To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

EOL announced

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

For announced

To all our customers

Regarding the change of names mentioned in the document, such as Mitsubishi Electric and Mitsubishi XX, to Renesas Technology Corp.

The semiconductor operations of Hitachi and Mitsubishi Electric were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Mitsubishi Electric, Mitsubishi Electric Corporation, Mitsubishi Semiconductors, and other Mitsubishi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Note : Mitsubishi Electric will continue the business operations of high frequency & optical devices and power devices.

EOL announced

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

User's Manual

16

7700 Family

Software Manual

MITSUBISHI 16-BIT SINGLE-CHIP MICROCOMPUTER / 7700 SERIES

EOL announced

Renesas Electronics
www.renesas.com

keep safety first in your circuit designs !

- Mitsubishi Electric Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of non-flammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Mitsubishi semiconductor product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Mitsubishi Electric Corporation or a third party.

- Mitsubishi Electric Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts or circuit application examples contained in these materials.

- All information contained in these materials, including product data, diagrams and charts, represent information on products at the time of publication of these materials, and are subject to change by Mitsubishi Electric Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor for the latest product information before purchasing a product listed herein.

- Mitsubishi Electric Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.

- The prior written approval of Mitsubishi Electric Corporation is necessary to reprint or reproduce in whole or in part these materials.

- If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
  Any diversion or reexport contrary to the export control laws and regulations of JAPAN and/or the country of destination is prohibited.

- Please contact Mitsubishi Electric Corporation or an authorized Mitsubishi Semiconductor product distributor for further details on these materials or the products contained therein.

EOL announced

Preface

This manual has been prepared to enable the users of the 7700 Family CMOS 16-bit microcomputers to better understand the instruction set and the features so that they can utilize the capabilities of the microcomputers to the fullest. This manual presents detailed descriptions of the instructions and addressing modes available for the 7700 Family microcomputers.
For the hardware descriptions of the 7700 Family microcomputers and descriptions of various development support tools (e.g., assembler, debugger, and so on.), please refer to the user's manuals and operating guidebooks for the respective hardware and software products.

EOL announced

Table of contents

CHAPTER 1. DESCRIPTION

1.1 Description 1-2

CHAPTER 2. CENTRAL PROCESSING UNIT (CPU)

2.1 Central Processing Unit (CPU) 2-2

CHAPTER 3. ADDRESSING MODES

3.1 Addressing Modes 3-2

3.2 Explanation of addressing mode 3-2

CHAPTER 4. INSTRUCTIONS

4.1 Instruction set 4-2

4.2 Description of instructions 4-7

4.3 Notes for programming 4-129

CHAPTER 5. INSTRUCTION EXECUTION SEQUENCE

5.1 Change of the CPU basic clock ϕ_{CPU} 5-2

5.2 Instruction execution sequence 5-3

CHAPTER 6. CPU INSTRUCTION EXECUTION SEQUENCE FOR EACH ADDRESSING MODE

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions 7-2
APPENDIX.2 Hexadecimal Instruction Code Table 7-18

EOL announced

CHAPTER 1

DESCRIPTION

EOL announced

1.1 Description

1.1 Description

The 7700 Family software offer the following features.

1.1.1 7700 Series, 7770 Series, and 7790 Series software features

- m and x flags are used to select between word and byte operation enabling most instructions to be implemented with 1-byte operation code (reduces application ROM size)
- Powerful addressing modes, and fast and compact instruction set
- Direct page mapping function and memory oriented software system by direct paging
- The usual 64K bytes program memory boundary can be ignored for the practical purposes, and programs can be written to utilize the full 16M bytes of memory space
- For data memory linear as well as bank memory accessing are supported
- Bit manipulation instructions and bit test and branch instructions can be used for memory and I/O accessing of the entire 16M bytes space
- Block transfer instruction capable of handling blocks of up to 64K bytes each
- Decimal arithmetic instruction execution requiring no software compensation

1.1.2 7750 Series software features

The 7750 Series software is based on the Mitsubishi original 16-bit microcomputer 7700 Series and provides enhanced signed operation instructions. The signed operation enhancement is realized by additional instructions, instruction code and execution cycle of 7750 Series are completely compatible with those of conventional 7700 Series instructions.

In addition to the 7700 Series, 7770 Series, and 7790 Series software features, the 7750 Series software offer the following features.

- Upward compatibility for the 7700 Series
- Signed operation enhancements through signed divide/multiply, arithmetic shift right, and sign/zero extension instruction support

EOL announced

In this manual, 7700 Series software is described unless otherwise noted.

CHAPTER 2
CENTRAL PROCESSING UNIT (CPU)

2.1 Central processing unit (CPU)

EOL announced

2.1 Central processing unit (CPU)

The CPU of 7700 Series has the ten registers as shown in Figure 2.1.1. Each of these registers is described below.

2.1.1 Accumulator (Acc)

Accumulators A and B are available and each can be used as 8-bit or 16-bit register as necessary.

(1) Accumulator A (A)

Accumulator A is the main register of the microcomputer. Data operations such as calculations, data transfer, and input/output are executed mainly through accumulator A. It consists of 16 bits, and the low-order 8 bits can be also used separately. The data length flag (m) determines whether the register is used as a 16-bit register or as an 8-bit register. It is used as a 16-bit register when the flag m is "0", and as an 8-bit register when the flag m is "1". The flag m is a part of the processor status register (PS) which is described later. When an 8-bit register is selected, the low-order 8 bits of the accumulator A are used and the contents of the high-order 8 bits are unchanged.

(2) Accumulator B (B)

Accumulator B is a 16-bit register with the same function as accumulator A. The instructions of 7700 Series can use accumulator B instead of accumulator A, but the use of accumulator B requires more instruction bytes and execution cycles than accumulator A. Accumulator B is also controlled by the data length flag m just as in accumulator A.

2.1.2 Index register X (X)

Index register X consists of 16 bits and the low-order 8 bits can be also used separately. The index register length flag (x) determines whether the register is used as a 16-bit register or as an 8-bit register. It is used as a 16-bit register when the flag x is "0" and as an 8-bit register when the flag x is "1". The flag x is a part of the processor status register (PS) which is described later. When an 8-bit register is selected, the low-order 8 bits of the index register X are used and the contents of the high-order 8 bits are unchanged. In addressing mode in which the index register X is used as the index register, the contents of this register is added to obtain the real address.

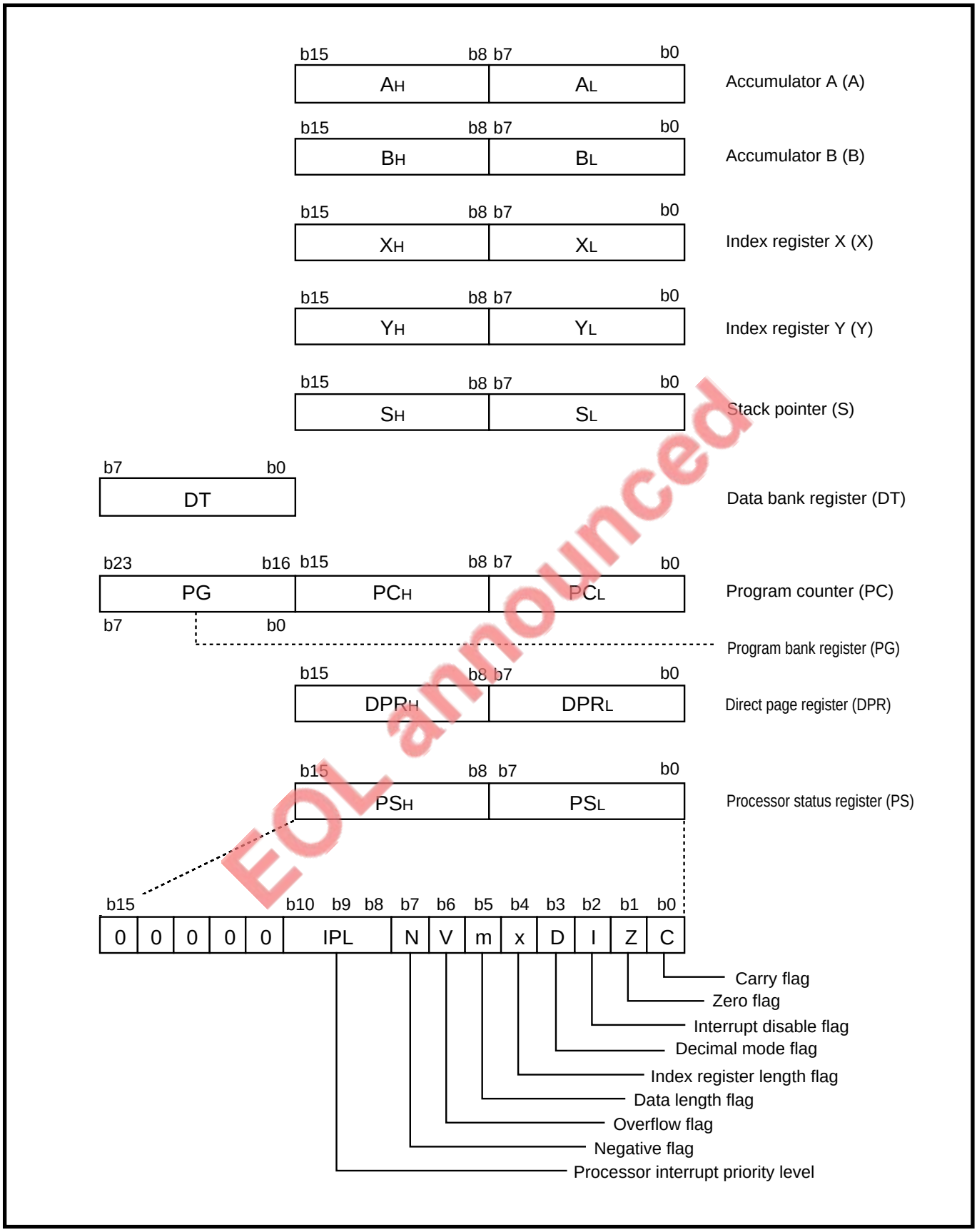
When executing the block transfer instruction **MVP** or **MVN**, the contents of the index register X indicate the low-order 16 bits of the source data address. The third byte of the **MVP** or **MVN** instruction is the high-order 8 bits of the source data address.

2.1.3 Index register Y (Y)

Index register Y is a 16-bit register with the same function as index register X. Just as in index register X, the index register length flag (x) determines whether this register is used as a 16-bit register or as an 8-bit register.

When executing the block transfer instruction **MVP** or **MVN**, the contents of the index register Y indicate the low-order 16 bits of the destination data address. The second byte of the **MVP** or **MVN** instruction is the high-order 8 bits of the destination data address.

EOL announced



EOL announced

Fig. 2.1.1 CPU registers structure

EOL announced

2.1.4 Stack pointer (S)

Stack pointer S is a 16-bit register. It is used for a subroutine call or an interrupt. It is also used for addressing modes using the stack. The contents of the stack pointer S indicate the address (stack area) for storing registers during subroutine calls and interrupts.

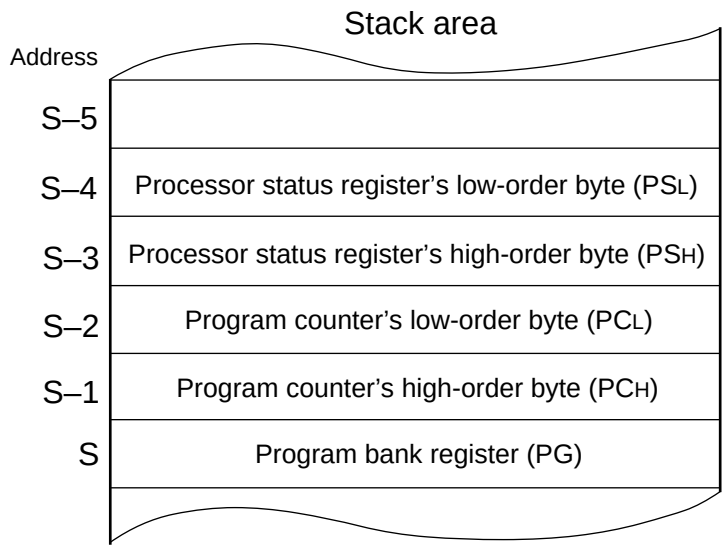
When an interrupt request is accepted, the contents of the program bank register PG is stored at the address indicated by the contents of the stack pointer S, and the contents of the stack pointer S are decremented by 1. Then the contents of the program counter PC and the processor status register PS (PC_H, PC_L, PS_H, PS_L) are stored. The contents of the stack pointer S after accepting an interrupt request are equal to the contents of the stack pointer S before the accepting of the interrupt request decremented by 5.

Figure 2.1.2 shows the stored registers before an interrupt routine.

When returning to the original routine after processing the interrupt routine by executing the **RTI** instruction, the registers stored in the stack area are restored to the original registers in the reverse sequence and the contents of the stack pointer are returned to the numerical value before the accepting of interrupt request. The same operation is performed during a subroutine call, but the contents of the processor status register PS are not automatically stored. (The contents of the program bank register PG may not be stored. It depends on the addressing mode.)

The user is responsible for storing registers other than those described above with a program during interrupts or subroutine calls.

Additionally, the stack pointer S must be initialized at the beginning of the program because its contents are undefined at reset. The stack area changes when subroutines are nested or when multiple interrupt requests are accepted. Therefore, make sure necessary data in the internal RAM are not destroyed when nesting subroutines.



❋ "S" is the initial address that the stack pointer (S) indicates at accepting an interrupt request.
The S's contents become "S–5" after storing the above registers.

Fig. 2.1.2 Stored registers before an interrupt routine

2.1.5 Program counter (PC)

Program counter PC is a 16-bit counter that indicates the low-order 16 bits of the next program memory address (24 bits) to be executed. The contents of the high-order program counter (PC_H) become "FF_{16}", and the low-order program counter (PC_L) become "FE_{16}" at reset. The contents of the program counter PC become the contents of the reset vector address (addresses $FFFE_{16}$, $FFFF_{16}$) after removing reset status. Figure 2.1.3 shows the program counter PC and the program bank register PG.

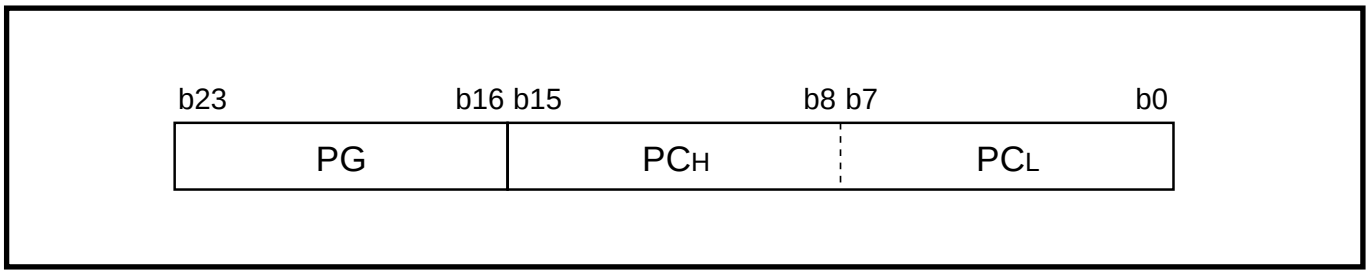


Fig. 2.1.3 Program counter PC and program bank register PG

2.1.6 Program bank register (PG)

Program bank register PG is an 8-bit register that indicates the high-order 8 bits (referred to as bank) of the next program memory address (24 bits) to be executed.

When a carry occurs after adding the contents of the program counter PC and other factors, the contents of the program bank register PG are automatically incremented by 1. When a borrow occurs after subtracting the contents of the program counter PC, the contents of the program bank register PG are automatically decremented by 1. Accordingly, there is no need to consider bank boundaries, usually.

This register is cleared to "00_{16}" at reset.

In single-chip mode, keep the program bank register PG "00_{16}" because the access within bank 0_{16} is only allowed. Be sure that prevent the program bank register PG from being set to a value other than "00_{16}" by executing the instructions of branch and so on.

This register is cleared to "00_{16}" at reset.

EOL announced

2.1.7 Data bank register (DT)

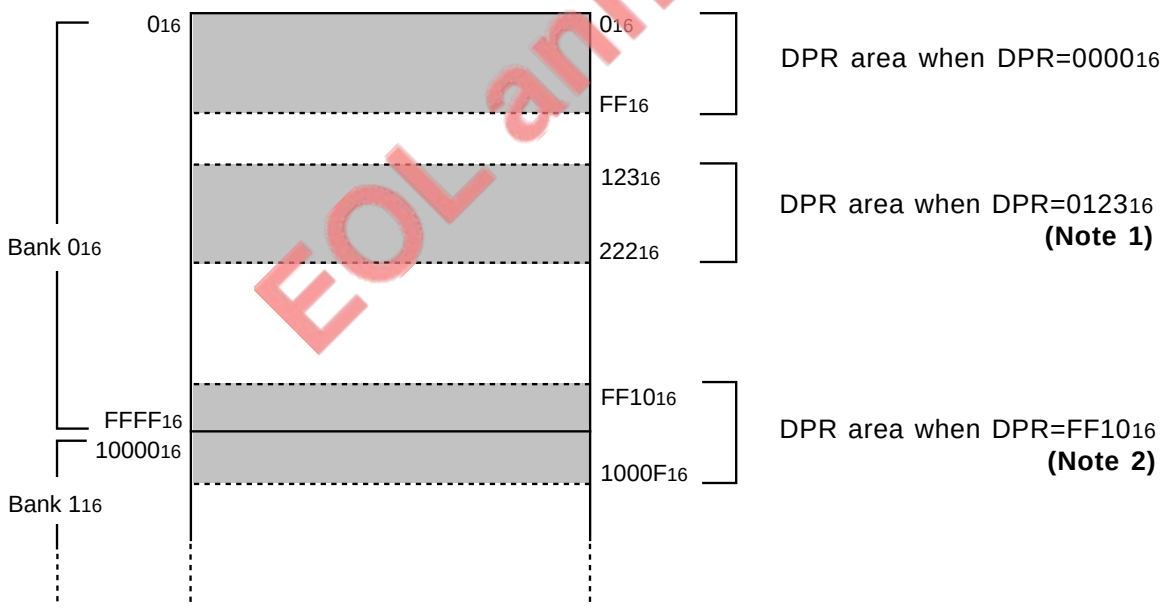
Data bank register DT is an 8-bit register. With some addressing modes using the data bank register DT, the contents of this register are used as the high-order 8 bits (bank) of a 24-bit address.
Use the **LDT** instruction to set the value in this register.
Set the only "00_{16}" in single-chip mode because the access within bank 0_{16} is only allowed.
This register is cleared to "00_{16}" at reset.

2.1.8 Direct page register (DPR)

Direct page register DPR is a 16-bit register. The contents of this register indicate the direct page area which is allocated in bank 0_{16} or in the area across banks 0_{16} and 1_{16}. This area can be accessed with 2 bytes✵ by using the direct page addressing mode.
The contents of the DPR are the base address (the lowest address) of the direct page area which extends to 256 bytes above this address. The DPR can contain a value from 0000_{16} to $FFFF_{16}$. However, set a value from 0000_{16} to $FF00_{16}$ in single-chip mode because the access within bank 0_{16} is only allowed.
If it contains a value equal to or more than "$FF01_{16}$", the direct page area spans the area across banks 0_{16} and 1_{16}. If the low-order 8 bits of the DPR is "00_{16}", the number of cycles required to generate an address is smaller by 1 cycle than the number if its contents are not "00_{16}". Accordingly, the low-order 8 bits of the DPR should usually be set to "00_{16}".
This register is cleared to "0000_{16}" at reset. Figure 2.1.4 shows a setting example of the direct page with the direct page register (DPR).

✵ With 3 bytes for **DIV** and **MPY** instructions, and 1 byte is added for all instructions when using accumulator B.

EOL announced



Note 1 : The number of execution cycles is incremented by 1 when the low-order 8 bits of the DPR are not "00_{16}".
Note 2 : The direct page area spans the area across banks 0_{16} and 1_{16} when the DPR is "$FF01_{16}$" or more.

Fig. 2.1.4 Setting example of direct page with direct page register (DPR)

2.1.9 Processor status register (PS)

Processor status register is an 11-bit register. It consists of the flags to indicate the result of operation and the processor interrupt priority level. The flags C, Z, V, and N are tested by branch instructions.
Figure 2.1.5 shows the structure of the processor status register.
The details of the processor status register flags are described below.

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	IPL			N	V	m	x	D	I	Z	C

Note: Bits 11 to 15 are always “0” when the contents of the processor status register are read.

Fig. 2.1.5 Processor status register structure

(1) Carry flag (C)

The carry flag is assigned to bit 0 of the processor status register. It contains the carry or borrow bit from the arithmetic and logic unit (ALU) after an arithmetic operation. This flag is also affected by shift and rotate instructions. This flag can be set with the **SEC** or **SEP** instruction and cleared with the **CLC** or **CLP** instruction.

(2) Zero flag (Z)

The zero flag is assigned to bit 1 of the processor status register. It is set to “1” when the result of an arithmetic operation or data transfer is zero, and cleared to “0” when otherwise. This flag can be set with the **SEP** instruction and cleared with the **CLP** instruction.
Note: This flag has no meaning in decimal mode addition (the **ADC** instruction).

(3) Interrupt disable flag (I)

The interrupt disable flag is assigned to bit 2 of the processor status register. It disables all maskable interrupts (interrupts other than watchdog timer, the **BRK** instruction, and zero division). Interrupts are disabled when this flag is “1”. When an interrupt request is accepted, this flag is automatically set to “1” to avoid multiple interrupts. This flag can be set with the **SEI** or **SEP** instruction and cleared with the **CLI** or **CLP** instruction. This flag is set to “1” at reset.

(4) Decimal mode flag (D)

The decimal mode flag is assigned to bit 3 of the processor status register. It determines whether addition and subtraction are performed in binary or decimal. Binary arithmetic is performed when this flag is “0”. When it is “1”, decimal arithmetic is performed with each word treated as two or four digits decimal (determined by the data length flag m). Decimal adjust is performed automatically. Decimal operation is possible only with the **ADC** and **SBC** instructions. This flag can be set with the **SEP** instruction and cleared with the **CLP** instruction. This flag is cleared to “0” at reset.

(5) Index register length flag (x)

The index register length flag is assigned to bit 4 of the processor status register. It determines whether the index register X and index register Y are used as a 16-bit register or an 8-bit register. The register is used as a 16-bit register when this flag x is “0”, and as an 8-bit register when it is “1”. This flag can be set with the **SEP** instruction and cleared with the **CLP** instruction. This flag is cleared to “0” at reset.

❋ When transferring between different bit lengths, the data is transferred with the length of the destination register, but except for the **TXA**, **TYA**, **TXB**, and **TYB** instructions.

EOL announced

(6) Data length flag (m)

The data length flag is assigned to bit 5 of the processor status register. It determines whether to treat data as a 16-bit unit or as an 8-bit unit. A data is treated as a 16-bit unit when this flag m is “0”, and as an 8-bit unit when it is “1”.

This flag can be set with the **SEM** or **SEP** instruction and cleared with the **CLM** or **CLP** instruction. This flag is cleared to “0” at reset.

❋ When transferring between different bit lengths, the data is transferred with the length of the destination register, but except for the **TXA**, **TYA**, **TXB**, and **TYB** instructions.

(7) Overflow flag (V)

The overflow flag is assigned to bit 6 of the processor status register. It is used when adding or subtracting a word as signed binary. In case the data length flag m is “0”, the overflow flag is set to “1” when the result of addition or subtraction is outside the range between –32768 and +32767, and cleared to “0” in all other cases. In case the data length flag m is “1”, the overflow flag is set to “1” when the result of addition or subtraction is outside the range between –128 and +127, and cleared to “0” in all other cases. The overflow flag can be set with the **SEP** instruction and cleared with the **CLV** or **CLP** instructions.

Note : This flag has no meaning in decimal mode.

(8) Negative flag (N)

The negative flag is assigned to bit 7 of the processor status register. It is set to “1” when the result of arithmetic operation or data transfer is negative (data bit 15 is “1” when the data length flag m is “0”, or data bit 7 is “1” when the data length flag m is “1”). It is cleared to “0” in all other cases. This flag can be set with the **SEP** instruction and cleared with the **CLP** instruction.

Note : This flag has no meaning in decimal mode.

(9) Processor interrupt priority level (IPL)

The processor interrupt priority level (IPL) is assigned to bits 8, 9, and 10 of the processor status register. These three bits determine the priority level of processor interrupts from level 0 to level 7. The interrupt is enabled when the interrupt priority level of a required interrupt (set with the interrupt control register) is higher than IPL. When an interrupt request is accepted, the IPL is stored in the stack and IPL is replaced by the interrupt priority level of the accepted interrupt request. This simplifies control of multiple interrupts.

There are no instructions to directly set or clear the IPL. It can be changed by placing the new IPL on the stack and updating the processor status register with the **PUL** or **PLP** instruction.

The contents of the IPL are cleared to “000_2” at reset.

EOL announced

CHAPTER 3

ADDRESSING MODES

3.1 Addressing modes
3.2 Explanation of addressing modes

EOL announced

3.1 Addressing Modes

When executing an instruction, the address of the memory location from which the data required for arithmetic operation is to be retrieved or to which the result of arithmetic operation is to be stored must be specified in advance. Address specification is also necessary when the control is to jump to a certain memory address during program execution. Addressing refers to the method of specifying the memory address.

The 7700 Series microcomputers support 28 different addressing modes, offering extremely versatile and powerful memory accessing capability.

3.2 Explanation of Addressing Modes

Each of the 28 addressing modes is explained on the pages indicated below:

Implied addressing mode 3-3
Immediate addressing mode 3-4
Accumulator addressing mode 3-6
Direct addressing mode 3-7
Direct bit addressing mode 3-9
Direct indexed X addressing mode 3-11
Direct indexed Y addressing mode 3-14
Direct indirect addressing mode 3-15
Direct indexed X indirect addressing mode 3-17
Direct indirect indexed Y addressing mode 3-20
Direct indirect long addressing mode 3-23
Direct indirect long indexed Y addressing mode 3-25
Absolute addressing mode 3-28
Absolute bit addressing mode 3-31
Absolute indexed X addressing mode 3-33
Absolute indexed Y addressing mode 3-36
Absolute long addressing mode 3-39
Absolute long indexed X addressing mode 3-41
Absolute indirect addressing mode 3-43
Absolute indirect long addressing mode 3-44
Absolute indexed X indirect addressing mode 3-45
Stack addressing mode 3-46
Relative addressing mode 3-49
Direct bit relative addressing mode 3-50
Absolute bit relative addressing mode 3-52
Stack pointer relative addressing mode 3-54
Stack pointer relative indirect indexed Y addressing mode 3-55
Block transfer addressing mode 3-58

EOL announced

Note. On the pages below, the instructions with the mark " * " can be used in the 7750 Series only.

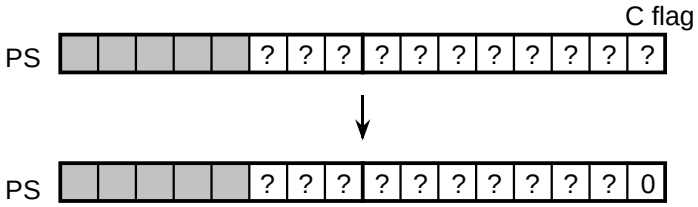
Implied

Mode : Implied addressing mode

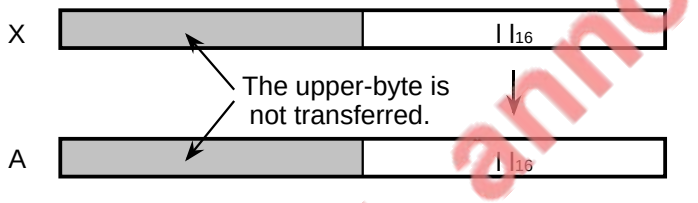
Function : The single-instruction inherently address an internal register.

Instruction : BRK, CLC, CLI, CLM, CLV, DEX, DEY, INX, INY, NOP, RTI, RTL, RTS, SEC, SEI, SEM, STP, TAD, TAS, TAX, TAY, TBD, TBS, TBX, TBY, TDA, TDB, TSA, TSB, TSX, TXA, TXB, TXS, TXY, TYA, TYB, TYX, WIT, XAB

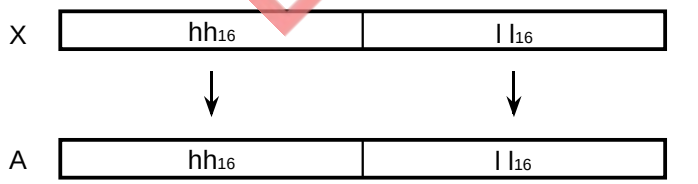
ex. : Mnemonic Machine Code
CLC 18_{16}



ex. : Mnemonic Machine Code
TXA $8A_{16}$
(m=1, x=1)



ex. : Mnemonic Machine Code
TXA $8A_{16}$
(m=0, x=0)



EOL announced

Immediate

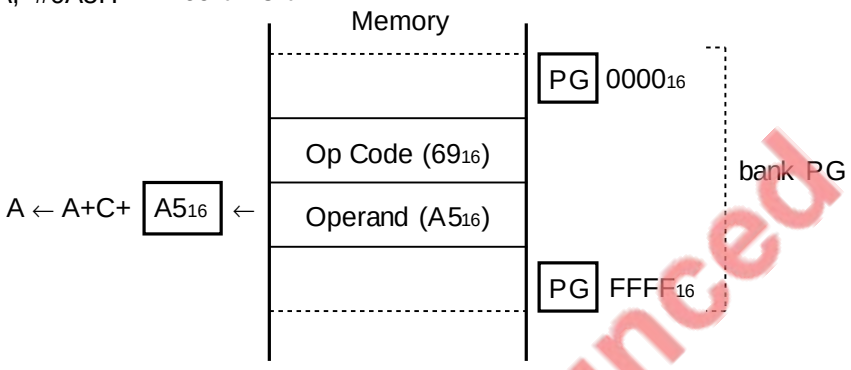
Mode : Immediate addressing mode

Function : A portion of the instruction is the actual data. Such instruction code may cross over the bank boundary.

Instruction : ADC, AND, CLP, CMP, CPX, CPY, DIV, DIVS*, EOR, LDA, LDT, LDX, LDY, MPY, MPYS*, ORA, RLA, SBC, SEP

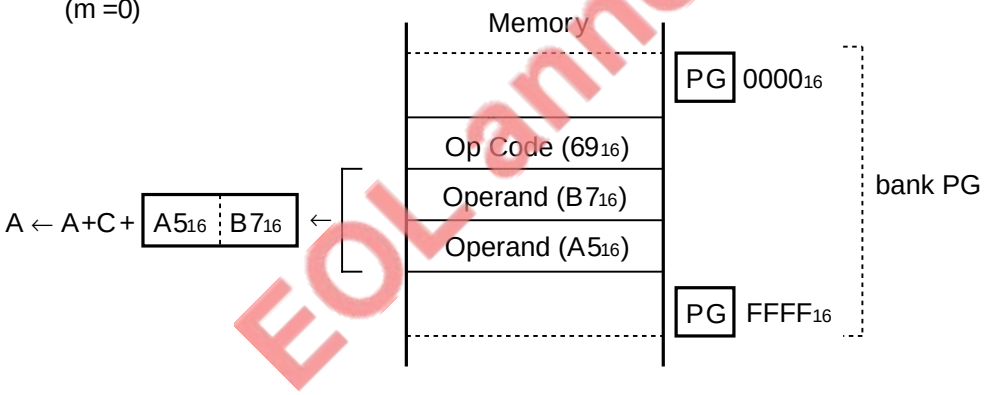
ex. : Mnemonic
ADC A, #0A5H
(m=1)

Machine Code
69_{16} $A5_{16}$



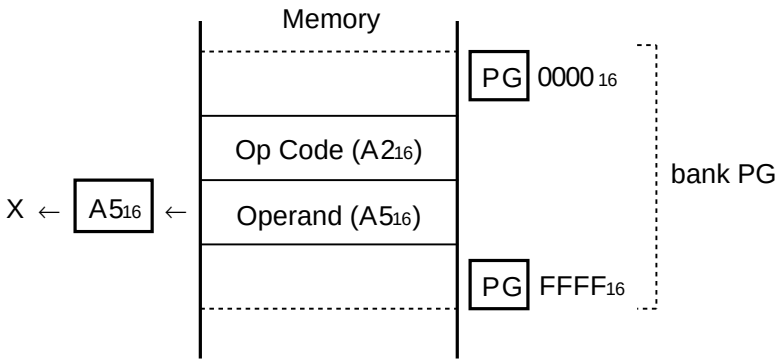
ex. : Mnemonic
ADC A, #0A5B7H
(m =0)

Machine Code
69_{16} $B7_{16}$ $A5_{16}$



ex. : Mnemonic
LDX #0A5H
(x=1)

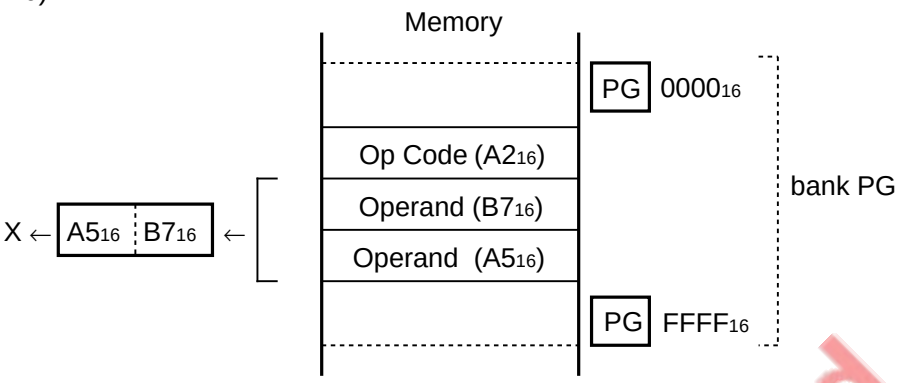
Machine Code
$A2_{16}$ $A5_{16}$



EOL announced

ex. : Mnemonic
LDX #0A5B7H
(x=0)

Machine Code
$A2_{16}$ $B7_{16}$ $A5_{16}$



EOL announced

Accumulator

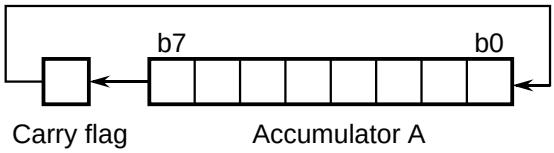
Mode : Accumulator addressing mode

Function : The contents of accumulator are the actual data.

Instruction : ASL, ASR*, DEC, EXTS*, EXTZ*, INC, LSR, ROL, ROR

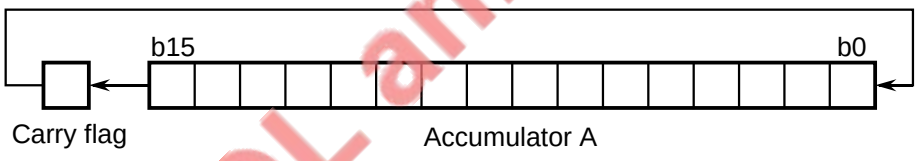
ex. : Mnemonic — Machine Code

ROL A (m=1) — $2A_{16}$



ex. : Mnemonic — Machine Code

ROL A (m=0) — $2A_{16}$



EOL announced

Direct

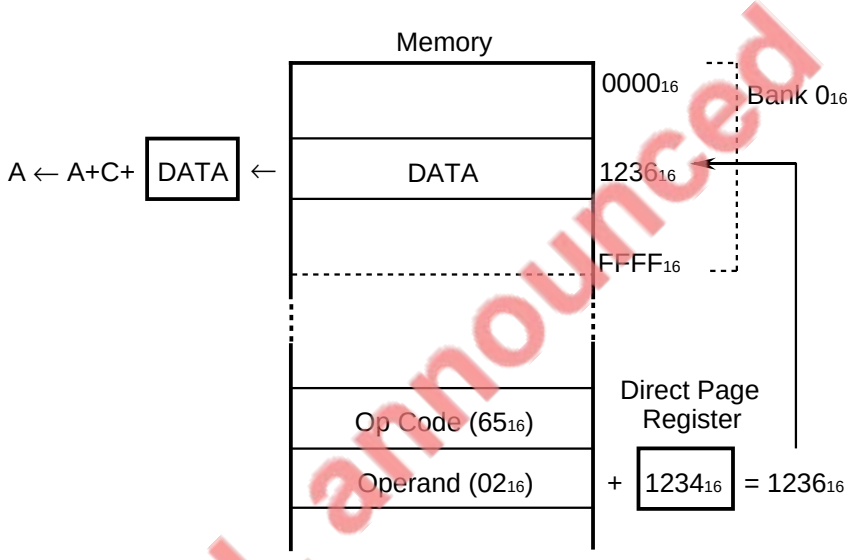
Mode : Direct addressing mode

Function : The contents of the bank 0_{16} memory location specified by the result of adding the second byte of the instruction to the contents of the direct page register become the actual data. If, however, addition of the instruction's second byte to the direct page register's contents result in a value that exceeds the bank 0_{16} range, the specified location will be in bank 1_{16}.

Instruction : ADC, AND, ASL, ASR*, CMP, CPX, CPY, DEC, DIV, DIVS*, EOR, INC, LDA, LDM, LDX, LDY, LSR, MPY, MPYS*, ORA, ROL, ROR, SBC, STA, STX, STY

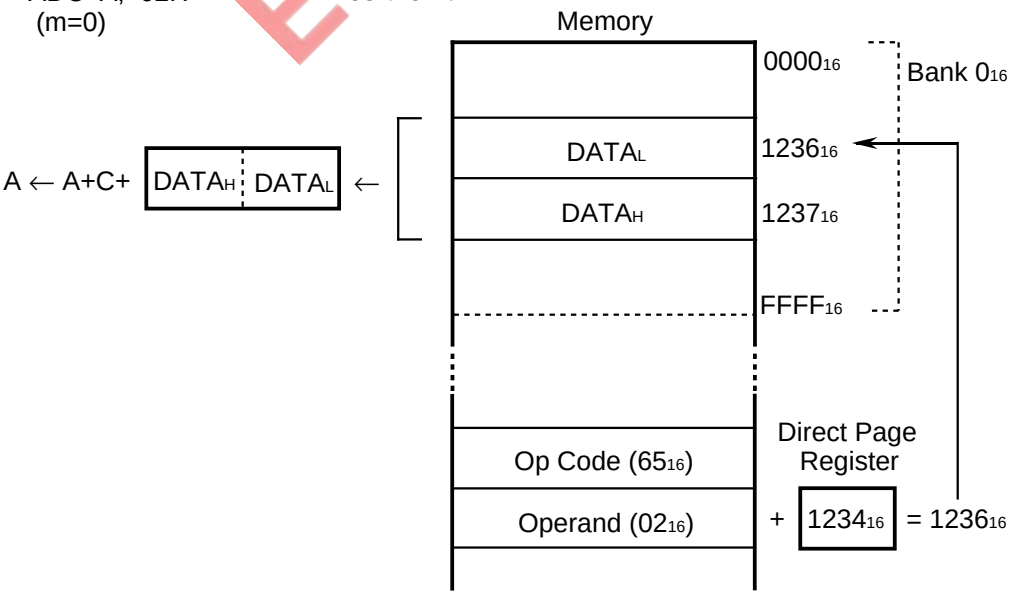
ex. : Mnemonic
ADC A, 02H
(m=1)

Machine Code
65_{16} 02_{16}



ex. : Mnemonic
ADC A, 02H
(m=0)

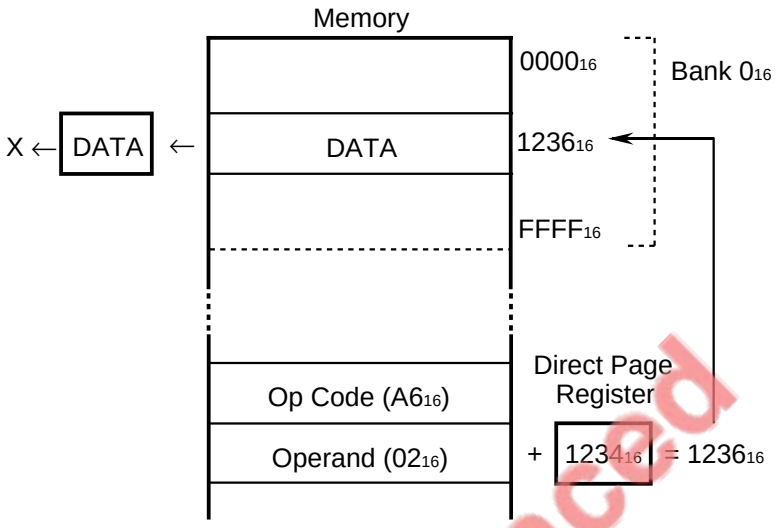
Machine Code
65_{16} 02_{16}



EOL announced

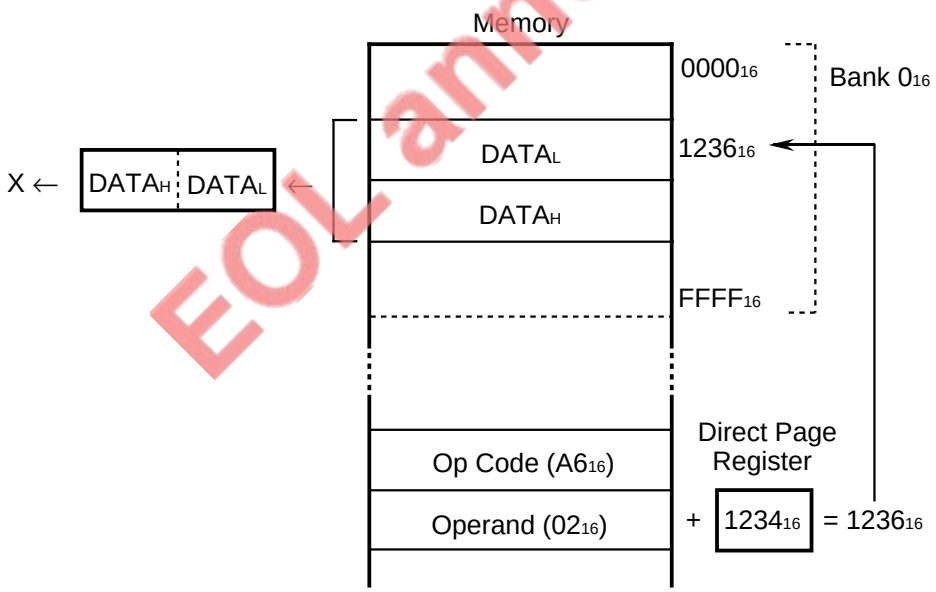
ex. : Mnemonic
LDX 02H
(x=1)

Machine Code
$A6_{16}$ 02_{16}



ex. : Mnemonic
LDX 02H
(x=0)

Machine Code
$A6_{16}$ 02_{16}



EOL announced

Mode : Direct bit addressing mode

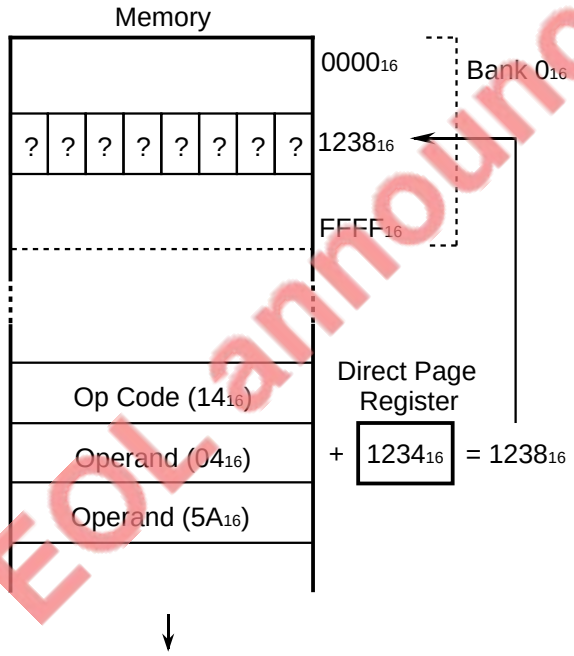
Function : Specifies the bank 0_{16} memory location by the value obtained by adding the instruction's second byte to the direct page register's contents, and specifies the positions of multiple bits in the memory location by the bit pattern in the third and fourth bytes of the instruction (third byte only when the m flag is set to 1). If, however, addition of the instruction's second byte to the direct page register's contents result in a value that exceeds the bank 0_{16} range, the specified location will be in bank 1_{16}.

Instruction : CLB, SEB

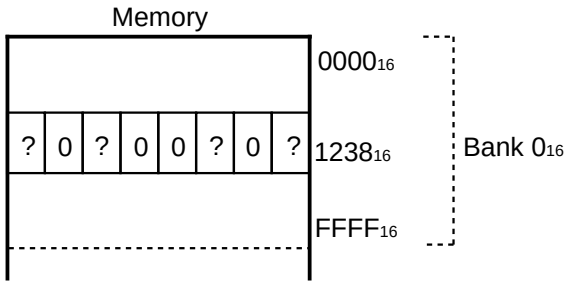
ex. : Mnemonic — Machine Code

CLB #5AH, 04H (m=1) — 14_{16} 04_{16} $5A_{16}$

(Before the instruction execution)



(After the instruction execution)

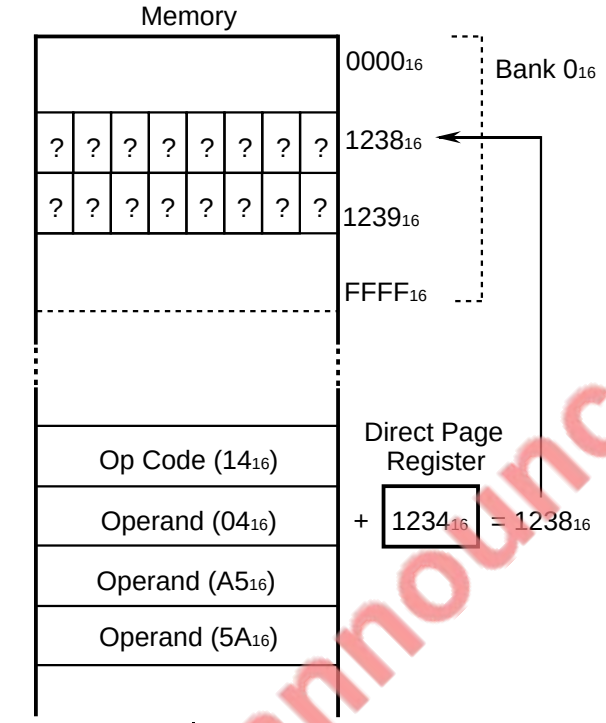


EOL announced

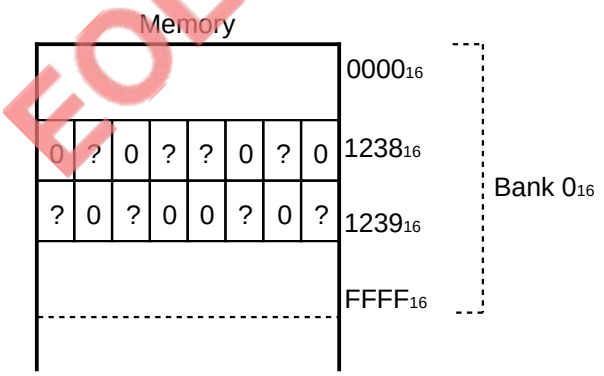
ex. : Mnemonic | Machine Code

CLB #5AA5H, 04H (m=0) | 14_{16} 04_{16} $A5_{16}$ $5A_{16}$

(Before the instruction execution)



(After the instruction execution)



EOL announced

Direct Indexed X

Mode : Direct indexed X addressing mode

Function : The contents of the bank 0_{16} memory location specified by the result of adding the second byte of the instruction, the contents of the direct page register and the contents of the index register X become the actual data. If, however, addition of the instruction's second byte, the direct page register's contents and the index register X's contents results in a value that exceeds the bank 0_{16} or bank 1_{16} range, the specified location will be in bank 1_{16} or bank 2_{16}.

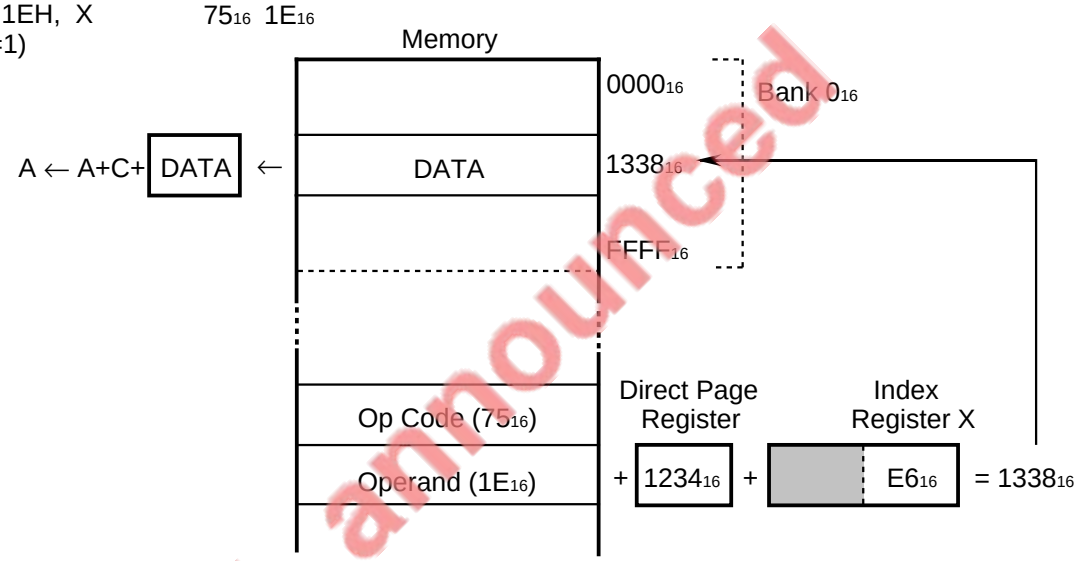
Instruction : ADC, AND, ASL, ASR*, CMP, DEC, DIV, DIVS*, EOR, INC, LDA, LDM, LDY, LSR, MPY, MPYS*, ORA, ROL, ROR, SBC, STA, STY

ex. : Mnemonic

ADC A, 1EH, X
(m=1, x=1)

Machine Code

75_{16} $1E_{16}$

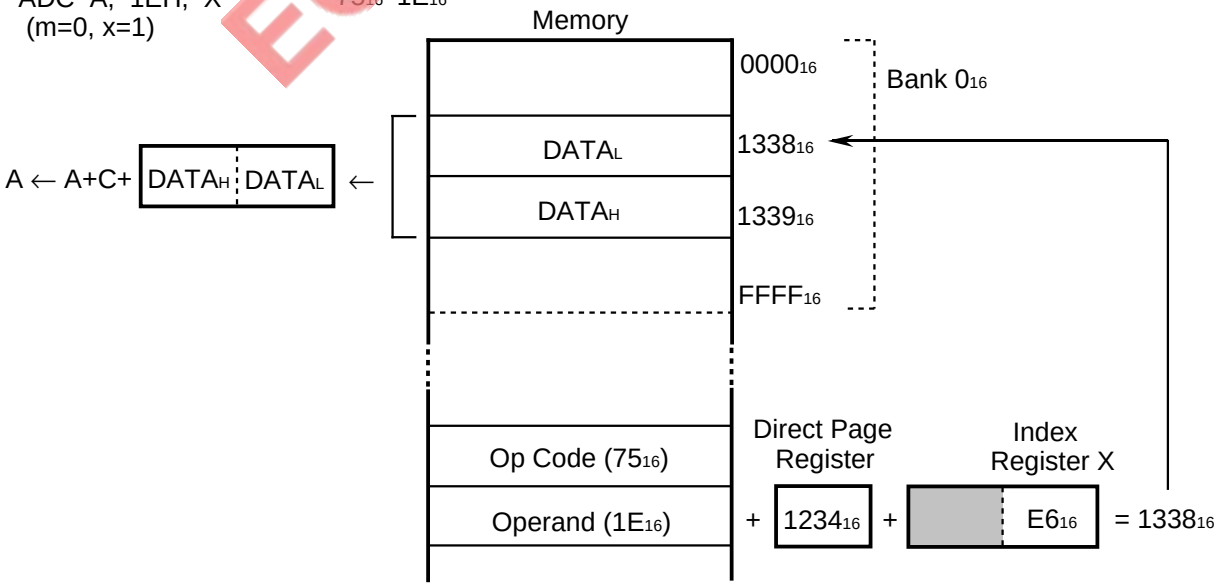


ex. : Mnemonic

ADC A, 1EH, X
(m=0, x=1)

Machine Code

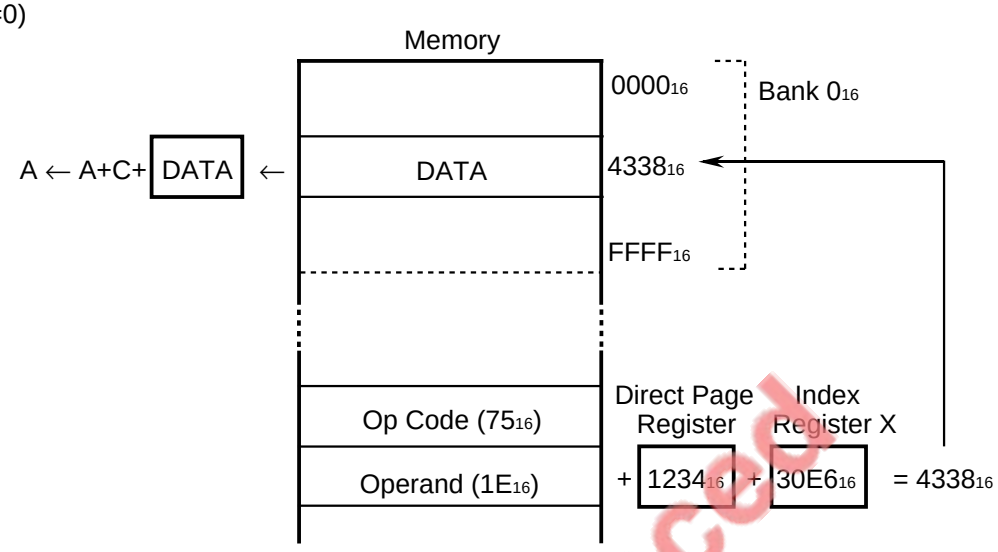
75_{16} $1E_{16}$



EOL announced

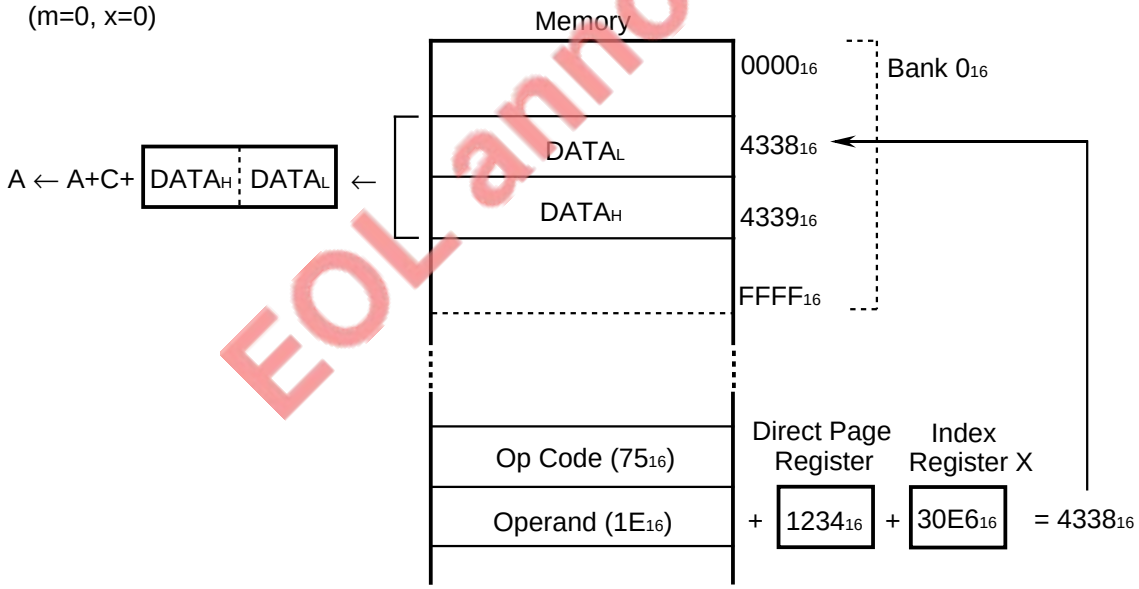
ex. : Mnemonic
ADC A, 1EH, X
(m=1, x=0)

Machine Code
75_{16} $1E_{16}$



ex. : Mnemonic
ADC A, 1EH, X
(m=0, x=0)

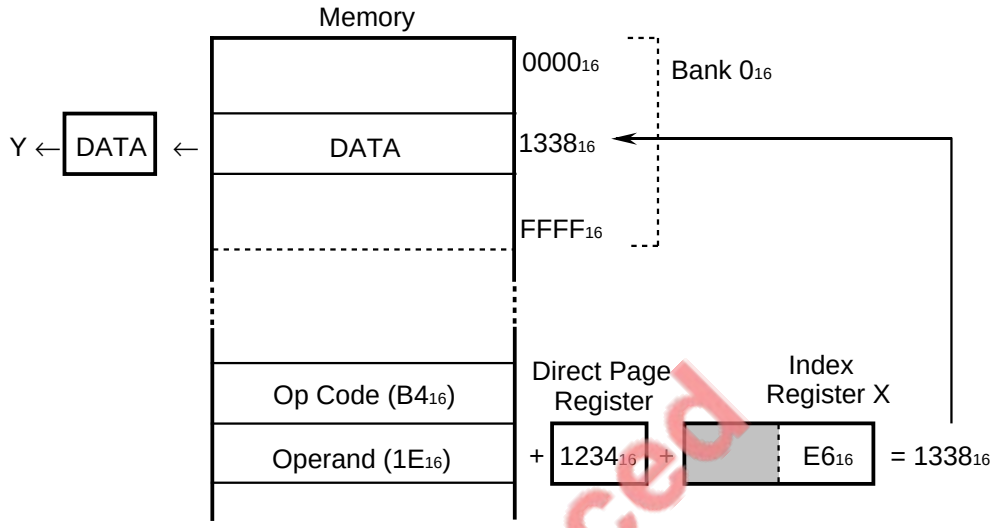
Machine Code
75_{16} $1E_{16}$



EOL announced

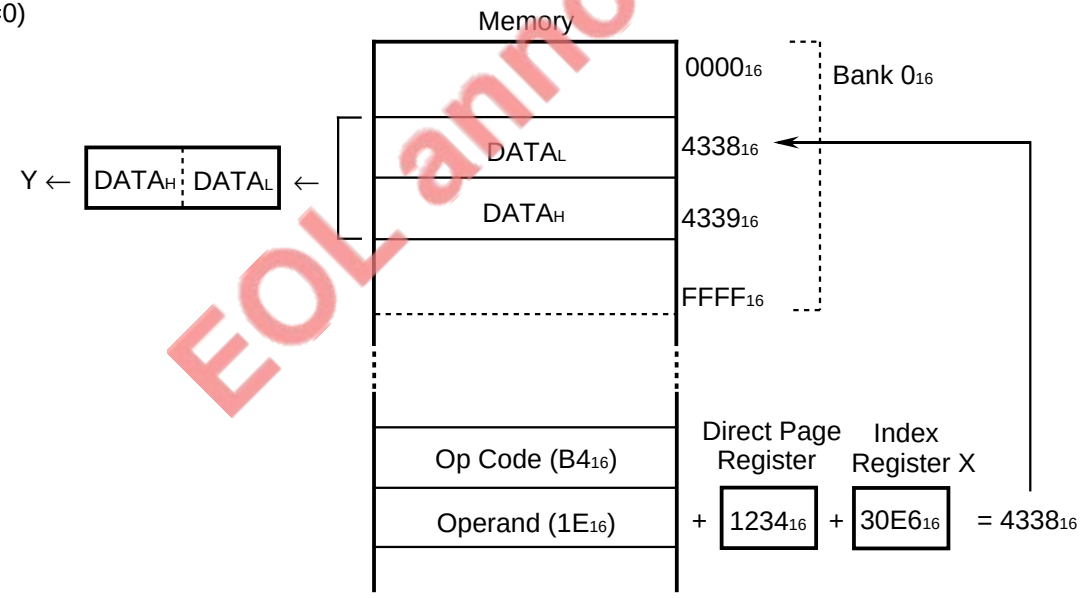
ex. : Mnemonic
LDY 1EH, X
(x=1)

Machine Code
$B4_{16}$ $1E_{16}$



ex. : Mnemonic
LDY 1EH, X
(x=0)

Machine Code
$B4_{16}$ $1E_{16}$



EOL announced

Direct Indexed Y

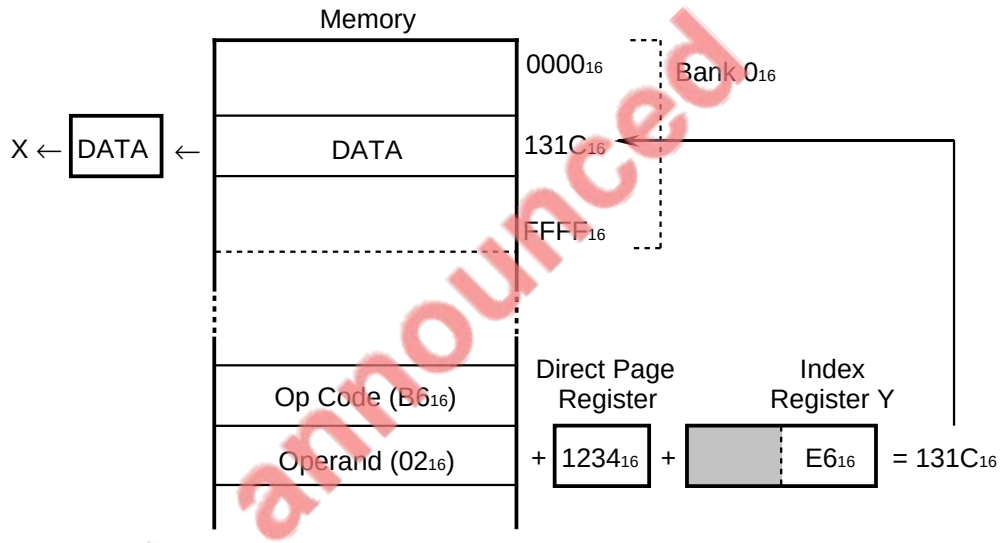
Mode : Direct indexed Y addressing mode

Function : The contents of the bank 0_{16} memory location specified by the result of adding the second byte of the instruction, the contents of the direct page register and the contents of the index register Y become the actual data. If, however, addition of the instruction's second byte, the direct page register's contents and the index register Y's contents results in a value that exceeds the bank 0_{16} or bank 1_{16} range, the specified location will be in bank 1_{16} or bank 2_{16}.

Instruction : LDX, STX

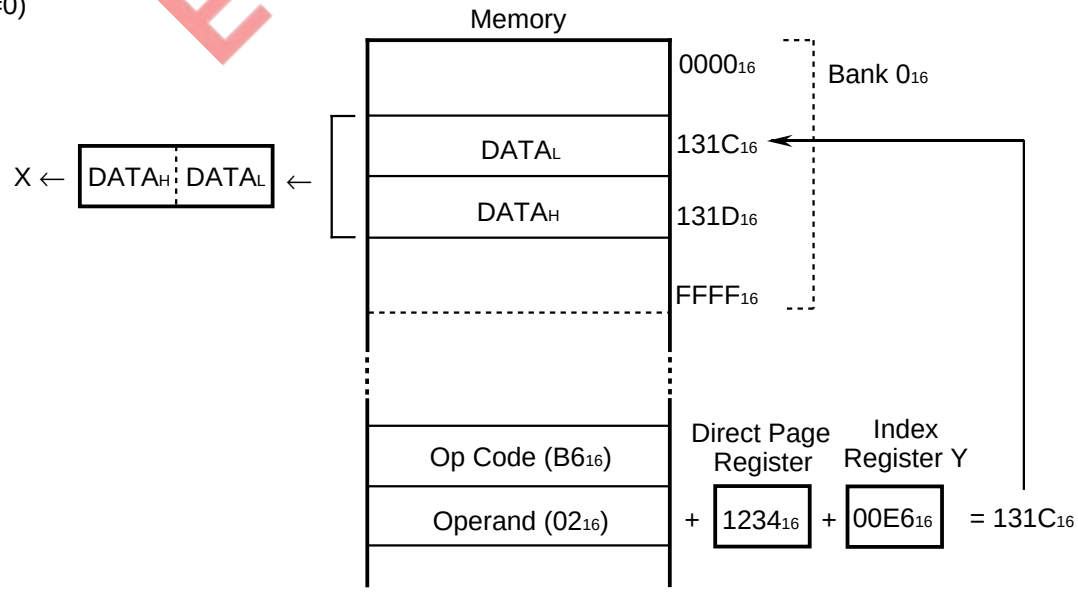
ex. : Mnemonic
LDX 02H, Y
(x=1)

Machine Code
$B6_{16}$ 02_{16}



ex. : Mnemonic
LDX 02H, Y
(x=0)

Machine Code
$B6_{16}$ 02_{16}



EOL announced

Direct Indirect

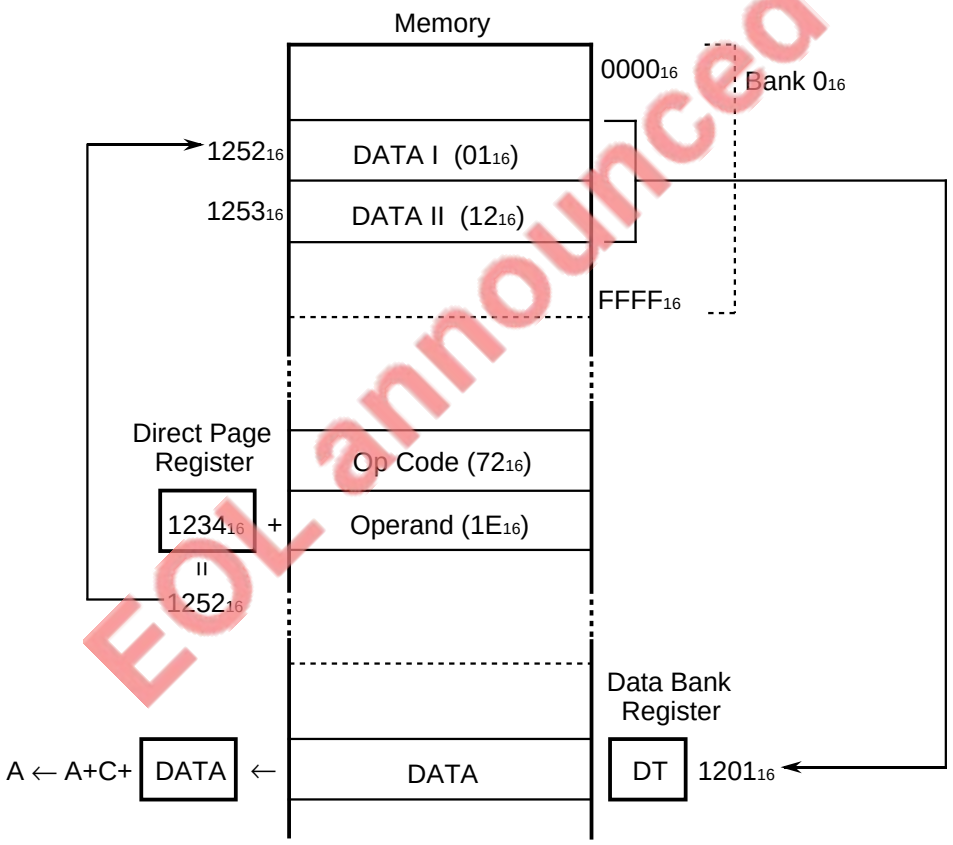
Mode : Direct indirect addressing mode

Function : The value obtained by adding the instruction's second byte to the contents of the direct page register specifies 2 adjacent bytes in memory bank 0_{16}, and the contents of these bytes in memory bank DT (DT is contents of data bank register) become the actual data. If, however, the value obtained by adding the instruction's second byte and the direct page register's contents exceeds the bank 0_{16} range, the specified location will be in bank 1_{16}.

Instruction : ADC, AND, CMP, DIV, DIVS*, EOR, LDA, MPY, MPYS*, ORA, SBC, STA

ex. : Mnemonic
ADC A, (1EH)
(m=1)

Machine Code
72_{16} $1E_{16}$



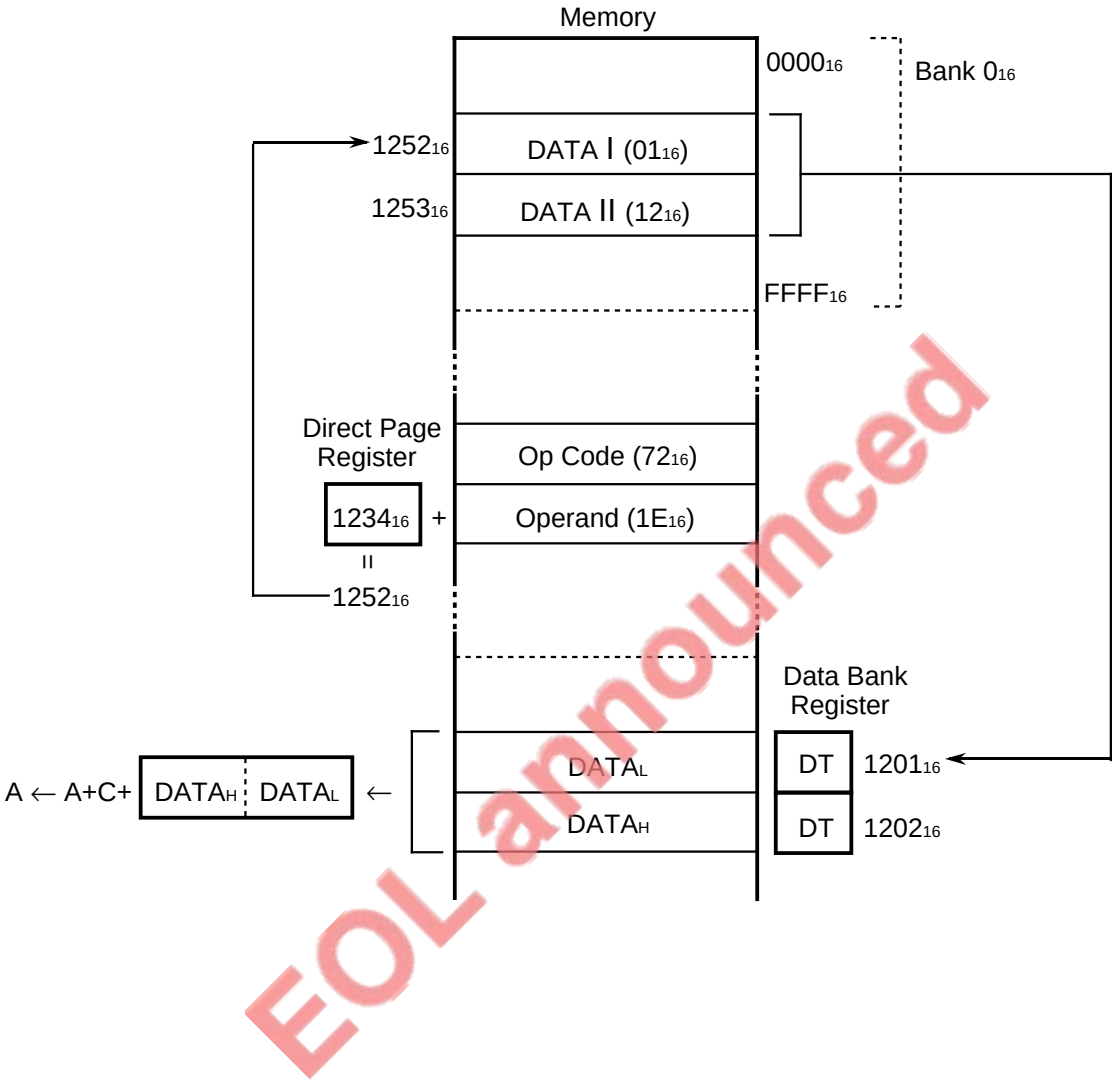
EOL announced

ex. : Mnemonic

ADC A, (1EH)
(m=0)

Machine Code

72_{16} $1E_{16}$



EOL announced

Direct Indexed X Indirect

Mode : Direct indexed X indirect addressing mode

Function : The value obtained by adding the instruction's second byte, the contents of the direct page register and the contents of the index register X specifies 2 adjacent bytes in memory bank 0_{16}, and the contents of these bytes in memory bank 0_{16}, and the contents of these bytes in memory bank DT (DT is contents of data bank register) become the actual data. If, however, the value obtained by adding the instruction's second byte, the direct page register's contents and the index register X's contents exceeds the bank 0_{16} or bank 1_{16} range, the specified location will be in bank 1_{16} or bank 2_{16}.

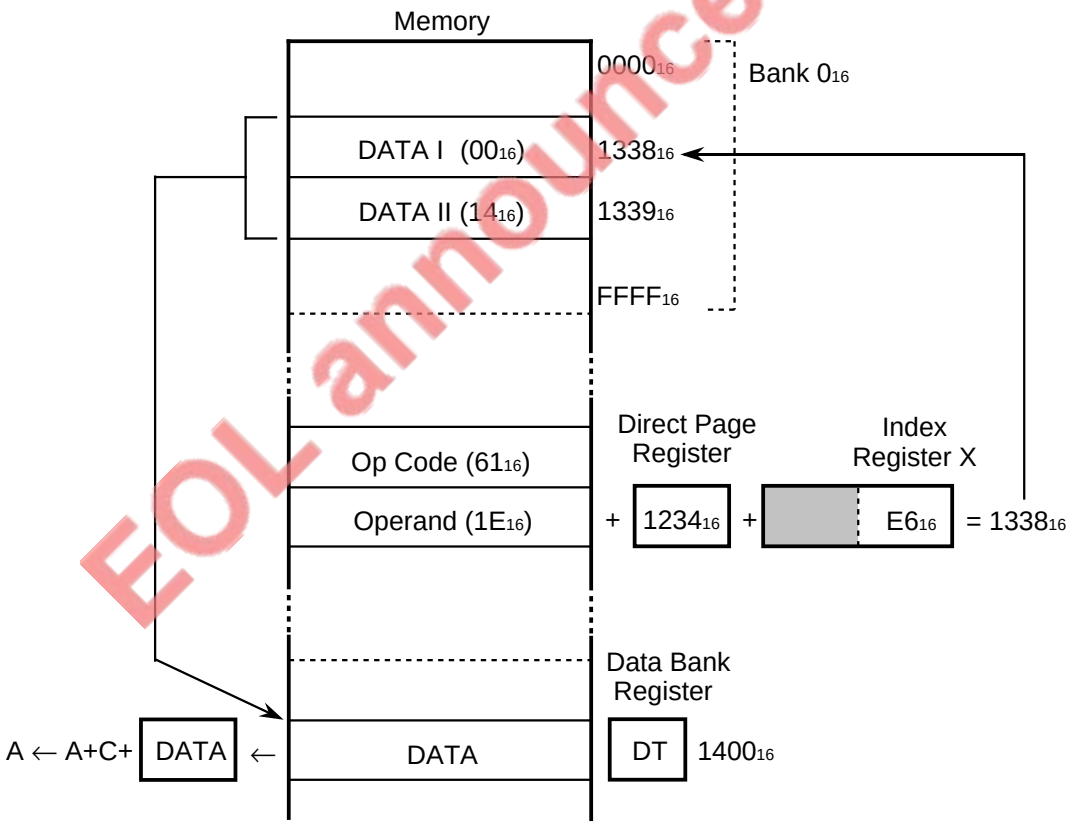
Instruction : ADC, AND, CMP, DIV, DIVS*, EOR, LDA, MPY, MPYS*, ORA, SBC, STA

ex. : Mnemonic

ADC A, (1EH, X)
(m=1, x=1)

Machine Code

61_{16} $1E_{16}$

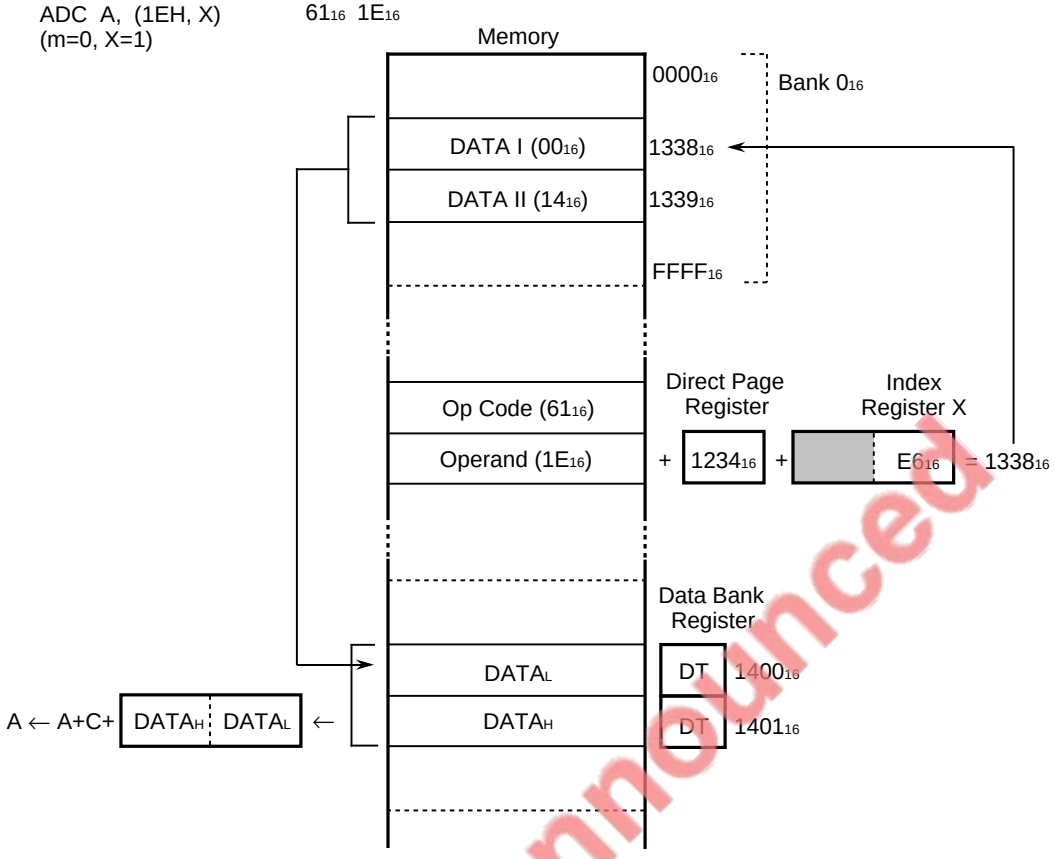


EOL announced

Direct Indexed X Indirect

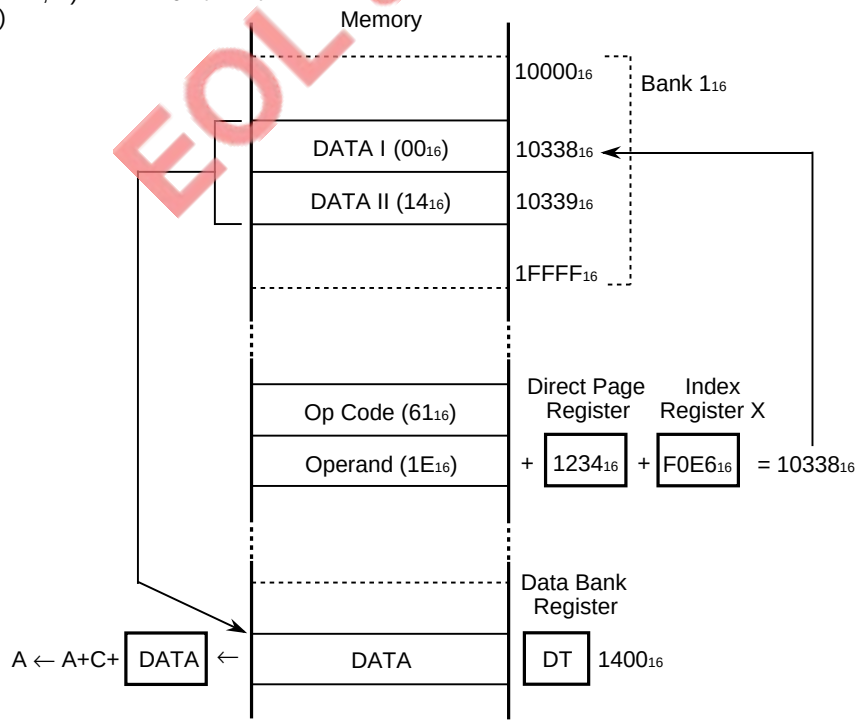
ex. : Mnemonic
ADC A, (1EH, X)
(m=0, X=1)

Machine Code
61_{16} $1E_{16}$



ex. : Mnemonic
ADC A, (1EH, X)
(m=1, x=0)

Machine Code
61_{16} $1E_{16}$

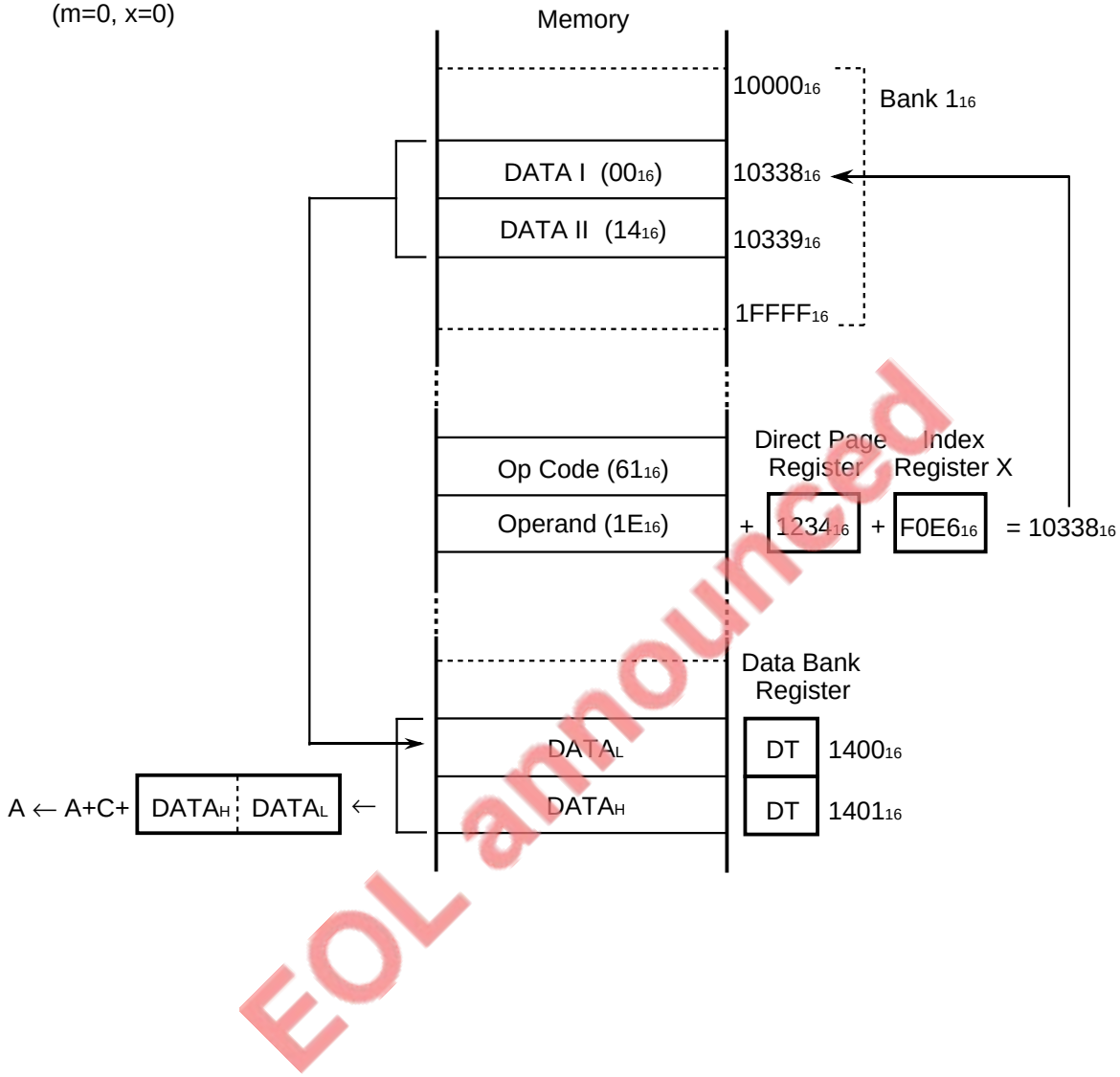


EOL announced

Direct Indexed X Indirect

ex. : Mnemonic
ADC A, (1EH, X)
(m=0, x=0)

Machine Code
61_{16} $1E_{16}$



EOL announced

Direct Indirect Indexed Y

Mode : Direct indirect indexed Y addressing mode

Function : The value obtained by adding the instruction's second byte and the contents of the direct page register specifies 2 adjacent bytes in memory bank 0_{16}.
The value obtained by adding the contents of these bytes and the contents of the index register Y specifies address of the actual data in memory bank DT (DT is contents of data bank register). If, however, the value obtained by adding the contents of the instruction's second byte and the direct page register exceeds the bank 0_{16} range, the specified location will be in bank 1_{16}. Also, if addition of the contents of memory and index register Y generate a carry, the bank number will be 1 larger than the contents of the data bank register.

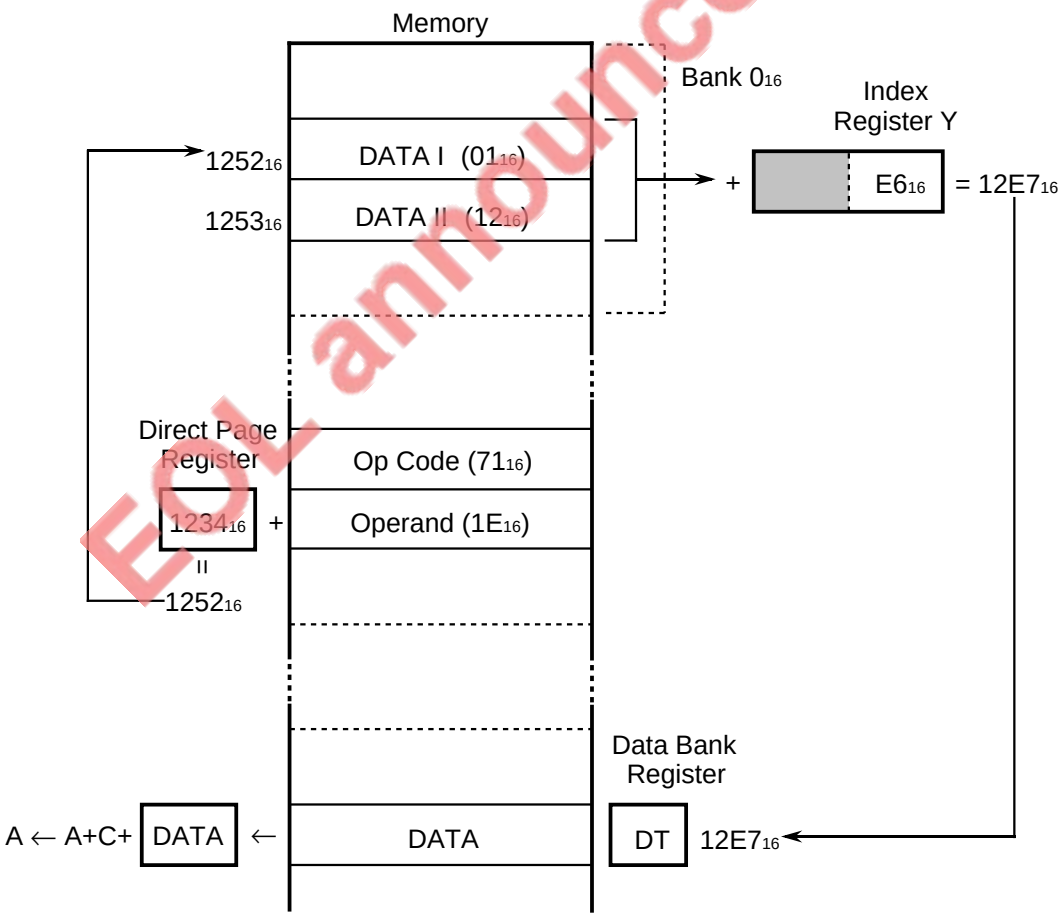
Instruction : ADC, AND, CMP, DIV, DIVS*, EOR, LDA, MPY, MPYS*, ORA, SBC, STA

ex. : Mnemonic

ADC A, (1EH), Y
(m=1, x=1)

Machine Code

71_{16} $1E_{16}$

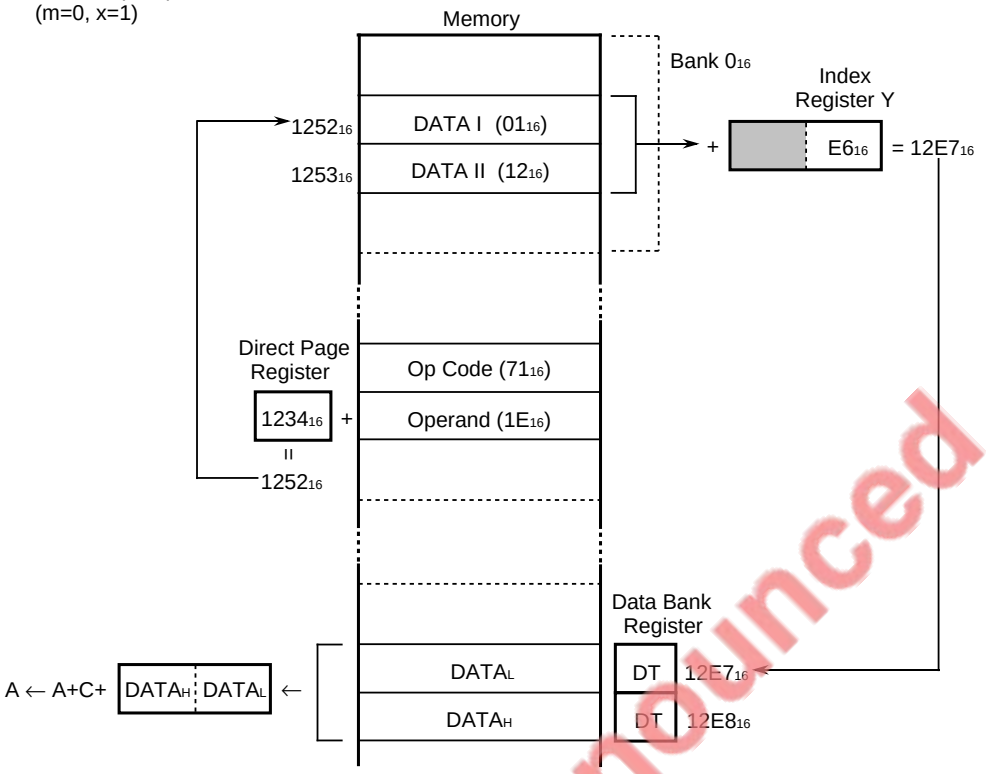


EOL announced

ex. : Mnemonic

ADC A, (1EH), Y
(m=0, x=1)

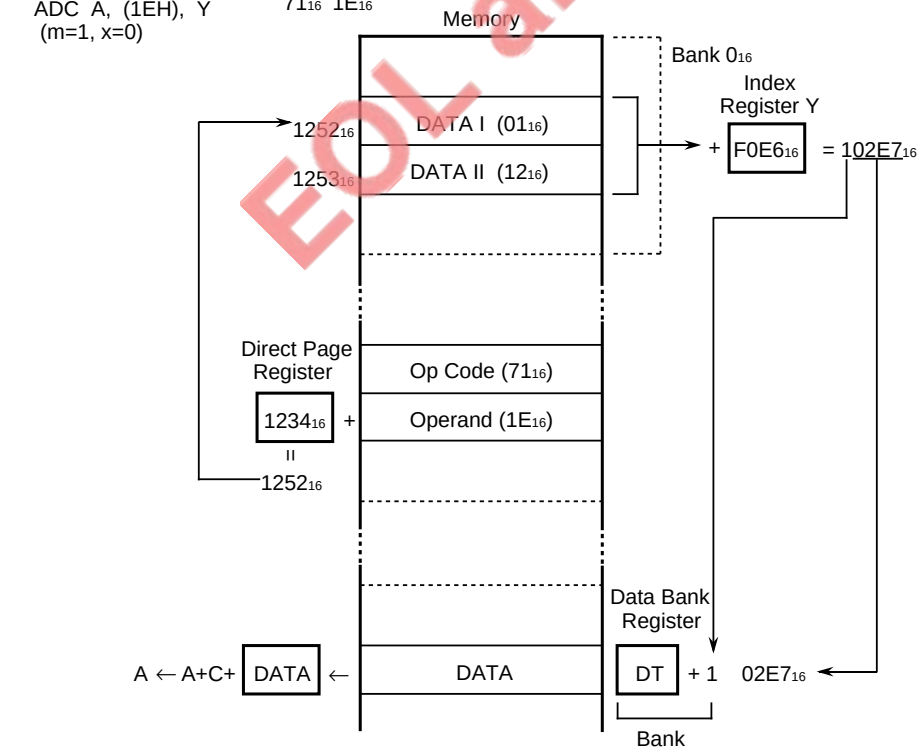
Machine Code
71_{16} $1E_{16}$



ex. : Mnemonic

ADC A, (1EH), Y
(m=1, x=0)

Machine Code
71_{16} $1E_{16}$



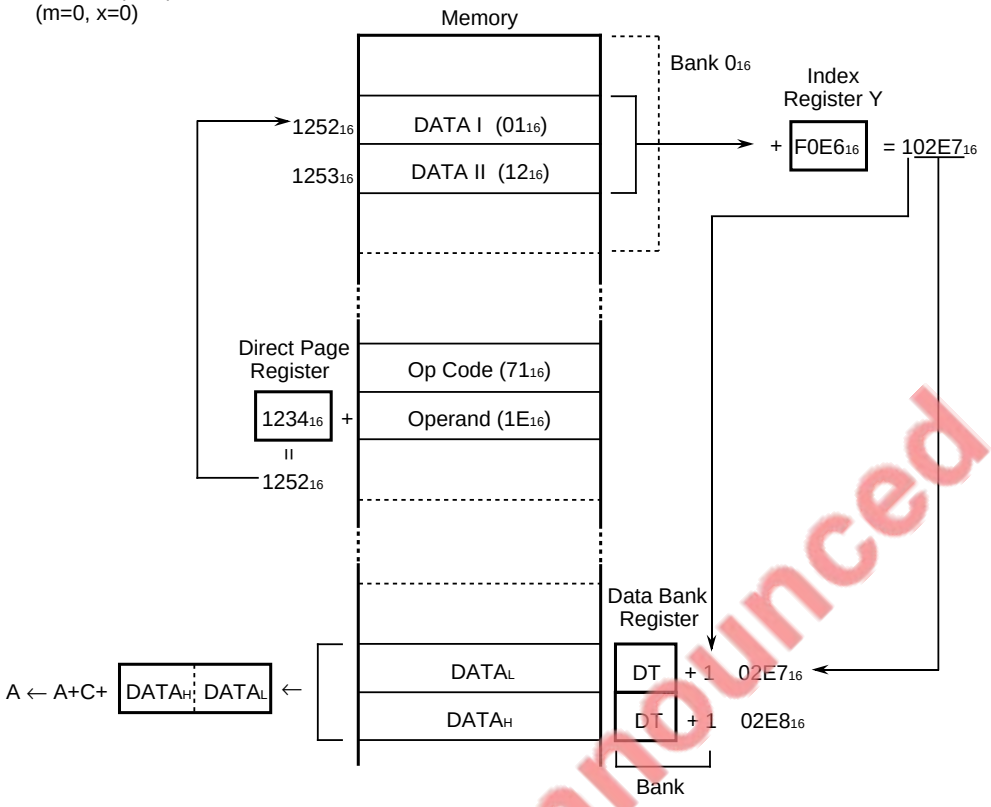
EOL announced

ex. : Mnemonic

ADC A, (1EH), Y
(m=0, x=0)

Machine Code

71_{16} $1E_{16}$



EOL announced

Direct Indirect Long

Mode : Direct indirect long addressing mode

Function : The value obtained by adding the instruction's second byte and the contents of the direct page register specifies 3 adjacent bytes in memory bank 0_{16}, and the contents of these bytes specify the address of the memory location that contains the actual data. If, however, the value obtained by adding the contents of the instruction's second byte and the direct page register exceeds the bank 0_{16} range, the specified location will be in bank 1_{16}. The 3 adjacent bytes memory location may be spread over two different banks.

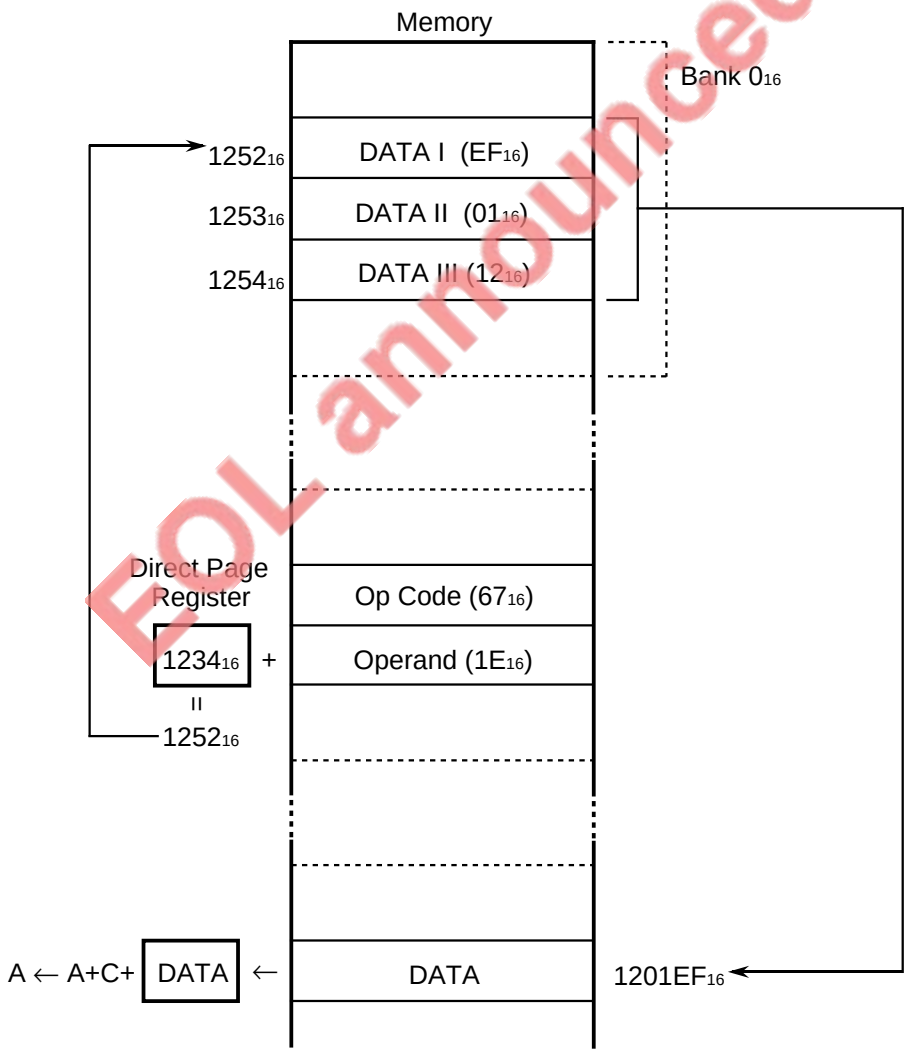
Instruction : ADC, AND, CMP, DIV, DIVS*, EOR, LDA, MPY, MPYS*, ORA, SBC, STA

ex. : Mnemonic

ADCL A, (1EH)
(m=1)

Machine Code

67_{16} $1E_{16}$



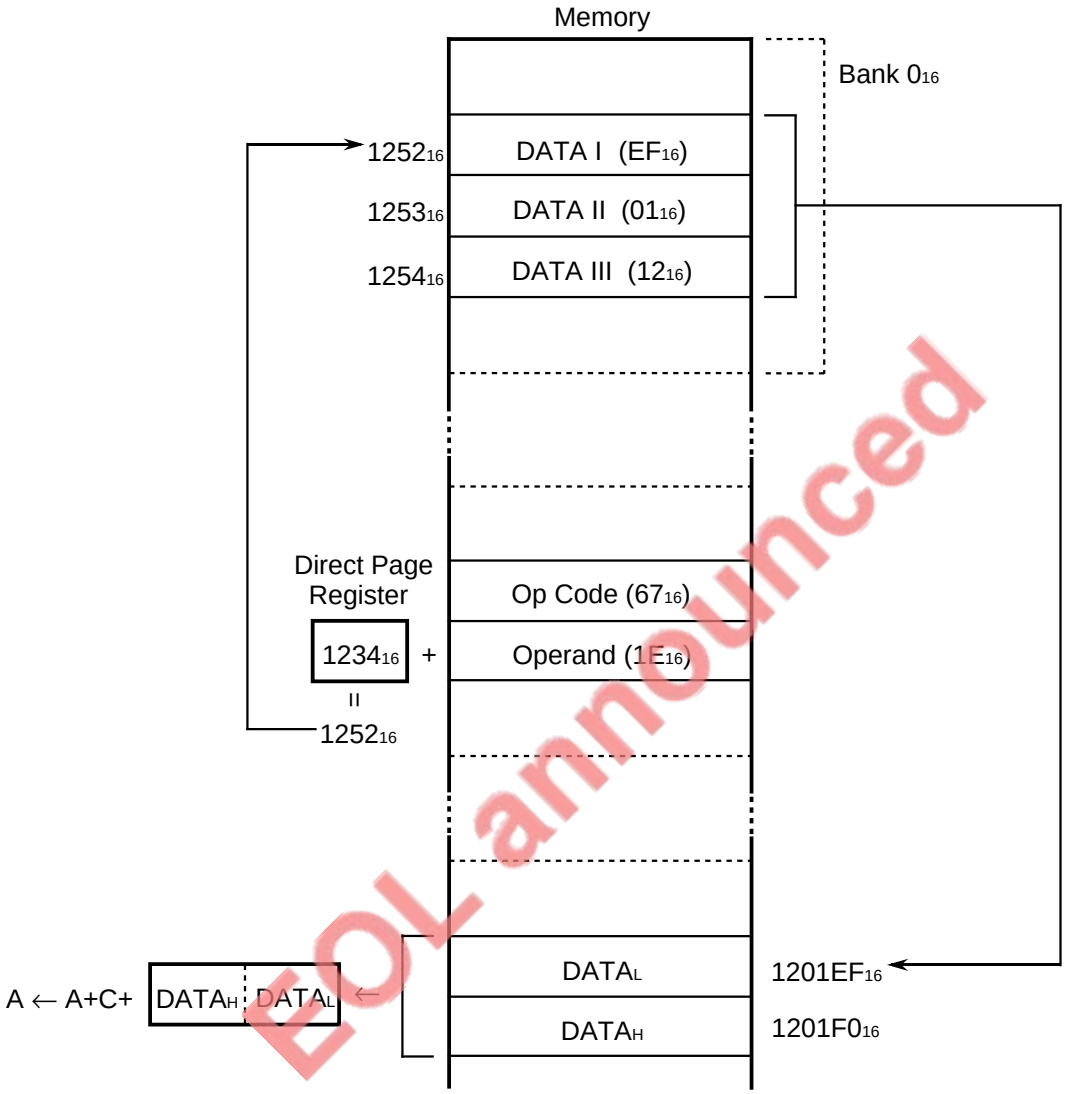
EOL announced

ex. : Mnemonic

ADCL A, (1EH)
(m=0)

Machine Code

67_{16} $1E_{16}$



EOL announced

Direct Indirect Long Indexed Y

Mode : Direct indirect long indexed Y addressing mode

Function : The value obtained by adding the instruction's second byte and the contents of the direct page register specifies 3 adjacent bytes in memory bank 0_{16}, and the value obtained by adding the contents of these bytes and the contents of the index register Y specifies the address of the memory location where the actual data is stored. If, however, the value obtained by adding the contents of the instruction's second byte and the direct page register exceeds the bank 0_{16} range, the specified location will be in bank 1_{16}. The 3 adjacent bytes memory location may be spread over two different banks.

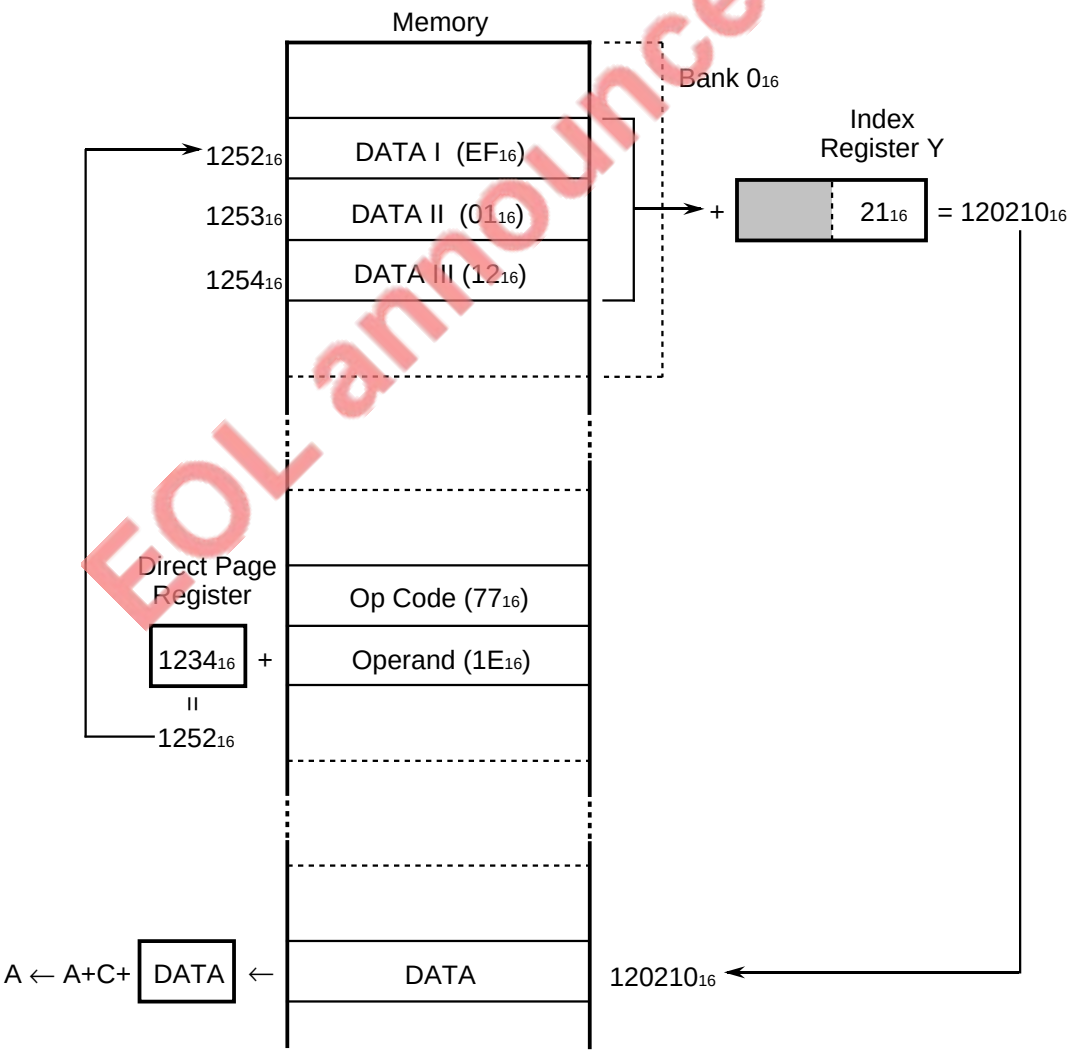
Instruction : ADC, AND, CMP, DIV, DIVS*, EOR, LDA, MPY, MPYS*, ORA, SBC, STA

ex. : Mnemonic

ADCL A, (1EH), Y
(m=1, x=1)

Machine Code

77_{16} $1E_{16}$

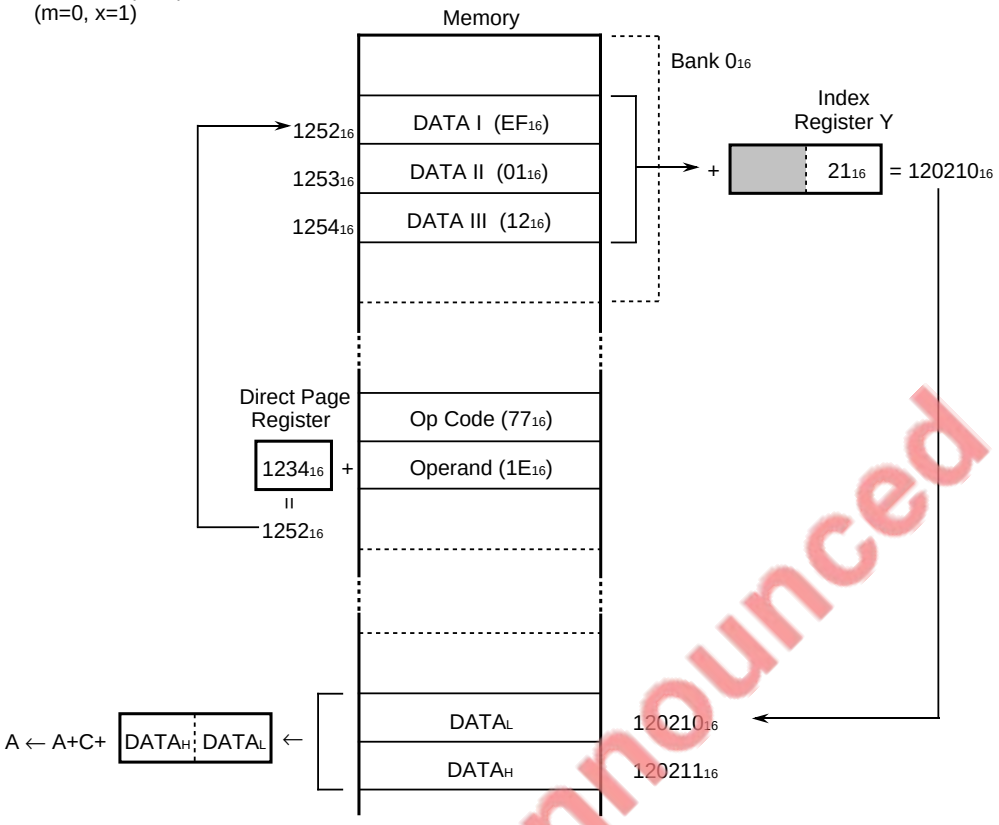


EOL announced

ex. : Mnemonic

ADCL A, (1EH), Y
(m=0, x=1)

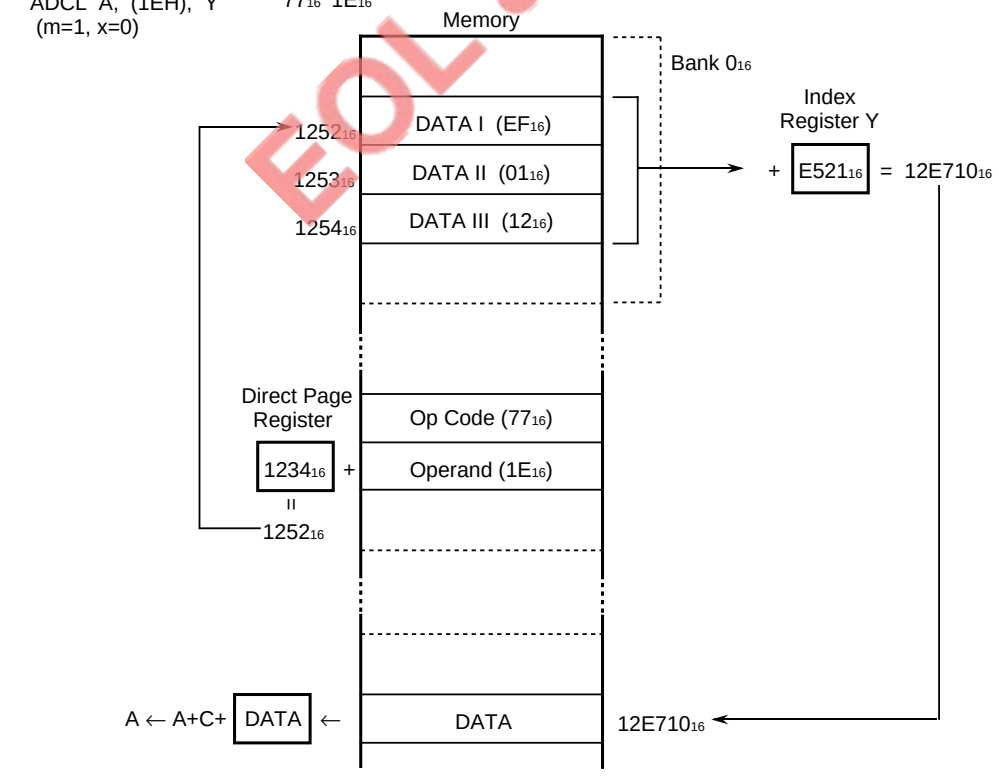
Machine Code
77_{16} $1E_{16}$



ex. : Mnemonic

ADCL A, (1EH), Y
(m=1, x=0)

Machine Code
77_{16} $1E_{16}$



EOL announced

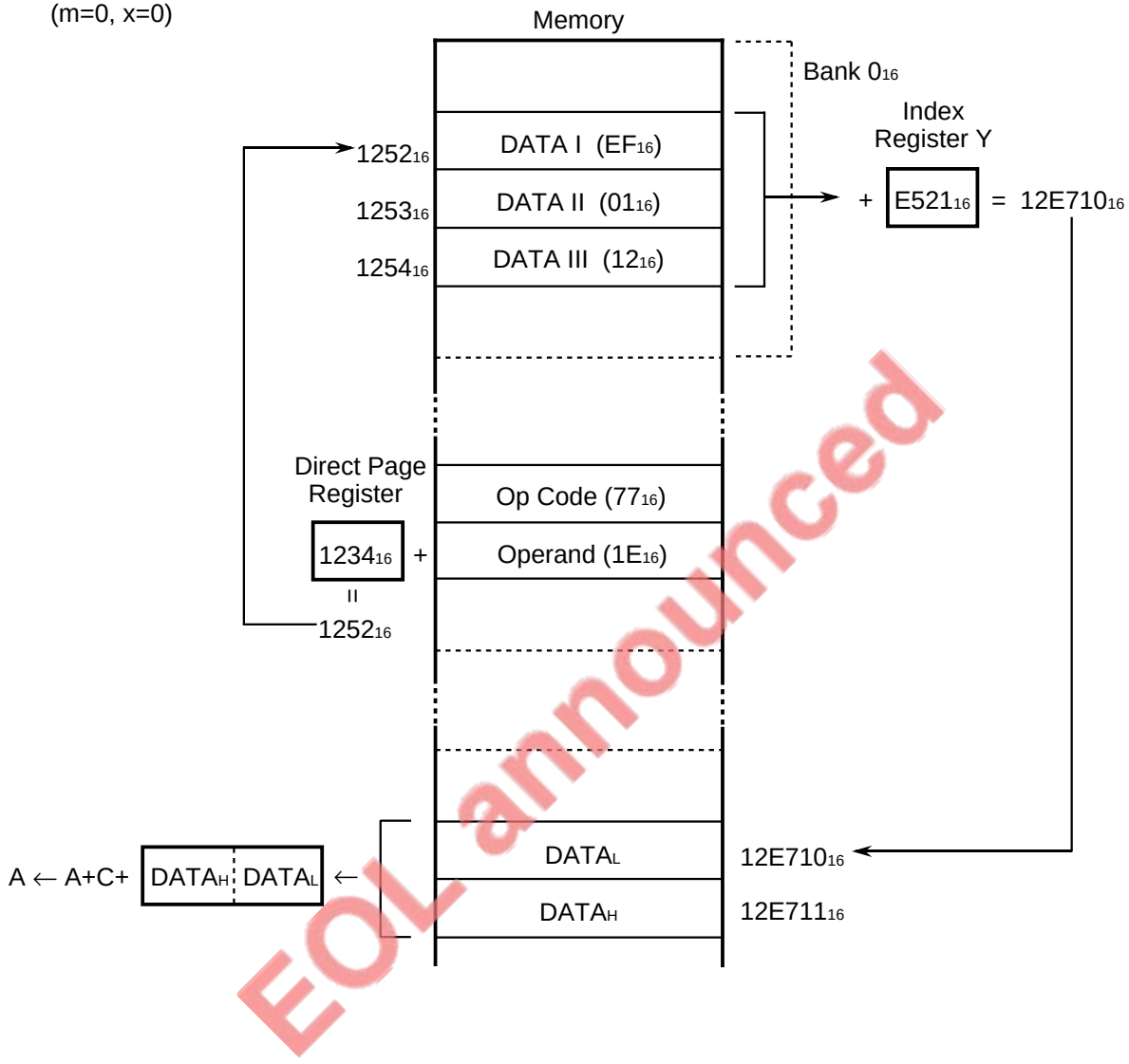
Direct Indirect Long Indexed Y

ex. : Mnemonic

ADCL A, (1EH), Y
(m=0, x=0)

Machine Code

77_{16} $1E_{16}$



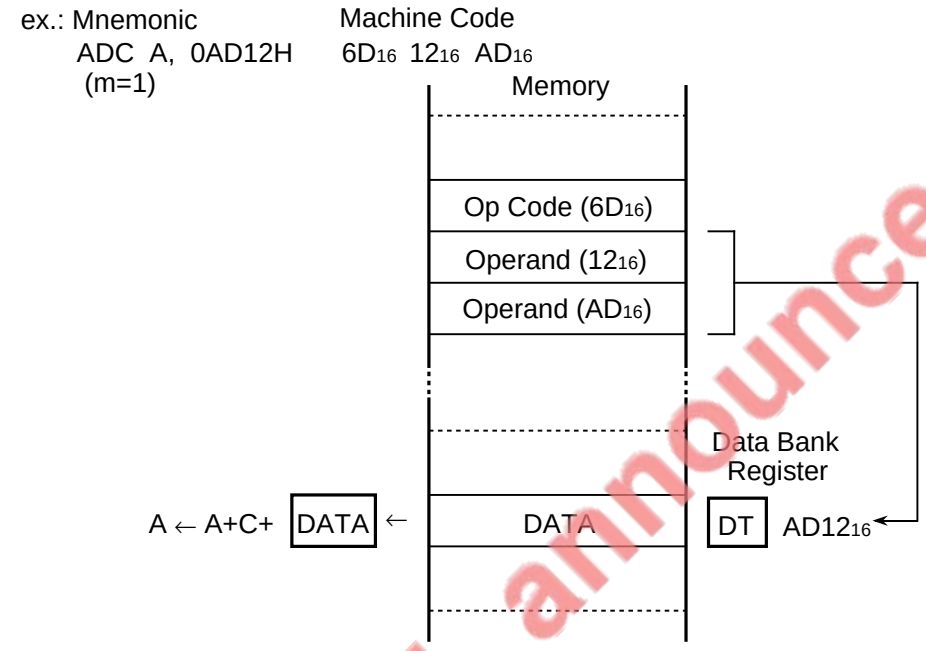
EOL announced

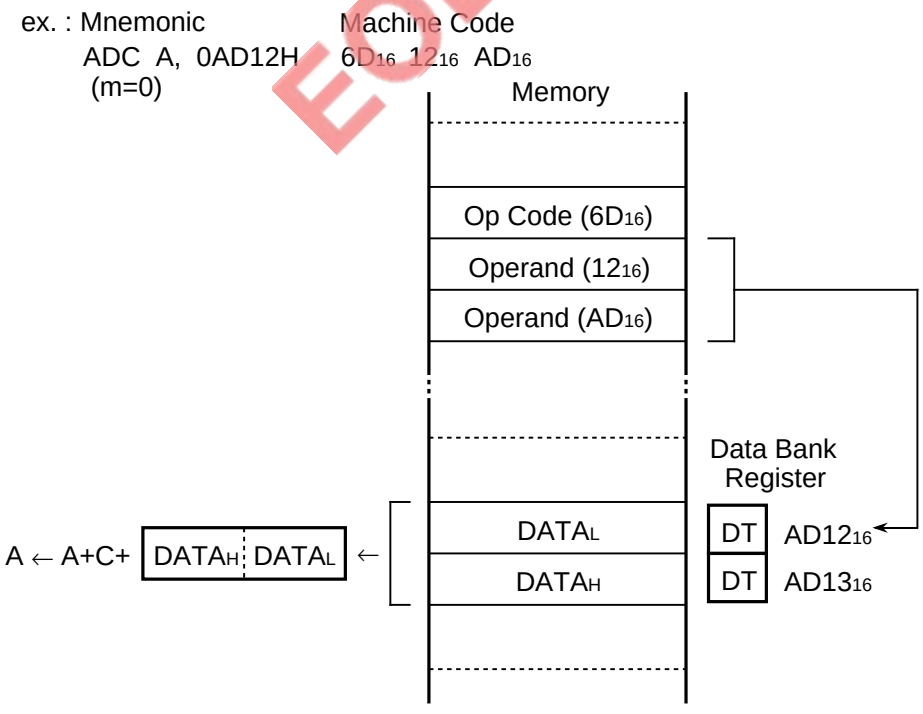
Absolute

Mode : Absolute addressing mode

Function : The contents of the memory locations specified by the instruction's second and third bytes and the contents of the data bank register are the actual data. Note that, in the cases of the JMP and JSR instructions, the instructions' second and third byte contents are transferred to the program counter.

Instruction : ADC, AND, ASL, ASR*, CMP, CPX, CPY, DEC, DIV, DIVS*, EOR, INC, JMP, JSR, LDA, LDM, LDX, LDY, LSR, MPY, MPYS*, ORA, ROL, ROR, SBC, STA, STX, STY

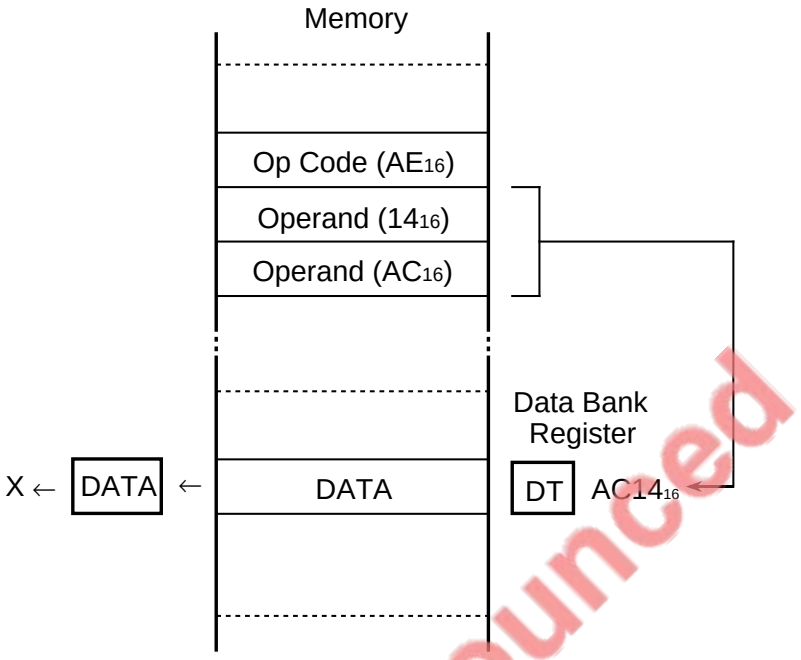




EOL announced

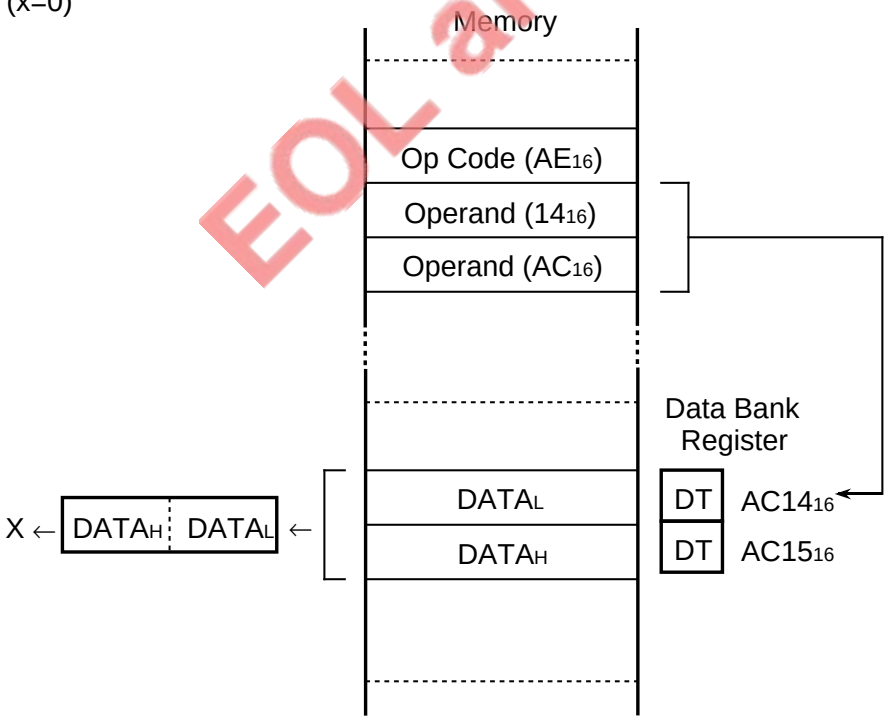
ex. : Mnemonic
LDX 0AC14H
(x=1)

Machine Code
AE_{16} 14_{16} AC_{16}



ex. : Mnemonic
LDX 0AC14H
(x=0)

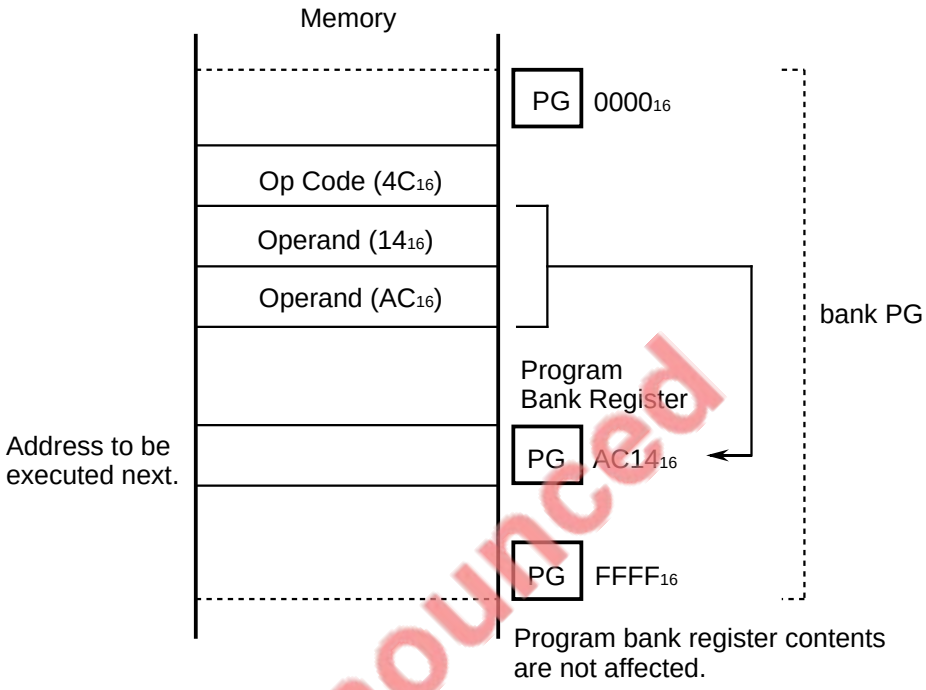
Machine Code
AE_{16} 14_{16} AC_{16}



EOL announced

ex. : Mnemonic
JMP 0AC14H

Machine Code
$4C_{16}$ 14_{16} AC_{16}



(Note) The branch destination bank must be considered carefully when a JMP or a JSR instruction is located near a bank boundary.
→Refer the description of a JMP instruction (Page 4-50).
Refer the description of a JSR instruction (Page 4-51).

EOL announced

Absolute Bit

Mode : Absolute bit addressing mode

Function : The contents of the instruction's second and third bytes and the contents of the data bank register specify the memory locations, and data for multiple bit positions in the memory locations are specified by a bit pattern specified in the instruction's fourth and fifth bytes (the fourth byte only if the m flag is set to 1).

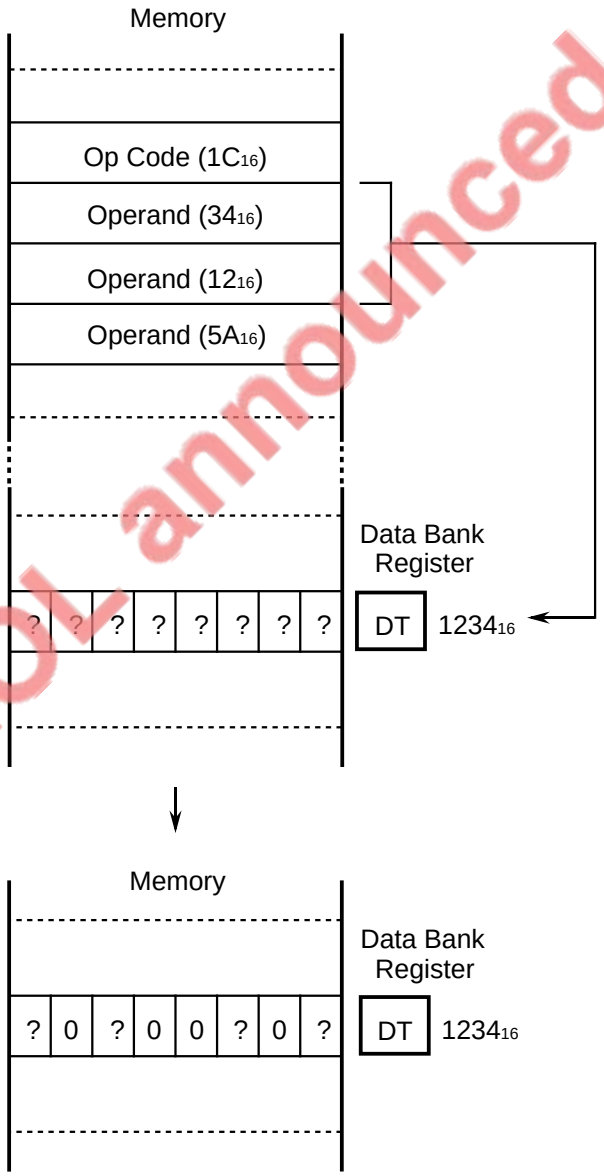
Instruction : CLB, SEB

ex. : Mnemonic

CLB #5AH, 1234H
(m=1)

Machine Code

$1C_{16}$ 34_{16} 12_{16} $5A_{16}$



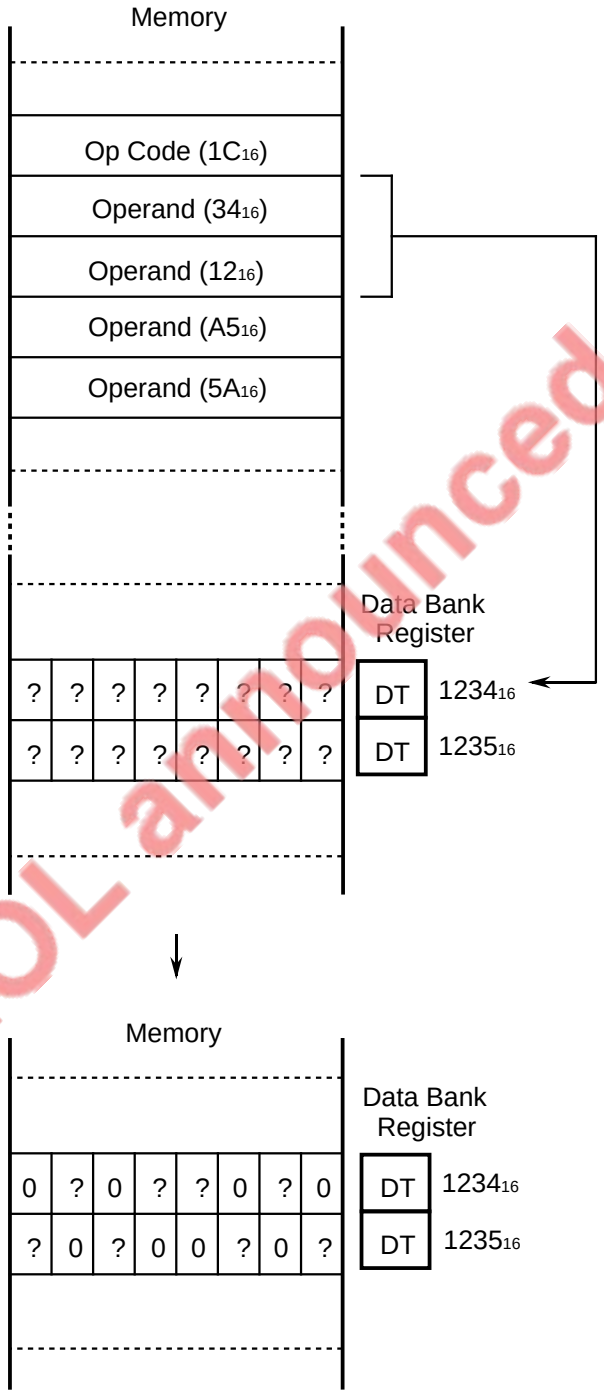
EOL announced

ex. : Mnemonic

CLB #5AA5H, 1234H
(m=0)

Machine Code

$1C_{16}$ 34_{16} 12_{16} $A5_{16}$ $5A_{16}$



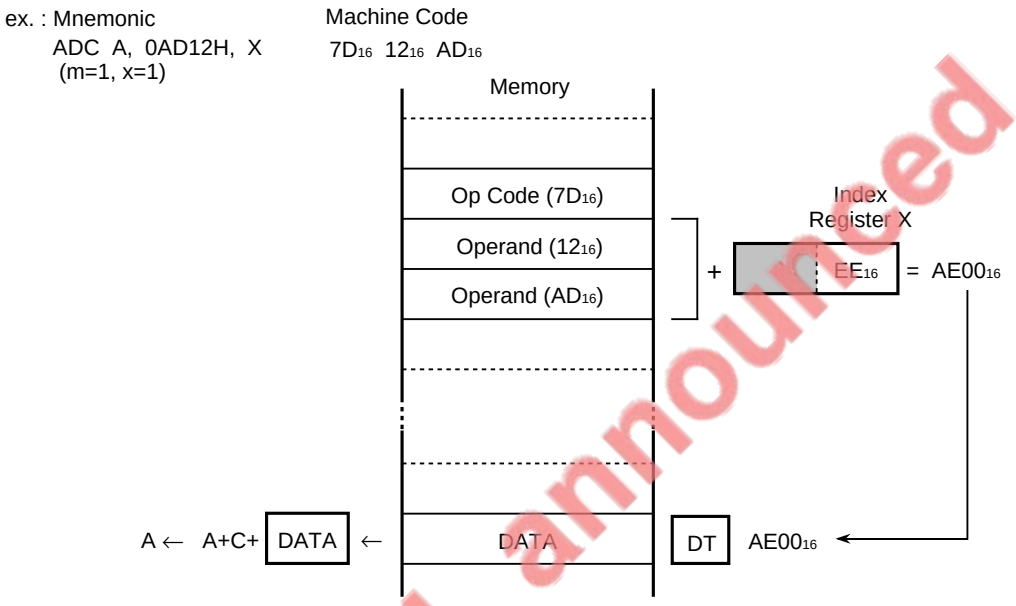
EOL announced

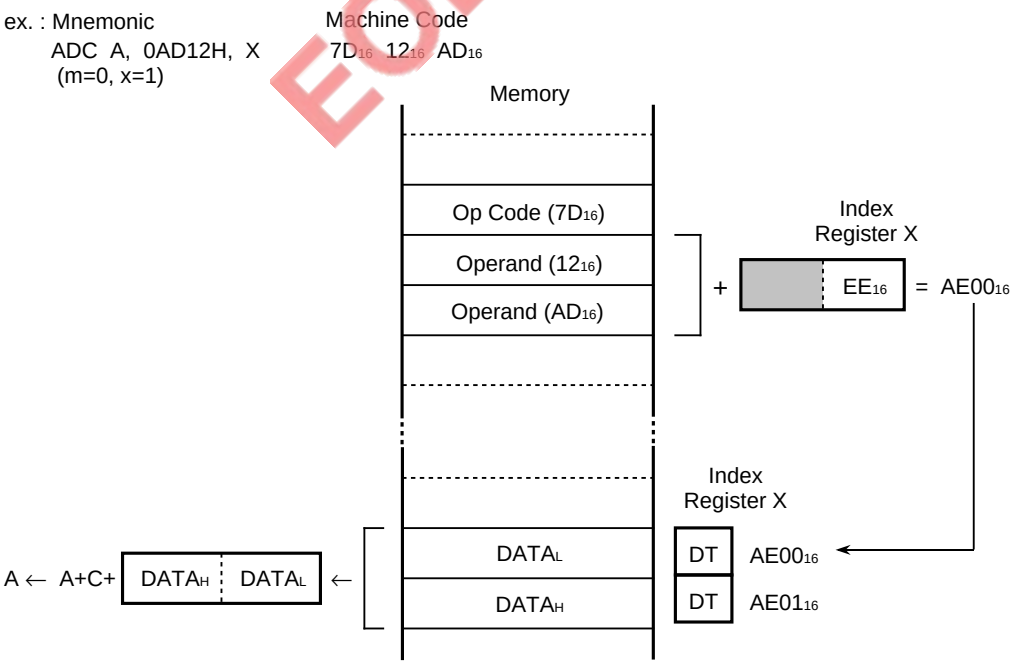
Absolute Indexed X

Mode : Absolute indexed X addressing mode

Function : The contents of the memory locations specified by a value resulting from addition of a 16-bit numeric value expressed by the instruction's second and third bytes with the contents of the index register X and the contents of the data bank register are the actual data. If, however, addition of the numeric value expressed by the instruction's second and third bytes with the contents of the index register X generates a carry, the bank number will be 1 larger than the contents of the data bank register.

Instruction : ADC, AND, ASL, ASR*, CMP, DEC, DIV, DIVS*, EOR, INC, LDA, LDM, LDY, LSR, MPY, MPYS*, ORA, ROL, ROR, SBC, STA

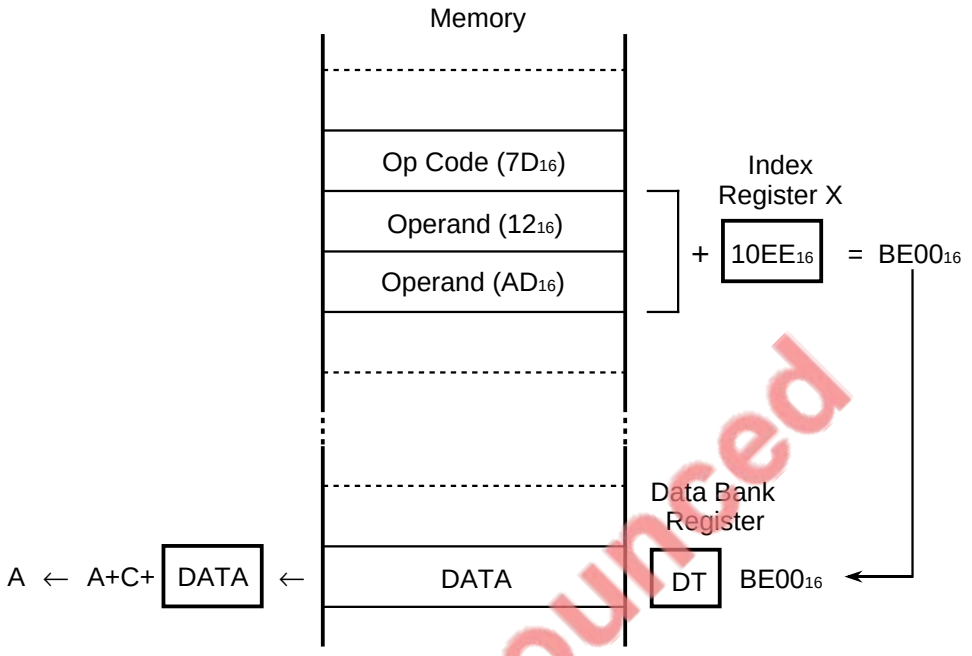




EOL announced

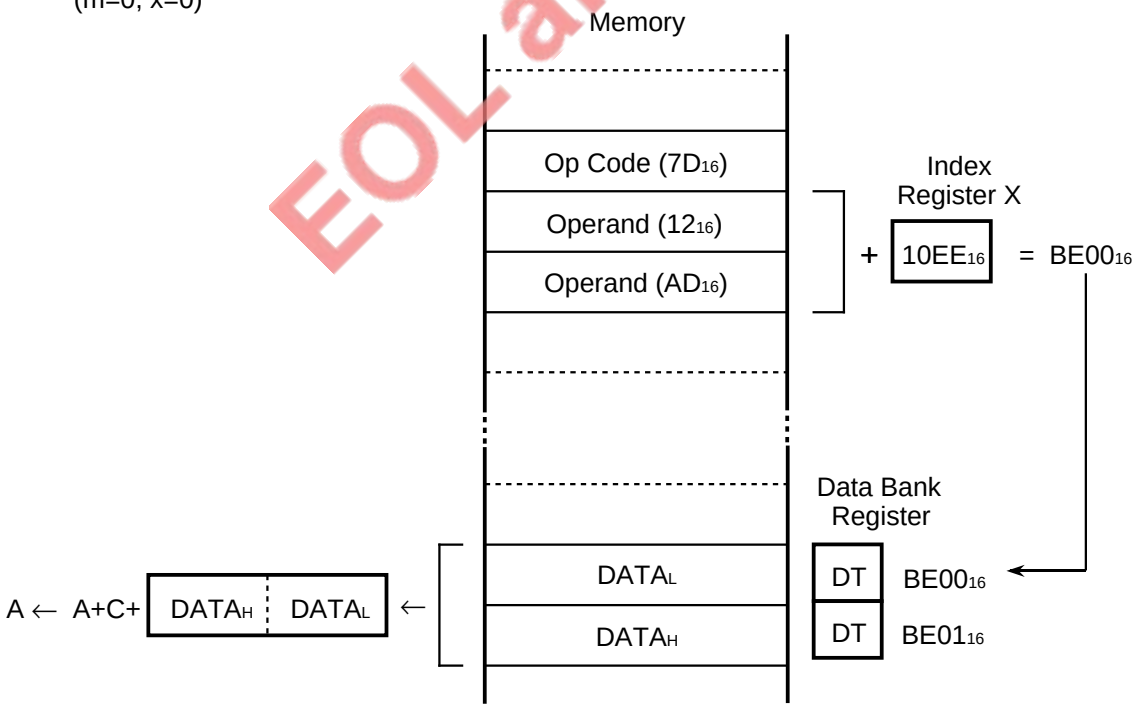
ex. : Mnemonic
ADC A, 0AD12H, X
(m=1, x=0)

Machine Code
$7D_{16}$ 12_{16} AD_{16}



ex. : Mnemonic
ADC A, 0AD12H, X
(m=0, x=0)

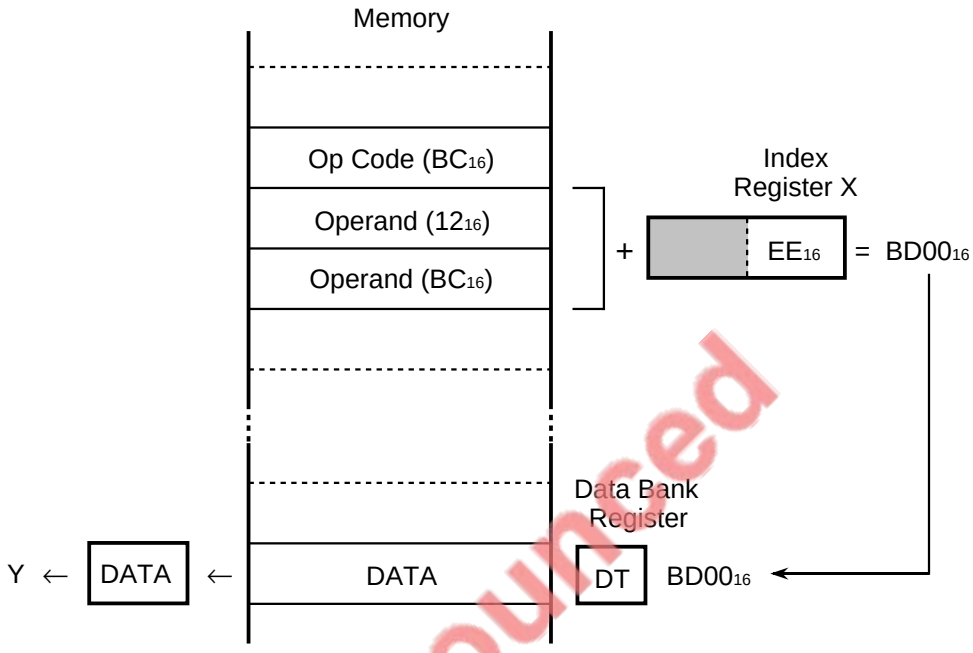
Machine Code
$7D_{16}$ 12_{16} AD_{16}



EOL announced

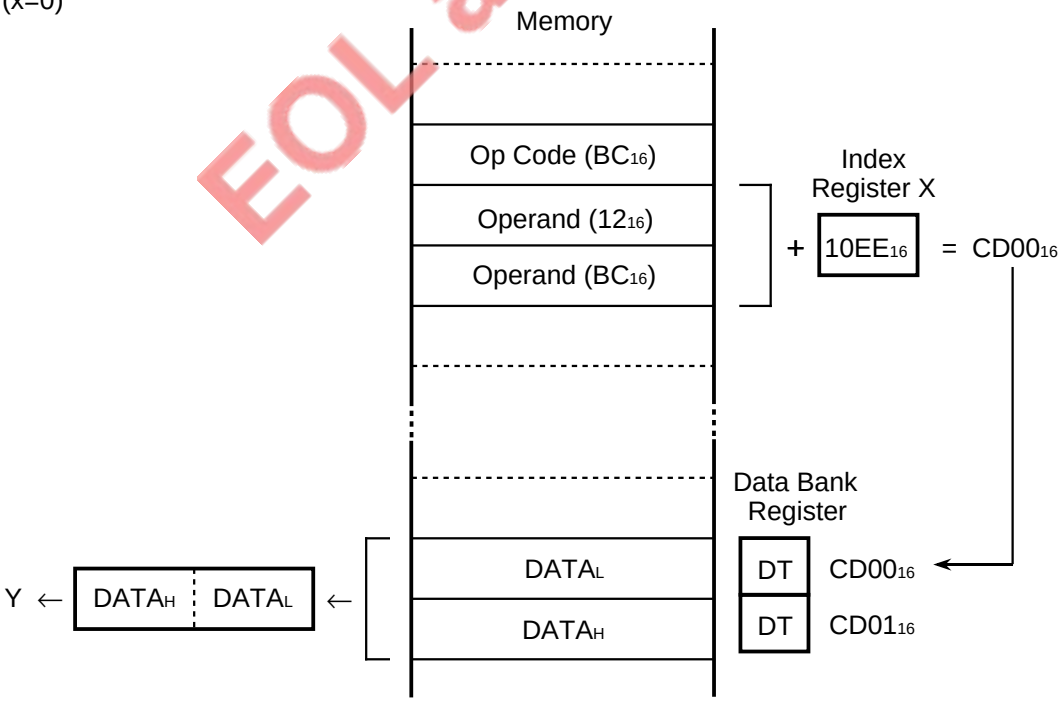
ex. : Mnemonic
LDY 0BC12H, X
(x=1)

Machine Code
BC_{16} 12_{16} BC_{16}



ex. : Mnemonic
LDY 0BC12H, X
(x=0)

Machine Code
BC_{16} 12_{16} BC_{16}



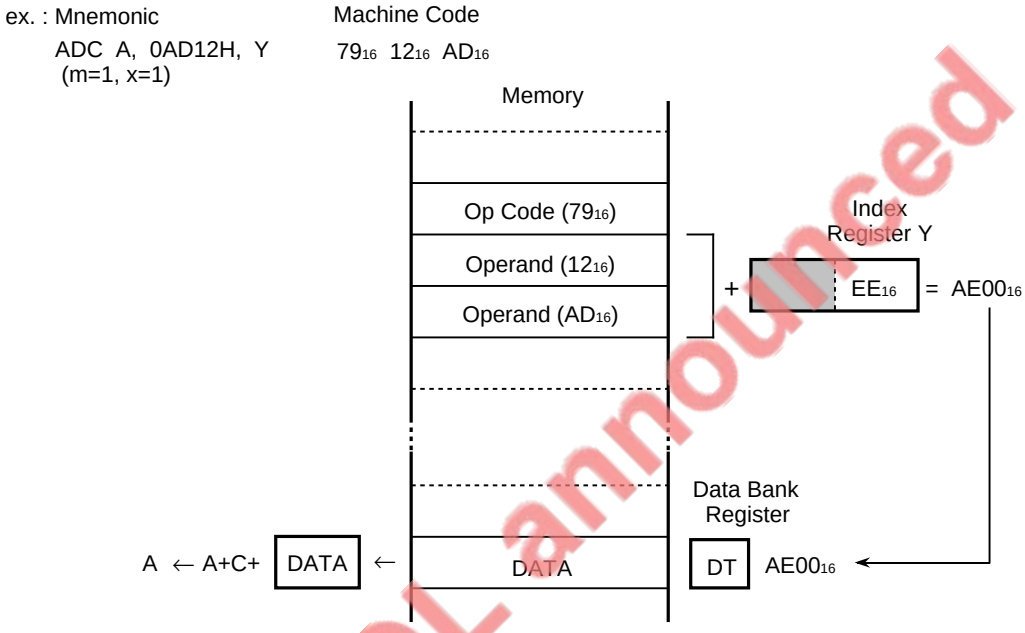
EOL announced

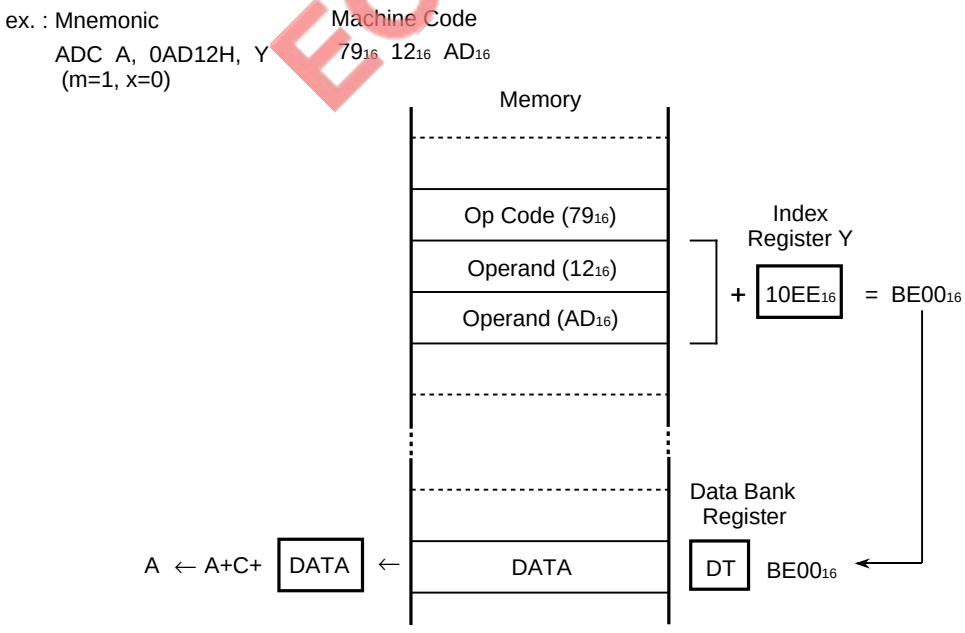
Absolute Indexed Y

Mode : Absolute indexed Y addressing mode

Function : The contents of the memory locations specified by a value resulting from addition of a 16-bit numeric value expressed by the instruction's second and third bytes with the contents of the index register Y and the contents of the data bank register are the actual data. If, however, addition of the numeric value expressed by the instruction's second and third bytes with the contents of the index register Y generates a carry, the bank number will be 1 larger than the contents of the data bank register.

Instruction : ADC, AND, CMP, DIV, DIVS*, EOR, LDA, LDX, MPY, MPYS*, ORA, SBC, STA



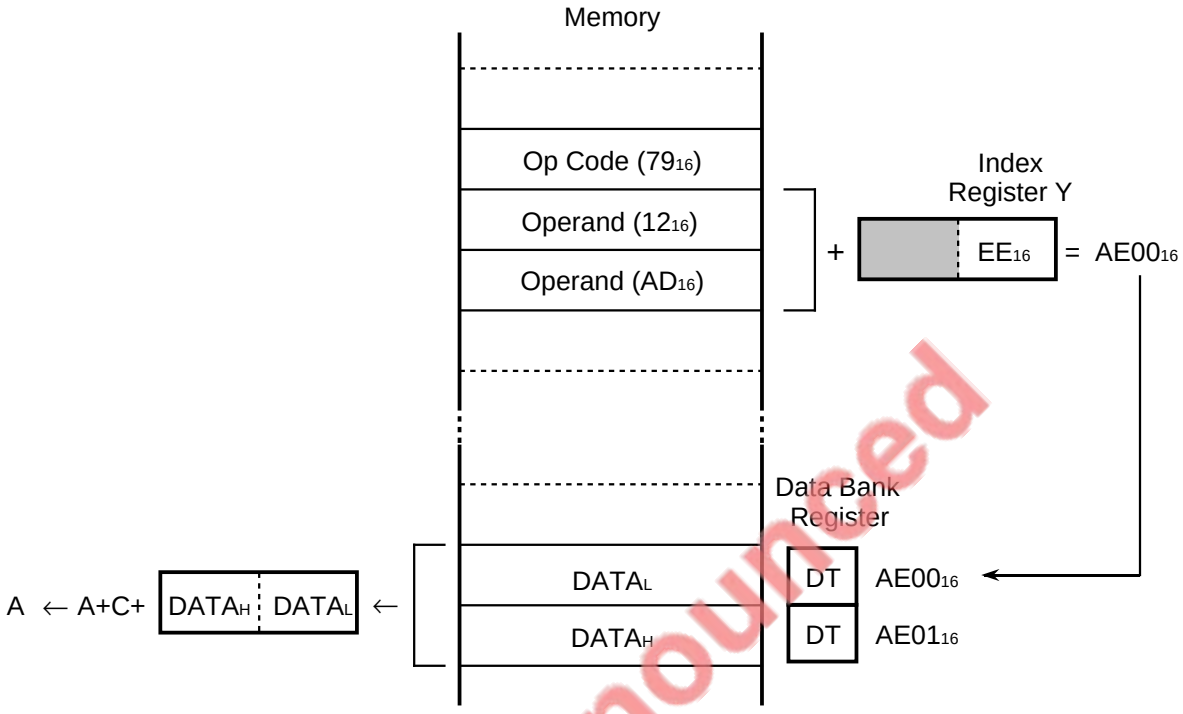


EOL announced

Absolute Indexed Y

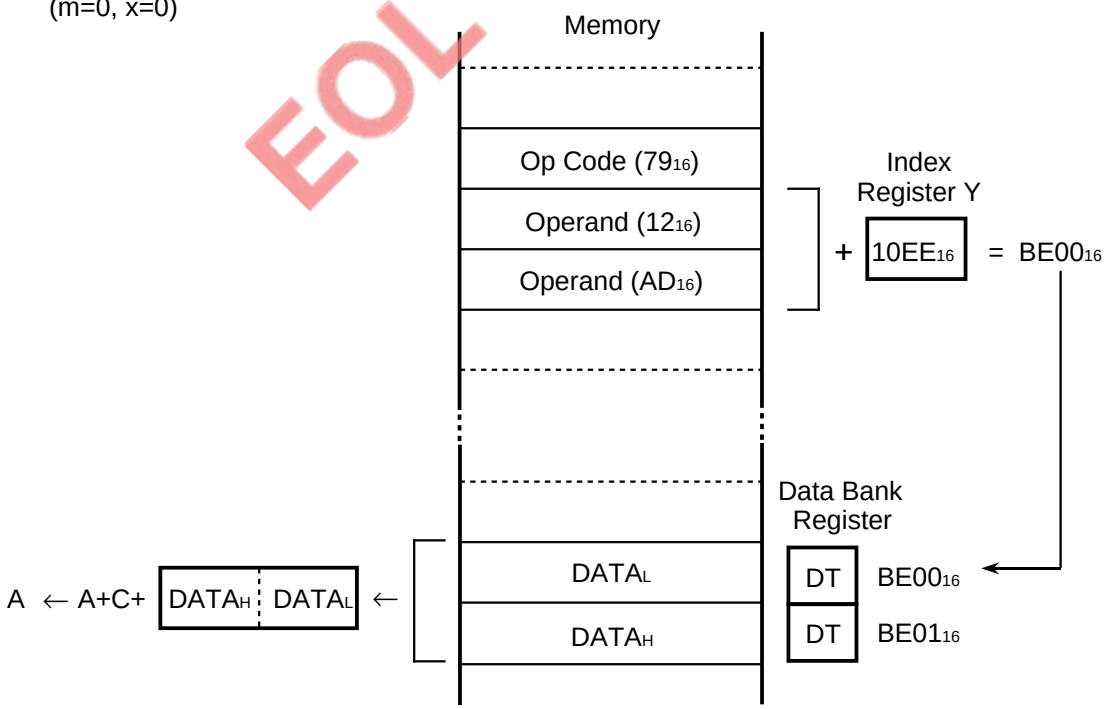
ex. : Mnemonic
ADC A, 0AD12H, Y
(m=0, x=1)

Machine Code
79_{16} 12_{16} AD_{16}



ex. : Mnemonic
ADC A, 0AD12H, Y
(m=0, x=0)

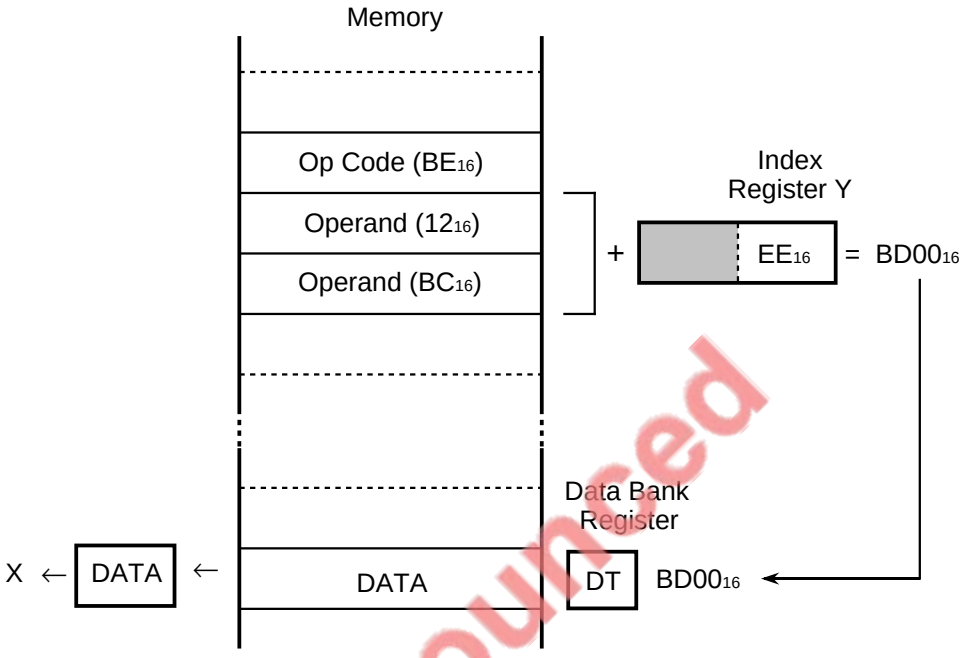
Machine Code
79_{16} 12_{16} AD_{16}



EOL announced

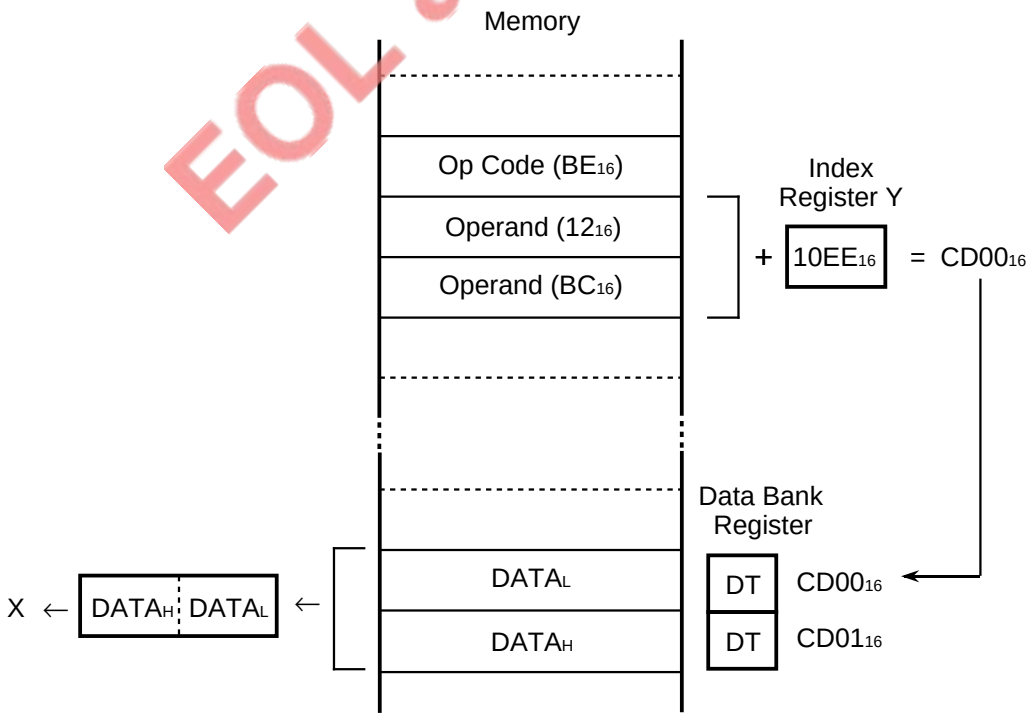
ex. : Mnemonic
LDX 0BC12H, Y
(x=1)

Machine Code
BE_{16} 12_{16} BC_{16}



ex. : Mnemonic
LDX 0BC12H, Y
(x=0)

Machine Code
BE_{16} 12_{16} BC_{16}



EOL announced

Absolute Long

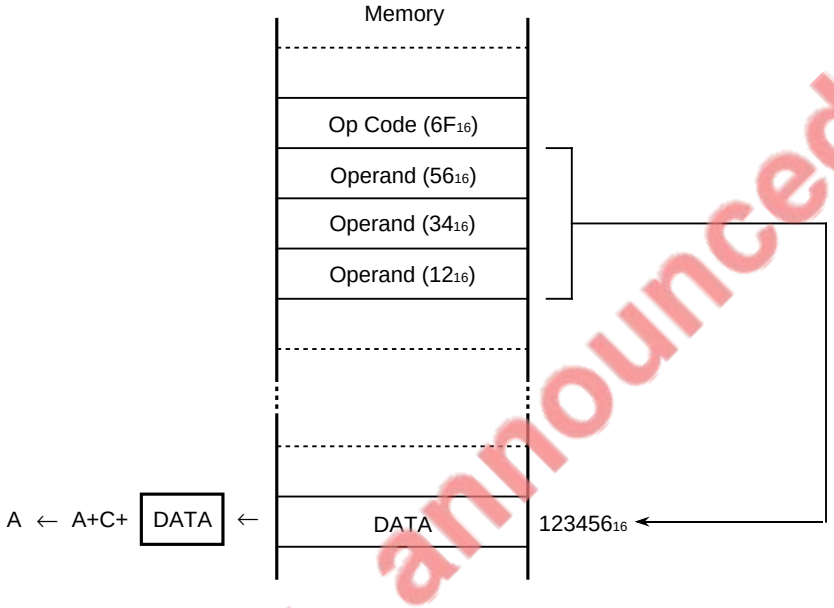
Mode : Absolute long addressing mode

Function : The contents of the memory locations specified by the instruction's second, third and fourth bytes become the actual data. Note that, in the cases of the JMP and JSR instructions, the instructions' second and third byte contents are transferred to the program counter and the fourth byte contents are transferred to the program bank register.

Instruction : ADC, AND, CMP, DIV, DIVS*, EOR, JMP, JSR, LDA, MPY, MPYS*, ORA, SBC, STA

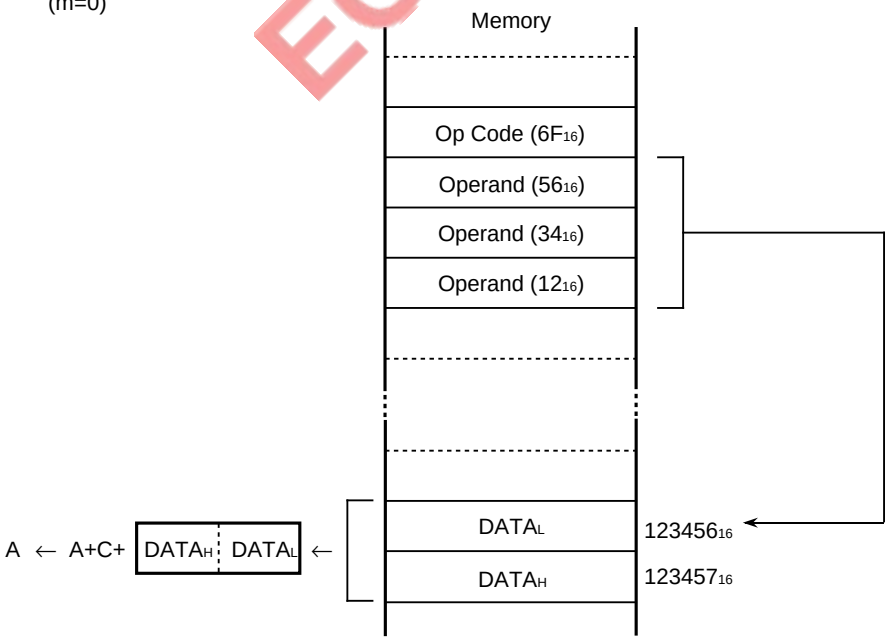
ex. : Mnemonic
ADC A, 123456H
(m=1)

Machine Code
$6F_{16}$ 56_{16} 34_{16} 12_{16}



ex. : Mnemonic
ADC A, 123456H
(m=0)

Machine Code
$6F_{16}$ 56_{16} 34_{16} 12_{16}

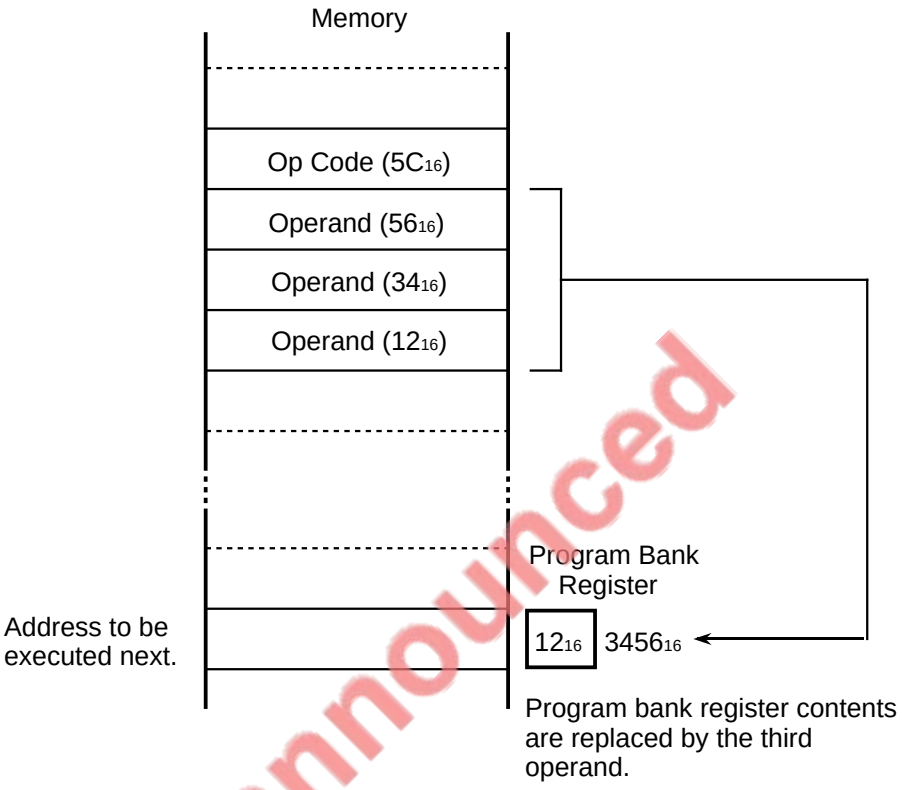


EOL announced

Absolute Long

ex. : Mnemonic
JMPL 123456H

Machine Code
$5C_{16}$ 56_{16} 34_{16} 12_{16}



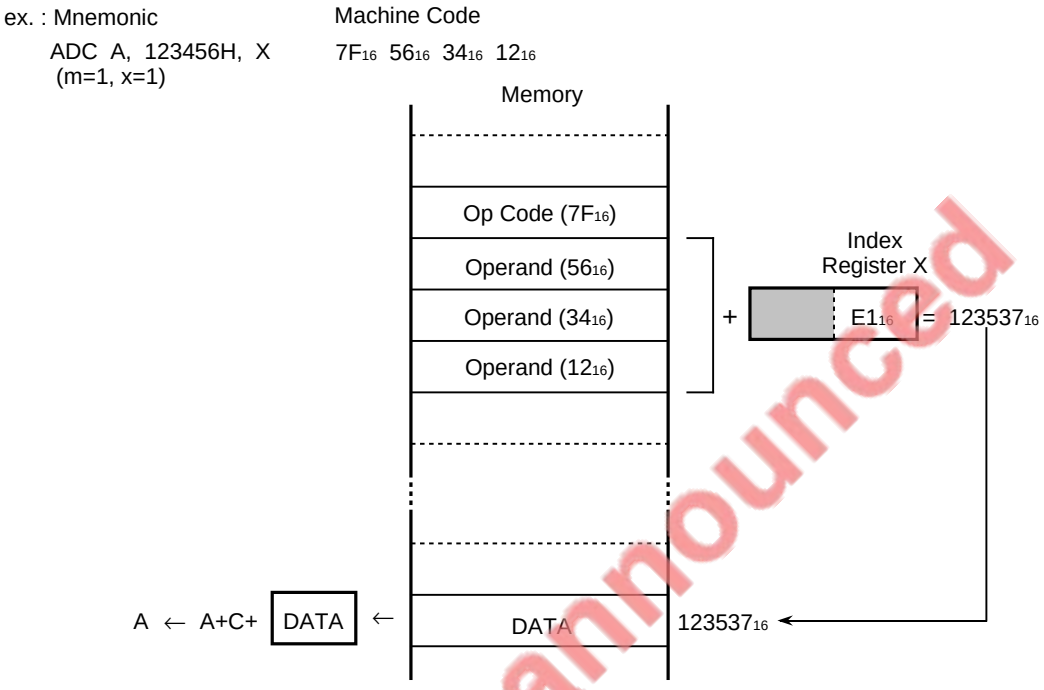
EOL announced

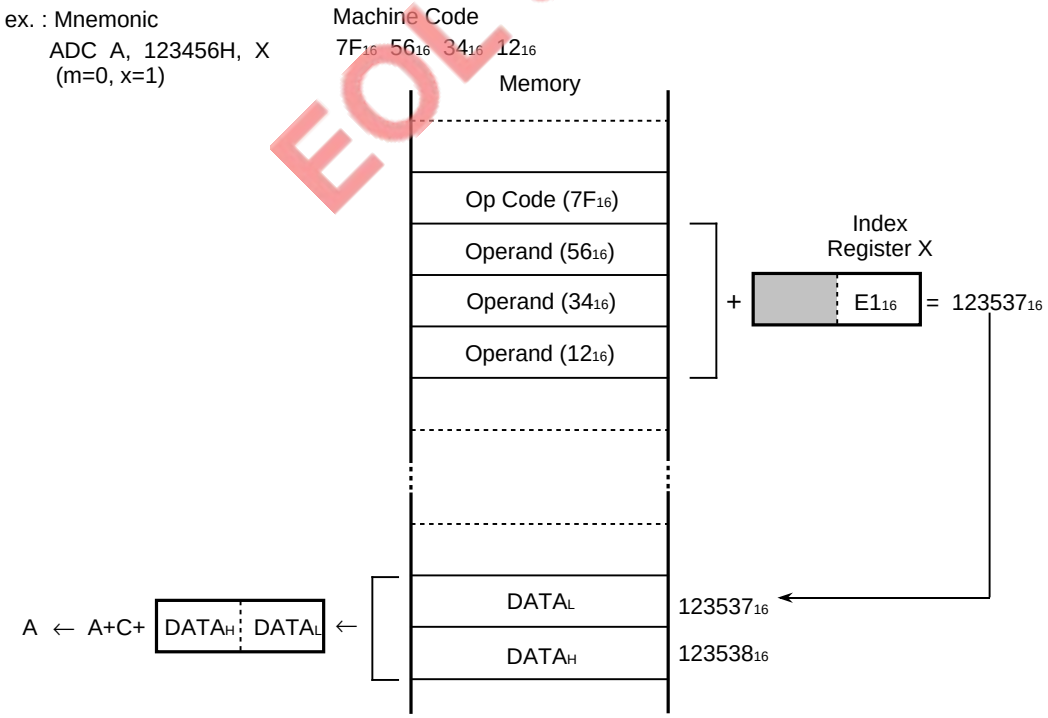
Absolute Long Indexed X

Mode : Absolute long indexed X addressing mode

Function : The contents of the memory location specified by adding the numeric value expressed by the instruction's second, third and fourth bytes with the contents of the index register X are the actual data.

Instruction : ADC, AND, CMP, DIV, DIVS*, EOR, LDA, MPY, MPYS*, ORA, SBC, STA



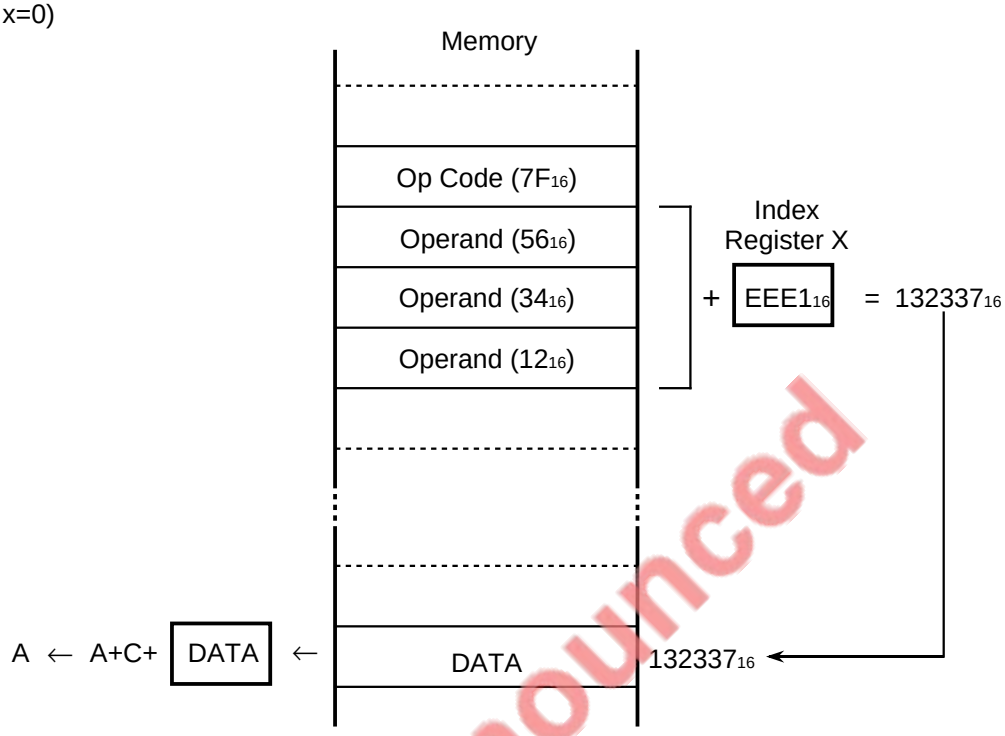


EOL announced

Absolute Long Indexed X

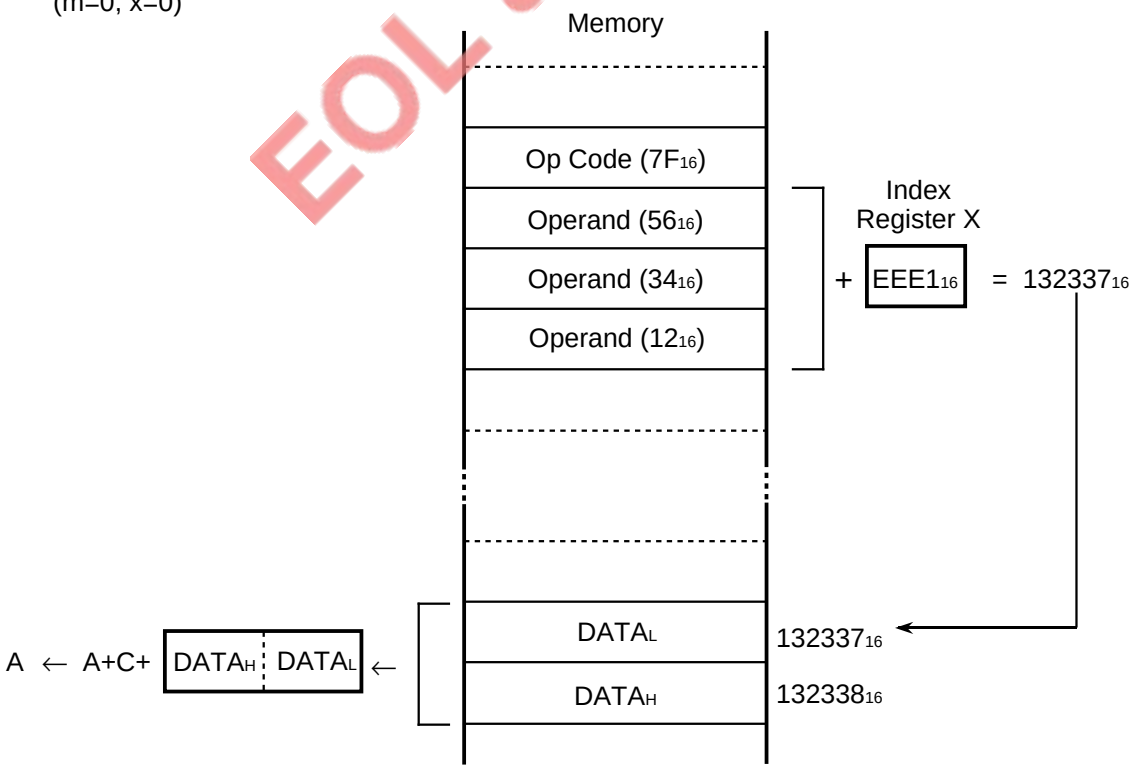
ex. : Mnemonic
ADC A, 123456H, X
(m=1, x=0)

Machine Code
$7F_{16}$ 56_{16} 34_{16} 12_{16}



ex. : Mnemonic
ADC A, 123456H, X
(m=0, x=0)

Machine Code
$7F_{16}$ 56_{16} 34_{16} 12_{16}



EOL announced

Absolute Indirect

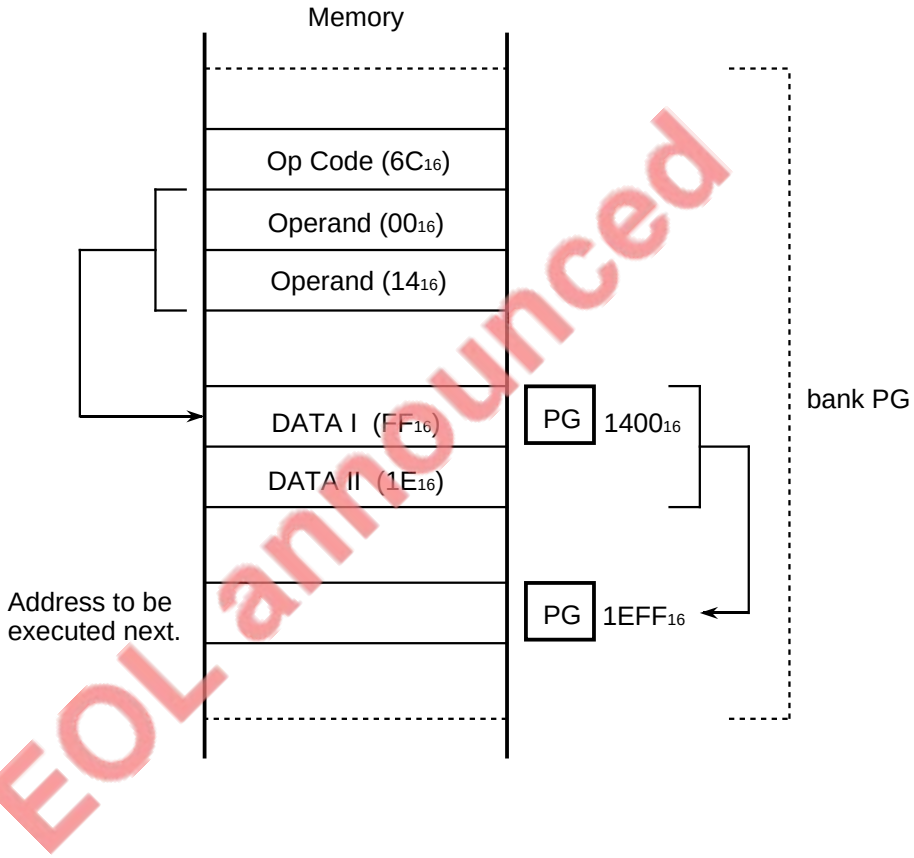
Mode : Absolute indirect addressing mode

Function : The instruction's second and third bytes specify 2 adjacent bytes in memory, and the contents of these bytes specify the address within the same program bank to which a jump is to be made.

Instruction : JMP

ex. : Mnemonic
JMP (1400H)

Machine Code
$6C_{16}$ 00_{16} 14_{16}



EOL announced

(Note) The branch destination bank must be considered carefully when a JMP instruction is located near a bank boundary.
→Refer the description of a JMP instruction (Page 4-50).

Absolute Indirect Long

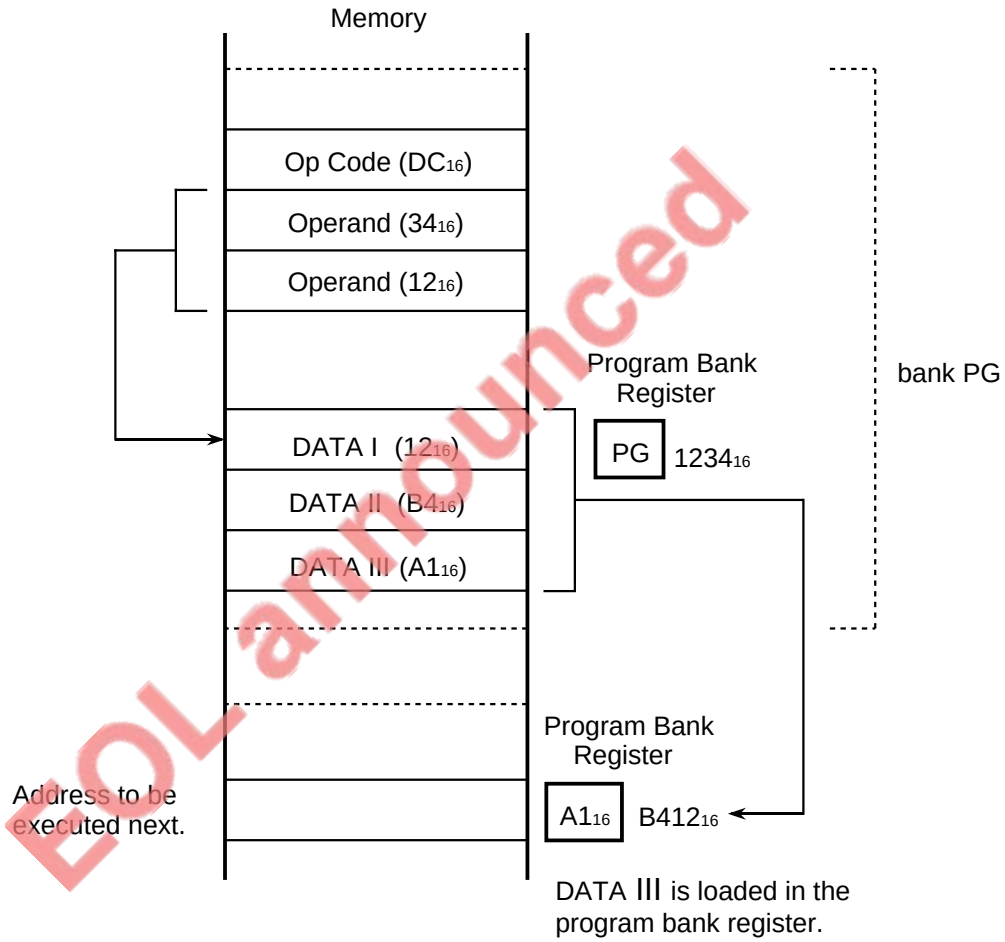
Mode : Absolute indirect long addressing mode

Function : The instruction's second and third bytes specify 3 adjacent bytes in memory, and the contents of these bytes specify the address to which a jump is to be made.

Instruction : JMP

ex. : Mnemonic
JMPL (1234H)

Machine Code
DC_{16} 34_{16} 12_{16}



EOL announced

(Note) The branch destination bank must be considered carefully when a JMP instruction is located near a bank boundary.
→Refer the description of a JMP instruction (Page 4-50).

Absolute Indexed X Indirect

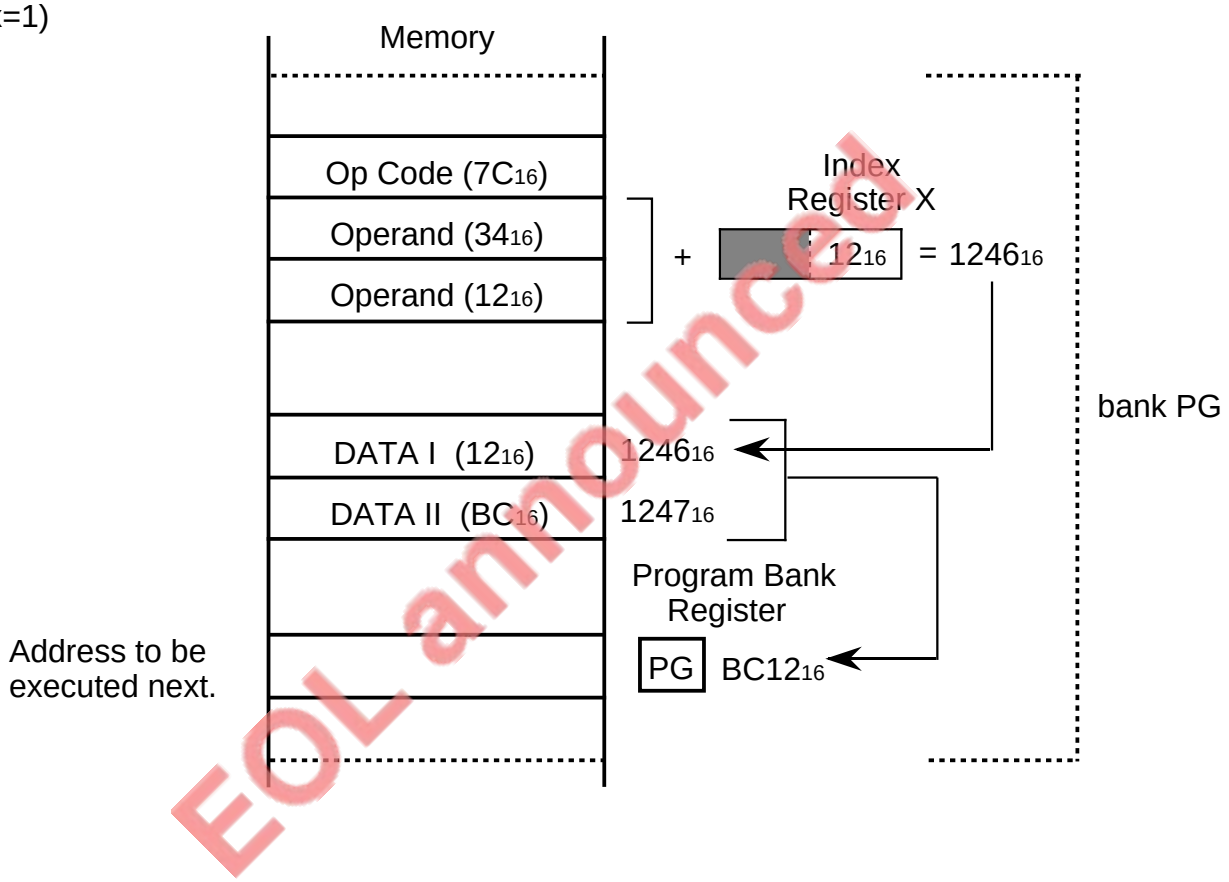
Mode : Absolute indexed X indirect addressing mode

Function : The value obtained by adding the instruction's second and third bytes and the contents of the index register X specifies 2 adjacent bytes in memory, and the contents of these bytes specify the address to which a jump is to be made.

Instruction : JMP, JSR

ex. : Mnemonic
JMP (1234H, X)
(x=1)

Machine Code
$7C_{16}$ 34_{16} 12_{16}



EOL announced

(Note) The branch destination bank must be considered carefully when a JMP or a JSR instruction is located near a boundary.
→Refer the description of a JMP instruction (Page 4-50).
Refer the description of a JSR instruction (Page 4-51).

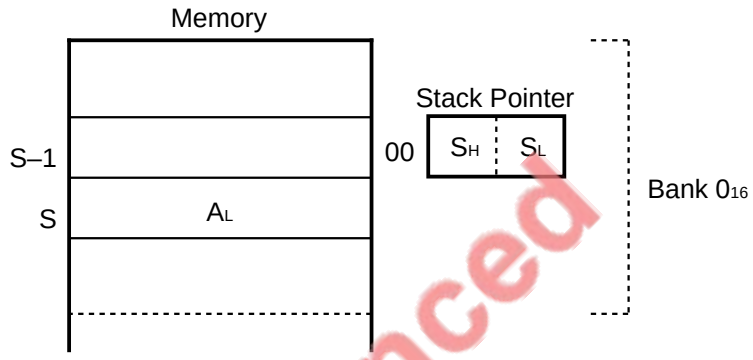
Mode : Stack addressing mode

Function : Register contents are saved to or restored from the memory location specified by the stack pointer. The stack pointer is set in bank-0.

Instruction : PEA, PEI, PER, PHA, PHB, PHD, PHG, PHP, PHT, PHX, PHY, PLA, PLB, PLD, PLP, PLT, PLX, PLY, PSH, PUL

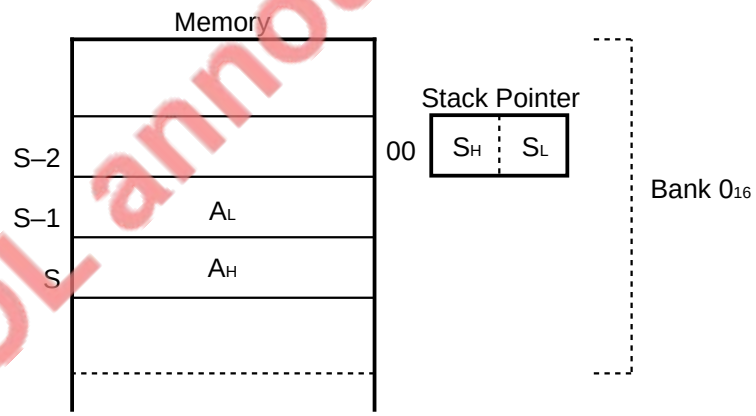
ex. : Mnemonic
PHA
(m=1)

Machine Code
48_{16}



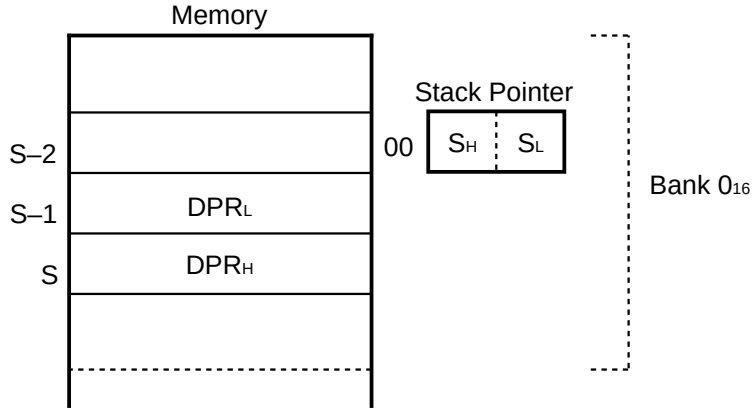
ex. : Mnemonic
PHA
(m=0)

Machine Code
48_{16}



ex. : Mnemonic
PHD

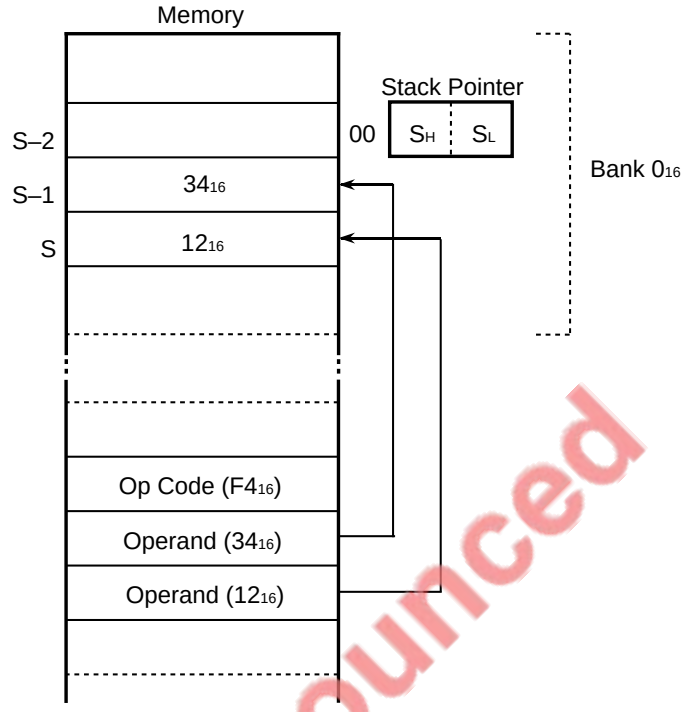
Machine Code
$0B_{16}$



EOL announced

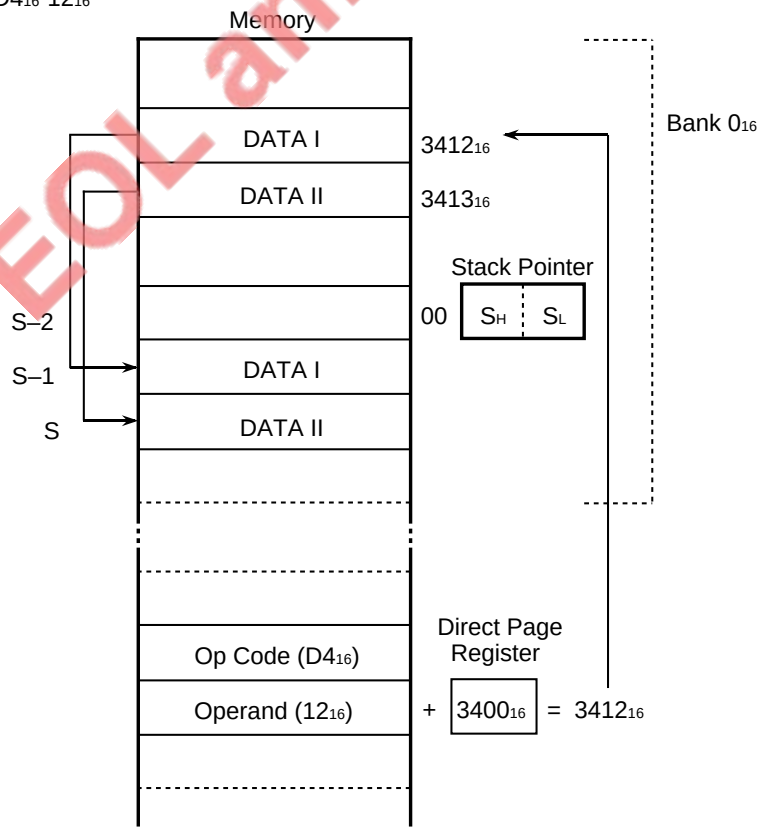
ex. : Mnemonic
PEA #1234H

Machine Code
$F4_{16}$ 34_{16} 12_{16}



ex. : Mnemonic
PEI #12H

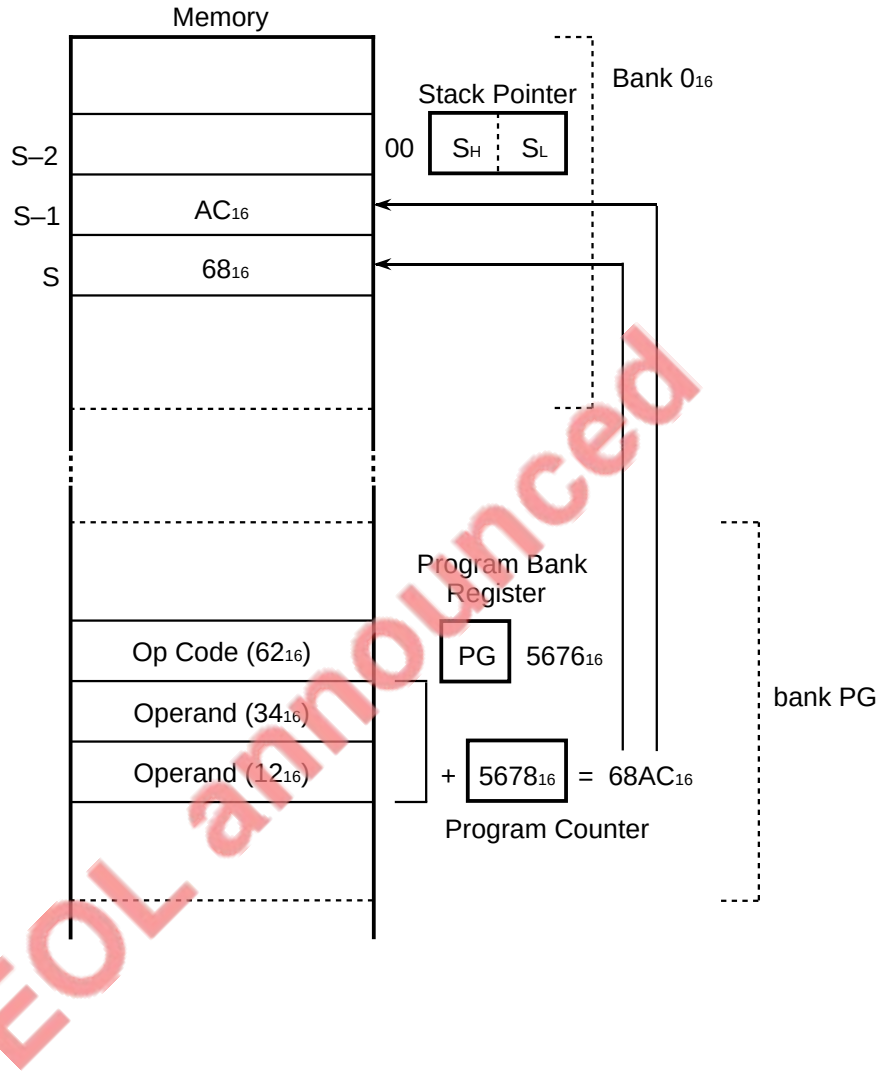
Machine Code
$D4_{16}$ 12_{16}



EOL announced

ex. : Mnemonic
PER #1234H

Machine Code
62_{16} 34_{16} 12_{16}



EOL announced

Relative

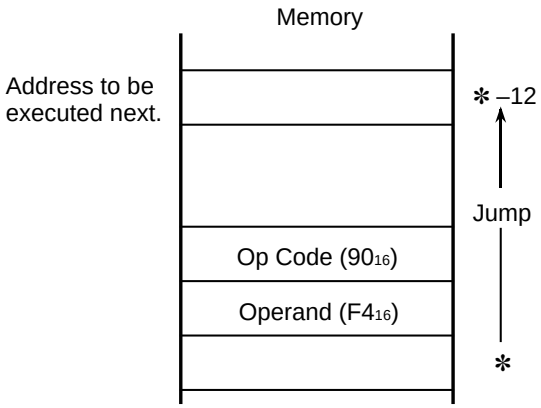
Mode : Relative addressing mode

Function : Branching occurs to the address specified by the value resulting from addition of the contents of the program counter and the instruction's second byte. In the case of a long branch by the BRA instruction, a 15-bit signed numeric value formed by the contents of the instruction's second and third bytes is added to the program counter contents. If the addition generates a carry or borrow, 1 is added to or subtracted from the program bank register.

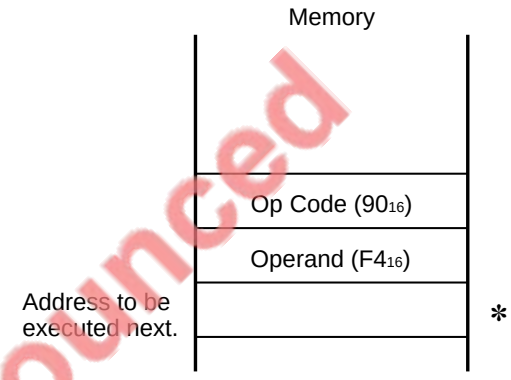
Instruction : BCC, BCS, BEQ, BMI, BNE, BPL, BRA, BVC, BVS

ex. : Mnemonic
BCC ✻ –12

Machine Code
90_{16} $F4_{16}$



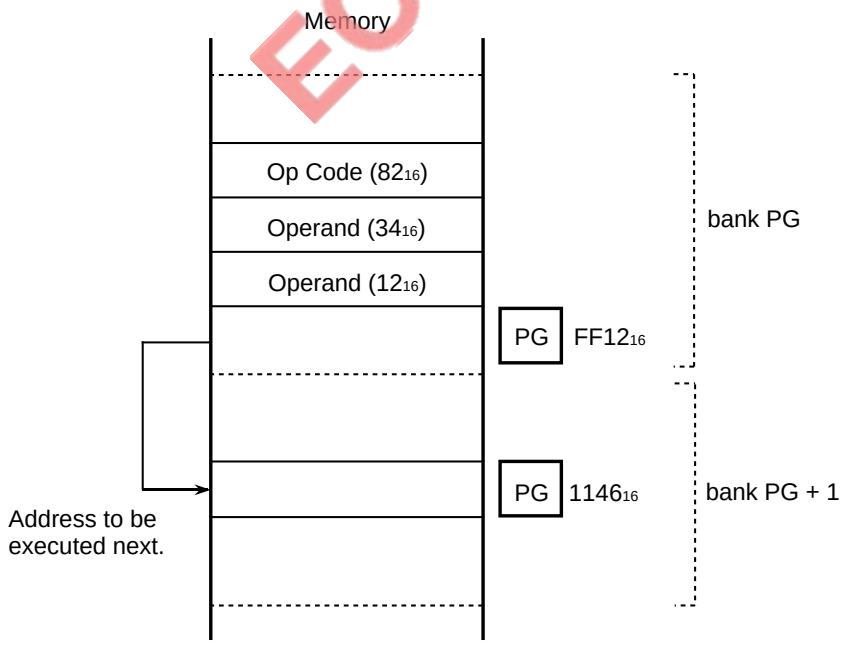
Branches to the address ✻ –12 if the carry flag (C) has been cleared "0".



Advances to the address ✻ if the carry flag (C) has been set "1".

ex. : Mnemonic
BRAL 1234H

Machine Code
82_{16} 34_{16} 12_{16}



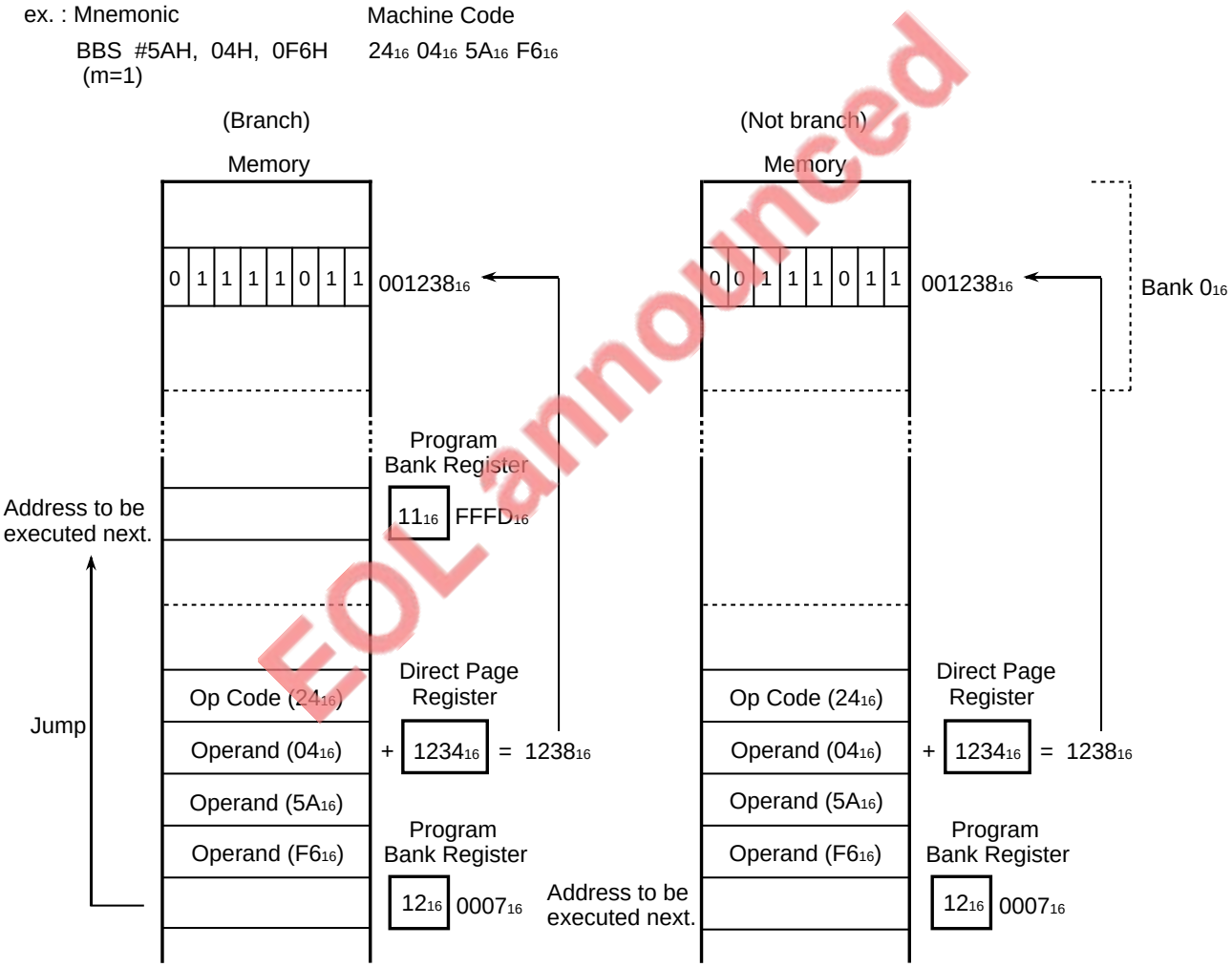
EOL announced

Direct Bit Relative

Mode : Direct bit relative addressing mode

Function : Specifies the bank 0_{16} memory location by the value obtained by adding the instruction's second byte to the direct page register's contents, and specifies the positions of multiple bits in the memory location by the bit pattern in the third and fourth bytes (the third byte only if the m flag is set to 1). Then, if the specified bits all satisfy the branching conditions, the instruction's fifth byte (or the fourth byte if the m flag is set to 1) is added to the program counter as a signed value, generating the branching destination address. If, however, addition of the instruction's second byte to the direct page register's contents result in a value that exceeds the bank 0_{16} range, the specified location will be in bank 1_{16}.

Instruction : BBC, BBS



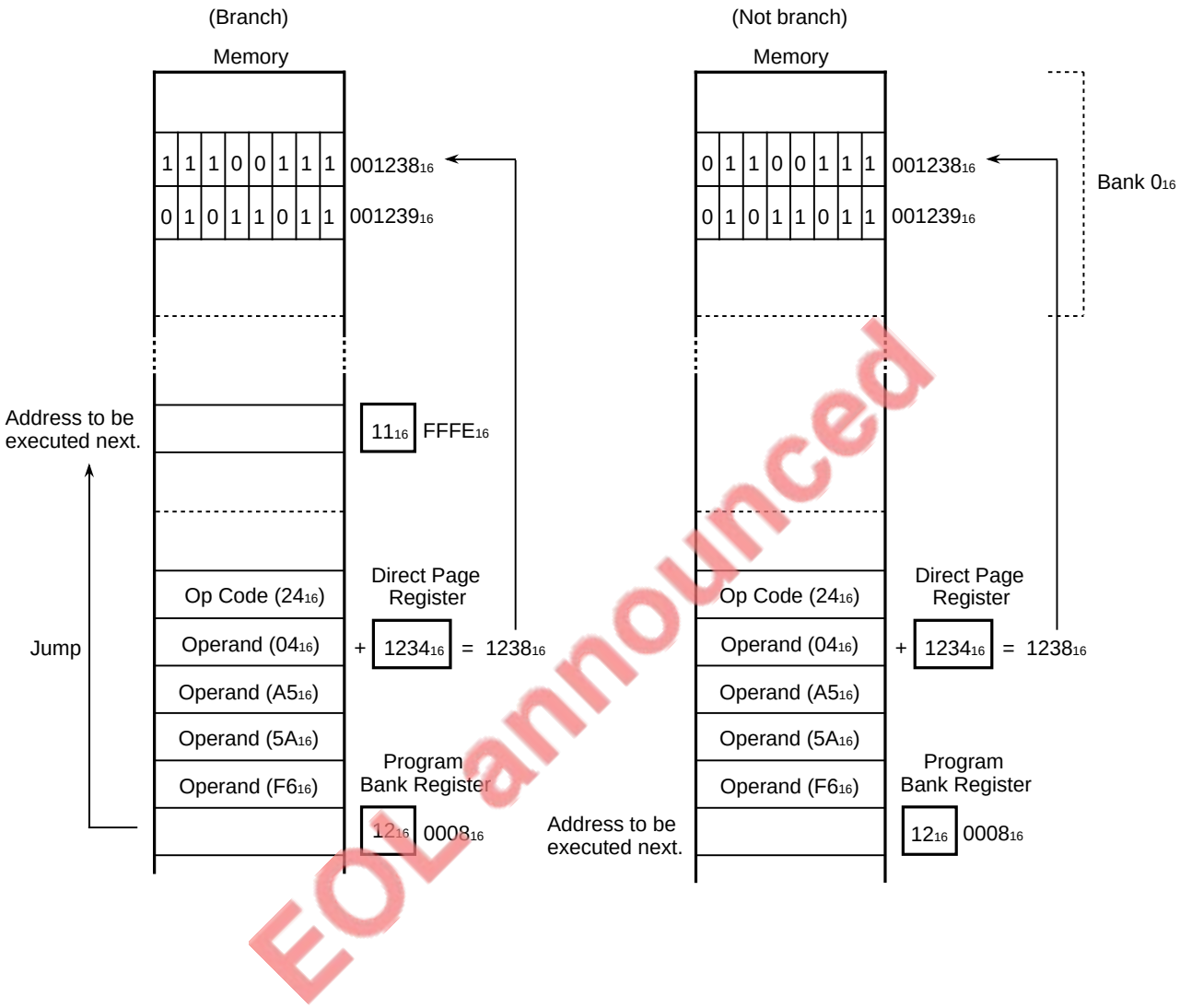
EOL announced

ex. : Mnemonic

BBS #5AA5H, 04H, 0F6H (m=0)

Machine Code

24_{16} 04_{16} $A5_{16}$ $5A_{16}$ $F6_{16}$



EOL announced

Absolute Bit Relative

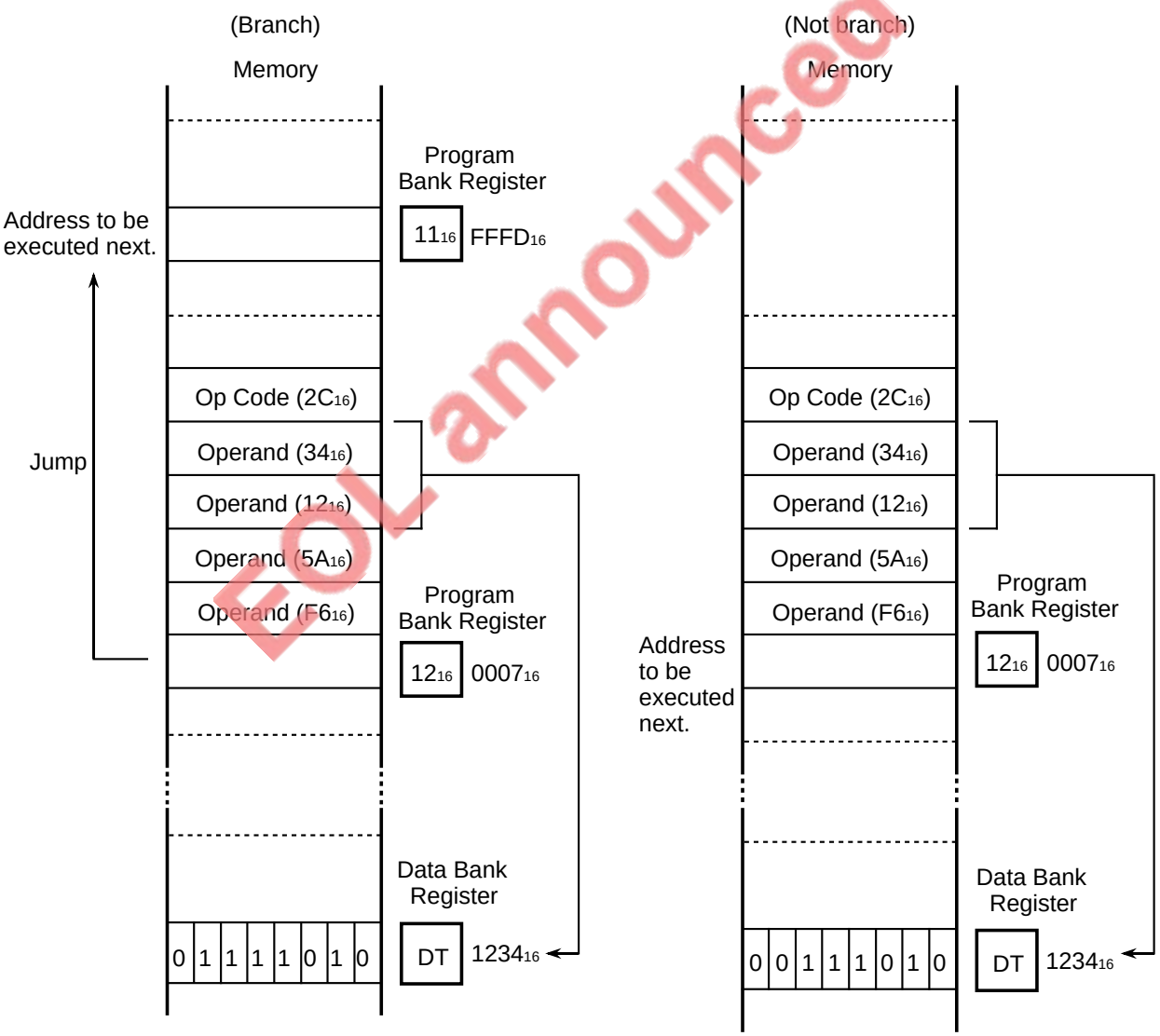
Mode : Absolute bit relative addressing mode

Function : The instruction's second and third bytes and the contents of the data bank register specify the memory location, and data for the memory location's multiple bits is specified by a bit pattern in the instruction's fourth and fifth bytes (the fourth byte only if the m flag is set to 1). Then, if the specified bits all satisfy the branching conditions, the instruction's sixth byte (or the fifth byte if the m flag is set to 1) is added to the program counter as a signed value, generating the branching destination address.

Instruction : BBC, BBS

ex. : Mnemonic
BBS #5AH, 1234H, 0F6H
(m=1)

Machine Code
$2C_{16}$ 34_{16} 12_{16} $5A_{16}$ $F6_{16}$



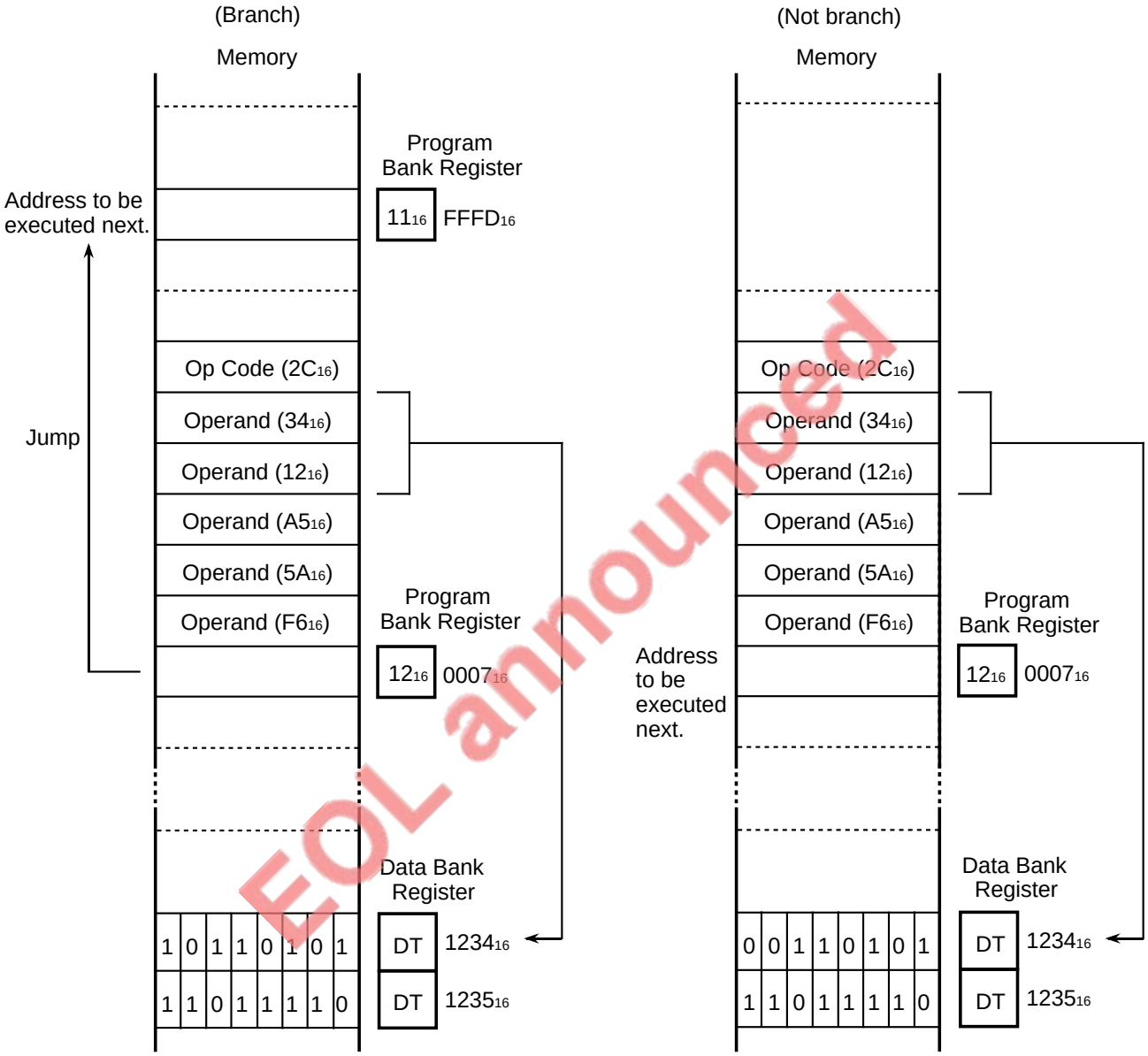
EOL announced

ex. : Mnemonic

BBS #5AA5H, 1234H, 0F6H (m=0)

Machine Code

$2C_{16}$ 34_{16} 12_{16} $A5_{16}$ $5A_{16}$ $F6_{16}$



EOL announced

Stack Pointer Relative

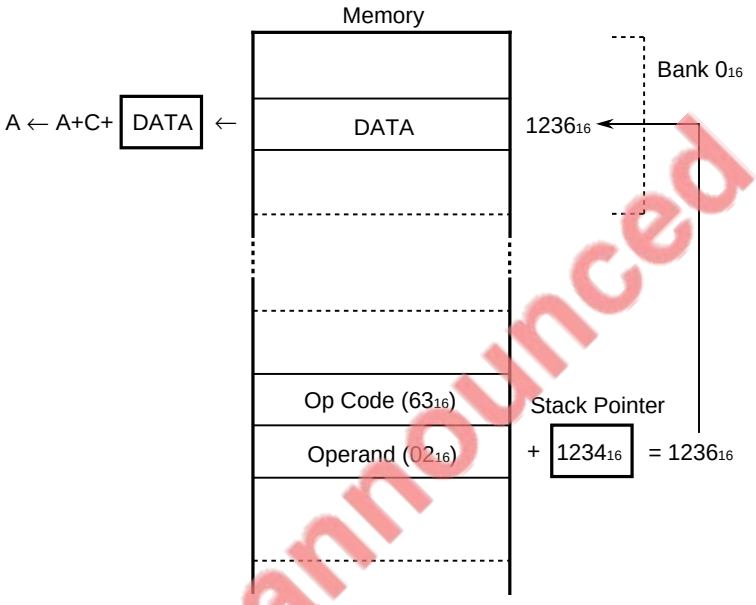
Mode : Stack pointer relative addressing mode

Function : The contents of a bank-0 memory location specified by the value resulting from addition of the instruction's second byte and the contents of the stack pointer become the actual data. If, however, the value obtained by adding the contents of the instruction's second byte and the stack pointer's contents exceeds the bank-0 range, the specified location will be in bank-1.

Instruction : ADC, AND, CMP, DIV, DIVS*, EOR, LDA, MPY, MPYS*, ORA, SBC, STA

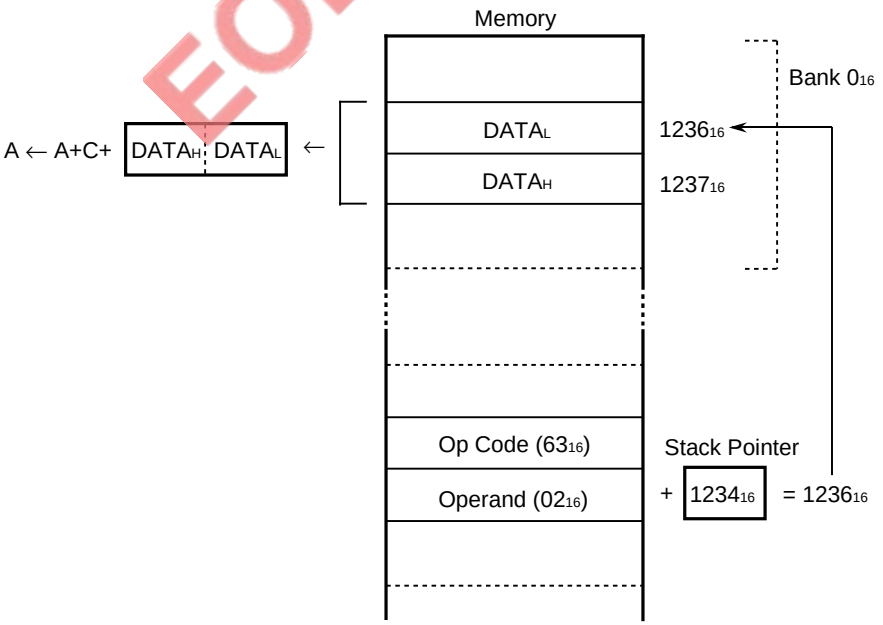
ex. : Mnemonic
ADC A, 02H, S
(m=1)

Machine Code
63_{16} 02_{16}



ex. : Mnemonic
ADC A, 02H, S
(m=0)

Machine Code
63_{16} 02_{16}



EOL announced

Stack Pointer Relative Indirect Indexed Y

Mode : Stack pointer relative indirect indexed Y addressing mode

Function : The value obtained by adding the instruction's second byte and the contents of the stack pointer specifies 2 adjacent bytes in memory. The value obtained by adding the contents of these bytes and the contents of the index register Y specifies address of the actual data in memory bank-DT (DT is contents of data bank register). If addition of the 2 bytes in memory with the contents of the index register Y generate a carry, the bank number will be 1 larger than the contents of the data bank register.

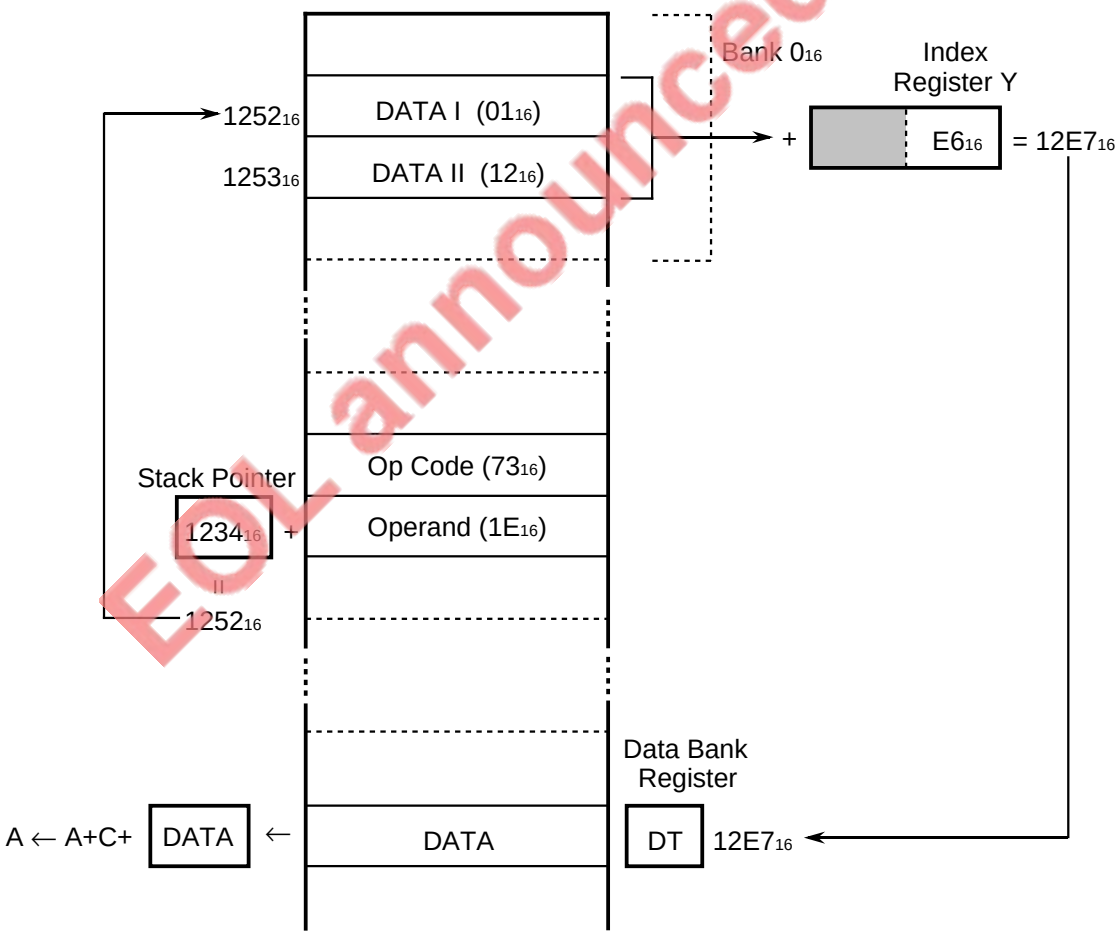
Instruction : ADC, AND, CMP, DIV, DIVS*, EOR, LDA, MPY, MPYS*, ORA, SBC, STA

ex. : Mnemonic

ADC A, (1EH, S), Y
(m=1, x=1)

Machine Code

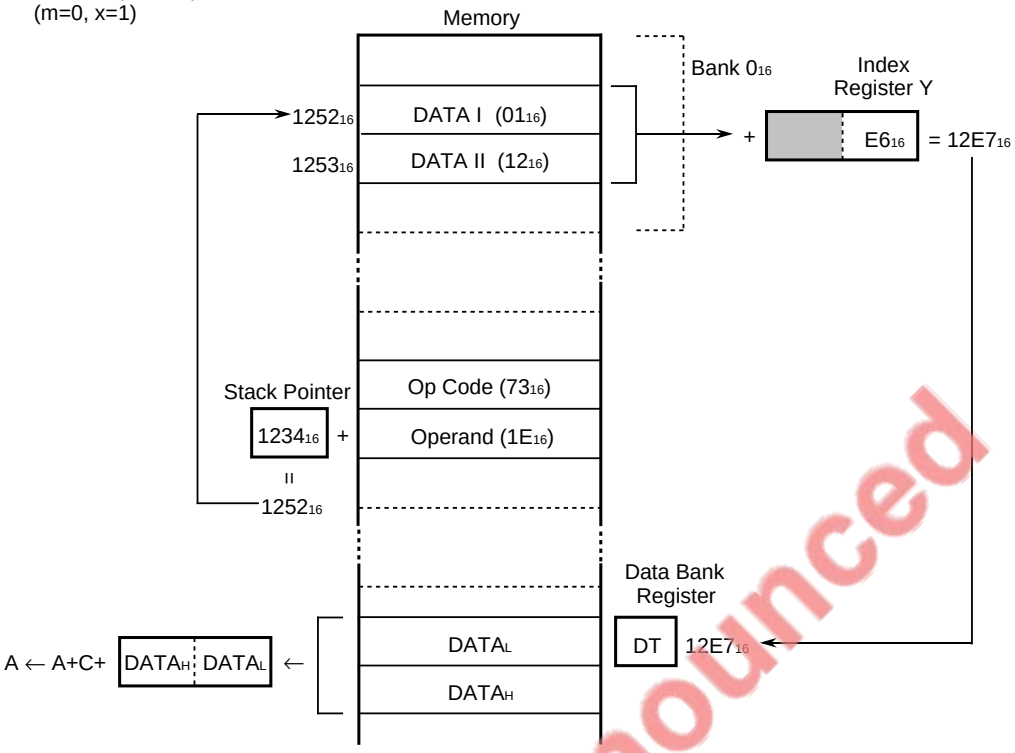
73_{16} $1E_{16}$



EOL announced

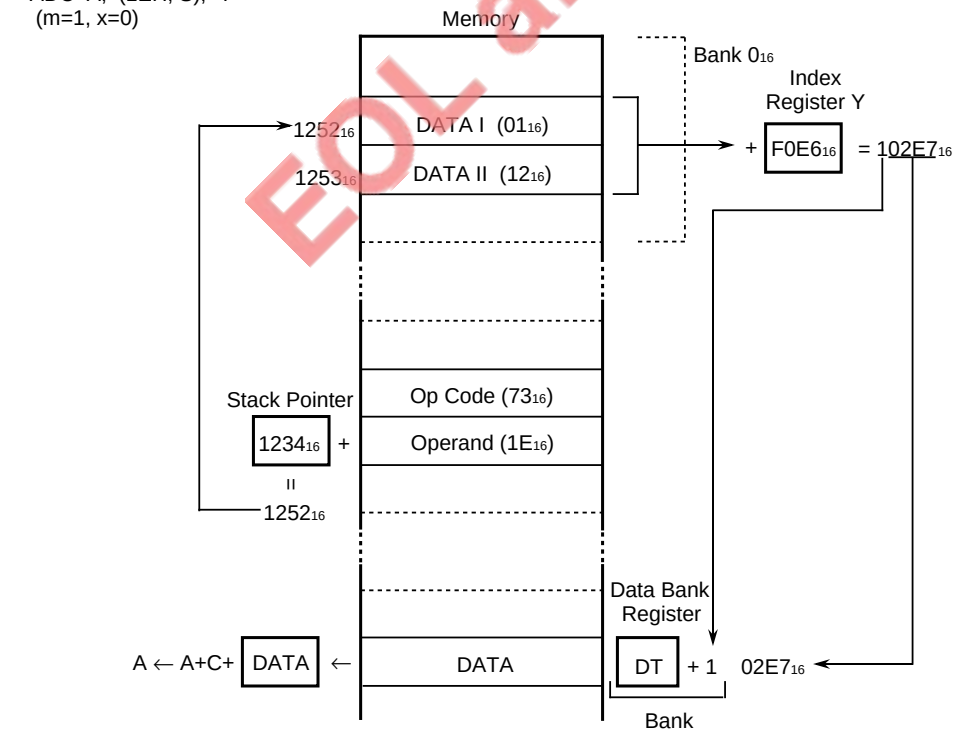
ex. : Mnemonic
ADC A, (1EH, S), Y
(m=0, x=1)

Machine Code
73_{16} $1E_{16}$



ex. : Mnemonic
ADC A, (1EH, S), Y
(m=1, x=0)

Machine Code
73_{16} $1E_{16}$



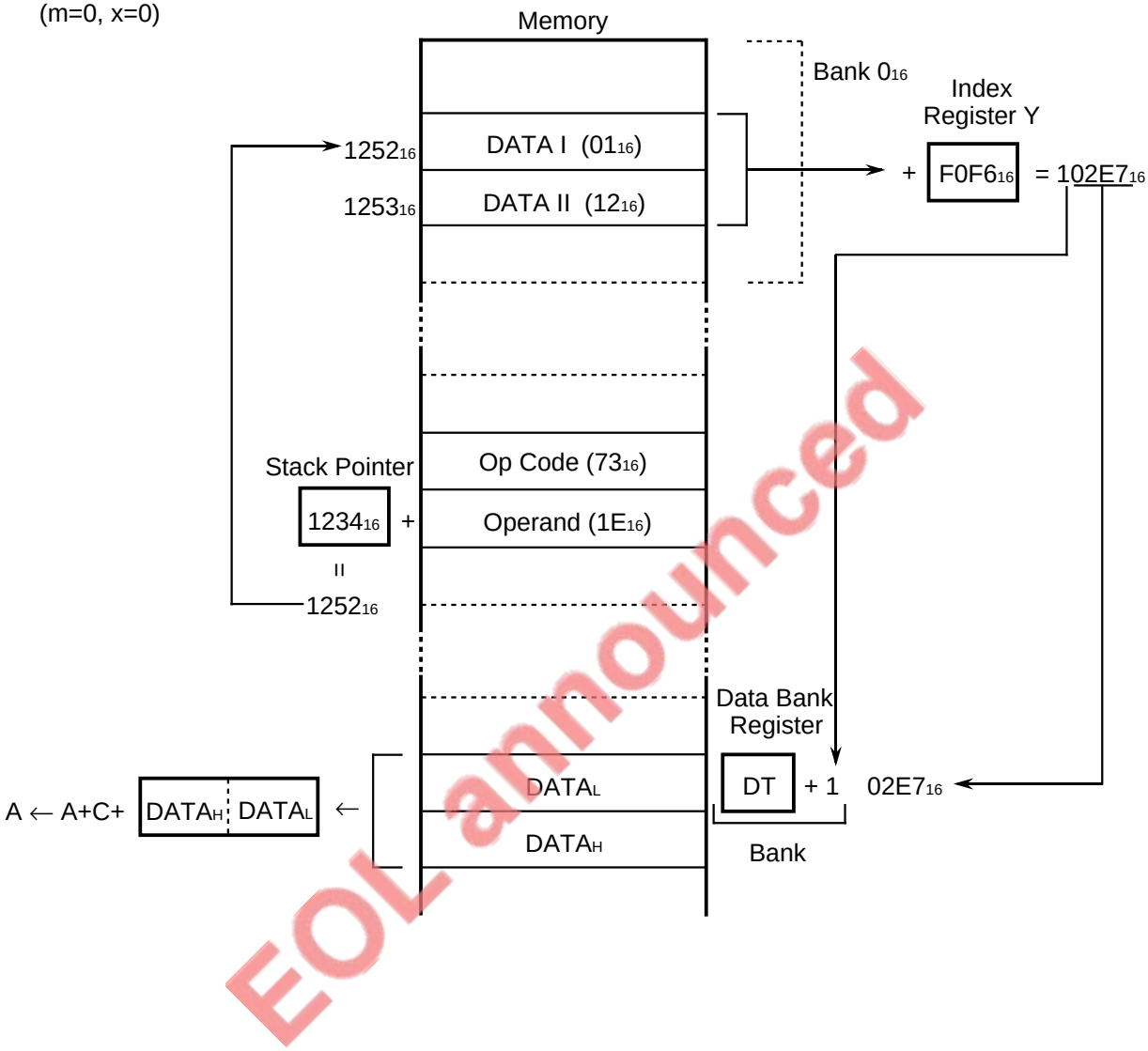
EOL announced

ex. : Mnemonic

ADC A, (1EH, S), Y
(m=0, x=0)

Machine Code

73_{16} $1E_{16}$



EOL announced

EOL announced

Mode : Block transfer addressing mode

Function : The instruction's second byte specifies the transfer-to data bank, and the contents of the index register Y specify the transfer-to address within the data bank. The instruction's third byte specifies the transfer-from data bank, and the contents of the index register X specify the address in the data bank where the data to be transferred is stored. The contents of the accumulator A constitute the number of bytes to be transferred. Upon termination of transfer, the contents of the data bank register will specify the transfer-to data bank. The MVN instruction is used for transfer to lower address location. In this case, the contents of the index registers X and Y are incremented each time data is transferred. The MVP instruction is used for transfer to higher address location. In this case, the contents of the index registers X and Y are decremented each time data is transferred. The block of data to be transferred may cross over the bank boundary.

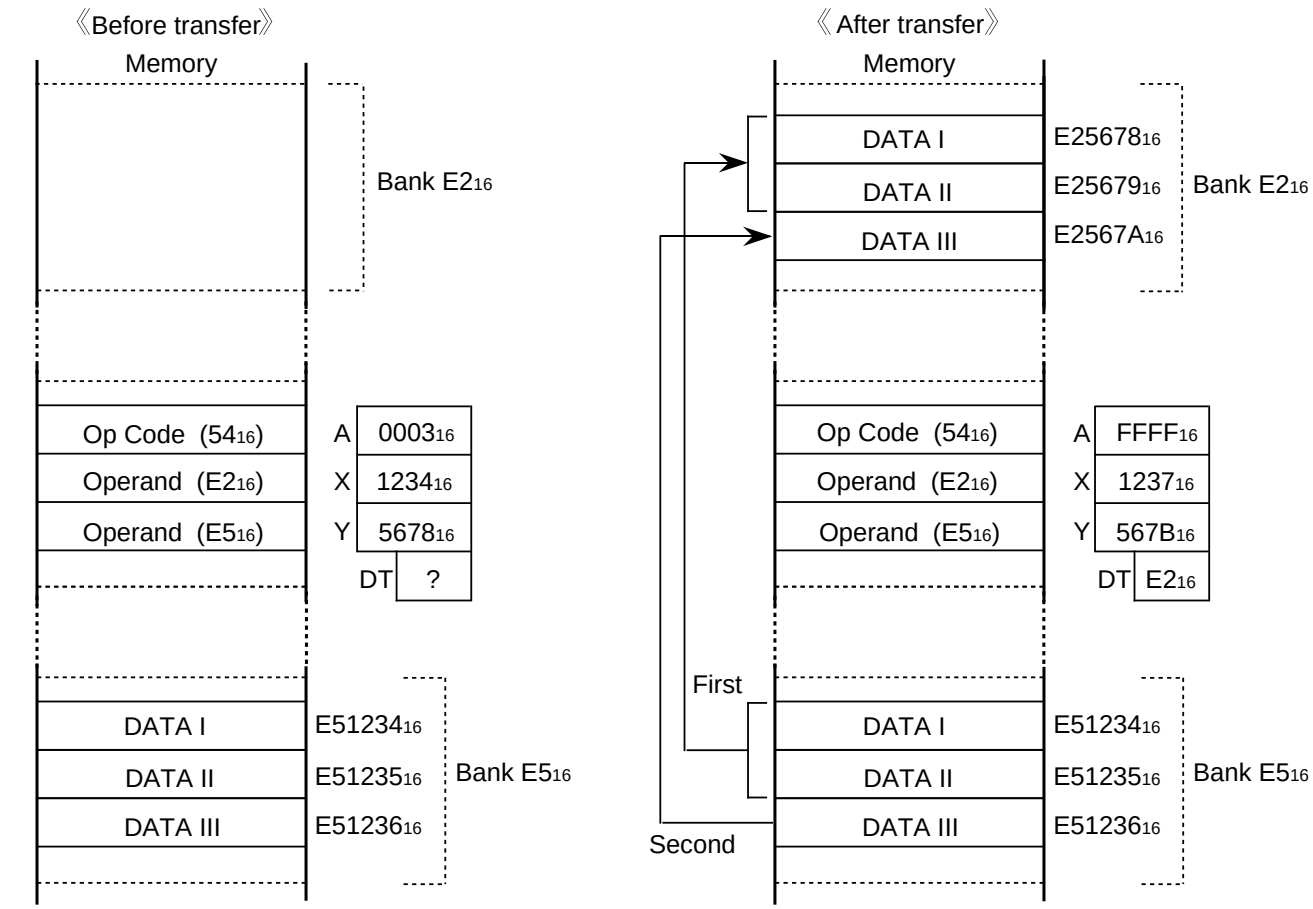
Instruction : MVN, MVP

ex. : Mnemonic

MVN 0E2H, 0E5H
(m=0, x=0)

Machine Code

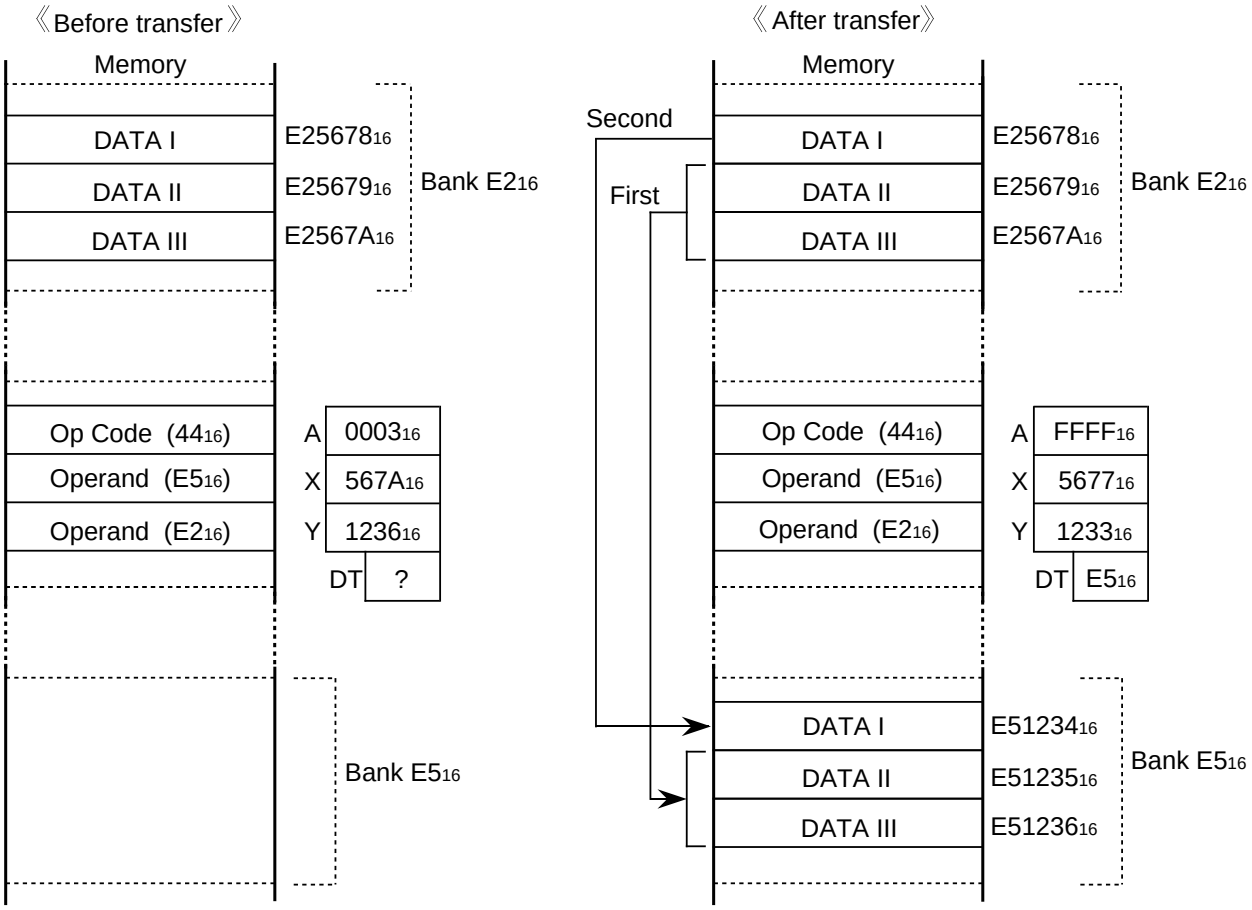
54_{16} $E2_{16}$ $E5_{16}$



EOL announced

ex. : Mnemonic — Machine Code

MVP 0E5H, 0E2H — 44_{16} $E5_{16}$ $E2_{16}$
(m=0, x=0)



(Note) For block transfer instruction, the transfer byte count and transfer source/destination address range change with the status of the m and x flags, but the transfer unit is unaffected.
The transfer unit is word (16 bits), but only 1 byte is transferred when transferring the last byte of an odd byte transfer.

EOL announced

CHAPTER 4

INSTRUCTIONS

4.1 Instruction set
4.2 Description of instructions
4.3 Notes for programming

4.1 Instruction Set

The 7700 Series, 7770 Series, and 7790 Series CPU uses the instruction set with 103 instructions. The 7750 Series CPU uses an extended instruction set (108 instructions) adding five additional instructions to their instruction set.

«Additional instructions»

Instructions	Mnemonic	Addressing mode
Multiply with sign	MPYS	Supports addressing mode equivalent to MPY instruction
Divide with sign	DIVS	Supports addressing mode equivalent to DIV instruction
Arithmetic shift right	ASR	Supports addressing mode equivalent to ASL instruction
Extension with sign	EXTS	Supported the accumulator addressing mode
Extension zero	EXTZ	

4.1.1 Data transfer instructions

The data transfer instructions move data between data and registers, between a register and the memory, between registers or between memory devices.

Category	Instruction	Description
Load	LDA	Loads the contents of memory into the accumulator.
	LDM	Loads an immediate value into the memory.
	LDT	Loads an immediate value into the data bank register.
	LDX	Loads the contents of memory into the index register X.
	LDY	Loads the contents of memory into the index register Y.
Store	STA	Stores the contents of the accumulator in the memory.
	STX	Stores the contents of the index register X in the memory.
	STY	Stores the contents of the index register Y in the memory.
Transfer	TAX	Transfers the contents of the accumulator A to the index register X.
	TXA	Transfers the contents of the index register X to the accumulator A.
	TAY	Transfers the contents of the accumulator A to the index register Y.
	TYA	Transfers the contents of the index register Y to the accumulator A.
	TSX	Transfers the contents of the stack pointer to the index register X.
	TXS	Transfers the contents of the index register X to the stack pointer.
	TAD	Transfers the contents of the accumulator A to the direct page register.
	TDA	Transfers the contents of the direct page register to the accumulator A.
	TAS	Transfers the contents of the accumulator A to the stack pointer.
	TSA	Transfers the contents of the stack pointer to the accumulator A.

EOL announced

Category	Instruction	Description
Transfer	TBD	Transfers the contents of the accumulator B to the direct page register.
	TDB	Transfers the contents of the direct page register to the accumulator B.
	TBS	Transfers the contents of the accumulator B to the stack pointer.
	TSB	Transfers the contents of the stack pointer to the accumulator B.
	TBX	Transfers the contents of the accumulator B to the index register X.
	TXB	Transfers the contents of the index register X to the accumulator B.
	TBY	Transfers the contents of the accumulator B to the index register Y.
	TYB	Transfers the contents of the index register Y to the accumulator B.
	TXY	Transfers the contents of the index register X to the index register Y.
	TYX	Transfers the contents of the index register Y to the index register X.
	MVN	Transfers a block of data from the lower addresses.
	MVP	Transfers a block of data from the higher addresses.
	PSH	Saves the contents of the specified register to the stack.
Stack operation	PUL	Restores the contents of stack to the specified register.
	PHA	Saves the contents of the accumulator A to the stack.
	PLA	Restores the contents of stack to the accumulator A.
	PHP	Saves the contents of the processor status register to the stack.
	PLP	Restores the contents of stack to the processor status register.
	PHB	Saves the contents of the accumulator B to the stack.
	PLB	Restores the contents of stack to the accumulator B.
	PHD	Saves the contents of the direct page register to the stack.
	PLD	Restores the contents of stack to the direct page register.
	PHT	Saves the contents of the data bank register to stack.
	PLT	Restores the contents of stack to the data bank register.
	PHX	Saves the contents of the index register X to the stack.
	PLX	Restores the contents of stack to the index register X.
	PHY	Saves the contents of the index register Y to the stack.
	PLY	Restores the contents of stack to the index register Y.
	PHG	Saves the contents of the program bank register to the stack.
	PEA	Saves a the numeric of 2 bytes to the stack.
	PEI	Saves the contents of 2 consecutive bytes in the direct page area to the stack.
	PER	Saves the result of adding a 16-bit numeric value to the program counter contents to the stack.
Exchange	XAB	Swaps the contents of the accumulator A with the contents of the accumulator B.

EOL announced

4.1.2 Arithmetic instructions

The arithmetic instructions perform addition, subtraction, multiplication, division, logical operation, comparison, rotation, shifting and sign/zero extension of register and memory contents.
The following table summarizes the arithmetic instructions supported:

Note. The instructions with the mark " * " can be used in the 7750 Series only.

Category	Instruction	Description
Addition, Subtraction, Multiplication, Division	ADC	Adds the contents of the accumulator, the contents of memory and the contents of the carry flag.
	SBC	Subtracts the contents of memory and the complement of the carry flag from the contents of the accumulator.
	INC	Increments the accumulator or memory contents by 1.
	DEC	Decrements the accumulator or memory contents by 1.
	INX	Increments the contents of the index register X by 1.
	DEX	Decrements the contents of the index register X by 1.
	INY	Increments the contents of the index register Y by 1.
	DEY	Decrements the contents of the index register Y by 1.
	MPY	Multiples the contents of the accumulator A and the contents of memory.
	MPYS*	Multiply the contents of the accumulator A and the contents of memory with sign.
	DIV	Divides the numeric value whose lower byte is the contents of the accumulator A and upper byte is the contents of the accumulator B by the contents of memory.
	DIVS*	Divides the numeric value whose lower byte is the contents of the accumulator A and upper byte is the contents of the accumulator B by the contents of memory with sign.
Logical operation	AND	Performs logical AND between the contents of the accumulator and the contents of memory.
	ORA	Performs logical OR between the contents of the accumulator and the contents of memory.
	EOR	Performs logical exclusive-OR between the contents of the accumulator and the contents of memory.
Comparison	CMP	Compares the contents of the accumulator with the contents of memory.
	CPX	Compares the contents of the index register X and the contents of memory.
	CPY	Compares the contents of the index register Y and the contents of memory.
Shifting, Rotation	ASL	Shifts the contents of the accumulator or memory to the left by 1 bit.
	ASR*	Shifts the contents of the accumulator or memory holding sign to the right by 1 bit.
	LSR	Shifts the contents of the accumulator or memory to the right by 1 bit.
	ROL	Links the contents of accumulator or memory with the carry flag, and rotates the result to the left by 1 bit.
	ROR	Links the contents of accumulator or memory with the carry flag, and rotates the result to the right by 1 bit.
	RLA	Rotates the contents of the accumulator A to the left by the specified number of bits.
Extension with sign / zero	EXTS*	Extend the low-order 8 bits of accumulator to 16 bits by sign extending.
	EXTZ*	Extend the low-order 8 bits of accumulator to 16 bits by zero extending.

EOL announced

4.1.3 Bit manipulation instructions

The bit manipulation instructions set the specified bits of the processor status register or memory to “1” or “0”.

The following table summarizes the bit manipulation instructions supported:

Category	Instruction	Description
Bit manipulation	CLB	Clears the specified memory bit to “0”.
	SEB	Sets the specified memory bit to “1”.
	CLP	Clears the specified bit of the processor status register’s lower byte (PSL) to “0”.
	SEP	Sets the specified bit of the processor status register’s lower byte (PSL) to “1”.

4.1.4 Flag manipulation instructions

The flag manipulation instructions set to “1” or clear to “0” the C, I, m and V flags.

The following table summarizes the flag manipulation instructions supported:

Category	Instruction	Description
Flag manipulation	CLC	Clears the contents of carry flag to “0”.
	SEC	Sets the contents of carry flag to “1”.
	CLM	Clears the contents of data length selection flag to “0”.
	SEM	Sets the contents of data length selection flag to “1”.
	CLI	Clears the contents of interrupt disable flag to “0”.
	SEI	Sets the contents of interrupt disable flag to “1”.
	CLV	Clears the contents of overflow flag to “0”.

EOL announced

4.1.5 Branching and return instructions

The branching and return instructions enable changing the program execution sequence.

The following table summarizes the branching and return instructions:

Category	Instruction	Description
Jump	JMP	Sets a new address in the program counter and jumps to the new address.
	BRA	Jumps to the address obtained by adding an offset value to the contents of the program counter.
	JSR	Saves the contents of the program counter to the stack and then jumps to the new address.

4.1 Instruction set

Category	Instruction	Description
Branch	BBC	Causes a branch if the specified memory bits are all “0”.
	BBS	Causes a branch if the specified memory bits are all “1”.
	BCC	Causes a branch if the carry flag is set to “0”.
	BCS	Causes a branch if the carry flag is set to “1”.
	BNE	Causes a branch if the zero flag is set to “0”.
	BEQ	Causes a branch if the zero flag is set to “1”.
	BPL	Causes a branch if the negative flag is set to “0”.
	BMI	Causes a branch if the negative flag is set to “1”.
	BVC	Causes a branch if the overflow flag is set to “0”.
	BVS	Causes a branch if the overflow flag is set to “1”.
Return	RTI	Returns from the interrupt routine to the original routine.
	RTS	Returns from a subroutine to the original routine. The program bank register contents are not restored.
	RTL	Returns from a subroutine to the original routine. The program bank register contents are restored.

4.1.6 Interrupt instruction (break instruction)

The interrupt instruction executes software interrupt.

Category	Instruction	Description
Break	BRK	Executes a software interrupt.

4.1.7 Special instructions

The special instructions listed below control the clock generator circuit.

Category	Instruction	Description
Special	WIT	Stops the internal clock.
	STP	Stops the oscillator.

4.1.8 Other instruction

Category	Instruction	Description
Other	NOP	Only advances the program counter.

EOL announced

4.2 Description of Instructions

This section describes the 7700 Family instructions at each instruction (Note 1). To the extent possible, each instruction is described using one page per instruction. Each instruction description page is headed by the instruction mnemonic, and the pages are arranged in alphabetical order of the mnemonics. For each instruction, operation and description of the instruction (Note 2, 3), status flag changes and a listing sorted by addressing modes of the assembler coding format (Note 4), machine code, bytes-count and cycles-count (Note 5) are presented.

Note 1. The instructions with the mark " * " can be used in the 7750 Series only.

Note 2. In the description of instruction operation, the change in the PC (program counter) is described only for instructions affecting the processing flow.

When an instruction is executed, the length of the instruction is added to content of the PC to form the address of the next instruction to be executed. If a carry occurs during this addition, PG (program bank register) is incremented by 1.

Note 3. In the description of each instruction, [Operation] indicates the contents of each register and memory after executing the instruction. The detailed operation sequence is omitted.

Note 4. The assembler coding formats shown are general examples, and they may differ from the actual formats for the assembler used. Please be sure to refer to the mnemonic coding description in the manual for the assembler actually used for programming.

Note 5. The cycles-counts shown are the minimum possible, and they vary depending on the following conditions:

- Value of direct page register's lower byte

  The cycles-count shown are for when the direct page register's lower byte (DPR_L) is 00_{16}. When using an addressing mode that uses the direct page register with $DPR_L \neq$"00_{16}", the cycles-count will be 1 more than the value shown.

- Number of bytes that have been loaded in the instruction queue buffer
- Whether the first address of the memory read/write is even- or odd-numbered in accessing the 16-bit data length.
- Accessing of an external memory are with BYTE=1 (using 8-bit external bus)
- Whether a wait is inserted in the bus cycle.

EOL announced

4.2 Description of Instructions

The table below lists the symbols that are used in this section:

Symbol	Description
C	Carry flag
Z	Zero flag
I	Interrupt disable flag
D	Decimal operation mode flag
x	Index register length selection flag
m	Data length selection flag
V	Overflow flag
N	Negative flag
IPL	Processor interrupt priority level
+	Addition
–	Subtraction
×	Multiplication
/	Division
Λ	Logical AND
V	Logical OR
∀	Exclusive OR
$\overline{\quad}$	Negation
←	Movement to the arrow direction
→	Movement to the arrow direction
⇄	Movement to the arrow direction
Acc	Accumulator
Acc_H	Accumulator's upper 8 bits
Acc_L	Accumulator's lower 8 bits
A	Accumulator A
A_H	Accumulator A's upper 8 bits
A_L	Accumulator A's lower 8 bits
B	Accumulator B
B_H	Accumulator B's upper 8 bits
B_L	Accumulator B's lower 8 bits
X	Index register X
X_H	Index register X's upper 8 bits
X_L	Index register X's lower 8 bits
Y	Index register Y
Y_H	Index register Y's upper 8 bits
Y_L	Index register Y's lower 8 bits
S	Stack pointer
PC	Program counter
PC_H	Program counter's upper 8 bits
PC_L	Program counter's lower 8 bits
REL	Relative address
PG	Program bank register
DT	Data bank register

EOL announced

Symbol	Description
DPR	Direct page register
DPRH	Direct page register's upper 8 bits
DPRL	Direct page register's lower 8 bits
PS	Processor status register
PSH	Processor status register's upper 8 bits
PSL	Processor status register's lower 8 bits
PSn	Processor status register's n-th bit
M	Memory contents
M(n)	Contents of memory location specified by operand (1 byte data)
M(n+1,n)	Contents of memory location specified by operand (1 word data)
M(m to n)	Contents of memory location specified by operand (plural bytes data)
M(S)	Contents of memory at address indicated by stack pointer
Mb	b-th memory location
ADDR	Value of 24-bit address' lower 16-bit (A_{15} to A_0)
BANK	Value of 24-bit address' upper 8-bit (A_{23} to A_{16})
ADG	Value of 24-bit address' upper 8-bit (A_{23} to A_{16})
ADH	Value of 24-bit address' middle 8-bit (A_{15} to A_8)
ADL	Value of 24-bit address' lower 8-bit (A_7 to A_0)
IMM	Immediate value
IMM16	16-bit immediate value
IMM8	8-bit immediate value
bn	n-th bit of data
dd	8-bit offset value
i	Number of transfer bytes or rotation
i1,i2	Number of registers pushed or pulled
imm	8-bit immediate value
immHimmL	16-bit immediate value (immH specifies the upper 8-bit, and immL specifies the lower 8-bit)
ll	8-bit address value
mmll	16-bit address value (mm specifies the upper 8-bit and ll specifies the lower 8-bit)
hhmmll	24-bit address value (hh specifies the upper 8-bit, mm specifies the middle 8-bit and ll specifies the lower 8-bit)
nn	8-bit data value
n1,n2	8-bit data value (Used when coding two 8-bit data side by side)
rr	Signed 8-bit data value
rr1rr2	Signed 16-bit data value (rr1 is the upper 8-bit value, and rr2 is the lower 8-bit value)

EOL announced

4.1 Instruction Set

The 7700 Series, 7770 Series, and 7790 Series CPU uses the instruction set with 103 instructions. The 7750 Series CPU uses an extended instruction set (108 instructions) adding five additional instructions to their instruction set.

«Additional instructions»

Instructions	Mnemonic	Addressing mode
Multiply with sign	MPYS	Supports addressing mode equivalent to MPY instruction
Divide with sign	DIVS	Supports addressing mode equivalent to DIV instruction
Arithmetic shift right	ASR	Supports addressing mode equivalent to ASL instruction
Extension with sign	EXTS	Supported the accumulator addressing mode
Extension zero	EXTZ	

4.1.1 Data transfer instructions

The data transfer instructions move data between data and registers, between a register and the memory, between registers or between memory devices.

Category	Instruction	Description
Load	LDA	Loads the contents of memory into the accumulator.
	LDM	Loads an immediate value into the memory.
	LDT	Loads an immediate value into the data bank register.
	LDX	Loads the contents of memory into the index register X.
	LDY	Loads the contents of memory into the index register Y.
Store	STA	Stores the contents of the accumulator in the memory.
	STX	Stores the contents of the index register X in the memory.
	STY	Stores the contents of the index register Y in the memory.
Transfer	TAX	Transfers the contents of the accumulator A to the index register X.
	TXA	Transfers the contents of the index register X to the accumulator A.
	TAY	Transfers the contents of the accumulator A to the index register Y.
	TYA	Transfers the contents of the index register Y to the accumulator A.
	TSX	Transfers the contents of the stack pointer to the index register X.
	TXS	Transfers the contents of the index register X to the stack pointer.
	TAD	Transfers the contents of the accumulator A to the direct page register.
	TDA	Transfers the contents of the direct page register to the accumulator A.
	TAS	Transfers the contents of the accumulator A to the stack pointer.
	TSA	Transfers the contents of the stack pointer to the accumulator A.

EOL announced

Category	Instruction	Description
Transfer	TBD	Transfers the contents of the accumulator B to the direct page register.
	TDB	Transfers the contents of the direct page register to the accumulator B.
	TBS	Transfers the contents of the accumulator B to the stack pointer.
	TSB	Transfers the contents of the stack pointer to the accumulator B.
	TBX	Transfers the contents of the accumulator B to the index register X.
	TXB	Transfers the contents of the index register X to the accumulator B.
	TBY	Transfers the contents of the accumulator B to the index register Y.
	TYB	Transfers the contents of the index register Y to the accumulator B.
	TXY	Transfers the contents of the index register X to the index register Y.
	TYX	Transfers the contents of the index register Y to the index register X.
	MVN	Transfers a block of data from the lower addresses.
	MVP	Transfers a block of data from the higher addresses.
	PSH	Saves the contents of the specified register to the stack.
Stack operation	PUL	Restores the contents of stack to the specified register.
	PHA	Saves the contents of the accumulator A to the stack.
	PLA	Restores the contents of stack to the accumulator A.
	PHP	Saves the contents of the processor status register to the stack.
	PLP	Restores the contents of stack to the processor status register.
	PHB	Saves the contents of the accumulator B to the stack.
	PLB	Restores the contents of stack to the accumulator B.
	PHD	Saves the contents of the direct page register to the stack.
	PLD	Restores the contents of stack to the direct page register.
	PHT	Saves the contents of the data bank register to stack.
	PLT	Restores the contents of stack to the data bank register.
	PHX	Saves the contents of the index register X to the stack.
	PLX	Restores the contents of stack to the index register X.
	PHY	Saves the contents of the index register Y to the stack.
	PLY	Restores the contents of stack to the index register Y.
	PHG	Saves the contents of the program bank register to the stack.
	PEA	Saves a the numeric of 2 bytes to the stack.
	PEI	Saves the contents of 2 consecutive bytes in the direct page area to the stack.
	PER	Saves the result of adding a 16-bit numeric value to the program counter contents to the stack.
Exchange	XAB	Swaps the contents of the accumulator A with the contents of the accumulator B.

EOL announced

4.1.2 Arithmetic instructions

The arithmetic instructions perform addition, subtraction, multiplication, division, logical operation, comparison, rotation, shifting and sign/zero extension of register and memory contents.
The following table summarizes the arithmetic instructions supported:

Note. The instructions with the mark " * " can be used in the 7750 Series only.

Category	Instruction	Description
Addition, Subtraction, Multiplication, Division	ADC	Adds the contents of the accumulator, the contents of memory and the contents of the carry flag.
	SBC	Subtracts the contents of memory and the complement of the carry flag from the contents of the accumulator.
	INC	Increments the accumulator or memory contents by 1.
	DEC	Decrements the accumulator or memory contents by 1.
	INX	Increments the contents of the index register X by 1.
	DEX	Decrements the contents of the index register X by 1.
	INY	Increments the contents of the index register Y by 1.
	DEY	Decrements the contents of the index register Y by 1.
	MPY	Multiples the contents of the accumulator A and the contents of memory.
	MPYS*	Multiply the contents of the accumulator A and the contents of memory with sign.
	DIV	Divides the numeric value whose lower byte is the contents of the accumulator A and upper byte is the contents of the accumulator B by the contents of memory.
	DIVS*	Divides the numeric value whose lower byte is the contents of the accumulator A and upper byte is the contents of the accumulator B by the contents of memory with sign.
Logical operation	AND	Performs logical AND between the contents of the accumulator and the contents of memory.
	ORA	Performs logical OR between the contents of the accumulator and the contents of memory.
	EOR	Performs logical exclusive-OR between the contents of the accumulator and the contents of memory.
Comparison	CMP	Compares the contents of the accumulator with the contents of memory.
	CPX	Compares the contents of the index register X and the contents of memory.
	CPY	Compares the contents of the index register Y and the contents of memory.
Shifting, Rotation	ASL	Shifts the contents of the accumulator or memory to the left by 1 bit.
	ASR*	Shifts the contents of the accumulator or memory holding sign to the right by 1 bit.
	LSR	Shifts the contents of the accumulator or memory to the right by 1 bit.
	ROL	Links the contents of accumulator or memory with the carry flag, and rotates the result to the left by 1 bit.
	ROR	Links the contents of accumulator or memory with the carry flag, and rotates the result to the right by 1 bit.
	RLA	Rotates the contents of the accumulator A to the left by the specified number of bits.
Extension with sign / zero	EXTS*	Extend the low-order 8 bits of accumulator to 16 bits by sign extending.
	EXTZ*	Extend the low-order 8 bits of accumulator to 16 bits by zero extending.

EOL announced

4.1.3 Bit manipulation instructions

The bit manipulation instructions set the specified bits of the processor status register or memory to “1” or “0”.

The following table summarizes the bit manipulation instructions supported:

Category	Instruction	Description
Bit manipulation	CLB	Clears the specified memory bit to “0”.
	SEB	Sets the specified memory bit to “1”.
	CLP	Clears the specified bit of the processor status register’s lower byte (PSL) to “0”.
	SEP	Sets the specified bit of the processor status register’s lower byte (PSL) to “1”.

4.1.4 Flag manipulation instructions

The flag manipulation instructions set to “1” or clear to “0” the C, I, m and V flags.

The following table summarizes the flag manipulation instructions supported:

Category	Instruction	Description
Flag manipulation	CLC	Clears the contents of carry flag to “0”.
	SEC	Sets the contents of carry flag to “1”.
	CLM	Clears the contents of data length selection flag to “0”.
	SEM	Sets the contents of data length selection flag to “1”.
	CLI	Clears the contents of interrupt disable flag to “0”.
	SEI	Sets the contents of interrupt disable flag to “1”.
	CLV	Clears the contents of overflow flag to “0”.

EOL announced

4.1.5 Branching and return instructions

The branching and return instructions enable changing the program execution sequence.

The following table summarizes the branching and return instructions:

Category	Instruction	Description
Jump	JMP	Sets a new address in the program counter and jumps to the new address.
	BRA	Jumps to the address obtained by adding an offset value to the contents of the program counter.
	JSR	Saves the contents of the program counter to the stack and then jumps to the new address.

4.1 Instruction set

Category	Instruction	Description
Branch	BBC	Causes a branch if the specified memory bits are all “0”.
	BBS	Causes a branch if the specified memory bits are all “1”.
	BCC	Causes a branch if the carry flag is set to “0”.
	BCS	Causes a branch if the carry flag is set to “1”.
	BNE	Causes a branch if the zero flag is set to “0”.
	BEQ	Causes a branch if the zero flag is set to “1”.
	BPL	Causes a branch if the negative flag is set to “0”.
	BMI	Causes a branch if the negative flag is set to “1”.
	BVC	Causes a branch if the overflow flag is set to “0”.
	BVS	Causes a branch if the overflow flag is set to “1”.
Return	RTI	Returns from the interrupt routine to the original routine.
	RTS	Returns from a subroutine to the original routine. The program bank register contents are not restored.
	RTL	Returns from a subroutine to the original routine. The program bank register contents are restored.

4.1.6 Interrupt instruction (break instruction)

The interrupt instruction executes software interrupt.

Category	Instruction	Description
Break	BRK	Executes a software interrupt.

4.1.7 Special instructions

The special instructions listed below control the clock generator circuit.

Category	Instruction	Description
Special	WIT	Stops the internal clock.
	STP	Stops the oscillator.

4.1.8 Other instruction

Category	Instruction	Description
Other	NOP	Only advances the program counter.

EOL announced

4.2 Description of Instructions

This section describes the 7700 Family instructions at each instruction (Note 1). To the extent possible, each instruction is described using one page per instruction. Each instruction description page is headed by the instruction mnemonic, and the pages are arranged in alphabetical order of the mnemonics. For each instruction, operation and description of the instruction (Note 2, 3), status flag changes and a listing sorted by addressing modes of the assembler coding format (Note 4), machine code, bytes-count and cycles-count (Note 5) are presented.

Note 1. The instructions with the mark " * " can be used in the 7750 Series only.

Note 2. In the description of instruction operation, the change in the PC (program counter) is described only for instructions affecting the processing flow.

When an instruction is executed, the length of the instruction is added to content of the PC to form the address of the next instruction to be executed. If a carry occurs during this addition, PG (program bank register) is incremented by 1.

Note 3. In the description of each instruction, [Operation] indicates the contents of each register and memory after executing the instruction. The detailed operation sequence is omitted.

Note 4. The assembler coding formats shown are general examples, and they may differ from the actual formats for the assembler used. Please be sure to refer to the mnemonic coding description in the manual for the assembler actually used for programming.

Note 5. The cycles-counts shown are the minimum possible, and they vary depending on the following conditions:

- Value of direct page register's lower byte

  The cycles-count shown are for when the direct page register's lower byte (DPR_L) is 00_{16}. When using an addressing mode that uses the direct page register with $DPR_L \neq$"00_{16}", the cycles-count will be 1 more than the value shown.

- Number of bytes that have been loaded in the instruction queue buffer
- Whether the first address of the memory read/write is even- or odd-numbered in accessing the 16-bit data length.
- Accessing of an external memory are with BYTE=1 (using 8-bit external bus)
- Whether a wait is inserted in the bus cycle.

EOL announced

4.2 Description of Instructions

The table below lists the symbols that are used in this section:

Symbol	Description
C	Carry flag
Z	Zero flag
I	Interrupt disable flag
D	Decimal operation mode flag
x	Index register length selection flag
m	Data length selection flag
V	Overflow flag
N	Negative flag
IPL	Processor interrupt priority level
+	Addition
–	Subtraction
×	Multiplication
/	Division
Λ	Logical AND
V	Logical OR
∀	Exclusive OR
$\overline{\quad}$	Negation
←	Movement to the arrow direction
→	Movement to the arrow direction
⇄	Movement to the arrow direction
Acc	Accumulator
Acc_H	Accumulator's upper 8 bits
Acc_L	Accumulator's lower 8 bits
A	Accumulator A
A_H	Accumulator A's upper 8 bits
A_L	Accumulator A's lower 8 bits
B	Accumulator B
B_H	Accumulator B's upper 8 bits
B_L	Accumulator B's lower 8 bits
X	Index register X
X_H	Index register X's upper 8 bits
X_L	Index register X's lower 8 bits
Y	Index register Y
Y_H	Index register Y's upper 8 bits
Y_L	Index register Y's lower 8 bits
S	Stack pointer
PC	Program counter
PC_H	Program counter's upper 8 bits
PC_L	Program counter's lower 8 bits
REL	Relative address
PG	Program bank register
DT	Data bank register

EOL announced

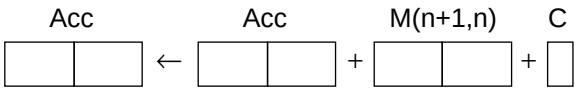
Symbol	Description
DPR	Direct page register
DPRH	Direct page register's upper 8 bits
DPRL	Direct page register's lower 8 bits
PS	Processor status register
PSH	Processor status register's upper 8 bits
PSL	Processor status register's lower 8 bits
PSn	Processor status register's n-th bit
M	Memory contents
M(n)	Contents of memory location specified by operand (1 byte data)
M(n+1,n)	Contents of memory location specified by operand (1 word data)
M(m to n)	Contents of memory location specified by operand (plural bytes data)
M(S)	Contents of memory at address indicated by stack pointer
Mb	b-th memory location
ADDR	Value of 24-bit address' lower 16-bit (A_{15} to A_0)
BANK	Value of 24-bit address' upper 8-bit (A_{23} to A_{16})
ADG	Value of 24-bit address' upper 8-bit (A_{23} to A_{16})
ADH	Value of 24-bit address' middle 8-bit (A_{15} to A_8)
ADL	Value of 24-bit address' lower 8-bit (A_7 to A_0)
IMM	Immediate value
IMM16	16-bit immediate value
IMM8	8-bit immediate value
bn	n-th bit of data
dd	8-bit offset value
i	Number of transfer bytes or rotation
i1,i2	Number of registers pushed or pulled
imm	8-bit immediate value
immHimmL	16-bit immediate value (immH specifies the upper 8-bit, and immL specifies the lower 8-bit)
ll	8-bit address value
mmll	16-bit address value (mm specifies the upper 8-bit and ll specifies the lower 8-bit)
hhmmll	24-bit address value (hh specifies the upper 8-bit, mm specifies the middle 8-bit and ll specifies the lower 8-bit)
nn	8-bit data value
n1,n2	8-bit data value (Used when coding two 8-bit data side by side)
rr	Signed 8-bit data value
rr1rr2	Signed 16-bit data value (rr1 is the upper 8-bit value, and rr2 is the lower 8-bit value)

EOL announced

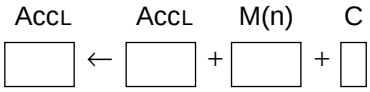
Function : Addition with carry

Operation : Acc ← Acc + M + C

When m=0



When m=1



Description : Adds the contents of the accumulator, memory and carry flag, and places the result in the accumulator.

Executed as binary addition if the decimal operation mode flag D is set to 0.

Executed as decimal addition if the decimal operation mode flag D is set to 1.

Status flags

IPL: Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0. Meaningless for decimal addition.

V : Set to 1 when binary addition of signed data result in a value outside the range of -32768 to +32767 (-128 to +127 if the data length selection flag m is set to 1). Otherwise, cleared to 0. Meaningless for decimal addition.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0. Meaningless for decimal addition.

C : When the data length selection flag m is set to 0, set to 1 if binary addition exceeds +65535 or if decimal addition exceeds +9999. Otherwise, cleared to 0. When the data length selection flag m is set to 1, set to 1 if binary addition exceeds +255 or if decimal addition exceeds +99. Otherwise, cleared to 0.

FOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	ADC A, #imm	69_{16}, imm	2	2
Direct	ADC A, dd	65_{16}, dd	2	4
Direct indexed X	ADC A, dd, X	75_{16}, dd	2	5
Direct indirect	ADC A, (dd)	72_{16}, dd	2	6
Direct indexed X indirect	ADC A, (dd, X)	61_{16}, dd	2	7
Direct indirect indexed Y	ADC A, (dd), Y	71_{16}, dd	2	8
Direct indirect long	ADCL A, (dd)	67_{16}, dd	2	10
Direct indirect long indexed Y	ADCL A, (dd), Y	77_{16}, dd	2	11
Absolute	ADC A, mmll	$6D_{16}$, ll, mm	3	4
Absolute indexed X	ADC A, mmll, X	$7D_{16}$, ll, mm	3	6
Absolute indexed Y	ADC A, mmll, Y	79_{16}, ll, mm	3	6
Absolute long	ADC A, hhmmll	$6F_{16}$, ll, mm, hh	4	6
Absolute long indexed X	ADC A, hhmmll, X	$7F_{16}$, ll, mm, hh	4	7
Stack pointer relative	ADC A, nn,S	63_{16}, nn	2	5
Stack pointer relative indirect indexed Y	ADC A, (nn, S), Y	73_{16}, nn	2	8

(Note 1) This table applies when using the accumulator A. If using the accumulator B, replace “A” with “B”. In this case, “42_{16}” is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

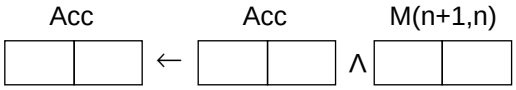
(Note 2) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

EOL announced

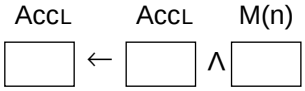
Function : Logical AND

Operation : Acc ← Acc ∧ M

When m=0



When m=1



Description : Performs logical AND between the contents of the accumulator and the contents of memory, and places the result in the accumulator.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	AND A, #imm	29_{16}, imm	2	2
Direct	AND A, dd	25_{16}, dd	2	4
Direct indexed X	AND A, dd, X	35_{16}, dd	2	5
Direct indirect	AND A, (dd)	32_{16}, dd	2	6
Direct indexed X indirect	AND A, (dd, X)	21_{16}, dd	2	7
Direct indirect indexed Y	AND A, (dd), Y	31_{16}, dd	2	8
Direct indirect long	ANDL A, (dd)	27_{16}, dd	2	10
Direct indirect long indexed Y	ANDL A, (dd), Y	37_{16}, dd	2	11
Absolute	AND A, mmll	$2D_{16}$, ll, mm	3	4
Absolute indexed X	AND A, mmll, X	$3D_{16}$, ll, mm	3	6
Absolute indexed Y	AND A, mmll, Y	39_{16}, ll, mm	3	6
Absolute long	AND A, hhmmll	$2F_{16}$, ll, mm, hh	4	6
Absolute long indexed X	AND A, hhmmll, X	$3F_{16}$, ll, mm, hh	4	7
Stack pointer relative	AND A, nn, S	23_{16}, nn	2	5
Stack pointer relative indirect indexed Y	AND A, (nn, S), Y	33_{16}, nn	2	8

(Note 1) This table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

(Note 2) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

EOL announced

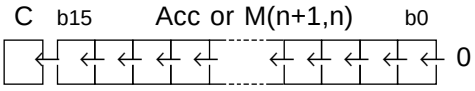
Function : Arithmetic shift left

Operation :

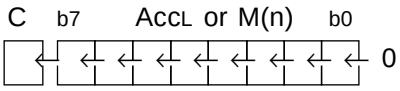
C Acc or M

1 bit shift to left ← 0

When m=0



When m=1



Description : Shifts all bits of the accumulator or memory one place to the left. Bit 0 is loaded with 0. The carry flag C is loaded from bit 15 (or bit 7 when the data length selection flag m is set to 1) of the data before the shift.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Set to 1 when bit 15 (or bit 7 when the data length selection flag m is set to 1) before the operation is 1. Otherwise, cleared to 0.

EOL announced

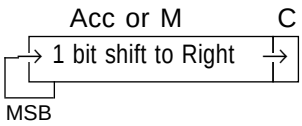
Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	ASL A	$0A_{16}$	1	2
Direct	ASL dd	06_{16}, dd	2	7
Direct indexed X	ASL dd, X	16_{16}, dd	2	7
Absolute	ASL mmll	$0E_{16}$, ll, mm	3	7
Absolute indexed X	ASL mmll, X	$1E_{16}$, ll, mm	3	8

(Note 1) The accumulator addressing mode's specification in this table applies when using the accumulator A. If using the accumulator B, replace “A” with “B”. In this case, “42_{16}” is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

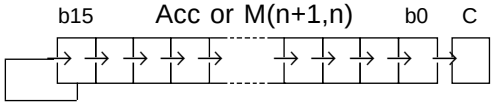
Function : Arithmetic shift right

This instruction can be used in the 7750 Series only.

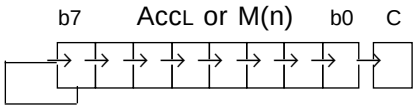
Operation :



When m=0



When m=1



Description : Shifts all bits of the accumulator or memory one place to the right. Bit 15 (or bit 7 when the data length selection flag m is set to 1) is loaded with the value before shift. The carry flag C is loaded from bit 0 of the data before the shift.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Set to 1 when bit 0 before the operation is 1. Otherwise, cleared to 0.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	ASR A	89_{16}, 08_{16}	2	5
Accumulator	ASR B	42_{16}, 08_{16}	2	5
Direct	ASR dd	89_{16}, 06_{16}, dd	3	10
Direct indexed X	ASR dd, X	89_{16},16_{16}, dd	3	10
Absolute	ASR mmll	89_{16}, $0E_{16}$, ll, mm	4	10
Absolute indexed X	ASR mmll, X	89_{16}, $1E_{16}$, ll, mm	4	11

Function : Branch on condition

Operation : Mb = 0 ? (b is the specified bits)

When M ∧ IMM = 0 (True) PC ← PC + n ± REL

When M ∧ IMM ≠ 0 (False) PC ← PC + n

❊ PG changes according to the result of the above PC operation

- if carry occurs in PC : PG ← PG + 1
- if borrow occurs in PC : PG ← PG – 1

❊ IMM is an immediate value indicating the bit to be tested with “1”.

❊ n is the number of instruction bytes in each addressing mode of the BBC instruction

❊ REL is relative value (–128 to +127) indicated by the last byte of the instruction

When m=0

M(n+1,n) IMM16

When m=1

M(n) IMM8

Description : The BBC instruction tests the specified bits (which may be specified simultaneously) of memory. The instruction causes a branch to the specified address when the specified bits are all 0. The branch address is specified by a relative address.

Status flags : Not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct bit relative	BBC #imm, dd, rr	34_{16}, dd, imm, rr	4	7
Absolute bit relative	BBC #imm, mmll, rr	$3C_{16}$, ll, mm, imm, rr	5	8

(Note 1) The bytes-count increases by 1 when operating on 16-bit data with the data length selection flag m set to 0.

(Note 2) The cycles-count increases by 2 when a branch occurs.

Function : Branch on condition

Operation : Mb = 1 ? (b is the specified bits)

When $\overline{M} \wedge IMM = 0$ (True) PC ← PC + n ± REL

When $\overline{M} \wedge IMM \neq 0$ (False) PC ← PC + n

❊ PG changes according to the result of the above PC operation

- if carry occurs in PC : PG ← PG + 1
- if borrow occurs in PC : PG ← PG – 1

❊ n is the number of instruction bytes of the BBS instruction

❊ REL is relative value (–128 to +127) indicated by the last byte of the instruction

When m=0

$\overline{M(n+1,n)}$ IMM16

When m=1

$\overline{M(n)}$ IMM8

Description : The BBS instruction tests the specified bits (which may be specified simultaneously) of memory. The instruction causes a branch to the specified address when the specified bits are all 1. The branch address is specified by a relative address.

Status flags : Not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct bit relative	BBS #imm, dd, rr	24_{16}, dd, imm, rr	4	7
Absolute bit relative	BBS #imm, mmll, rr	$2C_{16}$, ll, mm, imm, rr	5	8

(Note 1) The bytes-count increases by 1 when operating on 16-bit data with the data length selection flag m set to 0.

(Note 2) The cycles-count increases by 2 when a branch occurs.

Function : Branch on condition

Operation : C = 0 ?

When C = 0 (True) PC ← PC + 2 ± REL

When C = 1 (False) PC ← PC + 2

❊ PG changes according to the result of the above PC operation

- if carry occurs in PC : PG ← PG + 1
- if borrow occurs in PC : PG ← PG – 1

❊ 2 is the number of instruction bytes of the BCC instruction

❊ REL is relative value (–128 to +127) indicated by the 2nd byte of the instruction

Description : When the carry flag C is clear (0), the BCC instruction causes a branch to the specified address. The branch address is specified by a relative address.

When the carry flag C is set (1), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BCC rr	90_{16}, rr	2	4

(Note 1) The cycles-count increases by 2 when a branch occurs.

EOL announced

Function : Branch on condition

Operation : C = 1 ?

When C = 1 (True) PC ← PC + 2 ± REL

When C = 0 (False) PC ← PC + 2

❊ PG changes according to the result of the above PC operation

- if carry occurs in PC : PG ← PG + 1
- if borrow occurs in PC : PG ← PG – 1

❊ 2 is the number of instruction bytes of the BCS instruction

❊ REL is relative value (–128 to +127) indicated by the 2nd byte of the instruction

Description : When the carry flag C is set (1), the BCS instruction causes a branch to the specified address. The branch address is specified by a relative address.

When the carry flag C is clear (0), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BCS rr	$B0_{16}$, rr	2	4

(Note 1) The cycles-count increases by 2 when a branch occurs.

EOL announced

Function : Branch on condition

Operation : Z = 1 ?

When Z = 1 (True) PC ← PC + 2 ± REL

When Z = 0 (False) PC ← PC + 2

❊ PG changes according to the result of the above PC operation

- if carry occurs in PC : PG ← PG + 1
- if borrow occurs in PC : PG ← PG – 1

❊ 2 is the number of instruction bytes of the BEQ instruction

❊ REL is relative value (–128 to +127) indicated by the 2nd byte of the instruction

Description : When the zero flag Z is set (1), the BEQ instruction causes a branch to the specified address. The branch address is specified by a relative address.

When the zero flag Z is clear (0), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BEQ rr	$F0_{16}$, rr	2	4

EOL announced

Function : Branch on condition

Operation : N = 1 ?

When N = 1 (True) PC ← PC + 2 ± REL

When N = 0 (False) PC ← PC + 2

❊ PG changes according to the result of the above PC operation
- if carry occurs in PC : PG ← PG + 1
- if borrow occurs in PC : PG ← PG – 1

❊ 2 is the number of instruction bytes of the BMI instruction

❊ REL is relative value (–128 to +127) indicated by the 2nd byte of the instruction

Description : When the negative flag N is set (1), the BMI instruction causes a branch to the specified address. The branch address is specified by a relative address.
When the negative flag N is clear (0), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BMI rr	30_{16}, rr	2	4

(Note 1) The cycles-count increases by 2 when a branch occurs.

EOL announced

Function : Branch on condition

Operation : Z = 0 ?

When Z = 0 (True) PC ← PC + 2 ± REL

When Z = 1 (False) PC ← PC + 2

❊ PG changes according to the result of the above PC operation
- if carry occurs in PC : PG ← PG + 1
- if borrow occurs in PC : PG ← PG – 1

❊ 2 is the number of instruction bytes of the BNE instruction

❊ REL is relative value (–128 to +127) indicated by the 2nd byte of the instruction

Description : When the zero flag Z is clear (0), the BNE instruction causes a branch to the specified address. The branch address is specified by a relative address.
When the zero flag Z is set (1), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BNE rr	$D0_{16}$, rr	2	4

(Note 1) The cycles-count increases by 2 when a branch occurs.

EOL announced

Function : Branch on condition

Operation : N = 0 ?

When N = 0 (True) PC ← PC + 2 ± REL

When N = 1 (False) PC ← PC + 2

❊ PG changes according to the result of the above PC operation
- if carry occurs in PC : PG ← PG + 1
- if borrow occurs in PC : PG ← PG – 1

❊ 2 is the number of instruction bytes of the BPL instruction

❊ REL is relative value (–128 to +127) indicated by the 2nd byte of the instruction

Description : When the negative flag N is clear (0), the BPL instruction causes a branch to the specified address. The branch address is specified by a relative address.

When the negative flag N is set (1), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BPL rr	10_{16}, rr	2	4

(Note 1) The cycles-count increases by 2 when a branch occurs.

EOL announced

Function : Branch always

Operation : PC ← branch address (relative)

PC ← PC + n ± REL

❊ PG changes according to the result of the above PC operation
- if carry occurs in PC : PG ← PG + 1
- if borrow occurs in PC : PG ← PG – 1

❊ n is the number of instruction bytes in each addressing mode of the BRA instruction

❊ REL is relative value (–128 to +127) indicated by the last 1-byte or last 2-byte of the instruction

Branch area : For short relative –128 to +127

For long relative –32768 to +32767

Description : The BRA instruction causes a branch to the specified address. The branch address is specified by a relative address.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BRA rr	80_{16}, rr	2	4
	BRAL rr_1rr_2	82_{16}, rr_2, rr_1	3	4

EOL announced

Function : Software interrupt

Operation : Stack ← PG, PC, PS
I ← 1
PG, PC ← 00, Contents of BRK interrupt vector

PC ← PC + 2
M(S to S–4) ← PG, PC, PS
S ← S – 5
I ← 1
PG ← 00_{16}
PC ← M($FFFB_{16}$,$FFFA_{16}$)

(S) in just after instruction execution

(S) in just before instruction execution

Stack
PS_L
PS_H
PC_L
PC_H
PG

❊ "2" means the byte number of the BRK instruction, and "PC+2" is the address that stored the next instruction

Description : When the BRK instruction is executed, the CPU first saves the address where the next instruction is stored, and then saves the contents of the processor status register on the stack. Then, the CPU executes a branch to the address in bank-0 the lower portion of which is specified by the contents of $FFFA_{16}$ in bank-0 and the upper portion specified by the contents of $FFFB_{16}$ in bank-0.

EOL announced

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Set to 1.
Z : Not affected.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	BRK #nn	00_{16},EA_{16}	2	15

(Note 1) The instruction's second byte is ignored, so any value impossible.

Function : Branch on condition

Operation : V = 0 ?

When V = 0 (True) PC ← PC + 2 ± REL

When V = 1 (False) PC ← PC + 2

❊ PG changes according to the result of the above PC operation
- if carry occurs in PC : PG ← PG + 1
- if borrow occurs in PC : PG ← PG – 1

❊ 2 is the number of instruction bytes of the BRA instruction

❊ REL is relative value (–128 to +127) indicated by the 2nd byte of the instruction

Description : When the overflow flag V is clear (0), the BVC instruction causes a branch to the specified address. The branch address is specified by a relative address.

When the overflow flag V is set (1), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BVC rr	50_{16}, rr	2	4

(Note 1) The cycles-count increases by 2 when a branch occurs.

EOL announced

Function : Branch on condition

Operation : V = 1 ?

When V = 1 (True) PC ← PC + 2 ± REL

When V = 0 (False) PC ← PC + 2

❊ PG changes according to the result of the above PC operation
- if carry occurs in PC : PG ← PG + 1
- if borrow occurs in PC : PG ← PG – 1

❊ 2 is the number of instruction bytes of the BVS instruction

❊ REL is relative value (–128 to +127) indicated by the 2nd byte of the instruction

Description : When the overflow flag V is set (1), the BVS instruction causes a branch to the specified address. The branch address is specified by a relative address.
When the overflow flag V is clear (0), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BVS rr	70_{16}, rr	2	4

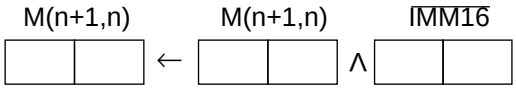
(Note 1) The cycles-count increases by 2 when a branch occurs.

EOL announced

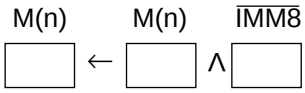
Function : Bit manipulation

Operation : Mb ← 0 (b is the specified bits)

When m=0



When m=1



❊ IMM is immediate value indicating the bit to be cleared with a “1” and is specified by the last 1 or 2 bytes of the instruction.

Description : The CLB instruction clears the specified memory bits to 0. Multiple bits to be cleared can be specified at one time.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct bit	CLB #imm, dd	14_{16}, dd, imm	3	8
Absolute bit	CLB #imm, mmll	$1C_{16}$, ll, mm, imm	4	9

(Note 1) The bytes-count increases by 1 when operating on 16-bit data with the data length selection flag m set to 0.

EOL announced

Function : Flag manipulation

Operation : C ← 0

Description : Clears the contents of carry flag C to 0.

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Not affected.
C : Cleared to 0.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	CLC	18_{16}	1	2

EOL announced

Function : Flag manipulation

Operation : I ← 0

Description : Clears the interrupt disable flag I to 0.

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Cleared to 0.
Z : Not affected.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	CLI	58_{16}	1	2

EOL announced

Function : Flag manipulation

Operation : m ← 0

Description : Clears the data length selection flag m to 0.

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Cleared to 0.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Not affected.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	CLM	$D8_{16}$	1	2

EOL announced

Function : Flag manipulation

Operation : PSLb ← 0 (b is the specified flags)

PSL ← PSL ∧ $\overline{IMM8}$

❊ IMM is a 1 byte immediate value indicating the flag to be cleared with a “1” and is specified by the second byte of the instruction.

b7	b6	b5	b4	b3	b2	b1	b0	
N	V	m	x	D	I	Z	C	PSL

Description : Clears the processor status flags specified by the bit pattern in the second byte of the instruction to 0.

Status flags : The specified status flags are cleared to “0”. IPL is not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	CLP #imm	$C2_{16}$, imm	2	4

EOL announced

Function : Flag manipulation

Operation : V ← 0

Description : Clears the overflow flag V to 0.

Status flags

IPL : Not affected.
N : Not affected.
V : Cleared to 0.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Not affected.
C : Not affected.

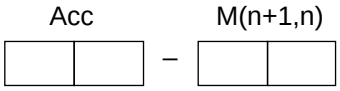
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	CLV	$B8_{16}$	1	2

EOL announced

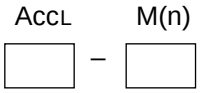
Function : Compare

Operation : Acc – M

When m=0



When m=1



Description : Subtracts the contents of memory from the contents of the accumulator. The accumulator and memory contents are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Set to 1 if the result of operation is 0 or larger. Otherwise, cleared to 0.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	CMP A, #imm	$C9_{16}$, imm	2	2
Direct	CMP A, dd	$C5_{16}$, dd	2	4
Direct indexed X	CMP A, dd, X	$D5_{16}$, dd	2	5
Direct indirect	CMP A, (dd)	$D2_{16}$, dd	2	6
Direct indexed X indirect	CMP A, (dd, X)	$C1_{16}$, dd	2	7
Direct indirect indexed Y	CMP A, (dd), Y	$D1_{16}$, dd	2	8
Direct indirect long	CMPL A, (dd)	$C7_{16}$, dd	2	10
Direct indirect long indexed Y	CMPL A, (dd), Y	$D7_{16}$, dd	2	11
Absolute	CMP A, mmll	CD_{16}, ll, mm	3	4
Absolute indexed X	CMP A, mmll, X	DD_{16}, ll, mm	3	6
Absolute indexed Y	CMP A, mmll, Y	$D9_{16}$, ll, mm	3	6
Absolute long	CMP A, hhmmll	CF_{16}, ll, mm, hh	4	6
Absolute long indexed X	CMP A, hhmmll, X	DF_{16}, ll, mm, hh	4	7
Stack pointer relative	CMP A, nn, S	$C3_{16}$, nn	2	5
Stack pointer relative indirect indexed Y	CMP A, (nn, S), Y	$D3_{16}$, nn	2	8

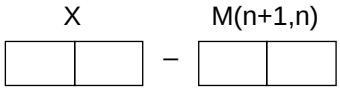
(Note 1) This table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

(Note 2) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

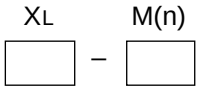
Function : Compare

Operation : X – M

When x=0



When x=1



Description : Subtracts the contents of memory from the contents of the index register X. The index register X and memory contents are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Set to 1 if the result of operation is 0 or larger. Otherwise, cleared to 0.

EOL announced

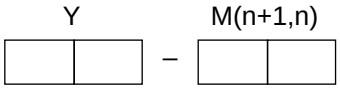
Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	CPX #imm	$E0_{16}$, imm	2	2
Direct	CPX dd	$E4_{16}$, dd	2	4
Absolute	CPX mmll	EC_{16}, ll, mm	3	4

(Note 1) When operating on 16-bit data in the immediate addressing mode with the index register length selection flag x set to 0, the bytes-count increases by 1.

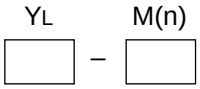
Function : Compare

Operation : Y – M

When x=0



When x=1



Description : Subtracts the contents of memory from the contents of the index register Y. The index register Y and memory contents are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Set to 1 if the result of operation is 0 or larger. Otherwise, cleared to 0.

EOL announced

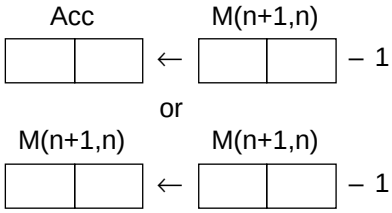
Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	CPY #imm	$C0_{16}$, imm	2	2
Direct	CPY dd	$C4_{16}$,dd	2	4
Absolute	CPY mmll	CC_{16}, ll, mm	3	4

(Note 1) When operating on 16-bit data in the immediate addressing mode with the index register length selection flag x set to 0, the bytes-count increases by 1.

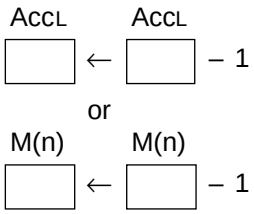
Function : Decrement

Operation : Acc ← Acc – 1 or M ← M – 1

When m=0



When m=1



Description : Subtracts 1 from the contents of the accumulator or memory.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

EOL announced

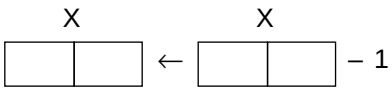
Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	DEC A	$1A_{16}$	1	2
Direct	DEC dd	$C6_{16}$, dd	2	7
Direct indexed X	DEC dd, X	$D6_{16}$, dd	2	7
Absolute	DEC mmll	CE_{16}, ll, mm	3	7
Absolute indexed X	DEC mmll, X	DE_{16}, ll, mm	3	8

(Note 1) The accumulator addressing mode's specification in this table applies when using the accumulator A. If using the accumulator B, replace “A” with “B”. In this case, “42_{16}” is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

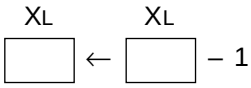
Function : Decrement

Operation : X ← X – 1

When x=0



When x=1



Description : Subtracts 1 from the contents of the index register X.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

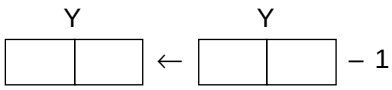
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	DEX	CA_{16}	1	2

EOL announced

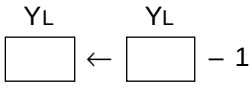
Function : Decrement

Operation : Y ← Y – 1

When x=0



When x=1



Description : Subtracts 1 from the contents of the index register Y.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

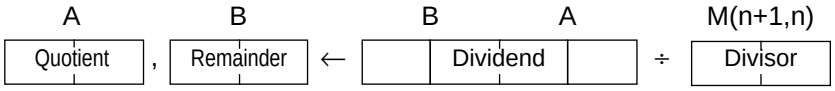
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	DEY	88_{16}	1	2

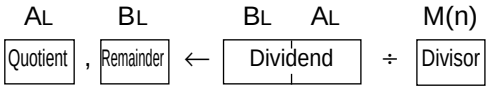
Function : Division (Unsigned)

Operation : A(quotient), B(remainder) ← (B, A) / M

When m=0



When m=1



Description : When the data length selection flag m is set to 0, a 32-bit data stored in the accumulators B (upper 16 bits) and A (lower 16 bits) are divided by a 16-bit data in memory. The quotient is placed in the accumulator A, and the remainder is placed in the accumulator B.

When the data length selection flag m is set to 1, a 16-bit data stored in the lower 8 bits of the accumulators B (upper 8 bits) and A (lower 8 bits) are divided by an 8 bit data in memory. The quotient is placed in the lower 8 bits of the accumulator A, and the remainder is placed in the lower 8 bits of the accumulator B.

If an overflow occurs as a result of the operation, the V flag is set and the content of the accumulator is unpredictable.

When divisor is 0, the zero division interrupt is generated, in which case the contents of the program bank register, program counter, and processor status register are saved on the stack and a branch occurs to the address in bank-0 as specified by the zero division interrupt vector. Accumulator contents are not changed. In this case, the content of the accumulator is unchanged.

EOL announced

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of quotient from the operation is 1. Otherwise, cleared to 0.

❊ When an overflow occurs as a result of the operation or divisor is 0, N flag is not affected.

V : Clear to 0.

❊ Set to 1 when an overflow occurs

❊ Not affected when divisor is 0

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the quotient from the operation is 0. Otherwise, cleared to 0. No changes occur when divisor is 0.

❊ When an overflow occurs as a result of the operation or divisor is 0, Z flag is not affected.

C : Clear to 0.

❊ Set to 1 when an overflow occurs

❊ Not affected when divisor is 0

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	DIV #imm	89_{16}, 29_{16}, imm	3	27
Direct	DIV dd	89_{16}, 25_{16}, dd	3	29
Direct indexed X	DIV dd, X	89_{16}, 35_{16}, dd	3	30
Direct indirect	DIV (dd)	89_{16}, 32_{16}, dd	3	31
Direct indexed X indirect	DIV (dd, X)	89_{16}, 21_{16}, dd	3	32
Direct indirect indexed Y	DIV (dd), Y	89_{16}, 31_{16}, dd	3	33
Direct indirect long	DIVL (dd)	89_{16}, 27_{16}, dd	3	35
Direct indirect long indexed Y	DIVL (dd), Y	89_{16}, 37_{16}, dd	3	36
Absolute	DIV mmll	89_{16}, $2D_{16}$, ll, mm	4	29
Absolute indexed X	DIV mmll, X	89_{16}, $3D_{16}$, ll ,mm	4	31
Absolute indexed Y	DIV mmll, Y	89_{16}, 39_{16}, ll ,mm	4	31
Absolute long	DIV hhmmll	89_{16}, $2F_{16}$, ll, mm, hh	5	31
Absolute long indexed X	DIV hhmmll, X	89_{16}, $3F_{16}$, ll, mm, hh	5	32
Stack pointer relative	DIV nn, S	89_{16}, 23_{16}, nn	3	30
Stack pointer relative indirect indexed Y	DIV (nn, S), Y	89_{16}, 33_{16}, nn	3	33

(Note 1) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

(Note 2) The cycles-count in this table are for 16-bit ÷ 8-bit operations. For 32-bit ÷ 16-bit operations, the cycles-count increases by 16.

(Note 3) The cycle count in this table and Note 2 are the value when the operation completes normally (no interrupt has occurred). If a zero divide interrupt has occurred, the cycle counts in the above table are decremented by 8 regardless of the data length.

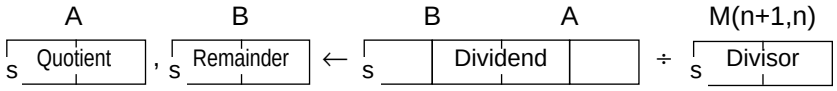
EOL announced

Function : Division (Signed)

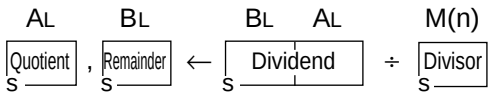
This instruction can be used in the 7750 Series only.

Operation : A(quotient), B(remainder) ← (B, A) / M

When m=0



When m=1



❊ "s" means a sign bit that is the most significant bit of the data

Description : When the data length selection flag m is set to 0, a signed 32-bit data stored in the accumulators B (upper 16 bits) and A (lower 16 bits) are divided by a signed 16-bit data in memory. As a result of the operation, the quotient is stored in accumulator A and the remainder is stored in accumulator B as signed 16-bit data.

When the data length selection flag m is set to 1, a signed 16-bit data stored in the lower 8 bits of the accumulators B (upper 8 bits) and A (lower 8 bits) are divided by a signed 8 bit data in memory. As a result of the operation, the quotient is stored in low-order 8 bits of accumulator A and the remainder is stored in low-order 8 bits of accumulator B as signed 8-bit data.

The sign of remainder becomes same as dividend.

When an overflow results from this operation (the quotient exceeds the extent –32767 to +32767 if m=0 or –127 to +127 if m=1) neglect removed out, the V flag is set. In this case, the content of the accumulator is unpredictable.

When divisor is 0, the zero division interrupt is generated, in which case the contents of the processor status register are saved on the stack and a branch occurs to the address in bank 0_{16} as specified by the zero division interrupt vector. And accumulator contents are not changed.

EOL announced

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of quotient from the operation is 1. Otherwise, cleared to 0.

❊ When an overflow occurs as a result of the operation or divisor is 0, N flag is not affected.

V : Clear to 0.

❊ Set to 1 when an overflow occurs

❊ Not affected when divisor is 0

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

❊ Set to 1 when divisor is 0

Z : Set to 1 when the quotient from the operation is 0. Otherwise, cleared to 0. No changes occur when divisor is 0.

❊ When an overflow occurs as a result of the operation or divisor is 0, Z flag is not affected.

C : Clear to 0.

❊ Set to 1 when an overflow occurs

❊ Not affected when divisor is 0

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	DIVS #imm	89_{16}, $A9_{16}$, imm	3	29
Direct	DIVS dd	89_{16}, $A5_{16}$, dd	3	31
Direct indexed X	DIVS dd, X	89_{16}, $B5_{16}$, dd	3	32
Direct indirect	DIVS (dd)	89_{16}, $B2_{16}$, dd	3	33
Direct indexed X indirect	DIVS (dd, X)	89_{16}, $A1_{16}$, dd	3	34
Direct indirect indexed Y	DIVS (dd), Y	89_{16}, $B1_{16}$, dd	3	35
Direct indirect long	DIVSL (dd)	89_{16}, $A7_{16}$, dd	3	37
Direct indirect long indexed Y	DIVSL (dd), Y	89_{16}, $B7_{16}$, dd	3	38
Absolute	DIVS mmll	89_{16}, AD_{16}, ll, mm	4	31
Absolute indexed X	DIVS mmll, X	89_{16}, BD_{16}, ll ,mm	4	33
Absolute indexed Y	DIVS mmll, Y	89_{16}, $B9_{16}$, ll ,mm	4	33
Absolute long	DIVS hhmmll	89_{16}, AF_{16}, ll, mm, hh	5	33
Absolute long indexed X	DIVS hhmmll, X	89_{16}, BF_{16}, ll, mm, hh	5	34
Stack pointer relative	DIVS nn, S	89_{16}, $A3_{16}$, nn	3	32
Stack pointer relative indirect indexed Y	DIVS (nn, S), Y	89_{16}, $B3_{16}$, nn	3	35

(Note 1) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

(Note 2) The cycles-count in this table are for 16-bit ÷ 8-bit operations. For 32-bit ÷ 16-bit operations, the cycles-count increases by 16.

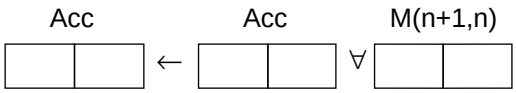
(Note 3) The cycle count in this table and Note 2 are the value when the operation completes normally (no interrupt has occurred). If a zero divide interrupt has occurred, the cycle counts in the above table are decremented by 10 regardless of the data length.

ECL announced

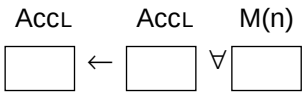
Function : Logical EXCLUSIVE OR

Operation : Acc ← Acc ∀ M

When m=0



When m=1



Description : Performs the logical EXCLUSIVE OR between the contents of the accumulator and the contents of memory, and places the result in the accumulator.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	EOR A, #imm	49_{16}, imm	2	2
Direct	EOR A, dd	45_{16}, dd	2	4
Direct indexed X	EOR A, dd, X	55_{16}, dd	2	5
Direct indirect	EOR A, (dd)	52_{16}, dd	2	6
Direct indexed X indirect	EOR A, (dd, X)	41_{16}, dd	2	7
Direct indirect indexed Y	EOR A, (dd), Y	51_{16}, dd	2	8
Direct indirect long	EORL A, (dd)	47_{16}, dd	2	10
Direct indirect long indexed Y	EORL A, (dd), Y	57_{16}, dd	2	11
Absolute	EOR A, mmll	$4D_{16}$, ll, mm	3	4
Absolute indexed X	EOR A, mmll, X	$5D_{16}$, ll, mm	3	6
Absolute indexed Y	EOR A, mmll, Y	59_{16}, ll, mm	3	6
Absolute long	EOR A, hhmmll	$4F_{16}$, ll, mm, hh	4	6
Absolute long indexed X	EOR A, hhmmll, X	$5F_{16}$, ll, mm, hh	4	7
Stack pointer relative	EOR A, nn, S	43_{16}, nn	2	5
Stack pointer relative indirect indexed Y	EOR A, (nn, S), Y	53_{16}, nn	2	8

(Note 1) This table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

(Note 2) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

Function : Extension with sign

This instruction can be used in the 7750 Series only.

Operation : ACCH ← 00_{16} or FF_{16}

When bit 7 of ACCL="0"

ACCH ← 00_{16}

ACCH	ACCL		ACCH	ACCL
00000000	0XXXXXXX	←	?	0XXXXXXX

When bit 7 of ACCL="1"

ACCH ← FF_{16}

ACCH	ACCL		ACCH	ACCL
11111111	1XXXXXXX	←	?	1XXXXXXX

❊ The high-order byte of Acc changes regardless of the m flag

Description : This instruction is used to extend the signed 8-bit data stored in the low-order byte of the accumulator to a 16-bit data.

If bit 7 of the accumulator is "0", bits 8 to 15 are set to "0". If bit 7 of the accumulator is "1", bits 8 to 15 are set to "1".

With this instruction, the high-order byte of accumulator changes regardless of the data length selection flag m, but the content of the data length selection flag m is unchanged.

Status flags

IPL : Not affected.
N : Set to 1, if bit 15 of the result of operation is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1, if the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	EXTS A	89_{16}, $8B_{16}$	2	8
Accumulator	EXTS B	42_{16}, $8B_{16}$	2	8

Function : Extension zero

This instruction can be used in the 7750 Series only.

Operation : ACCH ← 0016

ACCH	ACCL		ACCH	ACCL
0016		←	?	

❊ The high-order byte of Acc changes regardless of the m flag

Description : This instruction is used to extend the 8-bit data stored in the low-order byte of the accumulator to a 16-bit data.

Bits 8 to 15 of the accumulator are set to “0”.

With this instruction, the high-order byte of accumulator changes regardless of the data length selection flag m, but the content of the data length selection flag m is unchanged.

Status flags

IPL : Not affected.
N : Set to “0”
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1, if the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

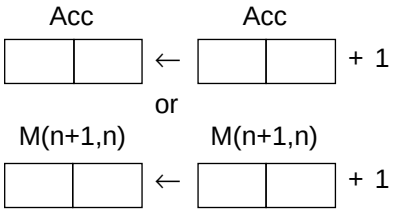
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	EXTZ A	89_{16}, AB_{16}	2	5
Accumulator	EXTZ B	42_{16}, AB_{16}	2	5

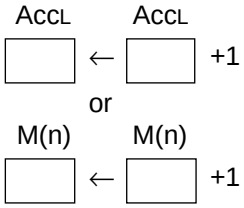
Function : Increment

Operation : Acc ← Acc + 1 or M ← M + 1

When m=0



When m=1



Description : Adds 1 to the contents of the accumulator or memory.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

EOL announced

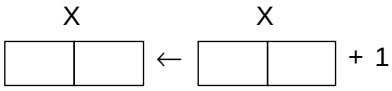
Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	INC A	$3A_{16}$	1	2
Direct	INC dd	$E6_{16}$, dd	2	7
Direct indexed X	INC dd, X	$F6_{16}$, dd	2	7
Absolute	INC mmll	EE_{16}, ll, mm	3	7
Absolute indexed X	INC mmll, X	FE_{16}, ll, mm	3	8

(Note 1) The accumulator addressing mode's specification in this table applies when using the accumulator A. If using the accumulator B, replace “A” with “B”. In this case, “42_{16}” is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

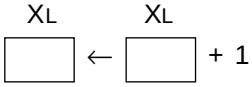
Function : Increment

Operation : X ← X + 1

When x=0



When x=1



Description : Adds 1 to the contents of the index register X.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

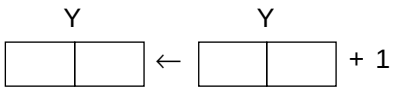
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	INX	$E8_{16}$	1	2

EOL announced

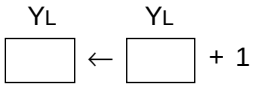
Function : Increment

Operation : Y ← Y + 1

When x=0



When x=1



Description : Adds 1 to the contents of the index register Y.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	INY	$C8_{16}$	1	2

EOL announced

Function : Jump always

Operation : [PG], PC ← specified address (absolute or indirect)

When the addressing mode is ...

absolute addressing mode,

PC ← ADDR

absolute long addressing mode,

PC ← ADDR
PG ← BANK

absolute indirect addressing mode,

PC ← M(ADDR+1, ADDR)

absolute indirect long addressing mode,

PC ← M(ADDR+1, ADDR)
PG ← M(ADDR+2)

absolute indexed X indirect addressing mode,

PC ← M(ADDR+X+1, ADDR+X)

❊ ADDR indicates the low-order 16 bits of a 24-bit address and is specified by bytes 2 and 3 of the instruction.

❊ BANK is the high-order 8 bits of a 24-bit address and is specified by byte 4 of the instruction.

EOL announced

Description : The JMP instruction causes a jump to the address specified for the addressing mode in use.

When this instruction is used in addressing mode other than absolute long, the content of PG is incremented by 1 and the branch destination becomes the next bank if the last byte of the instruction is at the topmost address (XXFFFF$_{16}$) of a bank or if the instruction spans across banks.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Absolute	JMP mmll	4C$_{16}$, ll, mm	3	2
Absolute long	JMPL hhmmll	5C$_{16}$, ll, mm, hh	4	4
Absolute indirect	JMP (mmll)	6C$_{16}$, ll, mm	3	4
Absolute indirect long	JMPL (mmll)	DC$_{16}$, ll, mm	3	8
Absolute indexed X indirect	JMP (mmll, X)	7C$_{16}$, ll, mm	3	6

Function : Jump to subroutine

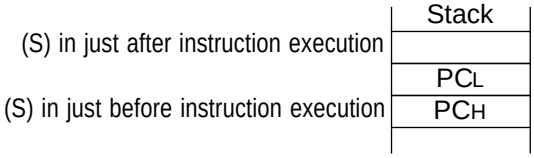
Operation : Stack ← [PG], PC

[PG], PC ← specified address (absolute or indirect)

When the addressing mode is ...

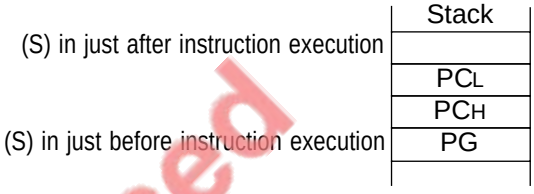
absolute addressing mode,

PC ← PC + 3
M(S,S–1) ← PC
S ← S – 2
PC ← ADDR



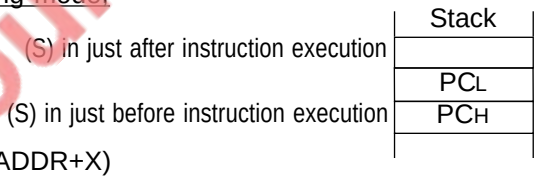
absolute long addressing mode,

PC ← PC + 4
M(S to S–2) ← PG, PC
S ← S – 3
PC ← ADDR
PG ← BANK



absolute indexed X indirect addressing mode,

PC ← PC + 3
M(S,S–1) ← PC
S ← S – 2
PC ← M(ADDR+X+1,ADDR+X)



EOL announced

❊ ADDR indicates the low-order 16 bits of a 24-bit address and is specified by bytes 2 and 3 of the instruction.

❊ BANK is the high-order 8 bits of a 24-bit address and is specified by byte 4 of the instruction.

Description : The contents of the program counter PC (or the program bank register PG and the program counter PC if absolute long addressing mode) are first saved on the stack, then a jump occurs to the address shown for each addressing mode.

When this instruction is used in addressing mode other than absolute long, the content of PG is incremented by 1 and the branch destination becomes the next bank if the last byte of the instruction is at the topmost address (XXFFFF16) of a bank or if the instruction spans across banks.

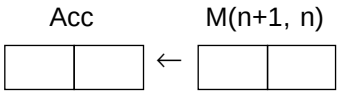
Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Absolute	JSR mmll	20_{16}, ll, mm	3	6
Absolute long	JSRL hhmmll	22_{16}, ll, mm, hh	4	8
Absolute indexed X indirect	JSR (mmll, X)	FC_{16}, ll, mm	3	8

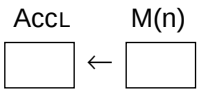
Function : Load

Operation : Acc ← M

When m=0



When m=1



Description : Loads the contents of memory into the accumulator.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	LDA A, #imm	$A9_{16}$, imm	2	2
Direct	LDA A, dd	$A5_{16}$, dd	2	4
Direct indexed X	LDA A, dd, X	$B5_{16}$, dd	2	5
Direct indirect	LDA A, (dd)	$B2_{16}$, dd	2	6
Direct indexed X indirect	LDA A, (dd, X)	$A1_{16}$, dd	2	7
Direct indirect indexed Y	LDA A, (dd), Y	$B1_{16}$, dd	2	8
Direct indirect long	LDAL A, (dd)	$A7_{16}$, dd	2	10
Direct indirect long indexed Y	LDAL A, (dd), Y	$B7_{16}$, dd	2	11
Absolute	LDA A, mmll	AD_{16}, ll, mm	3	4
Absolute indexed X	LDA A, mmll, X	BD_{16}, ll, mm	3	6
Absolute indexed Y	LDA A, mmll, Y	$B9_{16}$, ll, mm	3	6
Absolute long	LDA A, hhmmll	AF_{16}, ll, mm, hh	4	6
Absolute long indexed X	LDA A, hhmmll, X	BF_{16}, ll, mm, hh	4	7
Stack pointer relative	LDA A, nn, S	$A3_{16}$, nn	2	5
Stack pointer relative indirect indexed Y	LDA A, (nn, S), Y	$B3_{16}$, nn	2	8

(Note 1) This table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

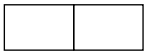
(Note 2) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

Function : Load

Operation : M ← IMM

When m=0

M(n+1, n)



← IMM16

When m=1

M(n)

← IMM8

Description : Loads an immediate value into memory.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct	LDM #imm, dd	64_{16}, dd, imm	3	4
Direct indexed X	LDM #imm, dd, X	74_{16}, dd, imm	3	5
Absolute	LDM #imm, mmll	$9C_{16}$, ll, mm, imm	4	5
Absolute indexed X	LDM #imm, mmll, X	$9E_{16}$, ll, mm, imm	4	6

(Note 1) When operating on 16-bit data with the data length selection flag m set to 0, the bytes-count increases by 1.

EOL announced

Function : Load

Operation : DT ← IMM8

DT
□ ← IMM8

Description : Loads an immediate value into the data bank register DT.

Status flags : Not affected.

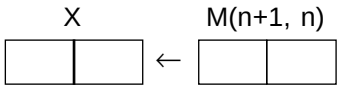
Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	LDT #imm	89_{16}, $C2_{16}$, imm	3	5

EOL announced

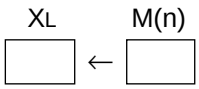
Function : Load

Operation : X ← M

When x=0



When x=1



Description : Loads the contents of memory into the index register X.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

EOL announced

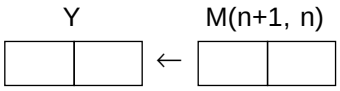
Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	LDX #imm	$A2_{16}$, imm	2	2
Direct	LDX dd	$A6_{16}$, dd	2	4
Direct indexed Y	LDX dd, Y	$B6_{16}$, dd	2	5
Absolute	LDX mmll	AE_{16}, ll, mm	3	4
Absolute indexed Y	LDX mmll, Y	BE_{16}, ll, mm	3	6

(Note 1) When operating on 16-bit data in the immediate addressing mode with the index register length selection flag x set to 0, the bytes-count increases by 1.

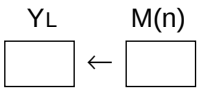
Function : Load

Operation : Y ← M

When x=0



When x=1



Description : Loads the contents of memory into the index register Y.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

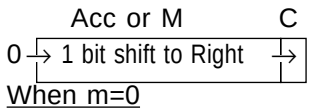
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	LDY #imm	$A0_{16}$, imm	2	2
Direct	LDY dd	$A4_{16}$, dd	2	4
Direct indexed X	LDY dd, X	$B4_{16}$, dd	2	5
Absolute	LDY mmll	AC_{16}, ll, mm	3	4
Absolute indexed X	LDY mmll, X	BC_{16}, ll, mm	3	6

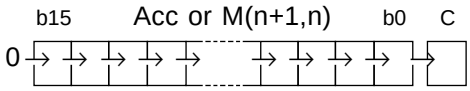
(Note 1) When operating on 16-bit data in the immediate addressing mode with the index register length selection flag x set to 0, the bytes-count increases by 1.

Function : Logical shift right

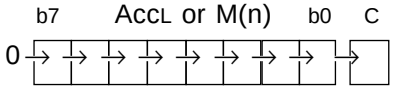
Operation :



When m=0



When m=1



Description : Shifts all bits of the accumulator or memory one place to the right. Bit 15 (or bit 7 if the data length selection flag m is set to 1) of the accumulator or memory is loaded with 0.

The carry flag C is loaded from bit 0 of the data before the shift.

Status flags

IPL : Not affected.
N : Cleared to “0”.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Set to 1 when bit 0 before the operation is 1. Otherwise, cleared to 0.

EOL announced

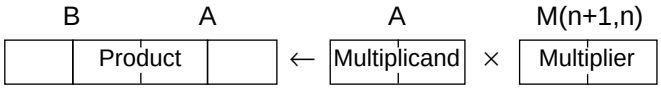
Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	LSR A	$4A_{16}$	1	2
Direct	LSR dd	46_{16}, dd	2	7
Direct indexed X	LSR dd, X	56_{16}, dd	2	7
Absolute	LSR mmll	$4E_{16}$, ll, mm	3	7
Absolute indexed X	LSR mmll, X	$5E_{16}$, ll, mm	3	8

(Note 1) The accumulator addressing mode's specification in this table applies when using the accumulator A. If using the accumulator B, replace “A” with “B”. In this case, “42_{16}” is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

Function : Multiplication (Unsigned)

Operation : B, A ← A × M

When m=0



When m=1

Description : When the data length selection flag m is set to 0, the contents of the accumulator A and the contents of memory are multiplied. Multiplication is performed as 16-bit × 16-bit, and the result is a 32-bit data which is placed in the accumulators B (upper 16 bits of the result) and A (lower 16 bits of the result).

When the data length selection flag m is set to 1, the lower 8-bit contents of the accumulator A and the contents of memory are multiplied. Multiplication is performed as 8-bit × 8-bit, and the result is a 16-bit data which is placed in the lower 8 bits of the accumulators B (upper 8 bits of the result) and A (lower 8 bits of the result).

Status flags

IPL : Not affected.

N : Set to 1 when bit 31 (or bit 15 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Cleared to 0.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	MPY #imm	89_{16}, 09_{16}, imm	3	16
Direct	MPY dd	89_{16}, 05_{16}, dd	3	18
Direct indexed X	MPY dd, X	89_{16}, 15_{16}, dd	3	19
Direct indirect	MPY (dd)	89_{16}, 12_{16}, dd	3	20
Direct indexed X indirect	MPY (dd, X)	89_{16}, 01_{16}, dd	3	21
Direct indirect indexed Y	MPY (dd), Y	89_{16}, 11_{16}, dd	3	22
Direct indirect long	MPYL (dd)	89_{16}, 07_{16}, dd	3	24
Direct indirect long indexed Y	MPYL (dd), Y	89_{16}, 17_{16}, dd	3	25
Absolute	MPY mmll	89_{16}, $0D_{16}$, ll, mm	4	18
Absolute indexed X	MPY mmll, X	89_{16}, $1D_{16}$, ll, mm	4	20
Absolute indexed Y	MPY mmll, Y	89_{16}, 19_{16}, ll, mm	4	20
Absolute long	MPY hhmmll	89_{16}, $0F_{16}$, ll, mm, hh	5	20
Absolute long indexed X	MPY hhmmll, X	89_{16}, $1F_{16}$, ll, mm, hh	5	21
Stack pointer relative	MPY nn, S	89_{16}, 03_{16}, nn	3	19
Stack pointer relative indirect indexed Y	MPY (nn, S), Y	89_{16}, 13_{16}, nn	3	22

(Note 1) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

(Note 2) The cycles-count in this table are for 8-bit × 8-bit multiplications. For 16-bit × 16-bit multiplications, the cycles-count increases by 8.

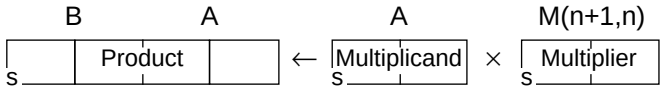
EOL announced

Function : Multiplication (Signed)

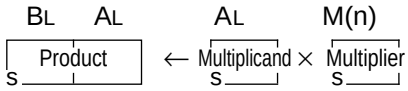
This instruction can be used in the 7750 Series only.

Operation : B, A ← A × M

When m=0



When m=1



❊ s indicates the sign bit and is the topmost bit of the data to be operated on.

Description : When the data length selection flag m is set to 0, the content of the accumulator A is multiplied by the content of memory as signed data. The multiplication is performed as a 16-bit × 16-bit operation, and the result is a 32-bit data which is placed in the accumulators B (upper 16 bits of the result) and A (lower 16 bits of the result). Bit 15 of accumulator B becomes the sign bit.

When the data length selection flag m is set to 1, the lower 8-bit content of the accumulator A is multiplied by the content of memory as signed data. The multiplication is performed as 8-bit × 8-bit operation, and the result is a 16-bit data which is placed in the lower 8 bits of the accumulators B (upper 8 bits of the result) and A (lower 8 bits of the result). Bit 7 of accumulator B becomes the sign bit.

Status flags

IPL : Not affected.

N : Set to 1 when bit 31 of the operation result which is bit 15 of accumulator B (or if the data length selection flag m is set to 1, bit 15 of the operation result which is bit 7 of accumulator B) is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	MPYS #imm	89_{16}, 89_{16}, imm	3	18
Direct	MPYS dd	89_{16}, 85_{16}, dd	3	20
Direct indexed X	MPYS dd, X	89_{16}, 95_{16}, dd	3	21
Direct indirect	MPYS (dd)	89_{16}, 92_{16}, dd	3	22
Direct indexed X indirect	MPYS (dd, X)	89_{16}, 81_{16}, dd	3	23
Direct indirect indexed Y	MPYS (dd), Y	89_{16}, 91_{16}, dd	3	24
Direct indirect long	MPYSL (dd)	89_{16}, 87_{16}, dd	3	26
Direct indirect long indexed Y	MPYSL (dd), Y	89_{16}, 97_{16}, dd	3	27
Absolute	MPYS mmll	89_{16}, $8D_{16}$, ll, mm	4	20
Absolute indexed X	MPYS mmll, X	89_{16}, $9D_{16}$, ll, mm	4	22
Absolute indexed Y	MPYS mmll, Y	89_{16}, 99_{16}, ll, mm	4	22
Absolute long	MPYS hhmmll	89_{16}, $8F_{16}$, ll, mm, hh	5	22
Absolute long indexed X	MPYS hhmmll, X	89_{16}, $9F_{16}$, ll, mm, hh	5	23
Stack pointer relative	MPYS nn, S	89_{16}, 83_{16}, nn	3	21
Stack pointer relative indirect indexed Y	MPYS (nn, S), Y	89_{16}, 93_{16}, nn	3	24

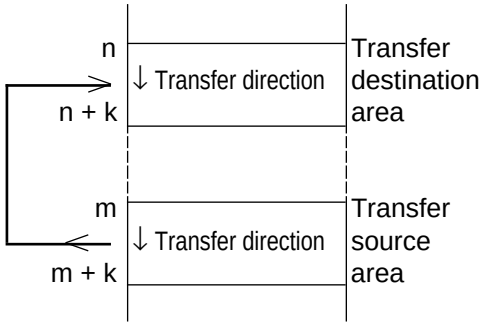
(Note 1) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

(Note 2) The cycles-count in this table are for 8-bit × 8-bit multiplications with the same sign. If the multiplier and multiplicand have different signs, three additional cycles are required. For 16-bit × 16-bit multiplications, the cycles-count increases by 8.

EOL announced

Function : Move

Operation : M(n to n + k) ← M(m to m + k)



A = 0 ?

When A=0

Instruction execution complete

When A≠0

Repeat operation

M(DTd: Y) M(DTs: X)

X ← X + 2

Y ← Y + 2

A ← A – 2

❊ DTd indicates the transfer destination bank and is specified with the second byte of the instruction.

❊ DTs indicates the transfer source bank and is specified with the third byte of the instruction.

❊ Values set in register before transfer

A: Transfer byte count

When m=0 The value 0 to 65535 can be set

When m=1 The value 0 to 255 can be set

X: Transfer source area beginning (lowermost) address

When x=0 The value 0 to 65535 can be set

When x=1 The value 0 to 255 can be set (Note 1)

Y: Transfer destination area beginning (lowermost) address

When x=0 The value 0 to 65535 can be set

When x=1 The value 0 to 255 can be set (Note 1)

❊ Content of register after transfer

A: $FFFF_{16}$

X: Transfer source area end (highermost) address + 1

Y: Transfer destination area beginning (highermost) address + 1

DT: Bank number of transfer destination

❊ Transfer is always performed two bytes at a time. Therefore, in the case of a 16-bit bus, the transfer time is shorter when transfer start addresses are even compared to when they are both odd addresses.

Note 1. This instruction is recommended to use at the x="0". Data in the area between $XX00_{16}$ and $XXFF_{16}$ can be only used because the higher bytes of index registers X and Y are not affected at x="1".

EOL announced

Description : Normally, a block of data is transferred from upper addresses to lower addresses. The transfer is performed in the ascending address order of the block being transferred. The target bank is specified by the instruction's second byte, and the address within the target bank is specified by the contents of the index register Y. The source bank is specified by the instruction's third byte, and the address within the source bank is specified by the contents of the index register X. The accumulator A is loaded with the bytes-count of the data to be transferred. As each byte of data is transferred, the index registers X and Y are incremented by 1, so that the index register X will become a value equal to 1 larger than the source address of the last byte transferred and the index register Y will become a value equal to 1 larger than the target address of the last byte received. The data bank register DT will become the target bank number, and the accumulator A will become $FFFF_{16}$.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Block transfer	MVN n_1, n_2	54_{16}, n_1, n_2	3	7+(i/2)×7

(Note 1)The cycles-count shown above is for when the number of bytes transferred, i, is an even number. If i is an odd number, the cycles-count is obtained as follows:

$7 + (i \div 2) \times 7 + 4$.

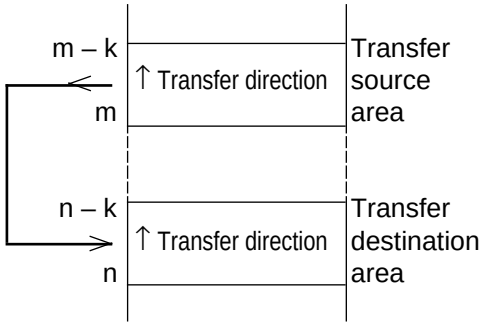
Note that $(i \div 2)$ denotes the integer part of the result of dividing i by 2.

EOL announced

Function : Move

Operation : M(n – k to n) ← M(m – k to m)

A = 0 ?
When A=0
Instruction execution complete
When A≠0
Repeat operation
X ← X – 1
Y ← Y – 1
M(DTd: Y) M(DTs: X)
X ← X – 1
Y ← Y – 1
A ← A – 2



❊ DTd indicates the transfer destination bank and is specified with the second byte of the instruction.

❊ DTs indicates the transfer source bank and is specified with the third byte of the instruction.

❊ Values set in register before transfer

A: Transfer byte count
When m=0 The value 0 to 65535 can be set
When m=1 The value 0 to 255 can be set

X: Transfer source area beginning (highermost) address
When x=0 The value 0 to 65535 can be set
When x=1 The value 0 to 255 can be set

Y: Transfer destination area beginning (highermost) address
When x=0 The value 0 to 65535 can be set
When x=1 The value 0 to 255 can be set

❊ Content of register after transfer

A: FFFF16
X: Transfer source area end (lowermost) address – 1
Y: Transfer destination area beginning (lowermost) address – 1
DT: Bank number of transfer destination

❊ Transfer is always performed two bytes at a time. Therefore, in the case of a 16-bit bus, the transfer time is shorter when transfer start addresses are odd compared to when they are both even addresses.

Note 1. This instruction is recommended to use at the x="0". Data in the area between XX0016 and XXFF16 can be only used because the higher bytes of index registers X and Y are not affected at x="1".

EOL announced

Description : Normally, a block of data is transferred from lower addresses to upper addresses. The transfer is performed in the descending address order of the block being transferred. The target bank is specified by the instruction's second byte, and the address within the target bank is specified by the contents of the index register Y. The source bank is specified by the instruction's third byte, and the address within the source bank is specified by the contents of the index register X. The accumulator A is loaded with the bytes-count of the data to be transferred. As each byte of data is transferred, the index registers X and Y are decremented by 1, so that the index register X will become a value equal to 1 less than the source address of the last byte transferred and the index register Y will become a value equal to 1 smaller than the target address of the last byte received. The data bank register DT will become the target bank number, and the accumulator A will become $FFFF_{16}$.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Block transfer	MVP n_1, n_2	44_{16}, n_1, n_2	3	9+(i/2)×7

(Note 1)The cycles-count shown above is for when the number of bytes transferred, i, is an even number. If i is an odd number, the cycles-count is obtained as follows:

$9 + (i \div 2) \times 7 + 5$.

Note that $(i \div 2)$ denotes the integer part of the result of dividing i by 2.

EOL announced

Function : No operation

Operation : PC ← PC + 1

❊ PG also changes depending on the result of the above operation on PC.

If a carry occurs in PC: PG ← PG + 1

Description : This instruction only causes the program counter to be incremented by 1 and nothing else.

Status flags : Not affected.

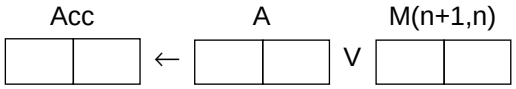
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	NOP	EA_{16}	1	2

EOL announced

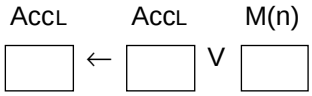
Function : Logical OR

Operation : Acc ← Acc V M

When m=0



When m=1



Description : Performs the logical OR between the contents of the accumulator and the contents of memory, and places the result in the accumulator.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	ORA A, #imm	09_{16}, imm	2	2
Direct	ORA A, dd	05_{16}, dd	2	4
Direct indexed X	ORA A, dd, X	15_{16}, dd	2	5
Direct indirect	ORA A, (dd)	12_{16}, dd	2	6
Direct indexed X indirect	ORA A, (dd, X)	01_{16}, dd	2	7
Direct indirect indexed Y	ORA A, (dd), Y	11_{16}, dd	2	8
Direct indirect long	ORAL A, (dd)	07_{16}, dd	2	10
Direct indirect long indexed Y	ORAL A, (dd), Y	17_{16}, dd	2	11
Absolute	ORA A, mmll	$0D_{16}$, ll, mm	3	4
Absolute indexed X	ORA A, mmll, X	$1D_{16}$, ll, mm	3	6
Absolute indexed Y	ORA A, mmll, Y	19_{16}, ll, mm	3	6
Absolute long	ORA A, hhmmll	$0F_{16}$, ll, mm, hh	4	6
Absolute long indexed X	ORA A, hhmmll, X	$1F_{16}$, ll, mm, hh	4	7
Stack pointer relative	ORA A, nn, S	03_{16}, nn	2	5
Stack pointer relative indirect indexed Y	ORA A, (nn, S), Y	13_{16}, nn	2	8

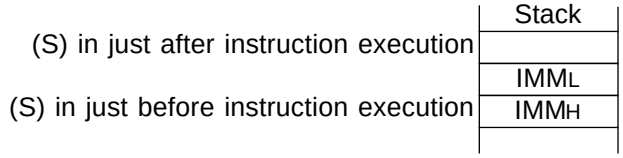
(Note 1) This table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

(Note 2) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

Function : Stack manipulation (push)

Operation : Stack ← IMM16

M(S,S – 1) ← IMM16
S ← S – 2



❊ IMM16 is an immediate value and IMMH indicates its high-order byte and IMML indicates its low-order byte.

Description : The instruction's third and second bytes are saved on the stack in this order.

Status flags : Not affected.

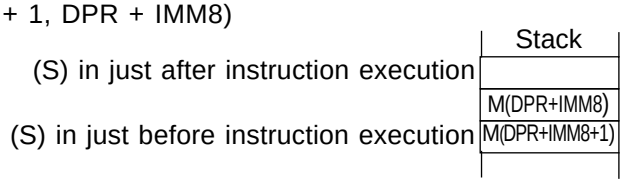
Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PEA #immHimmL	$F4_{16}$, immL, immH	3	5

EOL announced

Function : Stack manipulation (push)

Operation : Stack ← M(DPR + IMM8 + 1, DPR + IMM8)

M(S,S – 1) ← M(DPR + IMM8 + 1, DPR + IMM8)
S ← S – 2



❊ IMM8 is an 8-bit immediate value and is used as an offset from DPR.

Description : Saves the contents of the consecutive 2 bytes in the direct page as specified by the sum of the contents of the direct page register DPR and the instruction's second byte on the stack in the order of upper address first and lower address second.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PEI #imm	$D4_{16}$, imm	2	6

EOL announced

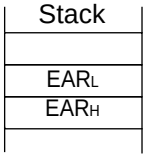
Function : Stack manipulation (push)

Operation : Stack ← PC + IMM16

EAR ← PC + IMM16
M(S, S – 1) ← EAR
S ← S – 2

(S) in just after instruction execution

(S) in just before instruction execution



❊ IMM16 is a 16-bit immediate value

❊ EAR is an execution address added PC and IMM16, and EAR_H indicates its high-order byte and EAR_L indicates its low-order byte.

Description : Saves the result of adding a 16-bit data consisting of an upper byte specified by the instruction's third byte and a lower byte specified by the instruction's second byte with the contents of the program counter on the stack in the order of the result's upper byte first and lower byte second.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PER #$imm_H imm_L$	62_{16}, imm_L, imm_H	3	5

EOL announced

EOL announced

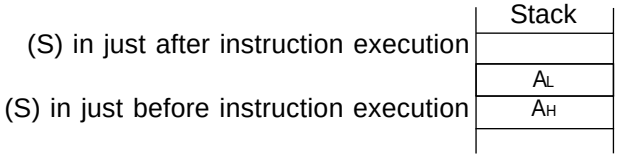
Function : Stack manipulation (push)

Operation : Stack ← A

When m=0

M(S, S – 1) ← A

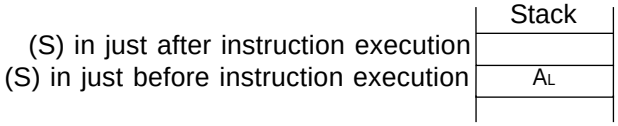
S ← S – 2



When m=1

M(S) ← AL

S ← S – 1



Description : Saves the contents of the accumulator A to the address specified by the stack pointer S. When the data length selection flag m is set to 0, the accumulator A's upper byte is saved on the stack first and then the lower byte. When the data length selection flag m is set to 1, only the accumulator A's lower byte is saved on the stack.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHA	48_{16}	1	4

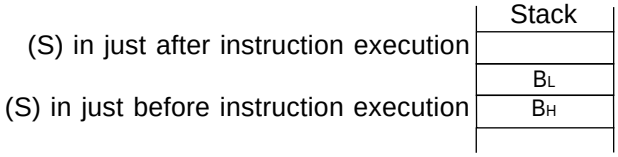
Function : Stack manipulation (push)

Operation : Stack ← B

When m=0

M(S, S – 1) ← B

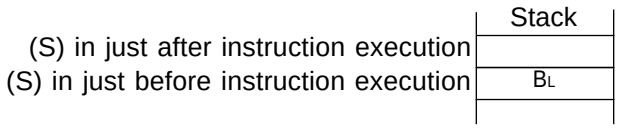
S ← S – 2



When m=1

M(S) ← BL

S ← S – 1



Description : Saves the contents of the accumulator B to the address indicated by the stack pointer S. When the data length selection flag m is set to 0, the accumulator B's upper byte is saved on the stack first and then the lower byte. When the data length selection flag m is set to 1, only the accumulator B's lower byte is saved on the stack.

Status flags : Not affected.

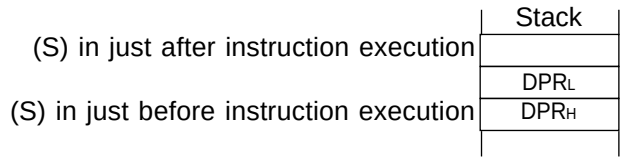
Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHB	42_{16}, 48_{16}	2	6

EOL announced

Function : Stack manipulation (push)

Operation : Stack ← DPR

M(S, S – 1) ← DPR
S ← S – 2



Description : Saves the contents of the direct page register DPR to the address indicated by the stack pointer S in the order of upper byte first and then lower byte.

Status flags : Not affected.

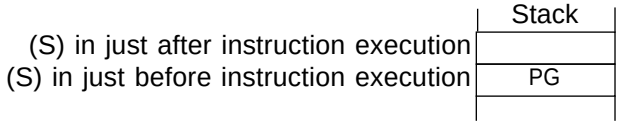
Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHD	$0B_{16}$	1	4

EOL announced

Function : Stack manipulation (push)

Operation : Stack ← PG

M(S) ← PG
S ← S – 1



Description : Saves the contents of the program bank register to the address indicated by the stack pointer S.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHG	$4B_{16}$	1	3

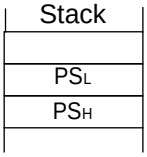
EOL announced

Function : Stack manipulation (push)

Operation : Stack ← PS

M(S, S – 1) ← PS
S ← S – 2

(S) in just after instruction execution
(S) in just before instruction execution



Description : Saves the contents of the processor status register PS to the address indicated by the stack pointer S in the order of upper byte and then lower byte.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHP	08_{16}	1	4

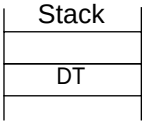
EOL announced

Function : Stack management (push)

Operation : Stack ← DT

M(S) ← DT
S ← S – 1

(S) in just after instruction execution
(S) in just before instruction execution



Description : Saves the contents of the data bank register DT to the address indicated by the stack pointer S.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHT	$8B_{16}$	1	3

EOL announced

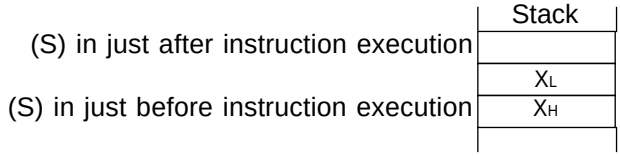
Function : Stack management (push)

Operation : Stack ← X

When x=0

M(S, S – 1) ← X

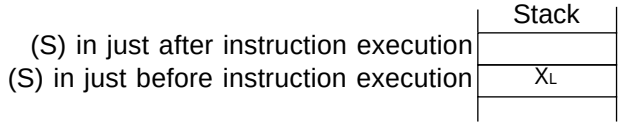
S ← S – 2



When x=1

M(S) ← XL

S ← S – 1



Description : Saves the contents of the index register X to the address indicated by the stack pointer S. When the index register length selection flag x is set to 0, the contents are saved in the order of upper byte and then lower byte. When the index register length selection flag x is set to 1, only the lower byte is saved on the stack.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHX	DA_{16}	1	4

EOL announced

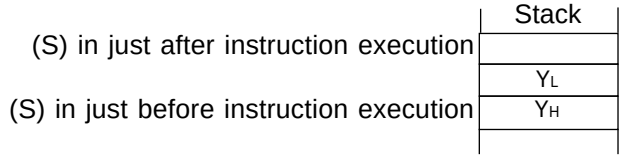
Function : Stack management (push)

Operation : Stack ← Y

When x=0

M(S, S – 1) ← Y

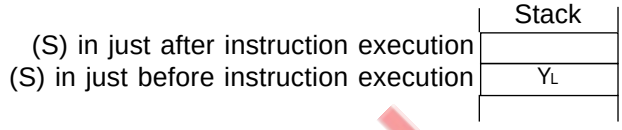
S ← S – 2



When x=1

M(S) ← YL

S ← S – 1



Description : Saves the contents of the index register Y to the address indicated by the stack pointer S. When the index register length selection flag x is set to 0, the contents are saved in the order of upper byte and then lower byte. When the index register length selection flag x is set to 1, only the lower byte is saved on the stack.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHY	$5A_{16}$	1	4

EOL announced

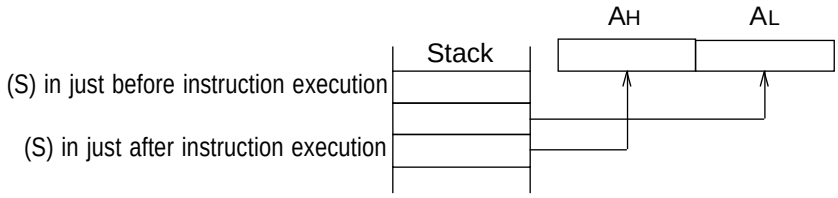
Function : Stack manipulation (pull)

Operation : A ← Stack

When m=0

A ← M(S+2, S+1)

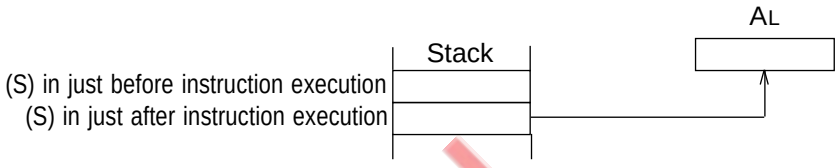
S ← S + 2



When m=1

AL ← M(S+1)

S ← S + 1



Description : The stack pointer S is incremented, and then restores the lower byte of the accumulator A with the data at the address indicated by the stack pointer S. Again, increments the stack pointer S and then restores the upper byte of the accumulator A with the data at the address indicated by the stack pointer S. When the data length selection flag m is set to 0, 2 bytes data are restored. When the data length selection flag m is set to 1, only 1 byte data is restored (to the lower byte of the accumulator A).

EOL announced

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

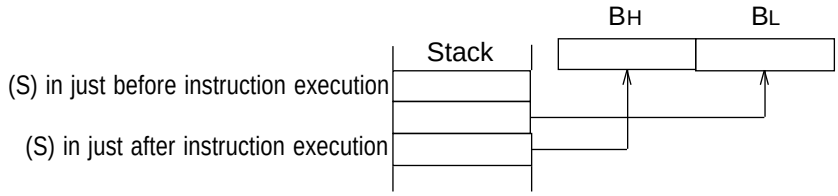
Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLA	68_{16}	1	5

Function : Stack manipulation (pull)

Operation : B ← Stack

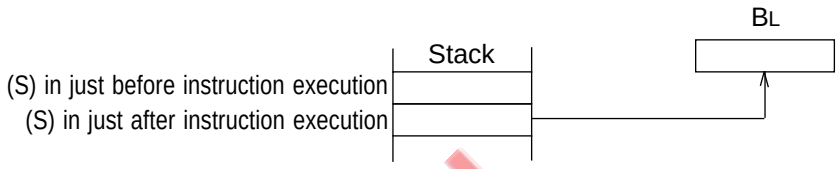
When m=0

B ← M(S+2, S+1)
S ← S + 2



When m=1

BL ← M(S+1)
S ← S + 1



Description : The stack pointer S is incremented, and then restores the lower byte of the accumulator B with the data at the address indicated by the stack pointer S. Again, increments the stack pointer S and then restores the upper byte of the accumulator B with the data at the address indicated by the stack pointer S. When the data length selection flag m is set to 0, 2 bytes data are restored. When the data length selection flag m is set to 1, only 1 byte data is restored (to the lower byte of the accumulator B).

EOL announced

Status flags

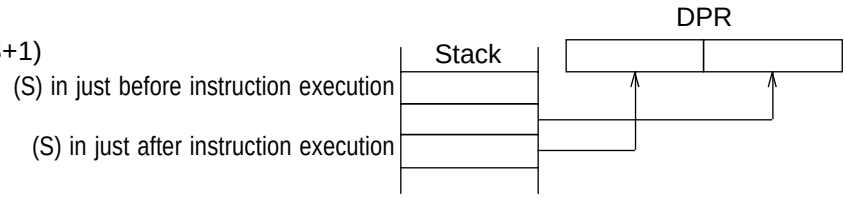
IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLB	42_{16}, 68_{16}	2	7

Function : Stack manipulation (pull)

Operation : DPR ← Stack

DPR ← M(S+2, S+1)
S ← S + 2



Description : The stack pointer S is incremented, and then the direct page register DPR is restored with the data at the address indicated by the stack pointer .

Status flags : Not affected.

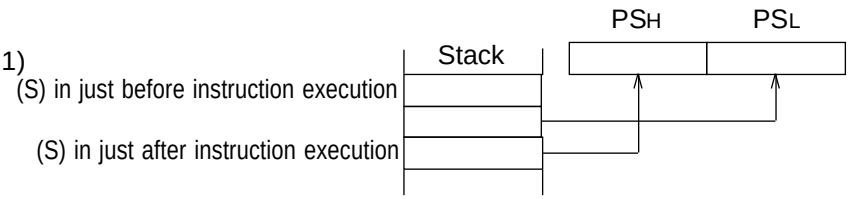
Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLD	$2B_{16}$	1	5

EOL announced

Function : Stack manipulation (pull)

Operation : PS ← Stack

PS ← M(S+2, S+1)
S ← S + 2



Description : The stack pointer S is incremented and then the processor status register PS is restored with the data at the address indicated by the stack pointer S.

Status flags : Changes to the values restored from the stack.

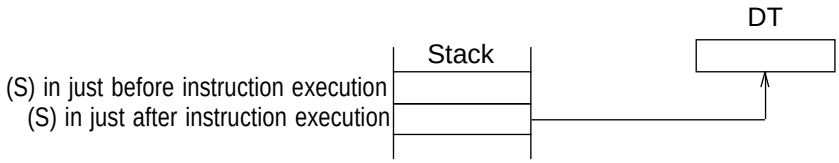
Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLP	28_{16}	1	6

EOL announced

Function : Stack manipulation (pull)

Operation : DT ← Stack

DT ← M(S+1)
S ← S + 1



Description : The stack pointer S is incremented, and then the data bank register DT is restored with the data at the address indicated by the stack pointer S.

Status flags

IPL : Not affected.
N : Set to 1 when bit 7 of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLT	AB_{16}	1	6

EOL announced

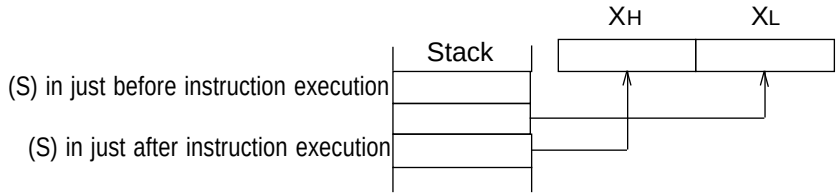
Function : Stack manipulation (pull)

Operation : X ← Stack

When x=0

X ← M(S+2, S+1)

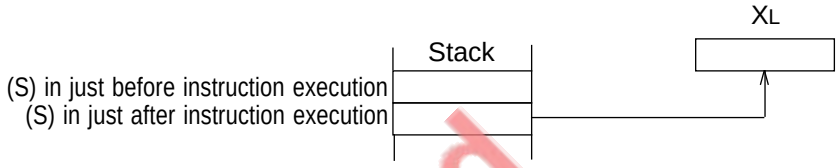
S ← S + 2



When x=1

XL ← M(S+1)

S ← S + 1



Description : The stack pointer S is incremented, and then restores the lower byte of the index register X with the data at the address indicated by the stack pointer S. Again, increments the stack pointer S and then restores the upper byte of the index register X with the data at the address indicated by the stack pointer S. When the index register length selection flag x is set to 0, 2 bytes are restored. When the index register length selection flag x is set to 1, only 1 byte is restored (to the lower byte of the index register X).

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLX	FA_{16}	1	5

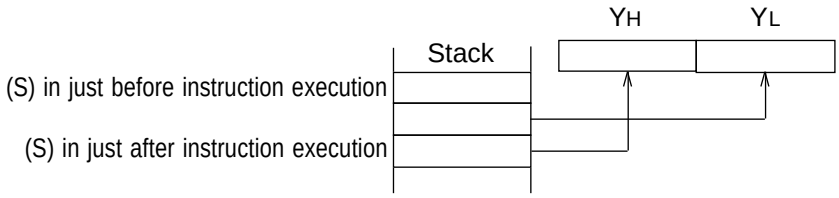
Function : Stack manipulation (pull)

Operation : Y ← Stack

When x=0

Y ← M(S+2, S+1)

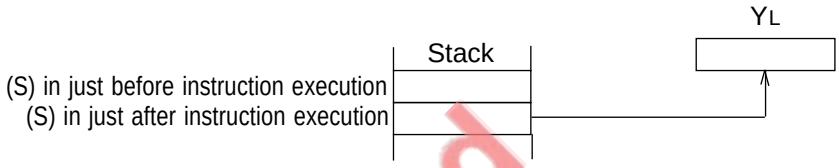
S ← S + 2



When x=1

YL ← M(S+1)

S ← S + 1



Description : The stack pointer S is incremented, and then restores the lower byte of the index register Y with the data at the address indicated by the stack pointer S. Again, increments the stack pointer S and then restores the upper byte of the index register Y with the data at the address indicated by the stack pointer S. When the index register length selection flag x is set to 0, 2 bytes are restored. When the index register length selection flag x is set to 1, only 1 byte is restored (to the lower byte of the index register Y).

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLY	$7A_{16}$	1	5

Function : Stack manipulation (push)

Operation : Stack ← Specified register of A, B, X, Y, DPR, DT, PG, PS

M(S to S – n) ← A, B, X, Y, DPR, DT, PG, PS
order to save ① ② ③ ④ ⑤ ⑥ ⑦ ⑧
S ← S – n – 1

❊ The immediate value in the second byte of the instruction is used to specify the registers to be saved.

❊ Among the registers being saved, the following registers are affected by the flags just before the instruction is executed.

- A, B registers
  When m=0 : The high-order and low-order bytes of the register are saved.
  When m=1 : The low-order bytes of the register are saved.
- X, Y registers
  When x=0 : The high-order and low-order bytes of the register is saved.
  When x=1 : The low-order byte of the register is saved.

❊ n indicates the number of data bytes to be saved.

Description : This instruction's second byte specifies the registers to be saved. The registers corresponding to the bits in the second byte that are 1 are saved on the stack. The bit and register correspondence is as shown below:

b7							b0
PS	PG	DT	DPR	Y	X	B	A

← Direction to save on the stack

When saving the registers to stack, registers A and B are affected by the m flag and registers X and Y are affected by the x flag.

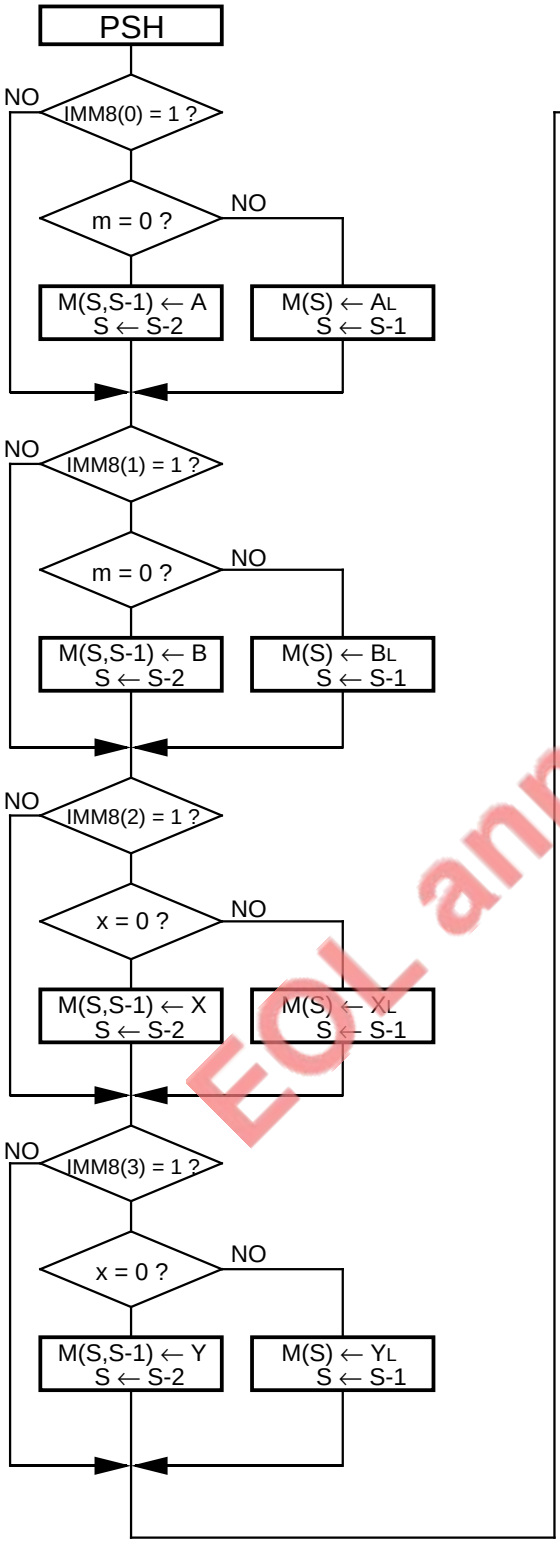
Status flags : Not affected.

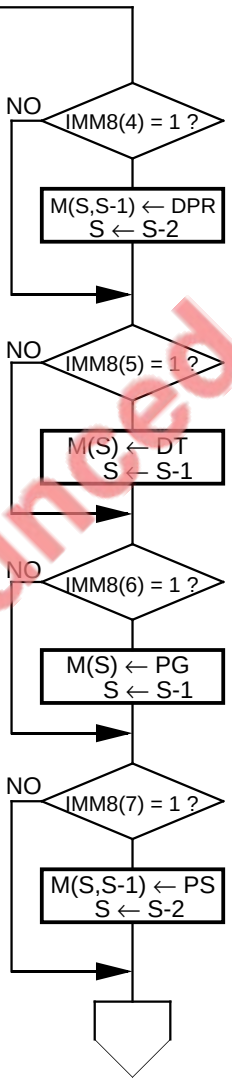
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PSH #nn	EB_{16}, nn	2	$12+2xi_1+i_2$

(Note 1) To the cycles-count shown above, the values shown below are added depending on the registers being saved. The count is 12 cycles when no registers are saved. i_1 in above table represents the number of registers (chosen from A, B, X, Y, DPR and PS) to be saved, and i_2 represents the number of registers (chosen from DT and PG) to be saved.

Register type	PS	PG	DT	DPR	Y	X	B	A
Cycles-count	2	1	1	2	2	2	2	2





❊ IMM8 is an immediate value in 1-byte and inside of () specifies the contents of the bit at the value.

EOL announced

Function : Stack manipulation (pull)

Operation : Specified register of A, B, X, Y, DPR, DT, PG, PS ← Stack

A, B, X, Y, DPR, DT, PG, PS ← M(S + 1 to S + n)
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ order to restore
S ← S + n

❊ The immediate value in the second byte of the instruction is used to specify the registers to be restored.

❊ Among the registers being restored, the following registers are affected by the flags in the restored PS or the flags just before the instruction is executed.

- A, B registers

  When m=0 : The high-order and low-order bytes of the register are restored.

  When m=1 : The low-order bytes of the register are restored.

- X, Y registers

  When x=0 : The high-order and low-order bytes of the register is restored.

  When x=1 : The low-order byte of the register is restored.

❊ n indicates the number of data bytes to be restored.

Description : This instruction's second byte specifies the registers to be restored. The registers corresponding to the bits in the second byte that are 1 are restored from the stack. The bit and register correspondence is as shown below:

b7							b0
PS		DT	DPR	Y	X	B	A

Direction to restore from the stack →

When restoring from stack, registers A and B are affected by the m flag in restored PS, and registers X and Y are affected by the x flag in restored PS. If PS is not restored, the registers are affected by the value of these flags just before instruction execution.

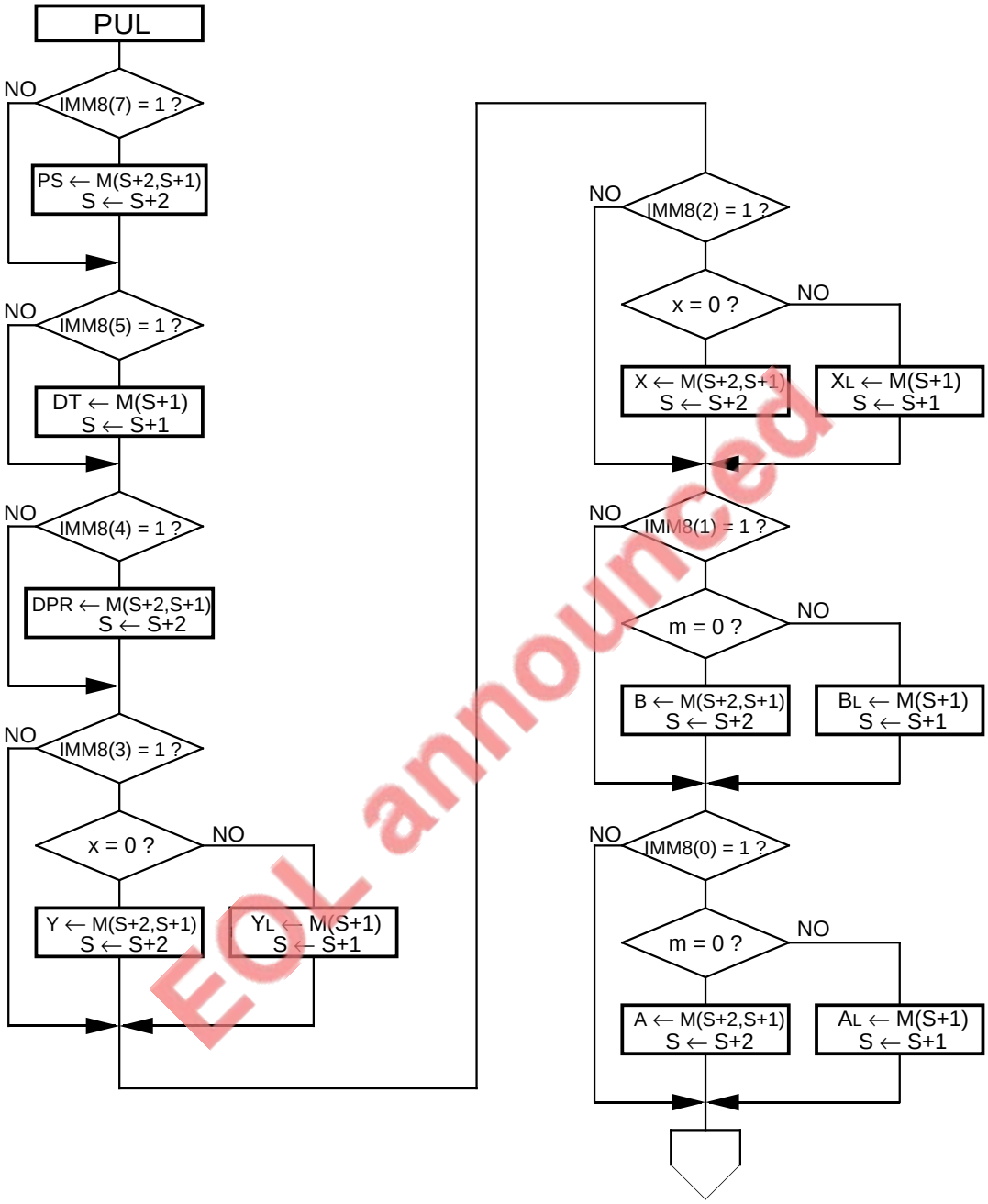
Status flags : When bit 7 of the instruction's second byte is 1, specifying that the program status register PS is to be restored, the status flags are restored to the values that had been restored from the stack. Otherwise, the status flags are not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PUL #nn	FB_{16}, nn	2	$14+3xi_1+4xi_2$

(Note 1) To the cycles-count shown above, the values shown below are added depending on the registers being restored. The count is 14 cycles when no registers are restored. i_1 in above table represents the number of registers (chosen from A, B, X, Y, PS and DT) to be saved. i_2=1 if DPR is to be restored, and i_2=0 if DPR is not to be restored.

Register type	PS	DT	DPR	Y	X	B	A
Cycles-count	3	3	4	3	3	3	3

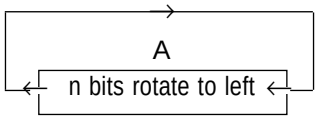


EOL announced

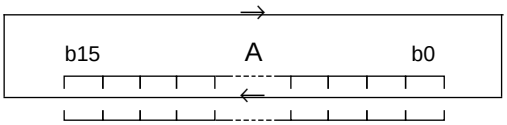
❊ IMM8 is an immediate value in 1-byte and inside of () specifies the contents of the bit at the value.

Function : n bits rotate to left

Operation :

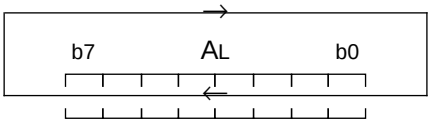


When m=0



n bits rotate to left
❊ n = 0 to 65535

When m=1



n bits rotate to left
❊ n = 0 to 255

Description : The contents of the accumulator A are rotated to the left by n bits. The value of n is specified by the instruction's third byte (or third and fourth bytes when m=0).

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	RLA #imm	89_{16}, 49_{16}, imm	3	6+i

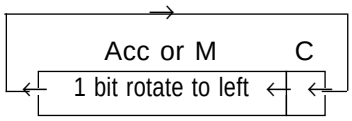
i: Number of rotation

(Note 1) When the data length selection flag m is 0, the bytes-count increases by 1.

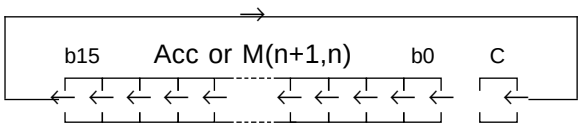
EOL announced

Function : Rotate to left

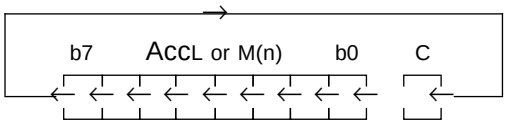
Operation :



When m=0



When m=1



Description : The carry flag C is linked to the accumulator or memory, and the combined contents are rotated by 1 bit to the left.

Bit 0 of the accumulator or memory is loaded with the content of the carry flag C before execution of this instruction, and the carry flag C is loaded with the content of bit 15 (or bit 7 if the data length selection flag m is set to 1) of the accumulator or memory before execution of this instruction.

EOL announced

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

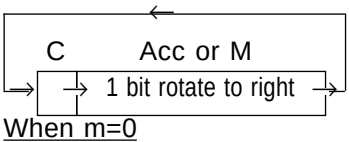
C : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) before execution of the instruction is 1. Otherwise, cleared to 0

Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	ROL A	$2A_{16}$	1	2
Direct	ROL dd	26_{16}, dd	2	7
Direct indexed X	ROL dd, X	36_{16}, dd	2	7
Absolute	ROL mmll	$2E_{16}$, ll, mm	3	7
Absolute indexed x	ROL mmll, X	$3E_{16}$, ll, mm	3	8

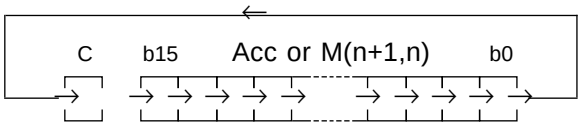
(Note 1) The accumulator addressing mode's specification in this table applies when using the accumulator A. If using the accumulator B, replace “A” with “B”. In this case, “42_{16}” is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

Function : Rotate to right

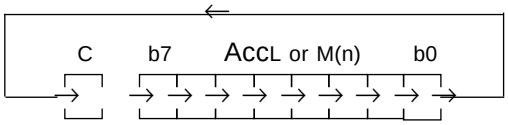
Operation :



When m=0



When m=1



Description : The carry flag C is linked to the accumulator or memory, and the combined contents are shifted by 1 bit to the right.

Bit 15 (or bit 7 if the data length selection flag m is set to 1) of the accumulator or memory is loaded with the content of the carry flag C, and the carry flag C is loaded with the content of bit 0 of the accumulator or memory before execution of this instruction.

EOL announced

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Set to 1 when bit 0 before execution of the instruction is 1. Otherwise, cleared to 0.

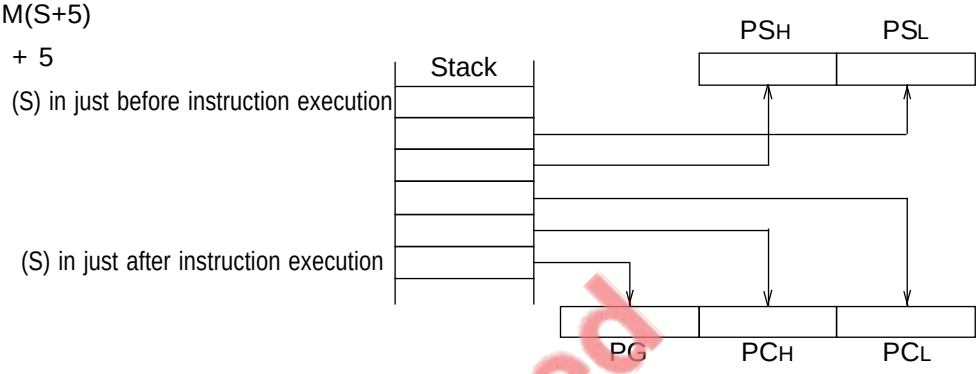
Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	ROR A	$6A_{16}$	1	2
Direct	ROR dd	66_{16}, dd	2	7
Direct indexed X	ROR dd, X	76_{16}, dd	2	7
Absolute	ROR mmll	$6E_{16}$, ll, mm	3	7
Absolute indexed X	ROR mmll, X	$7E_{16}$, ll, mm	3	8

(Note 1) The accumulator addressing mode's specification in this table applies when using the accumulator A. If using the accumulator B, replace “A” with “B”. In this case, “42_{16}” is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

Function : Return from subroutine

Operation : PG, PC, PS ← Stack (Saved content when interrupt occurred)

PS ← M(S+2, S+1)
PC ← M(S+4, S+3)
PG ← M(S+5)
S ← S + 5



Description : The contents of the processor status register PS, program counter PC, and program bank register PG, which are saved on the stack when the last interrupt was accepted, are restored these registers.

Status flags : Restored according to the values that had been on the stack.

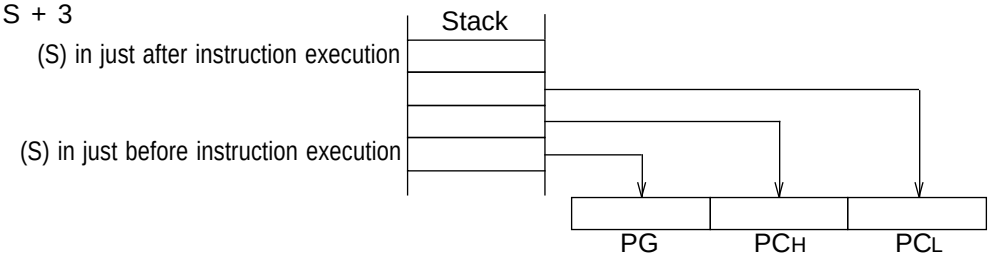
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	RTI	40_{16}	1	11

EOL announced

Function : Return from subroutine

Operation : PG, PC ← Stack (Subroutine long return address)

PC ← M(S+2, S+1)
PG ← M(S+3)
S ← S + 3



Description : The program counter PC and program bank register PG are restored according to the state previously saved on the stack.

Status flags : Not affected.

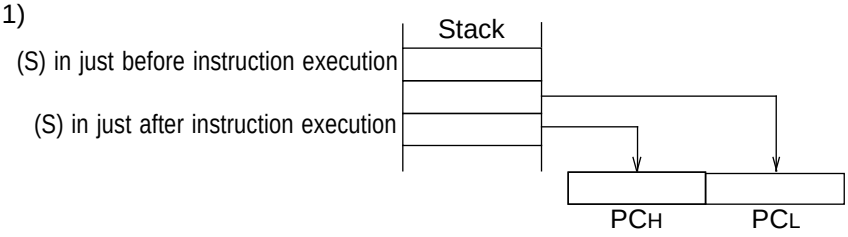
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	RTL	$6B_{16}$	1	8

EOL announced

Function : Return from subroutine

Operation : PC ← Stack (Subroutine return address)

PC ← M(S+2, S+1)
S ← S + 2



Description : The program counter PC is restored according to the state previously saved on the stack.
The contents of PG is added 1 when this instruction is at topmost address ($XXFFFF_{16}$) of a bank.

Status flags : Not affected.

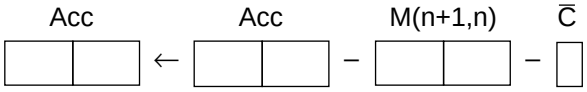
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	RTS	60_{16}	1	5

EOL announced

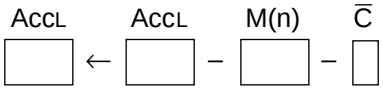
Function : Subtract with carry

Operation : Acc ← Acc – M – $\overline{C}$

When m=0



When m=1



Description : Subtracts the contents of memory and the 1's complements of carry flag from the contents of the accumulator , and places the result in the accumulator. Executed as a binary subtraction if the decimal operation mode flag D is set to 0. Executed as a decimal subtraction if the decimal operation mode flag D is set to 1.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0. Meaningless for decimal subtraction.

V : Set to 1 when binary subtraction of signed data results in a value outside the range of -32768 to +32767 (-128 to +127 if the data length selection flag m is set to 1). Otherwise, cleared to 0. Meaningless for decimal subtraction.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Set to 1 when the result of operation is equal to or larger than 0. Otherwise, cleared to 0, and a borrow is indicated.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	SBC A, #imm	$E9_{16}$, imm	2	2
Direct	SBC A, dd	$E5_{16}$, dd	2	4
Direct indexed X	SBC A,dd, X	$F5_{16}$, dd	2	5
Direct indirect	SBC A, (dd)	$F2_{16}$, dd	2	6
Direct indexed X indirect	SBC A,(dd, X)	$E1_{16}$, dd	2	7
Direct indirect indexed Y	SBC A,(dd), Y	$F1_{16}$, dd	2	8
Direct indirect long	SBCL A, (dd)	$E7_{16}$, dd	2	10
Direct indirect long indexed Y	SBCL A, (dd), Y	$F7_{16}$, dd	2	11
Absolute	SBC A,mmll	ED_{16},ll,mm	3	4
Absolute indexed X	SBC A, mmll, X	FD_{16}, ll, mm	3	6
Absolute indexed Y	SBC A, mmll, Y	$F9_{16}$, ll, mm	3	6
Absolute long	SBC A, hhmmll	EF_{16}, ll, mm, hh	4	6
Absolute long indexed X	SBC A, hhmmll, X	FF_{16}, ll, mm, hh	4	7
Stack pointer relative	SBC A, nn, S	$E3_{16}$, nn	2	5
Stack pointer relative indirect indexed Y	SBC A, (nn, S), Y	$F3_{16}$, nn	2	8

(Note 1) This table applies when using the accumulator A. If using the accumulator B, replace “A” with “B”. In this case, “42_{16}” is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

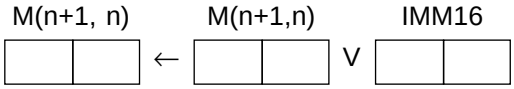
(Note 2) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

EOL announced

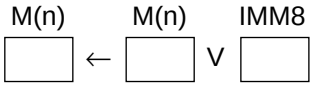
Function : Bit management

Operation : Mb ← 1 (b is the specified bits)

When m=0



When m=1



❊ IMM is an immediate value indicating the bit to be set with a “1” and is specified by the last 1 or 2 bytes of the instruction.

Description : The SEB instruction sets the specified memory bits to 1. Multiple bits to be set can be specified at one time.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct bit	SEB #imm, dd	04_{16}, dd, imm	3	8
Absolute bit	SEB #imm, mmll	$0C_{16}$, ll, mm, imm	4	9

(Note 1) When operating on 16-bit data with the data length selection flag m set to 0, the bytes-count increases by 1.

EOL announced

Function : Flag manipulation

Operation : C ← 1

Description : Sets the carry flag C to 1.

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Not affected.
C : Set to 1.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	SEC	38_{16}	1	2

EOL announced

Function : Flag manipulation

Operation : I ← 1

Description : Sets the interrupt disable flag I to 1.

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Set to 1.
Z : Not affected.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	SEI	78_{16}	1	2

EOL announced

Function : Flag manipulation

Operation : m ← 1

Description : Sets the data length selection flag m to 1.

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Set to 1.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Not affected.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	SEM	$F8_{16}$	1	2

EOL announced

Function : Flag manipulation

Operation : PSLb ← 1 (b is the specified flags)

PSL ← PSL V IMM8

❊ IMM8 is an immediate value indicating the bit to be set with a “1” and is specified by the last 1 or 2 bytes of the instruction.

b7	b6	b5	b4	b3	b2	b1	b0	
N	V	m	x	D	I	Z	C	PSL

Description : Sets the processor status flags specified by the bit pattern in the second byte of the instruction to 1.

Status flags : The specified status flags are set to “1”. IPL is not affected.

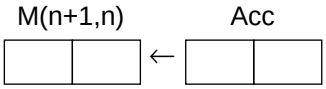
Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	SEP #imm	$E2_{16}$, imm	2	3

EOL announced

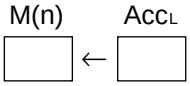
Function : Store

Operation : M ← Acc

When m=0



When m=1



Description : Stores the contents of the accumulator in memory.
The contents of the accumulator are not changed.

Status flags : Not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct	STA A, dd	85_{16}, dd	2	4
Direct indexed X	STA A, dd, X	95_{16}, dd	2	5
Direct indirect	STA A, (dd)	92_{16}, dd	2	7
Direct indexed X indirect	STA A, (dd, X)	81_{16}, dd	2	7
Direct indirect indexed Y	STA A, (dd), Y	91_{16}, dd	2	7
Direct indirect long	STAL A, (dd)	87_{16}, dd	2	10
Direct indirect long indexed Y	STAL A, (dd), Y	97_{16}, dd	2	11
Absolute	STA A, mmll	$8D_{16}$, ll, mm	3	5
Absolute indexed X	STA A, mmll, X	$9D_{16}$, ll, mm	3	5
Absolute indexed Y	STA A, mmll, Y	99_{16}, ll, mm	3	5
Absolute long	STA A, hhmmll	$8F_{16}$, ll, mm, hh	4	6
Absolute long indexed X	STA A, hhmmll, X	$9F_{16}$, ll, mm, hh	4	7
Stack pointer relative	STA A, nn, S	83_{16}, nn	2	5
Stack pointer relative indirect indexed Y	STA A, (nn, S), Y	93_{16}, nn	2	8

(Note 1) This table applies when using the accumulator A. If using the accumulator B, replace “A” with “B”. In this case, “42_{16}” is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

Function : Oscillation control

Operation : Stop the oscillation

Description : Resets the oscillator controlling flip-flop circuit to inhibit the oscillation of the oscillation circuit. To restart the oscillator, either an interrupt or reset must be executed.

Status flags : Not affected.

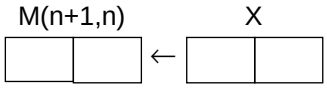
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	STP	DB_{16}	1	3

EOL announced

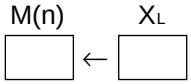
Function : Store

Operation : M ← X

When x=0



When x=1



Description : Stores the contents of the index register X in memory. The contents of the index register X remain the same.

Status flags : Not affected.

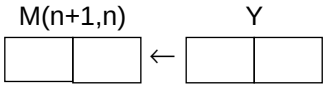
Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct	STX dd	86_{16}, dd	2	4
Direct indexed Y	STX dd, Y	96_{16}, dd	2	5
Absolute	STX mmll	$8E_{16}$, ll, mm	3	5

EOL announced

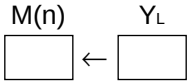
Function : Store

Operation : M ← Y

When x=0



When x=1



Description : Stores the contents of the index register Y in memory. The contents of the index register Y remain the same.

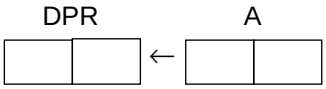
Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct	STY dd	84_{16}, dd	2	4
Direct indexed X	STY dd, X	94_{16}, dd	2	5
Absolute	STY mmll	$8C_{16}$, ll, mm	3	5

EOL announced

Function : Transfer

Operation : DPR ← A



Description : Loads the direct page register DPR with the contents of the accumulator A. Data is transferred as 16-bit data regardless of the status of the data length selection flag m. The contents of the accumulator A are not changed.

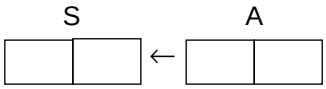
Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TAD	$5B_{16}$	1	2

EOL announced

Function : Transfer

Operation : S ← A



Description : Loads the stack pointer S with the contents of the accumulator A. Data is transferred as 16-bit data regardless of the status of the data length selection flag m. The contents of the accumulator A are not changed.

Status flags : Not affected.

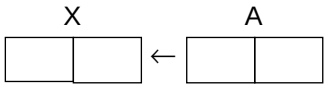
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TAS	$1B_{16}$	1	2

EOL announced

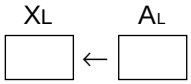
Function : Transfer

Operation : X ← A

When x=0



When x=1



Description : Loads the index register X with the contents of the accumulator A. The contents of the accumulator A are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

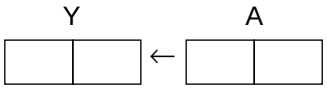
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TAX	AA_{16}	1	2

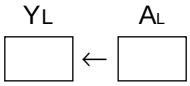
Function : Transfer

Operation : Y ← A

When x=0



When x=1



Description : Loads the index register Y with the contents of the accumulator A. The contents of the accumulator A are not changed.

Status flags

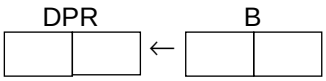
IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TAY	$A8_{16}$	1	2

EOL announced

Function : Transfer

Operation : DPR ← B



Description : Loads the direct page register DPR with the contents of the accumulator B. Data is transferred as 16-bit data regardless of the status of the data length selection flag m. The contents of the accumulator B are not changed.

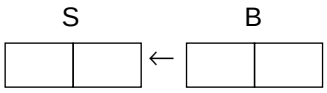
Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TBD	42_{16}, $5B_{16}$	2	4

EOL announced

Function : Transfer

Operation : S ← B



Description : Loads the stack pointer S with the contents of the accumulator B. Data is transferred as 16-bit data regardless of the status of the data length selection flag m. The contents of the accumulator B are not changed.

Status flags : Not affected.

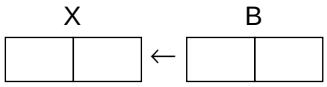
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TBS	42_{16}, $1B_{16}$	2	4

EOL announced

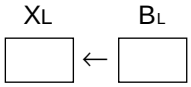
Function : Transfer

Operation : X ← B

When x=0



When x=1



Description : Loads the index register X with the contents of the accumulator B. The contents of the accumulator B are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

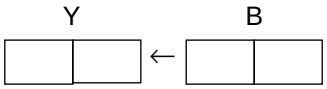
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TBX	42_{16}, AA_{16}	2	4

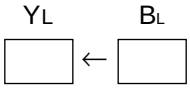
Function : Transfer

Operation : Y ← B

When x=0



When x=1



Description : Loads the index register Y with the contents of the accumulator B. The contents of the accumulator B are not changed.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

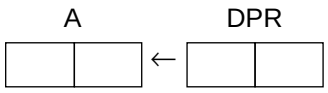
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TBY	42_{16}, $A8_{16}$	2	4

EOL announced

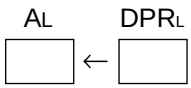
Function : Transfer

Operation : A ← DPR

When m=0



When m=1



Description : Loads the accumulator A with the contents of the direct page register DPR. The contents of the direct page register DPR are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

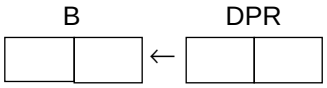
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TDA	$7B_{16}$	1	2

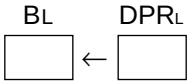
Function : Transfer

Operation : B ← DPR

When m=0



When m=1



Description : Loads the accumulator B with the contents of the direct page register DPR. The contents of the direct page register DPR are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

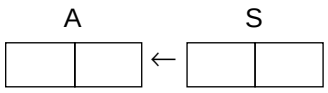
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TDB	42_{16}, $7B_{16}$	2	4

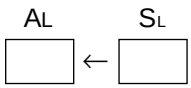
Function : Transfer

Operation : A ← S

When m=0



When m=1



Description : Loads the accumulator A with the contents of the stack pointer S. The contents of the stack pointer S are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

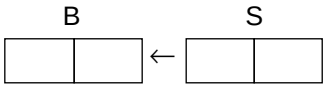
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TSA	$3B_{16}$	1	2

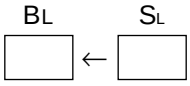
Function : Transfer

Operation : B ← S

When m=0



When m=1



Description : Loads the accumulator B with the contents of the stack pointer S. The contents of the stack pointer S are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

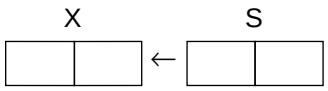
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TSB	42_{16}, $3B_{16}$	2	4

EOL announced

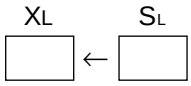
Function : Transfer

Operation : X ← S

When x=0



When x=1



Description : Loads the index register X with the contents of the stack pointer S. The contents of the stack pointer S are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

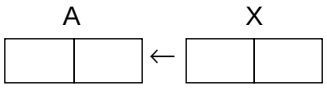
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TSX	BA_{16}	1	2

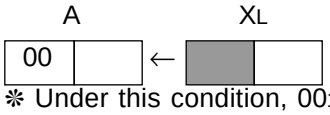
Function : Transfer

Operation : A ← X

When m=0 and x=0

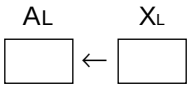


When m=0 and x=1



❊ Under this condition, 00_{16} is transferred to A_H regardless of X_H.

When m=1



Description : Loads the accumulator A with the contents of the index register X. The contents of the index register X are not changed.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

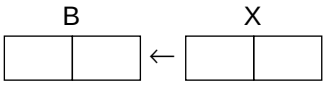
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TXA	$8A_{16}$	1	2

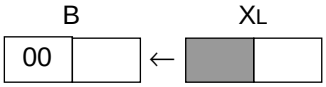
Function : Transfer

Operation : B ← X

When m=0 and x=0

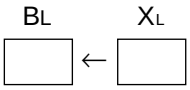


When m=0 and x=1



❊ Under this condition, 00_{16} is transferred to A_H regardless of X_H.

When m=1



Description : Loads the accumulator B with the contents of the index register X. The contents of the index register X are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

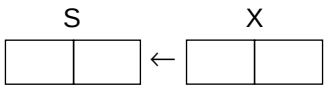
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TXB	42_{16}, $8A_{16}$	2	4

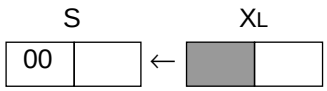
Function : Transfer

Operation : S ← X

When x=0



When x=1



Description : Loads the stack pointers with the contents of the index register X. The contents of the index register X are not changed.

Status flags : Not affected.

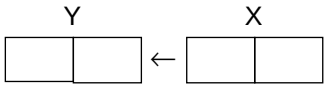
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TXS	$9A_{16}$	1	2

EOL announced

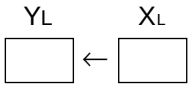
Function : Transfer

Operation : Y ← X

When x=0



When x=1



Description : Loads the index register Y with the contents of the index register X. The contents of the index register X are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

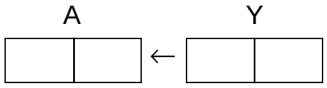
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TXY	$9B_{16}$	1	2

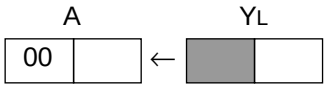
Function : Transfer

Operation : A ← Y

When m=0 and x=0

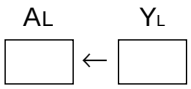


When m=0 and x=1



❊ Under this condition, 00_{16} is transferred to A_H regardless of Y_H.

When m=1



Description : Loads the accumulator A with the contents of the index register Y. The contents of the index register Y are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

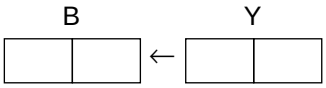
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TYA	98_{16}	1	2

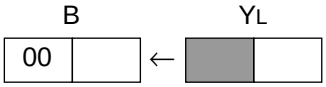
Function : Transfer

Operation : B ← Y

When m=0 and x=0

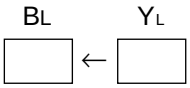


When m=0 and x=1



❊ Under this condition, 00_{16} is transferred to B_H regardless of Y_H.

When m=1



Description : Loads the accumulator B with the contents of the index register Y. The contents of the index register Y are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

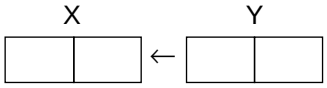
EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TYB	42_{16}, 98_{16}	2	4

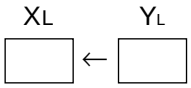
Function : Transfer

Operation : X ← Y

When x=0



When x=1



Description : Loads the index register X with the contents of the index register Y. The contents of the index register Y are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TYX	BB_{16}	1	2

Function : Clock control

Operation : Stop the CPU clock

Description : The WIT instruction stops the internal clock but not the oscillation of the oscillation circuit is not stopped. To restart the internal clock, either an interrupt or reset must be executed.

Status flags : Not affected.

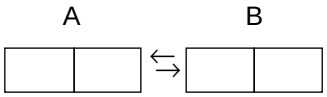
Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	WIT	CB_{16}	1	3

EOL announced

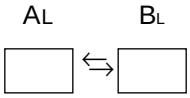
Function : Exchange

Operation : A ⇆ B

When m=0



When m=1



Description : Swaps the contents of the accumulators A and B.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the accumulator A after the operation is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the contents of the accumulator A is cleared to 0 by the operation. Otherwise, cleared to 0.

C : Not affected.

EOL announced

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	XAB	89_{16}, 28_{16}	2	6

4.3 Notes for Programming

Take care of the following when programming with the 7700 Family.

(1) The stack pointer S is undefined immediately after the reset is commanded. Always set the initial value.

Example)
```
LDX   #27FH
TXS
```

(2) The program bank register PG and the data bank register DT are disabled under the single chip mode. Do not set value other than "00_{16}" here.

(3) When "1" is set in the D-flag for decimal operation:

The C-flag alone is effective in the ADC instruction, while the Z, N, and V flags are disabled. The C and Z flags alone are effective in the SBC instruction, while the N and V flags are disabled. (Decimal operation can be done in the ADC and the SBC instructions alone.)

(4) Using the 16-bit immediate data with "1" (data length : 8 bits) in the data length selection flag m, or using the 8- bit immediate data with "0" (data length : 16 bits) in flag m, will cause the program run-away. The same rule is applied to the index register length selection flag x. Take care of the condition of these flags when coding the program.

(5) The 7700 Family can prefetch the instructions using the 3-byte instruction queue buffer. Keep in mind when creating the timer with the software, that the number of cycles shown in the list of machine language instructions is the minimum value. (Also see Chapter 5.)

(6) When value other than "00_{16}" is set in the lower order 8 bits of the direct page register DPR (DPR_L), the processing time will become 1 machine cycle longer than when "00_{16}" is set.

(7) The processing speed will deteriorate if a 16- bit data will be accessed from an odd address. Place the 16-bit data from an even address if the processing speed is important.

(8) The N and Z flags will change by execution of the PLA instruction, but the contents of the processor status register will not change if the accumulator A alone is recovered by the PUL instruction.

(9) The program bank register PG can be saved into the stack by setting "1" in bit 6 of the operation by the PSH instruction. However, the PG cannot be recovered by the PUL instruction.

(10) The code in the second byte of the BRK instruction will not affect the CPU.

EOL announced

MEMORANDUM

EOL announced

CHAPTER 5

INSTRUCTION EXECUTION SEQUENCE

5.1 Change of the CPU basic clock ϕCPU

5.2 Instruction execution sequence

EOL announced

INSTRUCTION EXECUTION SEQUENCE

5.1 Change of the CPU basic clock ϕ_{CPU}

The basic clock of the 7700 Family central processing unit (CPU) is the internal clock ϕ (divided by 2 of the oscillation frequency f(X_{IN})). The basic clock of the bus is an $\overline{E}$ derived from the internal clock ϕ, so data exchange between the CPU and the internal bus is done via the bus interface unit (BIU). The frequency of $\overline{E}$ is normally divided by 2 of the internal clock ϕ, but it becomes divided by 3 or 4 of ϕ, when accessing external memory while the wait is enabled by the wait bit (Note).

Note : The frequency of $\overline{E}$ is depend on the product. Therefore, refer to an user's manual or a data sheet (or a data book) to confirm it.

5.1 Change of the CPU Basic Clock ϕ_{CPU}

When the bus interface unit is not ready, the CPU extends the basic clock to synchronize with the bus, and waits till it is ready. As the CPU basic clock waits owing to some conditions, this clock will be called ϕ_{CPU} to be distinguished from the clock . The following are the cases in which the ϕ_{CPU} waits.

Causes for the ϕ_{CPU} to wait

<Cause 1>

When the CPU requests operation codes and operands, but the operation codes and operands in the instruction queue buffer did not reach the necessary number.

<Cause 2>

When the CPU tried to access data, but the bus interface unit was using the bus for fetching some data into the instruction queue buffer or writing data.

<Cause 3>

When the bus interface unit was reading data from the internal/external memory or I/O, according to the request of the CPU.

In addition to the above, the following are also causes for the ϕCPU to be extended.

- When 16-bit data is accessed from odd address.
- When external memory 16-bit data is accessed while the BYTE terminal level is “H”.
- When external memory is accessed with wait commanded by the wait bit.

EOL announced

The above conditions causes the execution time to differ each time, even with the same instruction and same addressing mode. Two example instructions are given in the next section to see the variation of the number of cycles according to the above conditions.

The “CPU execution sequence per addressing mode” of Chapter 6 is the CPU instruction execution sequences based on the ϕ_{CPU}. The number of cycles shown in “4.2 Instructions” and “Appendix 1 List of machine instructions” are the count for the shortest case, and cannot always be applied when calculating the actual cycles or the execution time of instructions.

5.2 Instruction Execution Sequence

This section describes how the instruction execution cycles change under various conditions.

- Example 1. ASL instruction Direct addressing mode
- Example 2. LDA instruction Direct indirect long addressing mode

Before observing the ϕ_{CPU} based CPU instruction execution sequence

The following table describes the ϕ_{CPU} based CPU instruction execution sequence symbols. The signals indicated in this execution sequence are all CPU internal signals, that show data exchange between the bus interface unit and the CPU. Accordingly, these signals cannot be observed from outside.

ϕ_{CPU} Based CPU Instruction Execution Sequence Symbols

Symbol	Description
ϕ_{CPU}	CPU basic clock
$A_{P(CPU)}$	Higher order 8 bits of the address (24 bits) of the program that the CPU is actually execution
$A_HA_{L(CPU)}$	Lower order 16 bits of the address (24 bits) of the program that the CPU is actually execution
$DATA_{(CPU)}$	Data information the CPU is processing
$R/\overline{W}_{(CPU)}$	Data read/write request to the data buffer in the bus interface unit
PG,PC	Contents of the program bank register (PG) and the program counter (PC)
AD_P	Data indicating the address (higher order 8 bits)
AD_H,AD_L	Data indicating the address (lower order 16 bits)
DPR_H	Contents of the higher order 8 bits of the direct page register
DPR_L	Contents of the lower order 8 bits of the direct page register (DPR_L = 0 in the examples)
D_H	Data to be fetched or written from the data buffer by the CPU (higher order 8 bits)
D_L	Data to be fetched or written from the data buffer by the CPU (lower order 8 bits)
dd	Contents of the operand (DPR_L = 0 in examples 1 and 2, so dd represents the lower order 8 bits of the address)

EOL announced

5.2 Instruction execution sequence

Before observing the ϕ based instruction execution sequence

The ϕ based instruction execution sequence symbols are shown in the following table. The signals in this execution sequence indicates data exchange of the bus interface unit with the memory and I/O. The internal instruction execution sequence of the CPU can be guessed from these signals.
However, the ϕ_{CPU} and the number of data in the instruction queue buffer shown here cannot be observed from the outside.

ϕ Based Execution Sequence Symbols

Symbol	Description
ϕ	Basic operation clock of the microcomputer (divided by 2 of f(XIN))
$\bar{E}$	Basic operation clock of the bus (divided by 2 of ϕ) ❊ No wait
hh	Higher order 8 bits of the address where the bus interface unit is to access to (bank)
mm	Middle order 8 bits of the address where the bus interface unit is to access to
ll	Lower order 8 bits of the address where the bus interface unit is to access to
DPR	Contents of the direct page
DPRH	Contents of the higher order 8 bits of the direct page register
DPRL	Contents of the lower order 8 bits of the direct page register
OP1	Data to be fetched into the instruction queue buffer by the bus interface
OP2	(Operation code or operand)
OP3	The subscript represents the fetch sequence.
:	
DL	Data to be fetched into the data buffer or data to be written into the memory by the bus interface unit
DH	
dd	Data obtained as the operand (The lower order 8 bits of the address are given in examples 1 and 2, because DPRL = 0.)
ADP	Higher order 8 bits of data that indicates the address (contents of the data address register)
ADH	Middle order 8 bits of data that indicates the address (contents of the data address register)
ADL	Lower order 8 bits of data that indicates the address (contents of the data address register)

EOL announced

The following are the cause of the "ϕ_{CPU} to queue" in the ϕ based instruction execution sequence.

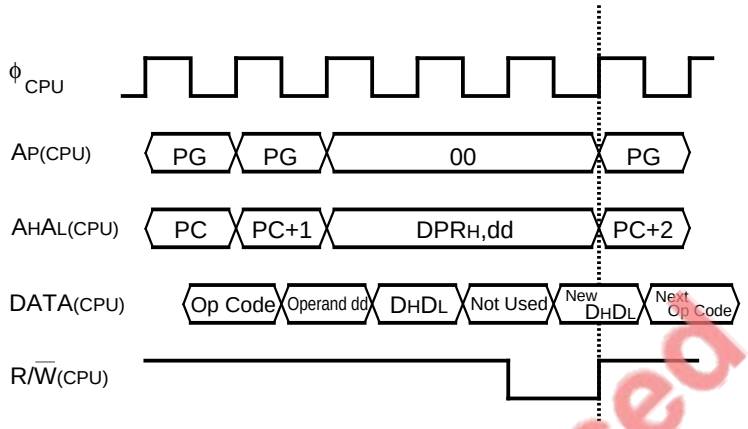
Cause 1
When the CPU required operation codes and operands, but the number of operation codes and operands did not reach the requested number.

Cause 2
When the CPU tried to access data, but the bus interface was using the bus for fetching data into the instruction queue buffer or for writing data.

Cause 3
When the bus interface unit is reading data from the internal/external memory or I/O, etc., according to the request of the CPU.

Example 1. ASL instruction / direct addressing mode ($DPR_L = 00_{16}$)

ϕ_{CPU} based CPU instruction execution sequence



The following examples 1-1 to 1-6 are examples of the ϕ_{CPU} based instruction execution sequences under various conditions.

Example 1-1 When the instruction queue buffer is vacant

Example 1-2 When two data are in the instruction queue buffer

Example 1-3 When three data are in the instruction queue buffer

Example 1-4 When 16-bit data is accessed from odd address

Example 1-5 When external memory is accessed from the BYTE terminal using 8-bit external bus width

Example 1-6 When external memory is accessed with wait by the wait bit

EOL announced

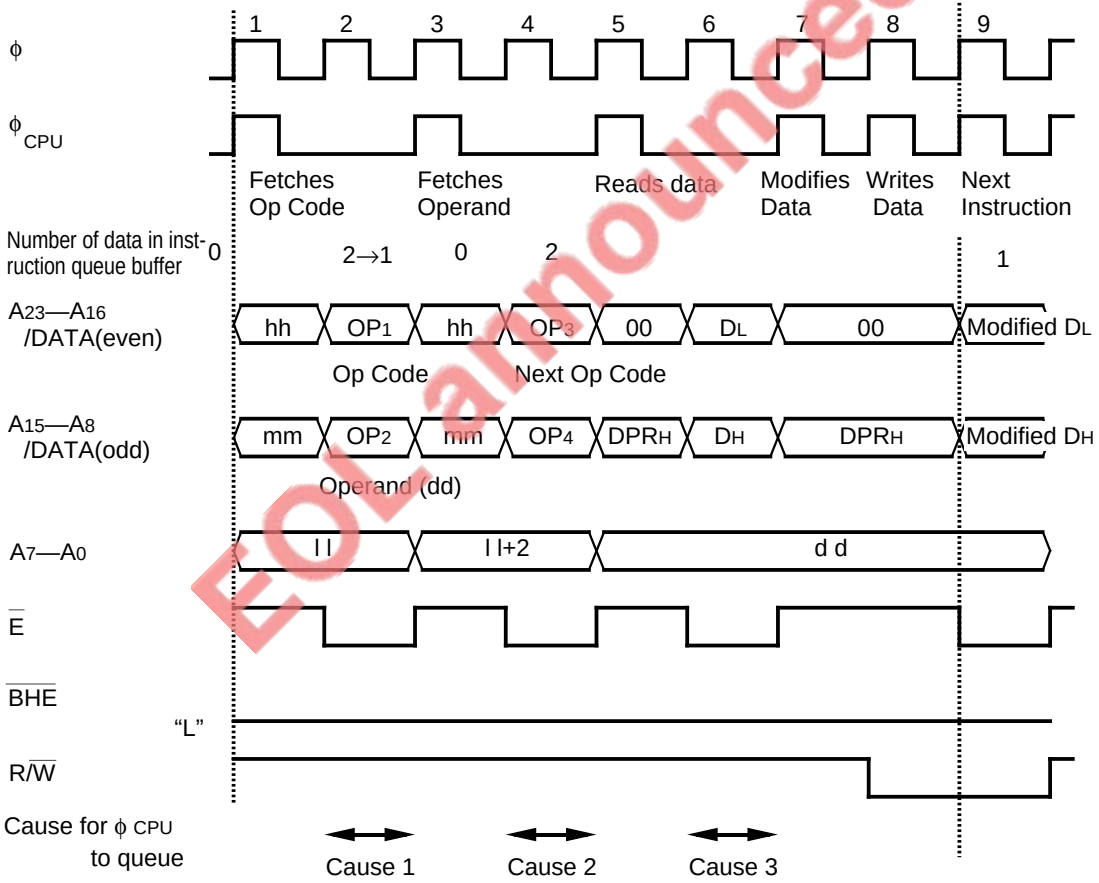
Example 1. ASL instruction / direct addressing mode (DPR_L = 00_{16})

(Example 1-1) When the instruction queue buffer is vacant

Conditions

- Number of data in the instruction queue buffer 0
- ROM, RAM ..... External memory is used (Note)
- Data length selection flag m ..... "0" (16-bit length)
- BYTE terminal level ..... "L" (External bus width is 16 bits)
- Contents of lower order bytes (PC_L) of the program counter ..... Even
- Contents of the operand (dd) ..... Even

ϕ based execution sequence



EOL announced

Note. The operation when internal ROM and internal RAM are used, will be as shown above, regardless of the level of the BYTE terminal. However, the address/data bus, $\overline{BHE}$, R/$\overline{W}$ signal cannot be observed from outside, when the mode is single-chip mode.

Example 1. ASL instruction / direct addressing mode ($DPR_L = 00_{16}$)

Operation of the CPU and bus interface unit under various cycles

ϕ No.	CPU	Bus interface unit
1	(No fetching can be done, because there are no operation codes in the instruction queue buffer.)	Fetches the instruction, because instruction queue buffer is vacant and the CPU is not using the bus.
2	Fetches the operation code.	Fetches 2-byte worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
3	Fetches the operand.	Prefetches the instruction, because the instruction queue buffer is vacant and the CPU is not using the bus.
4	(Waits till the bus used by the bus interface unit becomes vacant.)	Fetches 2 bytes worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
5	Waits for $\overline{E}$ to become "L", to read data.	
6	Reads data when $\overline{E}$ becomes "L".	
7	Modifies data.	
8	Writes data into the data buffer.	
9	Fetches the next operation code.	Writes the contents of the data buffer into the original address, when $\overline{E}$ becomes "L".

EOL announced

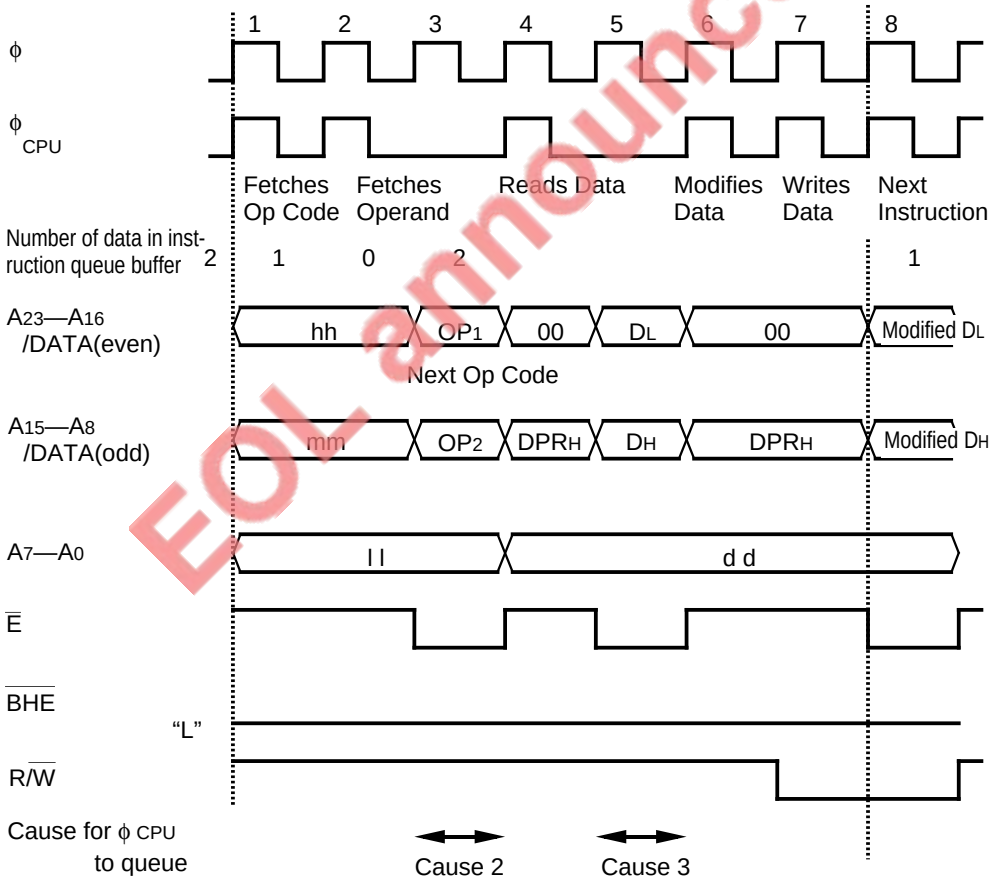
Example 1. ASL instruction / direct addressing mode (DPR_L = 00_{16})

(Example 1-2) When two data are in the instruction queue buffer

Conditions

- Number of data in the instruction queue buffer 2
- ROM, RAM ..... External memory is used (Note)
- Data length selection flag m ..... "0" (16-bit length)
- BYTE terminal level ..... "L" (External bus width is 16 bits)
- Contents of lower order bytes (PC_L) of the program counter ..... Even
- Contents of the operand (dd) ..... Even

ϕ based execution sequence



EOL announced

Note. The operation when internal ROM and internal RAM are used, will be as shown above, regardless of the level of the BYTE terminal. However, the address/data bus, $\overline{BHE}$, R/$\overline{W}$ signal cannot be observed from outside, when the mode is single chip mode.

Example 1. ASL instruction / direct addressing mode ($DPR_L = 00_{16}$)

Operation of the CPU and bus interface unit under various cycles

ϕ No.	CPU	Bus interface unit
1	Fetches operation code.	
2	Fetches operand (dd).	Prefetches the instruction, because the instruct queue buffer is vacant and the CPU is not using the bus.
3	(Waits till the bus used by the bus interface unit becomes vacant.)	Fetches 2-byte worth of data into the instruction queue buffer when $\overline{E}$ becomes “L”.
4	Waits for $\overline{E}$ to become “L”, to read data.	
5	Reads data when $\overline{E}$ becomes “L”.	
6	Modifies data.	
7	Writes data into the data buffer.	
8	Fetches the next operation code.	Writes the contents of the data buffer into the original address, when $\overline{E}$ becomes “L”.

EOL announced

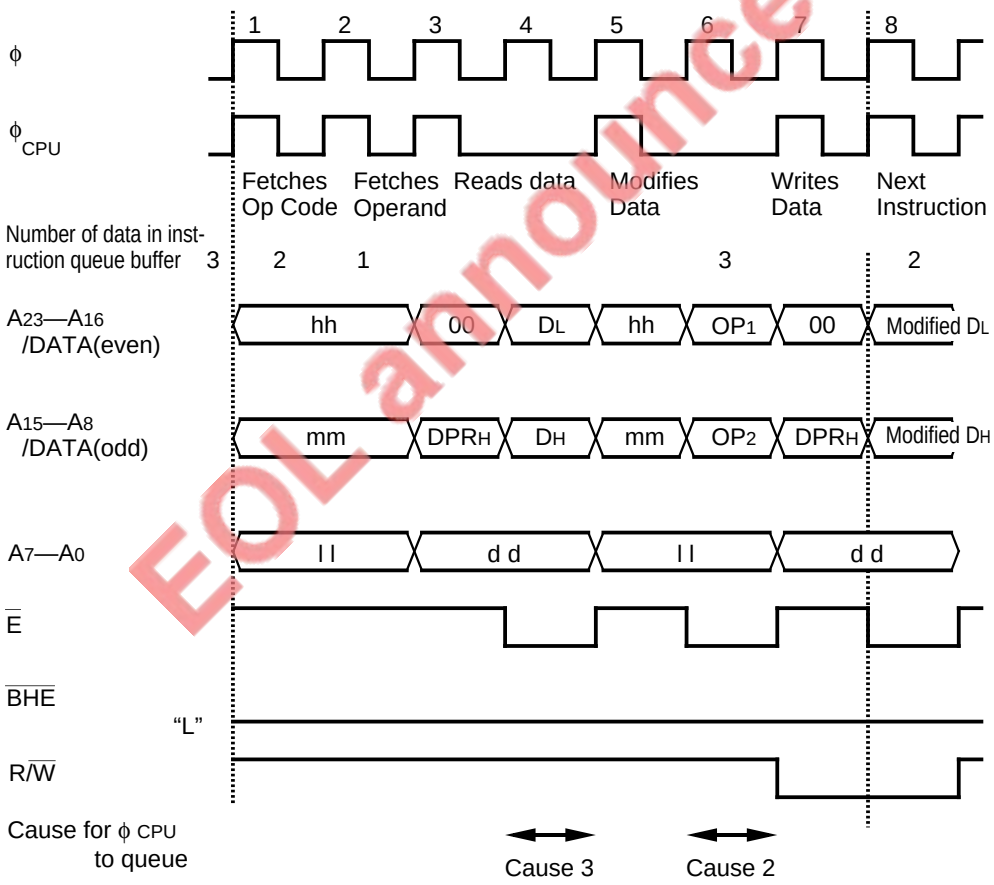
Example 1. ASL instruction / direct addressing mode (DPR_L = 00_{16})

(Example 1-3) When three data are in the instruction queue buffer

Conditions

- Number of data in the instruction queue buffer 3
- ROM, RAM ..... External memory is used (Note)
- Data length selection flag m ..... "0" (16-bit length)
- BYTE terminal level ..... "L" (External bus width is 16 bits)
- Contents of lower order bytes (PC_L) of the program counter ..... Even
- Contents of the operand (dd) ..... Even

ϕ based execution sequence



EOL announced

Note. The operation when internal ROM and internal RAM are used, will be as shown above, regardless of the level of the BYTE terminal. However, the address/data bus, $\overline{BHE}$, R/$\overline{W}$ signal cannot be observed from outside, when the mode is single chip mode.

Example 1. ASL instruction / direct addressing mode ($DPR_L = 00_{16}$)

Operation of the CPU and bus interface unit under various cycles

ϕ No.	CPU	Bus interface unit
1	Fetches operation code .	
2	Fetches operand (dd).	
3	Waits for $\overline{E}$ to become "L", to read data.	
4	Reads data when $\overline{E}$ becomes "L".	
5	Modifies data.	Prefetches the instruction, because there are two vacant instruction queue buffers and the CPU is not using the bus.
6	(Waits till the bus used by the bus interface unit becomes vacant.)	Fetches 2-byte worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
7	Writes data into the data buffer.	
8	Fetches the next operation code.	Writes the contents of the data buffer into the original address, as $\overline{E}$ becomes "L".

EOL announced

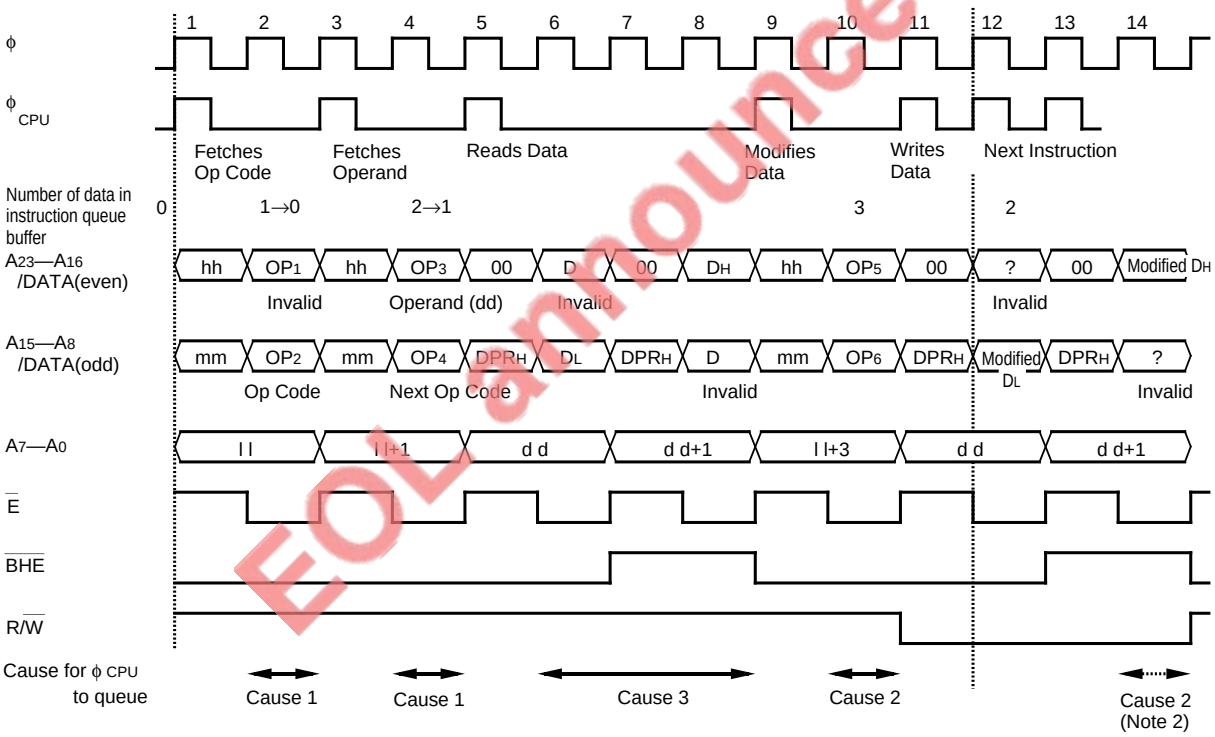
Example 1. ASL instruction / direct addressing mode (DPR_L = 00_{16})

(Example 1-4) When 16-bit data is accessed from odd address

Conditions

- Number of data in the instruction queue buffer 0
- ROM, RAM ..... External memory is used (Note 1)
- Data length selection flag m ..... "0" (16-bit length)
- BYTE terminal level ..... "L" (External bus width is 16 bits)
- Contents of lower order bytes (PC_L) of the program counter ..... Odd
- Contents of the operand (dd) ..... Odd

ϕ based execution sequence



EOL announced

Note 1. The operation when internal ROM and internal RAM are used, will be as shown above, regardless of the level of the BYTE terminal. However, the address/data bus, $\overline{BHE}$, R/$\overline{W}$ signal cannot be observed from outside, when the mode is single chip mode.

Note 2. At the ◄···► part

❊ When the CPU does not use the bus, ϕ_{CPU} corresponds with ϕ.

❊ When the CPU uses the bus, the ϕ_{CPU} queues till the writing in the bus interface unit completes. (the ϕ14 cycle)

Example 1. ASL instruction / direct addressing mode (DPR_L = 00_{16})

Operation of the CPU and bus interface unit under various cycles

ϕ No.	CPU	Bus interface unit
1	(No fetching can be done, because there are no operation codes in the instruction queue buffer.)	Fetches the instruction, because instruction queue buffer is vacant and the CPU is not using the bus.
2	Fetches operation code.	Fetches 1 odd address byte worth of data into the instruction queue buffer, when $\overline{E}$ becomes "L".
3	(No fetching can be done, because there are no operands in the instruction queue buffer.)	Fetches the instruction, because instruction queue buffer is vacant and the CPU is not using the bus.
4	Fetches operand (dd).	Fetches 2-byte worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
5	Waits for $\overline{E}$ to become "L", to read data.	
6	Reads data in the odd addresses (D_L) alone into the data buffer when $\overline{E}$ becomes "L".	
7	Waits for $\overline{E}$ to become "L", to read data.	
8	Reads data in the even addresses (D_H) alone into the data buffer when $\overline{E}$ becomes "L".	
9	Modifies data.	Prefetches the instruction, because there are two vacant positions in the instruction queue buffer, and the CPU is not using the bus.
10	(Waits till the bus used by the bus interface unit becomes vacant.)	Fetches 2 bytes worth of data into the instruction queue buffer, when $\overline{E}$ becomes "L".
11	Writes data into the data buffer.	Waits till $\overline{E}$ becomes "L" to write data.
12	Fetches the next operation code.	Writes the contents of the data buffer (D_L) into the original address (odd address), when $\overline{E}$ becomes "L".
13	?	Waits till $\overline{E}$ becomes "L" to write data.
14	?	Writes the contents of the data buffer (D_H) into the original address (even address), when $\overline{E}$ becomes "L".

EOL announced

When internal ROM or BYTE terminal level "L" external memory is used as the program memory, the instruction is fetched into the instruction queue buffer normally in 2-byte (word) unit of sequential even and odd addresses in this order. However, when the instruction must be fetched from odd address like after execution of the JMP instruction, the 1-byte of the first odd address alone is fetched into the instruction queue buffer (ϕ2 cycle), and the later instructions are fetched into the instruction queue buffer in 2-byte units (ϕ4, ϕ10 cycle).

The bus interface unit automatically selects whether to fetch one word or to fetch the 1 byte of odd address alone. The operation status can be observed from outside, according to the output of the $\overline{BHE}$ terminal and the address bus signal A_0, as long as the mode is not single-chip mode.

- When one word is fetched
  The output from both the $\overline{BHE}$ terminal and the address bus A_0 are at the "L" level.
- When 1 byte of odd address alone is fetched
  The output from the $\overline{BHE}$ terminal is "L", while the output from address bus A_0 is "H".

When internal RAM and external memory at BYTE terminal level "L" are used as the data memory, with data length selection flag m=0, both data read and write are normally done in 2-byte units of even and odd addresses, in this sequence. However, access can also be done when the word data is defined from an odd address. In other words, "H" is output first from address bus A_0 and then "L" from the $\overline{BHE}$ terminal to access to odd address alone. Next, "L" is output from A_0, and "H" from the $\overline{BHE}$ terminal to access to the even address (ϕ5 to ϕ8, ϕ11 to ϕ14 cycle).

Example 1. ASL instruction / direct addressing mode (DPRL = 00$_{16}$)

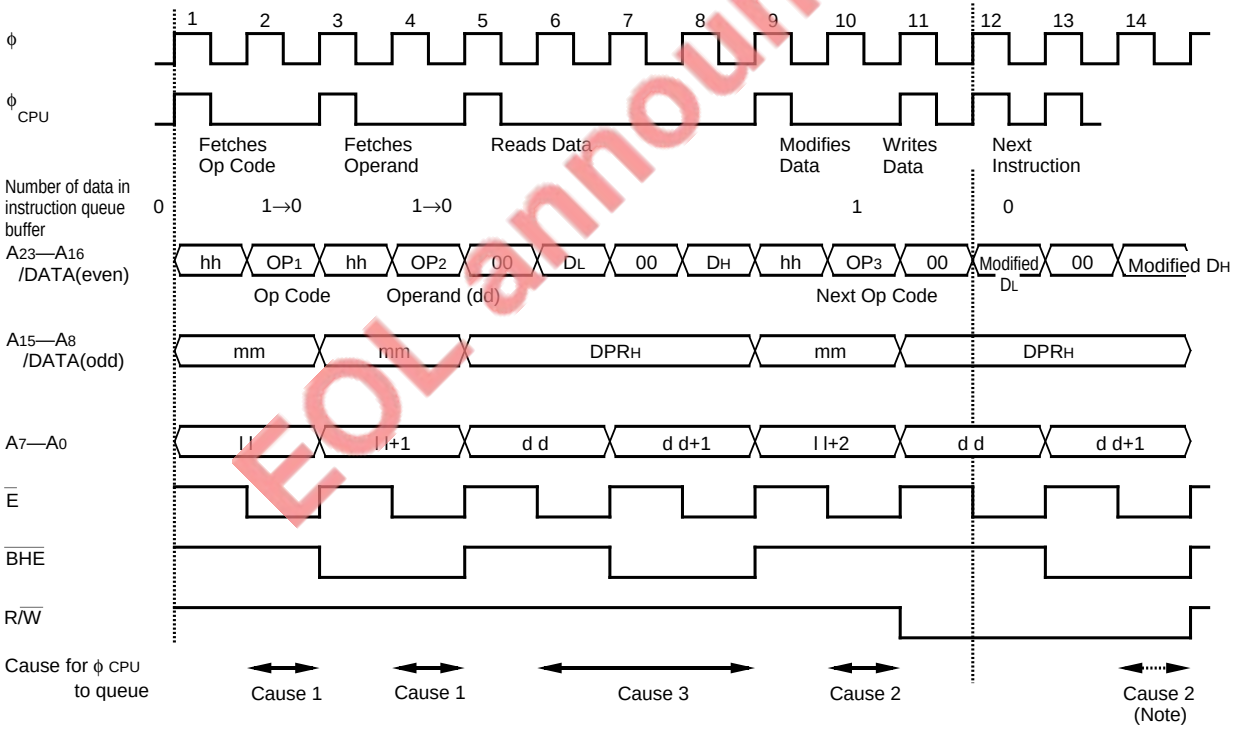
(Example 1-5) When external memory is accessed from the BYTE terminal using 8-bit external bus width

Conditions

- Number of data in the instruction queue buffer 0
- ROM, RAM ..... External memory is used
- Data length selection flag m ..... “0” (16-bit length)
- BYTE terminal level ..... “H” (External bus width is 8 bits)

EOL announced

ϕ based execution sequence



Note. At the ◄···► part
❊ When the CPU does not use the bus, ϕCPU corresponds with ϕ.
❊ When the CPU uses the bus, the ϕCPU queues till the writing in the bus interface unit completes. (the ϕ13 to ϕ14 cycle)

Example 1. ASL instruction / direct addressing mode (DPR_L = 00_{16})

Operation of the CPU and bus interface unit under various cycles

ϕ No.	CPU	Bus interface unit
1	(No fetching can be done, because there are no operation codes in the instruction queue buffer.)	Fetches the instruction, because the instruction queue buffer is vacant and the CPU is not using the bus.
2	Fetches operation code. ←	Fetches 1 odd address byte worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
3	(No fetching can be done, because there are no operands in the instruction queue buffer.)	Fetches the instruction, because instruction queue buffer is vacant and the CPU is not using the bus.
4	Fetches operand (dd). ←	Fetches 1-byte worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
5	Waits for $\overline{E}$ to become "L", to read data.	
6	Reads data (D_L) into the data buffer when $\overline{E}$ becomes "L".	
7	Waits for $\overline{E}$ to become "L", to read data.	
8	Reads data (D_H) alone into the data buffer when $\overline{E}$ becomes "L".	
9	Modifies data.	Prefetches the instruction, because there are two vacant positions in the instruction queue buffer, and the CPU is not using the bus.
10	(Waits till the bus used by the bus interface unit is vacant.)	Fetches 1 byte worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
11	Writes data into the data buffer.	Waits till $\overline{E}$ becomes "L" to write data.
12	Fetches the next operation code.	Writes the contents of the data buffer (D_L) into the original address (odd address), when $\overline{E}$ becomes "L".
13	?	Waits till $\overline{E}$ becomes "L" to write data.
14	?	Writes the contents of the data buffer (D_H) into the original address (even address), when $\overline{E}$ becomes "L".

EOL announced

The external bus width becomes 8 bits when the "H" level is applied to the BYTE terminal. (The width of the internal bus is 16 bits, regardless of the level of the BYTE terminal.) When external ROM is used under this mode, the instruction can only be fetched byte by byte. (ϕ2, ϕ4, ϕ10 cycle) When external RAM is used, the data can likewise only be handled byte by byte. Accordingly, when data length selection flag m = 0 is selected, it takes time worth 2 cycles of the enable output $\overline{E}$ for data read and write. (ϕ5 to ϕ8, ϕ11 to ϕ14 cycle)

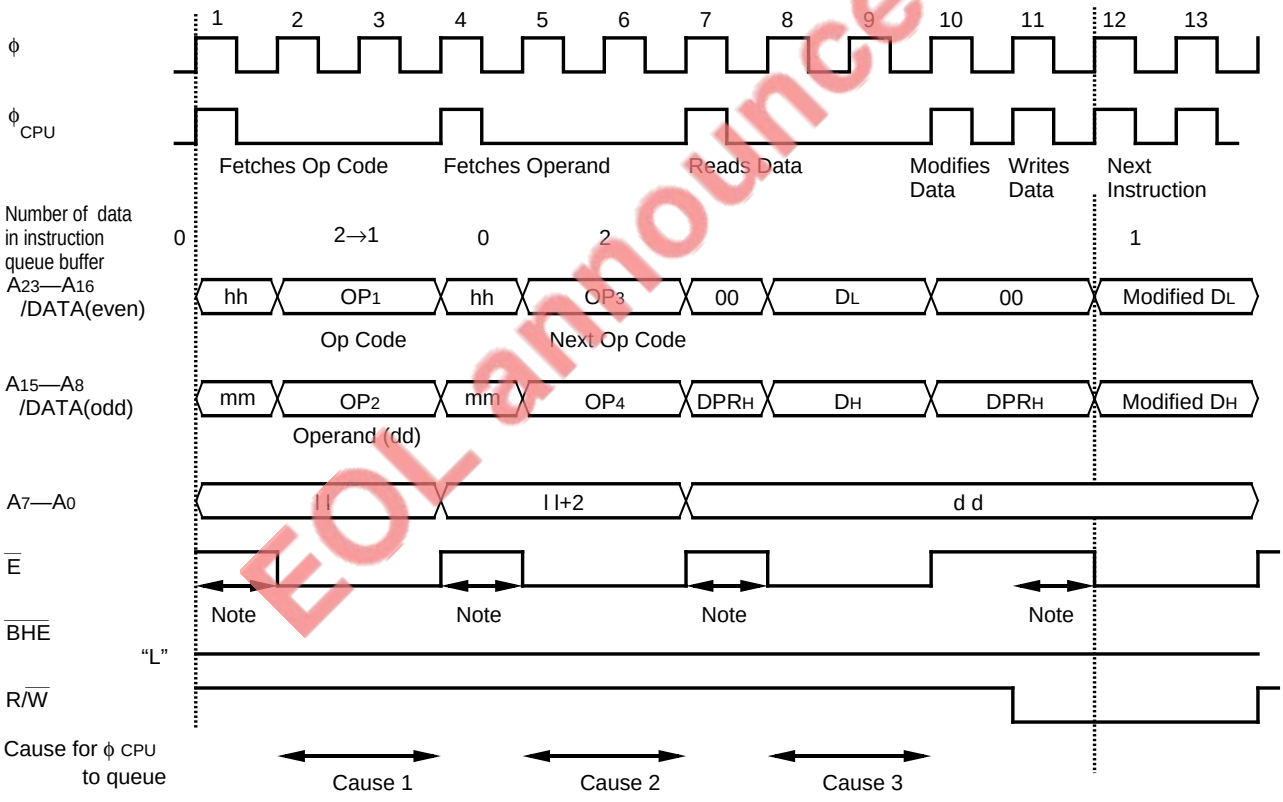
Example 1. ASL instruction / direct addressing mode (DPR_L = 00_{16})

(Example 1-6) When external memory is accessed with wait by the wait bit

Conditions

- Number of data in the instruction queue buffer 0
- ROM, RAM ..... External memory is used
- Data length selection flag m ..... “0” (16-bit length)
- BYTE terminal level ..... “L” (External bus width is 16 bits)
- Contents of lower order bytes (PCL) of the program counter ..... Even
- Contents of the operand (dd) ..... Even

ϕ based execution sequence



EOL announced

Note: This figure hows the case of the bus cycle becomes as 3 cycles of ϕ. If the bus cycle becomes 4 cycles of ϕ by wait, the “H” width of $\overline{E}$ ($\longleftrightarrow$) becomes as 2 cycles. Therefore, the ASL instruction execution time extended by 4 cycles from the above case.

Example 1. ASL instruction / direct addressing mode (DPR_L = 00_{16})

Operation of the CPU and bus interface unit under various cycles

ϕ No.	CPU	Bus interface unit
1	(No fetching can be done, because there are no operation codes in the instruction queue buffer.)	Fetches the instruction, because instruction queue buffer is vacant and the CPU is not using the bus.
2 3	Fetches the operation code ←	Fetches 2 bytes worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
4	Fetches operand (dd).	Prefetches the instruction because the instruction queue buffer is vacant and the CPU is not using the bus.
5 6	(Waits till the bus used by the bus interface unit becomes vacant.)	Fetches 2 bytes worth of data into the instruction queue buffer when becomes "L".
7	Waits till $\overline{E}$ becomes "L" to write data.	
8 9	Reads data when $\overline{E}$ becomes "L".	
10	Modifies data.	
11	Writes data into the data buffer.	
12	Fetches the next operation code.	
13	?	Writes the contents of the data buffer into the original address (odd address), when $\overline{E}$ becomes "L".

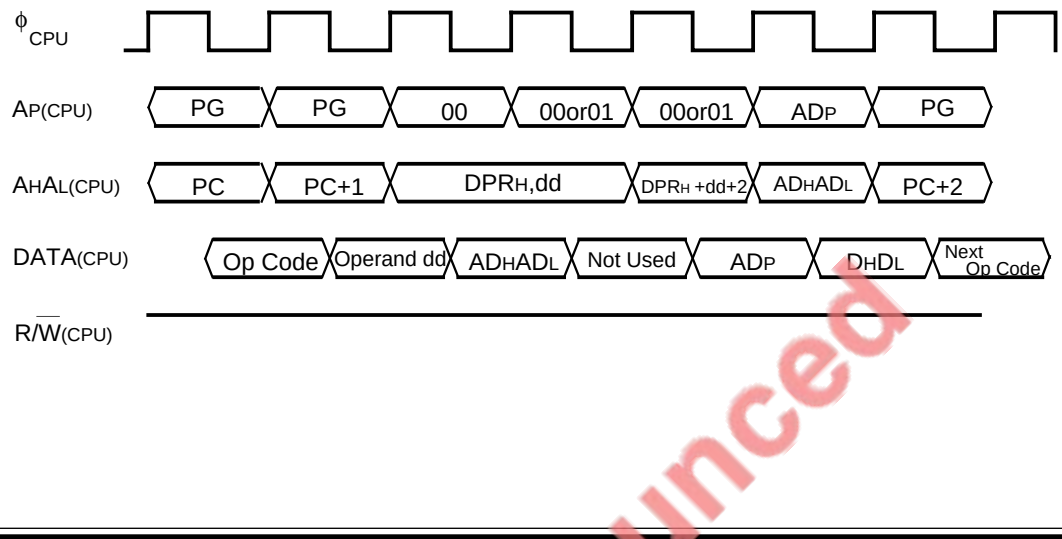
The conditions are the same as the Example 1-1, except when wait is commanded by the wait bit . When accessing to the external memory, the "L" width of enable output $\overline{E}$ is extended by 1 cycle of ϕ when compared to the case of no wait. Therefore, the ϕ_{CPU} wait interval is also extended by 1 cycle (ϕ2 to ϕ3, ϕ5 to ϕ6, ϕ8 to ϕ9 cycle).
If the bus cycle becomes 4 cycles, the ϕ_{CPU} wait interval is also extended by 2 cycles of ϕ because the "H" and "L" width extended by 1 cycle from the case no wait.

EOL announced

EOL announced

Example 2. LDA instruction / Direct indirect long addressing mode (DPR_L = 00_{16})

ϕ_{CPU} based CPU instruction execution sequence



EOL announced

Example 2. LDA instruction / Direct indirect long addressing mode (DPRL = 00_{16})

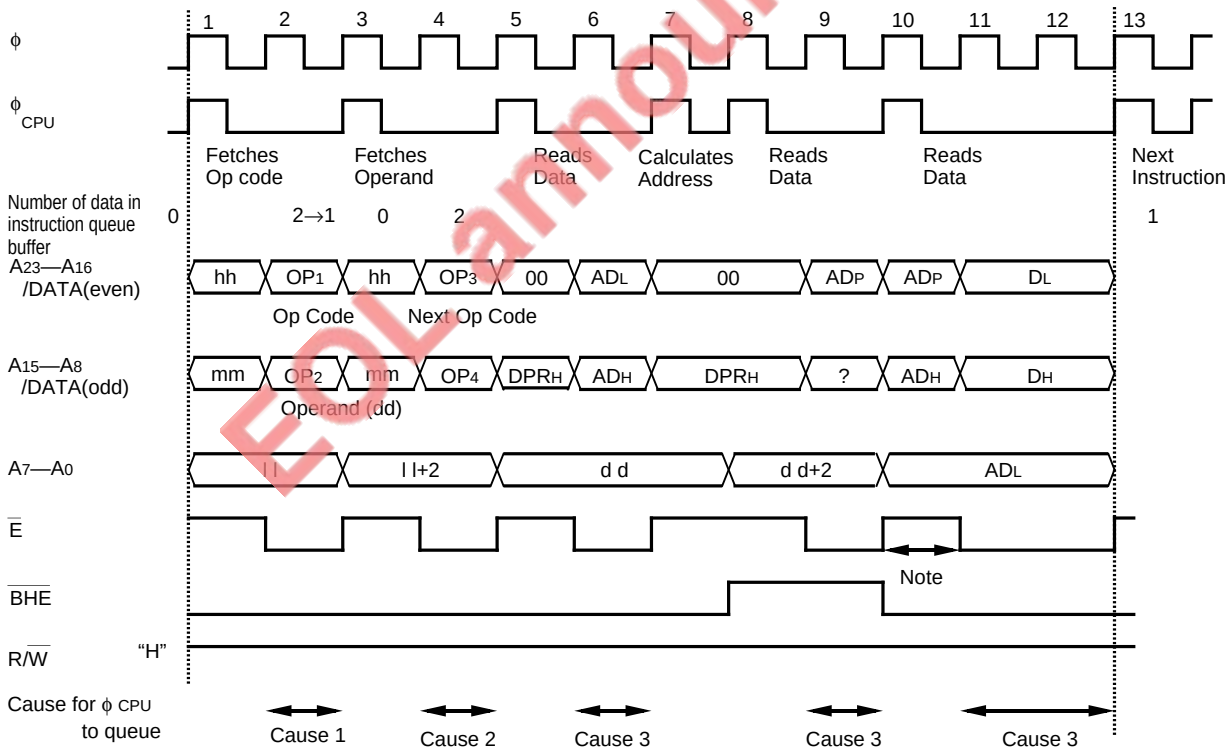
(Example 2-1) When the internal as well as the external memories are used together while wait is commanded by the wait bit.

Conditions

- Number of data in the instruction queue buffer 0
- Bank 0 ..... Internal ROM, RAM are used
  Bank 1 and after ..... External memory is used
- Data length selection flag m ..... "0" (16-bit length)
- BYTE terminal level ..... "L" (External bus width is 16 bits)
- Contents of lower order bytes (PC_L) of the program counter ..... Even
- Contents of the operand (dd) ..... Even
- Data indicated by the address AD_L ..... Even
  AD_P ..... 1 or more (bank 1 and after)

EOL announced

ϕ based execution sequence



Note: This figure hows the case of the bus cycle becomes as 3 cycles of ϕ. If the bus cycle becomes 4 cycles of ϕ by wait, the "H" width of $\overline{E}$ ($\longleftrightarrow$) becomes as 2 cycles.

Example 2. LDA instruction / Direct indirect long addressing mode (DPRL = 00_{16})

Operation of the CPU and bus interface unit under various cycles

ϕ No.	CPU	Bus interface unit
1	(No fetching can be done, because there are no operation codes in the instruction queue buffer.)	Fetches the instruction, because instruction queue buffer is vacant and the CPU is not using the bus.
2	Fetches the operation code .	Fetches 2 bytes worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
3	Fetches operand (dd).	Prefetches the instruction because the instruction queue buffer is vacant and the CPU is not using the bus.
4	(Waits till the bus used by the bus interface unit becomes vacant.)	Fetches 2 bytes worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
5	Waits for $\overline{E}$ to becomes "L", to read data (AD_H, AD_L) indicated by the address obtained by adding the contents of the operand (dd) and the DPR_L.	
6	Reads data when $\overline{E}$ becomes "L".	
7		Calculated address.
8	Waits for $\overline{E}$ to become "L", to read data (AD_P).	
9	Reads data when $\overline{E}$ becomes "L".	
10	Waits for $\overline{E}$ to become "L", to read the data (D_H, D_L) at the address specified by AD_P, AD_H, AD_L.	
11 12	Reads data when $\overline{E}$ becomes "L".	

The above is the case when bank 1 and after are used by the external memory under the memory expansion mode. The currently executed program is in bank 0. The contents of the lower order bytes of the direct page register DPR_L is "00_{16}", so the direct pages are all in bank 0. The access to the outside (ϕ10 to ϕ12 cycle) alone is affected by the wait bit, and access to the internal memory is not affected by the bit.

EOL-announced

CHAPTER 6

CPU INSTRUCTION EXECUTION SEQUENCE FOR EACH ADDRESSING MODE

EOL announced

CPU INSTRUCTION EXECUTION SEQUENCE FOR EACH ADDRESSING MODES

The following are the CPU instruction execution sequences for each addressing mode. The execution sequences shown here describe the internal operation of the CPU. Therefore, the signals are all CPU internal signals, and cannot be observed from outside. The CPU internal operation, the actual execution time, and the relation between signals that can be externally checked are described in Chapter 4 “Instruction Execution Sequence”.

The following are the signals and the symbols indicating the contents.

Symbol	Description
ϕ_{CPU}	CPU basic cycle
$AP_{(CPU)}$	Higher order 8 bits of the CPU internal address bus.
$AHAL_{(CPU)}$	Lower order 16 bits of the CPU internal address bus.
PG	Contents of the program bank register.
PC	Contents of the program counter.
	Others are data that indicates the address obtained as result of address calculation.
$DATA_{(CPU)}$	The CPU internal data bus. The signal is output with a half-cycle delay from the CPU internal address bus. The operation codes and the operands are fetched from the instruction buffer. They are not directly fetched from the memory indicated by the PG and PC of this cycle.
$R/\overline{W}_{(CPU)}$	Becomes “L” when the CPU writes data into the data buffer of the bus interface unit.

The accumulator used in the above instructions in the CPU instruction execution sequence is accumulator A. When accumulator B is used, the execution cycle will have the two cycles of a “42_{16}” that indicates accumulator B, and an internal processing cycle added at the front. (See the figure in the next page.)

The number of ϕ_{CPU} cycles differs in the addressing mode that uses the direct page register, according to whether the lower order 8 bits (DPR_L) are “00_{16}”. The number of cycles when $DPR_L = 00_{16}$ is 1 cycle (address calculation cycle) less than when $DPR_L \neq 00_{16}$.

The number of cycles differs in the PSH and PUL instructions according to the number and type of registers placed in (taken out of) the stack.

The number of cycles differs in the block transmission instruction (MVN, MVP), according to the number of the data transmitted.

Note. The instructions with the mark “ * ” can be used in the 7750 Series only.

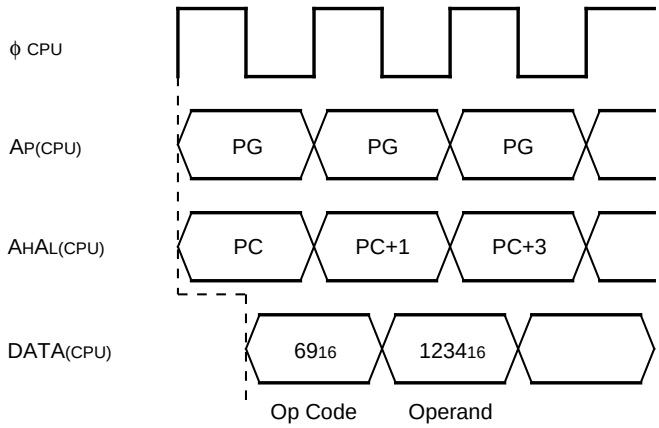
EOL announced

Variation of the execution cycles according to the accumulator used

ADC Instruction / Immediate addressing mode

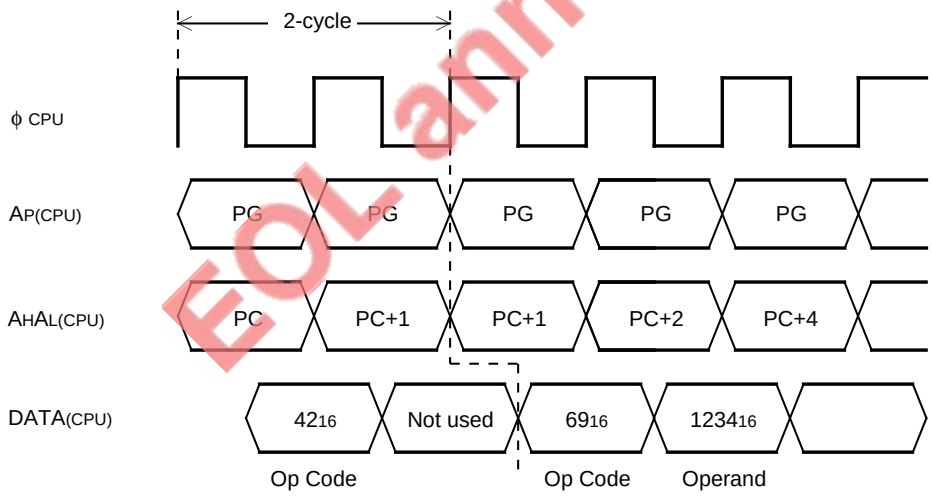
<<When accumulator A is used>>

Mnemonic : ADC A,#1234H Machine code : 69_{16} 34_{16} 12_{16}



<<When accumulator B is used>>

Mnemonic : ADC B,#1234H Machine code : 42_{16} 69_{16} 34_{16} 12_{16}

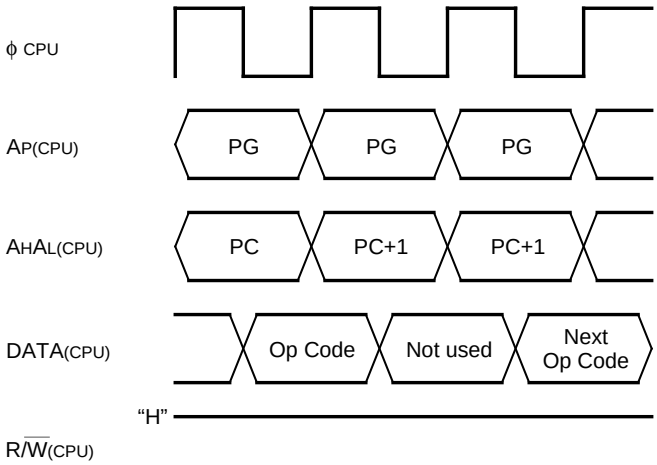


EOL announced

Implied

Instructions : CLC, CLI, CLM, CLV, DEX, DEY, INX, INY, NOP, SEC, SEI, SEM, TAD, TAS, TAX, TAY, TDA, TSA, TSX, TXA, TXS, TXY, TYA, TYX

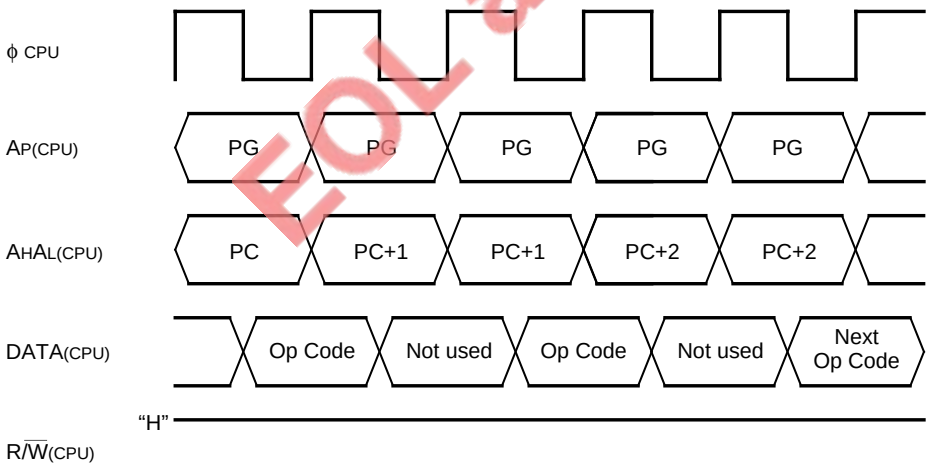
Timing :



EOL announced

Instructions : TBD, TBS, TBX, TBY, TDB, TSB, TXB, TYB

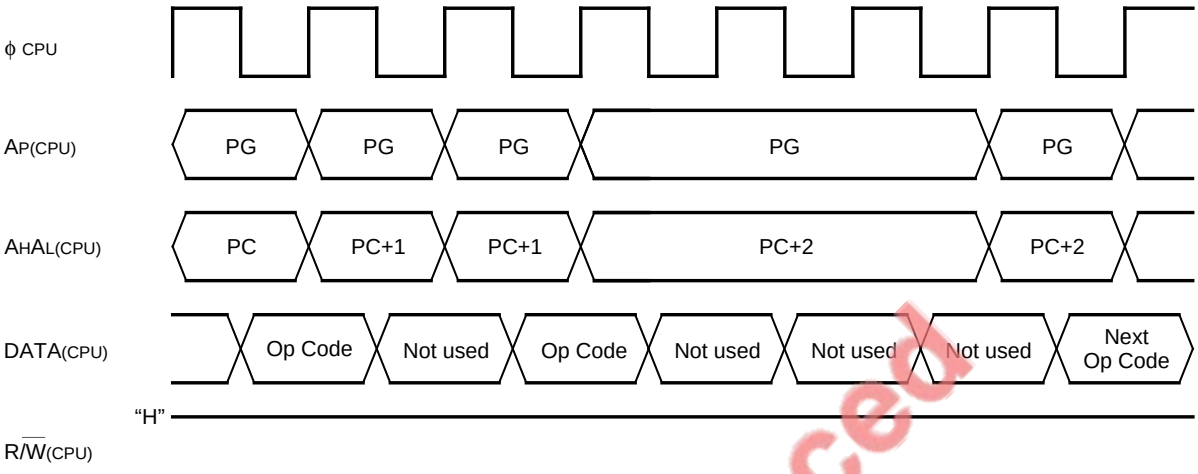
Timing :



Implied

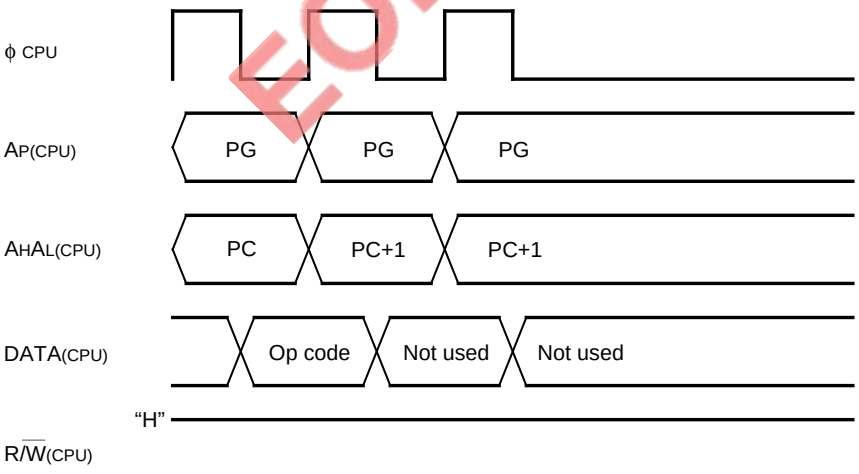
Instructions : XAB

Timing :



Instructions : STP, WIT

Timing :

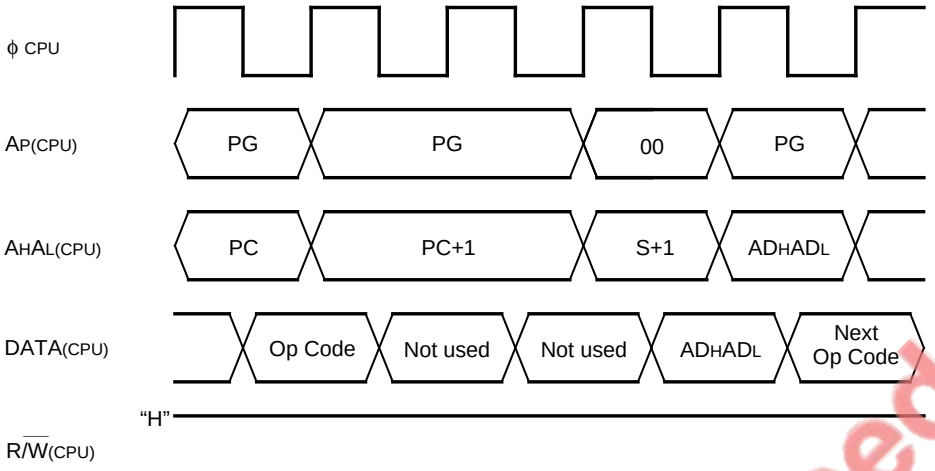


EOL announced

Implied

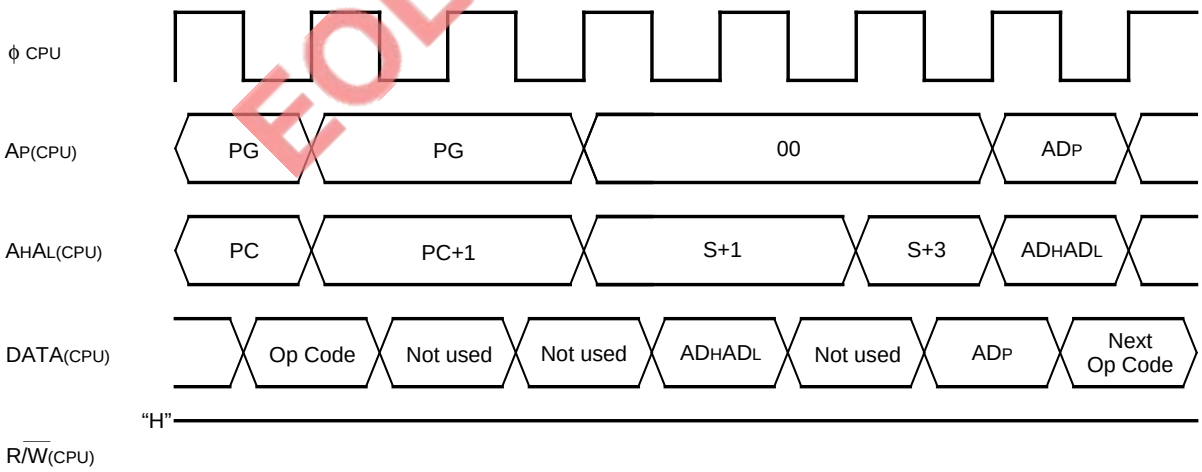
Instructions : RTS

Timing :



Instructions : RTL

Timing :

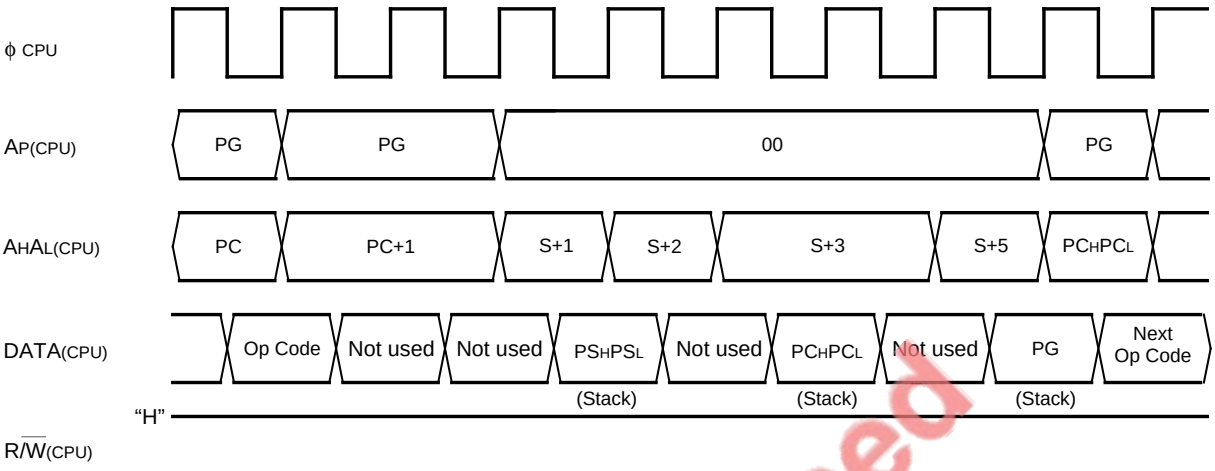


EOL announced

Implied

Instructions : RTI

Timing :

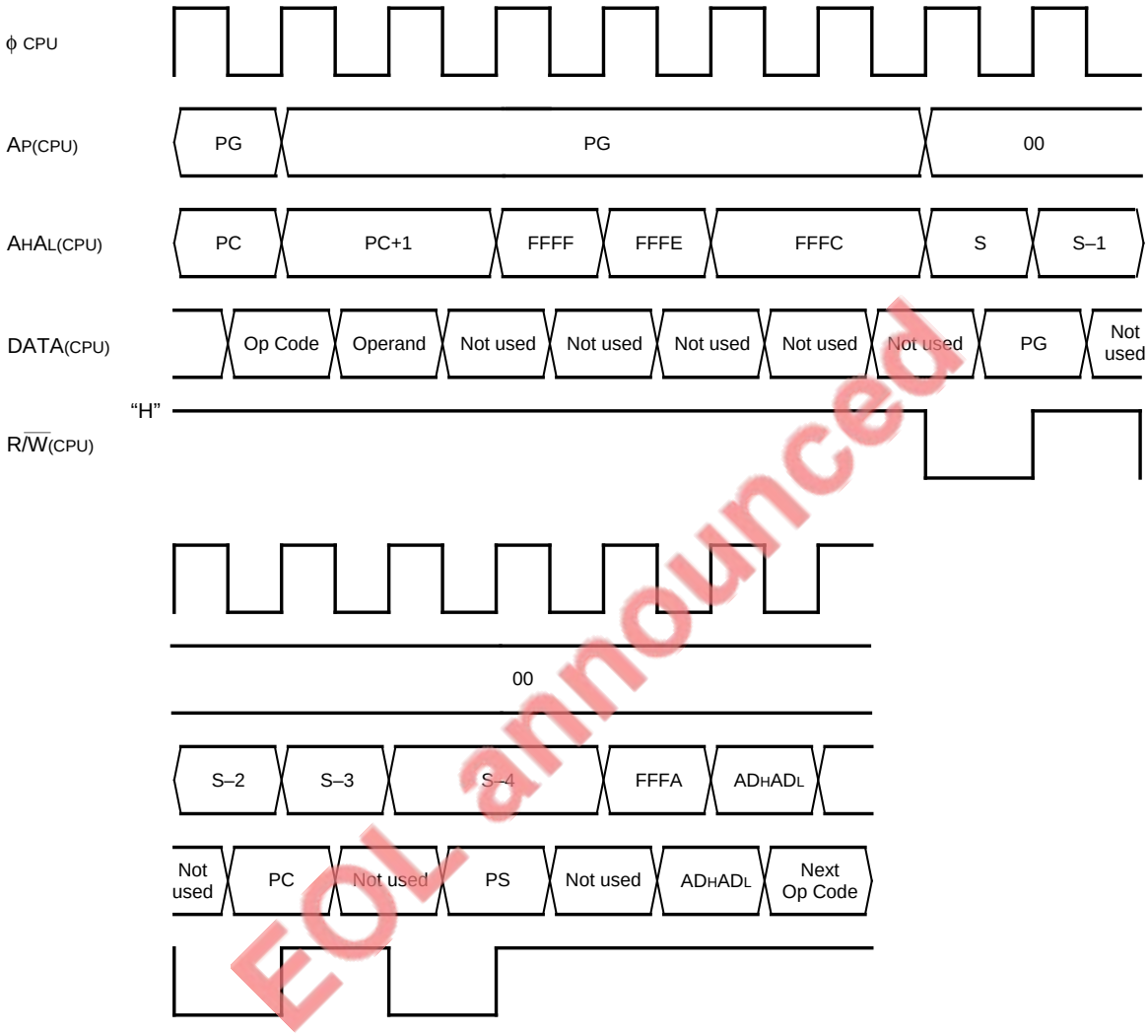


EOL announced

Implied

Instructions : BRK

Timing :

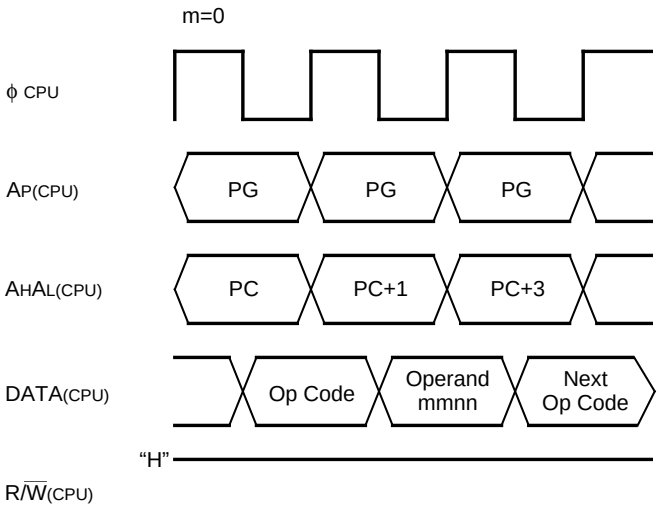


EOL announced

Immediate

Instructions : ADC, AND, CMP, EOR, LDA, ORA, SBC

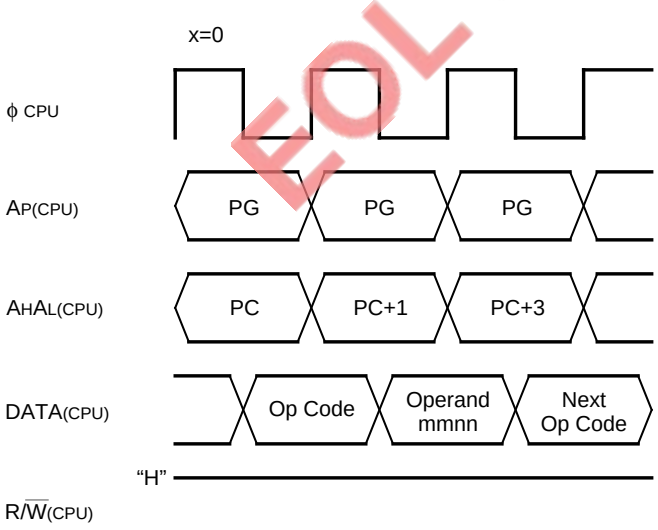
Timing :



When m=1, fetched operand at 2-nd cycle is 1-byte (nn).

Instructions : LDX, LDY, CPX, CPY

Timing :



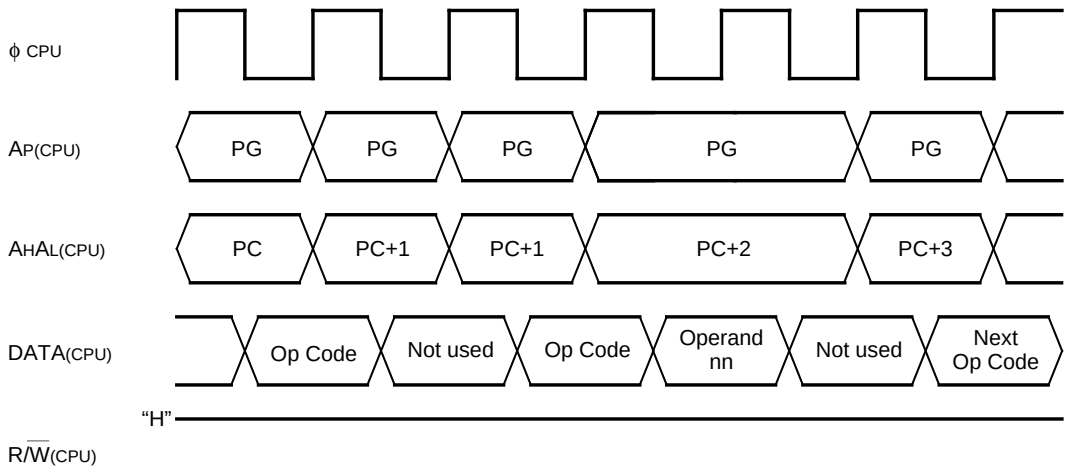
When x=1, fetched operand at 2-nd cycle is 1-byte (nn).

EOL announced

EOL announced

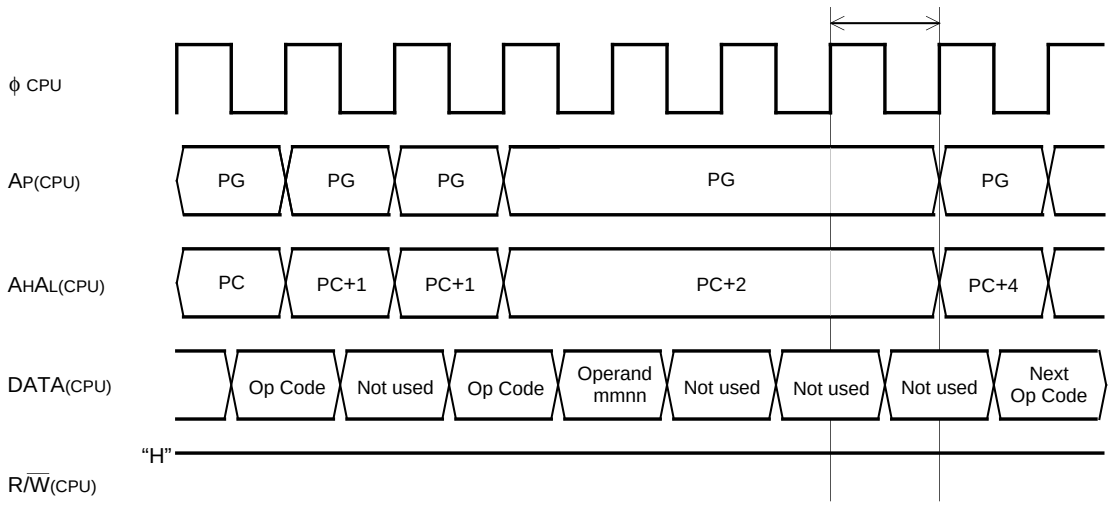
Instructions : LDT

Timing :



Instructions : RLA

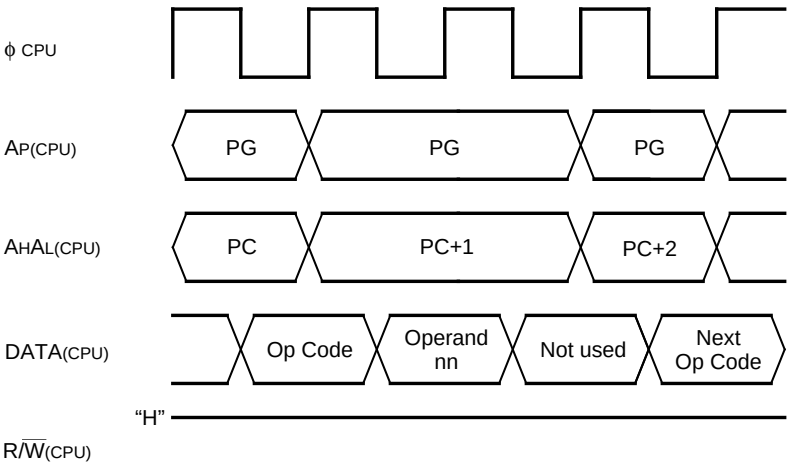
Timing :



This Figure is shown that shifted 1 bit. If shifted more than 2 bit, the cycle " ⟷ " is repeated each shift number worth. When 0 bit shift(not shifted), the cycle " ⟷ " is nothing.
When m=1, fetched operand at 4-th cycle is 1-byte (nn).

Instructions : SEP

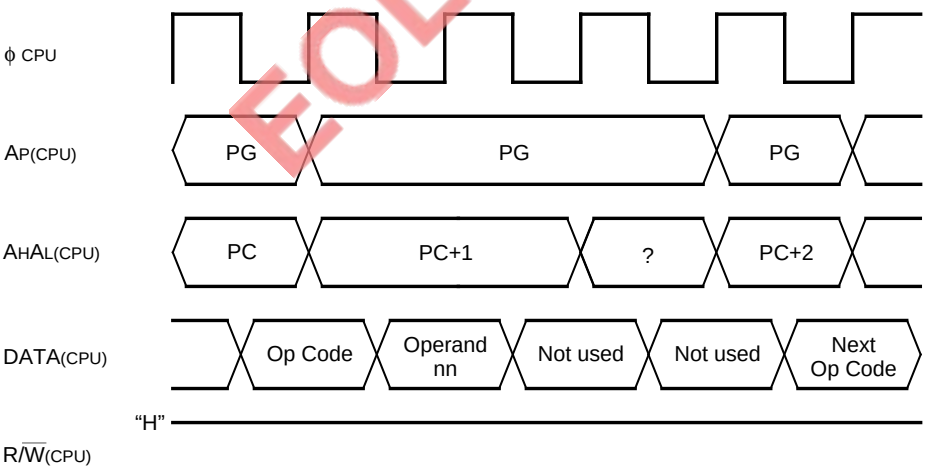
Timing :



EOL announced

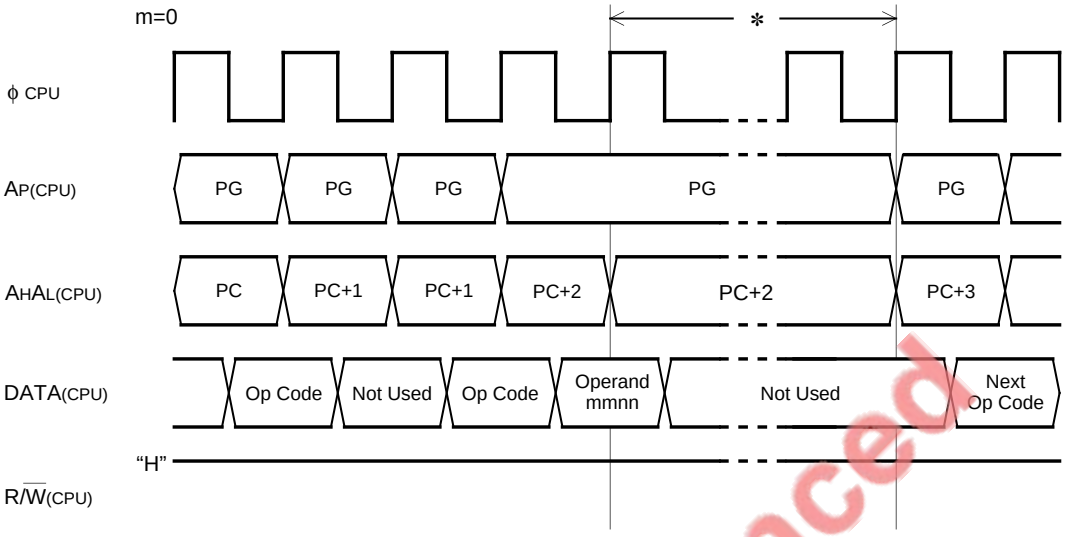
Instructions : CLP

Timing :



Instructions : DIV, DIVS*, MPY, MPYS*

Timing :



When m=1, fetched operand at 4-th cycle is 1-byte(nn).

(Note) The cycle number during ✻ is shown in following table

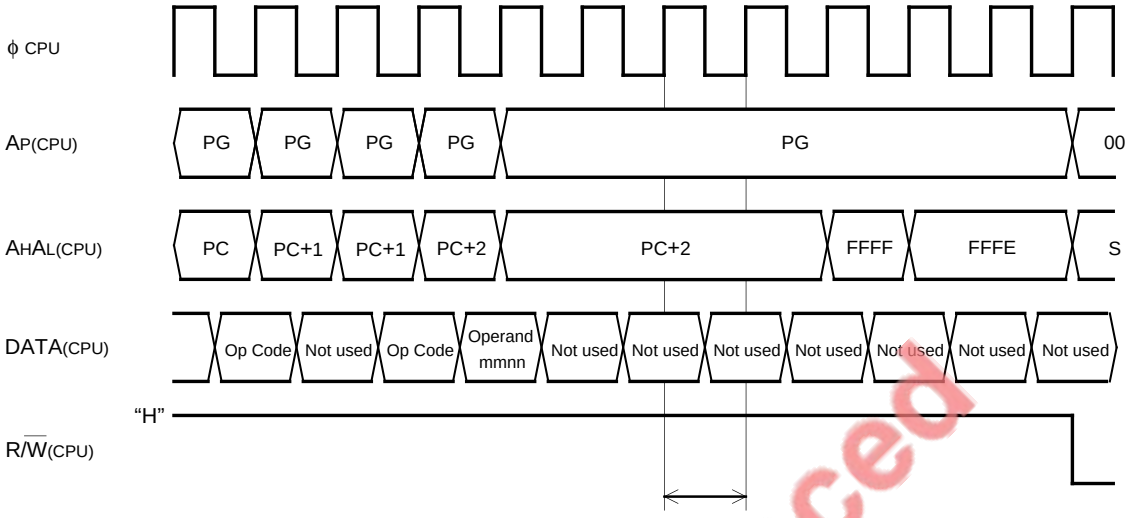
Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus✕Plus	22	14
	Minus✕Minus		
	Plus✕Minus	25	17
	Minus✕Plus		

- The contents of AHAL(CPU) during ✻ of the DIVS instruction is undefined.
- When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during ✻ is undefined.

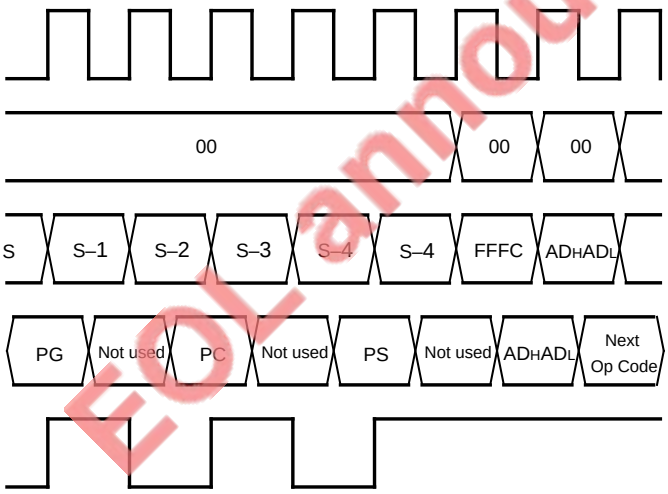
EOL announced

Instructions : DIV, DIVS* (case of zero division)

Timing :



At DIVS instruction, this cycle is nothing.

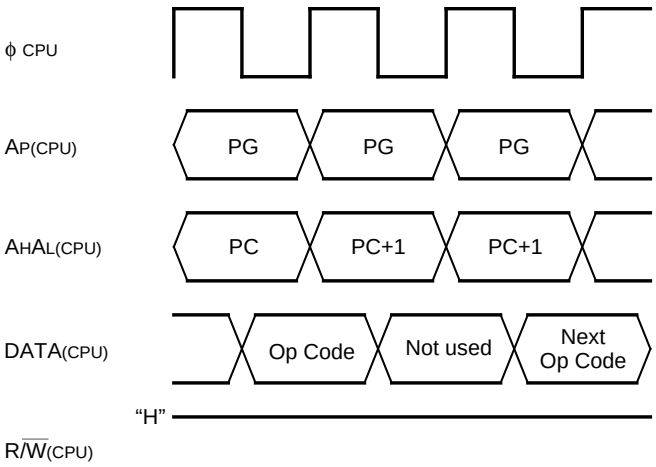


When m=1, fetched operand at 4-th cycle is 1-byte(nn).

EOL announced

Instructions : ASL, DEC, INC, LSR, ROL, ROR

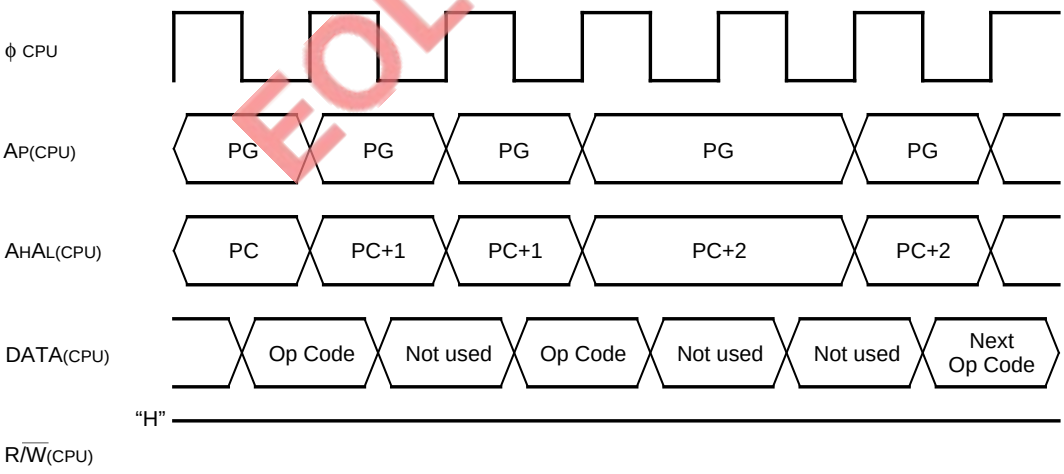
Timing :



EOL announced

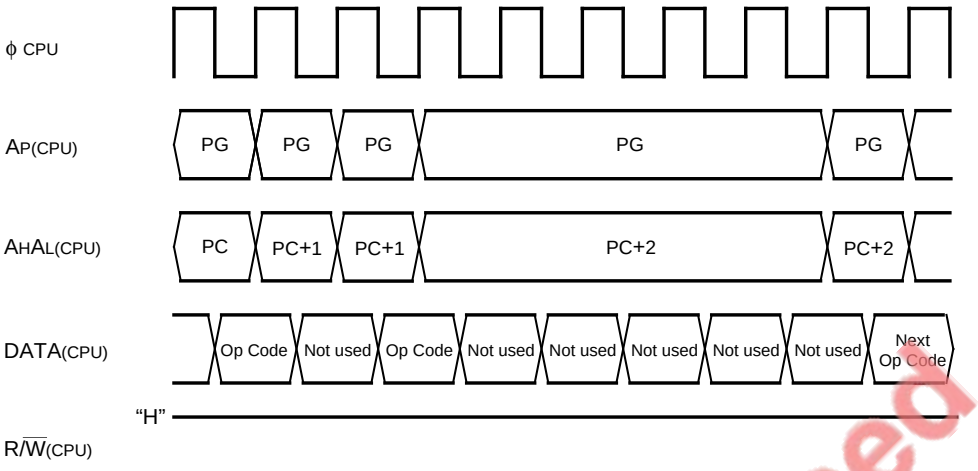
Instruction : ASR*, EXTZ*

Timing :



Instruction : EXTS*

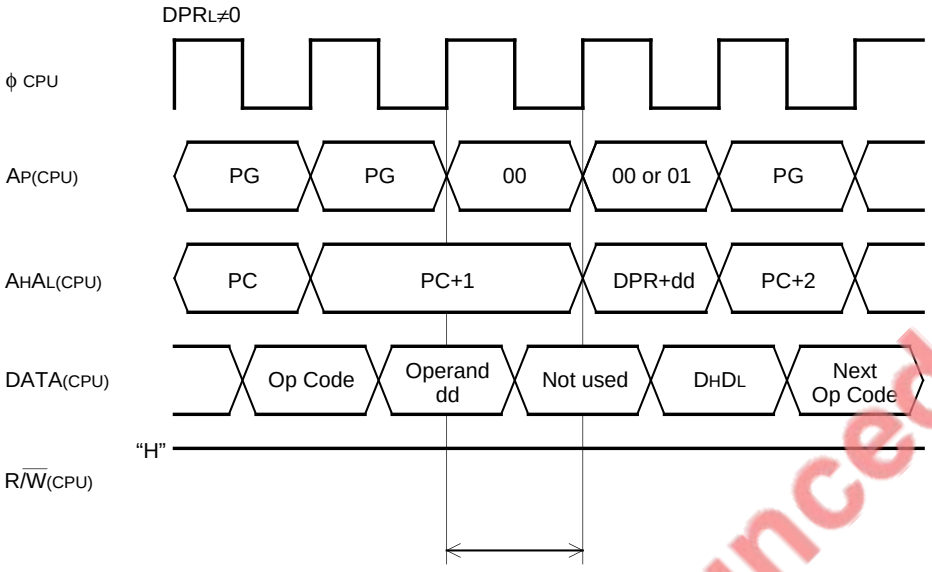
Timing :



EOL announced

Instruction : ADC, AND, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA, SBC

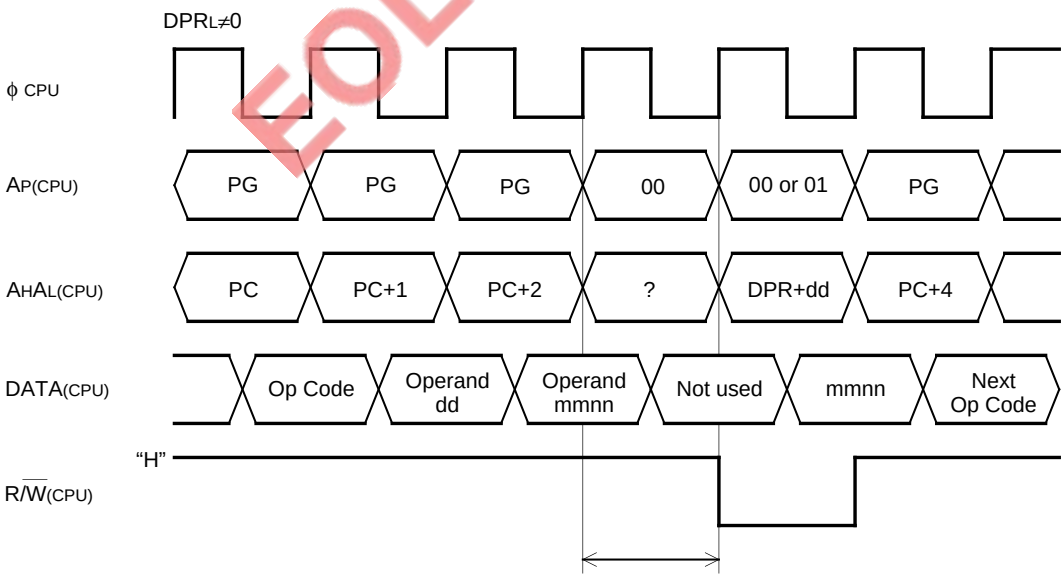
Timing :



When DPRL=0, this cycle is nothing.

Instruction : LDM

Timing :



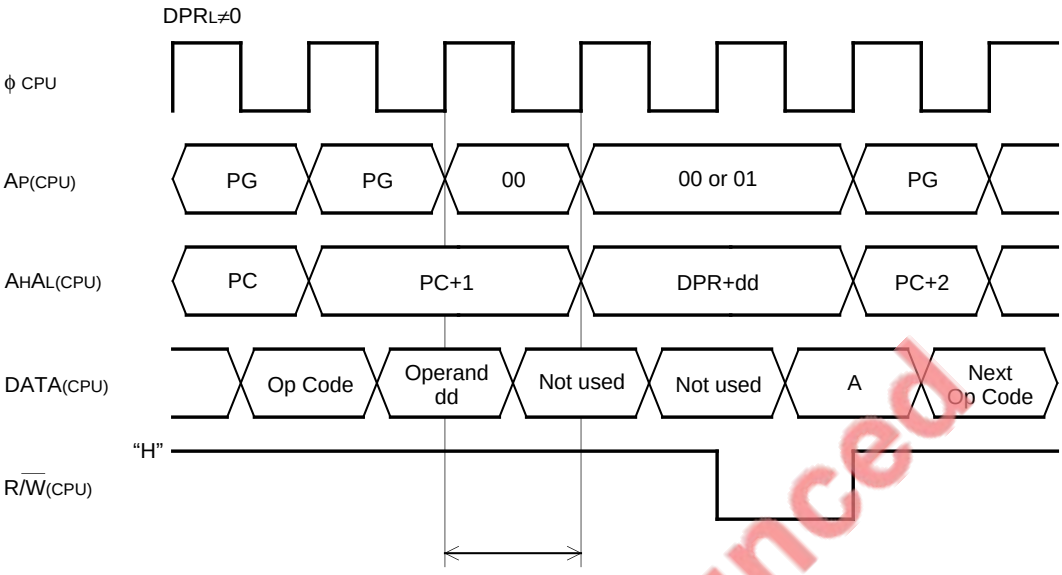
When DPRL=0, this cycle is nothing.

When m=1, fetched operand at 3-rd cycle and data at 5-th cycle are 1-byte(nn).

EOL announced

Instruction : STA, STX, STY

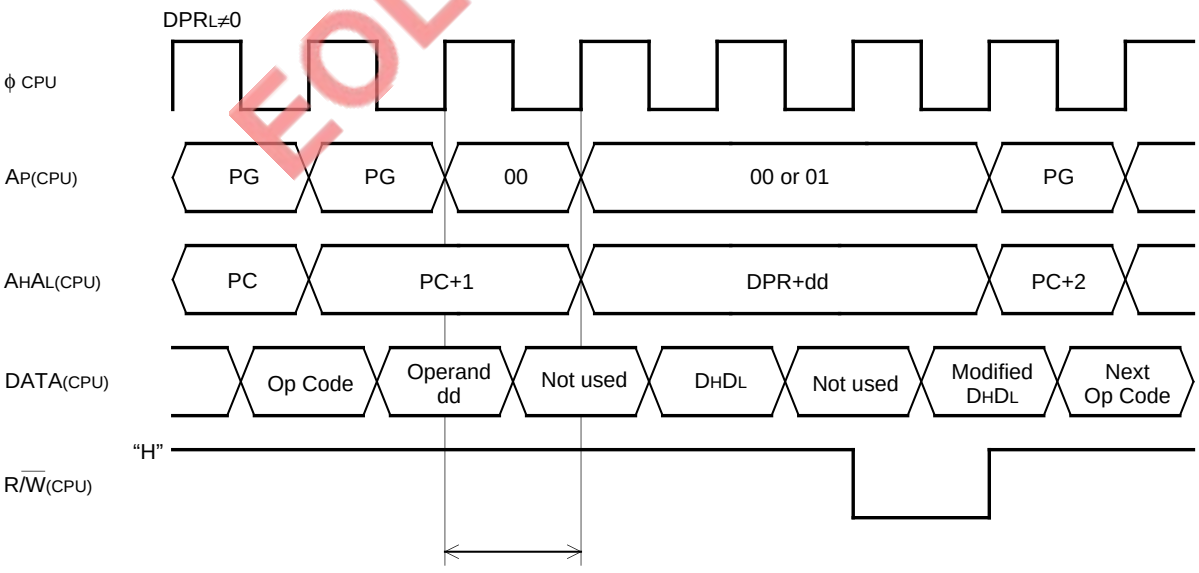
Timing :



When DPRL=0, this cycle is nothing.

Instruction : ASL, DEC, INC, LSR, ROL, ROR

Timing :

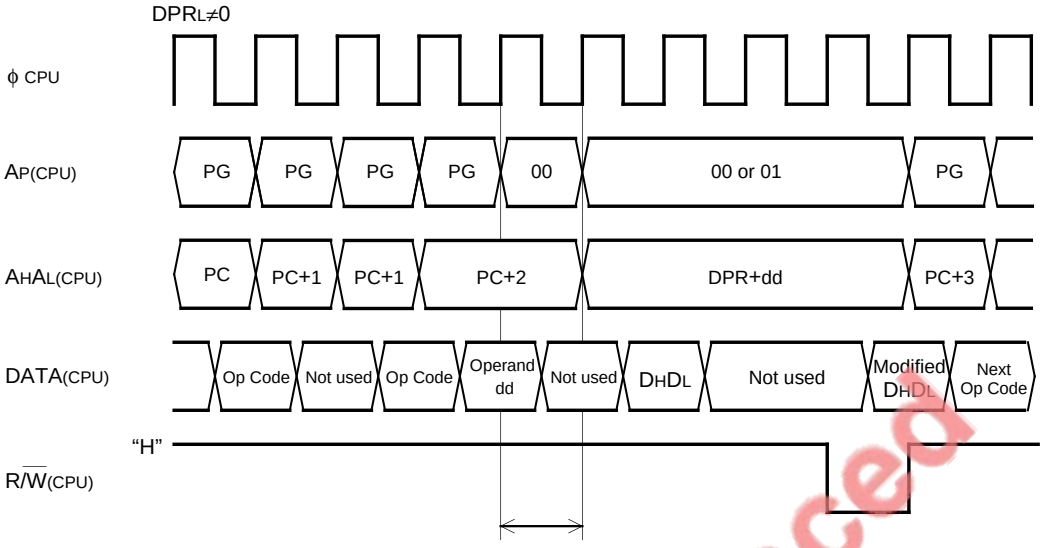


When DPRL=0, this cycle is nothing.

EOL announced

Instruction : ASR*

Timing :

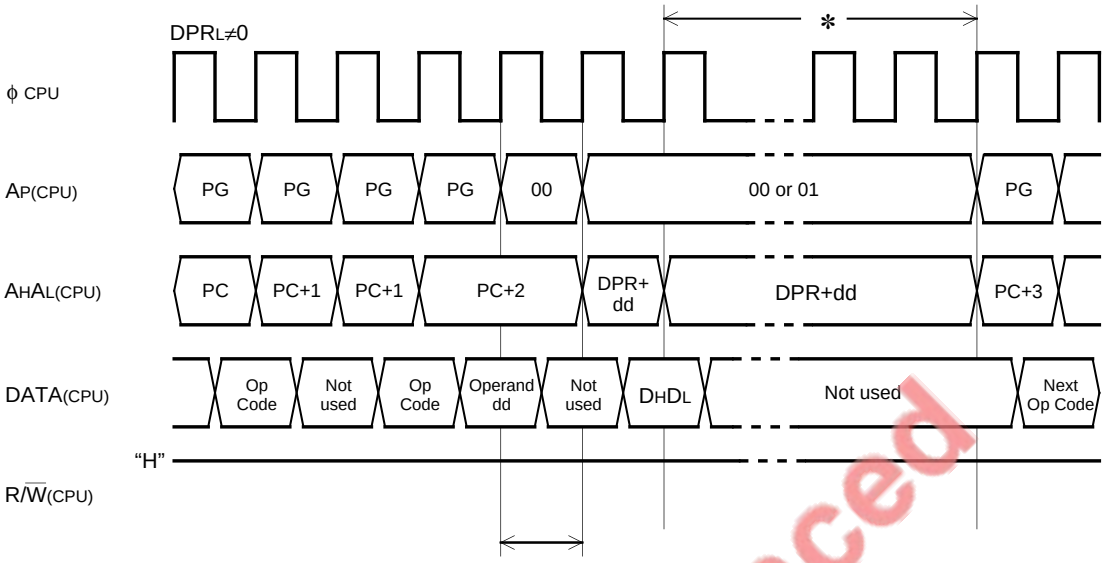


When DPRL=0, this cycle is nothing.

EOL announced

Instruction : DIV, DIVS*, MPY, MPYS*

Timing :



When DPRL=0, this cycle is nothing.

(Note) The cycle number during ✻ is shown in following table

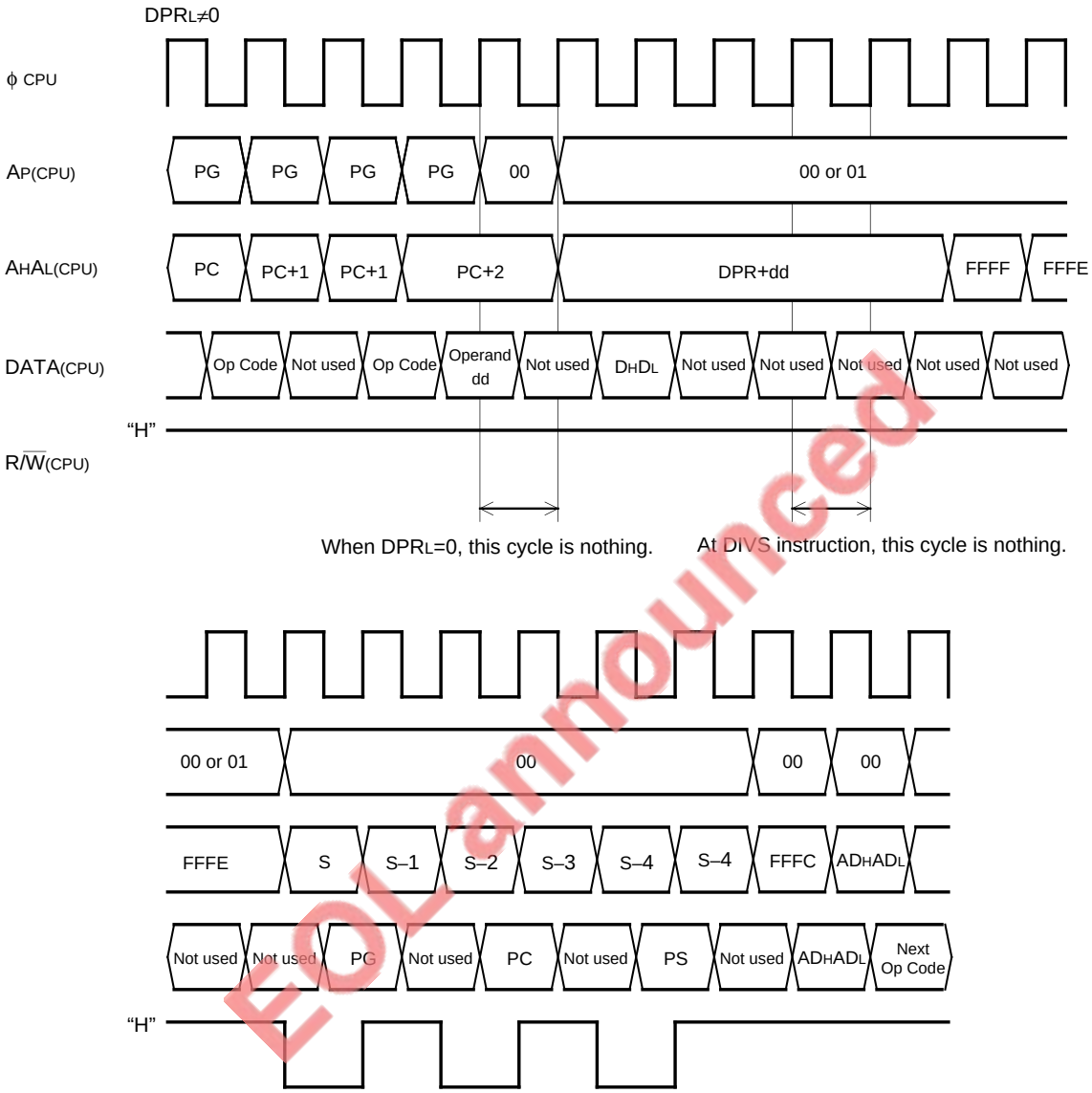
Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus×Plus	22	14
	Minus×Minus		
	Plus×Minus	25	17
	Minus×Plus		

- The contents of AHAL(CPU) during ✻ of the DIVS instruction is undefined.
- When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during ✻ is undefined.

EOL announced

Instruction : DIV, DIVS* (case of 0 division)

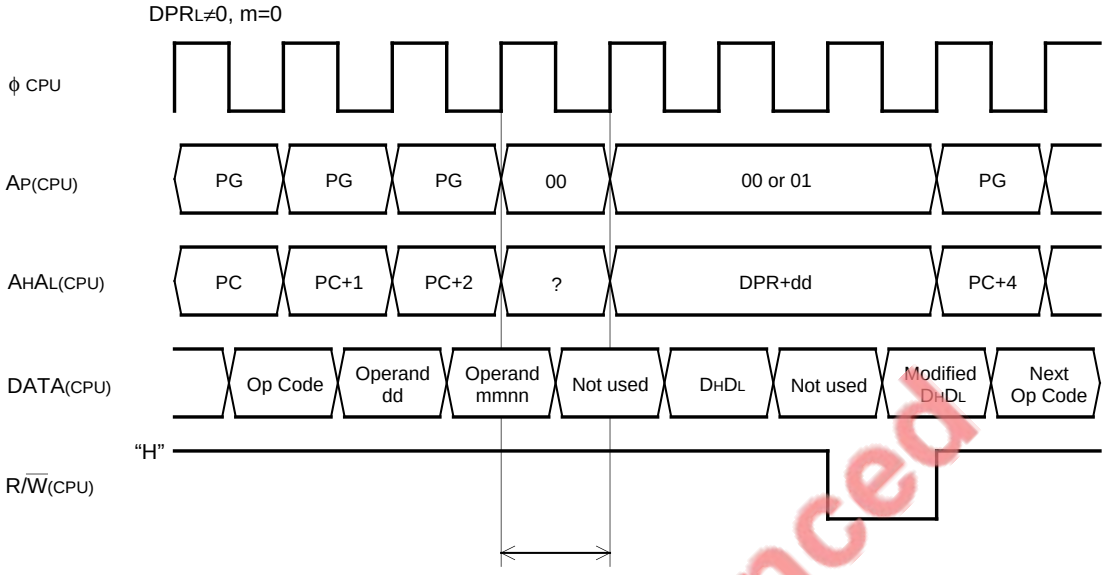
Timing :



EOL announced

Instruction : CLB, SEB

Timing :



When DPRL=0, this cycle is nothing.

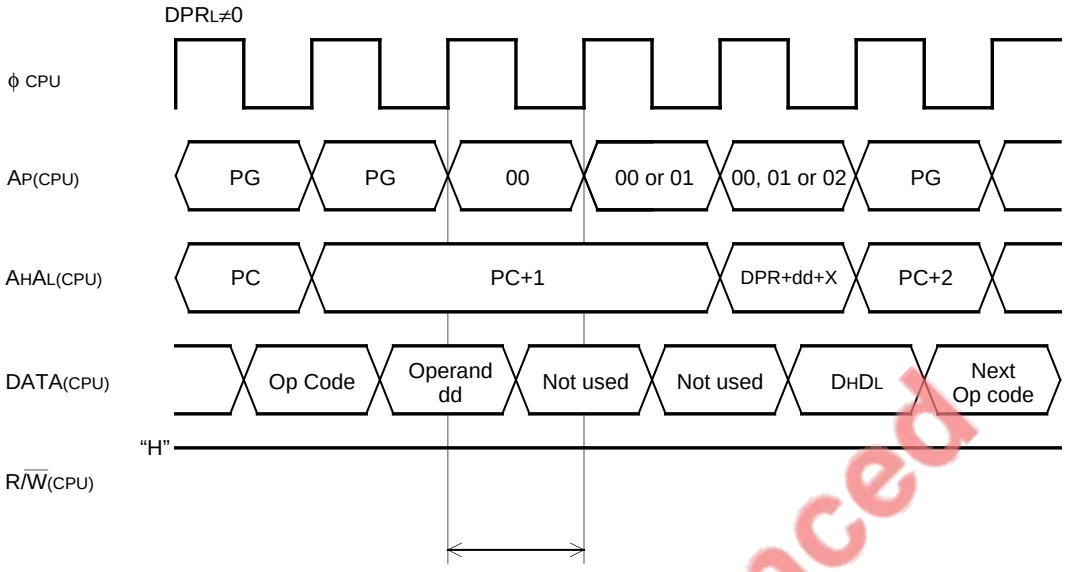
When m=1, fetched operands at 3-rd cycle is 1-byte (nn).

EOL announced

Direct Indexed X

Instruction : ADC, AND, CMP, EOR, LDA, LDY, ORA, SBC

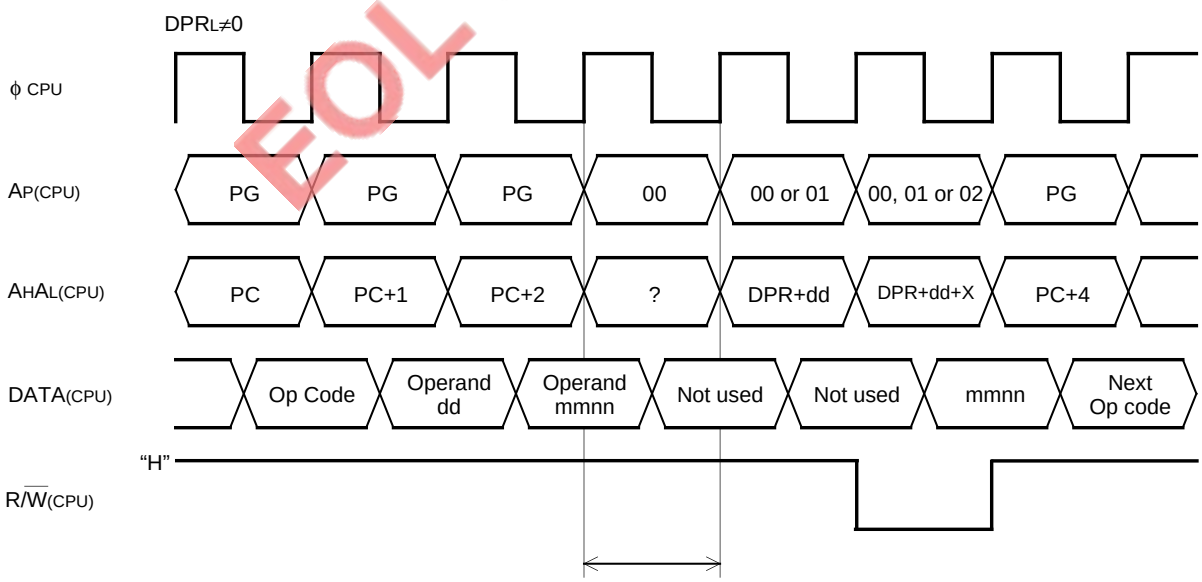
Timing :



When DPRL=0, this cycle is nothing.

Instruction : LDM

Timing :



When DPRL=0, this cycle is nothing.

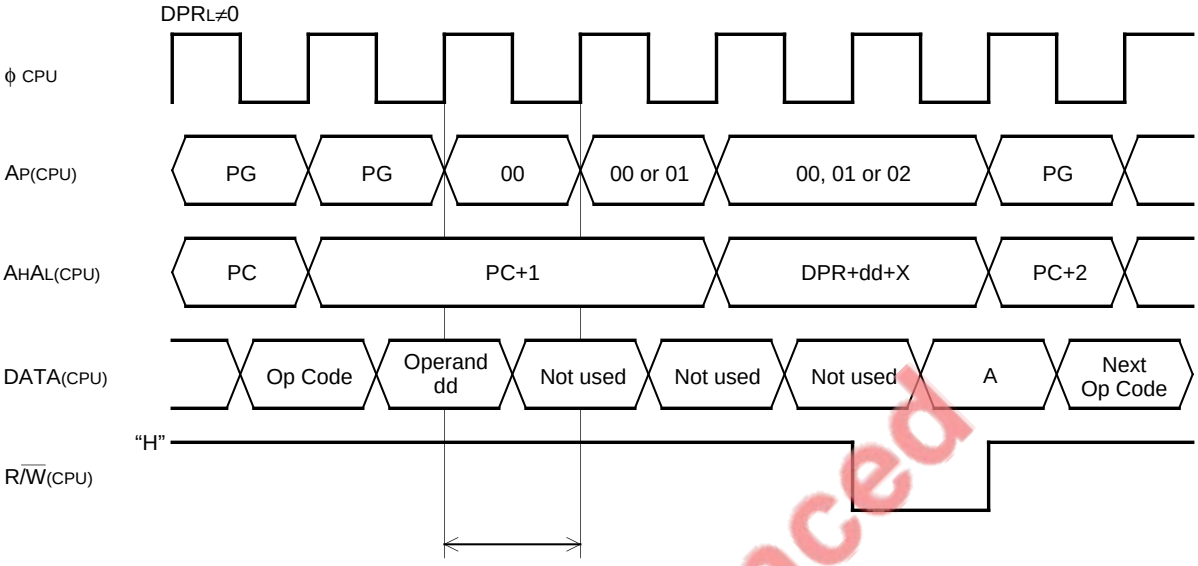
When m=1, fetched operand at 3-rd cycle and data at 6-th cycle are 1-byte(nn).

EOL announced

Direct Indexed X

Instruction : STA, STY

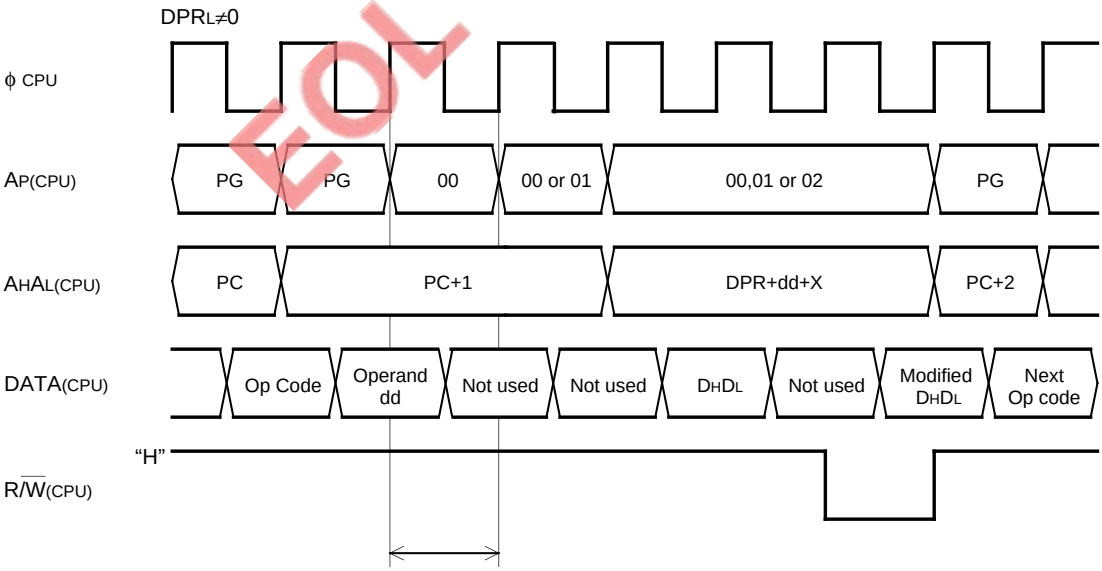
Timing :



When DPRL=0, this cycle is nothing.

Instruction : ASL, DEC, INC, LSR, ROL, ROR

Timing :



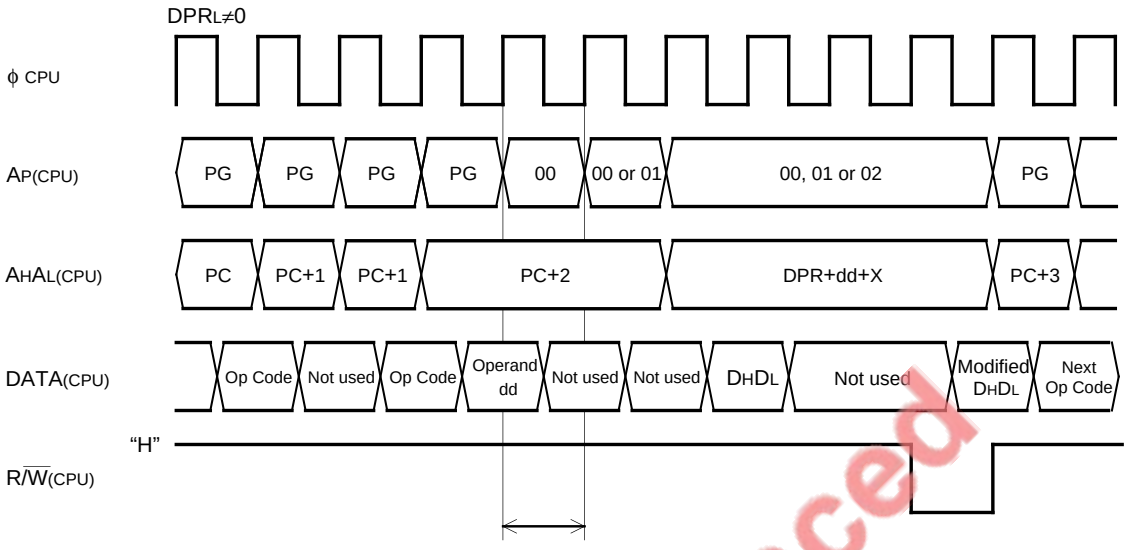
When DPRL=0, this cycle is nothing.

EOL announced

Direct Indexed X

Instruction : ASR*

Timing :

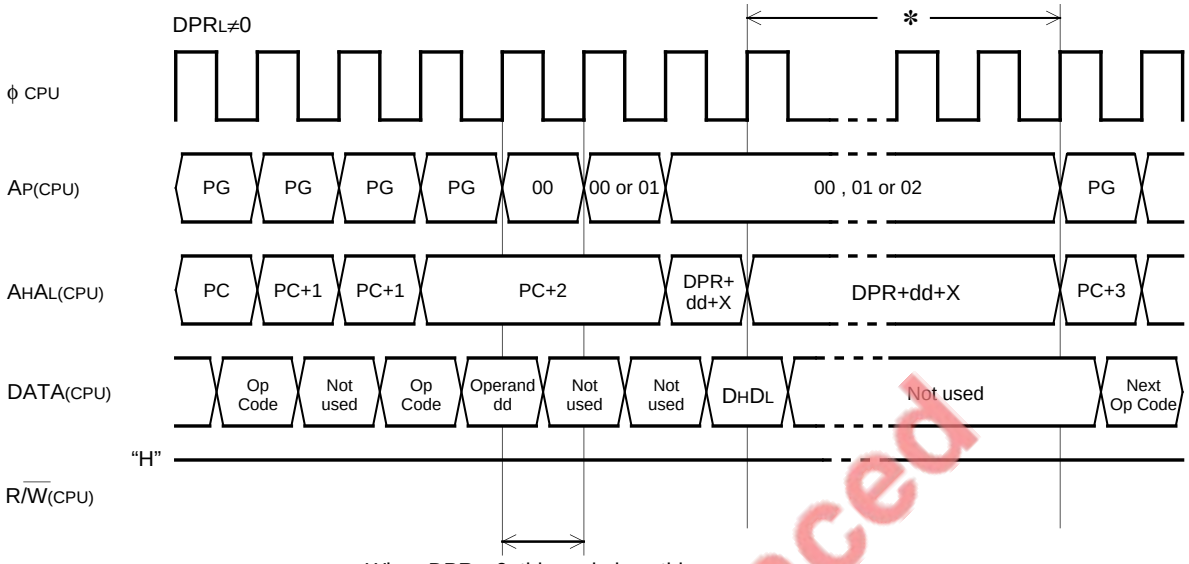


When DPRL=0, this cycle is nothing.

EOL announced

Instruction : DIV, DIVS*, MPY, MPYS*

Timing :



When DPRL=0, this cycle is nothing.

(Note) The cycle number during * is shown in following table

Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus×Plus	22	14
	Minus×Minus		
	Plus×Minus	25	17
	Minus×Plus		

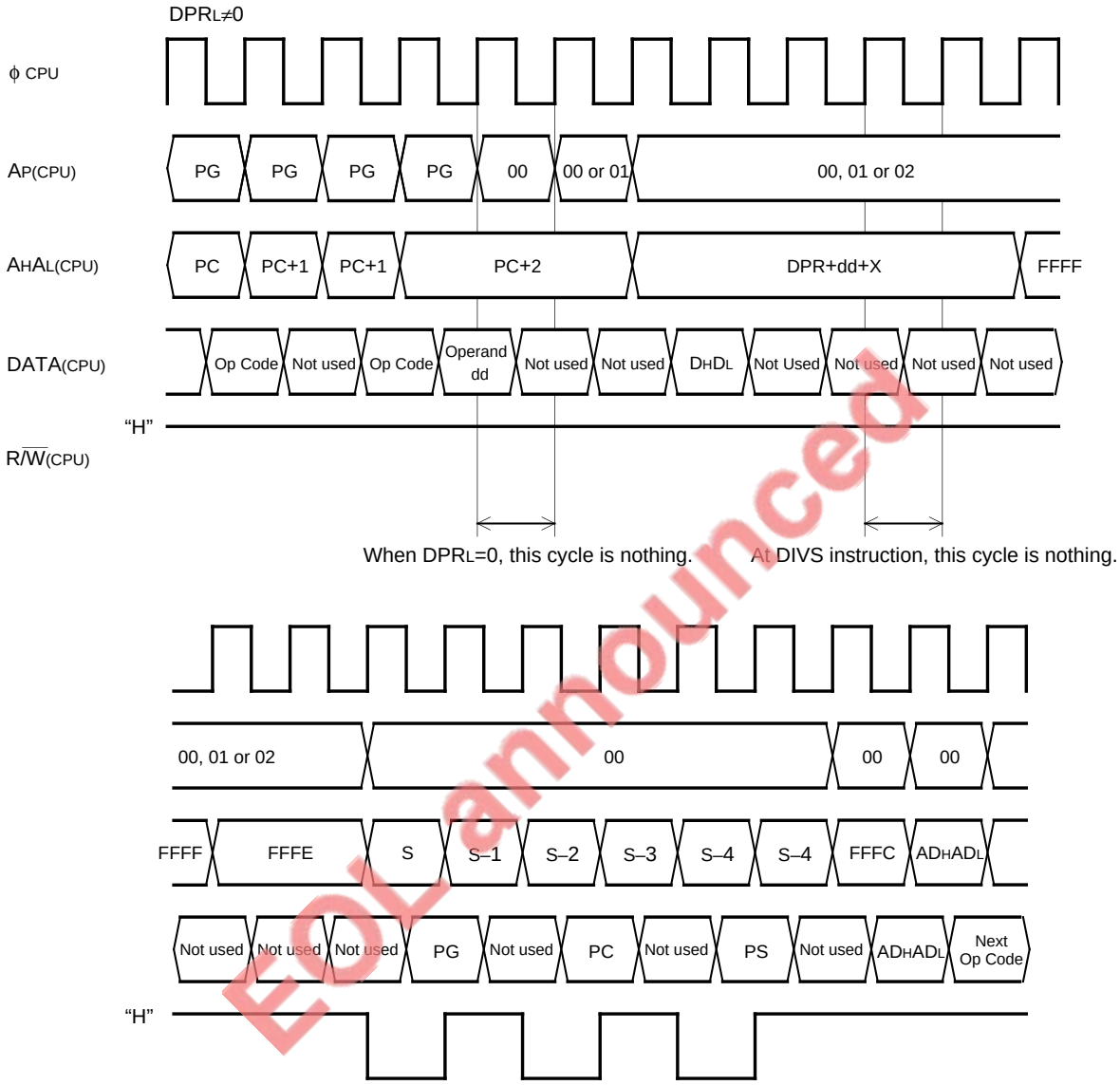
- The contents of AHAL(CPU) during * of the DIVS instruction is undefined.
- When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during * is undefined.

EOL announced

Direct Indexed X

Instruction : DIV, DIVS* (case of 0 division)

Timing :

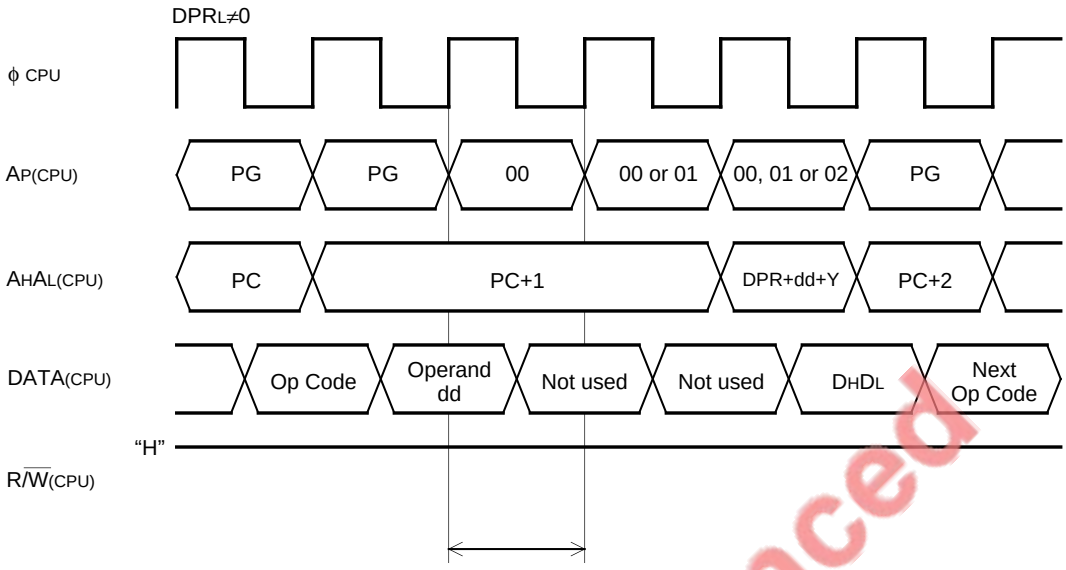


EOL announced

Direct Indexed Y

Instruction : LDX

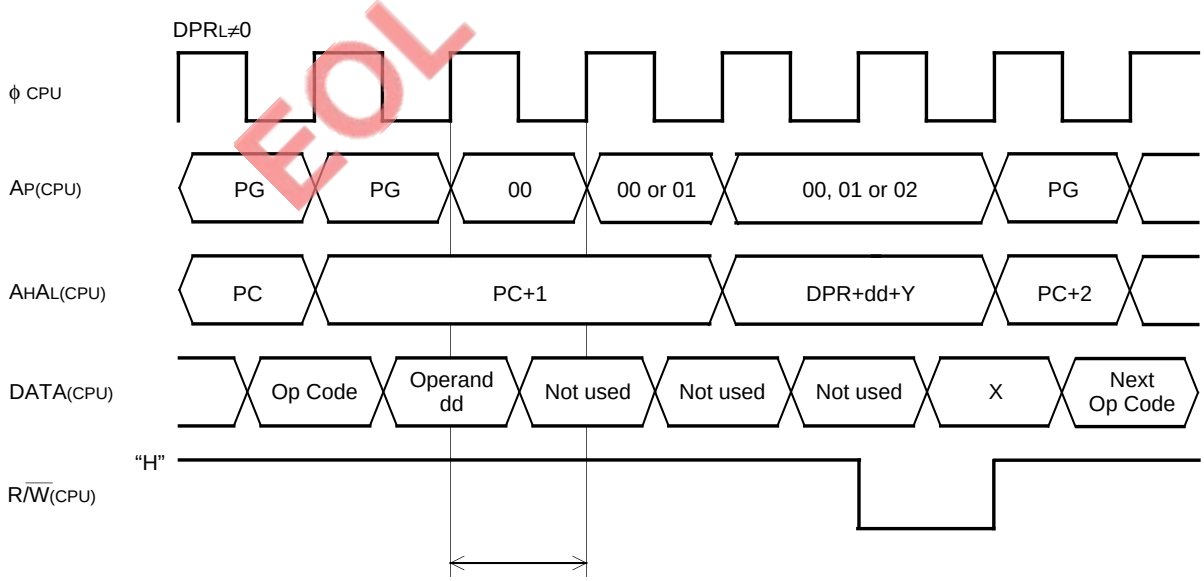
Timing :



When DPRL=0, this cycle is nothing.

Instruction : STX

Timing :



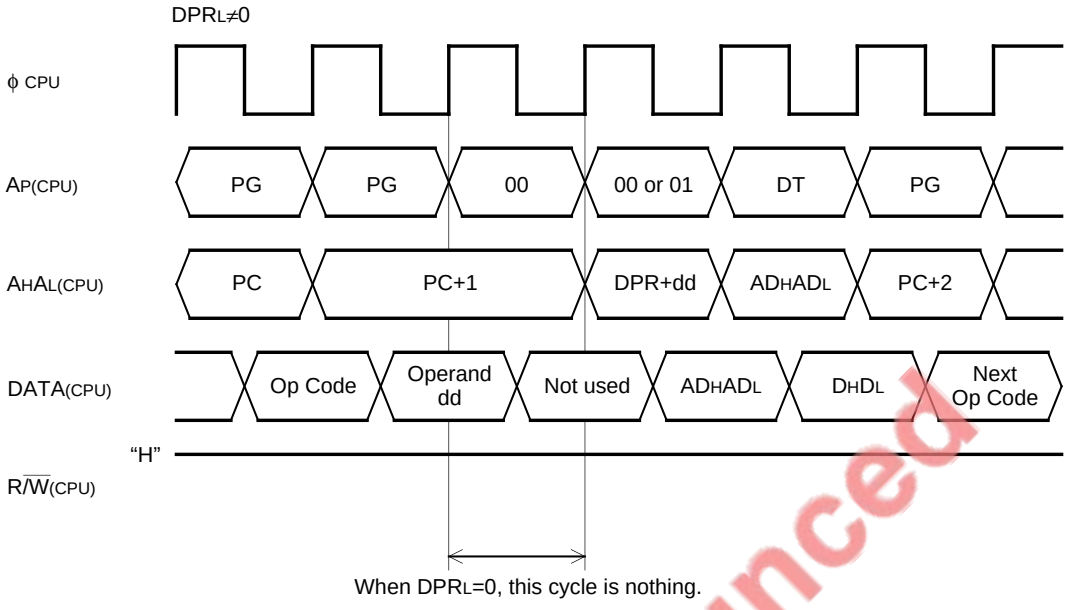
When DPRL=0, this cycle is nothing.

EOL announced

Direct Indirect

Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

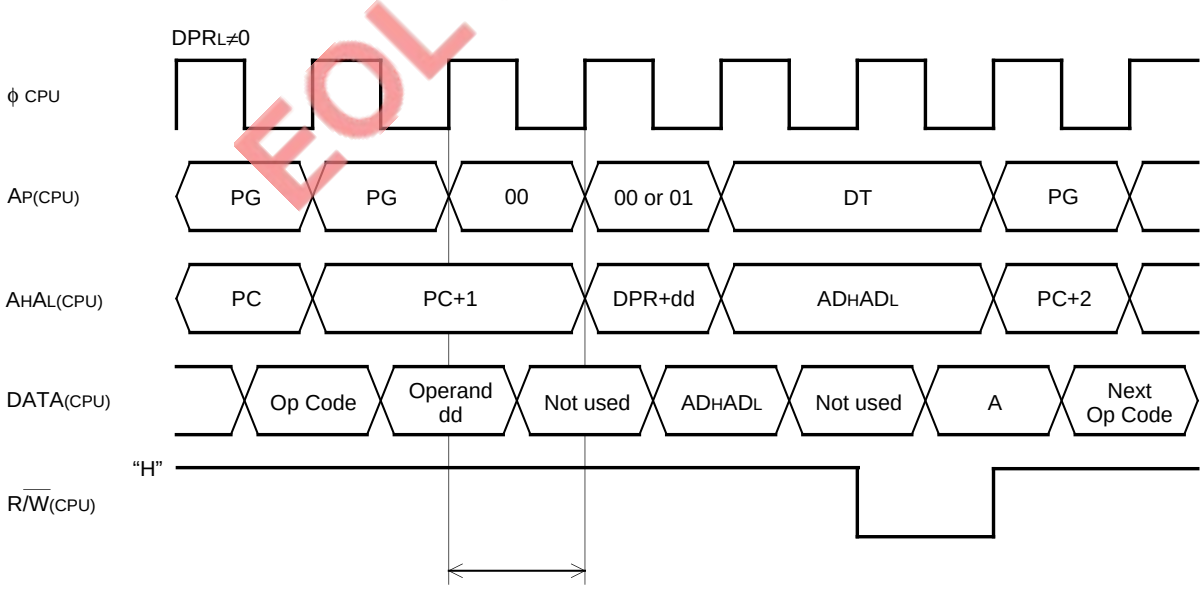
Timing :



When DPRL=0, this cycle is nothing.

Instruction : STA

Timing :

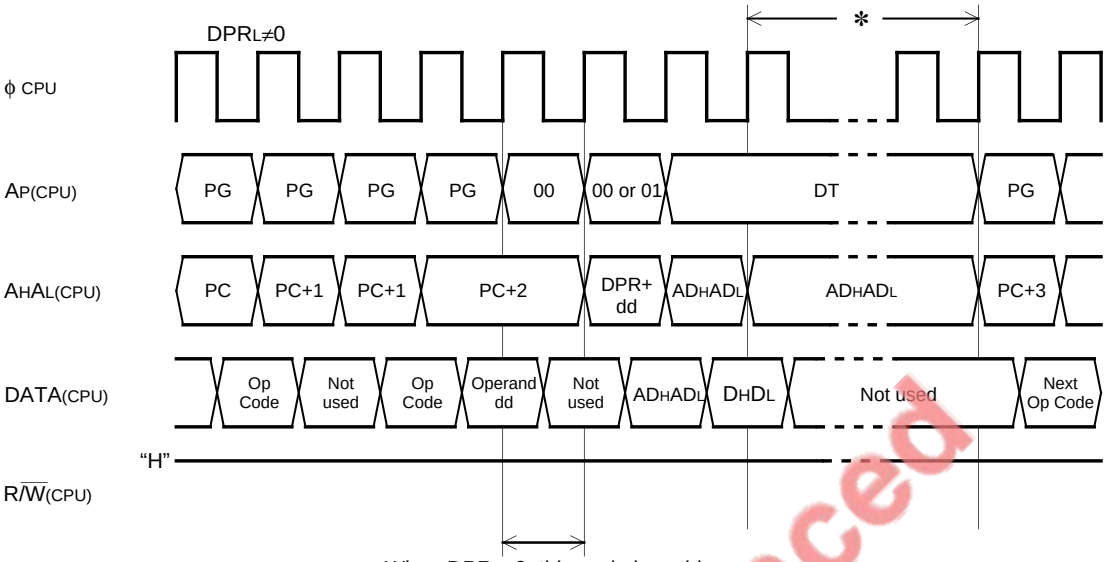


When DPRL=0, this cycle is nothing.

EOL announced

Instruction : DIV, DIVS*, MPY, MPYS*

Timing :



When DPRL=0, this cycle is nothing.

(Note) The cycle number during ✻ is shown in following table

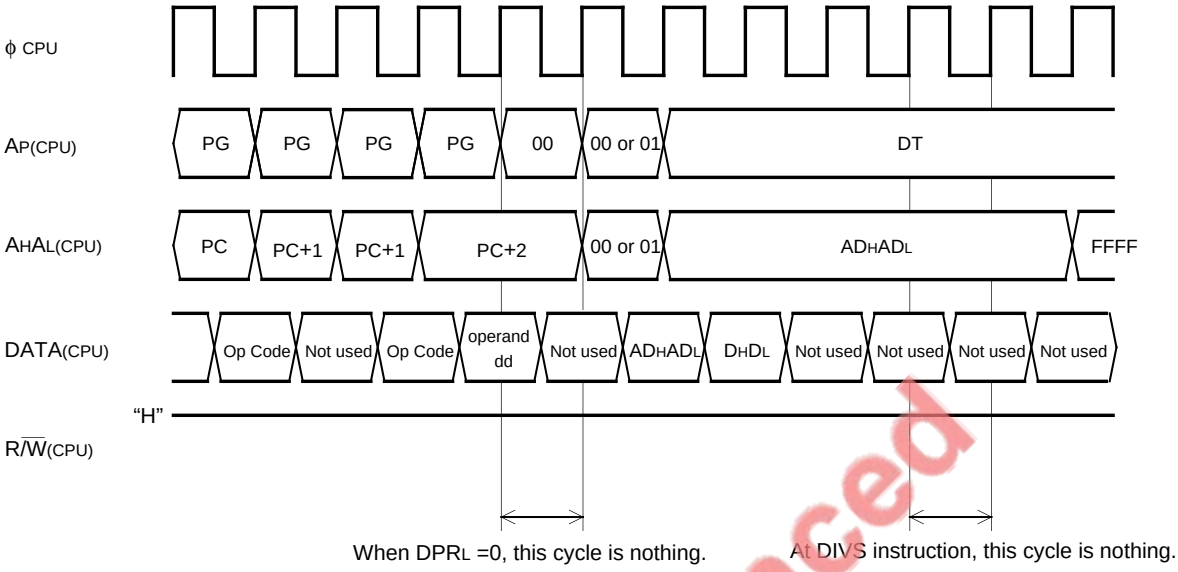
Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus×Plus	22	14
	Minus×Minus		
	Plus×Minus	25	17
	Minus×Plus		

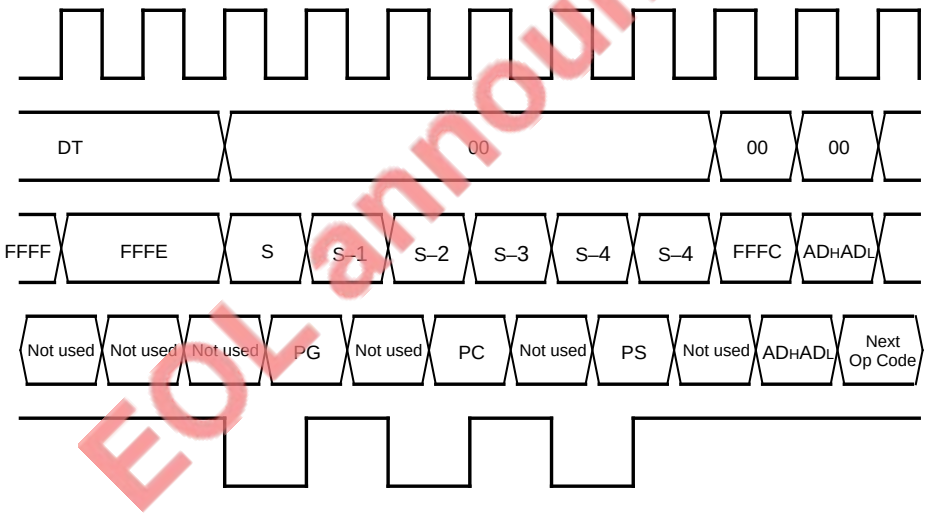
- The contents of AHAL(CPU) during ✻ of the DIVS instruction is undefined.
- When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during ✻ is undefined.

EOL announced

Instruction : DIV, DIVS* (case of 0 division)

Timing :



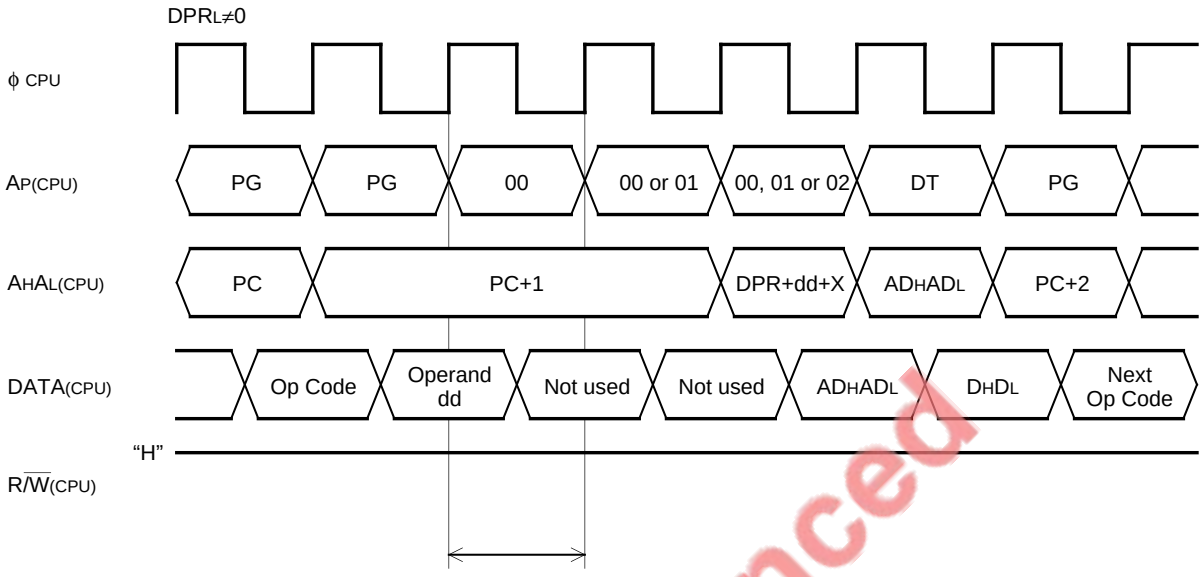


EOL announced

Direct Indexed X Indirect

Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

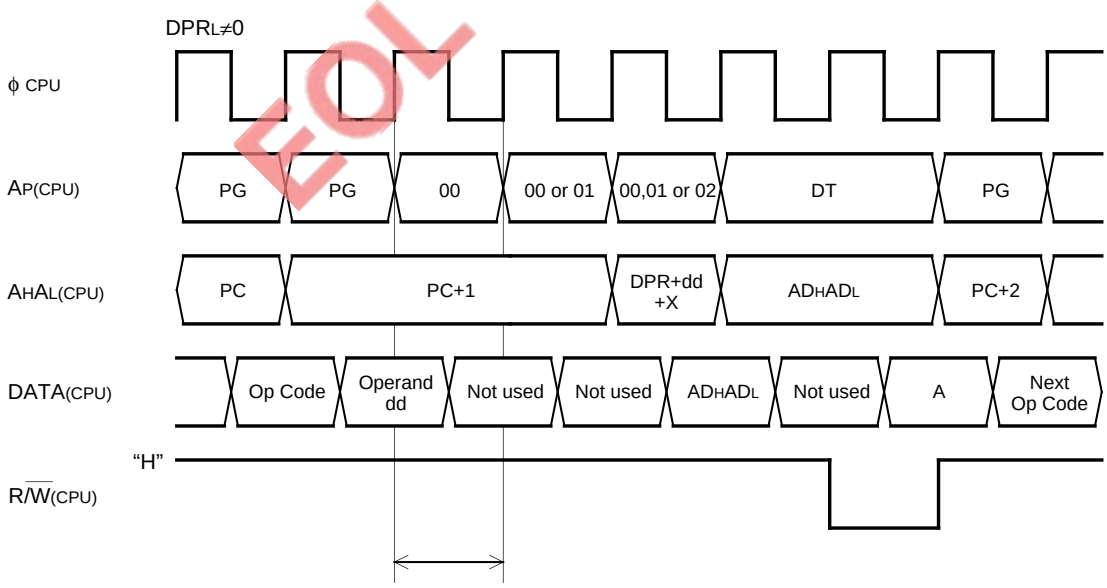
Timing :



When DPRL=0, this cycle is nothing.

Instruction : STA

Timing :



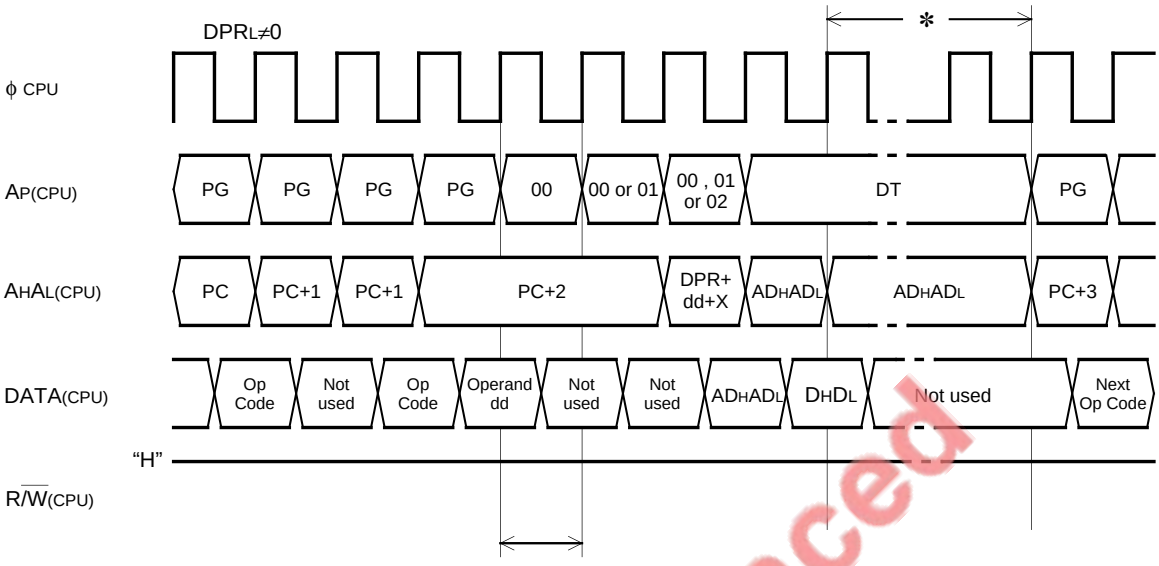
When DPRL=0, this cycle is nothing.

EOL announced

Direct Indexed X Indirect

Instruction : DIV, DIVS*, MPY, MPYS*

Timing :



When DPRL=0, this cycle is nothing.

(Note) The cycle number during ✻ is shown in following table

Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus×Plus	22	14
	Minus×Minus		
	Plus×Minus	25	17
	Minus×Plus		

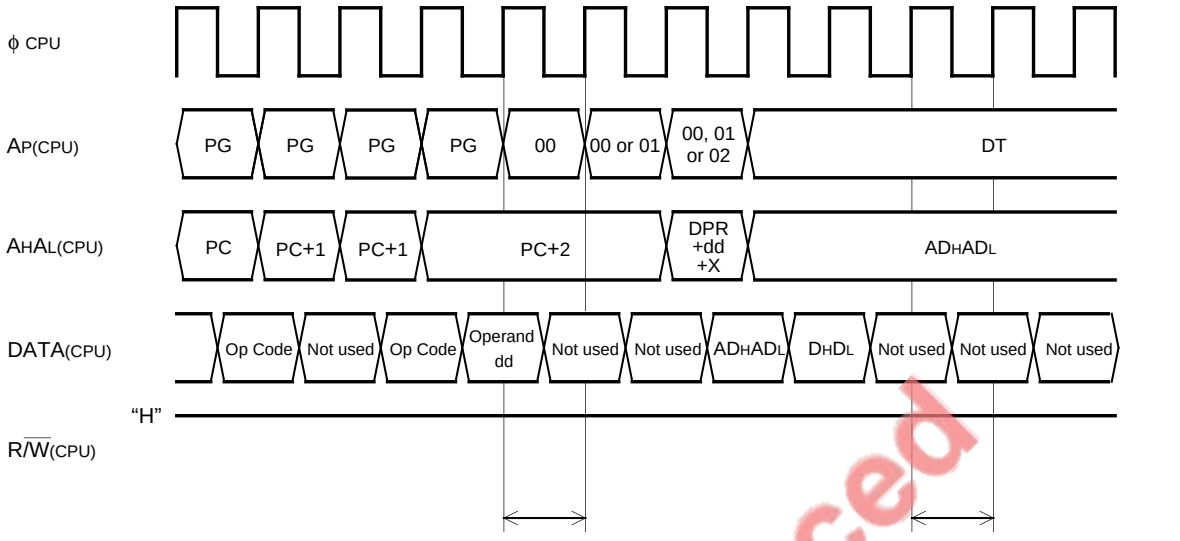
- The contents of AHAL(CPU) during ✻ of the DIVS instruction is undefined.
- When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during ✻ is undefined.

EOL announced

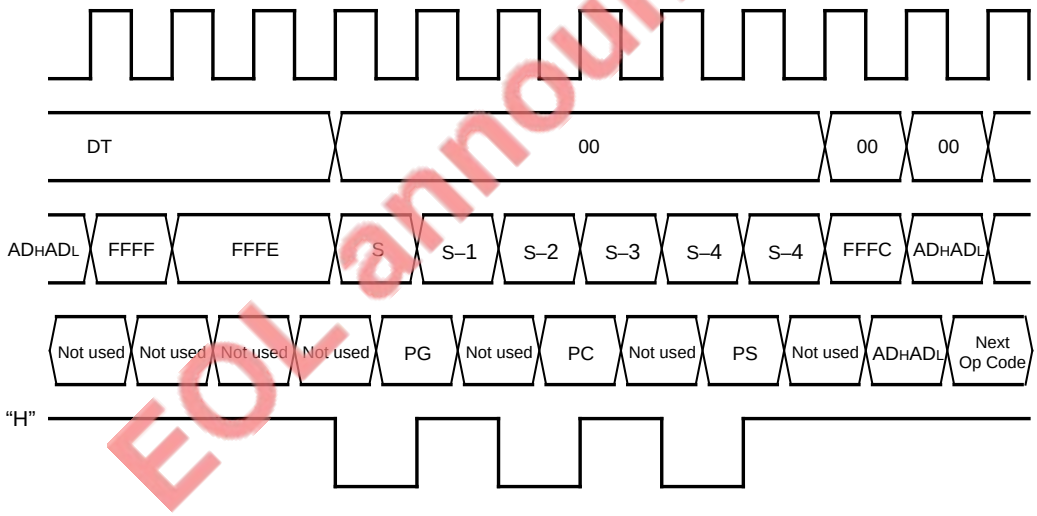
Direct Indexed X Indirect

Instruction : DIV, DIVS* (case of 0 division)

Timing :



When DPRL =0, this cycle is nothing. At DIVS instruction, this cycle is nothing.

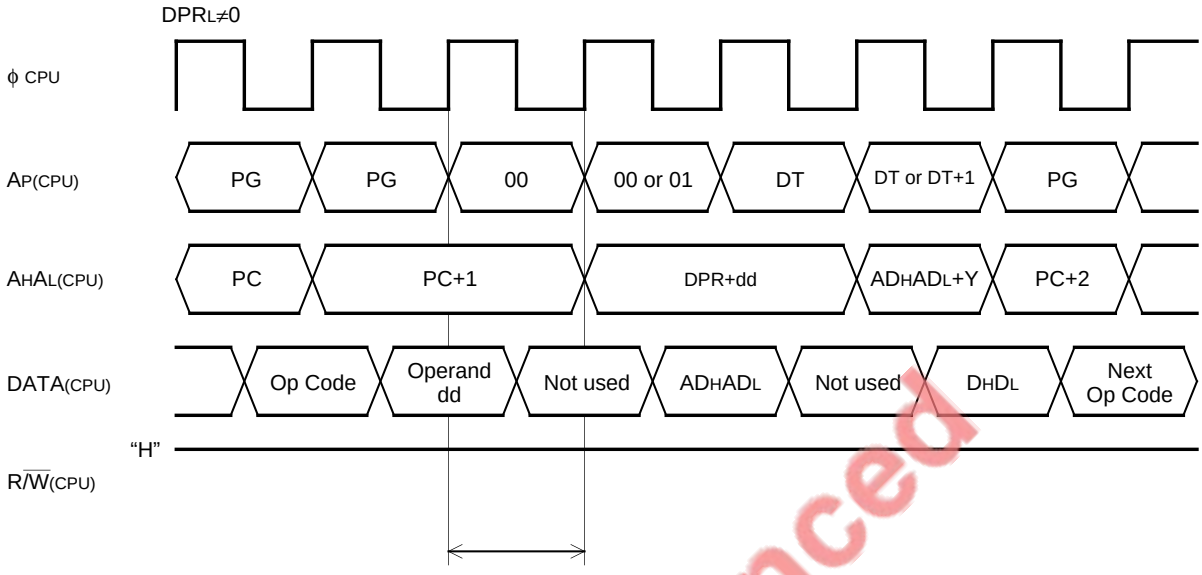


EOL announced

Direct Indirect Indexed Y

Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

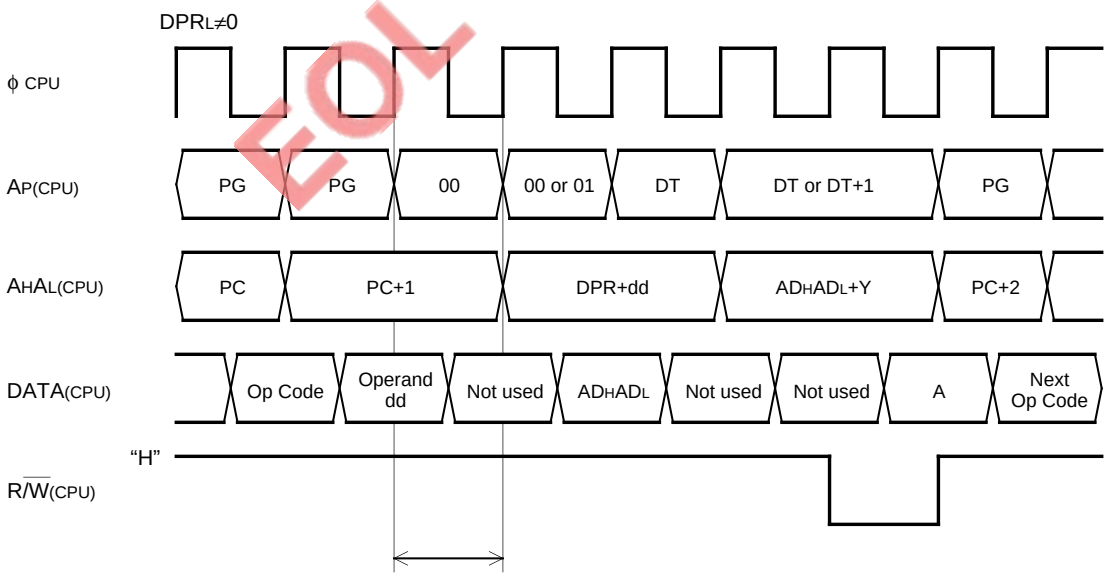
Timing :



When DPRL=0, this cycle is nothing.

Instruction : STA

Timing :

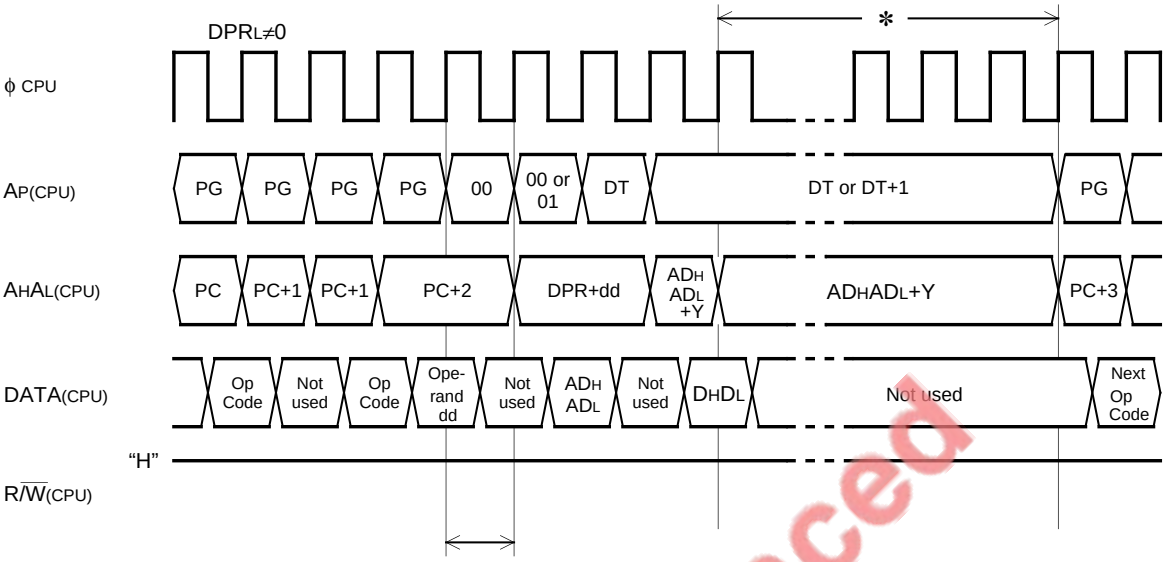


When DPRL=0, this cycle is nothing.

EOL announced

Instruction : DIV, DIVS*, MPY, MPYS*

Timing :



When DPRL=0, this cycle is nothing.

(Note) The cycle number during ✻ is shown in following table

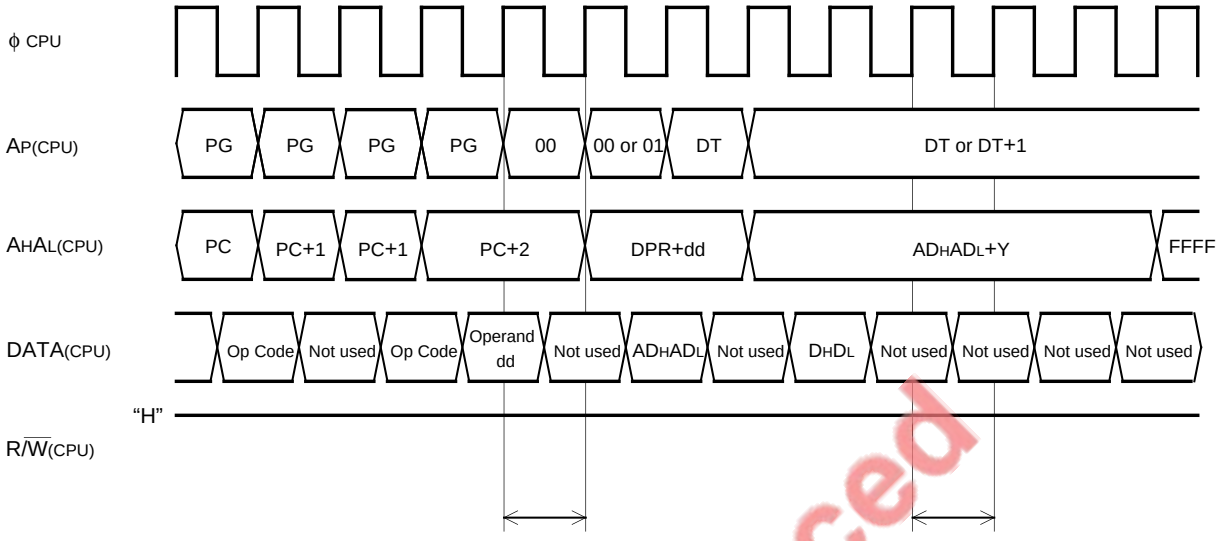
Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus×Plus	22	14
	Minus×Minus		
	Plus×Minus	25	17
	Minus×Plus		

- The contents of AHAL(CPU) during ✻ of the DIVS instruction is undefined.
- When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during ✻ is undefined.

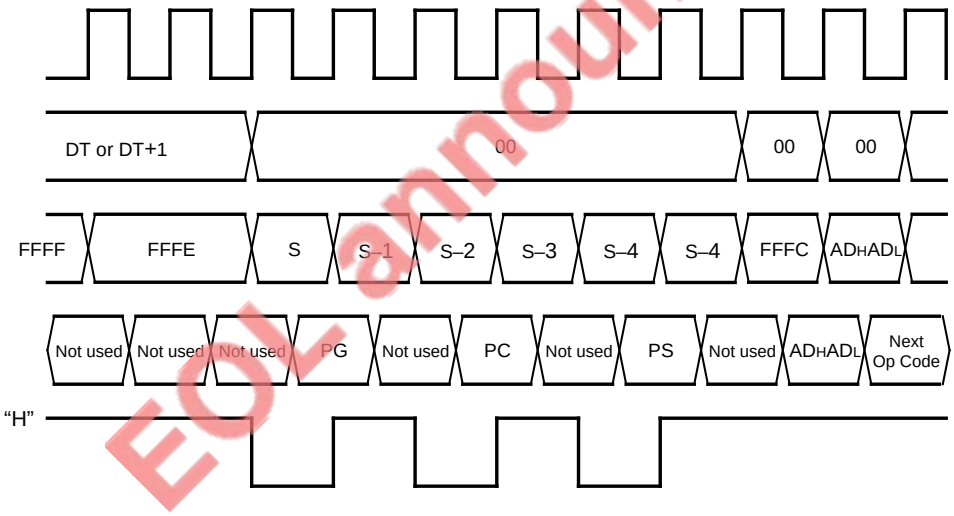
EOL announced

Instruction : DIV, DIVS* (case of 0 division)

Timing :



When DPRL =0, this cycle is nothing. At DIVS instruction, this cycle is nothing.

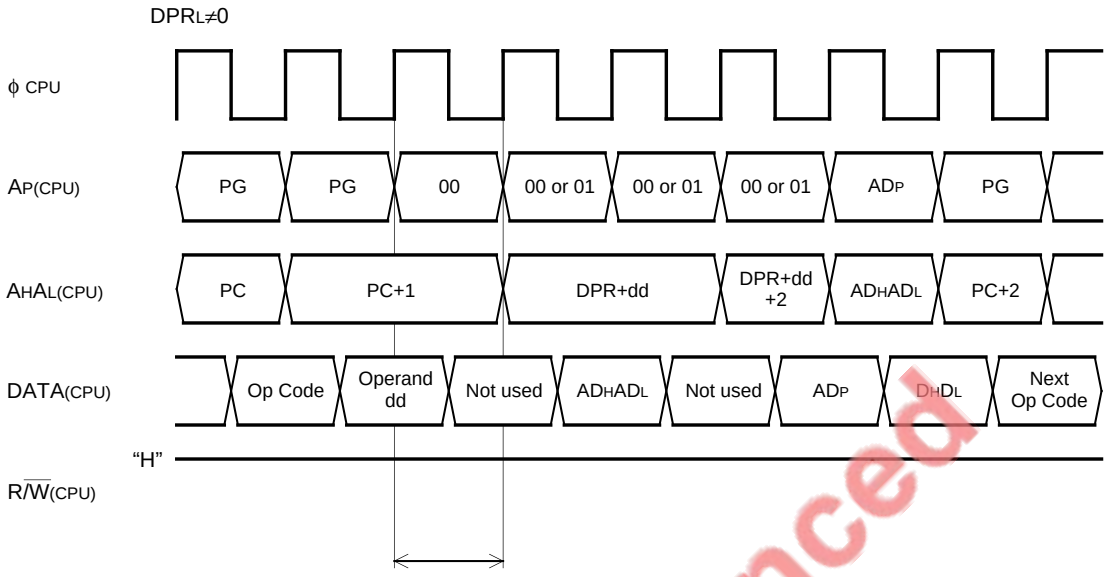


EOL announced

Direct Indirect Long

Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

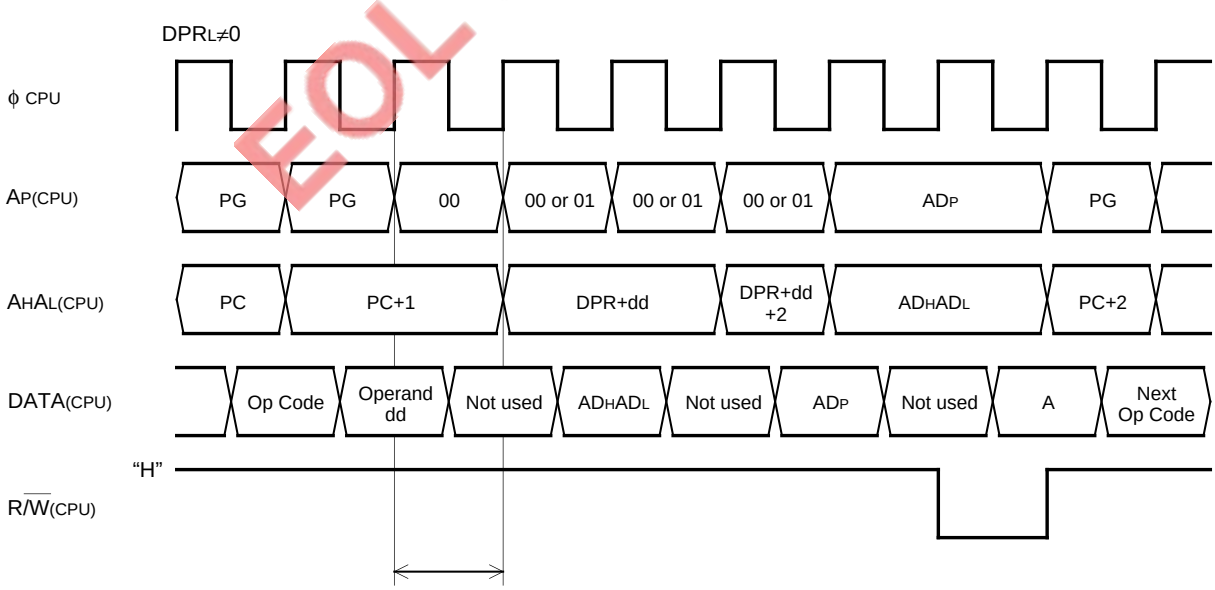
Timing :



When DPRL=0, this cycle is nothing.

Instruction : STA

Timing :



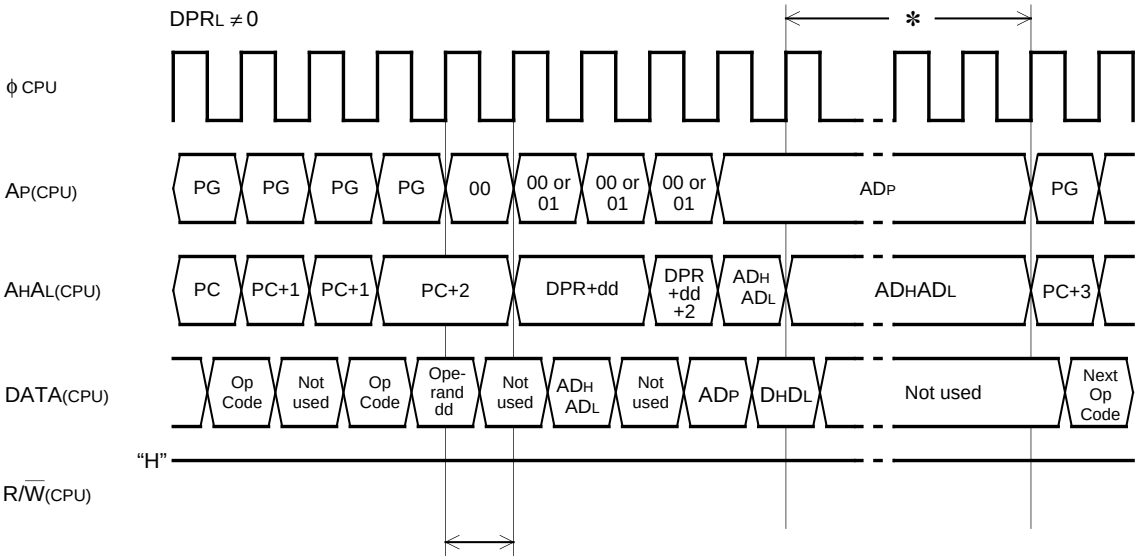
When DPRL=0, this cycle is nothing.

EOL announced

EOL announced

Instruction : DIV, DIVS*, MPY, MPYS*

Timing :



When DPRL=0, this cycle is nothing.

(Note) The cycle number during ✻ is shown in following table

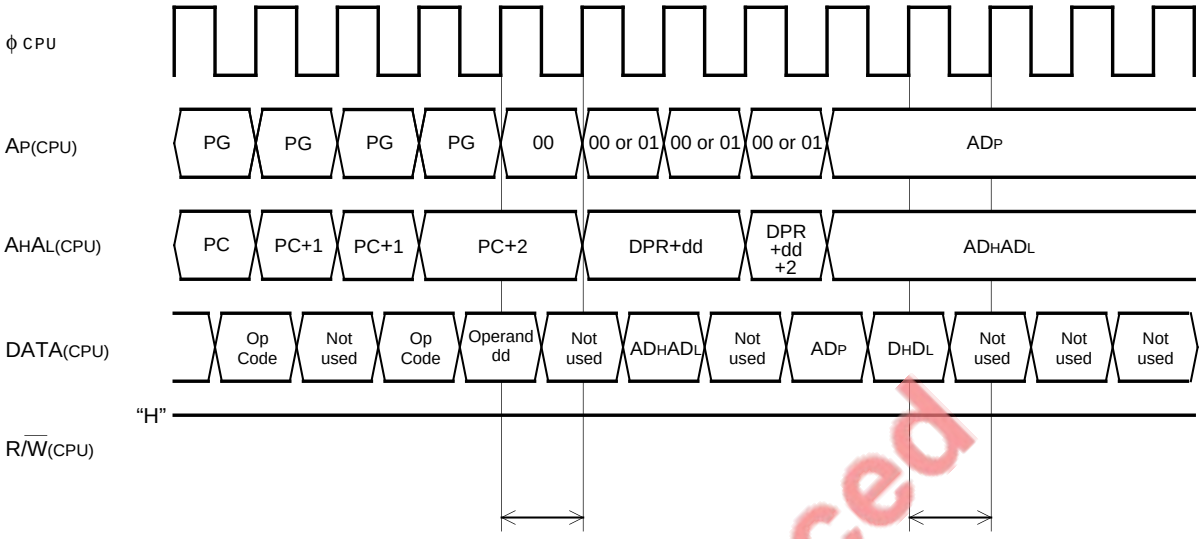
Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus×Plus	22	14
	Minus×Minus		
	Plus×Minus	25	17
	Minus×Plus		

- The contents of AHAL(CPU) during ✻ of the DIVS instruction is undefined.
- When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during ✻ is undefined.

Direct Indirect Long

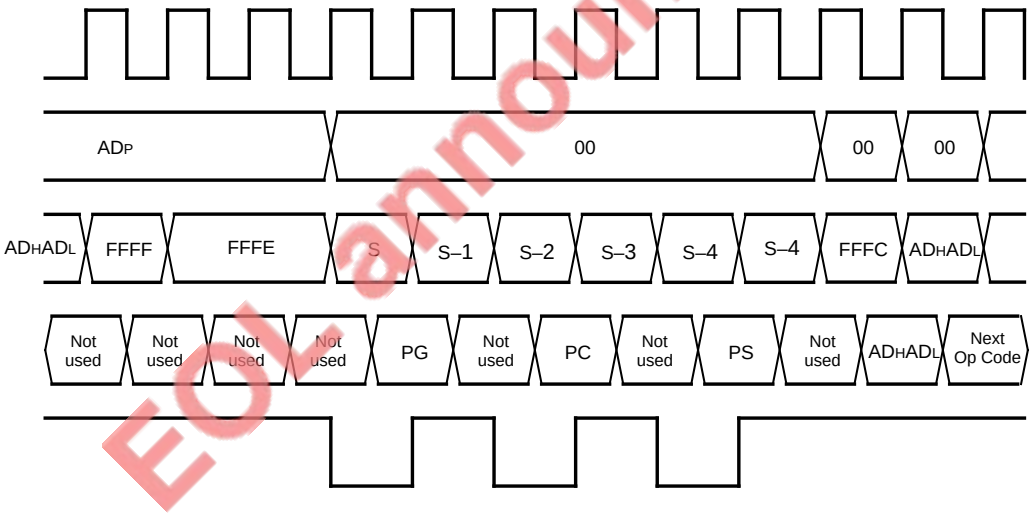
Instruction : DIV, DIVS* (case of 0 division)

Timing :



When DPRL=0, this cycle is nothing.

At DIVS instruction, this cycle is nothing.

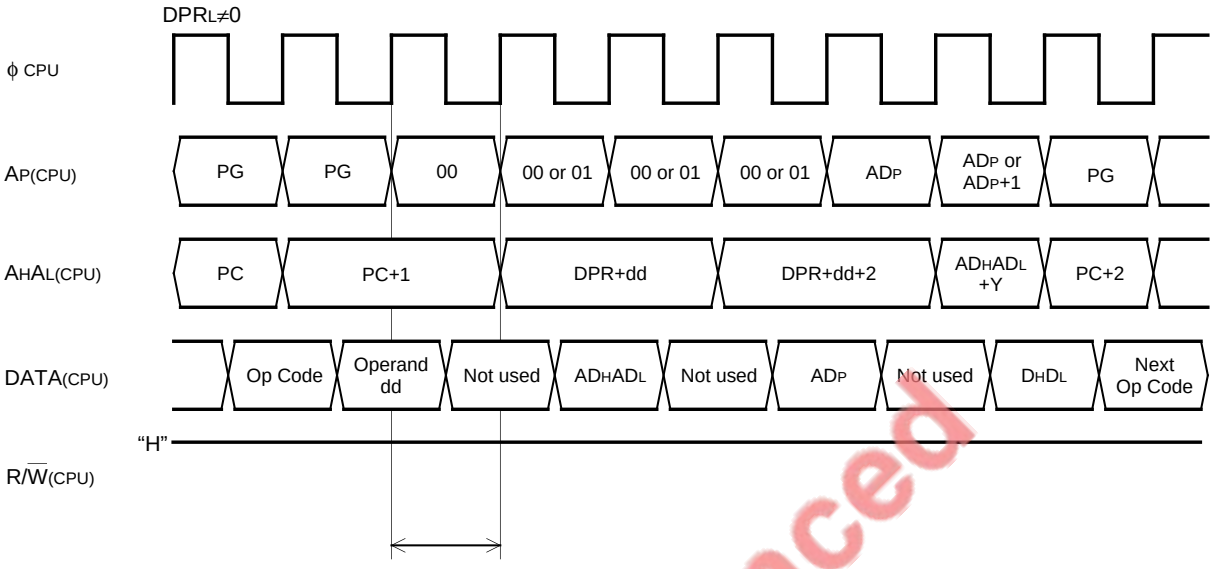


EOL announced

Direct Indirect Long Indexed Y

Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

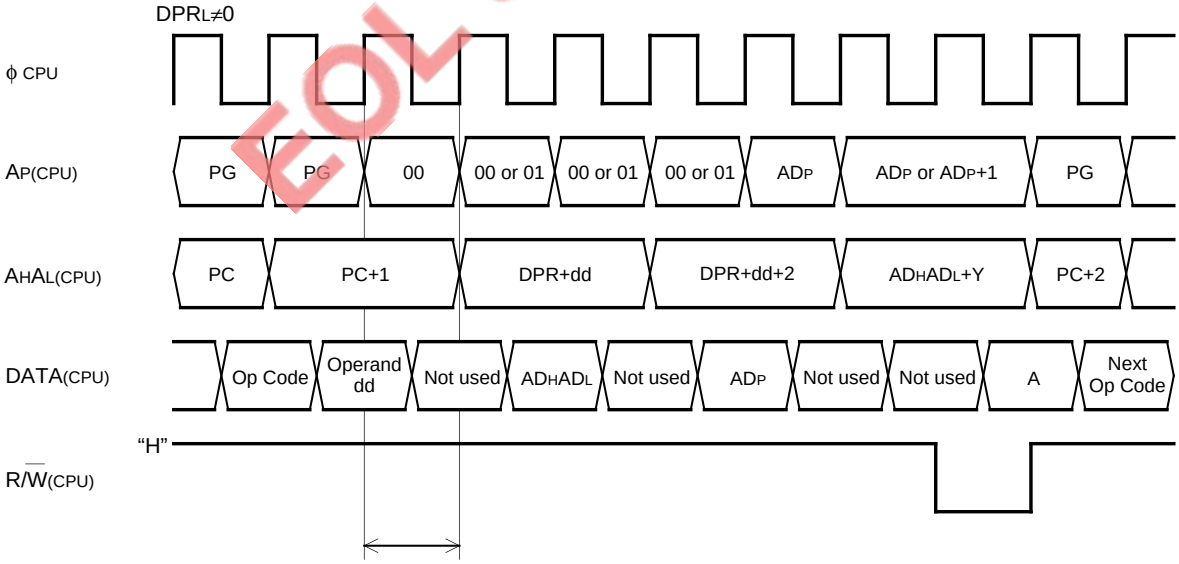
Timing :



When DPRL=0, this cycle is nothing.

Instruction : STA

Timing :



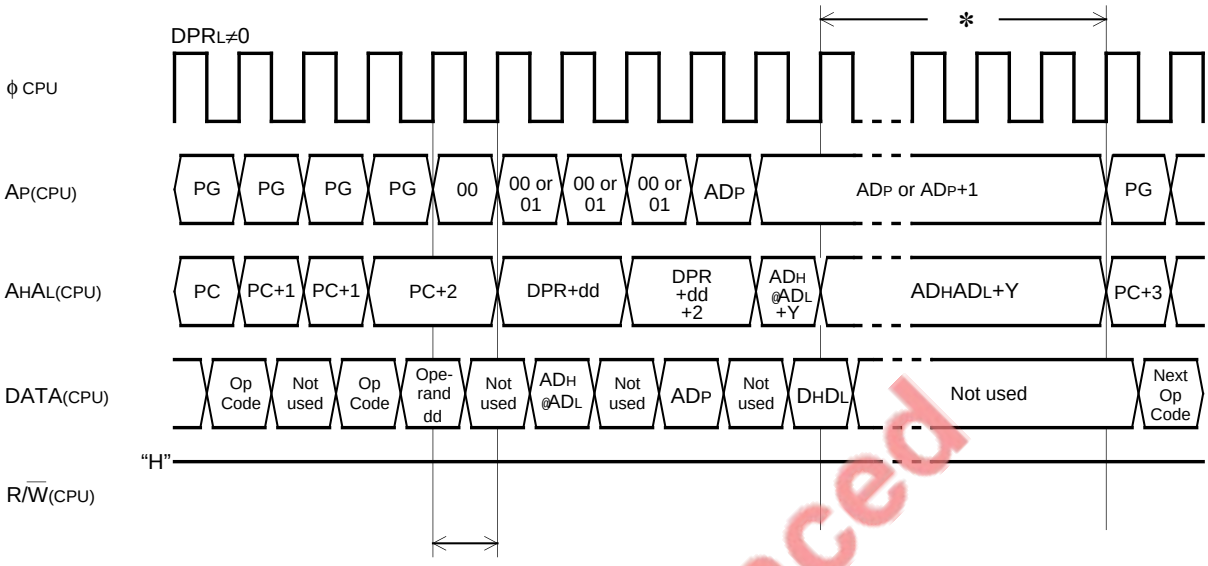
When DPRL=0, this cycle is nothing.

EOL announced

Direct Indirect Long Indexed Y

Instruction : DIV, DIVS*, MPY, MPYS*

Timing :



When DPRL=0, this cycle is nothing.

(Note) The cycle number during ✻ is shown in following table

Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus×Plus	22	14
	Minus×Minus		
	Plus×Minus	25	17
	Minus×Plus		

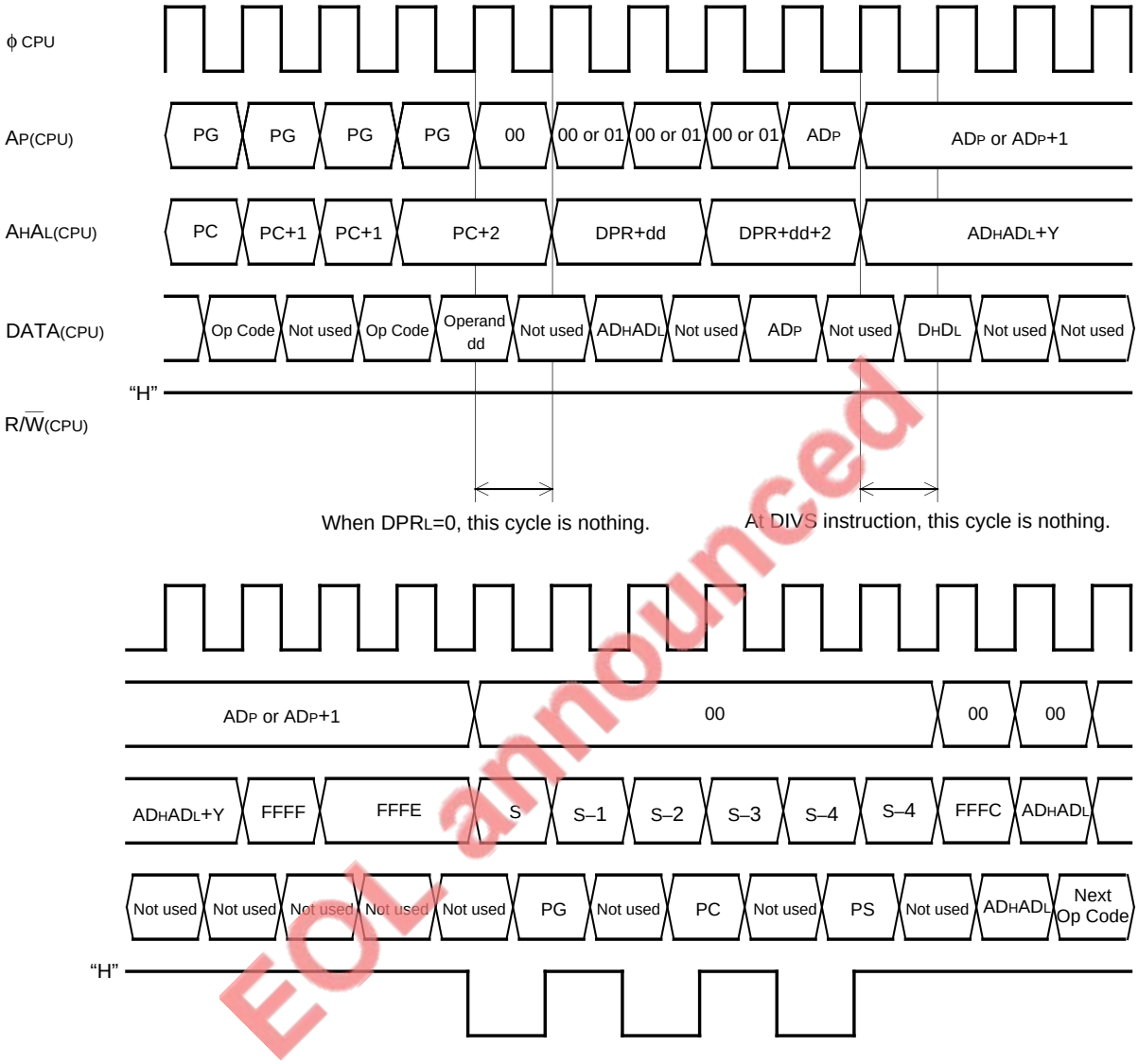
- The contents of AHAL(CPU) during ✻ of the DIVS instruction is undefined.
- When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during ✻ is undefined.

EOL announced

Direct Indirect Long Indexed Y

Instruction : DIV, DIVS* (case of 0 division)

Timing :

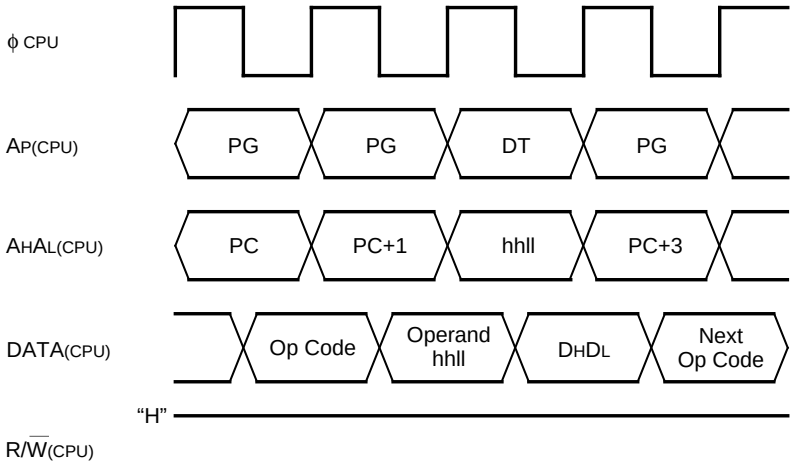


EOL announced

Absolute

Instruction : ADC, AND, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA, SBC

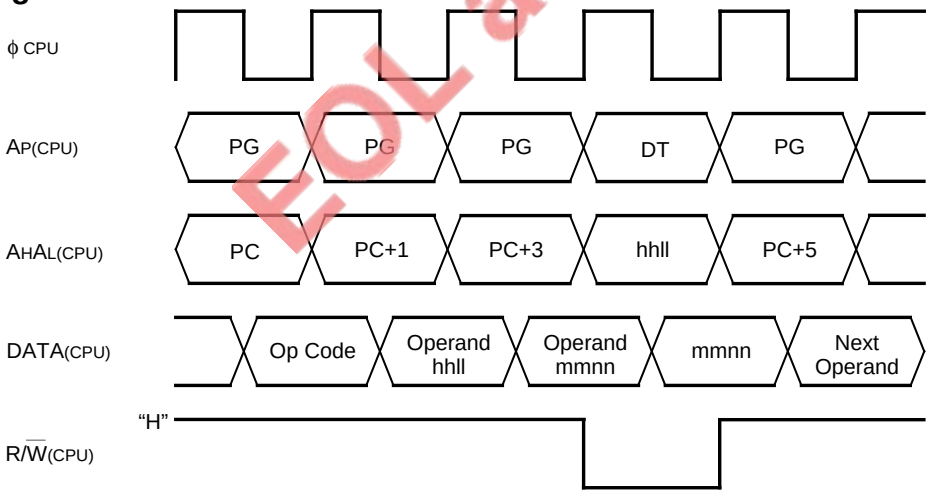
Timing :



EOL announced

Instruction : LDM

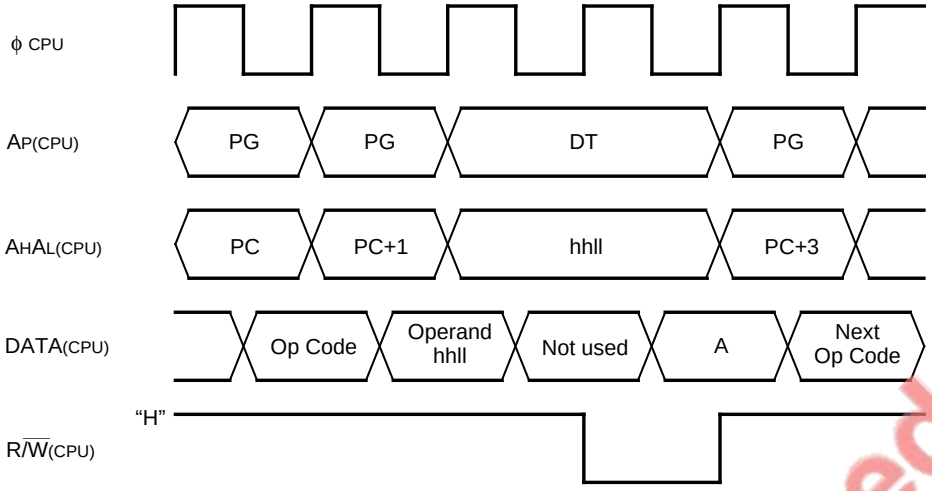
Timing :



When m=1, fetched operand at 3-rd cycle and data at 4-th cycle are 1-byte (nn).

Instruction : STA, STX, STY

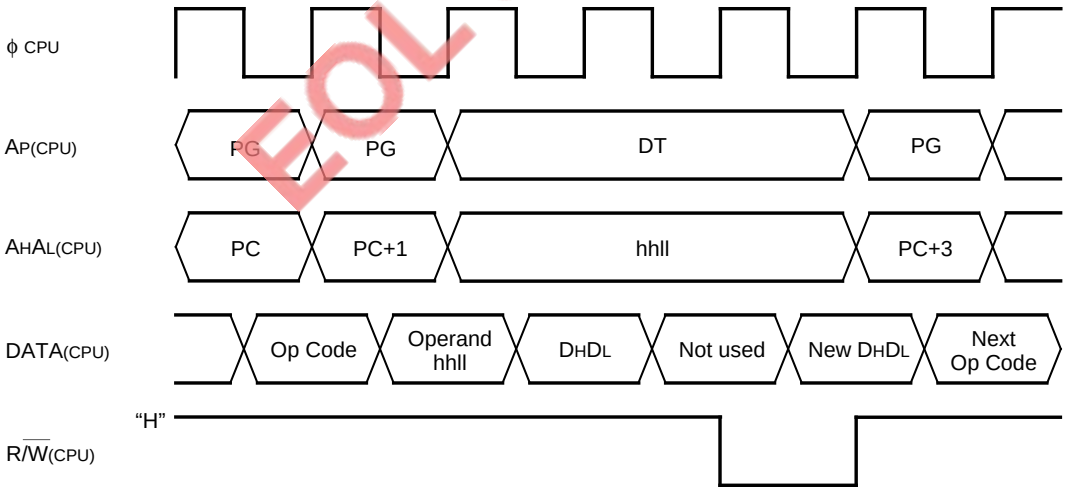
Timing :



EOL announced

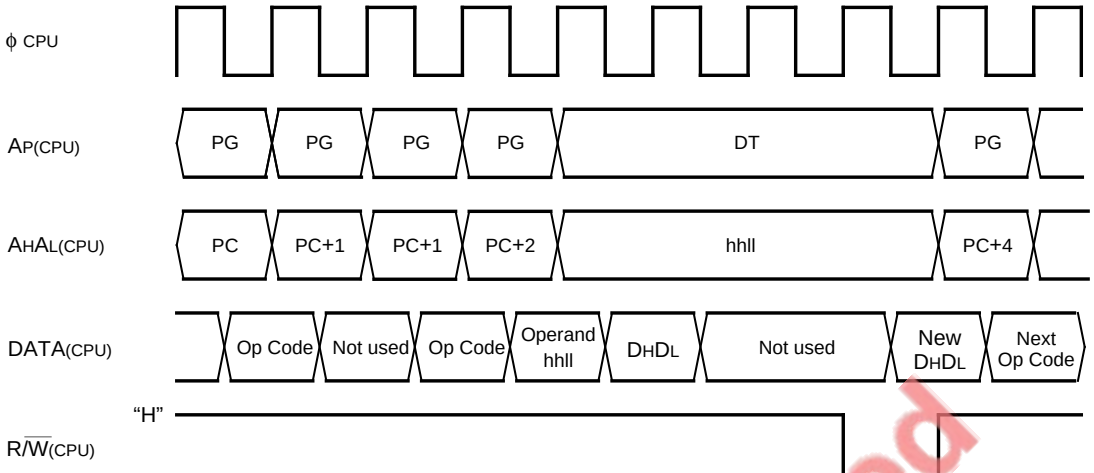
Instruction : ASL, DEC, INC, LSR, ROL, ROR

Timing :



Instruction : ASR*

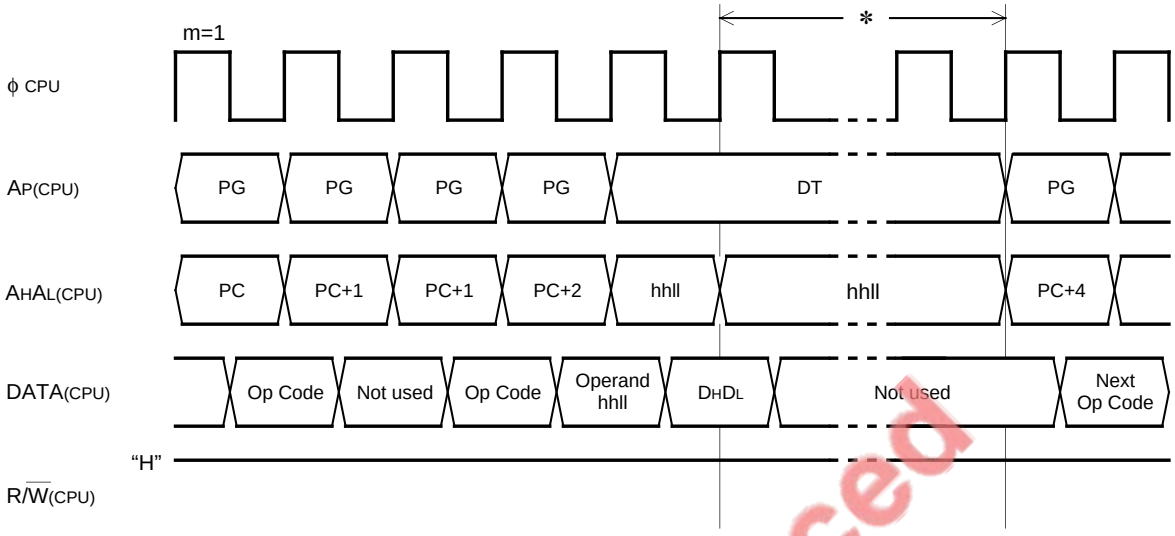
Timing :



EOL announced

Instruction : DIV, DIVS*, MPY, MPYS*

Timing :



(Note) The cycle number during ✻ is shown in following table

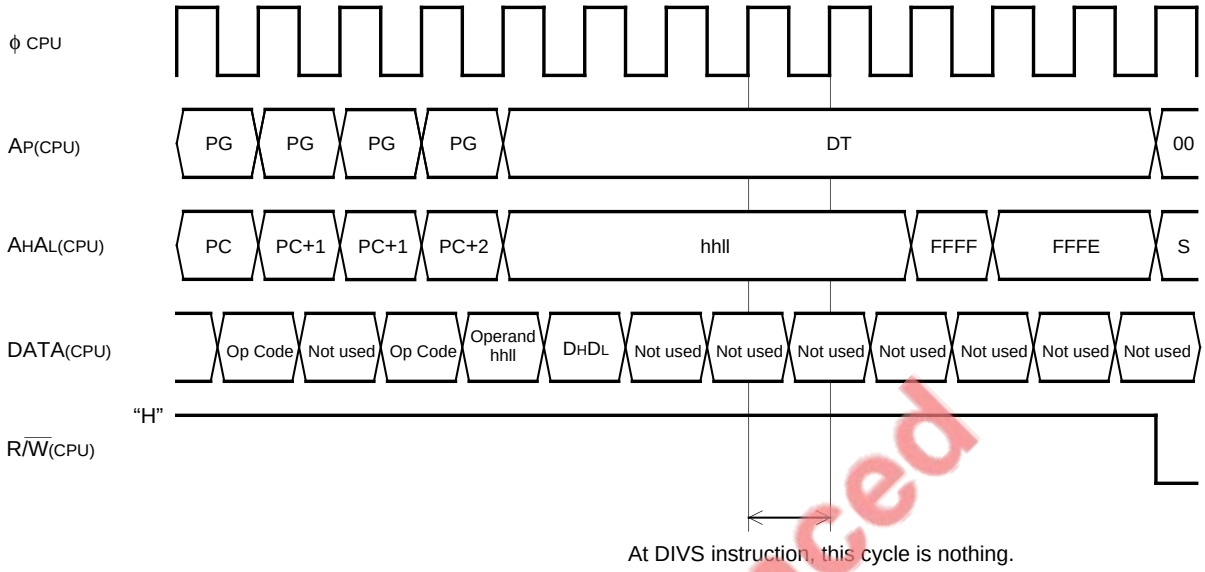
Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus×Plus	22	14
	Minus×Minus		
	Plus×Minus	25	17
	Minus×Plus		

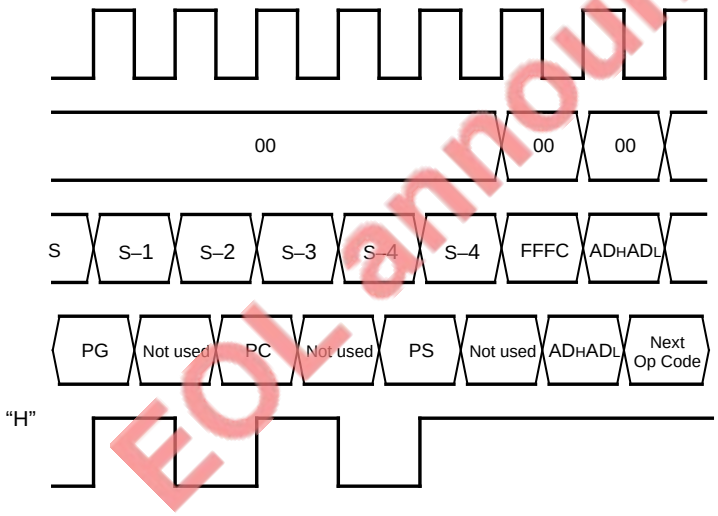
- The contents of AHAL(CPU) during ✻ of the DIVS instruction is undefined.
- When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during ✻ is undefined.

EOL announced

Instruction : DIV, DIVS* (case of 0 division)

Timing :

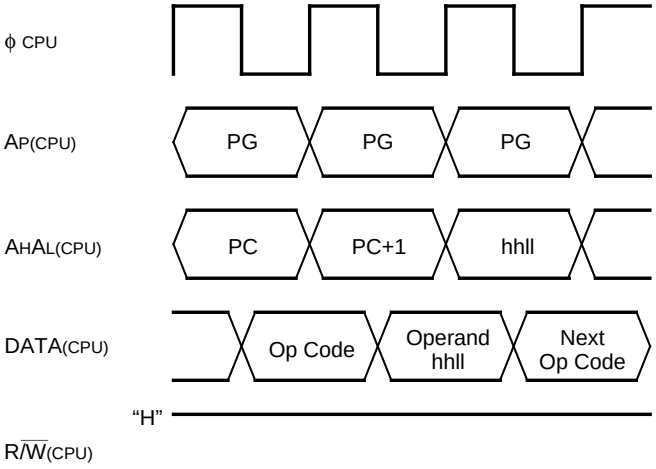




EOL announced

Instruction : JMP

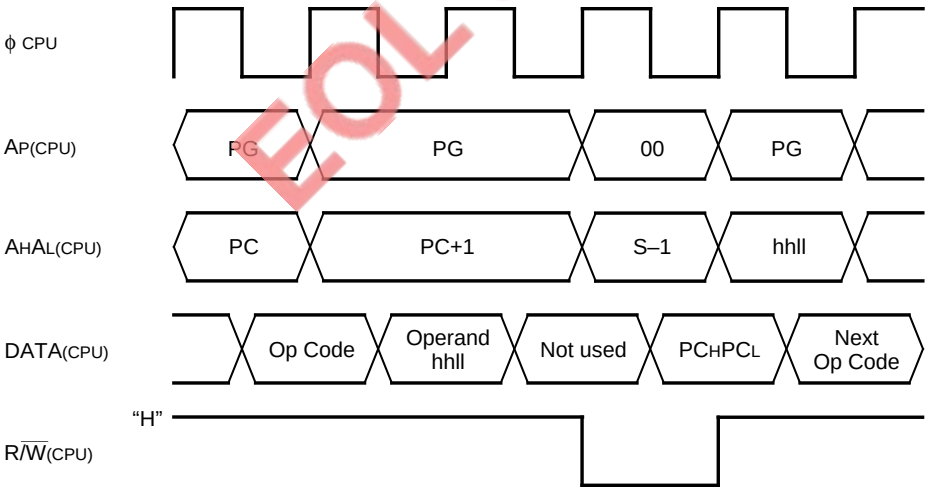
Timing :



EOL announced

Instruction : JSR

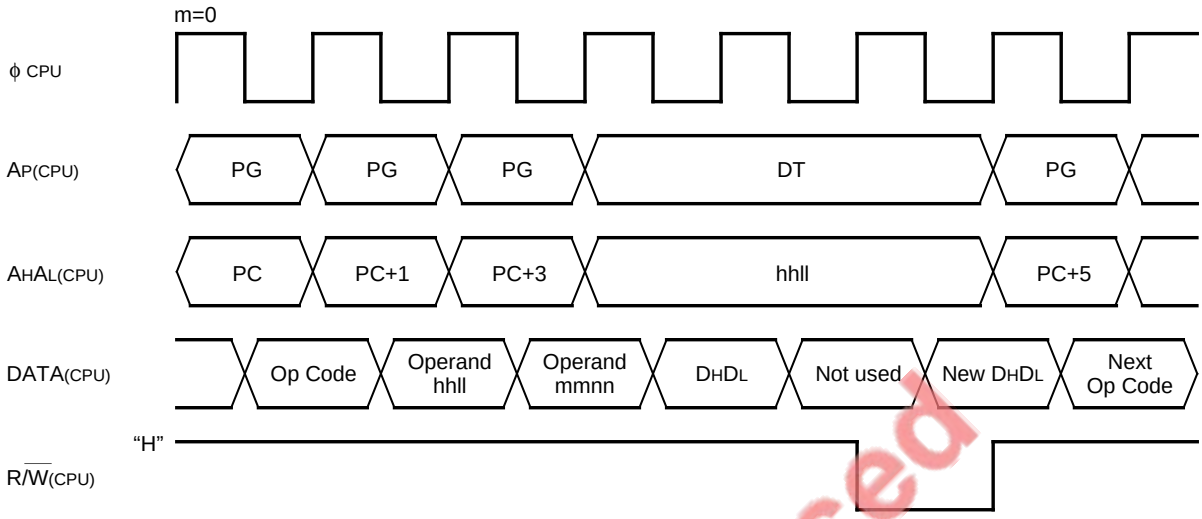
Timing :



Absolute Bit

Instruction : CLB, SEB

Timing :



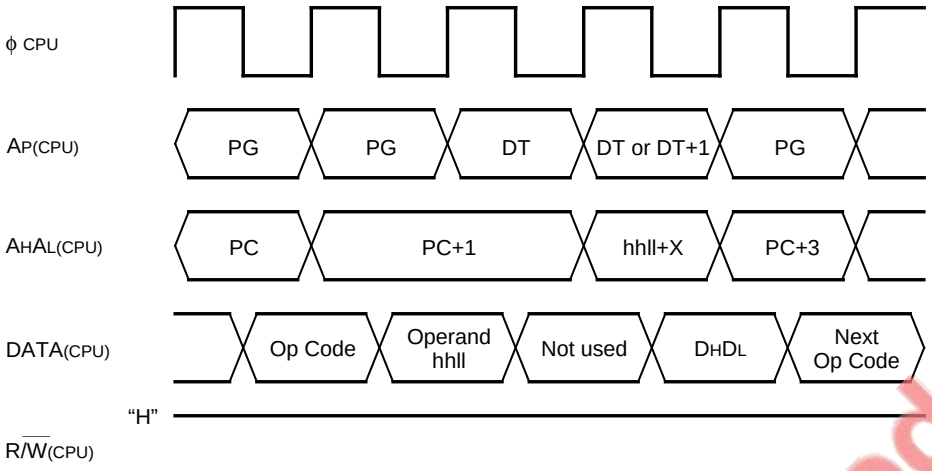
When m=1, fetched operand at 3-rd cycle is 1-byte (nn).

EOL announced

Absolute Indexed X

Instruction : ADC, AND, CMP, EOR, LDA, LDY, ORA, SBC

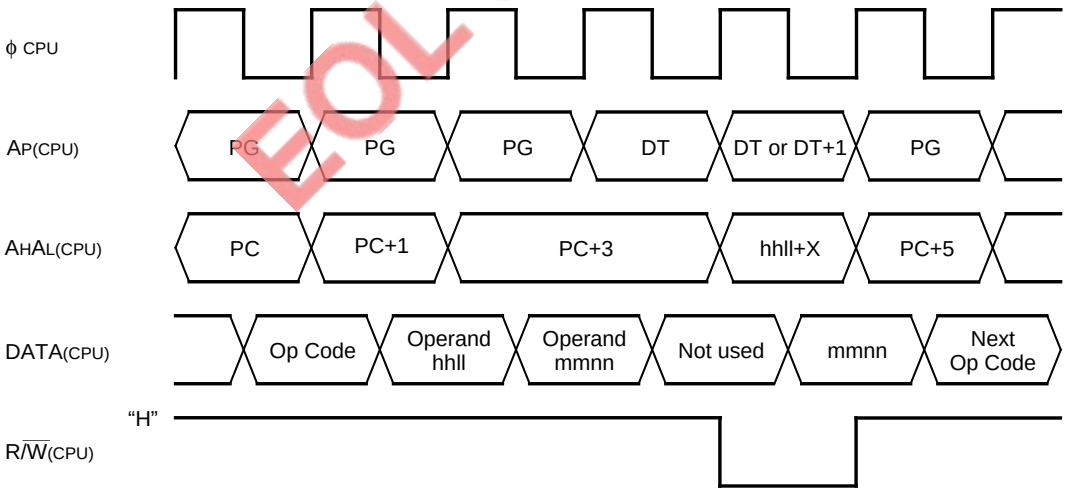
Timing :



EOL announced

Instruction : LDM

Timing :

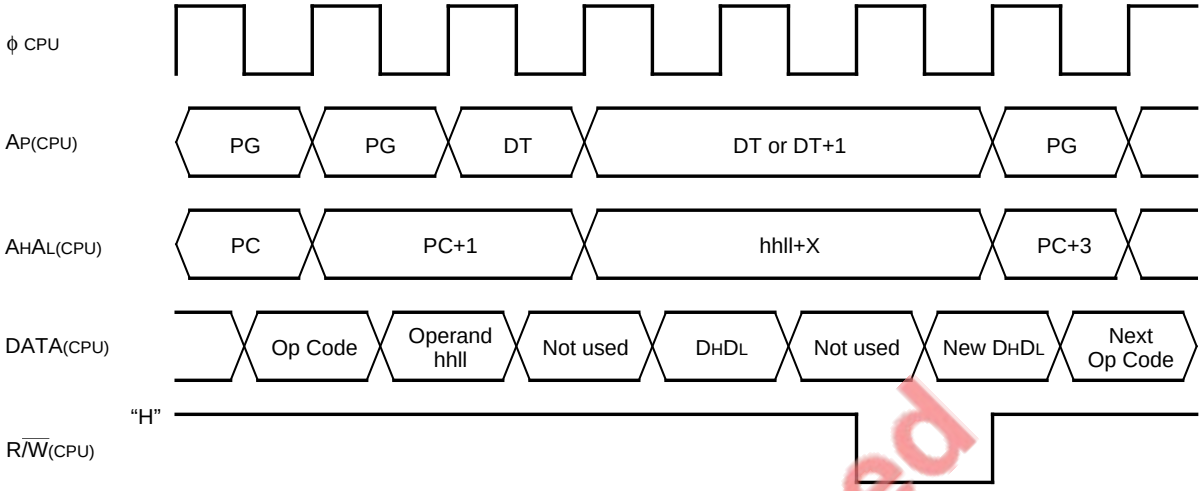


When m=1, fetched operand at 3-rd cycle and data at 5-th cycle are 1-byte (nn).

Absolute Indexed X

Instruction : ASL, DEC, INC, LSR, ROL, ROR

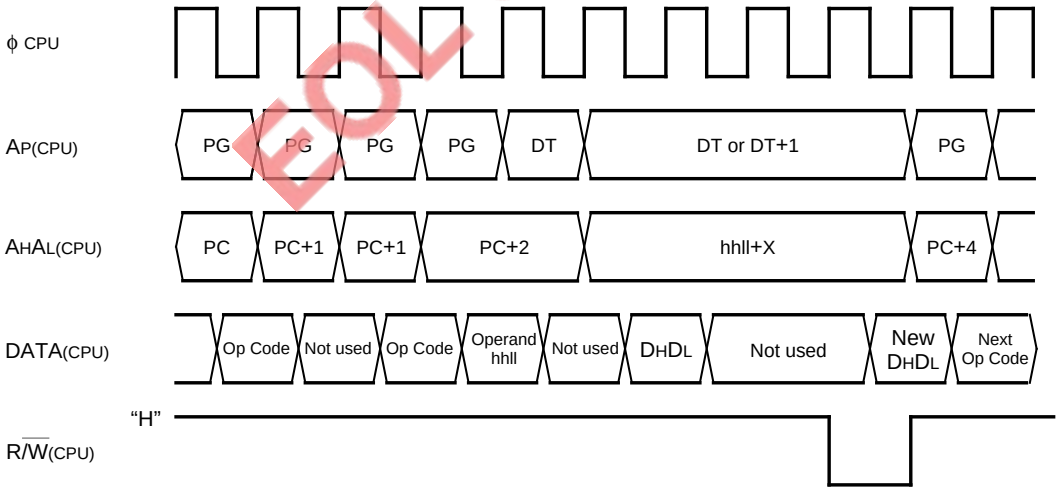
Timing :



EOL announced

Instruction : ASR*

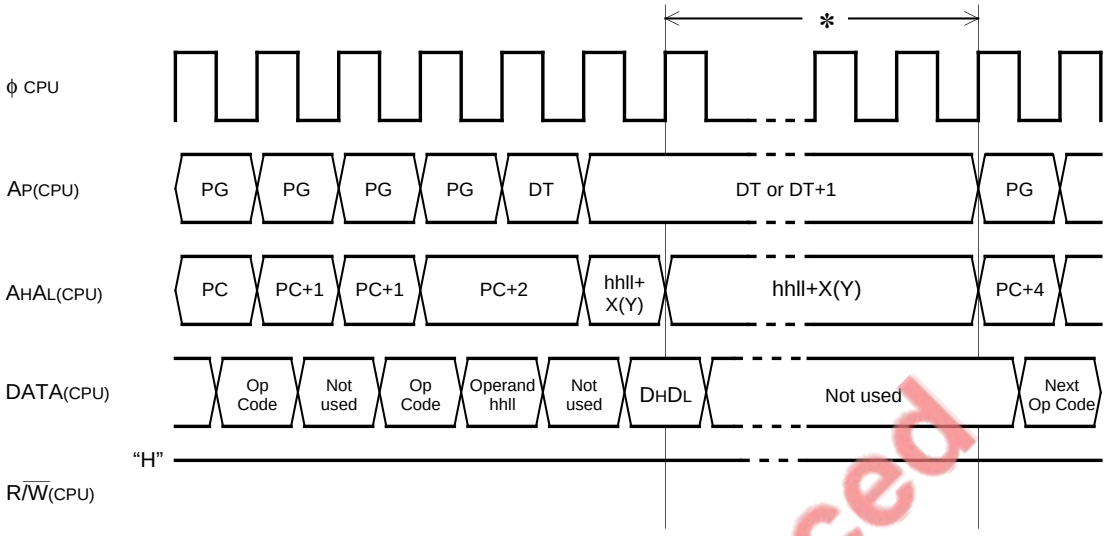
Timing :



Absolute Indexed X
Absolute Indexed Y

Instruction : DIV, DIVS*, MPY, MPYS*

Timing :



(Note) The cycle number during ✻ is shown in following table

Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus×Plus	22	14
	Minus×Minus		
	Plus×Minus	25	17
	Minus×Plus		

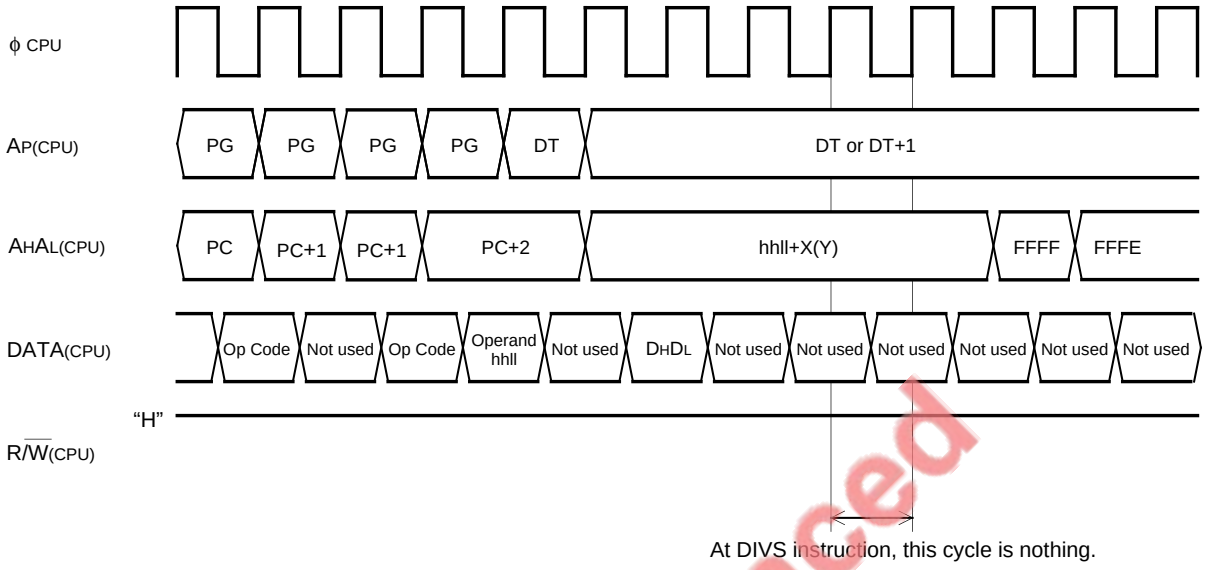
- The contents of AHAL(CPU) during ✻ of the DIVS instruction is undefined.
- When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during ✻ is undefined.

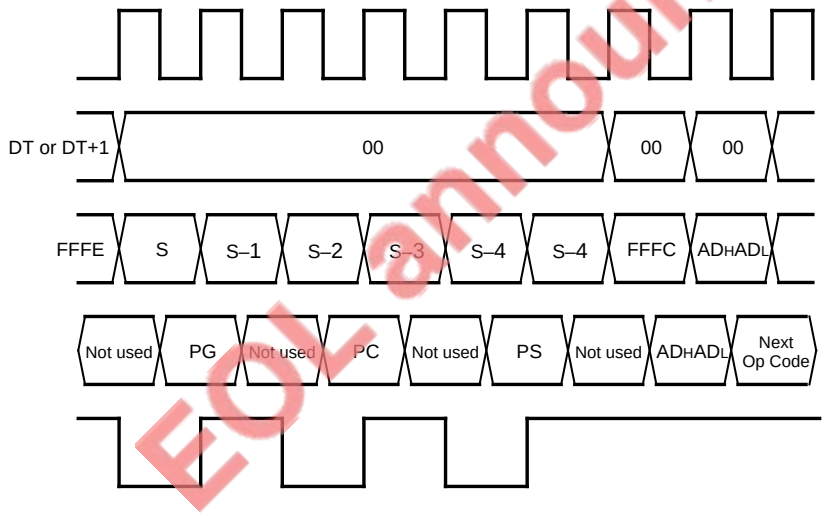
EOL announced

Absolute Indexed X
Absolute Indexed Y

Instruction : DIV, DIVS* (case of 0 division)

Timing :



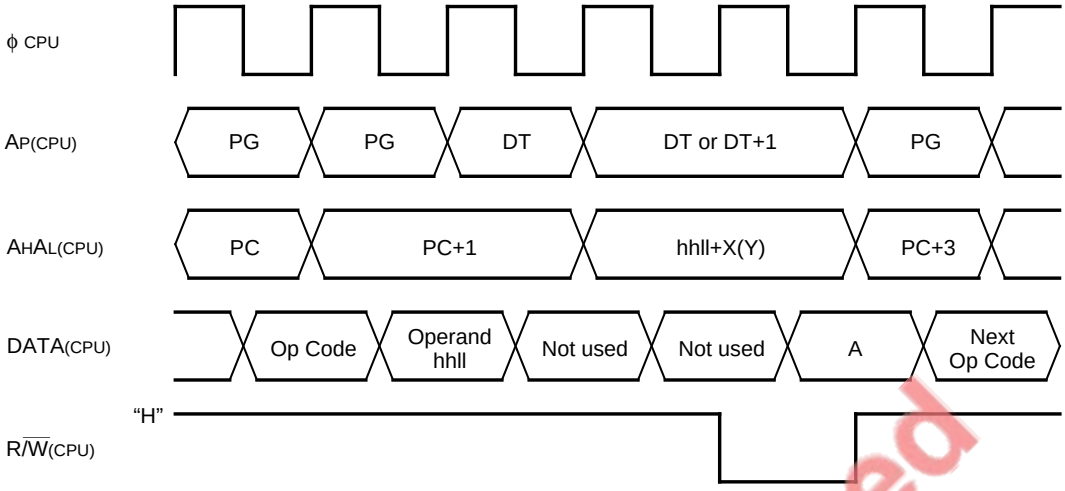


EOL announced

Absolute Indexed X
Absolute Indexed Y

Instruction : STA

Timing :

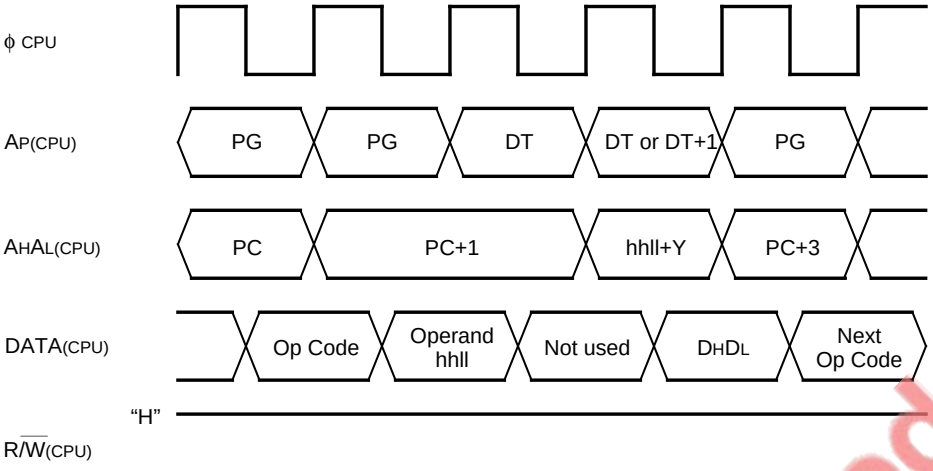


EOL announced

Absolute Indexed Y

Instruction : ADC, AND, CMP, EOR, LDA, LDX, ORA, SBC

Timing :

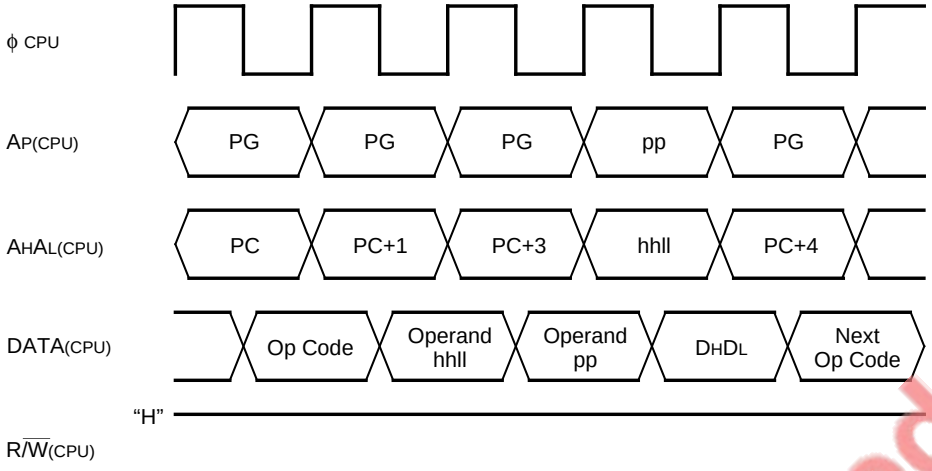


EOL announced

Absolute Long

Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

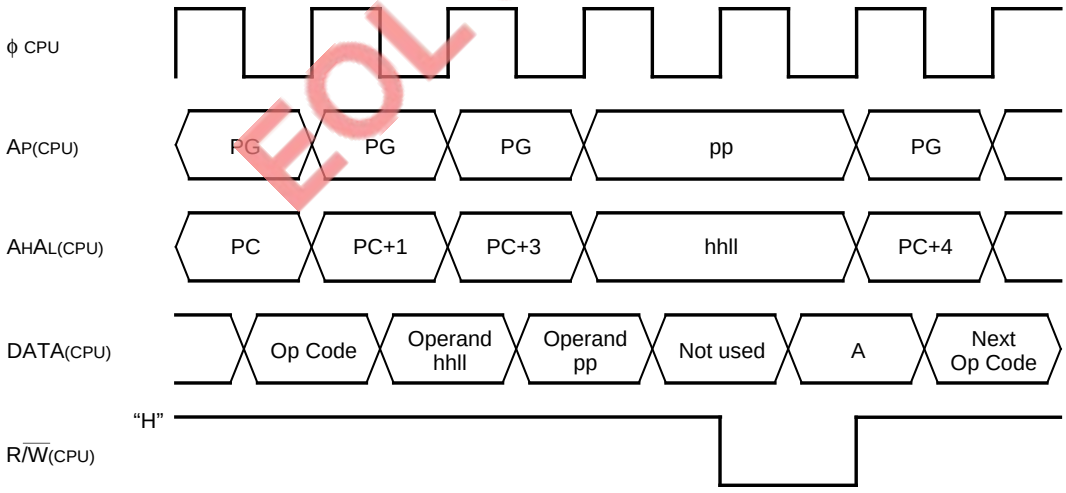
Timing :



EOL announced

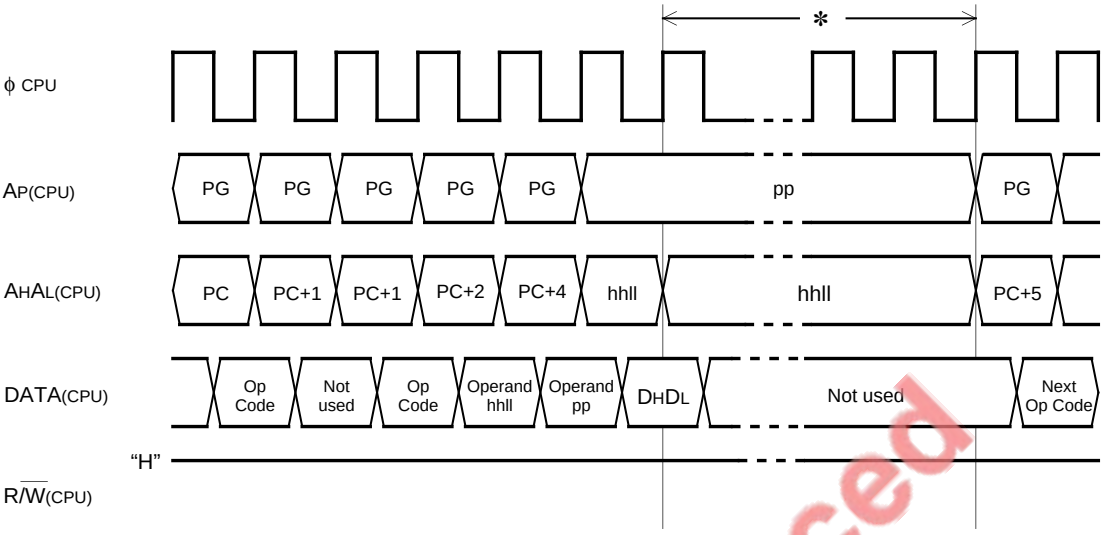
Instruction : STA

Timing :



Instruction : DIV, DIVS*, MPY, MPYS*

Timing :



(Note) The cycle number during ✻ is shown in following table

Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus×Plus	22	14
	Minus×Minus		
	Plus×Minus	25	17
	Minus×Plus		

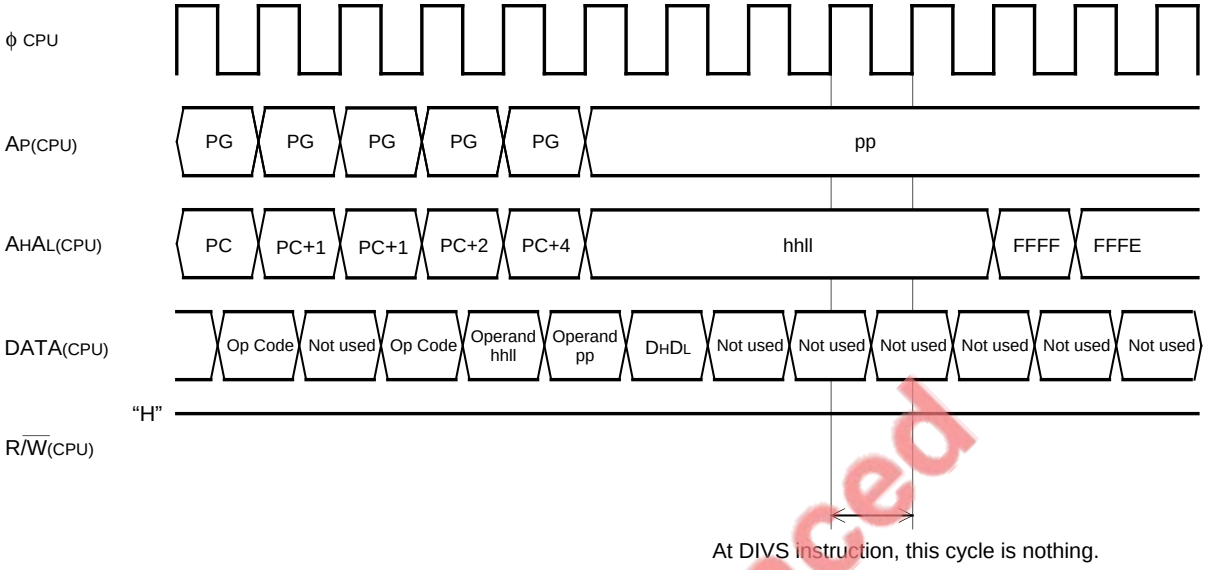
• The contents of AHAL(CPU) during ✻ of the DIVS instruction is undefined.
• When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during ✻ is undefined.

EOL announced

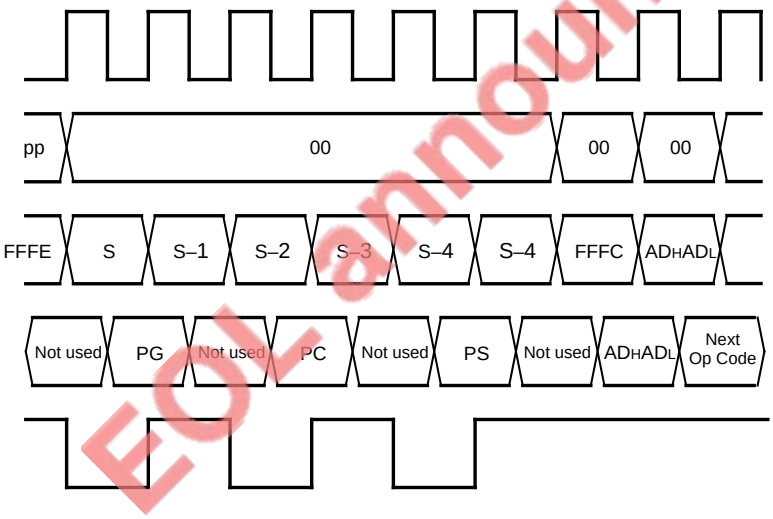
Absolute Long

Instruction : DIV , DIVS*(case of 0 division)

Timing :



At DIVS instruction, this cycle is nothing.

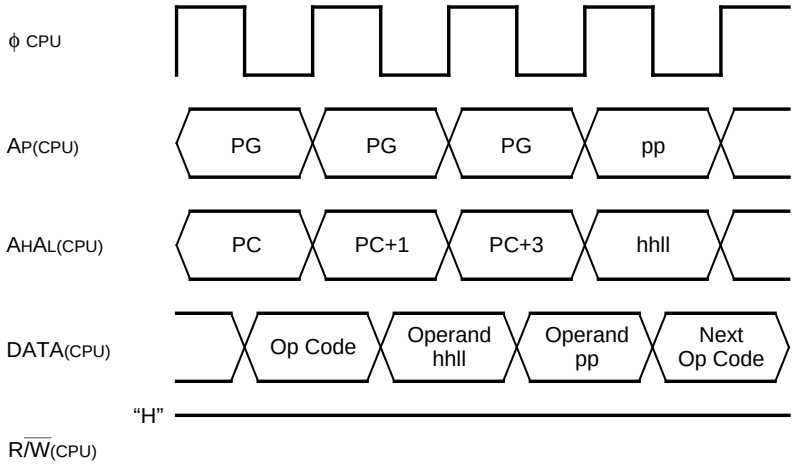


EOL announced

Absolute Long

Instruction : JMP

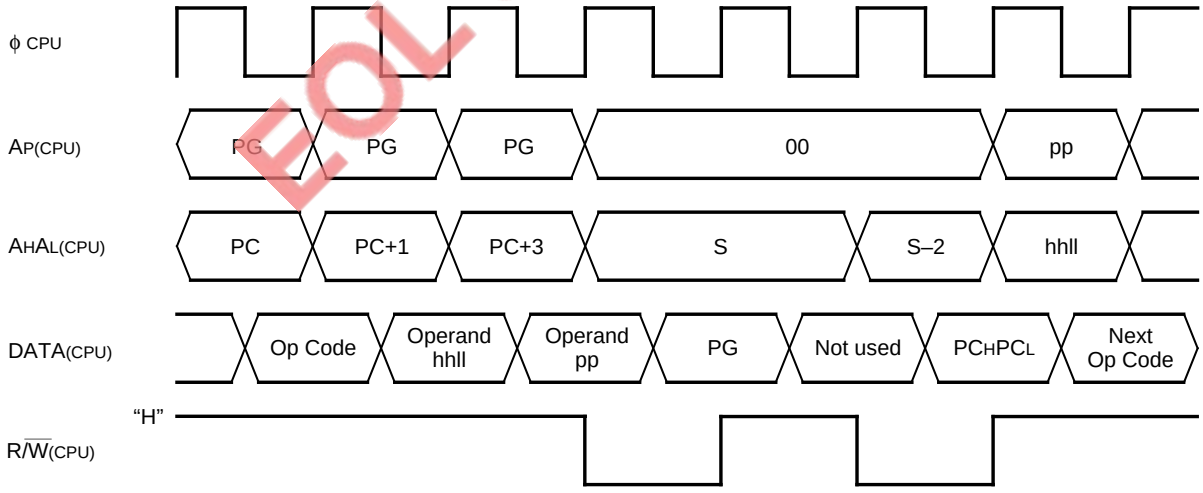
Timing :



EOL announced

Instruction : JSR

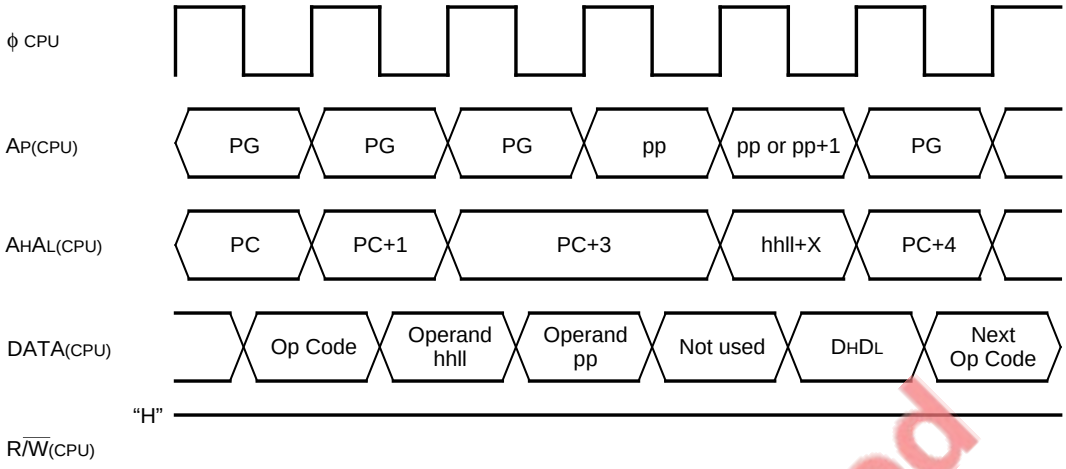
Timing :



Absolute Long Indexed X

Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

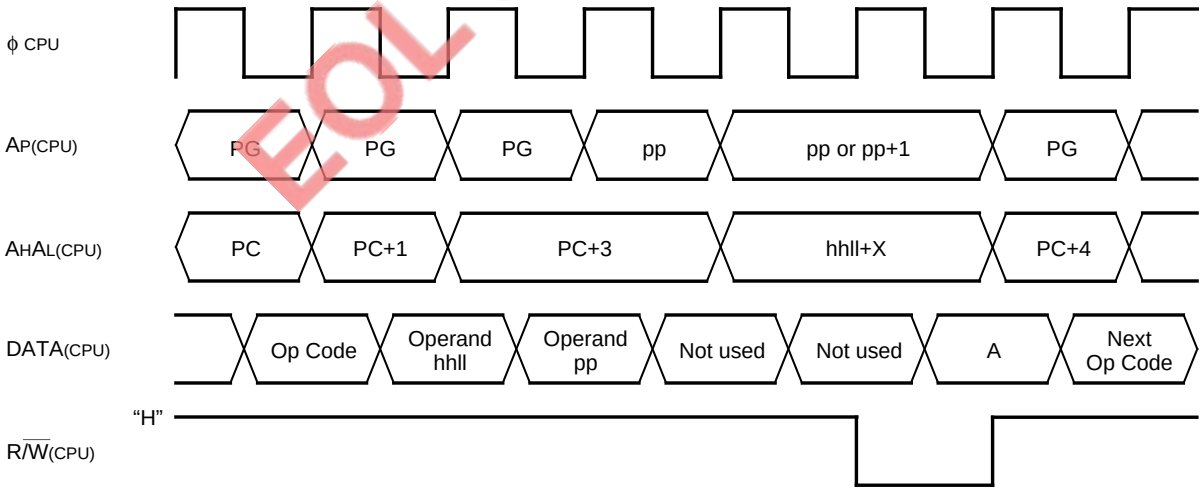
Timing :



EOL announced

Instruction : STA

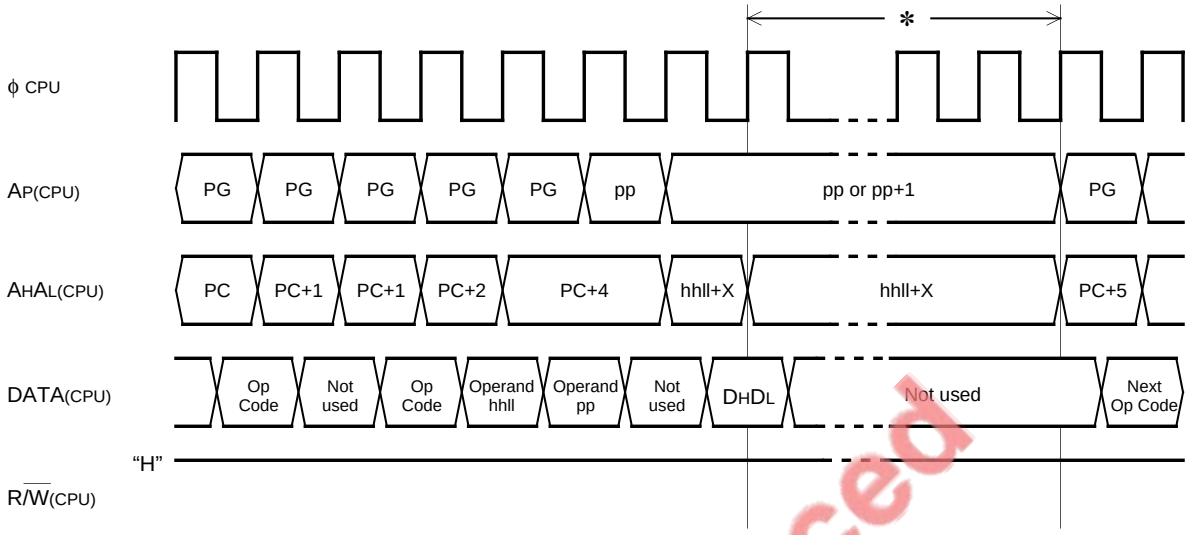
Timing :



Absolute Long Indexed X

Instruction : DIV, DIVS*, MPY, MPYS*

Timing :



(Note) The cycle number during ✻ is shown in following table

Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus×Plus	22	14
	Minus×Minus		
	Plus×Minus	25	17
	Minus×Plus		

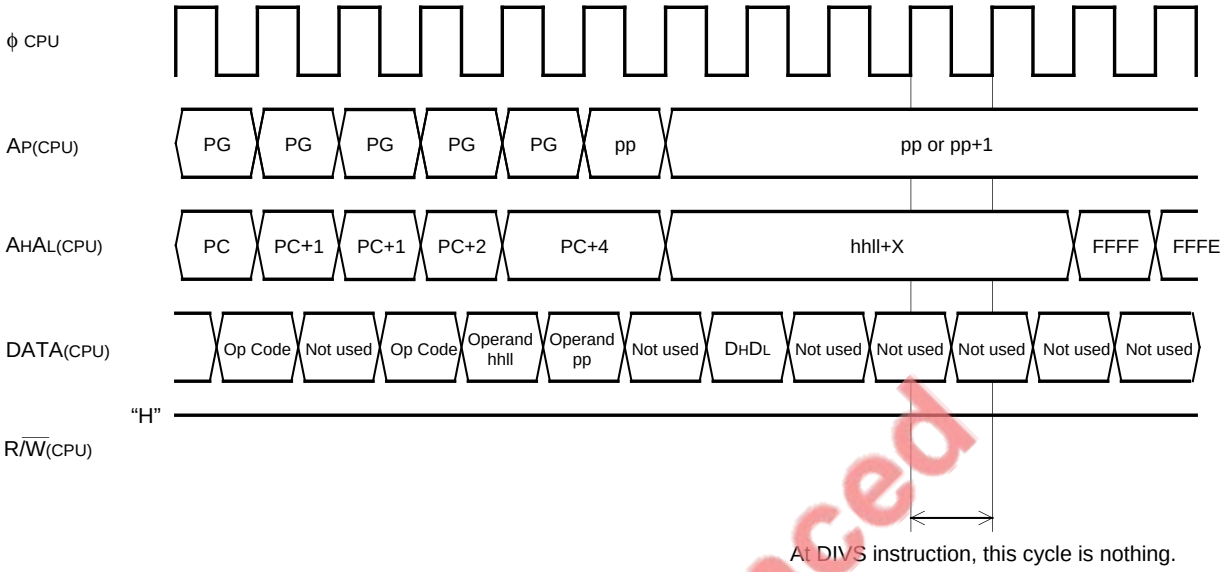
- The contents of AHAL(CPU) during ✻ of the DIVS instruction is undefined.
- When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during ✻ is undefined.

EOL announced

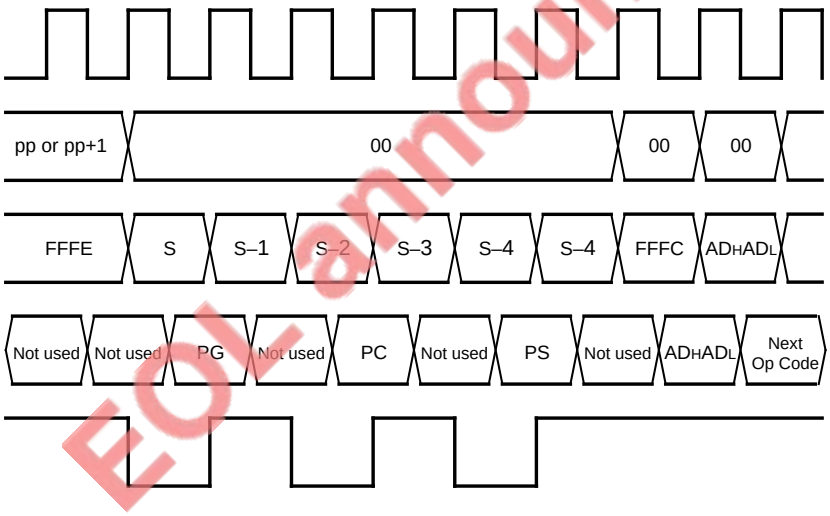
Absolute Long Indexed X

Instruction : DIV, DIVS* (case of 0 division)

Timing :



At DIVS instruction, this cycle is nothing.

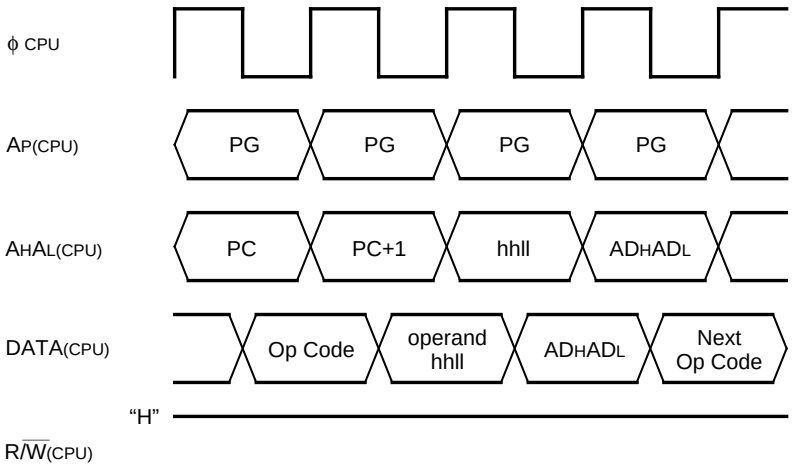


EOL announced

Absolute Indirect

Instruction : JMP

Timing :

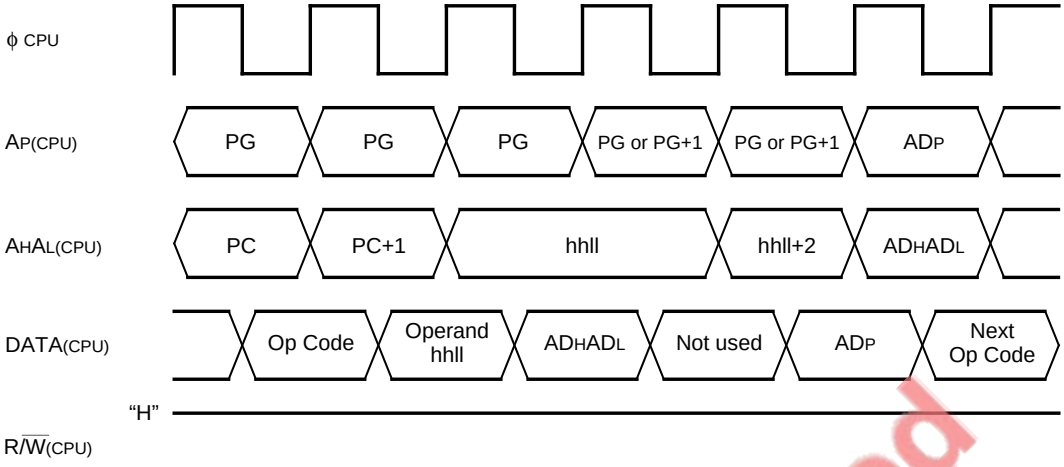


EOL announced

Absolute Indirect Long

Instruction : JMP

Timing :

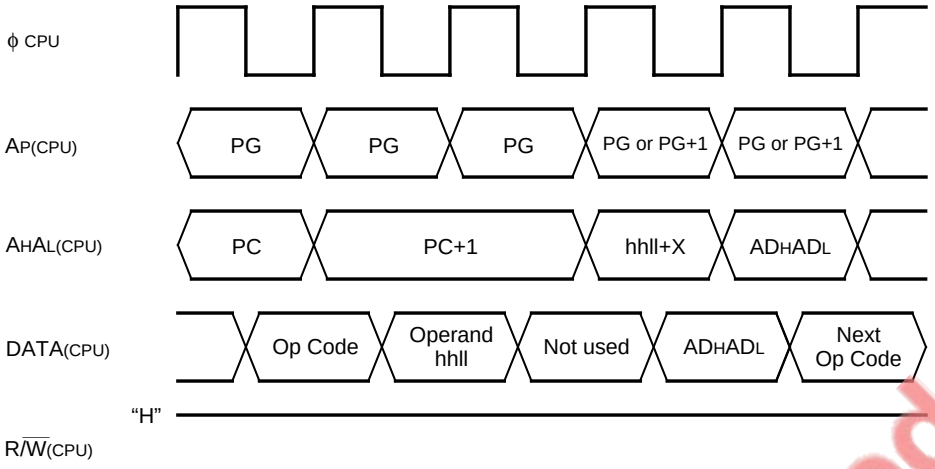


EOL announced

Absolute Indexed X Indirect

Instruction : JMP

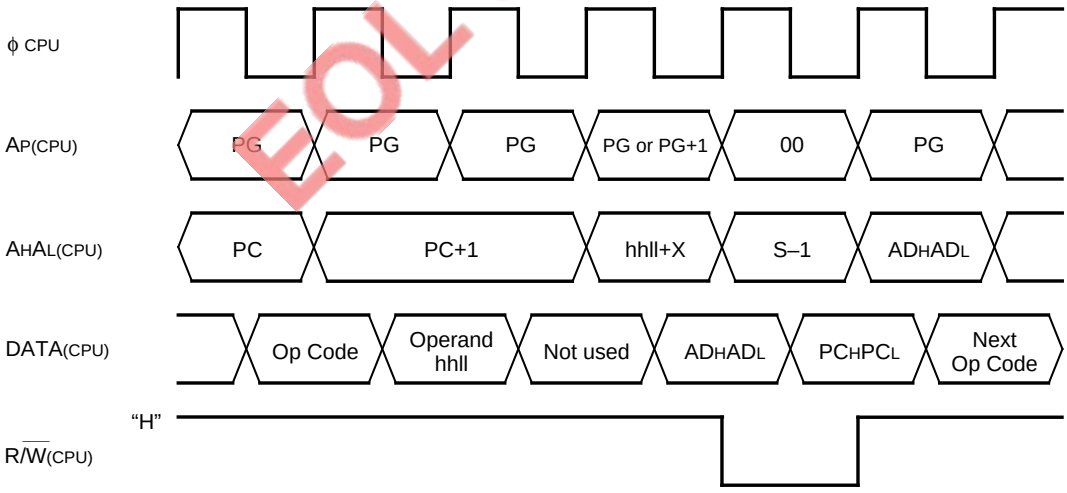
Timing :



EOL announced

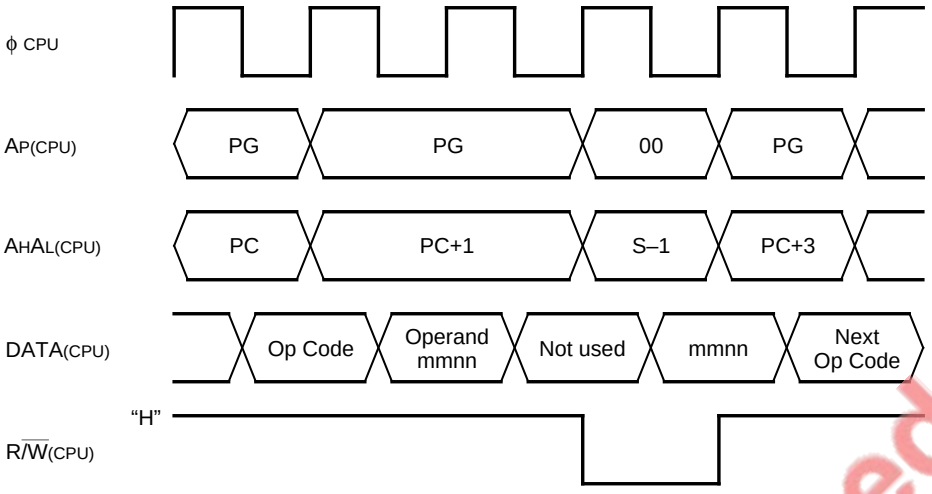
Instruction : JSR

Timing :



Instruction : PEA

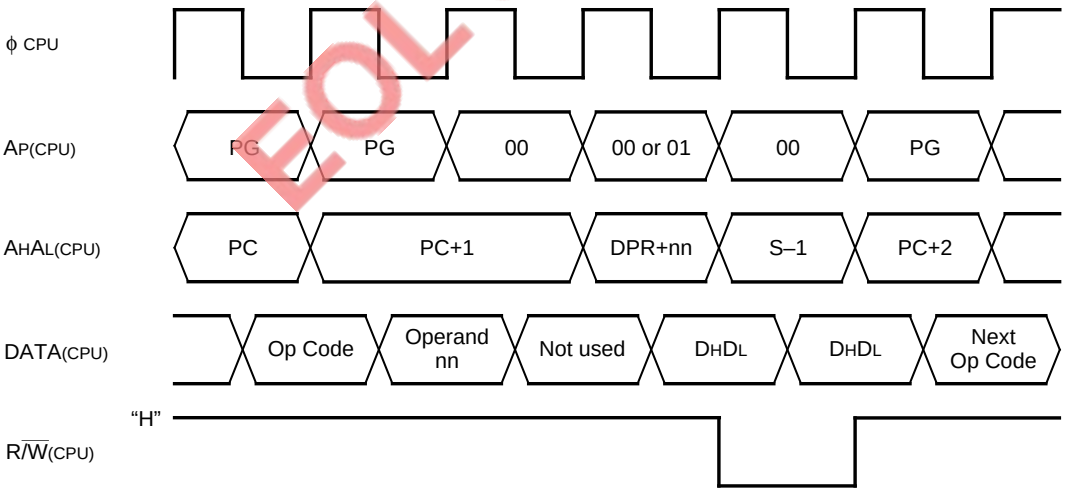
Timing :



EOL announced

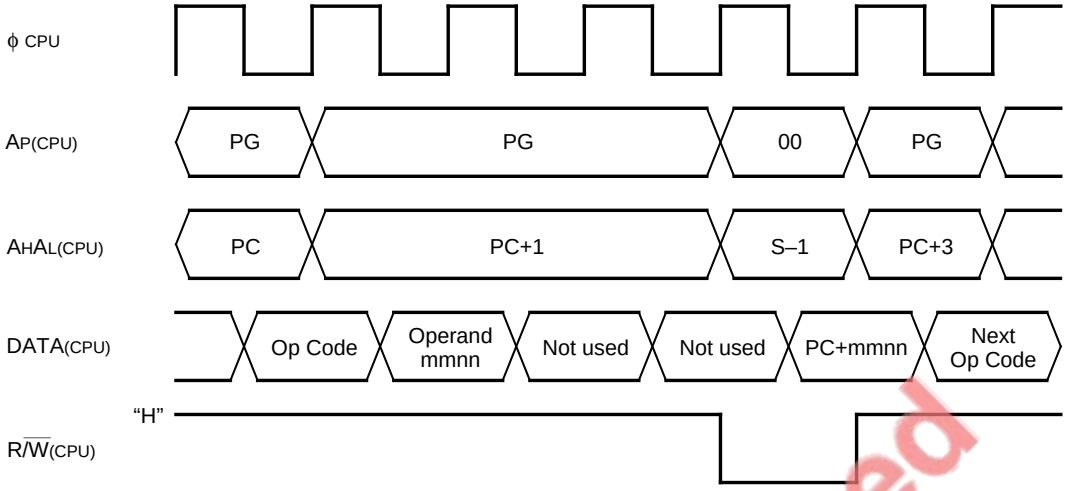
Instruction : PEI

Timing :



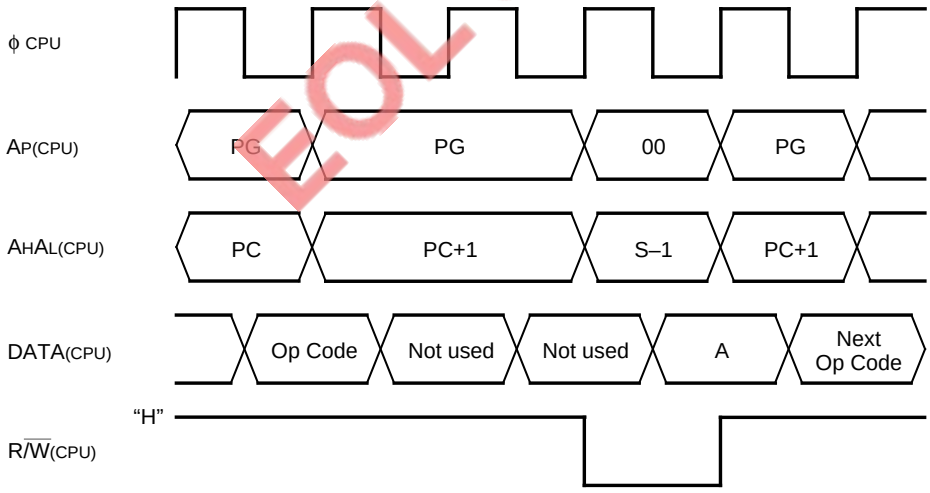
Instruction : PER

Timing :



Instruction : PHA, PHD, PHP, PHX, PHY

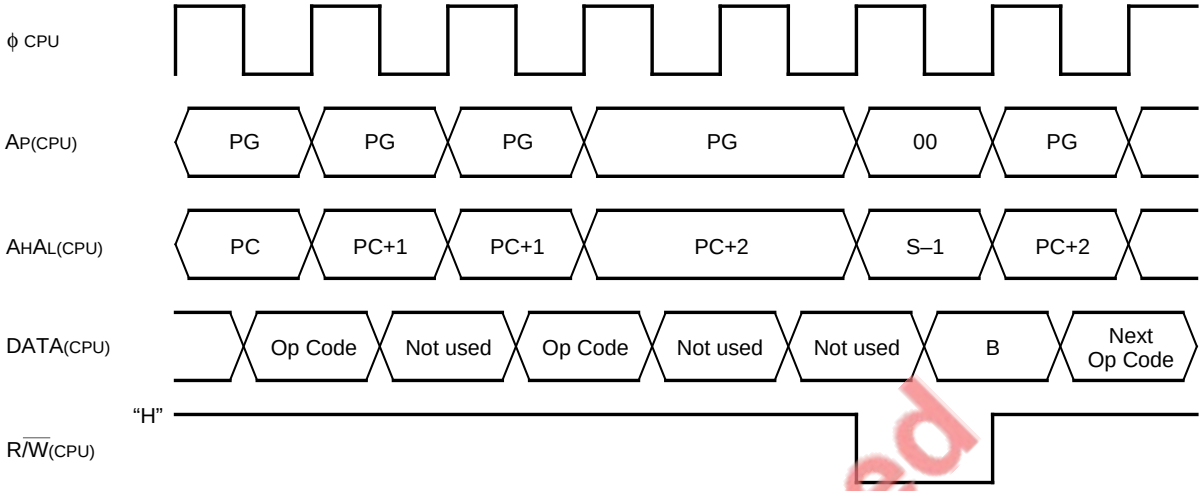
Timing :



EOL announced

Instruction : PHB

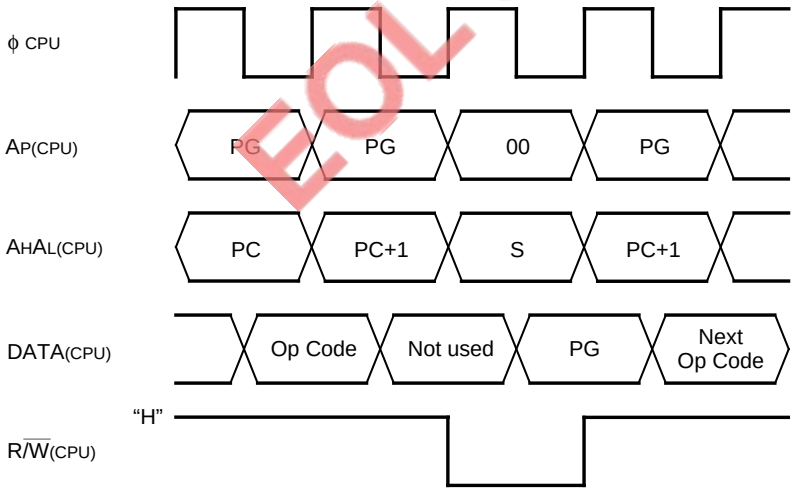
Timing :



EOL announced

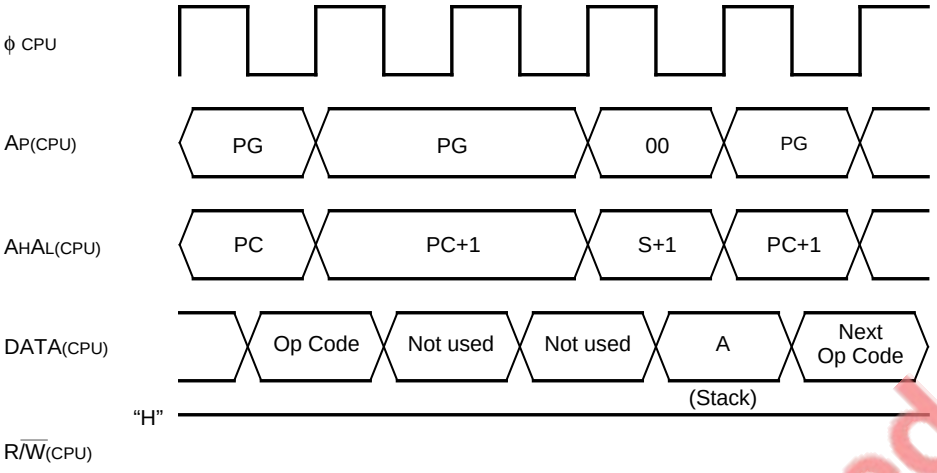
Instruction : PHG, PHT

Timing :



Instruction : PLA, PLD, PLX, PLY

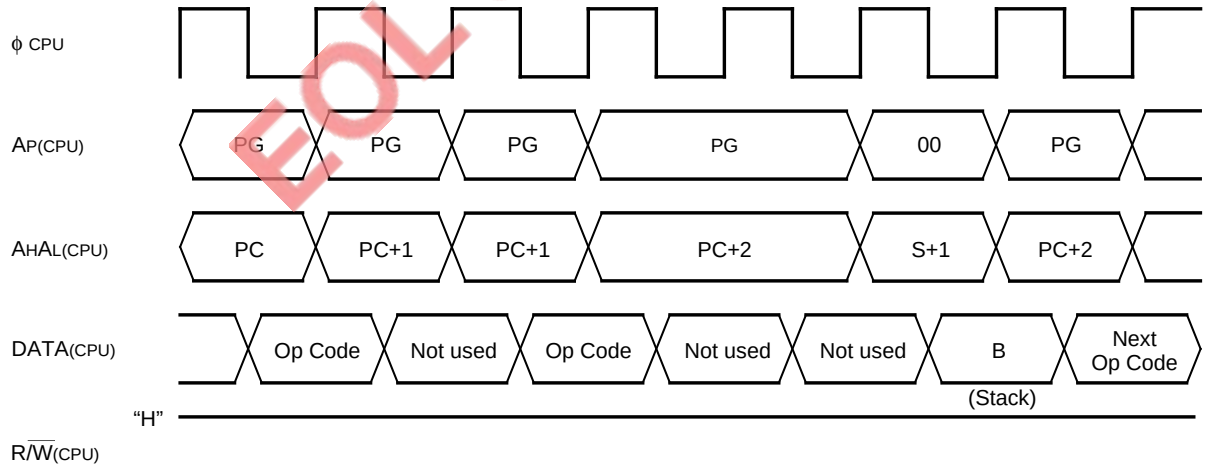
Timing :



EOL announced

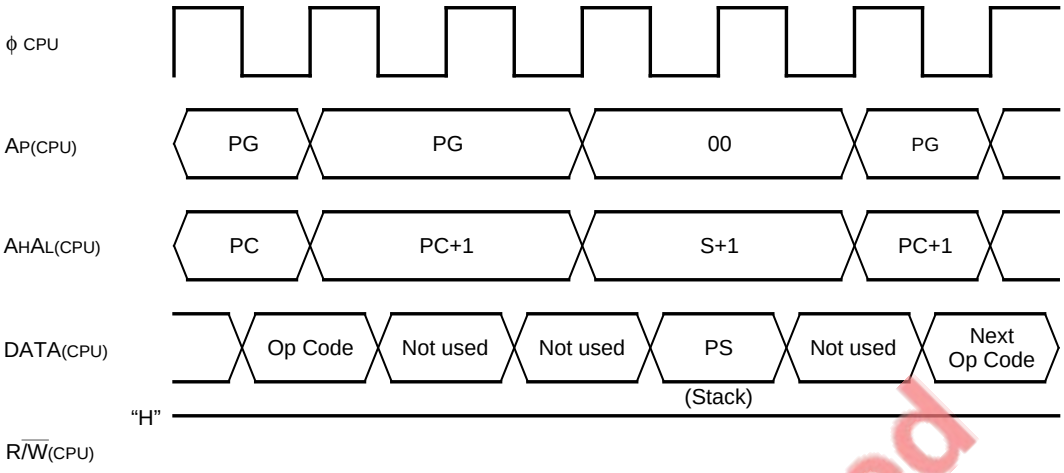
Instruction : PLB

Timing :



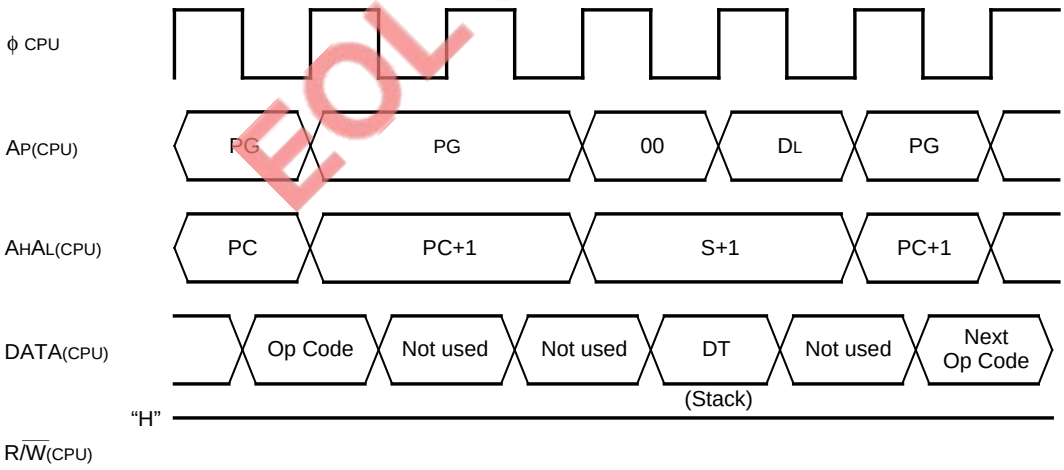
Instruction : PLP

Timing :



Instruction : PLT

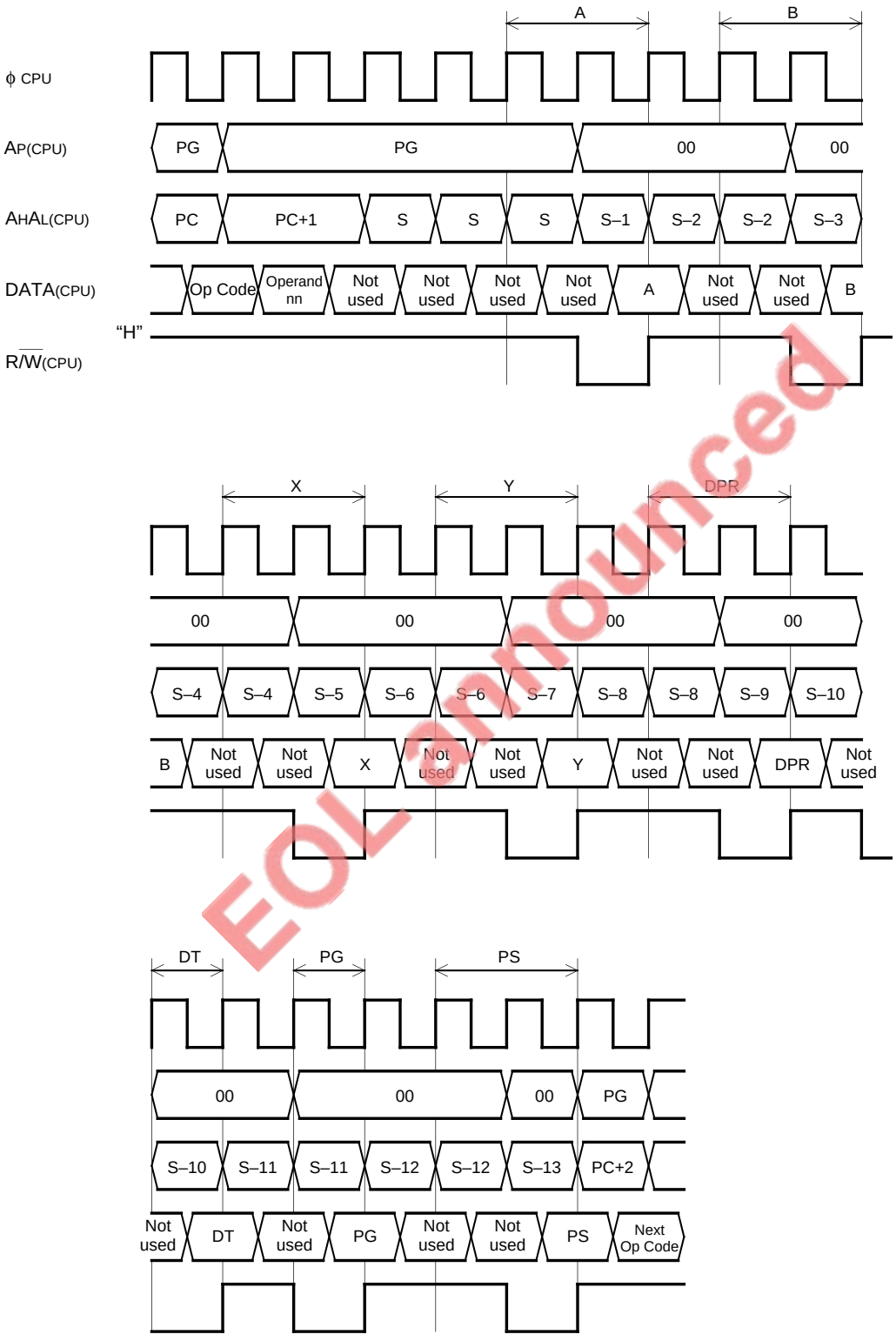
Timing :



EOL announced

Instruction : PSH

Timing :

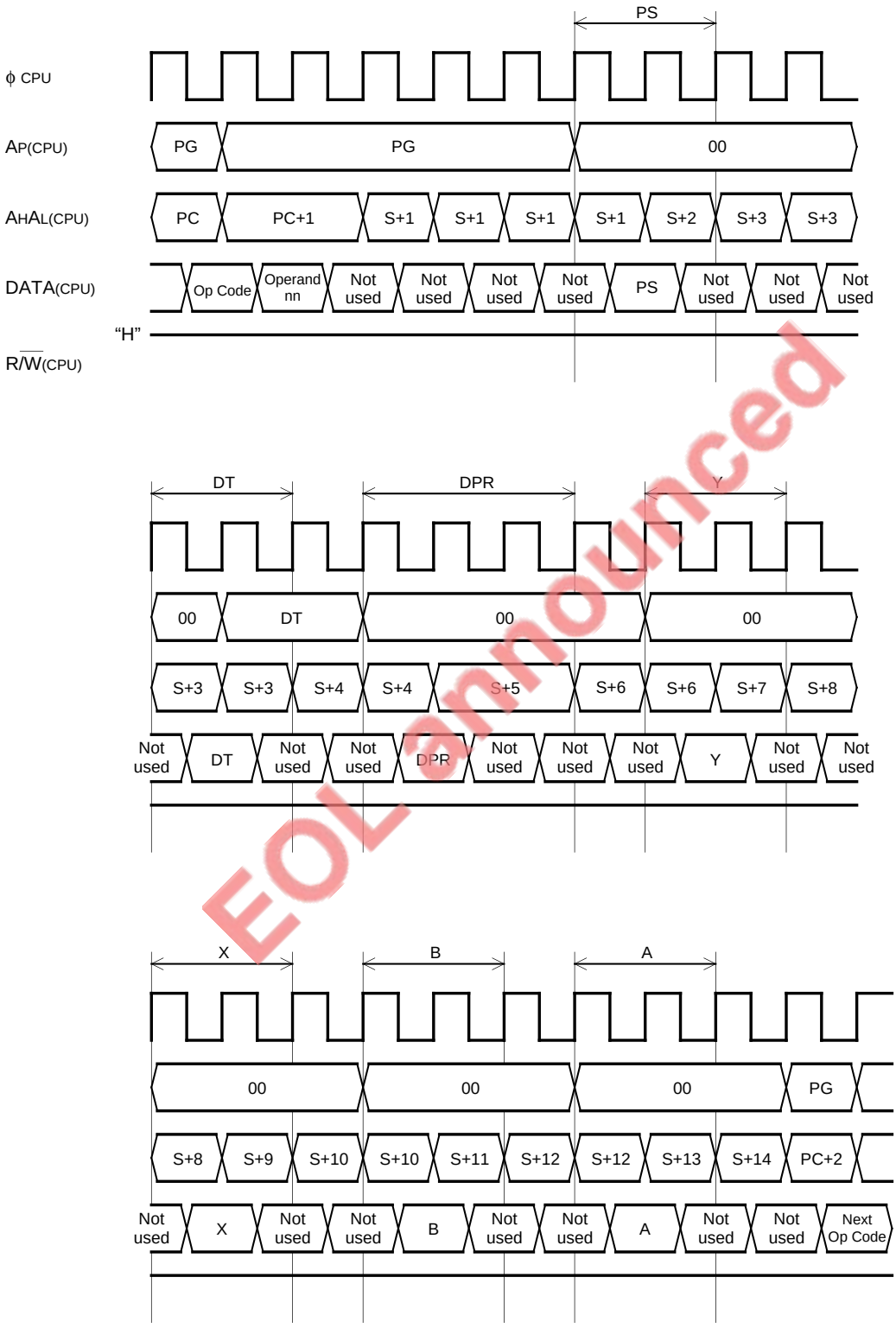


EOL announced

(Note) This figure is an example pushed all the registers by PSH instruction. If any register is not pushed, its cycle " ⟷ " is nothing.

Instruction : PUL

Timing :



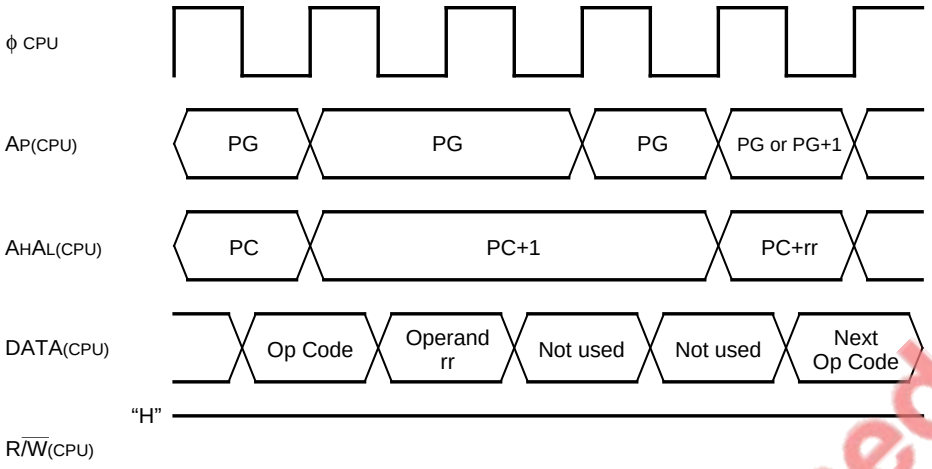
EOL announced

(Note) This figure is an example pushed all the registers by PUL instruction. If any register is not pushed, its cycle " ⟷ " is nothing.

Relative

Instruction : BRA

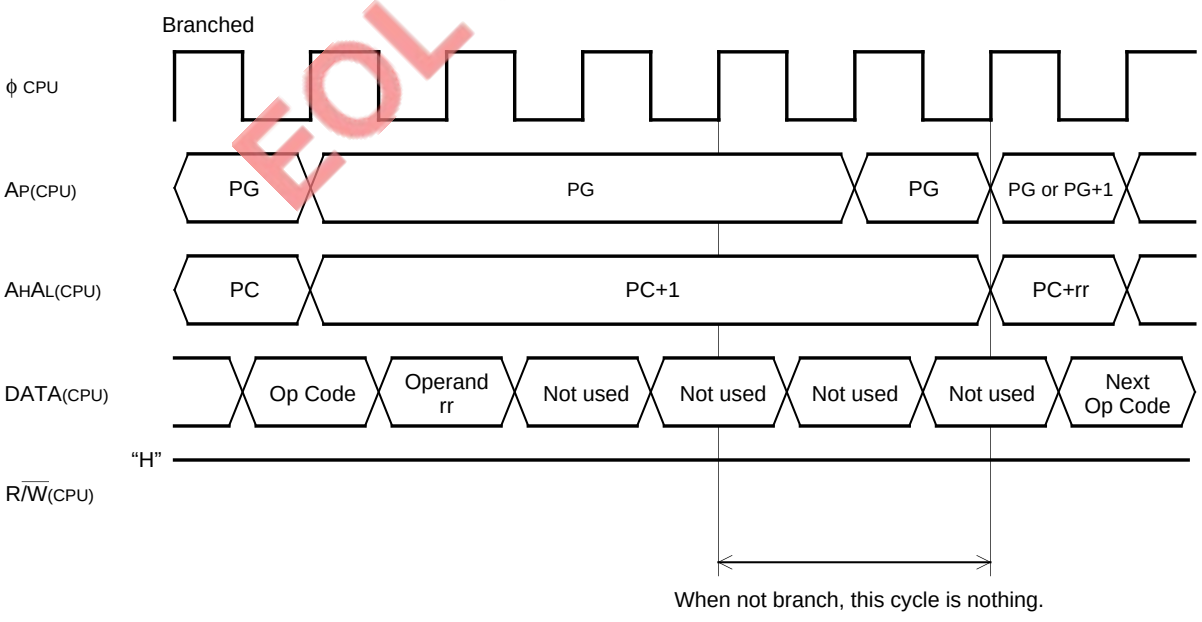
Timing :



EOL announced

Instruction : BCC, BCS, BEQ, BMI, BNE, BPL, BVC, BVS

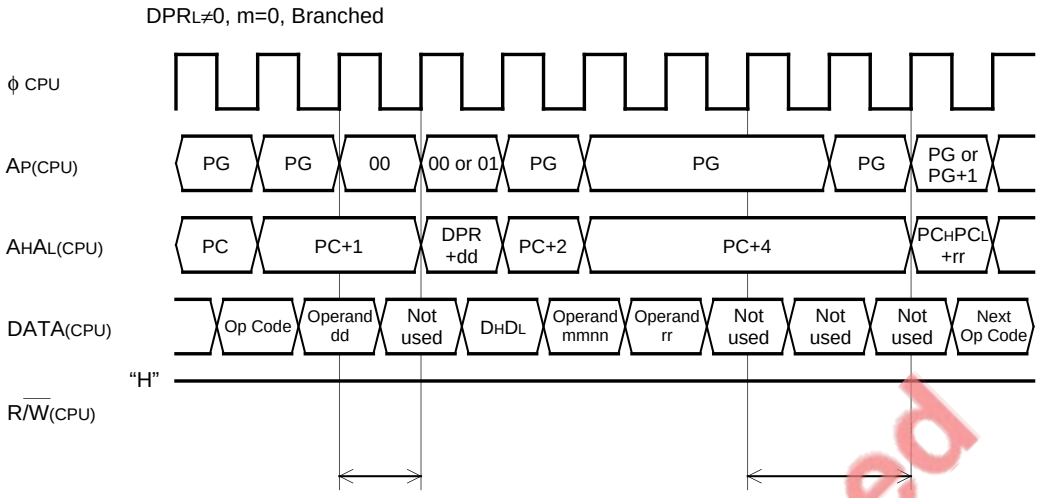
Timing :



Direct Bit Relative

Instruction : BBC, BBS

Timing :



When m=1, fetched operand at 5-th cycle is 1-byte (nn).

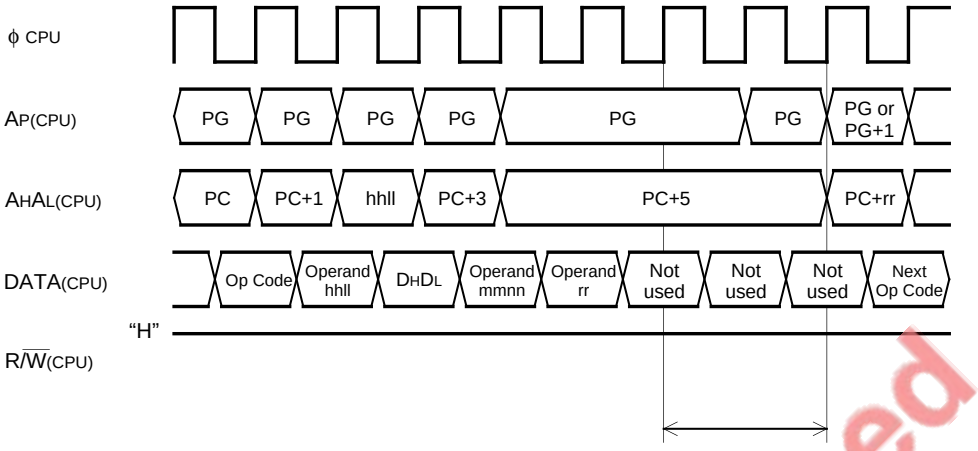
EOL announced

Absolute Bit Relative

Instruction : BBC, BBS

Timing :

m=0, Branched



When not branch, this cycle is nothing.

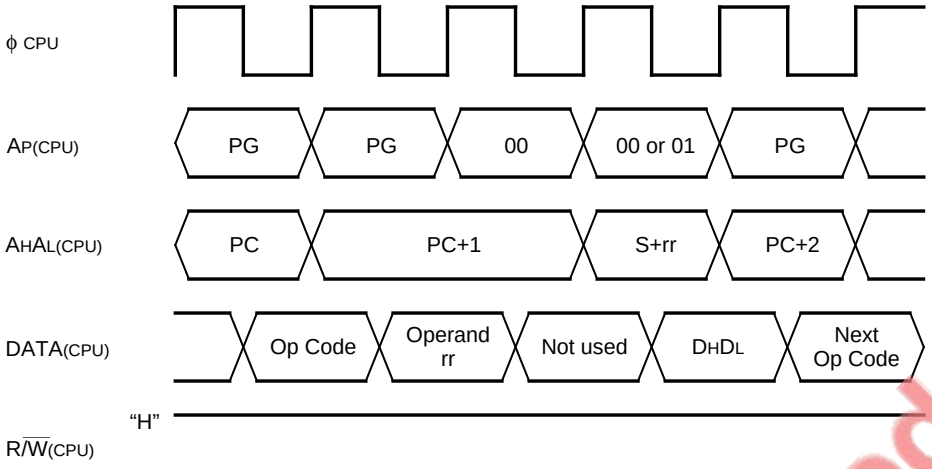
When m=1, fetched operand at 4-th cycle is 1-byte (nn).

EOL announced

Stack Pointer Relative

Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

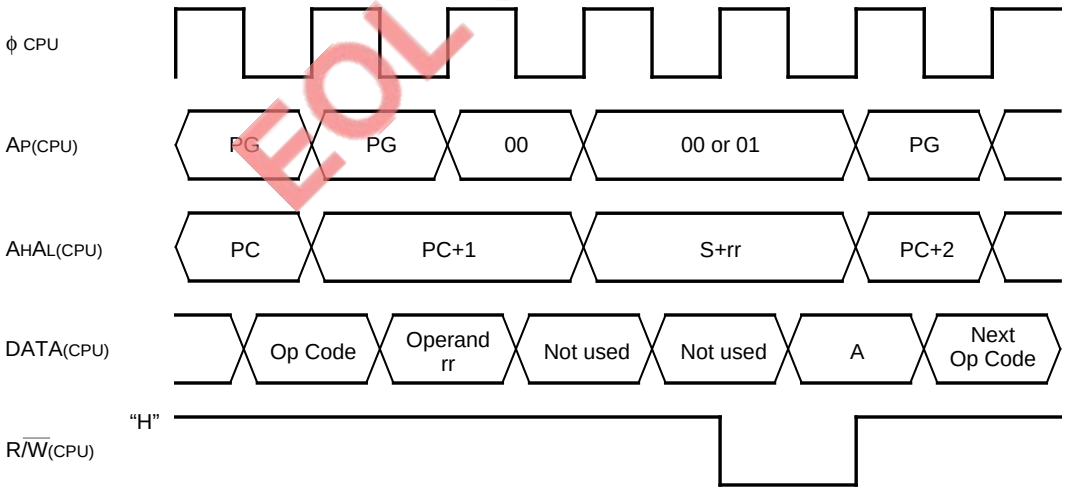
Timing :



EOL announced

Instruction : STA

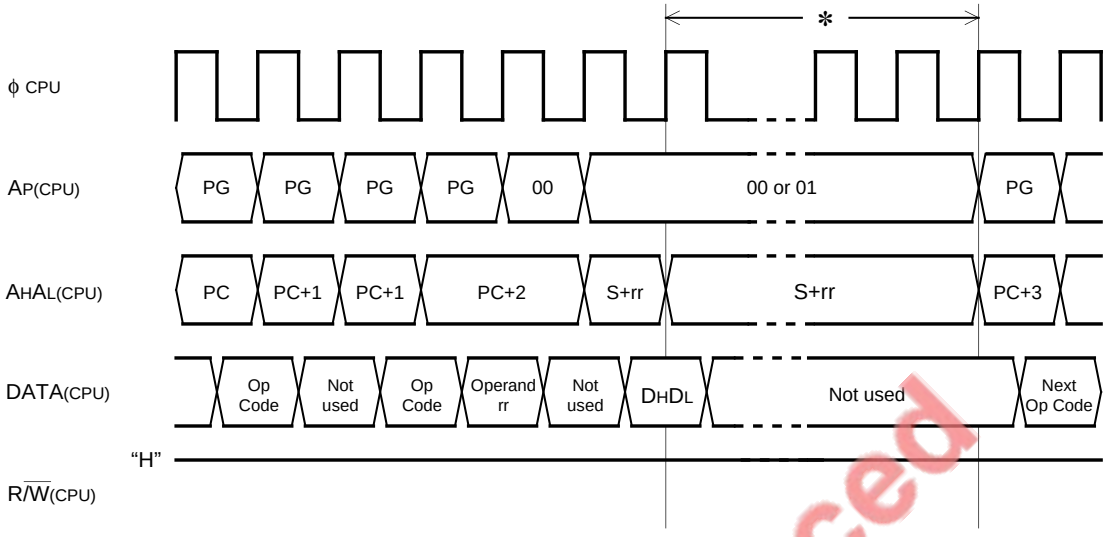
Timing :



Stack Pointer Relative

Instruction : DIV, DIVS*, MPY, MPYS*

Timing :



(Note) The cycle number during ✻ is shown in following table

Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus×Plus	22	14
	Minus×Minus		
	Plus×Minus	25	17
	Minus×Plus		

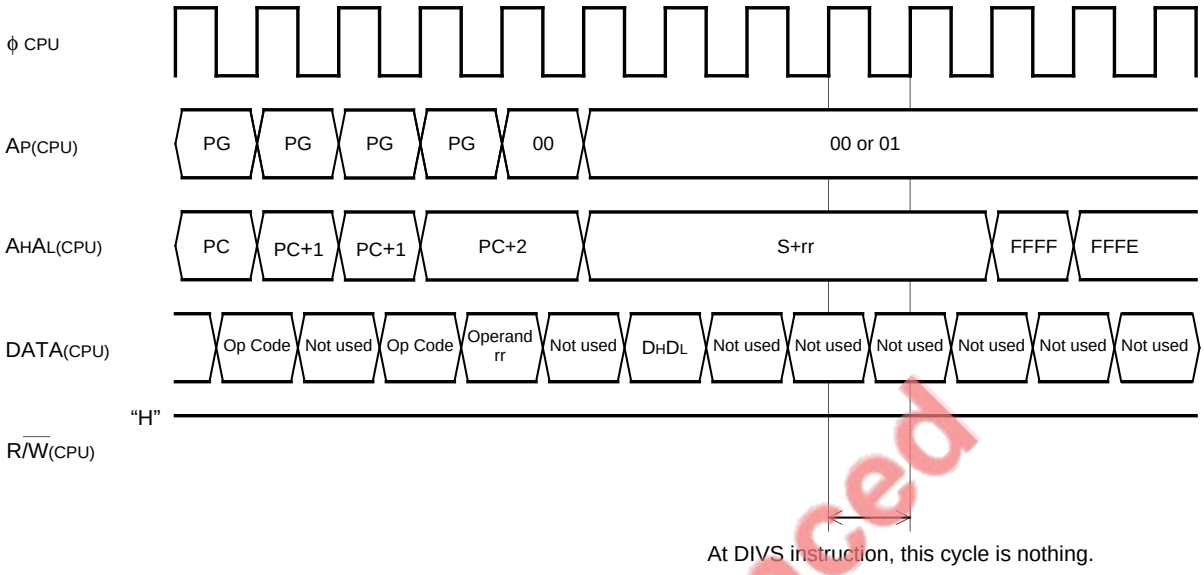
• The contents of AHAL(CPU) during ✻ of the DIVS instruction is undefined.
• When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during ✻ is undefined.

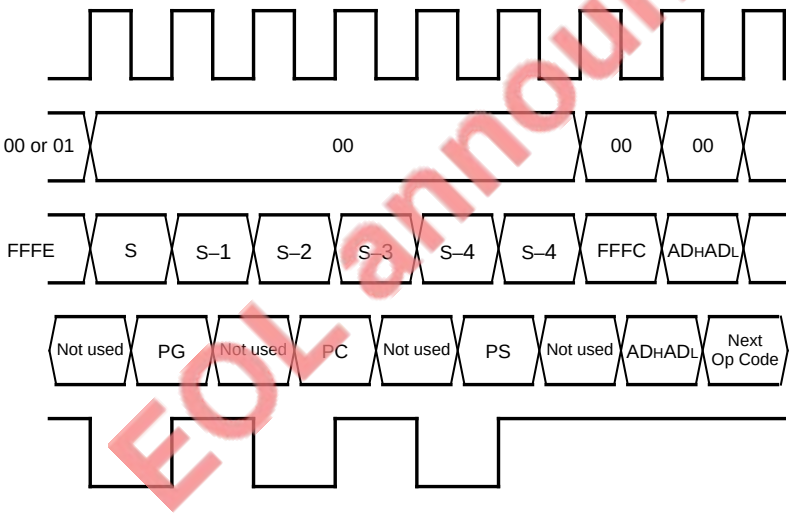
EOL announced

Stack Pointer Relative

Instruction : DIV, DIVS* (case of 0 division)

Timing :



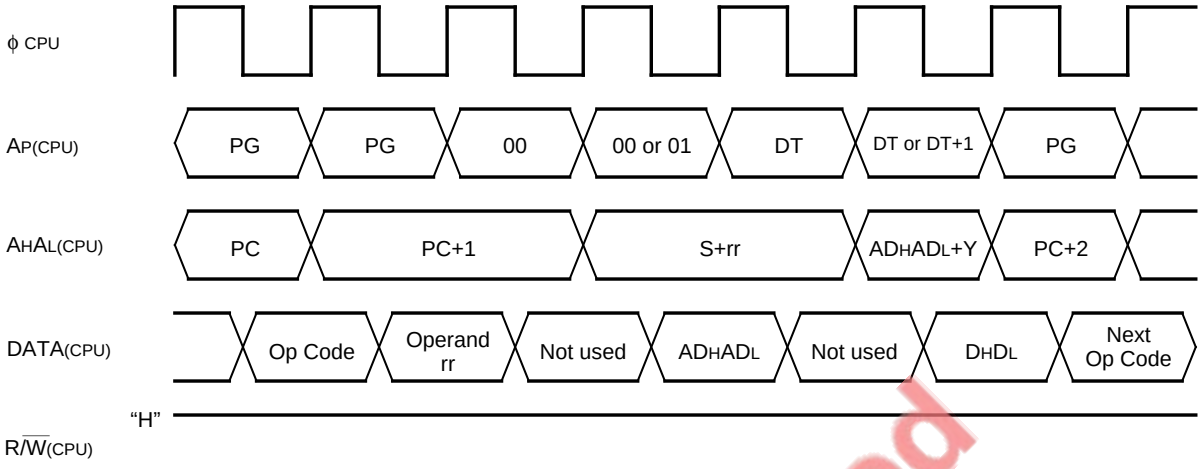


EOL announced

Stack Pointer Relative Indirect Indexed Y

Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

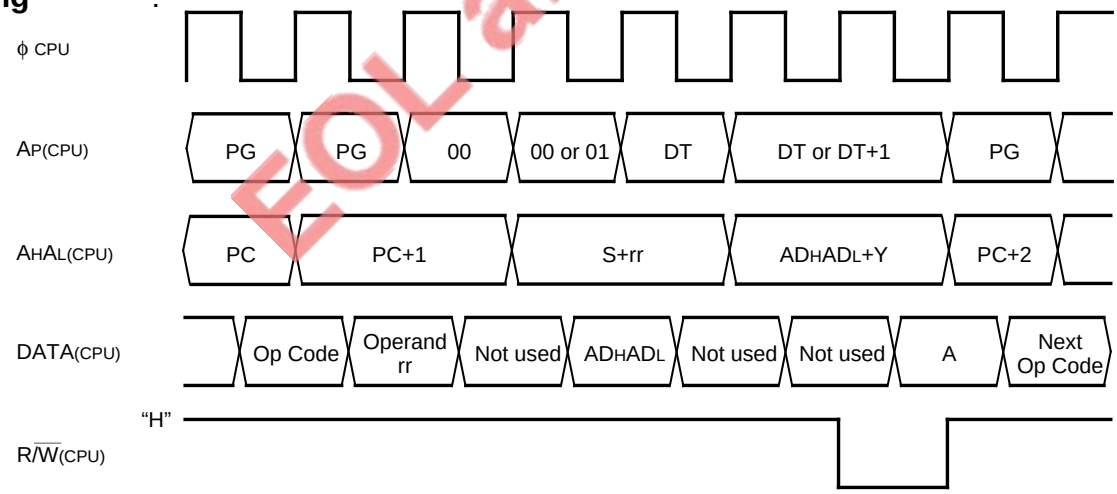
Timing :



EOL announced

Instruction : STA

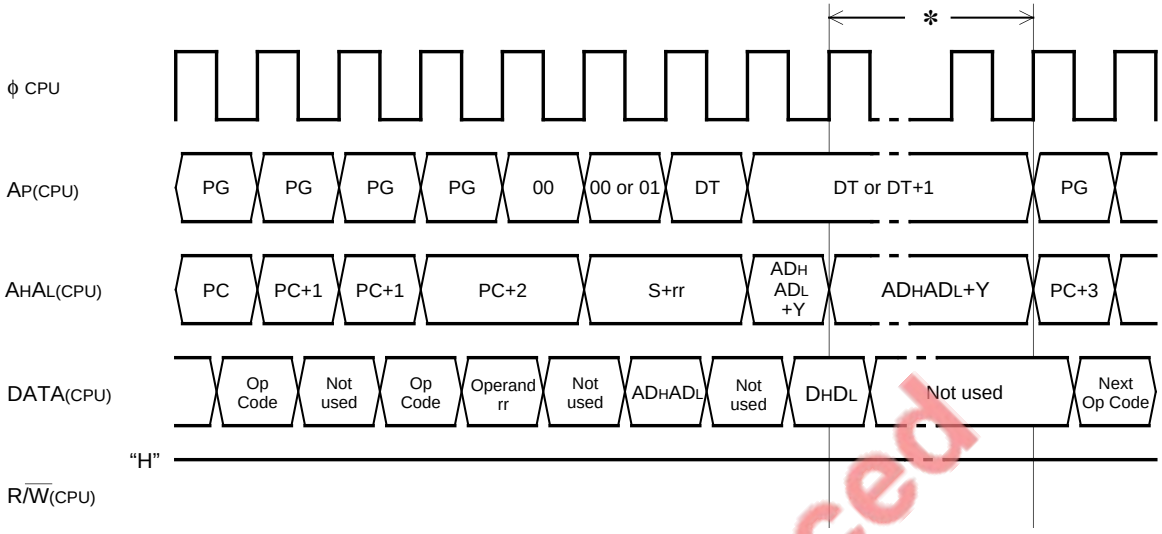
Timing :



Stack Pointer Relative Indirect Indexed Y

Instruction : DIV, DIVS*, MPY, MPYS*

Timing :



(Note) The cycle number during ✻ is shown in following table

Instruction	The contents of division	The number of cycles(cycle)	
		m=0	m=1
DIV	—	39	23
DIVS	Plus÷Plus	41	25
	Plus÷Minus		
	Minus÷Minus		
	Minus÷Plus	42	26
MPY	—	20	12
MPYS	Plus×Plus	22	14
	Minus×Minus		
	Plus×Minus	25	17
	Minus×Plus		

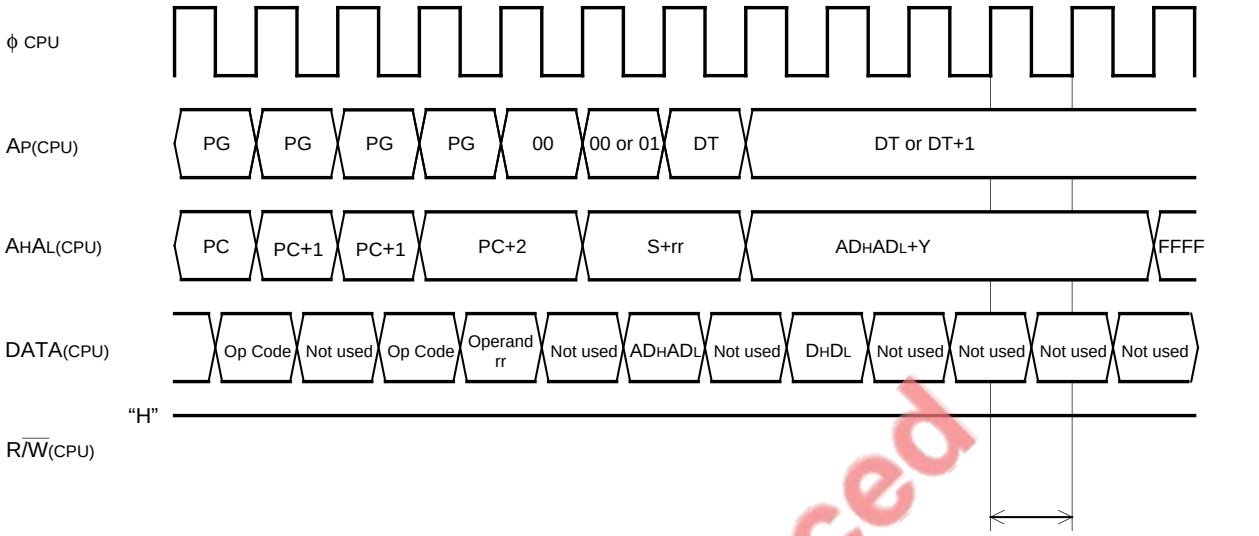
- The contents of AHAL(CPU) during ✻ of the DIVS instruction is undefined.
- When the multiplier and the multiplicand is different sign at the MPYS instruction, the contents of AHAL(CPU) of the last 3 cycles during ✻ is undefined.

EOL announced

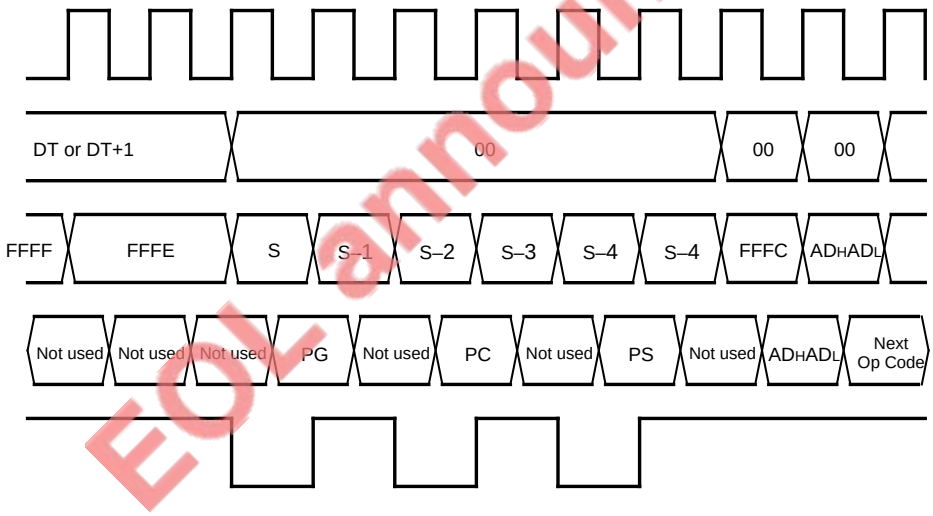
Stack Pointer Relative Indirect Indexed Y

Instruction : DIV , DIVS*(case of 0 division)

Timing :



At DIVS instruction, this cycle is nothing.

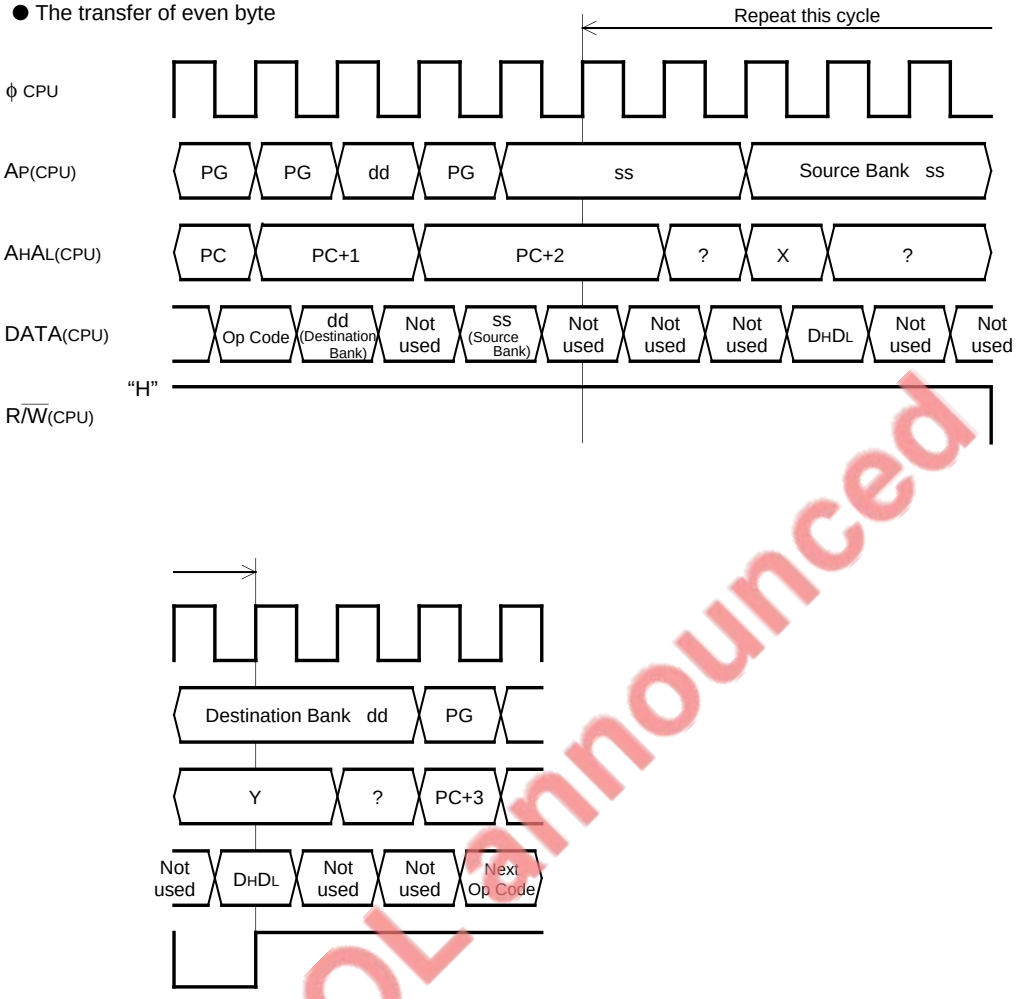


EOL announced

Instruction : MVN

Timing :

● The transfer of even byte



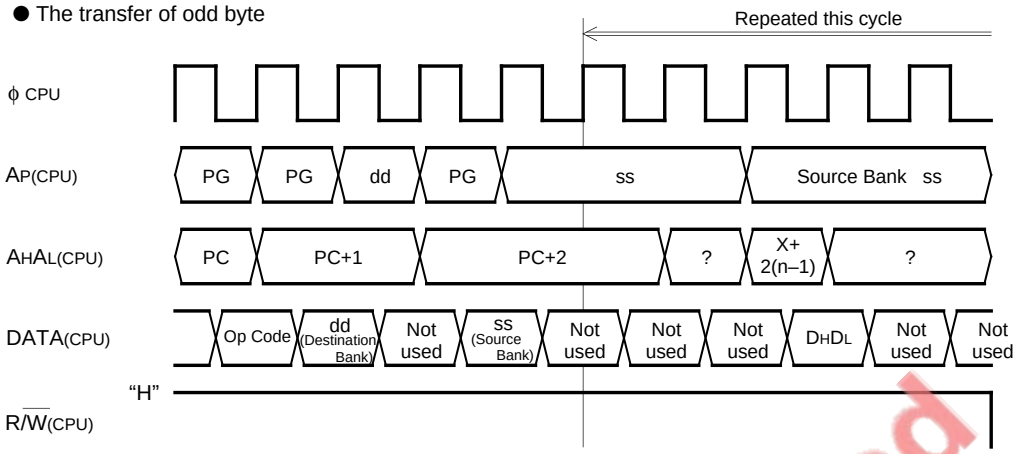
(Note) This figure is shown that transferred the 2-bytes data. If transferred more than 2-bytes data, the cycle " ⟷ " is repeated each 2-bytes.
The CPU instruction execution sequence is identical regardless of whether the transfer start address is even or odd.

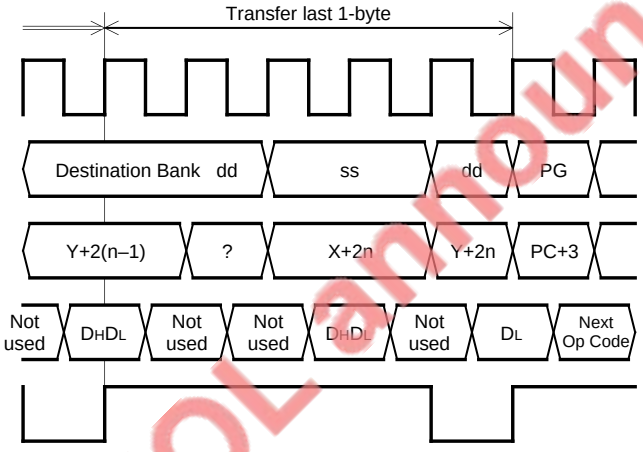
EOL Announced

Instruction : MVN

Timing :

● The transfer of odd byte





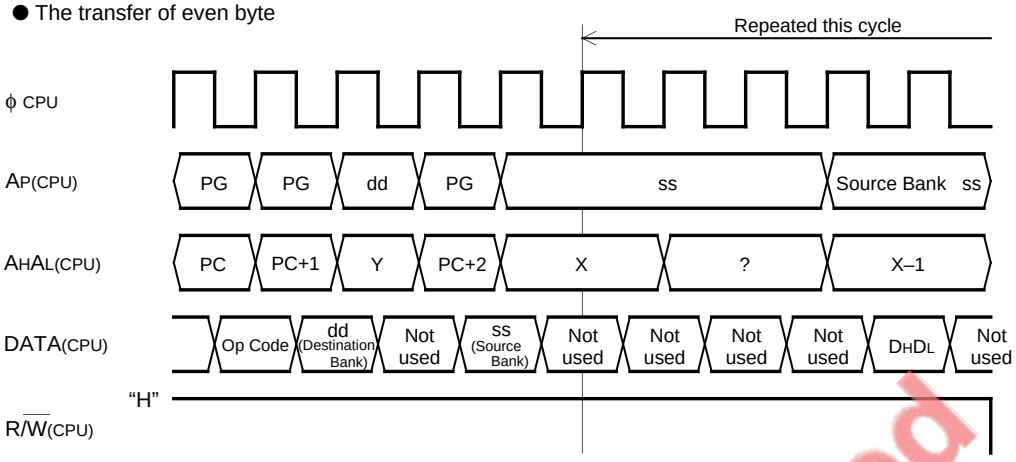
EOL announced

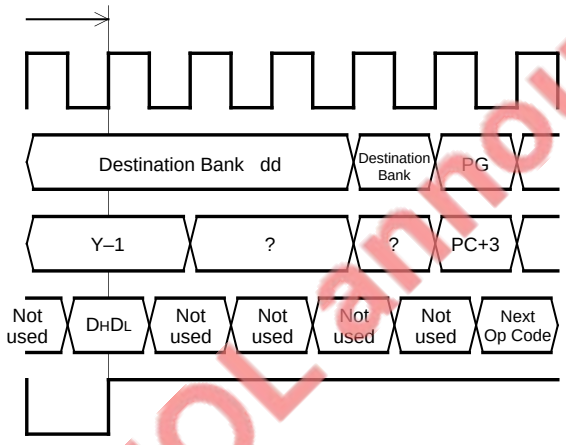
(Note) This figure is shown that transferred the 2-bytes. If transferred more than 2-bytes data, the cycle " ⟺ " is repeated each 2-bytes.
The CPU instruction execution sequence is identical regardless of whether the transfer start address is even or odd. The transfer of the last byte is performed by reading 2 byte and writing 1 byte.

Instruction : MVP

Timing :

● The transfer of even byte





EOL announced

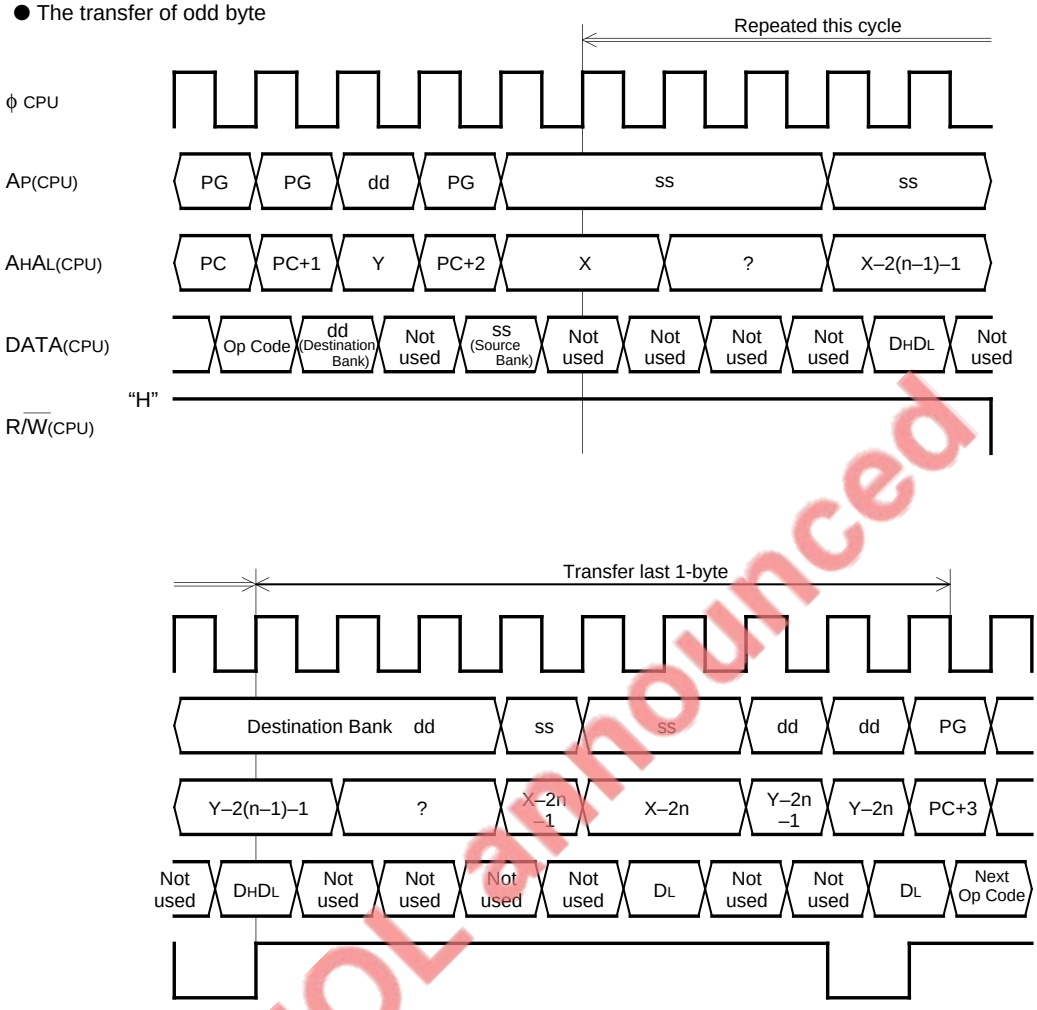
(Note) This figure is shown that transferred the 2-bytes data. If transferred more than 2-bytes, the cycle " ⟷ " is repeated each 2-bytes.
The CPU instruction execution sequence is identical regardless of whether the transfer start address is even or odd.

Block Transfer

Instruction : MVP

Timing :

● The transfer of odd byte



EOL announced

(Note) This figure is shown that transferred the 2-bytes data. If transferred more than 2-bytes, the cycle " ⟺ " is repeated each 2-bytes.
The CPU instruction execution sequence is identical regardless of whether the transfer start address is even or odd.

MEMORANDUM

EOL announced

EOL announced

APPENDIX

APPENDIX 1. Machine Instructions
APPENDIX 2. Hexadecimal Instruction Code Table

APPENDIX

APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																														
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
ADC (Note 1,2)	A_{CC},C ← A_{CC}+M+C	Adds the carry, the accumulator and the memory contents. The result is entered into the accumulator. When the D flag is "0", binary additions is done, and when the D flag is "1", decimal addition is done.				69	2	2				65	4	2				75	5	2				72	6	2	61	7	2	71	8	2
						42 69	4	3				42 65	6	3				42 75	7	3				42 72	8	3	42 61	9	3	42 71	10	3
AND (Note 1,2)	A_{CC} ← A_{CC} ∧ M	Obtains the logical product of the contents of the accumulator and the contents of the memory. The result is entered into the accumulator.				29	2	2				25	4	2				35	5	2				32	6	2	21	7	2	31	8	2
						42 29	4	3				42 25	6	3				42 35	7	3				42 32	8	3	42 21	9	3	42 31	10	3
ASL (Note 1)	m=0 C ← b_{15} ··· b_0 ← 0 m=1 C ← b_7 ··· b_0 ← 0	Shifts the accumulator or the memory contents one bit to the left. "0" is entered into bit 0 of the accumulator or the memory. The contents of bit 15 (bit 7 when the m flag is "1") of the accumulator or memory before shift is entered into the C flag.							0A	2	1	06	7	2				16	7	2												
									42 0A	4	2																					
ASR* (Note 1)	m=0 → b_{15} ··· b_0 → C m=1 → b_7 ··· b_0 → C	Shifts the accumulator or the memory contents one bit to the right. The bit 0 of the accumulator or memory is entered into the C flag. The contents of bit 15 (bit 7 when the m flag is "1") of the accumulator or memory before shift is entered into bit 15 (bit 7).							89 08	5	2	89 06	9	3				89 16	10	3												
									42 08	5	2																					
BBC (Note 3,5)	Mb=0 ?	Tests the specified bit of the memory. Branches when all the contents of the specified bit is "0".																															
BBS (Note 3,5)	Mb=1 ?	Tests the specified bit of the memory. Branches when all the contents of the specified bit is "1".																															
BCC (Note 3)	C=0 ?	Branches when the contents of the C flag is "0".																															
BCS (Note 3)	C=1 ?	Branches when the contents of the C flag is "1".																															
BEQ (Note 3)	Z=1 ?	Branches when the contents of the Z flag is "1".																															
BMI (Note 3)	N=1 ?	Branches when the contents of the N flag is "1".																															
BNE (Note 3)	Z=0 ?	Branches when the contents of the Z flag is "0".																															
BPL (Note 3)	N=0 ?	Branches when the contents of the N flag is "0".																															
BRA (Note 4)	PC←PC±offset PG←PG+1 (carry occured) PG←PG−1 (borrow occured)	Jumps to the address indicated by the program counter plus the offset value.																															
BRK	PC←PC+2 M(S)←PG S←S−1 M(S)←PC_H S←S−1 M(S)←PC_L S←S−1 M(S)←PS_H S←S−1 M(S)←PS_L S←S−1 I←1 PC_L←AD_L PC_H←AD_H PG←00_{16}	Executes software interruption.	00	15	2																												
BVC (Note 3)	V=0 ?	Branches when the contents of the V flag is "0".																															
BVS (Note 3)	V=1 ?	Branches when the contents of the V flag is "1".																															
CLB (Note 5)	Mb←0	Makes the contents of the specified bit in the memory "0".													14	8	3																
CLC	C←0	Makes the contents of the C flag "0".	18	2	1																												
CLI	I←0	Makes the contents of the I flag "0".	58	2	1																												
CLM	m←0	Makes the contents of the m flag "0".	58	2	1																												

EOL announced

APPENDIX.1 Machine Instructions

Addressing mode																																																						Processor status register										
L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
67	10	2	77	11	2	6D	4	3				7D	6	3	79	6	3	6F	6	4	7F	7	4																						63	5	2	73	8	2				•	•	•	N	V	•	•	•	•	Z	C
42 67	12	3	42 77	13	3	42 6D	6	4				42 7D	8	4	42 79	8	4	42 6F	8	5	42 7F	9	5																						42 63	7	3	42 73	10	3														
27	10	2	37	11	2	2D	4	3				3D	6	3	39	6	3	2F	6	4	3F	7	4																						23	5	2	33	8	2				•	•	•	N	•	•	•	•	•	Z	•
42 27	12	3	42 37	13	3	42 2D	6	4				42 3D	8	4	42 39	8	4	42 2F	8	5	42 3F	9	5																						42 23	7	3	42 33	10	3														
						0E	7	3				1E	8	3																																								•	•	•	N	•	•	•	•	•	Z	C
						89 0E	9	4				89 1E	11	4																																								•	•	•	N	•	•	•	•	•	Z	C
																																							34	7	4	3C	8	5										•	•	•	•	•	•	•	•	•	•	•
																																							24	7	4	2C	8	5										•	•	•	•	•	•	•	•	•	•	•
																																				90	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				B0	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				F0	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				30	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				D0	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				10	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				80	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				82	4	3																										
																																																						•	•	•	•	•	•	•	•	•	•	•
																																				50	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				70	4	2																•	•	•	•	•	•	•	•	•	•	•
									1C	9	4																																													•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	•	•	•	•	•	•	•	0
																																																						•	•	•	•	•	•	•	•	0	•	•
																																																						•	•	•	•	•	0	•	•	•	•	•

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																																
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y					
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
CLP	PSb←0	Specifies the bit position in the processor status register by the bit pattern of the second byte in the instruction, and sets "0" in that bit.				C2	4	2																											
CLV	V←0	Makes the contents of the V flag "0".	B8	2	1																														
CMP (Note 1,2)	A_{CC}−M	Compares the contents of the accumulator with the contents of the memory.				C9	2	2				C5	4	2				D5	5	2				D2	6	2	C1	7	2	D1	8	2			
						42 C9	4	3				42 C5	6	3				42 D5	7	3				42 D2	8	3	42 C1	9	3	42 D1	10	3			
CPX (Note 2)	X−M	Compares the contents of the index register X with the contents of the memory.				E0	2	2				E4	4	2																					
CPY (Note 2)	Y−M	Compares the contents of the index register Y with the contents of the memory.				C0	2	2				C4	4	2																					
DEC (Note 1)	A_{CC}←A_{CC}−1 or M←M−1	Decrements the contents of the accumulator or memory by 1.							1A	2	1	C6	7	2				D6	7	2															
									42 1A	4	2																									
DEX	X←X−1	Decrements the contents of the index register X by 1.	CA	2	1																														
DEY	Y←Y−1	Decrements the contents of the index register Y by 1.	88	2	1																														
DIV (Note 2,10)	A(quotient)←B,A/M B(remainder)	The numeral that places the contents of accumulator B to the higher order and the contents of accumulator A to the lower order is divided by the contents of the memory. The quotient is entered into accumulator A and the remainder into accumulator B.				89 29	27	3				89 25	29	3				89 35	30	3				89 32	31	3	89 21	32	3	89 31	33	3			
DIVS★ (Note 2,14)	A(quotient)←B,A/M B(remainder)with sign	The numeral with sign that places the contents of accumulator B to the higher order and the contents of accumulator A to the lower order is divided by the contents of the memory. The quotient is entered into accumulator A and the remainder into accumulator B.				89 A9	29	3				89 A5	31	3				89 B5	32	3				89 B2	33	3	89 A1	34	3	89 B1	35	3			
EOR (Note 1,2)	A_{CC}←A_{CC}¥M	Logical exclusive sum is obtained of the contents of the accumulator and the contents of the memory. The result is placed into the accumulator.				49	2	2				45	4	2				55	5	2				52	6	2	41	7	2	51	8	2			
						42 49	4	3				42 45	6	3				42 55	7	3				42 52	8	3	42 41	9	3	42 51	10	3			
EXTS★ (Note 1)	Bit 7 of A_{CC}=1 b15 b7 11111111 1 Bit 7 of A_{CC}=0 b15 b7 00000000 0	The signed 8-bit data stored in the low-order byte of the accumulator is extended to a 16-bit data.																																	
						89 8B	8	2																											
						42 8B	8	2																											
EXTZ★ (Note 1)	A_{CC} b15 b8 b7 b0 00000000	The 8-bit data stored in the low-order byte of the accumulator is extended to a 16-bit data. Bits 8 to 15 of the accumulator are set to "0".																																	
						89 AB	5	2																											
						42 AB	5	2																											
INC (Note 1)	A_{CC}←A_{CC}+1 or M←M+1	Increments the contents of the accumulator or memory by 1.							3A	2	1	E6	7	2				F6	7	2															
									42 3A	4	2																									
INX	X←X+1	Increments the contents of the index register X by 1.	E8	2	1																														
INY	Y←Y+1	Increments the contents of the index register Y by 1.	C8	2	1																														
JMP	ABS PC_L ← AD_L PC_H ← AD_H ABL PC_L ← AD_L PC_H ← AD_H PG←AD_G (ABS) PC_L ←(AD_H, AD_L) PC_H ←(AD_H, AD_L+1) L(ABS) PC_L ←(AD_H, AD_L) PC_H ←(AD_H, AD_L+1) PG ←(AD_H, AD_L+2) (ABS, X) PC_L ←(AD_H, AD_L+X) PC_H ←(AD_H, AD_L+X +1)	Places a new address into the program counter and jumps to that new address.																																	

PDL announced

APPENDIX.1 Machine Instructions

Addressing mode																																																						Processor status register										
L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
																																																						•	•	•	Specified flag becomes "0".							
																																																						•	•	•	•	0	•	•	•	•	•	•
C7	10	2	D7	11	2	CD	4	3				DD	6	3	D9	6	3	CF	6	4	DF	7	4																								C3	5	2	D3	8	2				•	•	•	N	•	•	•	•	•	Z	C
42 C7	12	3	42 D7	13	3	42 CD	6	4				42 DD	8	4	42 D9	8	4	42 CF	8	5	42 DF	9	5																								42 C3	7	3	42 D3	10	3														
						EC	4	3																																															•	•	•	N	•	•	•	•	•	Z	C
						CC	4	3																																															•	•	•	N	•	•	•	•	•	Z	C
						CE	7	3				DE	8	3																																									•	•	•	N	•	•	•	•	•	Z	•
																																																							•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
89 27	35	3	89 37	36	3	89 2D	29	4				89 3D	31	4	89 39	31	4	89 2F	31	5	89 3F	32	5																								89 23	30	3	89 33	33	3				•	•	•	N	V	•	•	•	•	Z	C
89 A7	37	3	89 B7	38	3	89 AD	31	4				89 BD	33	4	89 B9	33	4	89 AF	33	5	89 BF	34	5																								89 A3	32	3	89 B3	35	5				•	•	•	N	V	•	•	•	•	Z	C
47	10	2	57	11	2	4D	4	3				5D	6	3	59	6	3	4F	6	4	5F	7	4																								43	5	2	53	8	2				•	•	•	N	•	•	•	•	•	Z	•
42 47	12	3	42 57	13	3	42 4D	6	4				42 5D	8	4	42 59	8	4	42 4F	8	5	42 5F	9	5																								42 43	7	3	42 53	10	3														
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	0	•	•	•	•	•	Z	•
						EE	7	3				FE	8	3																																									•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
						4C	2	3										5C	4	4				6C	4	3	DC	8	3	7C	6	3																						•	•	•	•	•	•	•	•	•	•	•

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																														
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
JSR	ABS M(S)← PC_H S←S−1 M(S)← PC_L S←S−1 PC_L ← AD_L PC_H ← AD_H ABL M(S)← PG S←S−1 M(S)← PC_H S←S−1 M(S)← PC_L S ← S−1 PC_L ← AD_L PC_H ← AD_H PG ← AD_B (ABS, X) M(S)← PC_H S←S−1 M(S)← PC_L S←S−1 PC_L ←(AD_H, AD_L+X) PC_H ←(AD_H, AD_L+X+1)	Saves the contents of the program counter (also the contents of the program bank register for ABL) into the stack, and jumps to the new address.																																
LDA (Note 1,2)	Acc ← M	Enters the contents of the memory into the accumulator.				A9	2	2				A5	4	2				B5	5	2				B2	6	2	A1	7	2	B1	8	2
						42 A9	4	3				42 A5	6	3				42 B5	7	3				42 B2	8	3	42 A1	9	3	42 B1	10	3
LDM (Note 5)	M ← IMM	Enters the immediate value into the memory.										64	4	3				74	5	3															
LDT	DT ← IMM	Enters the immediate value into the data bank register.				89 C2	5	3																											
LDX (Note 2)	X ← M	Enters the contents of the memory into index register X.				A2	2	2				A6	4	2							B6	5	2												
LDY (Note 2)	Y ← M	Enters the contents of the memory into index register Y.				A0	2	2				A4	4	2				B4	5	2															
LSR (Note 1)	m=0 0→[b_{15}]···[b_0]→C m=1 0→[b_7]···[b_0]→C	Shifts the contents of the accumulator or the contents of the memory one bit to the right. The bit 0 of the accumulator or the memory is entered into the C flag. "0" is entered into bit 15 (bit 7 when the m flag is "1".)							4A	2	1	46	7	2				56	7	2															
									4A	4	2																								
MPY (Note 2,11)	B, A←A*M	Multiplies the contents of accumulator A and the contents of the memory. The higher order of the result of operation are entered into accumulator B, and the lower order into accumulator A.				89 09	16	3				89 05	18	3				89 15	19	3				89 12	20	3	89 01	21	3	89 11	22	3
MPYS* (Note 2,15)	B, A←A*M with sign	The content of the accumulator A is multiplied by the content of memory as signed data. The result is a 32-bit data which is placed in the accumulators B (upper 16 bits of the result) and A (lower 16 bits of the result).				89 89	18	3				89 85	20	3				89 95	21	3				89 92	22	3	89 81	23	3	89 91	24	3
MVN (Note 8)	Mn+i←Mm+i	Transmits the data block. The transmission is done from the lower order address of the block.																																
MVP (Note 9)	Mn−i←Mm−i	Transmits the data block. Transmission is done form the higher order address of the data block.																																
NOP	PC←PC+1	Advances the program counter, but performs nothing else.	EA	2	1																													
ORA (Note 1,2)	Acc←Acc∨M	Logical sum per bit of the contents of the accumulator and the contents of the memory is obtained. The result is entered into the accumulator.				09	2	2				05	4	2				15	5	2				12	6	2	01	7	2	11	8	2
						42 09	4	3				42 05	6	3				42 15	7	3				42 12	8	3	42 01	9	3	42 11	10	3

Not announced

APPENDIX.1 Machine Instructions

L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
						20	6	3										22	8	4										FC	8	3																						•	•	•	•	•	•	•	•	•	•	•
A7	10	2	B7	11	2	AD	4	3				BD	6	3	B9	6	3	AF	6	4	BF	7	4																						A3	5	2	B3	8	2				•	•	•	N	•	•	•	•	•	Z	•
42 A7	12	3	42 B7	13	3	42 AD	6	4				42 BD	B	4	42 B9	8	4	42 AF	B	5	42 BF	9	5																						42 A3	7	3	42 B3	10	3														
						9C	5	4				9E	6	4																																								•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	•	•	•	•	•	•	•	•
						AE	4	3							BE	6	3																																					•	•	•	N	•	•	•	•	•	Z	•
						AC	4	3				BC	6	3																																								•	•	•	N	•	•	•	•	•	Z	•
						4E	7	3				5E	8	3																																								•	•	•	0	•	•	•	•	•	Z	C
89 07	24	3	89 17	25	3	89 0D	18	4				89 1D	20	4	89 19	20	4	89 0F	20	5	89 1F	21	5																						89 03	19	3	89 13	22	3				•	•	•	N	•	•	•	•	•	Z	0
89 87	26	3	89 97	27	3	89 8D	20	4				89 9D	22	4	89 99	22	4	89 8F	22	5	89 9F	23	5																						89 83	21	3	89 93	24	3				•	•	•	N	•	•	•	•	•	Z	0
																																																			54	$7+\frac{i}{2}\times 7$	3	•	•	•	•	•	•	•	•	•	•	•
																																																			44	$9+\frac{i}{2}\times 7$	3	•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	•	•	•	•	•	•	•	•
07	10	2	17	11	2	0D	4	3				1D	6	3	19	6	3	0F	6	4	1F	7	4																						03	5	2	13	8	2				•	•	•	N	•	•	•	•	•	Z	•
42 07	12	3	42 17	13	3	42 0D	6	4				42 1D	8	4	42 19	8	4	42 0F	8	5	42 1F	9	5																						42 03	7	3	42 13	10	3														

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																														
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
PEA	M(S)←IMM$_2$ S←S−1 M(S)←IMM$_1$ S←S−1	The 3rd and the 2nd bytes of the instruction are saved into the stack, in this order.																															
PEI	M(S)←M((DPR)+IMM+1) S←S−1 M(S)←M((DPR)+IMM) S←S−1	Specifies 2 sequential bytes in the direct page in the 2nd byte of the instruction, and saves the contents into the stack.																															
PER	EAR←PC+IMM$_2$,IMM$_1$ M(S)←EAR$_H$ S←S−1 M(S)←EAR$_L$ S←S−1	Regards the 2nd and 3rd bytes of the instruction as 16-bit numerals, adds them to the program counter, and saves the result into the stack.																															
PHA	m=0 M(S)←A$_H$ S←S−1 M(S)←A$_L$ S←S−1 m=1 M(S)←A$_L$ S←S−1	Saves the contents of accumulator A into the stack.																															
PHB	m=0 M(S)←B$_H$ S←S−1 M(S)←B$_L$ S←S−1 m=1 M(S)←B$_L$ S←S−1	Saves the contents of accumulator B into the stack.																															
PHD	M(S)←DPR$_H$ S←S−1 M(S)←DPR$_L$ S←S−1	Saves the contents of the direct page register into the stack.																															
PHG	M(S)←PG S←S−1	Saves the contents of the program bank register into the stack.																															
PHP	M(S)←PS$_H$ S←S−1 M(S)←PS$_L$ S←S−1	Saves the contents of the program status register into the stack.																															
PHT	M(S)←DT S←S−1	Saves the contents of the data bank register into the stack.																															
PHX	x=0 M(S)←X$_H$ S←S−1 M(S)←X$_L$ S←S−1 x=1 M(S)←X$_L$ S←S−1	Saves the contents of the index register X into the stack.																															
PHY	x=0 M(S)←Y$_H$ S←S−1 M(S)←Y$_L$ S←S−1 x=1 M(S)←Y$_L$ S←S−1	Saves the contents of the index register Y into the stack.																															

EOL announced

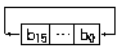
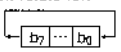
APPENDIX.1 Machine Instructions

Addressing mode																																																						Processor status register										
L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
																																	F4	5	3																			•	•	•	•	•	•	•	•	•	•	•
																																	D4	6	2																			•	•	•	•	•	•	•	•	•	•	•
																																	62	5	3																			•	•	•	•	•	•	•	•	•	•	•
																																	48	4	1																			•	•	•	•	•	•	•	•	•	•	•
																																	42 48	6	2																			•	•	•	•	•	•	•	•	•	•	•
																																	0B	4	1																			•	•	•	•	•	•	•	•	•	•	•
																																	4B	3	1																			•	•	•	•	•	•	•	•	•	•	•
																																	08	4	1																			•	•	•	•	•	•	•	•	•	•	•
																																	8B	3	1																			•	•	•	•	•	•	•	•	•	•	•
																																	DA	4	1																			•	•	•	•	•	•	•	•	•	•	•
																																	5A	4	1																			•	•	•	•	•	•	•	•	•	•	•

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																															
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y				
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#		
PLA	m=0 S←S+1 A_L←M(S) S←S+1 A_H←M(S) m=1 S←S+1 A_L←M(S)	Restores the contents of the stack on the accumulator A.																																
PLB	m=0 S←S+1 B_L←M(S) S←S+1 B_H←M(S) m=1 S←S+1 B_L←M(S)	Restores the contents of the stack on the accumulator B.																																
PLD	S←S+1 DPR_L←M(S) S←S+1 DPR_H←M(S)	Restores the contents of the stack on the direct page register.																																
PLP	S←S+1 PS_L←M(S) S←S+1 PS_H←M(S)	Restores the contents of the stack on the processor status register.																																
PLT	S←S+1 DT←M(S)	Restores the contents of the stack on the data bank register.																																
PLX	x=0 S←S+1 X_L←M(S) S←S+1 X_H←M(S) x=1 S←S+1 X_L←M(S)	Restores the contents of the stack on the index register X.																																
PLY	x=0 S←S+1 Y_L←M(S) S←S+1 Y_H←M(S) x=1 S←S+1 Y_L←M(S)	Restores the contents of the stack on the index register Y.																																
PSH (Note 6)	M(S)←A, B, X···	Saves the registers among accumulator, index register, direct page register, data bank register, program bank register, or processor status register, specified by the bit pattern of the second byte of the instruction into the stack.																																
PUL (Note 7)	A, B, X···←M(S)	Restores the contents of the stack to the registers among accumulator, index register, direct page register, data bank register, or processor status register, specified by the bit pattern of the second byte of the instruction.																																
RLA (Note 13)	m=0 n bit rotate left  m=1 n bit rotate left  	Rotates the contents of the accumulator A, n bits to the left.				89 49	6 + n	3																											

FOL announced

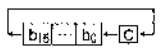
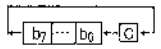
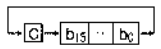
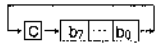
APPENDIX.1 Machine Instructions

Addressing mode																				Processor status register										
L(DIR)	L(DIR),Y	ABS	ABS,b	ABS,X	ABS,Y	ABL	ABL,X	(ABS)	L(ABS)	(ABS,X)	STK	REL	DIR,b,R	ABS,b,R	SR	(SR),Y	BLK				10	9	8	7	6	5	4	3	2	1	0
op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #	op n #				IPL			N	V	m	x	D	I	Z	C
											68 5 1										•	•	•	N	•	•	•	•	•	Z	•
											42 68 7 2										•	•	•	N	•	•	•	•	•	Z	•
											2B 5 1										•	•	•	•	•	•	•	•	•	•	•
											28 6 1										Value saved in stack.										
											AB 6 1										•	•	•	N	•	•	•	•	•	Z	•
											FA 5 1										•	•	•	N	•	•	•	•	•	Z	•
											7A 5 1										•	•	•	N	•	•	•	•	•	Z	•
											EB 12+2i1+i2 2										•	•	•	•	•	•	•	•	•	•	•
											FB 14+3i1+4i2 2										If restored the contents of PS, it becomes its value. And the other case is no change.										
																					•	•	•	•	•	•	•	•	•	•	•

EOL announced

APPENDIX

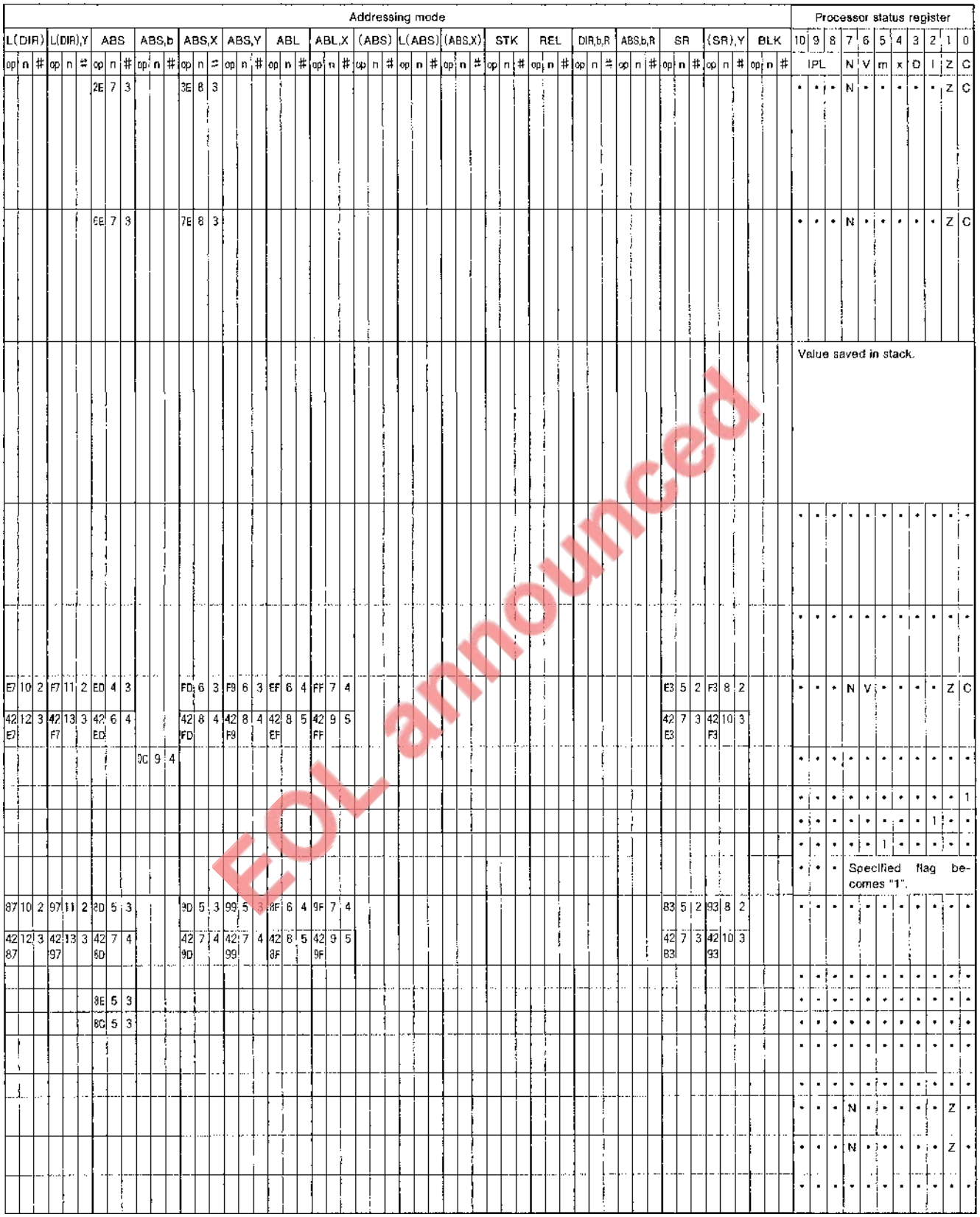
APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																														
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
ROL (Note 1)	m=0  m=1 	Links the accumulator or the memory to C flag, and rotates result to the left by 1 bit.							2A 42 2A	2 4	1 2	26	7	2				36	7	2														
ROR (Note 1)	m=0  m=1 	Links the accumulator or the memory to C flag, and rotates result to the right by 1 bit.							6A 42 6A	2 4	1 2	66	7	2				76	7	2														
RTI	S←S+1 PS_L←M(S) S←S+1 PS_H←M(S) S←S+1 PC_L←M(S) S←S+1 PC_H←M(S) S←S+1 PG←M(S)	Returns from the interruption routine.	40	11	1																												
RTL	S←S+1 PC_L←M(S) S←S+1 PC_H←M(S) S←S+1 PG←M(S)	Returns from the subroutine. The contents of the program bank register are also restored.	6B	8	1																												
RTS	S←S+1 PC_L←M(S) S←S+1 PC_H←M(S)	Returns from the subroutine. The contents of the program bank register are not restored.	60	5	1																												
SBC (Note 1,2)	A_{CC}, C←A_{CC}−M−$\overline{C}$	Subtracts the contents of the memory and the borrow from the contents the accumulator.				E9 42 E9	2 4	2 3				E5 42 E5	4 6	2 3				F5 42 F5	5 7	2 3				F2 42 F2	6 8	2 3	E1 42 E1	7 9	2 3	F1 42 F1	8 10	2 3
SEB (Note 5)	Mb←1	Makes the contents of the specified bit in the memory "1".													04	8	3															
SEC	C←1	Makes the contents of the C flag "1".	38	2	1																												
SEI	I←1	Makes the contents of the I flag "1".	78	2	1																												
SEM	m←1	Makes the contents of the m flag "1".	F8	2	1																												
SEP	PSb←1	Set the specified bit of the processor status register's lower byte (PS L) to "1".				E2	3	2																									
STA (Note 1)	M←A_{CC}	Stores the contents of the accumulator into the memory.										85 42 85	4 6	2 3				95 42 95	5 7	2 3				92 42 92	7 9	2 3	81 42 81	7 9	2 3	91 42 91	7 9	2 3
STP		Stops the oscillation of the oscillator.	DB	3	1																												
STX	M←X	Stores the contents of the index register X into the memory.										86	4	2							96	5	2										
STY	M←Y	Stores the contents of the index register Y into the memory.										84	4	2				94	5	2													
TAD	DPR←A	Transmits the contents of the accumulator A to the direct page register.	5B	2	1																												
TAS	S←A	Transmits the contents of the accumulator A to the stack pointer.	1B	2	1																												
TAX	X←A	Transmits the contents of the accumulator A to the index register X.	AA	2	1																												
TAY	Y←A	Transmits the contents of the accumulator A to the index register Y.	A8	2	1																												
TBD	DPR←B	Transmits the contents of the accumulator B to the direct page register.	42 5B	4	2																												

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions



EOL announced

APPENDIX

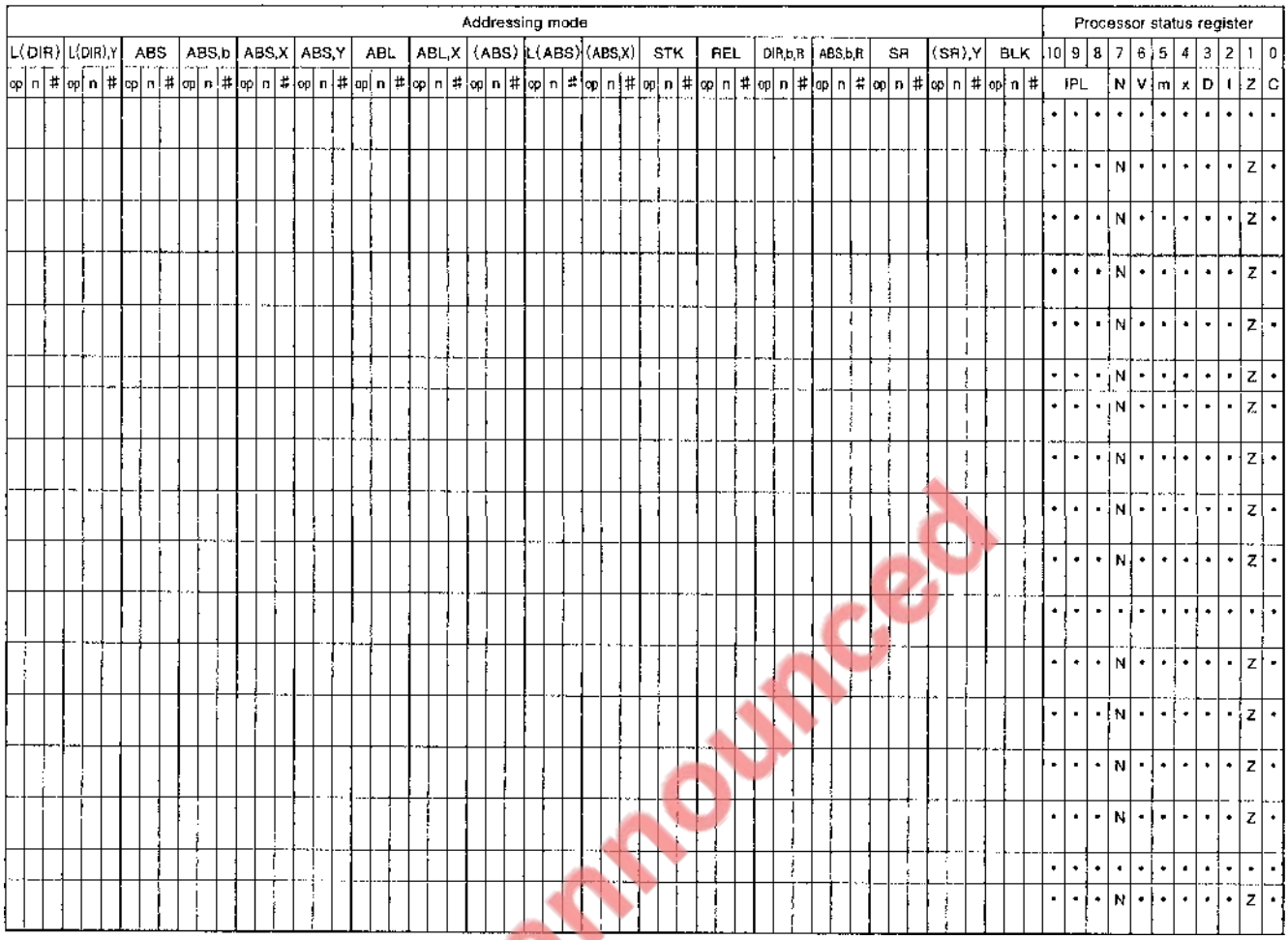
APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																															
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
TBS	S←B	Transmits the contents of the accumulator B to the stack pointer.	42 1B	4	2																													
TBX	X←B	Transmits the contents of the accumulator B to the index register X.	42 AA	4	2																													
TBY	Y←B	Transmits the contents of the accumulator B to the index register Y.	42 A8	4	2																													
TDA	A←DPR	Transmits the contents of the direct page register to the accumulator A.	7B	2	1																													
TDB	B←DPR	Transmits the contents of the direct page register to the accumulator B.	42 7B	4	2																													
TSA	A←S	Transmits the contents of the stack pointer to the accumulator A.	3B	2	1																													
TSB	B←S	Transmits the contents of the stack pointer to the accumulator B.	42 3B	4	2																													
TSX	X←S	Transmits the contents of the stack pointer to the index register X.	BA	2	1																													
TXA	A←X	Transmits the contents of the index register X to the accumulator A.	8A	2	1																													
TXB	B←X	Transmits the contents of the index register X to the accumulator B.	42 8A	4	2																													
TXS	S←X	Transmits the contents of the index register X to the stack pointer.	9A	2	1																													
TXY	Y←X	Transmits the contents of the index register X to the index register Y.	9B	2	1																													
TYA	A←Y	Transmits the contents of the index register Y to the accumulator A.	9B	2	1																													
TYB	B←Y	Transmits the contents of the index register Y to the accumulator B.	42 9B	4	2																													
TYX	X←Y	Transmits the contents of the index register Y to the index register X.	BB	2	1																													
WIT		Stops the internal clock.	CB	3	1																													
XAB	A⇆B	Exchanges the contents of the accumulator A and the contents of the accumulator B.	89 28	6	2																													

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions



EOL announced

APPENDIX

APPENDIX.1 Machine Instructions

The number of cycles shown in the table is described in case of the fastest mode for each instruction. The number of cycles shown in the table is calculated for DPR_L=0. The number of cycles in the addressing mode concerning the DPR when $DPR_L \neq 0$ must be incremented by 1.
The number of cycles shown in the table differs according to the bytes fetched into the instruction queue buffer, or according to whether the memory read/write address is odd or even. It also differs when the external region memory is accessed by BYTE="H".

Note 1. The operation code at the upper row is used for accumulator A, and the operation at the lower row is used for accumulator B.

Note 2. When setting flag m=0 to handle the data as 16-bit data in the immediate addressing mode, the number of bytes increments by 1.

Note 3. The number of cycles increments by 2 when branching.

Note 4. The operation code on the upper row is used for branching in the range of −128~+127, and the operation code on the lower row is used for branching in the range of −32768~+32767.

Note 5. When handling 16-bit data with flag m=0, the byte in the table is incremented by 1.

Note 6.

Type of register	A	B	X	Y	DPR	DT	PG	PS
Number of cycles	2	2	2	2	2	1	1	2

The number of cycles corresponding to the register to be pushed are added. The number of cycles when no pushing is done is 12. i_1 indicates the number of registers among A, B, X, Y, DPR, and PS to be saved, while I_2 indicates the number of registers among DT and PG to be saved.

Note 7.

Type of register	A	B	X	Y	DPR	DT	PS
Number of cycles	3	3	3	3	4	3	3

The number of cycles corresponding to the register to be pulled are added. The number of cycles when no pulling is done is 14. I_1 indicates the number of registers among A, B, X, Y, DT, and PS to be restored, while I_2=1 when DPR is to be restored.

Note 8. The number of cycles is the case when the number of bytes to be transfered is even.
When the number of bytes to be transfered is odd, the number is calculated as;

$$7 + (i/2) \times 7 + 4$$

Note that, (i/2) shows the integer part when i is divided by 2.

Note 9. The number of cycles is the case when the number of bytes to be transfered is even.
When the number of bytes to be transfered is odd, the number is calculated as;

$$9 + (i/2) \times 7 + 5$$

Note that, (i/2) shows the integer part when i is divided by 2.

Note 10. The number of cycles is the case in the 16-bit÷8-bit operation. The number of cycles is incremented by 16 for 32-bit÷16-bit operation.

Note 11. The number of cycles is the case in the 8-bit×8-bit operation. The number of cycles is incremented by 8 for 16-bit ×16-bit operation.

Note 12. When setting flag x=0 to handle the data as 16-bit data in the immediate addressing mode, the number of bytes increments by 1.

Note 13. When flag m is 0, the byte in the table is incremented by 1.

Note 14. The cycles-count in this table are for 8-bit×8-bit multiplications with the same sign. If the multiplier and multiplicand have different signs, three additional cycles are required. For 16-bit×16-bit multiplications, the cycles-count increases by 8.

Note 15. The cycles-count in this table are for 16-bit÷8-bit operations. For 32-bit÷16-bit operations, the cycles-count increases by 16.

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions

Symbols in machine instructions table

Symbol	Description	Symbol	Description
IMP	Implied addressing mode	$\forall$	Exclusive OR
IMM	Immediate addressing mode	$\overline{\ }$	Negation
A	Accumulator addressing mode	$\leftarrow$	Movement to the arrow direction
DIR	Direct addressing mode	A_{CC}	Accumulator
DIR, b	Direct bit addressing mode	A_{CCH}	Accumulator's upper 8 bits
DIR, X	Direct indexed X addressing mode	A_{CCL}	Accumulator's lower 8 bits
DIR, Y	Direct indexed Y addressing mode	A	Accumulator A
(DIR)	Direct indirect addressing mode	A_H	Accumulator A's upper 8 bits
(DIR, X)	Direct indexed X indirect addressing mode	A_L	Accumulator A's lower 8 bits
(DIR), Y	Direct indirect indexed Y addressing mode	B	Accumulator B
L (DIR)	Direct indirect long addressing mode	B_H	Accumulator B's upper 8 bits
L (DIR), Y	Direct indirect long indexed Y addressing mode	B_L	Accumulator B's lower 8 bits
ABS	Absolute addressing mode	X	Index register X
ABS, b	Absolute bit addressing mode	X_H	Index register X's upper 8 bits
ABS, X	Absolute indexed X addressing mode	X_L	Index register X's lower 8 bits
ABS, Y	Absolute indexed Y addressing mode	Y	Index register Y
ABL	Absolute long addressing mode	Y_H	Index register Y's upper 8 bits
ABL, X	Absolute long indexed X addressing mode	Y_L	Index register Y's lower 8 bits
(ABS)	Absolute indirect addressing mode	S	Stack pointer
L (ABS)	Absolute indirect long addressing mode	PC	Program counter
(ABS, X)	Absolute indexed X indirect addressing mode	PC_H	Program counter's upper 8 bits
STK	Stack addressing mode	PC_L	Program counter's lower 8 bits
REL	Relative addressing mode	PG	Program bank register
DIR, b, REL	Direct bit relative addressing mode	DT	Data bank register
ABS, b, REL	Absolute bit relative addressing mode	DPR	Direct page register
SR	Stack pointer relative addressing mode	DPR_H	Direct page register's upper 8 bits
(SR), Y	Stack pointer relative indirect indexed Y addressing mode	DPR_L	Direct page register's lower 8 bits
		PS	Processor status register
BLK	Block transfer addressing mode	PS_H	Processor status register's upper 8 bits
C	Carry flag	PS_L	Processor status register's lower 8 bits
Z	Zero flag	PS_b	Processor status register's b-th bit
I	Interrupt disable flag	M(S)	Contents of memory at address indicated by stack pointer
D	Decimal operation mode flag		
x	Index register length selection flag	M_b	b-th memory location
m	Data length selection flag	AD_G	Value of 24-bit address's upper 8-bit (A_{23}~A_{16})
V	Overflow flag	AD_H	Value of 24-bit address's middle 8-bit (A_{15}~A_8)
N	Negative flag	AD_L	Value of 24-bit address's lower 8-bit (A_7~A_0)
IPL	Processor interrupt priority level	op	Operation code
+	Addition	n	Number of cycle
−	Subtraction	#	Number of byte
*	Multiplication	i	Number of transfer byte or rotation
/	Division	i_1, i_2	Number of registers pushed or pulled
$\wedge$	Logical AND		
$\vee$	Logical OR		

EOL announced

APPENDIX

APPENDIX.2 Hexadecimal Instruction Code Table

INSTRUCTION CODE TABLE - 1

D7—D4 \ D3—D0	Hexadecimal notation	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	BRK	ORA A,(DIR,X)		ORA A,SR	SEB DIR,b	ORA A,DIR	ASL DIR	ORA A,L(DIR)	PHP	ORA A,IMM	ASL A	PHD	SEB ABS,b	ORA A,ABS	ASL ABS	ORA A,ABL
0001	1	BPL	ORA A,(DIR),Y	ORA A,(DIR)	ORA A,(SR),Y	CLB DIR,b	ORA A,DIR,X	ASL DIR,X	ORA A,L(DIR),Y	CLC	ORA A,ABS,Y	DEC A	TAS	CLB ABS,b	ORA A,ABS,X	ASL ABS,X	ORA A,ABL,X
0010	2	JSR ABS	AND A,(DIR,X)	JSR ABL	AND A,SR	BBS DIR,b,R	AND A,DIR	ROL DIR	AND A,L(DIR)	PLP	AND A,IMM	ROL A	PLD	BBS ABS,b,R	AND A,ABS	ROL ABS	AND A,ABL
0011	3	BMI	AND A,(DIR),Y	AND A,(DIR)	AND A,(SR),Y	BBC DIR,b,R	AND A,DIR,X	ROL DIR,X	AND A,L(DIR),Y	SEC	AND A,ABS,Y	INC A	TSA	BBC ABS,b,R	AND A,ABS,X	ROL ABS,X	AND A,ABL,X
0100	4	RTI	EOR A,(DIR,X)	Note 1	EOR A,SR	MVP	EOR A,DIR	LSR DIR	EOR A,L(DIR)	PHA	EOR A,IMM	LSR A	PHG	JMP ABS	EOR A,ABS	LSR ABS	EOR A,ABL
0101	5	BVC	EOR A,(DIR),Y	EOR A,(DIR)	EOR A,(SR),Y	MVN	EOR A,DIR,X	LSR DIR,X	EOR A,L(DIR),Y	CLI	EOR A,ABS,Y	PHY	TAD	JMP ABL	EOR A,ABS,X	LSR ABS,X	EOR A,ABL,X
0110	6	RTS	ADC A,(DIR,X)	PER	ADC A,SR	LDM DIR	ADC A,DIR	ROR DIR	ADC A,L(DIR)	PLA	ADC A,IMM	ROR A	RTL	JMP (ABS)	ADC A,ABS	ROR ABS	ADC A,ABL
0111	7	BVS	ADC A,(DIR),Y	ADC A,(DIR)	ADC A,(SR),Y	LDM DIR,X	ADC A,DIR,X	ROR DIR,X	ADC A,L(DIR),Y	SEI	ADC A,ABS,Y	PLY	TDA	JMP (ABS,X)	ADC A,ABS,X	ROR ABS,X	ADC A,ABL,X
1000	8	BRA REL	STA A,(DIR,X)	BRA REL	STA A,SR	STY DIR	STA A,DIR	STX DIR	STA A,L(DIR)	DEY	Note 2	TXA	PHT	STY ABS	STA A,ABS	STX ABS	STA A,ABL
1001	9	BCC	STA A,(DIR),Y	STA A,(DIR)	STA A,(SR),Y	STY DIR,X	STA A,DIR,X	STX DIR,Y	STA A,L(DIR),Y	TYA	STA A,ABS,Y	TXS	TXY	LDM ABS	STA A,ABS,X	LDM ABS,X	STA A,ABL,X
1010	A	LDY IMM	LDA A,(DIR,X)	LDX IMM	LDA A,SR	LDY DIR	LDA A,DIR	LDX DIR	LDA A,L(DIR)	TAY	LDA A,IMM	TAX	PLT	LDY ABS	LDA A,ABS	LDX ABS	LDA A,ABL
1011	B	BCS	LDA A,(DIR),Y	LDA A,(DIR)	LDA A,(SR),Y	LDY DIR,X	LDA A,DIR,X	LDX DIR,Y	LDA A,L(DIR),Y	CLV	LDA A,ABS,Y	TSX	TYX	LDY ABS,X	LDA A,ABS,X	LDX ABS,Y	LDA A,ABL,X
1100	C	CPY IMM	CMP A,(DIR,X)	CLP IMM	CMP A,SR	CPY DIR	CMP A,DIR	DEC DIR	CMP A,L(DIR)	INY	CMP A,IMM	DEX	WIT	CPY ABS	CMP A,ABS	DEC ABS	CMP A,ABL
1101	D	BNE	CMP A,(DIR),Y	CMP A,(DIR)	CMP A,(SR),Y	PEI	CMP A,DIR,X	DEC DIR,X	CMP A,L(DIR),Y	CLM	CMP A,ABS,Y	PHX	STP	JMP L(ABS)	CMP A,ABS,X	DEC ABS,X	CMP A,ABL,X
1110	E	CPX IMM	SBC A,(DIR,X)	SEP IMM	SBC A,SR	CPX DIR	SBC A,DIR	INC DIR	SBC A,L(DIR)	INX	SBC A,IMM	NOP	PSH	CPX ABS	SBC A,ABS	INC ABS	SBC A,ABL
1111	F	BEQ	SBC A,(DIR),Y	SBC A,(DIR)	SBC A,(SR),Y	PEA	SBC A,DIR,X	INC DIR,X	SBC A,L(DIR),Y	SEM	SBC A,ABS,Y	PLX	PUL	JSR (ABS,X)	SBC A,ABS,X	INC ABS,X	SBC A,ABL,X

Note 1. 42_{16} specifies the contents of the INSTRUCTION CODE TABLE-2. About the second word's codes, refer to the INSTRUCTION CODE TABLE-2.
Note 2. 89_{16} specifies the contents of the INSTRUCTION CODE TABLE-3. About the second word's codes, refer to the INSTRUCTION CODE TABLE-3.

EOL announced

APPENDIX

APPENDIX.2 Hexadecimal Instruction Code Table

INSTRUCTION CODE TABLE - 2 (The first word's code of each instruction is 42_{16})

D_7—D_4 \ D_3—D_0	Hexadecimal notation	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0		ORA B,(DIR,X)		ORA B,SR		ORA B,DIR		ORA B,L(DIR)		ORA B,IMM	ASL B			ORA B,ABS		ORA B,ABL
0001	1		ORA B,(DIR),Y	ORA B,(DIR)	ORA B,(SR),Y		ORA B,DIR,X		ORA B,L(DIR),Y		ORA B,ABS,Y	DEC B	TBS		ORA B,ABS,X		ORA B,ABL,X
0010	2		AND B,(DIR,X)		AND B,SR		AND B,DIR		AND B,L(DIR)		AND B,IMM	ROL B			AND B,ABS		AND B,ABL
0011	3		AND B,(DIR),Y	AND B,(DIR)	AND B,(SR),Y		AND B,DIR,X		AND B,L(DIR),Y		AND B,ABS,Y	INC B	TSB		AND B,ABS,X		AND B,ABL,X
0100	4		EOR B,(DIR,X)		EOR B,SR		EOR B,DIR		EOR B,L(DIR)	PHB	EOR B,IMM	LSR B			EOR B,ABS		EOR B,ABL
0101	5		EOR B,(DIR),Y	EOR B,(DIR)	EOR B,(SR),Y		EOR B,DIR,X		EOR B,L(DIR),Y		EOR B,ABS,Y		TBD		EOR B,ABS,X		EOR B,ABL,X
0110	6		ADC B,(DIR,X)		ADC B,SR		ADC B,DIR		ADC B,L(DIR)	PLB	ADC B,IMM	ROR B			ADC B,ABS		ADC B,ABL
0111	7		ADC B,(DIR),Y	ADC B,(DIR)	ADC B,(SR),Y		ADC B,DIR,X		ADC B,L(DIR),Y		ADC B,ABS,Y		TDB		ADC B,ABS,X		ADC B,ABL,X
1000	8		STA B,(DIR,X)		STA B,SR		STA B,DIR		STA B,L(DIR)			TXB			STA B,ABS		STA B,ABL
1001	9		STA B,(DIR),Y	STA B,(DIR)	STA B,(SR),Y		STA B,DIR,X		STA B,L(DIR),Y	TYB	STA B,ABS,Y				STA B,ABS,X		STA B,ABL,X
1010	A		LDA B,(DIR,X)		LDA B,SR		LDA B,DIR		LDA B,L(DIR)	TBY	LDA B,IMM	TBX			LDA B,ABS		LDA B,ABL
1011	B		LDA B,(DIR),Y	LDA B,(DIR)	LDA B,(SR),Y		LDA B,DIR,X		LDA B,L(DIR),Y		LDA B,ABS,Y				LDA B,ABS,X		LDA B,ABL,X
1100	C		CMP B,(DIR,X)		CMP B,SR		CMP B,DIR		CMP B,L(DIR)		CMP B,IMM				CMP B,ABS		CMP B,ABL
1101	D		CMP B,(DIR),Y	CMP B,(DIR)	CMP B,(SR),Y		CMP B,DIR,X		CMP B,L(DIR),Y		CMP B,ABS,Y				CMP B,ABS,X		CMP B,ABL,X
1110	E		SBC B,(DIR,X)		SBC B,SR		SBC B,DIR		SBC B,L(DIR)		SBC B,IMM				SBC B,ABS		SBC B,ABL
1111	F		SBC B,(DIR),Y	SBC B,(DIR)	SBC B,(SR),Y		SBC B,DIR,X		SBC B,L(DIR),Y		SBC B,ABS,Y				SBC B,ABS,X		SBC B,ABL,X

EOL announced

APPENDIX

APPENDIX.2 Hexadecimal Instruction Code Table

INSTRUCTION CODE TABLE - 3 (The first word's code of each instruction is 89_{16})

D_7—D_4 \ D_3—D_0	Hexadecimal notation	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0		MPY (DIR,X)		MPY SR		MPY DIR		MPY L(DIR)		MPY IMM				MPY ABS		MPY ABL
0001	1		MPY (DIR),Y	MPY (DIR)	MPY (SR),Y		MPY DIR,X		MPY L(DIR),Y		MPY ABS,Y				MPY ABS,X		MPY ABL,X
0010	2		DIV (DIR),X		DIV SR		DIV DIR		DIV L(DIR)	XAB	DIV IMM				DIV ABS		DIV ABL
0011	3		DIV (DIR),Y	DIV (DIR)	DIV (SR),Y		DIV DIR,X		DIV L(DIR),Y		DIV ABS,Y				DIV ABS,X		DIV ABL,X
0100	4										RLA IMM						
0101	5																
0110	6																
0111	7																
1000	8		MPYS* (DIR,X)		MPYS* SR		MPYS* DIR		MPYS* L(DIR)		MPYS* IMM		EXTS* A		MPYS* ABS		MPYS* ABL
1001	9		MPYS* (DIR),Y	MPYS* (DIR)	MPYS* (SR),Y		MPYS* DIR,X		MPYS* L(DIR),Y		MPYS* ABS,Y				MPYS* ABS,X		MPYS* ABL,X
1010	A		DIVS* (DIR),X		DIVS* SR		DIVS* DIR		DIVS* L(DIR)		DIVS* IMM		EXTZ* A		DIVS* ABS		DIVS* ABL
1011	B		DIVS* (DIR),Y	DIVS* (DIR)	DIVS* (SR),Y		DIVS* DIR,X		DIVS* L(DIR),Y		DIVS* ABS,Y				DIVS* ABS,X		DIVS* ABL,X
1100	C			LDT IMM													
1101	D																
1110	E																
1111	F																

EOL announced

Note 1. The code of each instruction first word is 89_{16}.
Note 2. "*" shows the instructions can be used in 7750 Series.

APPENDIX

APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																														
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
ADC (Note 1,2)	A_{CC},C ← A_{CC}+M+C	Adds the carry, the accumulator and the memory contents. The result is entered into the accumulator. When the D flag is "0", binary additions is done, and when the D flag is "1", decimal addition is done.				69	2	2				65	4	2				75	5	2				72	6	2	61	7	2	71	8	2
						42 69	4	3				42 65	6	3				42 75	7	3				42 72	8	3	42 61	9	3	42 71	10	3
AND (Note 1,2)	A_{CC} ← A_{CC} ∧ M	Obtains the logical product of the contents of the accumulator and the contents of the memory. The result is entered into the accumulator.				29	2	2				25	4	2				35	5	2				32	6	2	21	7	2	31	8	2
						42 29	4	3				42 25	6	3				42 35	7	3				42 32	8	3	42 21	9	3	42 31	10	3
ASL (Note 1)	m=0 C ← b_{15} ··· b_0 ← 0 m=1 C ← b_7 ··· b_0 ← 0	Shifts the accumulator or the memory contents one bit to the left. "0" is entered into bit 0 of the accumulator or the memory. The contents of bit 15 (bit 7 when the m flag is "1") of the accumulator or memory before shift is entered into the C flag.							0A	2	1	06	7	2				16	7	2												
									42 0A	4	2																					
ASR* (Note 1)	m=0 → b_{15} ··· b_0 → C m=1 → b_7 ··· b_0 → C	Shifts the accumulator or the memory contents one bit to the right. The bit 0 of the accumulator or memory is entered into the C flag. The contents of bit 15 (bit 7 when the m flag is "1") of the accumulator or memory before shift is entered into bit 15 (bit 7).							89 08	5	2	89 06	9	3				89 16	10	3												
									42 08	5	2																					
BBC (Note 3,5)	Mb=0 ?	Tests the specified bit of the memory. Branches when all the contents of the specified bit is "0".																															
BBS (Note 3,5)	Mb=1 ?	Tests the specified bit of the memory. Branches when all the contents of the specified bit is "1".																															
BCC (Note 3)	C=0 ?	Branches when the contents of the C flag is "0".																															
BCS (Note 3)	C=1 ?	Branches when the contents of the C flag is "1".																															
BEQ (Note 3)	Z=1 ?	Branches when the contents of the Z flag is "1".																															
BMI (Note 3)	N=1 ?	Branches when the contents of the N flag is "1".																															
BNE (Note 3)	Z=0 ?	Branches when the contents of the Z flag is "0".																															
BPL (Note 3)	N=0 ?	Branches when the contents of the N flag is "0".																															
BRA (Note 4)	PC←PC±offset PG←PG+1 (carry occured) PG←PG−1 (borrow occured)	Jumps to the address indicated by the program counter plus the offset value.																															
BRK	PC←PC+2 M(S)←PG S←S−1 M(S)←PC_H S←S−1 M(S)←PC_L S←S−1 M(S)←PS_H S←S−1 M(S)←PS_L S←S−1 I←1 PC_L←AD_L PC_H←AD_H PG←00_{16}	Executes software interruption.	00	15	2																												
BVC (Note 3)	V=0 ?	Branches when the contents of the V flag is "0".																															
BVS (Note 3)	V=1 ?	Branches when the contents of the V flag is "1".																															
CLB (Note 5)	Mb←0	Makes the contents of the specified bit in the memory "0".													14	8	3																
CLC	C←0	Makes the contents of the C flag "0".	18	2	1																												
CLI	I←0	Makes the contents of the I flag "0".	58	2	1																												
CLM	m←0	Makes the contents of the m flag "0".	58	2	1																												

EOL announced

APPENDIX.1 Machine Instructions

Addressing mode																																																						Processor status register										
L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
67	10	2	77	11	2	6D	4	3				7D	6	3	79	6	3	6F	6	4	7F	7	4																						63	5	2	73	8	2				•	•	•	N	V	•	•	•	•	Z	C
42 67	12	3	42 77	13	3	42 6D	6	4				42 7D	8	4	42 79	8	4	42 6F	8	5	42 7F	9	5																						42 63	7	3	42 73	10	3														
27	10	2	37	11	2	2D	4	3				3D	6	3	39	6	3	2F	6	4	3F	7	4																						23	5	2	33	8	2				•	•	•	N	•	•	•	•	•	Z	•
42 27	12	3	42 37	13	3	42 2D	6	4				42 3D	8	4	42 39	8	4	42 2F	8	5	42 3F	9	5																						42 23	7	3	42 33	10	3														
						0E	7	3				1E	8	3																																								•	•	•	N	•	•	•	•	•	Z	C
						89 0E	9	4				89 1E	11	4																																								•	•	•	N	•	•	•	•	•	Z	C
																																							34	7	4	3C	8	5										•	•	•	•	•	•	•	•	•	•	•
																																							24	7	4	2C	8	5										•	•	•	•	•	•	•	•	•	•	•
																																				90	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				B0	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				F0	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				30	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				D0	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				10	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				80	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				82	4	3																										
																																																						•	•	•	•	•	•	•	•	•	•	•
																																				50	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				70	4	2																•	•	•	•	•	•	•	•	•	•	•
									1C	9	4																																													•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	•	•	•	•	•	•	•	0
																																																						•	•	•	•	•	•	•	•	0	•	•
																																																						•	•	•	•	•	0	•	•	•	•	•

APPENDIX

APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																																
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y					
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
CLP	PSb←0	Specifies the bit position in the processor status register by the bit pattern of the second byte in the instruction, and sets "0" in that bit.				C2	4	2																											
CLV	V←0	Makes the contents of the V flag "0".	B8	2	1																														
CMP (Note 1,2)	A_{CC}−M	Compares the contents of the accumulator with the contents of the memory.				C9	2	2				C5	4	2				D5	5	2				D2	6	2	C1	7	2	D1	8	2			
						42 C9	4	3				42 C5	6	3				42 D5	7	3				42 D2	8	3	42 C1	9	3	42 D1	10	3			
CPX (Note 2)	X−M	Compares the contents of the index register X with the contents of the memory.				E0	2	2				E4	4	2																					
CPY (Note 2)	Y−M	Compares the contents of the index register Y with the contents of the memory.				C0	2	2				C4	4	2																					
DEC (Note 1)	A_{CC}←A_{CC}−1 or M←M−1	Decrements the contents of the accumulator or memory by 1.							1A	2	1	C6	7	2				D6	7	2															
									42 1A	4	2																									
DEX	X←X−1	Decrements the contents of the index register X by 1.	CA	2	1																														
DEY	Y←Y−1	Decrements the contents of the index register Y by 1.	88	2	1																														
DIV (Note 2,10)	A(quotient)←B,A/M B(remainder)	The numeral that places the contents of accumulator B to the higher order and the contents of accumulator A to the lower order is divided by the contents of the memory. The quotient is entered into accumulator A and the remainder into accumulator B.				89 29	27	3				89 25	29	3				89 35	30	3				89 32	31	3	89 21	32	3	89 31	33	3			
DIVS★ (Note 2,14)	A(quotient)←B,A/M B(remainder)with sign	The numeral with sign that places the contents of accumulator B to the higher order and the contents of accumulator A to the lower order is divided by the contents of the memory. The quotient is entered into accumulator A and the remainder into accumulator B.				89 A9	29	3				89 A5	31	3				89 B5	32	3				89 B2	33	3	89 A1	34	3	89 B1	35	3			
EOR (Note 1,2)	A_{CC}←A_{CC}¥M	Logical exclusive sum is obtained of the contents of the accumulator and the contents of the memory. The result is placed into the accumulator.				49	2	2				45	4	2				55	5	2				52	6	2	41	7	2	51	8	2			
						42 49	4	3				42 45	6	3				42 55	7	3				42 52	8	3	42 41	9	3	42 51	10	3			
EXTS★ (Note 1)	Bit 7 of A_{CC}=1 b15 b7 11111111 1 Bit 7 of A_{CC}=0 b15 b7 00000000 0	The signed 8-bit data stored in the low-order byte of the accumulator is extended to a 16-bit data.																																	
						89 8B	8	2																											
						42 8B	8	2																											
EXTZ★ (Note 1)	A_{CC} b15 b8 b7 b0 00000000	The 8-bit data stored in the low-order byte of the accumulator is extended to a 16-bit data. Bits 8 to 15 of the accumulator are set to "0".																																	
						89 AB	5	2																											
						42 AB	5	2																											
INC (Note 1)	A_{CC}←A_{CC}+1 or M←M+1	Increments the contents of the accumulator or memory by 1.							3A	2	1	E6	7	2				F6	7	2															
									42 3A	4	2																									
INX	X←X+1	Increments the contents of the index register X by 1.	E8	2	1																														
INY	Y←Y+1	Increments the contents of the index register Y by 1.	C8	2	1																														
JMP	ABS PC_L ← AD_L PC_H ← AD_H ABL PC_L ← AD_L PC_H ← AD_H PG←AD_G (ABS) PC_L ←(AD_H, AD_L) PC_H ←(AD_H, AD_L+1) L(ABS) PC_L ←(AD_H, AD_L) PC_H ←(AD_H, AD_L+1) PG ←(AD_H, AD_L+2) (ABS, X) PC_L ←(AD_H, AD_L+X) PC_H ←(AD_H, AD_L+X +1)	Places a new address into the program counter and jumps to that new address.																																	

PDL announced

APPENDIX

APPENDIX.1 Machine Instructions

Addressing mode

Row	L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X		
	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
1																								
2																								
3	C7	10	2	D7	11	2	CD	4	3				DD	6	3	D9	6	3	CF	6	4	DF	7	4
4	42 C7	12	3	42 D7	13	3	42 CD	6	4				42 DD	8	4	42 D9	8	4	42 CF	8	5	42 DF	9	5
5							EC	4	3															
6							CC	4	3															
7							CE	7	3				DE	8	3									
8																								
9																								
10	89 27	35	3	89 37	36	3	89 2D	29	4				89 3D	31	4	89 39	31	4	89 2F	31	5	89 3F	32	5
11	89 A7	37	3	89 B7	38	3	89 AD	31	4				89 BD	33	4	89 B9	33	4	89 AF	33	5	89 BF	34	5
12	47	10	2	57	11	2	4D	4	3				5D	6	3	59	6	3	4F	6	4	5F	7	4
13	42 47	12	3	42 57	13	3	42 4D	6	4				42 5D	8	4	42 59	8	4	42 4F	8	5	42 5F	9	5
14																								
15																								
16							EE	7	3				FE	8	3									
17																								
18																								
19							4C	2	3										5C	4	4			

Row	(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK		
	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
1																														
2																														
3																						C3	5	2	D3	8	2			
4																						42 C3	7	3	42 D3	10	3			
5																														
6																														
7																														
8																														
9																														
10																						89 23	30	3	89 33	33	3			
11																						89 A3	32	3	89 B3	35	5			
12																						43	5	2	53	8	2			
13																						42 43	7	3	42 53	10	3			
14																														
15																														
16																														
17																														
18																														
19	6C	4	3	DC	8	3	7C	6	3																					

Processor status register

Row	10	9	8	7	6	5	4	3	2	1	0
	IPL			N	V	m	x	D	I	Z	C
1	•	•	•	Specified flag becomes "0".							
2	•	•	•	•	0	•	•	•	•	•	•
3–4	•	•	•	N	•	•	•	•	•	Z	C
5	•	•	•	N	•	•	•	•	•	Z	C
6	•	•	•	N	•	•	•	•	•	Z	C
7	•	•	•	N	•	•	•	•	•	Z	•
8	•	•	•	N	•	•	•	•	•	Z	•
9	•	•	•	N	•	•	•	•	•	Z	•
10	•	•	•	N	V	•	•	•	•	Z	C
11	•	•	•	N	V	•	•	•	•	Z	C
12–13	•	•	•	N	•	•	•	•	•	Z	•
14	•	•	•	N	•	•	•	•	•	Z	•
15	•	•	•	0	•	•	•	•	•	Z	•
16	•	•	•	N	•	•	•	•	•	Z	•
17	•	•	•	N	•	•	•	•	•	Z	•
18	•	•	•	N	•	•	•	•	•	Z	•
19	•	•	•	•	•	•	•	•	•	•	•

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																														
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
JSR	ABS M(S)← PC_H S←S−1 M(S)← PC_L S←S−1 PC_L ← AD_L PC_H ← AD_H ABL M(S)← PG S←S−1 M(S)← PC_H S←S−1 M(S)← PC_L S ← S−1 PC_L ← AD_L PC_H ← AD_H PG ← AD_B (ABS, X) M(S)← PC_H S←S−1 M(S)← PC_L S←S−1 PC_L ←(AD_H, AD_L+X) PC_H ←(AD_H, AD_L+X+1)	Saves the contents of the program counter (also the contents of the program bank register for ABL) into the stack, and jumps to the new address.																																
LDA (Note 1,2)	Acc ← M	Enters the contents of the memory into the accumulator.				A9	2	2				A5	4	2				B5	5	2				B2	6	2	A1	7	2	B1	8	2
						42 A9	4	3				42 A5	6	3				42 B5	7	3				42 B2	8	3	42 A1	9	3	42 B1	10	3
LDM (Note 5)	M ← IMM	Enters the immediate value into the memory.										64	4	3				74	5	3															
LDT	DT ← IMM	Enters the immediate value into the data bank register.				89 C2	5	3																											
LDX (Note 2)	X ← M	Enters the contents of the memory into index register X.				A2	2	2				A6	4	2							B6	5	2												
LDY (Note 2)	Y ← M	Enters the contents of the memory into index register Y.				A0	2	2				A4	4	2				B4	5	2															
LSR (Note 1)	m=0 0→[b_{15}]···[b_0]→C m=1 0→[b_7]···[b_0]→C	Shifts the contents of the accumulator or the contents of the memory one bit to the right. The bit 0 of the accumulator or the memory is entered into the C flag. "0" is entered into bit 15 (bit 7 when the m flag is "1".)							4A	2	1	46	7	2				56	7	2															
									4A	4	2																								
MPY (Note 2,11)	B, A←A*M	Multiplies the contents of accumulator A and the contents of the memory. The higher order of the result of operation are entered into accumulator B, and the lower order into accumulator A.				89 09	16	3				89 05	18	3				89 15	19	3				89 12	20	3	89 01	21	3	89 11	22	3
MPYS* (Note 2,15)	B, A←A*M with sign	The content of the accumulator A is multiplied by the content of memory as signed data. The result is a 32-bit data which is placed in the accumulators B (upper 16 bits of the result) and A (lower 16 bits of the result).				89 89	18	3				89 85	20	3				89 95	21	3				89 92	22	3	89 81	23	3	89 91	24	3
MVN (Note 8)	Mn+i←Mm+i	Transmits the data block. The transmission is done from the lower order address of the block.																																
MVP (Note 9)	Mn−i←Mm−i	Transmits the data block. Transmission is done form the higher order address of the data block.																																
NOP	PC←PC+1	Advances the program counter, but performs nothing else.	EA	2	1																													
ORA (Note 1,2)	Acc←Acc∨M	Logical sum per bit of the contents of the accumulator and the contents of the memory is obtained. The result is entered into the accumulator.				09	2	2				05	4	2				15	5	2				12	6	2	01	7	2	11	8	2
						42 09	4	3				42 05	6	3				42 15	7	3				42 12	8	3	42 01	9	3	42 11	10	3

Not announced

APPENDIX

APPENDIX.1 Machine Instructions

Addressing mode																																																						Processor status register										
L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
						20	6	3										22	8	4										FC	8	3																						•	•	•	•	•	•	•	•	•	•	•
A7	10	2	B7	11	2	AD	4	3				BD	6	3	B9	6	3	AF	6	4	BF	7	4																						A3	5	2	B3	8	2				•	•	•	N	•	•	•	•	•	Z	•
42 A7	12	3	42 B7	13	3	42 AD	6	4				42 BD	8	4	42 B9	8	4	42 AF	8	5	42 BF	9	5																						42 A3	7	3	42 B3	10	3														
						9C	5	4				9E	6	4																																								•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	•	•	•	•	•	•	•	•
						AE	4	3							BE	6	3																																					•	•	•	N	•	•	•	•	•	Z	•
						AC	4	3				BC	6	3																																								•	•	•	N	•	•	•	•	•	Z	•
						4E	7	3				5E	6	3																																								•	•	•	0	•	•	•	•	•	Z	C
89 07	24	3	89 17	25	3	89 0D	18	4				89 1D	20	4	89 19	20	4	89 0F	20	5	89 1F	21	5																						89 03	19	3	89 13	22	3				•	•	•	N	•	•	•	•	•	Z	0
89 87	26	3	89 97	27	3	89 8D	20	4				89 9D	22	4	89 99	22	4	89 8F	22	5	89 9F	23	5																						89 83	21	3	89 93	24	3				•	•	•	N	•	•	•	•	•	Z	0
																																																			54	7 + $\frac{i}{2}\times 7$	3	•	•	•	•	•	•	•	•	•	•	•
																																																			44	9 + $\frac{i}{2}\times 7$	3	•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	•	•	•	•	•	•	•	•
07	10	2	17	11	2	0D	4	3				1D	6	3	19	6	3	0F	6	4	1F	7	4																						03	5	2	13	8	2				•	•	•	N	•	•	•	•	•	Z	•
42 07	12	3	42 17	13	3	42 0D	6	4				42 1D	8	4	42 19	8	4	42 0F	8	5	42 1F	9	5																						42 03	7	3	42 13	10	3														

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																														
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
PEA	M(S)←IMM$_2$ S←S−1 M(S)←IMM$_1$ S←S−1	The 3rd and the 2nd bytes of the instruction are saved into the stack, in this order.																															
PEI	M(S)←M((DPR)+IMM+1) S←S−1 M(S)←M((DPR)+IMM) S←S−1	Specifies 2 sequential bytes in the direct page in the 2nd byte of the instruction, and saves the contents into the stack.																															
PER	EAR←PC+IMM$_2$,IMM$_1$ M(S)←EAR$_H$ S←S−1 M(S)←EAR$_L$ S←S−1	Regards the 2nd and 3rd bytes of the instruction as 16-bit numerals, adds them to the program counter, and saves the result into the stack.																															
PHA	m=0 M(S)←A$_H$ S←S−1 M(S)←A$_L$ S←S−1 m=1 M(S)←A$_L$ S←S−1	Saves the contents of accumulator A into the stack.																															
PHB	m=0 M(S)←B$_H$ S←S−1 M(S)←B$_L$ S←S−1 m=1 M(S)←B$_L$ S←S−1	Saves the contents of accumulator B into the stack.																															
PHD	M(S)←DPR$_H$ S←S−1 M(S)←DPR$_L$ S←S−1	Saves the contents of the direct page register into the stack.																															
PHG	M(S)←PG S←S−1	Saves the contents of the program bank register into the stack.																															
PHP	M(S)←PS$_H$ S←S−1 M(S)←PS$_L$ S←S−1	Saves the contents of the program status register into the stack.																															
PHT	M(S)←DT S←S−1	Saves the contents of the data bank register into the stack.																															
PHX	x=0 M(S)←X$_H$ S←S−1 M(S)←X$_L$ S←S−1 x=1 M(S)←X$_L$ S←S−1	Saves the contents of the index register X into the stack.																															
PHY	x=0 M(S)←Y$_H$ S←S−1 M(S)←Y$_L$ S←S−1 x=1 M(S)←Y$_L$ S←S−1	Saves the contents of the index register Y into the stack.																															

EOL announced

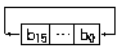
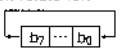
APPENDIX.1 Machine Instructions

Addressing mode																																																						Processor status register										
L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
																																	F4	5	3																			•	•	•	•	•	•	•	•	•	•	•
																																	D4	6	2																			•	•	•	•	•	•	•	•	•	•	•
																																	62	5	3																			•	•	•	•	•	•	•	•	•	•	•
																																	48	4	1																			•	•	•	•	•	•	•	•	•	•	•
																																	42 48	6	2																			•	•	•	•	•	•	•	•	•	•	•
																																	0B	4	1																			•	•	•	•	•	•	•	•	•	•	•
																																	4B	3	1																			•	•	•	•	•	•	•	•	•	•	•
																																	08	4	1																			•	•	•	•	•	•	•	•	•	•	•
																																	8B	3	1																			•	•	•	•	•	•	•	•	•	•	•
																																	DA	4	1																			•	•	•	•	•	•	•	•	•	•	•
																																	5A	4	1																			•	•	•	•	•	•	•	•	•	•	•

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																															
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y				
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#		
PLA	m=0 S←S+1 A_L←M(S) S←S+1 A_H←M(S) m=1 S←S+1 A_L←M(S)	Restores the contents of the stack on the accumulator A.																																
PLB	m=0 S←S+1 B_L←M(S) S←S+1 B_H←M(S) m=1 S←S+1 B_L←M(S)	Restores the contents of the stack on the accumulator B.																																
PLD	S←S+1 DPR_L←M(S) S←S+1 DPR_H←M(S)	Restores the contents of the stack on the direct page register.																																
PLP	S←S+1 PS_L←M(S) S←S+1 PS_H←M(S)	Restores the contents of the stack on the processor status register.																																
PLT	S←S+1 DT←M(S)	Restores the contents of the stack on the data bank register.																																
PLX	x=0 S←S+1 X_L←M(S) S←S+1 X_H←M(S) x=1 S←S+1 X_L←M(S)	Restores the contents of the stack on the index register X.																																
PLY	x=0 S←S+1 Y_L←M(S) S←S+1 Y_H←M(S) x=1 S←S+1 Y_L←M(S)	Restores the contents of the stack on the index register Y.																																
PSH (Note 6)	M(S)←A, B, X···	Saves the registers among accumulator, index register, direct page register, data bank register, program bank register, or processor status register, specified by the bit pattern of the second byte of the instruction into the stack.																																
PUL (Note 7)	A, B, X···←M(S)	Restores the contents of the stack to the registers among accumulator, index register, direct page register, data bank register, or processor status register, specified by the bit pattern of the second byte of the instruction.																																
RLA (Note 13)	m=0 n bit rotate left  m=1 n bit rotate left  	Rotates the contents of the accumulator A, n bits to the left.				89 49	6 + n	3																											

FOL announced

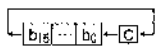
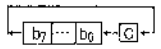
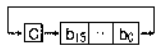
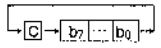
APPENDIX.1 Machine Instructions

Addressing mode																																																						Processor status register										
L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
																																	68	5	1																			•	•	•	N	•	•	•	•	•	Z	•
																																	42 68	7	2																			•	•	•	N	•	•	•	•	•	Z	•
																																	2B	5	1																			•	•	•	•	•	•	•	•	•	•	•
																																	2B	6	1																			Value saved in stack.										
																																	AB	6	1																			•	•	•	N	•	•	•	•	•	Z	•
																																	FA	5	1																			•	•	•	N	•	•	•	•	•	Z	•
																																	7A	5	1																			•	•	•	N	•	•	•	•	•	Z	•
																																	EB	12 + $2i_1+i_2$	2																			•	•	•	•	•	•	•	•	•	•	•
																																	FB	14 + $3i_1+4i_2$	2																			If restored the contents of PS, it becomes its value. And the other case is no change.										
																																																						•	•	•	•	•	•	•	•	•	•	•

EOL announced

APPENDIX

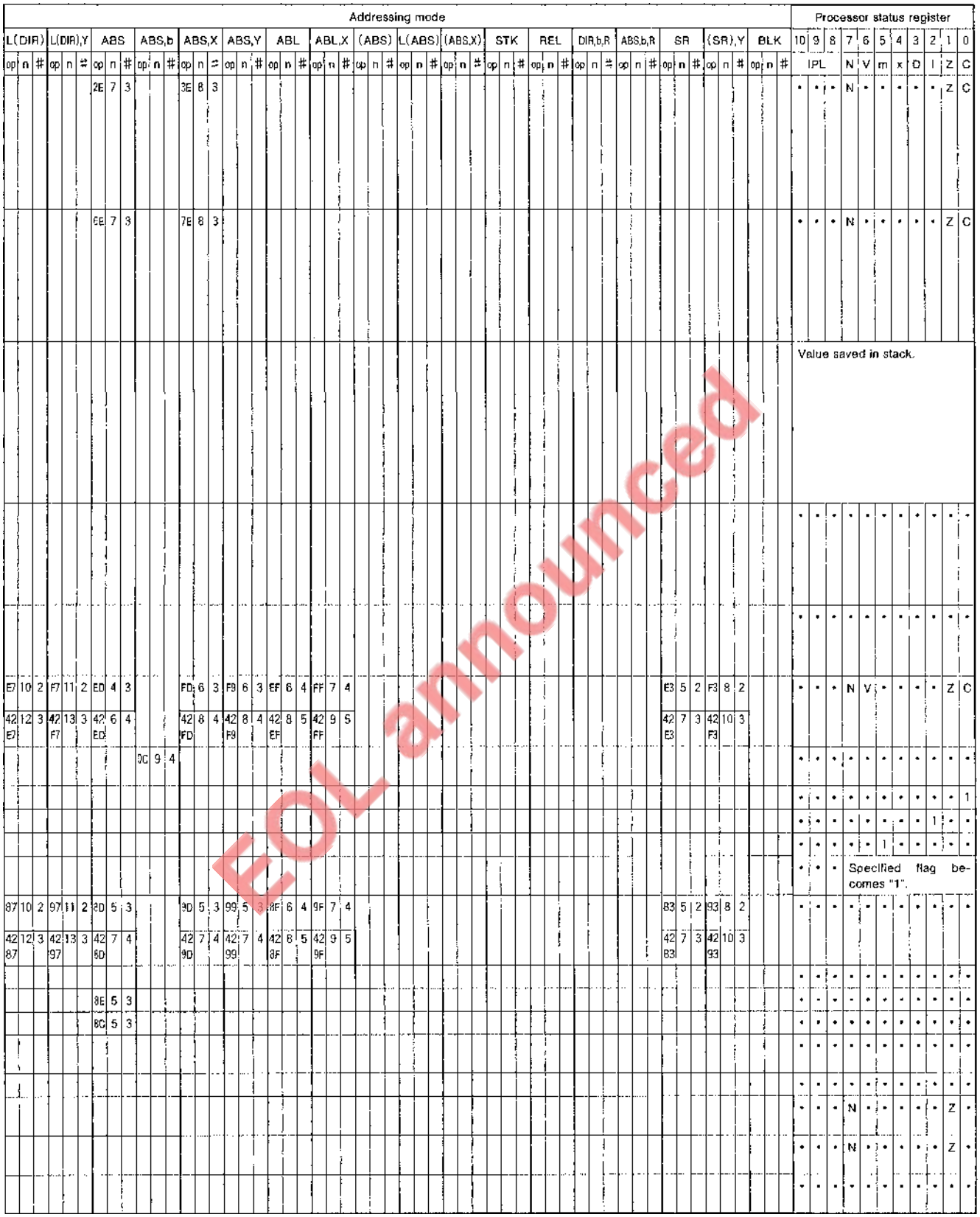
APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																														
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
ROL (Note 1)	m=0  m=1 	Links the accumulator or the memory to C flag, and rotates result to the left by 1 bit.							2A 42 2A	2 4	1 2	26	7	2				36	7	2														
ROR (Note 1)	m=0  m=1 	Links the accumulator or the memory to C flag, and rotates result to the right by 1 bit.							6A 42 6A	2 4	1 2	66	7	2				76	7	2														
RTI	S←S+1 PS_L←M(S) S←S+1 PS_H←M(S) S←S+1 PC_L←M(S) S←S+1 PC_H←M(S) S←S+1 PG←M(S)	Returns from the interruption routine.	40	11	1																												
RTL	S←S+1 PC_L←M(S) S←S+1 PC_H←M(S) S←S+1 PG←M(S)	Returns from the subroutine. The contents of the program bank register are also restored.	6B	8	1																												
RTS	S←S+1 PC_L←M(S) S←S+1 PC_H←M(S)	Returns from the subroutine. The contents of the program bank register are not restored.	60	5	1																												
SBC (Note 1,2)	A_{CC}, C←A_{CC}−M−$\overline{C}$	Subtracts the contents of the memory and the borrow from the contents the accumulator.				E9 42 E9	2 4	2 3				E5 42 E5	4 6	2 3				F5 42 F5	5 7	2 3				F2 42 F2	6 8	2 3	E1 42 E1	7 9	2 3	F1 42 F1	8 10	2 3
SEB (Note 5)	Mb←1	Makes the contents of the specified bit in the memory "1".													04	8	3															
SEC	C←1	Makes the contents of the C flag "1".	38	2	1																												
SEI	I←1	Makes the contents of the I flag "1".	78	2	1																												
SEM	m←1	Makes the contents of the m flag "1".	F8	2	1																												
SEP	PSb←1	Set the specified bit of the processor status register's lower byte (PS L) to "1".				E2	3	2																									
STA (Note 1)	M←A_{CC}	Stores the contents of the accumulator into the memory.										85 42 85	4 6	2 3				95 42 95	5 7	2 3				92 42 92	7 9	2 3	81 42 81	7 9	2 3	91 42 91	7 9	2 3
STP		Stops the oscillation of the oscillator.	DB	3	1																												
STX	M←X	Stores the contents of the index register X into the memory.										86	4	2							96	5	2										
STY	M←Y	Stores the contents of the index register Y into the memory.										84	4	2				94	5	2													
TAD	DPR←A	Transmits the contents of the accumulator A to the direct page register.	5B	2	1																												
TAS	S←A	Transmits the contents of the accumulator A to the stack pointer.	1B	2	1																												
TAX	X←A	Transmits the contents of the accumulator A to the index register X.	AA	2	1																												
TAY	Y←A	Transmits the contents of the accumulator A to the index register Y.	A8	2	1																												
TBD	DPR←B	Transmits the contents of the accumulator B to the direct page register.	42 5B	4	2																												

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions



EOL announced

APPENDIX

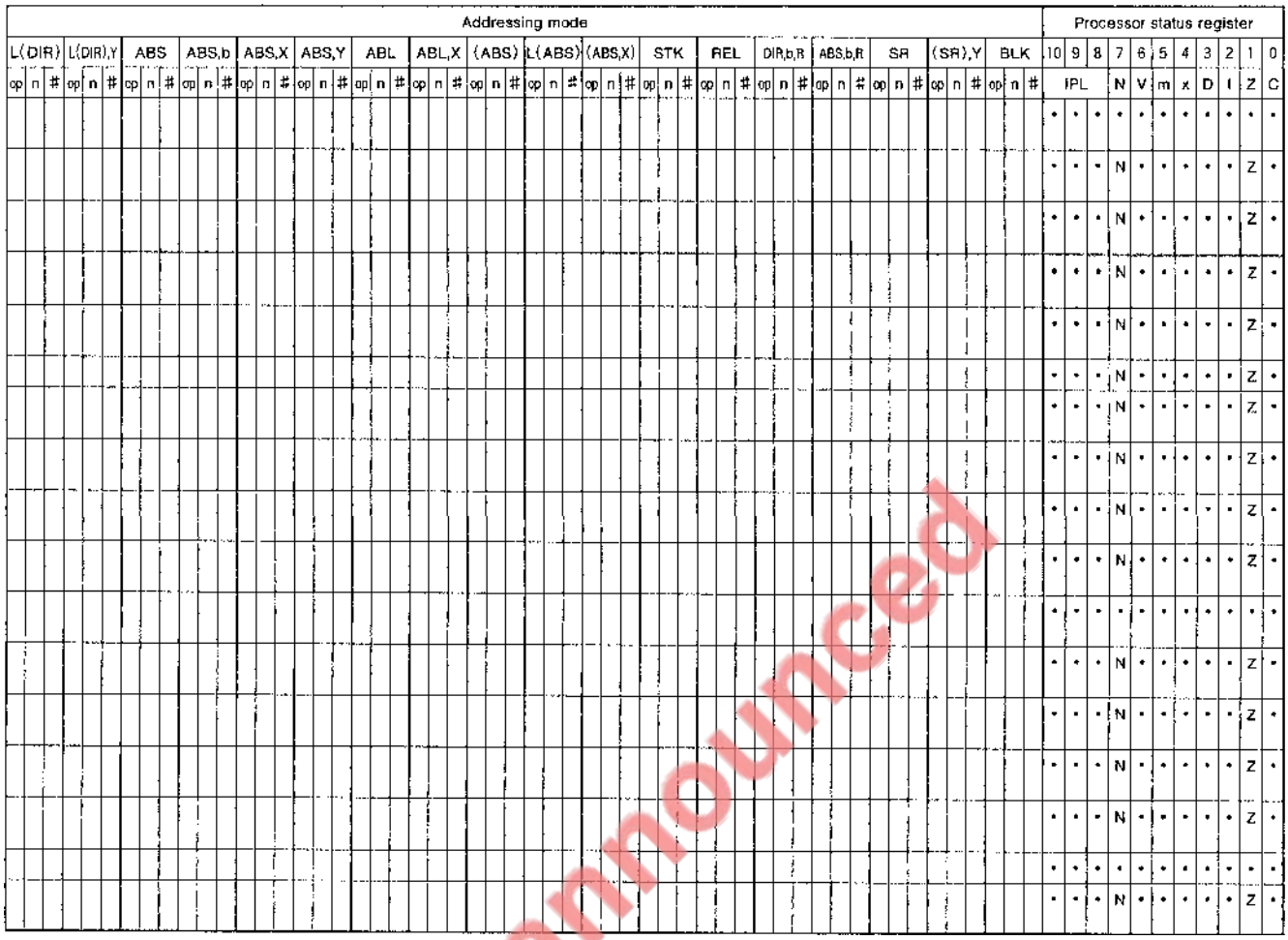
APPENDIX.1 Machine Instructions

Symbol	Function	Details	Addressing mode																															
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
TBS	S←B	Transmits the contents of the accumulator B to the stack pointer.	42 1B	4	2																													
TBX	X←B	Transmits the contents of the accumulator B to the index register X.	42 AA	4	2																													
TBY	Y←B	Transmits the contents of the accumulator B to the index register Y.	42 A8	4	2																													
TDA	A←DPR	Transmits the contents of the direct page register to the accumulator A.	7B	2	1																													
TDB	B←DPR	Transmits the contents of the direct page register to the accumulator B.	42 7B	4	2																													
TSA	A←S	Transmits the contents of the stack pointer to the accumulator A.	3B	2	1																													
TSB	B←S	Transmits the contents of the stack pointer to the accumulator B.	42 3B	4	2																													
TSX	X←S	Transmits the contents of the stack pointer to the index register X.	BA	2	1																													
TXA	A←X	Transmits the contents of the index register X to the accumulator A.	8A	2	1																													
TXB	B←X	Transmits the contents of the index register X to the accumulator B.	42 8A	4	2																													
TXS	S←X	Transmits the contents of the index register X to the stack pointer.	9A	2	1																													
TXY	Y←X	Transmits the contents of the index register X to the index register Y.	9B	2	1																													
TYA	A←Y	Transmits the contents of the index register Y to the accumulator A.	9B	2	1																													
TYB	B←Y	Transmits the contents of the index register Y to the accumulator B.	42 9B	4	2																													
TYX	X←Y	Transmits the contents of the index register Y to the index register X.	BB	2	1																													
WIT		Stops the internal clock.	CB	3	1																													
XAB	A⇆B	Exchanges the contents of the accumulator A and the contents of the accumulator B.	89 28	6	2																													

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions



EOL announced

APPENDIX

APPENDIX.1 Machine Instructions

The number of cycles shown in the table is described in case of the fastest mode for each instruction. The number of cycles shown in the table is calculated for DPR_L=0. The number of cycles in the addressing mode concerning the DPR when $DPR_L \neq 0$ must be incremented by 1.
The number of cycles shown in the table differs according to the bytes fetched into the instruction queue buffer, or according to whether the memory read/write address is odd or even. It also differs when the external region memory is accessed by BYTE="H".

Note 1. The operation code at the upper row is used for accumulator A, and the operation at the lower row is used for accumulator B.

Note 2. When setting flag m=0 to handle the data as 16-bit data in the immediate addressing mode, the number of bytes increments by 1.

Note 3. The number of cycles increments by 2 when branching.

Note 4. The operation code on the upper row is used for branching in the range of −128~+127, and the operation code on the lower row is used for branching in the range of −32768~+32767.

Note 5. When handling 16-bit data with flag m=0, the byte in the table is incremented by 1.

Note 6.

Type of register	A	B	X	Y	DPR	DT	PG	PS
Number of cycles	2	2	2	2	2	1	1	2

The number of cycles corresponding to the register to be pushed are added. The number of cycles when no pushing is done is 12. i_1 indicates the number of registers among A, B, X, Y, DPR, and PS to be saved, while I_2 indicates the number of registers among DT and PG to be saved.

Note 7.

Type of register	A	B	X	Y	DPR	DT	PS
Number of cycles	3	3	3	3	4	3	3

The number of cycles corresponding to the register to be pulled are added. The number of cycles when no pulling is done is 14. I_1 indicates the number of registers among A, B, X, Y, DT, and PS to be restored, while I_2=1 when DPR is to be restored.

Note 8. The number of cycles is the case when the number of bytes to be transfered is even.
When the number of bytes to be transfered is odd, the number is calculated as;

$$7 + (i/2) \times 7 + 4$$

Note that, (i/2) shows the integer part when i is divided by 2.

Note 9. The number of cycles is the case when the number of bytes to be transfered is even.
When the number of bytes to be transfered is odd, the number is calculated as;

$$9 + (i/2) \times 7 + 5$$

Note that, (i/2) shows the integer part when i is divided by 2.

Note 10. The number of cycles is the case in the 16-bit÷8-bit operation. The number of cycles is incremented by 16 for 32-bit÷16-bit operation.

Note 11. The number of cycles is the case in the 8-bit×8-bit operation. The number of cycles is incremented by 8 for 16-bit ×16-bit operation.

Note 12. When setting flag x=0 to handle the data as 16-bit data in the immediate addressing mode, the number of bytes increments by 1.

Note 13. When flag m is 0, the byte in the table is incremented by 1.

Note 14. The cycles-count in this table are for 8-bit×8-bit multiplications with the same sign. If the multiplier and multiplicand have different signs, three additional cycles are required. For 16-bit×16-bit multiplications, the cycles-count increases by 8.

Note 15. The cycles-count in this table are for 16-bit÷8-bit operations. For 32-bit÷16-bit operations, the cycles-count increases by 16.

EOL announced

APPENDIX

APPENDIX.1 Machine Instructions

Symbols in machine instructions table

Symbol	Description	Symbol	Description
IMP	Implied addressing mode	$\forall$	Exclusive OR
IMM	Immediate addressing mode	$\overline{\ }$	Negation
A	Accumulator addressing mode	$\leftarrow$	Movement to the arrow direction
DIR	Direct addressing mode	A_{CC}	Accumulator
DIR, b	Direct bit addressing mode	A_{CCH}	Accumulator's upper 8 bits
DIR, X	Direct indexed X addressing mode	A_{CCL}	Accumulator's lower 8 bits
DIR, Y	Direct indexed Y addressing mode	A	Accumulator A
(DIR)	Direct indirect addressing mode	A_H	Accumulator A's upper 8 bits
(DIR, X)	Direct indexed X indirect addressing mode	A_L	Accumulator A's lower 8 bits
(DIR), Y	Direct indirect indexed Y addressing mode	B	Accumulator B
L (DIR)	Direct indirect long addressing mode	B_H	Accumulator B's upper 8 bits
L (DIR), Y	Direct indirect long indexed Y addressing mode	B_L	Accumulator B's lower 8 bits
ABS	Absolute addressing mode	X	Index register X
ABS, b	Absolute bit addressing mode	X_H	Index register X's upper 8 bits
ABS, X	Absolute indexed X addressing mode	X_L	Index register X's lower 8 bits
ABS, Y	Absolute indexed Y addressing mode	Y	Index register Y
ABL	Absolute long addressing mode	Y_H	Index register Y's upper 8 bits
ABL, X	Absolute long indexed X addressing mode	Y_L	Index register Y's lower 8 bits
(ABS)	Absolute indirect addressing mode	S	Stack pointer
L (ABS)	Absolute indirect long addressing mode	PC	Program counter
(ABS, X)	Absolute indexed X indirect addressing mode	PC_H	Program counter's upper 8 bits
STK	Stack addressing mode	PC_L	Program counter's lower 8 bits
REL	Relative addressing mode	PG	Program bank register
DIR, b, REL	Direct bit relative addressing mode	DT	Data bank register
ABS, b, REL	Absolute bit relative addressing mode	DPR	Direct page register
SR	Stack pointer relative addressing mode	DPR_H	Direct page register's upper 8 bits
(SR), Y	Stack pointer relative indirect indexed Y addressing mode	DPR_L	Direct page register's lower 8 bits
		PS	Processor status register
BLK	Block transfer addressing mode	PS_H	Processor status register's upper 8 bits
C	Carry flag	PS_L	Processor status register's lower 8 bits
Z	Zero flag	PS_b	Processor status register's b-th bit
I	Interrupt disable flag	M(S)	Contents of memory at address indicated by stack pointer
D	Decimal operation mode flag		
x	Index register length selection flag	M_b	b-th memory location
m	Data length selection flag	AD_G	Value of 24-bit address's upper 8-bit (A_{23}~A_{16})
V	Overflow flag	AD_H	Value of 24-bit address's middle 8-bit (A_{15}~A_8)
N	Negative flag	AD_L	Value of 24-bit address's lower 8-bit (A_7~A_0)
IPL	Processor interrupt priority level	op	Operation code
+	Addition	n	Number of cycle
−	Subtraction	#	Number of byte
*	Multiplication	i	Number of transfer byte or rotation
/	Division	i_1, i_2	Number of registers pushed or pulled
$\wedge$	Logical AND		
$\vee$	Logical OR		

EOL announced

APPENDIX

APPENDIX.2 Hexadecimal Instruction Code Table

INSTRUCTION CODE TABLE - 1

D7—D4 \ D3—D0	Hexadecimal notation	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	BRK	ORA A,(DIR,X)		ORA A,SR	SEB DIR,b	ORA A,DIR	ASL DIR	ORA A,L(DIR)	PHP	ORA A,IMM	ASL A	PHD	SEB ABS,b	ORA A,ABS	ASL ABS	ORA A,ABL
0001	1	BPL	ORA A,(DIR),Y	ORA A,(DIR)	ORA A,(SR),Y	CLB DIR,b	ORA A,DIR,X	ASL DIR,X	ORA A,L(DIR),Y	CLC	ORA A,ABS,Y	DEC A	TAS	CLB ABS,b	ORA A,ABS,X	ASL ABS,X	ORA A,ABL,X
0010	2	JSR ABS	AND A,(DIR,X)	JSR ABL	AND A,SR	BBS DIR,b,R	AND A,DIR	ROL DIR	AND A,L(DIR)	PLP	AND A,IMM	ROL A	PLD	BBS ABS,b,R	AND A,ABS	ROL ABS	AND A,ABL
0011	3	BMI	AND A,(DIR),Y	AND A,(DIR)	AND A,(SR),Y	BBC DIR,b,R	AND A,DIR,X	ROL DIR,X	AND A,L(DIR),Y	SEC	AND A,ABS,Y	INC A	TSA	BBC ABS,b,R	AND A,ABS,X	ROL ABS,X	AND A,ABL,X
0100	4	RTI	EOR A,(DIR,X)	Note 1	EOR A,SR	MVP	EOR A,DIR	LSR DIR	EOR A,L(DIR)	PHA	EOR A,IMM	LSR A	PHG	JMP ABS	EOR A,ABS	LSR ABS	EOR A,ABL
0101	5	BVC	EOR A,(DIR),Y	EOR A,(DIR)	EOR A,(SR),Y	MVN	EOR A,DIR,X	LSR DIR,X	EOR A,L(DIR),Y	CLI	EOR A,ABS,Y	PHY	TAD	JMP ABL	EOR A,ABS,X	LSR ABS,X	EOR A,ABL,X
0110	6	RTS	ADC A,(DIR,X)	PER	ADC A,SR	LDM DIR	ADC A,DIR	ROR DIR	ADC A,L(DIR)	PLA	ADC A,IMM	ROR A	RTL	JMP (ABS)	ADC A,ABS	ROR ABS	ADC A,ABL
0111	7	BVS	ADC A,(DIR),Y	ADC A,(DIR)	ADC A,(SR),Y	LDM DIR,X	ADC A,DIR,X	ROR DIR,X	ADC A,L(DIR),Y	SEI	ADC A,ABS,Y	PLY	TDA	JMP (ABS,X)	ADC A,ABS,X	ROR ABS,X	ADC A,ABL,X
1000	8	BRA REL	STA A,(DIR,X)	BRA REL	STA A,SR	STY DIR	STA A,DIR	STX DIR	STA A,L(DIR)	DEY	Note 2	TXA	PHT	STY ABS	STA A,ABS	STX ABS	STA A,ABL
1001	9	BCC	STA A,(DIR),Y	STA A,(DIR)	STA A,(SR),Y	STY DIR,X	STA A,DIR,X	STX DIR,Y	STA A,L(DIR),Y	TYA	STA A,ABS,Y	TXS	TXY	LDM ABS	STA A,ABS,X	LDM ABS,X	STA A,ABL,X
1010	A	LDY IMM	LDA A,(DIR,X)	LDX IMM	LDA A,SR	LDY DIR	LDA A,DIR	LDX DIR	LDA A,L(DIR)	TAY	LDA A,IMM	TAX	PLT	LDY ABS	LDA A,ABS	LDX ABS	LDA A,ABL
1011	B	BCS	LDA A,(DIR),Y	LDA A,(DIR)	LDA A,(SR),Y	LDY DIR,X	LDA A,DIR,X	LDX DIR,Y	LDA A,L(DIR),Y	CLV	LDA A,ABS,Y	TSX	TYX	LDY ABS,X	LDA A,ABS,X	LDX ABS,Y	LDA A,ABL,X
1100	C	CPY IMM	CMP A,(DIR,X)	CLP IMM	CMP A,SR	CPY DIR	CMP A,DIR	DEC DIR	CMP A,L(DIR)	INY	CMP A,IMM	DEX	WIT	CPY ABS	CMP A,ABS	DEC ABS	CMP A,ABL
1101	D	BNE	CMP A,(DIR),Y	CMP A,(DIR)	CMP A,(SR),Y	PEI	CMP A,DIR,X	DEC DIR,X	CMP A,L(DIR),Y	CLM	CMP A,ABS,Y	PHX	STP	JMP L(ABS)	CMP A,ABS,X	DEC ABS,X	CMP A,ABL,X
1110	E	CPX IMM	SBC A,(DIR,X)	SEP IMM	SBC A,SR	CPX DIR	SBC A,DIR	INC DIR	SBC A,L(DIR)	INX	SBC A,IMM	NOP	PSH	CPX ABS	SBC A,ABS	INC ABS	SBC A,ABL
1111	F	BEQ	SBC A,(DIR),Y	SBC A,(DIR)	SBC A,(SR),Y	PEA	SBC A,DIR,X	INC DIR,X	SBC A,L(DIR),Y	SEM	SBC A,ABS,Y	PLX	PUL	JSR (ABS,X)	SBC A,ABS,X	INC ABS,X	SBC A,ABL,X

Note 1. 42_{16} specifies the contents of the INSTRUCTION CODE TABLE-2. About the second word's codes, refer to the INSTRUCTION CODE TABLE-2.
Note 2. 89_{16} specifies the contents of the INSTRUCTION CODE TABLE-3. About the second word's codes, refer to the INSTRUCTION CODE TABLE-3.

EOL announced

APPENDIX

APPENDIX.2 Hexadecimal Instruction Code Table

INSTRUCTION CODE TABLE - 2 (The first word's code of each instruction is 42_{16})

D_7—D_4 \ D_3—D_0	Hexadecimal notation	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0		ORA B,(DIR,X)		ORA B,SR		ORA B,DIR		ORA B,L(DIR)		ORA B,IMM	ASL B			ORA B,ABS		ORA B,ABL
0001	1		ORA B,(DIR),Y	ORA B,(DIR)	ORA B,(SR),Y		ORA B,DIR,X		ORA B,L(DIR),Y		ORA B,ABS,Y	DEC B	TBS		ORA B,ABS,X		ORA B,ABL,X
0010	2		AND B,(DIR,X)		AND B,SR		AND B,DIR		AND B,L(DIR)		AND B,IMM	ROL B			AND B,ABS		AND B,ABL
0011	3		AND B,(DIR),Y	AND B,(DIR)	AND B,(SR),Y		AND B,DIR,X		AND B,L(DIR),Y		AND B,ABS,Y	INC B	TSB		AND B,ABS,X		AND B,ABL,X
0100	4		EOR B,(DIR,X)		EOR B,SR		EOR B,DIR		EOR B,L(DIR)	PHB	EOR B,IMM	LSR B			EOR B,ABS		EOR B,ABL
0101	5		EOR B,(DIR),Y	EOR B,(DIR)	EOR B,(SR),Y		EOR B,DIR,X		EOR B,L(DIR),Y		EOR B,ABS,Y		TBD		EOR B,ABS,X		EOR B,ABL,X
0110	6		ADC B,(DIR,X)		ADC B,SR		ADC B,DIR		ADC B,L(DIR)	PLB	ADC B,IMM	ROR B			ADC B,ABS		ADC B,ABL
0111	7		ADC B,(DIR),Y	ADC B,(DIR)	ADC B,(SR),Y		ADC B,DIR,X		ADC B,L(DIR),Y		ADC B,ABS,Y		TDB		ADC B,ABS,X		ADC B,ABL,X
1000	8		STA B,(DIR,X)		STA B,SR		STA B,DIR		STA B,L(DIR)			TXB			STA B,ABS		STA B,ABL
1001	9		STA B,(DIR),Y	STA B,(DIR)	STA B,(SR),Y		STA B,DIR,X		STA B,L(DIR),Y	TYB	STA B,ABS,Y				STA B,ABS,X		STA B,ABL,X
1010	A		LDA B,(DIR,X)		LDA B,SR		LDA B,DIR		LDA B,L(DIR)	TBY	LDA B,IMM	TBX			LDA B,ABS		LDA B,ABL
1011	B		LDA B,(DIR),Y	LDA B,(DIR)	LDA B,(SR),Y		LDA B,DIR,X		LDA B,L(DIR),Y		LDA B,ABS,Y				LDA B,ABS,X		LDA B,ABL,X
1100	C		CMP B,(DIR,X)		CMP B,SR		CMP B,DIR		CMP B,L(DIR)		CMP B,IMM				CMP B,ABS		CMP B,ABL
1101	D		CMP B,(DIR),Y	CMP B,(DIR)	CMP B,(SR),Y		CMP B,DIR,X		CMP B,L(DIR),Y		CMP B,ABS,Y				CMP B,ABS,X		CMP B,ABL,X
1110	E		SBC B,(DIR,X)		SBC B,SR		SBC B,DIR		SBC B,L(DIR)		SBC B,IMM				SBC B,ABS		SBC B,ABL
1111	F		SBC B,(DIR),Y	SBC B,(DIR)	SBC B,(SR),Y		SBC B,DIR,X		SBC B,L(DIR),Y		SBC B,ABS,Y				SBC B,ABS,X		SBC B,ABL,X

EOL announced

APPENDIX

APPENDIX.2 Hexadecimal Instruction Code Table

INSTRUCTION CODE TABLE - 3 (The first word's code of each instruction is 89_{16})

D_7—D_4 \ D_3—D_0	Hexadecimal notation	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0		MPY (DIR,X)		MPY SR		MPY DIR		MPY L(DIR)		MPY IMM				MPY ABS		MPY ABL
0001	1		MPY (DIR),Y	MPY (DIR)	MPY (SR),Y		MPY DIR,X		MPY L(DIR),Y		MPY ABS,Y				MPY ABS,X		MPY ABL,X
0010	2		DIV (DIR),X		DIV SR		DIV DIR		DIV L(DIR)	XAB	DIV IMM				DIV ABS		DIV ABL
0011	3		DIV (DIR),Y	DIV (DIR)	DIV (SR),Y		DIV DIR,X		DIV L(DIR),Y		DIV ABS,Y				DIV ABS,X		DIV ABL,X
0100	4										RLA IMM						
0101	5																
0110	6																
0111	7																
1000	8		MPYS* (DIR,X)		MPYS* SR		MPYS* DIR		MPYS* L(DIR)		MPYS* IMM		EXTS* A		MPYS* ABS		MPYS* ABL
1001	9		MPYS* (DIR),Y	MPYS* (DIR)	MPYS* (SR),Y		MPYS* DIR,X		MPYS* L(DIR),Y		MPYS* ABS,Y				MPYS* ABS,X		MPYS* ABL,X
1010	A		DIVS* (DIR),X		DIVS* SR		DIVS* DIR		DIVS* L(DIR)		DIVS* IMM		EXTZ* A		DIVS* ABS		DIVS* ABL
1011	B		DIVS* (DIR),Y	DIVS* (DIR)	DIVS* (SR),Y		DIVS* DIR,X		DIVS* L(DIR),Y		DIVS* ABS,Y				DIVS* ABS,X		DIVS* ABL,X
1100	C			LDT IMM													
1101	D																
1110	E																
1111	F																

EOL announced

Note 1. The code of each instruction first word is 89_{16}.
Note 2. "*" shows the instructions can be used in 7750 Series.

EOL announced

MITSUBISHI SEMICONDUCTORS
7700 Family Software MANUAL

Sep. First Edition 1994

Editioned by
Committee of editing of Mitsubishi Semiconductor USER'S MANUAL

Published by
Mitsubishi Electric Corp., Semiconductor Marketing Division

This book, or parts thereof, may not be reproduced in any form without permission of Mitsubishi Electric Corporation.
©1994 MITSUBISHI ELECTRIC CORPORATION

7700 Family
Software Manual

EOL announced

Renesas Electronics Corporation
1753, Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8668 Japan