お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

740 ファミリ

740 ファミリ参考プログラム集

目次

1. ７４０ファミリの命令セットの特長

７４０ファミリの命令の特長は以下のようにまとめることができます。

(1) ＲＯＭ領域が有効に使える効率的な命令群と豊富なアドレッシングモード。
(2) アキュムレータ、メモリ、Ｉ／Ｏを問わず処理できるビット操作命令と、ビットテスト／ブランチ命令。
(3) 豊富な割り込みソースと処理機能。
(4) バイト単位処理、テーブル参照機能に優れたインデックス アドレッシング機能。
(5) ソフトウェア補正を必要としない１０進演算機能。
(6) アキュムレータを経由せずに実行のできるメモリ間、Ｉ／Ｏ間、そしてメモリとＩ／Ｏ間の演算機能。

2. ７４０ファミリの固有命令を生かした処理

2.1 メモリ間の演算

（1） 加算（ニーモニック ＡＤＣ）

ＡＤＣ命令は、Ｘ修飾演算モードフラグＴが１のとき、Ｍ（Ｘ）とメモリＭとキャリーフラグＣの内容を加算して、結果をＭ（Ｘ）及びキャリーフラグＣに入れます。このとき、アキュムレータＡの内容は変化しませんが、ステータスフラグは変化します。ただし、Ｍ（Ｘ）はインデックスレジスタＸが示す番地のメモリです。

例）

```
        ＣＬＣ                      ①
        ＳＥＴ                      ②
        ＬＤＸ      ＃ＡＤＤＡＴＡ   ③
        ＡＤＣ      ＄１０，Ｘ       ④
        ＣＬＴ                      ⑤
```

（説明）

① キャリーフラグＣの内容を０にします。

② Ｘ修飾演算モードフラグＴの内容を１にします。

③ ＃ＡＤＤＡＴＡ（例えば７０Ｈ）を、インデックスレジスタＸにロードします。

④ メモリの７０Ｈ番地の内容と８０Ｈ番地（７０Ｈ＋１０Ｈ）の内容とキャリーフラグＣの内容を加算して、結果を７０Ｈ番地及びキャリーフラグＣに入れます。

７０Ｈ番地 ← ７０Ｈ番地 ＋ （７０Ｈ＋１０Ｈ）番地 ＋ Ｃ

⑤ Ｘ修飾演算モードフラグＴの内容を０にします。

例えば、７０Ｈ番地の内容が５３Ｈで８０Ｈ番地の内容が２１Ｈとすると、①～⑤の命令実行後は、７０Ｈ番地の内容が７４Ｈでキャリーフラグ Ｃの内容が０となります。

（2） 減算（ニーモニック ＳＢＣ）

ＳＢＣ命令は、Ｘ修飾演算モードフラグＴが１のとき、Ｍ（Ｘ）の内容から、メモリＭの内容及び、キャリーフラグＣの内容の補数を減算して、結果をＭ（Ｘ）及びキャリーフラグＣに入れます。このとき、アキュムレータＡの内容は変化しませんが、ステータスフラグは変化します。ただし、Ｍ（Ｘ）はインデックスレジスタＸが示す番地のメモリです。

例）

```
ＳＥＣ                          ①
ＳＥＴ                          ②
ＬＤＸ      ＃ＳＢＤＡＴＡ      ③
ＳＢＣ      ＄１０，Ｘ          ④
ＣＬＴ                          ⑤
```

（説明）

① キャリーフラグＣの内容を１にします。

② Ｘ修飾演算モードフラグＴの内容を１にします。

③ ＃ＳＢＤＡＴＡ（例えば＄８０）を、インデックスレジスタＸにロードします。

④ メモリの８０Ｈ番地の内容から、９０Ｈ番地（８０Ｈ＋１０Ｈ）の内容及び、キャリーフラグＣの内容の補数を減算して、結果を８０Ｈ番地及びキャリーフラグＣに入れます。

８０Ｈ番地 ← ８０Ｈ番地 － （８０Ｈ＋１０Ｈ）番地 － Ｃ の補数

⑤ Ｘ修飾演算モードフラグＴの内容を０にします。

例えば、８０Ｈ番地の内容が５３Ｈで９０Ｈ番地の内容が２１Ｈとすると、①～⑤の命令実行後は、８０Ｈ番地の内容が３２Ｈでキャリーフラグ Ｃの内容が１となります。

2.2 データのビット判定分岐

（1） セット時分岐（ニーモニック ＢＢＳ）

ＢＢＳ命令は、アキュムレータＡ又はメモリＭの指定されたビットｉをテストします。そのビットが１であれば、指定されたアドレスに分岐します。分岐先のアドレスは、相対で示します。そのビットが０であれば、そのまま、次へ進みます。

例）

```
ＢＢＳ　　０，Ｐ０，ＳＴＡＲＴ
```

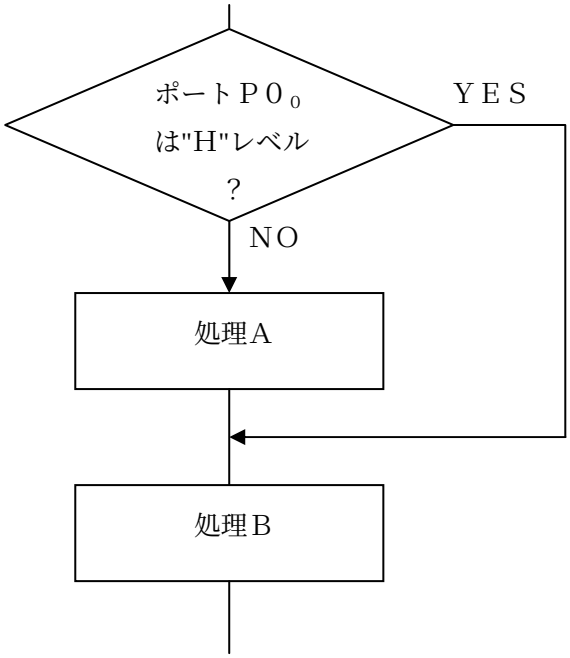
処理Ａ

ＳＴＡＲＴ：

処理Ｂ

（説明）

ポートＰ０のビット０が１であれば、ＳＴＡＲＴへ分岐します。ポートＰ０のビット０が０であれば、そのまま、次へ進みます。



（2）　クリア時分岐（ニーモニック　ＢＢＣ）

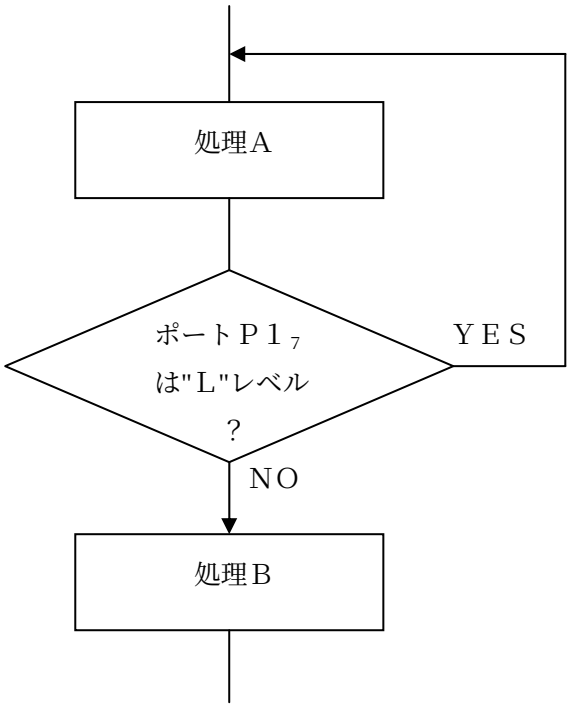
ＢＢＣ命令は、アキュムレータＡ又はメモリＭの指定されたビットｉをテストします。そのビットが０であれば、指定されたアドレスに分岐します。分岐先のアドレスは、相対で示します。そのビットが１であれば、そのまま、次へ進みます。

例）

ＲＥＳＥＴ：

（説明）

ポートＰ１のビット７が０であれば、ＲＥＳＥＴへ分岐します。ポートＰ１のビット７が１であれば、そのまま、次へ進みます。



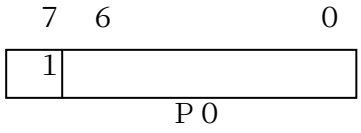
2.3 データのビット処理（セット／リセット）

(1) ビットセット（ニーモニック ＳＥＢ）
ＳＥＢ命令は、アキュムレータＡ又はメモリMの、指定されたビットｉの内容を１にします。

例）

```
ＳＥＢ　　７，Ｐ０
```

（説明）
ポートＰ０のビット７の内容を１にします。



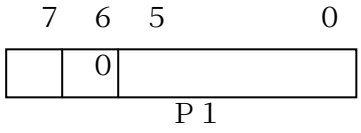
例えば、ポートＰ０の内容が５３Ｈとすると、命令実行後は、ポートＰ０の内容がＤ３Ｈとなります。

(2) ビットクリア（ニーモニック ＣＬＢ）
ＣＬＢ命令は、アキュムレータＡ又はメモリMの、指定されたビットｉの内容を０にします。

例）

```
ＣＬＢ　　６，Ｐ１
```

（説明）
ポートＰ１のビット６の内容を０にします。



例えば、ポートＰ１の内容が５３Ｈとすると、命令実行後は、ポートＰ１の内容が１３Ｈとなります。

2.4 データの回転シフト

（1） 左回転（ニーモニック ＲＯＬ）

ＲＯＬ命令は、アキュムレータＡ又はメモリＭをキャリーフラグＣとつなげて、その内容を左へ１ビット回転します。アキュムレータＡ又はメモリＭのビット０には、キャリーフラグＣの内容が入り、キャリーフラグＣには、アキュムレータＡ又はメモリＭのビット７の内容が入ります。

例）

```
ＲＯＬ　　Ｐ０
```

（説明）

ポートＰ０をキャリーフラグＣとつなげて、その内容を左へ１ビット回転します。

例えば、ポートＰ０の内容が５３Ｈでキャリーフラグの内容が１とすると、命令実行後は、ポートＰ０の内容がＡ７Ｈでキャリーフラグの内容が０となります。

（2） 左シフト（ニーモニック ＡＳＬ）

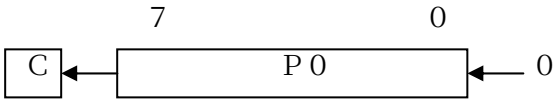
ＡＳＬ命令は、アキュムレータＡ又はメモリＭのすべてのビットを、１ビット左へシフトします。このとき、アキュムレータＡ又はメモリＭのビット０は０になります。また、キャリーフラグＣには、アキュムレータＡ又はメモリＭのビット７の内容が入ります。

例）

```
ＡＳＬ　　Ｐ０
```

（説明）

ポートＰ０のすべてのビットを、１ビット左へシフトします。



例えば、ポートＰ０の内容が５３Ｈでキャリーフラグの内容が１とすると、命令実行後は、ポートＰ０の内容がＡ６Ｈでキャリーフラグの内容が０となります。

（3）　右回転（ニーモニック　ＲＯＲ）

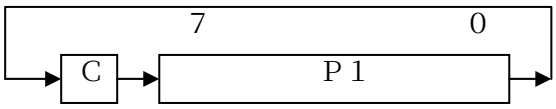
ＲＯＲ命令は、アキュムレータＡ又はメモリＭをキャリーフラグＣとつなげて、その内容を右へ１ビット回転します。アキュムレータＡ又はメモリＭのビット７には、キャリーフラグＣの内容が入り、キャリーフラグＣには、アキュムレータＡ又はメモリＭのビット０の内容が入ります。

例）

```
ＲＯＲ　　Ｐ１
```

（説明）

ポートＰ１をキャリーフラグＣとつなげて、その内容を右へ１ビット回転します。



例えば、ポートＰ１の内容が５３Ｈでキャリーフラグの内容が１とすると、命令実行後は、ポートＰ１の内容がＡ９Ｈでキャリーフラグの内容が１となります。

（4）　右シフト（ニーモニック　ＬＳＲ）

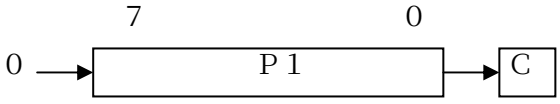
ＬＳＲ命令は、アキュムレータＡ又はメモリＭのすべてのビットを、１ビット右へシフトします。このとき、アキュムレータＡ又はメモリＭのビット７は０になります。また、キャリーフラグＣには、アキュムレータＡ又はメモリＭのビット０の内容が入ります。

例）

```
ＬＳＲ　　Ｐ１
```

（説明）

ポートＰ１のすべてのビットを、１ビット右へシフトします。



例えば、ポートＰ１の内容が５３Ｈでキャリーフラグの内容が１とすると、命令実行後は、ポートＰ１の内容が２９Ｈでキャリーフラグの内容が１となります。

（5）　上位と下位の交換（ニーモニック　ＲＲＦ）

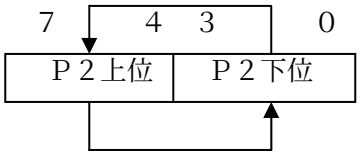
ＲＲＦ命令は、メモリMの内容を右へ４ビット回転します。その結果、メモリMの内容の上位４ビットと下位４ビットを交換したことになります。

例）

ＲＲＦ　　Ｐ２

（説明）

ポートＰ２の内容の上位４ビットと下位４ビットを交換します。



例えば、ポートＰ２の内容が５３Ｈとすると、命令実行後は、ポートＰ２の内容が３５Ｈとなります。

3. 基本処理プログラム例

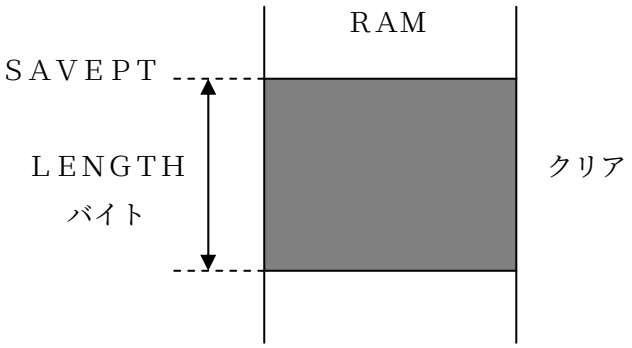
3.1 ＲＡＭクリア

(1) 概要

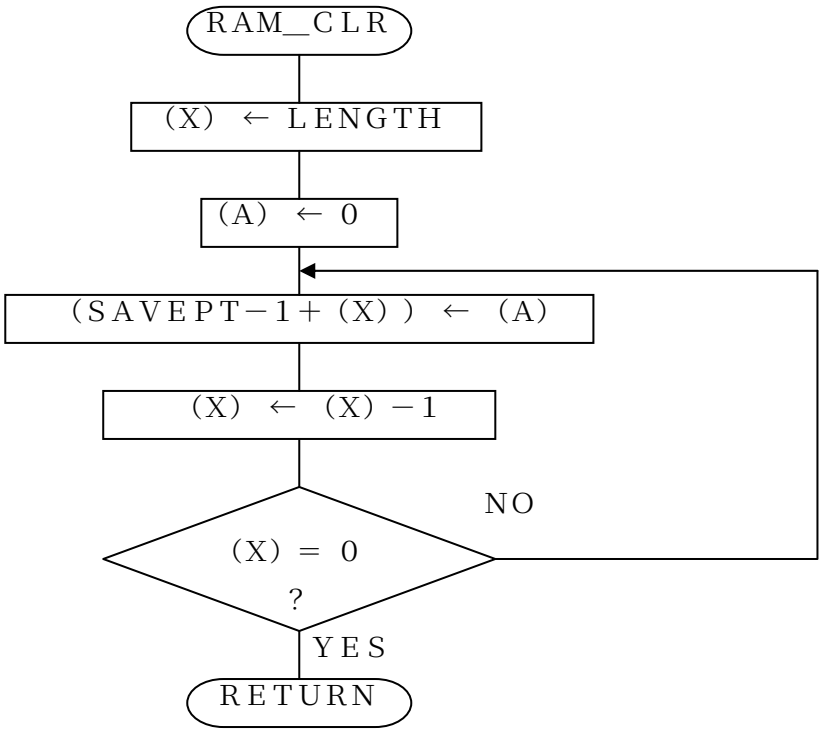
領域を指定して、ＲＡＭをクリアします。

(2) 説明

ＲＡＭアドレスＳＡＶＥＰＴより、ＬＥＮＧＴＨバイト分のＲＡＭをクリアします。



(3)　フローチャート



(4)　プログラムリスト

```
;*****************************************************
;
;       RAM clear routine
;
;*****************************************************
;
RAM_CLR:
        LDX   #LENGTH        ;RAM length
        LDA   #$00
RAMCL1: STA   SAVEPT-1,X     ;Clear from SAVEPT
        DEX                  ;  - to SAVEPT+LENGTH-1
        BNE   RAMCL1         ;Clear end ?
        RTS                  ;Yes
```

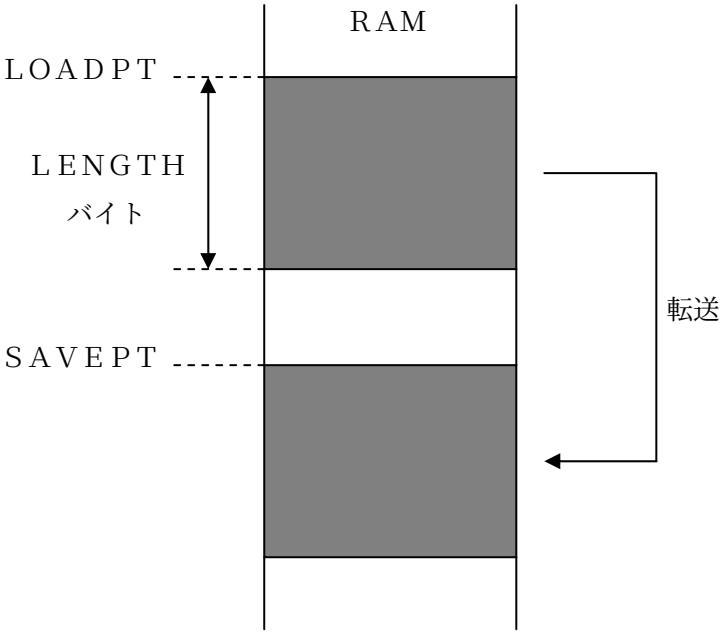
3.2 データ転送（ＲＡＭ）

（1） 概要

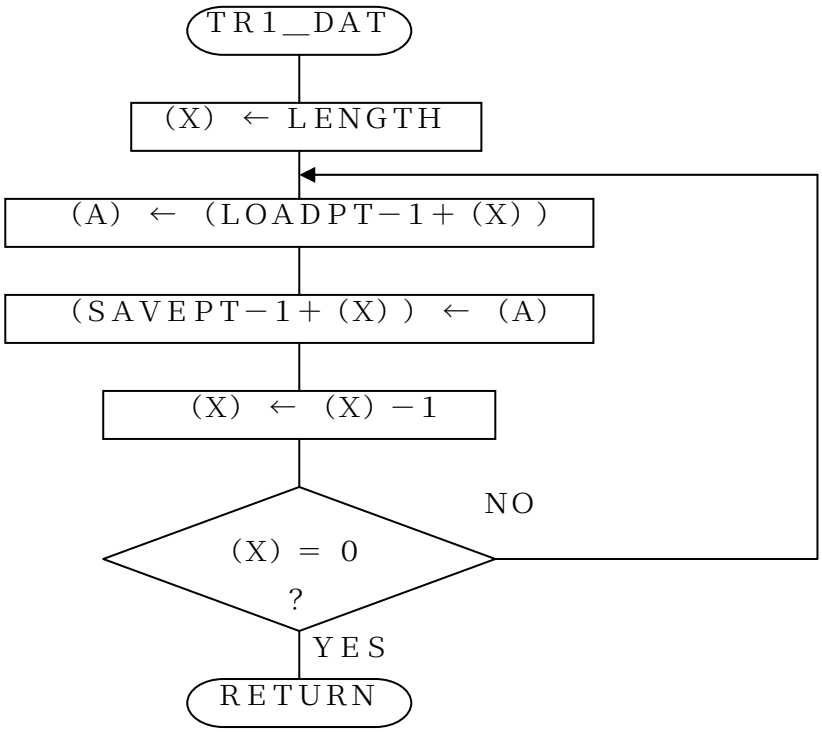
ＲＡＭ領域内でデータを転送します。

（2） 説明

ＲＡＭアドレスＬＯＡＤＰＴより、ＬＥＮＧＴＨバイト分のデータを、ＳＡＶＥＰＴから続くアドレスへ転送します。



(3)　フローチャート



(4)　プログラムリスト

```
;******************************************************
;
;       RAM data transfer routine
;
;******************************************************
;
TR1_DAT:
        LDX    #LENGTH         ;RAM length
TR1_01: LDA    LOADPT-1,X      ;Transfer data from LOADPT
        STA    SAVEPT-1,X      ;  - to SAVEPT
        DEX
        BNE    TR1_01          ;Transfer end ?
        RTS                    ;Yes
```

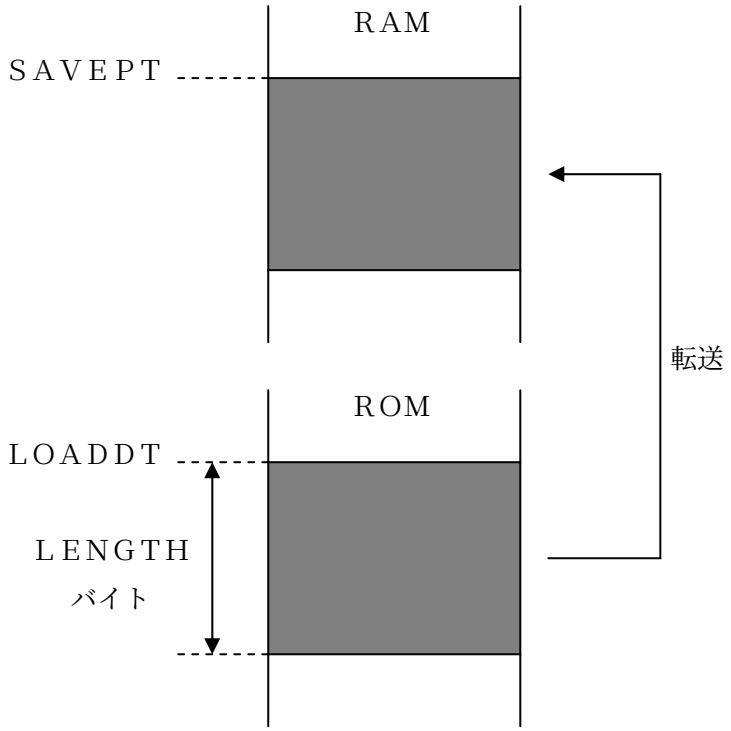
3.3 データ転送（ＲＯＭアドレス固定）

（1） 概要

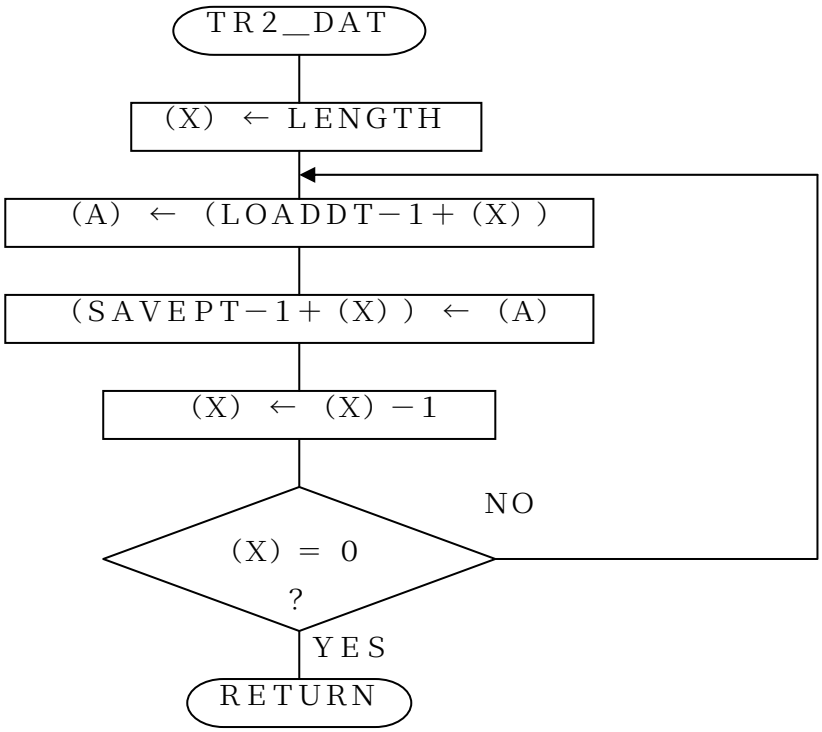
ＲＯＭ領域内のデータ(アドレス固定)を転送します。

（2） 説明

ＲＯＭアドレスＬＯＡＤＤＴより、ＬＥＮＧＴＨバイト分のデータを、ＳＡＶＥＰＴから続くＲＡＭアドレスへ転送します。



(3)　フローチャート



(4)　プログラムリスト

```
;*****************************************************
;
;       ROM data transfer routine ( address fixed )
;
;*****************************************************
;
TR2_DAT:
        LDX   #LENGTH         ;ROM length
TR2_01: LDA   LOADDT-1,X      ;Transfer data from LOADDT
        STA   SAVEPT-1,X      ;  - to SAVEPT
        DEX
        BNE   TR2_01          ;Transfer end ?
        RTS                   ;Yes
```

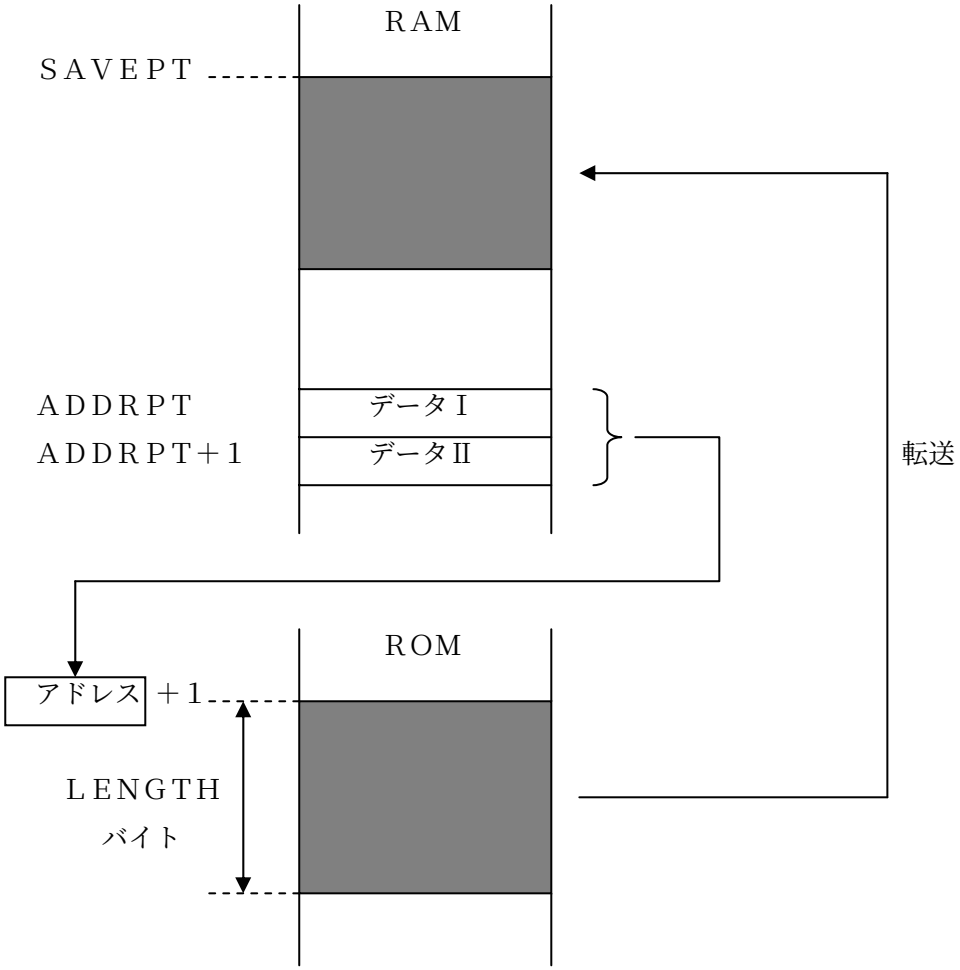
3.4 データ転送（ＲＯＭアドレス可変）

（1） 概要

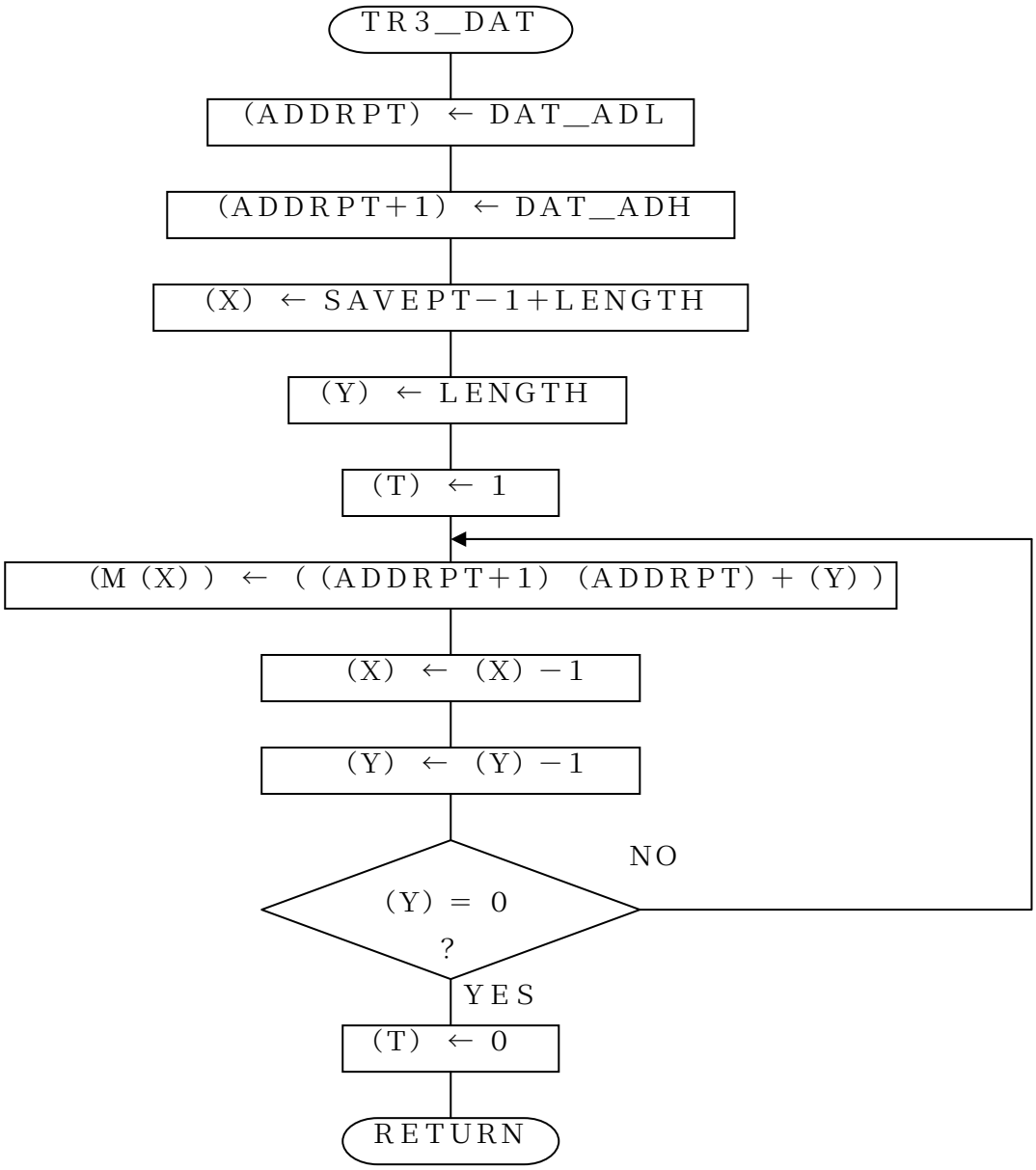
ＲＯＭ領域内のデータ(アドレス可変)を転送します。

（2） 説明

ＲＡＭアドレスＡＤＤＲＰＴ＋１、ＡＤＤＲＰＴの内容が示すＲＯＭアドレス＋１より、ＬＥＮＧＴＨバイト分のデータを、ＳＡＶＥＰＴから続くＲＡＭアドレスへ転送します。



（3）　フローチャート



注）M（X）はインデックスレジスタXが示す番地のメモリです。

(4)　プログラムリスト

```
;*****************************************************
;
;       ROM data transfer routine ( address float )
;
;*****************************************************
;
TR3_DAT:
        LDM    #DAT_ADL,ADDRPT
        LDM    #DAT_ADH,ADDRPT+1
;-----------------------------------------------
        LDX    #SAVEPT-1+LENGTH
        LDY    #LENGTH         ;ROM length
        SET                    ;Transfer data from
TR3_01: LDA    (ADDRPT),Y      ;  - (ADDRPT+1)(ADDRPT)+1
        DEX                    ;  - to SAVEPT
        DEY                    ;
        BNE    TR3_01          ;Transfer end ?
        CLT                    ;Yes
        RTS
```

3.5 データの並べ替え（ソート）

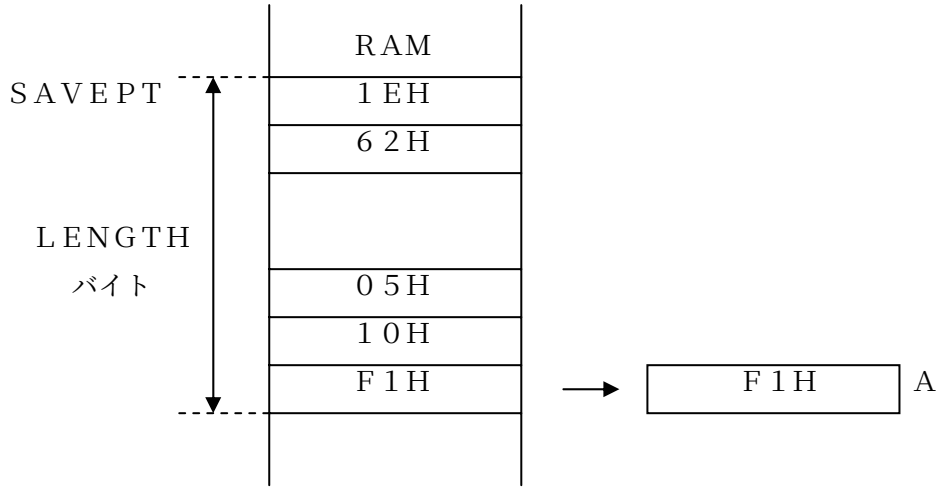
（1） 概要

ＲＡＭ領域内のデータを降順に並べ替えます。

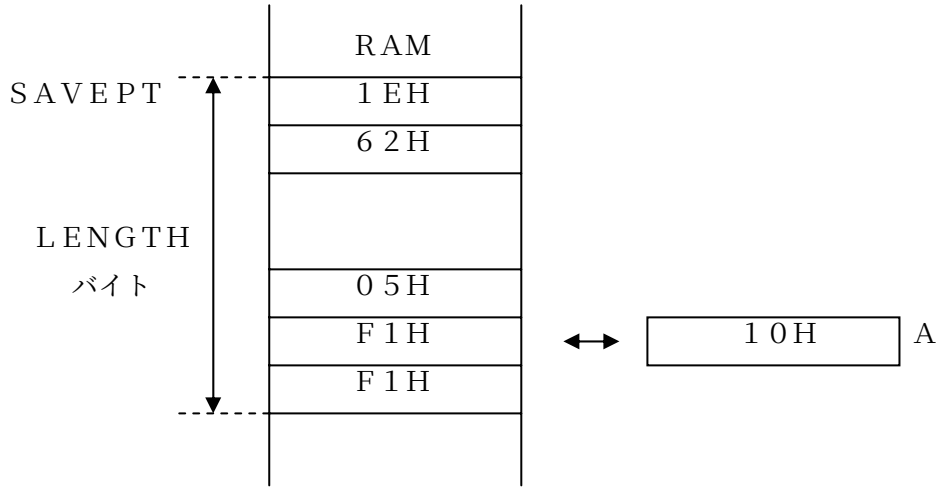
（2） 説明

ＲＡＭアドレスＳＡＶＥＰＴより、ＬＥＮＧＴＨバイト分のデータを、降順に並べ替えます。

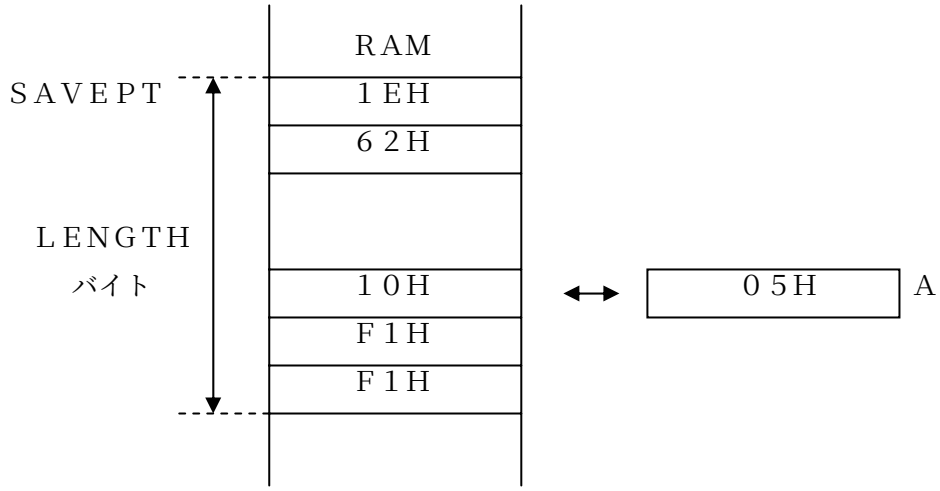
①まず、並べ替えを行なう領域の最上位アドレスのメモリ内容をアキュムレータＡに入れます。



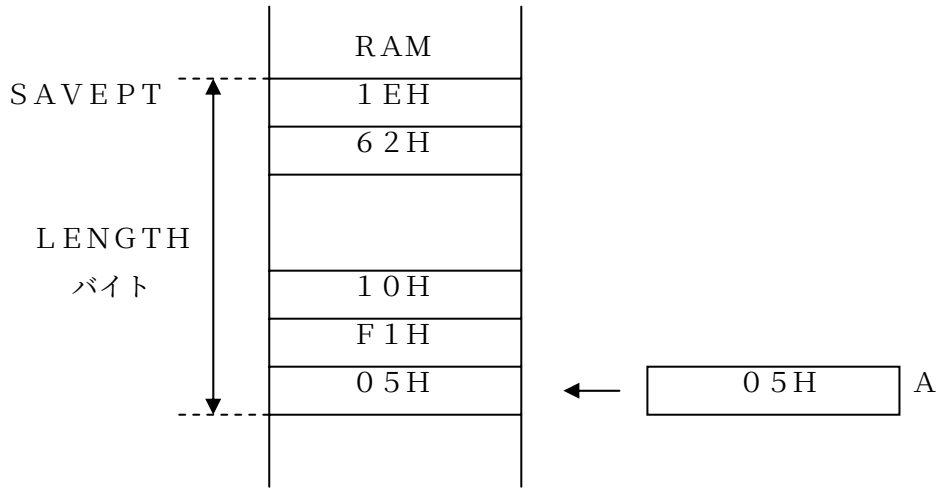
②次に、１つ下位アドレスのメモリの内容とアキュムレータＡの内容を比較します。この時、アキュムレータＡの内容のほうが大きいか等しい場合には、メモリとアキュムレータＡの内容を入れ替えます。



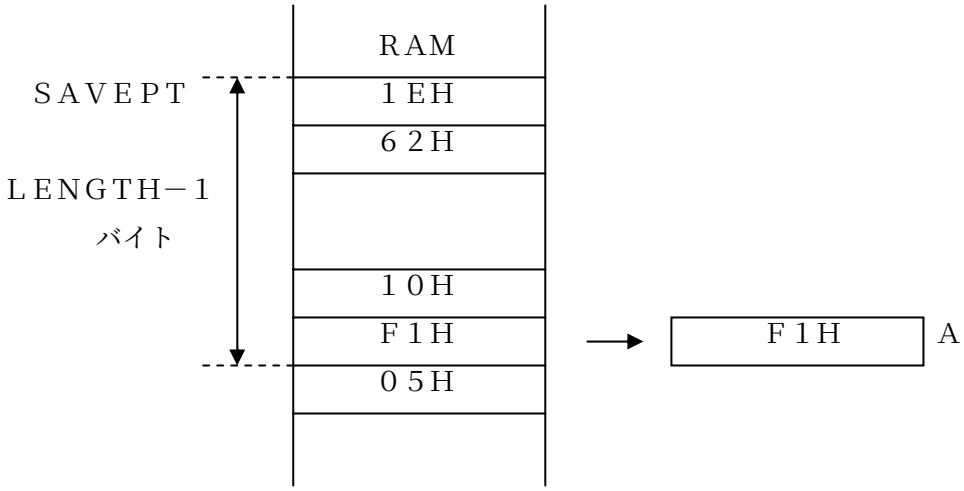
③さらに１つ下位アドレスのメモリの内容とアキュムレータＡの内容を比較します。この時も、アキュムレータＡの内容のほうが大きいか等しい場合には、メモリとアキュムレータＡの内容を入れ替えます。



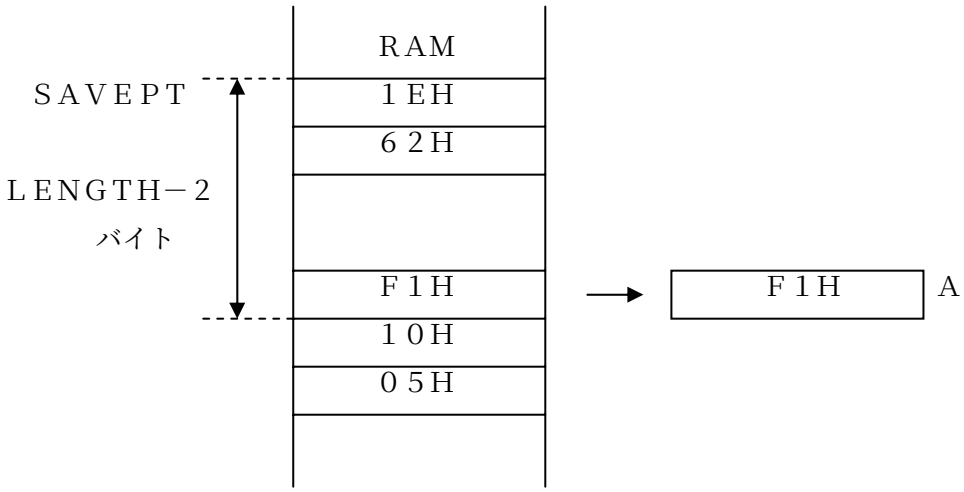
④このように、並べ替えを行なう領域の最下位アドレスまでこれを繰り返すと、アキュムレータＡには、最小値が入ります。このアキュムレータＡの内容を最上位アドレスのメモリに入れます。



⑤最上位アドレスをデクリメントした領域内（ＳＡＶＥＰＴよりＬＥＮＧＴＨ－１バイト分）で、①～④を実行します。



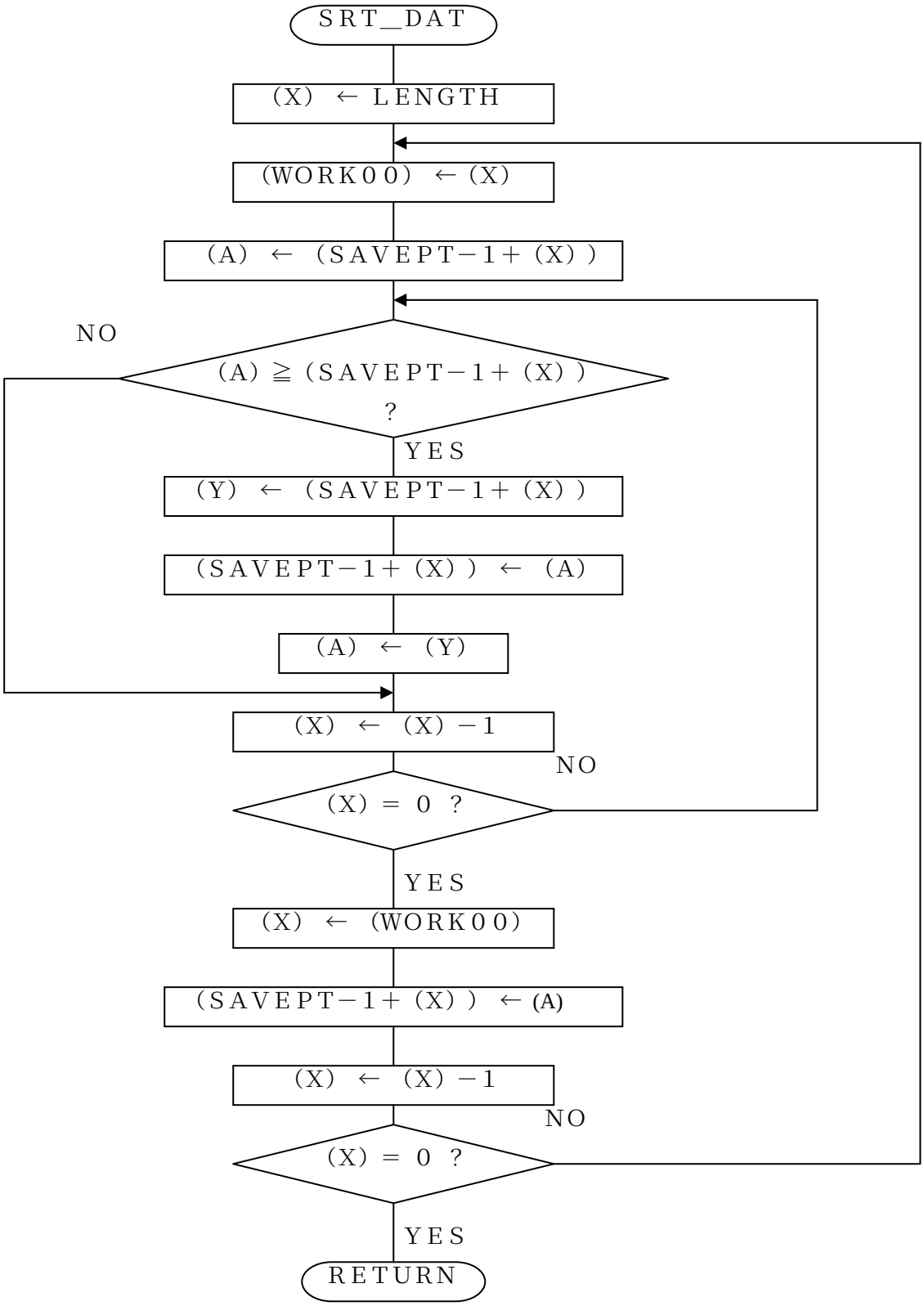
⑥さらに、最上位アドレスをデクリメントした領域内（ＳＡＶＥＰＴよりＬＥＮＧＴＨ－２バイト分）で、①～④を実行します。



⑦このように、最上位アドレスをデクリメントしていき、最上位アドレスが最下位アドレスと等しくなるまで①～④を実行することで、降順に並べ替えられます。

なお、②③のメモリとアキュムレータＡの内容の入れ替えを、アキュムレータＡの内容のほうが小さい場合に行なえば、昇順に並べ替えることができます。

(3)　フローチャート


SRT_DAT
(X) ← LENGTH
(WORK00) ← (X)
(A) ← (SAVEPT−1＋(X))
NO
(A) ≧ (SAVEPT−1＋(X)) ?
YES
(Y) ← (SAVEPT−1＋(X))
(SAVEPT−1＋(X)) ← (A)
(A) ← (Y)
(X) ← (X) −1
NO
(X) = 0 ?
YES
(X) ← (WORK00)
(SAVEPT−1＋(X)) ← (A)
(X) ← (X) −1
NO
(X) = 0 ?
YES
RETURN

(4)　プログラムリスト

```
;******************************************************
;
;       Sort routine
;
;******************************************************
;
SRT_DAT:
        LDX   #LENGTH        ;Data length
SRT_01: STX   WORK00
        LDA   SAVEPT-1,X     ;SAVEPT <--> SAVEPT-1+WORK00
SRT_02: CMP   SAVEPT-1,X     ;If use (BCS)
        BCC   SRT_03         ;  - then negative
        LDY   SAVEPT-1,X
        STA   SAVEPT-1,X
        TYA                  ;Minimum data set
SRT_03: DEX                  ;  - to A & Y
        BNE   SRT_02
        LDX   WORK00
        STA   SAVEPT-1,X     ;Minimum data set
        DEX                  ;Next area
        BNE   SRT_01         ;Sort end ?
        RTS                  ;Yes
```

3.6　１６ビットデータ加算（バイナリ）

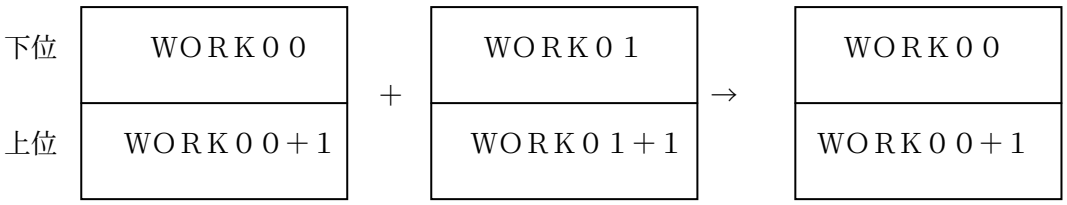
（1）　概要

１６ビットバイナリデータを加算します。

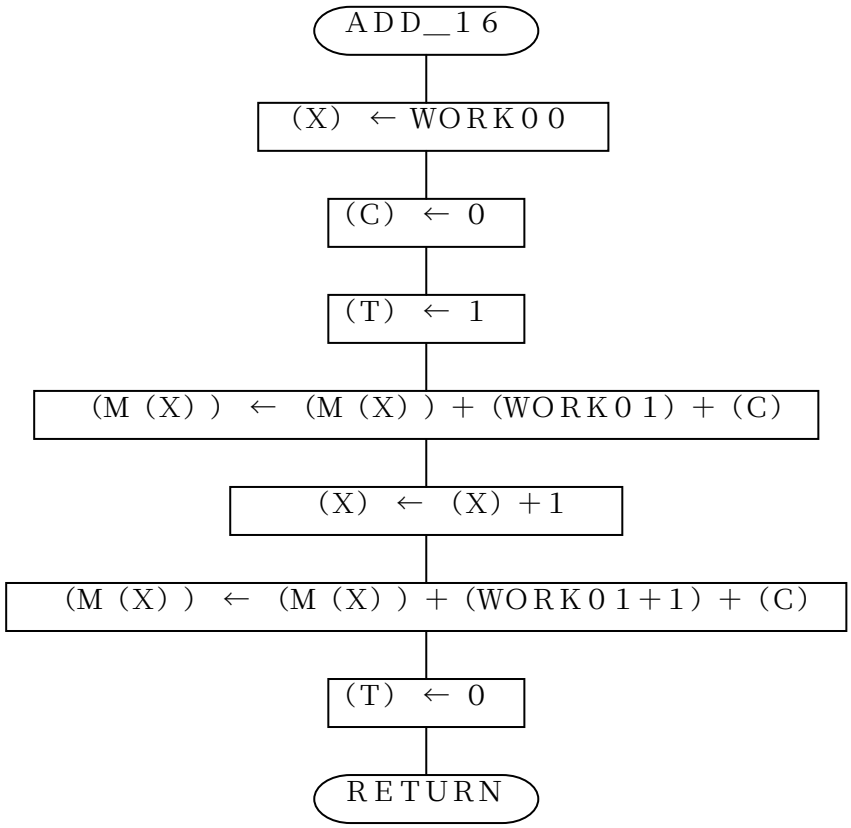
（2）　説明

ＷＯＲＫ００＋１、ＷＯＲＫ００の内容とＷＯＲＫ０１＋１、ＷＯＲＫ０１の内容を加算し、結果をＷＯＲＫ００＋１、ＷＯＲＫ００に入れます。

Ｘ修飾演算モードフラグＴを１にして、アキュムレータＡの内容を破壊せずに演算を行ないます。



(3)　フローチャート



注）M（X）はインデックスレジスタXが示す番地のメモリです。

(4)　プログラムリスト

```
;******************************************************
;
;       16 bits BIN. data addition routine
;
;******************************************************
;
ADD_16:
        LDX    #WORK00
        CLC                    ;C flag clear
        SET                    ;T flag set
        ADC    WORK01
        INX                    ;(WORK00+1)(WORK00) +
        ADC    WORK01+1        ;  (WORK01+1)(WORK01)
        CLT                    ;T flag clear
        RTS
```

3.7　１６ビットデータ減算（バイナリ）

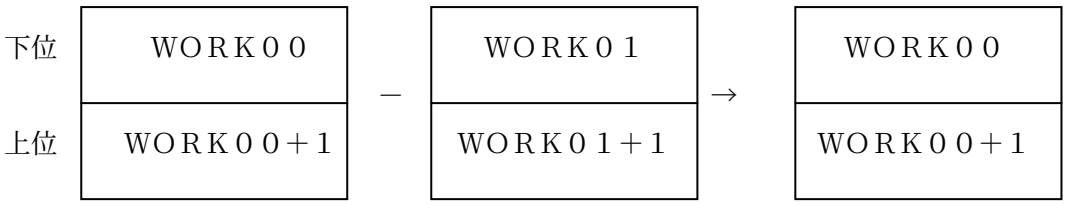
（1）　概要

１６ビットバイナリデータを減算します。

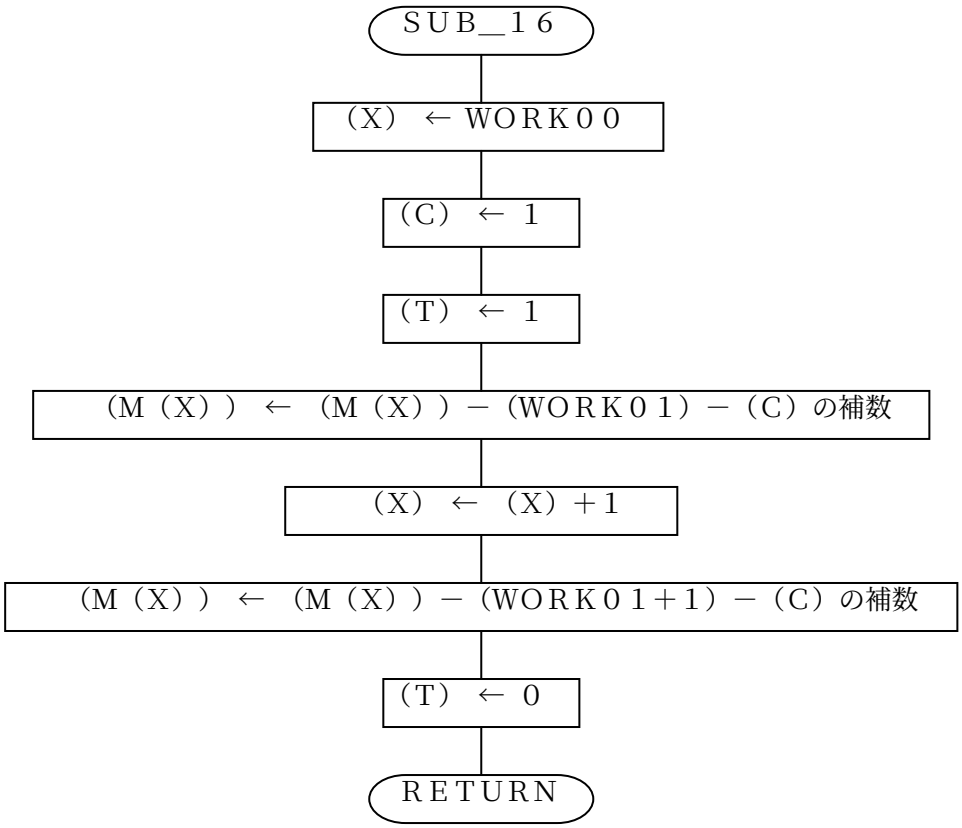
（2）　説明

ＷＯＲＫ００＋１、ＷＯＲＫ００の内容からＷＯＲＫ０１＋１、ＷＯＲＫ０１の内容を減算し、結果をＷＯＲＫ００＋１、ＷＯＲＫ００に入れます。

Ｘ修飾演算モードフラグＴを１にして、アキュムレータＡの内容を破壊せずに演算を行ないます。



(3)　フローチャート



注）M（X）はインデックスレジスタXが示す番地のメモリです。

(4)　プログラムリスト

```
;******************************************************
;
;       16 bits BIN. data subtraction routine
;
;******************************************************
;
SUB_16:
        LDX    #WORK00
        SEC                    ;C flag set
        SET                    ;T flag set
        SBC    WORK01
        INX                    ;(WORK00+1)(WORK00) -
        SBC    WORK01+1        ;  (WORK01+1)(WORK01)
        CLT                    ;T flag clear
        RTS
```

3.8　１６ビットデータ加算（ＢＣＤ）

（1）　概要

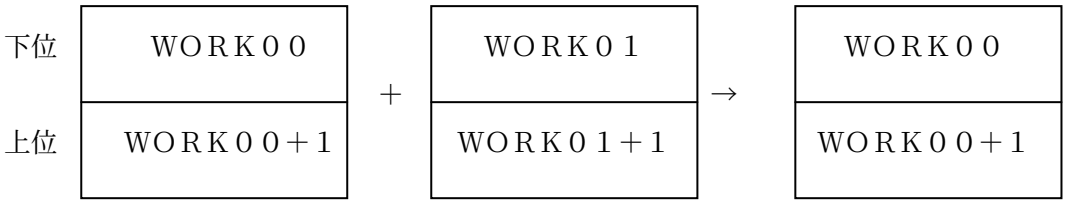
１６ビットＢＣＤデータを加算します。

（2）　説明

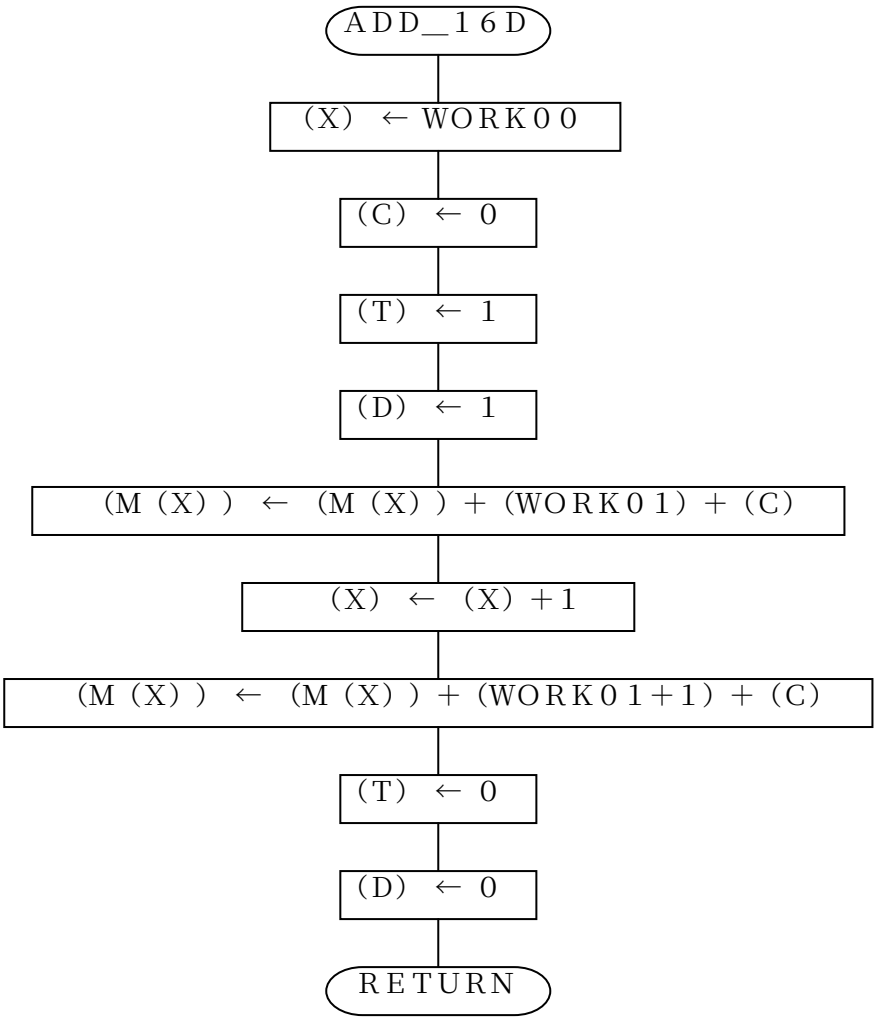
ＷＯＲＫ００＋１、ＷＯＲＫ００の内容とＷＯＲＫ０１＋１、ＷＯＲＫ０１の内容を加算し、結果をＷＯＲＫ００＋１、ＷＯＲＫ００に入れます。

１０進モードフラグＤを１にすることで、ＡＤＣ命令が１０進加算することを利用します。ただし、この場合、キャリーフラグＣが遅れて確定しますので、ＡＤＣ命令の次で、ＳＥＣ，ＣＬＣ，ＣＬＤの命令を実行しないようにしてください。

Ｘ修飾演算モードフラグＴを１にして、アキュムレータＡの内容を破壊せずに演算を行ないます。



(3)　フローチャート



注）M（X）はインデックスレジスタXが示す番地のメモリです。

(4)　プログラムリスト

```
;******************************************************
;
;       16 bits BCD data addition routine
;
;******************************************************
;
ADD_16D:
        LDX    #WORK00
        CLC                    ;C flag reset
        SET                    ;T flag set
        SED                    ;Decimal mode set
        ADC    WORK01
        INX                    ;(WORK00+1)(WORK00) +
        ADC    WORK01+1        ;  (WORK01+1)(WORK01)
        CLT                    ;T flag reset
        CLD                    ;Decimal mode clear
        RTS
```

3.9　１６ビットデータ減算（ＢＣＤ）

（1）　概要

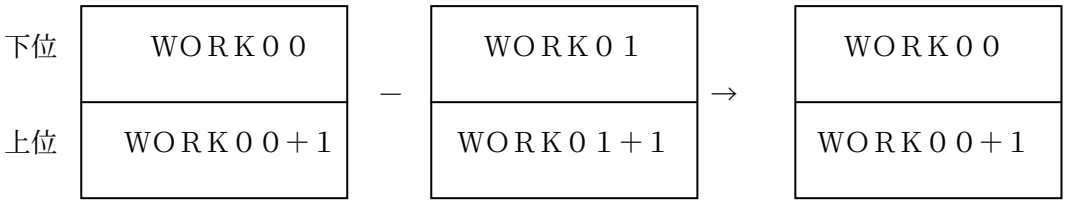
１６ビットＢＣＤデータを減算します。

（2）　説明

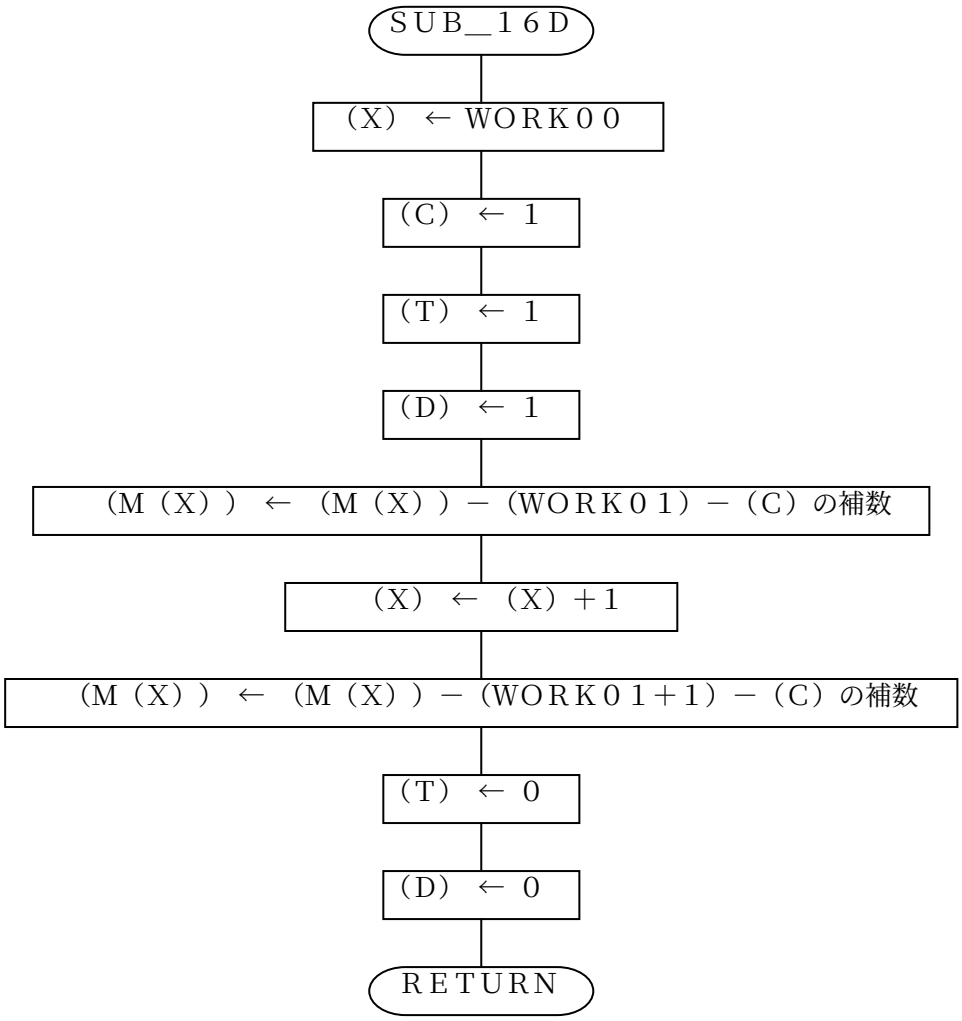
ＷＯＲＫ００＋１、ＷＯＲＫ００の内容からＷＯＲＫ０１＋１、ＷＯＲＫ０１の内容を減算し、結果をＷＯＲＫ００＋１、ＷＯＲＫ００に入れます。

１０進モードフラグＤを１にすることで、ＳＢＣ命令が１０進減算することを利用します。ただし、この場合、キャリーフラグＣが遅れて確定しますので、ＳＢＣ命令の次で、ＳＥＣ，ＣＬＣ，ＣＬＤの命令を実行しないようにしてください。

Ｘ修飾演算モードフラグＴを１にして、アキュムレータＡの内容を破壊せずに演算を行ないます。



(3)　フローチャート



注）M（X）はインデックスレジスタXが示す番地のメモリです。

(4)　プログラムリスト

```
;******************************************************
;
;       16 bits BCD data subtraction routine
;
;******************************************************
;
SUB_16D:
        LDX    #WORK00
        SEC                    ;C flag set
        SET                    ;T flag set
        SED                    ;Decimal mode set
        SBC    WORK01
        INX                    ;(WORK00+1)(WORK00) -
        SBC    WORK01+1        ;  (WORK01+1)(WORK01)
        CLT                    ;T flag reset
        CLD                    ;Decimal mode reset
        RTS
```

3.10　１６ビットデータ乗算（バイナリ）

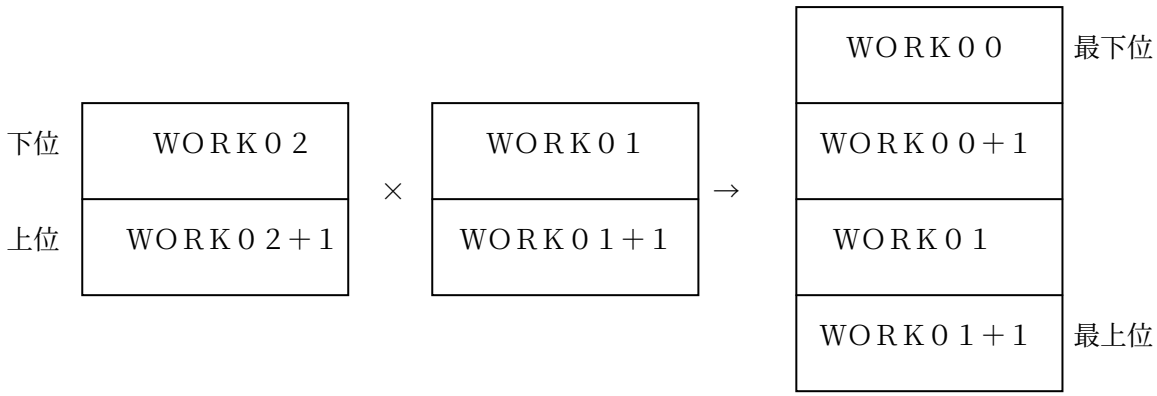
（1）　概要

１６ビットバイナリデータを乗算します。

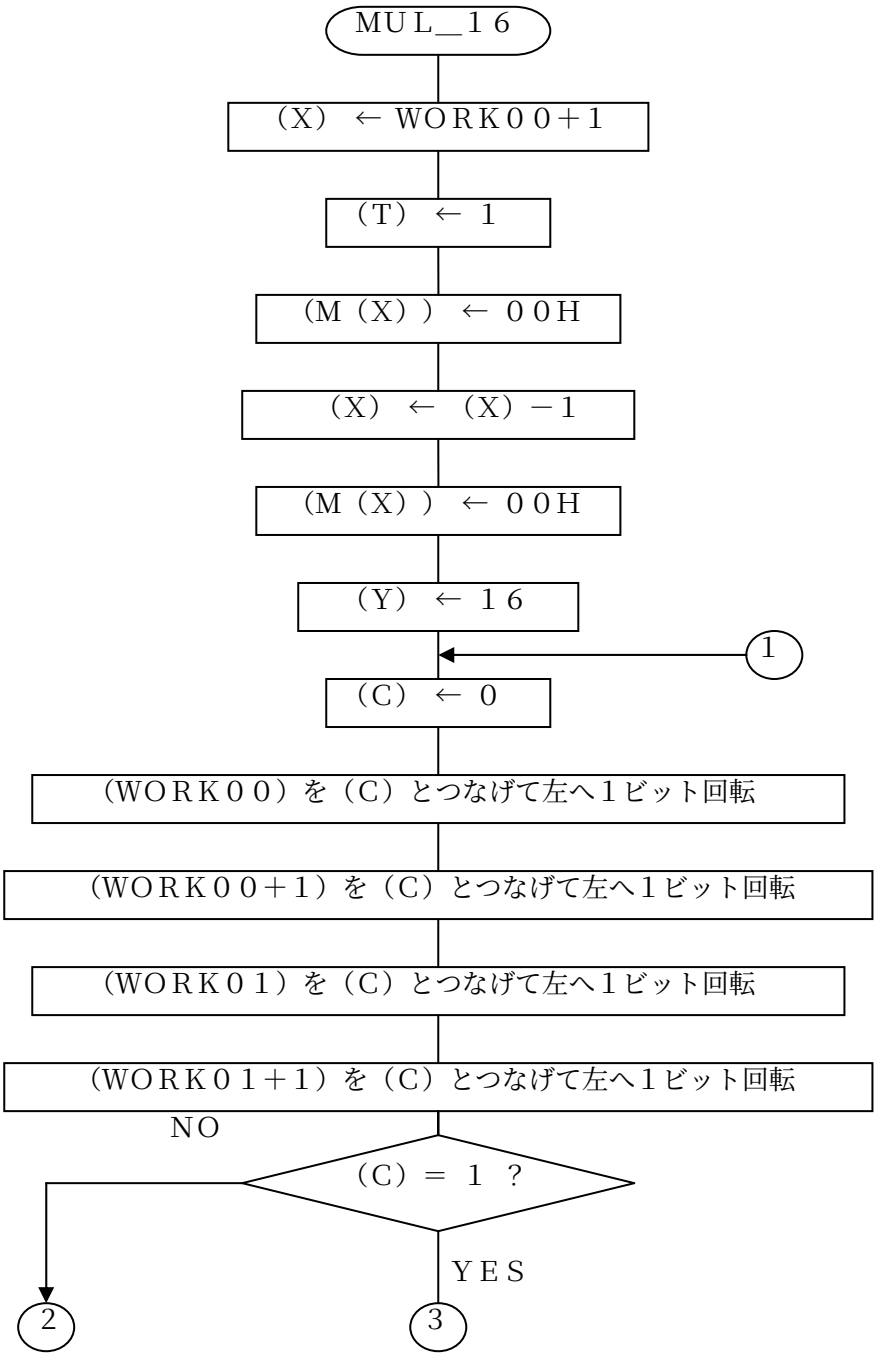
（2）　説明

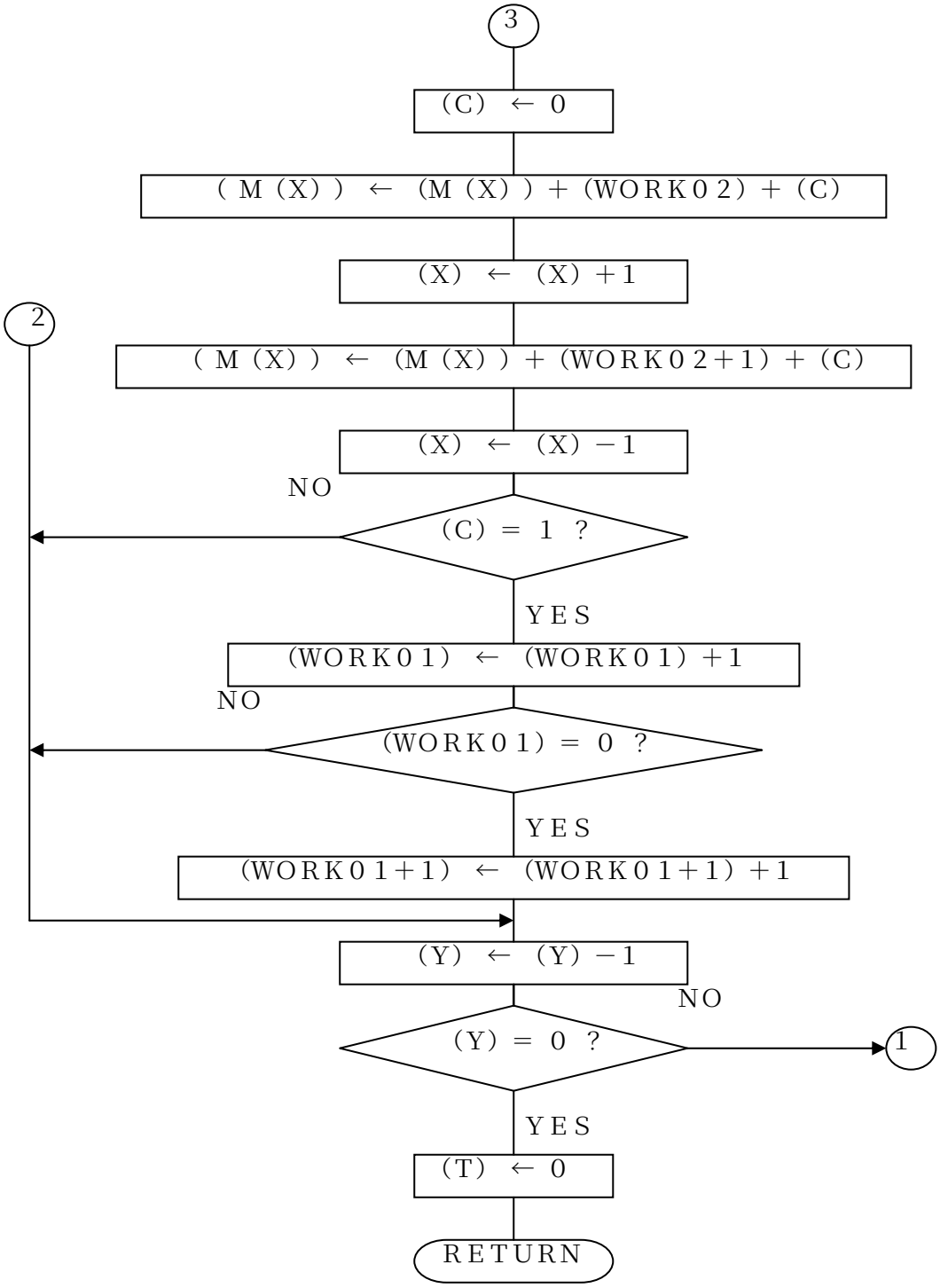
ＷＯＲＫ０２＋１、ＷＯＲＫ０２の内容とＷＯＲＫ０１＋１、ＷＯＲＫ０１の内容を乗算し、結果をＷＯＲＫ０１＋１、ＷＯＲＫ０１、ＷＯＲＫ００＋１、ＷＯＲＫ００に入れます。

Ｘ修飾演算モードフラグＴを１にして、アキュムレータＡの内容を破壊せずに演算を行ないます。



(3)　フローチャート


MUL_16
(X) ← WORK00+1
(T) ← 1
(M(X)) ← 00H
(X) ← (X)−1
(M(X)) ← 00H
(Y) ← 16
1
(C) ← 0
(WORK00)を(C)とつなげて左へ1ビット回転
(WORK00+1)を(C)とつなげて左へ1ビット回転
(WORK01)を(C)とつなげて左へ1ビット回転
(WORK01+1)を(C)とつなげて左へ1ビット回転
NO
(C) = 1 ?
YES
2
3



注）M（X）はインデックスレジスタXが示す番地のメモリです。

(4)　プログラムリスト

```
;*****************************************************
;
;       16 bits BIN. data multiplication routine
;
;*****************************************************
;
MUL_16:
        LDX   #WORK00+1      ;Product L addr. set
        SET                  ;T flag set
        LDA   #$00           ;Clear product L
        DEX
        LDA   #$00
        LDY   #16            ;Bit counter set
;-----------------------------------------------
MUL_01: CLC
        ROL   WORK00         ;Rotate product L
        ROL   WORK00+1
        ROL   WORK01         ;Rotate product H
        ROL   WORK01+1
        BCC   MUL_02         ;C flag 1 ?
        CLC                  ;Yes
        ADC   WORK02         ;Multiplicand + product L
        INX
        ADC   WORK02+1
        DEX
        BCC   MUL_02         ;Over flow ?
        INC   WORK01         ;Yes
        BNE   MUL_02         ;Over flow ?
        INC   WORK01+1       ;Yes
;-----------------------------------------------
MUL_02: DEY
        BNE   MUL_01         ;Multiple end ?
        CLT                  ;Yes
        RTS
```

3.11　１６ビットデータ除算（バイナリ）

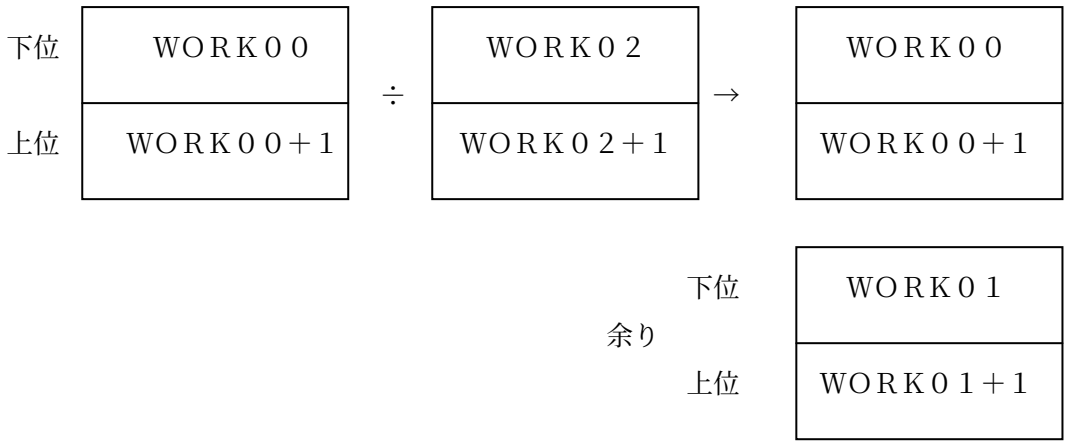
（1）　概要

１６ビットバイナリデータを除算します。

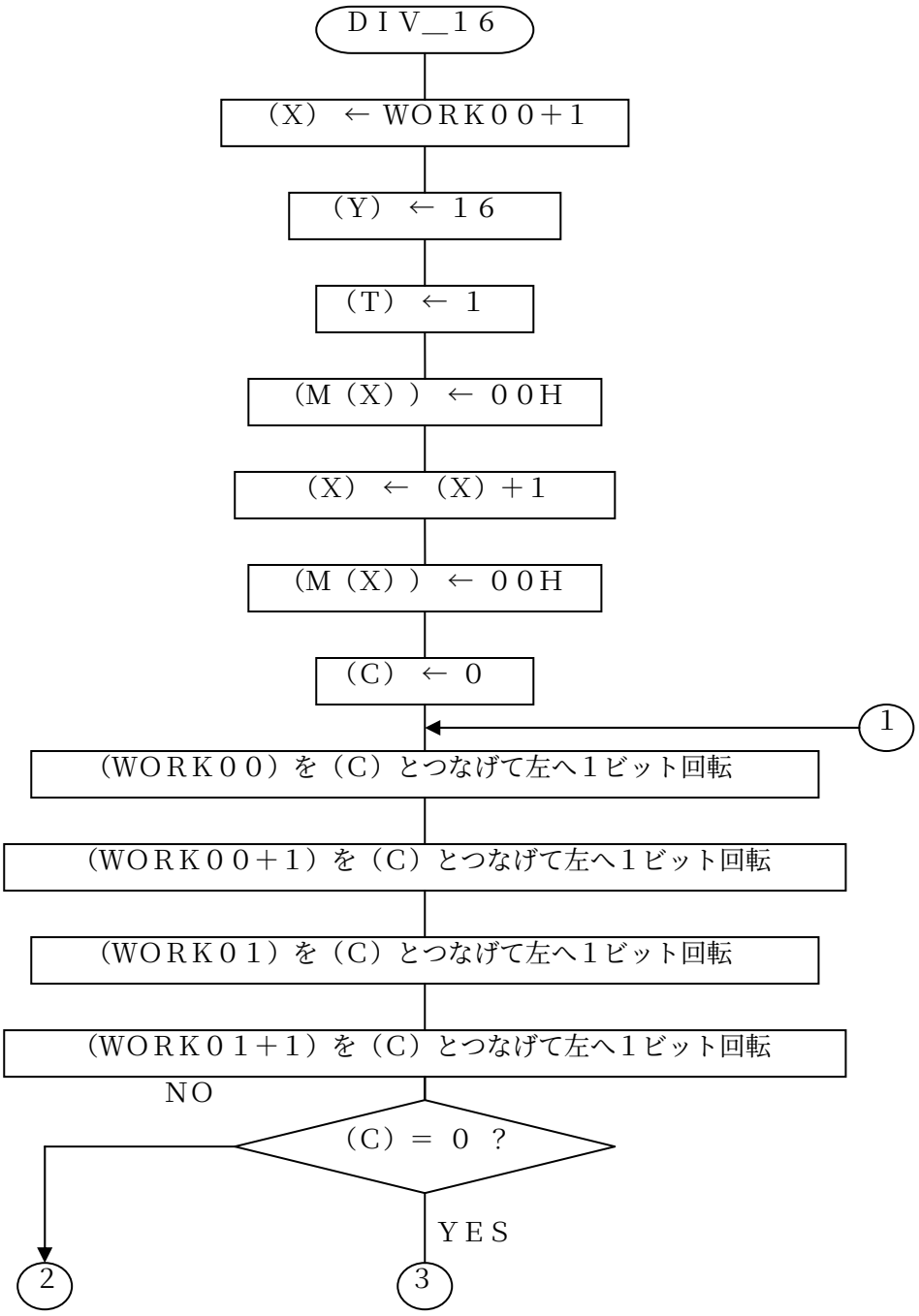
（2）　説明

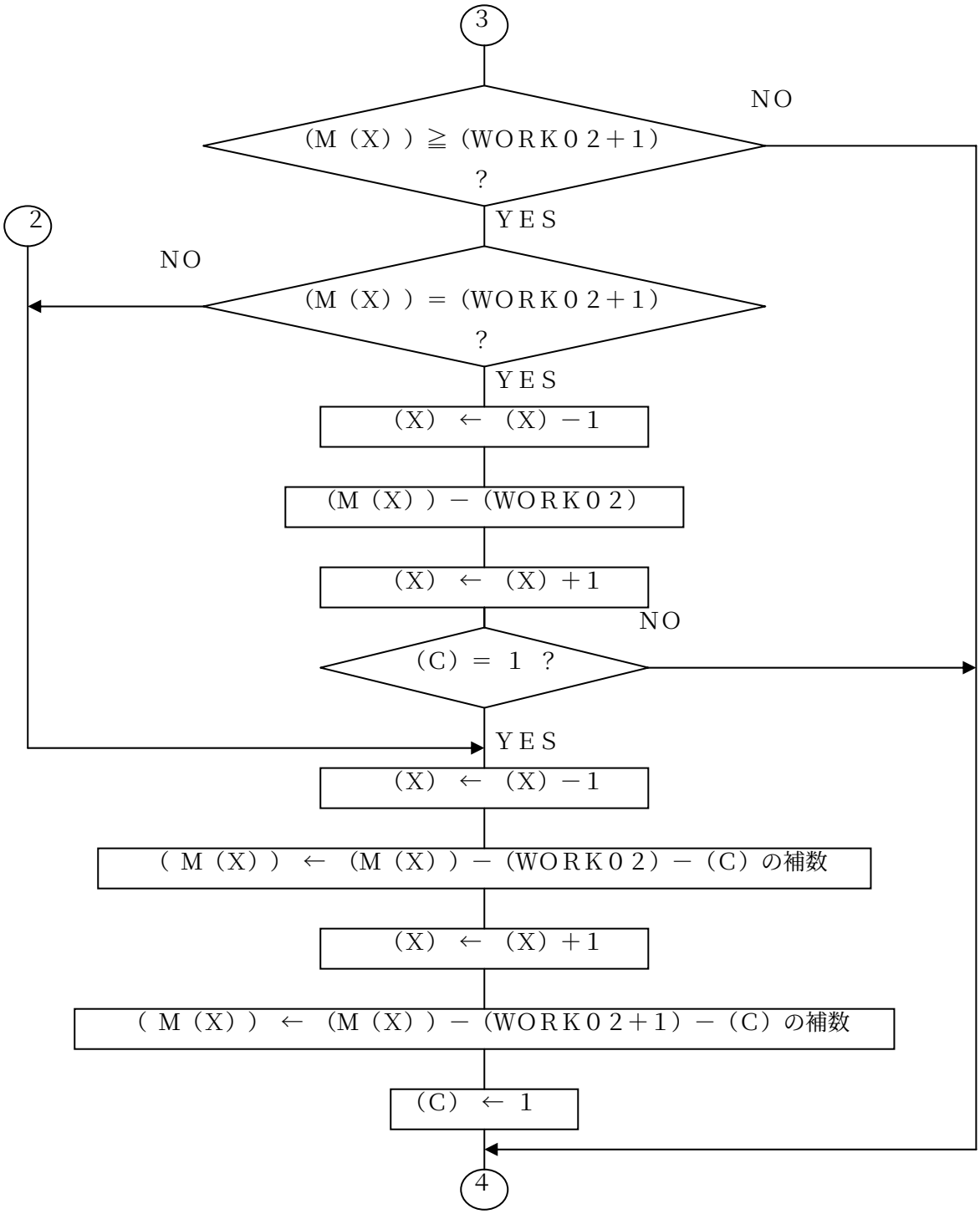
ＷＯＲＫ００＋１、ＷＯＲＫ００の内容をＷＯＲＫ０２＋１、ＷＯＲＫ０２の内容で除算し、商をＷＯＲＫ００＋１、ＷＯＲＫ００に、余りをＷＯＲＫ０１＋１、ＷＯＲＫ０１に入れます。

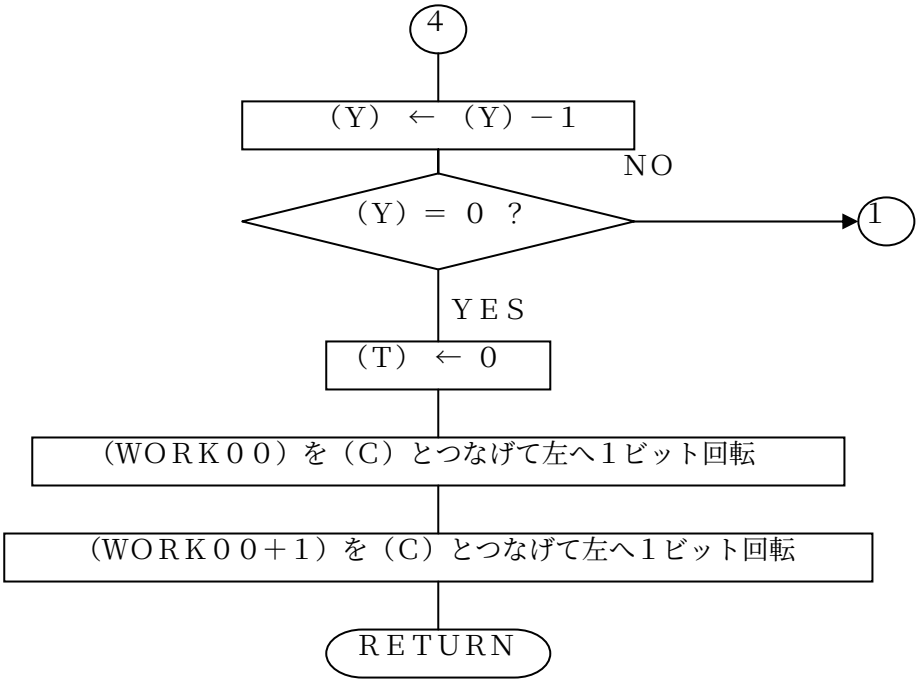
Ｘ修飾演算モードフラグＴを１にして、アキュムレータＡの内容を破壊せずに演算を行ないます。



(3)　フローチャート


ＤＩＶ＿１６
（X）← WORK００＋１
（Y）← １６
（T）← １
（M（X））← ００H
（X）← （X）＋１
（M（X））← ００H
（C）← ０
1
（WORK００）を（C）とつなげて左へ１ビット回転
（WORK００＋１）を（C）とつなげて左へ１ビット回転
（WORK０１）を（C）とつなげて左へ１ビット回転
（WORK０１＋１）を（C）とつなげて左へ１ビット回転
NO
（C）＝ ０ ？
ＹＥＳ
2
3


3
(M (X)) ≧ (WORK02+1)
?
NO
YES
2
NO
(M (X)) = (WORK02+1)
?
YES
(X) ← (X) −1
(M (X)) − (WORK02)
(X) ← (X) +1
NO
(C) = 1 ?
YES
(X) ← (X) −1
(M (X)) ← (M (X)) − (WORK02) − (C) の補数
(X) ← (X) +1
(M (X)) ← (M (X)) − (WORK02+1) − (C) の補数
(C) ← 1
4



注）M（X）はインデックスレジスタXが示す番地のメモリです。

(4)　プログラムリスト

```
;*****************************************************
;
;       16 bits BIN. data division routine
;
;*****************************************************
;
DIV_16:
        LDX     #WORK01         ;Surplus addr. set
        LDY     #16             ;Bit counter set
        SET                     ;T flag set
        LDA     #$00            ;Clear surplus
        INX
        LDA     #$00
        CLC
DIV_01: ROL     WORK00          ;Rotate quotient
        ROL     WORK00+1
        ROL     WORK01          ;Rotate surplus
        ROL     WORK01+1
        BCS     DIV_02          ;C flag 1 ?
;-----------------------------------------------
        CMP     WORK02+1        ;No
        BCC     DIV_03          ;Cannot divide ?
        BNE     DIV_02          ;No
        DEX
        CMP     WORK02
        INX
        BCC     DIV_03          ;Cannot divide ?
;-----------------------------------------------
DIV_02: DEX                     ;No
        SBC   WORK02            ;Surplus - divisor
        INX
        SBC   WORK02+1
        SEC
DIV_03: DEY
        BNE   DIV_01            ;Divide  end ?
;-----------------------------------------------
        CLT                     ;Yes
        ROL   WORK00            ;Rotate quotient
        ROL   WORK00+1
        RTS
```

4. 応用プログラム例

4.1 ファイルハンドリング（転送）

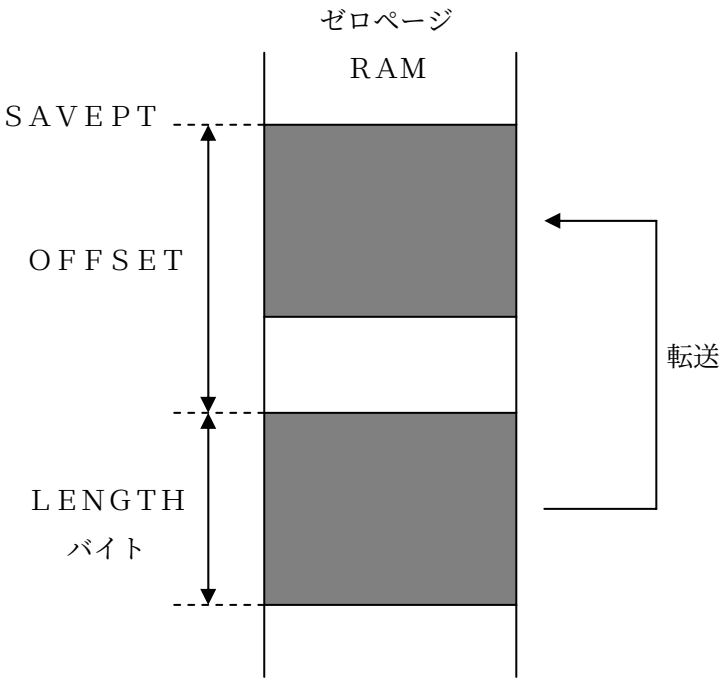
（1） 概要

ゼロページＲＡＭのある部分をファイルメモリとみなして、データ転送処理を行ないます。

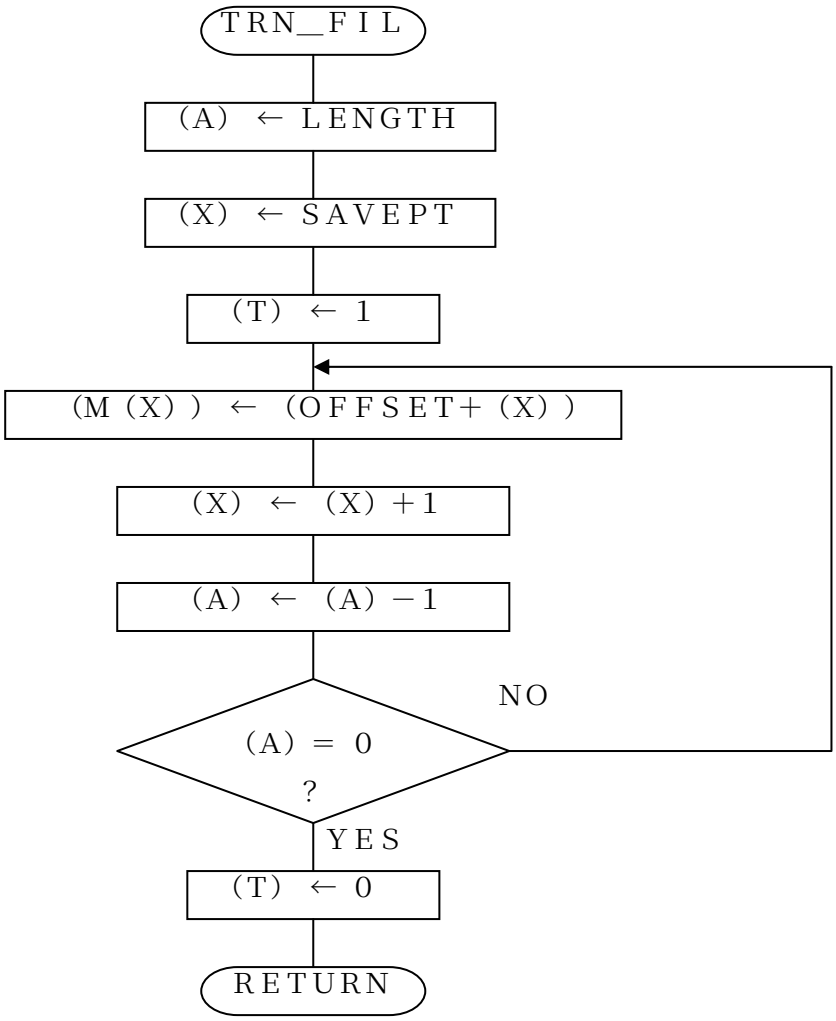
（2） 説明

ゼロページＲＡＭアドレスＳＡＶＥＰＴ＋ＯＦＦＳＥＴより、ＬＥＮＧＴＨバイト分のファイルメモリデータを、ＳＡＶＥＰＴから続くファイルメモリへ転送します。

Ｘ修飾演算モードフラグＴをセットして、メモリ間転送します。また、ポインタはダブルポインタ（ソース／ターゲット）とせず、シングルポインタ＋オフセットします。



(3)　フローチャート



注）M（X）はインデックスレジスタXが示す番地のメモリです。

(4)　プログラムリスト

```
;******************************************************
;
;       File handling (transfer)
;
;******************************************************
;
TRN_FIL:
        LDA   #LENGTH       ;File length
        LDX   #SAVEPT
        SET                 ;T flag set
TRN_01:
        LDA   OFFSET,X      ;Transfer data from
        INX                 ;  - SAVEPT + OFFSET
        DEC   A             ;  - to SAVEPT
        BNE   TRN_01        ;Transfer end ?
        CLT                 ;Yes
        RTS
```

4.2 ファイルハンドリング（交換）

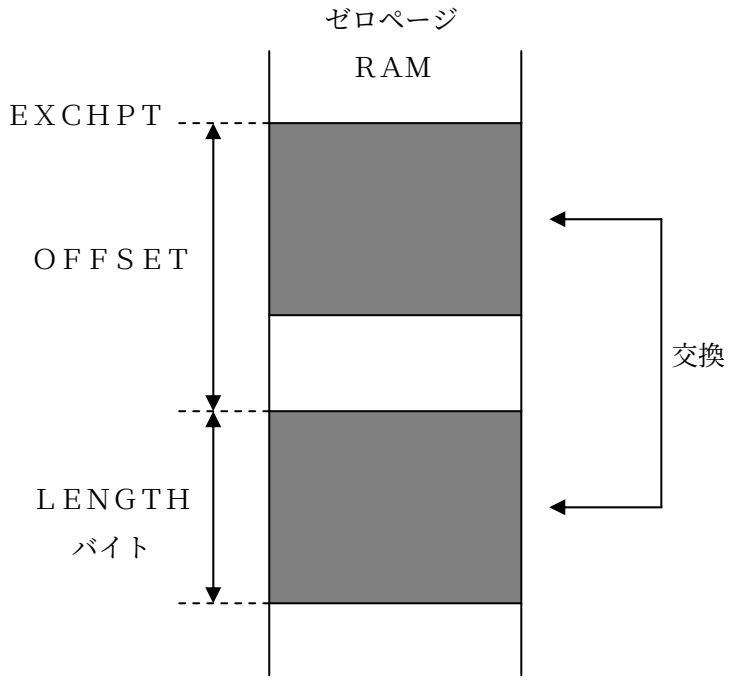
（1） 概要

ゼロページＲＡＭのある部分をファイルメモリとみなして、データ交換処理を行ないます。

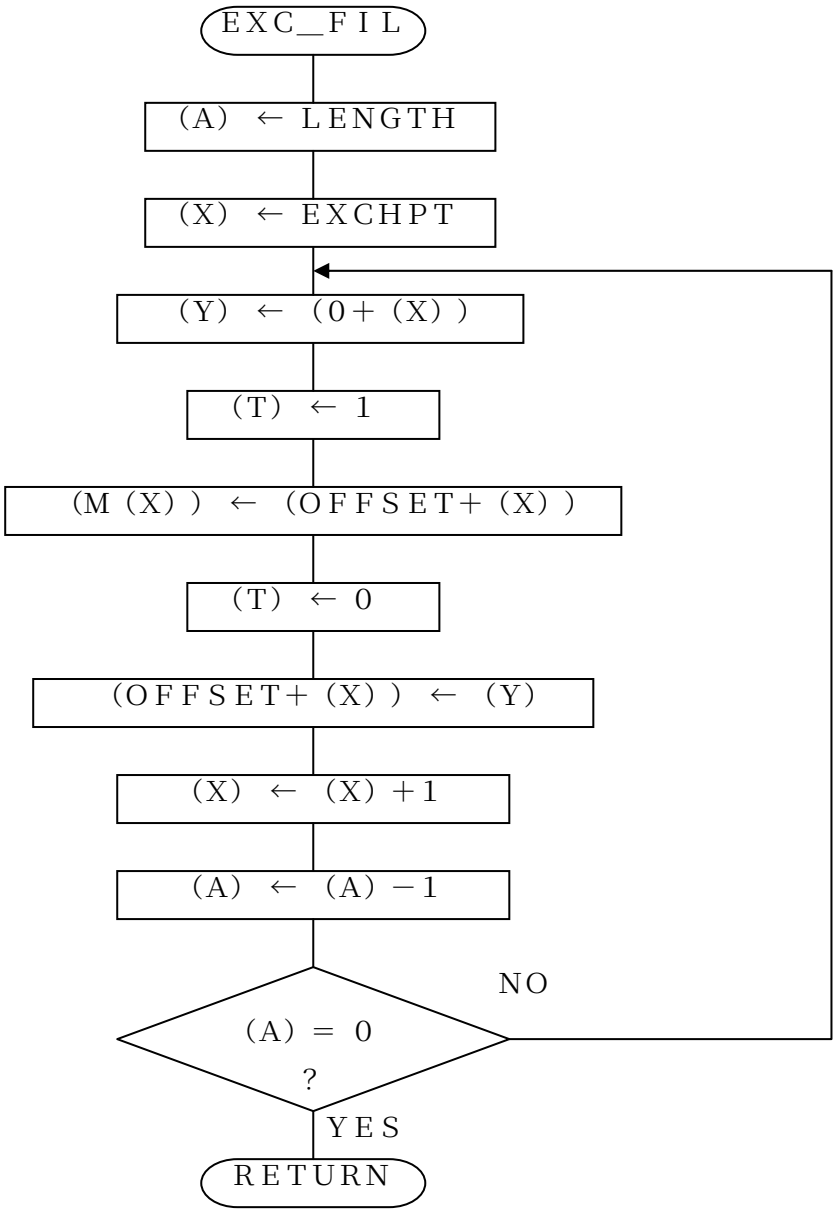
（2） 説明

ゼロページＲＡＭアドレスＥＸＣＨＰＴ＋ＯＦＦＳＥＴより、ＬＥＮＧＴＨバイト分のファイルメモリデータと、ＥＸＣＨＰＴより、ＬＥＮＧＴＨバイト分のファイルメモリデータとを交換します。

Ｘ修飾演算モードフラグＴをセットして、メモリ間転送します。また、ポインタはダブルポインタ（ソース／ターゲット）とせず、シングルポインタ＋オフセットします。



（3）　フローチャート



注）M（X）はインデックスレジスタXが示す番地のメモリです。

(4)　プログラムリスト

```
;*****************************************************
;
;       File handling (exchange)
;
;*****************************************************
;
EXC_FIL:
        LDA   #LENGTH        ;File length
        LDX   #EXCHPT
EXC_01:
        LDY   0,X            ;Exchange data of
        SET                  ;  - EXCHPT + OFFSET
        LDA   OFFSET,X         ;  - with EXCHPT
        CLT
        STY   OFFSET,X
        INX
        DEC   A
        BNE   EXC_01         ;Exchange end ?
        RTS                  ;Yes
```

4.3 コード変換（パックドＢＣＤ→アンパックドＢＣＤ）

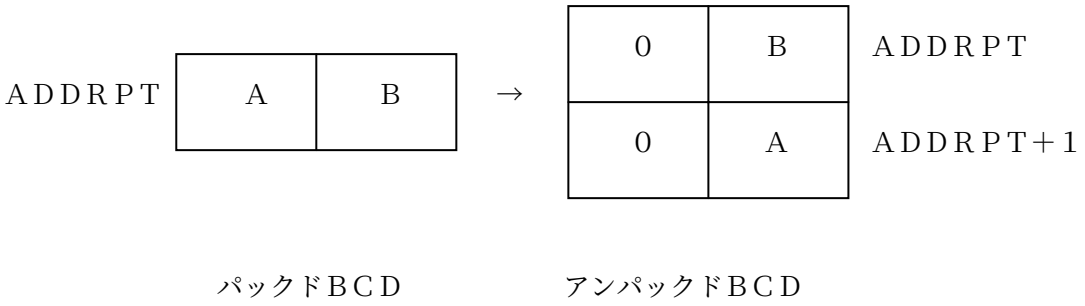
（1） 概要

パックドＢＣＤデータを、アンパックドＢＣＤデータに変換します。

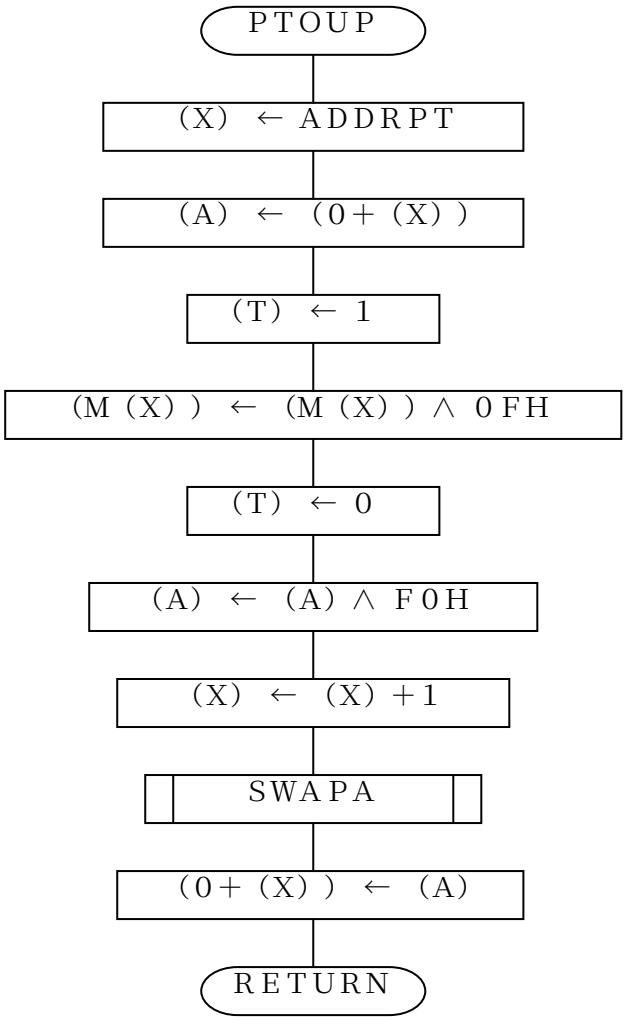
（2） 説明

ゼロページＲＡＭアドレスＡＤＤＲＰＴのパックドＢＣＤデータを、アンパックドＢＣＤに変換して、ＡＤＤＲＰＴとＡＤＤＲＰＴ＋１に入れます。

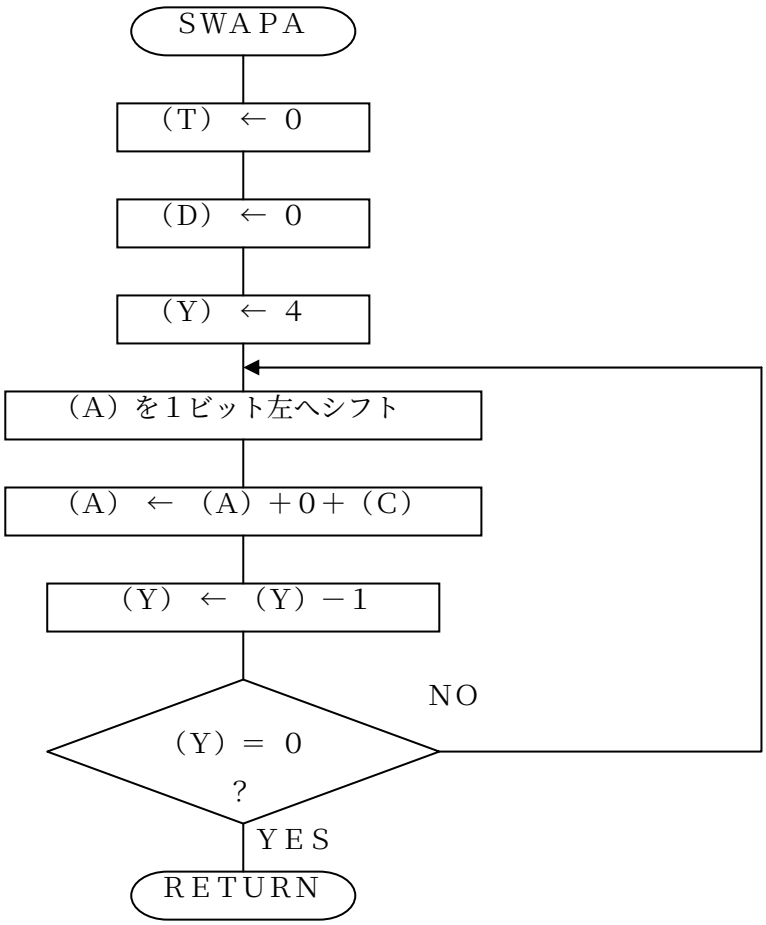
下記のように、パックドＢＣＤは、１バイトに２桁のＢＣＤ数値をパックしたものです。また、アンパックドＢＣＤは、パックドＢＣＤを２つに分割して上位４ビットを０にした２バイトのデータです。



（3）　フローチャート



注）M（X）はインデックスレジスタXが示す番地のメモリです。


SWAPA
（T）← 0
（D）← 0
（Y）← 4
（A）を１ビット左へシフト
（A）←（A）＋0＋（C）
（Y）←（Y）－1
（Y）＝ 0
？
NO
YES
RETURN

(4)　プログラムリスト

```
;******************************************************
;
;       Packed BCD -> unpacked BCD
;
;******************************************************
;
PTOUP:
        LDX   #ADDRPT
        LDA   0,X           ;Get packed BCD data
        SET                 ;T flag set
        AND   #0FH          ;Unpacked BCD data L
        CLT                 ;T flag clear
        AND   #0F0H
        INX
        JSR   SWAPA         ;Swap A
        STA   0,X           ;Unpacked BCD data H
        RTS
;******************************************************
;
;       Swap A register
;
;******************************************************
;
SWAPA:
        CLT                 ;T flag clear
        CLD                 ;Decimal mode clear
        LDY   #4
SWAPA1:
        ASL   A
        ADC   #0
        DEY
        BNE   SWAPA1
        RTS
```

4.4　コード変換（アンパックドＢＣＤ→パックドＢＣＤ）

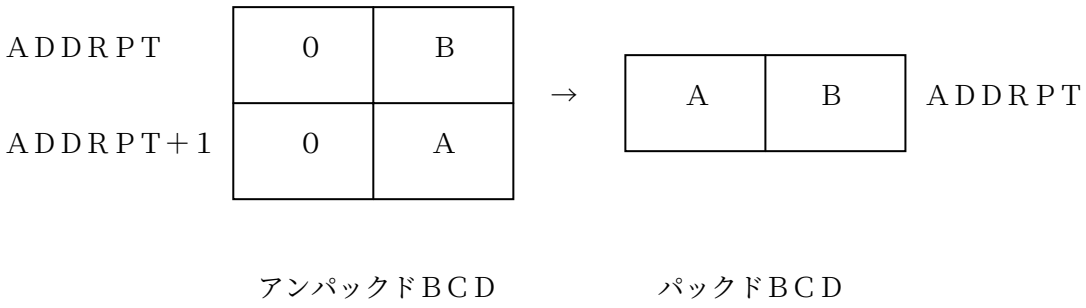
（1）　概要

アンパックドＢＣＤデータを、パックドＢＣＤデータに変換します。

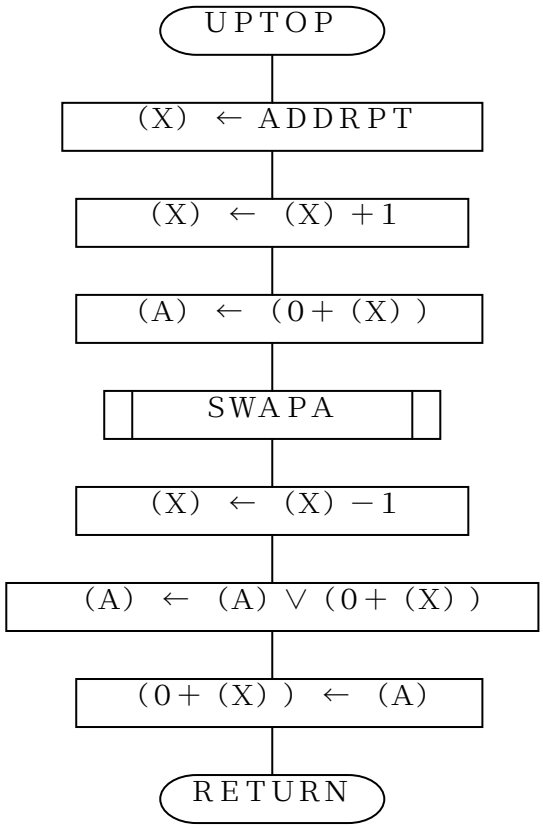
（2）　説明

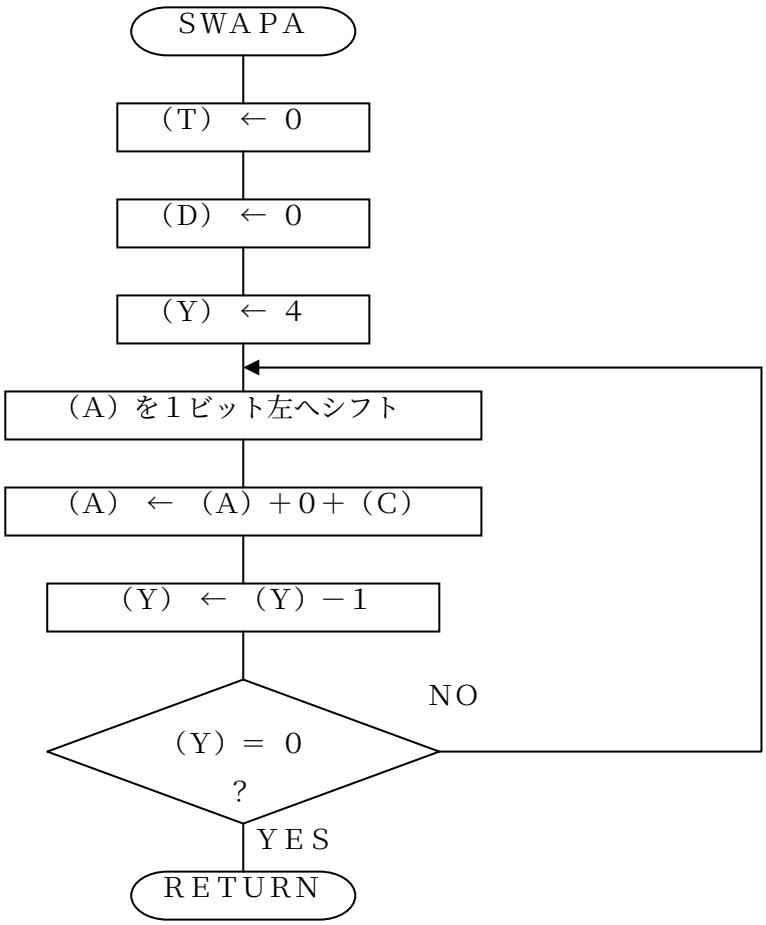
ゼロページＲＡＭアドレスＡＤＤＲＰＴ＋１とＡＤＤＲＰＴのアンパックドＢＣＤデータを、パックドＢＣＤに変換して、ＡＤＤＲＰＴに入れます。

下記のように、パックドＢＣＤは、１バイトに２桁のＢＣＤ数値をパックしたものです。また、アンパックドＢＣＤは、パックドＢＣＤを２つに分割して上位４ビットを０にした２バイトのデータです。



(3)　フローチャート




SWAPA
（T）← 0
（D）← 0
（Y）← 4
（A）を１ビット左へシフト
（A）←（A）＋0＋（C）
（Y）←（Y）－1
（Y）＝ 0
？
NO
YES
RETURN

(4)　プログラムリスト

```
;******************************************************
;
;       Unpacked BCD -> packed BCD
;
;******************************************************
;
UPTOP:
        LDX    #ADDRPT
        INX
        LDA    0,X             ;Get unpacked BCD data H
        JSR    SWAPA           ;Swap A
        DEX
        ORA    0,X             ;Packed BCD data
        STA    0,X
        RTS
;******************************************************
;
;       Swap A register
;
;******************************************************
;
SWAPA:
        CLT                    ;T flag clear
        CLD                    ;Decimal mode clear
        LDY    #4
SWAPA1:
        ASL    A
        ADC    #0
        DEY
        BNE    SWAPA1
        RTS
```

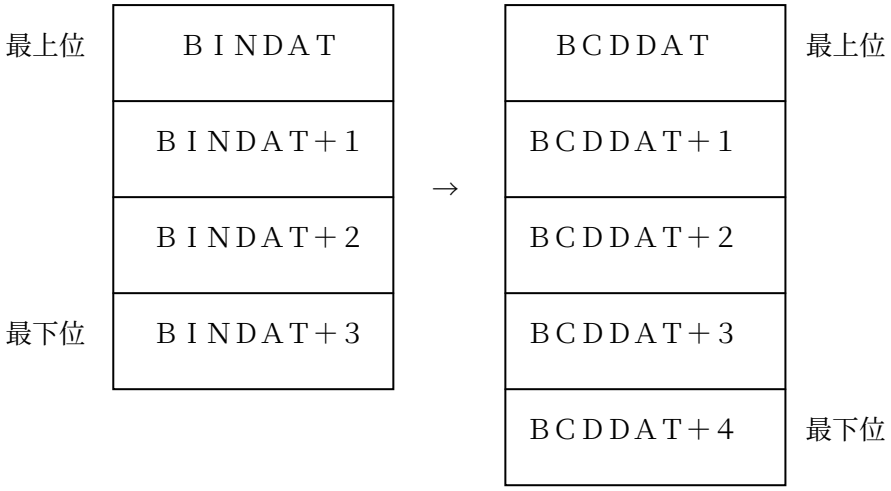
4.5　コード変換（ＢＩＮ→ＢＣＤ）

（1）　概要

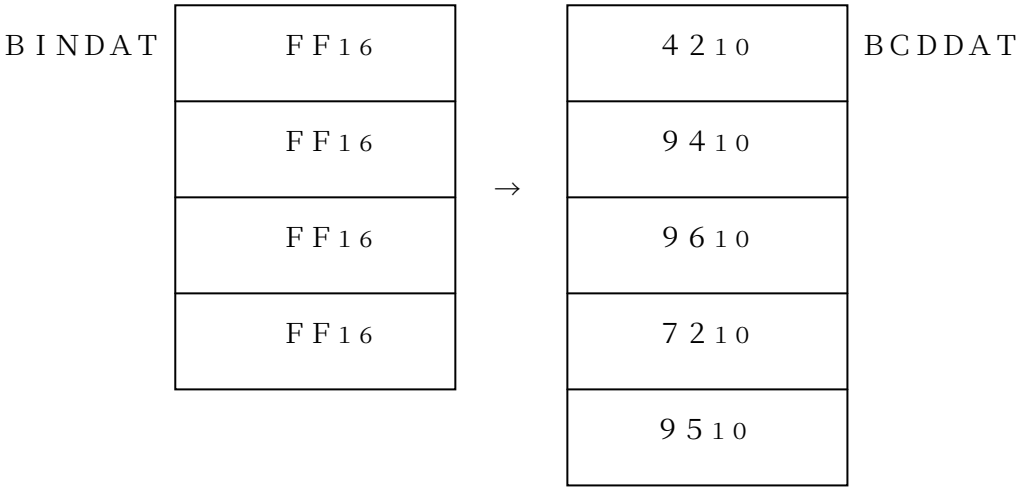
４バイトＢＩＮデータを、５バイトＢＣＤデータに変換します。

（2）　説明

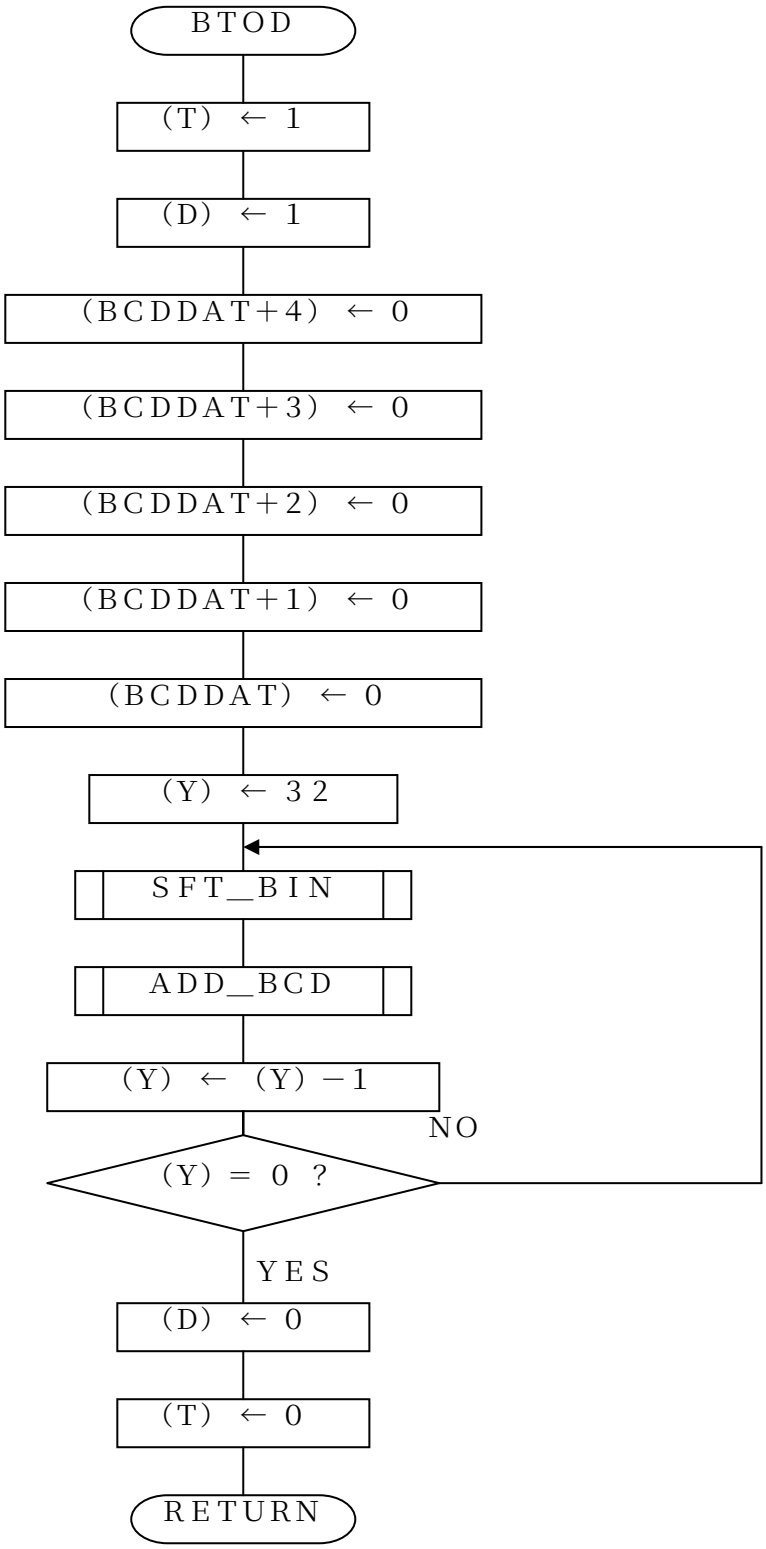
ゼロページＲＡＭアドレスＢＩＮＤＡＴ、ＢＩＮＤＡＴ＋１、ＢＩＮＤＡＴ＋２、ＢＩＮＤＡＴ＋３の４バイトのＢＩＮデータを、５バイトのＢＣＤデータに変換して、ＢＣＤＤＡＴ、ＢＣＤＤＡＴ＋１、ＢＣＤＤＡＴ＋２、ＢＣＤＤＡＴ＋３、ＢＣＤＤＡＴ＋４に入れます。

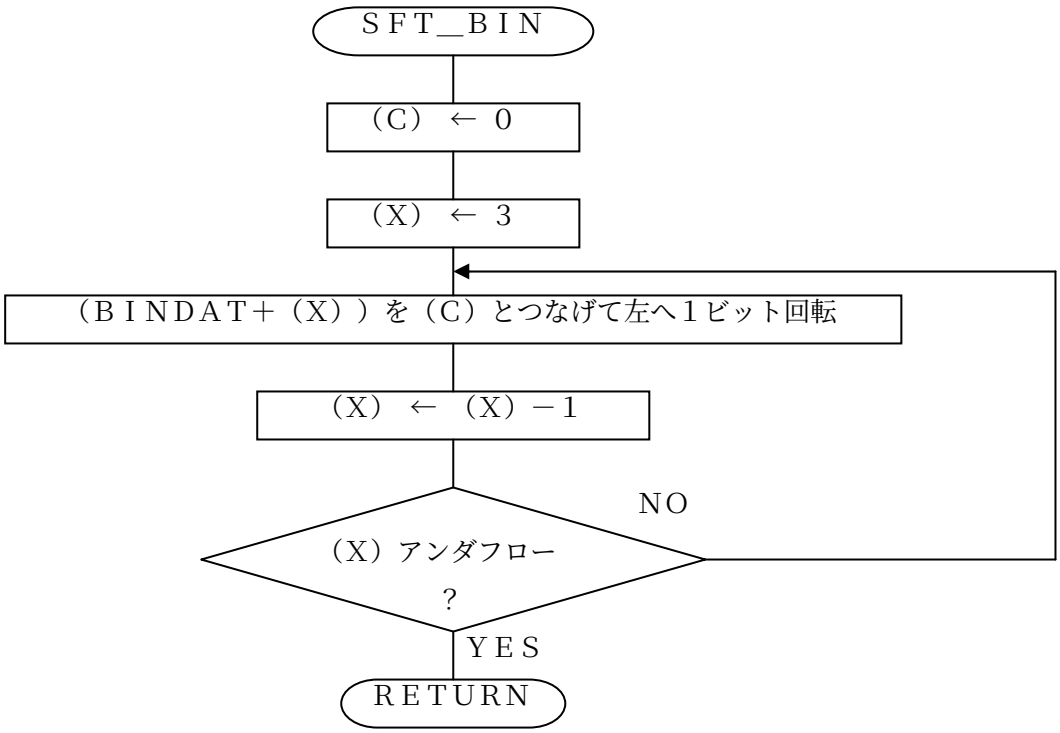


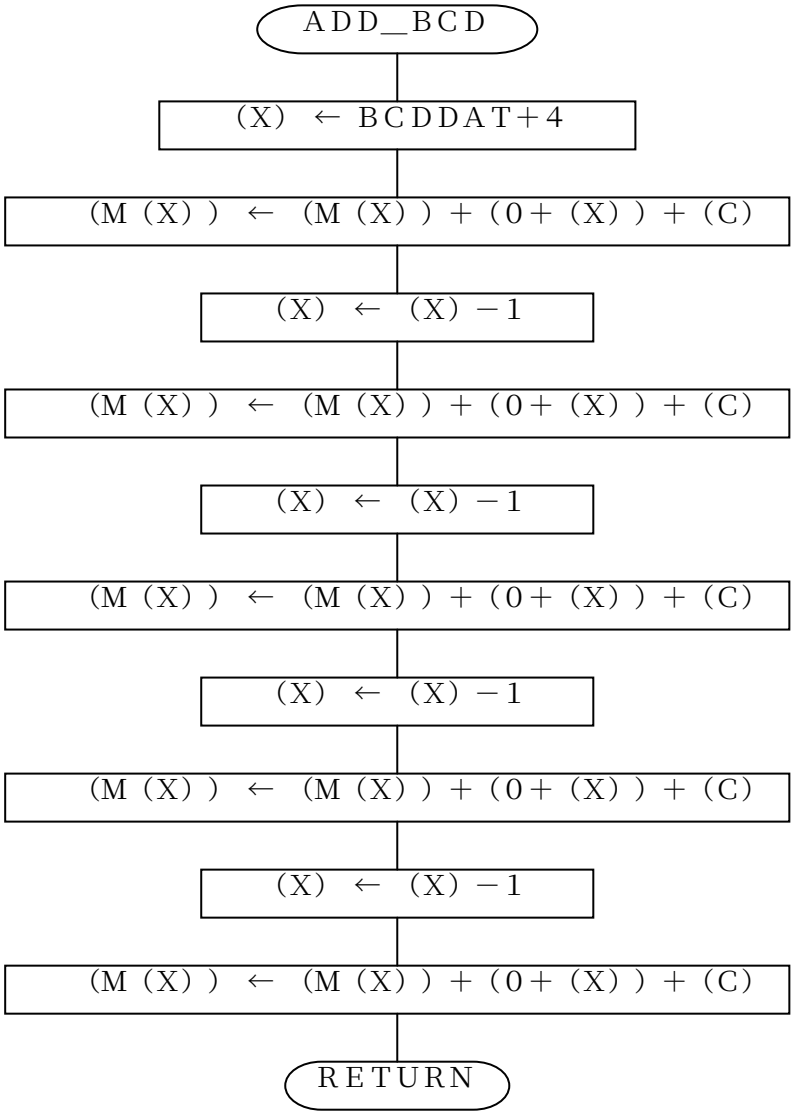
例）ＢＩＮデータがＦＦＦＦＦＦＦＦＨの場合、ＢＣＤデータは４２９４９６７２９５となります。



(3)　フローチャート


BTOD
(T) ← 1
(D) ← 1
(BCDDAT+4) ← 0
(BCDDAT+3) ← 0
(BCDDAT+2) ← 0
(BCDDAT+1) ← 0
(BCDDAT) ← 0
(Y) ← 32
SFT_BIN
ADD_BCD
(Y) ← (Y) −1
NO
(Y) = 0 ?
YES
(D) ← 0
(T) ← 0
RETURN


ＳＦＴ＿ＢＩＮ
（Ｃ）← ０
（Ｘ）← ３
（ＢＩＮＤＡＴ＋（Ｘ））を（Ｃ）とつなげて左へ１ビット回転
（Ｘ）← （Ｘ）－１
ＮＯ
（Ｘ）アンダフロー
？
ＹＥＳ
ＲＥＴＵＲＮ



注）Ｍ（Ｘ）はインデックスレジスタＸが示す番地のメモリです。

(4)　プログラムリスト

```
;******************************************************
;
;       BIN -> BCD
;
;******************************************************
;
BTOD:
        SET                     ;T flag set
        SED                     ;Decimal mode set
        LDM   #0,BCDDAT+4       ;Clear BCD result area
        LDM   #0,BCDDAT+3
        LDM   #0,BCDDAT+2
        LDM   #0,BCDDAT+1
        LDM   #0,BCDDAT
        LDY   #32               ;Yes
BTOD_01:
        JSR   SFT_BIN           ;Left shift BIN data
        JSR   ADD_BCD           ;2*(BCD)+C -> (BCD)
        DEY
        BNE   BTOD_01           ;Convert end ?
        CLD                     ;Yes
        CLT                     ;T flag clear
        RTS
;******************************************************
;
;       Left shift BIN data
;
;******************************************************
;
SFT_BIN:
        CLC                  ;C flag clear
        LDX   #3
SFT_01:
        ROL   BINDAT,X
        DEX
        BPL   SFT_01         ;Shift end ?
        RTS                  ;Yes
;******************************************************
;
;       2*(BCD)+C -> (BCD)
;
;******************************************************
;
ADD_BCD:
        LDX   #BCDDAT+4
        ADC   0,X
        DEX
        ADC   0,X
        DEX
        ADC   0,X
        DEX
        ADC   0,X
        DEX
        ADC   0,X
        RTS
```

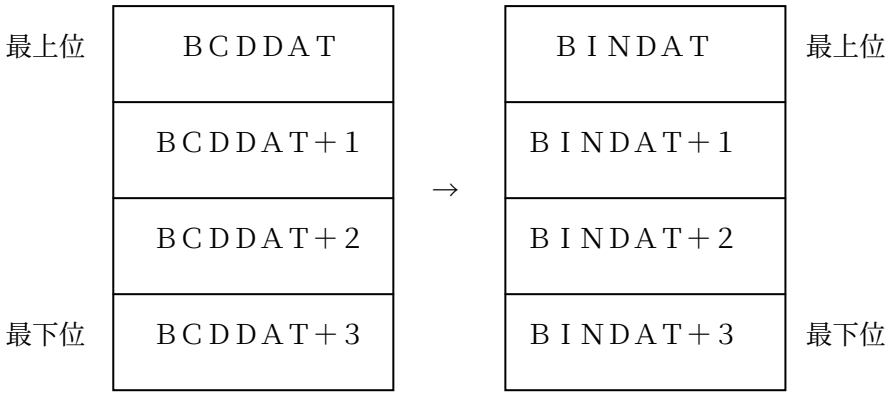
4.6　コード変換（ＢＣＤ→ＢＩＮ）

（1）　概要

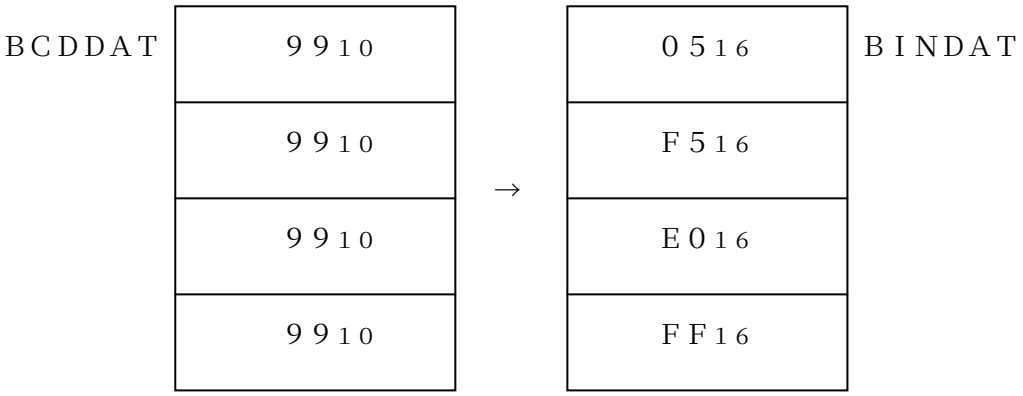
４バイトＢＣＤデータを、４バイトＢＩＮデータに変換します。

（2）　説明

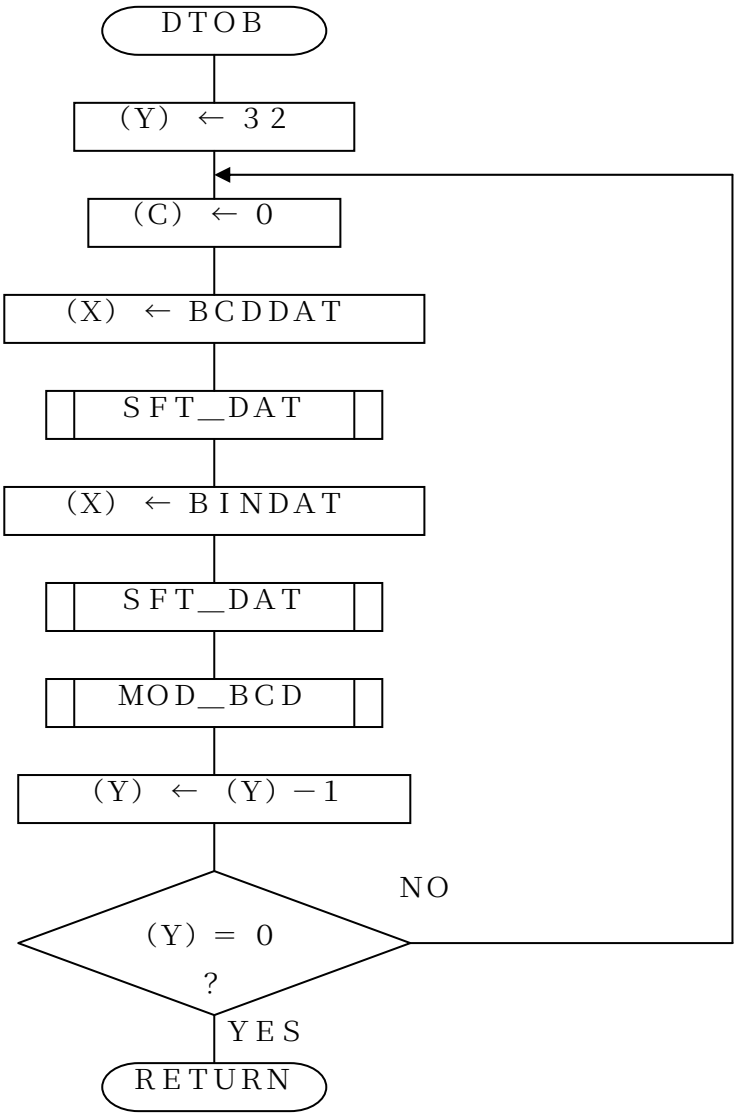
ゼロページＲＡＭアドレスＢＣＤＤＡＴ、ＢＣＤＤＡＴ＋１、ＢＣＤＤＡＴ＋２、ＢＣＤＤＡＴ＋３の４バイトのＢＣＤデータを、４バイトのＢＩＮデータに変換して、ＢＩＮＤＡＴ、ＢＩＮＤＡＴ＋１、ＢＩＮＤＡＴ＋２、ＢＩＮＤＡＴ＋３に入れます。

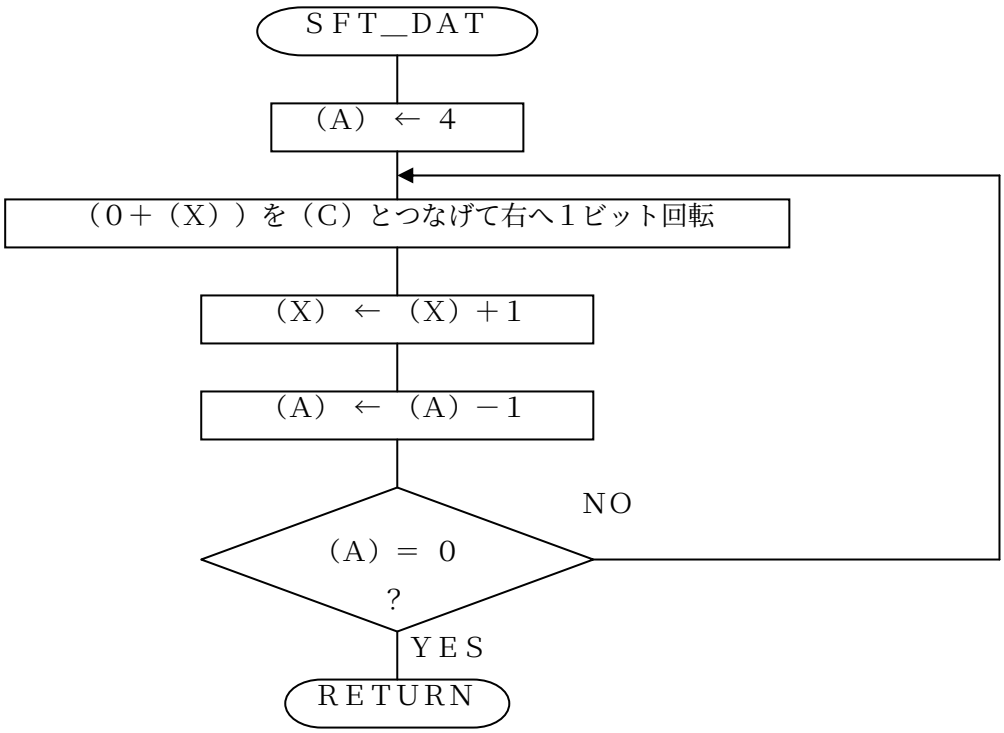


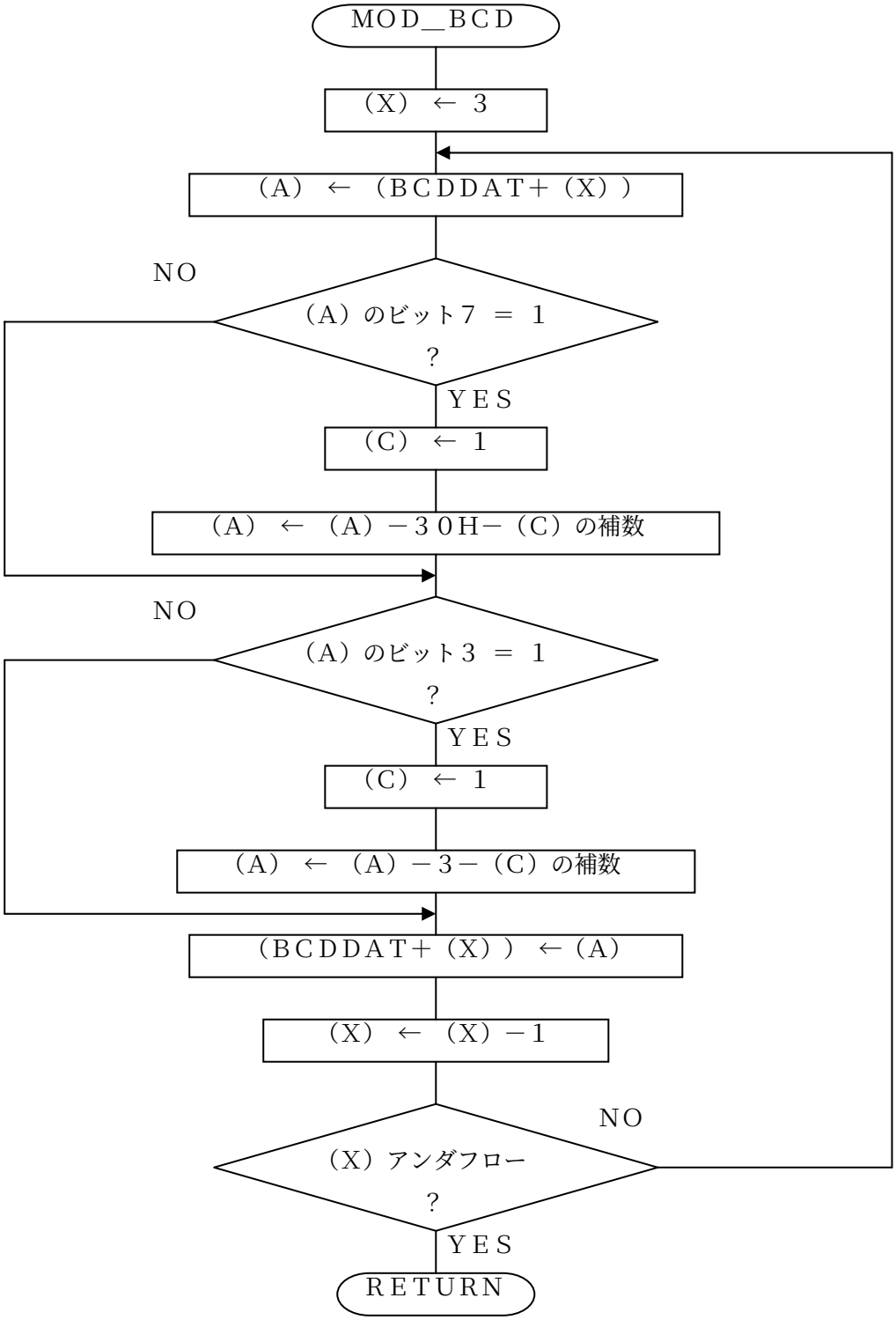
例）ＢＣＤデータが９９９９９９９９の場合、ＢＩＮデータは０５Ｆ５Ｅ０ＦＦＨとなります。



(3)　フローチャート


DTOB
(Y) ← 32
(C) ← 0
(X) ← BCDDAT
SFT_DAT
(X) ← BINDAT
SFT_DAT
MOD_BCD
(Y) ← (Y) −1
NO
(Y) = 0
?
YES
RETURN


ＳＦＴ＿ＤＡＴ
（Ａ）　←　４
（０＋（Ｘ））を（Ｃ）とつなげて右へ１ビット回転
（Ｘ）　←　（Ｘ）＋１
（Ａ）　←　（Ａ）－１
（Ａ）　＝　０
？
ＮＯ
ＹＥＳ
ＲＥＴＵＲＮ


MOD_BCD
（X）← 3
（A）← （BCDDAT＋（X））
NO
（A）のビット7 ＝ 1
？
YES
（C）← 1
（A）← （A）－30H－（C）の補数
NO
（A）のビット3 ＝ 1
？
YES
（C）← 1
（A）← （A）－3－（C）の補数
（BCDDAT＋（X））←（A）
（X）← （X）－1
NO
（X）アンダフロー
？
YES
RETURN

(4)　プログラムリスト

```
;******************************************************
;
;       BCD -> BIN
;
;******************************************************
;
DT0B:
        LDY    #32
DT0B_01:
        CLC
        LDX    #BCDDAT         ;Point to BCDDAT
        JSR    SFT_DAT         ;Right shift BCD data
        LDX    #BINDAT         ;Point to BINDAT
        JSR    SFT_DAT         ;Right shift BIN data
        JSR    MOD_BCD         ;Get modified BCD data
        DEY
        BNE    DT0B_01         ;Shift end ?
        RTS                    ;Yes !
;******************************************************
;
;       Right shift data
;
;******************************************************
;
SFT_DAT:
        LDA    #4
SFT_02:
        ROR    0,X
        INX
        DEC    A
        BNE    SFT_02
        RTS
;******************************************************
;
;       Modify BCD data
;
;******************************************************
;
MOD_BCD:
        LDX    #3
MOD_01:
        LDA    BCDDAT,X
        BBC    7,A,MOD_02
        SEC
        SBC    #30H
MOD_02:
        BBC    3,A,MOD_03
        SEC
        SBC    #3
MOD_03:
        STA    BCDDAT,X
        DEX
        BPL    MOD_01
        RTS
```

4.7　ＳＧＮ関数

（1）　概要

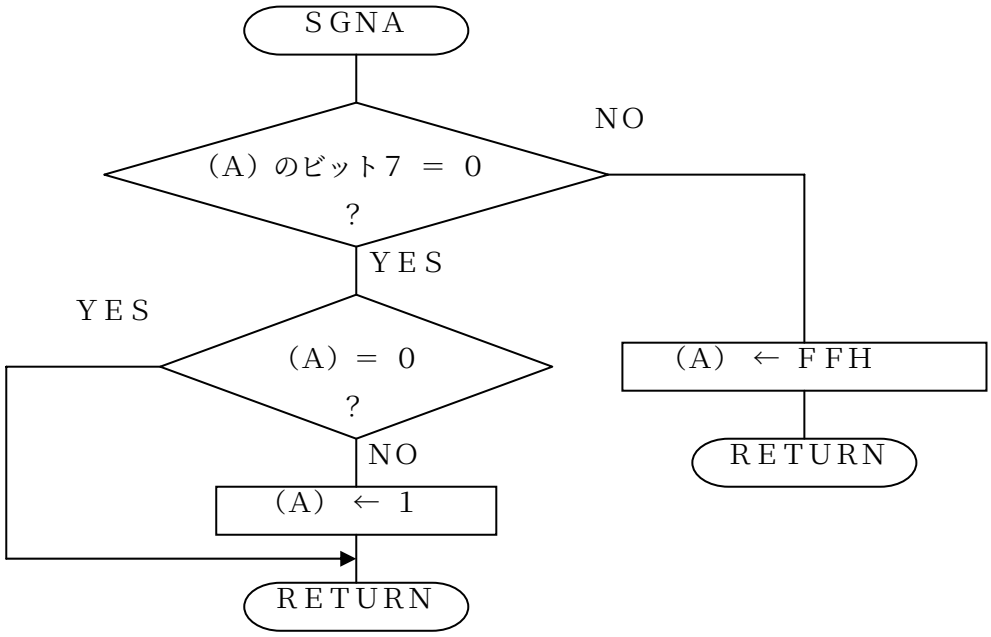
アキュムレータＡのＳＧＮ関数です。

（2）　説明

アキュムレータＡの内容に対するＳＧＮ値を作成し、その値をアキュムレータＡに入れます。アキュムレータＡのビット７は符号ビットです。この時のＳＧＮ関数は下記のようになります。

（Ａ）＞０のとき、ＳＧＮ（Ａ）＝１
（Ａ）＝０のとき、ＳＧＮ（Ａ）＝０
（Ａ）＜０のとき、ＳＧＮ（Ａ）＝－１

（3）　フローチャート



（4）　プログラムリスト

```
;*******************************************************
;
;       SGN(A)
;
;*******************************************************
;
SGNA:
        BBS     7,A,SGN_01
        CMP     #0
        BEW     SGN_02
        LDA     #1
SGN_02:
        RTS
SGN_01:
        LDA     #$FF
        RTS
```

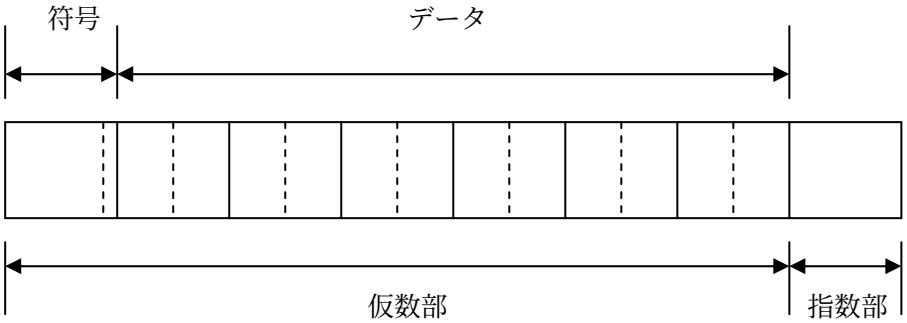
4.8 ＢＣＤ１２桁浮動小数点四則演算

（1） 概要

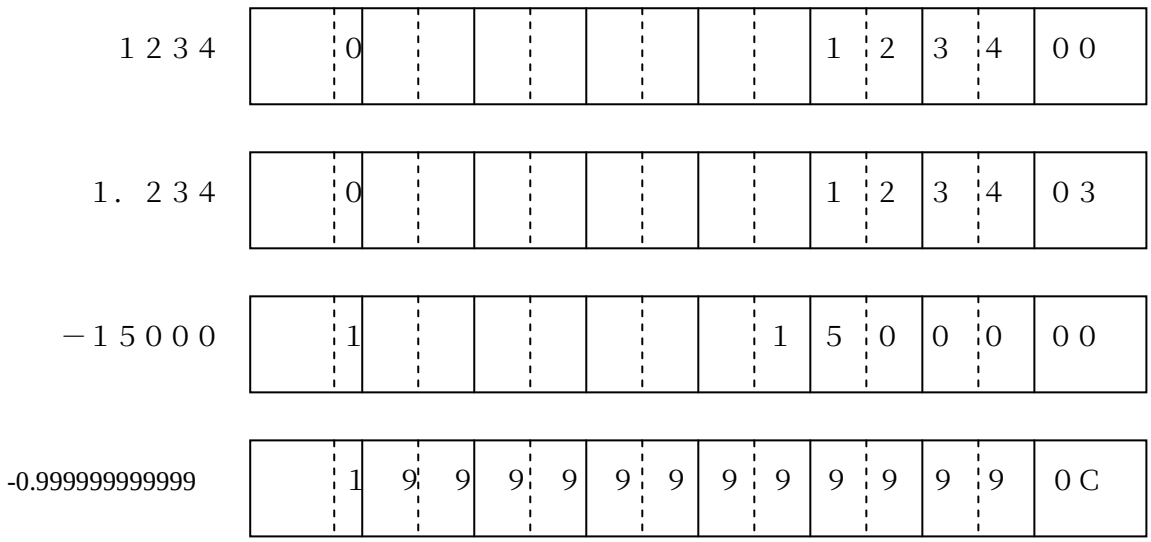
ＢＣＤ１２桁の浮動小数点の四則演算を行ないます。

（2） 説明

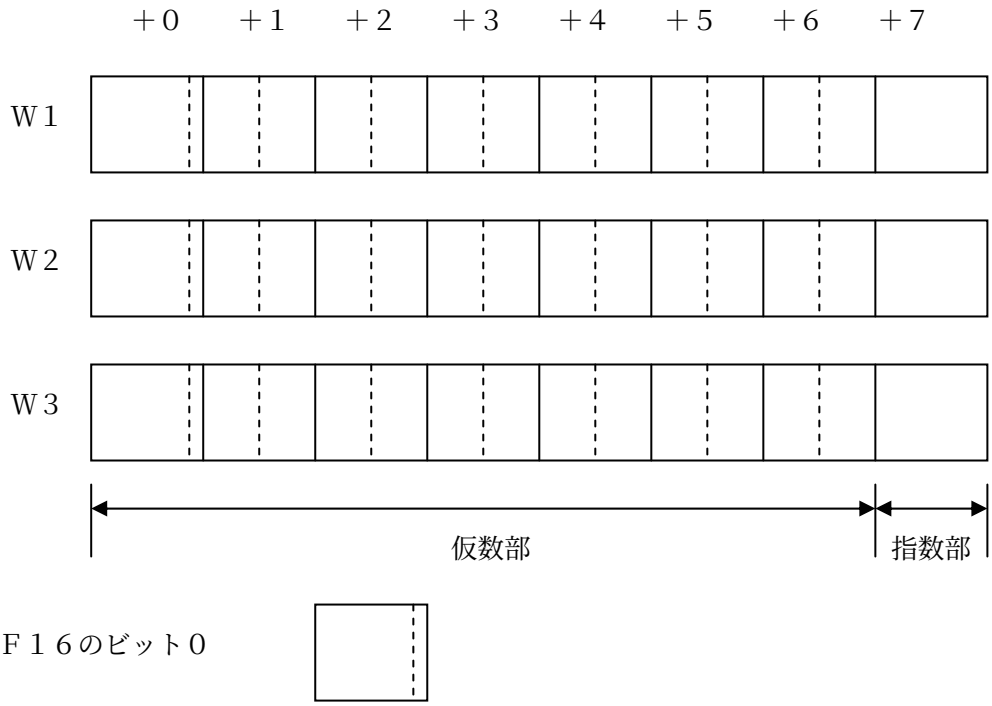
データフォーマットを下記に示します。このフォーマットは、仮数部７バイト（１バイト符号ビット、６バイトデータ）と指数部１バイトで表現されています。仮数部符号は０で正、１で負を表わし、仮数部データは有効数字１２桁（ＢＣＤ）となっています。指数部は、数値として００Ｈ～０ＣＨまでの値しかとることができず、０ＤＨ～ＦＦＨの値がセットされた場合は、演算結果が保証されません。



このフォーマットで表現された数の例を下記に示します。



このフォーマットは、下記に示されるＷ１、Ｗ２、Ｗ３の３つのファイル（ゼロページＲＡＭ領域）に使用されています。



- Ｗ１：被演算数及び結果がストアされます。
- Ｗ２：演算数がストアされます。アドレスとしては、Ｗ２＝Ｗ１＋８となります。
- Ｗ３：乗除算を行なう場合に、ワークファイルとして使用されます。アドレスとしては、Ｗ３＝Ｗ２＋８となります。
- Ｆ１６のビット０：エラーが発生した場合、１になります。

次に各演算のコーリングシーケンスを示します。加減算は、Ｗ２のビット３の値で判別しているので注意してください。

・乗算

①Ｗ１に被演算数をセットする。

②Ｗ２に演算数をセットする。

③ＪＳＲ　ＭＵＬＴ

・除算

①②は乗算と同じ

③ＪＳＲ　ＤＩＶ

・加算

①②は乗算と同じ

③ＪＳＲ　ＡＤＳＢ

・減算

①②は乗算と同じ

③Ｗ２のビット３を１にする。

④ＪＳＲ　ＡＤＳＢ

これらの処理を実行した後、ゼロページＲＡＭアドレスＦ１６のビット０のデータで演算結果がエラーかどうかチェックできます。（Ｆ１６のビット０）＝１ならばエラーです。下記に各演算に対するエラーを示します。

演算名	エラー
加算	演算結果が９９９９９９９９９９９９を超えた場合
減算	演算結果が－９９９９９９９９９９９９を超えた場合
乗算	演算結果が±９９９９９９９９９９９９を超えた場合
除算	• ０で除したとき • 演算結果が±９９９９９９９９９９９９を超えた場合

(3)　プログラムリスト

```
;*****************************************************
;
;       12 digits BCD number addition,subtraction
;       ,multiplication,division
;
;*****************************************************
;
;
;       RAM ASSIGN
;
;       0   1   2   3   4   5   6   7
; W1:  SIGN  MSD<-----------------LSD INDX
; W2:  SIGN  MSD<-----------------LSD INDX
; W3:  SIGN  MSD<-----------------LSD INDX
;
;
;*****************************************************
;
;       W1 <- W1/W2
;
;*****************************************************
;
DIVI:
        JSR    BFITL          ;W3<-W1 copy,W1<-0
        SBC    8,X            ;D=0,T=1,W1 INDX<-W1 INDX-W2 INDX
        AND    #0FH           ;W1 INDX MSD clear
        SEB    0,F16
        BCC    DIV1           ;W1 INDX<W2 INDX?
        CLB    0,F16          ;No
DIV1:   CLT                   ;T<-0
        SED                   ;Decimal mode
        JSR    SUB0           ;W1<-W1-W2
        BCS    DIV2           ;Borrow arise?
        TST    W1             ;Yes,then test W1
        BEQ    DIV3
        DEC    W1
DIV2:   CLD                   ;Binary mode
        RRF    W3+6
        CLC
        LDA    W3+6
        ADC    #10H
        STA    W3+6           ;W3 LSD<-W3 LSD+1
        RRF    W3+6
        BCC    DIV1           ;W3 LSD>=15?
ERROR:  SEB    0,F16          ;0  Division error
        CLD                   ;Decimal mode clear
        RTS
;
DIV3:   JSR    ADD0           ;W1<-W1+W2,add back
        LDA    W3+7
        CMP    #12
        BCS    DIV6
        INC    W3+7
;                             ;W1 MSD<--W1 LSD<--W3 MSD<--W3 LSD
DIV4:   LDY    #4             ;4 bits left rotate
DIV5:   LDX    #6
        CLC
DIV51:  ROL    W3,X
        DEX
        BNE    DIV51
        LDX    #6
        LDA    #7
DIV52:  ROL    W1,X
        DEX
        DEC    A
        BNE    DIV52
        DEY
        BNE    DIV5           ;Rotate end?
        INX
        SET                   ;T flag set
        AND    #0FH
        BRA    DIV1
```

```
;
DIV6:   LDA   W3+1
        AND   #0F0H
        BNE   DIV10
        BBS   0,F16,DIV7
        LDA   W1+7
        CMP   #12
        BCS   DIV10
DIV7:   INC   W1+7
        BBC   4,W1+7,DIV41
        CLB   4,W1+7
        CLB   0,F16
DIV41:
        BRA   DIV4
ERROR1:
        BRA   ERROR
;
DIV10:  BBS   0,F16,ERROR    ;Over flow error
        LDA   W1+7
        CMP   #13
        BCS   ERROR          ;Under flow error
DIV12:  LDY   #7             ;W1<-W3 copy
        LDX   #W1
        SET                  ;T<-1
DIV15:  LDA   16,X
        INX
        DEY
        BNE   DIV15
;
        CLT                  ;T<-0
        JSR   INDX           ;"0" condense
        CLD                  ;Decimal mode clear
        CLC
        LDA   W1
        ADC   W2
        STA   W1             ;Get sign bit
        CLB   0,F16          ;Normal return
        RTS
;*******************************************************
;
;       W1 <- W1*W2
;
;*******************************************************
;
MULT:
        JSR   BFITL          ;W3 <- W1 copy,W1 clear
        CLC                  ;C flag clear
        ADC   8,X            ;W1_INDX <- W1_INDX+W2_INDX
        CLB   0,F16
        BBC   4,W1+7,MULT2   ;W1_INDX <- W_INDX+W2_INDX
        SEB   0,F16          ;Yes
        CLB   4,W1+7
MULT2:
        CLT                  ;T flag clear
MULT7:
        LDA   W3+6
        AND   #$0F
        BNE   MULT3
        INC   W3+7           ;W1_MSD -> W1_LSD -> W3_MSD -> W3_LSD
MULT9:
        LDY   #4
MULT6:
        LDX   #0
        LDA   #7
        CLC
MULT4:
        ROR   W1,X
        INX
        DEC   A
        BNE   MULT4
        LDX   #1
        LDA   #6
```

```
MULT5:
        ROR    W3,X
        INX
        DEC    A
        BNE    MULT5
        DEY
        BNE    MULT6           ;4 bits right rotate end ?
        LDA    W3+7
        CMP    #12
        BCC    MULT7
        BBS    0,F16,MULT8
        LDX    #W1+1           ;W1 "0" test
MULT81:
        LDA    0,X
        BNE    MULT8
        INX
        CPX    #W1+7
        BNE    MULT81
        BRA    DIV12           ;Get result (to division routine)
MULT8:
        LDA    W1+7
        BNE    MULT10
        BBC    0,F16,ERROR1    ;Overflow error
        CLB    0,F16
MULT10:
        DEC    W1+7
        LDA    W1+7
        AND    #$0F
        STA    W1+7
        BRA    MULT9
MULT3:
        DEC    W3+6
        SED
        JSR    ADD0
        BCC    MULT31
        INC    W1
MULT31:
        BRA    MULT2
;*******************************************************
;
;       W1 <- W1-W2
;
;*******************************************************
;
SUB0:
        SEC
        LDA    #6
        LDX    #W1+6
        SET                    ;T flag set
SUB01:
        SBC    8,X
        DEX
        DEC    A
        BNE    SUB01
        CLT                    ;T flag clear
        RTS
;*******************************************************
;
;       W1 <- W1+W2
;
;*******************************************************
;
ADD0:
        CLC
        LDA    #6
        LDX    #W1+6
        SET                    ;T flag set
ADD01:
        ADC    8,X
        DEX
        DEC    A
        BNE    ADD01
        CLT                    ;T flag clear
        RTS
```

```
;******************************************************
;
;       Following "0" condense,then modify INDX and data
;
;******************************************************
;
INDX:
        TST    W1+7           ;Test W1 INDX
        BEQ    INDX1
        LDA    W1+6
        AND    #$0F
        BNE    INDX1          ;Valid data remain ?
        LDX    #W1            ;No
        JSR    SFR4           ;Condense to LSB direction
        DEC    W1+7           ;Modify INDX
        BRA    INDX           ;again
INDX1:
        RTS
;******************************************************
;
;       W1 -> W3 ,and  0 -> W1
;
;******************************************************
;
BFITL:
        CLT
        LDX    #W1
BFITL1:
        LDA    0,X
        STA    16,X
        LDA    #0
        STA    0,X
        INX
        CPX    #W1+7
        BNE    BFITL1
        LDM    #0,W3+7        ;W3_INDX initial
        CLD                   ;Decimal mode clear
        SET                   ;T flag set
        RTS
;******************************************************
;
;       WN 4 bits left shift
;
;******************************************************
;
SFL4:
        LDA    #4
SFL41:
        ASL    6,X
        ROL    5,X
        ROL    4,X
        ROL    3,X
        ROL    2,X
        ROL    1,X
        DEC    A
        BNE    SFL41
        RTS
;******************************************************
;
;       WN 4 bits right shift
;
;******************************************************
;
SFR4:
        LDA    #4
SFR41:
        LSR    1,X
        ROR    2,X
        ROR    3,X
        ROR    4,X
        ROR    5,X
        ROR    6,X
        DEC    A
        BNE    SFR41
        RTS
```

```
;******************************************************
;
;       Adjust INDX to W1 to W2
;
;******************************************************
;
ADIX:
        LDA   W1+7
        CMP   W2+7
        BEQ   ADIX1
        BCC   ADIX23
ADIX22:
        LDX   #W2
ADIX2:
        LDA   1,X
        AND   #$F0
        BEQ   ADIX21
        CPX   #W1
        BEQ   ADIX51
        LDX   #W1
ADIX5:
        JSR   SFR4
        DEC   7,X
        BRA   ADIX
ADIX23:
        LDX   #W1
        BRA   ADIX2
ADIX51:
        LDX   #W2
        BRA   ADIX5
ADIX21:
        JSR   SFL4
        INC   7,X
        BRA   ADIX
ADIX1:
        RTS
;******************************************************
;
;       Execute addition,subtraction
;
;******************************************************
;
ADSB:
        SED                 ;Decimal mode set
        CLT                 ;T flag clear
        JSR   ADIX          ;Adjust INDX
ADSB1:
        BBS   3,W2,ADSB6
        BBS   0,W1,ADSB7
        BBS   0,W2,ADSB10
ADSB9:
        JSR   ADD0          ;W2 <- W1+W2
        BCC   ADD2          ;C arise ?
        SEB   0,F16         ;Yes
        JSR   SFR4          ;1 digit right shift
        DEC   W1+7          ;INDX -1
        BMI   ADD3          ;FF ?
        CLC                 ;No
        LDA   W1+1
        ADC   #$10
        STA   W1+1
ADD2:
        CLB   0,F16
        JSR   INDX
ADD3:
        CLD                 ;Decimal mode clear
        RTS
ADSB7:
        BBS   0,W2,ADSB9
ADSB10:
        JSR   SUB0          ;W1 <- W1-W2
        BCS   SUB2
        INX
```

```
SUB3:
        LDA   #$99
        SEC
        SBC   0,X
        STA   0,X           ;Get 99-X
        INX
        CPX   #W1+7
        BNE   SUB3
        DEX
        SET
        LDY   #6            ;C=1
SUB4:
        ADC   #0
        DEX
        DEY
        BNE   SUB4
        CLT                 ;Get 100's complement
        COM   W1            ;Get sign bit in bit0
SUB2:
        CLB   0,F16
        JSR   INDX
        CLD                 ;Decimal mode clear
        RTS
ADSB6:
        BBS   0,W1,ADSB11
        BBS   0,W2,ADSB9
        BRA   ADSB10
ADSB11:
        BBS   0,W2,ADSB10
        BRA   ADSB9
```

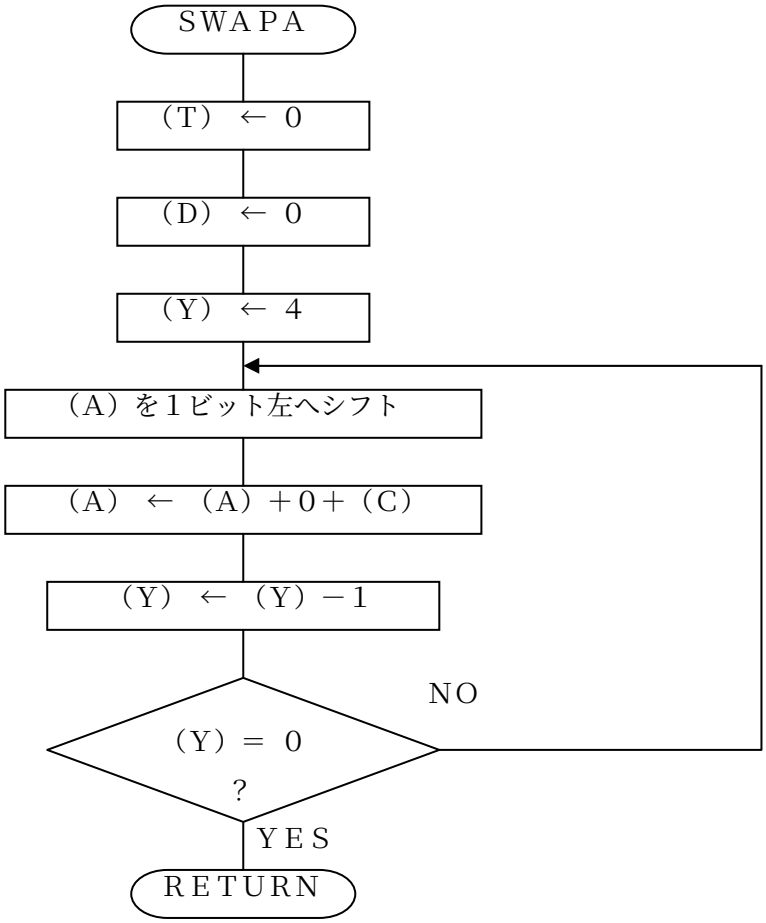
5. 代替命令

5.1 スワップアキュムレータ

(1) 概要

アキュムレータの上位４ビットと下位４ビットを置換します。

(2) フローチャート



(3)　プログラムリスト

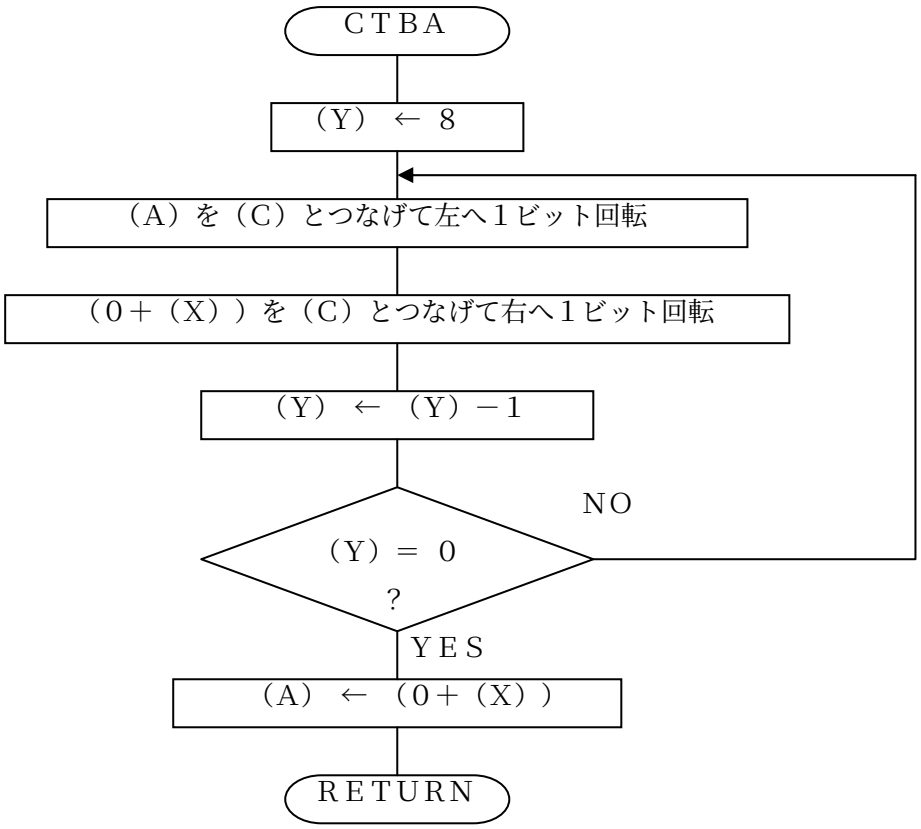
```
;******************************************************
;
;       Swap A register
;
;******************************************************
;
SWAPA:
        CLT                 ;T flag clear
        CLD                 ;Decimal mode clear
        LDY   #4
SWAPA1:
        ASL   A
        ADC   #0
        DEY
        BNE   SWAPA1
        RTS
```

5.2 カウンタビットアキュムレータ

（1） 概要

アキュムレータのビットデータの並びを逆にします。インデックスレジスタXで指定されるワークメモリを１バイト使用します。

（2） フローチャート



（3） プログラムリスト

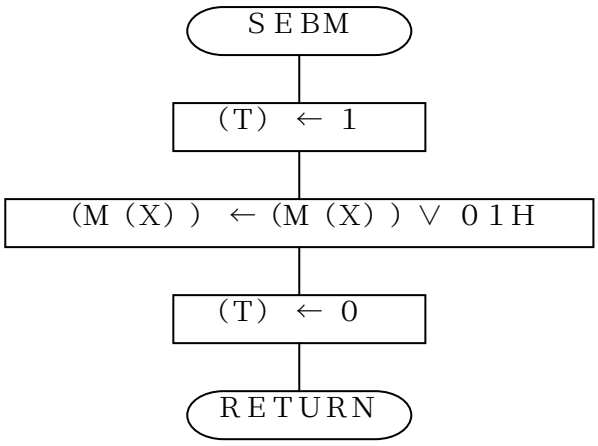
```
;******************************************************
;
;       Counter bit A register
;
;******************************************************
;
CTBA:
        LDY    #8
CTBA1:
        ROL    A
        ROR    0,X
        DEY
        BNE    CTBA1
        LDA    0,X
        RTS
```

5.3 メモリのセットビット

(1) 概要

インデックスレジスタXで指定されるメモリのビット0を”1”にします。

(2) フローチャート



注）M（X）はインデックスレジスタXが示す番地のメモリです。

(3) プログラムリスト

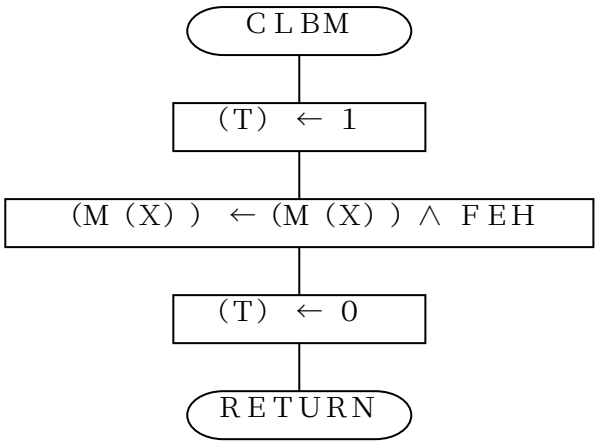
```
;******************************************************
;
;       Set bit 0 of M(X)
;
;******************************************************
;
SEBM:
        SET                 ;T flag set
        ORA   #1            ;Bit 0 set
        CLT                 ;T flag clear
        RTS
```

5.4 メモリのクリアビット

(1) 概要

インデックスレジスタXで指定されるメモリのビット0を”0”にします。

(2) フローチャート



注）M（X）はインデックスレジスタXが示す番地のメモリです。

(3) プログラムリスト

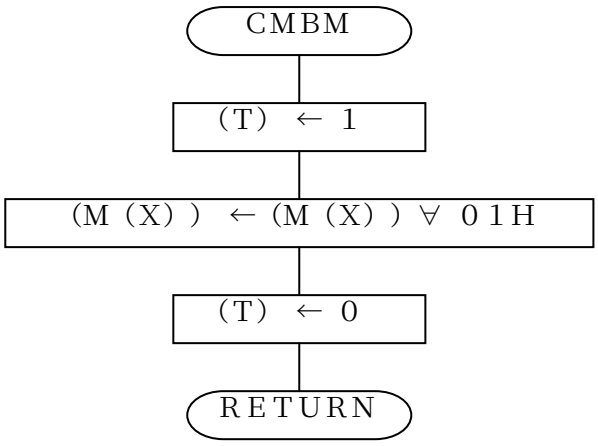
```
;******************************************************
;
;       Clear bit 0 of M(X)
;
;******************************************************
;
CLBM:
        SET                     ;T flag set
        AND   #$FE                ;Bit 0 clear
        CLT                     ;T flag clear
        RTS
```

5.5 メモリのビット反転

(1) 概要

インデックスレジスタXで指定されるメモリのビット0を反転します。

(2) フローチャート



注）M（X）はインデックスレジスタXが示す番地のメモリです。

(3) プログラムリスト

```
;******************************************************
;
;       Complement bit 0 of M(X)
;
;******************************************************
;
CMBM:
        SET                 ;T flag set
        EOR   #1            ;Bit 0 complement
        CLT                 ;T flag clear
        RTS
```

6. プログラム利用上の注意

プログラム利用に際しては、以下の点に注意してください。また、７４０ファミリソフトウエアマニュアルを参考にしてください。

(1) お客様のシステムにおいて十分な評価を行なってください。
(2) 本プログラムは、X修飾演算モードフラグTおよび１０進モードフラグDを、通常 ”０”として扱っています。X修飾演算モードフラグTあるいは１０進モードフラグDを ”１”にして本ルーチンを呼び出さないようにしてください。
(3) １０進モード時（D＝１）にＡＤＣ、ＳＢＣ命令を実行した時は、プロセッサステータスレジスタのネガティブフラグN、オーバフローフラグV、ゼロフラグZは無効となります。また、キャリーフラグCは演算の結果、桁上がりが発生すると ”１”にセット、桁借りが発生すると ”０”にクリアされます。
(4) １０進モード時（D＝１）、ＳＥＣ命令、ＣＬＣ命令、又はＣＬＤ命令は、ＡＤＣ命令又はＳＢＣ命令よりも一命令後に行なってください。

7. 参考ドキュメント

ソフトウェアマニュアル

740 ファミリ ソフトウェアマニュアル

最新版をルネサス テクノロジ ホームページから入手してください。

8. ホームページとサポート窓口

ルネサス テクノロジ ホームページ

http://www.renesas.com/jpn/products/mpumcu/index.html

ルネサス製品全般に関するお問合せ先

カスタマ・サポート・センター：csc@renesas.com

アプリケーションノートに関する技術的なお問合せ先

740 ファミリ MCU 技術サポート窓口：support_apl@renesas.com

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2000.11.07	—	応用技術資料として発行
1.01	2005.03.18	—	アプリケーションノートに様式変更して発行

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。