TOSHIBA

第 2 章　TLCS-870/C シリーズ　CPU

株式会社 東芝 セミコンダクター社

CMOS 8 **ビット マイクロコンピュータ**

TLCS-870/C **シリーズ**

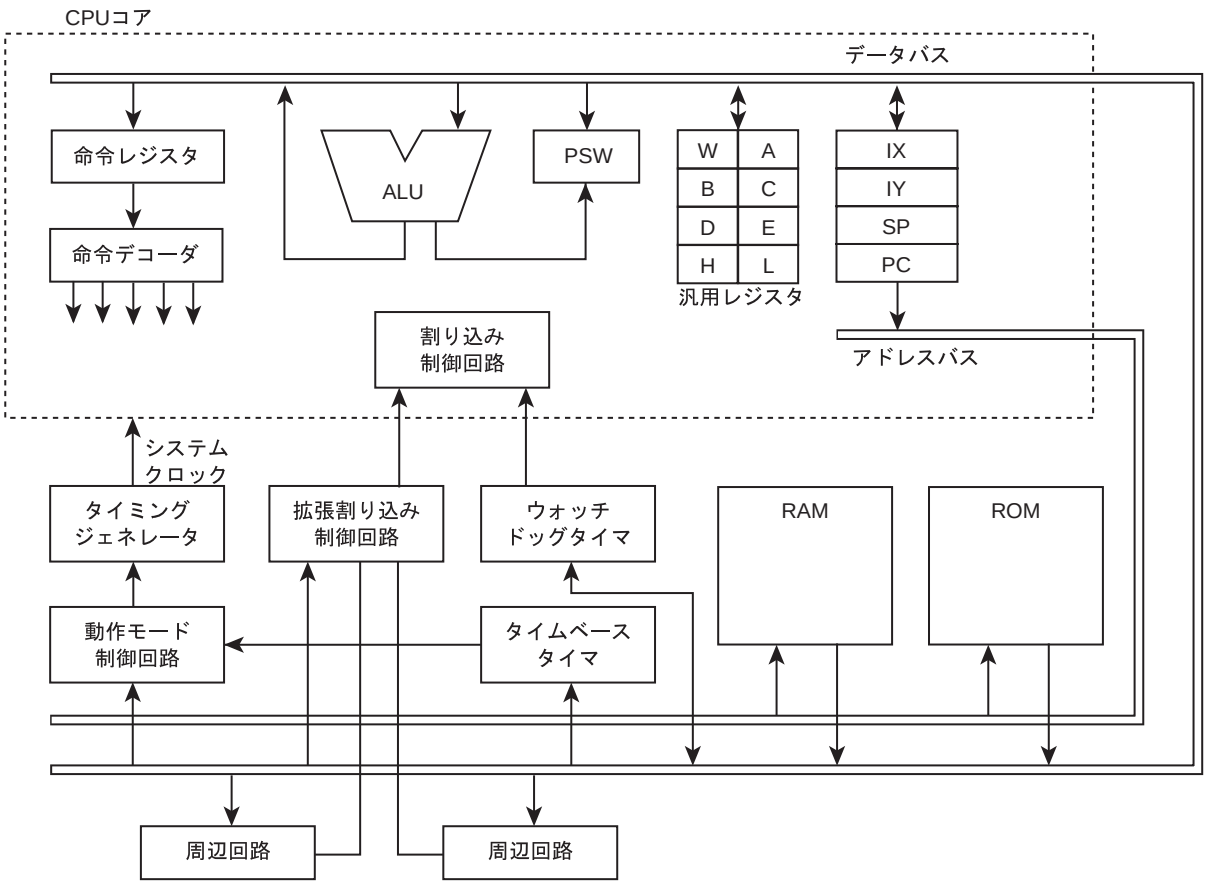
TLCS-870/C シリーズは、東芝オリジナルのコンパクトで高速 / 高機能 , 低消費電力の 8 ビットシングルチップマイクロコンピュータです。

特長

- 直交性のある豊富な命令セット : 132 種 731 命令
  TLCS-870/X を基に、小規模システム向けに最適化した命令セットで、より C コンパイラに向いています。
  - 乗除算 (8 ビット × 8 ビット , 16 ビット ÷ 8 ビット)
  - ビット操作 (Set/Clear/Complement/Load/Store/Test/Exclusive OR)
  - 16 ビット演算 / 転送
  - 1 バイト長のジャンプ / コール (Short relative jump/Vector call)
- レジスタ構成 (メモリ空間から独立)
  - 汎用レジスタ : 8 ビット幅 8 本 (16 ビット幅 4 本としても使用可)
  - 16 ビット汎用レジスタ : 16 ビット幅 2 本
  - プログラムカウンタ : 16 ビット幅 1 本
  - スタックポインタ : 16 ビット幅 1 本
  - プログラムステータスワード : 6 ビット幅 1 本
- メモリ空間
  プログラムおよびデータメモリ空間 : 64 K バイト
- メモリマップ I/O 方式
- 割り込み
  - 最大 32 要因まで対応可 (除くリセット)
  - エッジ選択、ノイズ除去機能付き外部割り込み
- 低消費電力動作モード (最大 9 種のモード)
  - STOP モード : 動作停止 (バッテリー / コンデンサバックアップ)
  - SLOW1 モード : 低周波クロックによる低周波動作 (高周波停止)
  - SLOW2 モード : 低周波クロックによる低周波動作 (高周波発振)
  - IDLE0 モード : CPU 停止、周辺ハードウエアのうち、タイムベースタイマのみ動作 (高周波クロック) 継続し、タイムベースタイマ設定の基準時間経過により解除。
  - IDLE1 モード : CPU 停止、周辺ハードウエアのみ動作 (高周波クロック)、割り込みにより解除 (CPU 再起動)。
  - IDLE2 モード : CPU 停止、周辺ハードウエアのみ動作 (高周波 / 低周波クロック)、割り込みにより解除 (CPU 再起動)。
  - SLEEP0 モード : CPU 停止、周辺ハードウエアのうち、タイムベースタイマ (TBT) のみ動作 (低周波クロック) 継続し、タイムベースタイマ設定の基準時間経過により解除。
  - SLEEP1 モード : CPU 停止、周辺ハードウエアのみ動作 (低周波クロック)、割り込みにより解除。
  - SLEEP2 モード : CPU 停止、周辺ハードウエアのみ動作 (高周波 / 低周波クロック)、割り込みにより解除。
- クロックギア (逓倍・分周)
- 高速処理
  - 最小命令実行時間 : 250 ns @16 MHz

- 低電圧 / 高速動作 (一部製品で適用)
  - 最低電源電圧 : 1.8 V/4.2 MHz 動作可能
- 外部メモリアクセス対応 (一部製品で適用)
- フェイルセーフ機能
  - ウォッチドッグタイマ (リセット or 割り込み)
  - アドレストラップ (リセット or 割り込み)
  - システムクロックリセット
  - 未定義命令割り込み
  - ソフトウエア割り込み

ブロック図



注) PSW: プログラムステータスワード

1. 動作説明

1.1 CPU コア機能

CPU コアは、CPU, システムクロック制御回路および割り込み制御回路から構成されています。

本章では、CPU コア , プログラムメモリおよびデータメモリについて説明します。

1.1.1 メモリアドレスマップ

TLCS-870/C シリーズのメモリは、ROM, RAM, SFR (スペシャルファンクションレジスタ), DBR (データバッファレジスタ), MEM1, MEM2 の 6 つのブロックで構成され、それらは 1 つの 64 K バイトアドレス空間上にマッピングされています。図 1-1 に TLCS-870/C のメモリアドレスマップの例を示します。また、汎用レジスタはアドレス空間にはマッピングされていません。実際の割り付けは、個別製品のデータシートを参照してください。

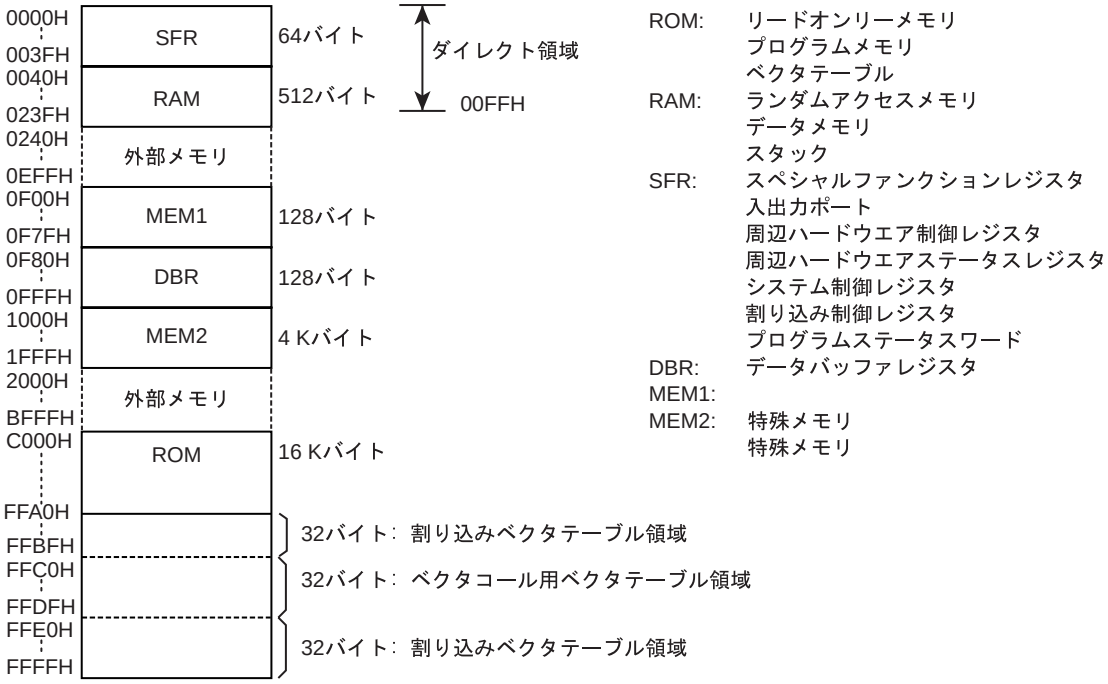


図 1-1 メモリアドレスマップ (例)

なお、アドレス 0000H~00FFH はダイレクト領域となっています。この領域に対しては、実行時間を短縮した命令による処理が可能です。

1.1.2 プログラムメモリ (ROM)

プログラムメモリの FFC0H~FFFFH 番地は、特定の用途にも使用されます。

1.1.2.1 割り込みベクタテーブル (FFA0H~FFBFH 番地 , FFE0H~FFFFH 番地)

リセットおよび割り込みのベクタ (2 バイト / ベクタ) を格納するテーブルで、16 ベクタあります。ベクタには、リセット解除からのスタートアドレス , 割り込みサービスルーチンのエントリーアドレスを格納します。

1.1.2.2 ベクタコール命令用ベクタテーブル (FFC0H~FFDFH 番地)

ベクタコール命令 [CALLV n] 用のベクタ (サブルーチン エントリーアドレス , 2 バイト / ベクタ) を格納するテーブルで、16 ベクタあります。通常コール命令 [CALL mn] は、一つの命令に 3 バイト長の ROM 容量が必要ですが、ベクタコール命令 [CALLV n] は 1 バイト長の命令で、使用頻度の高いサブルーチンコールに使うことによりメモリ効率を上げることができます。

	CALLV 命令	CALL 命令	
1 箇所から CALL	1 + 2 = 3 バイト	3 バイト	→ CALLV と CALL の効率同じ
2 箇所から CALL	2 + 2 = 4 バイト	6 バイト	→ CALLV の方が効率良い
3 箇所から CALL	3 + 2 = 5 バイト	9 バイト	
4 箇所から CALL	4 + 2 = 6 バイト	12 バイト	

*CALLV の +2 はベクタコール領域に配置されたジャンプ先アドレスの 2 バイト

（ プログラム例 ）ベクタコール

CALLV で呼ばれる定数を 2 倍して FFC0H を加えた値をアドレスと
するメモリから呼び出した 16 ビットアドレスにジャンプします。

```
                CALLV   02H
                ..
                ..
PSUB _ 2:
                LD      (HL), A
                DEC     A
                ..
                ..
;
VECTOR_CALL             SECTION CODE ABS = 0FFC0H
                DW      PSUB _ 0        ; 00H
                DW      PSUB _ 1        ; 01H
                DW      PSUB _ 2        ; 02H
                ..                                  最大 16 個
                ..
                DW      PSUB _ F        ; 0FH
;
```

プログラムメモリには、プログラムおよび固定データが格納されます。次に実行すべき命令は、プログラムカウンタの内容が示すアドレスから読み出されます。ジャンプ命令は相対ジャンプまたは絶対ジャンプ命令で、ジャンプ命令に関してプログラムメモリにはページ , バンクといった境界概念はありません。

例 : ジャンプ命令とプログラムカウンタの関係

1. 5 ビット相対ジャンプ命令 [JRS cc, $ + 2 + d]
   E8C4H: JRS T, $ + 2 + 08H の場合
   JF = 1 のとき、プログラムカウンタの内容に 08H を加算した E8CEH にジャンプします (プログラムカウンタの内容は実行命令の置かれたアドレス +2 になっています。従って、この場合のプログラムカウンタの値は E8C4H + 2 = E8C6H となります)。
2. 8 ビット相対ジャンプ命令 [JR cc, $ + 2 + d]/[JR cc, $ + 3 + d]
   E8C4H: JR Z, $ + 2 + 80H の場合
   ZF = 1 のとき、プログラムカウンタの内容に FF80H (–128) を加算した E846H にジャンプします。
3. 16 ビット絶対ジャンプ命令 [JP a]
   E8C4H: JP 0C235H の場合
   無条件に C235H 番地にジャンプします。絶対ジャンプ命令は 64 K バイトの全空間内の任意のアドレスにジャンプできます。

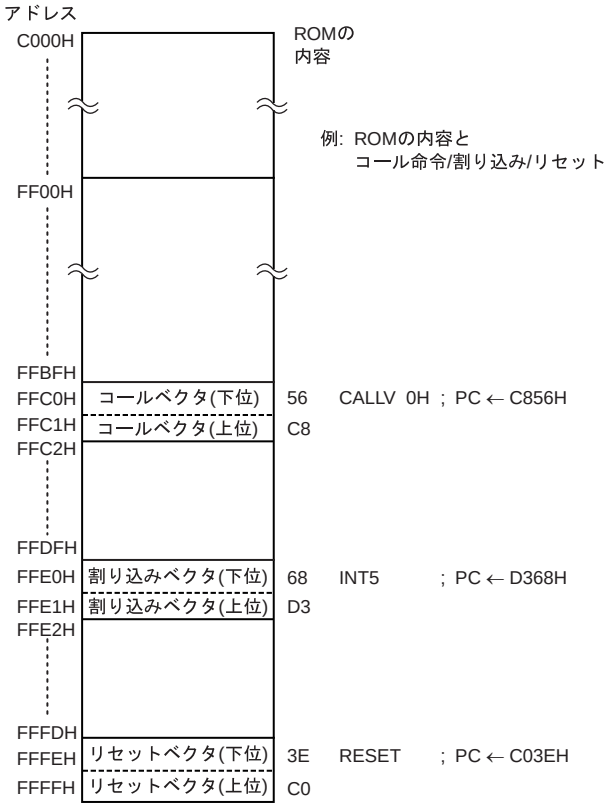


図 1-2 プログラムメモリマップ (例)

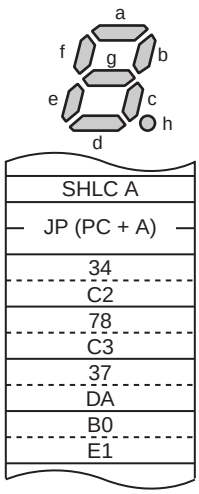
TLCS-870/C シリーズは、プログラムメモリに格納された固定データの読み出しに、データメモリをアクセスする命令と同じ命令を使用します。さらに、レジスタオフセット相対アドレッシングモード (PC + A) の命令も使用でき、コード変換 , テーブルルックアップ , 多方向分岐処理などが容易にプログラミングできます。

(プログラム例) HL レジスタペアで指定されるアドレスの ROM 内容をアキュムレータに読み出す処理。

```
LD          A, (HL)          ; A ← ROM (HL)
```

（プログラム例）テーブルルックアップ処理 (アセンブラ)

```
TABLE1:
        DB      3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 6FH
SUB1:
        LD      HL, TABLE1      ; 7seg テーブルの先頭アドレスをロード
        LD      C, 07H          ; 表示させたい 7seg データ ( この場合 "7") をロード
        LD      A, (HL + C)     ; A ← 7seg ポート出力データ
                (P3), A         ; Port3 ← A
```



注） $ **は** ADD **命令の先頭アドレス。**DB **はバイトデータの定義命令。**

（プログラム例）アキュムレータの内容 (0 ≦ A ≦ 3) による多方向分岐処理

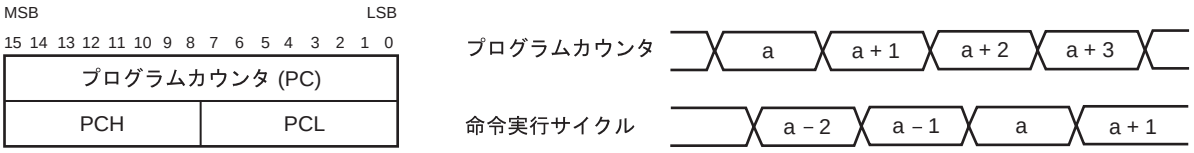
```
SHLC    A           ; if A = 00H then PC ← C234H
JP      (PC + A)      if A = 01H then PC ← C378H
                      if A = 02H then PC ← DA37H
                      if A = 03H then PC ← E1B0H
DW      0C234H, 0C378H, 0DA37H, 0E1B0H
```

注） DW **はワードデータの定義命令。ワード** = 2 **バイト。**

1.1.3 プログラムカウンタ (PC)

プログラムカウンタは、次に実行すべき命令の格納されているプログラムメモリのアドレスを指す 16 ビットのレジスタです。リセット解除時、ベクタテーブル (FFFFH, FFFEH 番地) に格納されているリセット ベクタがプログラムカウンタにロードされますので、任意のアドレスからプログラムの実行を開始することができます。例えば、FFFFH, FFFEH 番地にそれぞれ、C0H, 3EH が格納されている場合、リセット解除後 C03EH 番地から実行開始します。

TLCS-870/C シリーズは、パイプライン処理 (命令先行フェッチ) を行っていますので、プログラムカウンタは、常に 2 アドレス先を指します。例えば、C123H 番地に格納されている 1 バイト命令の実行中、プログラムカウンタの内容は、C125H です。



(a) 構成　　(b) プログラムカウンタと命令実行サイクル

図 1-3 プログラムカウンタ

1.1.4 データメモリ (RAM)

データメモリには内部データメモリ (内蔵ROM または内蔵RAM) および外部データメモリ (RAM または ROM) があります。

データメモリの内容は、電源投入時 不定になりますので、初期設定で RAM クリアならびに RAM の初期値設定を行ってください。

（ プログラム例 ）RAM クリア (データメモリが 4 K バイト内蔵されている製品の例)

```
        LD      HL, 0040H       ; スタートアドレスの設定
        LD      A, H            ; 初期化データ (00H) の設定
        LD      BC, 0FFFH       ; バイト数の設定 (RAM 容量 – 1)
SRAMCLR: LD     (HL), A
        INC     HL
        DEC     BC
        JRS     F, SRAMCLR
```

1.1.5 汎用レジスタ

TLCS-870/C は、汎用レジスタとして、W, A, B, C, D, E, H, L の 8 つの 8 ビットレジスタを内蔵しています。各レジスタは、WA, BC, DE, HL のペアで、16 ビットレジスタとしても使用できます。

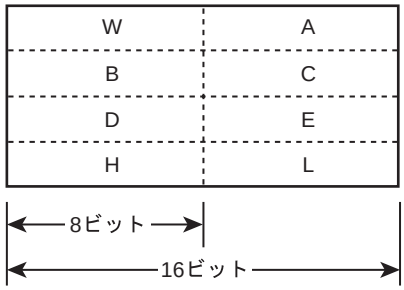


図 1-4 汎用レジスタ

W, A, B, C, D, E, H, L は 8 ビットの転送・演算に、WA, BC, DE, HL は 16 ビットの転送・演算に用いられます。また、汎用レジスタとしての機能のほかに、次の機能を有しています。

1.1.5.1 A

A レジスタは、レジスタ間接ビット指定のビット操作命令における、ビット指定レジスタとして使用されます。

(プログラム例)

```
①   SET   (56H).A        ; 0056H 番地の、A の内容で指定されるビットを、
                           "1" にセットします。
②   CPL   (IX + 03H).A   ; IX に即値 03H を符号拡張加算した値で指定される
                           アドレスのメモリの、A の内容で指定されるビットを
                           反転します。
```

また、レジスタオフセット相対アドレッシング (PC + A) のオフセットレジスタとしても使用されます。

(プログラム例)

```
    LD    A, (PC + A)    ; PC に A の内容を符号拡張加算した値で指定される
                         アドレスのメモリの内容を A にロードします。
```

1.1.5.2 C

C レジスタは、除算命令における除数レジスタとして使用されます。除算命令実行後に被除数の上位バイトに余りが、被除数の下位バイトに商がそれぞれ格納されます。

(プログラム例)

```
    DIV   WA, C          ; A ← WA ÷ C,  W ← 余り
```

また、レジスタオフセット相対アドレッシング (HL + C) オフセットレジスタとしても使用されます。

(プログラム例)

```
    LD    A, (HL + C)    ; HL に C の内容を符号拡張加算した値で指定される
                         アドレスのメモリの内容を A にロードします。
```

1.1.5.3 DE

DE レジスタは、レジスタ間接アドレッシングのアドレス指定レジスタとして使用されます。

(プログラム例)

```
    LD    A, (DE)        ; DE で指定されるアドレスのメモリの内容を A に
                         ロードします。
```

1.1.5.4 HL

HL レジスタは、レジスタ間接アドレッシングのアドレス指定レジスタ, インデックスアドレッシングのインデックスレジスタとして使用されます。

（プログラム例）

```
①   LD   A, (HL)        ; HL で指定されるアドレスのメモリの内容を
                          A にロードします。
②   LD   A, (HL + 52H)  ; HL に即値 52H を符号拡張加算した値で指定される
                          アドレスのメモリの内容を A にロードします。
③   LD   A, (HL + C)    ; HL に C の内容を符号拡張加算した値で指定される
                          アドレスのメモリの内容を A にロードします。
```

汎用レジスタは、リセットで初期化されません。電源投入時の値は不定となります。

1.1.6 16 ビット汎用レジスタ (IX, IY)

TLCS-870/C は、2 つの 16 ビット汎用レジスタ IX, IY を内蔵しています。これらは 16 ビット長のレジスタでレジスタ間接アドレッシングアドレス指定レジスタ , インデックスアドレッシングのインデックスレジスタとして使用されます。

（プログラム例）

```
①   LD   A, (IX)        ; IX で指定されるアドレスのメモリの内容を A に
                          ロードします。
②   LD   A, (IY + 52H)  ; IY に即値 52H を符号拡張加算した値で指定される
                          アドレスのメモリの内容を A にロードします。
```

また、16 ビット長の汎用レジスタとして、転送・演算に使用することもできます。

（プログラム例）

```
①   LD   IX, (3AH)      ; IX に 003AH 番地の内容をロードします。
②   ADD  IX, 5678H      ; IX に即値 5678H を加算し、結果を IX に格納します。
```

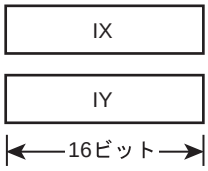


図 1-5 16 ビット汎用レジスタ

16 ビット汎用レジスタは、リセットで初期化されません。電源投入時の値は不定となります。

1.1.7 プログラムステータスワード (PSW)

プログラムステータスワードは、

- ジャンプステータスフラグ (JF)
- ゼロフラグ (ZF)
- キャリーフラグ (CF)
- ハーフキャリーフラグ (HF)
- サインフラグ (SF)

• オーバフローフラグ (VF)

の 6 つのフラグから構成され、SFR 内の 003FH 番地に割り付けられています。

PSW のアクセスは、専用命令で行います。また、メモリアクセス命令で読み出すこともできます。

（プログラム例）

```
①    LD      PSW, 00H        ; PSW に 00H を書き込む
②    PUSH    PSW             ; PSW の内容をスタックに退避
③    POP     PSW             ; スタックの内容を PSW にリストア
④    LD      A, (3FH)        ; A レジスタに PSW の内容を転送
```

PSW に対し、メモリアクセス命令での書き込みを行った場合、データは書き込まれず、その命令で定まった変化をします。

（プログラム例）

```
     LD      (3FH), 00H      ; PSW = 00H にはならず、[LD (x), n] 命令の指定に従い
                             JF が "1" になり、ほかのフラグは変化しません。
```

割り込み受け付け時、PSW はプログラムカウンタとともにスタックに退避されます。また、割り込みリターン命令 [RETI], [RETN] 命令の実行によりスタック上のデータが PSW にリストアされます。

PSW の各種フラグは、リセットで初期化されません。電源投入時の値は不定となります。

PSW の構成

PSW (003FH)	7	6	5	4	3	2	1	0
	JF	ZF	CF	HF	SF	VF	–	–

フラグは、命令で指定される条件に従いセット / クリアされます。また、ハーフキャリーフラグ以外のフラグは条件付きジャンプ命令 [JR cc, a], [JRS cc, a] のジャンプ条件 cc となり得ます。

cc	意味	条件
T	True	JF = 1
F	False	JF = 0
Z	Zero	ZF = 1
NZ	Not zero	ZF = 0
CS	Carry set	CF = 1
CC	Carry clear	CF = 0
VS	Overflow set	VF = 1
VC	Overflow clear	VF = 0
M	Minus	SF = 1
P	Plus	SF = 0
EQ	Equal	ZF = 1
NE	Not equal	ZF = 0
LT	Unsigned less than	CF = 1
GE	Unsigned greater than or equal	CF = 0
LE	Unsigned less than or equal	(CF ∨ ZF) = 1

cc	意味	条件
GT	Unsigned greater than	(CF-∨-ZF) = 0
SLT	Signed less than	(SF-∨-VF) = 1
SGE	Signed greater than or equal	(SF-∨-VF) = 0
SLE	Signed less than or equal	ZF ∨ (SF-∨-VF) = 1
SGT	Signed greater than	ZF ∨ (SF-∨-VF) = 0

1.1.7.1 ゼロフラグ (ZF)

演算結果または転送データが 00H (8 ビット演算 / 転送時)/0000H (16 ビット演算時) のとき “1” にセットされ、その他のときは “0” にクリアされます。ビット操作命令では、指定ビットの内容が “0” のとき “1” にセットされ、指定ビットの内容が “1” のとき “0” にクリアされます (ビットテスト)。乗算命令の場合、積の上位 8 ビットが 00H のとき、除算命令の場合、剰余が 00H のとき、“1” にセットされ、その他のときは “0” にクリアされます。

1.1.7.2 キャリーフラグ (CF)

演算時のキャリーまたはボローがセットされます。除算命令の場合、除数が 00H のとき (Divided by zero error)、または商が 100H 以上のとき (Quotient-overflow error)、“1” にセットされます。シフト / ローテート命令では、レジスタからシフトアウトされるデータがセットされます。

ビット操作命令では、1 ビット長のレジスタ (ブーリアンアキュムレータ) として機能します。また、キャリーフラグ操作命令によりセット / クリア / 反転ができます。

（ プログラム例 ）ビット操作 (0007H 番地のビット 5 の内容と 009AH 番地のビット 0 の内容とで排他的論理和をとり、結果を 0001H 番地のビット 2 に書き込みます)。

```
LD      CF, (07H).5     ; (0001H)2 ← (0007H)5 ∨ (009AH)0
XOR     CF, (9AH).0
LD      (01H).2, CF
```

1.1.7.3 ハーフキャリーフラグ (HF)

8 ビット演算時、4 ビット目へのキャリーまたは 4 ビット目からのボローがセットされます。HF は、BCD データの加減算の際の十進補正用のフラグです ([DAA r], [DAS r] 命令による十進補正)。

（ プログラム例 ）BCD 演算 (A = 19H, B = 28H のとき、次の命令を実行すると、A は 47H になります)。

```
ADD     A, B            ; A ← 41H, HF ← 1, CF ← 0
DAA     A               ; A ← 41H + 06H = 47H ( 十進補正 )
```

1.1.7.4 サインフラグ (SF)

算術演算の演算結果の MSB が “1” のとき、“1” にセットされ、その他のときは “0” にクリアされます。

1.1.7.5 オーバフローフラグ (VF)

算術演算の演算結果にオーバフローが生じたときに “1” にセットされ、その他のときは “0” にクリアされます。例えば、加算命令で 2 つの正の数を加算した結果が負の数になった場合、あるいは 2 つの負の数を加算した結果が正の数になった場合に、VF は “1” にセットされます。

1.1.7.6 ジャンプステータスフラグ (JF)

通常、“1” にセットされるフラグで、命令に従いゼロまたはキャリー情報がセットされ、条件付きジャンプ命令 [JR T/F, a], [JRS T/F, a] (T, F は条件コード) のジャンプ条件となります。

（プログラム例）ジャンプステータスフラグと条件付きジャンプ命令

```
INC     A
JRS     T, SLABEL1      ; 直前の演算命令で桁上げが発生した場合ジャンプ
                          します。
        :
LD      A, (HL)
JRS     T, SLABEL2      ; 直前の命令で JF は “1” にセットされますので、
        :                 無条件ジャンプと見なされます。
```

例 : WA レジスタ , HL レジスタ , データメモリの 00C5H 番地 , CF, HF, SF, VF の内容がそれぞれ “219AH”, “00C5H”, “D7H”, “1”, “0”, “1”, “0” のとき、下記命令を実行すると、A, WA レジスタおよび各フラグは次のようになります。

命令	実行後のアキュムレータ	実行フラグ					
		JF	ZF	CF	HF	SF	VF
ADDC A, (HL)	72	1	0	1	1	0	1
SUBB A, (HL)	C2	1	0	1	0	1	0
CMP A, (HL)	9A	0	0	1	0	1	0
AND A, (HL)	92	0	0	1	0	1	0
LD A, (HL)	D7	1	0	1	0	1	0
ADD A, 66H	00	1	1	1	1	0	0
INC A	9B	0	0	1	0	1	0
ROLC A	35	1	0	1	0	1	0
RORC A	CD	0	0	0	0	1	0
ADD WA, F508H	16A2	1	0	1	0	0	0
MUL WA	13DA	0	0	1	0	1	0
SET A.5	BA	1	1	1	0	1	0

1.1.8 スタック、スタックポインタ

1.1.8.1 スタック

スタックは、サブルーチンコール命令実行時または割り込み受け付け時に、その処理ルーチンへジャンプするに先立ってプログラムカウンタの内容 (戻り番地) やプログラムステータスワードの内容などをセーブするエリアです。

サブルーチンコール命令 [CALL mn], [CALLV n] 実行時、戻り番地が (上位バイト、下位バイトの順に) スタックに退避 (プッシュダウン) されます。ソフトウエア割り込み命令 [SWI] 実行時および割り込み受け付け時は、まずプログラムステータスワードの内容がスタックに退避され、次に戻り番地が退避されます。

処理ルーチンから復帰する場合、サブルーチンリターン命令 [RET] を実行することによりスタックからプログラムカウンタへ、割り込みリターン命令 [RETI], [RETN] を実行することによりスタックからプログラムカウンタおよびプログラムステータスワードへ、それぞれの内容がリストア (ポップアップ) されます。

スタックは、データメモリ内の任意のエリアに設定できます。

1.1.8.2 スタックポインタ

スタックポインタは、スタックの先頭番地を指す 16 ビットのレジスタです。スタックポインタは、サブルーチンコール、プッシュ命令実行時、および割り込み受け付け時にポストデクリメントされ、リターン、ポップ命令実行時にプリインクリメントされます。従って、スタックはアドレスの若い方に向かって深くなります。

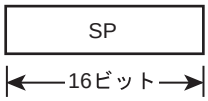


図 1-6 スタックポインタ

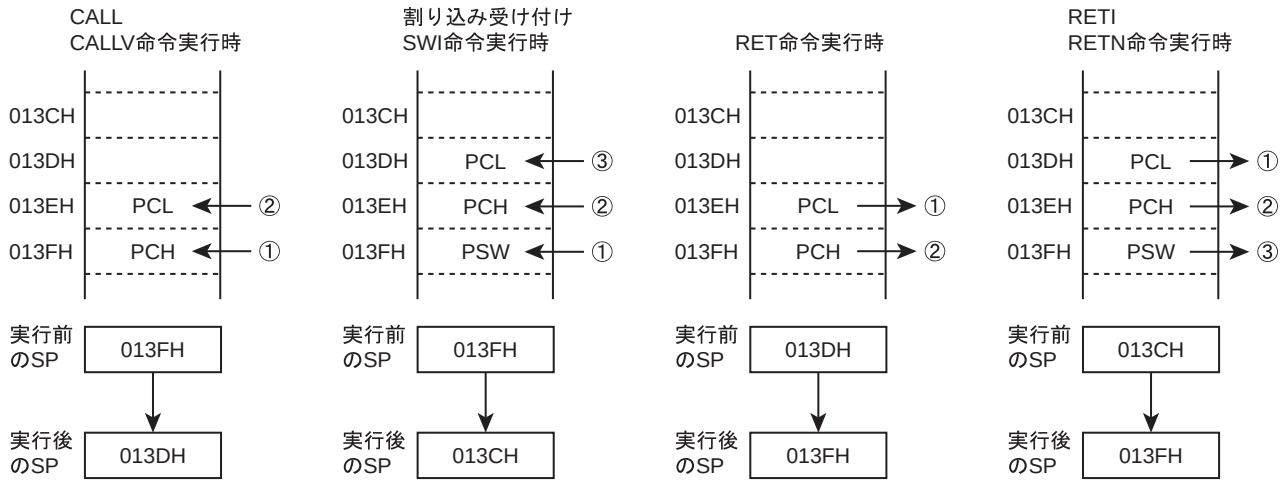
スタックのアクセスとスタックのポインタの変化を図 1-7 に示します。

スタックポインタは、ハードウエアリセットで初期化されません。電源投入時の値は不定となります。従って、プログラムで初期化 (スタックの最高位アドレスをセット) する必要があります。

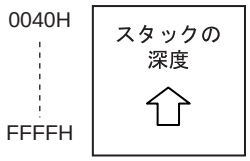
スタックポインタは、インデックスレジスタと同様に転送・演算命令で操作できるほか、インデックスアドレッシングのインデックスレジスタとして使用できます。

(プログラム例)

```
①   LD    SP, 043FH      ; SP ← 043FH
②   LD    HL, SP         ; HL ← SP
③   LD    SP, SP+04H     ; SP ← SP + 04H
④   ADD   SP, 5678H      ; SP ← SP + 5678H
⑤   LD    A, (SP+12H)    ; A ← アドレス (SP + 12H) のメモリの内容
```



(a) スタックのアクセス例 (プッシュ/ポップ)



(b) スタックの深度

図 1-7 スタック