
第 2 章 CPU

ハイライト

本章では次のトピックについて説明します。

2.1	はじめに	2-2
2.2	プログラマ用モデル	2-4
2.3	ソフトウェア スタックポインタ	2-7
2.4	CPU レジスタ説明	2-10
2.5	算術演算論理ユニット (ALU).....	2-13
2.6	乗算と除算のサポート	2-14
2.7	コンパイラとの親和性を持つアーキテクチャ	2-17
2.8	複数ビット シフトのサポート	2-17
2.9	命令フローの種類	2-18
2.10	プログラム フロー ループ制御	2-20
2.11	アドレス レジスタ 従属関係	2-22
2.12	レジスタ マップ	2-25
2.13	関連するアプリケーション ノート	2-26
2.14	改版履歴	2-27

2.1 はじめに

PIC24F CPU モジュールは、強化版命令セットを持った 16 ビット (データ) の改良ハーバードアーキテクチャとなっています。CPU は、オペコード部が可変長の 24 ビット命令ワードを持っています。プログラムカウンタ (PC) は 24 ビット幅で、最大 4M x 24 ビットまでのユーザープログラム空間をアドレスできます。単一サイクルの命令フェッチメカニズムにより、スループットを維持し、予測実行を可能としています。プログラムフローを変える命令や、ダブルワード移動命令 (MOV.D)、さらにテーブル命令を除くすべての命令が単一サイクルで実行されます。REPEAT 命令によりオーバーヘッドのないループ構造がサポートされ、この間のどこでも割り込みが可能です。

PIC24F デバイスはプログラマ用モデルとして、16 個の 16 ビット作業レジスタを持っています。各作業レジスタは、データ、アドレス、アドレスオフセット用レジスタとして機能します。16 番目の作業レジスタ (W15) は、割り込みや CALL の際のソフトウェアスタックポインタとして動作します。15 番目の作業レジスタ (W14) は、スタックフレームポインタとして LNK または UNLK 命令により使用されます。

データメモリ空間の上位 32k バイトは、オプションで、8 ビットのプログラム空間可視化ページレジスタ (PSVPAG) で定義された任意の 16k ワード境界のプログラム空間にマッピングすることができます。データをプログラム空間にマッピングできる特徴は、どの命令でもプログラム空間をデータ空間のようにアクセスできるようにします。プログラムスペース可視化についての詳細は、第 4.4 項「データ空間からのプログラム空間可視化」を参照して下さい。

命令セットのアーキテクチャ (ISA) は、PIC18F のものよりかなり強化されていますが、受け入れられるレベルの下位互換性を維持しています。すべての PIC18F の命令とアドレッシングモードが、直接あるいは簡単なマクロのいずれかでサポートされています。多くの ISA 強化はコンパイラの効率化に必要なために行われています。

コアは、内在型 (オペランドなし)、相対、リテラル、メモリ直接アドレッシングモードをサポートし、さらに 3 グループのアドレッシングモード (MODE1、MODE2、MODE3) をサポートしています。全モードがレジスタ直接と各種のレジスタ間接アドレッシングモードをサポートしています。各グループごとに 7 種類までのアドレッシングモードを提供します。命令はそれぞれの機能に必要なアドレッシングモードを持っています。

さらに、LDWLO と STWLO という 2 つの特別な移動命令には、符号付き 10 ビットオフセットを持つレジスタ間接アドレッシングモードも用意されています。詳しくは第 32 章「命令セット」を参照して下さい。

大部分の命令に対し、コアはデータ (またはプログラムデータ) 読み出し、作業レジスタ (データ) 読み出し、データメモリ書き込み、そしてプログラム (命令) メモリ読み出しを 1 命令サイクルで可能としています。その結果、3 つの命令パラメータをサポートすることができるので、 $A + B = C$ という演算を 1 サイクルで実行できます。

高速 17 ビット × 17 ビット乗算器が内蔵されていて、コアの算術演算能力とスループットを大幅に強化しています。この乗算器は、符号付き、符号なしおよび混在モードの 16 ビット × 16 ビットまたは 8 ビット × 8 ビットの整数の乗算をサポートしています。全乗算命令が 1 サイクルで実行されます。

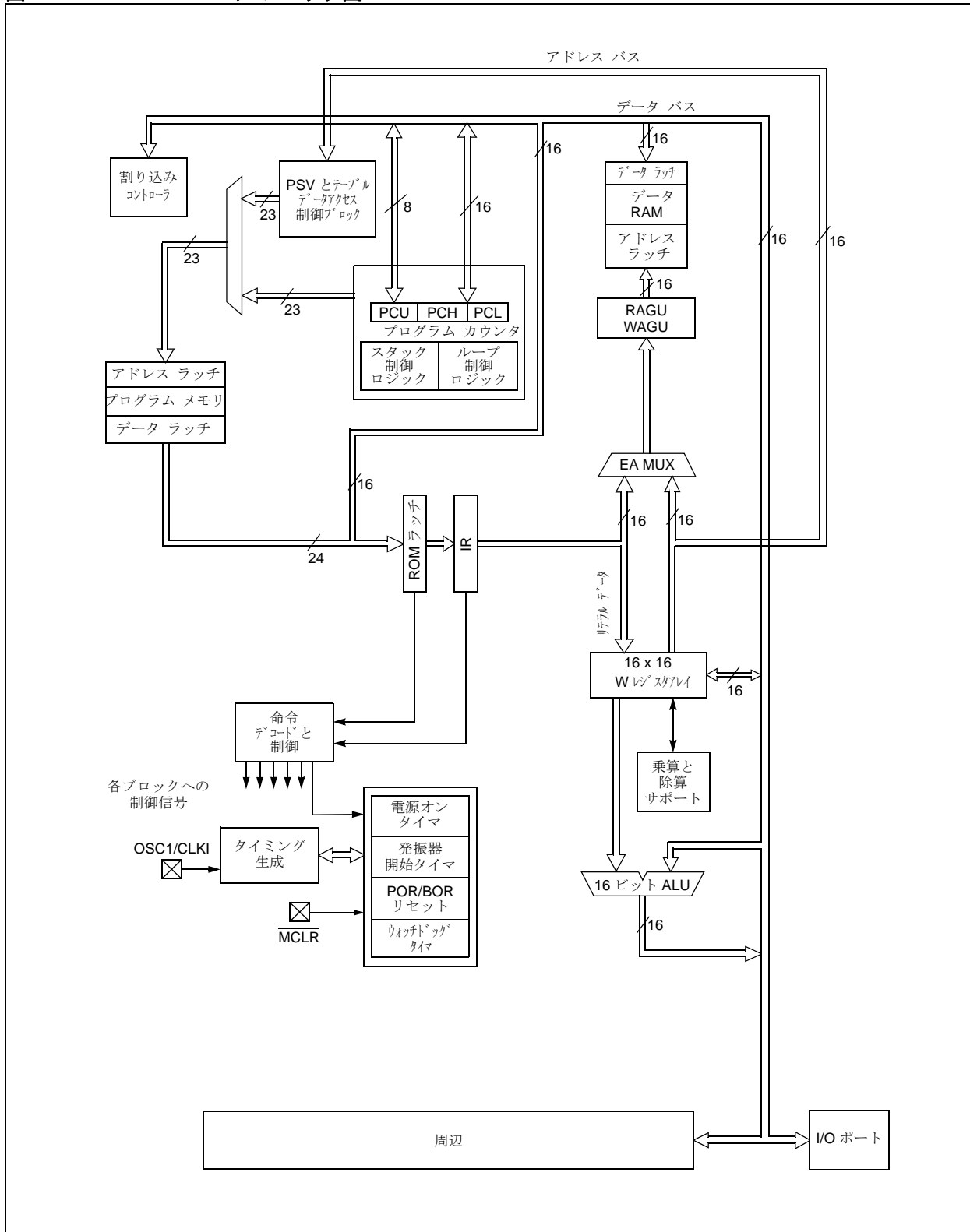
16 ビットの ALU は、格納不要の反復除算アルゴリズムをサポートする整数除算支援ハードウェアで強化されています。これと REPEAT 命令のループメカニズムとの協働と、反復除算命令の選択により、32 ビット (または 16 ビット) ÷ 16 ビットの符号付き、符号なしの整数除算を実行します。全除算演算が 19 サイクルで完了し、どのサイクル境界でも割り込み可能です。

PIC24F は、ベクタ方式の例外機構を持っていて、最大 8 種までのマスク不可のトラップと割り込みをサポートしています。どの割り込みも 7 レベルの優先順位に割付けできます。

CPU のブロック図を図 2-1 に示します。

PIC24F ファミリ リファレンス マニュアル

図 2-1: PIC24F CPU コア ブロック図



2.2 プログラマ用モデル

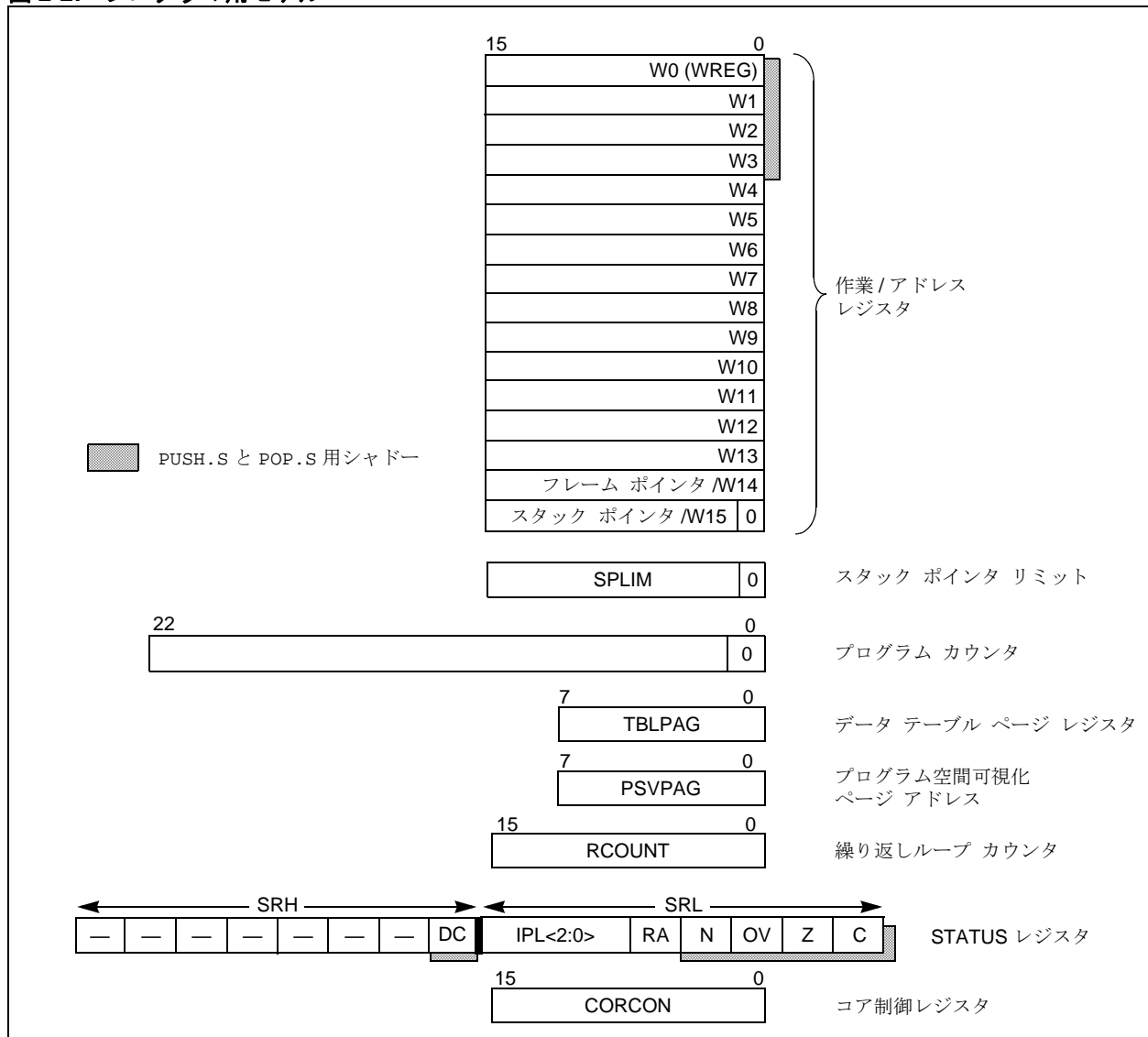
PIC24F のプログラマ用モデルを図 2-2 に示します。プログラマ用モデルの全レジスタはメモリにマップされていて、命令で直接扱うことができます。各レジスタの説明を表 2-1 に示します。

表 2-1: プログラマ用モデルのレジスタ説明

レジスタ名	説明
W0 ~ W15	作業レジスタアレイ
PC	23 ビットプログラムカウンタ
SR	ALU ステータス レジスタ
SPLIM	スタック ポインタ リミット値レジスタ
TBLPAG	テーブルメモリ ページアドレスレジスタ
PSVPAG	プログラム空間可視化 ページアドレスレジスタ
RCOUNT	繰り返しループカウンタレジスタ
CORCON	CPU 制御レジスタ

プログラマ用モデルに関連する全レジスタは表 2-5 のようにメモリにマップされています。

図 2-2: プログラマ用モデル



2.2.1 作業レジスタ アレイ

16 個の作業レジスタは、データ、アドレス、アドレス オフセット レジスタとして機能します。W レジスタの機能は、それをアクセスする命令のアドレッシング モードにより決定されます。

PIC24F の命令は、レジスタとファイル レジスタ命令の 2 つの命令タイプに分けられます。レジスタ命令は、各 W レジスタをデータ値またはアドレス オフセット値として使用します。次に例を示します。

例 2-1: レジスタ命令

```
MOV    W0, W1           ; W0 の内容を W1 に移動
MOV    W0, [W1]         ; W0 を W1 に格納されているアドレスに移動
ADD    W0, [W4], W5     ; W0 の内容を W4 で指定されたアドレスの内容に加算
                          ; 結果を W5 に格納
```

2.2.1.1 W0 とファイル レジスタ命令

W0 は特別な作業レジスタで、ファイル レジスタ命令で使用できる唯一の作業レジスタです。ファイル レジスタ命令は、命令のオペコード中に含まれるメモリアドレスと W0 に対して動作します。W1-W15 はファイル レジスタ命令では対象レジスタとして指定できません。

ファイルレジスタ命令は、1 個の W レジスタしか持たない既存の PICmicro® デバイスとの下位互換性のために用意されたものです。ラベル「WREG」が、ファイルレジスタ命令で W0 を指定するためにアセンブラ構文の中で用いられます。次に例を示します。

例 2-2: ファイル レジスタ命令

```
MOV    WREG, 0x0100     ; W0 の内容をアドレス 0x0100 に移動
ADD    0x0100, WREG     ; W0 をアドレス 0x0100 に加算し結果を W0 に格納
```

注: アドレッシング モードと命令構文に関する詳細は、『dsPIC30F プログラマ リファレンス マニュアル』(DS70030) を参照して下さい。

2.2.1.2 W レジスタ メモリ マッピング

W レジスタはメモリにマッピングされているので、次に示すように、ファイル レジスタ命令で W レジスタをアクセスすることができます。

例 2-3: ファイル レジスタ命令で W レジスタをアクセス

```
MOV    0x0004, W10      ; MOV W2, W10 と同じ
  
ここで
  
0x0004 は W2 のメモリアドレスです。
```

さらに、W レジスタをアドレス ポインタとオペランド指定用と同時に使う命令も実行できます。次に例を示します。

例 2-4: W レジスタをアドレス ポインタとオペランド指定として使う

```
MOV    W1, [W2++]
  
ここで
  
W1 = 0x1234
W2 = 0x0004           ; [W2] は W2 をアドレス指定
```

例 2-4 では、W2 の内容は 0x0004 です。しかし W2 はアドレス ポインタとして使用されているので、メモリの 0x0004 番地を指しています。また、W2 はこのメモリアドレスにマップされています。この例はあまり起きないことですが、実行するまで検出できません。PIC24F では、データ書き込みが優先されるので、この例の結果は W2=0x1234 となります。

2.2.1.3 Wレジスタとバイトモード命令

Wレジスタアレイを対象とするバイト命令は、対象レジスタの最下位バイトのみに影響します。作業レジスタはメモリマップされているため、下位バイト、上位バイトともバイト幅のデータメモリ空間アクセスにより操作可能です。

2.2.2 シャドーレジスタ

表 2-5 に示すレジスタにはシャドーレジスタを持っているものがあります。シャドーレジスタは一時的な保存レジスタとして使われ、イベント発生時に、元のレジスタと互いに入れ替え転送ができます。どのシャドーレジスタも直接アクセスすることはできません。シャドーレジスタでは、**PUSH.S** と **POP.S** 命令がシャドー化するために使用されます。

2.2.2.1 PUSH.S、POP.S とシャドーレジスタ

PUSH.S と **POP.S** 命令は、関数呼び出しまたは割り込みサービスルーチン (ISR) で、高速コンテキスト保存/復元のために使用されます。**PUSH.S** 命令は以下のレジスタ値をそれぞれのシャドーレジスタに転送します。

- W0 ~ W3
- SR (N、OV、Z、C、DC ビットのみ)

POP.S 命令はシャドーレジスタからの値をそれぞれのレジスタに復元します。**PUSH.S** と **POP.S** の命令を使用するコード例を例 2-5 に示します。

例 2-5: PUSH.S と POP.S 命令

```
MyFunction:
    PUSH.S           ; Wレジスタ、MCU ステータス保存
    MOV    #0x03, W0 ; リテラル値を W0 へ
    ADD   RAM100     ; W0 を RAM100 の内容に加算
    BTSC  SR, #Z     ; 結果が 0?
    BSET  Flags, #IsZero ; もしそうなら、フラグセット
    POP.S           ; W レジ、MCU ステータスを復元
    RETURN
```

PUSH.S 命令はシャドーレジスタに保存されている内容に上書きします。シャドーレジスタの深度は一段階のみなので、シャドーレジスタを複数のソフトタスクで使用する場合は注意が必要です。

シャドーレジスタを使用するタスクの実行中には、それよりも優先順位の高いシャドーレジスタを使用するタスクが割り込まないようにして下さい。優先順位の高いタスクが低優先のタスクに割り込むと、優先順位の低いタスクで保存されたシャドーレジスタの内容は、高優先のタスクによって上書きされます。

2.2.3 未初期化 Wレジスタ リセット

すべての RESET で、Wレジスタアレイ (W15 以外) はクリアされ、書き込むまで未初期化と見なされます。未初期化レジスタをアドレスポインタとして使用しようとするデバイスではリセットされます。詳細は第7章「リセット」を参照して下さい (入手につきましてはマイクロチップのウェブサイト www.micorchip.com を参照してください)。

Wレジスタを初期化するためにはワード書き込みをする必要があります。バイト書き込みは初期化検出ロジックに影響しません。

2.3 ソフトウェア スタックポインタ

W15 は専用のソフトスタックポインタとして使用され、例外処理、サブルーチン呼び出しおよびその戻りで自動的に更新されます。しかし、W15 は他の W レジスタと同様に命令によっても参照可能です。これにより、スタックポインタの読み取り、書き込み、および操作が容易になります (例、スタックフレームの生成)。

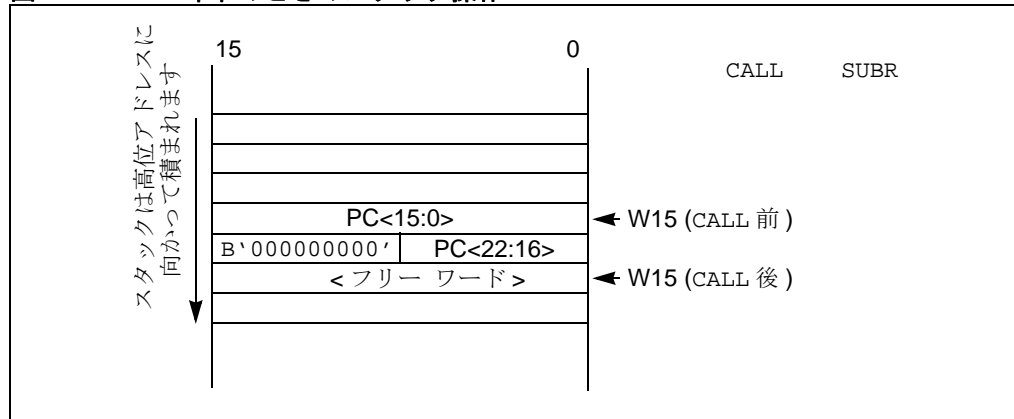
注： 誤った配置指定でスタック アクセスされるのを回避するため、W15<0> はハードウェアで「0」に固定化されています。

W15 はすべてのリセット時 0x0800 に初期化されます。このアドレスにより、スタックポインタ (SP) がすべての PIC24F デバイスで有効な RAM を示すことが保証され、ユーザソフトによる SP の初期化の前にマスク不可トラップ例外を起こさずにスタック利用が可能になります。ユーザーは初期化の際、データ空間内の任意の位置に SP をプログラムし直すことができます。

スタックポインタは、常に最初に使用可能なフリーのワード位置を指しており、ソフトウェアスタックとして低い方から高いアドレスに向かって使用されていきます。図 2-3 に示すように、スタックをポップ (読み出す) 前に減少し、スタックプッシュ (書き込み) 後に増加します。

プログラムカウンタ (PC) がスタックにプッシュされると、PC<15:0> が最初に利用可能なスタックワードにプッシュされ、PC<22:16> が二番目の利用可能なスタック位置にプッシュされます。CALL 命令時の PC プッシュでは、プログラムカウンタの MSB は図 2-3 のようにプッシュの前にゼロ拡張されます。例外処理の間は、プログラムカウンタの MSB には CPU ステータスレジスタ、SR の下位の 8 ビットが連結されます。したがって、SRL の内容は割り込み処理の際には自動的に保存されます。

図 2-3: CALL 命令のときのスタック操作



2.4 CPU レジスタ説明

2.4.1 SR: CPU ステータス レジスタ

PIC24F CPU は 16 ビットのステータス レジスタ (SR) を持っています。その LSB は下位ステータス レジスタ (SRL) として参照され、SR の上位バイトは SRH ととして参照されます。SR レジスタの詳細説明をレジスタ 2-1 に示します。

SRL には MCU ALU 操作の全フラグと、CPU の割り込み優先レベル ステータス ビット IPL<2:0>、さらに **REPEAT** ループのアクティブ ステータス ビット RA(SR<4>) が含まれています。例外処理中は、SRL はプログラム カウンタ (PC) の MSB と結合されてワード値としてスタックに格納されます。

SRH はデジット キャリー ビット DC (SR<8>) のみを含みます。

SR ビットは次の項目を除き読み書き可能です。

RA ビット (SR<4>) : RA は読み出し専用ビット

IPL<2:0>: レジスタが無効化されたとき (NSTDIS = 1) は、IPL<2:0> ビットは読み出し専用

注: 命令ごとの SR ビットへの影響の説明に関しては『dsPIC30F プログラマ リファレンス マニュアル』(DS70030) を参照して下さい。

PIC24F ファミリ リファレンス マニュアル

レジスタ 2-1: SR: CPU ステータス レジスタ

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	DC
ビット 15							ビット 8
R/W-0 ⁽²⁾	R/W-0 ⁽²⁾	R/W-0 ⁽²⁾	R-0	R/W-0	R/W-0	R/W-0	R/W-0
IPL<2:0>			RA	N	OV	Z	C
ビット 7							ビット 0

凡例:

R = 読み出し可

W = 書き込み可

U = 未実装、読むと「0」

-n = POR 後の値

'1' = セット

'0' = クリア

x = 不定

bit 15-9 **未実装:** 読むと「0」

bit 8 **DC:** MCU ALU ハーフ キャリー / ボロー ビット

1 = 第 4 下位ビットよりキャリーアウト (バイトサイズデータ) または第 8 下位ビット (ワードサイズデータ) の結果からキャリーアウトが発生した

0 = 第 4 下位ビットよりキャリーアウト (バイトサイズデータ) または第 8 下位ビット (ワードサイズデータ) の結果からキャリーアウト発生なし

bit 7-5 **IPL<2:0>:** CPU 割り込み優先レベル ステータス ビット⁽¹⁾

111 = CPU 割り込みレベル 7 (15)、ユーザー割り込み禁止

110 = CPU 割り込みレベル 6 (14)

101 = CPU 割り込みレベル 5 (13)

100 = CPU 割り込みレベル 4 (12)

011 = CPU 割り込みレベル 3 (11)

010 = CPU 割り込みレベル 2 (10)

001 = CPU 割り込みレベル 1 (9)

000 = CPU 割り込みレベル 0 (8)

bit 4 **RA:** REPEAT ループ アクティブ ビット

1 = REPEAT ループ実行中

0 = REPEAT ループは実行中ではない

bit 3 **N:** MCU ALU 負ビット

1 = 結果は負

0 = 結果は負でない (ゼロか正)

bit 2 **OV:** MCU ALU オーバー フロー ビット

このビットは符号付算術演算で使用 (2 の補数)。符号ビットを変更する大きさのオーバー フローが起きたことを示す

1 = 符号付算術演算でオーバー フローが発生した (この算術演算内)

0 = オーバー フローは発生していない

bit 1 **Z:** MCU ALU ゼロ ビット

1 = 最後の演算結果がゼロ

0 = 最後の演算の結果はゼロではない

bit 0 **C:** MCU ALU キャリー / ボロー ビット

1 = 最上位ビットの結果からキャリーアウトが発生した

0 = 最上位ビットの結果からキャリーアウトは発生していない

注 1: IPL<2:0> ビットは IPL<3> ビット (CORCON<3>) と結合されて CPU の割り込み優先レベルとなります。括弧の中の値は IPL<3> = 1 の場合の IPL を示しています。IPL<3> = 1 のときはユーザー割り込み禁止となります。

2: IPL<2:0> ステータス ビットは NSTDIS = 1 (INTCON1<15>) のときは読み出し専用となります。

レジスタ 2-2: CORCON: コア制御レジスタ

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
ビット 15						ビット 8	

U-0	U-0	U-0	U-0	R/C-0	R/W-0	U-0	U-0
—	—	—	—	IPL3 ⁽¹⁾	PSV	—	—
ビット 7						ビット 0	

凡例:	C = クリア可
R = 読み出し可	W = 書き込み可
U = 未実装、読むと「0」	
-n = POR 後の値	'1' = セット
	'0' = クリア
	x = 不定

- bit 15-4 **未実装**: 読むと「0」
 - bit 3 **IPL3**: CPU 割り込みレベル ステータス ビット⁽¹⁾
 1 = CPU 割り込みレベルは7より高い
 0 = CPU 割り込みレベルは7以下
 - bit 2 **PSV**: データ空間のプログラム空間可視化有効化ビット
 1 = プログラム空間のデータ空間への可視化有効化
 0 = プログラム空間のデータ空間への可視化無効化
 - bit 1-0 **未実装**: 読むと「0」
- 注 1: IPL3 = 1 のときユーザー割り込みは禁止。

2.4.2 他の PIC24F CPU コントロール レジスタ

以下に示されているレジスタは PIC24F の CPU コアに関連するレジスタですが、これらについての詳細については本書の他の章で説明されています。

2.4.2.1 TBLPAG: テーブル ページ アドレス ポインタ

TBLPAG レジスタは、テーブルの読み出しと書き込み操作時、プログラム メモリ アドレスの上位 8 ビットを保持するために使用します。テーブル命令は、プログラム メモリ 空間とデータメモリ空間の間のデータ転送に使用します。詳細は **第 4 章 「プログラム メモリ」** を参照してください (入手につきましてはマイクロチップのウェブサイト www.microchip.com をチェックしてください)。

2.4.2.2 PSVPAG: プログラム空間可視化ページアドレスポインタ

プログラム空間可視化により、ユーザーはプログラム メモリ空間の 32 キロバイトの区間をデータアドレス空間の上位 32 キロバイトにマップできます。この特徴により、データメモリを操作する PIC24F 命令を介して、定数データに透過的にアクセスできます。PSVPAG レジスタは、データアドレス空間にマップするプログラム メモリ空間の 32 キロバイト領域を選択します。PSVPAG レジスタの詳細情報は、**第 4 章 「プログラム メモリ」** を参照して下さい(入手につきましてはマイクロチップのウェブサイト www.microchip.com を参照してください)。

2.4.2.3 DISICNT: 割り込みカウント レジスタの無効化

DISICNT レジスタは **DISI** 命令で使用され、優先レベルが 1 ~ 6 の割り込みを指定されたサイクル数だけ禁止とします。詳細については**第 8 章 「割り込み」** を参照して下さい。

2.5 算術演算論理ユニット (ALU)

PIC24F の ALU は 16 ビット幅であり、加算、減算、単ビットシフトおよび論理演算が可能で、特にことわりない限り、算術演算は 2 の補数であるのが自然です。算術演算によって、ALU は SR レジスタのキャリー/ボロー (C)、ゼロ (Z)、負 (N)、オーバーフロー (OV) および ハーフ キャリー/ボロー (DC) のステータスビットの値に影響を及ぼします。C および DC ステータスビットは減算演算の際にそれぞれボロー ビットとデジットボロー ビットとして機能します。

ALU は使用される命令のモードによって 8 ビットまたは 16 ビット動作を実行します。命令のアドレッシングモードに応じて、ALU 演算データは W レジスタ アレイまたはデータメモリから取得されます。同様に、ALU からの出力データは W レジスタ アレイまたはデータメモリに書き込まれます。

各命令、アドレッシングモードおよび 8 ビット /16 ビット命令モードに影響される SR ビットの詳細は、「**dsPIC30F プログラマリファレンスマニュアル**」(DS70030) を参照して下さい。

注 1: バイト演算は 16 ビットの ALU を使用し 8 ビットを超える結果を生成します。しかし、PICmicro デバイスとの下位互換性の維持のため、ALU はすべてのバイト演算の結果をバイトとして(つまり MSB は変更なく)書き戻し、結果の LSB のみの状態によって CPU ステータス レジスタ SR が更新されます。

2: バイトモードで実行されたすべてのレジスタ命令は、W レジスタの LSB のみに影響します。W レジスタの MSB は、メモリマップされた W レジスタの内容にアクセスするファイルレジスタ命令で変更できます。

2.6 乗算と除算のサポート

2.6.1 概要

PIC24F コアには 17 ビット x 17 ビットの乗算器が含まれていて、次のような符号なし、符号つき、あるいは混合符号の演算を実行できます。

1. 16 ビット x 16 ビット符号つき
2. 16 ビット x 16 ビット符号なし
3. 符号つき 16 ビット x 符号なし 5 ビット (リテラル)
4. 符号なし 16 ビット x 符号なし 16 ビット
5. 符号なし 16 ビット x 符号なし 5 ビット (リテラル)
6. 符号なし 16 ビット x 符号つき 16 ビット
7. 符号なし 8 ビット x 符号なし 8 ビット

除算ブロックは 32 ビット /16 ビットと 16 ビット /16 ビットの符号つきと符号なしの整数除算をサポートしていて、以下のデータサイズの演算が可能です。

1. 符号つき 32 ビット / 符号つき 16 ビット除算
2. 符号つき 32 ビット / 符号なし 16 ビット除算
3. 符号つき 16 ビット / 符号つき 16 ビット除算
4. 符号なし 16 ビット / 符号なし 16 ビット除算

2.6.2 乗算器

図 2-8 に乗算器のブロック図を示します。乗算器は、16 ビットの符号つき、符号なし、符号混合の乗算命令と、PIC18F の符号なし乗算命令 **MULWF** (**MUL.w** と **MUL.b**) をサポートします。すべての乗算命令はレジスタ直接アドレッシング モードのみをサポートしています。32 ビットの演算結果 (**MULWF** 以外のすべての乗算命令による) は、禁止されている W15:W14 以外の任意の隣接 W レジスタ ペアに書き込まれます。

MULWF 命令はバイトまたはワードの演算を直接実行できます。格納先は常に W レジスタ アレイ内の W3:W2 レジスタ ペアになります。バイト乗算の 16 ビットの結果は W2 になり (W3 は使用しない)、ワード乗算の 32 ビットの結果は W3:W2 となります。

注：乗算命令の結果のレジスタ ペアは並んでいる (つまり奇数 : 偶数) 必要があり、奇数側に上位結果が格納され、偶数側に下位結果が格納されます。例えば、W3:W2 は認められますが、W4:W3 は認められません。

乗算命令 (特別なケースの **MULWF** を除く) の被乗数は、W アレイ (1 ワード目) とデータ空間 (2 ワード目) から引き出されます。**MULWF** のときは、W2 (1 ワード目または 1 バイト目) と、13 ビットの絶対アドレスのデータ空間 (2 ワード目または 2 バイト目) からゼロ拡張されて引き出されます。

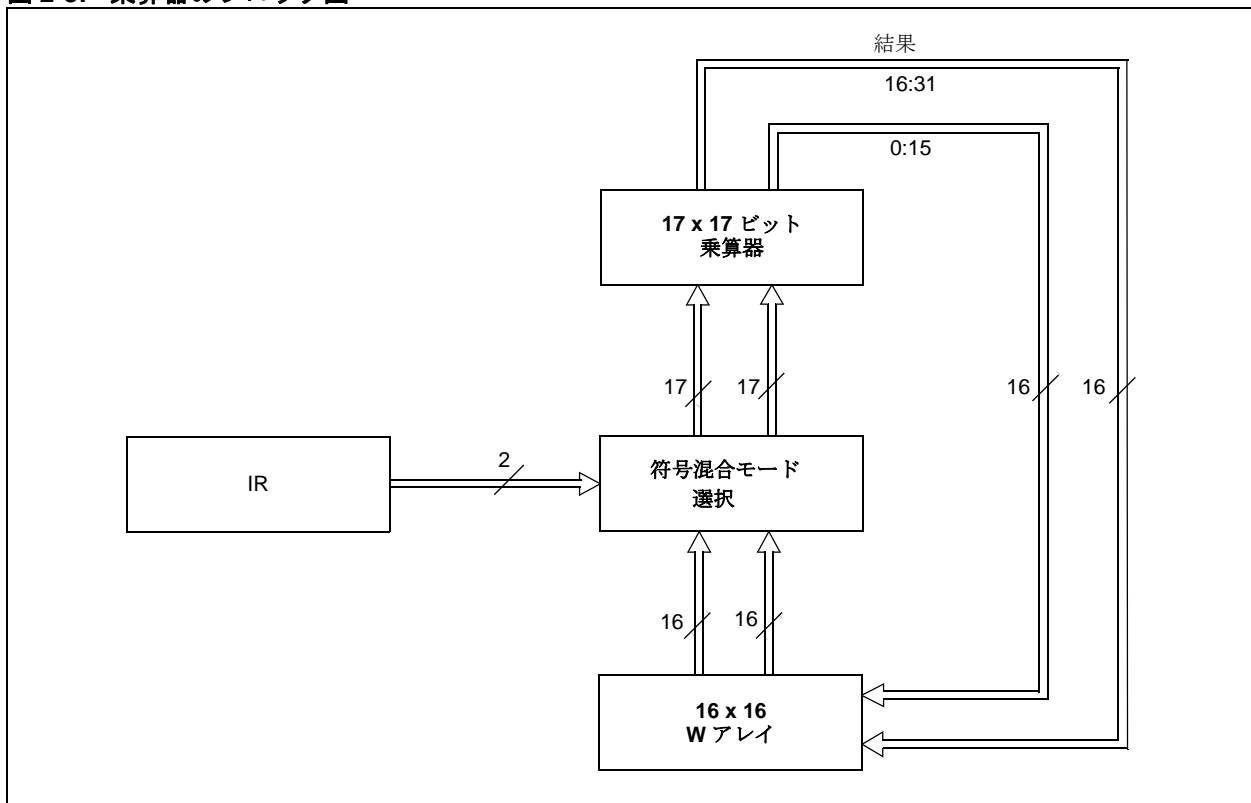
図 2-8 に示したように、これらの命令の結果の W アレイとデータバス (W アレイ経由で) への書き戻しには、追加のデータバスが供給されます。

2.6.2.1 単一と混合モード整数

すべてのオペランドをゼロまたは符号拡張した 17 ビットとして扱う単純なデータ処理ロジックであるため、符号なし、符号つき、符号混合の乗算を符号つきの値として実行できます。すべての符号なしオペランドは、常に 17 ビット目にゼロ拡張した乗算器への入力値として扱います。すべての符号つきオペランドは常に 17 ビット目に符号拡張した乗算器への入力値として扱います。

符号なし 16 ビット乗算では、乗算器から符号なし 32 ビットの結果が生成されます。符号つき 16 ビット乗算では、乗算器から 30 ビットのデータと 2 ビットの符号が生成されます。16 ビットの符号混合 (符号つき / 符号なし) の乗算では、乗算器から 31 ビットのデータと 1 ビットの符号が生成されます。

図 2-8: 乗算器のブロック図



2.6.3 除算器

PIC24Fの特徴は、32ビット/16ビットと16ビット/16ビットの両方の符号つき、符号なしの整数の除算演算を1個の命令の繰り返し除算で実装していることです。

すべての除算命令の商はW0に、余りはW1となります。16ビットの符号つきと符号なしDIV命令は、16ビット除数用のWレジスタと、32ビット被除数用の(並んだ)Wレジスタペア(W(m+1):Wm)に任意のWレジスタを指定できます。除算アルゴリズムでは、除数の1ビットあたり1サイクルかかりますので、32ビット/16ビット命令と16ビット/16ビット命令は両方とも同じ実行サイクル数となります。

除算命令は、REPEAT ループの中で実行する必要があります。その他の実行方法(例えば個別の除算命令を並べるなど)では正しく機能しません。なぜなら、命令実行の機能がRCOUNTに依存しているためです。除算フローで自動的にREPEATの設定をするわけではないので、表2-2に示すような正確なオペランド値設定を明示的に実行する必要があります(REPEATは対象の命令を{オペランド値+1}回だけ実行する)。

表 2-2: 除算実行回数

命令	説明	繰り返し回数	REPEAT オペランド 値	合計実行時間 (REPEAT 含む)
DIV.SD	符号付除算: W(m+1):Wm/Wn ÷ W0; Rem ÷ W1	18	17	19
DIV.SW	符号つき除算: Wm/Wn ÷ W0; Rem ÷ W1	18	17	19
DIV.UD	符号なし除算: W(m+1):Wm/Wn ÷ W0; Rem ÷ W1	18	17	19
DIV.UW	符号なし除算: Wm/Wn ÷ W0; Rem ÷ W1	18	17	19

繰り返し中の中間データは毎繰り返しの後にすべてW1:W0に格納されます。N、C、Zのステータスフラグは繰り返しごとに制御情報として伝達されながら使用されます。このように除算命令は19サイクルで実行されるとなっていますが、除算繰り返し中は、他のREPEATループ同様に割り込み可能です。

ゼロ割は算術演算エラー トラップを発生します。除算器は除算命令の最初のサイクルで評価されるので、トラップの例外処理が開始される前に1サイクル目は実行されます。詳細は第8章「割り込み」を参照して下さい。

2.7 コンパイラとの親和性を持つアーキテクチャ

コアアーキテクチャは、C コンパイラ効率率 (コードサイズと速度) が良くなるように設計されています。

- ほとんどの命令が、データ (またはプログラム データ) メモリの読み出し、作業レジスタ (データ) の読み出し、データ メモリへの書き込みとプログラム (命令) メモリの読み出しを 1 命令サイクル内で実行できます。その結果、3 個のパラメータを持つ命令がサポートされ、 $A + B = C$ の演算が 1 サイクルで実行できます。
- 命令のアドレッシングモードが非常に柔軟で、コンパイラのニーズにマッチするようになっています。
- 16 個の 16 ビットの作業レジスタがあり、それぞれがデータ、アドレス、アドレス オフセットレジスタとして機能します。一つのレジスタ (W15) が割り込みと CALL 時のソフトウェアスタックとして動作します。
- 全データ空間に対しリニアな間接アクセスがサポートされており、さらにメモリ直接アドレス範囲も 8K バイトと広がっていて、16 ビットの直接アドレスによる読み出し書き込み命令も追加されています。
- 任意の作業レジスタを使用し、新たなテーブル読み出し書き込み命令を介することで、プログラム空間 (ユーザーとテスト空間) の 32k ワード (64K バイト) ページのリニアな間接アクセスがサポートされています。
- データ空間の一部をプログラム空間にマッピングすることが可能なので、PSV モードを使用して定数データをデータ空間にあるようにアクセスできます。

2.8 複数ビットシフトのサポート

PIC24F コアは、シフトブロックを使用することにより、1 サイクルの複数ビット算術シフトおよび論理シフトをサポートしています。さらに ALU による 1 ビットシフトもサポートしています。複数ビットシフトは、最大 15 ビットの算術右シフトと、最大 15 ビットの左シフトを 1 サイクルで実行することができます。

シフト操作を行う命令のまとめを以下の表 2-3 に示します。

表 2-3: 1 ビットおよび複数ビットシフト操作に使用する命令

命令	説明
ASR	レジスタの 1 ビット以上の算術右シフト
SL	レジスタの 1 ビット以上の左シフト
LSR	レジスタの 1 ビット以上の論理右シフト

すべての複数ビットシフト命令は、オペランドのソース、結果の格納先とも、レジスタ直接アドレッシングモードのみのサポートとなります。

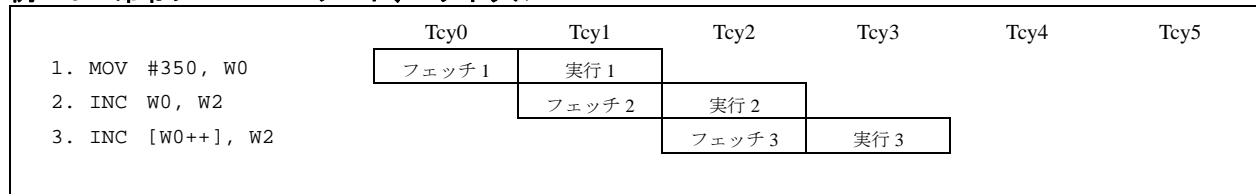
2.9 命令フローの種類

PIC24F アーキテクチャのほとんどの命令は、プログラムメモリの1ワードで構成され、1サイクルで実行されます。命令プリフェッチメカニズムによって1サイクル(1Tcy)実行が実現されています。ただし、2つまたは3つの命令サイクルを必要とする命令もあります。その結果、PIC24F アーキテクチャには6つの異なるタイプの命令フローがあります。以下にそのフローを示します。

1. 1命令ワード、1命令サイクル

これらの命令は例2-6に示したように1サイクルで実行されます。

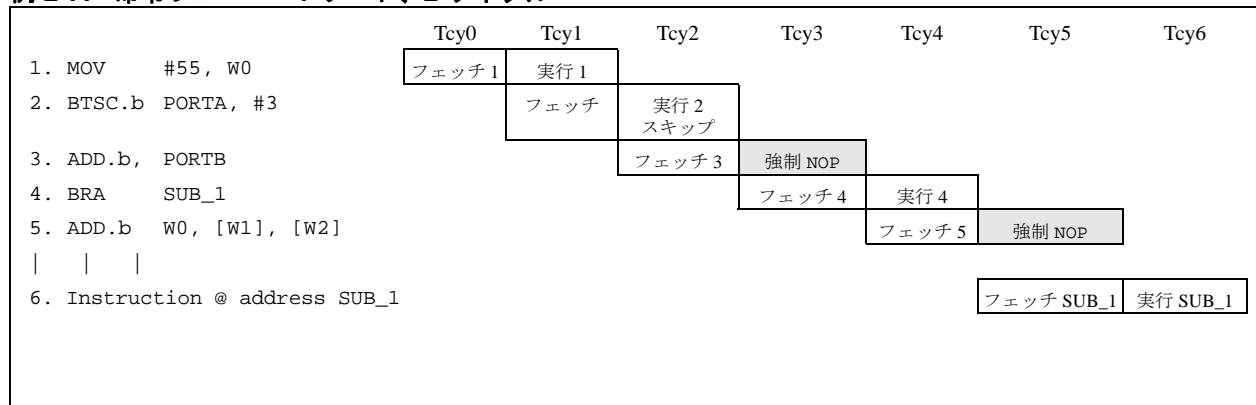
例 2-6: 命令フロー — 1ワード、1サイクル



2. 1命令ワード、2命令サイクル

この命令には相対分岐、相対 CALL、スキップとリターン命令が含まれます。命令がプログラムカウンタ(PC)を変更するとき(インクリメント以外で)、パイプラインによるフェッチが破棄されます。このため例2-7に示したように、命令実行には2命令サイクルを必要とします。

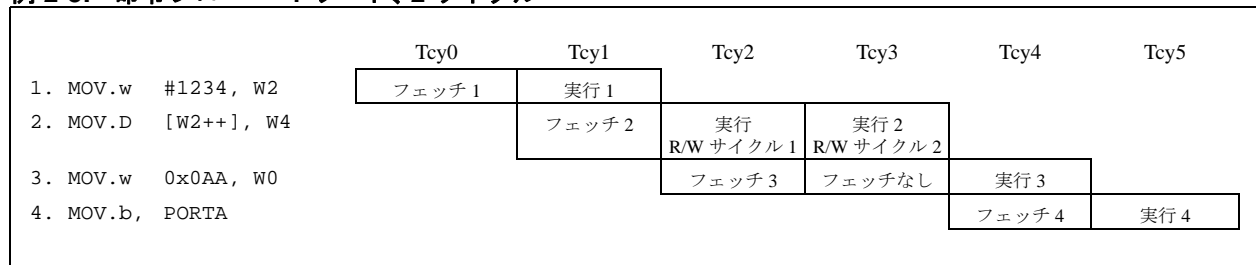
例 2-7: 命令フロー — 1ワード、2サイクル



3.1 命令ワード、2命令サイクル(ダブルワード)

このタイプの命令は LDDW と STDW だけです(ダブルワードを読み書きする)。データへのアクセスは順番ですから、この命令が完了するには、例2-8に示したように2命令サイクルが必要です。

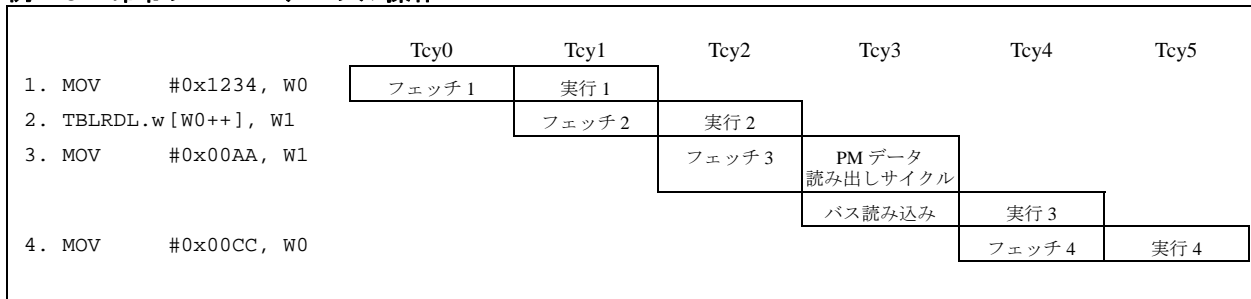
例 2-8: 命令フロー — 1ワード、2サイクル



4.1 命令ワード、2 命令サイクルテーブル操作

これらの命令はフェッチを一時停止して、読み取りまたは書き込みサイクルをプログラムメモリに挿入します。テーブル処理実行時にフェッチした命令は1サイクルの間保存され、例 2-9 に示したようにテーブル処理の次のサイクルで実行されます。

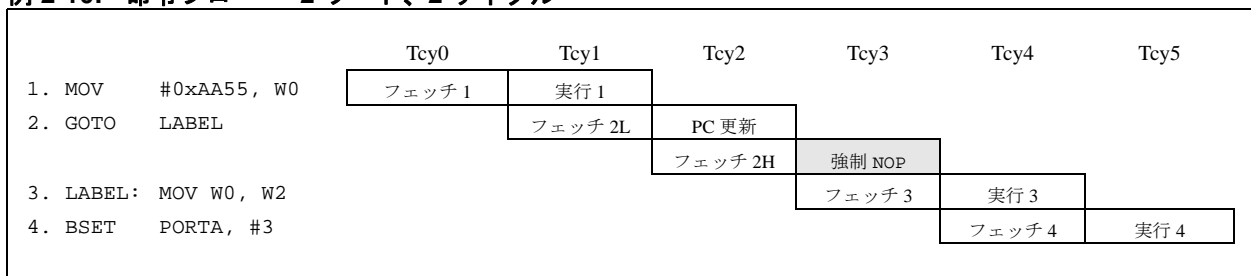
例 2-9: 命令フロー – テーブル操作



5.2 命令ワード、2 命令サイクル (GOTO, CALL)

これらの命令では、命令のあとのフェッチにデータを含みます。その結果、例 2-10 に示したように 2 命令サイクルとなります。2 ワード命令の 2 ワード目がエンコードされるとき、NOP として実行し、CPU が次の命令の最初のワードを先にフェッチしないようにしています。これは 2 ワード命令がスキップ命令でスキップされる場合に重要です (例 2-13 参照)。

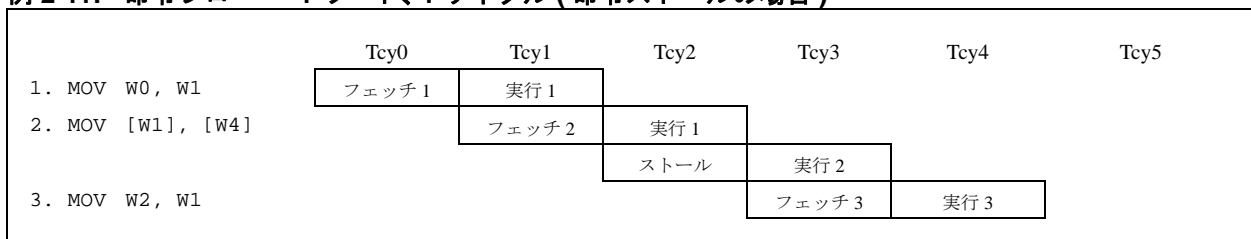
例 2-10: 命令フロー – 2 ワード、2 サイクル



6. アドレスレジスタ従属関係

これらの命令は、データ空間の読み出しと書き込み間にデータアドレスの従属関係があってストールする場合があります。この場合リソースの衝突を解決するため 2.11 項「アドレスレジスタ 従属関係」で説明したように 1 サイクルが追加されます。

例 2-11: 命令フロー – 1 ワード、1 サイクル (命令ストールの場合)



2.10 プログラム フロー ループ制御

PIC24F は無条件の自動プログラム ループ制御を提供するため **REPEAT** 命令をサポートしています。**REPEAT** 命令は単一命令プログラム ループを実装するため使用されます。この命令は CPU ステータス レジスタ SR 内のコントロール ビットを使用し、CPU 動作を一時的に変更します

2.10.1 REPEAT ループ

REPEAT 命令はそれに続く命令を指定回数反復実行させます。命令に含まれているリテラル値または W レジスタ値のいずれかで、反復カウンタ値を指定できます。W レジスタ オプションにより、ループ回数をソフトウェア変数にすることができます。**REPEAT** ループにある命令は少なくとも 1 回実行されます。**REPEAT** ループ用反復回数は 14 ビットリテラル値 + 1 または $W_n + 1$ です。

REPEAT 命令の 2 種類の構文は次のようになります。

例 2-12: REPEAT 命令の構文

```
REPEAT #lit14          ; RCOUNT <-- lit14
(Valid target Instruction)

または

REPEAT Wn             ; RCOUNT <-- Wn
(Valid target Instruction)
```

2.10.1.1 REPEAT 動作

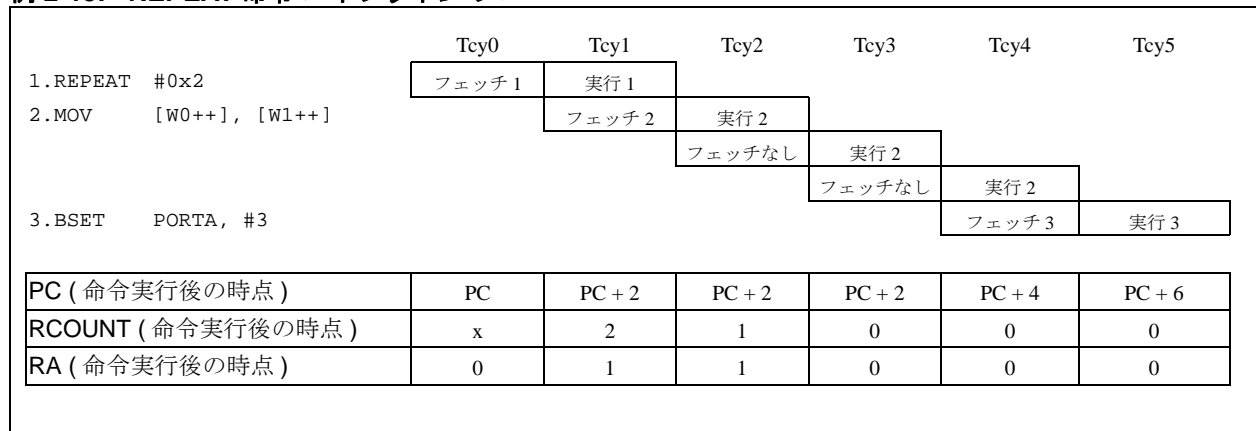
REPEAT 動作ループ回数はメモリマップされた 14 ビット RCOUNT レジスタに保存されています。RCOUNT は **REPEAT** 命令で初期化されます。RCOUNT 値がゼロでなければ、**REPEAT** 命令は繰り返しアクティブ ステータス ビット RA(SR<4>) を「1」にセットします。

RA は読み出し専用で、ソフトウェアでは変更できません。**REPEAT** ループ カウンタ値が「0」より大きい間は、プログラム カウンタ (PC) は増分されません。つまり PC 増分は RCOUNT=0 まで禁止されています。**REPEAT** ループの命令フロー例は、例 2-13 を参照して下さい。

ループカウンタ値が「0」のときは、**REPEAT** は **NOP** と同じとなり、RA (SR<4>) ビットはセットされません。**REPEAT** ループは本来開始するまで禁止されているため、対象命令が次の命令をプリフェッチする間に一度だけ実行されます。(すなわち、通常実行フロー)

注: REPEAT 命令の直後の命令 (すなわち、対象命令) は常に少なくとも 1 回実行されます。この命令は常に 14 ビットリテラルまたは W レジスタ オペランドに指定されている値より 1 回多く実行されます。

例 2-13: REPEAT 命令 パイプライン フロー



2.10.1.2 REPEAT ループ中の割り込み

REPEAT ループにはいつでも割り込みができます。

例外処理中は、**RA** 状態がスタックに保存されるため、ユーザーは、任意回数時点でネストされた割り込みからさらなる **REPEAT** ループを実行することができます。**SRL** をスタックした後、**ISR** 実行中に通常の実行フローがリストアされるようにするため **RA** ステータスビットはクリアされます。

注: **REPEAT** ループが割り込まれて **ISR** 処理の間、**ISR** 内で他の **REPEAT** 命令を実行する前に、**RCOUNT** (繰り返しカウントレジスタ) をスタックへ待避しなくてはなりません。

注: もし **REPEAT** が **ISR** の中で使われた場合は、**RETFIE** の前に **RCOUNT** 値をスタックから戻さなければなりません。

RETFIE を使用して **ISR** から **REPEAT** ループへ復帰するのに特別な処理は必要ありません。割り込みは **RETFIE** 命令の第3サイクルの間に繰り返し命令をあらかじめフェッチします。**SRL** レジスタを **POP** したとき待避していた **RA** ビットが復旧し、セットされていたれば、中断されていた **REPEAT** ループが再開されます。

注: 繰り返し命令 (**REPEAT** ループ内の対象の命令) がプログラム空間可視化 (**PSV**) を使用してプログラム空間 (**PS**) のデータにアクセスする場合、例外処理から復帰後最初に実行される時には2命令サイクルが必要です。ループの最初の繰り返しと同様、タイミング制限のため、最初の命令は **PS** にあるデータには1サイクルではアクセスできません。

2.10.1.2.1 REPEAT ループの早期終了

割り込まれた **REPEAT** ループは、**ISR** 処理内でソフトウェアにより **RCOUNT** レジスタを消去すれば通常よりも早く終了させることができます。

2.10.1.3 REPEAT 命令に関する制限事項

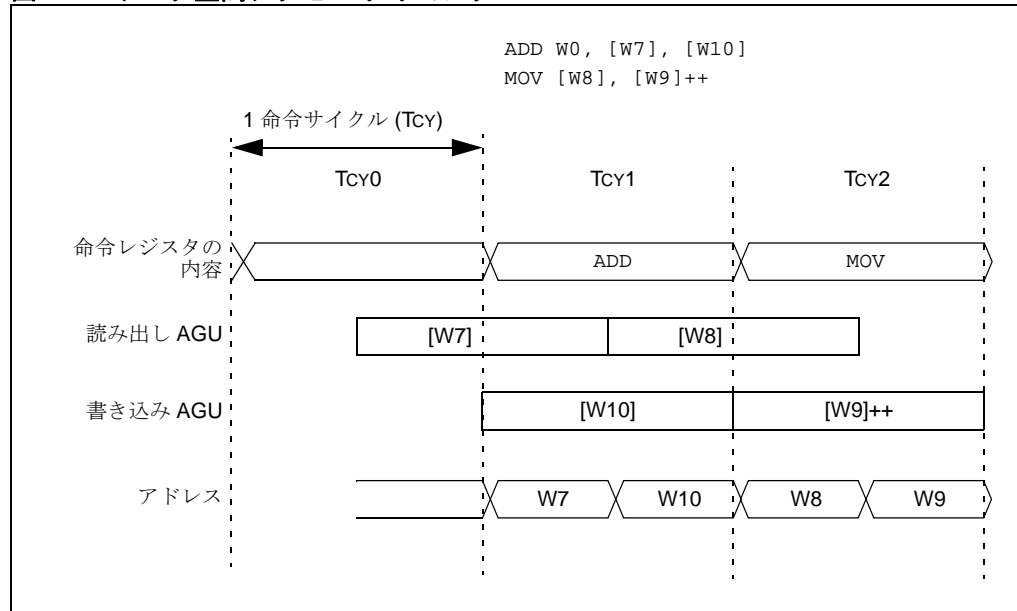
以下の場合を除いて、**REPEAT** 命令には任意の命令を続けることができます。

1. プログラム フロー制御命令 (任意の分岐、比較とスキップ、サブルーチン呼び出し、リターンなど)
2. 別の **REPEAT** 命令
3. **DISI**、**ULNK**、**LNK**、**PWRSVAV**、**RESET** 命令
4. **MOV.D** 命令

2.11 アドレス レジスタ 従属関係

PIC24F アーキテクチャは、ほとんどの命令用にデータメモリ読み出し（元）とデータメモリ書き込み（先）をサポートします。アドレス生成ユニット（AGU）がデータメモリ読み出したまたは書き込みに必要な有効アドレス（EA）の計算を完了するには、それぞれ1命令サイクルかかります。このタイミングのため、図 2-9 で示されるように、各命令用のデータメモリ読み書き動作が部分的に重複します。この重複のため、「書き込み後読み取り」（RAW）データ従属関係が命令境界をまたいで発生する可能性があります。RAW データ従属関係は PIC24F CPU によって実行時に検出および処理されます。

図 2-9: データ空間アクセス タイミング



2.11.1 書き込み後読み取り従属関係ルール

実行中の命令で W_n 作業レジスタが書き込み先として使用され、プリフェッチした次の命令で読み取られる W_n レジスタが同じ場合、次のルールが適用されます。

1. 書き込み先（実行中の命令）が W_n の内容を変更しない場合、ストールは発生しません。

または

2. 読み取り元（プリフェッチした命令）が W_n を使用して EA を計算しない場合、ストールは発生しません。

各命令サイクルの間に、PIC24F ハードウェアは RAW データ従属関係が発生しようとしていないか自動的に確認します。上記の条件が満たされない場合、CPU はプリフェッチした命令を実行する前に自動的に1命令サイクルの遅延を挿入します。命令のストールにより、次の（プリフェッチした）命令が書き込まれたデータを使用する前に、 W レジスタに書き込むために十分な時間が提供されます。

表 2-4: 書き込み後読み出し従属関係のまとめ

Wn を使う 実行先アドレス モード	Wn を使う 実行元アドレス モード	ステータス	例 (Wn = W2)
直接	直接	許可	ADD.w W0, W1, W2 MOV.w W2, W3
直接	間接	ストール	ADD.w W0, W1, W2 MOV.w [W2], W3
直接	修飾つき間接	ストール	ADD.w W0, W1, W2 MOV.w [W2++], W3
間接	直接	許可	ADD.w W0, W1, [W2] MOV.w W2, W3
間接	間接	許可	ADD.w W0, W1, [W2] MOV.w [W2], W3
間接	修飾つき間接	許可	ADD.w W0, W1, [W2] MOV.w [W2++], W3
修飾つき間接	直接	許可	ADD.w W0, W1, [W2++] MOV.w W2, W3
間接	間接	ストール	ADD.w W0, W1, [W2] MOV.w [W2], W3 ; W2=0x0004 (mapped W2)
間接	修飾つき間接	ストール	ADD.w W0, W1, [W2] MOV.w [W2++], W3 ; W2=0x0004 (mapped W2)
修飾つき間接	間接	ストール	ADD.w W0, W1, [W2++] MOV.w [W2], W3
修飾つき間接	修飾つき間接	ストール	ADD.w W0, W1, [W2++] MOV.w [W2++], W3

2.11.2 命令ストール サイクル

命令ストールは、基本的には次の読み出しオペレーションの前に先の書き込みが完了する時間を確保するために、命令の読み出しフェーズの前に 1 命令サイクルの待ち時間を追加することです。割り込みの待ち時間の観点からすれば、ストール サイクルは割り込み検出した命令に続く命令と関連していることに注意する必要があります(つまりストール サイクルは常に命令実行サイクルに先行します)。

RAW データ従属関係が検出されると、PIC24F は命令ストールを開始します。命令ストールの間に、以下のようなイベントが発生します。

1. 実行中の書き込み操作 (先行命令による) は正常に完了します。
2. データ空間は命令ストール後までアドレスされません。
3. プログラムカウンタ (PC) 増分は命令ストール後まで禁止されます。
4. 更なる命令フェッチは命令ストール後まで禁止されます。

2.11.2.1 命令ストール サイクルと割り込み

割り込みが命令ストールを発生させる 2 つの隣接命令と同時発生する場合、起きうる結果は以下の 2 つのうちいずれかです。

1. 割り込みが最初の命令と同時に発生した場合。このような状況では、最初の命令は完了し、第 2 の命令は ISR が完了した後に実行されます。この場合、例外処理の与える時間で最初の命令の書き込みフェーズが完了するため、ストール サイクルは第 2 の命令から消去されます。
2. 割り込みが第 2 の命令と同時に発生した場合。このような状況では、第 2 の命令と追加のストールサイクルが、ISR の前に実行されます。この場合、第 2 の命令に関連するストールサイクルは正常に実行されます。しかし、ストール サイクルは例外処理タイミングに効果的に吸収されます。なぜなら、例外処理では通常 2 命令サイクルが割り込み発生の前に挿入されるためです。

2.11.2.2 命令ストール サイクルとフロー変更命令

CALL と **RCALL** 命令は W15 を使用してスタックに書き込みを行うため、次の命令のソース読み取りが W15 を使用する場合、それに先立って命令ストールが強制挿入されます。

RETFIE と **RETURN** 命令は読み取り操作のみ行うので、次の命令に先立って命令ストールが強制挿入されることはありません。しかし、**RETLW** 命令は最後のサイクル中に W レジスタに書き込みを行うので、ストールが強制挿入されることに留意して下さい。

GOTO とブランチ命令は書き込み操作を行わないため、命令ストールが挿入されることはありません。

2.11.2.3 命令ストールと REPEAT ループ

命令ストールサイクルの追加以外、RAW データ従属関係は **REPEAT** 動作に影響しません。

REPEAT ループ内でプリフェッチした命令は、ループを完了するか例外が発生するまで変更されません。レジスタ従属関係チェックは複数の命令の境界で行われますが、PIC24F は **REPEAT** ループの際、同一命令の操作元と操作先を比較することで効率的に行います。

2.11.2.4 命令ストールとプログラム空間可視化 (PSV)

プログラム空間可視化 (PSV) が有効化され、有効アドレス (EA) が PSV 窓の範囲内に入る場合、読み取りまたは書き込みサイクルはプログラム空間内のアドレスに書き換えられます。プログラム空間からデータにアクセスするには3命令サイクルを要します。

PSV アドレス空間で操作する命令は、他の命令と同様、命令ストールに従います。命令ストールと PSV サイクルが同時に命令の最初で起きますが、これらを一緒にまとめることはできません。ストールが PSV サイクルと一緒に起きたら、ストールサイクルがまず先に実行され、その後 PSV サイクルが実行され、最後に命令サイクルが実行されます。

次のようなコードを考えてみて下さい。

```
ADD    W0, [W1], [W2++]    ; PSV = 1, W1=0x8000, PSVPAG=0xAA
MOV    [W2], [W3]
```

この命令シーケンスを実行するには5命令サイクルが必要です。W1 経由の PSV アクセスのため、2命令サイクルが追加されます。さらに、W2 による RAW データ従属関係を解決するためにもう1命令ストールサイクルが挿入されます。

ストールした命令では、図 2-9 に示すように、最初のサイクルの Q1 の立ち上がりで ROM ラッチが IR に転送され、次に2番目のサイクルの Q3 の立ち上がりでフラッシュのデータが ROM ラッチに転送されます。

2.12 レジスタ マップ

表 2-5 に PIC24F CPU に関連するレジスタのまとめを示します。

表 2-5: コア SFR メモリ マップ (ユーザー モード)

名称	ビット 15	ビット 14	ビット 13	ビット 12	ビット 11	ビット 10	ビット 9	ビット 8	ビット 7	ビット 6	ビット 5	ビット 4	ビット 3	ビット 2	ビット 1	ビット 0	リセット
W0	作業レジスタ 0																0000
W1	作業レジスタ 1																0000
W2	作業レジスタ 2																0000
W3	作業レジスタ 3																0000
W4	作業レジスタ 4																0000
W5	作業レジスタ 5																0000
W6	作業レジスタ 6																0000
W7	作業レジスタ 7																0000
W8	作業レジスタ 8																0000
W9	作業レジスタ 9																0000
W10	作業レジスタ 10																0000
W11	作業レジスタ 11																0000
W12	作業レジスタ 12																0000
W13	作業レジスタ 13																0000
W14	作業レジスタ 14																0000
W15	作業レジスタ 15																0800
SPLIM	Stack Pointer Limit																xxxx
PCL	プログラムカウンタ、下位ワード																0000
PCH	—	—	—	—	—	—	—	—	プログラムカウンタ、上位バイト								0000
TBLPAG	—	—	—	—	—	—	—	—	テーブル ページ アドレス ポインタ								0000
PSVPAG	—	—	—	—	—	—	—	—	プログラム メモリ 空間 可視化 ページ アドレス ポインタ								0000
RCOUNT	繰り返しループカウンタ																xxxx
SR	—	—	—	—	—	—	—	DC	IPL2	IPL1	IPL0	RA	N	OV	Z	C	0000
CORCON	—	—	—	—	—	—	—	—	—	—	—	—	IPL3	PSV	—	—	0000
DISICNT	—	—	割り込み禁止カウンタ														xxxx

凡例: x = リセット時不定、— = 未実装、読むと「0」。リセット時の値は 16 進数で示す。

注 1: 特定のコア レジスタ マップの詳細は個別のデバイスのデータシートを参照して下さい。

2.13 関連するアプリケーションノート

この項では、マニュアルのこの章に関連するアプリケーションノートをリストアップします。これらのアプリケーションノートは、特に PIC24F デバイス ファミリー用に書かれているわけではありませんが、その概念は適切であり、変更あるいは制限事項も考慮に入れて使用可能です。現状、CPU に関連するアプリケーションノートは次の通りです。

タイトル	アプリケーション ノート #
現在関連するアプリケーション ノートはありません。	

注: PIC24F ファミリー デバイスに関するその他のアプリケーション ノートやコード例についてはマイクロチップ ウェブ サイト (www.microchip.com) をご覧下さい。

2.14 改版履歴

リビジョン A (2006 年 4 月)

本文書の初版リリース。

ノート: