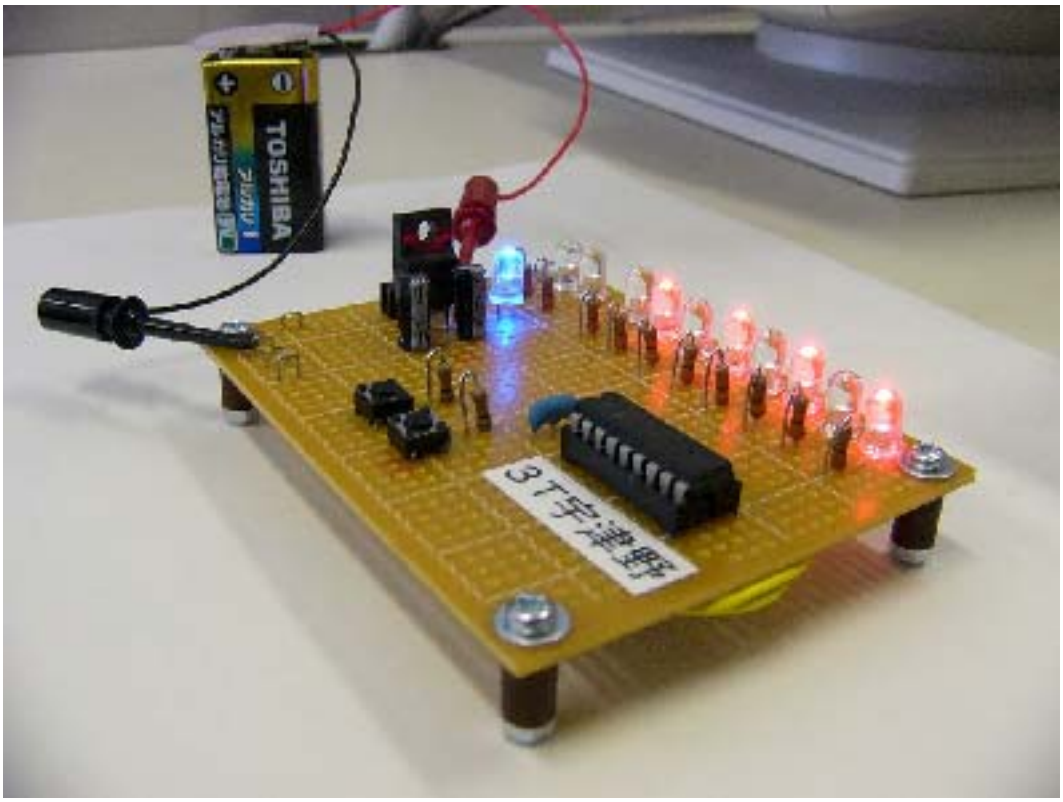
PICアセンブラの基礎



	年		組		番	氏名	

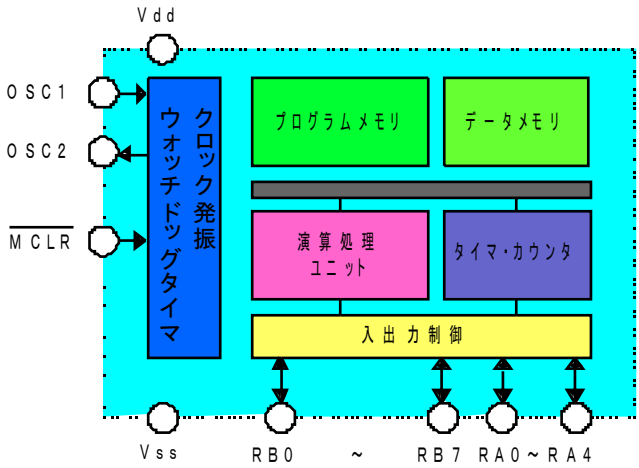
群馬県立利根実業高等学校
工業技術科情報技術コース

１．ＰＩＣとは？

ＰＩＣ(ピック)とは、Peripheral Interface Controllerの頭文字から名付けられ、周辺インターフェイス・コントローラを意味する。米国のMicrochip Technology社により開発されたワンチップマイコン(マイクロコントローラ)製品のシリーズ名称である。用途により数多くの種類が用意されている。

今回使用するＰＩＣ１６Ｆ８４Ａは、外形がＤＩＰ型１８ピンのＩＣである。この中には、演算装置やプログラムを実装するメモリ、タイマなどの周辺装置もすべて内蔵されており、数個の部品や電源を接続すれば、立派なワンチップマイコンとして動作する優れものである。

→



ＰＩＣシリーズは次の３つに大きく分類できる。

①命令長１２ビット：アーキテクチャのロー・レンジ

②命令長１４ビット：アーキテクチャのミッド・レンジ

③命令長１６ビット：アーキテクチャのハイエンド

今回使用するＰＩＣ１６Ｆ８４Ａは中位のミッド・レンジシリーズに属する。

特徴

① 低価格（１個２５０円～５００円程度）

② 低消費電力（動作電圧：２．５～５．５V、電流：３０μＡ～２ｍＡ）

③ 命令数が少ない（３５個の命令のみ）

④ 開発環境ソフト（ＭＰＬＡＢ）がフリーソフトとして無料提供

⑤ PICライタ（５０００円～７０００円程度）

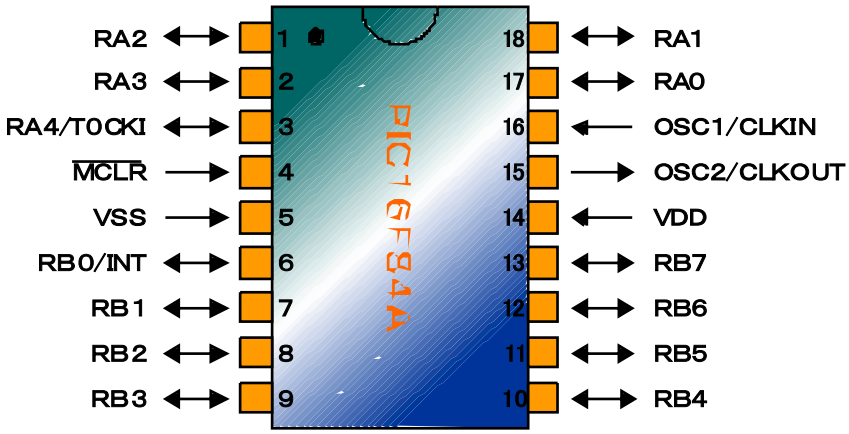
⑥ フラッシュプログラムメモリ搭載で何度もプログラムの書き換えが可能（１０００回程度）

⑦ RISC（縮小セット命令コンピュータ）設計、１命令を１マシン・サイクルで高速処理

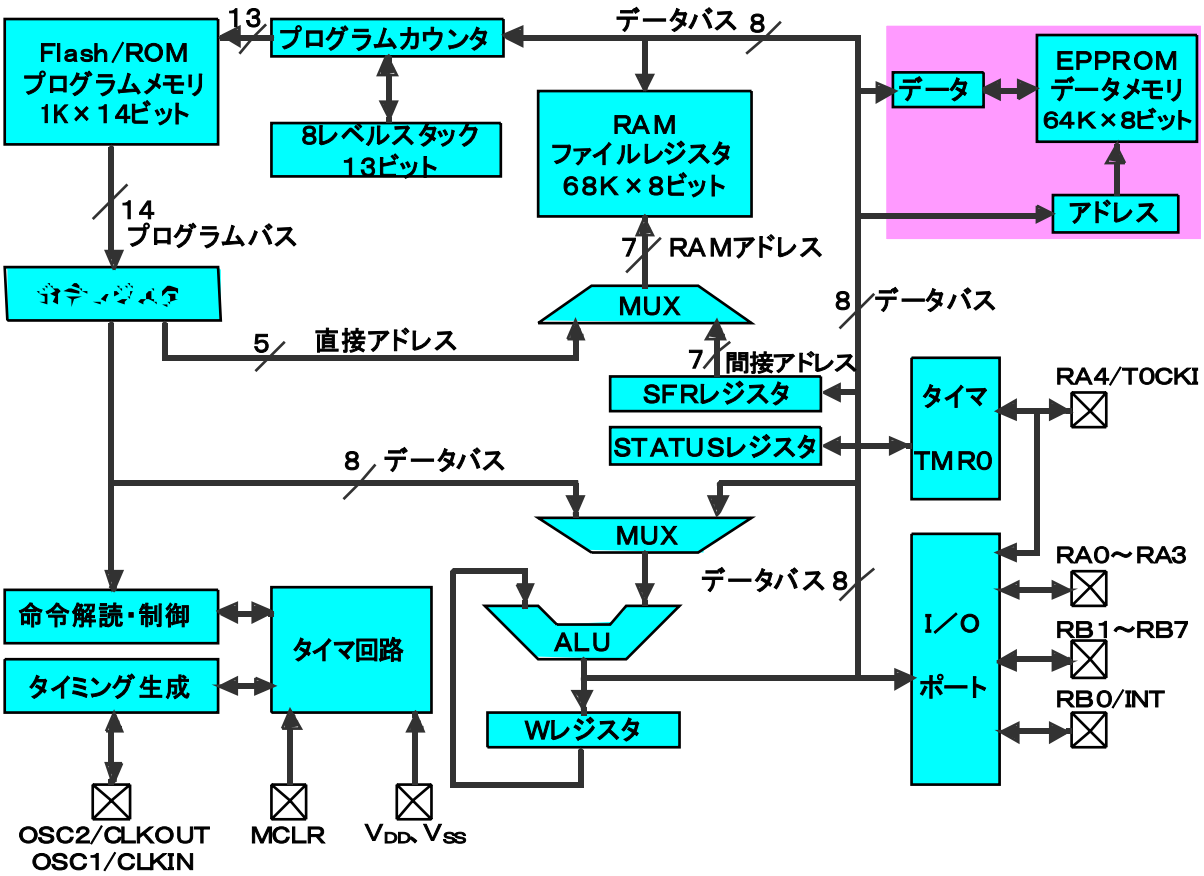
⑧ １命令の実行時間は、使用するＯＳＣ１０ＭＨｚで、０．４μｓ

２．ＰＩＣ１６Ｆ８４Ａの概要

２－１．外観とピン配置



２－２．内部の基本構成

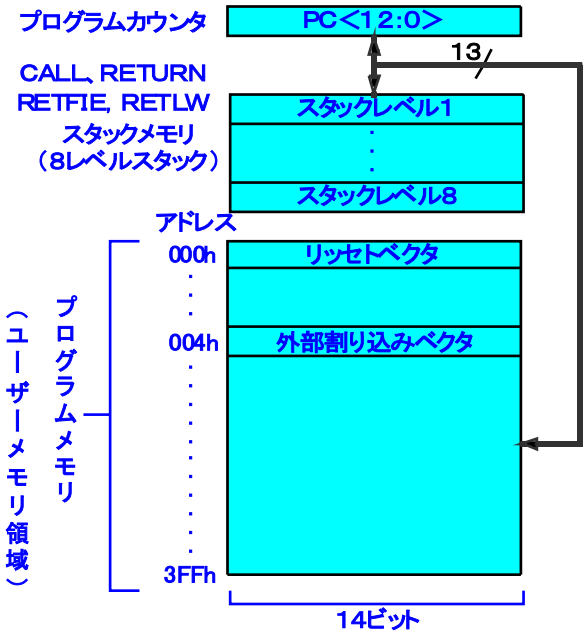


２－３プログラムメモリとスタック配置

①プログラムメモリ

プログラムメモリは、プログラムを格納する専用メモリで、フラッシュメモリで構成される。フラッシュメモリは、電源を切ってもデータが消失しない不揮発性のメモリであり、プログラムライタを使用して何度（１０００回程度といわれている）でも電気的にデータを書きかえることができる。

ＰＩＣ１６Ｆ８４Ａの場合、１命令１４ビット幅で、アドレス000h～3FFh番地までの１ｋワードが格納できる。



②スタックメモリ（８レベルスタック）

サブルーチンや割り込み発生時の戻り番地を記憶しておくメモリで、レベル１～レベル８までである。このため８回のサブルーチンのネスティングが可能となる。

ＣＡＬＬ命実行時や割り込み発生時のＰＣ値に＋１した値をスタックメモリに記憶しておき、ＲＥＴＵＲＮ命令やＲＥＴＦＩＥ命令、ＲＥＴＬＷ命令が実行されたときに、その値をプログラムカウンタに呼び戻し、サブルーチンや割り込み処理から抜け出す。

③プログラムカウンタ（ＰＣ）

プログラムの実行順序を管理するカウンタで、プログラムメモリのアドレスを順次生成する。プログラムカウンタが示すアドレスの順にプログラムは実行される。

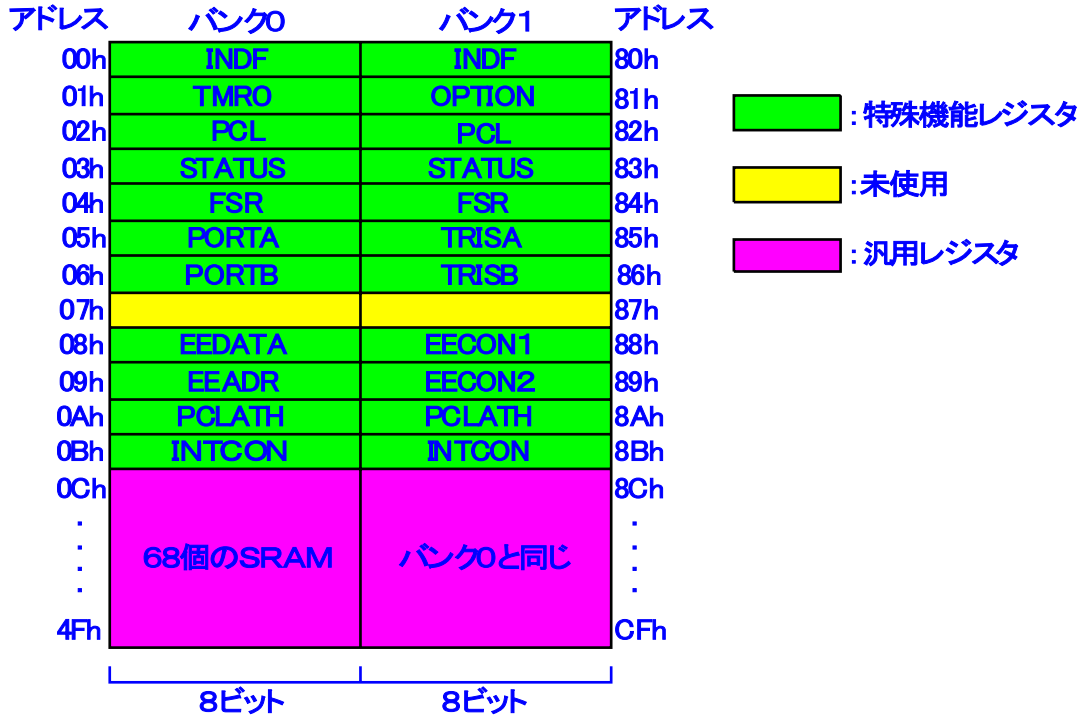
通常はプログラムがアドレス000h番地（リセットベクタ）からスタートするので、ＰＣ＝０の状態からスタートし、ＰＣ＋１→ＰＣとすることによ順次プログラムメモリのアドレスを生成する。

スキップ時にはＰＣ＋２→Ｐとすることにより、１命令ジャンプしたプログラムメモリのアドレスを生成する。

割り込み時はプログラムがアドレス004h番地（外部割り込みベクタ）からスタートするので、ＰＣ＝４の状態からスタートし、ＰＣ＋１→ＰＣとすることにより順次プ

ログラムメモリのアドレスを生成する。

２－４．ファイルレジスタ（データメモリ）



アドレス指定して使用するレジスタである。ＰＩＣ１６Ｆ８４ＡではアドレスOOh～4Fh番地までの８０バイトまでアクセスできる。（ただし、０７ｈ番地は未使用）

また、バンクを切り替えることによりアドレス80h～8Bh番地にもアクセスできる。（ただし、８７ｈ番地は未使用）

アドレス8Ch～CFh番地はバンク０にマップされているため、アクセスできない。

①特殊機能レジスタ（アドレスOOh～0Bh番地および80h～8Bh番地）

ＰＩＣの動作を指定するための特別な役割を持つレジスタである。

よく使う特殊機能レジスタについては、後で説明する。

②汎用レジスタ（アドレス0Ch～4Fh番地）

変数データを扱う汎用データメモリ（ＳＲＡＭ）であり、ユーザーが自由に使用できるレジスタある。

③バンクの切り替えについて

バンク０→アドレスOOh～4Fh番地とアクセス可能

（ただし、アドレス07h番地を除く）

バンク１→アドレス80h～8Bh番地とアクセス可能

（ただし、アドレス87h番地を除く）

バンク０→バンク１の切り替えはアドレス03h番地のSTATUSレジスタのビット5（RP0）に1を書き込むことにより行う。

バンク１→バンク０の切り替えはアドレス83h番地のSTATUSレジスタのビット5（RP0）に0を書き込むことにより行う。

パワーオンリセット後は、STATUSレジスタのビット5（RP0）に0が書き込まれており、バンク０となっている。

２－５．特殊機能レジスタ

よく使う特殊機能レジスタのみを説明する。

２－５－１．ＳＴＡＴＵＳレジスタ

STATUSレジスタの内容

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	
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	
0	0	0	1	1	x	x	x	パワーオンリセット後の内容
						x: 0か1か不定		

よく使うビットについてのみ説明する。

RP1：バンク0とバンク1を切り替える時→0

RP0：RP1ビットが0であることを前提に

RP0ビットを0にした時→バンク0を選択

RP0ビットを1にした時→バンク1を選択

となり、バンク0とバンク1を切り替えることができる。

通常はバンク0が選択されている。

z：ＡＬＵとＷレジスタにより、演算を行った際、

演算の結果が00000000の時→1

演算の結果が00000000以外の時→0

になる。

DC：ＡＬＵとＷレジスタにより、演算を行った際、

演算の結果、4ビット目(ビット3)から桁上げがあった時→1

そうでないとき→0

演算の結果、4ビット目(ビット3)にボローがあった時→0

そうでないとき→1

になる。

C：ＡＬＵとＷレジスタにより、演算を行った際、

演算の結果、最上位ビット(ビット7)から桁上げがあった時→1

そうでないとき→0

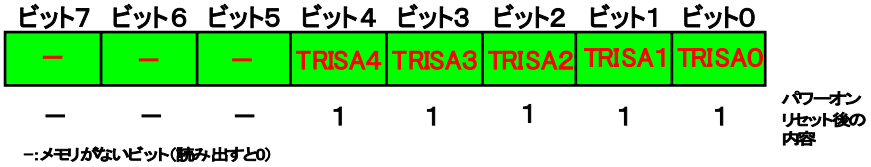
演算の結果、最上位ビット(ビット7)にボローがあった時→0

そうでないとき→1

になる。

２－５－２．ＴＲＩＳＡレジスタ

TRISAレジスタの内容



ポートRA0～RA4を入力端子として使うのか出力端子としてつかうのかによって各ビットに1、0を書き込む。TRISA0～TRISA4はポートRA0～RA4に対応している。

入力端子として使う　→　1

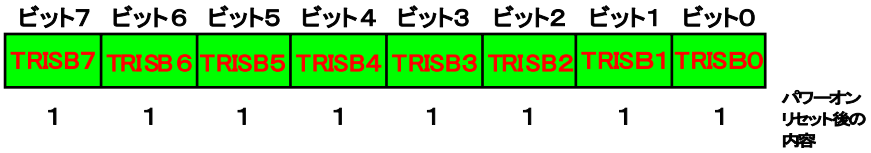
出力端子として使う　→　0

たとえば、TRISAレジスタに00000011を書き込めばポートRA0、RA1は入力端子、ポートRA2、RA3、RA4は出力端子として使うことになる。

ちなみにパワーオンリセット後は、ポートRA0～RA4をすべて入力端子として使う設定となっている。

２－５－３．ＴＲＩＳＡレジスタ

TRISBレジスタの内容



ポートRB0～RB7を入力端子として使うのか出力端子としてつかうのかによって各ビットに1、0を書き込む。TRISB0～TRISB7はポートRB0～RB7に対応している。

入力端子として使う　→　1

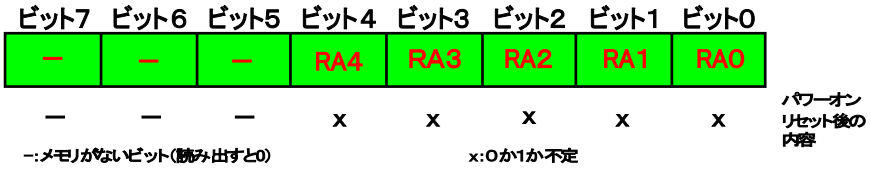
出力端子として使う　→　0

たとえば、TRISBレジスタに11110000を書き込めばポートRB0、RB1、RB2、RB3は出力端子、ポートRB4、RB5、RB6、RB7は出力端子として使うことになる。

ちなみにパワーオンリセット後は、ポートRB0～RB7をすべて入力端子として使う設定となっている。

２－５－４．ＰＯＲＴＡレジスタ

PORTAレジスタの内容



ポートRA0～RA4を入力端子として使っている時は、各ポートからの入力信号の有無に

より1、0が各ビットに書き込まれる。

入力信号あり　→　1

入力信号なし　→　0

ポートRA0～RA4を出力端子として使っている時は、各ビットに書き込んだ1、0により、出力信号が各ポートから送り出される。

1　→　出力信号あり

0　→　出力信号なし

２－５－５．ＰＯＲＴＢレジスタ

PORTBレジスタの内容

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	
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	
x	x	x	x	x	x	x	x	パワーオンリセット後の内容

x:0か1か不定

ポートRB0～RB7を入力端子として使っている時は、各ポートからの入力信号の有無により1、0が各ビットに書き込まれる。

入力信号あり　→　1

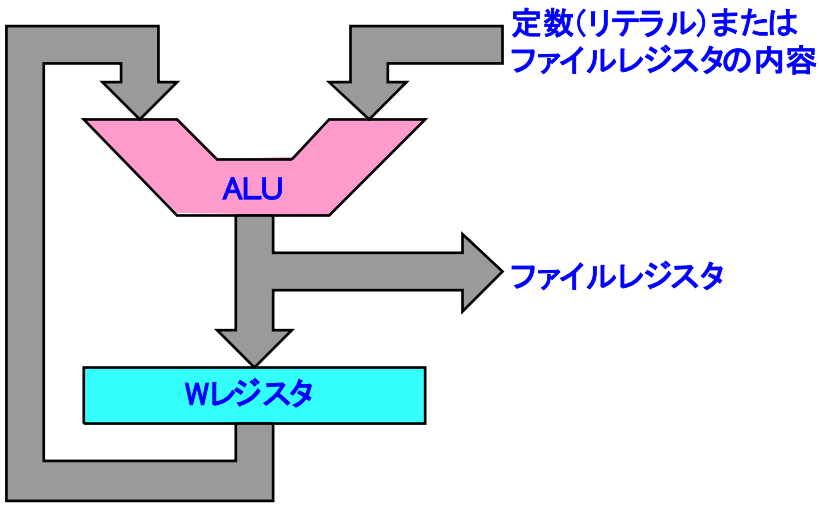
入力信号なし　→　0

ポートRB0～RB7を出力端子として使っている時は、各ビットに書き込んだ1、0により、出力信号が各ポートから送り出される。

1　→　出力信号あり

0　→　出力信号なし

２－６．ＡＬＵとＷレジスタ



①ＡＬＵ（Arithmetic Logic Unit）

算術論理演算装置である。各命令の指示に従って、各種レジスタやＷレジスタの内容との演算が行われる。Ｗレジスタと協調動作する。

②Ｗレジスタ（ワーキングレジスタ）

演算結果などを一時的に格納する演算用レジスタである。

各種レジスタやＡＬＵ、Ｗレジスタはデータが取り出された後も別のデータが格納されるまで前のデータを保持している。

３．ＰＩＣ１６Ｆ８４Ａのアセンブラ命令

３－１．命令の種類

ＰＩＣ１６Ｆ８４Ａは次の３５個の命令を持つ。（詳細は一覧表参照）

転送命令		MOVF、MOVWF、MOVLW
演算命令	算術演算	ADDWF、ADDLW、SUBWF、SUBLW、INCF、DECF
	論理演算	COMF、ANDWF、ANDLW、IORWF、IORLW、XORWF XORLW、
	ローテイト演算	RLF、RRF
	ビット演算	BCF、BSF
	その他	CLRF、CLRW、SWAPF
分岐命令	条件分岐	INCFSZ、DECFSZ、BTFSC、BTFSS
	無条件分岐	GOTO
	サブルーチン命令	CALL、RETFIE、RETLW、RETURN
その他		NOP、CLRWDT、SLEEP

３－２．擬似命令

アセンブラに対する制御命令で、アセンブル時に機械語に変換されない。

LIST命令：使用するマイコンの種類を指定する。
INCLUDE命令：指定したファイルとソースプログラムを取り込む。
_ _CONFIG命令：マイコンで使用する特殊機能を設定する。
EQU命令：数値やアドレス値をラベルに割り当てる。
ORG命令：機械語を格納するプログラムメモリの先頭番地を指定す。
END命令：ソースプログラムの終了を示す。

３－２．命令の形式

アセンブラ命令は、ラベル、ニーモニック、オペランド、コメントで構成される。

①ラベル：ユーザーが必要に応じてファイルレジスタやプログラムメモリに対するアドレス(番地)の代わりとなるラベルを記述する。

記述ルール

・行の先頭より記述する。

・英文字またはアンダーバー(_)で始まる半角３２文字以内の英数文字で記述する。

②ニーモニック：命令を記述する。

記述ルール

・ラベルとの間に１文字以上のスペースかコロン(:)を記述する。

・ラベルを省略した場合は､行の先頭から１文字以上のスペースを記述する。

③オペランド：命令の対象となるファイルレジスタのアドレス(f)、結果格納先指定子(d)、８ビットファイルレジスタ内のビット番号(b)、定数(k)を記述する。命令によっては必要ない場合もある。

記述ルール

・f：ファイルレジスタのアドレス00h～4Fhおよび80h～8Bhまたはレジスタ名を記述する。

・d：結果の格納先がWレジスタの場合0を記述、ファイルレジスタの場合は1を記述する。

・b：８ビットファイルレジスタ内のビット番号0～7を記述する。

・k：８ビットで表現できる定数（リテラル）0～255を記述する。

④コメント(注釈)：必要に応じてコメント(注釈)を記述する。

記述ルール

・セミコロン(;)を記述すると、それ以降の記述はすべて命令とは無関係となり､アセンブル時は無視される｡命令の説明などを記述するときに使う。

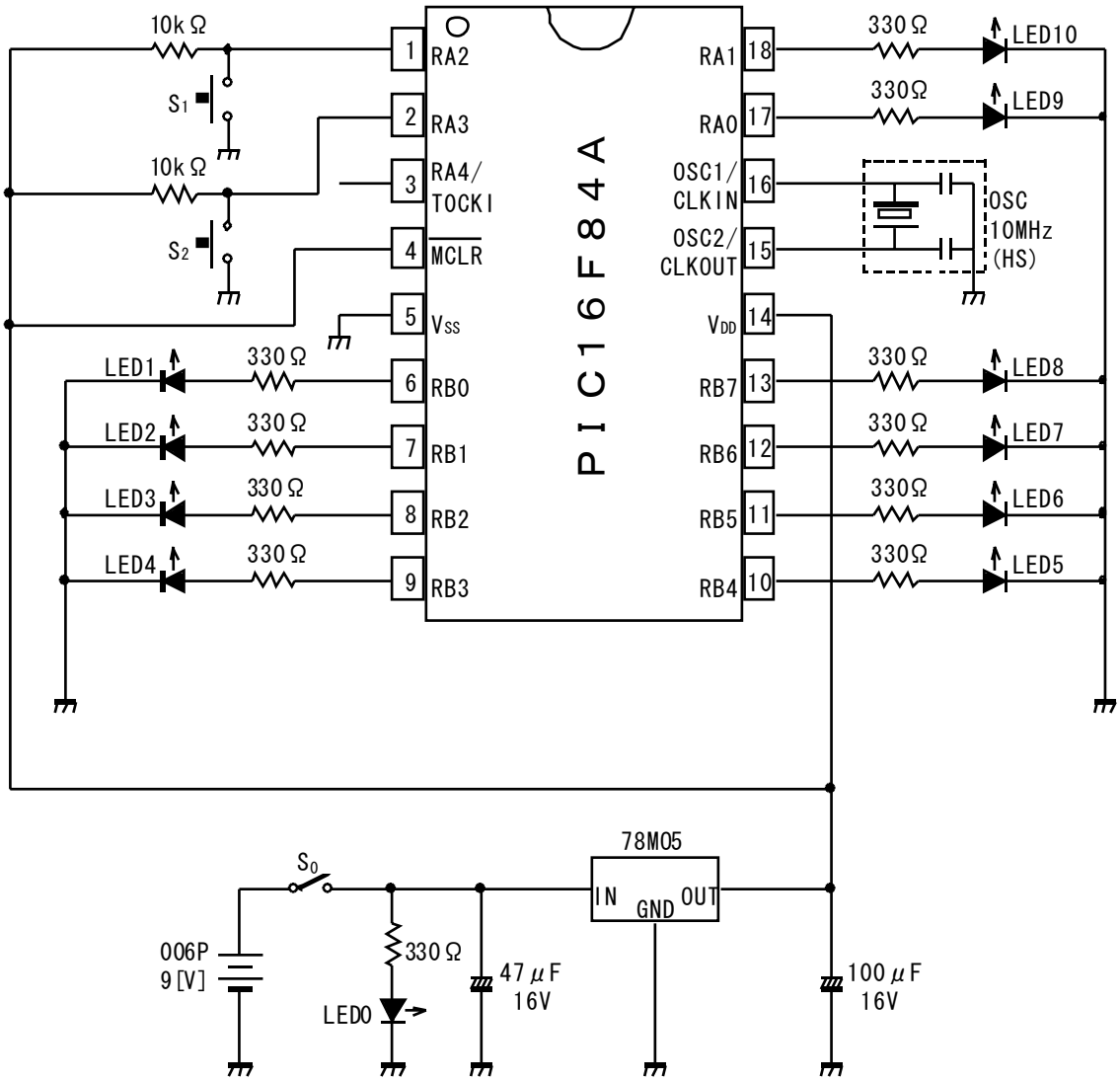
３－４．数値の記述

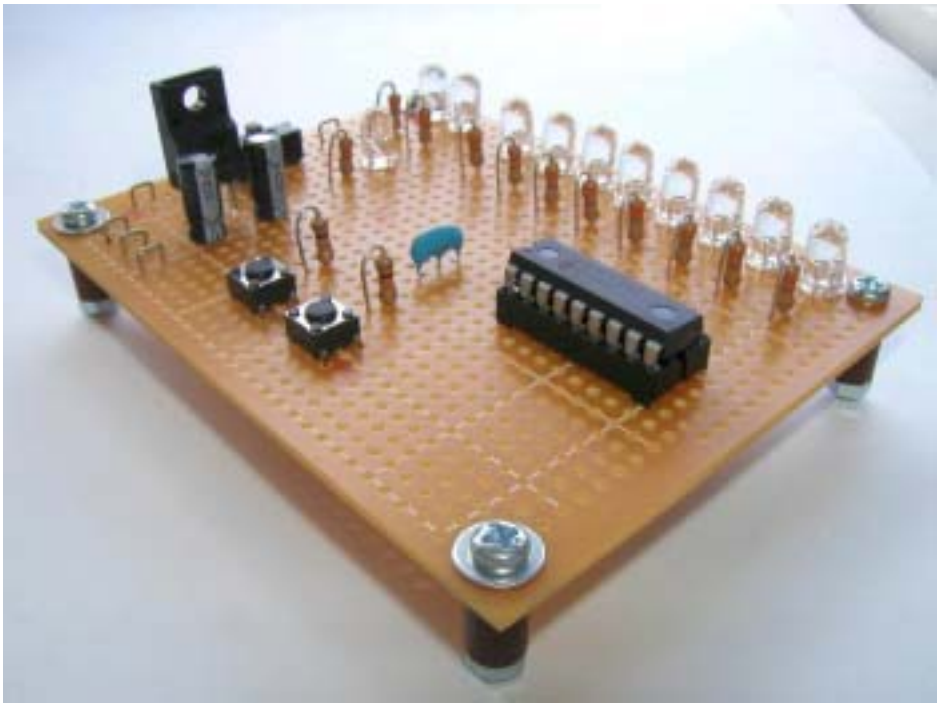
プログラムレジスタやファイルレジスタのアドレス値、定数をプログラム中で記述する時には、次に示す書式のいずれかとする。通常16進数で記述されることが多い。

記述形式	書式	記述例(10進数の10を記述する場合)
10進数	D'値'	D'10'
2進数	B'値'	B'00001010'
8進数	O'値'	O'12'
16進数	H'値'	H'0A'
	値H	0AH
	0x値	0x0A

今回は、アドレス値と定数を区別するため、アドレス値には値H、定数にはH'値'を用いる。

３．制作したＬＥＤ点滅制御ボード





５．簡単なプログラムの理解

ＬＥＤを右図のように点灯させるプログラムを作ろう。

５－１．フローチャートを作ろう。

５－２．プログラムを書いてみよう。

ラベル	ニーモニック	オペランド	コメント

５－３．各命令を理解しよう。

① LIST P=PIC16F84A

② INCLUDE "P16F84A.INC"

③ __CONFIG _HS_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF

④ ORG 0

⑤ BSF STATUS,RP0

⑥ CLRF TRISB

⑦ BCF STATUS,RP0

⑧ CLRF PORTB

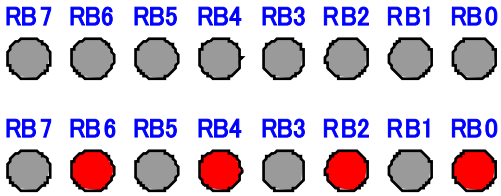
⑨ MOVLW H'55'

⑩ MOVWF PORTB

⑪ END

５－４．タイマルーチンってなんだ？

先に作ったプログラムをもとにタイマルーチンを使って右の２つの点灯状態を１秒間隔で繰り返すプログラムを作ろう。



タイマルーチンってなんだ？

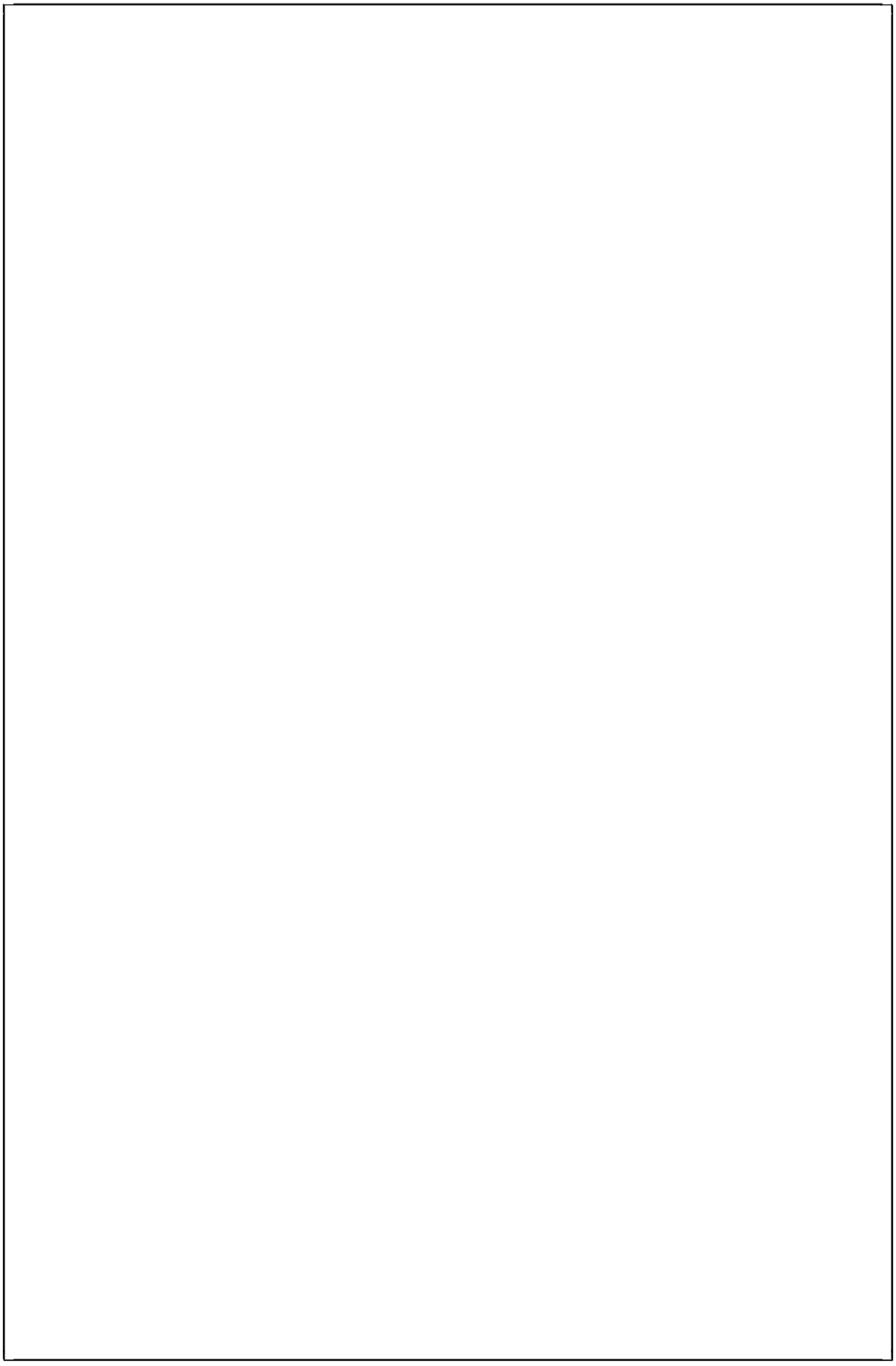
```
CNT1    EQU     0CH
CNT2    EOU     0DH
CNT3    EQU     0EH

TIMER1  MOVLW   H'3E'
        MOVWF   CNT1
LOOP1   NOP
        DECFSZ  CNT1,1
        GOTO    LOOP1
        RETURN

TIMER2  MOVLW   H'64'
        MOVWF   CNT2
LOOP2   CALL    TIMER1
        DECFSZ  CNT2,1
        GOTO    LOOP2
        RETURN

TIMER3  MOVLW   H'64'
        MOVWF   CNT3
LOOP3   CALL    TIMER2
        DECFSZ  CNT3,1
        GOTO    LOOP3
        RETURN
```

0.1ms(0.0001s)タイマ

10ms(0.01s)タイマ

1sタイマ

フローチャート



プログラム

```
LIST  P=PIC16F84A
INCLUDE       "P16F84A.INC"
__CONFIG _HS_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF
```

ラベル	ニーモニック	オペランド	コメント

６．プログラミングに挑戦

６－１．下図のようにＬＥＤを点灯させるプログラムを作ろう。

フローチャート

プログラム

```
LIST  P=PIC16F84A
INCLUDE        "P16F84A.INC"
__CONFIG _HS_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF
```

ラベル	ニーモニック	オペランド	コメント

６－２．ポートRA0とRA1も出力ポートに設定し、下図のようにＬＥＤを点灯させるプログラムを作ろう。

ポートRA0とRA1を出力端子として使うにはどうしたらいいのかな？考えてみよう。

フローチャート

プログラム

```
LIST  P=PIC16F84A
INCLUDE       "P16F84A.INC"
__CONFIG _HS_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF
```

ラベル	ニーモニック	オペランド	コメント

６－３．下図のようにＬＥＤを点灯させるプログラムを作ろう。

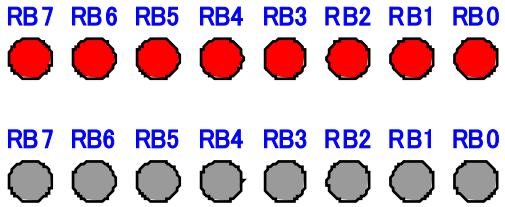
フローチャート

プログラム

```
LIST  P=PIC16F84A
INCLUDE        "P16F84A.INC"
__CONFIG _HS_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF
```

ラベル	ニーモニック	オペランド	コメント

６－４．全部点灯と全部消灯を１秒間隔で繰り返すプログラムを作ろう。



フローチャート

プログラム

```
LIST  P=PIC16F84A
INCLUDE       "P16F84A.INC"
__CONFIG _HS_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF
```

ラベル	ニーモニック	オペランド	コメント

６－５．自分でどのように点灯や点滅させたいかを決め、それを実現するプログラムを作ろう。

どのように点灯や点滅させたいかを考えてみよう。

フローチャート

プログラム

```
LIST  P=PIC16F84A
INCLUDE       "P16F84A.INC"
__CONFIG _HS_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF
```

ラベル	ニーモニック	オペランド	コメント