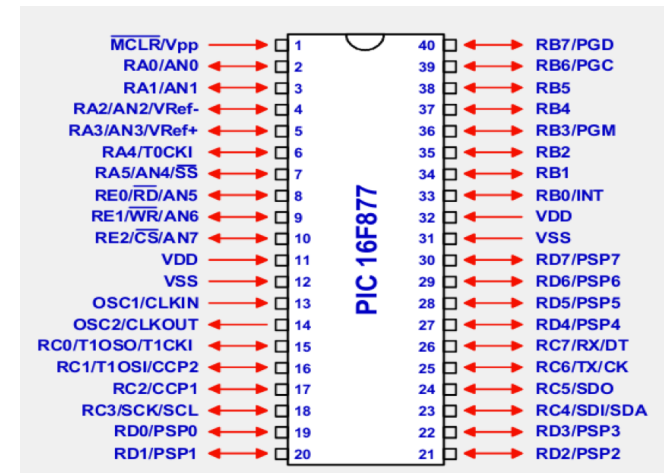


le PIC16f877

cours n°2
LI326

Boitier pic16f877 40 broches



Architecture

Caractéristiques pic16f877

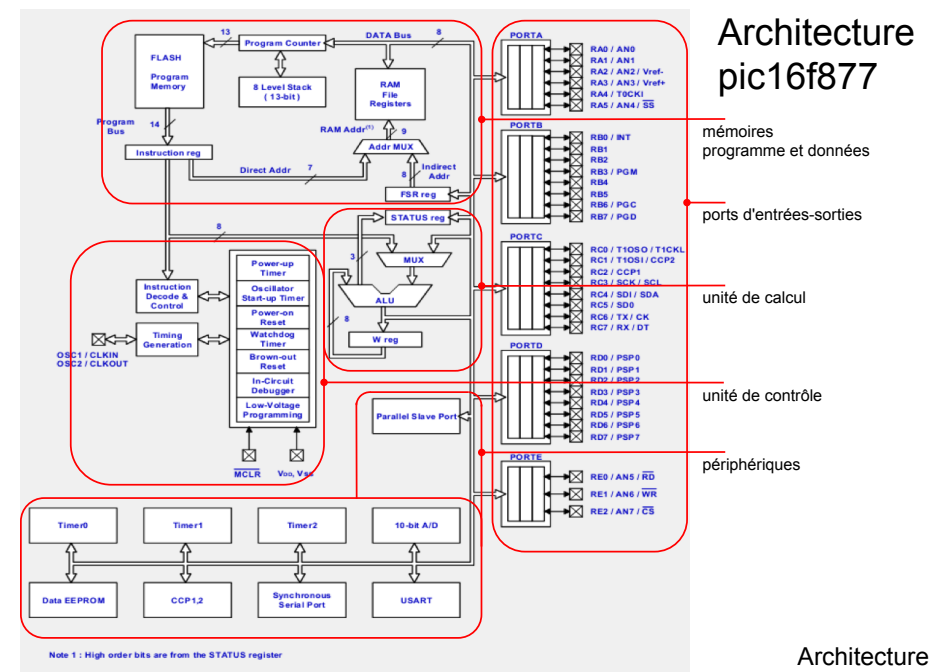
Fonctionne à **20 MHz** maximum.

- mais utilise 4 cycles par instruction donc 5 millions d'instructions par seconde ou **200 nanosecondes par instruction**

Possède :

- 35 instructions (composant RISC),
- 8Ko de mémoire Flash interne pour le programme,
- 368 octets de RAM, ce sont plutôt des registres (FILE)
- 256 octets de d'EEProm,
- 3 compteurs / timer de 8 ou 16 bits (PWM)
- 1 Watchdog,
- 8 entrées analogiques 10bits / comparateur de tension
- 1 port série (RS232), 1 port I2C / SPI
- 15 sources d'interruption,
- 33 entrées/sorties numériques configurables individuellement, disposés en 5 ports nommés de A à E,
- un mode SLEEP.

Architecture



Architecture pic16f877

mémoires programme et données

ports d'entrées-sorties

unité de calcul

unité de contrôle

périphériques

Architecture

File Address	File Address	File Address	File Address
Indirect addr.(1)	Indirect addr.(1)	Indirect addr.(1)	Indirect addr.(1)
00h	80h	100h	180h
TMR0	OPTION_REG	TMR0	OPTION_REG
01h	81h	101h	181h
PCL	PCL	PCL	PCL
02h	82h	102h	182h
STATUS	STATUS	STATUS	STATUS
03h	83h	103h	183h
FSR	FSR	FSR	FSR
04h	84h	104h	184h
PORTA	TRISA		185h
05h	85h	105h	
PORTB	TRISB	PORTB	TRISB
06h	86h	106h	186h
PORTC	TRISC		187h
07h	87h	107h	
PORTD(1)	TRISD(1)		188h
08h	88h	108h	
PORTE(1)	TRISE(1)		189h
09h	89h	109h	
PCLATH	PCLATH	PCLATH	PCLATH
0Ah	8Ah	10Ah	18Ah
INTCON	INTCON	INTCON	INTCON
0Bh	8Bh	10Bh	18Bh
PIR1	PIE1	EEDATA	ECON1
0Ch	8Ch	10Ch	18Ch
PIR2	PIE2	EEDATH	ECON2
0Dh	8Dh	10Dh	18Dh
TMR1L	PCON	EEDATH	Reserved(2)
0Eh	8Eh	10Eh	18Eh
TMR1H		EEDATH	Reserved(2)
0Fh	8Fh	10Fh	18Fh
T1CON			190h
10h	90h	110h	191h
TMR2	SSPCON2		192h
11h	91h	111h	193h
T2CON	PR2		194h
12h	92h	112h	195h
SSPBUF	SSPADD		196h
13h	93h	113h	197h
SSPCON	SSPSTAT		198h
14h	94h	114h	199h
CCPR1L			200h
15h	95h	115h	
CCPR1H			201h
16h	96h	116h	
CCP1CON			202h
17h	97h	117h	
RCSTA	TXSTA		203h
18h	98h	118h	
TXREG	SPBRG		204h
19h	99h	119h	
1Ah			205h
RCREG			206h
1Bh	9Bh	11Bh	
CCPR2L			207h
1Ch	9Ch	11Ch	
CCPR2H			208h
1Dh	9Dh	11Dh	
CC2CON			209h
1Eh	9Eh	11Eh	
ADRESH	ADRESL		210h
1Fh	9Fh	11Fh	
ADCON0	ADCON1		211h
20h	A0h	120h	
			1A0h
General Purpose Register 96 Bytes	General Purpose Register 80 Bytes	General Purpose Register 16 Bytes	General Purpose Register 16 Bytes
Bank 0	Bank 1	Bank 2	Bank 3
7Fh	EFh	16Fh	1EFh
	FDh	170h	1F0h
	FFh	17Fh	1FFh
		accesses 70h - 7Fh	

Données Registres

- 4 bancs
- SFR
Special File Register pour les périphériques
- GPR
General Purpose Reg pour les programmes
- certains registres adresses multiples
\$0 = \$80 = \$100 = \$180
\$70 = \$F0 = \$170 = \$1F0

Mémoire

Assembleur à une adresse

Cas général d'une instruction (dit 3 adresses)
register_dest = operande1 OPER operande2

Pour le PIC

une des operandes est W
Le registre de destination est une des operandes

exemple MIPS :

add \$rd, \$rs, \$rt c.-à-d. \$rd = \$rs + \$rs (p.ex. add \$6,\$4,\$3)

en PIC (notez que la destination est à la fin) :

addwf 23,1 reg23 = reg23 + W
addwf 23,0 W = reg23 + W

Pour certaines instructions on n'a qu'une seule opérande

Assembleur

Assembleur PIC

Un programme en assembleur comporte une *instruction* par ligne.
Une ligne se découpe en quatre colonnes

1. la 1ère colonne commence en tout début de ligne et contient au plus un mot appelé symbole ou étiquette associé à l'adresse de la case mémoire de l'instruction ou de la donnée courante. On peut aussi attribuer une valeur quelconque à une étiquette.
2. la 2ème colonne commence après un espace ou une tabulation. elle contient une instruction, une macro-instruction ou une directive.
3. la 3ème colonne est séparée par un espace ou une tabulation contient les arguments séparés par des virgules de la 2ème colonne.
4. la 4ème colonne commence par un ; s'achève au retour chariot contient un commentaire.

Table 29-1: Midrange Instruction Set

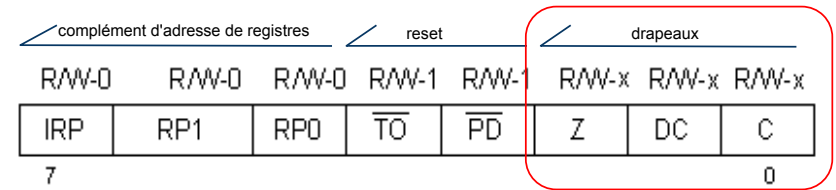
Mnemonic, Operands	Description	Cycles	14-Bit Instruction Word		Status Affected	Notes
			MSb	LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d Add W and f	1	00	0111	0000	C,DC,Z
ANDWF	f, d AND W with f	1	00	0101	0000	Z
CLRF	f Clear f	1	00	0001	1111	Z
CLRW	- Clear W	1	00	0001	0000	Z
COMF	f, d Complement f	1	00	1001	0000	Z
DECf	f, d Decrement f	1	00	0011	0000	Z
DECFSZ	f, d Decrement f, Skip if 0	1(2)	00	1011	0000	Z
INCF	f, d Increment f	1	00	1010	0000	Z
INCFSZ	f, d Increment f, Skip if 0	1(2)	00	1111	0000	Z
IORWF	f, d Inclusive OR W with f	1	00	0100	0000	Z
MOVF	f, d Move f	1	00	1000	0000	Z
MOVWF	f Move W to f	1	00	0000	1111	Z
NOF	- No Operation	1	00	0000	0000	
RLF	f, d Rotate Left f through Carry	1	00	1101	0000	C
RRF	f, d Rotate Right f through Carry	1	00	1100	0000	C
SUBWF	f, d Subtract W from f	1	00	0010	0000	C,DC,Z
SWAPF	f, d Swap nibbles in f	1	00	1110	0000	Z
XORWF	f, d Exclusive OR W with f	1	00	0110	0000	Z
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b Bit Clear f	1	01	00bb	bfff	1,2
BSF	f, b Bit Set f	1	01	01bb	bfff	1,2
BTFSZ	f, b Bit Test f, Skip if Clear	1(2)	01	10bb	bfff	3
BTFSZ	f, b Bit Test f, Skip if Set	1(2)	01	11bb	bfff	3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k Add literal and W	1	11	111x	kkkk	C,DC,Z
ANDLW	k AND literal with W	1	11	1001	kkkk	Z
CALL	k Call subroutine	2	10	0kjk	kkkk	
CLRWDT	- Clear Watchdog Timer	1	00	0000	0110	TOPD
GOTO	k Go to address	2	10	1kjk	kkkk	
IORLW	k Inclusive OR literal with W	1	11	1000	kkkk	Z
MOVLW	k Move literal to W	1	11	00xx	kkkk	
RETFIE	- Return from interrupt	2	00	0000	0000	1001
RETLW	k Return with literal in W	2	11	01xx	kkkk	
RETURN	- Return from Subroutine	2	00	0000	0000	1000
SLEEP	- Go into standby mode	1	00	0000	0110	0011
SUBLW	k Subtract W from literal	1	11	110x	kkkk	C,DC,Z
XORLW	k Exclusive OR literal with W	1	11	1010	kkkk	Z

Assembleur

Exercices pour comprendre

\$20 = 0	clrf 0x20
\$20 = 1	clrf 0x20 incf 0x20,F
\$20 = \$20 + 1	incf 0x20
\$20 = \$20 + \$21	movf 0x21, W addwf 0x20,F
\$20 = \$21 - 2	movlw 2 addwf 0x21,W movwf 0x20
\$20 = \$20 - \$21	movf 0x20,W subwf 0x21,F

Registre status



R = Readable bit W = Writable bit

U = Unimplemented bit, read as '00' - n = Value at power-on reset

Le registre status est modifié par la plupart des instructions

Assembleur

Branchements (principe)

Comparaisons

goto adresse
branch_if_cond operande1, operande2, adresse

Pour le PIC

goto adresse (ATTENTION l'adresse n'est pas complète)
skip_if_notcond
goto adresse

exemple MIPS :

beq \$4,\$5,label

en PIC :

movf 23,0
xorwf 22,0
btfs status,2
goto label

Branchements (exemples)

Égalité registre/registre et registre/immédiat

```
; si (REG1 == REG2) goto LAB
; beq REG1, REG2, LAB
movf REG1, W
xorwf REG2, W
btfs STATUS, Z
goto LAB
```

```
; si (REG1 != REG2) goto LAB
; bne REG1, REG2, LAB
movf REG1, W
xorwf REG2, W
btfs STATUS, Z
goto LAB
```

```
; si (REG == IMM) goto LAB
; beq REG, IMM, LAB
movlw IMM
xorwf REG, W
btfs STATUS, Z
goto LAB
```

```
; si (REG1 != IMM) goto LAB
; bne REG1, IMM, LAB
movlw IMM
xorwf REG, W
btfs STATUS, Z
goto LAB
```

Comparaisons registre/registre sur des nombres non signés

```
; si (REG1 < REG2) goto LAB
; bit REG1, REG2, LAB
movf REG2, W
subwf REG1, W
btfs STATUS, C
goto LAB
```

```
; si (REG1 > REG2) goto LAB
; bgt REG1, REG2, LAB
movf REG1, W
subwf REG2, W
btfs STATUS, C
goto LAB
```

```
; si (REG1 <= REG2) goto LAB
; ble REG1, REG2, LAB
movf REG1, W
subwf REG2, W
btfs STATUS, C
goto LAB
```

```
; si (REG1 >= REG2) goto LAB
; bge REG1, REG2, LAB
movf REG2, W
subwf REG1, W
btfs STATUS, C
goto LAB
```

Assembleur

Exercices pour comprendre

\$20 = SOMME de 0 à 100

```

movlw 100
movwfw 0x21
clrf 0x20
do
    movfw 0x21,W
    addwfw 0x20,F
    decfsz 0x21
    goto do
    
```

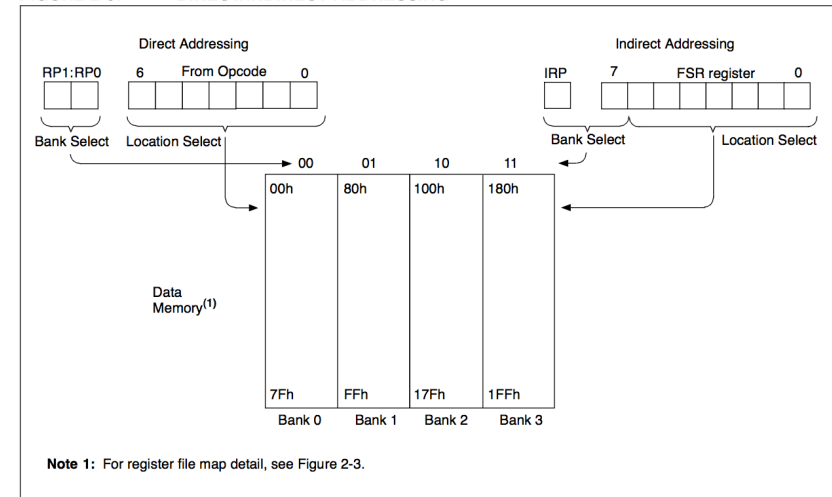
\$20 = PGCD (\$70, \$71)

```

goto L3
L0 si $70 > $71 goto L2
L1 $70 = $70 - $71
    goto L3
L2 $71 = $71 - $70
L3 si $70 != $71 goto L0
    
```

Calcul des operandes

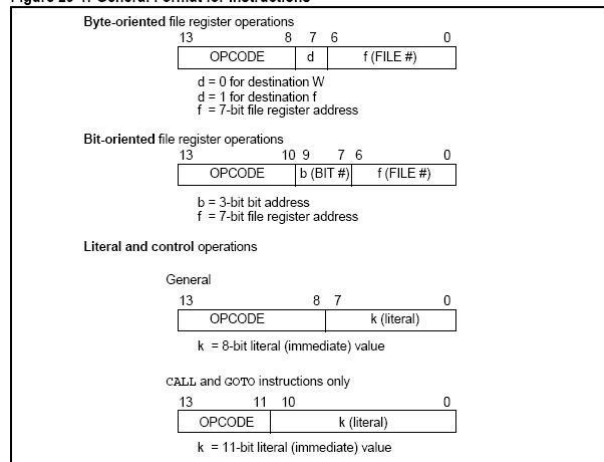
FIGURE 2-6: DIRECT/INDIRECT ADDRESSING



Assembleur

Codage des instructions

Figure 29-1: General Format for Instructions

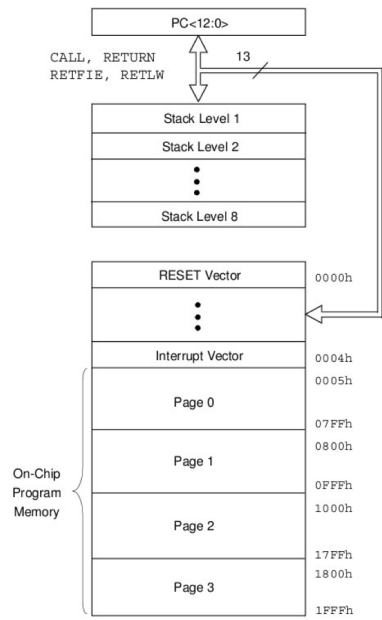


Il manque 2 bits pour adresser les registres 7 au lieu de 9
il manque 2 bits pour fabriquer les adresses d'instructions

Accès aux registres

- GPR standard
 - Ecrire dans les bits RP0, RP1 du registre STATUS
 - (bcf | bsf) STATUS, (rp0 | rp1)
 - ou banksel REGISTRE
 - Accéder au registre
- GPR répliqué dans les 4 bancs
 - Accéder au registre
- GPR répliqué dans les 2 bancs
 - Ecrire dans les bits RP0 du registre STATUS
 - (bcf | bsf) STATUS, rp0
 - ou banksel REGISTRE
 - Accéder au registre

File Address	File Address	File Address	File Address
Indirect addr. ⁽¹⁾ 00h	Indirect addr. ⁽¹⁾ 80h	Indirect addr. ⁽¹⁾ 100h	Indirect addr. ⁽¹⁾ 180h
OPTION_REG 01h	OPTION_REG 81h	TMR0 101h	OPTION_REG 181h
PCL 02h	PCL 82h	PCL 102h	PCL 182h
STATUS 03h	STATUS 83h	STATUS 103h	STATUS 183h
FSR 04h	FSR 84h	FSR 104h	FSR 184h
PORTA 05h	TRISA 85h	PORTB 105h	TRISA 185h
PORTB 06h	TRISB 86h	PORTC 106h	TRISB 186h
PORTC 07h	TRISC 87h	PORTD 107h	TRISC 187h
PORTD 08h	TRISD 88h	PORTF 108h	TRISD 188h
PORTF 09h	TRISF 89h	PORTG 109h	TRISF 189h
PCLATH 0Ah	PCLATH 8Ah	PCLATH 10Ah	PCLATH 18Ah
INTCON 0Bh	INTCON 8Bh	INTCON 10Bh	INTCON 18Bh
PIE1 0Ch	PIE1 8Ch	EEDATA 10Ch	EEDATA 18Ch
PIE2 0Dh	PIE2 8Dh	EEDADR 10Dh	EEDADR 18Dh
TMR1L 0Eh	PCON 8Eh	EEDATH 10Eh	Reserved ⁽²⁾ 18Eh
TMR1H 0Fh	PCON 8Fh	EEDATH 10Fh	Reserved ⁽²⁾ 18Fh
TCON 10h	SSPCON2 91h	110h	190h
TMRP 11h	SSPCON2 91h	111h	191h
TCON 12h	SSPCON2 92h	112h	192h
SSPBUF 13h	SSPAD 93h	113h	193h
SSPCON 14h	SSPSTAT 94h	114h	194h
CCPR1L 15h	95h	115h	195h
CCPR1H 16h	96h	116h	196h
CCP1CON 17h	97h	117h	197h
RCSTA 18h	TXSTA 98h	118h	198h
TXREG 19h	SPBRG 99h	119h	199h
RCREG 1Ah	9Ah	11Ah	19Ah
CCP2L 1Bh	9Bh	11Bh	19Bh
CCP2H 1Ch	9Ch	11Ch	19Ch
CCP2CON 1Dh	9Dh	11Dh	19Dh
ADRESH 1Eh	ADRESL 9Eh	11Eh	19Eh
ADCON0 1Fh	ADCON1 9Fh	11Fh	19Fh
20h	Adh	120h	1A0h
General Purpose Register 86 Bytes	accesses 70h-7Fh	General Purpose Register 80 Bytes	accesses 1EFh
General Purpose Register 80 Bytes	accesses F0h	General Purpose Register 80 Bytes	accesses 1F0h
General Purpose Register 80 Bytes	accesses 70h-7Fh	General Purpose Register 80 Bytes	accesses 1FFh



Programme

- 4 pages de programmes
- Pile hardware pour les adresses
- 2 adresses pour le traitement des événements matériels
 - reset
 - interruption

Mémoire

Saut à une instruction quelconque

- GOTO
 - Ecrire dans les bits de PCLATH le numéro du banc
 - goto address
- Exemple
 - goto 0x1000


```
bcfPCLATH,3
bsfPCLATH,4
goto 0x1000
```

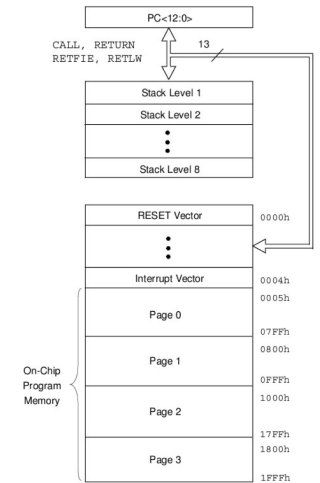
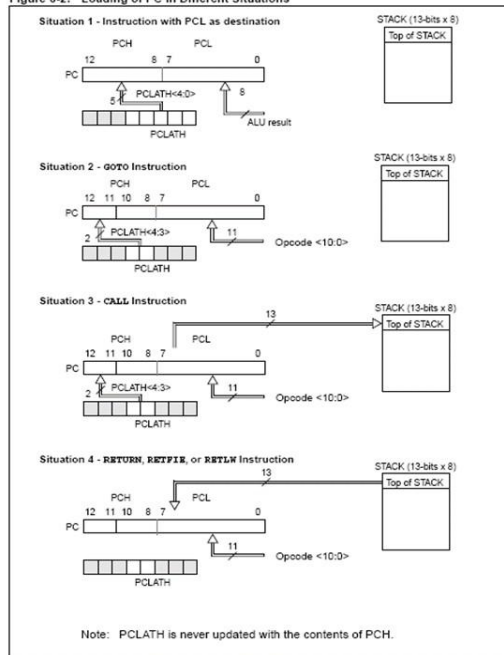


Figure 6-2: Loading of PC in Different Situations



Courtesy Microchip Technology Inc.

Calcul du PC

- Le PC est sur 13 bits
- Les 8 bits de poids faible sont accessibles en lecture/écritures
- Les 5 bits de poids fort sont accessibles en écriture seule

Assembleur

Hello World
(façon microcontrôleur)

GPIO principe

Les ports à usage général sont des entrées / sorties (GPIO)

Pour chaque broche, nous avons deux bits de commande

- 1 bit pour définir de sens : entrée (1), sortie (0)
- 1 bit pour lire la donnée (si entrée) ou l'écrire (si sortie)

Les registres de direction sont nommés TRISx

Les registres de données sont nommés PORTx

Par exemple :

TRISD / PORTD sur 8 bits

Directives assembleur

LIST <expression> [, <expression>] *

n=nnn Sets the number of lines per page
p= <symbol> Sets the current processor
r= [oct | dec | hex] Sets the radix

INCLUDE "p16f877.inc"

déclaration des noms de registres
macro-définitions
définition de fonctions

ORG <expression>

définition de l'adresse à partir de laquelle
le programme va être placé.

<symbol> EQU <expression> (cf CONSTANT)
associe définitivement <expression> à <symbol>

<symbol> SET <expression> (cf VARIABLE)
associe temporairement <expression> à <symbol>

__CONFIG <expression>

Définition du registre de configuration du pic
_CP_OFF : le code peut être relu
_WDT_OFF: pas de timer watch dog

END

fin du programme

p16f877.inc (extrait)

```

W      EQU  H'0000'
F      EQU  H'0001'

INDF   EQU  H'0000'
TMRO   EQU  H'0001'
PCL    EQU  H'0002'
STATUS EQU  H'0003'
FSR    EQU  H'0004'
PORTA  EQU  H'0005'
PORTB  EQU  H'0006'
PORTC  EQU  H'0007'

C      EQU  H'0000'
DC     EQU  H'0001'
Z      EQU  H'0002'
NOT_PD EQU  H'0003'
NOT_TO EQU  H'0004'
IRP    EQU  H'0007'

RP0    EQU  H'0005'
RP1    EQU  H'0006'
    
```

Hello World !

```

list    p=16f877      ; définit le processeur cible
include "p16f877.inc" ; déclaration des noms de registres
    
```

```

__CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _HS_OSC & _LVP_OFF
    
```

```

ORG    0
    
```

initialisation

```

bcf    STATUS,RP1    ; STATUS(RP1) <- 0
bsf    STATUS,RP0    ; STATUS(RP0) <- 1 permet de désigner le banc 1
bcf    TRISD,0       ; place le bit 0 du port D en sortie
bcf    STATUS,RP0    ; STATUS(RP0) <- 0 revient sur le banc 0
    
```

main

```

movlw  1              ;
movwf  PORTD          ; met 1 sur le bit 0 sur port D
    
```

loop

```

xorwf  PORTD,f        ; inverse la valeur du bit 0
goto   loop           ; on boucle
    
```

END

; directive de fin de programme

programme

Outils de développement

- éditeur
gvim-emacs-kate-gedit
- assembleur
gasm
- simulateur
gpsim
- programmeur
picprog
(ou un bootloader)

