le PIC16f877

cours n°2
LI326

Plan

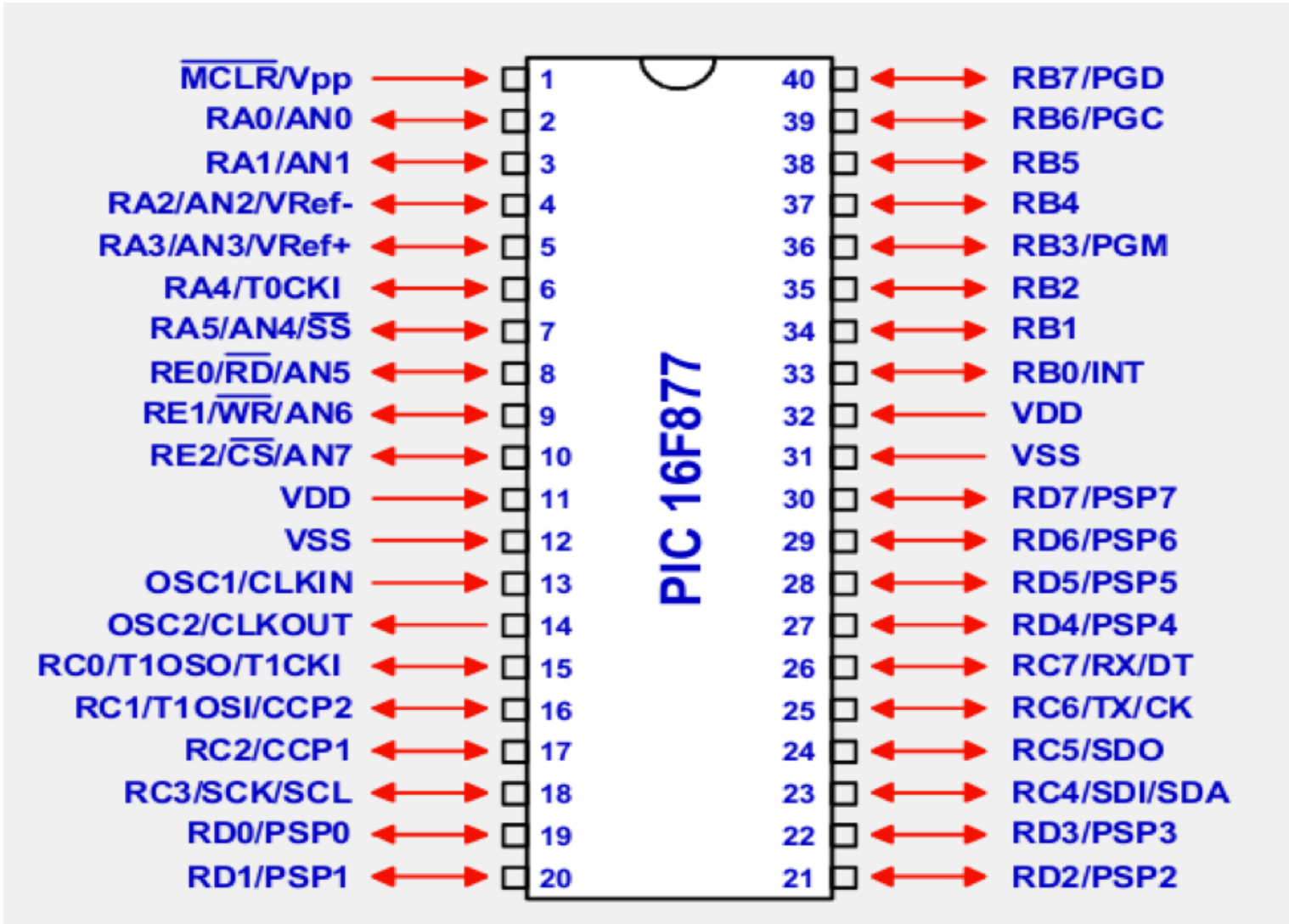
- Architecture du PIC16F877
- Plan mémoire
- Langage assembleur
- Programme “hello world”
- Outils de développement

Caractéristiques pic16f877

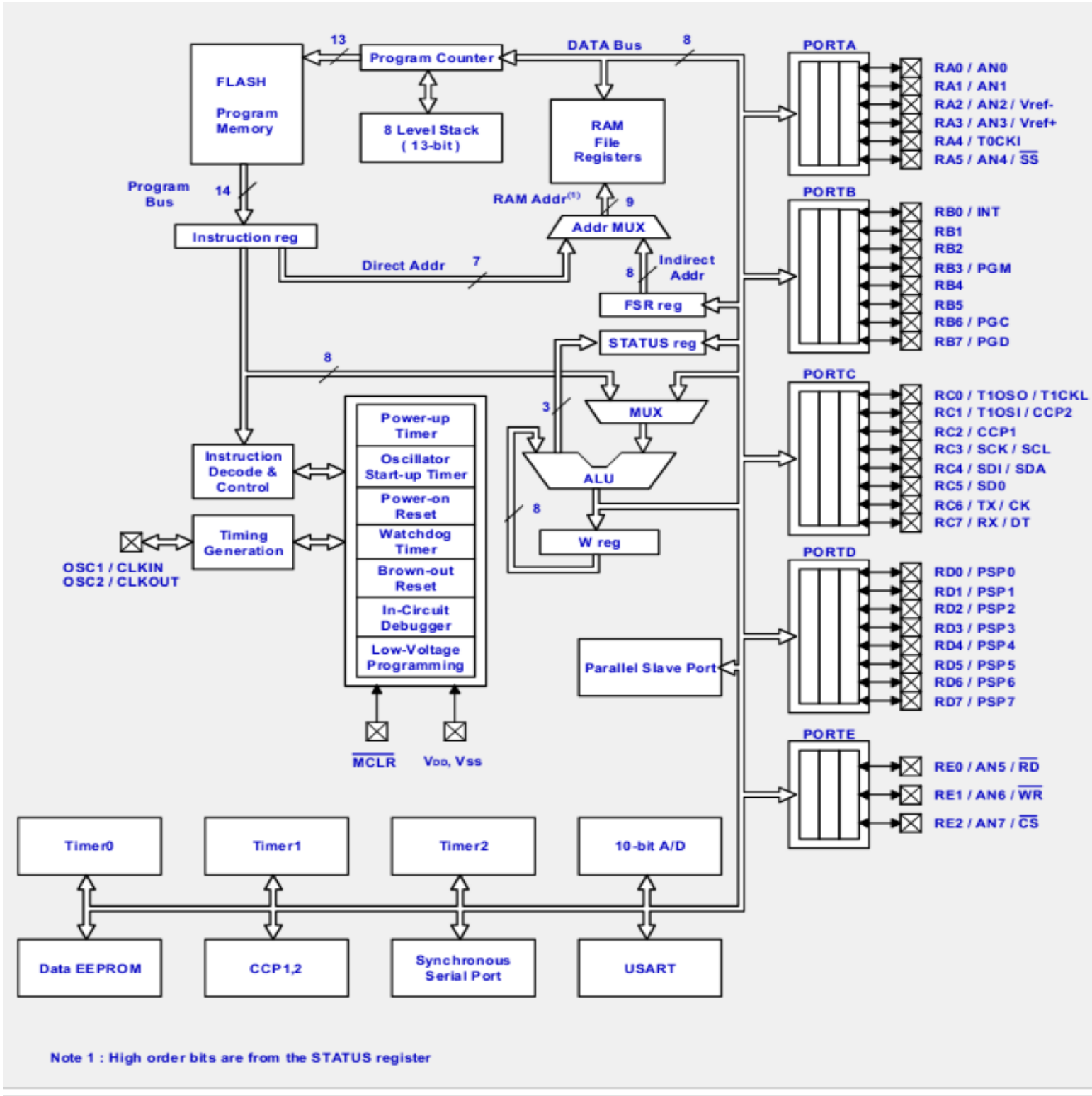
Fonctionne à 20 MHz maximum.
Possède :

- 35 instructions (composant RISC),
- 8Ko de mémoire Flash interne pour le programme,
- 368 octets de RAM,
- 256 octets de d'EEprom,
- 3 compteurs / timer de 8 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.

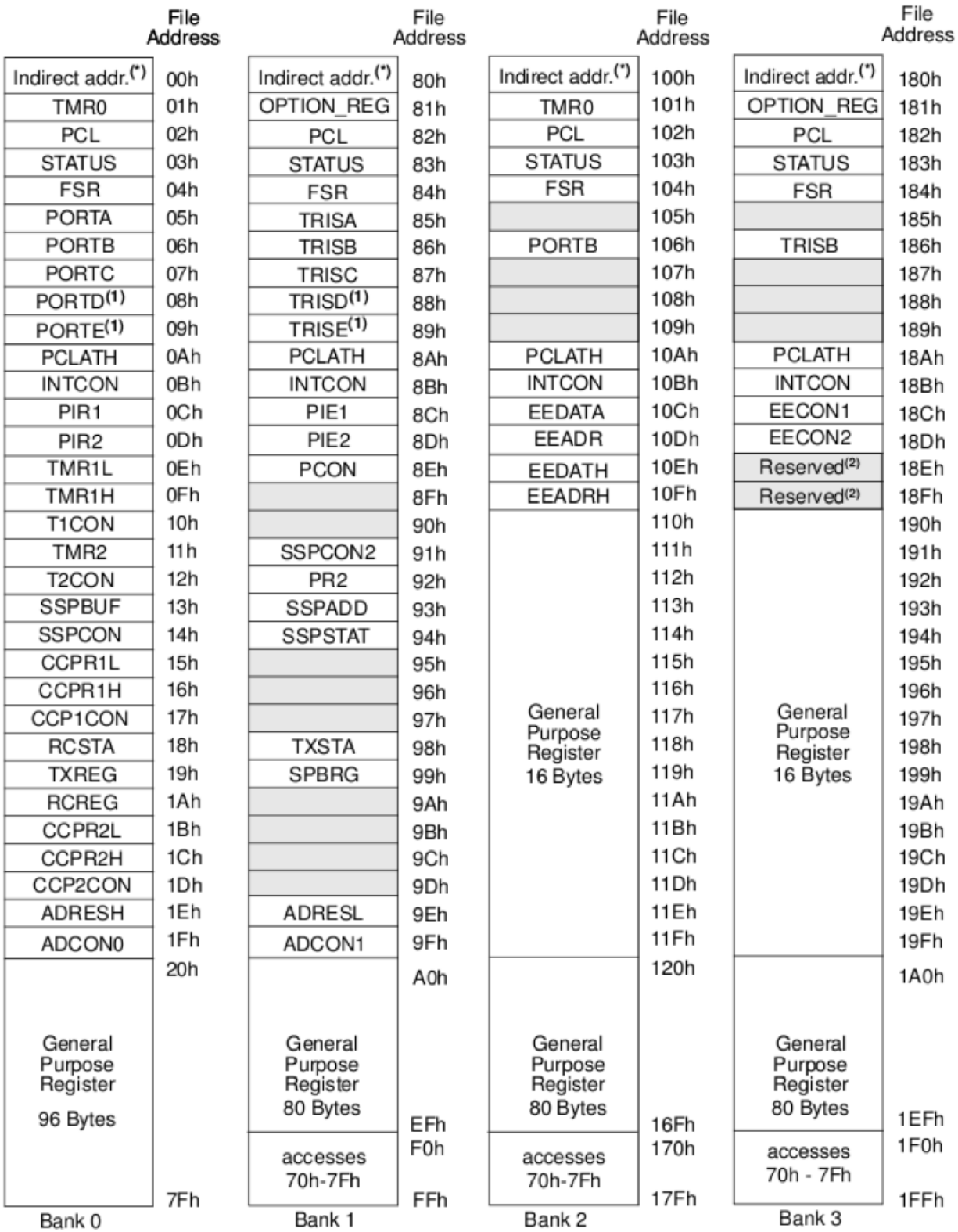
Boitier pic16f877 40 broches



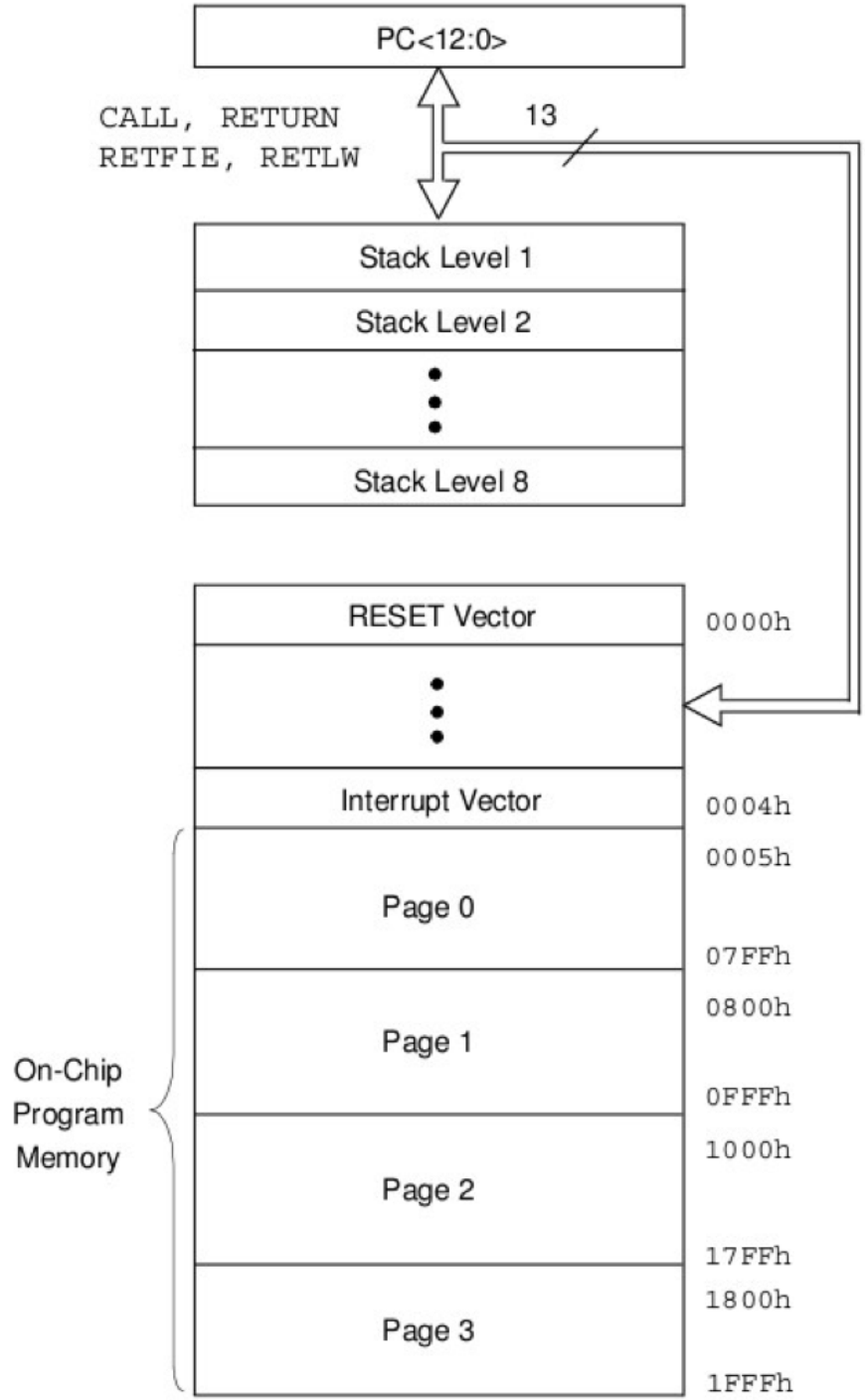
Architecture pic16f877



Registres



- 4 bancs
- certains registres adresses multiples
- SFR
  Special File Register
  pour les périphériques



Programme

- 4 pages
- Pile hardware
- 2 adresses pour le traitement des événements matériels

Format instruction assembleur / 1

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 opérandes

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

Instructions

Table 29-1: Midrange Instruction Set

Mnemonic, Operands		Description	Cycles	14-Bit Instruction Word MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Assembleur

Format instruction assembleur / 2

Cas général d'un branchement

```
goto    adresse
branch_if_cond  operande1, operande2, adresse
```

Pour le PIC

```
goto    adresse
```

(mais l'adresse n'est pas complète)

```
skip_if_notcond
    goto    adresse
```

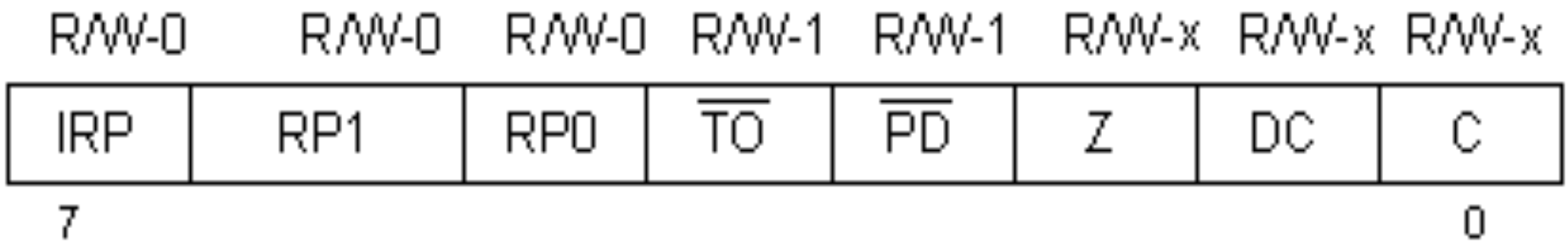
exemple MIPS :

```
beq    $4,$5,label
```

en PIC :

```
movf    23,0
xorwf   22,0
btfsc   status,2
goto    label
```

Registre status



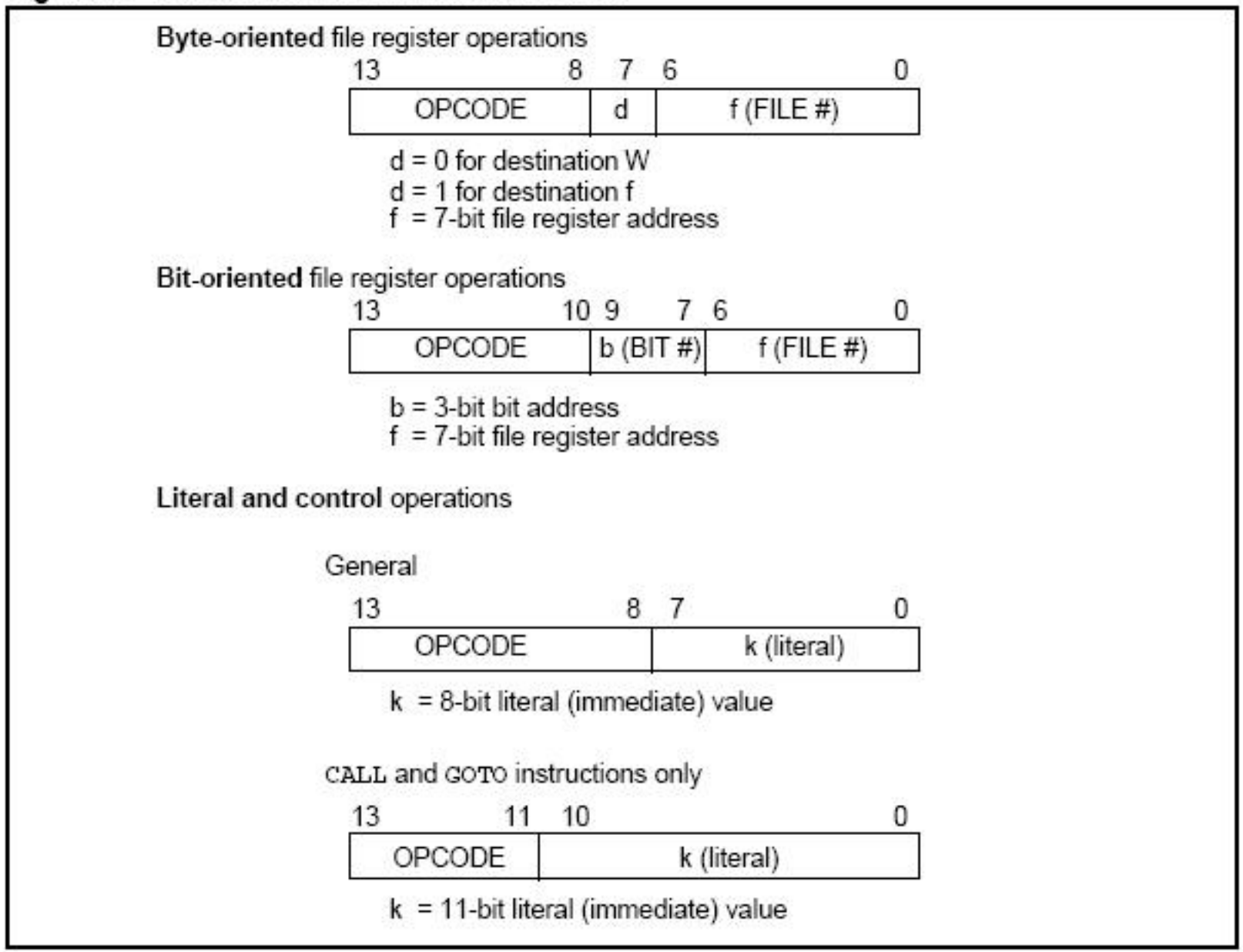
R = Readable bit **W** = Writable bit

U = Unimplemented bit, read as '0' - n = Value at power-on reset

Le registre status est modifié par la plupart des instructions
Ce qui va être utilisé pour faire des sauts conditionnels

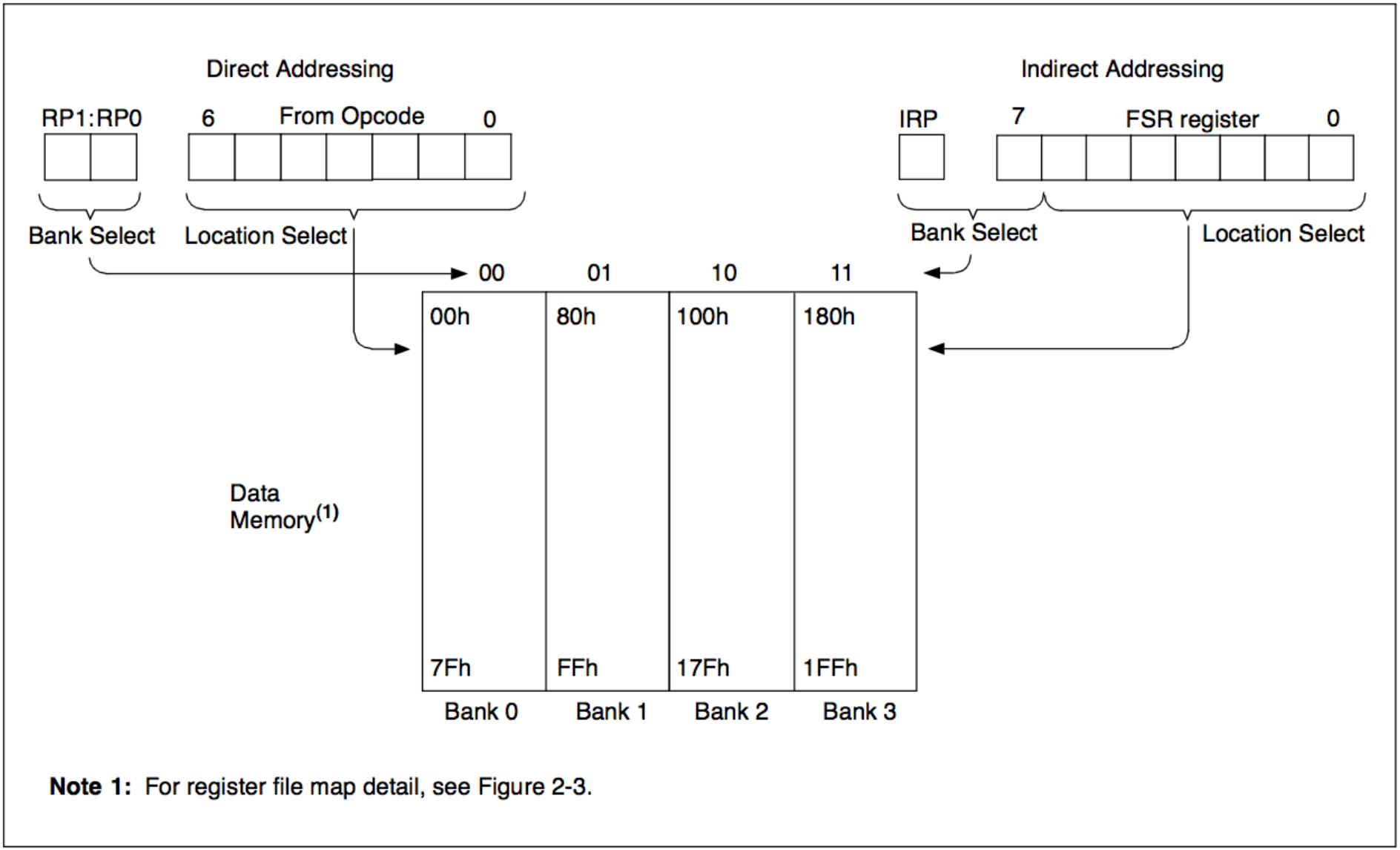
Codage des instructions

Figure 29-1: General Format for Instructions



Calcul des operandes

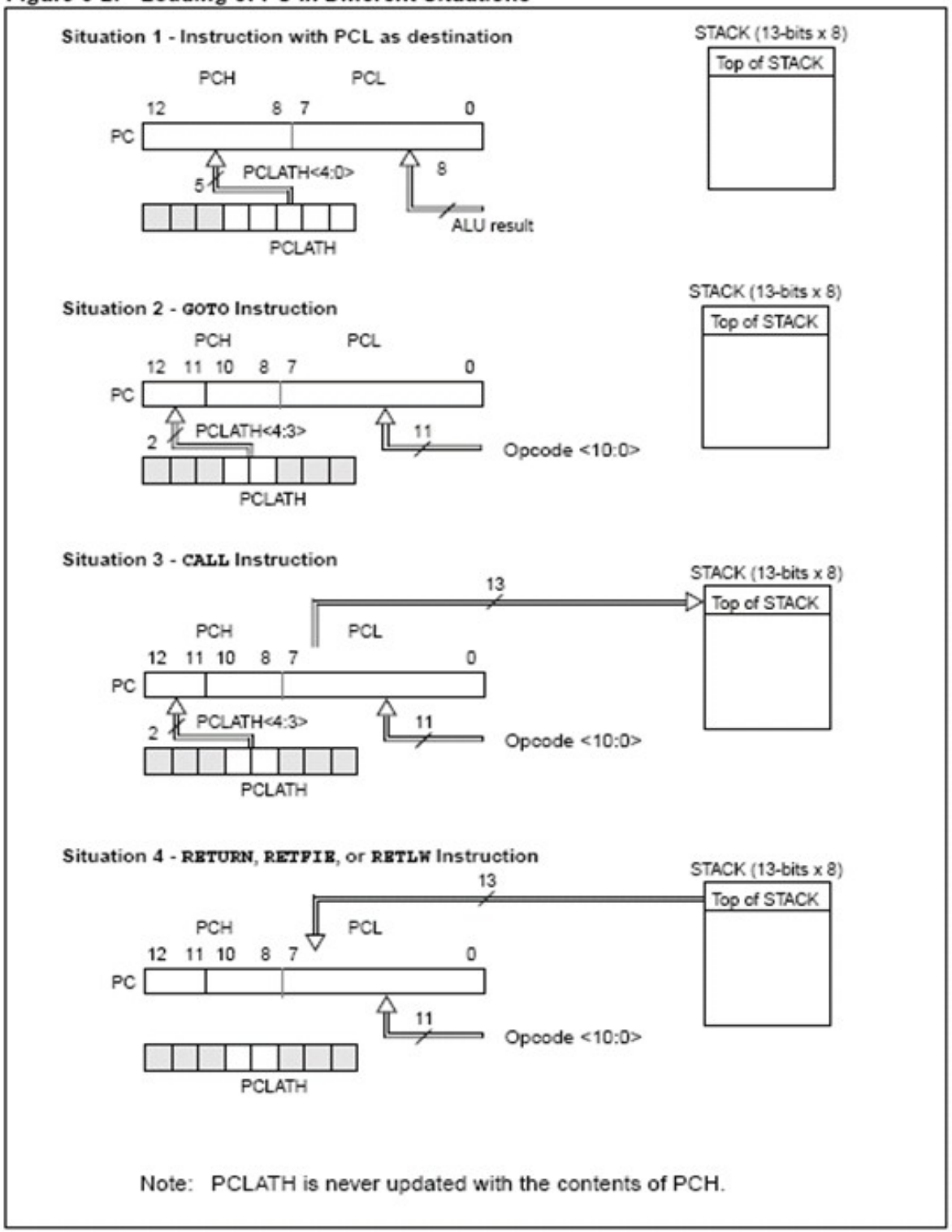
FIGURE 2-6: DIRECT/INDIRECT ADDRESSING



Note 1: For register file map detail, see Figure 2-3.

Assembleur

Figure 6-2: Loading of PC In Different Situations



Note: PCLATH is never updated with the contents of PCH.

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 accessible en écriture seule

Assembleur

GPIO principe

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

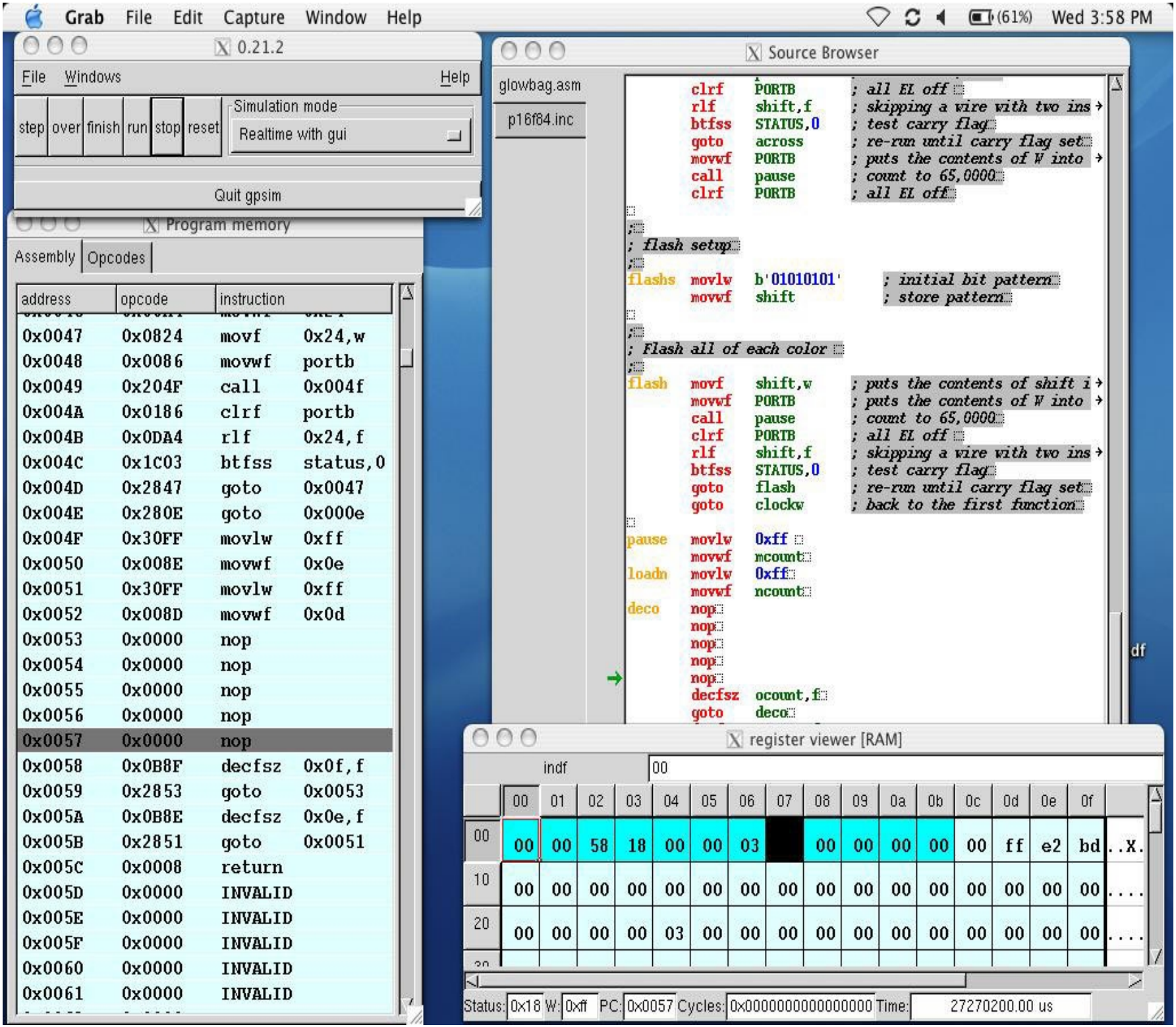
Pour chaque broche, nous avons deux bits de commande
- 1 bit pour définir de sens : entrée (1), sortie (0)
- 1 bit pour lire la donnée (si entrée) ou l'écrire (si sortie)

Les registres de direction sont nommés TRISx
Les registres de données sont nommés PORTx

Par exemple :
TRISD / PORTD sur 8 bits

Outils de développement

- éditeur : gvim / emacs / kate / gedit
- assembleur : gpasm
- simulateur : gpsim
- programmateur : picprog
  (remplacer après par un bootloader)


Grab File Edit Capture Window Help
(61%) Wed 3:58 PM
0.21.2
File Windows Help
step over finish run stop reset
Simulation mode
Realtime with gui
Quit gpsim
Program memory
Assembly Opcodes
address opcode instruction
0x0047 0x0824 movf 0x24,w
0x0048 0x0086 movwf portb
0x0049 0x204F call 0x004f
0x004A 0x0186 clrf portb
0x004B 0x0DA4 rlf 0x24,f
0x004C 0x1C03 btfss status,0
0x004D 0x2847 goto 0x0047
0x004E 0x280E goto 0x000e
0x004F 0x30FF movlw 0xff
0x0050 0x008E movwf 0x0e
0x0051 0x30FF movlw 0xff
0x0052 0x008D movwf 0x0d
0x0053 0x0000 nop
0x0054 0x0000 nop
0x0055 0x0000 nop
0x0056 0x0000 nop
0x0057 0x0000 nop
0x0058 0x0B8F decfsz 0x0f,f
0x0059 0x2853 goto 0x0053
0x005A 0x0B8E decfsz 0x0e,f
0x005B 0x2851 goto 0x0051
0x005C 0x0008 return
0x005D 0x0000 INVALID
0x005E 0x0000 INVALID
0x005F 0x0000 INVALID
0x0060 0x0000 INVALID
0x0061 0x0000 INVALID
Source Browser
glowbag.asm
p16f84.inc
clrf PORTB ; all EL off
rlf shift,f ; skipping a wire with two ins
btfss STATUS,0 ; test carry flag
goto across ; re-run until carry flag set
movwf PORTB ; puts the contents of W into
call pause ; count to 65,0000
clrf PORTB ; all EL off
; flash setup
flashs movlw b'01010101' ; initial bit pattern
movwf shift ; store pattern
; Flash all of each color
flash movf shift,w ; puts the contents of shift i
movwf PORTB ; puts the contents of W into
call pause ; count to 65,0000
clrf PORTB ; all EL off
rlf shift,f ; skipping a wire with two ins
btfss STATUS,0 ; test carry flag
goto flash ; re-run until carry flag set
goto clockw ; back to the first function
pause movlw 0xff
movwf mcount
loadn movlw 0xff
movwf ncount
deco nop
nop
nop
nop
nop
decfsz ocount,f
goto deco
register viewer [RAM]
indf 00
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00 00 00 58 18 00 00 03 00 00 00 00 00 ff e2 bd ..X.
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
20 00 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 ....
Status: 0x18 W: 0xff PC: 0x0057 Cycles: 0x0000000000000000 Time: 27270200.00 us

GPSIM

Hello World !

```
        list    p=16f877            ; definit le processeur cible
        include "p16f877.inc"   ; declaration 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
        return

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