Université Badji Mokhtar – Licence Electronique
Matière : Systèmes à microprocesseurs
Enseignant : Dr S.BENSAOULA

PIC16f84 : Jeu d'instructions et modes d'adressages

1. Les instructions

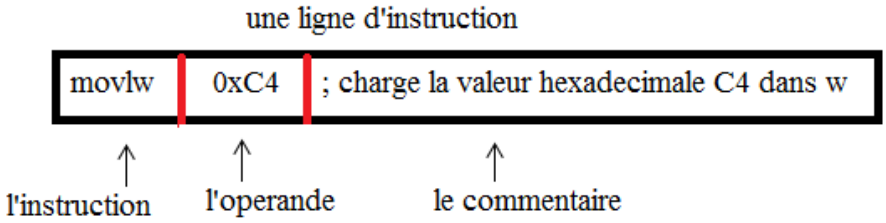
Les PICs sont conçus selon une architecture RISC. Programmer avec un nombre d'instructions réduit permet de limiter la taille de leur codage et donc de la place mémoire et du temps d'exécution.

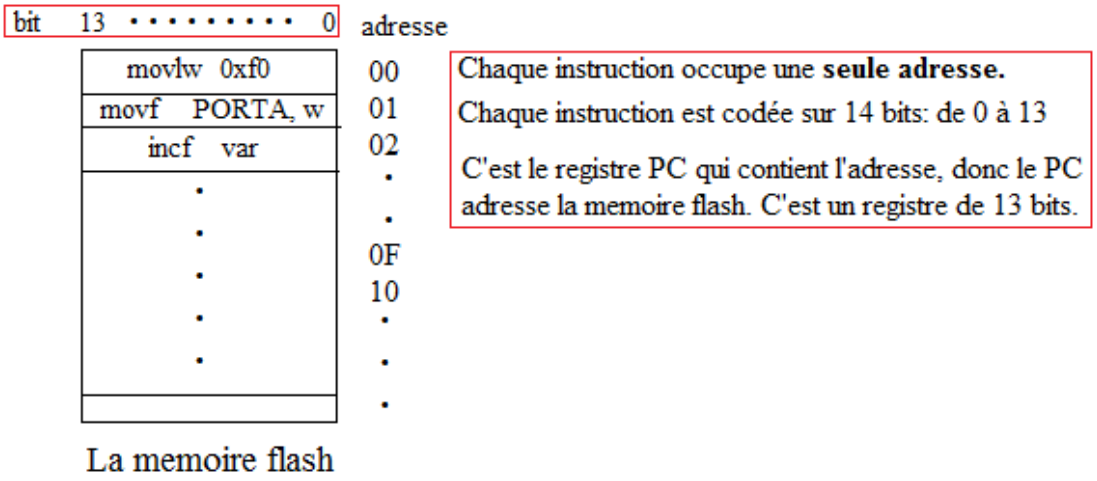
Toutes les instructions du pic 16f84 sont codées sur 14 bits. Elles sont regroupées en trois grands types:

- Instructions orientées octets
- Instructions orientées bits
- Instructions de contrôle

Le registre de travail w joue un rôle particulier dans un grand nombre d'instructions.

exemple d'une ligne d'instruction :





2. Les modes d'adressages

Il existe trois types d'accès à une donnée en mémoire RAM :

- adressage immédiat: la donnée est contenue dans l'instruction.
- adressage direct: la donnée est contenue dans un registre.
- adressage indirect : l'adresse de la donnée est contenue dans un pointeur

Ces trois types d'accès sont appelés modes d'adressages.

2.1 Adressage immédiat

La donnée est contenue dans la ligne d'instruction (champ opérande)

exemples :

```
movlw   0xC4          ; transfert la valeur hexadécimale C4 dans w
movlw   .255          ; transfert la valeur décimale 255 dans w
movlw   B'00001111'   ; transfert la valeur binaire 00001111 dans w
```

2.2 Adressage direct

La donnée est contenue dans un registre. Ce dernier peut être un nom (par exemple w) ou une adresse mémoire,

exemples :

```
movwf   PORTB   ; copie (ou transfert) le contenu de w dans le PORTB : adressage direct
                ; c'est une écriture dans le PORTB

movf    PORTB,w ; transfert le contenu du PORTB dans le registre w : adressage direct
movf    0x2B, w ; transfert le contenu de l'adresse 0x2B dans w: adressage direct
                ; c'est une lecture de la position mémoire d'adresse 0x2B
```

Le PORTA peut lui aussi être utilisé exactement comme le PORTB.

autre forme syntaxique:

```
movf    0x2B, 0     ; copie le contenu de l'adresse 0x2B dans w
```

cette fois-ci le registre w est remplacé par 0

Remarque : une adresse s'écrit toujours en hexadécimale

exemples : l'adresse 9 : s'écrit 0x09
l'adresse 100 : s'écrit 0x64

2.3 Adressage indirect

2.3.1 Le pointeur : définition

Un **pointeur** est un registre qui contient une adresse. Donc lorsque ce pointeur est utilisé, son contenu est une adresse.

L'adressage indirect utilise un pointeur. Dans le pic 16f84 un seul pointeur est disponible pour l'adressage indirect, c'est le **registre FSR**.

Quand on veut lire ou écrire dans une position mémoire en RAM, en utilisant l'adressage indirect, on doit d'abord chargée l'adresse de cette position dans le registre FSR.

Le pic exécute l'adressage indirect en faisant intervenir un deuxième registre, c'est le registre **INDF**. Ce registre se trouve à l'adresse 0 dans les deux banques (bank0 et bank1); il n'a pas d'existence physique. On doit le faire apparaitre dans les lignes d'instructions de l'adressage indirect.

2.3.2 Lecture par adressage indirect

exemple

```
movlw   0x1A   ; charge 1A dans w
movwf    FSR     ; transfert w dans FSR : on dit que le registre FSR pointe l'adresse  0x1A
movf     INDF, 0   ; copie le contenu de l'adresse 0x1A dans w : c'est une lecture
```

exécution par le pic de l'instruction : movf INDF, 0

1. le pic lit le contenu de FSR : c'est l'adresse 0x1A,
2. ensuite il copie le contenu de l'adresse 0x1A dans w : c'est la lecture d'une position mémoire par adressage indirect.

2.3.3 Ecriture par adressage indirect

exemple

```
 movlw  0x0D          ; charge 0D dans w
 movwf   FSR          ; copie w dans FSR : on pointe l'adresse 0x0D
 movlw   0xff    ; charge w avec 255 (ou autre valeur)
movwf   INDF   ; copie w dans la position mémoire d'adresse 0x0D : c'est une écriture,
```

exécution par le pic de l'instruction : movwf INDF

1. Le pic lit le contenu de FSR : c'est l'adresse 0x0D,
2. ensuite il copie le contenu de w dans cette adresse : c'est l'écriture dans une position mémoire par adressage indirect.

Conclusion

- ✓ **la lecture :** c'est lorsqu'une donnée est copiée dans le registre w
  cette donnée peut provenir soit d'un port, soit de la RAM

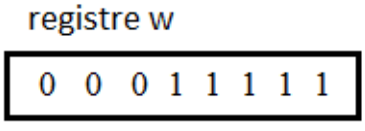
- ✓ **l'écriture :** c'est lorsqu'une donnée sort du registre w
  cette donnée peut être envoyée soit vers un port ou vers la RAM.

- ✓ **l'adressage indirect** est utilisé lorsqu'on veut lire ou écrire dans un tableau de données en RAM.

3. Représentation de l'information dans le pic

```
movlw  .31              ; charge 31 dans w: opérande décimale
movlw  0x1f             ; charge 31 dans w : opérande hexadécimale
movlw  B'00011111'      ; charge 31 dans w : opérande binaire
```

C'est trois écritures sont équivalentes, et la valeur 31 est représentée dans w comme suit :



Le jeu d'instructions

Mnemonic, Operands		Description	Cycles	14-Bit Opcode 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	