Informatique Embarquée (IE)

(Embedded Systems)

Cours n°2

Le traitement d'une instruction, Pipeline, CISC/RISC
Accumulateur W, ALU, Registre d'état
Langages Assembleur et C, Compilateur C, Linker

olivier.lourme@univ-lille.fr

Le traitement d'une instruction : intro

Une Instruction en Langage Machine (ILM) =
une action élémentaire effectuée par le µP ou µC :

- Copier le contenu d'un registre (GPR ou SFR) vers un autre;
- Inverser tous les bits d'un registre;
- Réaliser une addition;
- Etc.

Des opérations très simples !

C'est dans la mémoire de programme qu'on stocke les ILM.

Ne pas confondre ILM avec Instruction en Langage Evolué (ILE) :

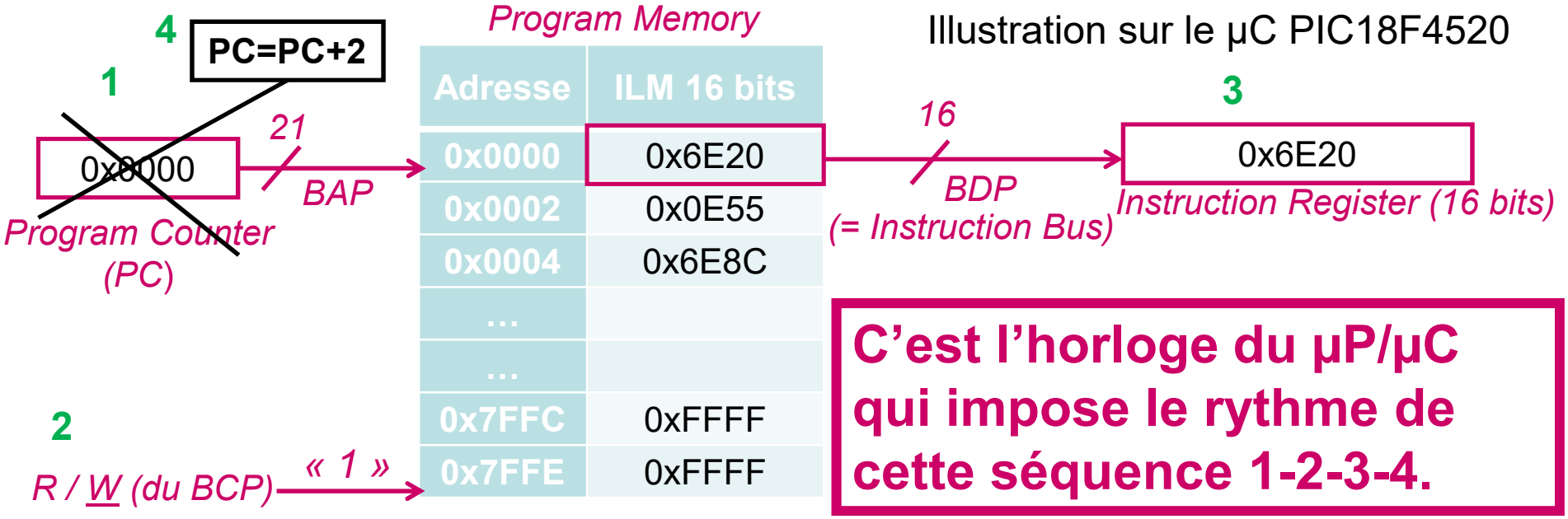
Par exemple, en langage C (langage évolué), une boucle avec les 2 instructions :

```
for(i=0; i < N; i++) {
   printf("%d", i);
}
```

La succession très rapide d'opérations très simples permet un travail complexe.

sera, grâce au travail du compilateur C, décomposée en une suite
plus ou moins grande d'ILM afin que le µP ou µC exécute le travail demandé.

Le traitement d'une instruction : 1 - *Fetch*



C'est l'horloge du µP/µC qui impose le rythme de cette séquence 1-2-3-4.

1. Le registre ***Program Counter*** (PC) dépose sur le Bus d'Adresse-Programme (BAP) l'adresse de l'instruction à récupérer, par exemple 0x0000 au *reset* du µC.
2. Le signal binaire R/W du Bus de Contrôle-Programme (BCP) est ajusté pour provoquer une **lecture de la mémoire de programme**.
3. Le contenu désigné est copié vers le **Registre d'Instruction** grâce au Bus de Données-Programme (BDP) appelé aussi **Bus d'Instruction**.
4. Le PC est incrémenté de 2 unités pour désigner la prochaine instruction.

Le traitement d'une instruct. : 2 - *Execute*

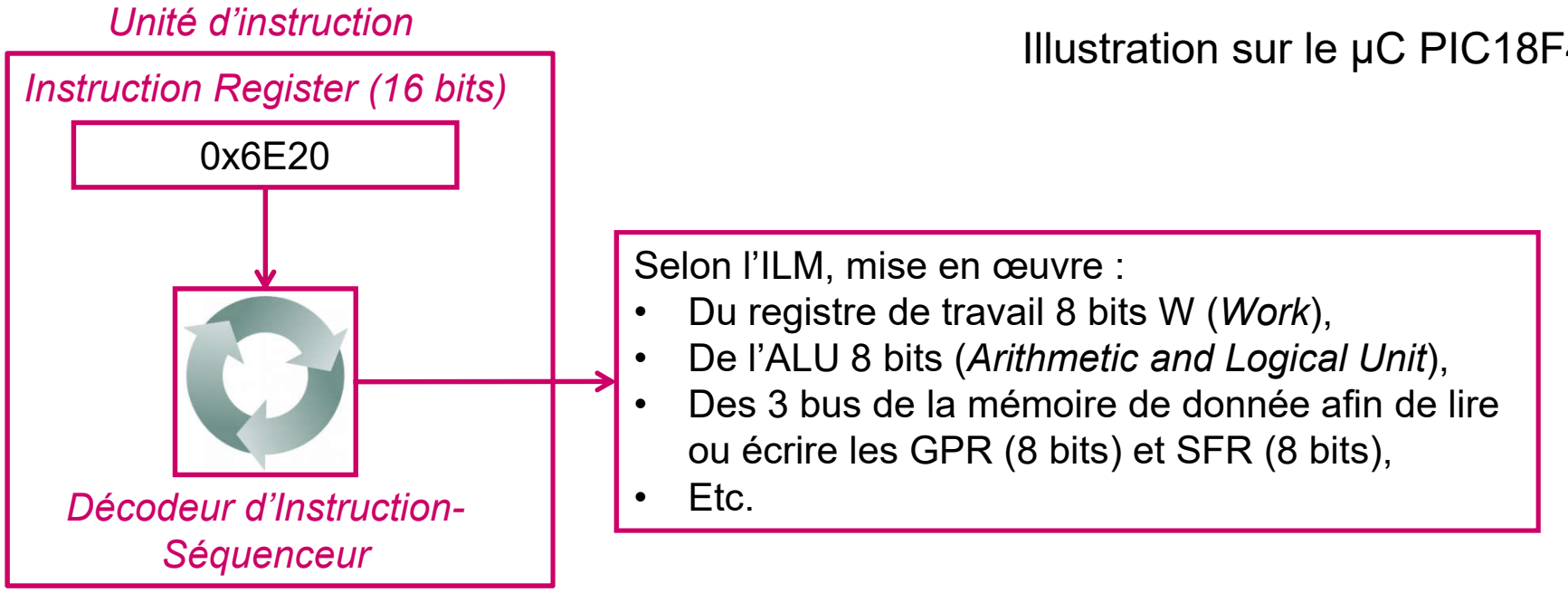


Illustration sur le µC PIC18F4520

- La phase d'*Execute* assure l'exécution proprement dite de ce que l'ILM est censée faire.
- Pour cela, un bloc « Décodeur d'Instruction-Séquenceur » (DIS) va analyser l'ILM stockée dans le Registre d'Instruction et mettre en œuvre, dans le bon ordre et toujours à un rythme imposé par l'horloge du µC, les éléments du µC permettant l'exécution de l'instruction.
- Unité d'instruction = Registre d'Instruction + Décodeur d'Instruction-Séquenceur

Le traitement d'une instruct. : 2 - *Execute*

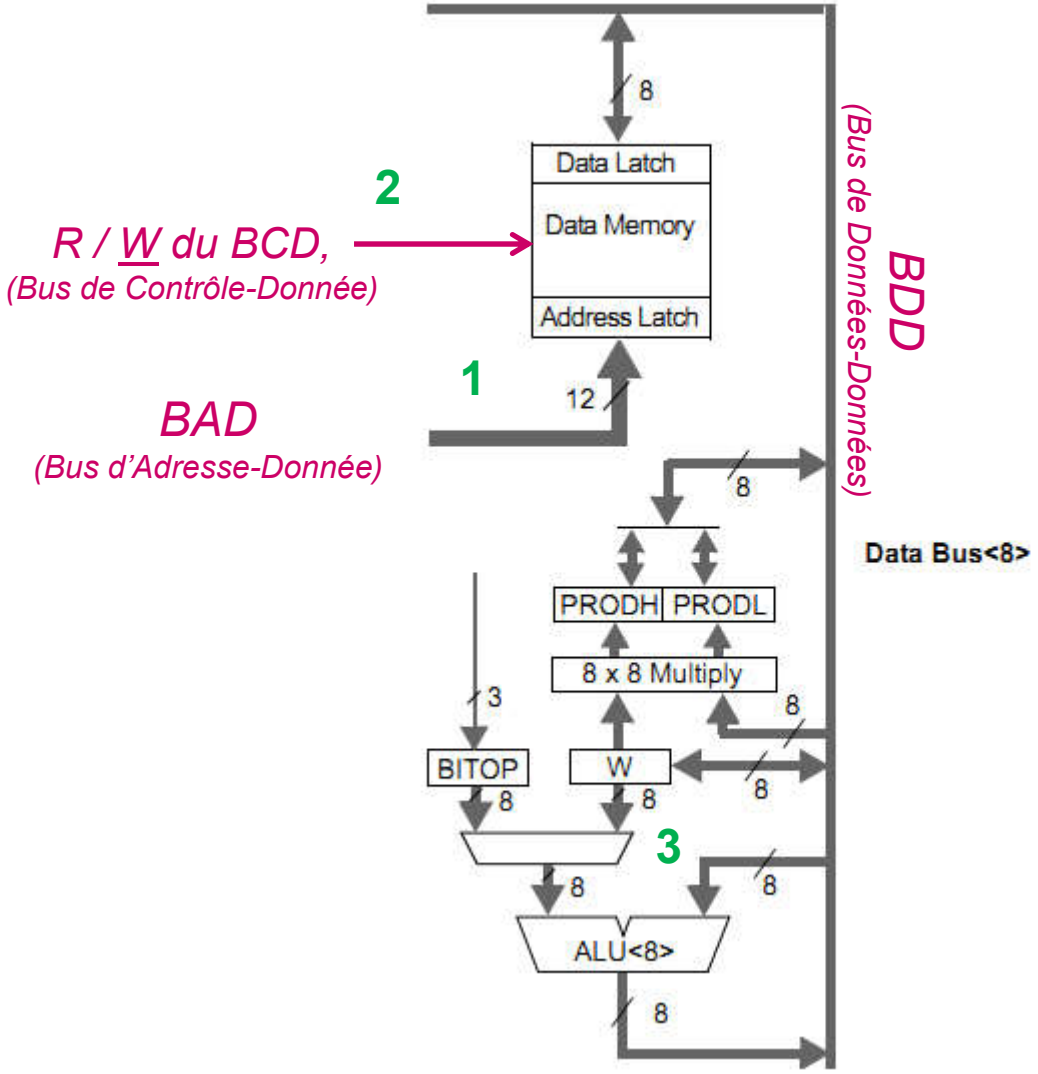
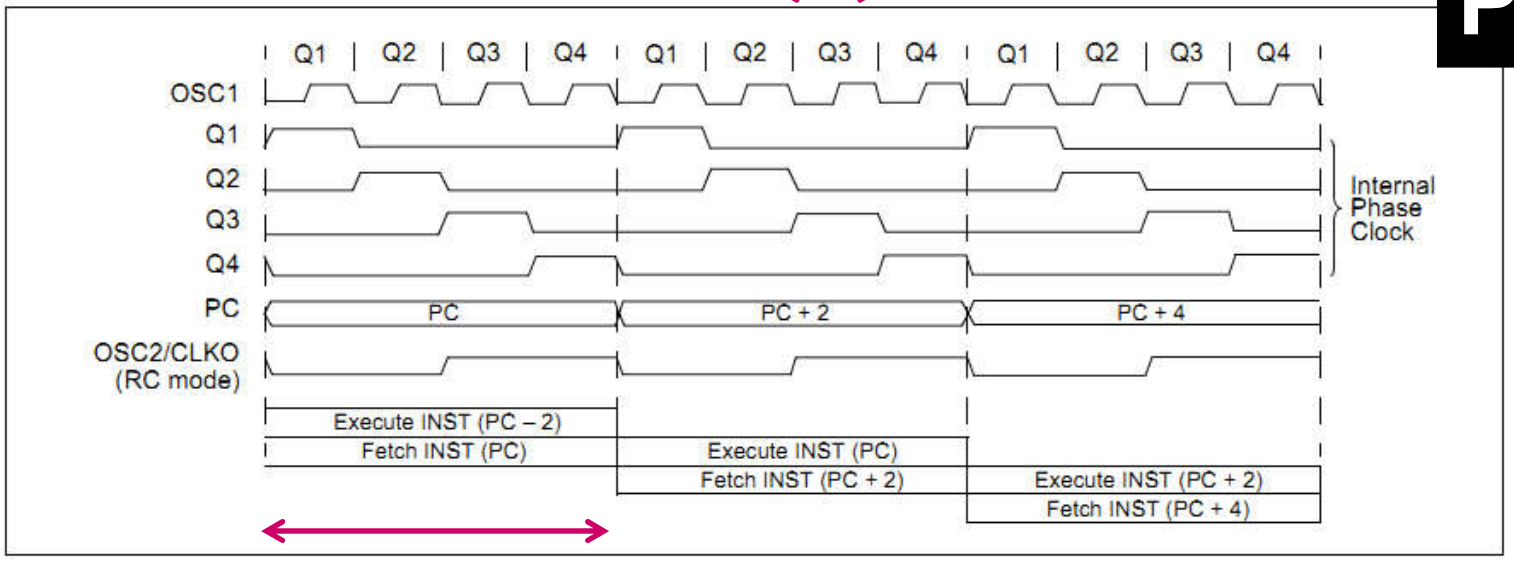


Illustration sur le µC PIC18F4520

- Par exemple, l'ILM 16 bits 0x6E20 a pour effet de copier le contenu du registre de travail 8 bits W vers le GPR 8 bits d'@ 0x020 de la mémoire de données.
- Le DIS va pour cette instruction déclencher une séquence en trois étapes:
  - 1: Ecrire sur le BAD l'adresse 0X020.
  - 2: Mettre R/W du BCD à 0 pour annoncer à *Data Memory* qu'une écriture va avoir lieu à l'adresse 0X020.
  - 3: Mettre le contenu de W sur le BDD pour qu'il arrive à l'@0X020 de la *Data Memory*.

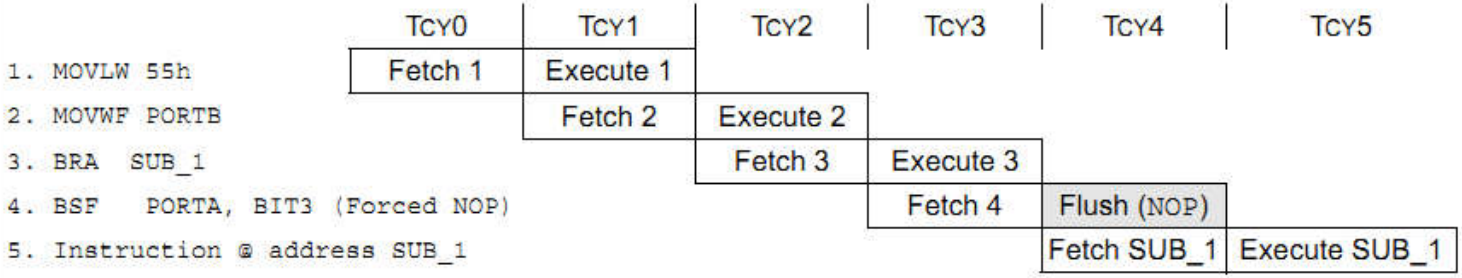
Pipeline

FIGURE 5-3: CLOCK/INSTRUCTION CYCLE ⟷ *1 période d'horloge*



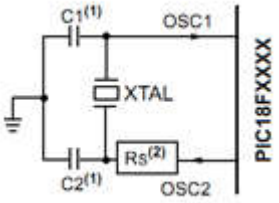
1 cycle instruction = 1 Tcy = 4 périodes d'horloge

EXAMPLE 5-3: INSTRUCTION PIPELINE FLOW



All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

Horloge
=
oscillateur à quartz :



(page 57 de 39631E.pdf)

Nature des ILM : RISC ou CISC ?

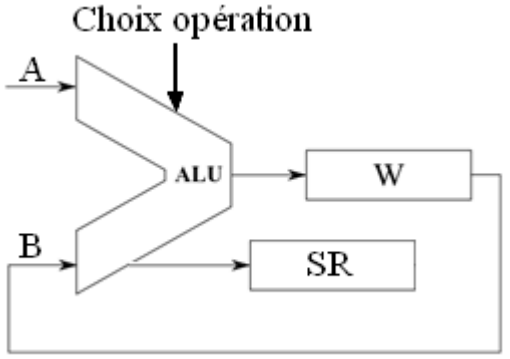
	RISC (*Reduced Instruction Set Code*) : Microchip, DSP	CISC (*Complex Instruction Set Code*) : Intel, Power PC
Description	Chaque instruction est très simple. Exemple PIC : additionner le contenu du registre W avec le contenu d'une case mémoire. Tout est « câblé », « silicium », très simple.	Les instructions font des choses complexes. Exemple : additionner les contenus de deux cases mémoire. Une instruction est une séquence de micro-instructions. Un « microcode » est inscrit sur silicium.
Avantage / Inconvénient	++ Unité d'Instruction simple. Les instructions ont toutes la même taille (par exemple toujours 2 octets sur PIC18F4520).	- - Le microcode à graver sur silicium coûte cher en développement et production. Les instructions ont des tailles variables.
Avantage / Inconvénient	++ Un cycle fixe / instruction	- - Temps d'exécution / instruction variable
Avantage / Inconvénient	- - Programme long (donc mémoire de programme bien remplie)	++ Programme compact

Accumulateur W, ALU, Registre d'état (1)

Illustration sur le µC PIC18F4520

➔ ***Ces 3 éléments constituent le <u>cœur du traitement de l'information</u> au sein du µC/µP. C'est là que se font les calculs.***

- ALU « 8 bits » : *Arithmetic and Logical Unit*. Elle effectue les **opérations arithmétiques** (addition, soustraction, etc. mais pas la multiplication) **et logiques** (décalages, ET, OU , etc.).
  En entrée : deux opérandes 8 bits (A et B) et le choix de l'opération à effectuer.
  En sortie : le résultat (registre 8 bits W) et un registre spécial : le Registre d'Etat (*Status Register* = *SR*).
- Registre W : Registre de travail (*Work*). Vu le câblage proposé, si par exemple l'opération choisie est une addition alors on a :
  Nouveau W = A + Ancien W. D'où le nom d'**Accumulateur**.



Note : dans le PIC18F4520 :
- Il y a aussi une MPU.
- Mais il n'y a pas de FPU.

Accumulateur W, ALU, Registre d'état (2)

REGISTER 5-2: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC(1)	C(2)
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **N:** Negative bit
This bit is used for signed arithmetic (2's complement). It indicates whether the result was negative (ALU MSB = 1).
1 = Result was negative
0 = Result was positive

bit 3 **OV:** Overflow bit
This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude which causes the sign bit (bit 7) to change state.
1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

bit 2 **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit Carry/$\overline{\text{borrow}}$ bit(1)
For ADDWF, ADDLW, SUBLW and SUBWF instructions:
1 = A carry-out from the 4th low-order bit of the result occurred
0 = No carry-out from the 4th low-order bit of the result

bit 0 **C:** Carry/$\overline{\text{borrow}}$ bit(2)
For ADDWF, ADDLW, SUBLW and SUBWF instructions:
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Illustration
sur le µC PIC18F4520

(page 67 de 39631E.pdf)

Le langage assembleur

Illustration sur le µC PIC18F4520

- Il est pénible de retenir que l'ILM 0x6E20 permet de copier le contenu du registre de travail 8 bits W et de le déposer dans le GPR 8 bits d'@ 0x020 de la mémoire de données.
- Et il y a beaucoup d'ILM possibles (= **jeu d'instructions**) ! Il n'est pas possible de retenir leur code machine.
- Remarque : chaque µC / µP a son propre jeu d'instruction (lié à l'architecture).

➔ ***A chaque travail possible, on associe un mnémonique d'assemblage plus simple à retenir pour un humain :***

- Par exemple, le **mnémonique** permettant de copier le contenu du registre de travail 8 bits W pour le déposer dans un GPR 8 bits est : `MOVWF`.
- En langage d'assemblage 0x6E20 s'écrit `MOVWF 0x20`. 0x20 est l'**opérande**.

➔ ***Correspondance 1/1 : Une ILM = Une instruction en assembleur.***

instruction = mnémonique + opérande(s)

Jeu d'instructions PIC18F4520 (extrait)

TABLE 24-2: PIC18FXXXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
BYTE-ORIENTED OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, Skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4
CPFSGT	f, a	Compare f with WREG, Skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4
CPFSLT	f, a	Compare f with WREG, Skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	1100 1111	ffff ffff	ffff ffff	ffff ffff	None	
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	1, 2

Jeu d'environ 80 instructions

Logiciel d'assemblage

Illustration sur le µC PIC18F4520

➔ ***Le programmeur écrit un programme en assembleur.***

➔***Un logiciel d'assemblage génère à partir de ce programme en assembleur un programme suite d'ILM dit « programme objet ».***

Programme en assembleur file1.asm		mpasmwin	Programme en ILM file1.o
MOVLW	0x00	→	0x0E00
MOVWF	TRISD, ACCESS		0x6E95
MOVLW	0x55		0x0E55
MOVWF	LATD, ACCESS		0x6E8C
4 lignes			***4 lignes***

➔ ***Le programme en ILM sera ensuite logé à l'endroit adéquat de la mémoire de programme.***

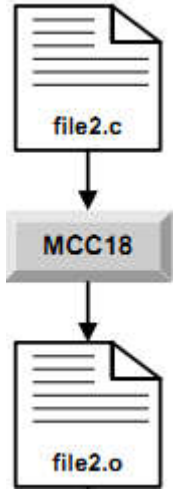
Ecriture de programmes en langage C (1)

- Même si écrire un programme en assembleur est plus lisible que l'écrire en ILM, cela reste très pénible car c'est un **langage de bas niveau** propre à un processeur et à son architecture.
- Par exemple, une boucle est très facile à écrire en C avec un `for`, c'est beaucoup plus compliqué en assembleur.
- Le langage C est un **langage évolué** (dit aussi de **haut niveau**). Un programme écrit en langage C est *a priori* non dépendant du processeur cible.

➔ ***Le programmeur écrit un programme en langage C.***

➔ ***Un logiciel** (ex. : MCC18 ou gcc) **appelé compilateur C (C-compiler) génère à partir du programme en C un programme en assembleur qu'il convertit ensuite en une suite d'ILM dit « programme objet » (= programme en langage machine).***

- Comme le compilateur est installé sur un PC mais que son produit est à destination d'une cible différente, on parle de « **compilation croisée** ».



Ecriture de programmes en langage C (2)

Illustration sur le µC PIC18F4520

➔ ***Exemple sur un extrait de programme en langage C :***

Compilateur C MCC18

Programme en C file1.c (extrait)
TRISD = 0x00;
LATD = 0x55;

1 →

Programme en assembleur file1.asm (extrait)	
MOVLW	0x00
MOVWF	TRISD, ACCESS
MOVLW	0x55
MOVWF	LATD, ACCESS

2 →

Programme en ILM file1.o (extrait)
0x0E00
0x6E95
0x0E55
0x6E8C

2 lignes *4 lignes* *4 lignes*

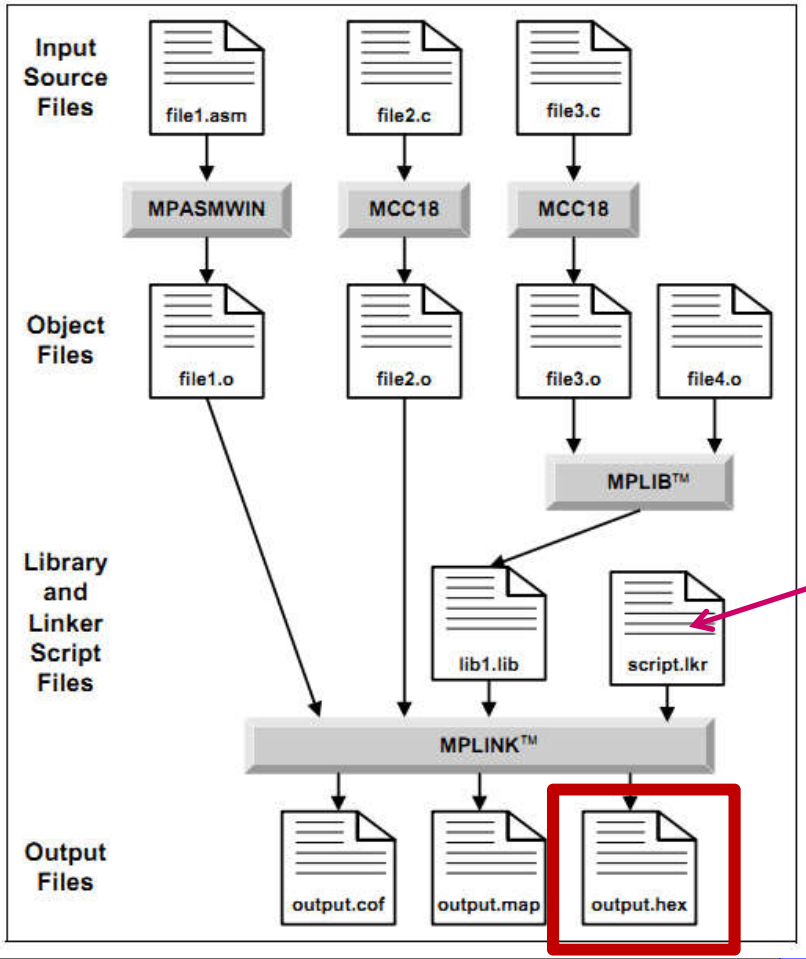
➔ ***Le taux d'expansion n'est pas 1/1 dans la 1ère étape.***

Chaîne de dvpt de projet : *linker*, etc.

EXECUTION FLOW

An example of the flow of execution of the language tools is illustrated in Figure 1-3.

FIGURE 1-3: LANGUAGE TOOLS EXECUTION FLOW



Traduction :

library :

linker :

linker script :

build :

Fichier script de l'éditeur de liens

In the above example, two C files are compiled by MPLAB C18, `file2.c` and `file3.c`, and an assembly file, `file1.asm`, is assembled by MPASM. These result in object files, named `file1.o`, `file2.o` and `file3.o`.

A precompiled object file, `file4.o`, is used with `file3.o` to form a library called `lib1.lib`. Finally, the remaining object files are combined with the library file by the linker.

MPLINK also has as an input linker script, `script.lkr`. MPLINK produces the output files, `output.cof` and `output.map`, and the HEX file, `output.hex`.