CHAPITRE EN6

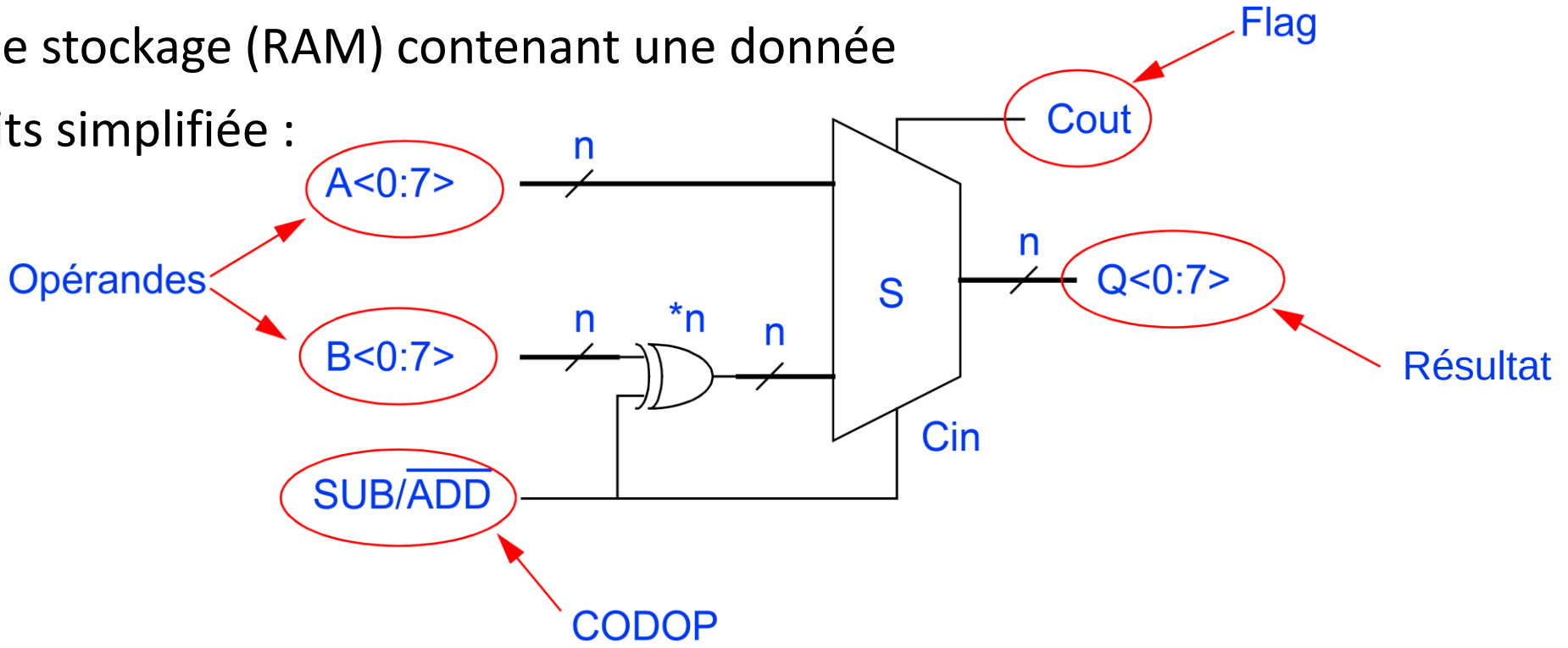
Architecture élémentaire des ordinateurs: l'unité arithmétique et logique (ALU)

SOMMAIRE

DÉFINITIONS

- **Unité Arithmétique et Logique (ALU) :** circuit contenant plusieurs opérateurs
- **Opérateur :** circuit combinatoire qui reçoit en entrée un ou deux opérandes,
  leur applique une opération (ET, OU, INV, addition, soustraction, décalage, etc...)
  en fonction d'un code opération pré-défini (CODOP)
  et fournit en sortie le résultat correspondant ainsi qu'un ou plusieurs indicateurs (Flags)
- **Registre :** zone de stockage (RAM) contenant une donnée

Exemple d'ALU 8 bits simplifiée :



DÉFINITIONS – Critères de performance

- Nombre de bits de l'ALU = largeur maximale des mots binaires traités en une opération
  - Remarque :
    - il est possible de travailler avec des mots plus larges, mais plusieurs opérations successives seront nécessaires.
    - Par exemple, l'addition de deux mots de 16 bits avec l'ALU de l'exemple précédent nécessite plusieurs opérations successives sur les différentes parties des opérandes.

- Temps de traitement dans les opérateurs
  - Exemple du PIC18Fxxxx: 16 MIPS (Mega Instructions Per Second) max.

- Nombre d'ALU **physiquement** présentes
  - possibilité ou non de réaliser des opérations en parallèle (processeurs mono-cœurs / processeurs multi-cœurs)

- Diversité des opérateurs = diversité des instructions disponibles pour la programmation
  - Exemple du PIC18F:
    - Opérations arithmétiques: Add, Substract, Multiply, Increment, Decrement, Negate
    - Opérations logiques: AND, OR, XOR, Complement

DÉFINITIONS – Programme

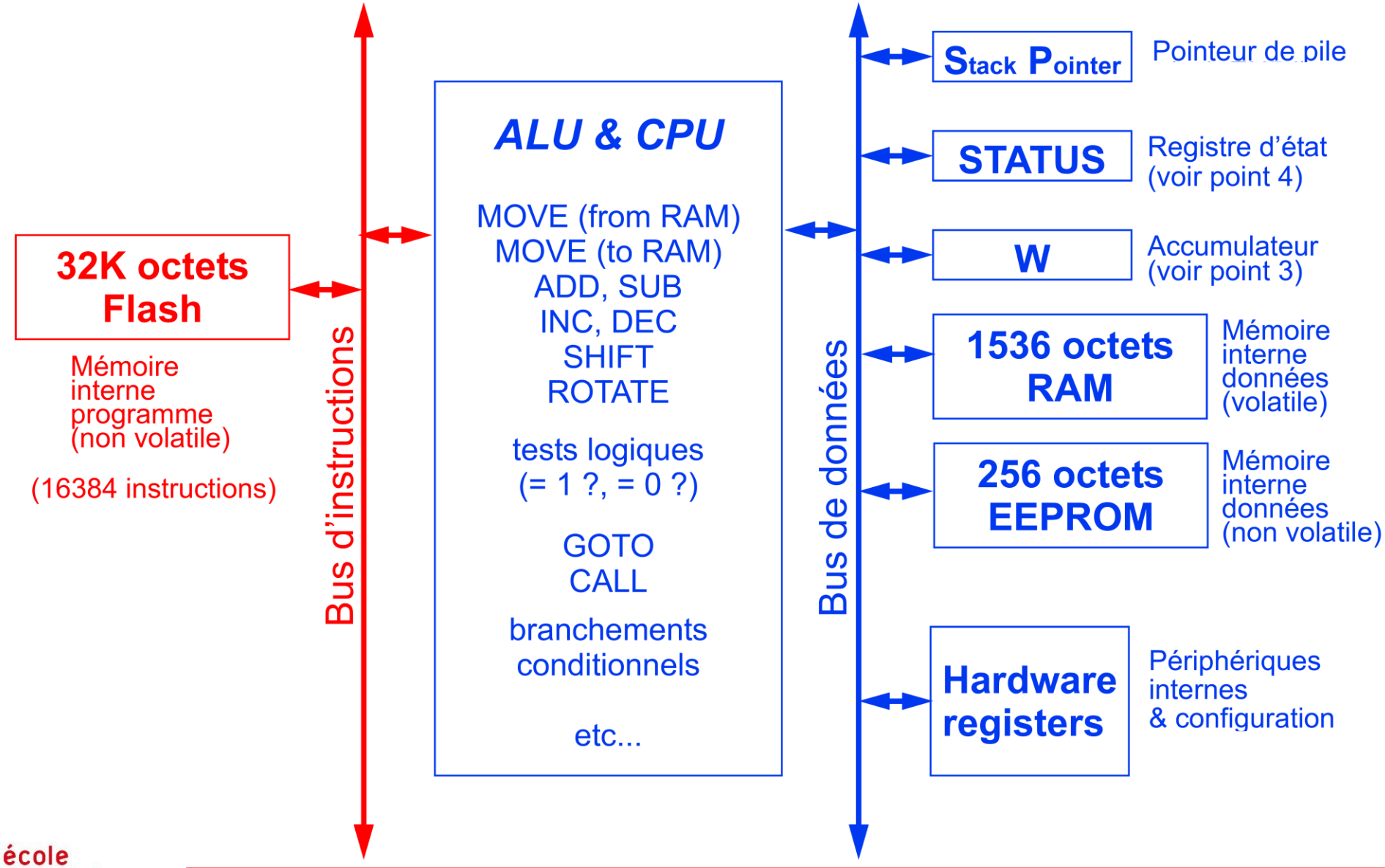
- Programme exécuté par un processeur :
  - suite d'opérations (instructions) effectuées séquentiellement
  - (en langage **assembleur**: 1 instruction = 1 commande compréhensible par le processeur)

- La liste des instructions est stockée dans la mémoire programme
  - après avoir exécuté une instruction, on passe à celle qui est stockée juste à la suite dans la mémoire
  - (sauf en cas de branchement : CALL routine, GOTO, etc...)

- Informations contenues dans une instruction :
  - que faut-il faire ?
  - avec quelle donnée ?
  - où placer le résultat ?
  - etc...

- Cas de la famille PIC18 :
  - 1 instruction codée sur 2 octets (16 bits) dans la majorité des cas

ALU ET REGISTRES DU PIC18F45K20



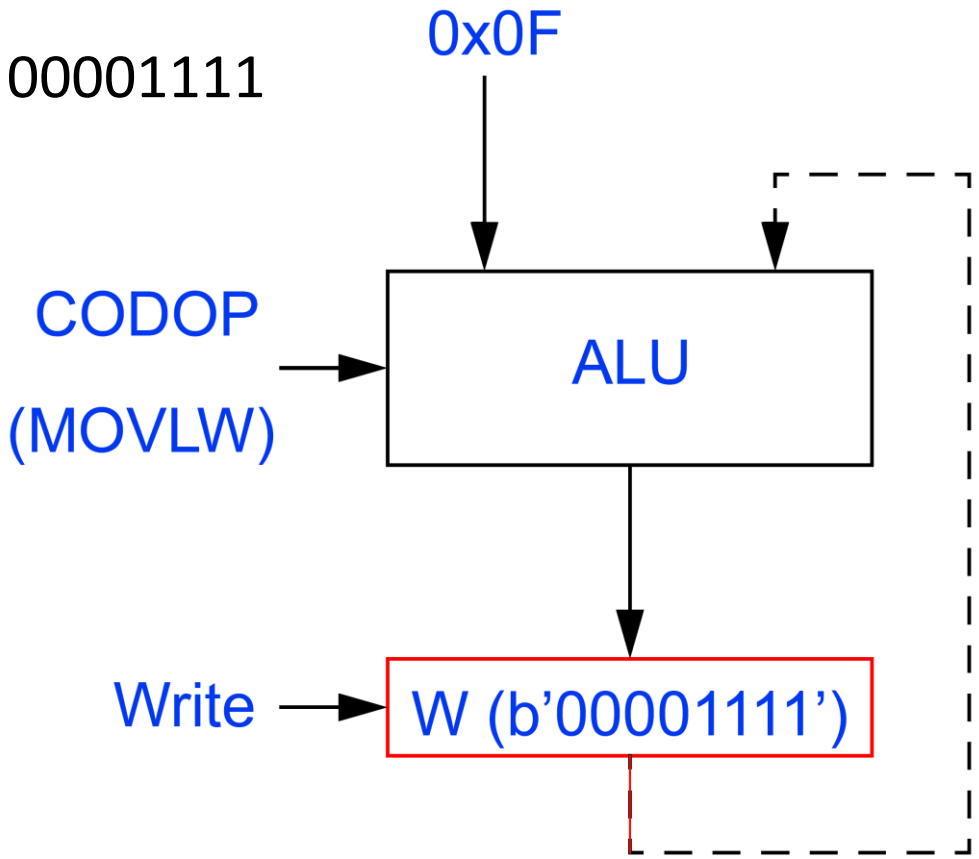
UN REGISTRE PARTICULIER: L'ACCUMULATEUR – Opérations simples

- Accumulateur (W ou WREG ou Work Register): registre particulier qui contient un opérande avant d'exécuter l'instruction et qui peut contenir le résultat après exécution de l'instruction

Exemple 1: MOVLW 0x0F ; W contient 00001111

Move literal to W valeur

(notations équivalentes:
0x0F
h'0F'
b'00001111'
d'15')



literal (ou adressage immédiat):

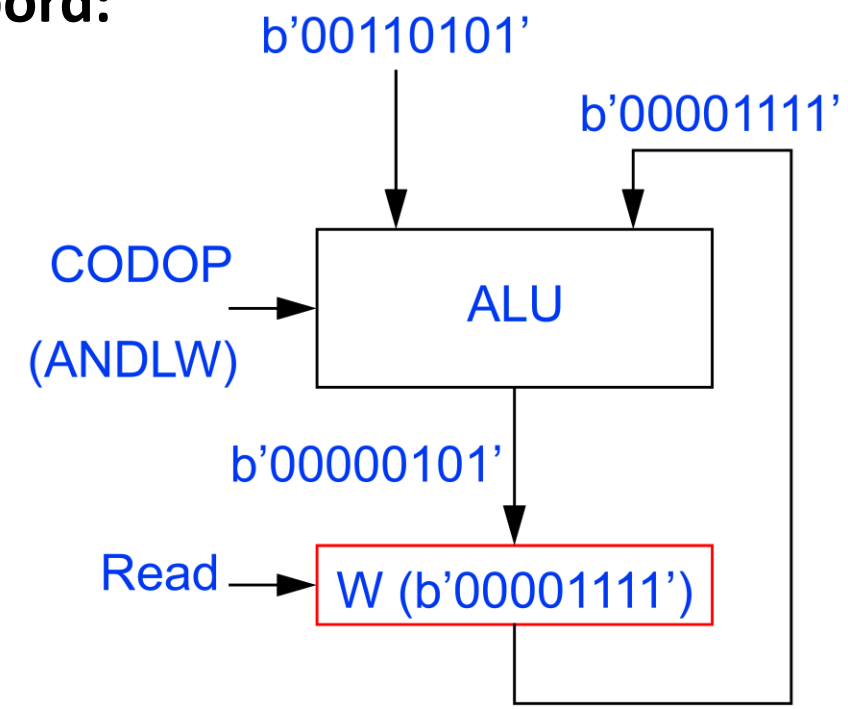
on donne directement une valeur

UN REGISTRE PARTICULIER: L'ACCUMULATEUR – Opérations simples

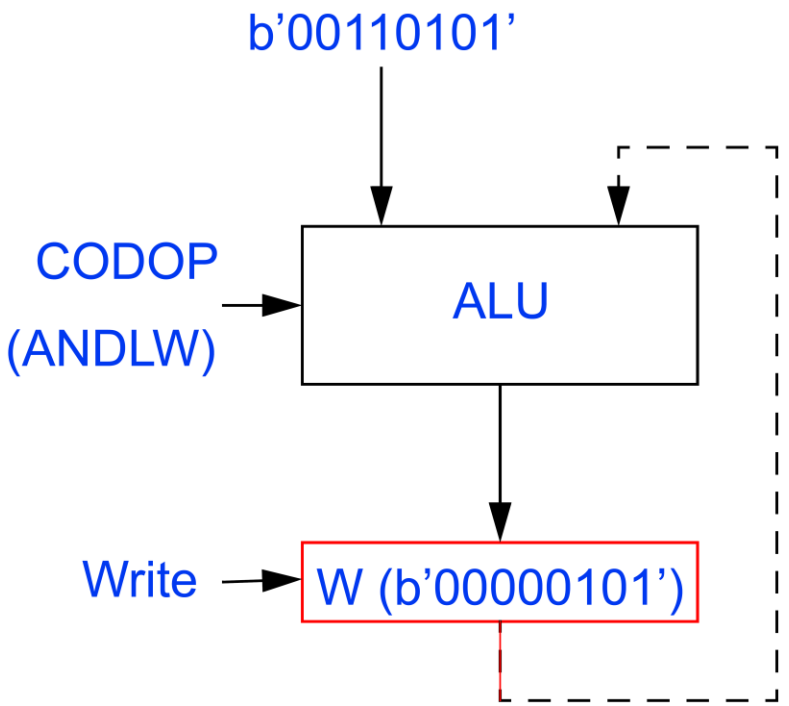
Exemple 1: ANDLW b'00110101' ; W contient maintenant le résultat du ET logique
; de 00001111 et de 00110101

AND literal with W

D'abord:



Puis:



UN REGISTRE PARTICULIER: L'ACCUMULATEUR – Opérations « complexes »

Exemple 3: faire la somme de R1 et R2, stocker le résultat dans R1
(R1 et R2 sont deux registres (deux "cases") de la mémoire RAM,
qui contiennent chacun une donnée codée sur un octet)

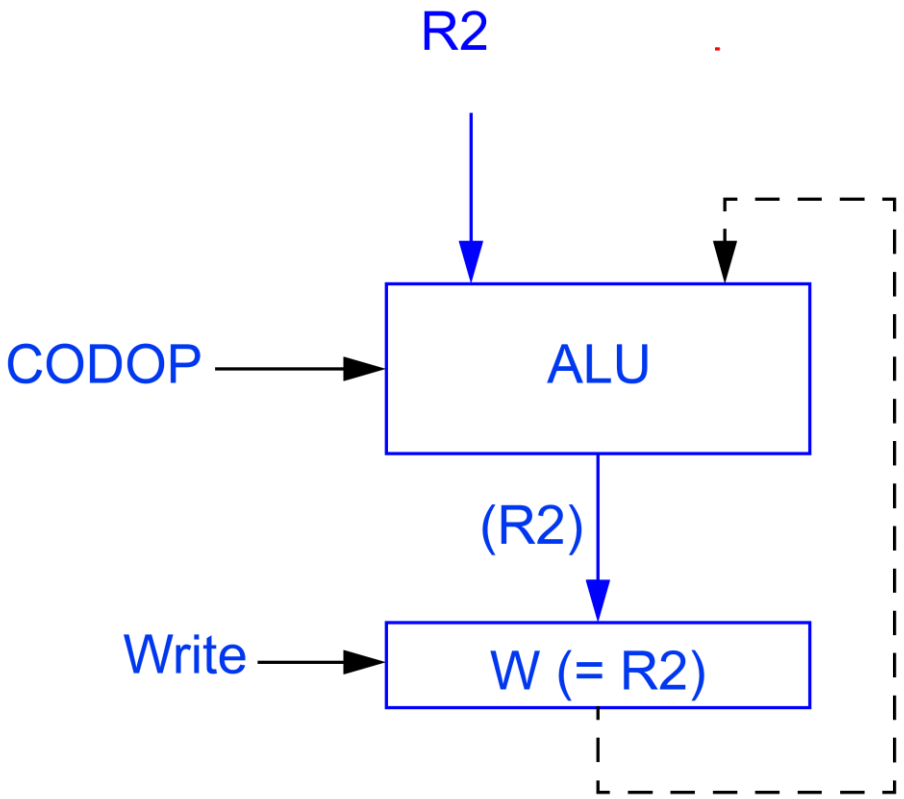
MOVF R2, 0 ; charge l'accumulateur avec le contenu de R2

Move f

Adresse du registre R2

Destination (résultat placé dans W)

(possible de n'indiquer que son nom: le compilateur le remplacera par son adresse)



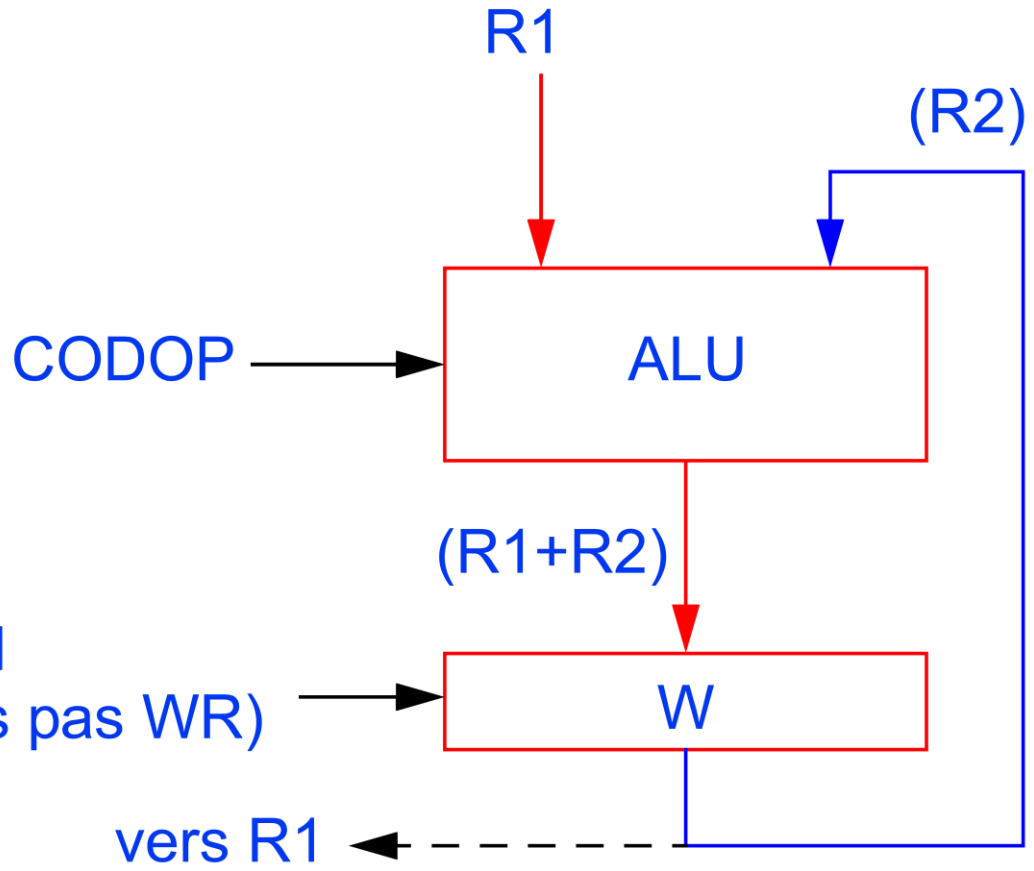
UN REGISTRE PARTICULIER: L'ACCUMULATEUR – Opérations « complexes »

ADDWF R1, 1

ADD W and f

Adresse du registre R1

Destination (résultat placé dans R1)



Remarque :

A la fin, W contient toujours la donnée du registre R2.

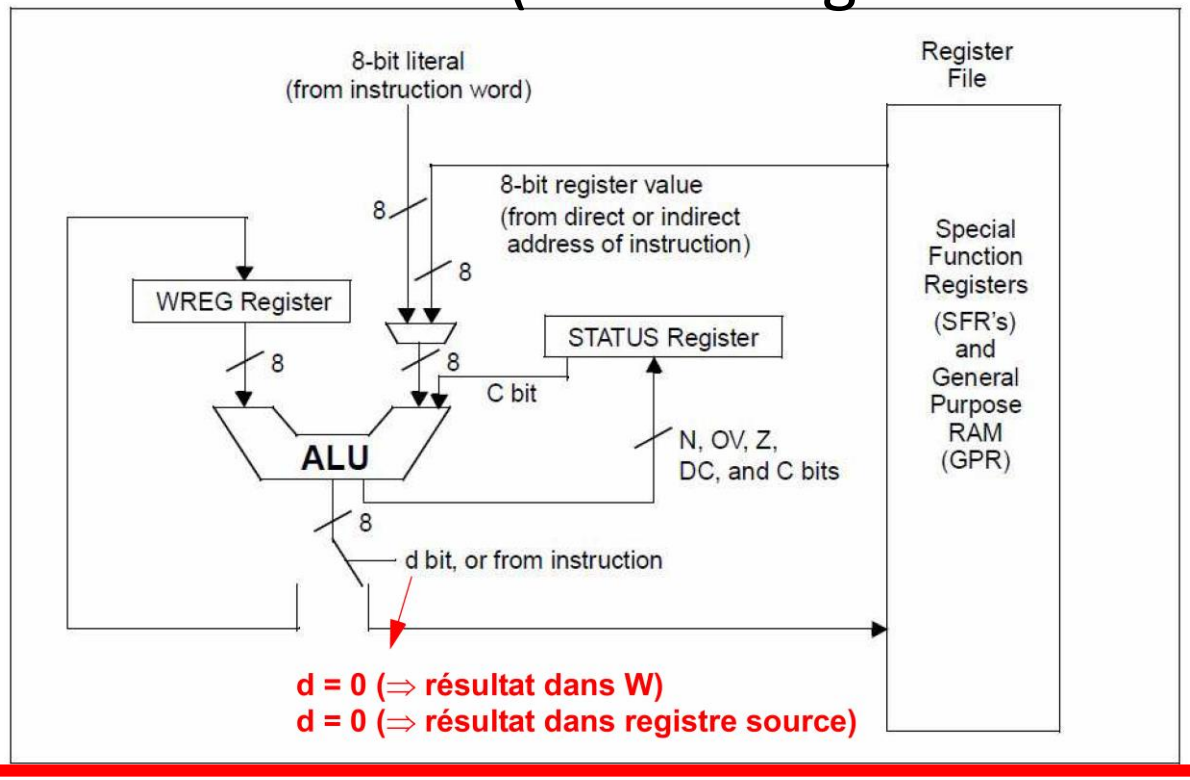
Avec **ADDWF R1, 0,** il aurait contenu à la fin le résultat de R1 + R2

UN REGISTRE PARTICULIER: L'ACCUMULATEUR – *destination d'une opération*

- Sur certains micro-contrôleurs, le résultat d'une opération est systématiquement dans l'accumulateur (exemple du mC ST7)

  ⇒ il faut une instruction supplémentaire pour stocker le résultat dans la RAM

- Sur les PIC18, le résultat d'une opération est au choix dans l'accumulateur ou remis directement dans la RAM (dans le registre source)



LE REGISTRE D'ÉTAT – Exemple sur PIC18: le registre STATUS (registre d'état)

- Contient le mot d'état (i.e. l'ensemble des indicateurs générés par la ***dernière instruction***)
- Les bits d'état sont notamment utilisés par les instructions de ***branchement conditionnel***.

bit C (Carry): retenue sur opération 8 bit
(utilisé notamment par BC (Branch if Carry) et BNC)

bit DC (Digit carry): retenue entre 4ème et 5ème bit lors d'une addition (utile en BCD)

bit Z (Zero): le résultat de la dernière opération est 0
(utilisé par BZ et BNZ)

bit OV (Overflow): le résultat de la dernière opération en complément en 2 entraîne un dépassement de dynamique sur 8 bits
(utilisé par BOV et BNOV)

bit N (Negative): c'est la recopie du MSB qui indique le signe en codage complément à 2
(utilisé par BN et BNN)

LE REGISTRE D'ÉTAT – Utilisations du bit "Carry"

- La retenue (Carry) permet de réaliser des traitements sur des mots de plus de 8 bits

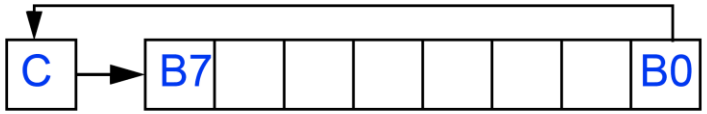
Exemple 1 : addition de deux mots de 16 bits (M1,M2) résultat dans M3

pour le LSB:
```
MOVF LSBM1, 0
ADDWF LSBM2, 0
MOVWF LSBM3
```

pour le MSB:
```
MOVF MSBM1, 0
ADDWFC MSBM2, 0
MOVWF MSBM3
```

Exemple 2 : décalage d'un mot de 16 bits M1 d'une position vers la droite

```
rotate:   RRCF   register,1
```
(Rotate Right through Carry)




```
BCF   STATUS, C
RRCF  MSBM1, 1
```

⟹

MSBM1:	0	M7						M1	Carry: M0
LSBM1:	L7							L0	

```
RRCF  LSBM1, 1
```

⟹

MSBM1:	0	M7						M1	
LSBM1:	M0	L7						L1	Carry: L0