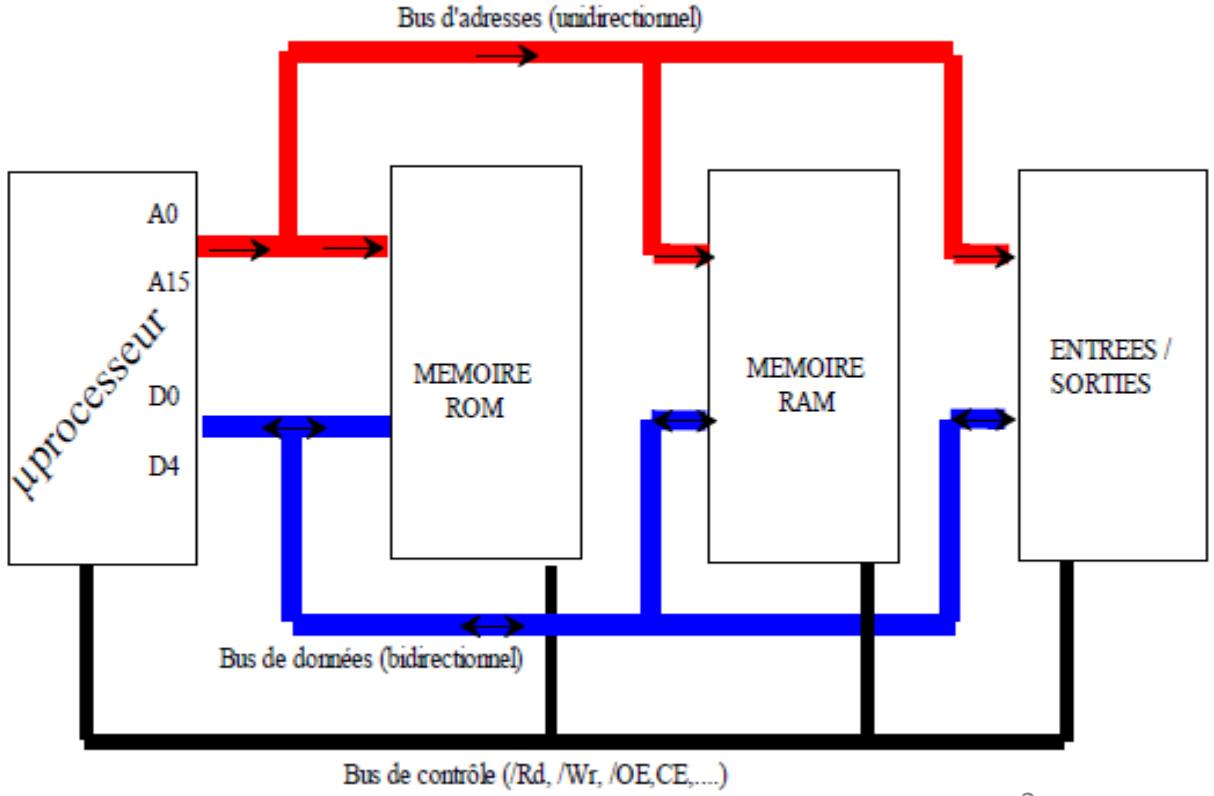
Architecture d’un microprocesseur

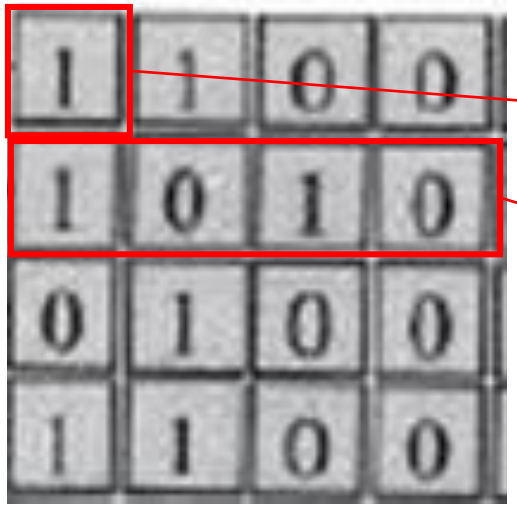
Architecture d'un ordinateur

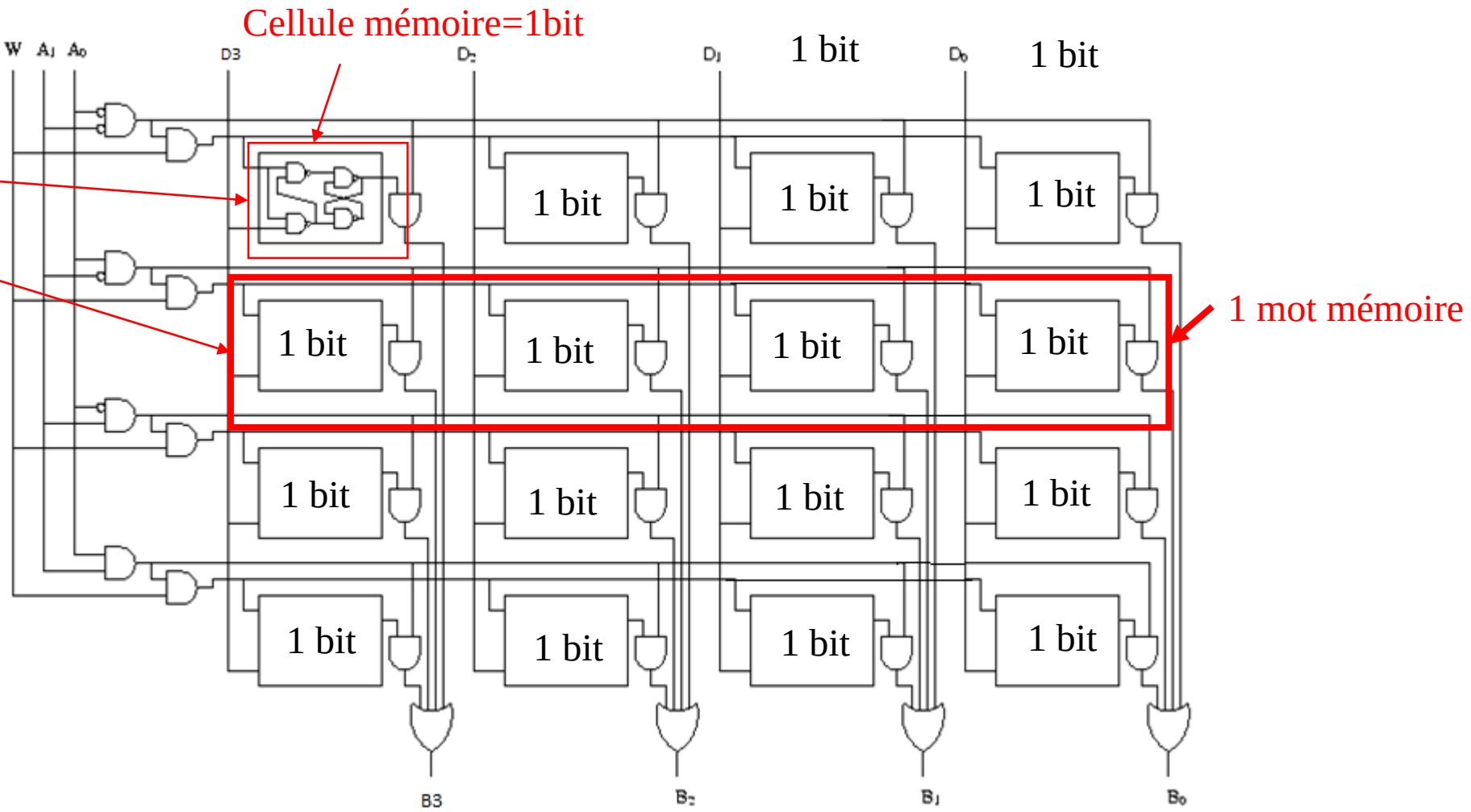
- Un ordinateur contient essentiellement
  - Microprocesseur (un CPU: Central Processing Unit=Unité centrale de traitement): son rôle est d'exécuter les programmes
  - Des mémoires: espaces de stockage des programmes et des données
    - Mémoire vive (RAM): barrettes mémoire
    - Mémoire morte (ROM):BIOS
  - Des entrées /sorties:
    - Interagir avec l'extérieur: moniteur, clavier, souris, etc.
    - Faire un stockage de masse: disque dur, clé USB, etc.
  - Des bus qui connectent ces composants



La mémoire

- Organisation
  - Organisation en grille





Exemple: mémoire de 4x 4bits

La mémoire

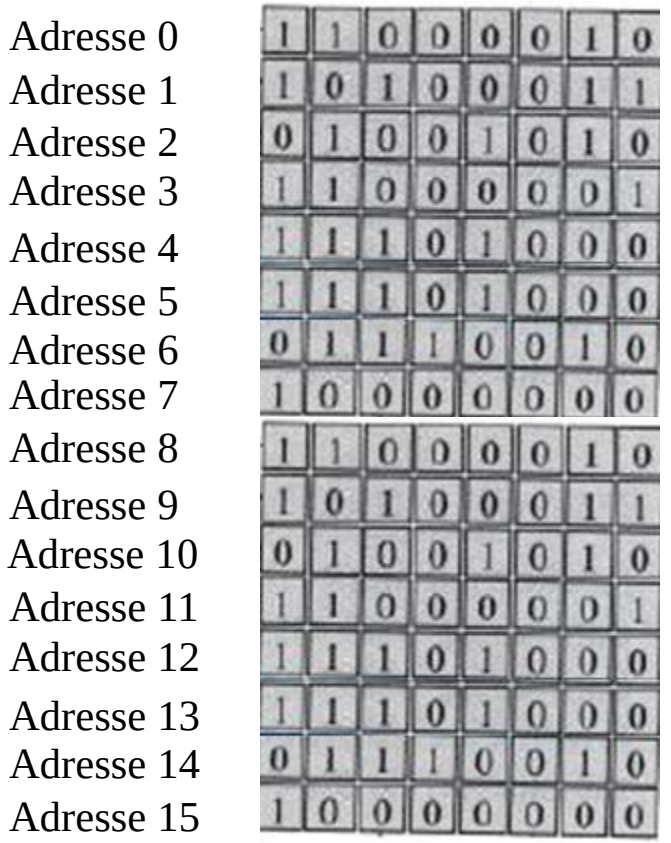
- La capacité

La capacité= taille d'un mot mémoire x nombre de mots dans la mémoire

Exemple: 8bits x 16 = 16 octets

La mémoire

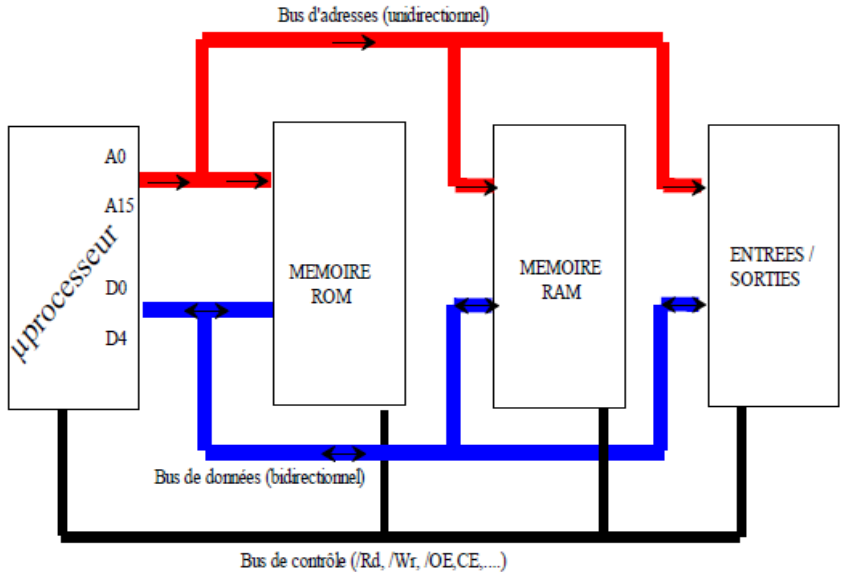
- Le uprocesseur communique avec la mémoire en lisant ou en modifiant le contenu d'un mot mémoire
- Pour accéder à un mot mémoire, le uprocesseur doit connaitre son emplacement (son adresse)
- Chaque mot mémoire a une adresse différente indiquant son emplacement
- L'adresse d'un mot mémoire est indépendante de son contenu



Le bus

- Les types de bus
  1) Bus d'adresse
     - Permet au micro-processeur d'indiquer l'adresse de la donnée à lire/écrire dans la mémoire ou l'E/S
     - Unidirectionnel: le micro-processeur envoie l'adresse de la donnée à lire/écrire

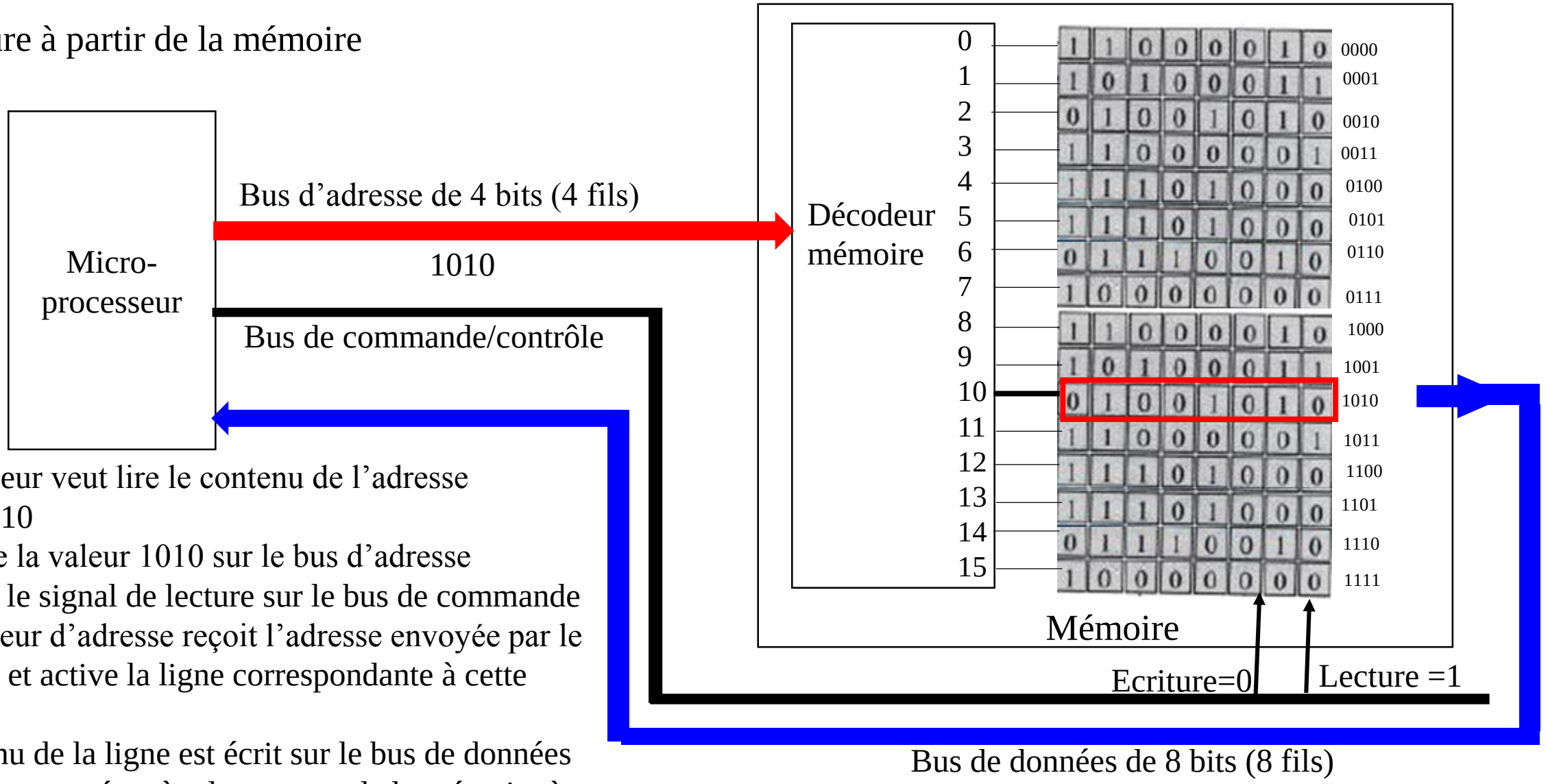
  2) Bus de données
     Deux sens
     a) Du processeur vers une mémoire ou une E/S
        Le processeur envoie une donnée à écrire dans une mémoire ou une E/S
     b) De la mémoire ou l'E/S vers le processeur
        Le processeur lit à partir de la mémoire ou l'E/S
        La mémoire ou l'E/S envoie la donnée sur le bus de données



  3) Bus de contrôle/commande
     - Permet entre autres d'indiquer s'il s'agit d'une lecture ou une écrire, de sélectionner la mémoire ou l'E/S à lire
     - Des fils dans les deux sens
       - Sortant du micro-processeur: Exemples: écriture/lecture, sélection de la mémoire, etc.
       - Vers le micro-processeur: Exemple: indiquer si le composant est prêt à communiquer avec le microprocesseur

Communication processeur/mémoire

- Lecture à partir de la mémoire

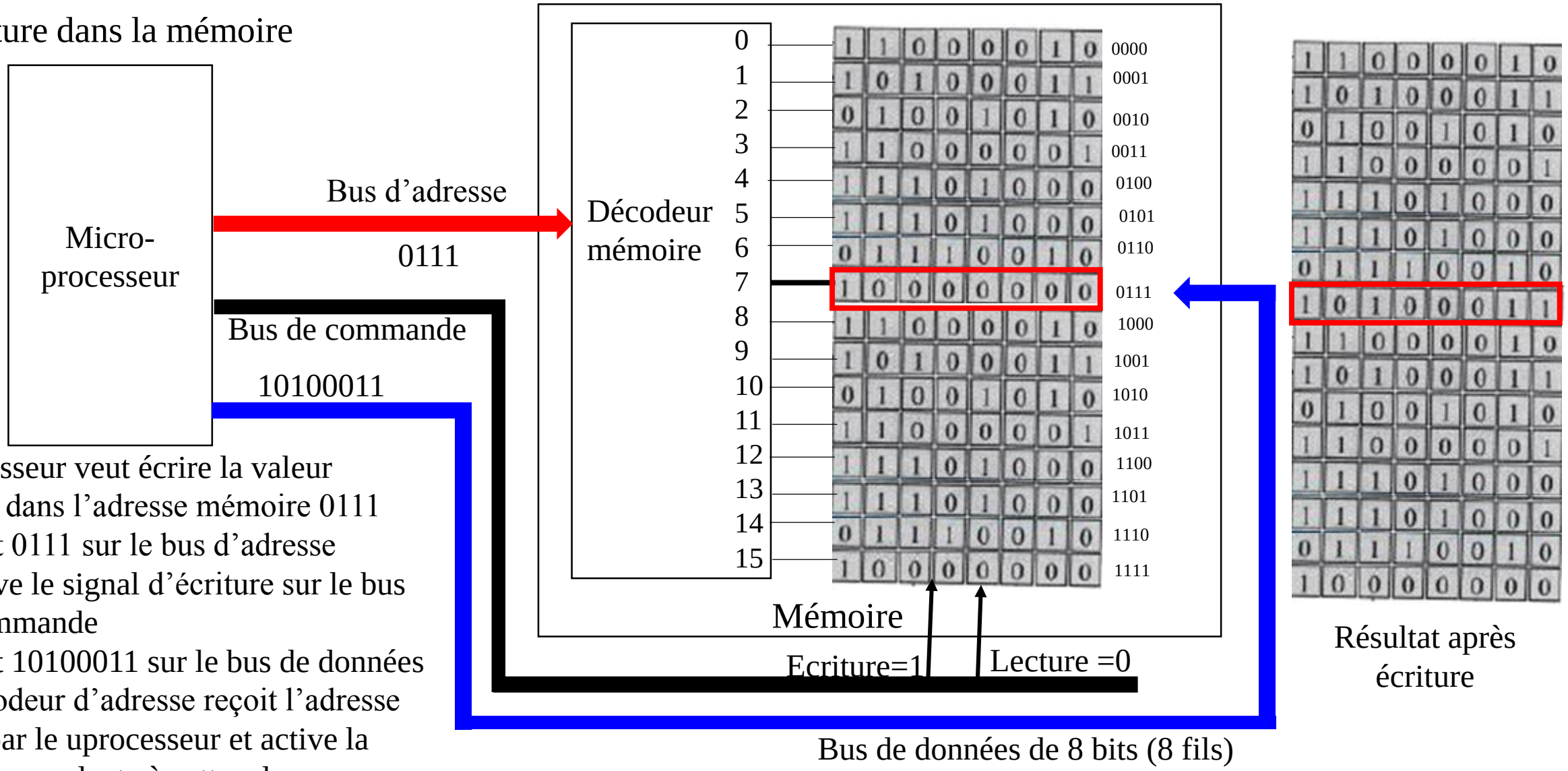


Le uprocesseur veut lire le contenu de l'adresse mémoire 1010

1) Il envoie la valeur 1010 sur le bus d'adresse
2) Il active le signal de lecture sur le bus de commande
3) Le décodeur d'adresse reçoit l'adresse envoyée par le uprocesseur et active la ligne correspondante à cette adresse
4) Le contenu de la ligne est écrit sur le bus de données
5) Le uprocesseur récupère le contenu de la mémoire à partir du bus de données

Communication processeur/mémoire

- Ecriture dans la mémoire



Le uprocesseur veut écrire la valeur 10100011 dans l'adresse mémoire 0111

1) Il écrit 0111 sur le bus d'adresse
2) Il active le signal d'écriture sur le bus de commande
3) Il écrit 10100011 sur le bus de données

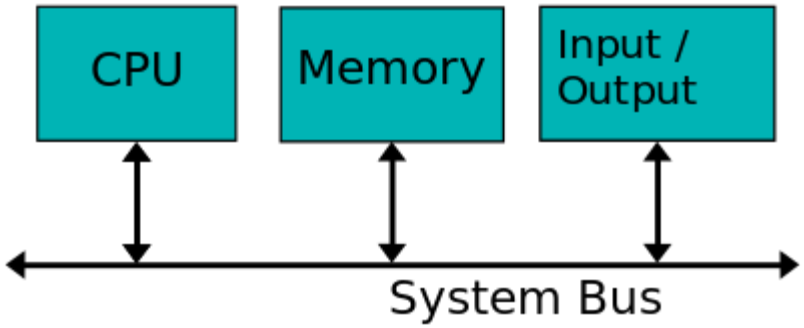
3) Le décodeur d'adresse reçoit l'adresse envoyée par le uprocesseur et active la ligne correspondante à cette adresse

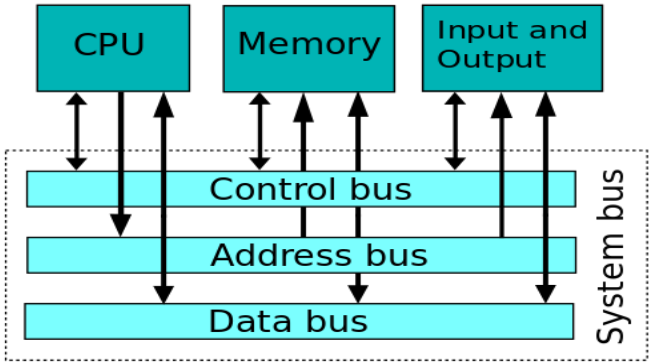
4) Le contenu du bus de données est écrit dans la ligne activée

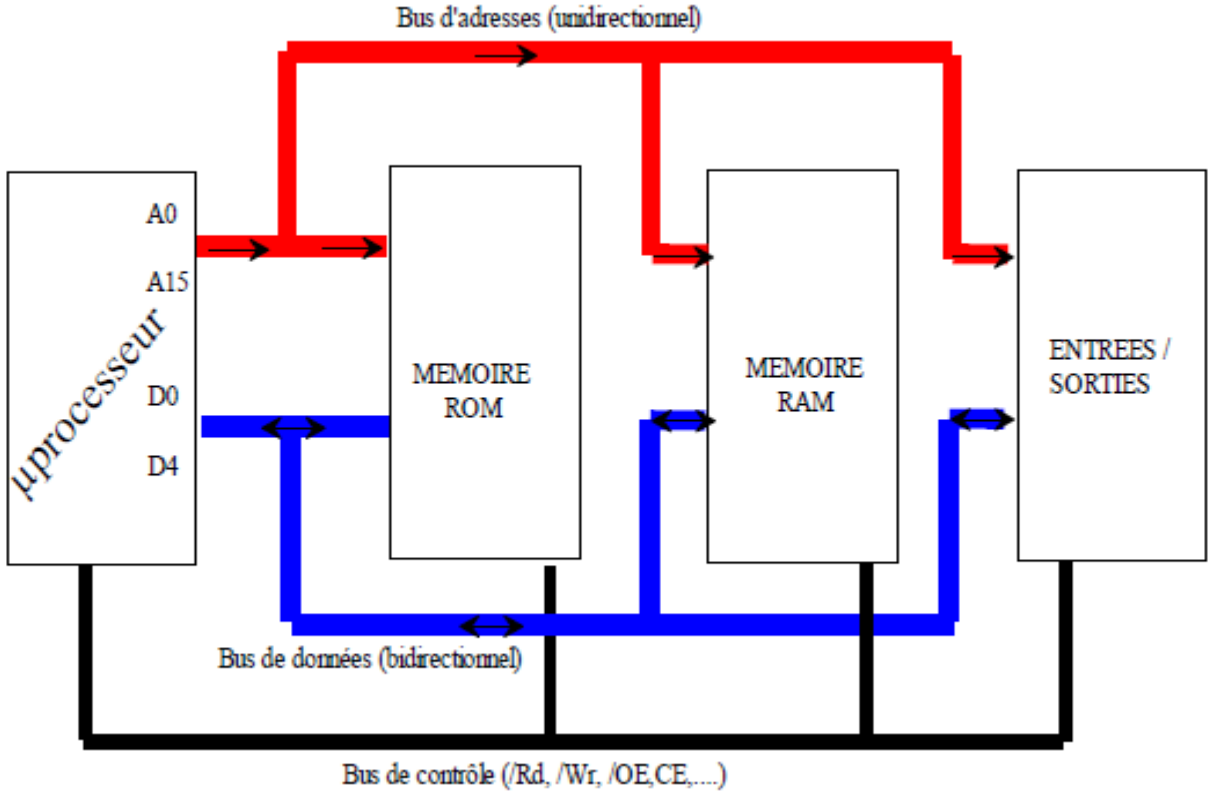
Le bus système

- Trois types de bus
  - Bus d'adresse
  - Bus de données
  - Bus de contrôle/ commande

- Ces trois bus forment le **bus système**





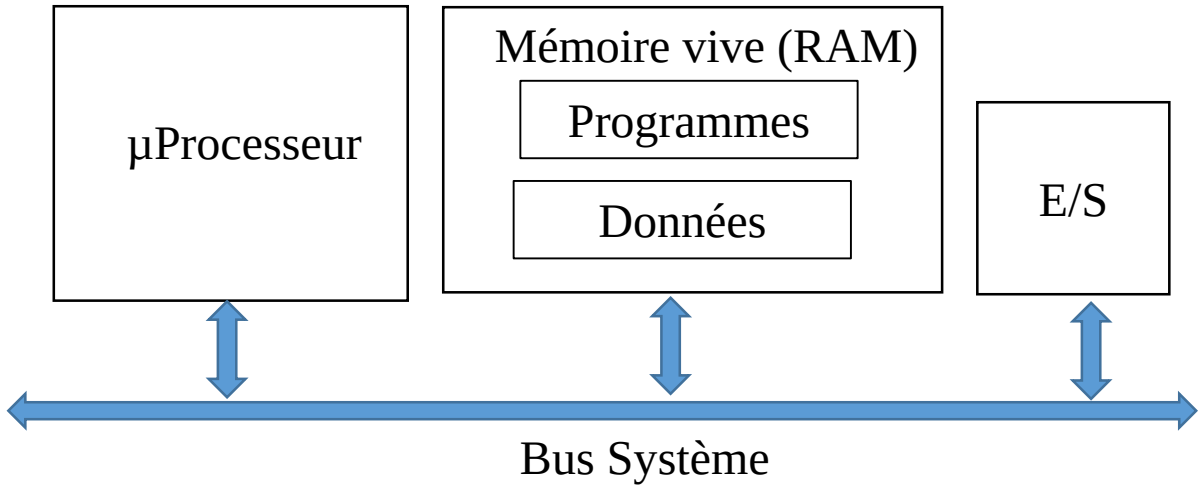


Deux architectures de base

- Un ordinateur a souvent une des architectures suivantes
  - Architecture de Von Neumann
  - Architecture de Harvard

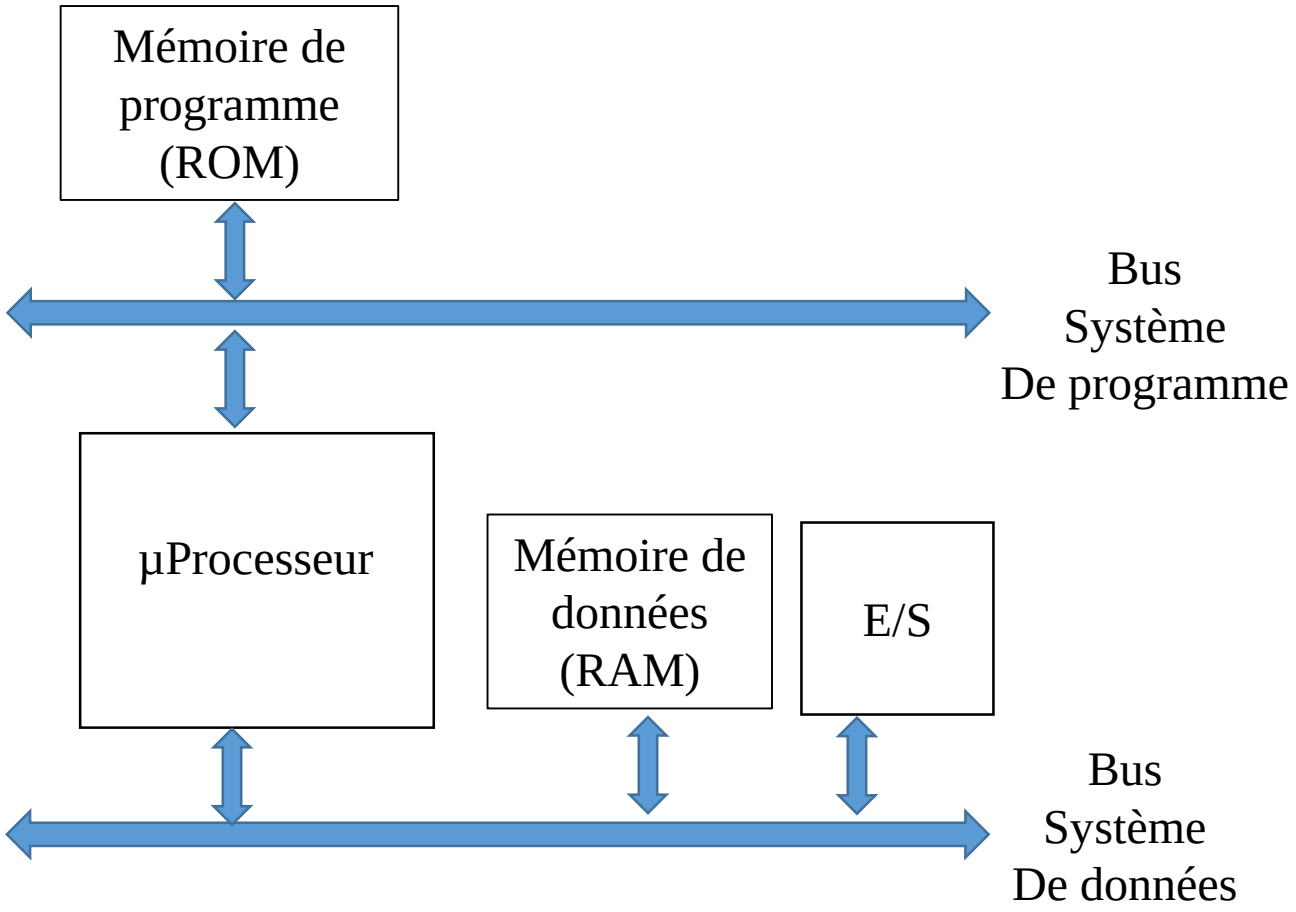
Deux architectures de base

- Architecture de Von Neumann
  - Particularité:
    - La mémoire contient les programmes (instructions) et les données (variables, la pile, etc.)
    - **Un seul bus** pour le transfert des instructions et des données
  - Avantage: architecture simple: le processeur accède à la même mémoire pour lire les instructions et les données
  - Inconvénient:
    - Rapidité limitée: le processeur ne peut accéder qu'à une instruction ou une donnée à un instant donné, il ne peut pas faire les deux en même temps



Deux architectures de base

- Architecture Harvard
  - Particularité:
    - Deux mémoires séparées
      - Une mémoire de programme,
      - Une mémoire de données
    - Le processeur utilise **deux bus différents** pour accéder au programme et aux données



- Avantage:
  - Plus rapide: le processeur peut accéder à une instruction et une données en même temps
- Inconvénient:
  - Une implémentation plus complexe du processeur qui doit gérer l'accès aux deux mémoires
  - Deux bus → un coût plus élevé

Deux architectures de base

L'architecture de Von Neumann

- Utilisation: Ordinateurs
- Raison:
  - Applications variées parmi lesquelles il y a celles qui nécessitent une grande mémoire de programmes et d'autres nécessitent une grande mémoire de données

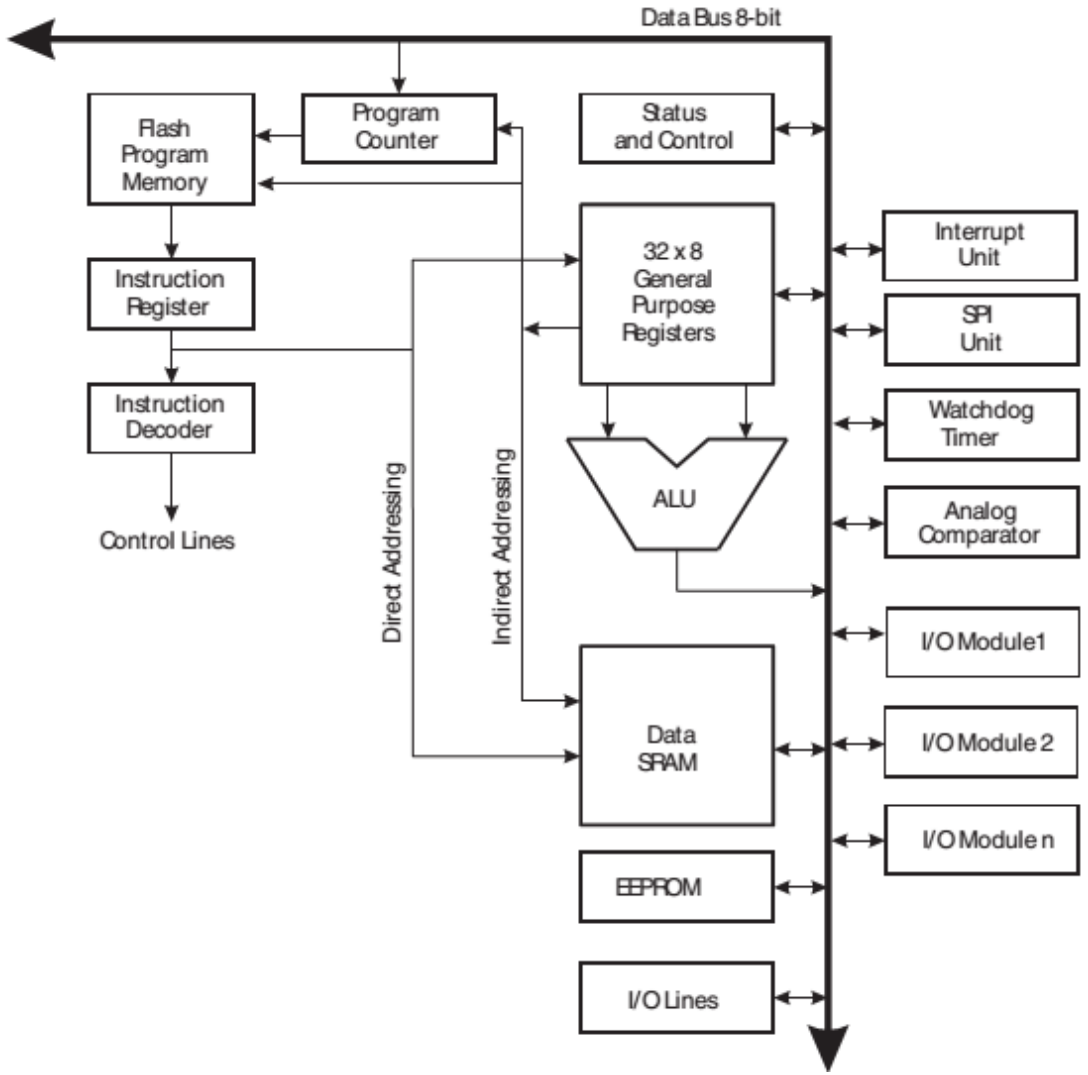
  → Avoir une seule mémoire permet une meilleure utilisation des ressources mémoire

L'architecture Harvard

- Utilisations:
  - Certains microcontrôleurs (exemple: Arduino Uno)
  - Processeurs de traitement de signal (DSP)
- Raison:
  - Pour un système embarqué qui exécute la même application, l'architecture Harvard convient très bien puisqu'on peut optimiser les tailles des mémoires données et programme selon le besoin

Deux architectures de base

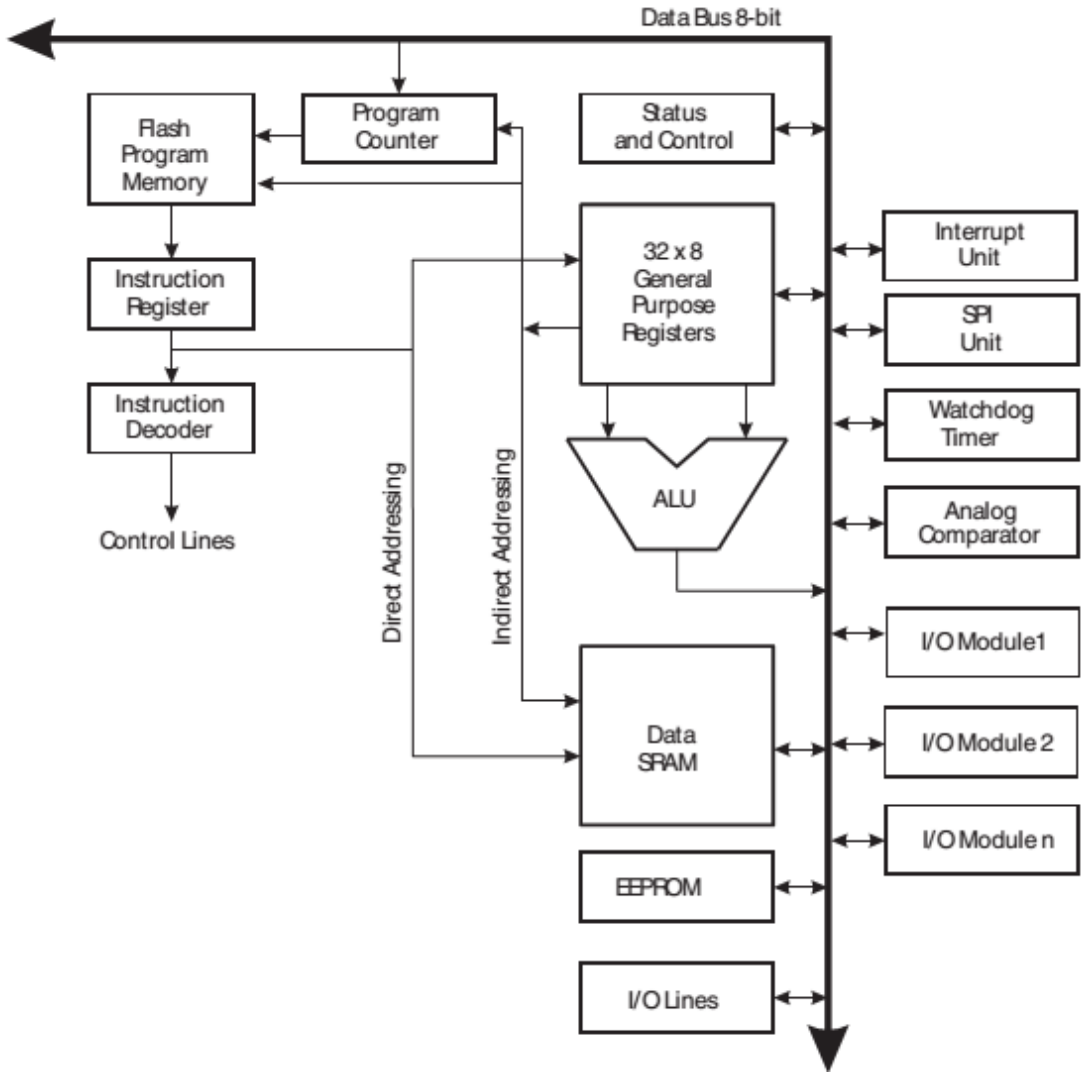
- Exercice



A quel type d'architecture cela correspond-il ?

Deux architectures de base

- Solution



A quel type d'architecture cela correspond-il ?

- Le chemin d'accès à la mémoire programme est différent de l'accès aux données

→ architecture Harvard.

Deux architectures de base

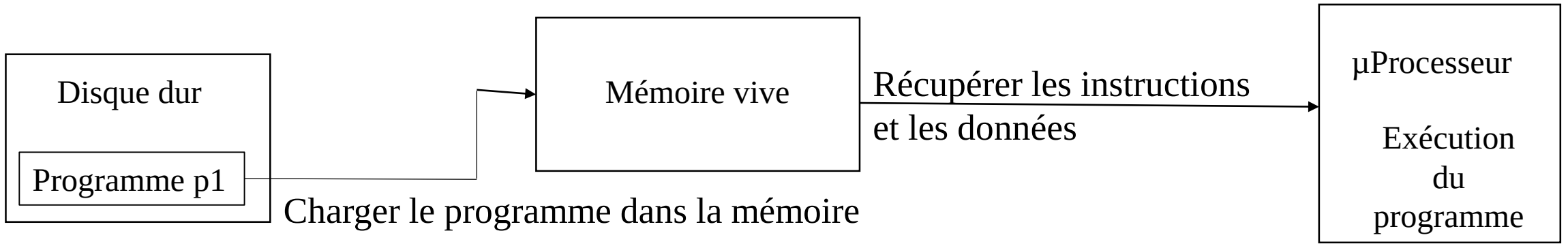
- Exercice
  - On connecte à un microprocesseur une mémoire ROM d'instructions et une mémoire RAM de données sur le même bus. La ROM est mappée entre les adresses A0000h et A7FFFh et la RAM entre 62000h et 62FFFh. A quel type d'architecture cela correspond-il ?

Deux architectures de base

- Solution
  - On connecte à un microprocesseur une mémoire ROM d'instructions et une mémoire RAM de données sur le même bus. La ROM est mappée entre les adresses A0000h et A7FFFh et la RAM entre 62000h et 62FFFh. A quel type d'architecture cela correspond-il ?

  - Le même bus pour les instructions et les données → architecture de Von Neumann

Exécution des programmes



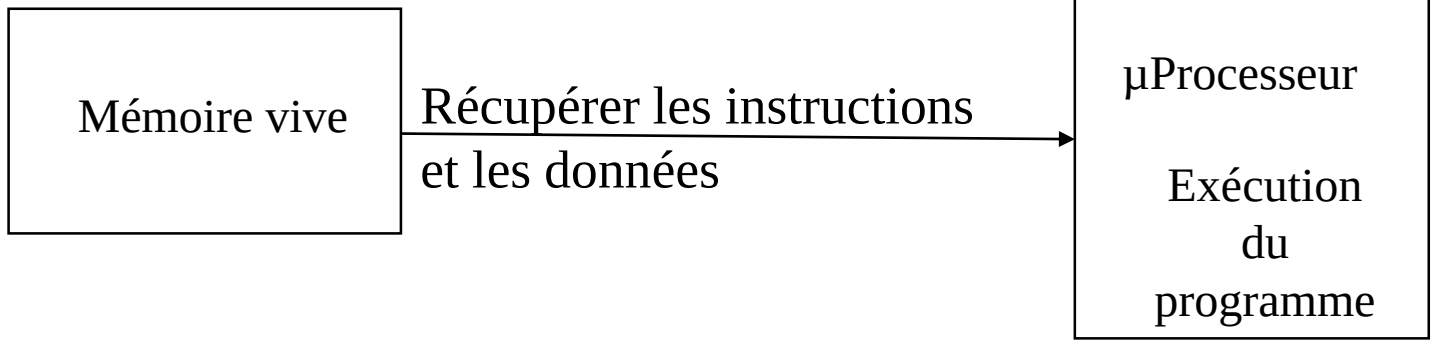
Traitement séquentiel des instructions

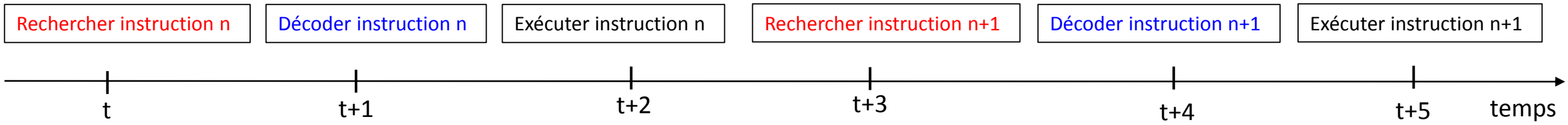
Cycle FDX (Fetch Decode Execute)

1. Lire une instruction de la mémoire,
2. Décoder cette instruction : traduire en commandes internes
3. Exécuter cette instruction

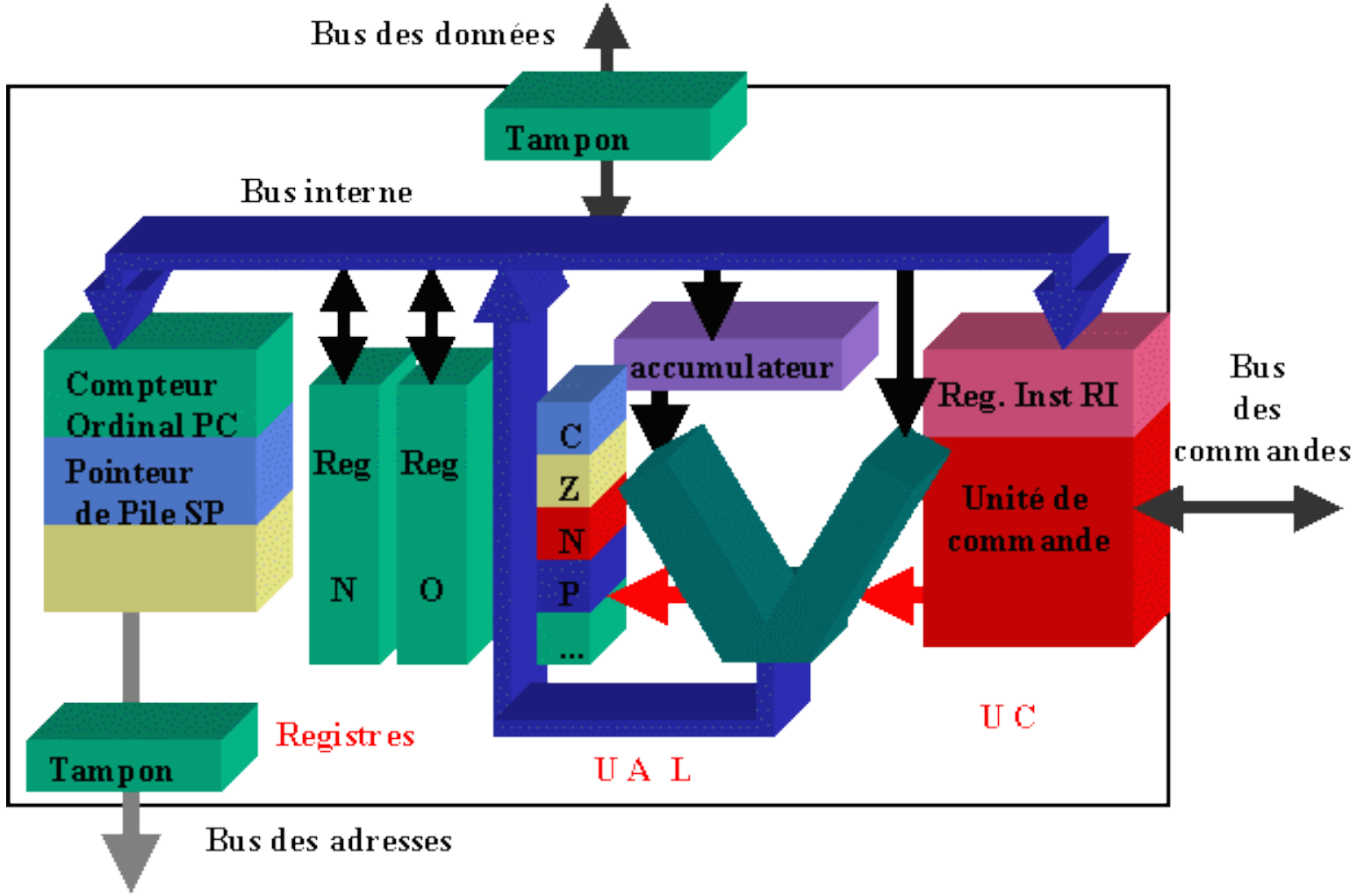
On trouvera en permanence le cycle suivant :

1. recherche d'instruction (Fetch)
2. décodage d'instruction (Decode)
3. exécution de l'instruction (Execute)
4. recherche de l'instruction suivante (Fetch)
5. décodage d'instruction (Decode)
6. exécution de l'instruction (Execute)
7. et ainsi de suite…

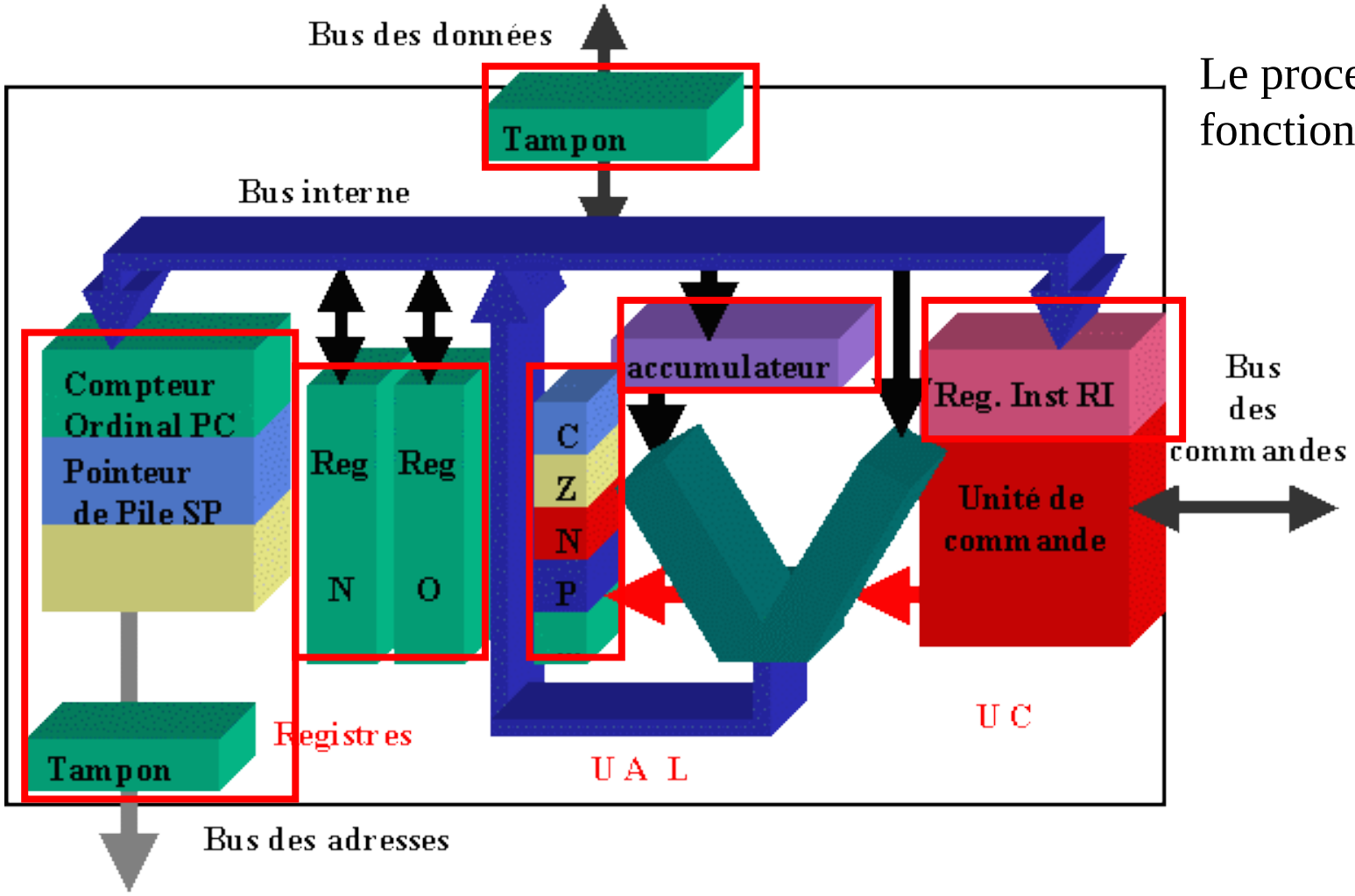




Structure interne d'un µProcesseur



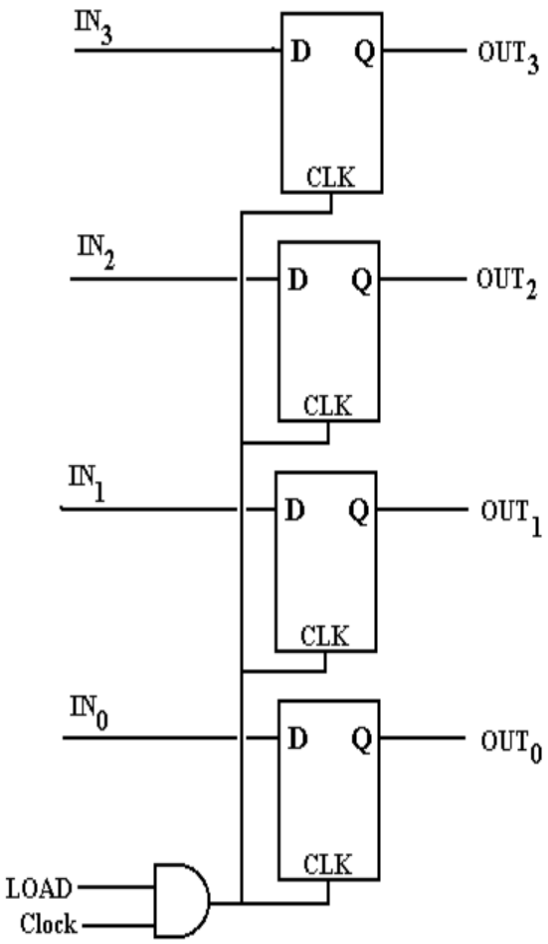
Structure interne d'un µProcesseur



Le processeur contient des registres qui ont des fonctionnalités diverses

Structure interne d'un µProcesseur

- Les registres: ensemble de bascules synchrones qui servent à stocker des informations



Registre à 4 bits

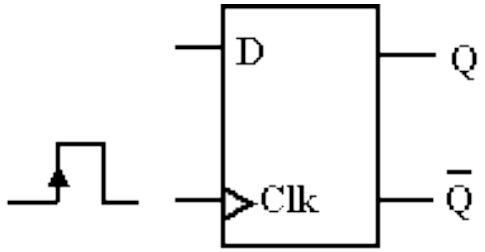


Table de vérité

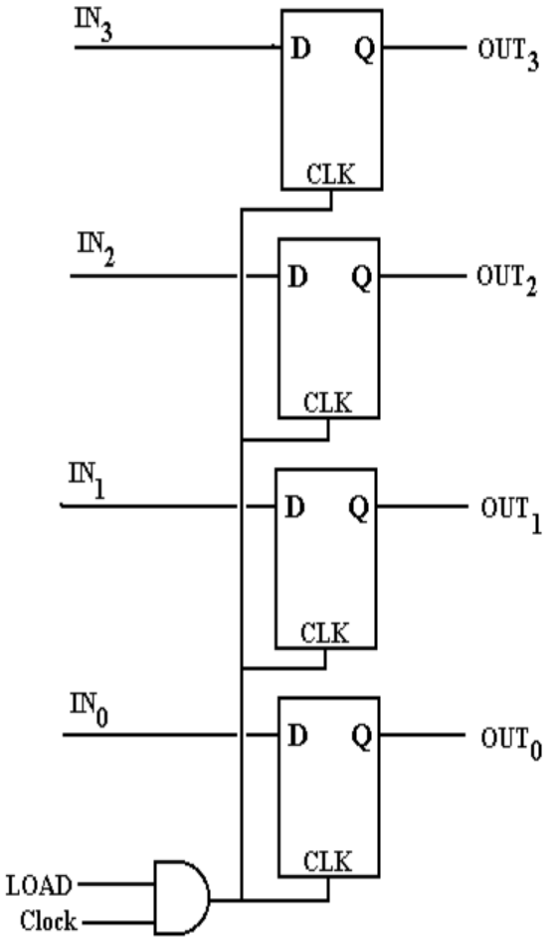
D	H	Q_n	$\overline{Q_n}$
0	↗	0	1
1	↗	1	0
X	0	Q_{n-1}	$\overline{Q_{n-1}}$
X	1	Q_{n-1}	$\overline{Q_{n-1}}$

La sortie de bascule ne change de valeur que sur front de l'horloge
→ Elle est capable de maintenir son « contenu » (fonction mémoire) en absence du front d'horloge

- Si on veut maintenir le contenu des bascules, Load=0, la sortie de la porte ET est 0, les signaux d'horloge qui vont vers les bascules sont nuls, donc les bascules maintiennent leurs contenus
- Si on veut écrire une nouvelle donnée dans les bascules, on la met sur les entrées D, et on met Load=1, la sortie de la porte ET est égal au signal Clock (signal carré), les bascules D changent de contenu sur front montant, après un cycle d'horloge on peut remettre LOAD à 0

Structure interne d'un µProcesseur

- Les registres: ensemble de bascules synchrones qui servent à stocker des informations



Registre à 4 bits

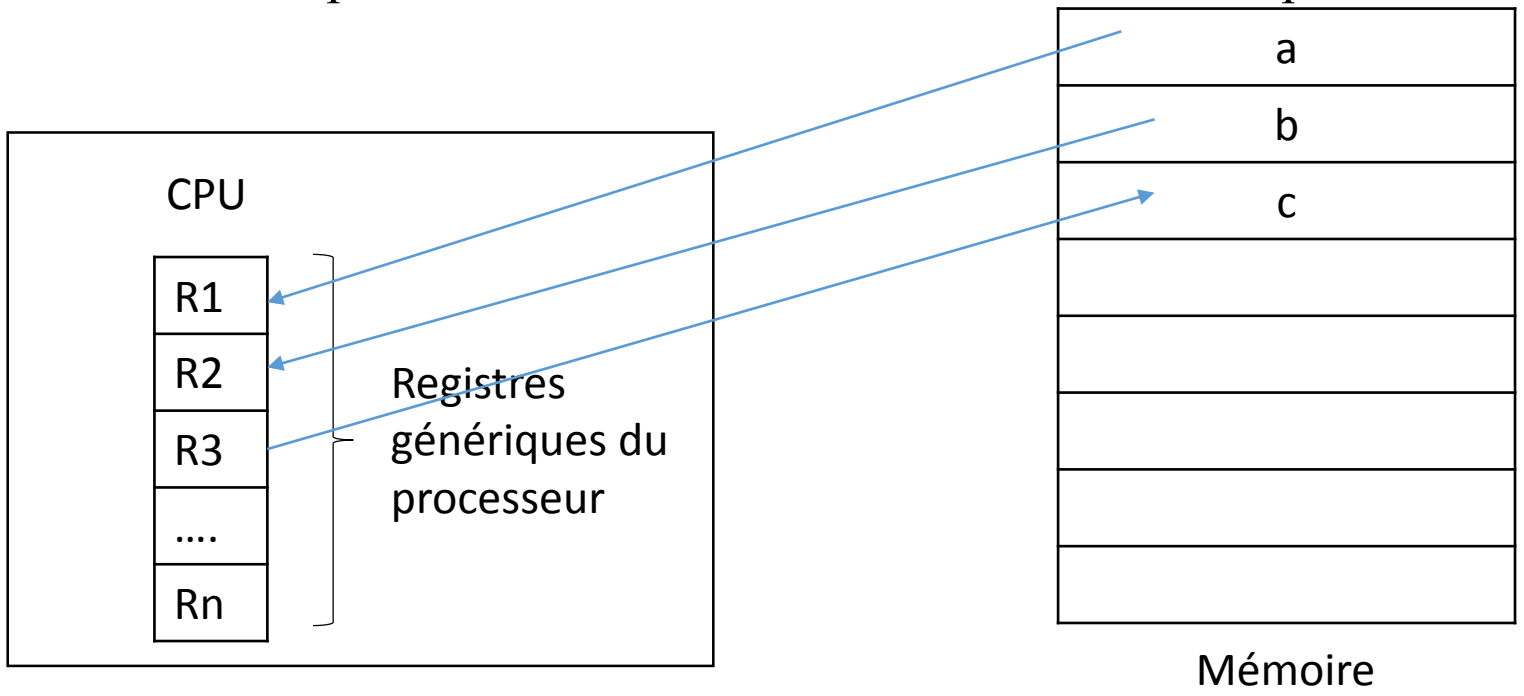
Attention:

- Les registres sont des endroits de stockage internes au processeur
- La mémoire principale (RAM) se trouve à l'extérieur du processeur

Structure interne d'un µProcesseur

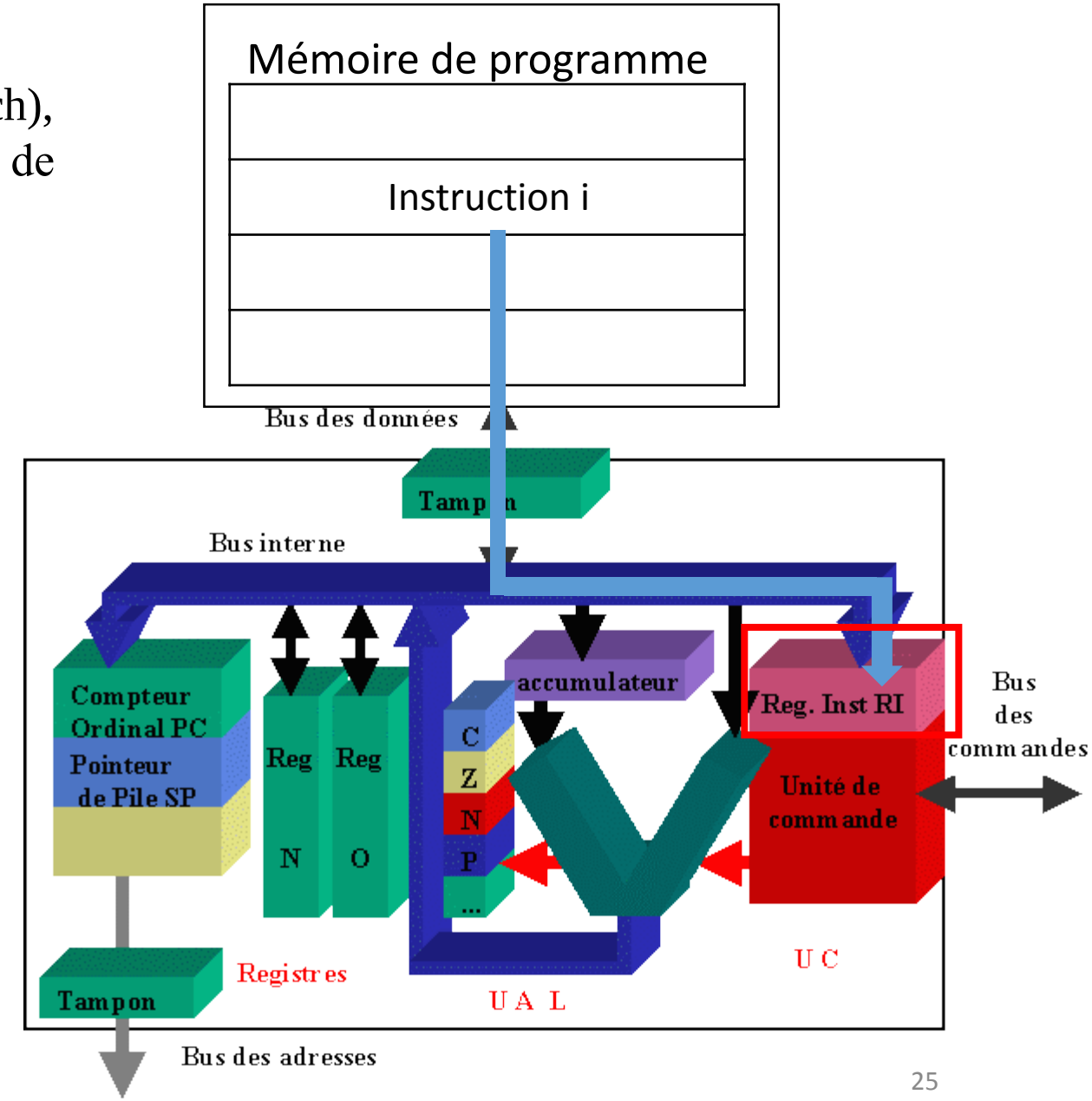
Exemple d'utilisation:

- Le processeur contient des registres génériques qu'il utilise pour ses calculs intermédiaires
  - Exemple: Le processeur veut exécuter l'instruction **c=a+b;** (a, b et c sont trois variables entières stockées dans la mémoire)
  - Le processeur envoie à la mémoire des requêtes pour récupérer les valeurs de a et de b et les met dans les registres R1 et R2 respectivement, il fait l'addition et stocke le résultat dans le registre R3
  - Le contenu de R3 est par la suite écrit dans la case mémoire correspondante à l'adresse de c



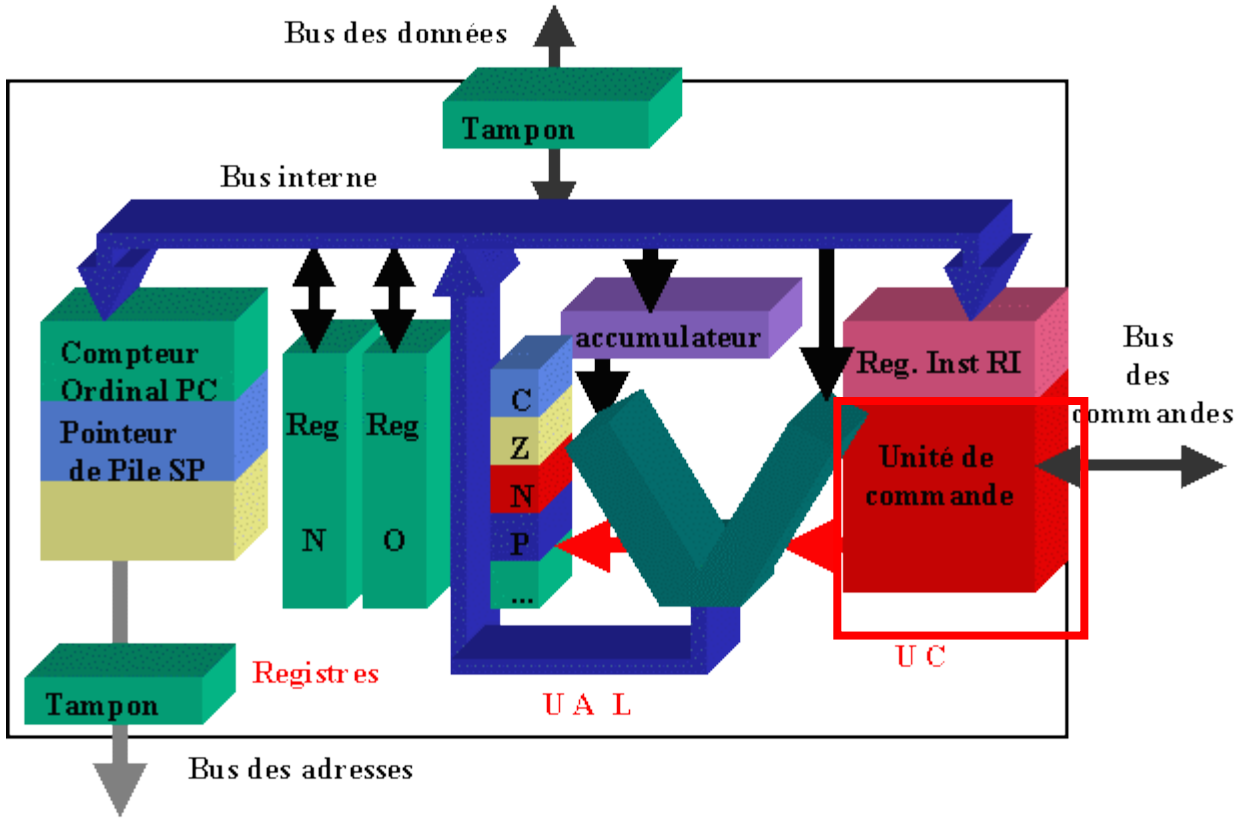
Structure interne d'un µProcesseur

- Registre d'instruction (RI)
  - Dans la phase de recherche de l'instruction (Fetch), l'instruction est récupérée à partir de la mémoire de programme et chargée dans le RI



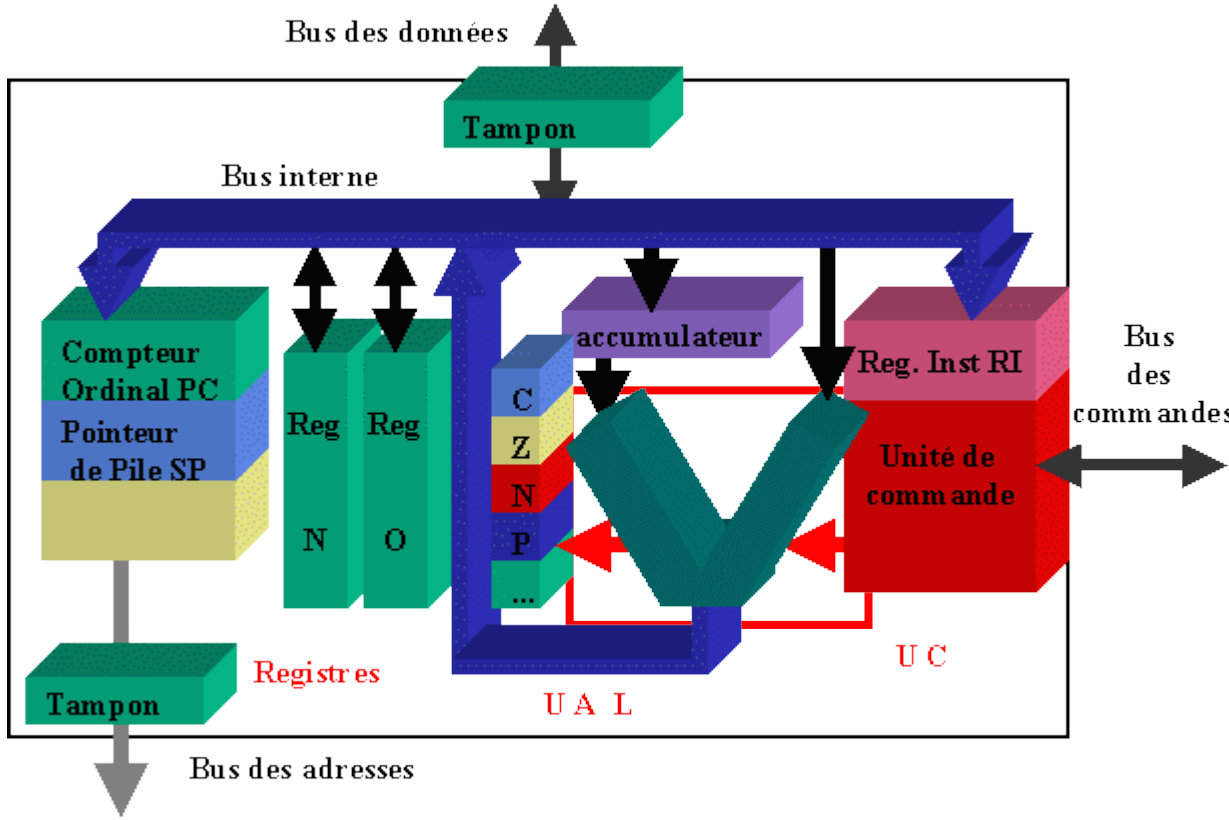
Structure interne d'un µProcesseur

- Unité de contrôle
  - Récupère le contenu du RI
  - Décode l'instruction: compare le code de l'instruction reçue avec la table des instructions du processeur (jeu d'instructions)
  - Donne les ordres à la logique d'interface de bus pour aller chercher les données dans la mémoire
  - Donne les ordres à l'ALU pour exécuter une instruction
  - Gère le travail des différents sous-systèmes et leur cohabitation dans le même boîtier.



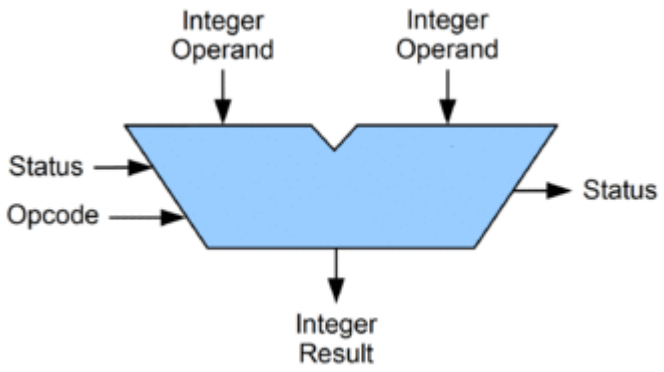
Structure interne d'un µProcesseur

- Unité Arithmétique et Logique (ALU)
  - Destinée à faire des calculs
    - Opération arithmétiques : addition , soustraction, incrémentation, décrémentation, etc.
    - Opération logiques : ET, OU, comparaison, décalage, etc.
  - Reçoit des données venant des accumulateurs, des registres, …
  - Reçoit des ordres de l'unité de commande pour savoir quelles opérations effectuer et dans quel ordre

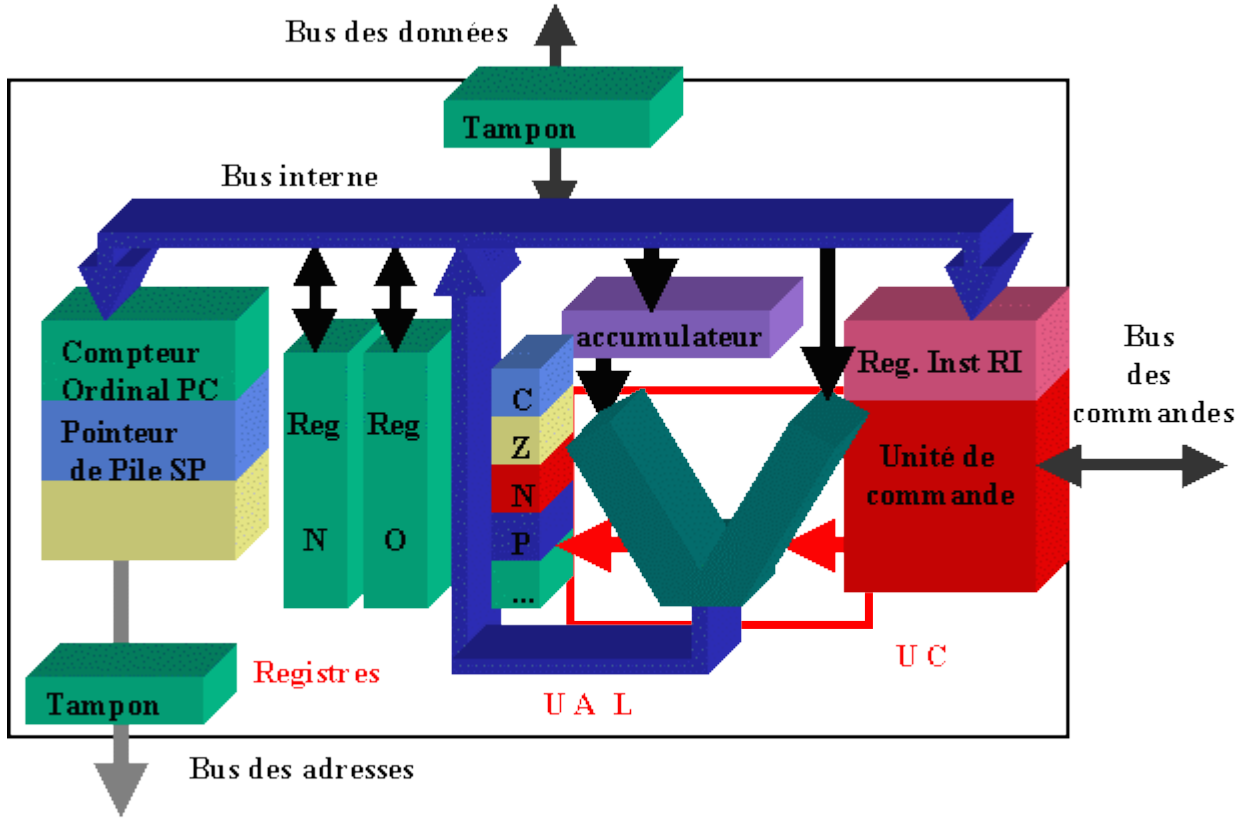


Structure interne d'un µProcesseur

- Unité Arithmétique et Logique (ALU)



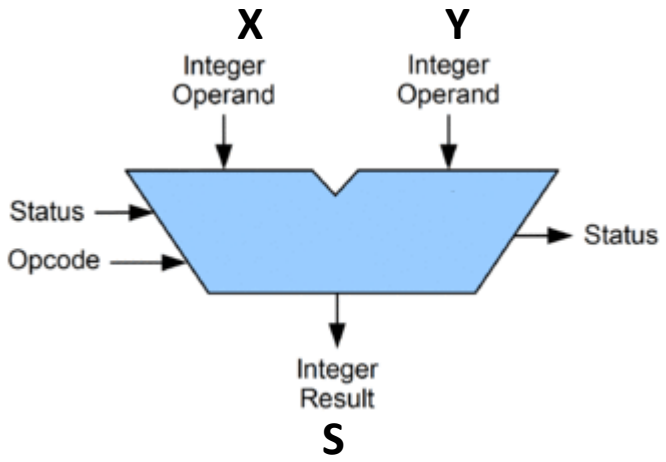
- L'Opcode correspond au code de l'opération à exécuter (ADD, MUL, etc.) parmi les instructions disponibles dans le jeu d'instructions du processeur



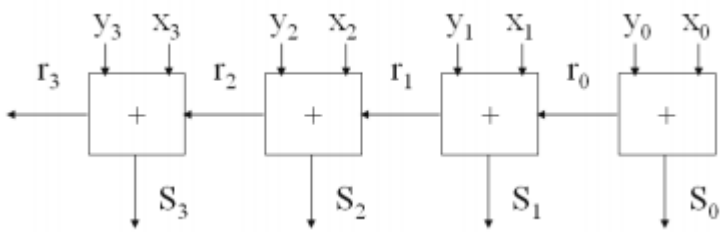
Structure interne d'un µProcesseur

- Unité Arithmétique et Logique (ALU)

  - L'addition dans l'UAL est implémentée par un ensemble d'additionneurs 1-bit (pour un processeur 32-bit, on en a 32)
  - Un additionneur 1-bit a deux entrées d'1 bit chacune et deux sorties (la somme et la retenue)



Carry, Overflow, …



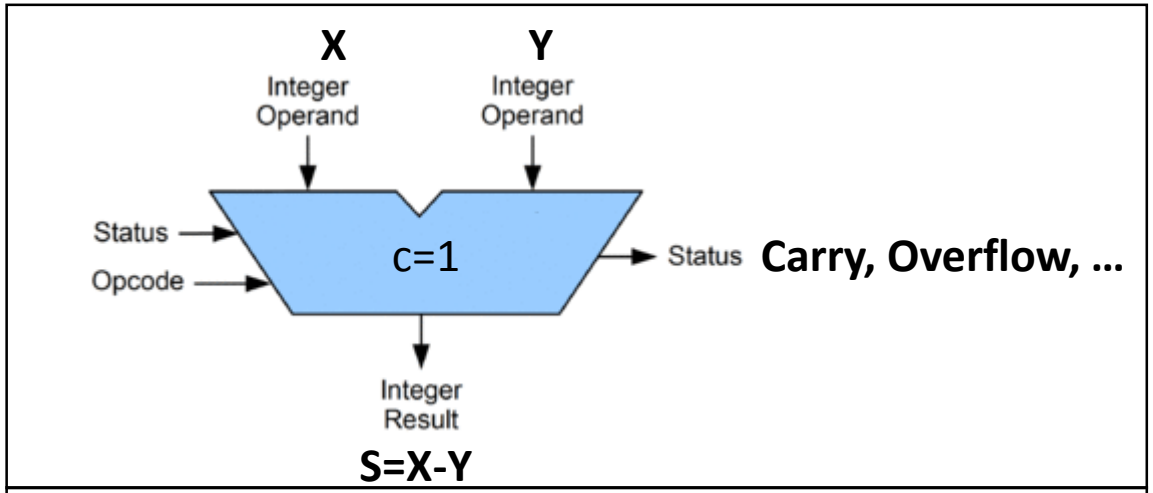
Additionneur sur 4-bits

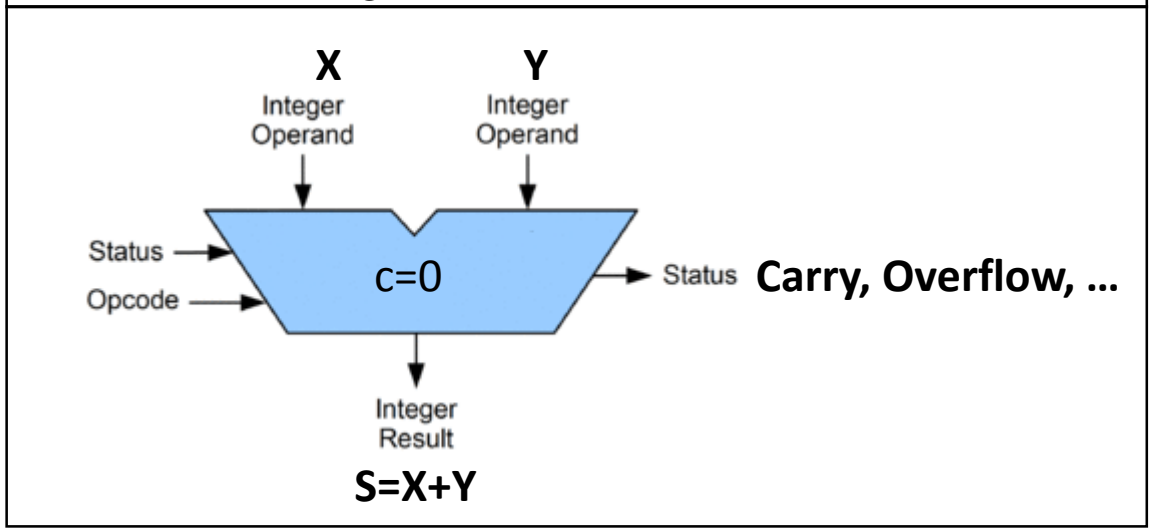
r=	1	0	0	0
X=		1	0	1	0
Y=		1	0	0	1
	1	0	0	1	1
	r3		S		

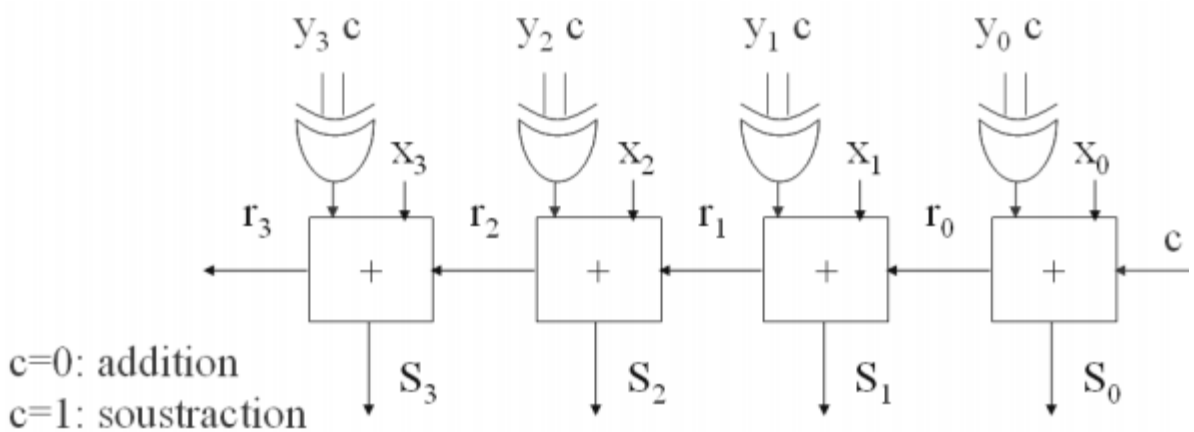
  - L'UAL prend en compte l'état du processeur (Status) avant l'opération et le met à jour après l'opération (retenue, débordement, etc.)

Structure interne d'un µProcesseur

- Unité Arithmétique et Logique (ALU)
  - La soustraction
    - L'UAL utilise le même matériel pour faire l'addition et la soustraction







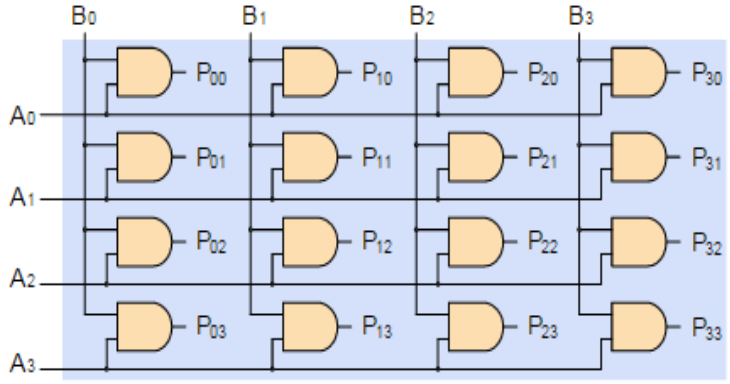
Un additionneur/soustracteur sur 4 bits

Structure interne d'un µProcesseur

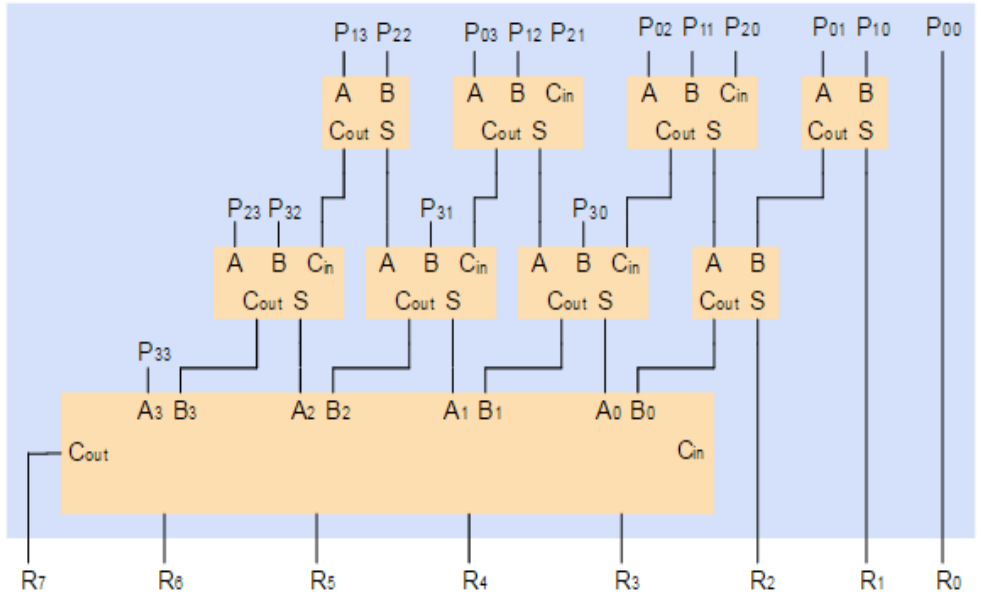
- Unité Arithmétique et Logique (ALU)

  - La multiplication est basée sur des multiplicateurs et des additionneurs

$$
\begin{array}{r}
A_3\ A_2\ A_1\ A_0 \\
\times\ B_3\ B_2\ B_1\ B_0 \\
\hline
P_{03}\ P_{02}\ P_{01}\ P_{00} \\
P_{13}\ P_{12}\ P_{11}\ P_{10} \\
P_{23}\ P_{22}\ P_{21}\ P_{20} \\
P_{33}\ P_{32}\ P_{31}\ P_{30} \\
\hline
R_3\ R_2\ R_1\ R_0\ R_3\ R_2\ R_1\ R_0
\end{array}
$$



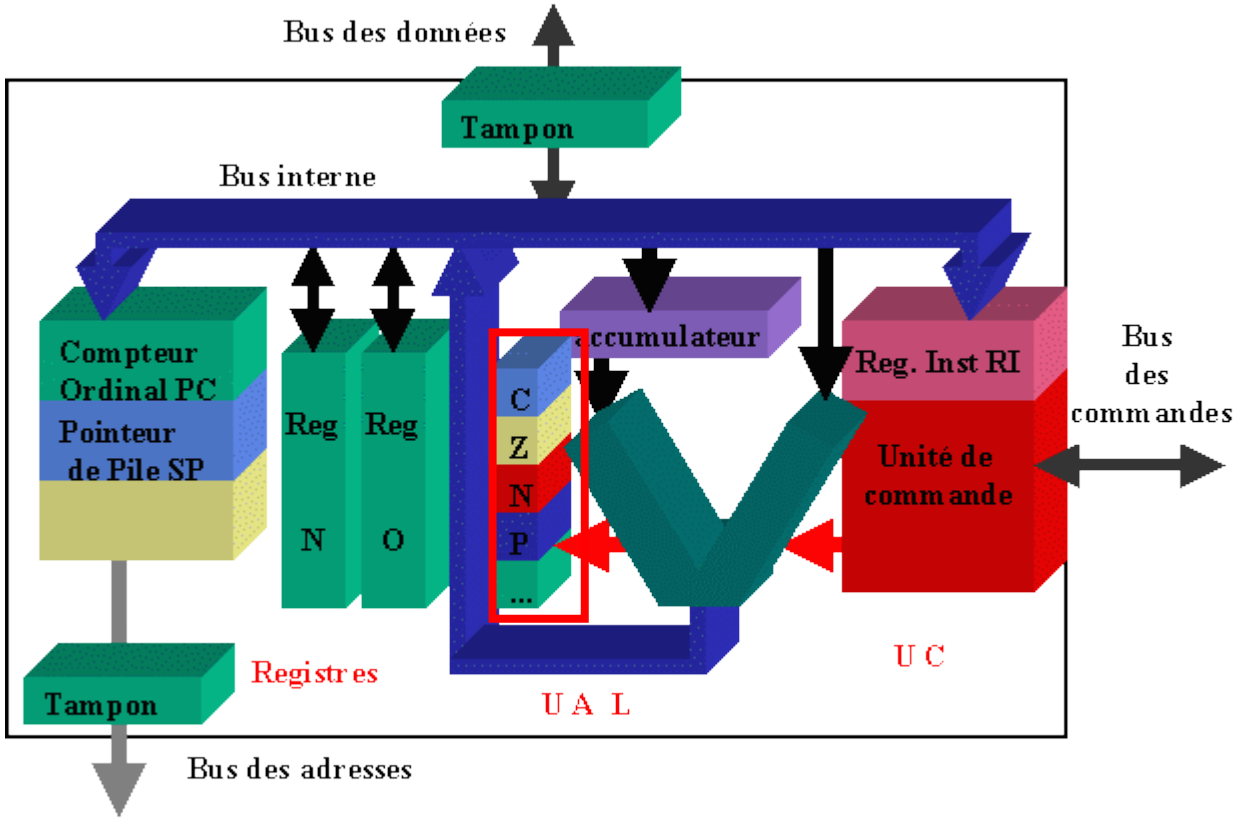
Multiplicateur 4-bit

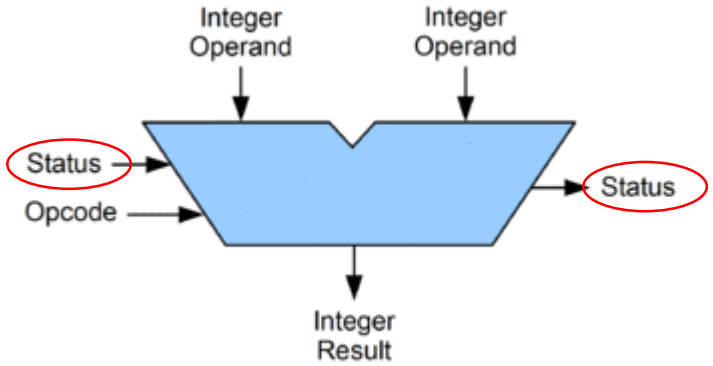


Additionneurs

Structure interne d'un µProcesseur

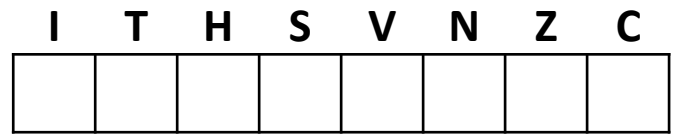
- Le registre d'état
  - C'est un ensemble de bits qui indique l'état du processeur.





Structure interne d'un µProcesseur

- Le registre d'état
  - C'est un ensemble de bits qui indique l'état du processeur.



  - Exemple: Le registre d'état d'un processeur AVR (Arduino) contient 8 bits:
    - C: Carry Flag. Indique si la dernière opération du processeur a généré une retenue (addition par exemple)
    - Z: Zero Flag. Indique si le résultat de la dernière opération du processeur est nul
    - N: Negative Flag. Ce bit est a 1 si le bit de signe est à 1
    - V: Two's complement overflow indicator. Indique si la dernière opération du processeur a généré un overflow
    - S: N $\oplus$ V. Indique le signe effectif du résultat de la dernière opération du processeur
    - H: Half Carry Flag. Indique si la dernière opération du processeur sur les 4 premiers bits a généré une retenue
    - T: Transfer. Utilisé pour des opérations spéciales du processeur.
    - I: Global Interrupt Enable/Disable Flag. Permet d'active/désactiver les interruptions

Remarque: les bits C, Z, N, V et S sont mis à jour automatiquement par le processeur après chaque instruction.

Structure interne d'un µProcesseur

- Le registre d'état
  - Effet de l'addition sur le registre d'état
    - Pour détecter si le résultat est correct, il faut regarder l'overflow (les deux dernières retenues)
    - V=0 → résultat correct
    - V=1 → résultat incorrect

Exemple: format sur 8 bits

```
  0 0 1 1 1 1 1
    0 0 1 0 0 1 0 1    (37)
+
    0 0 0 1 1 1 1 1    (31)
  -------------------
  0 1 0 0 0 1 0 0      (68)
```

I	T	H	S	V	N	Z	C
-	-	1	0	0	0	0	0

Le processeur donne le résultat: 68 (résultat correct)

```
  1 0 1 1 1 1 1 1
    1 0 1 0 0 1 0 1    (-91)
+
    1 0 0 1 1 1 1 1    (-97)
  -------------------
  1 0 1 0 0 0 1 0 0    (68)
```

I	T	H	S	V	N	Z	C
-	-	1	1	1	0	0	1

Le processeur donne le résultat: 68 (résultat incorrect)

Structure interne d'un µProcesseur

- Exercice

1) Donner le résultat binaire de l'addition sur 8 bits entre 11110100 et 11100001 (binaires signés). Est-ce que cette addition génère un overflow/dépassement ? Justifier votre réponse.

2) Donner les valeurs des bits suivants du registre d'état après l'exécution de l'addition :

   - Le bit C (Carry/retenue)
   - Le Bit Z (Zéro)
   - Le Bit V (Overflow)/dépassement
   - Le bit N (Négatif)

Structure interne d'un µProcesseur

- Solution

1) Donner le résultat binaire de l'addition sur 8 bits entre 11110100 et 11100001 (binaires signés). Est-ce que cette addition génère un overflow/dépassement ? Justifier votre réponse.

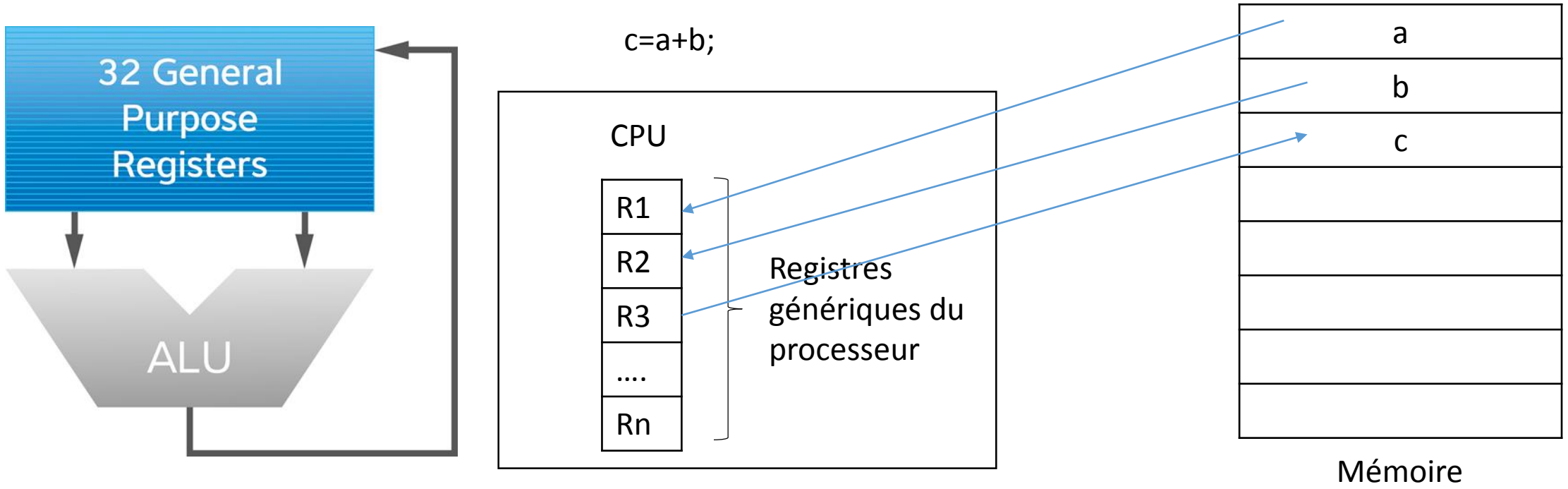
   - Le résultat binaire est 11010101, il n'y a pas eu de dépassement, les deux dernières retenues sont identiques.

1) Donner les valeurs des bits suivants du registre d'état après l'exécution de l'addition :

   - Le bit C (Carry/retenue) →1
   - Le Bit Z (Zéro) →0
   - Le Bit V (Overflow)/dépassement →0
   - Le bit N (Négatif) →1

Structure interne d'un µProcesseur

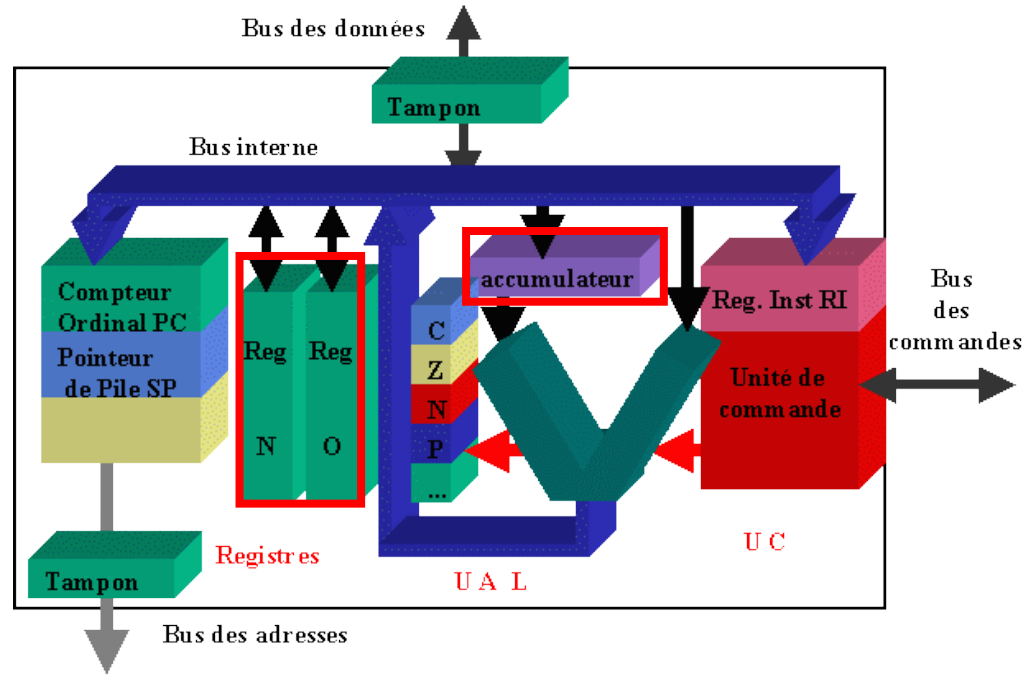
- Registres génériques

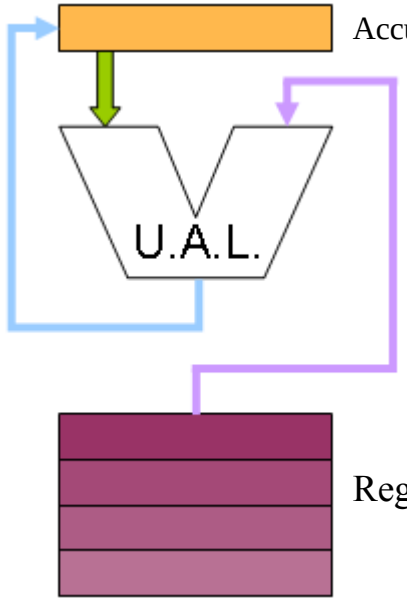


- Ils sont utilisés pour les calculs intermédiaires
- Quand on parle de processeur 32 bits/64 bits, on se réfère à la taille de ses registres génériques (c'est la taille des données que le processeur peut manipuler en une seule instruction machine)
  Exemple: un processeur 32-bits fait une opération sur des données de 64 bits en deux temps.

Structure interne d'un µProcesseur

- Accumulateur
  - Certains processeurs ont un accumulateur
  - L'UAL utilise l'accumulateur pour stocker le résultat d'une opération, qui sert d'opérande pour l'opération suivante
  - L'accumulateur est souvent un opérande implicite à ces opérations
  - Exemple: Pour exécuter l'instruction X=Y+Z en langage C, un processeur à accumulateur la traduirait en ces instructions machine:
    - `Load Y` (charger Y dans l'accumulateur)
    - `ADD Z` (ajouter Z au contenu de l'accumulateur)
    - `Store X` (stocker le contenu de l'accumulateur dans X)



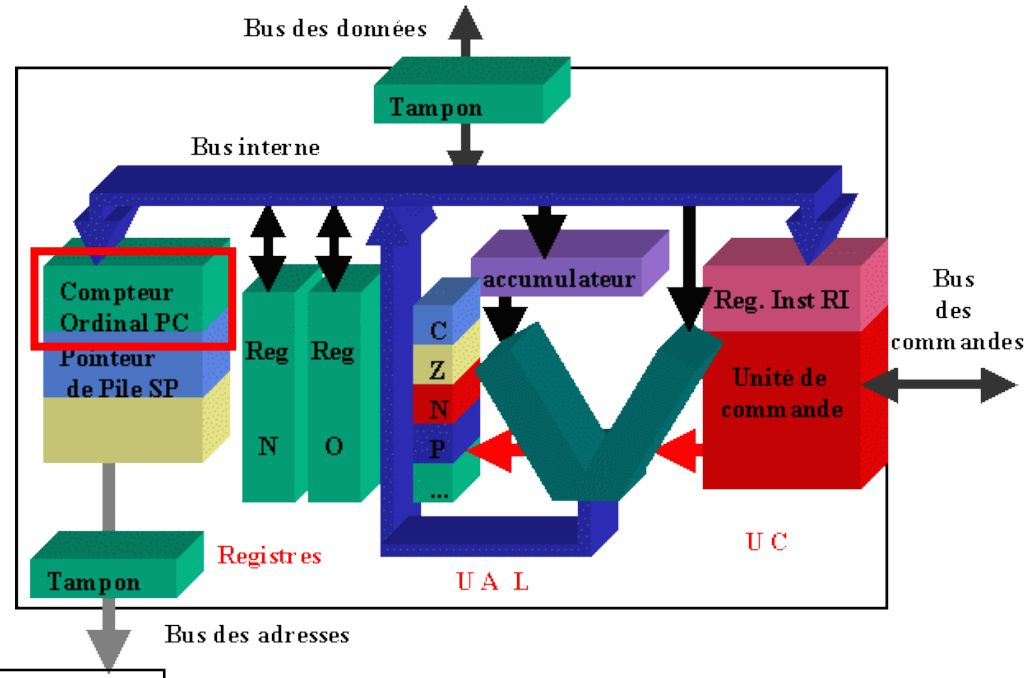


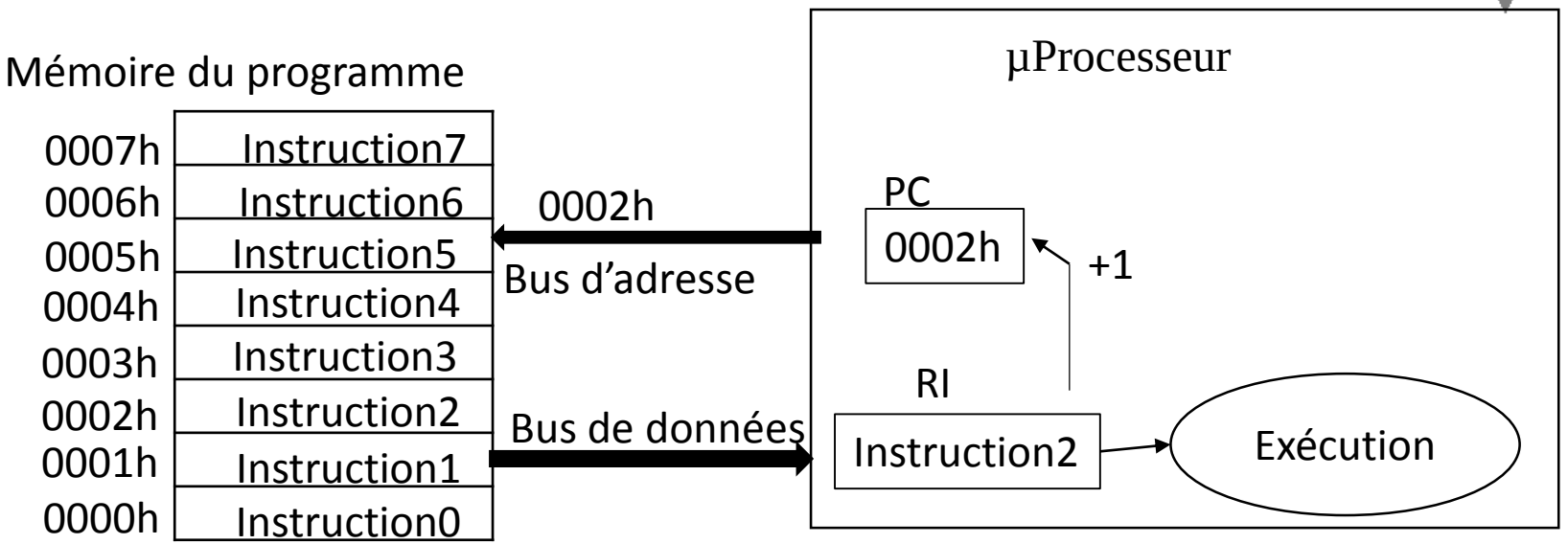
- L'accumulateur est un registre où les résultats intermédiaires de l'UAL sont versés
- Avantage: L'accès à ce registre par le processeur est très rapide, ce qui permet d'accélérer l'exécution des programmes

Registres génériques du processeur (non connectés directement à l'UAL)

Structure interne d'un µProcesseur

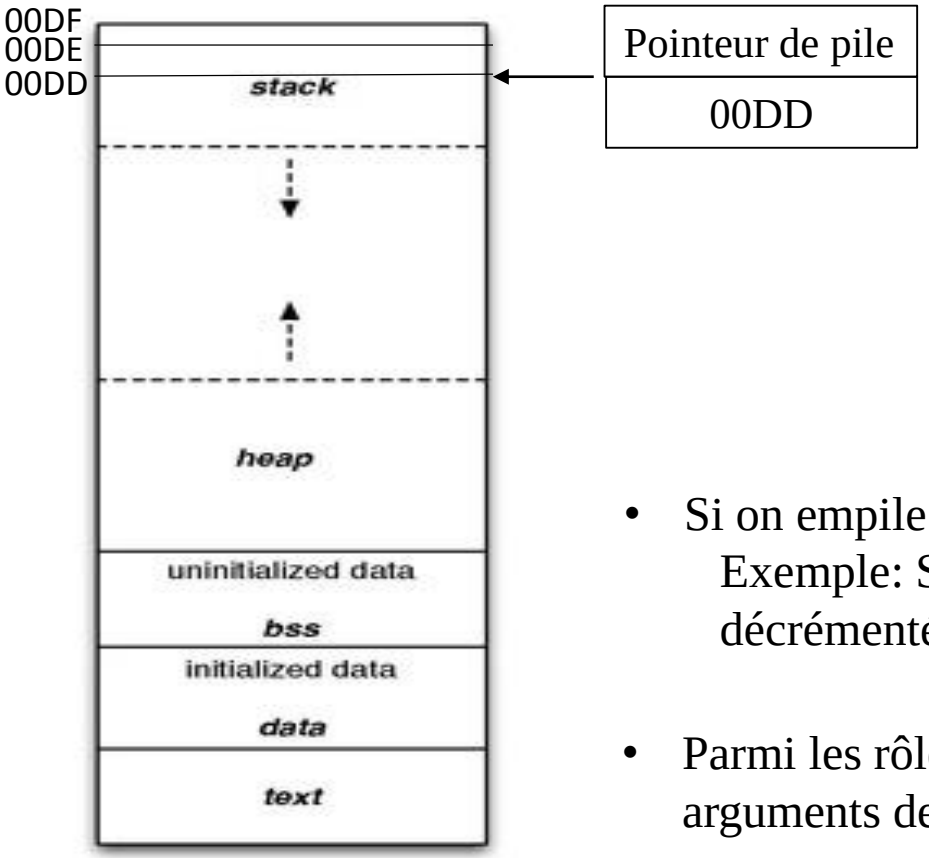
- Compteur de programme (PC)
  - Appelé aussi compteur ordinal
  - Il contient **l'<u>adresse</u> de la prochaine instruction** à exécuter
  - Son contenu est envoyé sur le bus d'adresse pour aller chercher la prochaine instruction
  - AUTOMATIQUEMENT incrémenté entre les instructions afin d'aller chercher l'instruction suivante de manière séquentielle.



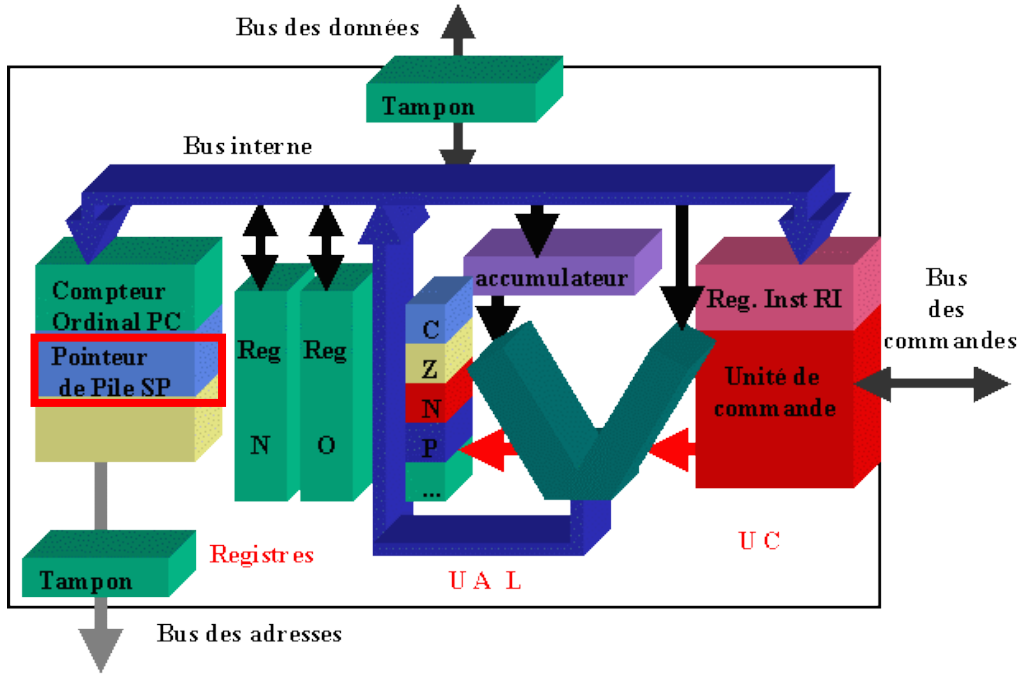


Structure interne d'un µProcesseur

- Pointeur de pile
  - Le pointeur de pile est un registre spécial qui contient l'adresse mémoire sur laquelle pointe le haut de la pile



Mémoire (RAM)



- Si on empile des données dans la pile, le pointeur sera décrémenté
  Exemple: Si on ajoute une données de 2 octets à la pile, le pointeur est décrémenté de 2

- Parmi les rôles de la pile: stocker les adresses de retour des fonctions, stocker les arguments des fonctions

- Pointeur de pile
  - Exemple d'utilisation pour l'appel d'une fonction

```
int main(){
….
a=a+5;
F(b);
a=a+b;
}

void F(int b){
printf(« b=%d\n »,b);
}
```

- À t=tn
  PC= 0350h
  1) On exécute l'instruction a=a+5
  2) PC← PC+1

PC

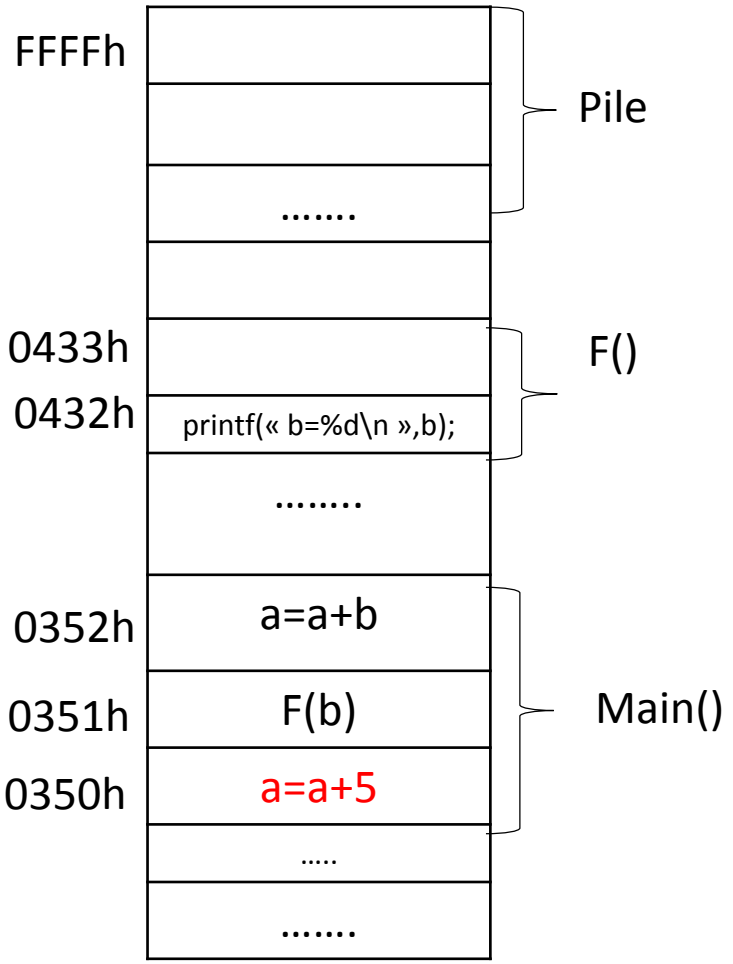
0350h

0351h

SP

Stack Pointer: Pointeur de pile

FFFFh



Structure interne d'un µProcesseur

- Pointeur de pile
  - Exemple d'utilisation pour l'appel d'une fonction

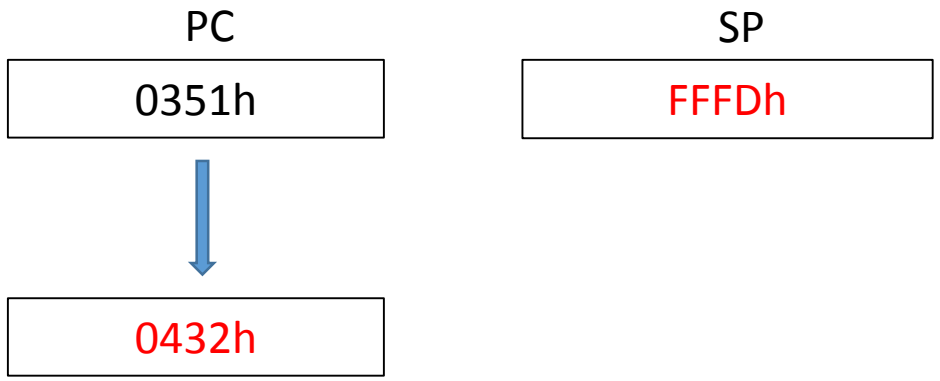
```
int main(){
….
a=a+5;
F(b);
a=a+b;
}

void F(int b){
printf(« b=%d\n »,b);
}
```

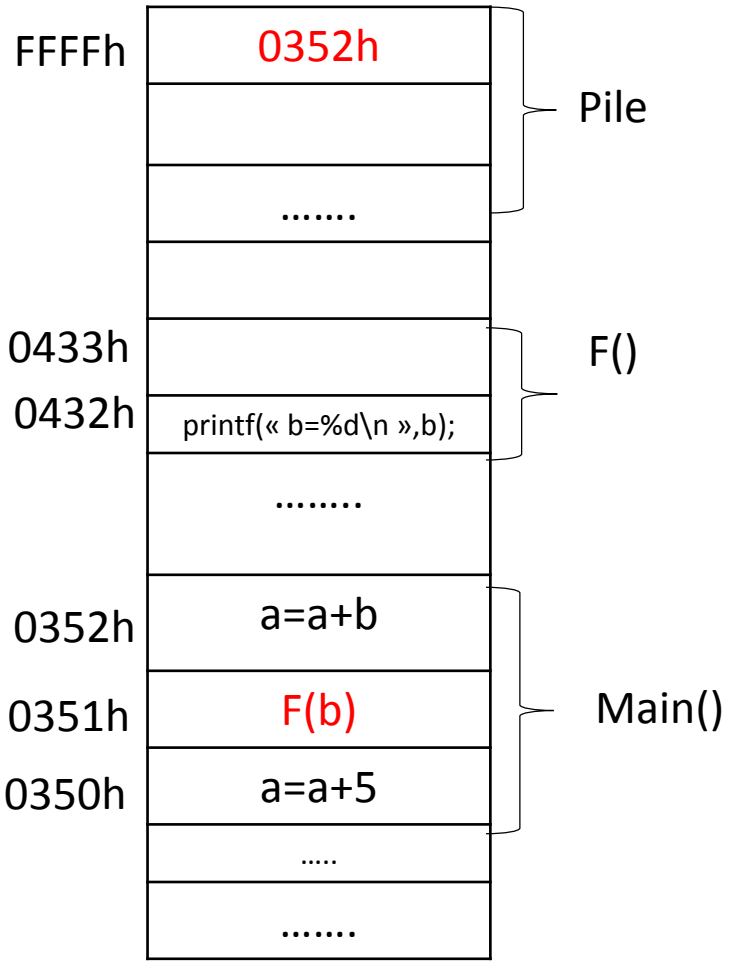
- À t=tn+1

  PC=0351 (appel d'une fonction)

  1) sauvegarde de l'adresse 0352h dans la pile

  2)PC← 0432h (adresse de début de la fonction)

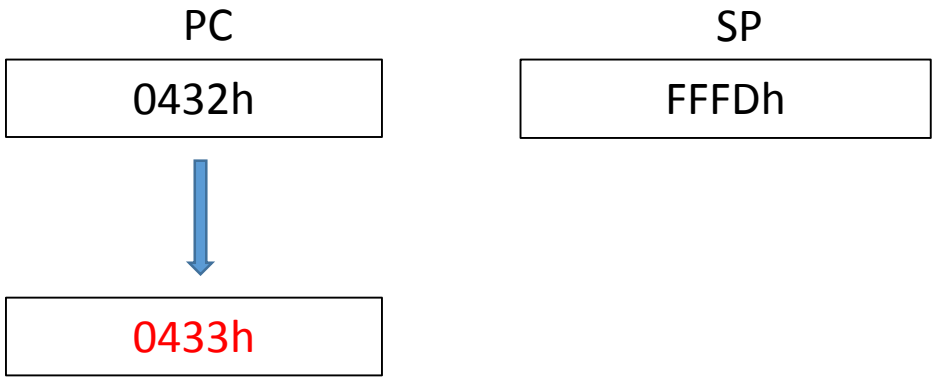


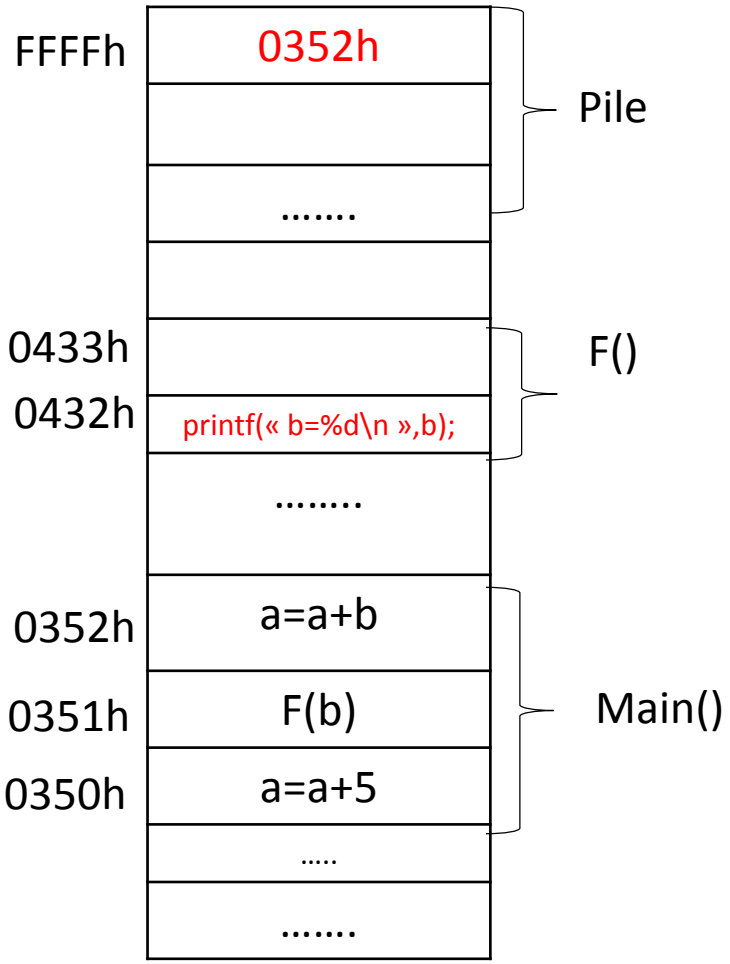


- Pointeur de pile
  - Exemple d'utilisation pour l'appel d'une fonction

```
int main(){
….
a=a+5;
F(b);
a=a+b;
}

void f(int b){
printf(« b=%d\n »,b);
}
```

- À t=tn+2
  PC=0432
  1) Exécution de la fonction
  2) PC← 0433h



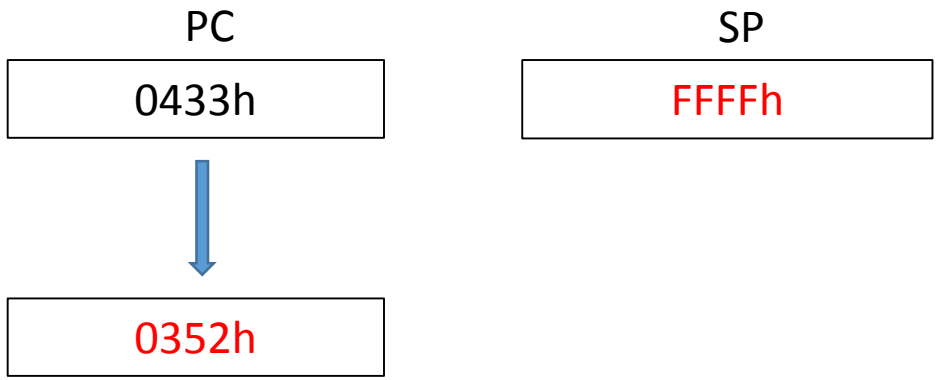


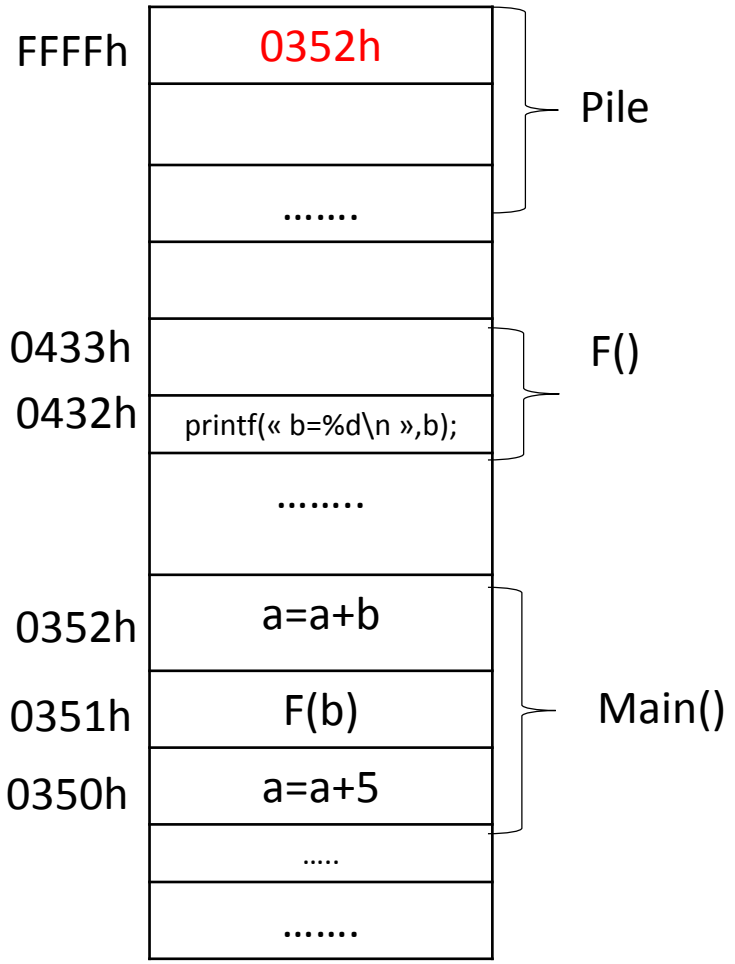
Structure interne d'un µProcesseur

- Pointeur de pile
  - Exemple d'utilisation pour l'appel d'une fonction

```
int main(){
….
a=a+5;
F(b);
a=a+b;
}

void f(int b){
printf(« b=%d\n »,b);
}
```

- À t=tn+3
  PC=0433
  1) Retour de la fonction
  2) PC← 0352 (récupérée à partir de la pile)

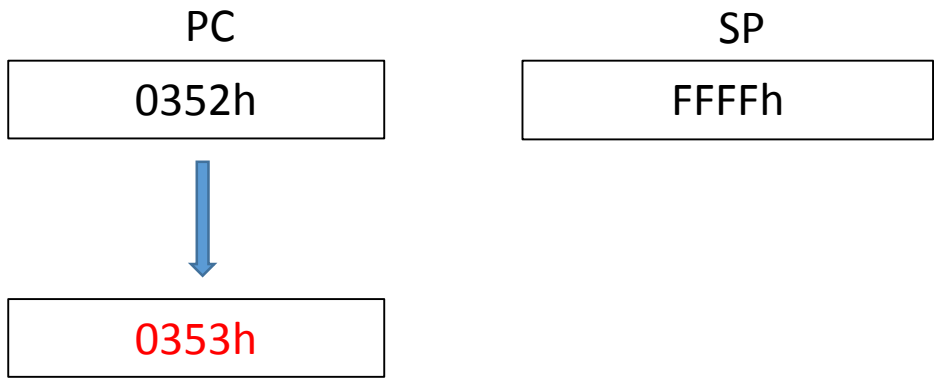


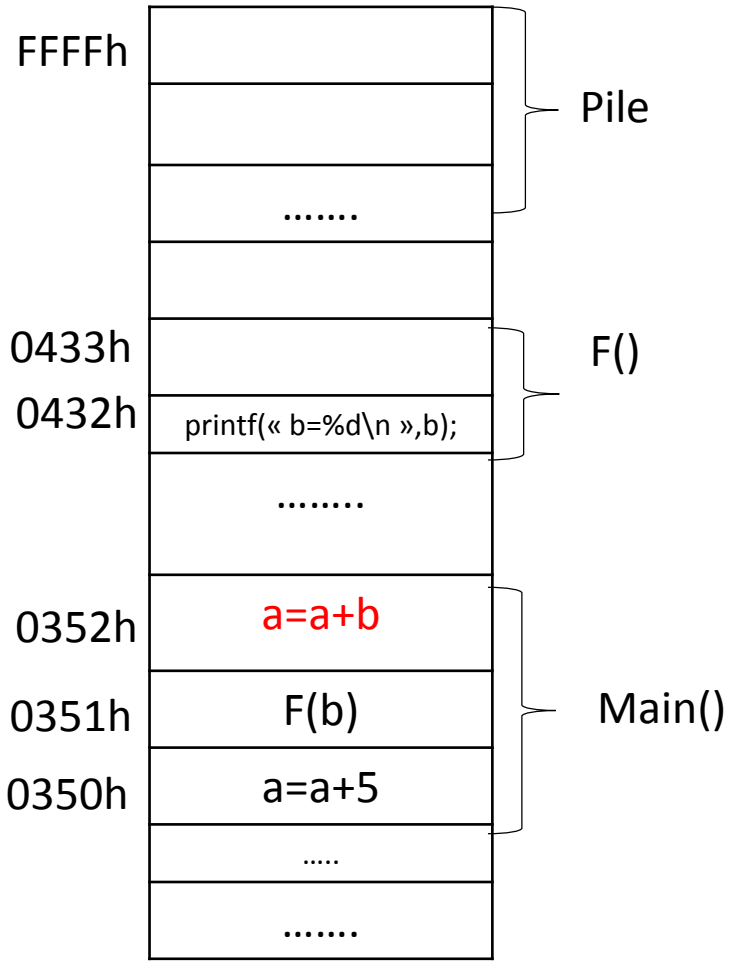


- Pointeur de pile
  - Exemple d'utilisation pour l'appel d'une fonction

```
int main(){
….
a=a+5;
F(b);
a=a+b;
}

void f(int b){
printf(« b=%d\n »,b);
}
```

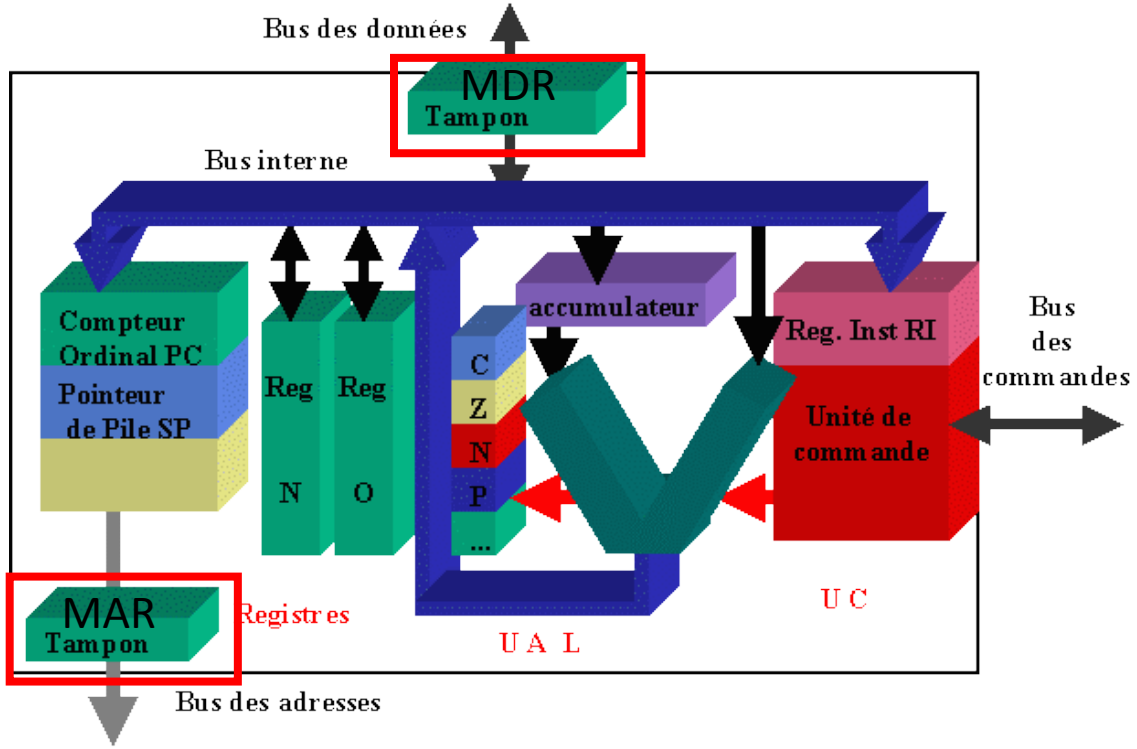
- À t=tn+4
  PC=0352
  1) On exécute a=a+b
  2) PC← PC+1





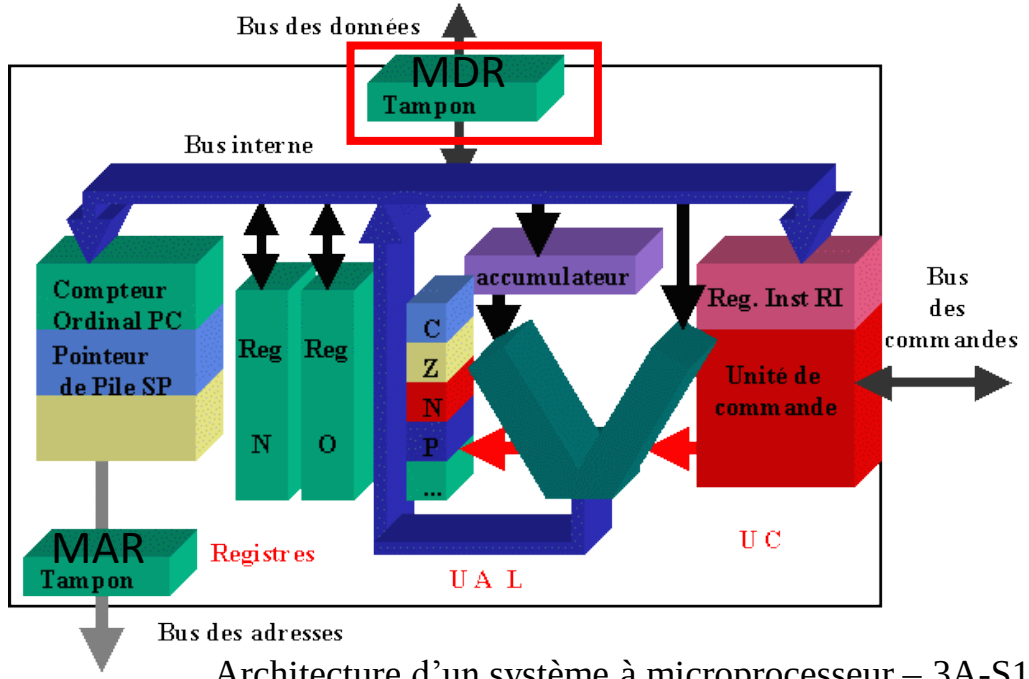
Structure interne d'un µProcesseur

- Les tampons/ buffers
  - Le CPU contient deux tampons/buffers pour stocker les adresses et les données échangées avec le bus
  - Cela permet au processeur et à la mémoire/ E/S de travailler indépendamment.
  - Exemple: le processeur récupère les données quand il est prêt et quand les données sont disponibles dans le buffer.



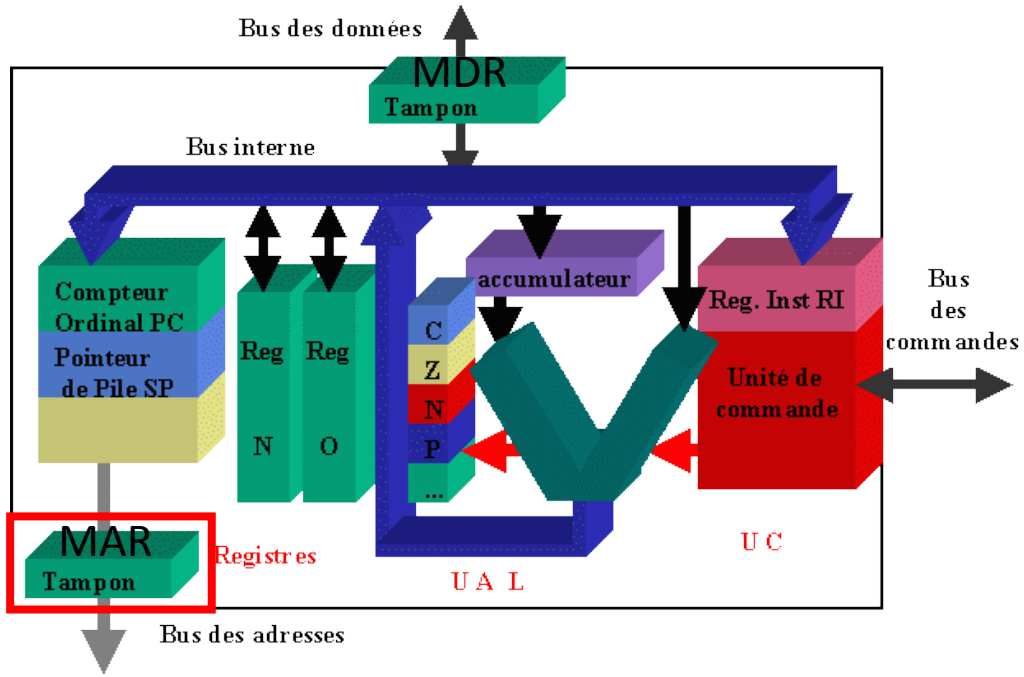
Structure interne d'un µProcesseur

- Les tampons/ buffers
  - Le MDR: Memory Data Register
    - Ce registre stocke les données que le processeur veut échanger avec le bus (avec une mémoire ou une E/S)
    - Utilisée dans deux sens
      - Après une demande de lecture envoyée à la mémoire principale, la donnée concernée est stockée dans le MDR avant d'être utilisée par le processeur (le processeur peut être en train de faire un autre traitement, le bus peut donc être utilisé par d'autres composants)
      - Si le processeur veut écrire une donnée dans la mémoire, il l'écrit d'abord dans le MDR et la donnée sera envoyée à la mémoire quand le bus sera libre



Structure interne d'un µProcesseur

- Les tampons/ buffers
  - Le MAR: Memory Address Register
    - Ce registre stocke l'adresse de la case mémoire à laquelle le processeur veut accéder
    - Utilisé dans un seul sens (processeur vers mémoire)
    - Lorsque le processeur veut accéder à une adresse mémoire, il écrit cette adresse dans le MAR. L'adresse sera envoyée sur le bus quand il sera libre

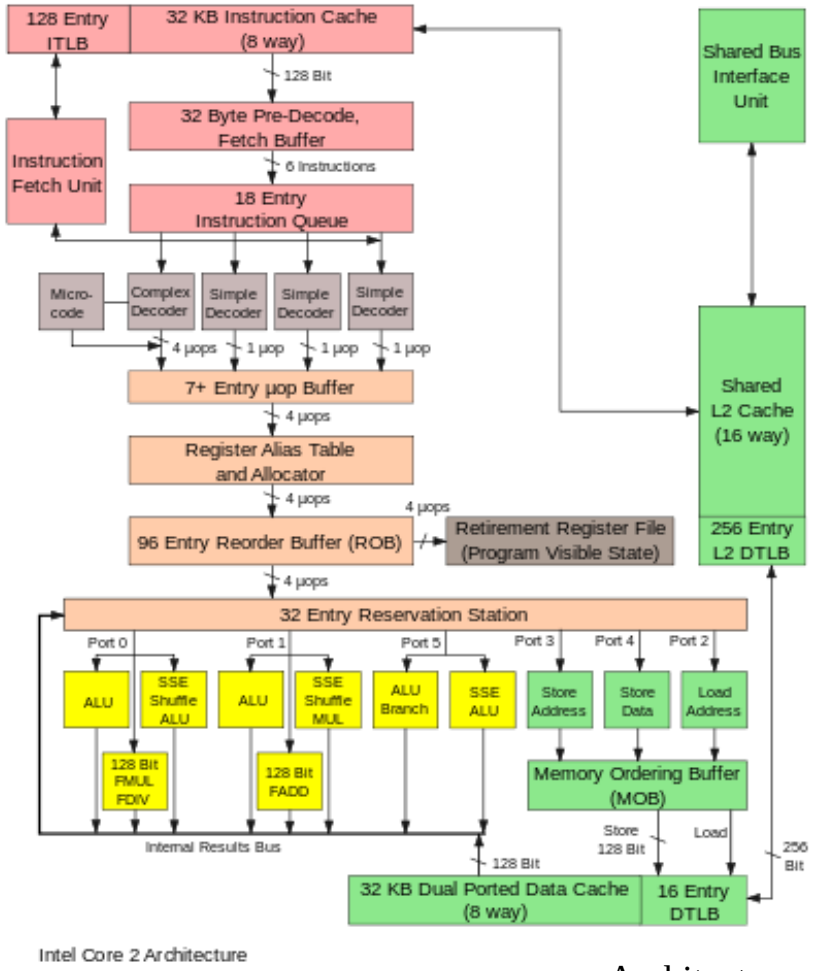


Les unités de calculs

- Dans les processeurs modernes, chaque processeur/cœur a plusieurs unités de calcul

→Possibilités de traiter plusieurs instructions en même temps selon la nature de l'instruction

→ Accélération du temps d'exécution



Intel Core 2 Architecture

Exemple d'architecture d'un intel dual core

Unités de calcul :

- Plusieurs ALU (calcul sur les entiers et en virgule fixe)
- Plusieurs FPU (Floating Point Units: calcul sur les flottants) (Exemple: FMUL, FDIV dans la figure)
- etc.

Les modules qui interviennent dans le cycle d'exécution d'une instruction

- Séquencement du cycle FDX
  - Fetch: recherche de l'instruction
    - Le processeur met dans le buffer MAR, le contenu du compteur ordinal (l'adresse de la prochaine instruction) et envoie une commande de lecture de mémoire
    - La mémoire répond en envoyant l'instruction qui se trouve à cette adresse
    - L'instruction est reçue dans le buffer MDR puis transférée au registre RI
  - Decode: décodage de l'instruction
    - L'unité de contrôle décode l'instruction qui se trouve dans le RI
    - Pour les instructions arithmétiques et logiques, les signaux de contrôle nécessaires à l'exécution sont envoyée par l'unité de contrôle à l'UAL
    - S'il s'agit d'une instruction mémoire (charger ou stocker une donnée), l'adresse correspondante est écrite dans le buffer MAR
  - Execute: exécution de l'instruction
    - Pour les instructions logiques et arithmétiques, l'UAL exécute la commande envoyée par l'unité de contrôle
    - Pour les instructions de chargement et de stockage: la donnée est chargée de la mémoire vers un registre ou inversement

Le cycle d'exécution d'une instruction

- Remarques
  - C'est l'unité de contrôle du processeur qui gère le séquencement de l'exécution des instructions
  - On a présenté un cycle d'exécution en trois étapes mais le nombre d'étapes dépond du processeur
  - On peut trouver, par exemple, des cycles à 5 étapes: Fetch, decode, execute, Memory access, Write-back
    - L'étape Memory Access correspond aux accès mémoire pour les instructions de stockage mémoire
    - L'étape Write-back correspond à l'écriture du résultat de l'exécution dans un registre