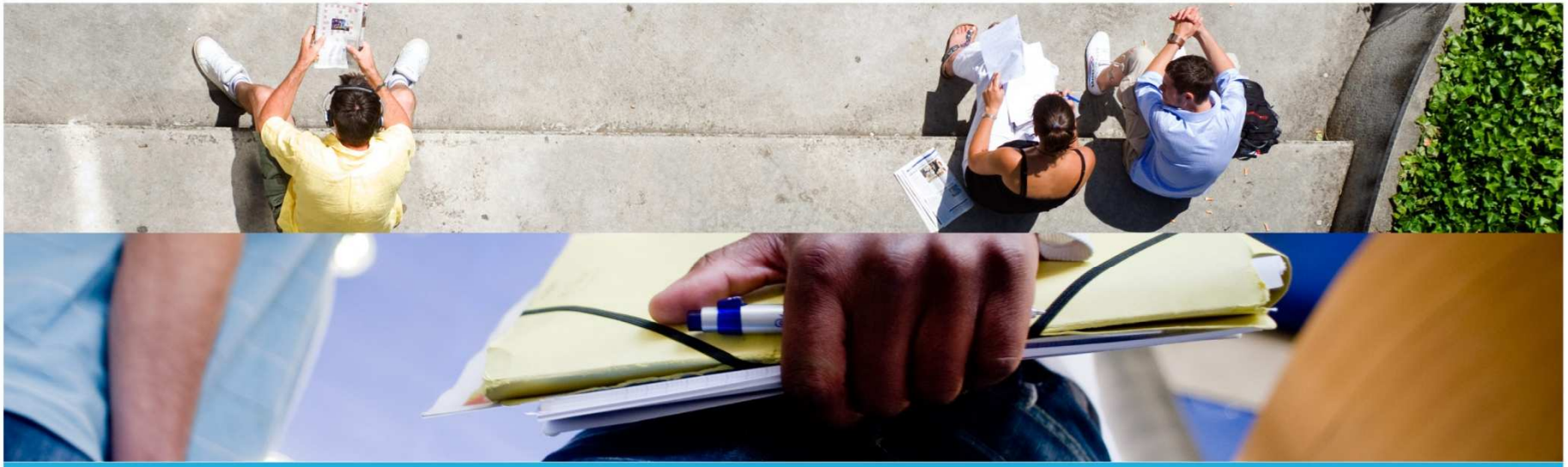


ARCHITECTURE DES SYSTEMES A MICROPROCESSEUR

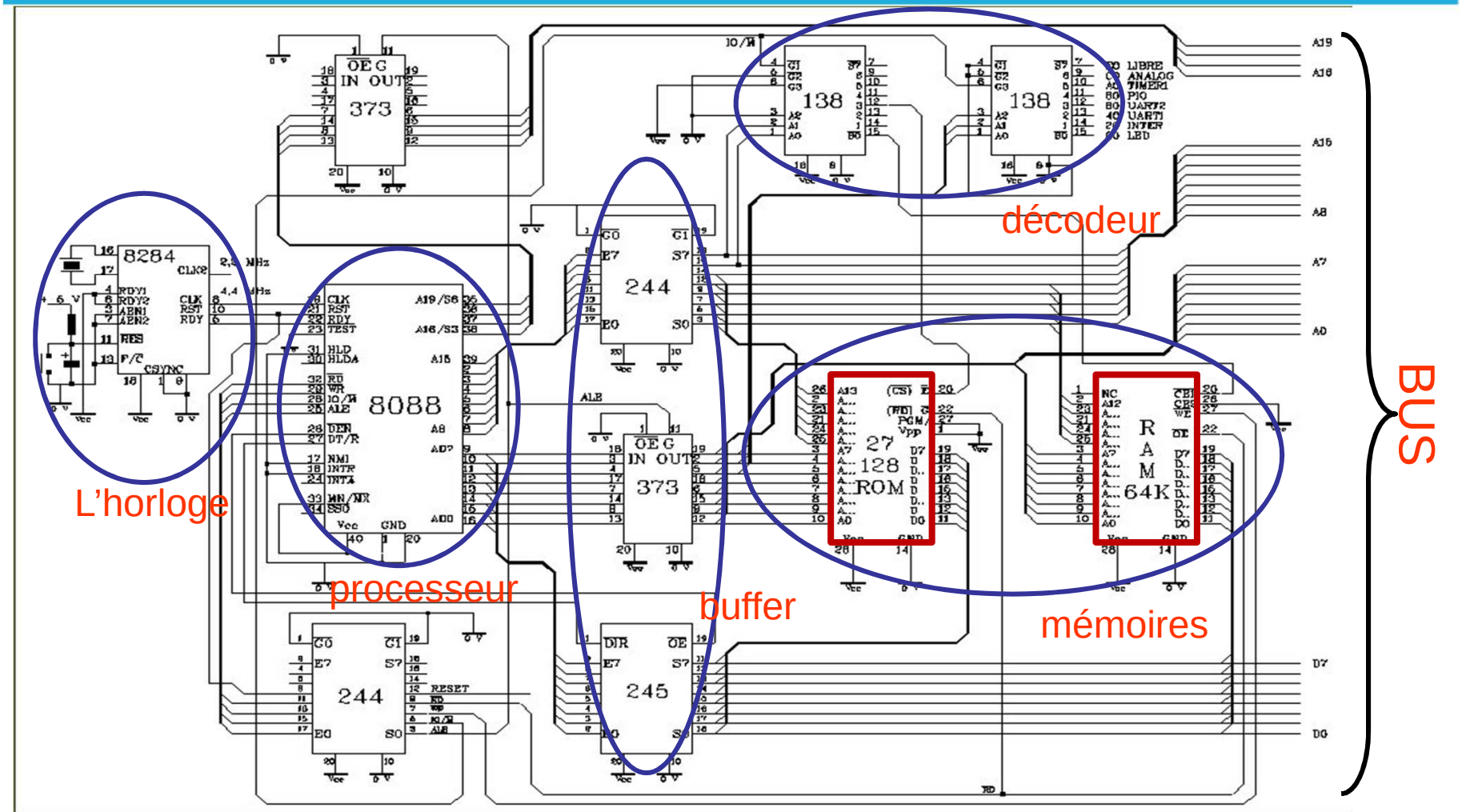
Architecture et structure interne d’un ordinateur

N.ABOUCHI

membre de UNIVERSITÉ DE LYON

CPE LYON
ÉCOLE SUPÉRIEURE DE CHIMIE PHYSIQUE ÉLECTRONIQUE DE LYON

Exemple d'un système à microprocesseur (premier PC)



Les rôles

L'unité centrale, composée par le microprocesseur, est chargée d'interpréter et d'exécuter les instructions d'un programme, de lire ou de sauvegarder les résultats dans la mémoire et de communiquer avec les unités d'échanges.

La mémoire principale contient les instructions du ou des programmes en cours d'exécution et les données associées à ce ou ces programmes.

Les interfaces d'entrées-sorties permettent d'assurer la communication entre le microprocesseur et les périphériques (clavier, écran, imprimante, réseau, etc.).

Le bus est un ensemble de fils qui assure la transmission du même type d'information (**bus de données, bus d'adresses, bus de commande**).

On **caractérise** le microprocesseur par la fréquence de **l'horloge** qui permet de cadencé ses activités, le nombre d'instructions qu'il est capable d'exécuter par secondes (MIPS, MFLOPS), la taille des données qu'il est capable de manipuler (bits) et l'espace mémoire qu'il est capable d'adresser.

Microprocesseur : définition

Un microprocesseur est un circuit intégré complexe qui intègre sur une même puce des fonctions logiques combinatoires (multiplexeurs, codeurs, décodeurs, comparateurs, etc.), des fonctions logiques séquentielles (registres, compteur, mémoires, etc.) et des éléments de calculs arithmétiques et logiques (additionneurs, soustracteurs, multiplieurs, diviseurs, opérateurs logiques, etc.).

Que fait le microprocesseur ?

Le microprocesseur est capable de :

1. **lire les instructions** d'un programme,
2. **les interpréter**,
3. **lire les données**,
4. **exécuter les instructions,**
5. **sauvegarder les résultats** dans la mémoire ou les communiquer aux périphériques d'entrées sorties.

Eléments fonctionnels de bases

Eléments combinatoires :

- multiplexeurs,
- codeurs et décodeurs,
- comparateurs

Plus l'horloge

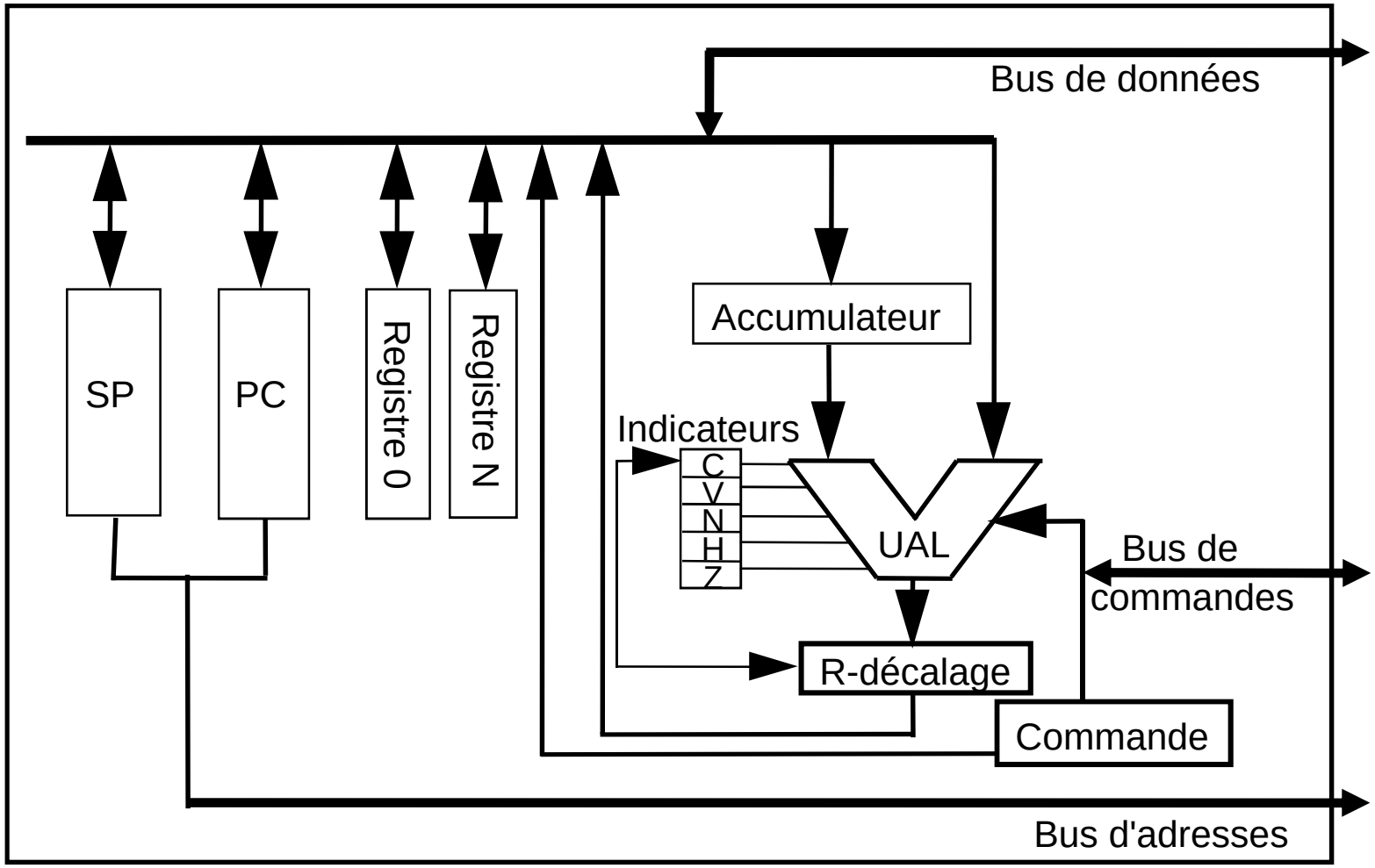
Eléments séquentiels :

- registres (bascules, registres à décalage, registres universels),
- compteurs (synchrone, asynchrones, binaires, BCD).

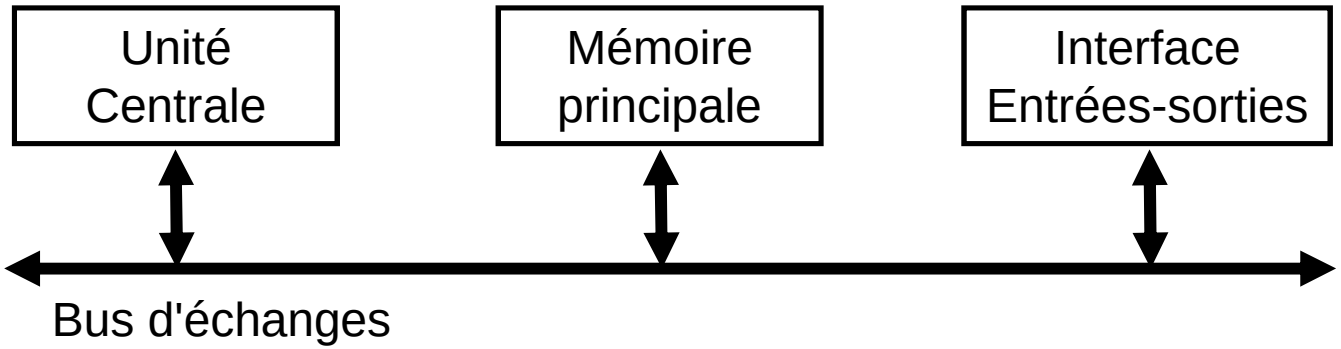
Eléments arithmétiques et logiques :

- additionneurs et soustracteurs,
- multiplieurs et diviseurs,
- opérateurs arithmétiques et logiques,
- etc.

Organisation fonctionnels de bases



Principe du modèle Von Neumann



Caractéristiques

- Jeu d'instructions propre (défini lors de la construction de la machine).
- Programme = suite de traitement, disponibles sur la machine considérée.
- Programme et données peuvent être stockés dans la mémoire du système.
- Certaines instructions autorisent des ruptures de séquence conditionnelles.

Jeu d'instructions

Le jeu d'instructions décrit toute les opérations élémentaires que le microprocesseur est capable d'exécuter. Différents types d'instructions sont possibles et différents types d'accès aux opérandes sont aussi disponibles :

- ❑ Transfert de données
  - o charger ou sauver en mémoire,
  - o transferts de registre à registre, etc.
- ❑ Opérations arithmétiques
  - o addition, soustraction,
  - o division, multiplication, etc.
- ❑ Opérations logiques
  - o ET, OU, NON, NAND,
  - o comparaison, test, etc.
- ❑ Contrôle de séquence
  - o Branchement,
  - o test, etc.

- ❑ Codage des instructions
  - o un nombre entier d'octets,
  - o un code instruction,
  - o un champ opérande
  - o donnée, ou une référence à une donnée.

- ❑ Modes d'adressages
  - o Adressage de registre
  - o Adressage immédiat
  - o Adressage direct
  - o Adressage indirect

une instruction peut être codée par 1 ou plusieurs octets.

Architecture CISC

Complex Instruction Set Computer. Ordinateur à jeu d'instruction complexe, (80X86, Pentium, etc.)

Architecture RISC

Reduced Instruction Set Computer. Ordinateur à jeu d'instruction réduit, plus rapide que les CISC car on peut mieux optimiser leur câblage et parce qu'ils ont des vitesse d'horloge plus importante, leur petite taille et leur faible nombre de transistors les faisant moins chauffer. (PowerPC, RS-6000, etc.)

Architecture CISC

Pourquoi : à l'origine pour compenser la lenteur des mémoires par rapport aux microprocesseurs. Plutôt que de coder une opération complexe par plusieurs instructions simple, utiliser une seule instruction complexe pour réaliser l'opération. **Le but étant de diminuer le nombre d'accès mémoires**.

Comment : pour une tâche donnée, une machine CISC exécute un petit nombre d'instructions complexes nécessitants chacune un plus grand nombre de cycles d'horloge. Le code machine de ces instructions varie d'une instruction à l'autre et nécessite donc un décodeur complexe (micro-code)

Architecture RISC

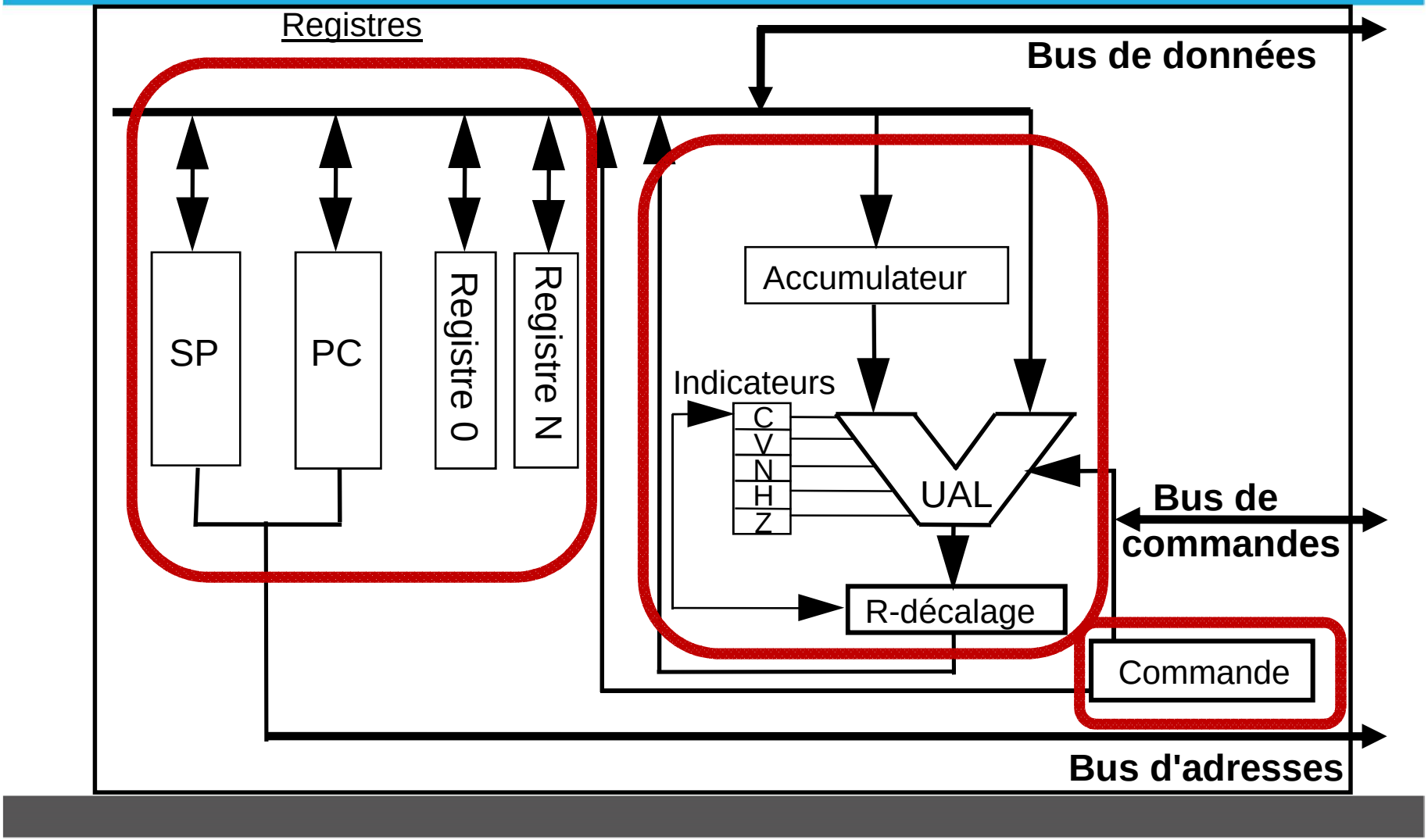
Pourquoi : 80% des traitements des langages de haut niveau utilisent seulement 20% des instructions du microprocesseur. D'où l'idée de réduire le jeu d'instructions à celles le plus couramment utilisées et d'en améliorer la vitesse de traitement.

Comment : instructions en nombre réduit réalisées à partir de séquenceur câblé et exécutée en un cycle d'horloge. Les accès à la mémoire s'effectue seulement à partir de deux instructions (Load et Store). Les instructions complexes doivent être réalisées à partir de d'instructions élémentaires, ce qui nécessite des compilateurs très évolués.

Petite comparaison

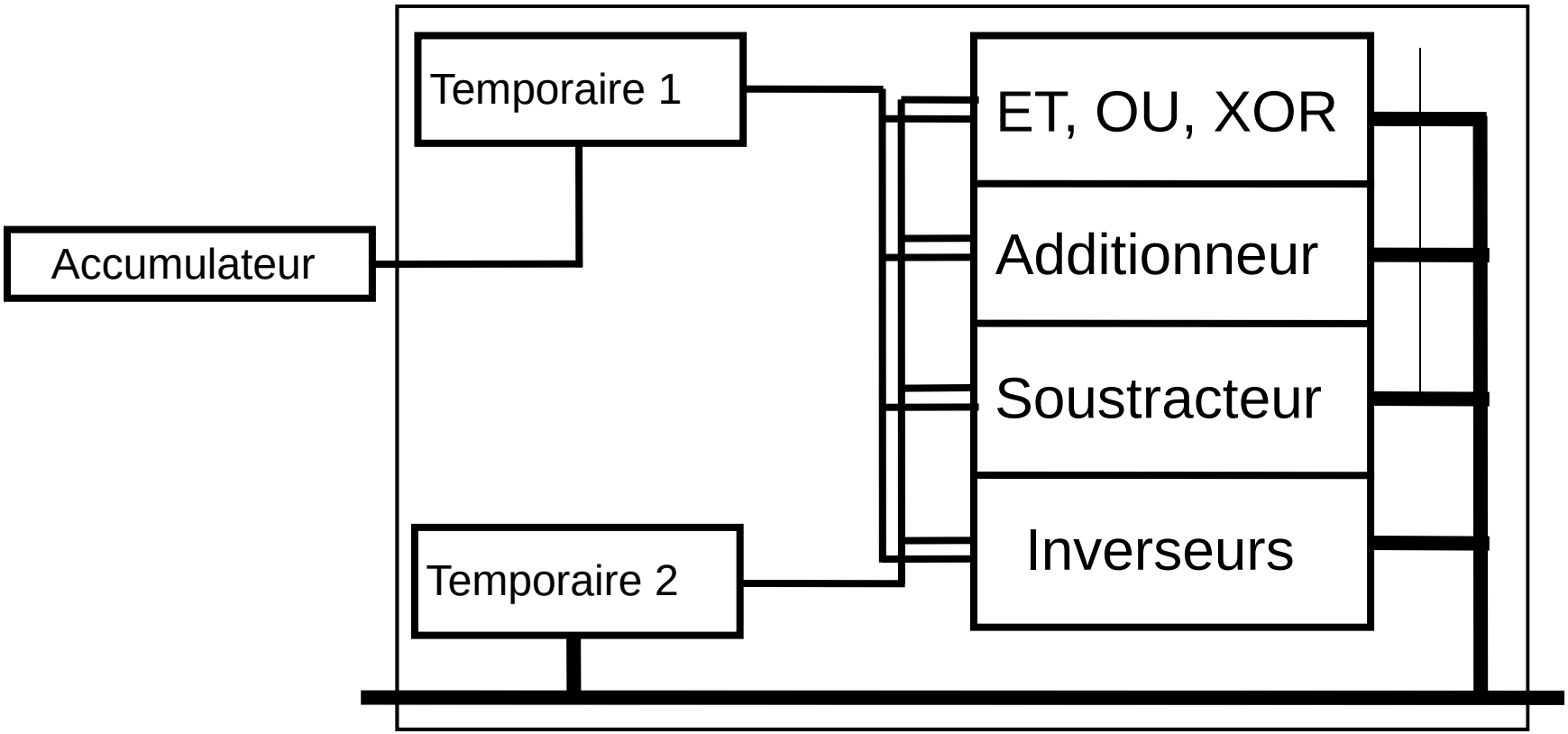
Architecture RISC	Architecture CISC
Instructions simples sur un seul cycle	Instructions sur plusieurs cycles
Format fixe	Instructions au format variable
Décodeur simple (câblé)	Décodeur complexe (microcode)
Beaucoup de registres	Peu de registres
Accès mémoire par LOAD et STORE	Accès mémoire par toutes instructions
Peu de modes d’adressage	Beaucoup de modes d’adressage
Compilateur complexe	Compilateur simple

Architecture d’un microprocesseur



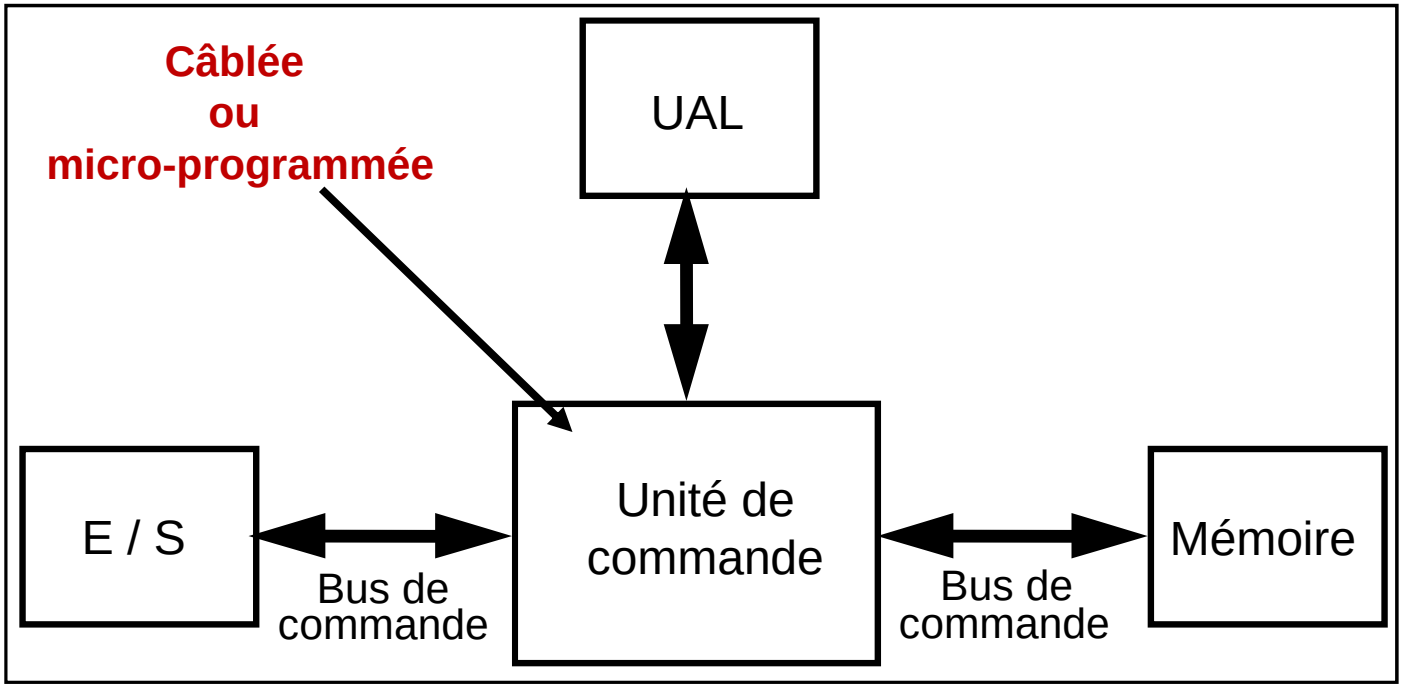
L’unité arithmétique et logique (UAL)

Effectue les opérations arithmétiques et logiques portant sur deux nombres au maximum ainsi que les opérations de décalage.



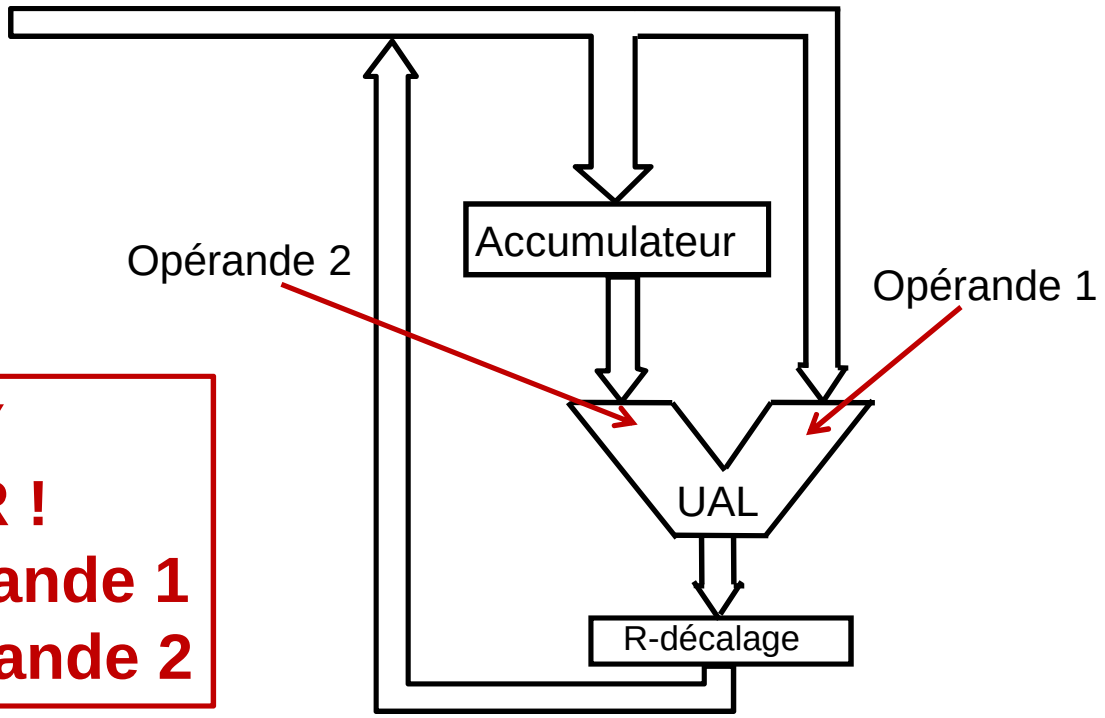
L’unité de commande

✓Assure le bon ordre du déroulement des séquences du programme,
✓élabore les signaux de synchronisation,
✓gère les ordres échangés par l’UAL, les entrées, les sorties, et la mémoire,
✓assure la recherche en mémoire, le décodage et l’exécution des instructions.



L’accumulateur

- ✓ Registre particulier placé à l’une des entrées de l’UAL.
- ✓ La plupart des instructions arithmétiques et logiques utilisent l’accumulateur.
- ✓ Il peut être référencé en entrée et en sortie.



X = X + Y
A = A + R !
X = opérande 1
U = opérande 2

Les indicateurs d'états : exemples

- Caractérisent les états de fonctionnement du microprocesseur.
- Interviennent en particulier pour les branchement conditionnels.

C	N	H	Z	P	I	-	-

C : retenue arithmétique

N ou S : signe

H ou AC : retenue intermédiaire

Z : zéro

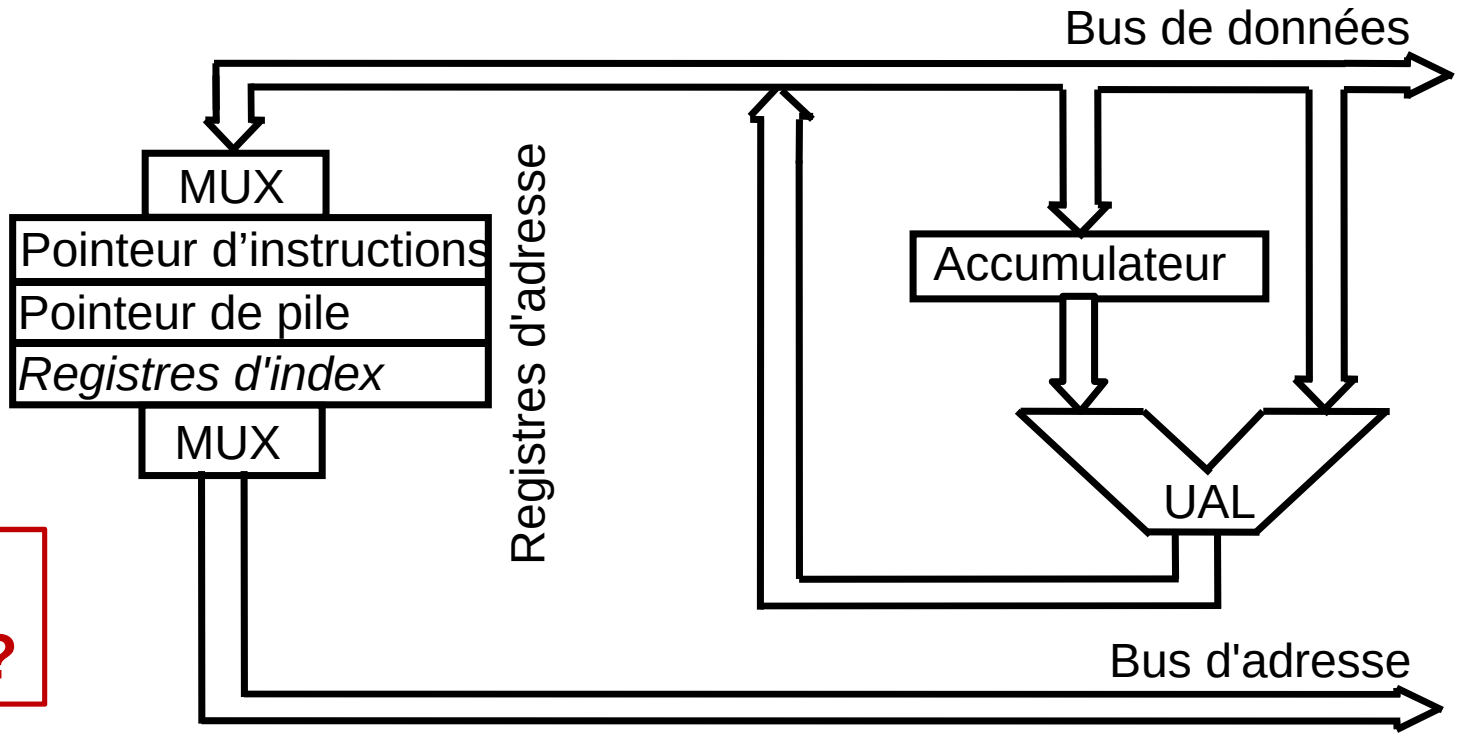
P : Parité

I : interruption

Remarque : La plupart des instructions exécutées par le processeur modifieront l'ensemble ou une partie des indicateurs d'état. Aussi, il est toujours important de se reporter au tableau fourni par le fabricant sur lequel figurent les bits d'état qui seront modifiés par les instructions.

Les registres d ’adresses

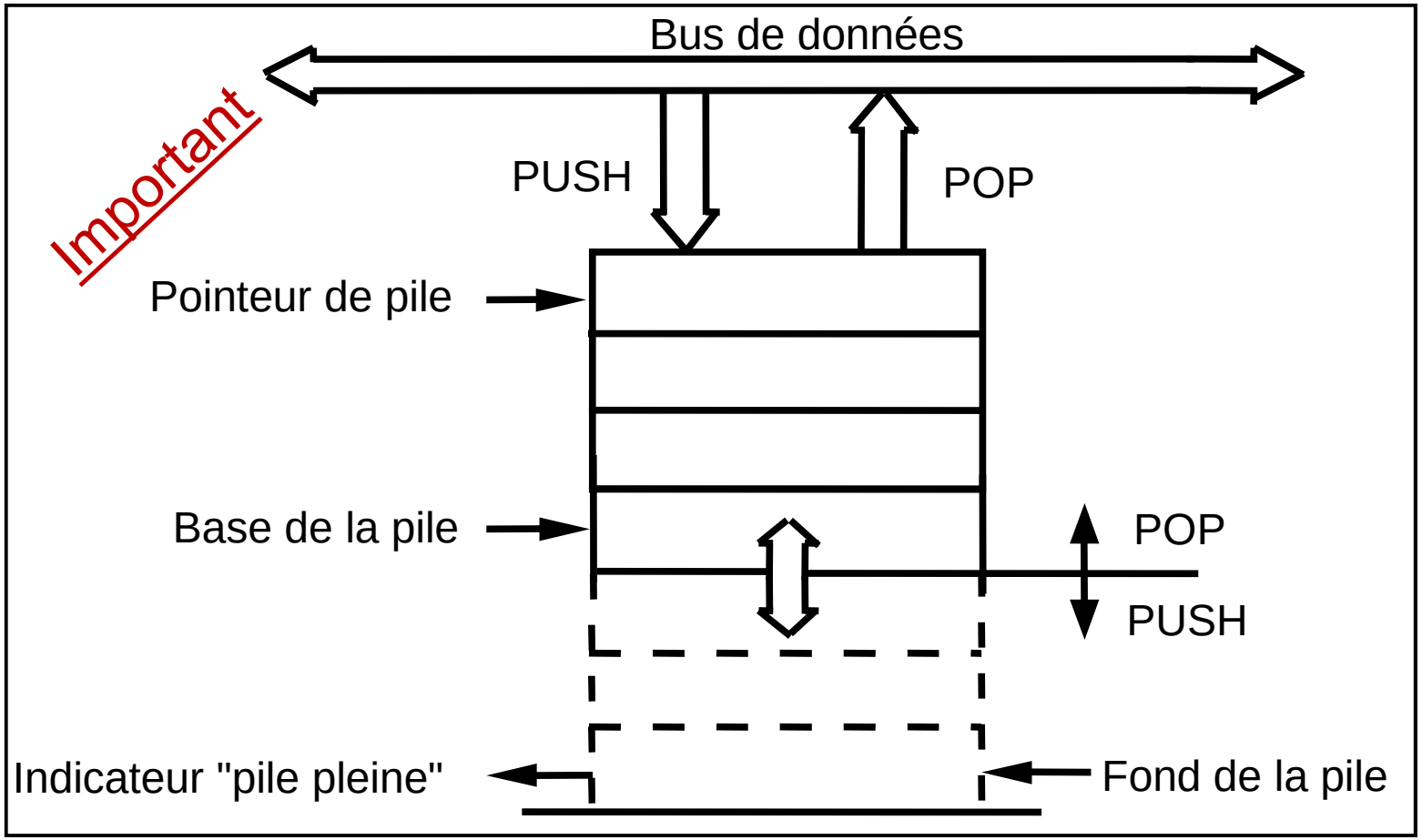
- ✓ Sont destinés au stockage des adresses.
- ✓ Sont connectés au bus des adresses.
- ✓ Tous les microprocesseurs comportent au moins un registre d’adresses, le PC.



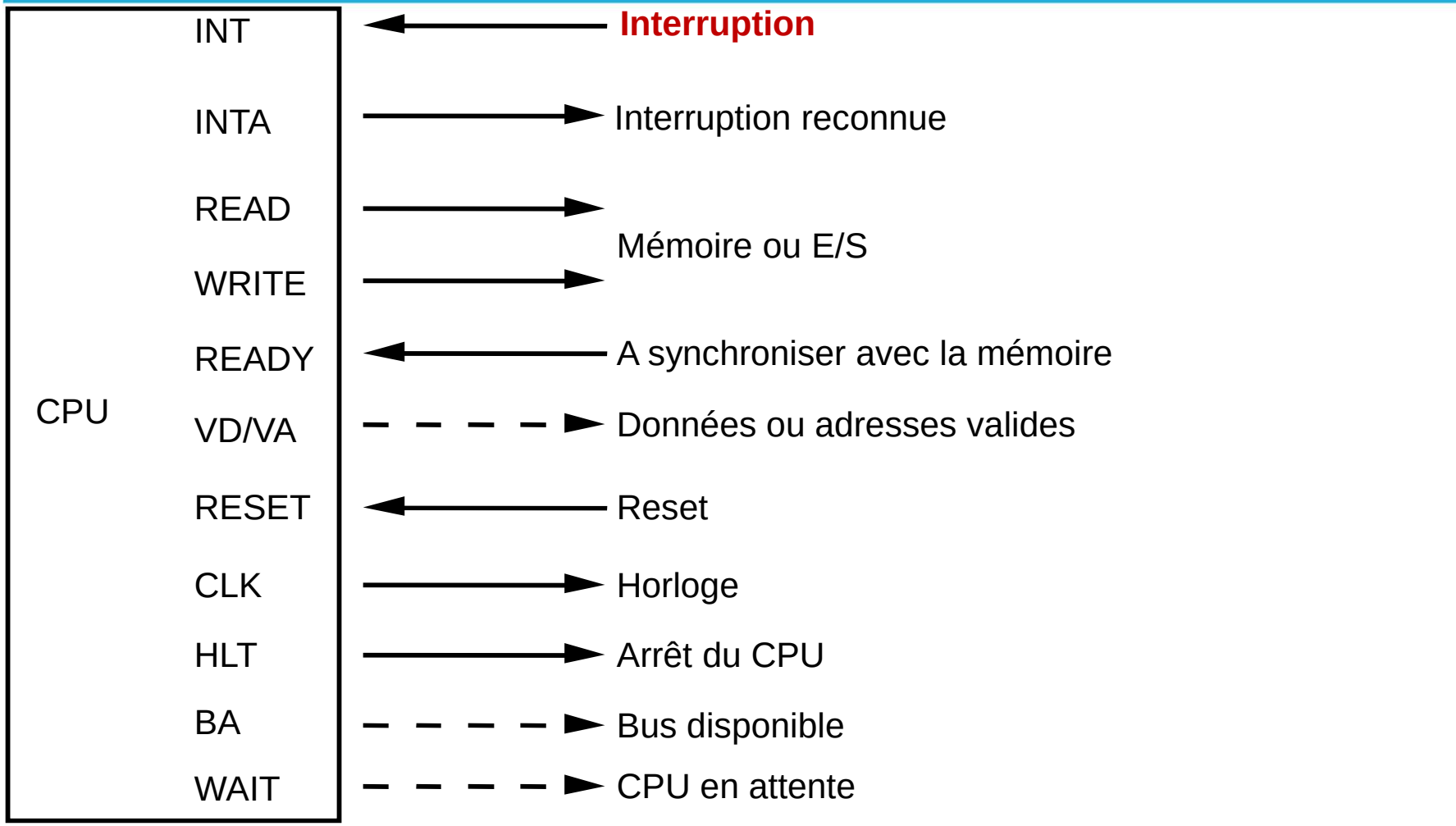
MOV A, R : comment ?

Les registres spéciaux : la pile

C’est une structure LIFO implantée dans la mémoire RAM externe



Bus de commande : exemples

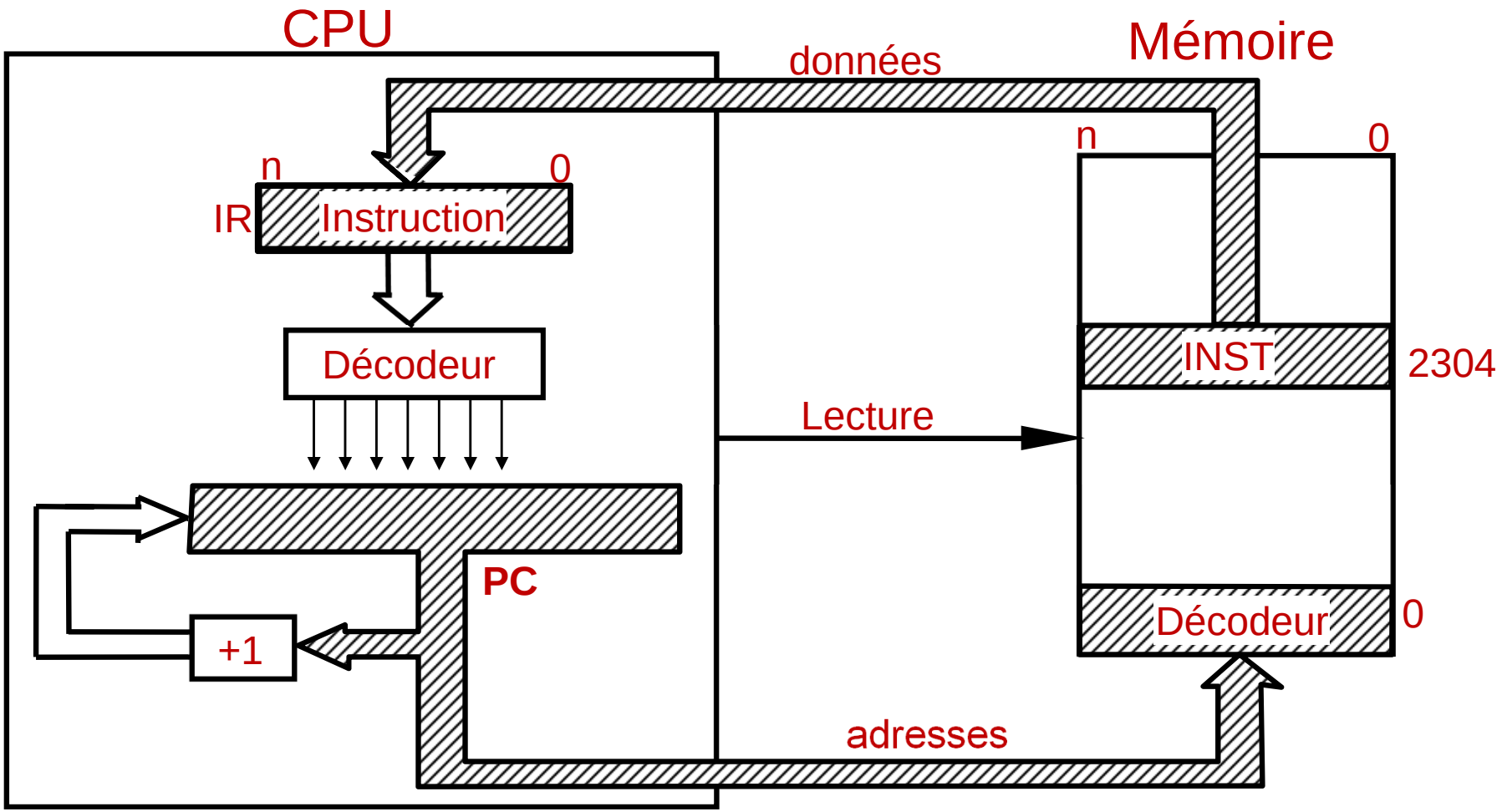


Exécution d'une instruction

Chaque instruction est exécutée en une séquence de cinq phases :

- la phase recherche de l'instruction (fetch),
- la phase décodage de l'instruction (decode),
- la phase recherche des données,
- la phase exécution de l'instruction (execute),
- la phase mémorisation des résultats.

Illustration :
la phase recherche d'une instruction



Exercice : une machine fictive (questions)

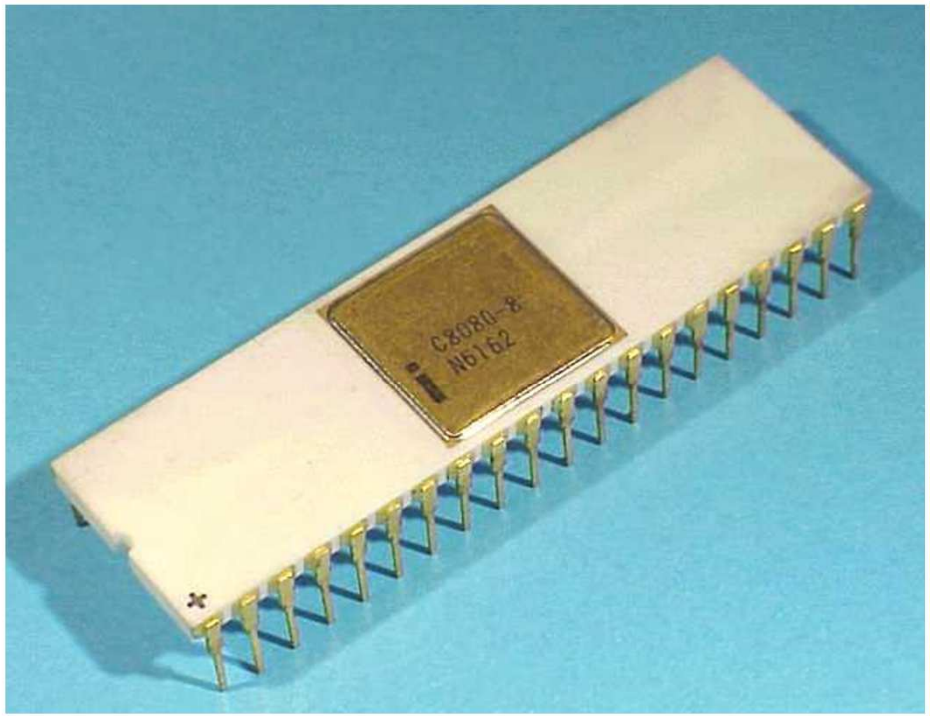
La machine sur laquelle porte l'exercice comporte les éléments suivants :

- ✓ Une mémoire principale de 32 octets,
- ✓ Un registre de travail de 8 bits (nommé ACCU) constituant une des deux entrées de l'UAL avant exécution de l'opération et le résultat de l'opération ensuite (l'Unité Arithmétique et Logique (UAL) est capable d'exécuter seulement les opérations d'addition et de soustraction).
- ✓ Un registre de 2 bits (nommé FLAGS) qui sont positionnés en fonction du résultat de l'UAL (Bit0=1 si le résultat est nul, Bit1=1 si le résultat est négatif).
- ✓ Un registre compteur d'instruction (IP) qui contient l'adresse de la prochaine instruction à exécuter.
- ✓ Un registre instruction (INST) qui contient le code de l'instruction courante.

Exercice : une machine fictive (énoncé)

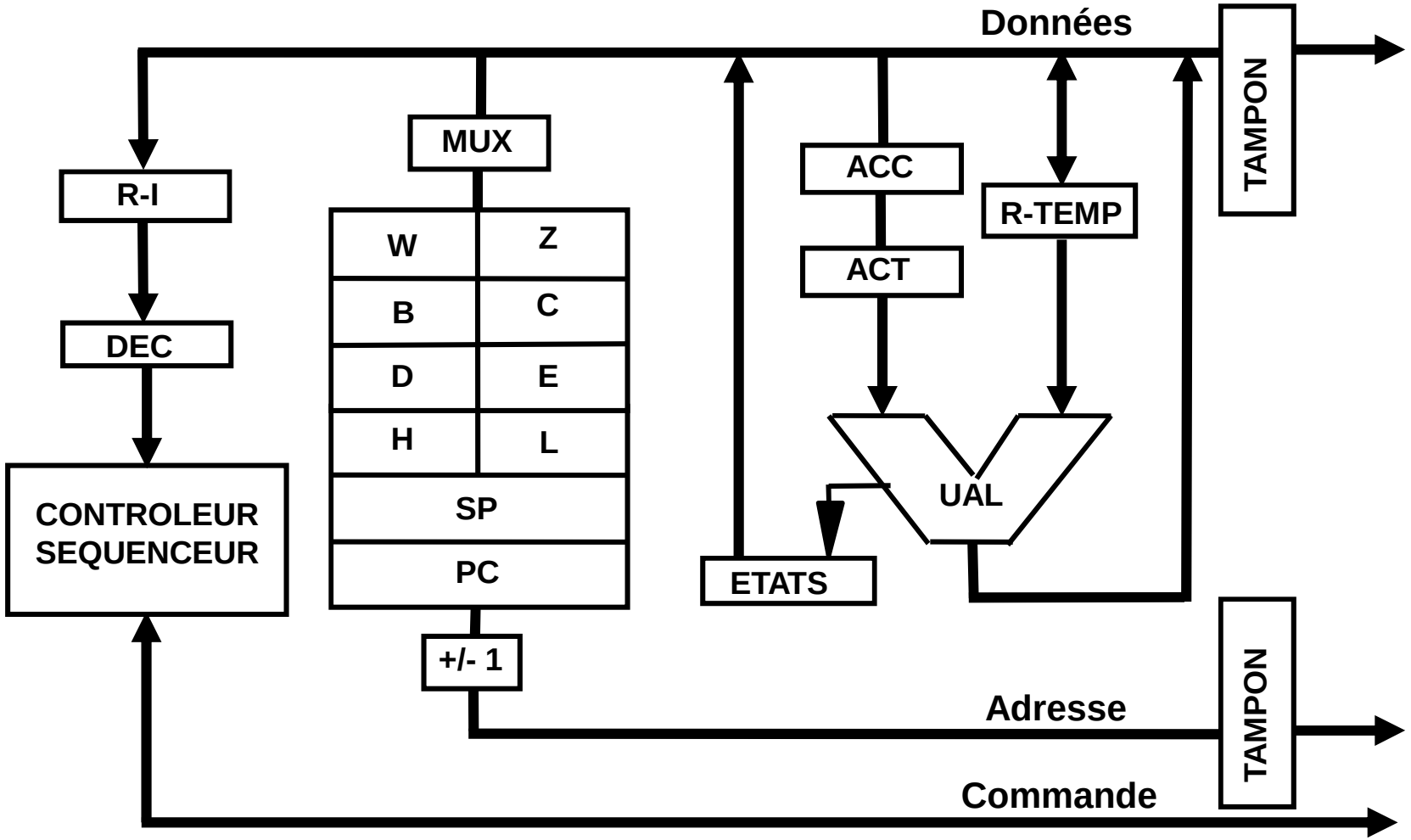
- Donner le schéma de principe de cette machine ?
- Définition du langage binaire : Combien de bits faut-il pour représenter une adresse ?
- En supposant que le nombre d'instructions disponibles est de 8 et que chaque instruction est représentée sous la forme [Codop Adresse], combien de bits faut-il pour coder une instruction ? (Codop = Code opératoire).
- Définition du langage symbolique : La liste des instructions est la suivante: addition, soustraction, chargement du registre ACCU (depuis la mémoire), rangement du registre ACCU (dans la mémoire), branchement, branchement si nul, branchement si négatif, fin. Donner un code binaire (Codop) et un nom symbolique à chacune d'entre elles.
- Ecrire un programme (en langage symbolique) dont la première ligne sera codée à l'adresse 10H de la mémoire, permettant d'ajouter deux nombres stockés aux adresses 01H et 02H de la mémoire et rangeant le résultat à l'adresse 03H de la mémoire.

Exemple : le microprocesseur 8080 de INTEL

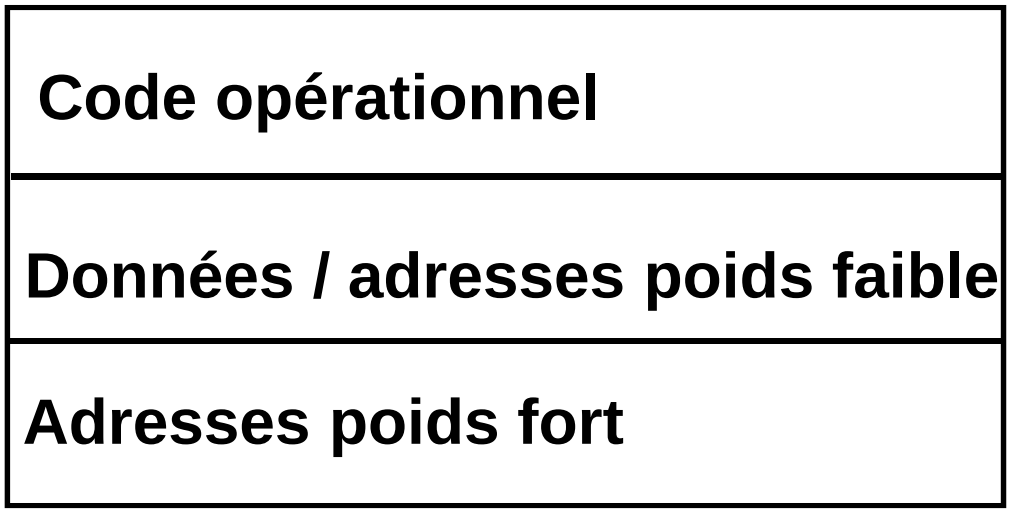


Technologie : 6 µm
Vitesse : 2 – 3 MHz
Transistors : 6000

Le 8080 : Architecture interne



Codage des instructions



MOV r1, r2 01DDDSSS

ADI data 11000110 data

LDA adr 00111010 adresse

Codage en mémoire : exemple

Langage haut niveau

(for, if…then, write, etc…)

LDA 1234	:	3A 34 12
MOV C,B	:	48
ADI 55	:	C6 55

Langage assembleur

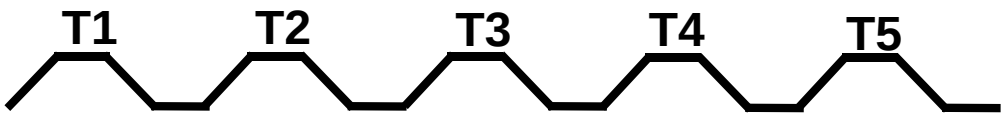
Mémoire

3000H	00111010	3A
	00110100	34
	00010010	12
	01001000	48
	11000110	C6
	01010101	55

Langage machine

Exécutions d'instructions sur le 8080

L'exécution d'une instruction nécessite entre un et cinq cycle machines (accès mémoire). Chaque cycle machine correspond à trois, quatre ou cinq cycle horloge.



La **phase de recherche de l'instruction** correspond aux états T1, T2 et T3 du cycle machine M1.

T1 : Contenu du registre d 'adresse (PC) dans le bus d 'adresses.

T2 : PC = PC +1 (& décodage et préparation de l 'instruction).

T3 : Transfert de l 'instruction dans le registre d 'instruction.

T4 : Décodage de l 'instruction.

T5 : Fonction du type de l 'instruction.

Exemple : MOV D,C

Mnémonique		T1	T2	T3	T4	T5
MOV r1,r2	01DDDSSS	Envoi de PC	PC = PC + 1	INST → IR	(SSS) → TMP	(TMP) → DDD
	Code opérationnel					

T1 : Contenu du registre d ’adresse (PC) dans le bus d ’adresses.

T2 : PC = PC +1 (& décodage et préparation de l ’instruction).

T3 : Transfert de l ’instruction dans le registre d ’instruction.

T4 : Décodage de l ’instruction et transfert de C vers TMP.

T5 : Transfert de TMP vers D.

Durée = 5 fois le cycle horloge

Exemple : LDA addr

T1 de M1 : Contenu du registre d ’adresse (PC) dans le bus d ’adresses.

T2 de M1 : PC = PC +1 (& décodage et préparation de l ’instruction).

T3 de M1 : Transfert de l ’instruction dans le registre d ’instruction.

T4 de M1 : Décodage de l ’instruction.

T1 de M2 : Contenu du registre d ’adresse (PC) dans le bus d ’adresses.

T2 de M2 : PC = PC +1 (& décodage et préparation de la donnée).

T3 de M2 : Transfert du poids faible de l ’adresse dans le registre Z.

T1 de M3 : Contenu du registre d ’adresse (PC) dans le bus d ’adresses.

T2 de M3 : PC = PC +1 (& décodage et préparation de la donnée).

T3 de M3 : Transfert du poids fort de l’adresse dans le registre W.

T1 de M4 : Contenu du registre d ’adresse WZ dans le bus d ’adresses

T2 de M4 : (& décodage et préparation de la donnée).

T3 de M4 : Transfert de la donnée dans ll ’accumulateur.

Exemple : JMP addr

T1 de M1 : Contenu du registre d ’adresse (PC) dans le bus d ’adresses.

T2 de M1 : PC = PC +1 (& décodage et préparation de l ’instruction).

T3 de M1 : Transfert de l ’instruction dans le registre d ’instruction.

T4 de M1 : Décodage de l ’instruction.

T1 de M2 : Contenu du registre d ’adresse (PC) dans le bus d ’adresses.

T2 de M2 : PC = PC +1 (& décodage et préparation de la donnée).

T3 de M2 : Transfert du poids faible de l ’adresse dans le registre Z.

T1 de M3 : Contenu du registre d ’adresse (PC) dans le bus d ’adresses.

T2 de M3 : PC = PC +1 (& décodage et préparation de la donnée).

T3 de M3 : Transfert du poids fort de l ’adresse dans le registre W.

T1 de $M1_{n+1}$: Contenu du registre d ’adresse WZ dans le bus d ’adresses

T2 de $M1_{n+1}$: PC = WZ +1 (& décodage et préparation de la donnée).

Exemple : CALL adresse

T1 de M1 : Contenu du registre d ’adresse (PC) dans le bus d ’adresses.
T2 de M1 : PC = PC +1 (& décodage et préparation de l ’instruction).
T3 de M1 : Transfert de l ’instruction dans le registre d ’instruction.
T4 de M1 : décodage de l ’instruction.
T5 de M1 : SP = SP -1.
T1 de M2 : Contenu du registre d ’adresse (PC) dans le bus d ’adresses.
T2 de M2 : PC = PC +1 (& décodage et préparation de la donnée).
T3 de M2 : Transfert du poids faible de l ’adresse dans le registre Z.
T1 de M3 : Contenu du registre d ’adresse (PC) dans le bus d ’adresses.
T2 de M3 : PC = PC +1 (& décodage et préparation de la donnée).
T3 de M3 : Transfert du poids fort de l ’adresse dans le registre W.
T1 de M4 : Contenu du registre SP dans le bus d ’adresses.
T2 de M4 : SP = SP -1 (& décodage et préparation de la donnée).
T3 de M4 : Transfert du poids fort du PC vers la pile.
T1 de M5 : Contenu du registre SP dans le bus d ’adresses.
T2 de M5 : (& décodage et préparation de la donnée).
T3 de M5 : Transfert du poids faible du PC vers la pile.
T1 de $M1_{n+1}$: Contenu du registre d ’adresse WZ dans le bus d ’adresses
T2 de $M1_{n+1}$: PC = WZ +1 (& décodage et préparation de la donnée).

Exemple : RET

T1 de M1 : Contenu du registre d ’adresse (PC) dans le bus d ’adresses.
T2 de M1 : PC = PC +1 (& décodage et préparation de l ’instruction).
T3 de M1 : Transfert de l ’instruction dans le registre d ’instruction.
T4 de M1 : décodage de l ’instruction.
T1 de M2 : Contenu du registre SP dans le bus d ’adresses.
T2 de M2 : SP = SP +1 (& décodage et préparation de la donnée).
T3 de M2 : Transfert de l ’octet de la pile vers Z.
T1 de M3 : Contenu du registre SP dans le bus d ’adresses.
T2 de M3 : SP = SP +1 (& décodage et préparation de la donnée).
T3 de M3 : Transfert de l ’octet de la pile vers W.
T1 de $M1_{n+1}$: Contenu du registre d ’adresse WZ dans le bus d ’adresses
T2 de $M1_{n+1}$: PC = WZ +1 (& décodage et préparation de la donnée).

Exercice : exécution d'un programme sur microprocesseur de type 8080

Soit le programme assembleur 8080 suivant :

Adresse	Mnémonique	Code machine (instruction)
0050 H	MVI H , #00H;	0 0 1 0 0 1 1 0
. . . . . .	MVI L , #80H;	0 1 0 1 0 0 0 1
. . . . . .	JMP 0070H;	1 1 0 0 0 0 1 1
. . . . . .	. . . . . .;	. . . . .
0070 H	ADD M;	1 0 0 0 0 1 1 0
. . . . . .	. . . . . .;	. . . . .

Représenter, en précisant les adresses, le contenu de la mémoire sachant que la position mémoire M contient la valeur AAH.

Sachant que l'accumulateur contient la valeur D5H et que le pointeur de code contient la valeur 0050H, dérouler pas a pas le programme et donner le contenu des registres A, H, L, W, Z, PC et le registre d'états après l'exécution du programme.

Calculer le temps nécessaire à l'exécution de ce programme sachant que l'horloge utilisé est de 2 Mhz.

Exercice : exécution d'un programme sur microprocesseur de type 8080

Signification des instructions :

Adresse	Mnémonique	Code machine (**instruction**)
0050 H	MVI H , #00H;	0 0 1 0 0 1 1 0
. . . . . .	MVI L , #80H;	0 1 0 1 0 0 0 1
. . . . . .	JMP 0070H;	1 1 0 0 0 0 1 1
. . . . . .	. . . . . .;	. . . . .
0070 H	ADD M;	1 0 0 0 0 1 1 0
. . . . . .	. . . . . .;	. . . . .

MVI H , #00H : le registre H est initialisé à la valeur 00H.
MVI L , #80H : le registre L est initialisé à la valeur 00H.
JMP 0070H : saut à l'adresse 0070H.
ADD M : A = A + [M], M = contenu de ce qui est mémorisé à l'adresse [H:L]

Questions à retenir

- Quels sont les bus externes standards d'un microprocesseur, lequel est totalement monodirectionnel.
- Quelles sont les différences entre les architectures de type Von Neumann et les architectures de type Harvard.
- Quelles sont les différences entre les jeux d'instructions CISC et RISC.
- Sur les machines séquentielles chaque instruction est exécutée en une séquence de cinq phases, rappeler ces cinq phases.
- Quel est le rôle du registre d'état.
- Quel est le rôle du pointeur de code.
- Quelle est le rôle de la pile dans un système à microprocesseur.
- Quel est le rôle du pointeur de pile.
- Comment calculer la capacité d'adressage mémoire d'un microprocesseur.
- A quoi sert la mémoire dans un système à microprocesseur.
- A quoi sert le décodeur d'adresses.