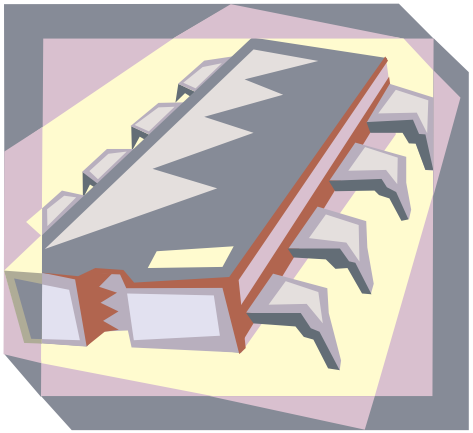
Ecole polytechnique de l'université de Nice-Sophia Antipolis

Département Electronique



Systèmes à Microprocesseurs

Cycle Ingénieur Première Année

S. Bilavarn

Polytech'Sophia – Département Electronique

bilavarn@unice.fr

Plan

- Ch1 – Historique
- Ch2 – Généralités sur les architectures
- Ch3 – Traitement des données sur ARM
- Ch4 – Jeu d'instructions ARM
- Ch5 – Programmation structurée en assembleur ARM
- Ch6 – Exécution du processeur
- Ch7 – Représentation en mémoire

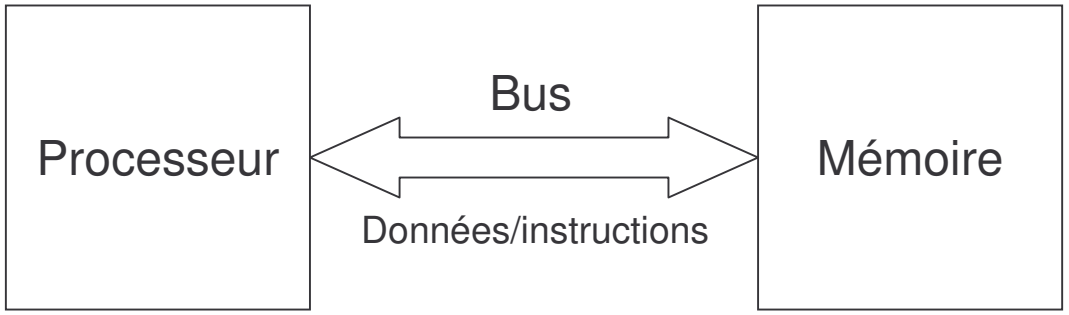
Ecole polytechnique
de l'université de Nice-Sophia Antipolis
Département Electronique

Généralités sur les microprocesseurs

- Types d'architecture
- Principaux éléments
- Structure de l'unité centrale
- Notions et représentation

Architecture de Von Neuman

- Les premières machines de calcul se basaient sur des programmes fixes (non modifiables, ex: calculatrice)
- L'architecture de Von Neuman (1945) est un modèle qui utilise une structure unique pour les instructions et les données
  - Notion de programme stocké en mémoire
  - Possibilité de programmer la machine
  - Séparation entre le stockage et le processeur
- Inconvénient
  - Architecture limitée par l'élément le plus lent (transfert mémoire)



Architecture de Harvard

- L'architecture de Harvard sépare physiquement la mémoire de données de la mémoire de programme
  - Permet de transférer simultanément les données et les instructions à exécuter
  - L'accès à chaque mémoire par deux bus distincts
  - Plus rapide que le modèle Von Neuman, mais structure plus complexe

Processeur CISC

- Complex Instruction Set Computing
- Avant les années 80, la tendance était d'augmenter le nombre d'instructions du processeur de manière à faciliter son utilisation.
  - En une seule instruction, traitement de séquences complexes d'opérations
  - Permettait d'éviter l'appel à plusieurs instructions et donc plusieurs accès (lents) à la mémoire
- Inconvénient: seule une petite proportion du jeu d'instruction est utilisée en pratique

- Exemples: Motorola 68000, Intel Pentium

Processeur RISC

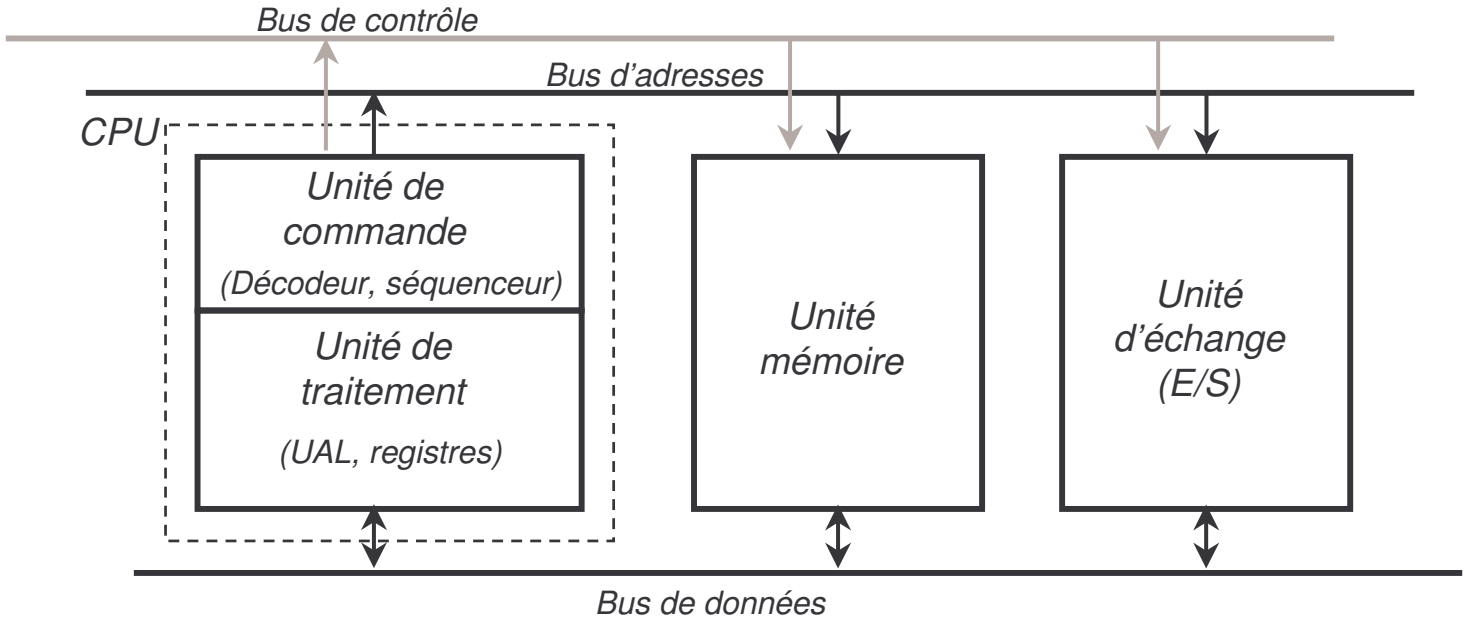
- Reduced Instruction Set Computer
- 1980: conception du premier processeur RISC (Berkeley)
  - Architecture beaucoup plus simple à concevoir
  - Niveau de performance similaire
- Principe:
  - Format fixe d'instruction (32-bit), 1 instruction = 1 cycle
  - Architecture load-store: les instructions de traitement opèrent sur des registres et sont séparées des instructions d'accès à la mémoire
  - Une file de registres importante (32) pour permettre à l'architecture load-store d'opérer efficacement
- Ex: ARM (Advanced RISC Machine)

Généralités sur les microprocesseurs

- Types d'architecture
- Principaux éléments
- Structure de l'unité centrale
- Notions et représentation

Unités d'un microprocesseur standard

- 3 unités fondamentales:
  - Unité centrale de traitement (CPU)
  - Unité de mémoire
  - Unité d'échange (modules d'entrées-sorties)

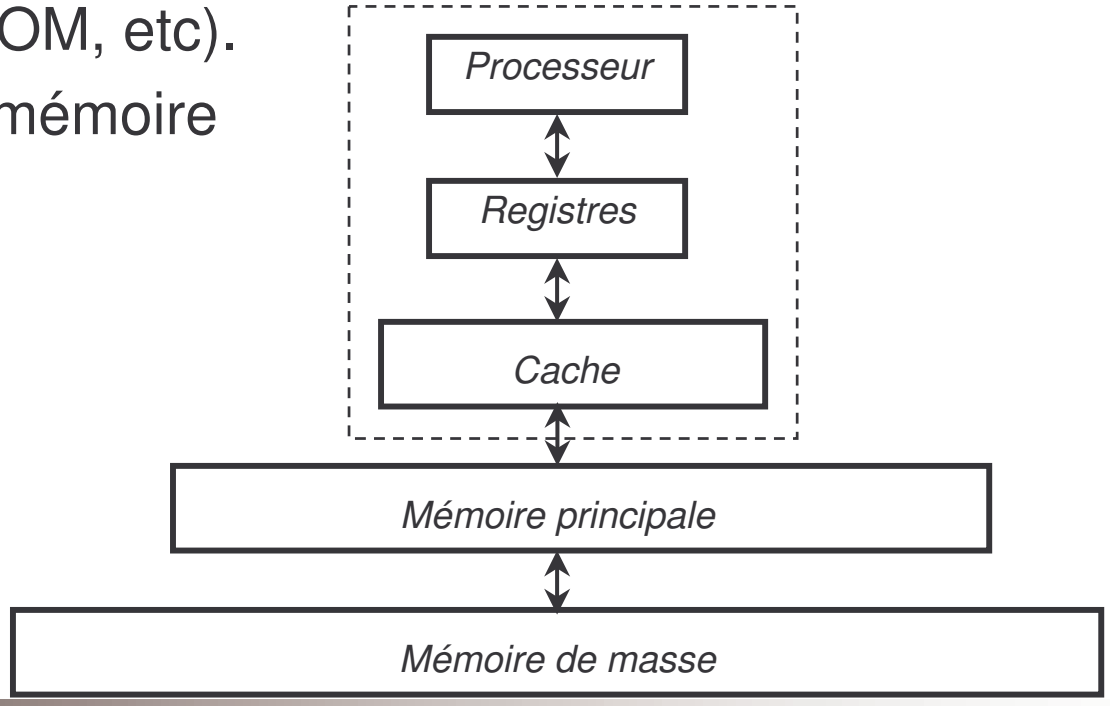


Les unités fonctionnelles

- La CPU (Central Processing Unit) regroupe trois modules:
  - L'UAL (Unité Arithmétique et Logique)
    - Réalise les opérations arithmétiques (additions, soustractions, …) et logiques (ET, OU, …).
  - Les registres
    - Stockent les données en cours de traitement.
  - L'unité de commande (UC)
    - Séquenceur (logique de contrôle): régir le séquencement de l'ensemble du système en gérant les signaux de commande et de contrôle, l'occupation des bus d'adresse et de données.
    - Décodeur: chercher, décoder et exécuter les instructions.

Les unités fonctionnelles

- La mémoire
  - La *mémoire principale* sert à stocker les programmes en cours d'exécution et les données manipulées.
  - La *mémoire de masse* sert à stocker les programmes ou données qui ne sont pas utilisées dans l'immédiat (disques durs, CD ROM, etc).
  - Hiérarchie mémoire

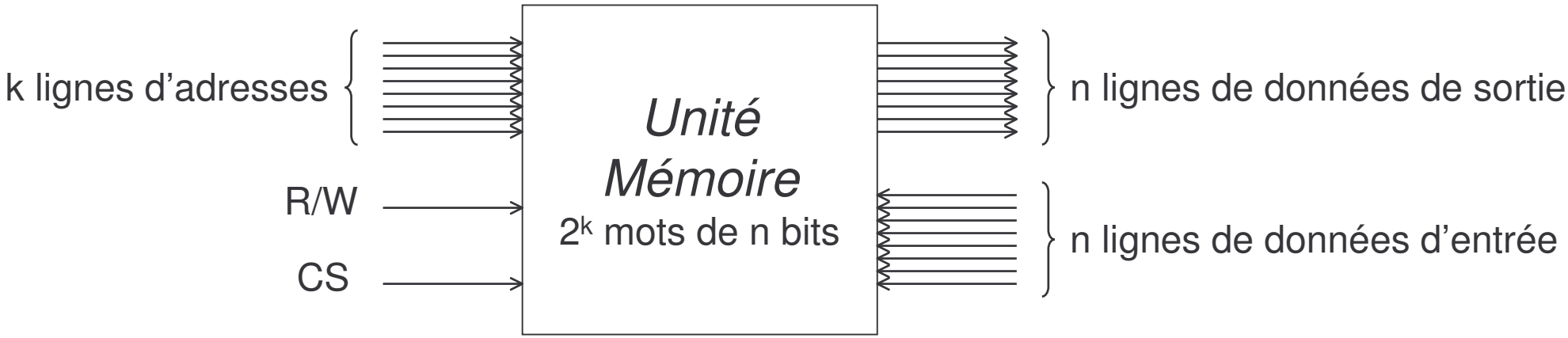


Les liaisons entre unités (bus)

- Les communications entre l'unité centrale (CPU) et les autres modules se font par l'intermédiaire de bus.
- 3 types de bus:
  - Le *bus de commande* assure la synchronisation des opérations du système.
    - Transporte les signaux de contrôle entre UC et autres modules.
  - Le *bus de données* transmet les données échangées par différents modules.
    - Bidirectionnel (lecture, écriture)
    - Format des données (8,16, 32-bit)
  - Le *bus d'adresses* sert à sélectionner la source ou la destination des données.
    - Unidirectionnel
    - Transporte des adresses, sa largeur détermine la capacité d'adressage. Ex: 16-bit 65536 données.

Organisation de la mémoire principale

- k lignes d'adresses permettent l'accès aux cellules mémoire
  - La taille d'un bloc mémoire est 2^k.
- Une ligne de commande R/W
  - Mémorisation de l'information, accès en écriture (R/W = 0)
  - Restitution de l'information, accès en lecture (R/W = 1)
- Port d'écriture
- Port de lecture



Organisation de la mémoire principale

- Exemple: k = 16, n = 8
  - Bus d'adresse de 16 bits, chaque ligne d'adresse peut prendre la valeur 0 ou 1

  → 2^{16} = 65536 adresses
  - Bus de données 8 bits

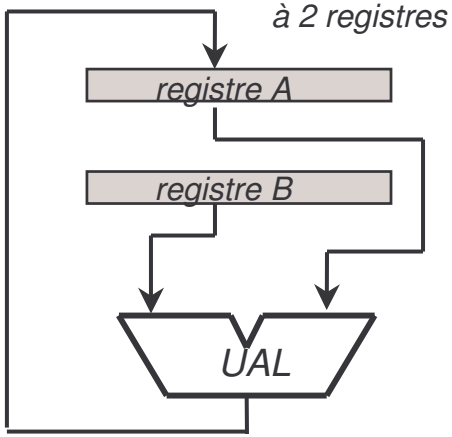
  → Capacité mémoire de 65536 mots de 8 bits
    - 65536 octets
    - 65536/1024 = 64 Koctets
    - 65536*8 = 524288 bits
    - 524288/1024 = 512 Kbits

Symbole	Préfixe	Capacité		
1 K	Kilo	2^{10}	=	1024
1 M	Méga	2^{20}	=	1048576
1 G	Giga	2^{30}	=	1073741824
1 T	Téra	2^{40}	=	1099511627776

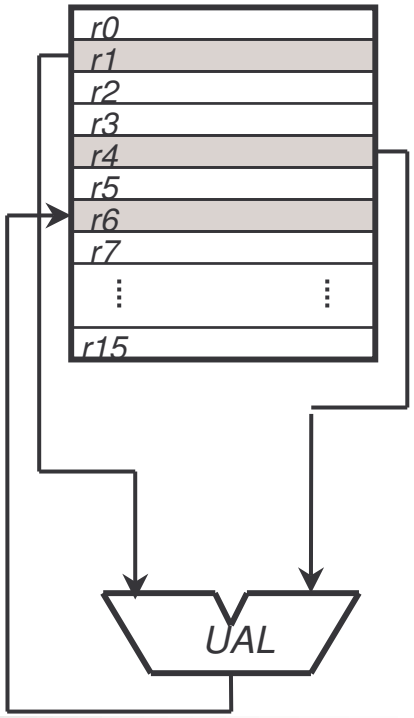
Registres/file de registres

- Les registres sont utilisés comme intermédiaire entre l'UAL et la mémoire car elles n'ont pas de temps d'accès.
- Entrées/sorties de l'UAL
- Un registre contient un mot
- Résultats intermédiaires de calculs

Structure simple à 2 registres



Organisation en file de registres

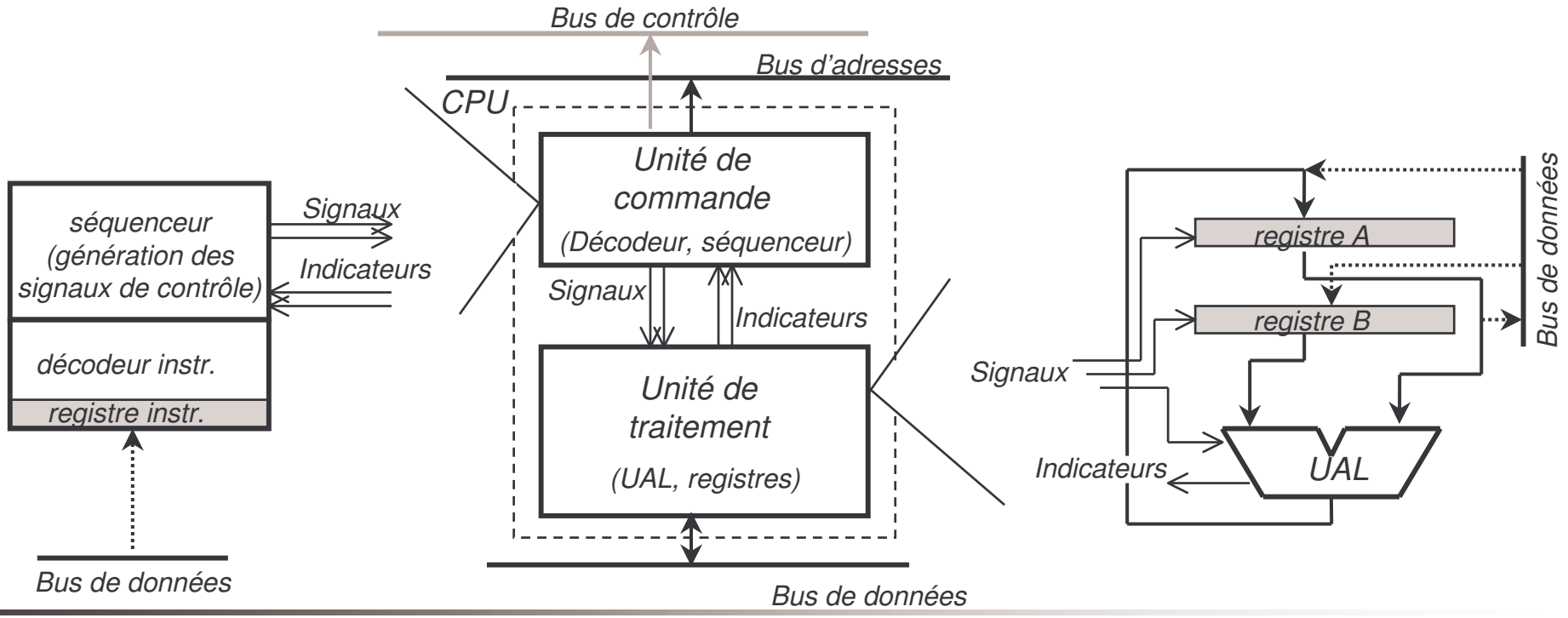


Généralités sur les microprocesseurs

- Types d'architecture
- Principaux éléments
- Structure de l'unité centrale
- Notions et représentation

Structure de l'unité centrale

- Unité de traitement
  - En charge d'effectuer les opérations de traitement
- Unité de commande
  - Commande l'ensemble en fonction des instructions du programme



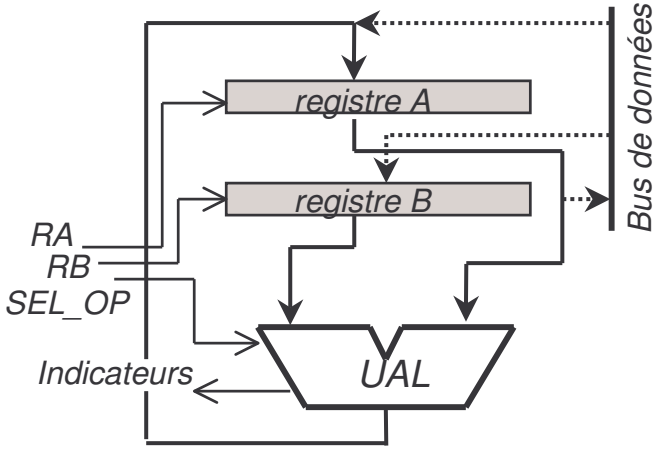
Principe de l'unité de traitement

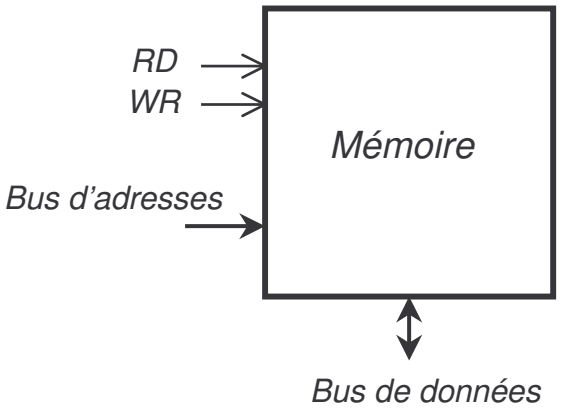
Algorithme

```
Nbre1, Nbre2, Res : Octet
A, B : Registre
Début
  A := Nbre1;
  B := Nbre2;
  A := A + B;
  res := A;
Fin
```

Micro-Instructions → **Signaux**

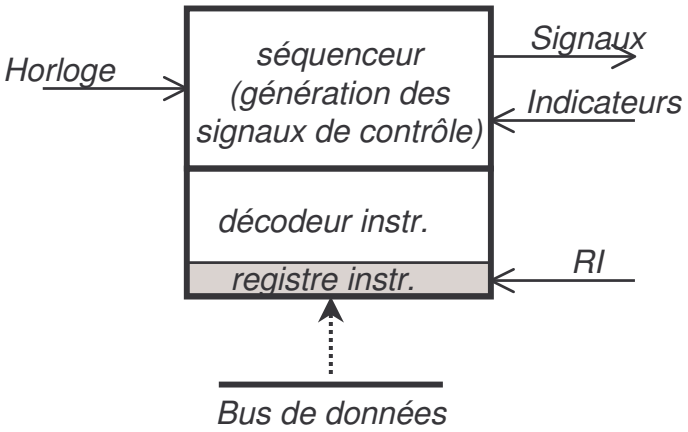
- Bus d'adresses ← @Nbre1
- Lecture mémoire → *RD*
- A ← Bus de données → *RA*
- Bus d'adresses ← @Nbre2
- Lecture mémoire → *RD*
- B ← Bus de données → *RB*
- ADD → *SEL_OP*
- Bus d'adresses ← @res
- Bus de données ← A
- Écriture mémoire → *WR*

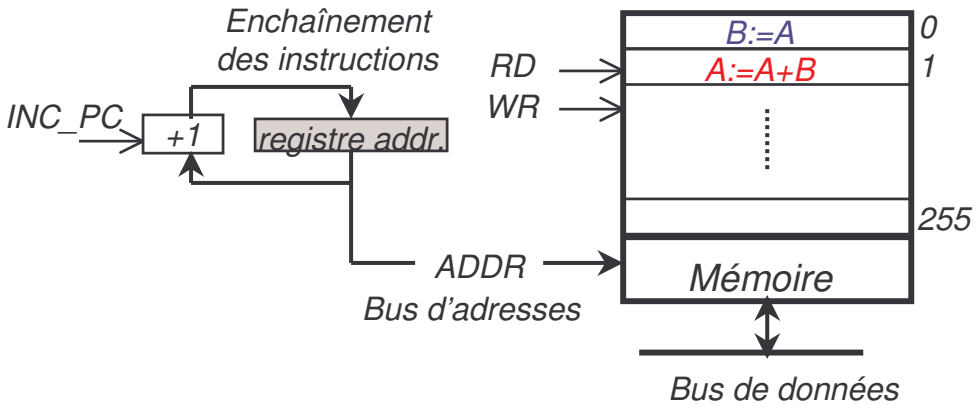




Principe de l'unité de commande

Algorithme	Description	Signaux
A, B : Registre Début B := A; A := A + B; Fin	Les instructions sont contenues en mémoire à partir de l'adresse 0. PC contient l'adresse de l'instruction courante Bus d'adresses ← 0 Registre instr. ← B := A Action: lect. A, positionnement bus, écr. B PC ← PC + 1 Bus d'adresses ← 1 Registre instr. ← A := A + B Action: sélection de l'addition, écr. A	*ADDR, RD* *RI* *RB* *INC_PC* *ADDR, RD* *RI* *SEL_OP, RA*





Généralités sur les microprocesseurs

- Types d'architecture
- Principaux éléments
- Structure de l'unité centrale
- Notions et représentation

Notions

- Programme: séquence d'instructions décrivant au microprocesseur ce qu'il doit faire.
- Exécution: réalise une séquence d'instructions.
- Langage de programmation: langage pour écrire des instructions au microprocesseur.
- Langages principaux
  - Langage machine ou code objet (binaire)
  - Assembleur (mnémoniques)
  - Haut niveau (C, Pascal)
- Compilateur: transforme un programme en langage de haut niveau en séquence d'instruction machine (code objet).

Le jeu d'instructions

- Chaque microprocesseur reconnaît un ensemble d'instruction appelé jeu d'instructions
  - CISC: 75 – 100 instructions
  - RISC: 10- 30 instructions
- Une instruction est définie par son code opératoire qui est une valeur numérique binaire
  - On utilise une notation symbolique: les mnémoniques
  - Un programme constitué de mnémoniques est appelé programme assembleur
- Langage machine et assembleur
  - Assembleur: logiciel de traduction du code source écrit en langage assembleur (mnémoniques)
  - Langage machine: codes binaires correspondant aux instructions et exploitable par la machine

Représentation de l'information (1)

- Types de données de base
  - Nombres entiers (0, -23, 0, 6, 65635)
  - Nombres réels (0.3, -3.2, 5, 10e3)
  - Booléens (true, false)
  - Caractères ('a', 'b', '?', '&', '0')
- Systèmes de numération
  - Décimal: 0 … 9, décomposition en puissances de 10
  - Hexadécimal: 0 à 9 et A…F, décomposition en puissances de 16
  - Binaire: 0 ou 1, décomposition en puissances de 2
- Décimal et hexadécimal sont utilisables pour l'être humain
- La représentation binaire est utilisable par l'ordinateur

Représentation de l'information (2)

- Mot binaire
  - Bit (Binary digit)
  - Quartet (nibble) 4 bits
  - Octet (byte) 8 bits
- Terminologie utilisée sur les processeurs 32-bit (y compris ARM)
  - Demi-mot (half-word) 16 bits
  - Mot (word) 32 bits
  - Double-mot (double word) 64 bits
- /!\ la terminologie peut changer d'une famille à l'autre
  - Motorola, Intel
    - word 16 bits, long word 32 bits, quad word 64 bits

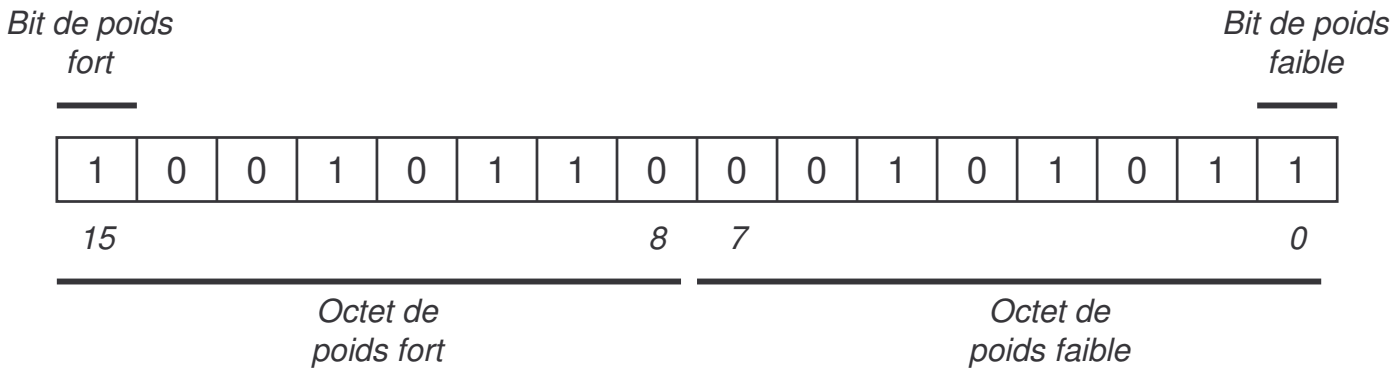
Représentation des nombres entiers (1)

- Binaire pur sur n bits
  - Représentation des entiers entre 0 et 2^n-1

- Exemples:
  - $(00000000)_{bin} = (0)_{dec}$
  - $(11111111)_{bin} = (2^8\text{-}1)_{dec} = (255)_{dec}$
  - $(11001001)_{bin} = 1x\ 2^7 + 1x\ 2^6 + 0x\ 2^5 + 0x\ 2^4 + 1x\ 2^3 + 0x\ 2^2 + 0x\ 2^1 + 1x\ 2^0$
    $= (128 + 64 + 8 + 1)_{dec}$
    $= (201)_{dec}$

Représentation des nombres entiers (2)

- Vocabulaire
  - Bit de poids fort: bit le plus significatif (MSB, Most Significant Bit)
  - Bit de poids faible: bit le moins significatif (LSB, Least Significant Bit)



Représentation des nombres entiers (3)

- Correspondance entre binaire et hexadécimal
  - Un chiffre hexadécimal pour 4 bits
  - Exemples:
    - $(53)_{hex} = (0101\ 0011)_{bin}$
    - $(FF)_{hex} = (1111\ 1111)_{bin}$
  - Nombres négatifs
    - En décimal on fait figurer le signe –
    - En hexadécimal par convention, on traduit directement à partir du binaire
  - Exemple en complément à deux sur 8 bits
    - $(-1)_{dec} = (1111\ 1111)_{bin} = (FF)_{hex}$

Représentation des caractères

- Le code ASCII
  - American Standard Code for Information Interchange
  - 1 caractère = 1 octet
  - Exemples:

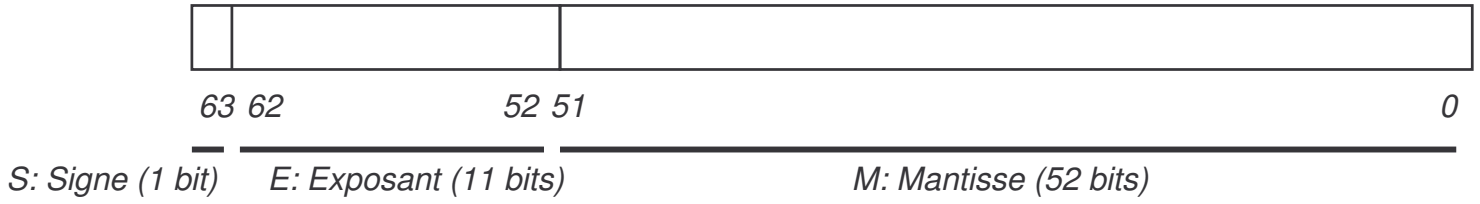
Décimal	Caractère	Décimal	Caractère	Décimal	Caractère
32	ESPACE	48	0	91	[
33	!	49	1	92	\
34	‘’	…	…	63	]
35	#	57	9	…	
36	$	…	…	…	
37	%	65	A	97	a
38	&	66	B	98	b
39	’	…	…	…	…
40	(	90	Z	122	z

Représentation des nombres réels (1)

- Le standard IEEE 754 (32-bit, simple précision)

$$(-1)^{S}*(1 + M/2^{23}) * 2^{E-127}$$

- Le standard IEEE 754 (64-bit, double précision)



$$(-1)^{S}*(1 + M/2^{52}) * 2^{E-1023}$$

Représentation des nombres réels (2)

- Exemple:

01000100 01000001 01010100 01000101

- Signe: 0 positif
- Exposant: 10001000 = $(136)_{dec}$
- Mantisse: 1000001 01010100 01000101 = $(4281413)_{dec}$

- Résultat: $(1 + 4281413/2^{23}) * 2^{136-127} =$

$= 1.5103842 * 2^{9} = 773.31671$