

Systèmes d'exploitation et microprocesseurs

Principe des microprocesseurs

UBO, L3 informatique, parcours CDA
Stéphane Rubini

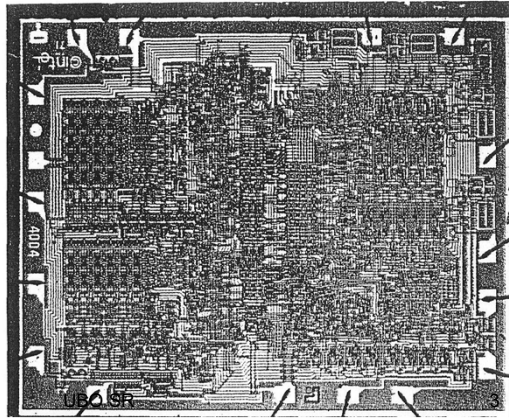
Le triste travail d'un processeur ...



- lecture de l'instruction
(chercher en mémoire une valeur codant les opérations à effectuer)
- mise à jour du compteur ordinal (ou de programme)
(préparation pour l'exécution de l'instruction suivante)
- décodage de l'instruction
(quelles sont les opérations codées dans l'instruction ?)
- lecture des opérandes de l'instruction
(chercher les données à traiter)
- exécution de l'instruction
(traitement des données)
- rangement des résultats
(enregistrement du résultat du traitement)

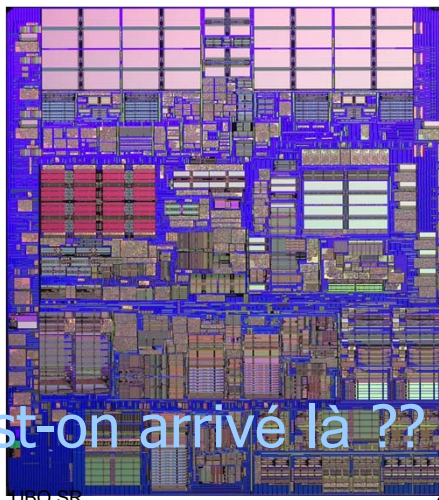
Du premier microprocesseur, l'Intel 4004 ...

- Processeur 4 bits
- Conçu en 1971
- 2300 transistors



... à l'IBM PPC970 (Apple G5)

- IBM, Apple, Motorola
- 64 bits
- 55 millions de transistors



Pourquoi en est-on arrivé là ??



Plan

- Représentation des nombres
- Circuits logiques
 - Combinatoires
 - Séquentiels
- Architecture d'un microprocesseur
 - Structure générale
 - Accès mémoire
 - Jeu d'instructions
 - Interruptions

UBO SR

5



La représentation des nombres entiers

- Base de numération B
- Chiffres : B symboles différents, représentant des quantités de 0 à B-1
- Représentation implicite :
$$N = x_{n-1}x_{n-2} \dots x_i \dots x_0$$
- Représentation explicite :
$$N = \sum_{i=0}^{n-1} B^i \cdot x_i$$
- B^i est le « poids » du chiffre dans le nombre.

UBO SR

6

Les bases utilisées en informatique

- La base 10 bien sûr ! Sauf exception, les programmeurs comptent en base 10 comme le quidam ordinaire.
- La base 2 : les opérateurs de calcul sont souvent prévus pour calculer dans cette base, car leur réalisation en est simplifiée.
- La base 16 ou hexadécimal : elle est utilisée lorsque le programmeur doit raisonner en base 2. Les conversions de la base hexadécimal vers la base 2 se calculent très facilement de tête.

UBO SR

7

Pourquoi la base 16 (hexadécimal)

- Remarque : $16^0 = 2^0, 16^1 = 2^4, 16^2 = 2^8, \dots$
- $$\begin{aligned} N_2 &= \sum_{i=0}^{n-1} 2^i \cdot x_i \\ &= \sum_{j=0}^{\frac{n}{4}-1} 2^{4j} \cdot (2^3 x_{4j+3} + 2^2 x_{4j+2} + 2^1 x_{4j+1} + 2^0 x_{4j}) \\ &= \sum_{j=0}^{\frac{n}{4}-1} 16^j \cdot (2^3 x_{4j+3} + 2^2 x_{4j+2} + 2^1 x_{4j+1} + 2^0 x_{4j}) \\ N_{16} &= \sum_{j=0}^{\frac{n}{4}-1} 16^j \cdot y_j \end{aligned}$$
- La conversion de 4 chiffres binaires donne un chiffre hexadécimal

UBO SR

8



Conversion des chiffres hexadécimaux

$$(0)_{16}=(0000)_2$$

$$(1)_{16}=(0001)_2$$

$$(2)_{16}=(0010)_2$$

$$(3)_{16}=(0011)_2$$

$$(4)_{16}=(0100)_2$$

$$(5)_{16}=(0101)_2$$

$$(6)_{16}=(0110)_2$$

$$(7)_{16}=(0111)_2$$

$$(8)_{16}=(1000)_2$$

$$(9)_{16}=(1001)_2$$

$$(A)_{16}=(1010)_2$$

$$(B)_{16}=(1011)_2$$

$$(C)_{16}=(1100)_2$$

$$(D)_{16}=(1101)_2$$

$$(E)_{16}=(1110)_2$$

$$(F)_{16}=(1111)_2$$

UBO SR

9



Format des nombres : définitions

- *Bit* (ou chiffre binaire) : 0 ou 1
- Octet (ou *byte*) : un nombre codé sur 8 bits
- Mot (ou *word*) : un nombre codé sur 16, 32, ou 64 bits actuellement ; le format dépend de l'ordinateur considéré.
- Mot long (ou *long word*) : un nombre codé sur 32 ou 64 bits.

UBO SR

10

Complément à 2

- Le « complément à 2 » permet de coder des nombres entiers positifs ou négatifs. Il est utilisé dans la grande majorité des ordinateurs.
- Si $x < 0$, sur n bits, le codage de x en complément à 2 est égal à $2^n - |x|$.
- Si $x \geq 0$, le codage de x en complément à 2 est x .

UBO SR

11

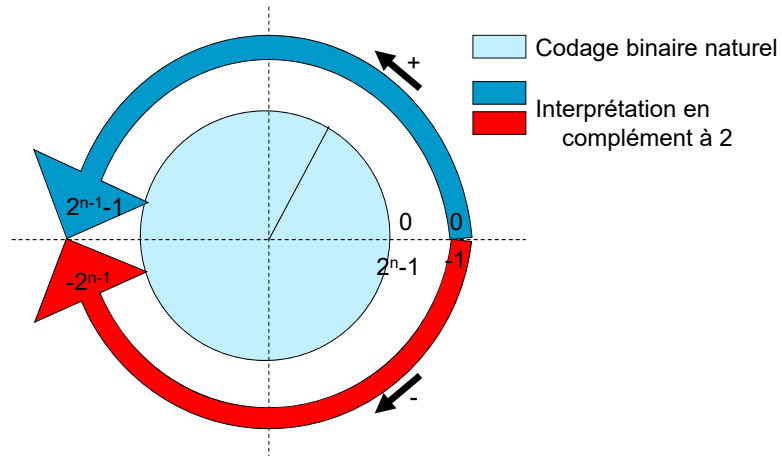
Propriétés du complément à 2

- Une seule représentation du 0
- Soit $\text{code}(x)$ le codage en complément à 2 de x .
- Sur n bits, $\text{code}(x) + \text{code}(y) = \text{code}(x+y)$, quelque soit les signes de x et y .
- Plus mathématiquement :
$$\text{code}(x) + \text{code}(y) \equiv \text{code}(x+y) \pmod{2^n}$$

UBO SR

12

Arithmétique modulo 2^n



UBO SR

13

Méthode pratique pour calculer le complément à 2

- Soit x un nombre entier codé sur n bits.
- Pour trouver le complément à 2 de ce nombre :
 1. On calcule \bar{x} , le complément à 1 de x en prenant le complément logique de chaque chiffre du nombre x
 2. On ajoute 1 au complément à 1
- Exemple :
 - $x=01010011$
 - $\bar{x}=10101100$
 - Code $(-x)=\bar{x}+1=10101100+00000001=10101101$

UBO SR

14

Ordre de grandeur des nombres représentables

Bits	4	8	16	32
Ordinaux	16	256	65536	4294967296
Entiers positifs	7	127	32767	2147483647
Entiers négatifs	-8	-128	-32768	-2147483648

Nombres positifs et négatifs codés en complément à 2

Remarque : les processeurs hautes performances actuels codent les nombres entiers sur 64 bits.

UBO SR

15

Représentation des nombres réels

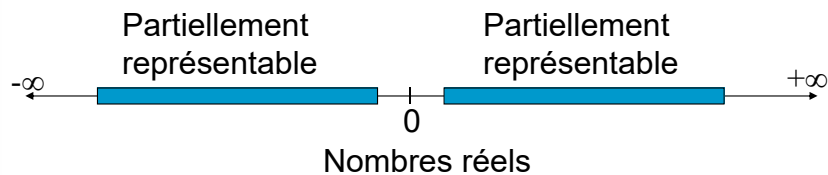
- La notation scientifique est utilisée $N=m.B^e$ où m est la mantisse, B la base et e l'exposant. En informatique, on parle de nombres en virgule flottante.
- Exemple :
 $3,14=3,14.10^0=0,314.10^1=314.10^{-2}$
- En pratique, la virgule flottante est ennuyeuse ; les nombres sont « normalisés » avec un chiffre avant la virgule.

UBO SR

16

Codage des nombres réels

- Les nombres réels (ou plus exactement un sous-ensemble des nombres réels) sont codés sur 4 octets (simple précision) ou 8 octets (double précision).
- Norme IEEE754

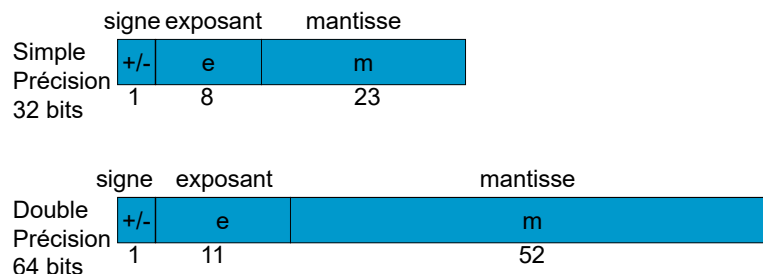


UBO SR

17

IEEE754 : format de représentation

- Simple précision : de $1,175494351 \cdot 10^{-38}$ à $3,402823466 \cdot 10^{+38}$
- Double précision : de $2,2250738585072014 \cdot 10^{-308}$ à $1,7976931348623157 \cdot 10^{+308}$



UBO SR

18

IEEE754 : valeurs représentées

- L'exposant est codé par la méthode dite par excès : la constante 127 est ajoutée à l'exposant réel. Ainsi, seuls des nombres positifs sont manipulés lors des calculs internes.
- La base de numération utilisée est la base 2.
- En simple précision :
$$N = -1^s \times (1+m) \times 2^{e-127}$$
- En double précision :
$$N = -1^s \times (1+m) \times 2^{e-1023}$$

UBO SR

19

IEEE754 : représentations particulières

Zéro	+/-	0	0
+/- infini	+/-	111...111	0
NaN Not a Number	+/-	111...111	≠0

UBO SR

20

Plan

- Représentation des nombres
- Circuits logiques
 - Combinatoires
 - Séquentiels
- Architecture d'un microprocesseur
 - Structure générale
 - Accès mémoire
 - Jeu d'instructions
 - Interruptions

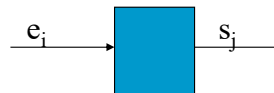
UBO SR

21

Circuits logiques

- Circuit logique combinatoire : l'état des sorties s_j dépend uniquement de l'état courant des entrées e_i .

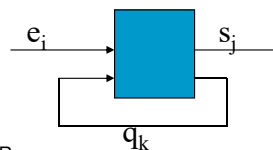
$$s_j = f(\dots, e_i, \dots)$$



- Circuit logique séquentiel : l'état des sorties s_j dépend de l'état courant des entrées e_i et de l'état (\dots, q_k, \dots) de la machine

$$s_j = f(\dots, e_i, \dots, q_k, \dots)$$

$$q_k = g(\dots, e_i, \dots, q_k, \dots)$$

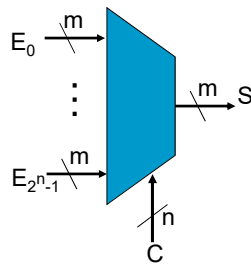


UBO SR

22

Multiplexeur 2^n vers 1

- Un multiplexeur est un dispositif combinatoire permettant de « connecter » une voie d'entrée, désignée par les signaux de contrôle, à la sortie.
- $2^n \times m$ entrées, n entrées de contrôle, m sorties
- $S = \text{mux}(C, E_0, \dots, E_i, \dots, E_{2^n-1}) = E_C$



Multiplexeur 2 vers 1

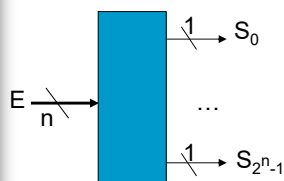
E_0	E_1	C	S
a	x	0	a
x	b	1	b

UBO SR

23

Décodeur n vers 2^n

- Un décodeur n vers 2^n est un dispositif combinatoire. Une seule des sorties est active à un instant donnée. La sortie active est sélectionnée par son numéro d'ordre, codé en binaire naturel, sur le bus d'entrée.
- $S_i = 1$ pour $i=E$,
 $S_i = 0$ pour $i \neq E$



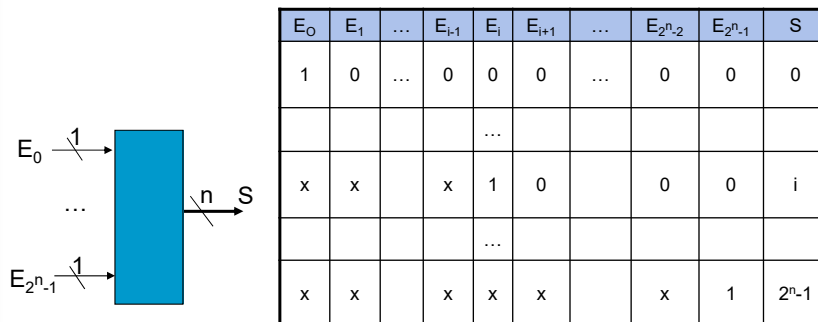
E	S_0	S_1	...	S_{i-1}	S_i	S_{i+1}	...	S_{2^n-2}	S_{2^n-1}
0	1	0	...	0	0	0	...	0	0
...					...				
i	0	0		0	1	0		0	0
...					...				
2^n-1	0	0		0	0	0		0	1

UBO SR

24

Encodeur 2^n vers n

- Un encodeur 2^n vers n est un dispositif combinatoire. La sortie encode le numéro de l'entrée active de rang le plus élevé.
- $S=i$ si $E_i=1$ et $E_j=0$ avec $j>i$

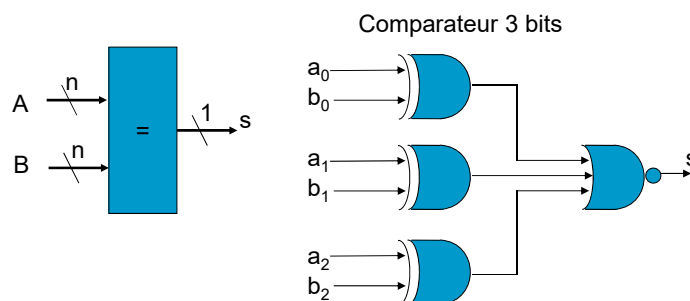


UBO SR

25

Comparateur

- Un multiplexeur est un dispositif combinatoire permettant de « comparer » deux ensembles de signaux. Ces ensembles peuvent représenter des nombres.
- $2 \times n$ entrées, 1 sortie
- $s=1$ si quelque soit $0 \leq i < n$, $a_i=b_i$
sinon $s=0$

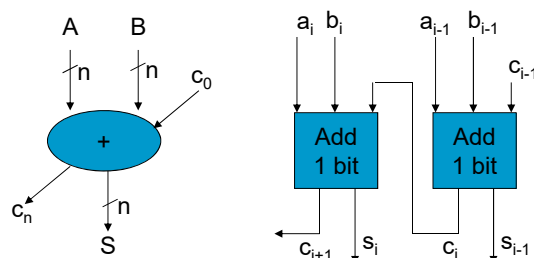


UBO SR

26

Opérateurs arithmétiques et logiques

- Utilisés pour « traiter » les données
- Mise en œuvre de complexité variable selon les opérations à réaliser et les performances souhaitées.
- Le + simple : additionneur à propagation de retenue

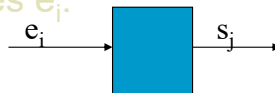


27

Circuits logiques

- Circuit logique combinatoire : l'état des sorties s_j dépend uniquement de l'état courant des entrées e_i .

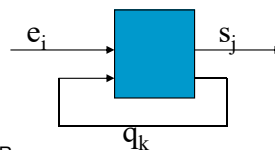
$$s_j = f(\dots, e_i, \dots)$$



- Circuit logique séquentiel : l'état des sorties s_j dépend de l'état courant des entrées e_i et de l'état (\dots, q_k, \dots) de la machine

$$s_j = f(\dots, e_i, \dots, q_k, \dots)$$

$$q_k = g(\dots, e_i, \dots, q_k, \dots)$$

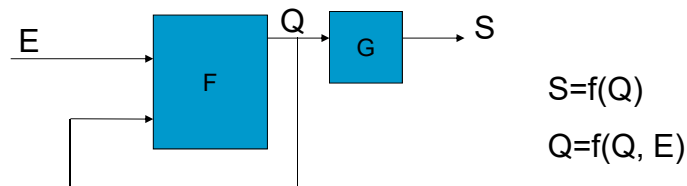


UBO SR

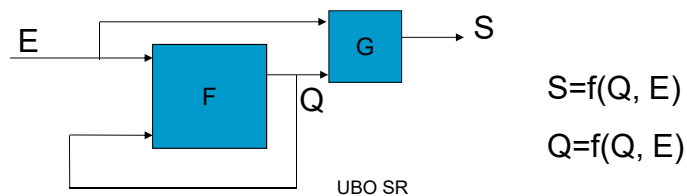
28

Machines particulières

■ Machines de Moore



■ Machines de Mealy



29

Mode de fonctionnement

■ Mode fondamental ou asynchrone

- Toutes les entrées ont un statut identique, les modifications des variables d'entrées se répercutent à tout moment sur les sorties.

■ Mode impulsionnel ou synchrone

- Le circuit est contrôlé par des entrées d'horloge. Les variables d'entrées sont considérées lorsque l'un des signaux d'horloge est actif. L'état interne de la machine change normalement une seule fois dans chaque période d'activité de l'horloge.

UBO SR

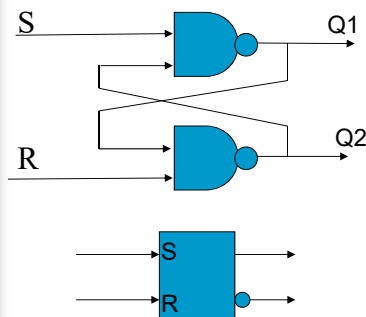
30

Bascule RS

- Une bascule RS (Reset-Set) est un circuit séquentiel asynchrone.

$$Q1 = S \cdot Q2$$

$$Q2 = R \cdot Q1$$



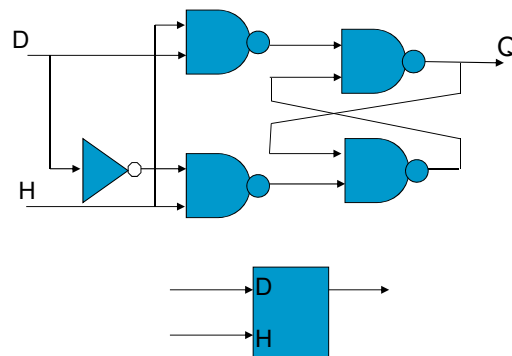
R	S	Q1-	Q2-	Q1	Q2
1	1	0	1	0	1
1	1	1	0	1	0
0	1	X	X	0	1
1	0	X	X	1	0
0	0	X	X	1	1

UBO SR

31

Bascule D (verrou)

- Une bascule D (verrou ou latch) est un circuit séquentiel synchrone.
- La sortie Q recopie l'entrée D lorsque le signal d'horloge H est actif.



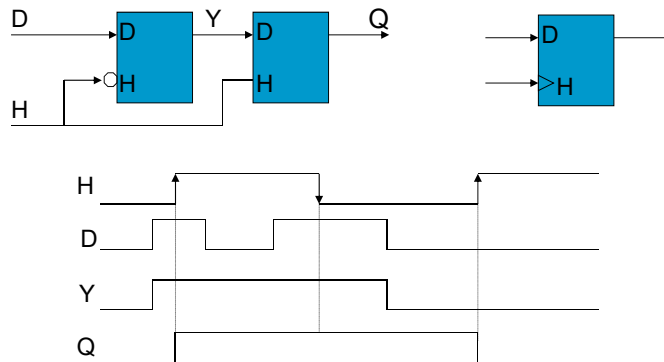
D	H	Q
0	0	Q-
1	0	Q-
0	1	0
1	1	1

UBO SR

32

Bascule D (flip-flop)

- Une bascule D (flip-flop) est un circuit séquentiel synchrone.
- La sortie Q recopie l'entrée D lorsque le signal d'horloge H passe de l'état 0 à l'état 1 (front montant).
- Structure maître-esclave

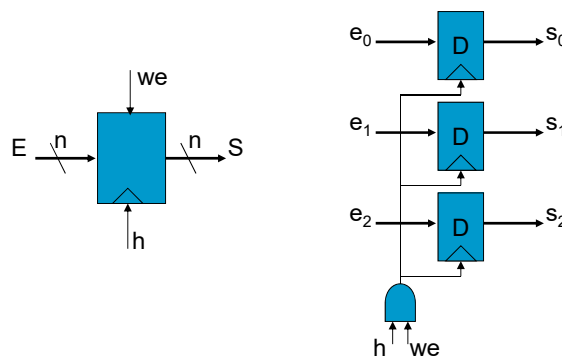


UBO SR

33

Registres

- N bascules D sont associées pour enregistrer un mot.
- Une entrée « write enable », optionnelle, permet de valider ou non le signal d'horloge

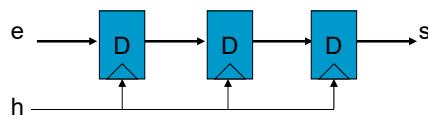


UBO SR

34

Registres à décalage

- n bascules D sont banchées en « série ».
- T : période d'horloge
Si e change uniquement au front d'horloge, on a:
 $s(t) = e(t - nT)$
- Le registre à décalage implante une fonction retard.



UBO SR

35

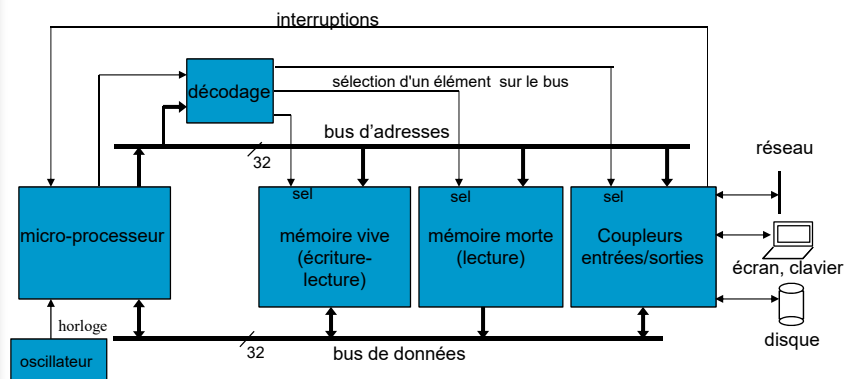
Plan

- Représentation des nombres
- Circuits logiques
 - Combinatoires
 - Séquentiels
- Architecture d'un microprocesseur
 - Structure générale
 - Accès mémoire
 - Jeu d'instructions
 - Interruptions

UBO SR

36

Architecture générale d'un ordinateur

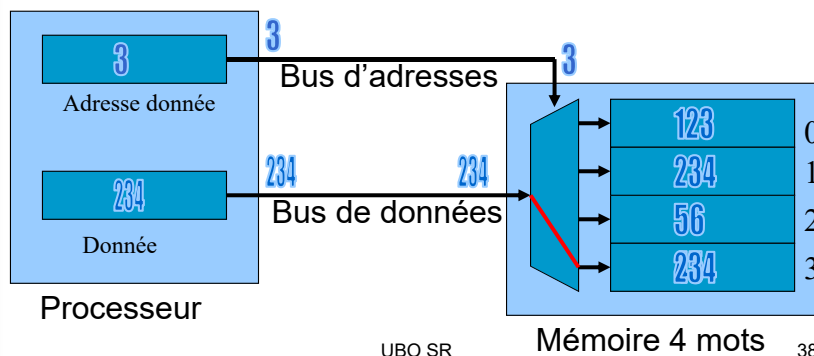
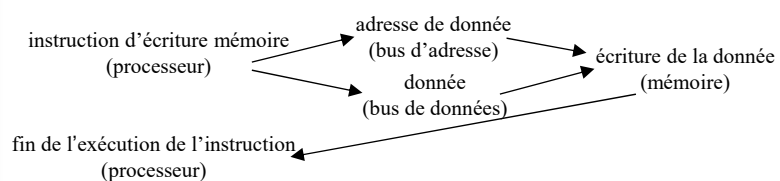


UBO SR

37

Cycle d'accès mémoire

Écriture d'une donnée



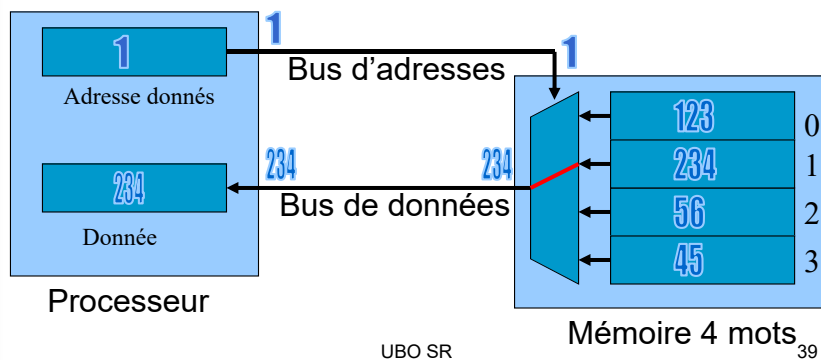
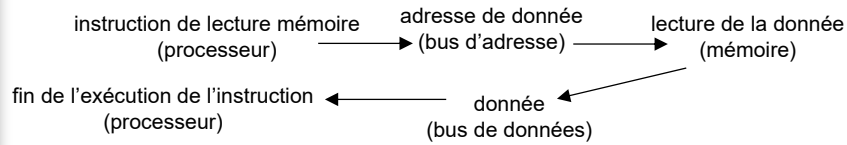
UBO SR

Mémoire 4 mots

38

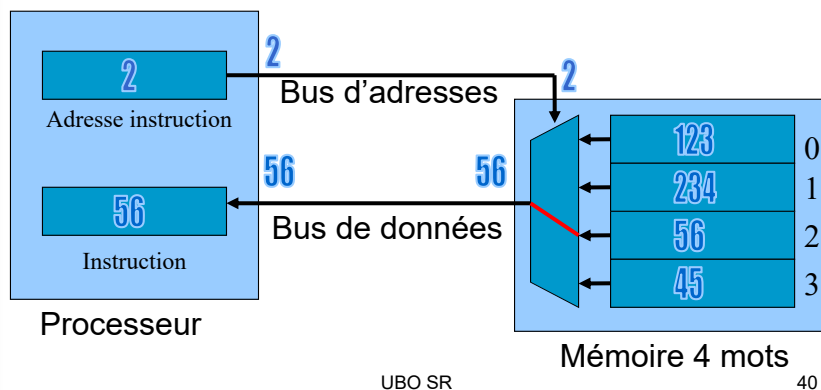
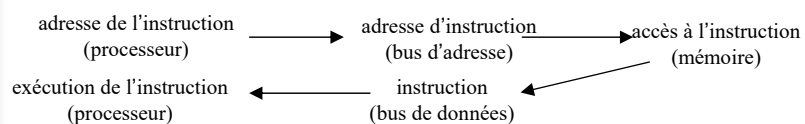
Cycle d'accès mémoire

Lecture d'une donnée



Cycle d'accès mémoire

Lecture d'une instruction



Organisation de la mémoire

- La mémoire des ordinateurs est généralement organisée et adressée par octets.
- Lorsque le processeur accède à un mot mémoire (sur 32 bits par exemple), plusieurs octets sont échangés avec la mémoire.
- L'ordre d'assemblage des octets au sein du mot mémoire peut varier d'une architecture à l'autre.

UBO SR

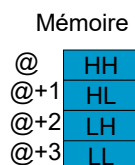
41

Organisation de la mémoire

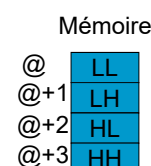
Mot de 32 bits



■ **Big-Endian**
(gros boutiste)



■ **Little-Endian**
(petit boutiste)



UBO SR

Exemple :
305419896 = 0x12345678

Stockage à partir
de l'adresse 1000

1000	18	0x12
1001	52	0x34
1002	86	0x56
1003	120	0x78

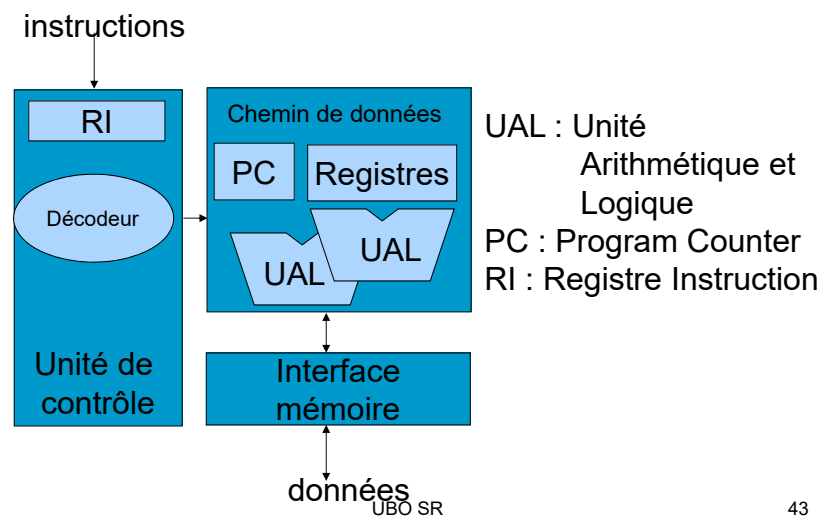
idem, en hexa

Stockage à partir
de l'adresse 1000

1000	120	0x78
1001	86	0x56
1002	52	0x34
1003	18	0x12

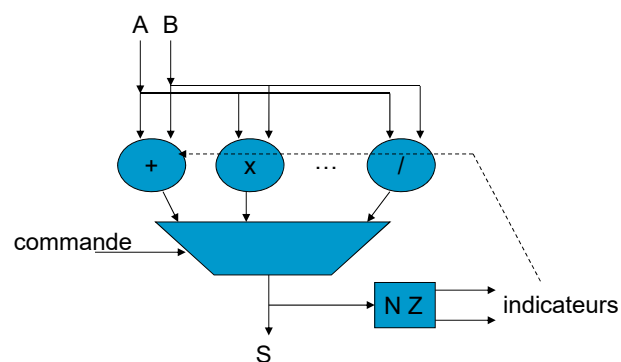
idem, en hexa

Blocs fonctionnels d'un processeur



43

Structure interne d'une UAL



UBO SR

44

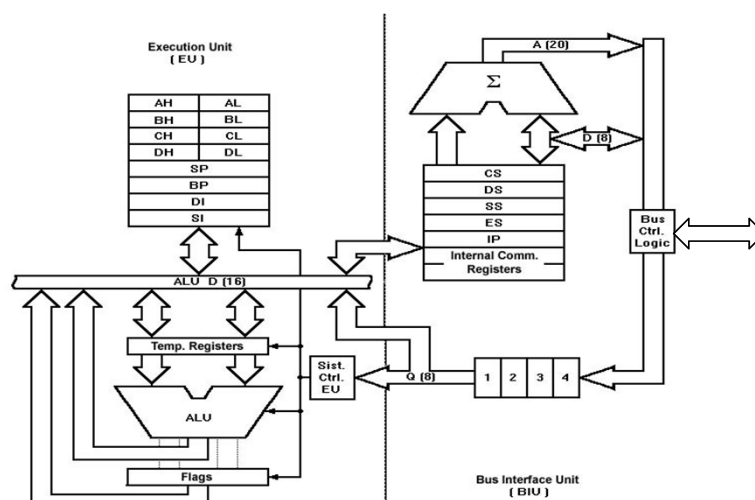
Exemple : Microprocesseur Intel 8086

- 2 unités fonctionnelles :
 - l'unité interface bus (Bus Interface Unit, BPU), chargée du calcul des adresses mémoires des données à transférer avec l'unité d'exécution ;
 - l'unité d'exécution (Execution Unit, EU), qui reçoit les instructions et les opérandes (données) et exécute les instructions.

UBO SR

45

Microprocesseur Intel 8086



UBO SR

Juin 1978

46



Les registres internes

- Registres généraux (entre 10 et 100)
 - Pour les données du programme
 - Accès plus rapide que la mémoire
- Registres d'adressages
 - Utilisés pour le stockage/calcul des adresses mémoires (index, segment)
- Registres spéciaux
 - Compteur de programme (PC ou IP)
 - Pointeur de pile
 - Registre de statut
 - Registre d'instruction (RI)

UBO SR

47



Rôle de l'unité de contrôle

- Contrôle principalement le chemin de données en fonction de l'instruction à exécuter (« courante »)
- L'instruction est codée sous la forme d'une valeur numérique (le code de l'instruction)
- Le décodeur instruction interprète le code de l'instruction, et produit des signaux de contrôle qui provoquent le traitement correspondant dans le chemin de données

UBO SR

48



Jeu d'instructions

- Les opérations qu'un processeur « sait » faire constituent son **jeu d'instructions**.
- Le jeu d'instructions est défini dans un document appelé ISA (Instruction Set Architecture)
 - Opérations et modes d'adressage
 - Codage numérique de ces opérations (format de codage)
- Deux processeurs différents peuvent implanter le même jeu d'instructions (processeurs compatibles, capable d'exécuter les mêmes binaires).

UBO SR

49



Catégorie d'instructions

- Opérations arithmétiques
- Transfert Registre-Registre, Registre-mémoire
- Branchements, conditionnel ou pas, appel de sous-programmes
- Interruptions logicielles
- Échanges co-processeurs
- Contrôle du processeur

UBO SR

50

Codage des instructions

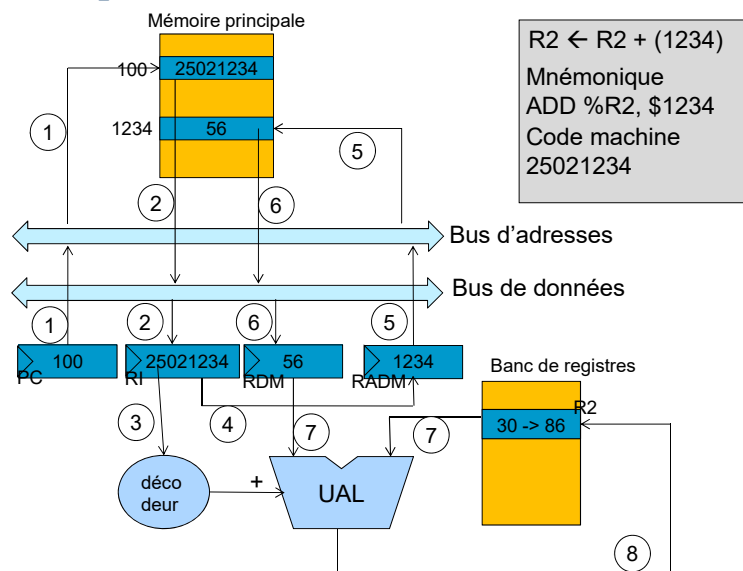
3	3	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0									
Cond	0	0	I																											
Cond	0	0	0	0	0	0	A	S				Rd																		
Cond	0	0	0	0	1	U	A	S				RdHi	RdLo																	
Cond	0	0	0	1	0	B	0	0				Rn																		
Cond	0	0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1							
Cond	0	0	0	P	U	0	W	L				Rn																		
Cond	0	0	0	P	U	1	W	L				Rn																		
Cond	0	0	0	P	U	1	W	L				Rn																		
Cond	0	1	I	P	U	B	W	L				Rn																		
Cond	0	1	1																											
Cond	1	0	0	P	U	S	W	L				Rn																		
Cond	1	0	1	L																										
Cond	1	1	0	P	U	N	W	L				Rn																		
Cond	1	1	1	0																										
Cond	1	1	1	0	CP	Opc						CRn																		
Cond	1	1	1	0	CP	Opc						CRn																		
Cond	1	1	1	0	CP	Opc						CRn																		
Cond	1	1	1	1																										

Source : ARM, ARM7TDMI-S data sheet

UBO SR

51

Exemple : exécution d'une instruction



UBO SR

52



Les interruptions

- Une interruption est un événement asynchrone produit par un dispositif matériel associé au microprocesseur. Les dispositifs d'entrées /sorties utilisent très souvent ce procédé pour avertir le processeur d'une situation à prendre en compte.
- La réception d'une interruption provoque l'exécution immédiate d'un sous-programme chargé de répondre à la situation marquée par l'interruption.
- Après la fin du sous-programme, le processeur reprend l'exécution de son programme à l'endroit exact où il a été interrompu.

UBO SR

53



Les interruptions (suite)

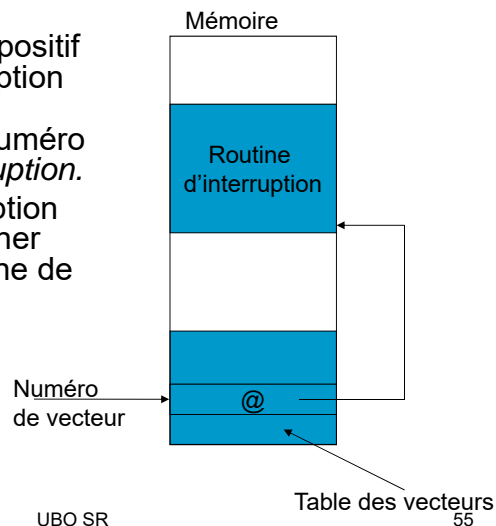
- Un niveau de priorité est généralement associé aux interruptions (de 0 à 7 par exemple).
- Le processeur peut ignorer certaines interruptions, ou certains niveaux de priorité. D'autres ne peuvent en aucun cas être ignorée (le *reset* par exemple).
- Le traitement d'une interruption de niveau *i* ne peut pas être interrompu par une autre interruption de niveau inférieur à *i*.

UBO SR

54

Vecteurs d'interruptions

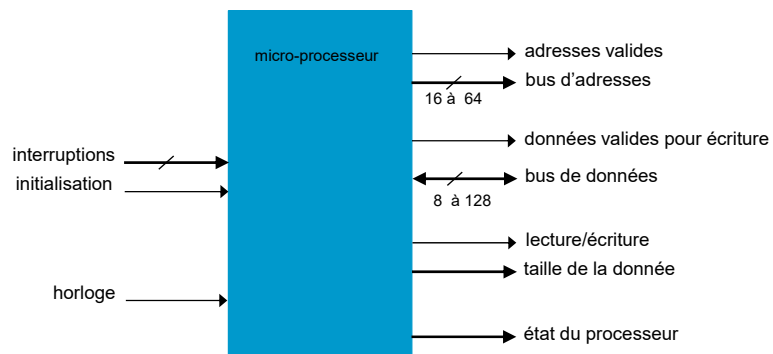
- Dans certaines implantations, le dispositif d'origine de l'interruption doit produire simultanément un numéro de *vecteurs d'interruption*.
- Le vecteur d'interruption permet de sélectionner l'adresse de la routine de traitement qui sera exécutée.



Les exceptions ou déroutements

- Une routine de traitement d'exception est un sous-programme qui est appelé automatique lorsqu'une condition particulière est détectée au cours de l'exécution du programme.
- Une division par 0, un code opération inconnu ou un accès mémoire protégé produisent par exemple des exceptions.
- Les processeurs disposent aussi d'instructions qui produisent volontairement des exceptions (*trap*). Ce mécanisme peut être utilisé pour implanter des appels systèmes sûrs.

Signaux externes d'un microprocesseur



Le boîtier d'un micro-processeur peut comporter jusqu'à plusieurs centaines de broches !

UBO SR

57