Cours d'Architecture des Ordinateurs

Didier Teifreto

Laboratoire d'Informatique de Besançon

teifreto@univ-fcomte.fr

http://lifc.univ-fcomte.fr/PEOPLE/teifreto/Teifreto.html

mardi 22 août 2000

But du cours d'A.S.I.

- Comprendre la structure matérielle et logicielle d'un microprocesseur moderne):
  - structure interne
  - structure externe
- Comprendre la structure matérielle et logicielle d'un ordinateur (PC)
- Apprendre à programmer en assembleur 8086
- 3 x 18h d'apprentissage.

Plan du cours

1 - Historique
2- Principe de fonctionnement
3- Evaluation des performances
4- Le jeu d'instructions
5- Codage des nombres
6- Le matériel de l 'ordinateur
7- Pipeline et Prédiction de Branchement
8- Hiérarchie Mémoire
9- Les Entrées / Sorties
10- Architectures parallèles
11- x86 et PC

Bibliographie - Web

- Organisation et conception des ordinateurs : L 'interface matériel/logiciel 1994 DUNOD A. Patterson et J.L. Hennessy http://www.mkp.com
- Architecture des Ordinateurs : Une approche quantitative 1996 ITP JL Hennessy et A. Patterson http://www.mkp.com/
- http://lifc.univ-fcomte.fr/PEOPLE/teifreto/Teifreto.html.
  - cours, TD et TP
  - Liens utilisés pour l'architecture
    - D'autres cours d'architectures
    - La programmation en Assembleur
    - Le PC

1| Historique www

La Préhistorique -3000 à 1833

-3000 : Chine, l'octogone à trigramme

-500 : Moyen Orient, l'abaque et le boulier.

1642 : Pascal, met au point la Pascaline (+,-)

1666 : Moreland, la multiplication par additions successives.

1833 : Babbage, imagine la machine à différences puis une machine analytique (UC, mémoire, registres, cartes perforées) jamais terminée

Historique 1840 -1944

1840 : Ada Lovelace, principe itérations successives : algorithme en honneur de Al Khowarizmi (820).

1854 : Boole, Algèbre de Boole

1858 : Le premier cable transatlantique (2ème en 1866)

1904 : John Fleming, Diode (tube à vide) Lee DeForest, Triode 1907

1937 : Alan M. Turing, Machine de Turing

1938 : Thèse de Shannon, BInary digiT (// architecture et Booléen)

1940 - 1945 : Calculateurs mécaniques et à relais électrique

Les premiers ordinateurs

1945 : John Von Neumann définit l'architecture de Von Neumann toujours utilisée de nos jours.

1946 : ENIAC (première génération) 30 Tonnes / 72 m^2/140 K watts/ 18000 tubes.350 multiplications /seconde 5000 additions /seconde. programmation par fils.

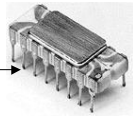
1947 : Invention du Transistor 1Kw /3500transistors/ (interrupteur commandé électroniquement)

1950 : Langage Assembleur et Compilateur

1956 : premier ordinateur à transistors (deuxième génération) 83000 */s . Notion de système d'exploitation

Circuits intégrés

1957 : Langage FORTRAN/LISP



1958 : Circuit intégré par Texas Instruments

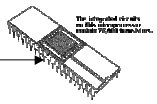
1965 : Loi de Moore . Complexité des circuits x2 / 2 ans

1968 : Ordinateurs à Circuits intégrés (troisième génération) et langage pascal. Notion de Multiprogrammation de temps partagé, de temps réel.

1971 : Intel lance le premier microprocesseur (60000 instructions /s 4004) et la mémoire intégrée , VLSI

Micro ordinateur

1973 : Premier ordinateur à microprocesseur (quatrième génération) Altair, ...



1978 : Intel lance le 8086 330 000 instruction /seconde

1981 : multitude de micro-ordinateurs Apple, IBM PC,

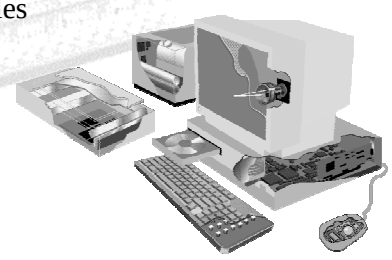
1982 : Intel lance le 80286 900 000 instruction /seconde

1985 : Intel lance le 80386 900 000 instruction /seconde

1986 : Premières machines parallèles

Depuis 1986 : Toujours quatrième générations VLSI

Traitement distribué, réseaux, machines virtuelles



Capacité d'intégration

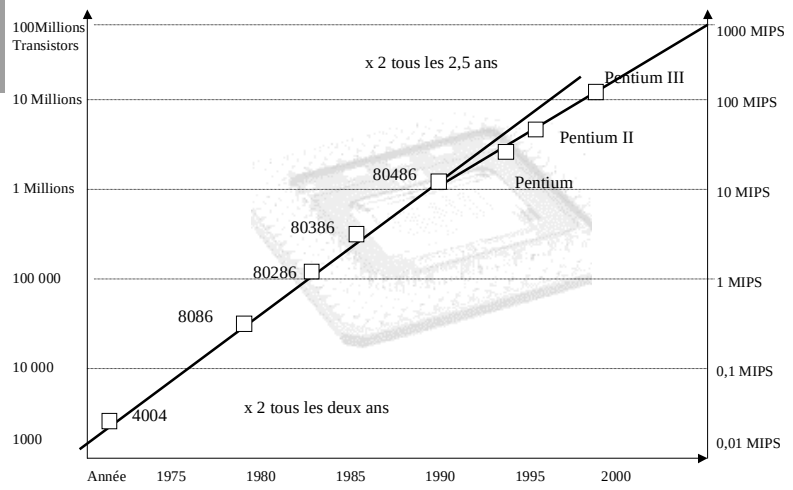


Schéma fonctionnel du 8086

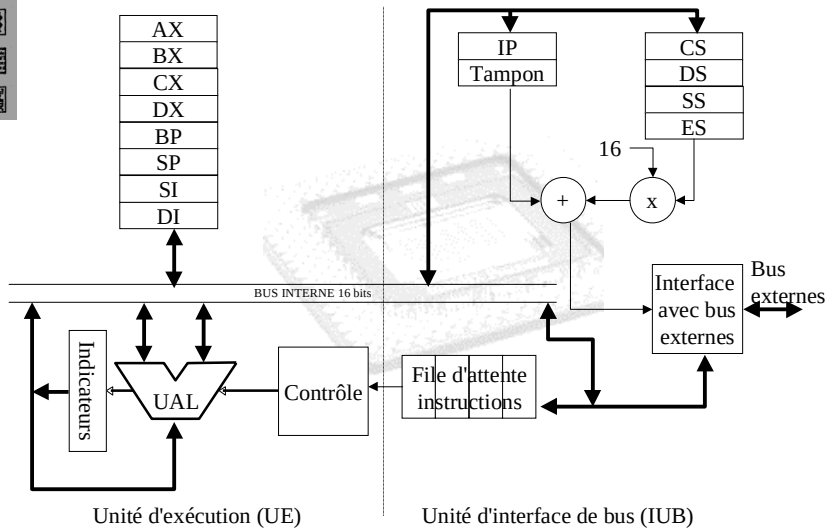
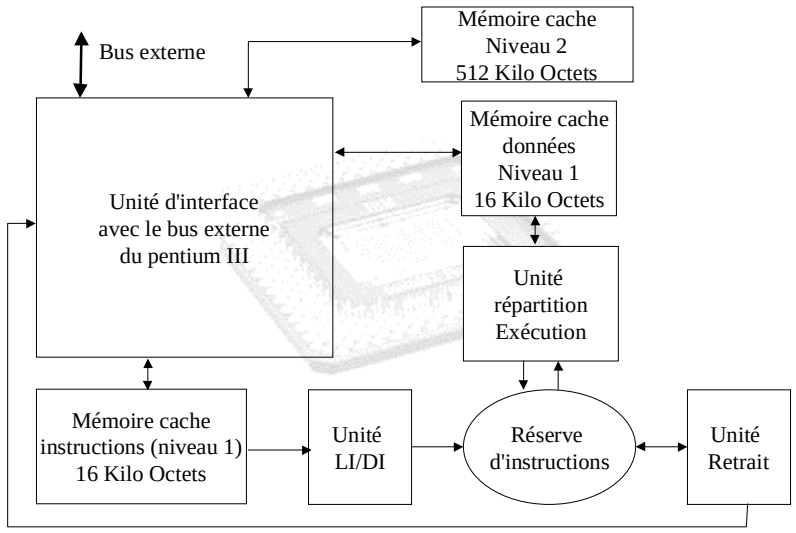
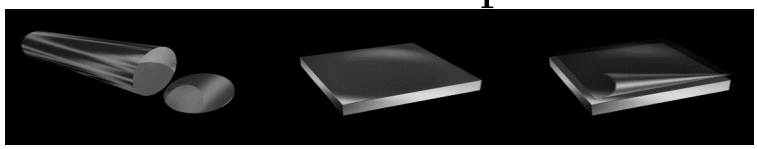


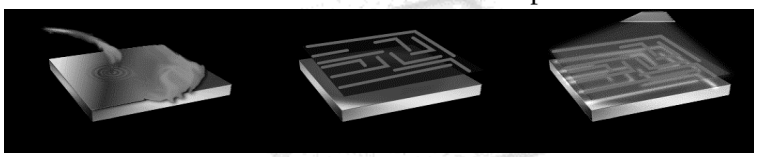
Schéma fonctionnel Pentium



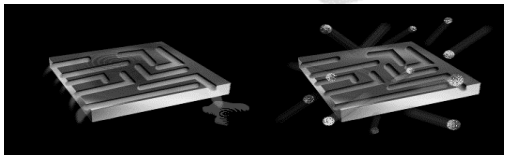
Fabrication du microprocesseur



Barre de silicium Wafer Dépôt couche isolante



Dépôt couche photosensible Masquage Insolation



Photogravure et
Dopage des
zones dégagées =
Semi-conducteur

2| Principe de fonctionnement

Machine de Turing

Alan Turing : Mathématicien(1912- 1954)
Machine permettant de calculer le résultat de fonctions décidables.

La machine contient

- Une bande
  - de longueur infinie
  - séparée en cases contenant les symboles 0 et 1
  - se déplaçant à gauche ou à droite
- Une unité de commande
  - Lire une case sur la bande
  - En fonction de l'état interne, et de la valeur lue
    écrire une valeur dans la même case
  - En fonction de l'état interne, déplacer la bande
    d'une case (gauche ou droite) puis changer d'état

Application

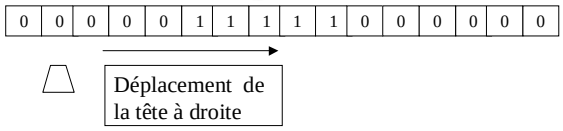
- **Syntaxe de la commande :**

état initial, symbole lu→ état suivant, symbole à écrire,COMMANDE

avec COMMANDE G : Gauche, D : Droite, S : Stop

- Exemple 2,0 →3,1,D

Si la machine est dans l'état 2, si le symbole lu sur la bande vaut 0, alors écrire 1 sur la bande, passer dans l'état 3 et déplacer la bande à droite.



Exemple : calcul e0 ET e1

2 données : si (e0=1 et e1=1) alors S=1 sinon S=0

- Etat 0
  - si Sj=1 alors aller dans l'état 1 - DROITE
  - si Sj=0 alors aller dans l'état 3 - DROITE
- Etat 1
  - si Sj=1 alors aller dans l'état 2 - DROITE
  - si Sj=0 alors aller dans l'état 4 - DROITE
- Etat 2 : ∀Sj alors écrire 1 et STOP
- Etat 3 : ∀Sj alors écrire Sj - DROITE
- Etat 4 : ∀Sj alors écrire 0 et STOP

0,1 → 1,1,D
0,0 → 3,0,D

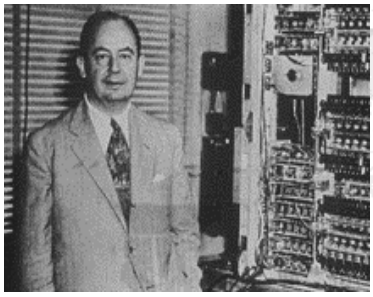
1,1 → 2,1,D
1,0 → 4,0,D

2,0 → 2,1,S
2,1 → 2,1,S

3,0 → 4,0,D
3,1 → 4,1,D

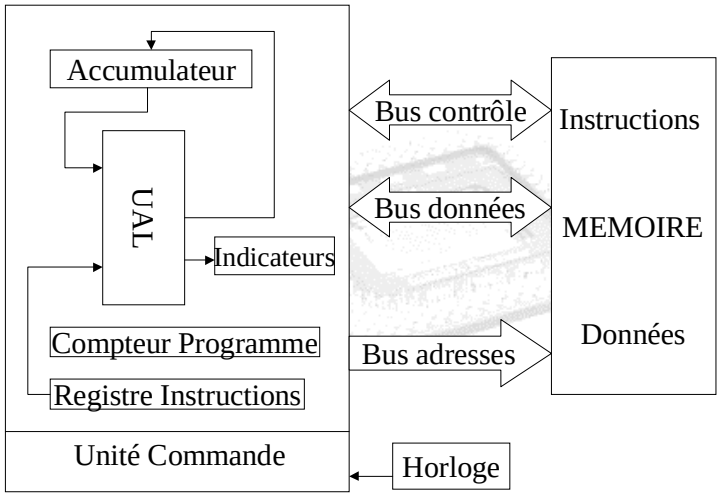
4,0 → 4,0,S
4,1 → 4,0,S

Von Neumann



- 1903-1957
- Mathématicien
- concept de programme enregistré
- Machine de Von Neumann
- Goulot d'étranglement

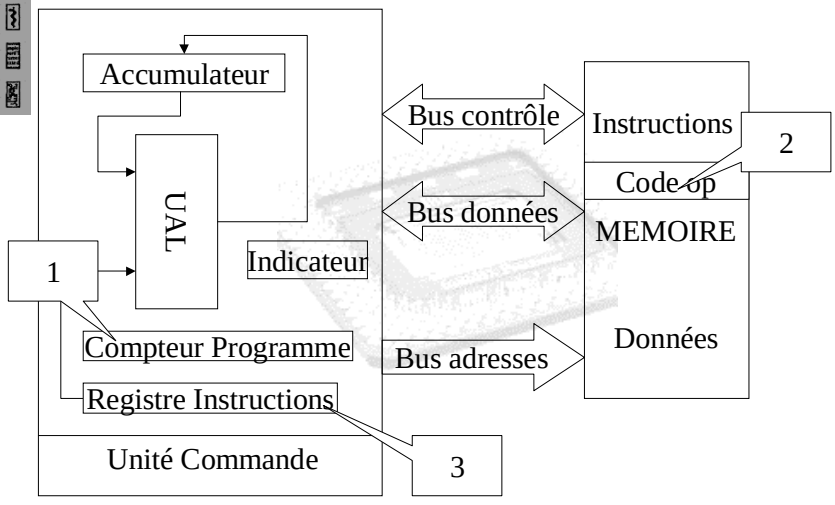
Schéma fonctionnel



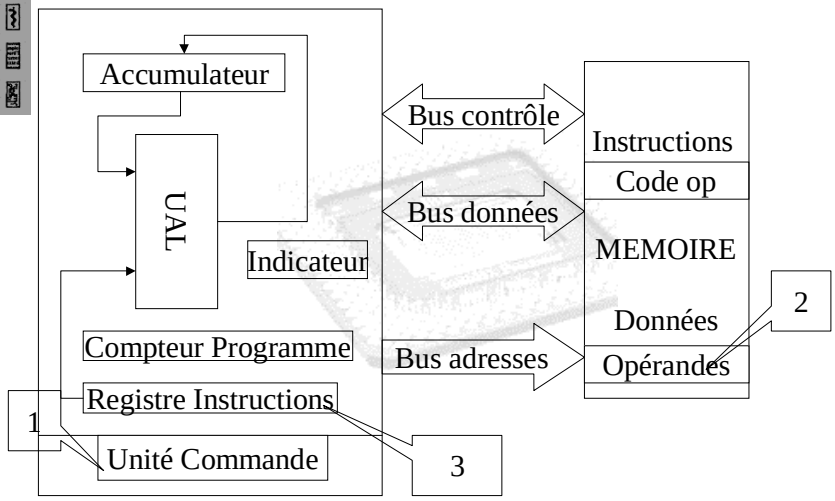
Exemple

- Programme enregistré en mémoire
  - instructions A ← 4 + 3
    - code opération du ← 45 et du ← + 54
    - opérandes 4 et 3
  - données
    - données en attente (mémoire)
    - données en cours de traitement accumulateur (ou registre)
- Bus : ensemble de connections
  - adresse mémoire
  - donnée
  - contrôle
- Accumulateur et Registre = mémoire locale

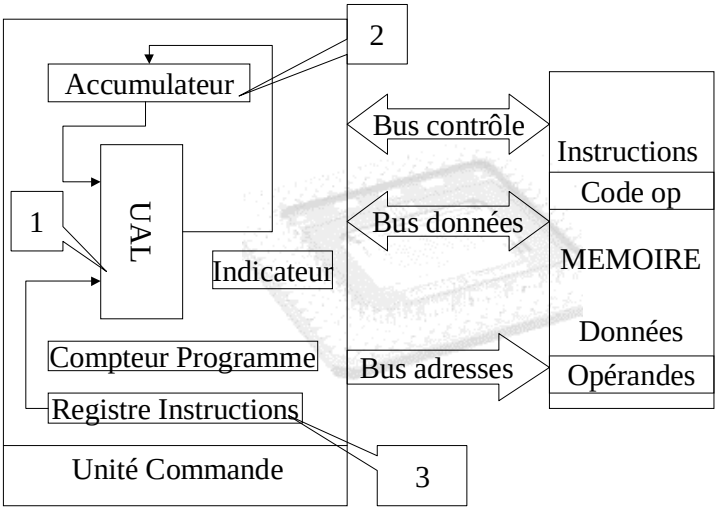
Phase 1 Lecture instruction



Phase 2 Décodage - Données



Phase 3 Exécution



Exemple d ’opération

- A ← 4 + 3
  - A ← 4
  - A ← A + 3
  - Initialement CP=100 RI=0 A=0

Mémoire	Adresse
45	100
4	101
54	102
3	103

Déroulement du programme

- CP =100 RI=0 A=0
- Lecture du code instruction et décodage ←
  - CP =101 RI=45 A=0
- Décodage, lecture de l’opérande 4, exécution
  - CP =102 RI=4 A=4
- Lecture du code instruction ← +
  - CP =103 RI=54 A=4
- Décodage, lecture de l’opérande 3 et addition
  - CP =104 RI=3 A=7

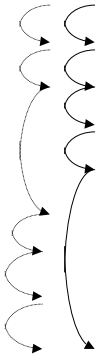
Instructions de la machine

	Instructions arithmétiques
+ ,-	+ 50 A ← A + contenu mémoire 50
	Instructions de transfert
->	-> 50 Contenu de la mémoire 50 ← A
=>	=> 50 idem précédente et A← 0
	Instructions de branchement
J	J 1010 L'instruction suivante est en 1010
JGT	JGT 1010 Aller à l'adresse 1010 si Accumulateur est positif
JLE	JLE 1010 Aller à l'adresse 1010 si Accumulateur est positif ≤ 0

Exemple de calcul

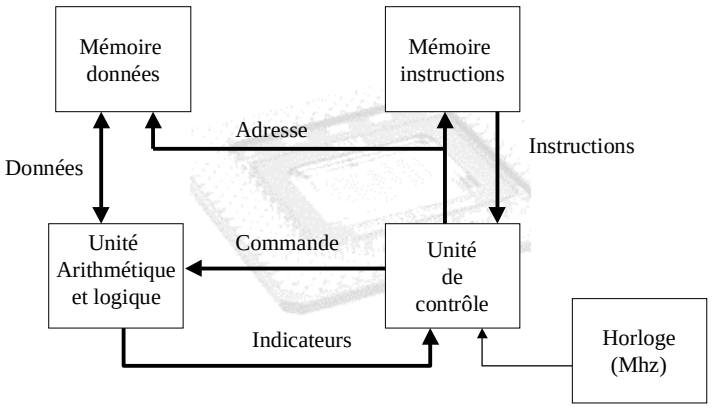
```
.START 1000        ; Début du programme 1000h
10          11     ; En mémoire 10 il y a 11
20          12     ; Mémoire 20 ← 12

1000        =>     20    ; Mémoire 20 ← A et A=0
1001        +      10    ; A ←0 + Contenu mémoire 10
1002        ->     20    ; Mémoire 20 ← A
1003        H      0
;La mémoire 20 contient maintenant la valeur 11
```

Calcul de la valeur absolue

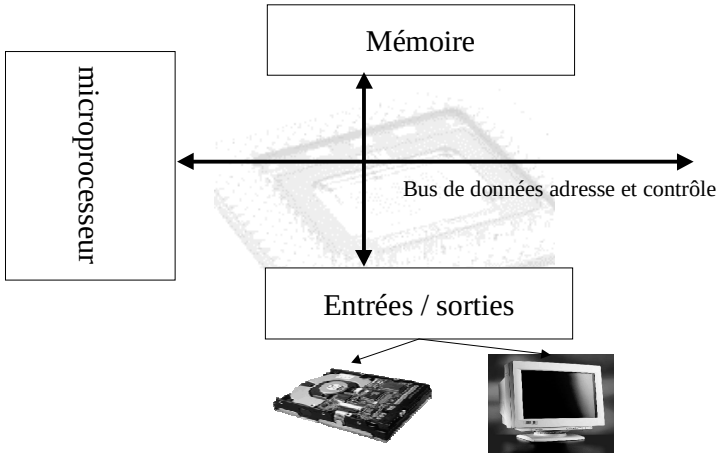


```
.START 100
50      10      ;(cas -10 en pointillés)
51      20      ; résultat
100     =>      51      ; A ← 0
101     +       50      ; A ← mémoire 50
102     JLE     105
103     ->      51      ; Résultat positif
104     J       108
105     =>      51
106     -       50      ; A ← 0 - (-10)
107     ->      51      ; résultat positif
108     H       0
```


1947 - AIKEN - HARVARD



Système complet



3| Evaluation des performances

Unités de mesure

$$MIPS = \frac{NI}{Temps\ exécution \times 10^6}$$

$$MIPS = \frac{Fréquence\ horloge \uparrow}{\downarrow CPI \times 10^6}$$

Dépend du programme de test - Des programmes standards permettent la comparaison

$$MFLOP = \frac{NI\ Réels}{Temps\ exécution \times 10^6}$$

La loi d'Amdalh

$$Accélération = \frac{Temps\ exécution\ sans\ amélioration}{Temps\ exécution\ avec\ amélioration} > 1$$

$$Fraction_{améliorée} = \frac{Temps\ exécution\ partie\ pouvant\ être\ améliorée}{Temps\ exécution\ totale\ sans\ amélioration} \leq 1$$

$$Accélération_{améliorée} = \frac{Temps\ exécution\ partie\ pouvant\ être\ améliorée}{Temps\ exécution\ partie\ avec\ amélioration} \geq 1$$

$$Accélération = \frac{1}{(1-Fa) + \frac{Fa}{Aa}}$$

Equation des performances

Temps exécution UC = Nombre cycle UC x *Temps de cycle*

$$CPI = \frac{\text{Nombre cycle UC}}{NI}$$

Temps exécution UC = NI x *CPI* x *Temps de cycle*

- *Temps de cycle : 1/ Fréquence horloge* ↑
- *CPI : Jeu d'instruction* ↓
- *NI : Compilateurs* ↓

$$\text{Temps exécution UC} = \sum_i (NI_i \times CPI_i) \times \text{Temps de cycle}$$

4| Le jeu d'instructions

Classification jeux d'instruction

Les opérandes sont dans

- la pile (opérandes ou sommet de la pile) (1)
- l'accumulateur (une opérande dans accu) (2)
- dans les registres généraux et/ou la mémoire
  - registre, mémoire (quelques registres) (3)
  - chargement-rangement (grand nombre de registres) (4)
- calcul de C = A+B

x86

(1)	(2)	(3)	(4)
PUSH A	LOAD A	MOV AX,A	LOAD R1,A
PUSH B	ADD B	ADD AX,B	LOAD R2,B
ADD	STORE C	MOV C,AX	ADD R3,R2,R1
POP C			STORE C,R3

Registres

GPR (4)

- 32 registres généraux
  - 32 bits
  - r0 = 0
  - notés : r0 à r31
  - r31 = adresse retour
- 32 registres flottants
  - 64 bits

Registres (3)

x86

AX : Accumulateur

BX : Base

CX : Comptage

DX : Accumulateur auxiliaire

SI : Index Source

DI : Index Destination

Opérations UAL

- Nombres d'opérandes
  - Jeux destructifs (C)

x86

a← a+b

  - Jeux d'instruction non destructifs (D)

x86

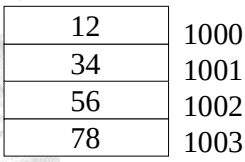
c← a+b

- Machine CISC (année 75)
  - registre / mémoire , quelques registres, destructif
- Machine RISC (année 90)
  - chargement / rangement, beaucoup de registres GPR, non destructif

Interprétation adresses mémoire

x86

- Little endian
  - mot [1000] = 3412h
  - double mot [1000] =78563412h

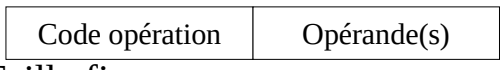


- Big endian
  - mot [1000] = 1234h
  - double mot [1000] =12345678h
- Attention lors du transfert de données entre deux machines différentes

Modes d'adressage du x86

- But : Trouver l 'opérande
  - registre : nom du registre en clair AX
  - immédiate : dans le code (constante) 100
  - direct : on spécifie l'adresse de l'opérande [100]
  - indirect : l 'adresse se trouve dans un registre [SI] [DI]
  - indexé basé : adresse = registre base + registre index + valeur immédiate [BX+SI+10] [BX+DI+10]

Format des instructions



- Taille fixe
  - décodage facile
  - programme plus long
  - temps de chargement d 'une instruction constant

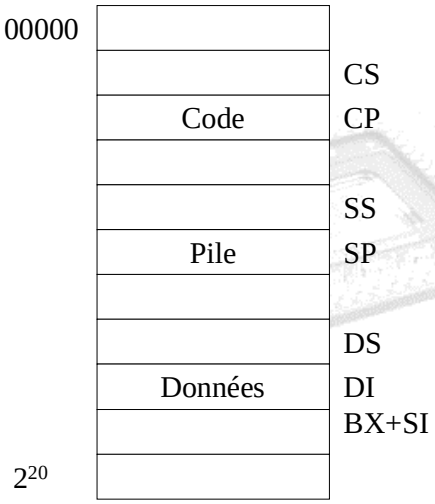
x86

- Taille variable
  - décodage compliqué
  - programme de taille optimale

Instructions x86

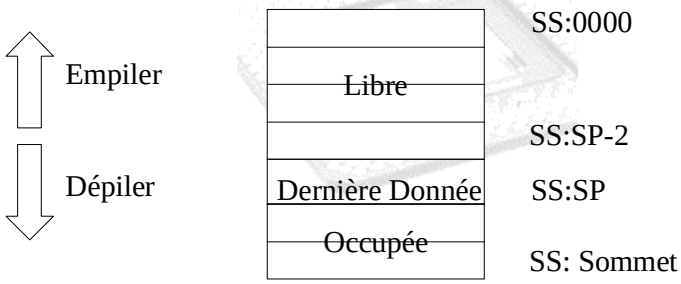
- Transfert
  - MOV Destination,Source Destination ← Source
  - PUSH registre et POP registre
- Arithmétique et logique
  - ADD Destination,Source (sub,and) Destination ← Destination+ Source
  - ADC Destination,Source (subb) Destination ← Destination+ Source+C
  - CMP Destination,Source (test) ← Destination-Source
- Branchement
  - JMP Etiquette (inconditionnel) Aller à l'étiquette
  - J? Etiquette (jc, jz, jnc, jnz…) Si Indicateur Aller à l'étiquette
- Procédure
  - CALL Etiquette Appel sous-programme à l'étiquette
  - RET Retour de sous-programme

Notion de Segment x86



Notion de Pile x86

- Type de la pile : LIFO
- Données codées sur cases mémoires uniquement (16 bits)



Fonction : Empiler et Dépiler

Procédure Empiler(entier V)
début
SP ← SP -2
[SP] ← v
fin

pile pleine SP = 0

fonction Pleine():Entier
debut
Si SP=0 alors Pleine ← 1
sinon Pleine ← 0
fin

fonction Dépiler:Entier
début
SP ← SP+2;
retourner [SP-2];
fin

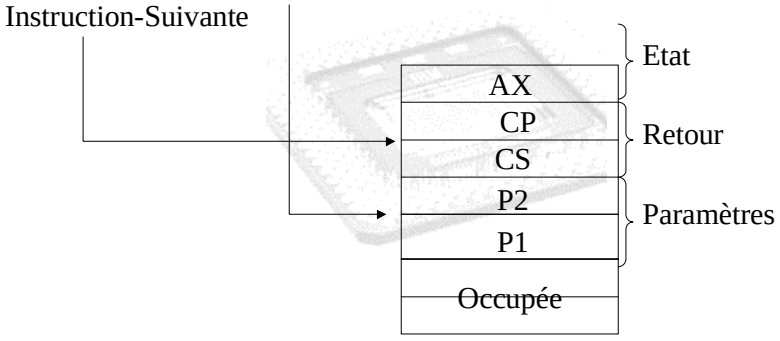
Lorsque la pile est vide SP est au sommet

Les procédures x86

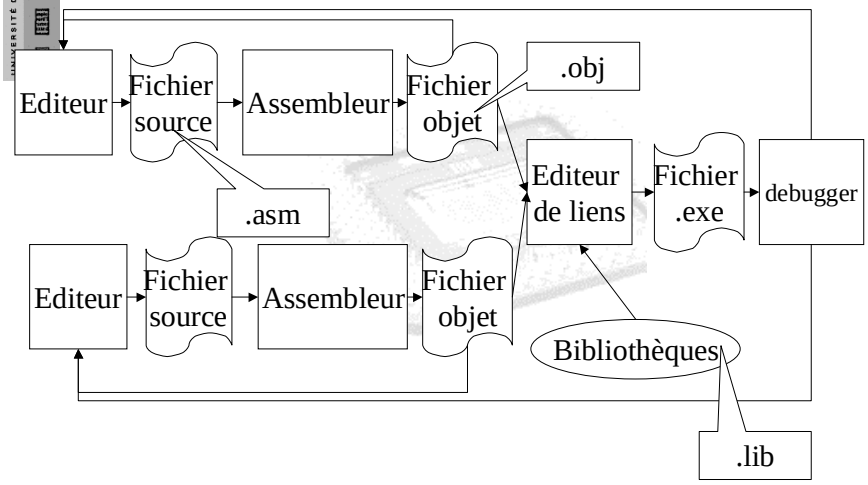
- A l'appel on place dans la pile :
  - Les paramètres (Programme - non obligatoire)
  - L'adresse de retour (instruction suivante - Processeur)
- Exécution du sous-programme (modification de CP et de CS)
- Au début de la procédure
  - Le contexte (Etat du processeur - Programme)
- A la fin de la procédure :
  - Restitution du contexte (Programme)
  - Instruction suivante du programme (Processeur)
  - Suppression des paramètres (Programme - non obligatoire)

Appel des procédures x86

Procédure MaProc(P1 entier,P2 entier)
Instruction-Suivante



Création d'un exécutable



Assembleur x86 - Les bases

Directives

```
.model small
.stack 100h
.data
.code
    mov ax,@data
    mov ds,ax
    ….
    mov ah,4ch
    int 21h
end
```

Instructions

Opérandes

```
.model small
.stack 100h
.data
    n1 DB -10
    n2 DB ?
.code
    mov ax,@data
    mov ds,ax
    mov ax,n1
Si: cmp ax,0
    jbe sinon
alors:
    mov n2,ax
    Jmp FinSi
sinon:
    mov ax,0
    sub ax,n1
    mov n2,ax
FinSi:
    mov ah,4ch
    int 21h
end
```

Etiquettes

5| Codage des nombres

Nombres entiers non signés (≥0)

- En informatique
  - Binaire 0 et 1
  - Hexadécimal (base 16) 0 à 9 et A a F
- Notation positionnelle (conversion base b $\rightarrow$10)
  - $N_{(b)} = d_{n-1}\, d_{n-2} \ldots d_1\, d_0$
  - $N_{(10)} = b^{n-1}d_{n-1} + b^{n-2}d_{n-2} \ldots + b^1d_1 + b^0d_0$

poids rang

  - avec n chiffres $0 \leq N \leq b^n - 1$ soit b^n valeurs

Binaire ⇔ hexadécimal

	Double mot	Mot (word)	Octet (byte)	
Bits	32	16	8	4
Valeur maximale	2^{32}-1	2^{16}-1 65535	2^{8}-1 255	2^{4}-1 15
Digit	8	4	2	1
Valeur Maximale	16^{8}-1 FFFFFFFF	16^{4}-1 FFFF	16^{2}-1 FF	16^{1}-1 F

Conversion base 10 → base b

- divisions successives $254_{(10)} = ?_{(2)}$

254| 2
0 127| 2
1 63| 2
1 31 | 2
1 15 | 2
1 7 | 2
1 3 | 2
1 1

254| 16
14 15

E F

254= 1111 1110 b = FE h

Addition binaire

1001 1101 B = 157
+ 0110 1101 A = 109
11111 1010 retenue
10000 1010 = 266

+	0	1
0	00	01
1	01	10

Résultat sur 9 bits
dépassement non
signé : CARRY

Nombres entiers signés

- format prédéfini
- bit de poids fort a un poids de $-b^{n-1}$
  - $N_{(10)} = -b^{n-1}d_{n-1} + b^{n-2}d_{n-2} \ldots + b^1d_1 + b^0d_0$
  - avec n chiffes $b^{(n-1)} \leq N \leq b^{(n-1)} - 1$
  - b^{n-1} valeurs négatives b^{n-1}-1 valeurs positives et 0
  - $-128 \leq N$ 8 bits ≤ 127
  - $-32768 \leq N \leq 32767$
- Complément à deux 4 0100
  - complément à 1 -8+2+1=-5 1011
  - + 1 -8+4 = -4 1100

soustraction binaire

1001 1101 B = -99
+ 0110 1101 A = 109
11111 1010 retenue
10000 1010 = 10

-	0	1
0	00	01
1	11	00

Juste ?? CARRY =1

Dépassement signé = OVERFLOW

1001 1101 B = -99
- 0110 1101 A = 109
11111 0000 retenue
11101 0000 = -48

résultat FAUX

BCD

- 1 chiffre décimal = 1 groupe de 4 bits
- Sur 8 bits $0 \leq N \leq 99$
- 64_{10}= 0110 0100
- Correction des opérations

0110 0100 64_{10}
+1000 1000 88_{10}
1110 1100 EC_{16}

x86

FAUX instruction de correction décimale DAA

Nombres fractionnaires

- Notation positionnelle (virgule fixe)
  - $N_{(b)} = d_{n-1}\, d_{n-2} \ldots d_1\, d_0\, d_{-1}\, d_{-2} \ldots d_{-(m-1)}\, d_{-m}$
  - $N_{(10)} = b^{n-1}d_{n-1} + b^{n-2}d_{n-2} \ldots + b^1 d_1 + b^0 d_0$
    $+ b^1 d_{-1} + b^{-2}d_{-2} \ldots + b^{-(m-1)}d_{-(m-1)} + b^{-m}d_{-m}$
- Représentation virgule flottante
  - mantisse x b $^{\text{exposant}}$
  - binaire 1,xxxxx . 2 $^{\text{exposant}}$
  - normalisation : 1 non représenté

Conversion base 10 → base b

- multiplications successives

$0{,}58_{10} = ?_{16}$

0,58* 16	= 9,28	**9**
0,28* 16	= 4,48	**4**
0,48* 16	= 7,68	**7**
0,68* 16	= 10,88	**A**
0,88*16	= 14,08	**E**

0,947AE ….

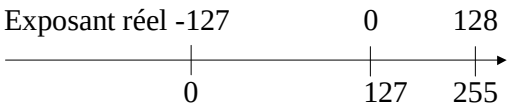
Format IEEE 754

S de M 1bit	Exposant 8 bits	Mantisse 23 bits	FSP

S de M 1bit	Exposant 11 bits	Mantisse 52 bits	FDP

valeur exposant codé = exposant réel + excédent

- *excédent* = Exposant maximum /2
- exposant minimum correspond à *-excédent*
- exposant maximum correspond à *excédent +1*



Caractéristiques IEEE 754

	Simple précision	Double précision
Taille (bits)	32	64
Excédent	+127	+1023
Exposant normalisé	[-126,+127]	[-1022,+1023]
Plus petit nombre	$1{,}0\ 2^{-126} \approx 10^{-39}$	$1{,}0\ 2^{-1022} \approx 10^{-308}$
Plus grand nombre	$1{,}111_2\ 2^{127} \approx 10^{38}$	$1{,}111\ 2^{1023} \approx 10^{308}$
Exposant = 0	Dénormalisé nombre < Plus petit Si mantisse = 0 alors nombre représenté = 0	
Exposant = MAXI (128 ou 1024)	Si mantisse = 0 alors nombre représenté = $+\infty$ Si mantisse $\neq 0$ alors nombre non représentable	

Addition flottante

Addition : $m_1 2^{e1} + m_2 2^{e2} m$

$1{,}101\ 2^0 = 1{,}625$

$+ \quad 1{,}111\ 2^1 = 3{,}75$

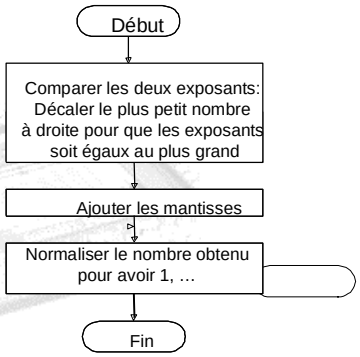
erreur 5,375

$0{,}1101\ 2^1$

$+ \quad 1{,}1110\ 2^1$

$10{,}1011\ 2^1$

Normalisation $1{,}01011\ 2^2$



Nouveaux formats flottants

- Format 80 bits Extension norme IEEE 754

S de M 1bit	Exposant 15 bits	Mantisse 64 bits

- Format deux réels (SIMD)

S1 1bit	E1 7 bits	M1 32 bits	S2 1bit	E2 7 bits	M2 32 bits

6 Le matériel de l'ordinateur

Algèbre de Boole

- Variable à deux états
- Fonctions à n variables S= f(A,B)

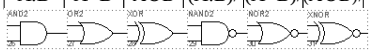
A	non	oui
0	1	0
1	0	1

n=1

Symbole de la porte 

A	B	and	or	⊕	nand	nor	id
0	0	0	0	0	1	1	1
0	1	0	1	1	1	0	0
1	0	0	1	1	1	0	0
1	1	1	1	0	0	0	1
		A.B	A+B	A⊕B	(A.B)/	(A+B)/	(A⊕B)/

n=2

Symbole de la porte 

Théorèmes fondamentaux

Associativité	(A+B)+C = A+(B)+C) (A.B).C = A.(B.C)		
Commutativité	A+B = B+A	A.B = B.A	
Distributivité	A+(B.C) = (A+B) . (A+C) A.(B+C) = (A.B) + (A.C)		
Loi d 'Identité	A+0 = A	A.1=A	A//=A
Loi du 0 et 1	A+1 = 1	A.0=0	
Loi d 'inversion	A+A/ = 1	A.A/=0	

Théorème De Morgan

(A+B+C...)/ = A/.B/.C/. ... (A.B.C...)/ = A/+B/+C/+ .../

Isoler la valeur d'un bit

Valeur	0	0	1	1	0	0	0	1
Masque	0	0	0	0	0	0	1	0
	0	0	0	0	0	0	0	0

Valeur	0	0	1	1	0	0	1	1
Masque	0	0	0	0	0	0	1	0
	0	0	0	0	0	0	1	0

Si (Valeur ET 02) = 2
alors
Le bit 1 est à 1
sinon
Le bit 1 est à 0
FinSI

Si (Valeur ET 02) ≠ 0
alors
Le bit 1 est à 0
sinon
Le bit 1 est à 1
FinSI

Forcer la valeur d'un bit

Valeur	0	0	1	1	0	0	0	1
Masque	0	0	0	0	0	0	1	0
Valeur	0	0	1	1	0	0	1	1

Valeur	0	0	1	1	0	0	1	1
Masque	0	0	0	0	0	0	1	0
Valeur	0	0	1	1	0	0	1	1

Valeur ← Valeur OU 02h

Conception d'un circuit

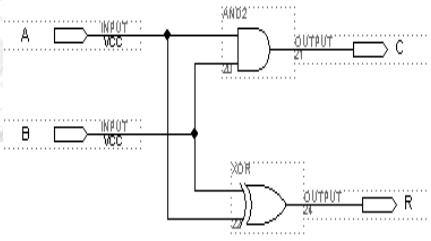
- 1 - La table de vérité
- 2 - L'équation

  (somme de produits)

  C=A.B

  R = AB/+A/B = A ⊕ B
- 3 - Schéma
  - portes ET OU NON

A	B	C	R
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Conception d'un additionneur

1. La table de vérité

2 L'équation

R=A/B/C+A/BC/+AB/C/+ABC

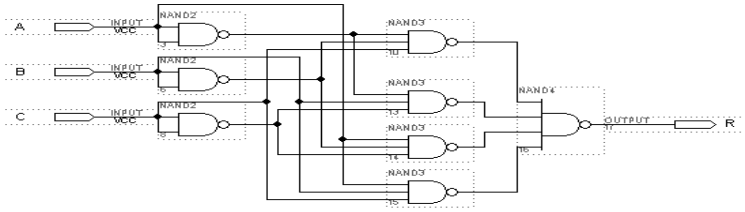
C+=A/BC+AB/C+ABC/+ABC

3. Schéma avec des portes nands (R)

R = (A/B/C+A/BC/+AB/C/+ABC)//

R =((A/B/C)/.(A/BC/)/.(AB/C/)/.(ABC)/)/

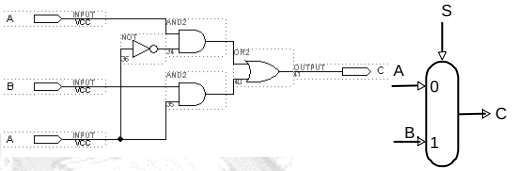
A	B	C	C+	R
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



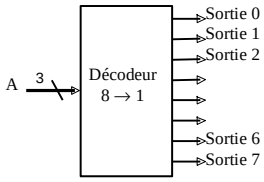
Autres circuits

- Multiplexeur *Si S=0 alors C= A sinon C=B)* C=S/A+ SB

A	B	S	C
0	X	0	0
1	X	0	1
X	0	1	0
X	1	1	1



- Décodeur



Suivant A

cas 0: Sortie 0=1

cas1 : Sortie 1=1

…

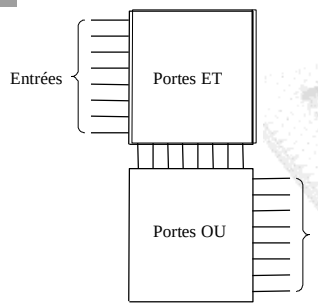
cas 7: Sortie 7=1

Fin Suivant

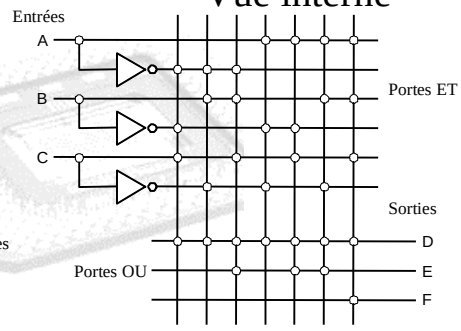
Réseaux logiques programmables

E=A/BC+AB/C+ABC/

Vue externe

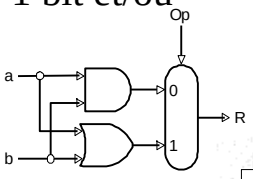


Vue interne



Unité Arithmétique et Logique

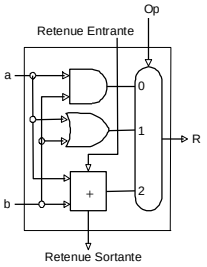
- 1 bit et/ou



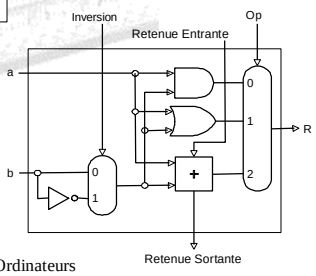
Inversion	op	op2	op1	Fonction
0	0	0	0	a ET b
0	1	0	1	a OU b
0	2	1	0	a + b
1	2	1	0	a - b
1	3	1	1	a<b

-b = b/ +1
a-b=a+b/+1

- 1 bit et/ou/+



et/ou/+/-

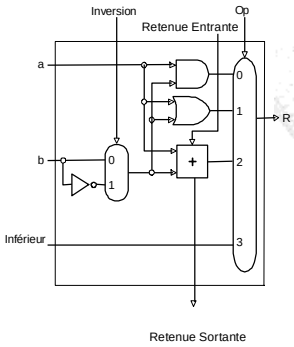


UAL 1 bit complète

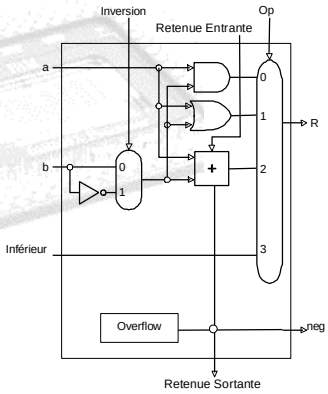
Si a<b alors r=1 sinon r=0

Si a-b<0 alors r=1 sinon r=0 (c'est la valeur du bit de signe)

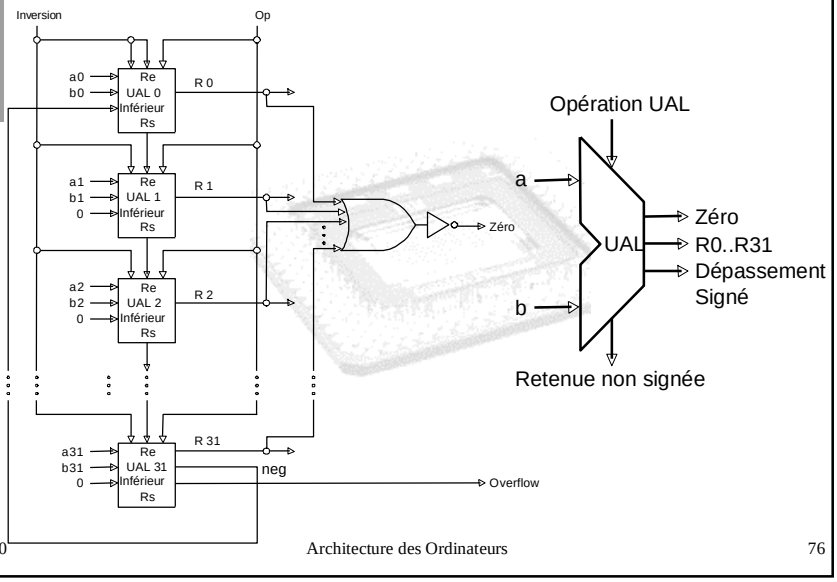
UAL bit 0 à n-2



UAL bit n-1



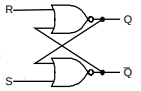
UAL n bits. Vue interne /externe



Circuits séquentiels - Bascules RS

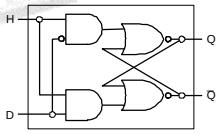
R	S	Q-	Q	
0	0	Q-	Q-	mémorisation
1	0	x	0	Remise à zéro
0	1	x	1	mise à 1
1	1	x	?	indéfini

Bascule RS



Bascule D niveau

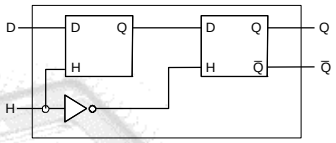
H	D	Q-	Q
0	x	Q-	Q-
1	1	x	1
1	0	x	0



Bascule D et trois états

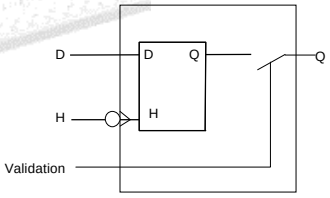
Bascule D front

H	D	Q-	Q
x	x	x	Q-
↓	1	x	1
↓	0	x	0

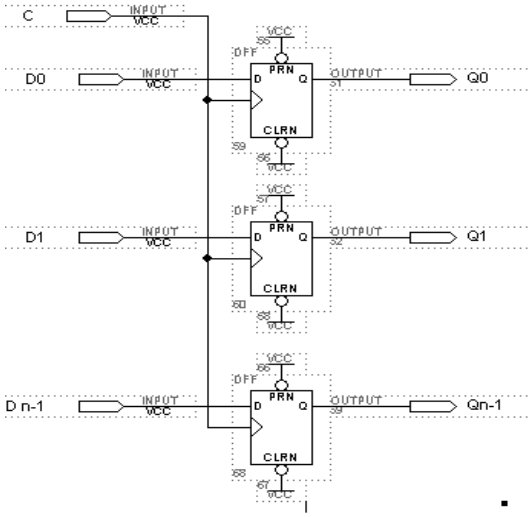


Bascule D front et validation

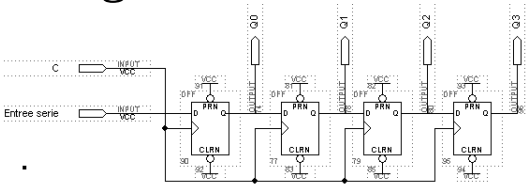
Val	H	D	Q-	Q
1	x	x	x	Q-
1	↓	1	x	1
1	↓	0	x	0
0	x	0	x	HI

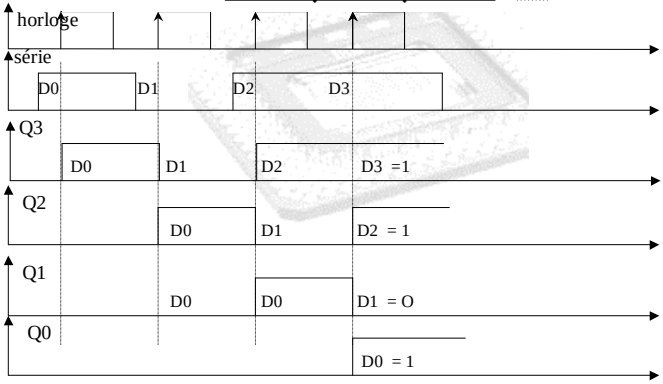


Registre parallèle synchrone

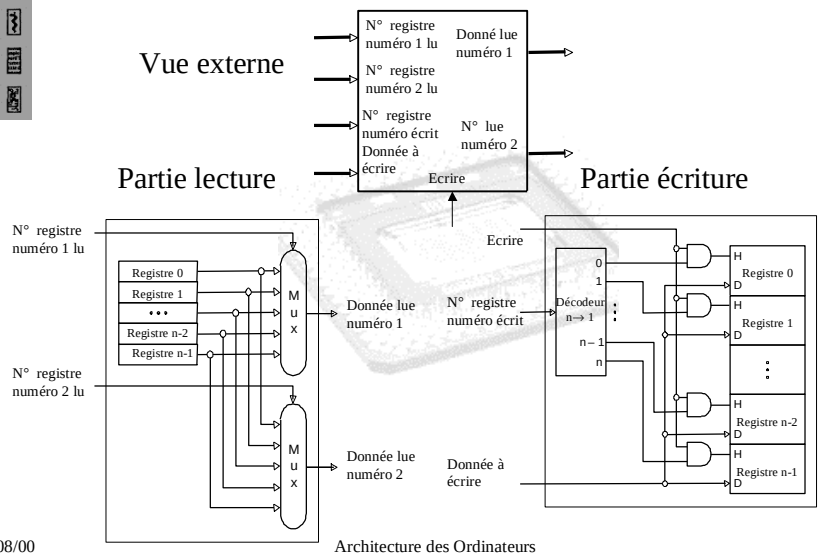


Registre à entrée série

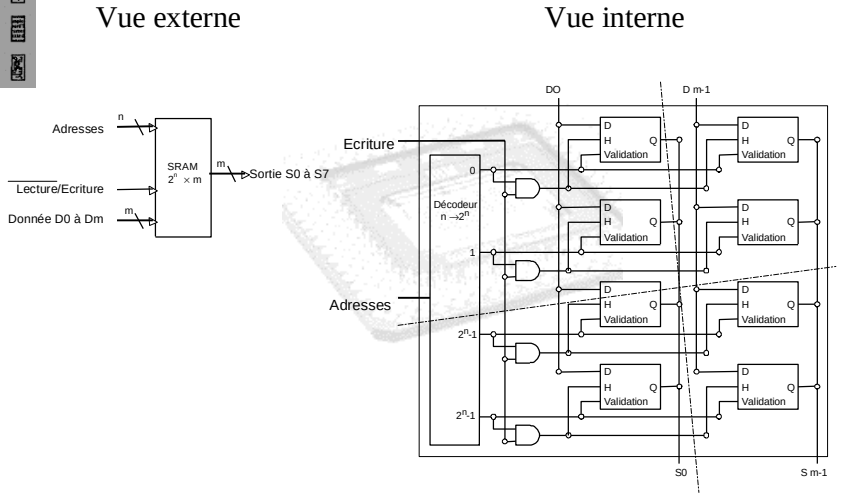




Registre 2 ports lecture / 1 écriture



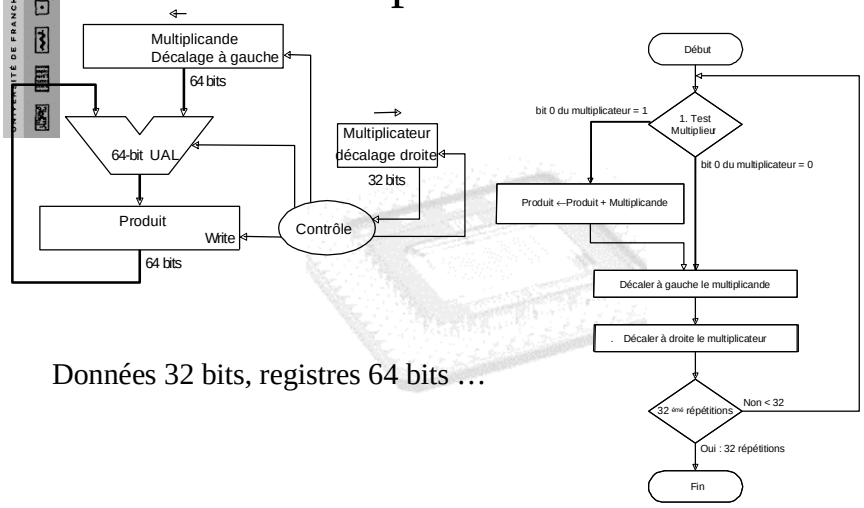
Mémoire SRAM 2^n x m bits



Multiplication principe

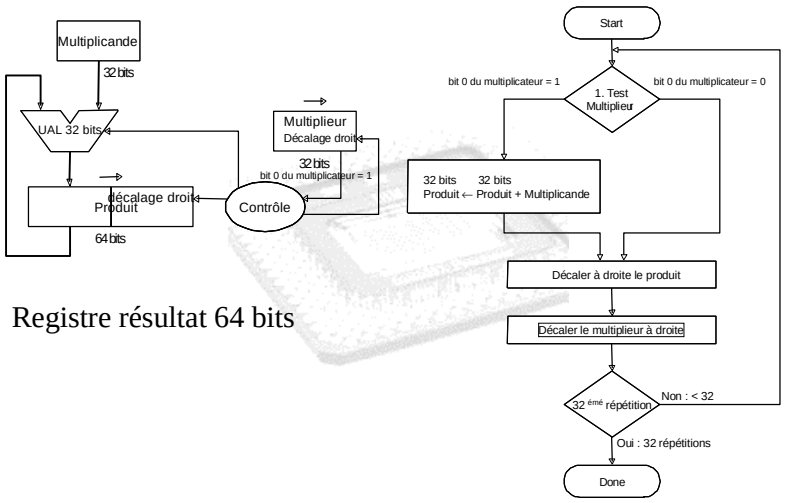
Multiplicande	1 0 0 0	8
Multiplicateur	x 1 0 0 1	x 9
+	1 0 0 0	72
+	0 0 0 0	
+	0 0 0 0	
+	1 0 0 0	
	1 0 0 1 0 0 0	

Multiplication v1



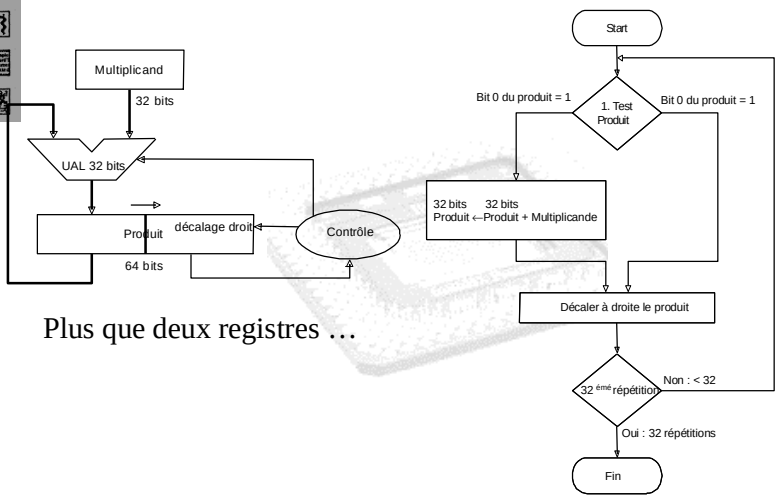
Données 32 bits, registres 64 bits …

Multiplication v2



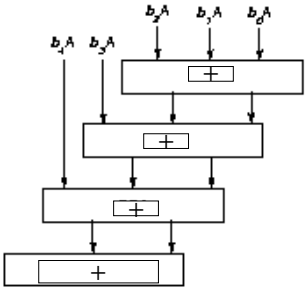
Registre résultat 64 bits

Multiplication finale



Plus que deux registres …

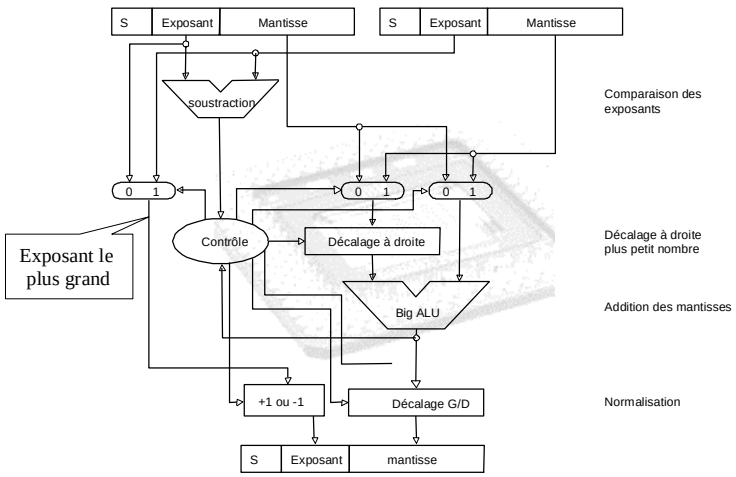
Accélération de la multiplication



Multiplieur en réseau

	1 0 0 0	A
x	1 0 0 1	b3 b2 b1 b0
	1 0 0 0	A et b0
	0 0 0 0	A et b1
	0 0 0 0	A et b2
	1 0 0 0 .	A et b3
	1 0 0 1 0 0 0	

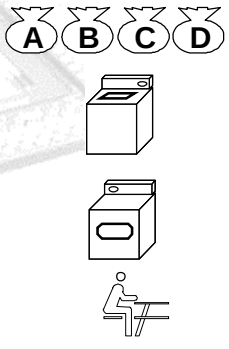
Addition flottante



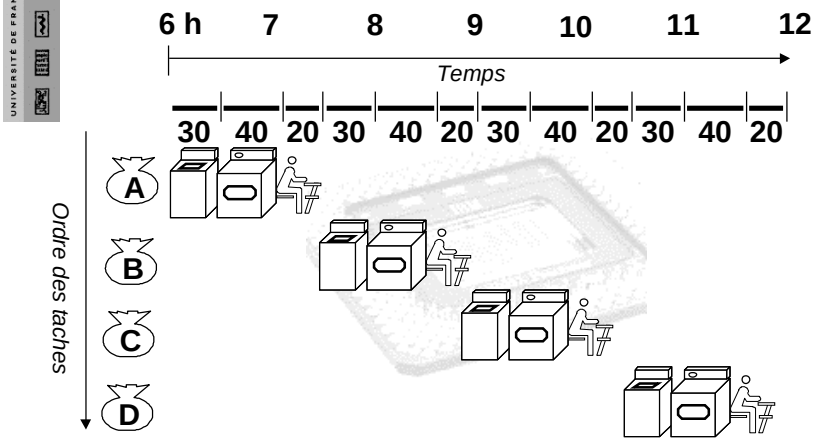
7| Pipeline et prédiction de branchement

Pipeline: La laverie

- Anne, Bernard, Claude, David ont chacun du linge à nettoyer, sécher, repasser
- Laver : 30minutes
- Sécher : 40 minutes
- Repasser 20 minutes
- Total : 90 minutes

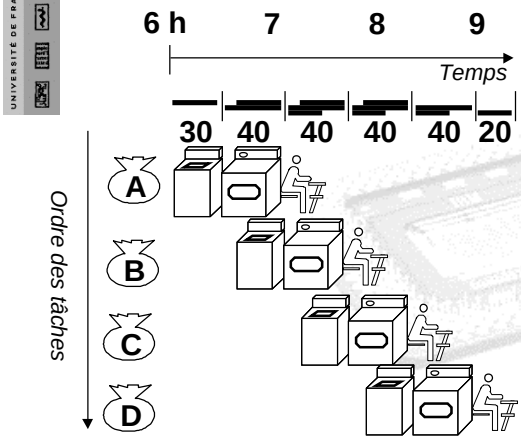


Laverie séquentielle



- Total 90mn x 4 charges = 360 mn

Principe du pipeline



- La latence (temps d'une charge ne change pas)
- La vitesse est limitée par le traitement le plus long
- Plusieurs tâches en même temps
- Augmentation potentielle de la vitesse : 3
- Avec les temps différents gain en vitesse de 90/ 40 =2,25
- Gain réel 360/210= 1,7
- Temps de chargement du linge réduisent la vitesse globale.

Total : 40x4+30+20 = 210 mn

Le Pipeline

- Plusieurs instructions se recouvrent dans leur exécution.
- *débit* $_{(sans\ pipeline)}$ *= Nombre instructions / Temps*
- Etage du pipeline = Etape
- Cycle Machine=Temps dans un étage+Transfert
- => Equilibrer le temps dans chaque étage
- *Débit* $_{avec}$ *=* $\frac{Nombres\ instructions \times Nbs\ Etages}{Temps\ d'exécution}$

Pipeline de base à 5 étages

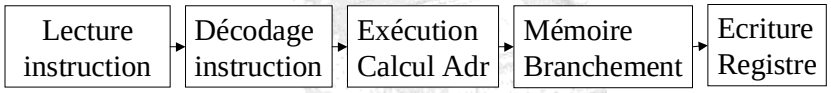
LI : Lecture instruction suivante / calcul CP suivant

DI : Décodage instruction, affectation registre temporaire

EX : Exécution de l'instruction ou calcul d'adresse

M : accès mémoire éventuel ou modification CP éventuelle

ER : mise à jour des registres

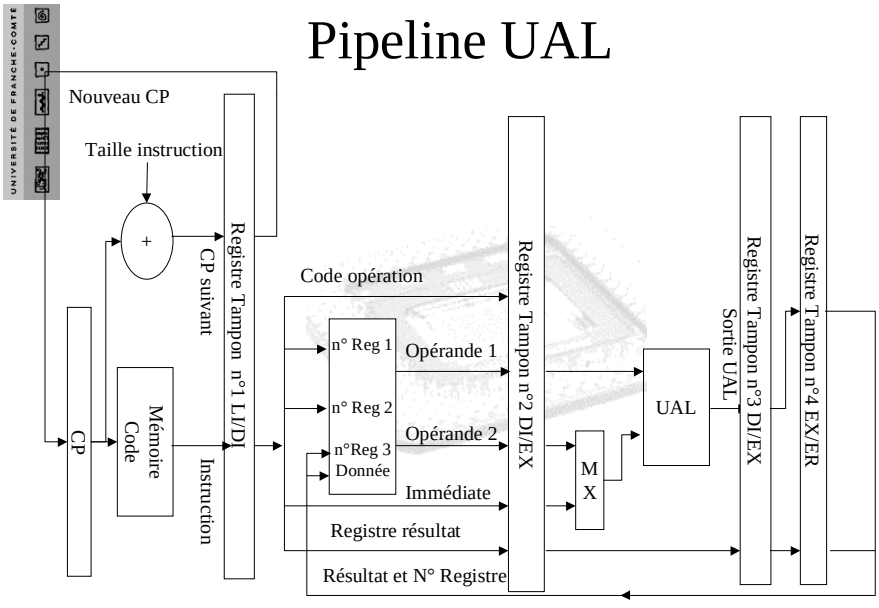


Cycle Horloge	1	2	3	4	5	6	7	8
Instruction i	LI	DI	EX	M	ER			
Instruction i+1		LI	DI	EX	M	ER		
Instruction i+2			LI	DI	EX	M	ER	
Instruction i+3				LI	DI	EX	M	ER

Instructions U.A.L.

- Rd ← Rs1 opération Rs2
- Rd ← Rs1 opération immédiate
  - LI : Lire instruction suivante, calcul de CP suivant
  - DI : détermination Opérande 1, opérande 2, code opération op , numéro registre résultat
  - EX : calcul résultat Opérande 1 op Opérande 2
  - M : Rien
  - ER : Rd ← Résultat

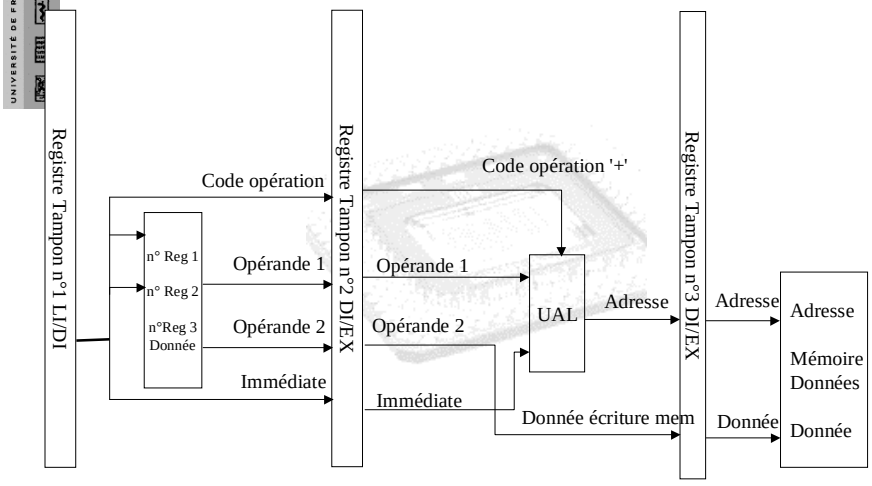
Pipeline UAL



Instruction écriture mémoire

- [Rs1+immédiate] ←Rs2
  - LI : Lire instruction suivante, calcul CP suivant (idem opération UAL)
  - DI : détermination Opérande 1, opérande 2, immédiate, code opération '+'
  - EX : calcul adresse de la donnée
  - M : accès mémoire écriture de la donnée
  - ER : rien

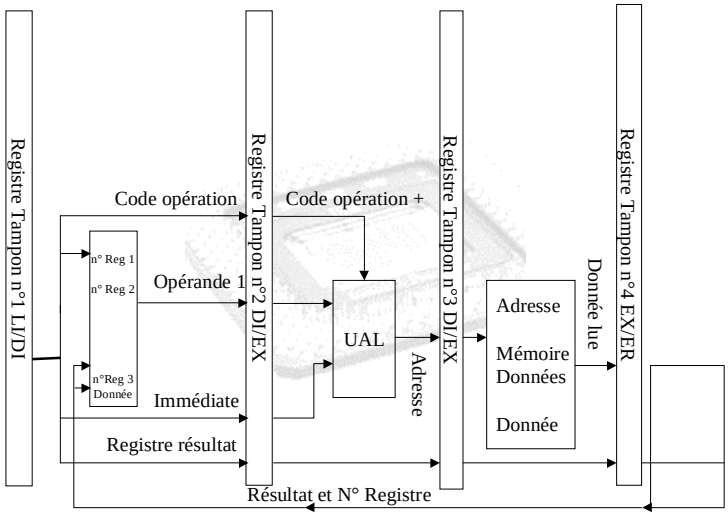
Pipeline Ecriture Mémoire



Instruction chargement mémoire

- Rd ←[Rs1+immédiate]
  - LI : Lire instruction suivante, calcul CP suivant (idem opération UAL)
  - DI : détermination Opérande 1, immédiate, code opération '+'
  - EX :calcul adresse de la donnée
  - M : Accès mémoire en lecture
  - ER : Ecriture dans Rd de la valeur lue en mémoire

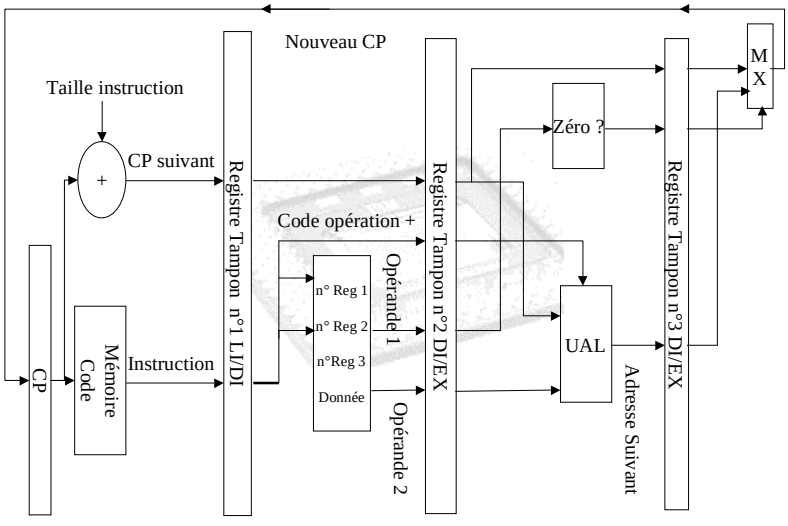
Pipeline Ecriture Mémoire



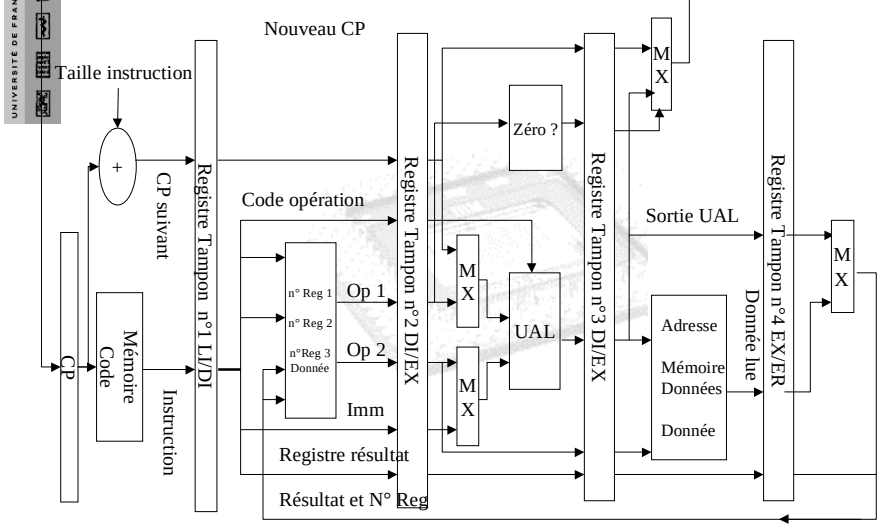
Instruction de branchement

- Si Rs1=0 aller à CP Suivant + Rs2
  - LI : Lire instruction suivante
  - DI : calcul opérande 1, opérande 2, code opération +, immédiate
  - EX : calcul de l'adresse CP Suivant + Rs2, Calcul Rs1=0
  - M : Modifier CP avec CP suivant ou CP Suivant +Rs2
  - ER : rien

Pipeline Branchement



Microprocesseur version 1



Réduction nombre d'étage

- Instructions UAL peuvent éviter le cycle M
- Instructions de Branchement peuvent éviter le cycle ER

Cycle Horloge	1	2	3	4	5	6	7	8
Inst UAL	LI	DI	EX	ER				
Lect Mem		LI	DI	EX	M	ER		
Inst UAL			LI	DI	EX	?	ER	
Inst UAL				LI	DI	EX	?	ER
Inst Branch.					LI	DI	EX	M

•Gain de temps non évident …

Les aléas

– Structurels : conflits de ressources
  • Machine avec un seul banc mémoire (programme + données)
– De données : LAE (lecture après écriture)
  • 3 cycles perdus

Cycle Horloge	1	2	3	4	5	6	7	8
R1←R2+R3	LI	DI	EX	M	ER			
R4 ←R1-R5		LI	DI	**X**	**X**	DI	EX	M
R6 ←R1.R7			LI	**X**	**X**	**X**	DI	EX
R8 ←R1+R9				**X**	**X**	**X**	LI	DI
R10 ←R3-R5					LI	**X**	**X**	LI

Bulle

Les aléas de données

– Opération UAL
  • résolution matérielle : technique d'envoi (EX/M/ER/Registres)
  • 0 cycle perdu

Cycle Horloge	1	2	3	4	5	6	7	8
R1←R2+R3	LI	DI	EX	M	ER			
R4 ←R1-R5		LI	DI	EX	M	ER		
R6 ←R1.R7			LI	DI	EX	M	ER	
R8 ←R1+R9				LI	DI	EX	M	ER
R10 ←R3-R5					LI	DI	EX	M

registres

Les aléas de données

– Opération Mémoire
  - résolution matérielle : technique d ’envoi
  - insuffisant 1 cycle perdu

Cycle Horloge	1	2	3	4	5	6	7	8
R1←[R2+0]	LI	DI	EX	M	ER	registres		
R4 ←R1-R5		LI	DI	**X**	EX	M	ER	
R6 ←R1.R7			LI	**X**	DI	EX	M	ER
R8 ←R1+R9				**X**	LI	DI	EX	M
R10 ←R3-R5						LI	DI	EX

Les aléas de données

- résolution logicielle : Réorganisation du code par le compilateur
- Exécution dans le désordre : gérée par le matériel
- 0 cycle perdu

Cycle Horloge	1	2	3	4	5	6	7	8
R1←[R2+0]	LI	DI	EX	M	ER	registres		
R10←R3 - R5		LI	DI	EX	M	ER		
R4 ←R1-R5			LI	DI	EX	M	ER	
R6 ←R1.R7				LI	DI	EX	M	ER

Les aléas de branchement

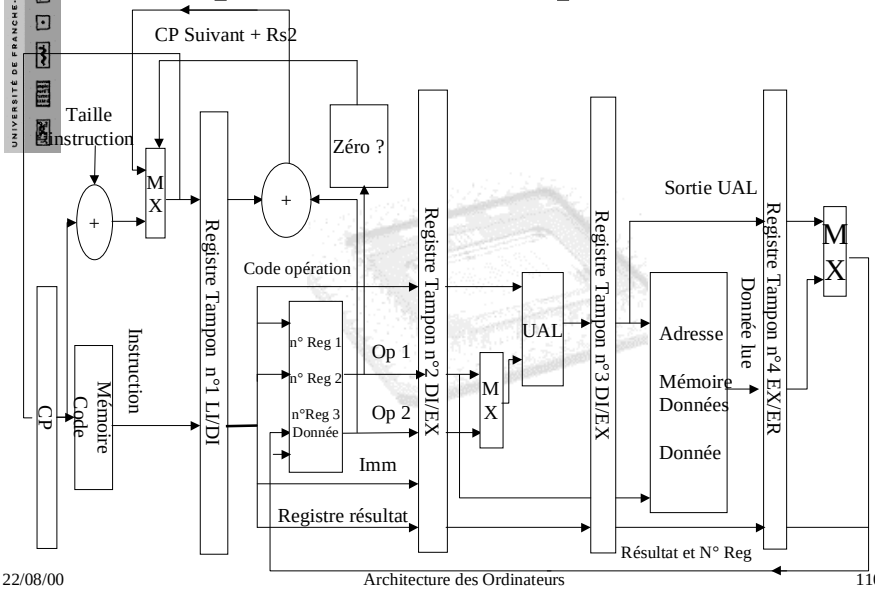
- Blocage pipeline durant deux ou trois cycles perdus

Cycle Horloge	1	2	3	4	5	6	7	8
Si R2=0 aller a	LI	DI	EX	M	ER			
Instruction i+1		LI	?	?	DI	EX	M	ER
Inst suivante			LI	LI	LI	DI	EX	M

- Calcul anticipé de la condition (DI) 0 OU 1 cycle perdu

Cycle Horloge	1	2	3	4	5	6	7	8
Si R2=0 aller a	LI	DI	EX	M	ER			
Instruction i+1		LI	DI	EX	M	ER		
Inst suivante	no	LI	LI	DI	EX	M	ER	

Microprocesseur complet version2



Prédiction de branchement

i1:R1←100
i2: R1←R1-1
i3: R2 ← R2+1
i4:Si R1 <> 0 aller à i2
i5:

99 erreurs : 99 cycles perdus

Cycle Horloge	1	2	3	4	5	6	7	8
i1 *R1←100*	LI	DI	EX	M	ER	registres		
i2 *R1←R1-1*		LI	DI	EX	M	ER		
i3*R2 ←R2+1*			LI	DI	EX	M	ER	
i4 *Si R1 <> 0*				LI	DI	EX	M	ER
i5					LI			
i2 *R1←R1-1*						LI	DI	EX

Prédiction de branchement

– Principe de localité : Même code effectué souvent (ici 99% du temps)
  • En conservant l'état antérieur des branchements
    – « PRIS » ou « NON PRIS »
  • Variable booléenne.

Cycle Horloge	1	2	3	4	5	6	7	8
i4 *Si R1 <> 0*	LI	DI	EX	M	ER			
i5 *R1←R1-1*		LI						
i2*R1← R1-1*			LI	DI	EX	M	ER	
...				LI	DI	EX	M	ER
i4 *Si R1 <> 0*					LI	DI	EX	M
i2 *R1←R1-1*						LI	DI	EX

Non Pris

Pris

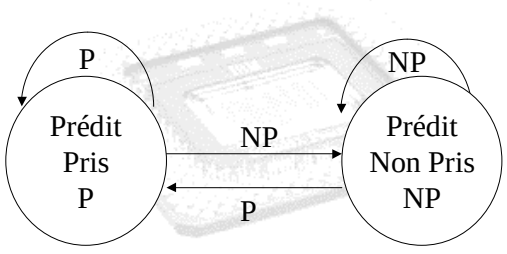
Prédiction de branchement

- BTB (Branch Target Buffer)
  - mémoire interne au processeur contenant un ensemble de lignes (état prédiction 1 bit)

Adresse Instruction	Adresse Cible	Etat Prédiction	
i4	?	Non Pris	R1=100
i4	i2	Pris	R1=99
			R1=0
i4	i2	Non Pris	R1=-1

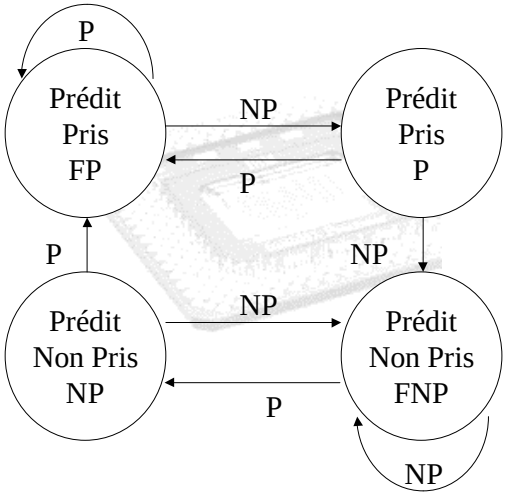
Taux prédiction correcte = $\frac{98}{100}$

Algorithme de prédiction de branchement 1 bit



1 bit : 2 états P/NP

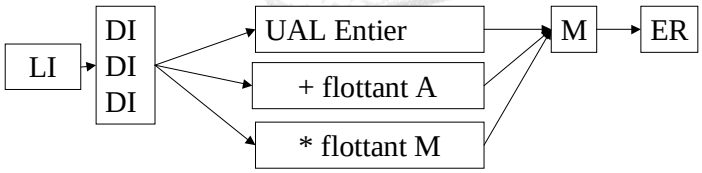
Algorithme de prédiction de branchement 2 bits



2 bits:
4 états
FNP
NP
P
FP

Opérations multicycles

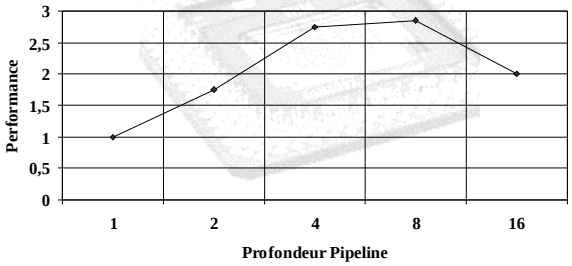
- Opérations flottantes (plus d ’un cycle)
- Trois unités d’exécution en parallèle (superscalaire ordre 3)
  - UAL nombre entier : 1 étage
  - Multiplieur flottant et entier : 3 étages
  - Addition flottante : 3 étages (3 cycles entre deux divisions)



Pièges

L ’augmentation de la profondeur du pipeline accroît toujours les performances

- impossible d ’équilibrer les temps dans les étages
- le temps de mémorisation est constant



Equations des performances

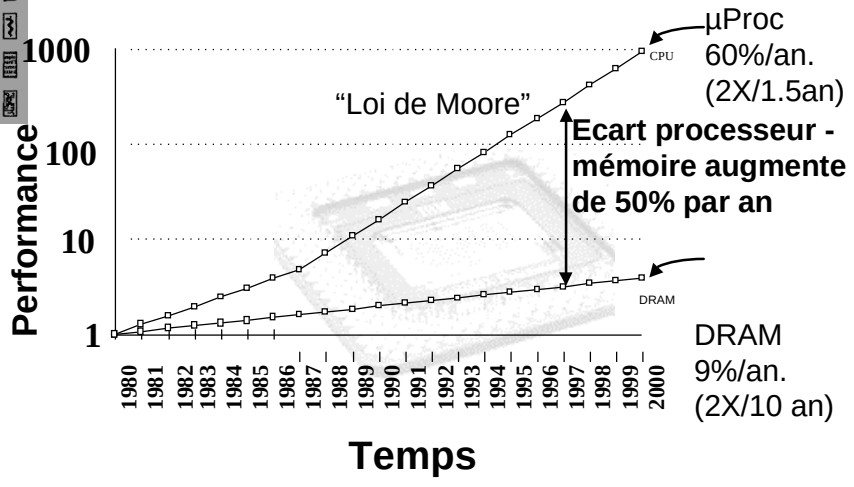
Accélération = CPI non pipeliné x Temps cycle non pipeliné
CPI pipeliné x Temps cycle pipeliné

CPI pipeliné= CPI idéal + Cycle de suspension
= 1 + Cycle de suspension par instruction

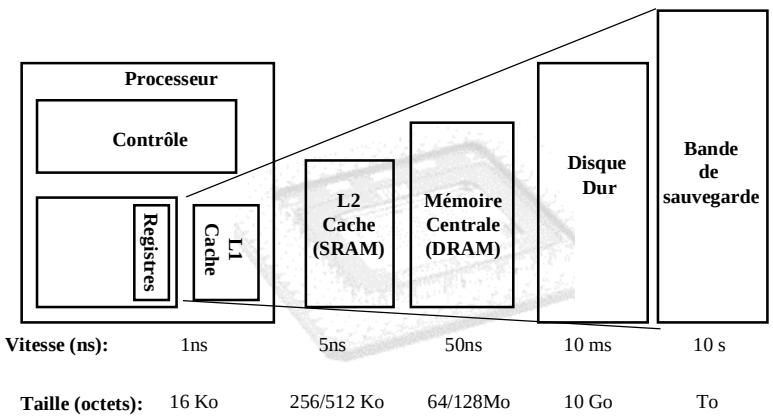
Si même fréquence horloge
Accélération = Profondeur pipeline
1+ Cycle de suspension

8| Hiérarchie Mémoire

Vitesse UC /RAM



Hiérarchie mémoire actuelle



Cache Unifié / Cache séparé (Données Instructions)

Principe de localité

- Localité temporelle
  - données ou instructions utilisées récemment le seront dans un futur proche (boucles, tableaux de données)
  - instruction 90% du temps sur 10% d'instructions
- localité spatiale
  - données ou instructions proches seront utilisées en même temps (tableau, instructions séquentielles)
  - accès à des petites portions de l'espace mémoire.

⇒Mettre dans le processeur les instructions et les données en cours d'utilisation

⇒Rendre les cas fréquents plus rapides

Le cache

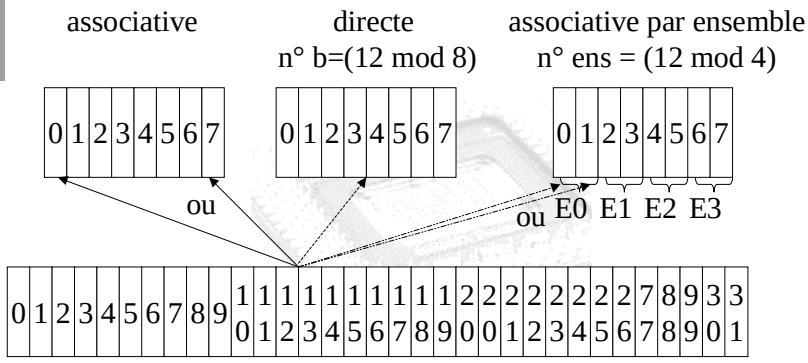
- Q1 : Où placer un bloc dans le cache ?
- Q2 : Comment trouver un bloc dans le cache ?
- Q3 : Quel bloc doit être celui remplacé ?
- Q4 : Que se passe t-il lors d'une écriture ?

- On ne raisonne plus en octet mais en bloc (ensemble d'octets - localité spatiale)

Q1 : Placer un bloc dans le cache

- Une place unique (correspondance directe)
  (numéro bloc) modulo (nombre blocs cache)
- N'importe où (correspondance associative)
- Ensemble de blocs restreint (correspondance associative par ensemble de n blocs)
  (numéro bloc) modulo (nombre ensembles dans le cache)

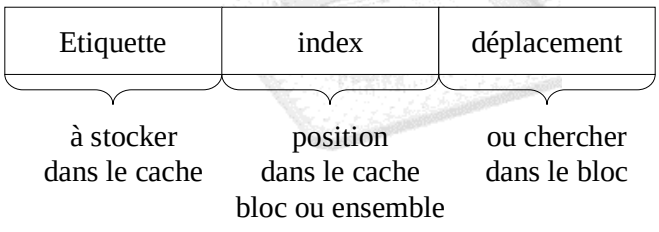
Fonctions de correspondance



Mémoire 32 blocs

Q2 : Trouver

- Une adresse contient
  - offset dans le bloc
  - étiquette
  - numéro de bloc dans le cache



Exemple

- Taille bloc = 1Ko
- Mémoire centrale 32 Mo = $2^{25} \rightarrow$ 32 K blocs
- Mémoire cache 256 Ko $\rightarrow$ 256 blocs $\rightarrow$ 64 ensembles

fonction de correspondance directe :

7 bits	8 bits	10

correspondance associative par ensemble de 4 blocs

9 bits	6 bits	10

fonction de correspondance associative

15 bits	0 bits	10

Organisation d'un bloc du cache

- bit V = 1 si bloc occupé

Etiquette	V	n octets (données)

- Comparaison simultanée de toutes les étiquettes des blocs présents dans l'ensemble.
- Si bloc absent "pénalité d'accès au cache" pour vider le bloc en cours dans la mémoire et charger le nouveau bloc.

Q3 : Remplacer

- Si tous les blocs de l'ensemble sont occupés, il faut en enlever un
  - Au hasard : facile à réaliser
  - L.R.U (Least Recent Used) Evite de supprimer des informations bientôt nécessaires

Q4 : Ecrire

- Ecriture simultanée → Données dans le Cache et dans le cache de niveau inférieur (ou MC)
  - Augmentation trafic mémoire
  - cohérence MC / Cache

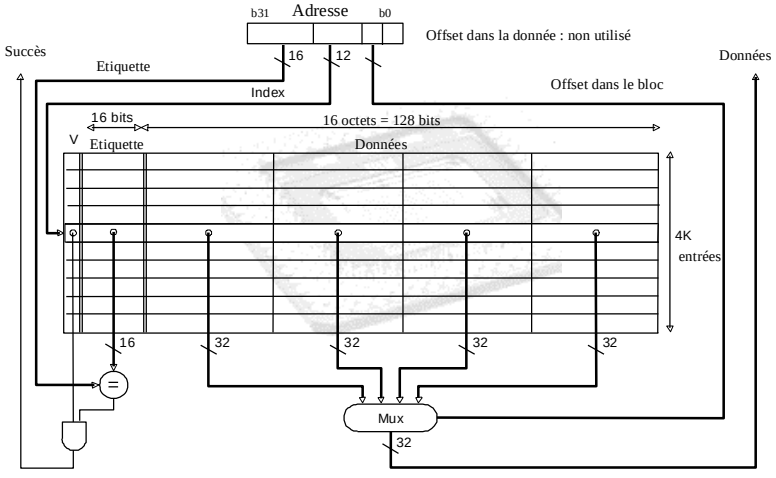
Ecrire

- Réécriture (recopie, rangement) → Données dans Cache et copie dans le cache de niveau inférieur (ou MC) quand on a besoin du bloc
  - 1 bit modifié
  - Diminution du trafic mémoire
  - incohérence cache / MC
- Si la donnée n'a jamais été utilisée
  - Ecriture allouée : chargement du bloc, puis modification donnée dans ce bloc (localité temporelle, on espère avoir besoin encore de ce bloc)
  - Ecriture non allouée : le bloc est modifié dans le niveau inférieur

Exemple de cache direct

- Mémoire : 8 bits
- Données et Adresse processeur : 32 bits
- 2^{30} doubles mots de 32 bits en mémoire
- Cache de 64 Kilos octets - 16 Kilos doubles mots
- Blocs de 16 octets = 4 (2^2)mots = Offset 2 bits
- Cache 4 Kilos blocs = 2^{12} = Index 12 bits
- Mémoire centrale 2^{28} blocs
- Le cache 1 parmi $2^{28}/2^{12} = 2^{16}$ = Etiquette 16 bits

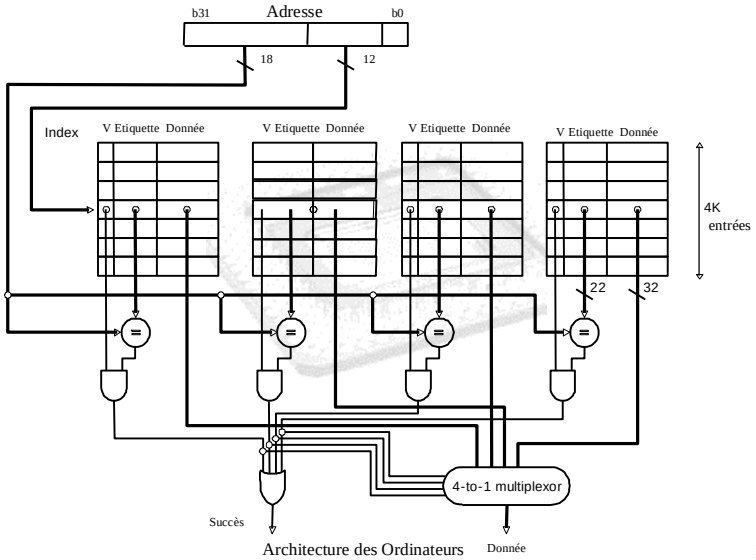
Exemple de cache direct contenant des blocs de 4 mots



Exemple de cache associatif par ensemble de 4 blocs de 1 mot

- Mémoire : 8 bits
- Données et Adresse processeur : 32 bits
- 2^{30} doubles mots de 32 bits en mémoire
- Cache de 64 Kilos octets - 16 Kilos doubles mots
- Blocs de 4 octets = 1 (2^0) mots = Offset 0 bit
- Cache 4 Kilos Ensemble = 2^{12} = Index 12 bits
- Mémoire centrale 2^{30} blocs
- Le cache 1 parmi $2^{30}/2^{12} = 2^{18}$ = Etiquette 18 bits

Cache associatif par ensemble 4



Pénalité d'échec

- En cas d'échec cache, accès mémoire

  4 cycles pour envoyer envoi adresse mémoire

  +24 cycles d'attente pour accès à un double-mot

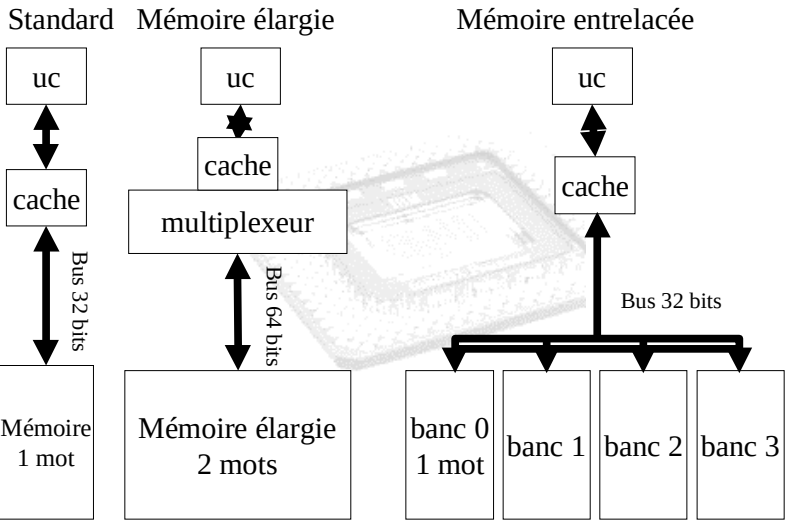
  +4 cycles récupérer la donnée double-mot

  32 cycles pour mettre une donnée dans le cache

- 1 bloc du cache contient 4 doubles-mots

  32 x 4 = 128 cycles pour 4 doubles mots

Réduction de la pénalité d'échec



Résultats

- Mémoire standard
  - 4 x 32 = 128 cycles
- Mémoire élargie
  - 2 mots x 32 + 2 (MX) = 66 cycles
- Mémoire entrelacée

  4 cycles pour l'envoi des adresses

  24 cycles d'attente

  4 x 4 récupérer les données

  44 cycles

Equation des performances

Temps exécution UC = (Nombre cycle UC+Cycle suspension) x Temps de cycle

Avec

Cycle suspension = *Nombre échecs* x *Pénalité d'échecs*

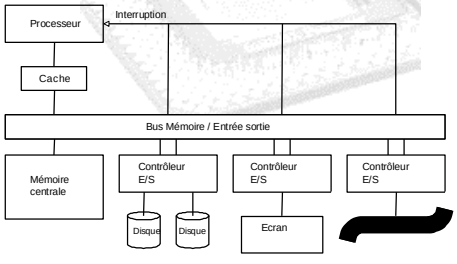
= *NI x Echecs par instruction* x *Pénalité*

= *NI x Accès mémoire par instruction* x *Taux échecs (%)* x *Pénalité*

9| Les Entrées / Sorties

Introduction

- Conception Processeur : Augmentation performances
- Conception E/S : extensibilité, tolérance pannes + performances
- E/S aussi importantes que le processeur



Exemple loi d'Amdalh

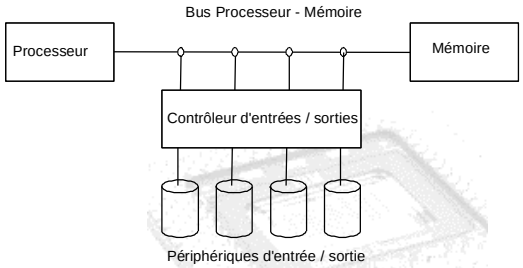
Programme 100 secondes dont E/S 10secondes
Gain vitesse microprocesseur 50% an . Gain E/S 0%
Quel sera le gain en vitesse d'exécution du programme en 5 ans?

année	Temps UC	Temps exécution
0	90s	100
1	90/1,5 = 60s	70
2	45/1,5=40s	50
…		
5	12s	22s
Accélération	7,5	4,5

Caractéristiques

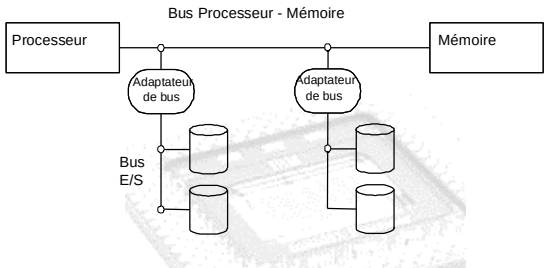
- Comportement
  - Entrée, Sortie ou Entrée/sortie
- Partenaire → Quantité d'information à transférer
  - humain (clavier, souris, quantité faible)
  - machine (disque, réseaux, quantité forte)
- Débit
  - entre périphérique et mémoire
    - humain (lent 0,01Ko/s)
    - machine(rapide 2Mo/s)

Type de bus - Bus commun



• Bus court : les transfert mémoire doit être rapide.
• Spécifique à un processeur donné
• Bus de données et adresse communs : attente longue lors des entrées/sorties

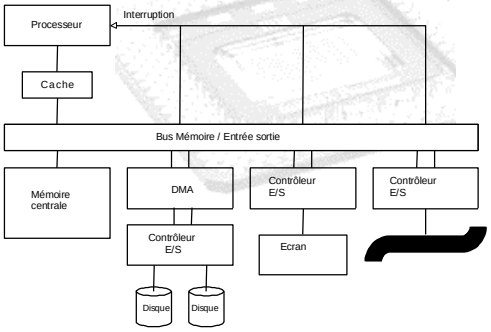
Bus dédiés au E/S



• Bus E/S : longs - lents
• Adaptateur : permet de convertir les vitesses de bus
• Bus standardisés (PCI,SCSI,…)
• Une connexion aux bus processeur-mémoire, les autres bus d ' E/S sont connectés en dessous.

DMA

- Pas d'accès au microprocesseur pour les transferts
- microprocesseur « disponible » sur ses caches



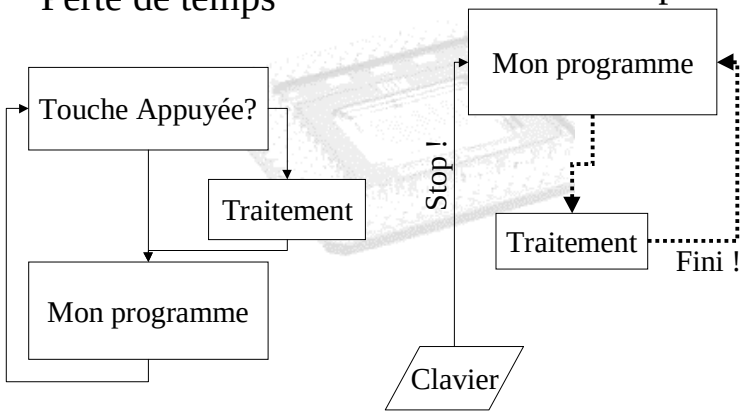
Communication E/S processeur

Interrogation

Perte de temps

Interruption :

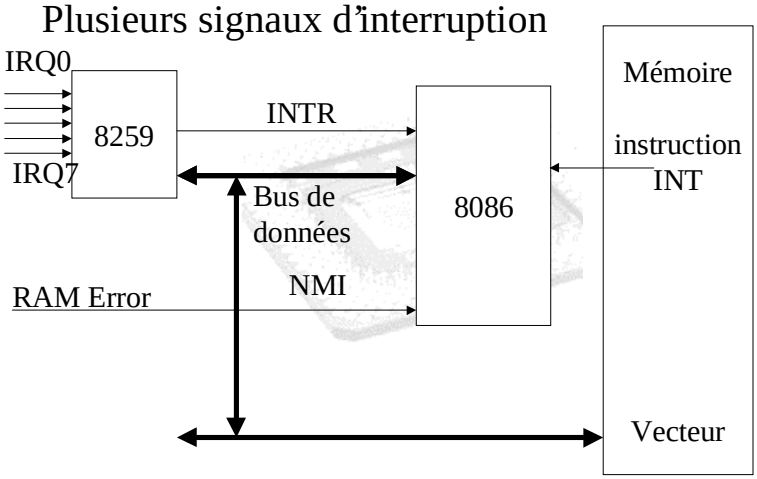
Gain de temps



Contrôleur d'interruption

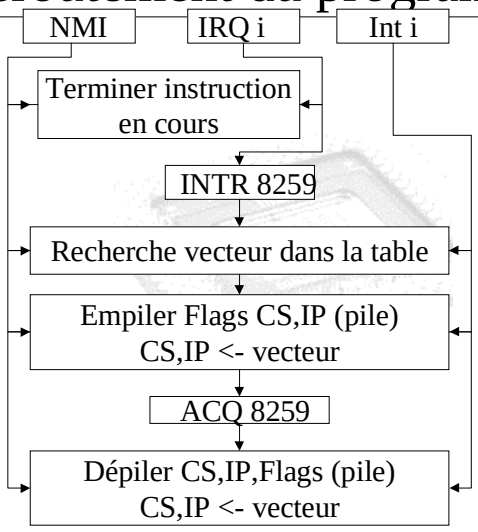
x86

Plusieurs signaux d'interruption

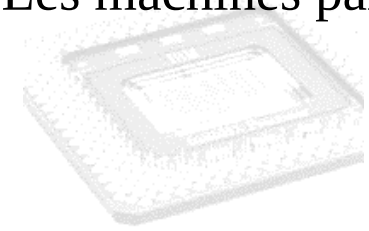


Déroutement du programme

x86



10 Les machines parallèles

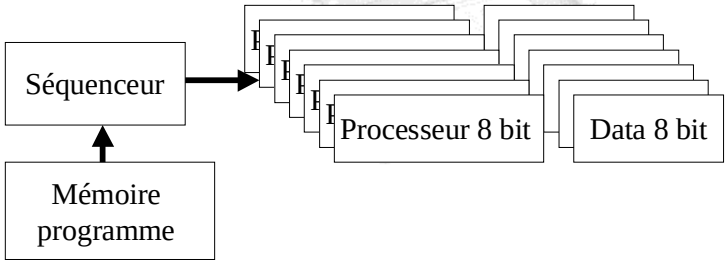


Classification de Flynn

- SISD Une instruction, 1 donnée
- SIMD Une instruction,1 vecteur de données
- MISD
  - Aucune machine commerciale
- MIMD

SIMD

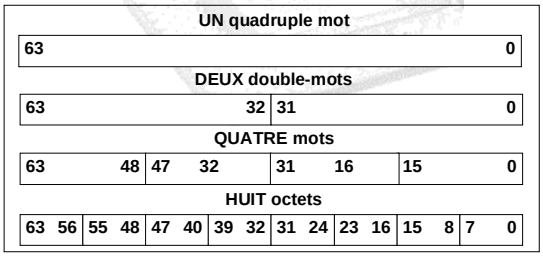
- Même instruction sur plusieurs processeurs
- Processeur a sa propre mémoire de données
- Une seule mémoire programme



Intel MultiMédia eXtension

x86

- Créée en 1996, intégrée aux Pentium III
- Accélère les calculs sur les entiers
- Composée de 57 instructions
- Manipule des blocs 64 bits décomposables en



Les instructions MMX

- Les instructions agissent sur les paquets de données. Pas de passage entre deux paquets.
- En arithmétique saturée, le résultat issu d'un débordement est saturé à la valeur extrême correspondante : Addition de deux octets
  - •Arithmétique classique : F0h + 60h = 50h + carry
  - •Arithmétique saturée: F0h + 60h = FFh
  - •Exemple de valeurs de saturation : octet

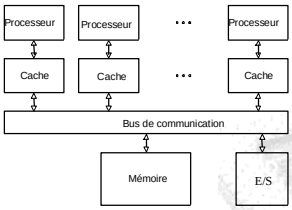
	Minimum	Maximum
Signée	00h (0)	FFh (255)
Non Signée	80h (-127)	7Fh (127)

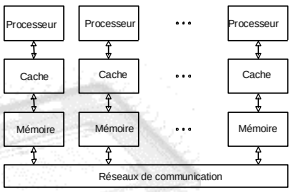
Intérêt :
pas de test de débordement (CF),
pas de correction à effectuer.

MIMD

- Processeur a sa propre mémoire instruction
- Processeur a sa propre mémoire de données
- Les microprocesseurs sont standards
  - 2 à 32 processeurs : mémoire partagée centralisée, UMA (Uniform Memory Access)
  - Mémoire distribuée sur chaque processeur
  - Combinaison des deux
    - mémoire distribuée sur chaque nœud
    - plusieurs processeurs par nœud

2 Architectures possibles

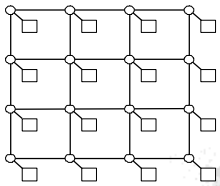


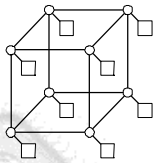


Machine à bus unique
Mémoire centralisée

Machine à réseaux
Mémoire distribuée

Topologie des réseaux





Réseaux 2 dimensions
16 processeurs

Carré = Processeur
Rond : Commutateur

Beaucoup de commutateurs
à traverser

Réseaux 3 dimensions
8 processeurs

Carré = Processeur
Rond : Commutateur

Moins de commutateurs
à traverser

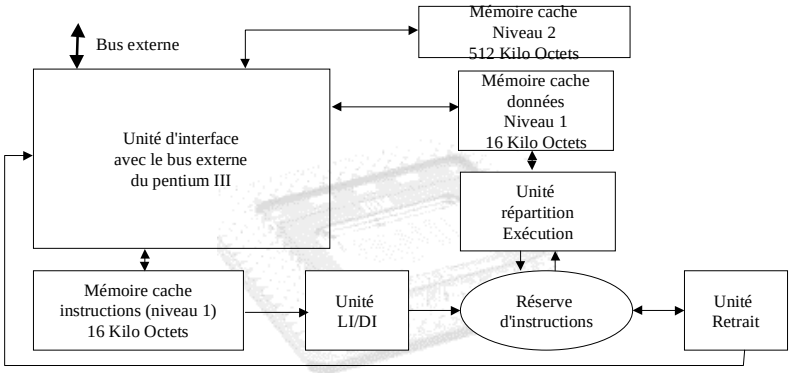
11| x86 et PC

Evolution de la famille x86

	An	MIPS	Mhz	Tr	Data	Adr	Reg	Cache
8086	1978	0,8	8	29k	16	1M	16	x
286	1982	2,7	12,5	134k	16	16M	16	x
386	1985	6	20	275k	32	4GB	32	L1 8k
486	1989	20	25	1,2M	32	4GB	32	L1 16k
Pentium	1993	100	80	3,1M	64	4GB	32	L1 16k
PII	1997	466	266	7M	64	64GB	32	L1 16k L2 512
PIII	1999	1000	500	8,2M	64	64GB	32 +128	L1 32 L2 512

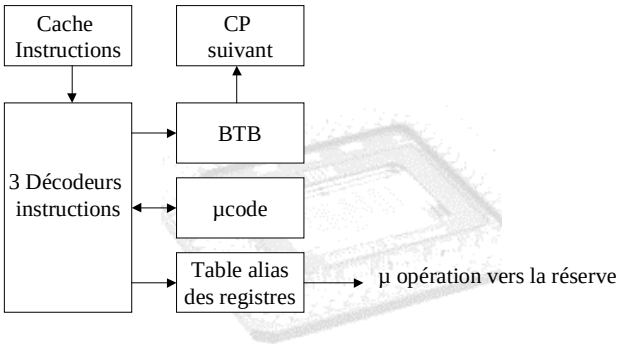
Mhz : Fréquence du microprocesseur
Tr: Nombre de transistors pour le microprocesseur. (hors cache)
data : Taille du bus de donnée.
Adr : Taille de la mémoire.
Reg : Nombre de registre dans le microprocesseur
Cache : Taille du cache

Architecture du Pentium III



- 3 machines indépendantes
- Réserve d'instructions (exécution dans le désordre)
- Analyse du flux de données : réorganisation du code
- Branchement prédictif multiple

Lecture et Décodage Instruction

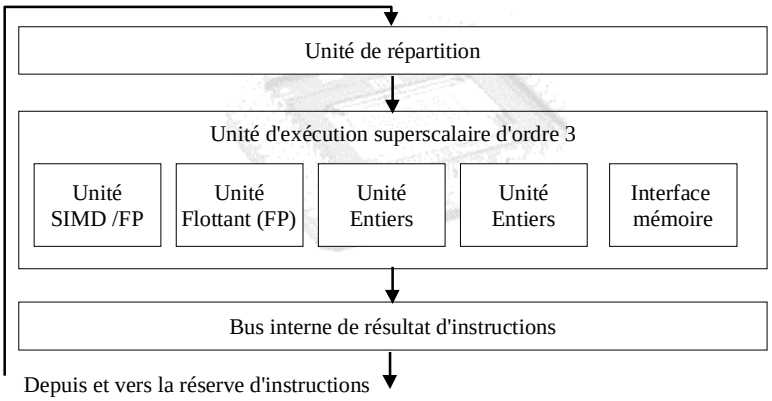


Conversion en µopérations (une à quatre par instruction)
Les autres plus complexes utilisent le microcode
Conversion du nom des registres en registres internes

Unité de répartition et d'exécution

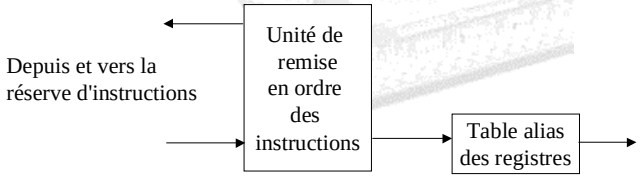
Distribution à l'unité d'exécution correcte

Retour dans la réserve après exécution.



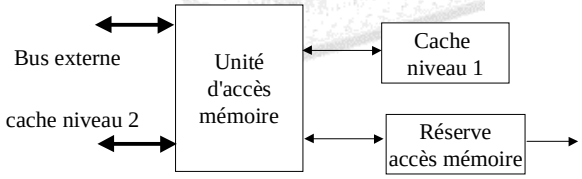
Unité de retrait

- Supprime les micro opérations de la réserve dans l'ordre du programme
- Redonner au registres leurs noms d'origine

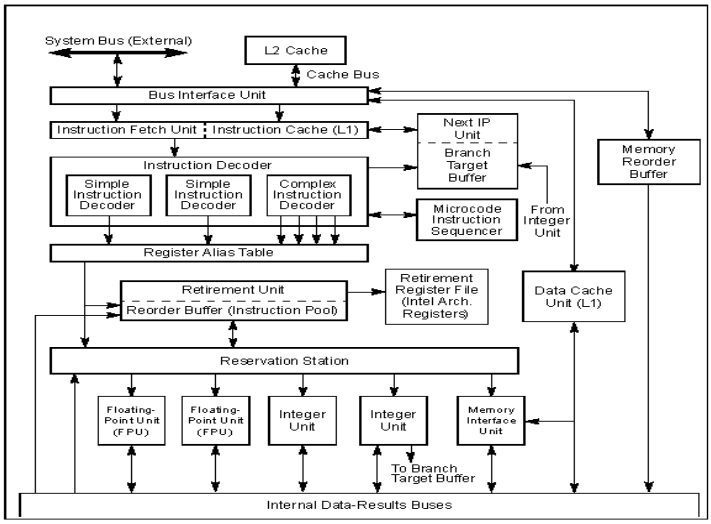


Bus interface

- Accès mémoire non bloquants avec une réserve des accès mémoire à effectuer.
- Bus rapide pour le cache de niveau 2



Architecture PIII



Architecture AMD Athlon

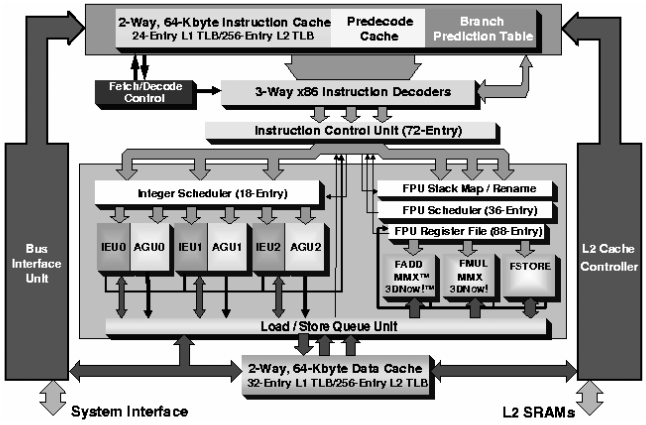
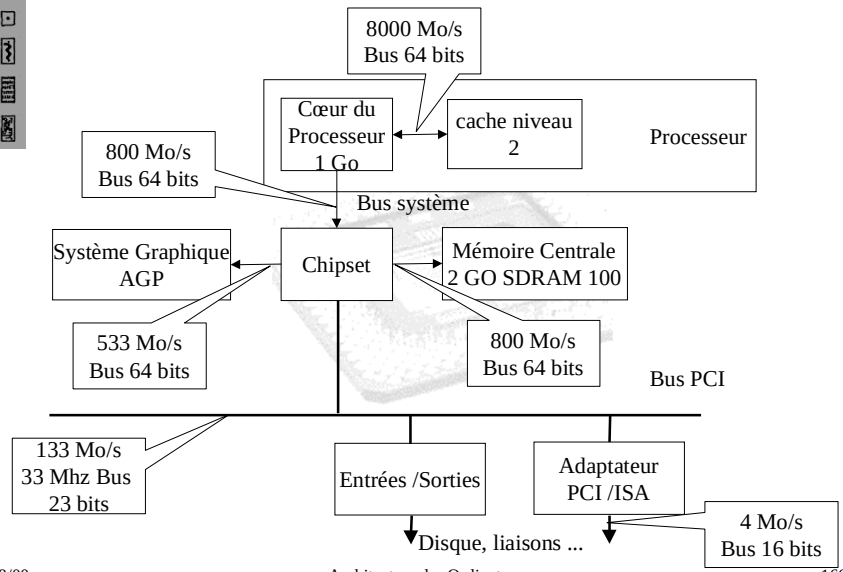


Figure 1. AMD Athlon™ Processor Block Diagram

Architecture du PC



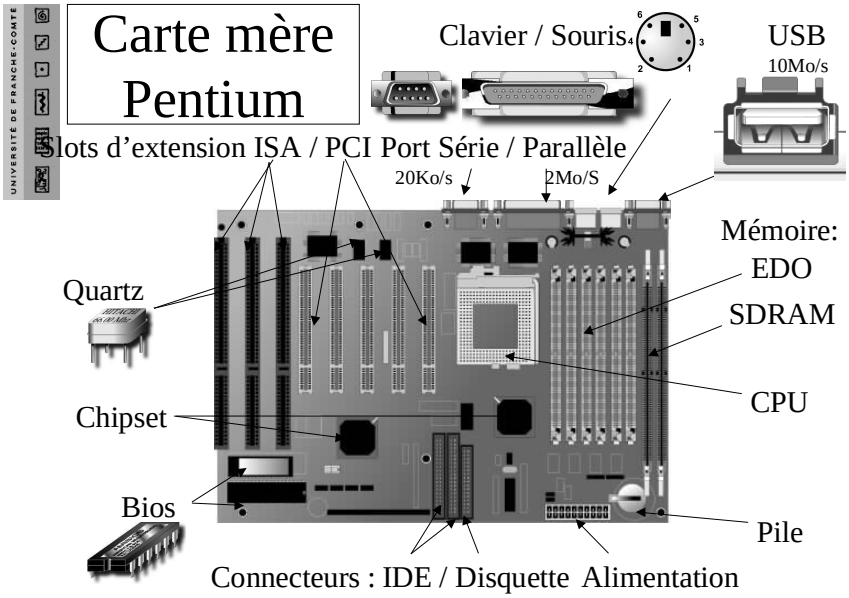
Caractéristiques des bus - 1

- bus ISA : Industry Standard Architecture - 1984
  - (16 bits - 8 Mhz - synchronisé par processeur)
- bus EISA : Extended ISA
  - (32 bits - 10 Mhz - 33 Mo / seconde)
- bus PCI : Peripheral Component Interconnect
  - (32 bits - 33 Mhz - 132 Mo /seconde - Indépendant processeur, plug and play)
  - Transfert donnée en mémoire sans passer par le processeur
- AGP : Accelerated Graphics Port
  - (vidéo 66 Mhz - 64 bits 528 Mo/s)
  - 4 fois débit PCI, Gestion mémoire améliorée
  - Réduction du trafic sur bus PCI

Caractéristiques des bus - 2

- bus IDE : Integrated Drive Electronic
  - (disque de 20 à 200Mo - 4 Mo/s)
- bus EIDE : Enhanced IDE (disque > 500 Mo - 16 Mo/s)
  - gère quatre disques (2 canaux)
  - gère la DMA
  - Ultra DMA 33 (DMA synchrone) 33 Mo /s
  - Plug and play
  - gestion CRC (cyclic redundancy code)
- SCSI : Small computer signal Interface
  - (4 Mo/s à 80 Mo/s)

Carte mère Pentium



Connecteurs


EISA


PCI


AGP


SIMM (36 bits par ensemble de 2)


DIMM (72 bits)


USB Remplace ports séries trop lents
127 périphériques chaînés (clavier, souris, modem
débit 10 Mo/s

La mémoire

- DRAM (mémoire centrale) 32 Mo minimum
  - (1 transistor) rafraîchissement périodique (8 ms)
    - EDO (Extended Data Output) - 50 ns 5 -2 - 2 - 2 (66 Mhz)
    - Bande passante maxi 264 Mo/s
    - SDRAM (synchronous DRAM) Synchrone horloge système 5-1-1-1
- SRAM (mémoire cache) Une bascule D
  - niveau 1 : intégré au processeur 64 Ko
  - niveau 2 : 512 ko minimum