

Base des Systèmes Informatiques

Adrien Revault d'Allonnes

<http://www.allonnes.me>

Premier semestre 2009

1 Architecture des ordinateurs

- Représentation de l'information
- Les composants élémentaires des ordinateurs
 - Unité centrale
 - Unités de mémoire
- Circuits combinatoires, circuits séquentiels
- Cycle d'exécution des instructions, langage assembleur

2 Systèmes d'exploitation

- Fonctions des systèmes d'exploitation
- Notions de processus
 - Communication de processus
 - Synchronisation de processus
- UNIX
 - Système de fichiers
 - Commandes de base
 - Programmation shell

3 Réseau

- Architecture en couches des réseaux informatiques

Architecture des ordinateurs

Cours 1 : représentation des données

Adrien Revault d'Allonnes

<http://www.allonnes.me>

Premier semestre 2009

Informations codées : le Morse

A	DI DAH	. _	N	DAH DI	_ .
B	DAH DI DI DI	_ . . .	O	DAH DAH DAH	_ _ _
C	DAH DI DAH DI	_ . _ .	P	DI DAH DAH DI	. _ _ .
D	DAH DI DI	_ . .	Q	DAH DAH DI DAH	_ _ . _
E	DI	.	R	DI DAH DI	. _ .
F	DI DI DAH DI	. . _ .	S	DI DI DI	. . .
G	DAH DAH DI	_ _ .	T	DAH	_
H	DI DI DI DI	. . .	U	DI DI DAH	. . _
I	DI DI	. .	V	DI DI DI DAH	. . . _
J	DI DAH DAH DAH	. _ _ _	W	DI DAH DAH	. _ _
K	DAH DI DAH	_ . _	X	DAH DI DI DAH	_ . . _
L	DI DAH DI DI	. _ . .	Y	DAH DI DAH DAH	_ . _ _
M	DAH DAH	_ _	Z	DAH DAH DI DI	_ _ . .

Informations codées : le Morse

A	DI DAH	. _	N	DAH DI	_ .
B	DAH DI DI DI	_ . . .	O	DAH DAH DAH	_ _ _
C	DAH DI DAH DI	_ . _ .	P	DI DAH DAH DI	. _ _ .
D	DAH DI DI	_ . .	Q	DAH DAH DI DAH	_ _ . _
E	DI	.	R	DI DAH DI	. _ .
F	DI DI DAH DI	. . _ .	S	DI DI DI	. . .
G	DAH DAH DI	_ _ .	T	DAH	_
H	DI DI DI DI	. . .	U	DI DI DAH	. . _
I	DI DI	. .	V	DI DI DI DAH	. . . _
J	DI DAH DAH DAH	. _ _ _	W	DI DAH DAH	. _ _
K	DAH DI DAH	_ . _	X	DAH DI DI DAH	_ . . _
L	DI DAH DI DI	. _ . .	Y	DAH DI DAH DAH	_ . _ _
M	DAH DAH	_ _	Z	DAH DAH DI DI	_ _ . .

ADRIEN = . _ . . . _ _ .

Types d'informations

- Texte
- Nombres
 - Entiers naturels (positifs)
 - Entiers relatifs (signés)
 - Réels
 - ...
- Images
- Sons
- Films
- ...

L'ordinateur manipule des données 'codées en binaire'

Représentation de l'information

Unité d'information : 1 bit (Binary digiT)

1 bit $\in \{0, 1\}$

Représentation en binaire $\equiv$ suite de bits

Une suite de bits sans contexte ne veut rien dire

- 8 bits = 1 octet
- 1024 octets = 2^{10} octets = 1 kilo-octet = 1Ko
- 1024 Ko = 2^{20} octets = 1 méga-octet = 1Mo
- 1024 Mo = 2^{30} octets = 1 giga-octet = 1Go
- 1024 Go = 2^{40} octets = 1 téra-octet = 1To
- ...

Représentation(s) d'entiers naturels

Tout entier n peut s'écrire dans une 'base de numération' $b \in \mathbb{N}$.

$$n = \sum_{i=0}^{\infty} a_i b^i, \text{ où } a_i \in \{0, 1, \dots, b-1\}, \forall i$$

Base	a_i	Expression	Résultat
16 (hexadécimal)	$\{0, \dots, 9, A, \dots, F\}$	$1 \times 16^3 + 1 \times 16^2 + 9 \times 16^1 + E \times 16^0$	$119E_{(16)}$
10 (décimal)	$\{0, \dots, 9\}$	$4 \times 10^3 + 5 \times 10^2 + 1 \times 10^1 + 0 \times 10^0$	$4510_{(10)}$
8 (octal)	$\{0, \dots, 7\}$	$1 \times 8^4 + 0 \times 8^3 + 6 \times 8^2 + 3 \times 8^1 + 6 \times 8^0$	$10636_{(8)}$
2 (binaire)	$\{0, 1\}$	$1 \times 2^{12} + 0 \times 2^{11} + 0 \times 2^{10} + 0 \times 2^9 + 1 \times 2^8 + 0 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$	$1000110011110_{(2)}$

TAB.: Exemples d'écritures de 4510

Changements de base

- Passer de l'écriture décimale à n'importe quelle autre
- Encadrement par les plus grandes puissances, ex. binaire :
 - $2^{13} = 8192 > 4510 > 4096 = 2^{12}$
 - $4510 - 2^{12} = 414$
 - $2^9 = 512 > 414 > 256 = 2^8$
 - ...
- Réalisable en binaire, puisque le reste $a_i \in \{0, 1\}$ mais long et fastidieux
- Complicé en octal ou hexadécimal (bases 8 ou 16)

Changements de base : décimal à binaire

- Astuces
- Divisions entières successives

$$\begin{array}{r} 50 \quad | \underline{2} \\ 0 \quad 25 \quad | \underline{2} \\ \swarrow 1 \quad 12 \quad | \underline{2} \\ \quad 0 \quad 6 \quad | \underline{2} \\ \quad \quad 0 \quad 3 \quad | \underline{2} \\ \quad \quad \quad 1 \quad 1 \quad | \underline{2} \\ \quad \quad \quad \quad 1 \quad 0 \end{array}$$

- Restes dans le sens inverse : $110010_{(2)} = 50_{(10)}$

Changements de base : binaire à octal ou hexa

- Sur n bits on écrit tous les chiffres de 0 à $2^n - 1$
- Sur 3 bits on écrit tous les chiffres de 0 à $2^3 - 1 = 7$
 - Pour un nombre exprimé en binaire, chaque groupe de 3 bits représente un a_i de l'écriture en octal
 - Exemple : $10100001_{(2)} = 010\ 100\ 001_{(2)} = 2\ 4\ 1_{(8)} = 241_{(8)}$
- Sur 4 bits on écrit tous les chiffres de 0 à $2^4 - 1 = 15$
 - De même, chaque groupe de 4 bits représente un a_i de l'écriture en hexadécimal
 - Exemple : $10100001_{(2)} = 1010\ 0001_{(2)} = A\ 1_{(16)} = A1_{(16)} = 161_{(10)}$

Représentation (binaire) d'entiers relatifs

- Codage habituel : complément à 2
- Nombres de longueur fixe, exemple nombre de huit bits :
 $0101\ 0110_{(2)} (= 86_{(10)})$
 - Inverser tous les bits : $1010\ 1001_{(2)}$
 - Ajouter 1 :
$$1010\ 1010_{(2)} = -1 \times 2^7 + 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^1$$
$$= -128 + 32 + 8 + 2 = -86_{(10)}$$
- Nombres positifs de $0000\ 0000_{(2)} = 0_{(10)}$ à
 $0111\ 1111_{(2)} = 127_{(10)}$
- Nombres négatifs de $1000\ 0000_{(2)} = -128_{(10)}$ à
 $1111\ 1111_{(2)} = -1_{(10)}$

Représentations alternatives d'entiers relatifs

Il existe d'autres représentations des nombres négatifs :

- Complément à 1

- Inversion de tous les bits
- Nombres positifs de $0000\ 0000_{(2)} = 0_{(10)}$ à $0111\ 1111_{(2)} = 127_{(10)}$
- Nombres négatifs de $1000\ 0000_{(2)} = -127_{(10)}$ à $1111\ 1111_{(2)} = -0_{(10)}$
- Deux façons d'écrire 0

- Par excédent à 2^{m-1} pour un nombre de m bits

- Sur huit bits $-86 + 2^7 = 42$ entier naturel, non signé
- Nombres négatifs de $0000\ 0000_{(2)} = -128_{(10)}$ à $0111\ 1111_{(2)} = -1_{(10)}$
- Nombres positifs de $1000\ 0000_{(2)} = 0_{(10)}$ à $1111\ 1111_{(2)} = 127_{(10)}$

- Nombres positifs sur 8 bits
de $0000\ 0000_{(2)} = 0_{(10)}$ à $0111\ 1111_{(2)} = 127_{(10)}$
- Plus généralement, pour un nombre sur m bits de 0 à $2^{m-1} - 1$
- Nombres négatifs sur 8 bits
de $1000\ 0000_{(2)} = -128_{(10)}$ à $1111\ 1111_{(2)} = -1_{(10)}$
- Plus généralement, pour un nombre sur m bits de -2^{m-1} à -1

Un peu d'arithmétique

L'addition se fait comme d'habitude, avec retenues

NB : $0 + 1 = 1$ et $1 + 1 = 10$ où le bit de gauche est la retenue

- $0010\ 0101_{(2)} + 0101\ 0000_{(2)} = 0111\ 0101_{(2)}$
($\Leftrightarrow 37_{(10)} + 80_{(10)} = 117$)
- $0010\ 0101_{(2)} + 0111\ 0000_{(2)} \neq 1001\ 0101_{(2)}$
($\Leftrightarrow 37_{(10)} + 112_{(10)} \neq -107$) $\rightarrow$ dépassement de capacité
- Dépassement possible que si les deux nombres sont de même signe

Représentation de réels

Nombres à virgule flottante

Représentation économique de réels : (M, B, E)

- M : la mantisse
- B : la base
- E : l'exposant
- Soit $q \in \mathbb{R}$, q peut s'écrire : $q = M \times B^E$
 - $3,1416 = 31416 \times 10^{-4} = 0,31416 \times 10^1 = \dots$

Représentation de réels, standard IEEE

- Standard IEEE754 : codés sur 32 ou 64 bits (base 2)

32 bits, simple :

- Signe : 1 bit
- Exposant : 8 bits en excédent à 127
- Mantisse : 23 bits

64 bits, double :

- Signe : 1 bit
- Exposant : 11 bits en excédent à 1023
- Mantisse : 52 bits

- Un réel trop proche de zéro peut provoquer un débordement inférieur
- Un réel trop grand peut provoquer un débordement supérieur

Représentation de caractères

Un caractère : tout symbole du clavier

- Code ASCII

American Standard Code for Information Interchange

- Code à 7 bits $\Rightarrow 2^7 = 128$ symboles disponibles

- 0110001 : 1
- 1000001 : A
- 1100001 : a
- 0100000 : a
- ...

- Unicode

- Code à 16 bits $\Rightarrow 2^{16} = 65\,536$ symboles disponibles

- Près de 200 000 signes en usage dans le monde

- Existe aussi sur 32 bits

Architecture des ordinateurs

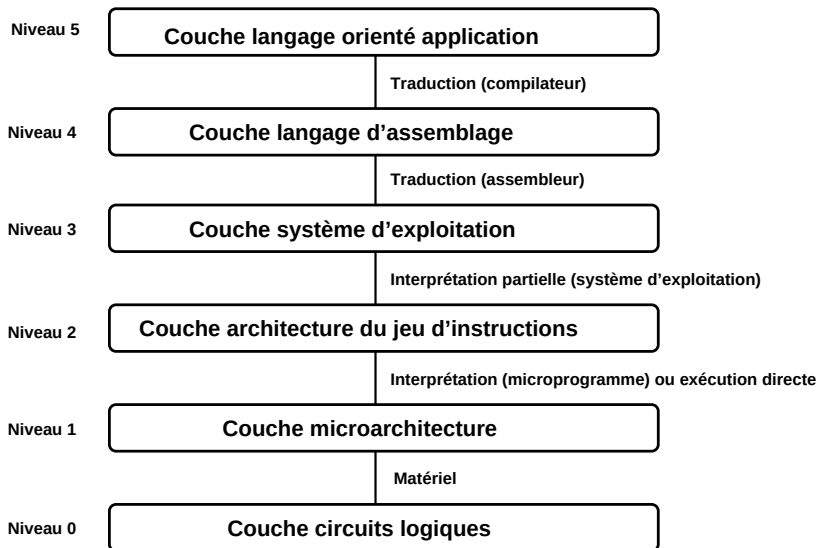
Cours 2 : composants élémentaires des ordinateurs

Adrien Revault d'Allonnes

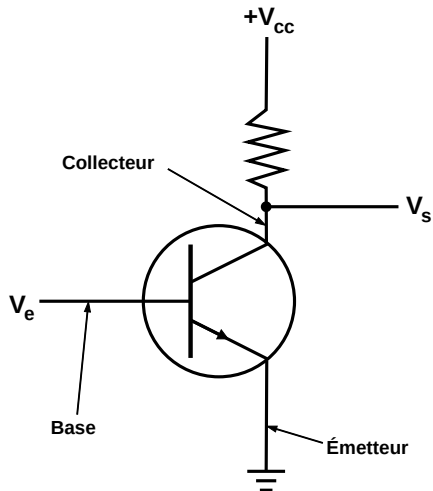
<http://www.allonnes.me>

Premier semestre 2009

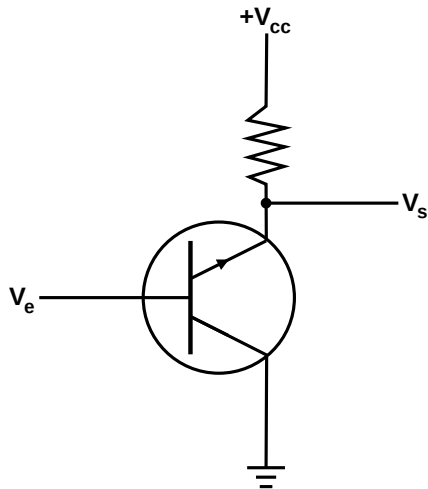
Ordinateur à six couches



Transistors : l'unité de base



Transistor inverseur



Transistor positif

Combinaison de transistors

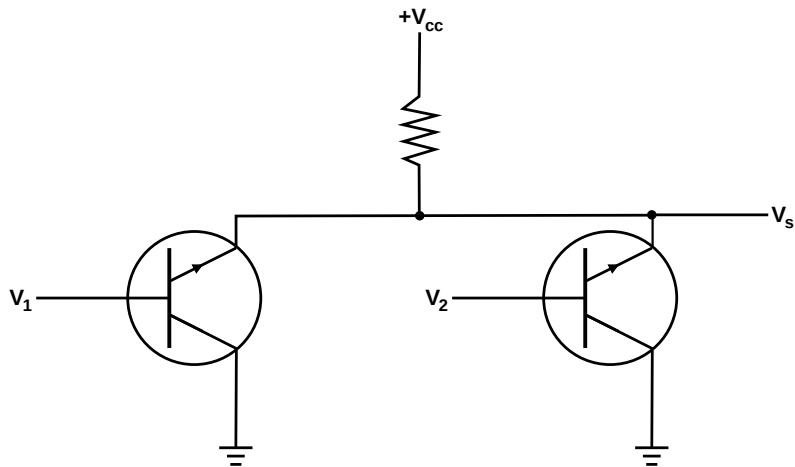


FIG.: Porte OU

Portes élémentaires

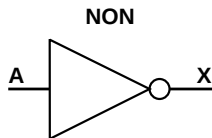


FIG.: Porte NON

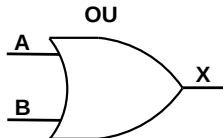


FIG.: Porte OU

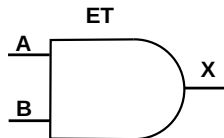


FIG.: Porte ET

Tables de vérité correspondantes

A	X
0	1
1	0

A	B	X
0	0	0
1	0	1
0	1	1
1	1	1

A	B	X
0	0	0
1	0	0
0	1	0
1	1	1

Portes complètes élémentaires

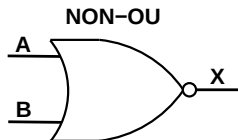


FIG.: Porte NON-OU

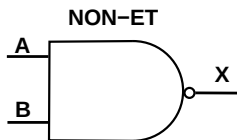


FIG.: Porte NON-ET

Tables de vérité correspondantes

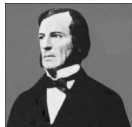
A	B	X
0	0	1
1	0	0
0	1	0
1	1	0

A	B	X
0	0	1
1	0	1
0	1	1
1	1	0

Algèbre de Boole, pourquoi ?

- Le processeur exécute des programmes
- Un programme est une suite d'instructions
- Une instruction se décompose en opérations élémentaires
- Une opération élémentaire est réalisée par des fonctions logiques
- Une fonction logique est réalisée par des circuits électroniques, physiques

- George Boole, mathématicien anglais du XIXème siècle



- Algèbre sur deux valeurs de vérité : $\{0, 1\}$, $\{F, V\}$, ...
- Raisonnement par exhaustion des cas
⇒ Tables de vérité
- Opérateurs logiques classiques : ET, OU, NON
- Autres opérateurs : XOR (OU-EX), NAND (NON-ET), NOR (NON-OU)
- Construction de fonctions sur les valeurs de vérités des variables
e.g. $1 + 1 = 2$ est Vrai

Nom	Forme ET	Forme OU
Loi d'identité	$1.A=A$	$0+A=A$
Loi de nullité	$0.A=0$	$1+A=1$
Loi d'idempotence	$A.A=A$	$A+A=A$
Loi d'inversion	$A.\bar{A}=0$	$A+\bar{A}=1$
Loi commutative	$A.B=B.A$	$A+B=B+A$
Loi associative	$(A.B).C=A.(B.C)$	$(A+B)+C=A+(B+C)$
Loi distributive	$(A+B).C=(A.C)+(B.C)$	$A.(B+C)=(A.B)+(A.C)$
Loi d'absorbtion	$A.(A+B)=A$	$A+(A.B)=A$
Loi de DeMorgan	$\overline{A.B}=\bar{A}+\bar{B}$	$\overline{A+B}=\bar{A}.\bar{B}$

Implémentation d'un circuit de fonction booléenne

Exemple : la fonction majoritaire à trois variables

$F(A,B,C)=M$, où $M=1 \Leftrightarrow$ au moins deux de ses variables sont vraies

1) Écrire sa table de vérité

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Implémentation d'un circuit de fonction booléenne

Exemple : la fonction majoritaire à trois variables

$F(A,B,C)=M$, où $M=1 \Leftrightarrow$ au moins deux de ses variables sont vraies

2) Conserver les lignes où M est vrai

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Implémentation d'un circuit de fonction booléenne

Exemple : la fonction majoritaire à trois variables

$F(A,B,C)=M$, où $M=1 \Leftrightarrow$ au moins deux de ses variables sont vraies

3) Conserver les lignes où $M = 1$

A	B	C	M
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

Implémentation d'un circuit de fonction booléenne

Exemple : la fonction majoritaire à trois variables

$F(A,B,C)=M$, où $M=1 \Leftrightarrow$ au moins deux de ses variables sont vraies

- 4) Écrire chaque ligne comme la conjonction (ET) des termes vrais et de l'inverse (NON) des termes faux

A	B	C	M
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

$$\bar{A}.B.C$$

Implémentation d'un circuit de fonction booléenne

Exemple : la fonction majoritaire à trois variables

$F(A,B,C)=M$, où $M=1 \Leftrightarrow$ au moins deux de ses variables sont vraies

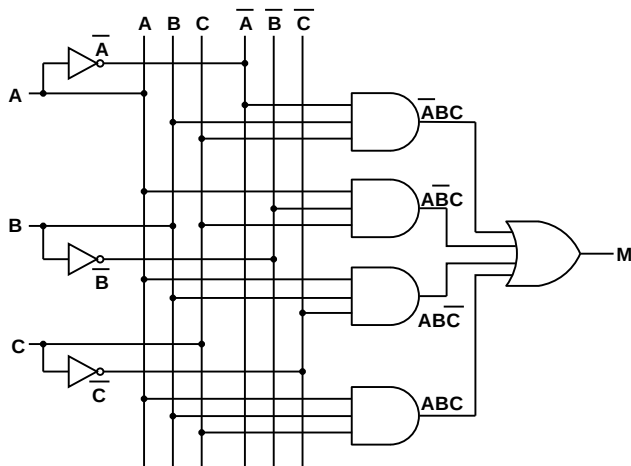
5) La fonction est la disjonction (OU) de ces expressions

A	B	C	M
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

$$M=(\overline{A}.B.C)+(A.\overline{B}.C)+(A.B.\overline{C})+(A.B.C)$$

Implémentation d'un circuit de fonction booléenne

Dans le circuit, les portes correspondantes remplacent les opérateurs



$$M = (\bar{A}.B.C) + (A.\bar{B}.C) + (A.B.\bar{C}) + (A.B.C)$$

Architecture des ordinateurs

Cours 3 : circuits

Adrien Revault d'Allonnes

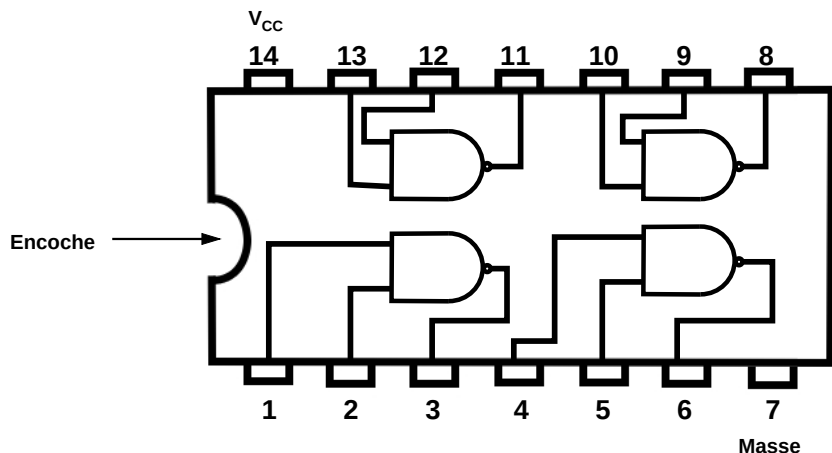
<http://www.allonnes.me>

Premier semestre 2009

- Les portes ne sont ni fabriquées ni vendues à l'unité
⇒ Circuits intégrés logiques, CI, puces, chips, ...
- Circuits SSI (Small Scale Integrated) : de 1 à 10 portes
- Circuits MSI (Medium Scale Integrated) : de 10 à 100 portes
- Circuits LSI (Large Scale Integrated) : de 100 à 100 000 portes
- Circuits VLSI (Very Large Scale Integrated) : > 100 000 portes

Circuits intégrés logiques

- Circuits SSI (Small Scale Integrated) : de 1 à 10 portes



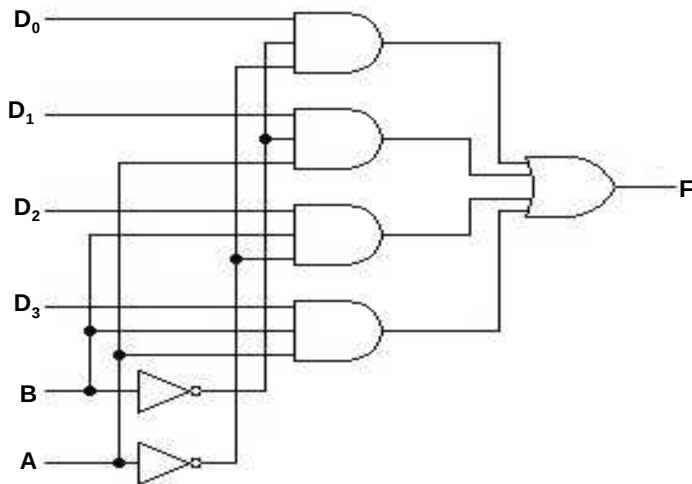
- Circuits combinatoires :

- Ne dépend que de la valeur de ses entrées
- Aucune notion de temps

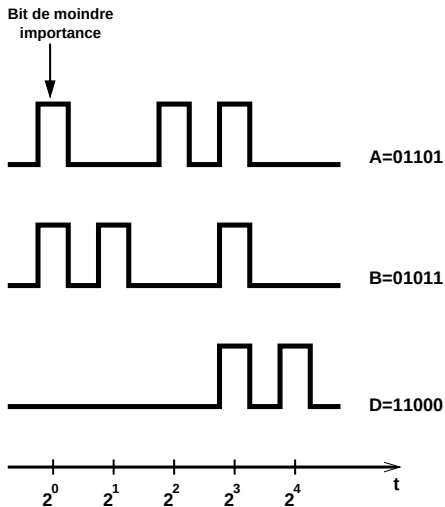
- Circuits séquentiels :

- Fonction de l'état du système
- Contient des éléments mémoires
- Comportement variable avec le temps
- Synchronisation des états fonction du cycle d'horloge

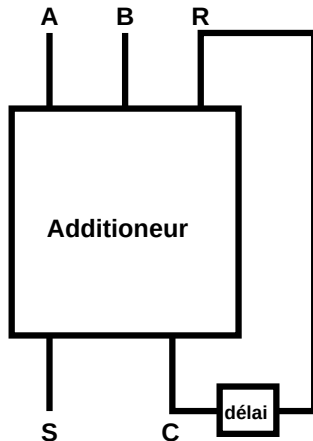
Un circuit combinatoire : le multiplexeur



Un circuit séquentiel : l'additionneur séquentiel



Un circuit séquentiel : l'additionneur séquentiel



Structure d'un ordinateur

Cours 4 : unité centrale

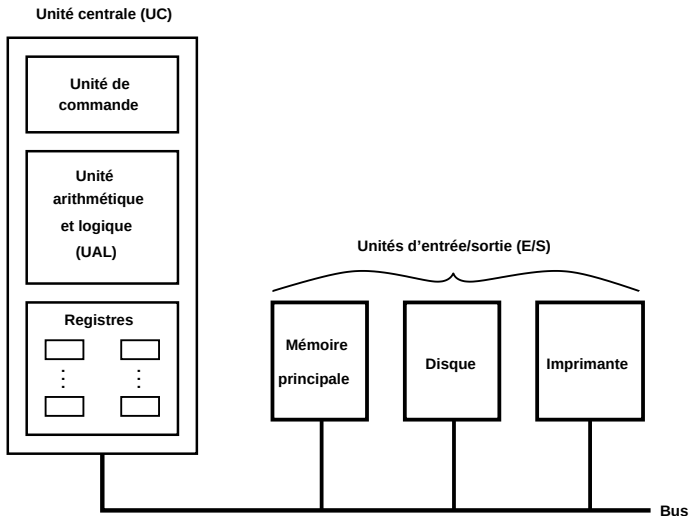
Adrien Revault d'Allonnes

<http://www.allonnes.me>

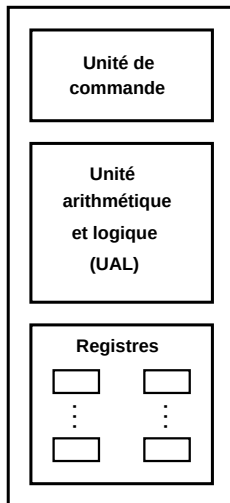
Premier semestre 2009

- Unité centrale (UC, CPU, processeur, ...) : cerveau
- Mémoires
- Dispositifs d'entrée/sortie (périphériques)
- Bus : connecteur entre différents composants

Structure d'un ordinateur



Unité centrale (UC)



Cerveau de l'ordinateur : exécute les programmes. Constitué de

- Unité de commande
 - Extrait chaque instruction de la mémoire principale
 - Détermine le type de l'instruction
- Unité arithmétique et logique (UAL)
 - Effectue les opérations : additions, ET, ...
- Registres : mémoire interne à grande vitesse
 - Stocke temporairement les informations de commande
 - Compteur ordinal (Program Counter) : pointe l'instruction suivante
 - Registre instruction : stocke l'instruction en cours

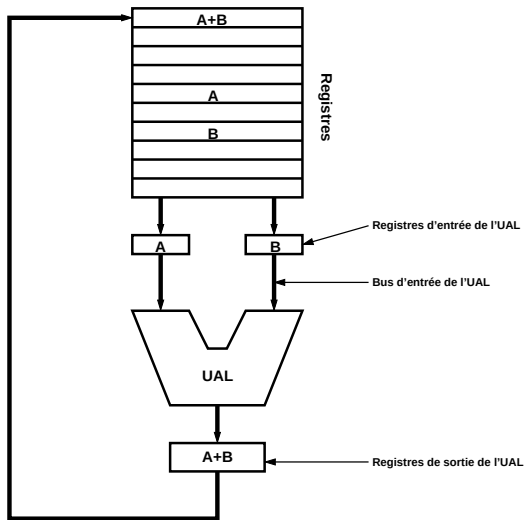
- Le processeur exécute des programmes
- Un programme est une suite d'instructions
- Une instruction se décompose en opérations élémentaires

Structure de l'unité centrale

- Unité de commande
 - Extrait chaque instruction de la mémoire principale
 - Détermine le type de l'instruction
- Unité arithmétique et logique (UAL)
 - Effectue les opérations : additions, ET, ...
- Registres : mémoire interne à grande vitesse
 - Stocke temporairement les informations de commande
 - Compteur ordinal (Program Counter) : pointe l'instruction suivante
 - Registre instruction : stocke l'instruction en cours

- Instructions registre-mémoire
 - Chargement de données de la mémoire aux registres
 - Renvoi de données de registres à la mémoire
 - Instructions registre-registre
 - Registres vers registres d'entrée UAL
 - Opération
 - Registres sortie UAL vers registres
- ⇒ Cycle du chemin de données

Exemple de cycle du chemin de données : $A + B$



Exécution d'une instruction (chargement-décodage-exécution)

- ➊ Récupérer instruction dans la mémoire, charger dans RI
- ➋ Modifier PC pour qu'il pointe vers adresse instruction suivante
- ➌ Déterminer genre d'instruction chargée
- ➍ Si instruction nécessite un mot mémoire, le localiser
- ➎ Charger le mot, si besoin, dans un registre UC
- ➏ Exécuter instruction
- ➐ Loop -> 1

Structure d'un ordinateur

Cours 5 : assembleur

Adrien Revault d'Allonnes

<http://www.allonnes.me>

Premier semestre 2009

Langage machine difficile et fastidieux $\Rightarrow$ rare et cher.

Langage d'assemblage :

- Représentation symbolisée langage sous-jacent
- Traduit langage niveau inférieur
- Interprété par le niveau correspondant
- Traducteur : assembleur

80X86 : La classe des puces d'Intel allant du 8086 au 80486.

⇒ Pentiums

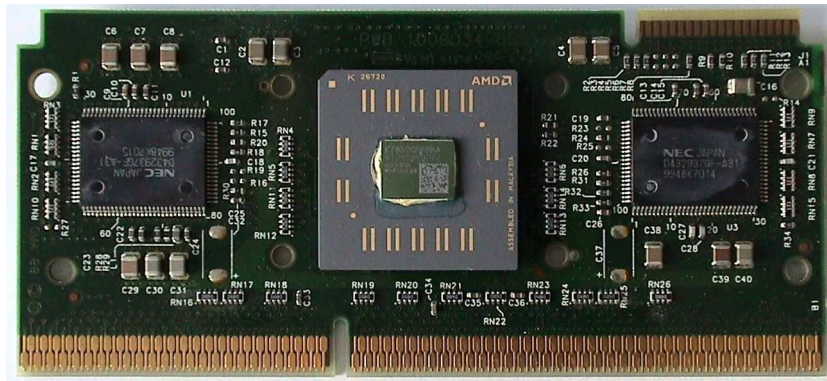
- L'Intel 8086 est un microprocesseur CISC 16 bits fabriqué par Intel en 1978, qui fait partie de la famille des x86.
- Il est basé sur des registres 16 bits, et dispose d'un bus externe lui aussi sur 16 bits.
- Il contient 29 000 transistors, est gravé en 3 μm et peut adresser 1 Mo de RAM.
- Sa puissance varie de 0.33 MIPS (lorsqu'il est cadencé à 4.77 MHz comme dans l'IBM PC) jusqu'à 0.75 MIPS pour la version 10 MHz.

Architecture x86 (merci Wikipédia)



Intel P4

Architecture x86 (merci Wikipédia)



AMD Athlon

- Quatre accumulateurs 16 bits : AX, BX, CX et DX
 - Peuvent être accédés sous formes de deux registres 8 bits e.g. AX $\rightarrow$ AH (High = poids fort) et AL (Low = poids faible)
- Pointeur d'instruction IP : $2^{16} = 64\text{Ko}$
 - D'où registres segments
- Pointeur de pile (SP : Stack Pointer)
- Pointeur de base (BP : Base Pointer)
- Index source (SI)
- Index destination (DI)
- Adressage mémoire sur 20 bits $\rightarrow$ 1 Mo (32 bits $\rightarrow$ 4 Go)
- Registres segments mémoire
 - Code Segment CS
 - Data Segment DS
 - Stack Segment SS
 - Extra EX

- Instructions de taille variable : de 1 à 6 octets
 - Premier octet : code opération
 - Suite : 0, 1, 2 ou 3 opérandes (sur 8 ou 16 bits)
- Exemples
 - B8 00 00 : met AX à 0
 - A1 01 40 : transfert contenu adresse 140 dans AX
 - A3 01 40 : transfert AX dans adresse 140
 - B8 00 00 : $AX = AX + \text{contenu adresse 140}$

Exemples

- B8 00 00 : met AX à 0
⇒ MOV AX,0
- A1 01 40 : transfère contenu adresse 140 dans AX
⇒ MOV AX,[140]
- A3 01 40 : transfère AX dans adresse 140
⇒ MOV [140],AX
- B8 00 00 : AX = AX + contenu adresse 140
⇒ ADD AX,[140]

- `<src>` : source (registre, mémoire ou valeur immédiate)
- `<dest>` : destination
- `<reg>` : registre
- e.g. `MOV <dest>, <src>`

Plus de détails dans le TD

Structure d'un ordinateur

Cours 6 : mémoire

Adrien Revault d'Allonnes

<http://www.allonnes.me>

Premier semestre 2009

- Mémoire : zone de stockage des programmes et données.
- Ensemble de cellules **adressables** : accessible par son adresse ($\neq$ registre).
- Mémoire de n octets : adresses de 0 à n-1.
- Formée de mots (1, 2, 4 ou 8 octets)
- Temps d'accès $\approx$ de 10^{-9} à 10^{-6} secondes

- 8 bits = 1 octet
- 1024 octets = 2^{10} octets = 1 kilo-octet = 1Ko
- 1024 Ko = 2^{20} octets = 1 méga-octet = 1Mo
- 1024 Mo = 2^{30} octets = 1 giga-octet = 1Go
- 1024 Go = 2^{40} octets = 1 téra-octet = 1To
- ...

- Les mémoires font parfois des erreurs
- Ajoût, à chaque mot mémoire, d'un code de contrôle

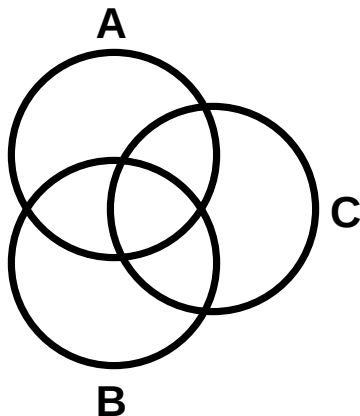
e.g.

- Mot mémoire : m bits de données et r bits de contrôle
- Longueur totale : $n = m + r \Rightarrow$ mot code de n bits
- Distance de Hamming : nombres de bits différents entre deux mots codes

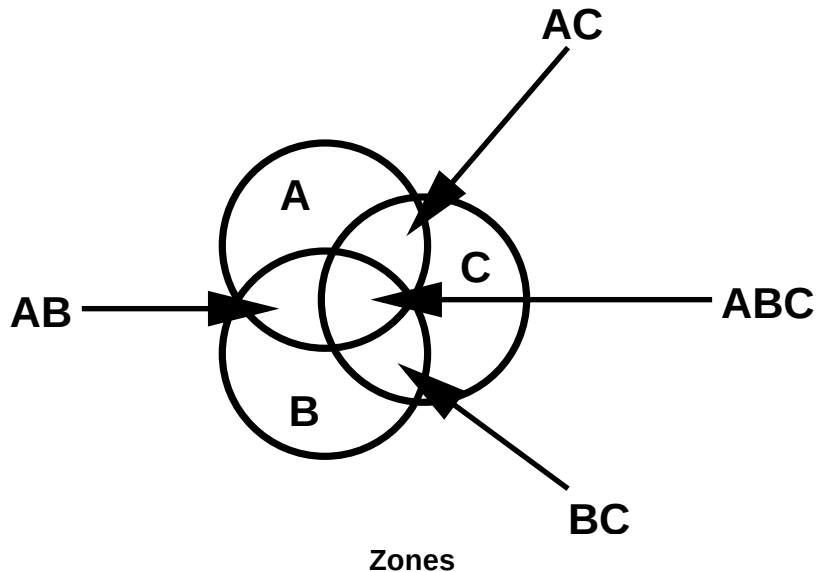
e.g. distance de Hamming entre 1000 1001 et 1011 0001 = 3

i.e. il faut trois bits en erreur pour passer de l'un à l'autre

- Soit un mot mémoire de longueur n (m bits de données et r bits de contrôle, $n=m+r$)
- Nombre de mots mémoires valides : 2^m
- Nombre total de mots mémoires : 2^n
- Distance de Hamming d'un code : distance minimale entre deux mots valides
- Un code avec une distance d peut détecter, au plus, $d-1$ erreurs
i.e. un mot avec d erreurs est – *peut être* – un mot valide

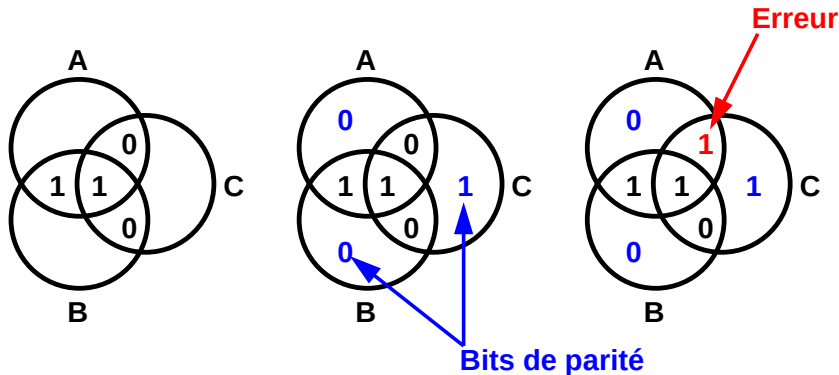


Codes autocorrecteurs - diagramme de Venn



Codes autocorrecteurs - diagramme de Venn

On écrit 1100 dans les zones AB, ABC, AC et BC



On ajoute un bit de parité dans chacune des zones A, B et C
Si une erreur d'un bit se produit, elle ne correspond qu'à une zone

Systèmes d'exploitation

Cours 7 : fonctions

Adrien Revault d'Allonnes

<http://www.allonnes.me>

Premier semestre 2009

- Gestion des ressources de l'ordinateur
- Initialisation puis fonctionnement machine
- À partir d'un noyau :
Bibliothèque de programmes en langage machine
- Point de vue du programmeur :
 - Bibliothèque d'appels de fonctions
- Point de vue du utilisateur, encapsulée par :
 - Langage de commandes
 - Interface graphique

- Gestion des programmes
- Gestion de la mémoire
- Système de fichiers
- Gestion des périphériques

Caractéristiques des systèmes d'exploitation

- Mono-tâche / multi-tâches

Mono-utilisateur / Multi-utilisateurs

- Gestion de la mémoire virtuelle

Swapping / paging

- Système de fichiers

Arborescence $\approx$ graphe

- Gestion des périphériques

- PC

- MS-DOS : Microsoft Disk Operating System
- Windows 98, 2000, XP
- Windows NT

- Mac

- MacOS 9
- MacOS X
- ...

- Unix

- SunOS
- Linux
- MacOS X
- ...

Sur un système multi-utilisateurs :

- Identification : login + mot de passe

Interaction avec l'OS :

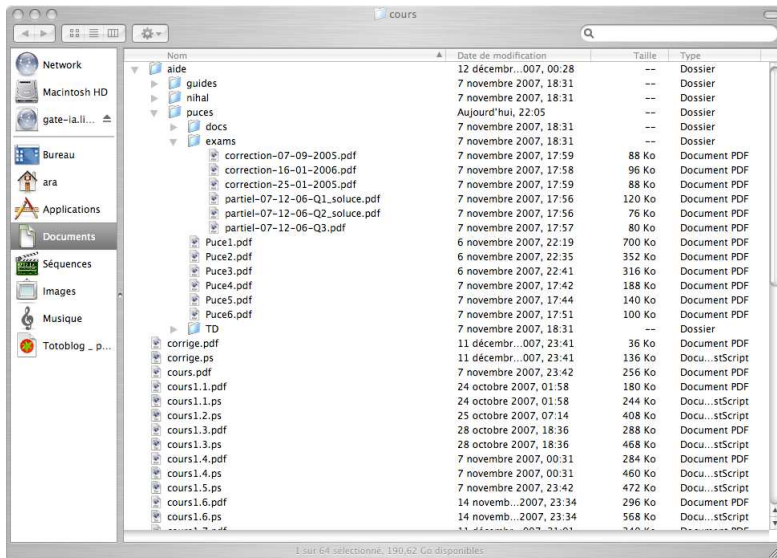
- Interface graphique (fenêtres, souris, ...)
- Interface textuelle (terminal, commandes, ...)

- Interface textuelle
 - Windows : bat (batch)
 - Unix : shell (bash)
- Interface graphique
 - Windows : 3.1, 98, XP
 - Unix : X-Window
 - MacOS X : Quartz

- Périphériques divers et variés
 - Écran, clavier, souris
 - Imprimante, scanner, ...
 - Disques, clés USB, ...
- Gestionnaire : driver

- Fichier : regroupement (stockage) de données de même nature
- Identification (par l'utilisateur) : nom & emplacement
- Système fichiers : organisation et méthode d'accès aux données enregistrées

Système de fichiers



- Répertoire : noeud interne

- Racine Unix : /
- Racine Windows : C\

- Fichier : feuille

- Nom complet : chemin de la racine à la feuille

e.g. /Users/ara/Documents/cours/aide/puces/exams/correction-07-09-2005.pdf

- La structure interne du fichier correspond au type de données
- On associe par convention une extension (.□ □ □) à une structure
e.g.
 - mp3, ... : son compressé
 - mpg, avi, ... : animation, film
 - txt : texte brut
 - jpg, gif, ... : images
- On peut en général aussi spécifier des informations comme :
 - Dates : création, modification, dernière ouverture, ...
 - Propriétaire et droits : contexte multi-utilisateurs

- Processus : tâche, programme en cours d'exécution
- Ressources nécessaires : UC, mémoire, périphériques, ...
- État d'un processus : CO, SP, registres, ...
- Le système d'exploitation gère :
 - l'allocation de ressources aux processus
 - la création, la fermeture de processus
 - la suspension, reprise des processus
 - la synchronisation entre processus
- Des processus peuvent s'exécuter en parallèle :
 - Simultanéité réelle : multiprocesseurs
 - Simultanéité apparente : exécution entrelacée

Processus : gestion de la mémoire

Elle est le rôle du système d'exploitation.

Il doit :

- savoir quels processus utilisent quelles parties de la mémoire
- savoir quels processus demandent de la mémoire et combien
- allouer de la mémoire quand elle est disponible
- libérer la mémoire

La mémoire principale ne peut pas contenir tous les processus actifs.

La mémoire secondaire (disques) est, en général, utilisée pour stocker provisoirement les processus inactifs.

La mémoire principale et secondaire forment l'unité logique appelée mémoire virtuelle.

Le système d'exploitation doit donc gérer ensemble la RAM et la mémoire disque.

Deux types de gestion de la mémoire virtuelle :

- Mécanisme de va-et-vient (swapping)
 - Processus inactif placé en mémoire disque
 - Ses ressources sont utilisées par un autre
- Mécanisme de demande de pages
 - Processus sont découpés en pages mémoire
 - L'OS swap les pages

Systèmes d'exploitation

Cours 8 : l'exemple d'UNIX

Adrien Revault d'Allonnes

<http://www.allonnes.me>

Premier semestre 2009

- Désigné par un nom
- Caractéristiques :
 - Type
 - Droits d'accès
 - Taille
 - Dates
 - ...
- Types :
 - Fichier : -
 - Répertoire : d
 - Lien symbolique : l

e.g.

```
-rw-rw-rw- 1 ara ara 36905 Dec 13 01:24 slides.tex
drwxrwxrwx 6 ara ara 204 Dec 12 00:28 aide
-rw-r--r-- 1 ara ara 337532 Dec 12 00:12 cours.pdf
```

Fichiers sous Unix : droits

Chaque fichier a :

- Un propriétaire
- Des droits :
 - Lecture (read : r)
 - Écriture (write : w)
 - Exécution (execute : x)

⇒ Spécifiés pour :

- Le propriétaire
- Les membres du groupe du propriétaire
- Le reste du monde

e.g.

```
-rw-rw-rw- 1 ara ara 36905 Dec 13 01:24 slides.tex
drwxrwxrwx 6 ara ara 204 Dec 12 00:28 aide
-rw-r--r-- 1 ara ara 337532 Dec 12 00:12 cours.pdf
```

Types de processus

- Processus système : appartiennent au super-utilisateur root
- Démons : permanents en veille (e.g. inetd démon telnet)
- Processus utilisateur : lancés par l'utilisateur

Identification de processus (commande ps [-options])

- Identifiant attribué par SE à la création : pid (process identifier)
- Propriétaire effectif : à qui appartient le programme
- Propriétaire réel : qui exécute le programme
- Un programme peut avoir plusieurs processus

Interruption de processus

- Signal au programme (e.g. CTRL Q, CTRL C, ...)
- Depuis le terminal : envoi d'un signal `kill [-signal] pid`

- Processus communique par des canaux :
 - Lecture entrée standard (`stdin`) : clavier
 - Écriture sortie standard (`stdout`) ou erreur (`stderr`) : écran
- Redirection de canaux :
 - `echo "bonjour" > test` :
écrit la chaîne 'bonjour' dans le fichier 'test'
 - `cat < test` : écrit le contenu de 'test' à l'écran :
bonjour
- Chaînage de processus :
 - `ls -l | grep zip | wc -l : ?`

Commandes de base

- `ls [-options] [<répertoire>]` : affiche contenu d'un répertoire (courant, par défaut)
- `cd [<répertoire>]` : change répertoire (home, par défaut)
- `mkdir <répertoire>` : crée un répertoire
- `pwd` : affiche chemin absolu du répertoire courant
- `mv <file1> <file2>` : déplace un fichier
- `cp [-options] <src> <dest>` : copie un fichier
- `rmv <file>` : supprime un fichier
- `cat [-options] [<file>]` : lit un fichier vers la sortie standard
- `grep [-options] patron [<file>]` : cherche le patron dans un fichier et écrit les lignes le contenant sur stdout
- `man <commande>` : affiche la doc de la commande

Systèmes d'exploitation

Cours 9 : programmation shell

Adrien Revault d'Allonnes

<http://www.allonnes.me>

Premier semestre 2009

Programmation shell : caractères spéciaux

- `?` : n'importe quel caractère non vide
- `*` : n'importe quel **s** caractère **s**
- `"` : délimite une chaîne protégée
i.e. seuls les variables et les métacaractères (`*`, `?`) sont interprétés
- `'` : délimite une chaîne constante
i.e. même protection que chaînes protégées sans interprétation
- `'` : délimite une commande shell qui est interprétée
i.e. le résultat de la commande est affecté à la variable
- `\` : caractère de protection de n'importe quel caractère
e.g.
 - `\*` : caractère `*` et non méta-caractère `*`
 - `\\` : caractère `\`
- `#` : commentaires (dièse + espace)
- `#!/bin/sh` : première ligne d'un script, chemin du shell à utiliser

Programmation shell : variables spéciales

- `$#` : nombre de paramètres de la ligne de commande
- `$n` ($0 \leq n \leq 9$) valeur du nième paramètre
 - `$0` : nom de la commande (pas comptabilisé dans `$#`)
 - `$n` : chaîne vide si `$# < n`
- `$@` : l'ensemble des arguments
- **shift** [*n*] : décale les paramètres de *n* places à gauche (défaut 1)
Attention : comportement non standard si le nombre de paramètres n'est pas suffisant

Syntaxe d'une commande de test (**mots clefs**) :

```
if [ test ]  
then  
    bloc d'instructions  
elif [ test ]  
then  
    bloc d'instructions  
else  
    bloc d'instructions  
fi
```

Les tests suivants sont possibles

Tests d'expressions numériques

Test	Condition
$e1 < e2$	$e1 -lt e2$
$e1 > e2$	$e1 -gt e2$
$e1 \leq e2$	$e1 -le e2$
$e1 \geq e2$	$e1 -ge e2$
$e1 = e2$	$e1 -eq e2$
$e1 \neq e2$	$e1 -ne e2$

Tests d'expressions alphanumériques

Test	Condition
ch1 = ch2	ch1 = ch2
ch1 $\neq$ ch2	ch1 != ch2
ch1 vide	-z ch1
ch1 est le nom d'un fichier	-f ch1
ch1 est le nom d'un répertoire	-d ch1
négation de condition	! condition
ou	[cond1] [cond2]
et	[cond1] && [cond2]

Tests

ATTENTION : l'espace entre les éléments est obligatoire.

`[$x=$y]` ne marchera pas alors que `[ $x = $y ]` est correct.

Préférer `[ "$x" = "$y" ]` : si les variables n'ont pas de valeurs "" est interprété comme la chaîne vide par shell.

Boucle for :

```
for var in liste de valeurs  
do  
    bloc d'instructions  
done
```

Boucle while :

```
while [ test ]  
do  
    bloc d'instructions  
done
```

Boucle repeat :

```
until [ test ]  
do  
    bloc d'instructions  
done
```

Programmation shell : boucles

Test valeurs variable :

```
case var in
    profil) bloc d'instructions;;
esac
```

Où "profil" est une séquence de caractères pouvant contenir des méta-caractères comme "*" ou "?", e.g.

```
case $# in
    1) echo "1 variable" ;;
    2) echo "2 variables" ;;
    *) echo "0 ou plus de 2 variables" ;;
esac
```

Faites un script qui vous dise bonjour en affichant votre login :

- Quand vous tapez `saluer`
- Quand vous tapez `saluer [votre prénom]`

Faites un script qui vous dise bonjour en affichant votre login :

- Quand vous tapez sa luer
- Quand vous tapez sa luer [votre prénom]

```
#!/bin/sh
if [ -n "$1" ]; then LOGIN=$1; else LOGIN='whoami';
fi
echo "Bonjour, $LOGIN"
```

Faire un script qui affiche la phrase :

Le script [params] a [n] paramètre(s), que voici :
[paramètres]

en remplaçant les mots entre crochets par leur valeur, e.g.

```
ara ~$ ./monscript bla ble bli blo blu
```

Le script monscript a 5 paramètre(s), que voici :
bla ble bli blo blu

Faire un script qui affiche la phrase :

Le script [params] a [n] paramètre(s), que voici :
[paramètres]

en remplaçant les mots entre crochets par leur valeur, e.g.

```
ara ~$ ./monscript bla ble bli blo blu
```

Le script monscript a 5 paramètre(s), que voici :
bla ble bli blo blu

```
#!/bin/sh
```

```
echo "Le script $0 a $# paramètre(s), que voici :  
$@"
```

Créez une commande qui, lorsqu'elle est appelée, renvoie le nombre d'arguments qui lui ont été fournis, ainsi que le premier de ces arguments.

Créez une commande qui, lorsqu'elle est appelée, renvoie le nombre d'arguments qui lui ont été fournis, ainsi que le premier de ces arguments.

```
#!/bin/sh  
echo "J'ai reçu $# arguments"  
echo "Le premier est : $1"
```

Réseau

Cours 10 : architecture en couches

Adrien Revault d'Allonnes

<http://www.allonnes.me>

Premier semestre 2009

Communication : historique

- 1810 – Samuel Thomas van Soemmaring
Télégraphe à bulles (26 fils)
- 1816 – Francis Ronalds
Première expérience d'un télégraphe électrique
- 1837 – William Cooke, Charles Wheatstone
Télégraphe électrique (Angleterre)
- 1837 – Samuel Morse, Alfred Vail
Télégraphe électrique (États-Unis)

- 1850 – William Thomson (Lord Kelvin)
Développement technique du câble sous-marin
- 1866 – William Gisborne, Cyrus Field
Premier câble transatlantique fonctionnel
- 1876 – Alexandre Graham Bell
Téléphone (prise de brevet)
- 1897 – Guglielmo Marconi
Prise de brevet et transmission télégraphique sans fil sur 16 km

Communication : historique

- 1901 – Guglielmo Marconi
Première transmission télégraphique sans fil à travers l'Atlantique
- 1906 – Reginald Fessenden
Première transmission sans fil de la parole à travers l'Atlantique dans les deux sens
- 1907 – Édouard Belin
Première transmission d'une photographie
- 1915 – Compagnie Marconi
Développement à grande échelle de la radio parlée
- 1923 – John Baird
Première télévision mécanique

- Connexion de plusieurs machines entre elles
 - utilisateurs et applications échangent des informations
- Le terme réseau en fonction de son contexte peut désigner :
 - l'ensemble des machines
 - l'infrastructure informatique d'une organisation avec les protocoles qui sont utilisés
 - e.g. internet
 - la façon dont les machines d'un site sont interconnectées
 - e.g. Ethernet
 - e.g. Token Ring
 - e.g. réseau en étoile
 - e.g. réseau en bus
 - le protocole qui est utilisé pour que les machines communiquent
 - e.g. TCP/IP
 - e.g. NetBeui (protocole Microsoft)
 - e.g. DecNet (protocole DEC)
 - e.g. IPX/SPX

Lorsque l'on parle de réseau, il faut bien comprendre le sens du mot.

Pour faire circuler l'information sur un réseau on peut utiliser principalement deux stratégies :

- L'information est envoyée de façon complète
 - risques d'erreurs
 - problèmes sous-jacents trop complexes à résoudre

⇒ pas utilisée
- L'information est fragmentée en petits morceaux (paquets)
 - chaque paquet est envoyé séparément sur le réseau
 - les paquets sont réassemblés sur la machine destinataire

⇒ On parle réseau à **commutations de paquets**

Modèle de référence : OSI de chez ISO

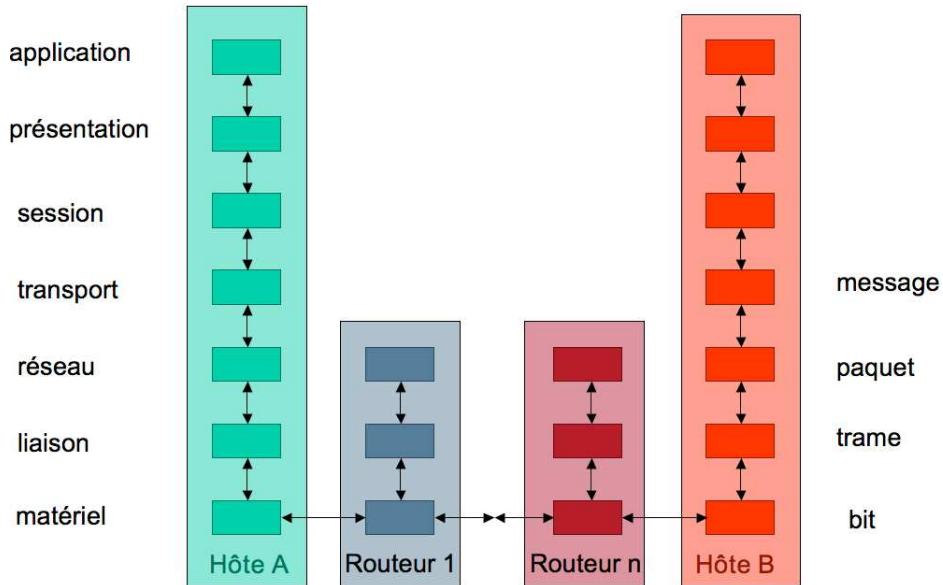
Couche	Fonctionnalité
7	Application
6	Présentation
5	Session
4	Transport
3	Réseau
2	Liaison
1	Matériel

Open System Interconnection : norme ISO 7498

ou

Modèle basique de référence pour
l'interconnexion des systèmes ouverts

Modèle de référence : OSI de chez ISO



OSI : la couche 1 – Matériel

S'occupe des problèmes strictement matériels (support physique pour le réseau), on doit également préciser toutes ses caractéristiques.

- Pour du câble :
 - Type (coaxial, paires torsadées,...)
 - si un blindage est nécessaire
 - le type du signal électrique envoyé (tension, intensité,...)
 - nature des signaux (carrés, sinusoïdaux,...)
 - limitations (longueur, nombre de stations,...)
 - ...
- Pour des communications hertziennes :
 - Fréquences
 - Type de modulation (Phase, Amplitude,...)
 - ...
- Pour de la fibre optique :
 - Couleur du laser
 - Section du câble
 - Nombre de brins
 - ...

- **La couche 2 Liaison**

Dans cette couche on cherche à savoir comment deux stations sur le même support physique (cf. couche 1) vont être identifiées. Pour ce faire, on peut par exemple assigner à chaque station une adresse (cas des réseaux Ethernet,...)

- **La couche 3 Réseau**

Le rôle de cette couche est de trouver un chemin pour acheminer un paquet entre 2 machines qui ne sont pas sur le même support physique.

- **La couche 4 Transport**

La couche transport doit normalement permettre à la machine source de communiquer directement avec la machine destinataire. On parle de communication de bout en bout (end to end).

- **La couche 5 Session**

Cette couche a pour rôle de transmettre cette fois les informations de programmes à programmes.

- **La couche 6 Présentation**

A ce niveau on doit se préoccuper de la manière dont les données sont échangées entre les applications.

- **La couche 7 Application**

Dans la couche 7 on trouve normalement les applications qui communiquent ensemble. (Courrier électronique, transfert de fichiers,...)

Caractéristiques des réseaux

- Type des connexions

- point à point
- multipoint

- Topologie

- maillage
- bus
- anneau
- étoile
- ...

- Taille du réseau

- LAN
- MAN
- WAN
- Internet

- Liaison (connexion) point à point

un canal est dédié spécifiquement à la connexion de deux machines

- Réseau point à point

ensemble de liaisons point à point

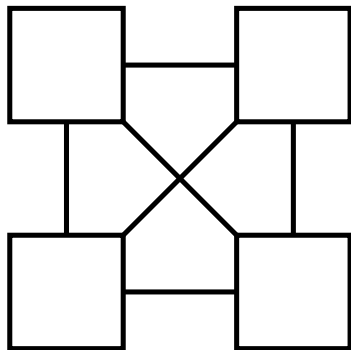


FIG.: Câblage en maille

Chaque machine est reliée à toutes les autres par un câble (obsolète)

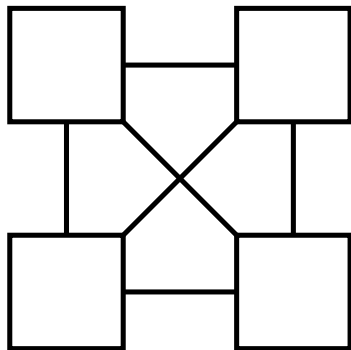


FIG.: Câblage en maille

Chaque machine est reliée à toutes les autres par un câble (obsolète)
Le câblage en maile n'est plus utilisé car trop coûteux en câble

Architectures des réseaux : point à point

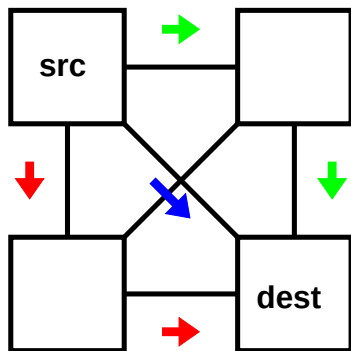


FIG.: Câblage en maille

Chaque machine est reliée à toutes les autres par un câble (obsolète)
Problèmes du routage

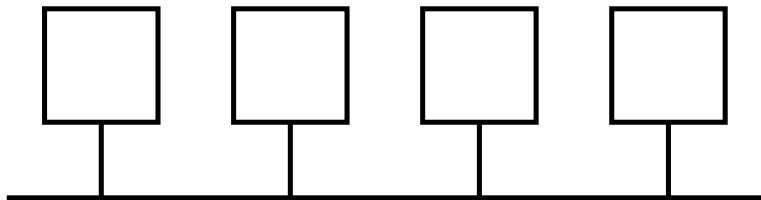


FIG.: Câblage en bus

Chaque machine est reliée à un câble appelé bus (e.g. Ethernet)

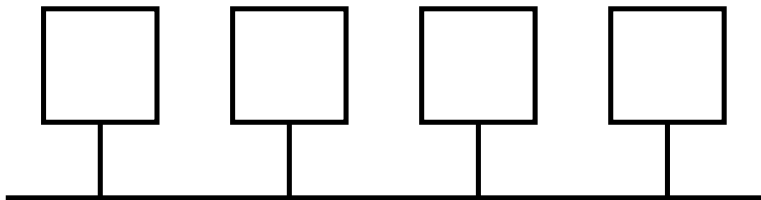


FIG.: Câblage en bus

- Par son architecture le câblage en bus avec des protocoles CSMA/CD convient mal à un environnement temps réel
- Deux machines peuvent monopoliser le câble

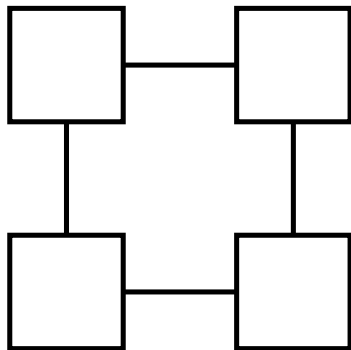


FIG.: Câblage en anneau

Chaque machine est reliée à une autre pour former un anneau

- Architecture en anneau avec un protocole à base de jeton : le délai maximum pour transmettre une information entre 2 machines peut être calculé
⇒ peut servir dans un environnement temps réel
- Le câblage en anneau nécessite plus de câble puisqu'il faut reboucler la dernière machine sur la première.
- Le câblage en anneau peut être perturbé par la panne d'une seule machine.

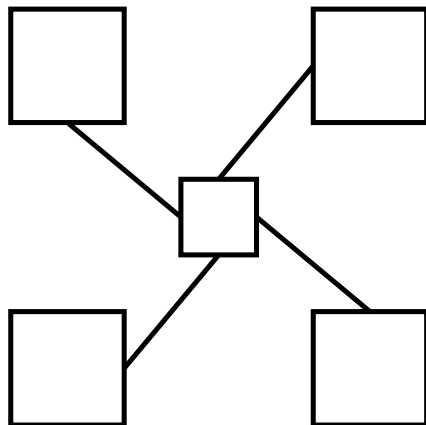


FIG.: Câblage en étoile

Chaque machine est reliée par un câble à un appareil actif

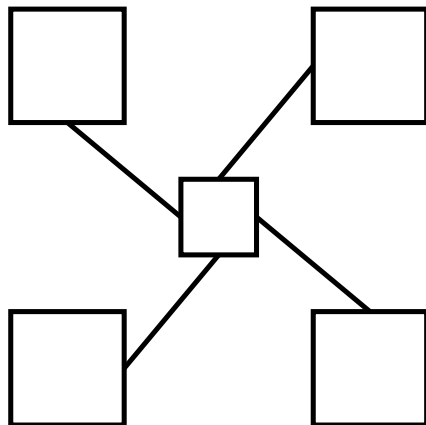


FIG.: Câblage actuel

Point faible : centre de l'étoile. S'il tombe tout le réseau est paralysé

- LAN (Local Area Network)
- Liaison multipoint
 - bus
 - anneau
 - étoile
- Dimension : jusque quelques kilomètres
- Taux d'erreur : peu élevé

- Ethernet (IEEE 802.3)
10 Mb/s - 100 Mb/s
- Token Ring (IEEE 802.5)
4 Mb/s - 16 Mb/s
- Token bus (IEEE 802.4)
- FDDI (Fiber Distributed Data Interface)
100 Mb/s

- MAN (Metropolitan Area Network)
- Fédération de réseaux locaux
- Deux bus unidirectionnels
 - artères à gros débit
 - épine dorsales (backbones)
- Dimension : un campus, une ville
- Débit : 64 Kb/s à 155,52 Mb/s

- WAN (Wide Area network)
- Constitution :
 - hôtes
 - routeurs
- Dimension : un pays, un continent
- Taux d'erreur : relativement élevé

- WAN (Wide Area network)
- Constitution :
 - hôtes
 - routeurs
- Dimension : un pays, un continent
- Taux d'erreur : relativement élevé
- Interconnected Networks
- Interconnexion de réseaux

TCP/IP vs. OSI

Fonctionnalité TCP/IP	Couche	Fonctionnalité
Application	7	Application
	6	Présentation
Transport	5	Session
	4	Transport
Interface avec le réseau	3	Réseau
Matériel		
	1	Matériel

Base des Systèmes Informatiques

Adrien Revault d'Allonnes

<http://www.allonnes.me>

Premier semestre 2009

Sources et remerciements

Ce cours a été établi à partir de deux sources clés :

- **'Architecture de l'ordinateur'**
d'**Andrew Tanenbaum**
Pearson éducation, 5ème édition, d'abord
- **'De la puce au web'**
Cours d'**Émmanuel Chailloux**
Laboratoire d'Informatique de Paris VI, surtout

La dernière partie, traitant de réseaux, s'inspire copieusement des cours de :

- **Bruno Péan**, pour le fond
École Internationale des Sciences du Traitement de l'Information
- **Christophe Lecoutre**, pour sa clarté
Centre de Recherche en Informatique de Lens, IUT de Lens
- **Jean-Luc Dekeyser**, pour l'adresse de Bruno Péan, notamment
Laboratoire d'Informatique Fondamentale de Lille

et bien sûr, Wikipedia