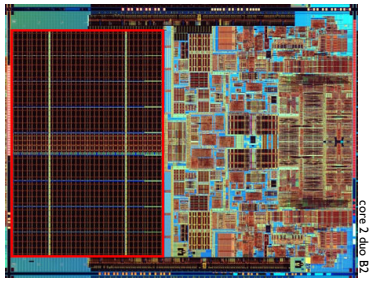
Mise à jour: Janvier 2012

Architecture des Ordinateurs

[Archi/Lycée]



Nicolas Bredèche

Maître de Conférences
Université Paris-Sud

bredeche@lri.fr

UNIVERSITÉ PARIS SUD

Ressources bibliographiques utilisées pour ce cours :

- http://www.lri.fr/~cecile/ENSEIGNEMENT/POLYARCHI/
- http://www.lri.fr/~cecile/ENSEIGNEMENT/IFIPS/IFIPSA/Cours/
- http://www.lri.fr/~temam/enseignement/x/

- http://cas.ee.ic.ac.uk/people/nachiket/teaching/
- http://people.ee.duke.edu/~sorin
- http://www.csee.umbc.edu/~younis/CMSC611/

- Computer Architecture: A Quantitative Approach (Hennessy, Patterson)
- Architecture des ordinateurs (Schwarz), Eyrolles, 2005.
- Architecture et technologie des ordinateurs. (Zanella, Ligier). Dunod, 2005.
- Architecture des machines et des systèmes informatiques. (Cazes, Delacroix), 2011.

Et aussi: http://www.wikipedia.fr/ et http://www.commentcamarche.net/

Objectif: de la mécanique au fonctionnel

source: T. Dumartin (note de cours)

Objectif du module

- Acquérir un...
  - Savoir
    - comprendre comment est représentée l'information
    - comprendre comment est fait un ordinateur (matériel/logiciel)
    - comprendre comment un réseau informatique fonctionne
    - comprendre comment programmer un robot
  - Savoir faire
    - manipuler l'information (codage, calcul)
    - dessiner (sur papier) un circuit logique, et l'optimiser
    - programmer en assembleur simplifié
    - programmer un protocole de communication simplifié
    - programmer un comportement simple de robot

Plan du module

- Intervenants
  - Chargé de cours: Nicolas Bredèche
  - Chargé de TP: Nicolas Galichet et Jean-Marc Montanier
  - Contact: prenom.nom@lri.fr
- Organisation
  - Cours 1 : Architecture - représentation de l'information
  - Cours 2 : Architecture - conception matérielle
  - Cours 3 : Architecture - conception logicielle
  - Cours 4 : Réseaux informatiques
  - Cours 5 : Robotique
- Evaluation: TPs (sauf 1er TP)

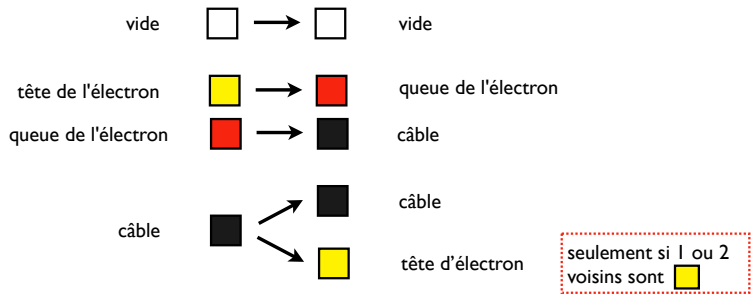
Plan du Cours #1

- Architecture des ordinateurs
  - introduction générale
  - représentation de l'information
    - encodage
    - changement de représentation
    - erreur d'arrondi
  - fonction logique
    - portes logiques
    - optimisation des portes

Une petite digression

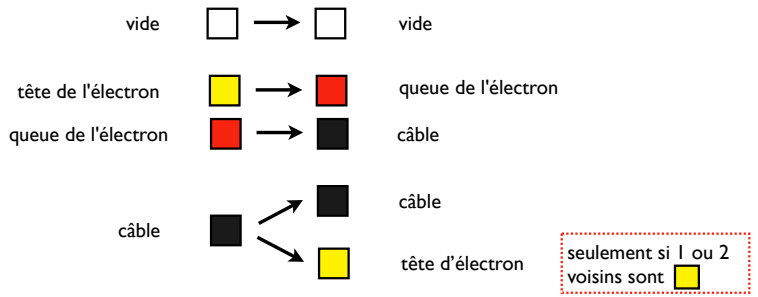
WireWorld

[Silverman, 1987]

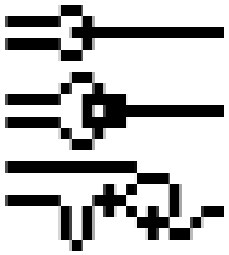


WireWorld

[Silverman, 1987]



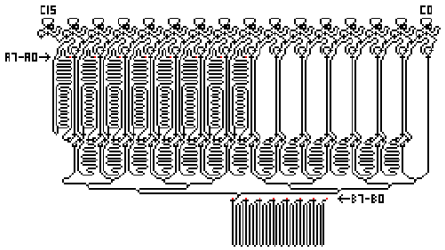
EXERCICE: 1. exécuter les trois versions de ce montage
2. à quoi ce montage peut il servir?



porte OR

porte XOR

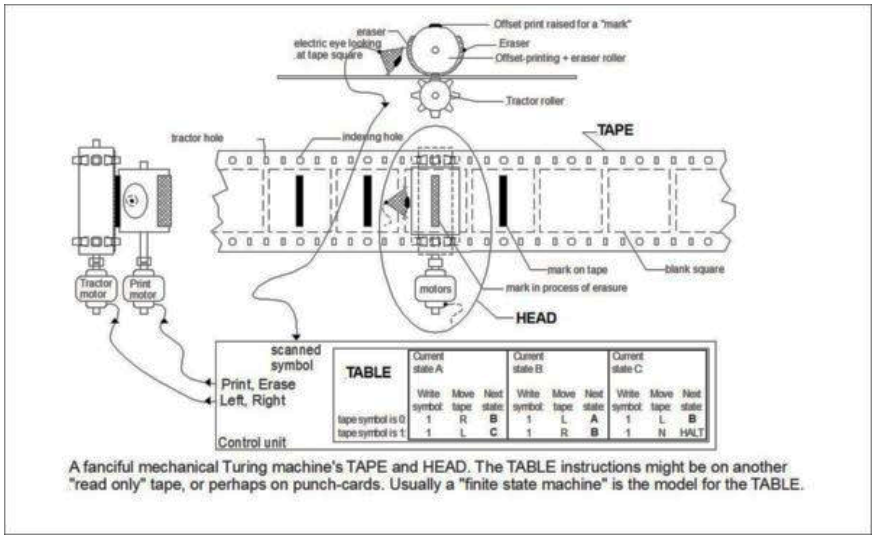
porte AND



Multiplication de deux nombres 8 bits dans WireWorld
[Gardner, 2002]

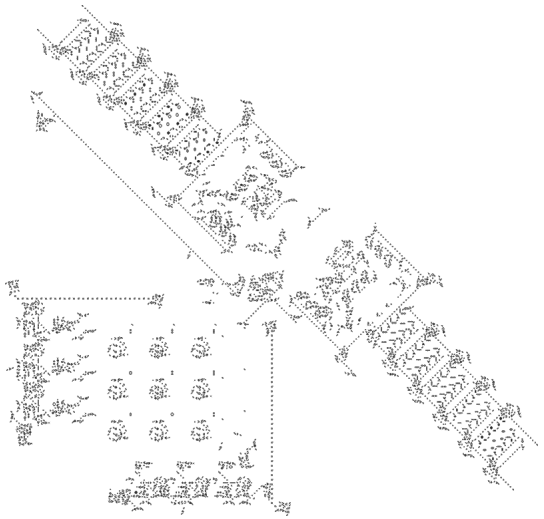
http://mathworld.wolfram.com/WireWorld.html

L'ordinateur fonctionne dans un monde <u>discret</u>, sur une base <u>binaire</u>

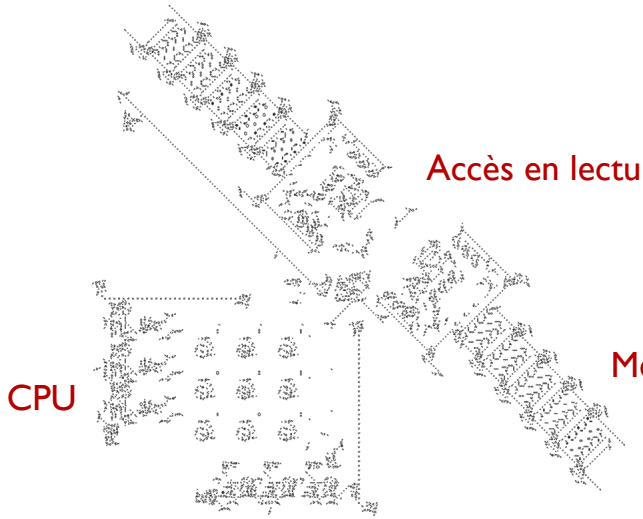


A fanciful mechanical Turing machine's TAPE and HEAD. The TABLE instructions might be on another "read only" tape, or perhaps on punch-cards. Usually a "finite state machine" is the model for the TABLE.

Machine de Turing (ou Automate de Turing)
(1936)



une machine de Turing dans le jeu de la vie
(design par P. Rendell, 2002)

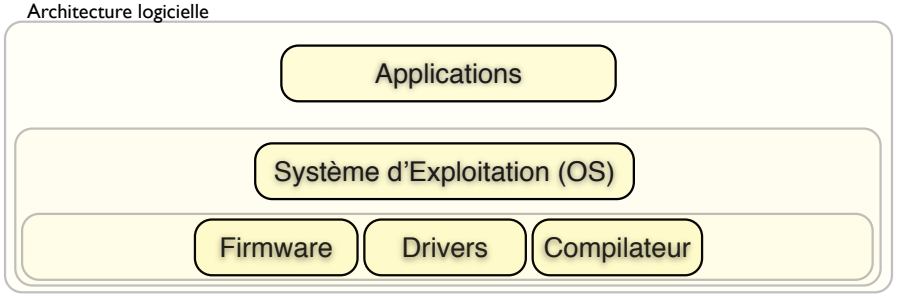


une machine de Turing dans le jeu de la vie
(design par P. Rendell, 2002)

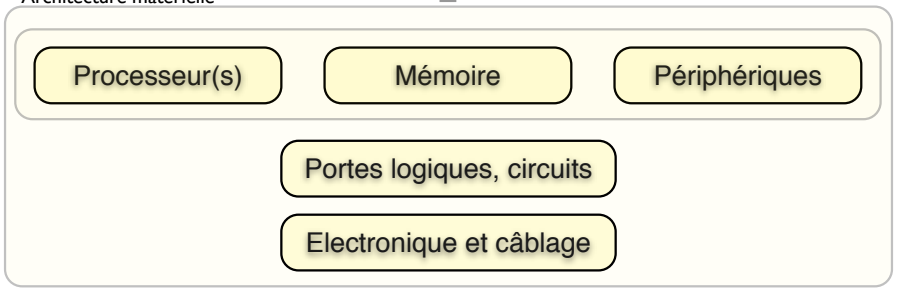
Un ordinateur est constitué de plusieurs
modules spécialisés qui interagissent entre eux

Introduction

Architecture logicielle



Architecture matérielle



Architecture logicielle

Applications

Système d'Exploitation (OS)

Firmware Drivers Compilateur

Objet de ce module:

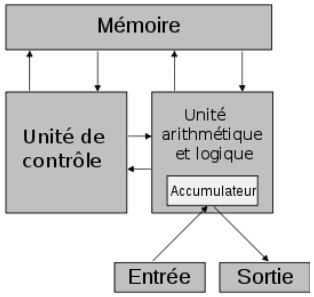
Architecture matérielle

Processeur(s) Mémoire Périphériques

Portes logiques, circuits

Electronique et câblage

Architecture de von Neumann



L'**architecture de von Neumann** décompose l'ordinateur en 4 parties distinctes

1. L'**unité arithmétique et logique** (UAL ou ALU en anglais) ou unité de traitement : son rôle est d'effectuer les opérations de base ;
2. L'**unité de contrôle**, chargée du séquençage des opérations ;
3. La **mémoire** qui contient à la fois les données et le programme qui dira à l'unité de contrôle quels calculs faire sur ces données. On distingue:
   a. la mémoire volatile (programmes et données en cours de fonctionnement)
   b. la mémoire permanente (programmes et données de base de la machine).
4. Les dispositifs d'**entrée-sortie**, qui permettent de communiquer avec le monde extérieur.

source: wikipedia

Architecture de von Neumann

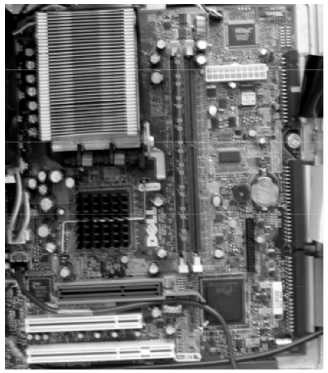
On parle ici du modèle d'architecture...

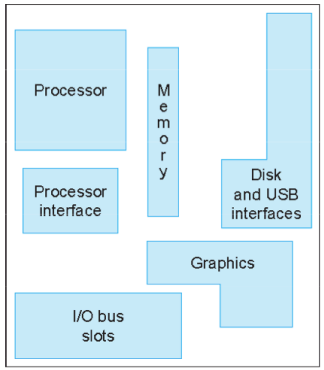
- de votre ordinateur personnel
...
- de votre téléphone portable
- de votre appareil photo numérique
...
- du GPS de votre voiture
- de votre voiture (de son ordinateur de bord)
- et sinon, du panneau indiquant le temps au prochain bus
...
- de votre machine à laver, de votre four micro-ondes
- de votre futur compteur électrique
...

a. la mémoire volatile (programmes et données en cours de fonctionnement)
b. la mémoire permanente (programmes et données de base de la machine).
4. Les dispositifs d'entrée-sortie, qui permettent de communiquer avec le monde extérieur.

source: wikipedia

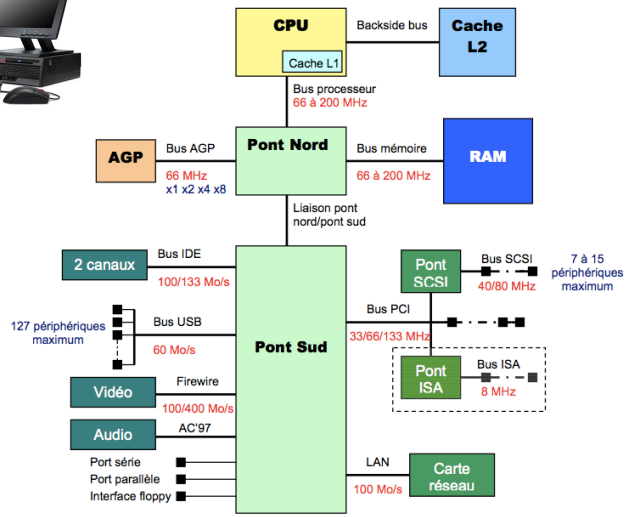
Carte mère PC
«motherboard»





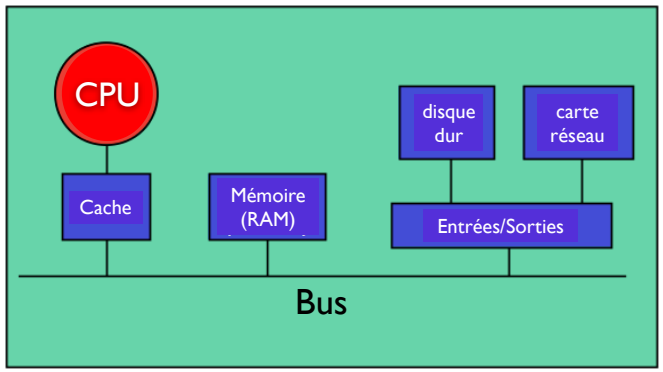
source: http://www.csee.umbc.edu/~younis/CMSC611

Exemple typique



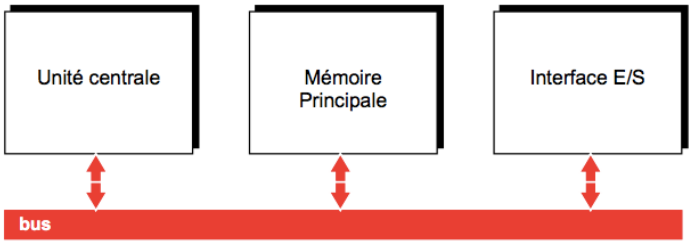
source:T. Dumartin (note de cours)

Vue d'ensemble



traduit de: Daniel Sorin, Duke University

les bus



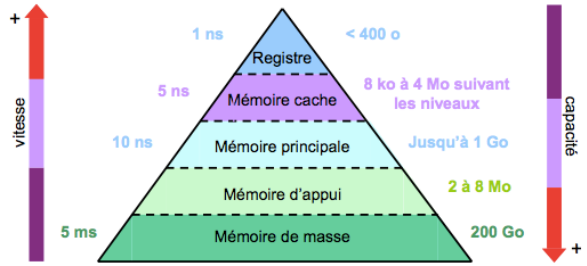
Bus d'adresse
permet la sélection de l'emplacement mémoire à lire/écrire

Bus de données
permet l'écriture/lecture des informations en mémoire

Bus de commande
permet de synchroniser les flux d'information entre les deux bus précédents

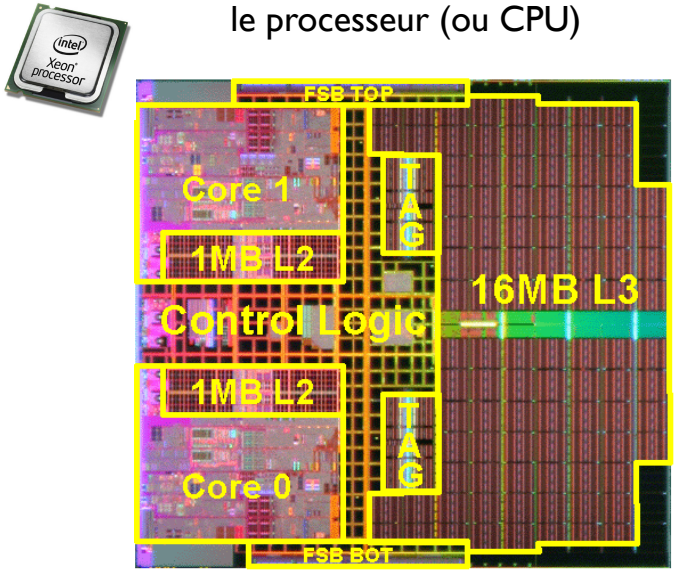
source:T. Dumartin (note de cours)

la mémoire



source: T. Dumartin (note de cours)

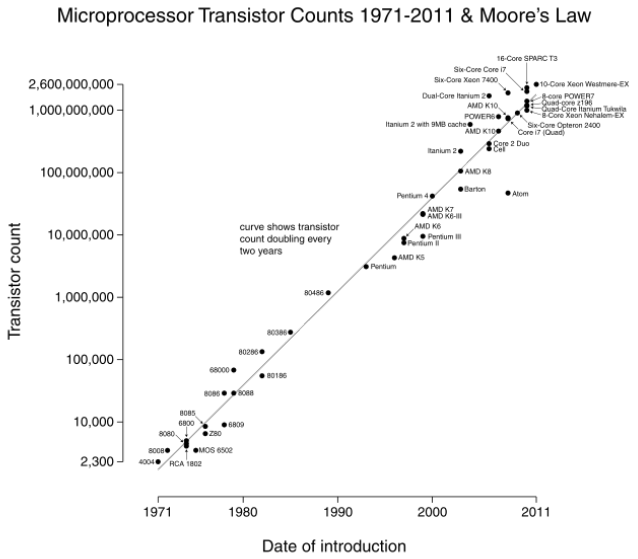
le processeur (ou CPU)



Intel 65nm Xeon MP processor «Tulsa» (2006)

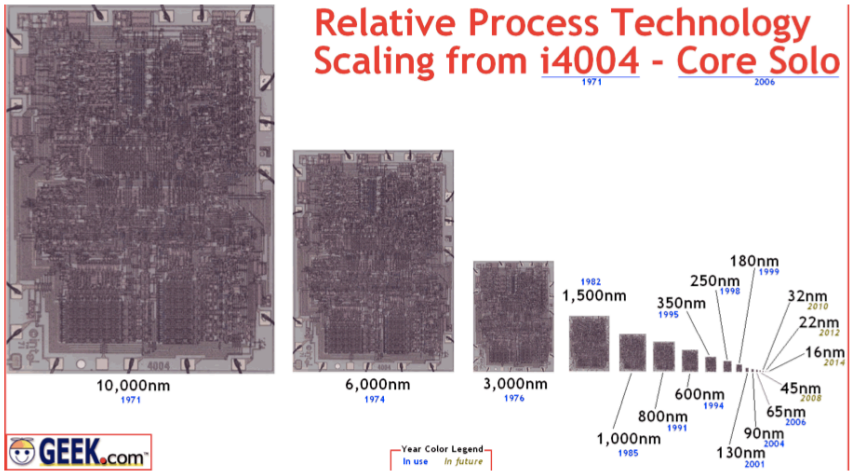
source: http://www.realworldtech.com/page.cfm?ArticleID=RWT021906030756

évolution du nombre de composants



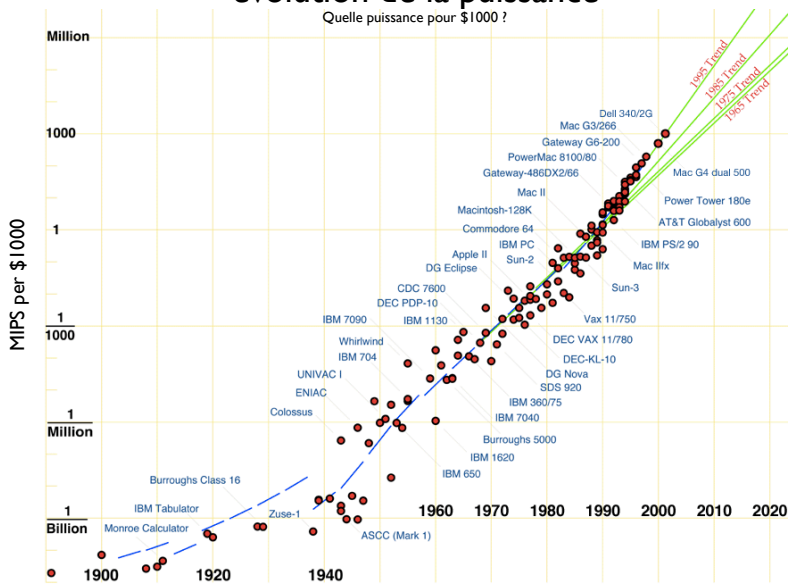
source: wikipedia

évolution de la place occupée



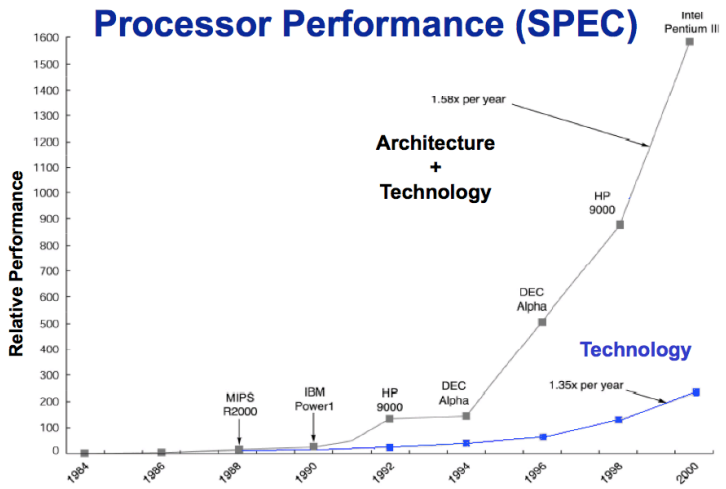
source: MIT course (Nachiket Kapre)

évolution de la puissance



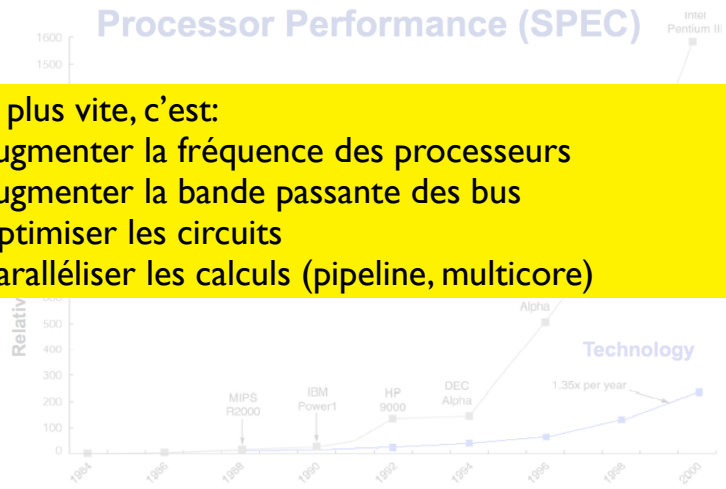
source: http://www.mocom2020.com/2009/05/evolution-of-computer-capacity-and-costs/

Quelles sont les causes de l'augmentation de la puissance ?



source: David Paterons (depuis: M.Younis, CMCS 611)

Quelles sont les causes de l'augmentation de la puissance ?



Aller plus vite, c'est:
- augmenter la fréquence des processeurs
- augmenter la bande passante des bus
- optimiser les circuits
- paralléliser les calculs (pipeline, multicore)

source: David Paterons (depuis: M.Younis, CMCS 611)

- Concevoir l'architecture des ordinateurs:
  - Compétence en ingénierie («computer engineering»)
  - Compétence en informatique («computer science»)
- Objectifs:
  - Aller plus vite...
  - En restant (si possible) compatible avec l'existant

Portes logiques

comment calculer?

porte logique «universelle» 1/2

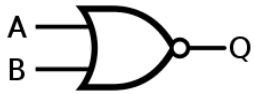
NAND

$$\overline{A \cdot B}$$

INPUT		OUTPUT
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

source: wikipedia

porte logique «universelle» 2/2



NOR

$$\overline{A + B}$$

INPUT		OUTPUT
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

source: wikipedia

Type	Distinctive shape	Rectangular shape	Boolean algebra between A & B	Truth table
AND			$A \cdot B$	INPUT A B / OUTPUT A AND B: 0 0 0; 0 1 0; 1 0 0; 1 1 1
OR			$A + B$	INPUT A B / OUTPUT A OR B: 0 0 0; 0 1 1; 1 0 1; 1 1 1
NOT			$\overline{A}$	INPUT A / OUTPUT NOT A: 0 1; 1 0

source: wikipedia

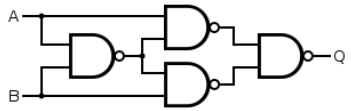
Type	Distinctive shape	Rectangular shape	Boolean algebra between A & B	Truth table
AND			$A \cdot B$	INPUT A B / OUTPUT A AND B: 0 0 0; 0 1 0; 1 0 0; 1 1 1
OR			$A + B$	INPUT A B / OUTPUT A OR B: 0 0 0; 0 1 1; 1 0 1; 1 1 1
NOT			$\overline{A}$	INPUT A / OUTPUT NOT A: 0 1; 1 0

exercice:

INPUT A	INPUT B	OUTPUT A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

?

source: wikipedia



...On peut aussi le faire avec des portes ET et OU

INPUT A	INPUT B	OUTPUT A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

source: wikipedia

NAND			$\overline{A \cdot B}$	INPUT A B / OUTPUT A NAND B: 0 0 1; 0 1 1; 1 0 1; 1 1 0
NOR			$\overline{A + B}$	INPUT A B / OUTPUT A NOR B: 0 0 1; 0 1 0; 1 0 0; 1 1 0
XOR			$A \oplus B$	INPUT A B / OUTPUT A XOR B: 0 0 0; 0 1 1; 1 0 1; 1 1 0
XNOR			$\overline{A \oplus B}$ or $A \odot B$	INPUT A B / OUTPUT A XNOR B: 0 0 1; 0 1 0; 1 0 0; 1 1 1

source: wikipedia

Addition

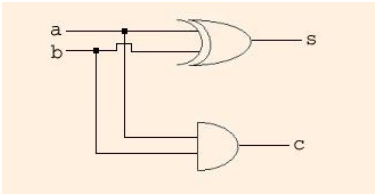
avec calcul de retenue

a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

exercice!

Addition

avec calcul de retenue



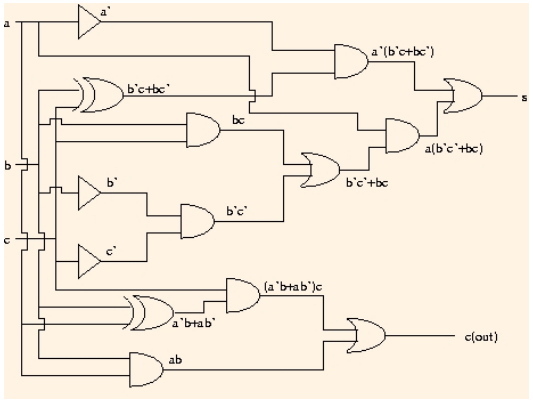
«half-adder»

a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

source: http://www.ucblueash.edu/koehler/comath/25.html

Addition

avec prise en charge de retenue et calcul de retenue



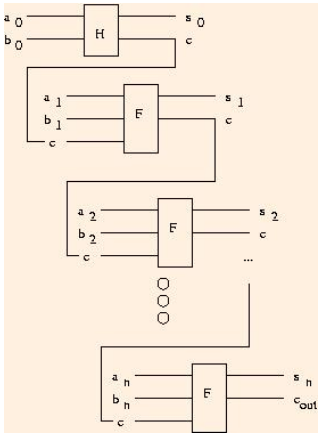
«full-adder»

a	b	c (in)	s	c (out)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

source: http://www.ucblueash.edu/koehler/comath/25.html

Addition

deux nombres de *n* bits



source: http://www.ucblueash.edu/koehler/comath/25.html

Codage

comment représenter l'information?

- Représentation en machine
  - bases 2, 8, 10, 16
  - passage d'une base à une autre
  - complément à deux
- Représentation des nombres réels
  - de 2^32 (en nombres entiers) à 2^232 valeurs en 32 bits
  - représentation en virgule flottante (norme IEEE 754)
    - définit la représentation, le comportement en cas de dépassement de capacité et garanti un arrondi exact pour les opération élémentaires (+,-,*,/,Sqroot)

- Représentation en machine
  - bases 2, 8, 10, 16
  - passage d'une base à une autre
  - complément à deux
- Représentation des nombres réels
  - de 2^32 (en nombres entiers) à 2^232 valeurs en 32 bits
  - représentation en virgule flottante (norme IEEE 754)
    - définit la représentation, le comportement en cas de dépassement de capacité et garanti un arrondi exact pour les opération élémentaires (+,-,*,/,Sqroot)

En base b, on utilise b chiffres. Notons a_i la suite des chiffres utilisés pour écrire un nombre

$$x = a_n\, a_{n-1} ... a_1 a_0$$

a_0 est le chiffre des unités.

- En décimal, $b = 10$, $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$;
- En binaire, $b = 2$, $a_i \in \{0, 1\}$: 2 chiffres binaires, ou bits ;
- En hexadécimal, $b = 16$, $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$ (on utilise les 6 premières lettres comme des chiffres).

Représentation des nombres entiers

En base 10, on écrit par exemple 1996 pour représenter le nombre

$$1996 = 1\ 10^3 + 9\ 10^2 + 9\ 10^1 + 6\ 10^0$$

Dans le cas général, en base b, le nombre représenté par une suite de chiffres $a_n a_{n-1} ... a_1 a_0$ est donné par :

$$a_n a_{n-1} ... a_1 a_0 = \sum_{i=0}^{n} a_i b^i$$

a_0 est le chiffre de poids faible, et a_n le chiffre de poids fort.
Exemple en base 2 :

$$(101)_2 = \ 1\ 2^2 + 0\ 2^1 + 1\ 2^0 \ = 4 + 0 + 1 = 5$$

La notation $(\ )_b$ indique que le nombre est écrit en base b.

Passage d'une base quelconque à la base 10

Il suffit d'écrire le nombre comme ci-dessus et d'effectuer les opérations en décimal.
Exemple en hexadécimal :

$$(AB)_{16} = 10\ 16^1 + 11\ 16^0 \ = 160 + 11 \ = (171)_{10}$$

(en base 16, A représente 10, B 11, et F 15).

Passage de la base 10 vers une base quelconque

Nombres entiers On procède par divisions successives. On divise le nombre par la base, puis le quotient obtenu par la base, et ainsi de suite jusqu'a obtention d'un quotient nul.
La suite des restes obtenus correspond aux chiffres dans la base visée, $a_0 a_1 ... a_n$.
Exemple : soit à convertir $(44)_{10}$ vers la base 2.

$$\begin{array}{lll} 44 = 22 \text{ x } 2 + 0 & \Longrightarrow & a_0 = 0 \\ 22 = 11 \text{ x } 2 + 0 & \Longrightarrow & a_1 = 0 \\ 11 = 2 \text{ x } 5 + 1 & \Longrightarrow & a_2 = 1 \\ 5 = 2 \text{ x } 2 + 1 & \Longrightarrow & a_3 = 1 \\ 2 = 1 \text{ x } 2 + 0 & \Longrightarrow & a_4 = 0 \\ 1 = 0 \text{ x } 2 + 1 & \Longrightarrow & a_5 = 1 \end{array}$$

Donc $(44)_{10} = (101100)_2$.

Cas des bases 2, 8 et 16

Ces bases correspondent à des puissances de 2 ($2^1, 2^3$ et 2^4), d'où des passages de l'une à l'autre très simples. Les bases 8 et 16 sont pour cela très utilisées en informatique, elles permettent de représenter rapidement et de manière compacte des configurations binaires.

La base 8 est appelée notation *octale*, et la base 16 notation *hexadécimale*.

Chaque chiffre en base 16 (2^4) représente un paquet de 4 bits consécutifs. Par exemple :

$$(10011011)_2 = (1001\ \ 1011)_2 = (9B)_{16}$$

De même, chaque chiffre octal représente 3 bits.

On manipule souvent des nombres formés de 8 bits, nommés *octets*, qui sont donc notés sur 2 chiffres hexadécimaux.

Opérations arithmétiques

Les opérations arithmétiques s'effectuent en base quelconque b avec les mêmes méthodes qu'en base 10. Une retenue ou un report apparait lorsque l'on atteint ou dépasse la valeur b de la base.

Entiers naturels

Les entiers naturels (positifs ou nuls) sont codés sur un nombre d'octets fixé (un octet est un groupe de 8 bits). On rencontre habituellement des codages sur 1, 2 ou 4 octets, plus rarement sur 64 bits (8 octets, par exemple sur les processeurs DEC Alpha).

Un codage sur n bits permet de représenter tous les nombres naturels compris entre 0 et $2^n - 1$. Par exemple sur 1 octet, on pourra coder les nombres de 0 à $255 = 2^8 - 1$.

On représente le nombre en base 2 et on range les bits dans les cellules binaires correspondant à leur poids binaire, de la droite vers la gauche. Si nécessaire, on complète à gauche par des zéros (bits de poids fort).

Entiers relatifs

Il faut ici coder le signe du nombre. On utilise le codage en *complément à deux*, qui permet d'effectuer ensuite les opérations arithmétiques entre nombres relatifs de la même façon qu'entre nombres naturels.

1. **Entiers positifs ou nuls :** On représente le nombre en base 2 et on range les bits comme pour les entiers naturels. Cependant, la cellule de poids fort est toujours à 0 : on utilise donc $n-1$ bits.
   Le plus grand entier positif représentable sur n bits en relatif est donc $2^{n-1}-1$.
2. **Entiers négatifs :** Soit x un entier positif ou nul représenté en base 2 sur $n-1$ bits

$$x = \sum_{i=0}^{n-2} \alpha_i\, 2^i, \text{ avec } \alpha_i \in \{0, 1\}$$

La représentation de $-x$ est obtenue par complémentation à 2^{n-1} de x. On dit *complément à deux*.

Pour obtenir le codage d'un nombre x négatif, on code en binaire sa valeur absolue sur $n-1$ bits, puis on complémente (ou inverse) tous les bits et on ajoute 1.

Exemple : soit à coder la valeur -2 sur 8 bits. On exprime 2 en binaire, soit) 00000010. Le complément à 1 est 11111101. On ajoute 1 et on obtient le: résultat : 1111 1110.

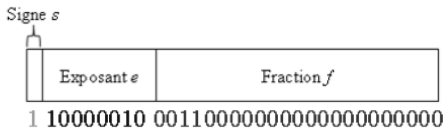
Remarques :

(a) le bit de poids fort d'un nombre négatif est toujours 1 ;

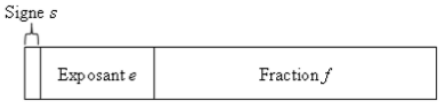
(b) sur n bits, le plus grand entier positif est $2^{n-1} - 1 = 011\ldots1$;

(c) sur n bits, le plus petit entier négatif est -2^{n-1}.

- Représentation en machine
  - bases 2, 8, 10, 16
  - passage d'une base à une autre
  - complément à deux
- Représentation des nombres réels
  - de 2^32 (en nombres entiers) à 2^232 valeurs en 32 bits
  - représentation en virgule flottante (norme IEEE 754)
    - définit la représentation, le comportement en cas de dépassement de capacité et garanti un arrondi exact pour les opération élémentaires (+,-,*,/,Sqroot)



Nombre de bits	Taille de s	Taille de f	Taille de e	E_{min}	E_{max}
32 (simple précision)	1	23	8	-126	+127
64 (double précision)	1	52	11	-1022	+1023



$$(-1)^s \times \left(1 + \sum_{i=1}^{23} f_i 2^{-i}\right) \times 2^{e-E_{\max}}$$

f_i correspond au $i^{ème}$ bit de la fraction f.

A= 1 10000010 00110000000000000000000

s= 1 (-) , e=130-127=3, f=2^{-3}+2^{-4}=0,125+0,0625

A= -1,1875 x 2^3 = -9,5

Exceptions. Afin d'identifier des valeurs extrêmes (valeurs proches de 0, ou ∞) ainsi que des expressions indécidables ($\sqrt{-1}, \frac{0}{0}, \infty - \infty$), la norme introduit des conventions particulières :

e- E_{max}	f	Nombre représenté	Cas
E_{min}-1	0	±0	Zéro
E_{min}-1	≠0	$(-1)^s \times 0.f \times 2^{E\min}$	Nombres dénormalisés (petites valeurs)
$E_{min} \leq \leq E_{max}$		$(-1)^s \times 1.f \times 2^{e-E\max}$	Nombres normalisés
E_{max}+1	0	±∞	Infini
E_{max}+1	≠0	*NaN*	Expressions indécidables

Arrondi. La norme propose quatre types d'arrondi : vers 0, vers +∞, vers -∞, au plus près. En théorie, l'utilisateur a la possibilité de sélectionner le mode d'arrondi, mais en pratique ce choix est souvent imposé dans le langage de programmation.

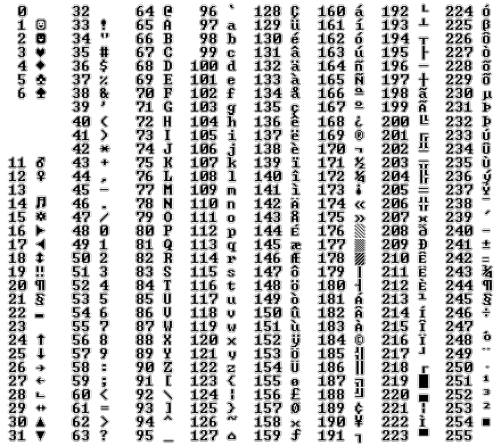
Exemple : Addition avec arrondi au plus près.

On veut additionner les deux nombres suivants (norme IEEE 754):
R_1 = 0 00000010 0010...10
R_2 = 0 00000010 1010...00
On effectue d'abord l'addition des deux mantisses normalisées 1,0010...01 avec 1,1010...00 et le résultat est : 10,1100...01. Comme la partie entière contient 2 bits (10,...), il faut normaliser en décalant à droite d'un bit (1,01100...01). Le 23ème bit de la mantisse correspond donc au dernier 0 (1,01100...01). Comme le préconise la norme IEEE 754, on a effectué le calcul sur plus de 23 bits, et pour ramener la mantisse à 23 bits, on doit appliquer la politique d'arrondi. On suppose quel'on utilise ici l'arrondi par excès, et donc la présence d'un 1 dans le 24e bit implique que le 23e bit doit devenir un 1. Le résultat final est : 0 00000010 01100...01.
en raison de l'arrondi, il y a eu une perte de précision.

Limite:

Exemple : L'addition n'est pas associative ; e→3 bits, f→4 bits.

$$(-1{,}0111\times 2^4 + 1{,}0111\times 2^4) + 1{,}1000\times 2^{-3} = 1{,}1000\times 2^{-3}$$
$$-1{,}0111\times 2^4 + (1{,}0111\times 2^4 + 1{,}1000\times 2^{-3}) = -1{,}0111\times 2^4 + 1{,}0111\times 2^4 = 0$$

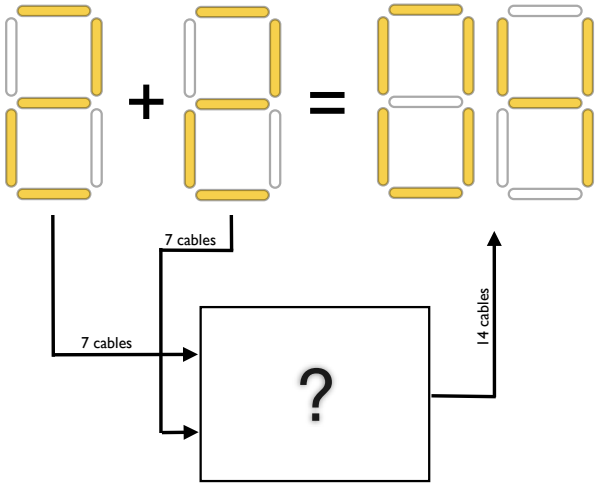


0		32		64	@	96	`	128	Ç	160	á	192	└	224	ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	ô
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	ò
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	õ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
		39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
		40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
		41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
		42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13		45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	☼	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
16	►	48	0	80	P	112	p	144	É	176	░	208	ð	240	
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
19	‼	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	§	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	▬	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23		55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	↑	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	∟	60	<	92	\	124	¦	156	£	188	╝	220	▄	252	³
29	↔	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	▲	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	

Table ASCII

il s'agit d'une norme établissant une correspondance
entre des codes binaires et la représentation de caractères

Pour réfléchir un peu...



Ecrire le circuit logique qui additionne les deux chiffres.

Remarque : il va falloir décoder les entrées et ré-encoder la sortie

Fin du cours #1