l'informatique

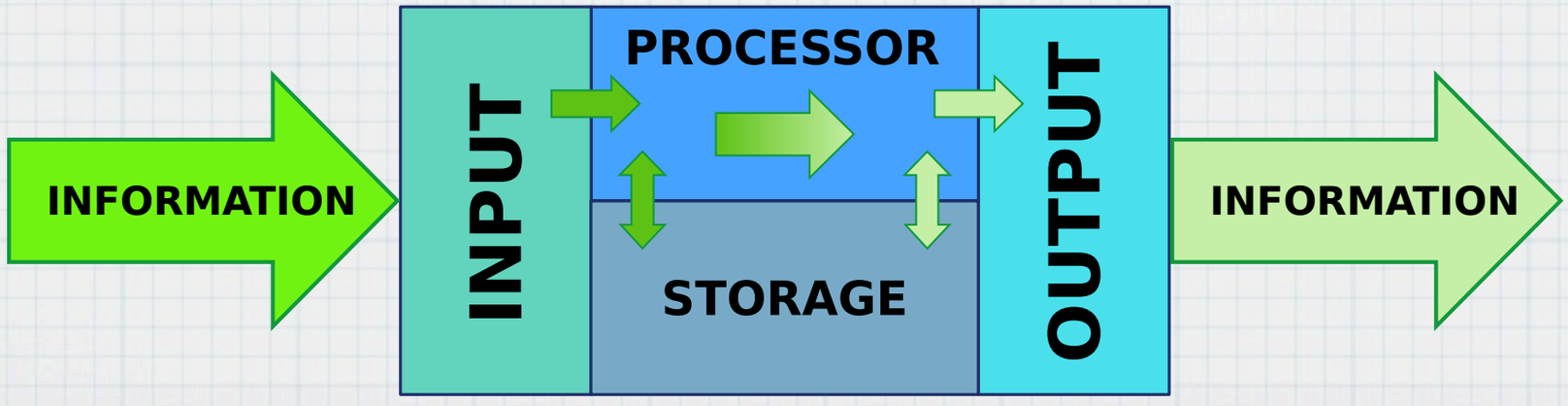
Chapitre 1 : kézako ?

Définition

* **Informatique**

nom féminin
(de *information* et *automatique*)
1 Science du traitement automatique et rationnel de l'information considérée comme le support des connaissances et des communications.

2 Ensemble des applications de cette science, mettant en œuvre des matériels (ordinateurs) et des logiciels.

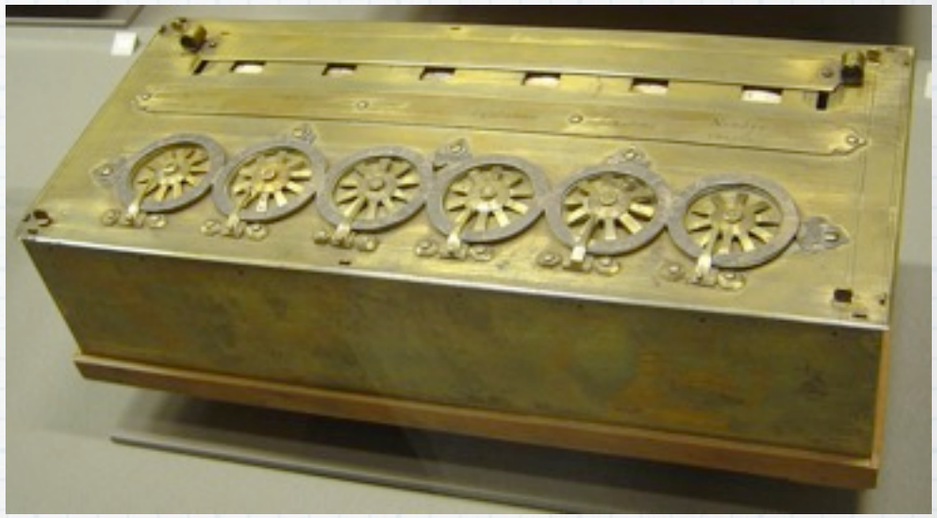


Calcul

* Calcul vient du latin calculus, soit cailloux

* Boulier

Machine de calculs mécaniques



* Pascaline (1640)
* Soustracteur (1820)
  Charles Babbage

Cartes perforées

* Inventées par Basile Bouchon (1725)

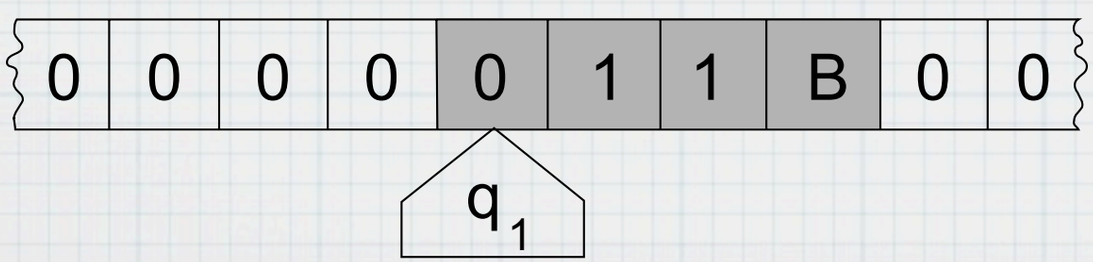
  Améliorées par Jean-Baptiste Falcon (1728)

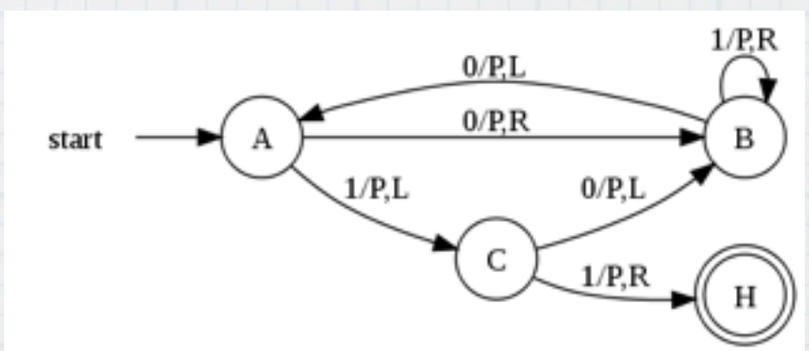
* Automates (flûtes, métiers à tisser...)

* Automate = aucun calcul.

Machine universelle

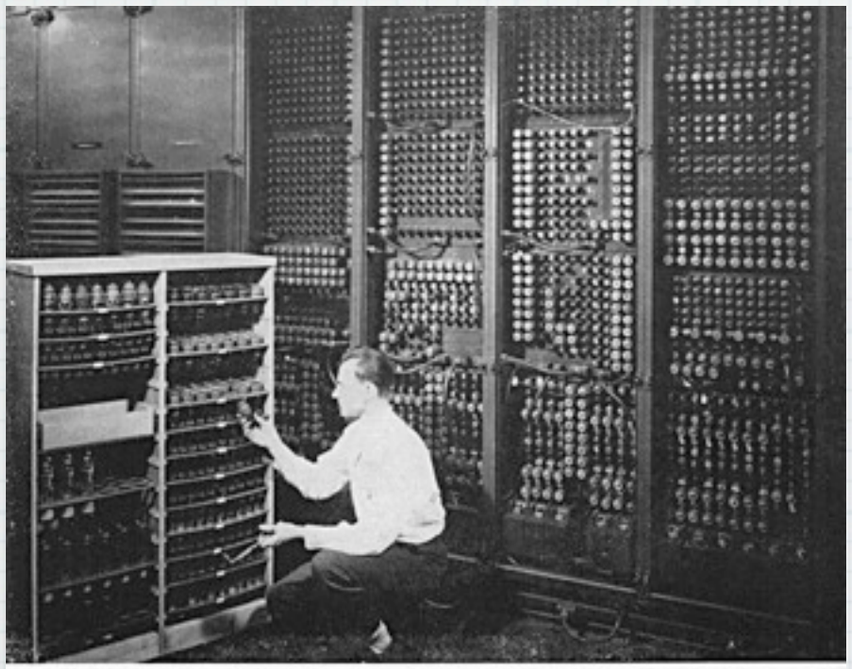
* Question (Hilbert, 1928) : existe-t-il une machine vérifiant une propriété (algorithme de nombre premier) à partir d'une valeur ?
* Voici une machine universelle (Turing & Church, 1936) :





Electronic Numerical Integrator Analyser and Computer (1946)

* 5000 additions/s
* 67 mètres carrés
* 17 468 tubes à vide

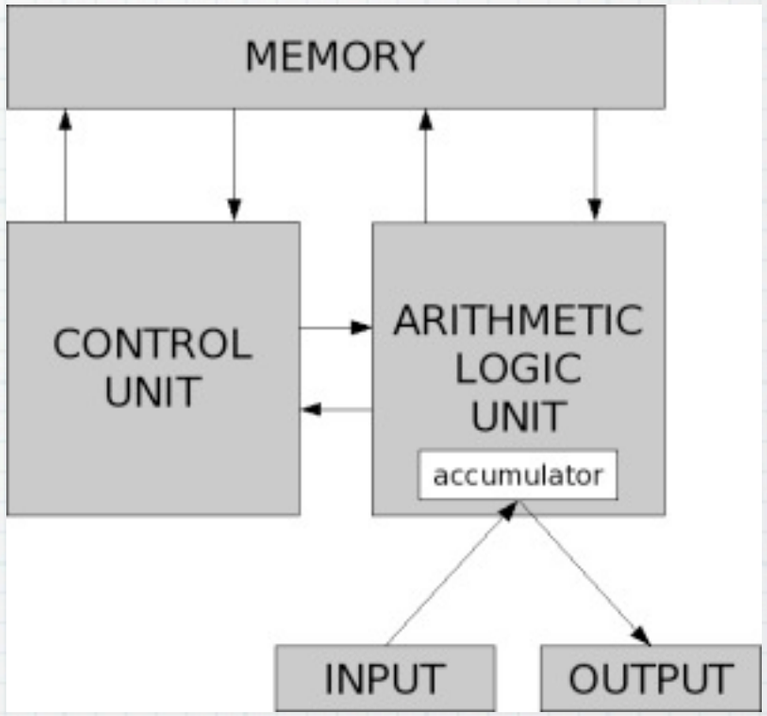


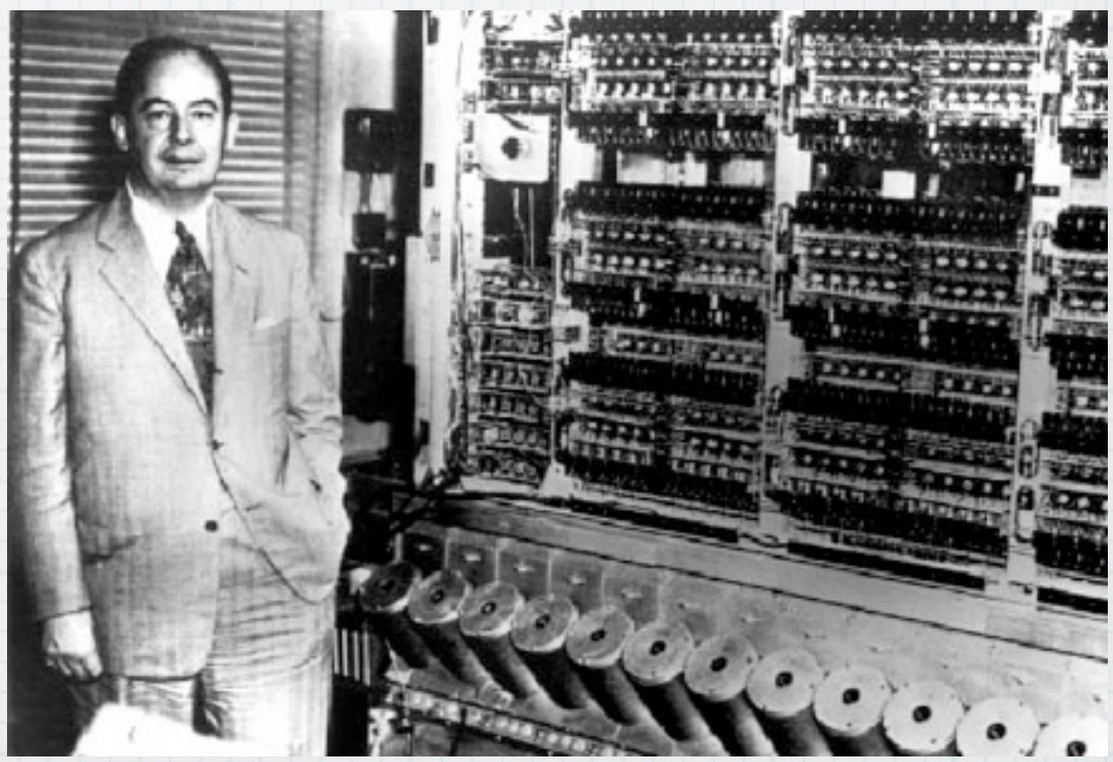
Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

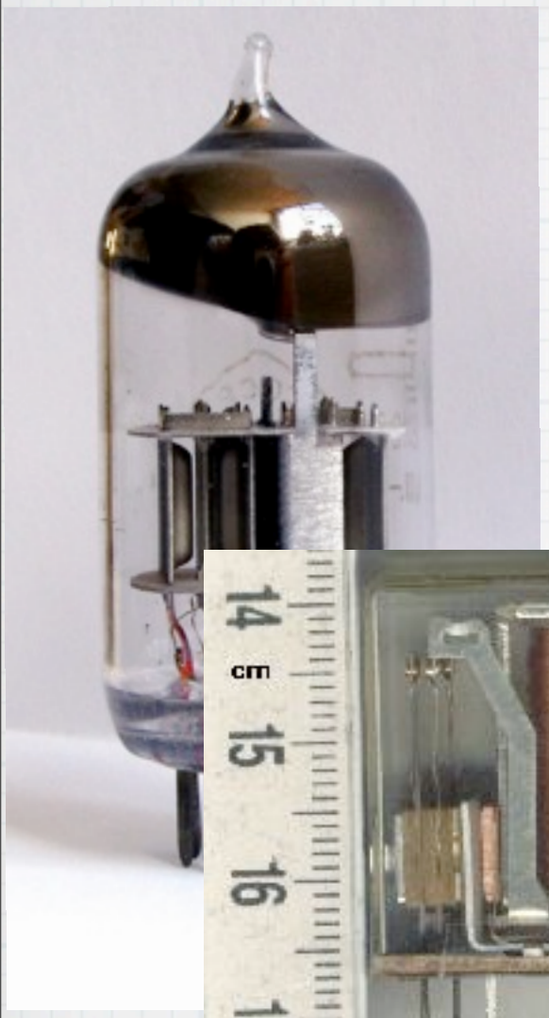
* Turing-complet
* Bug = insecte grillé
* Pas de mémorisation du programme

Von Neumann

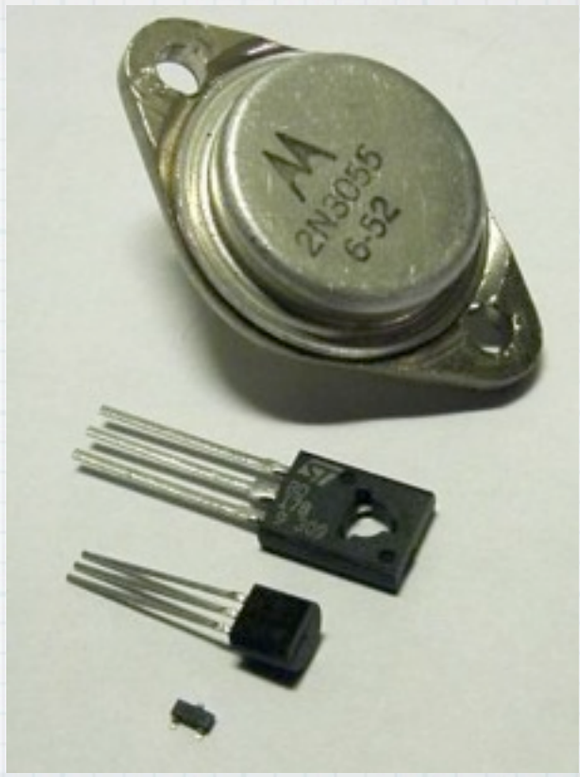
* Nouvelle architecture
* Turing-complet
* Architecture actuellement utilisée
* EDVAC (1948)
  * 7T
  * 6 000 tubes







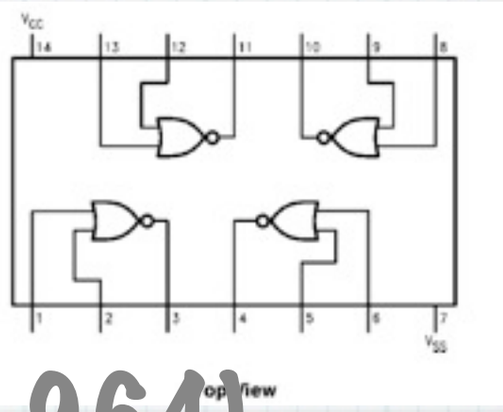
IIe génération



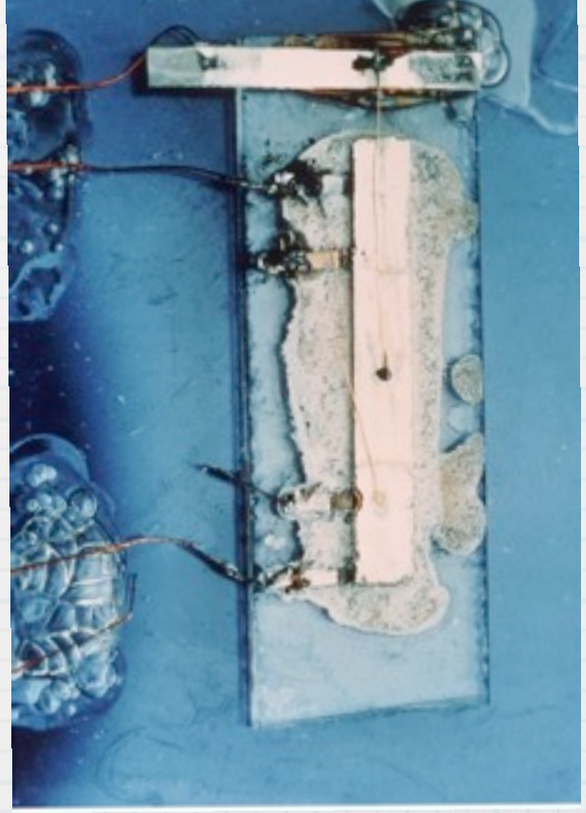
* Transistor :
  * + petit
  * + rapide
  * dure plus longtemps

* Université Manchester (1953) 92 transistors
* IBM 608 (1957 - 83 210$)
  * 4 500 inst./s
  * 3 000 transistors

IIIe génération

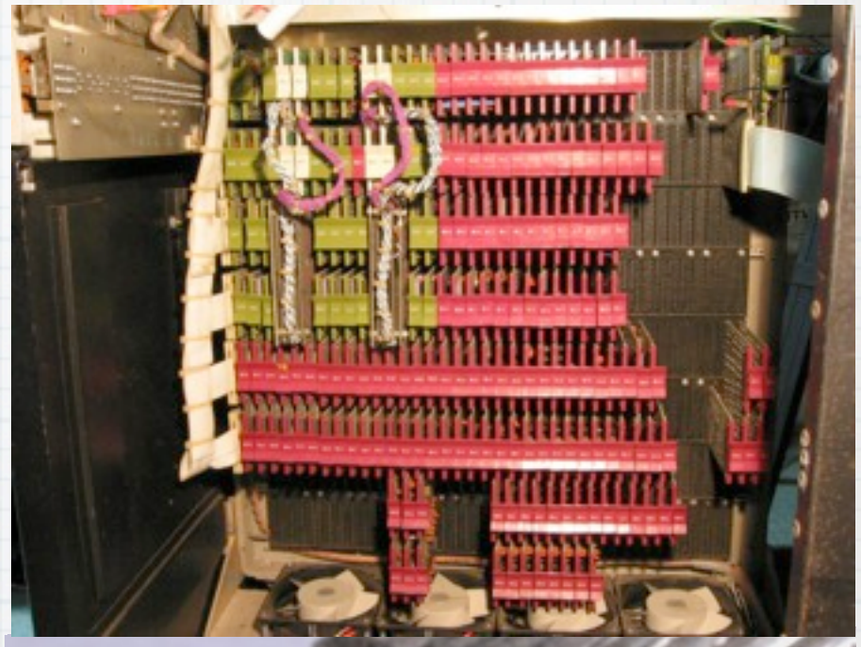


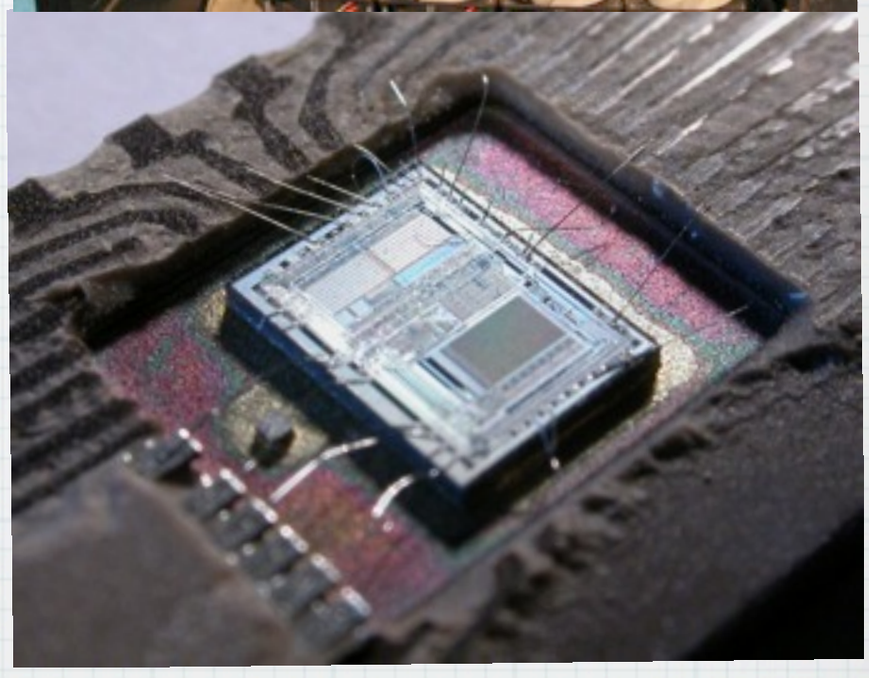
* Circuit intégré (1956) ordinateur (1964)
* Début des interface CLI (1961)
* Toujours plus petit et rapide (premier : 5 transistors)

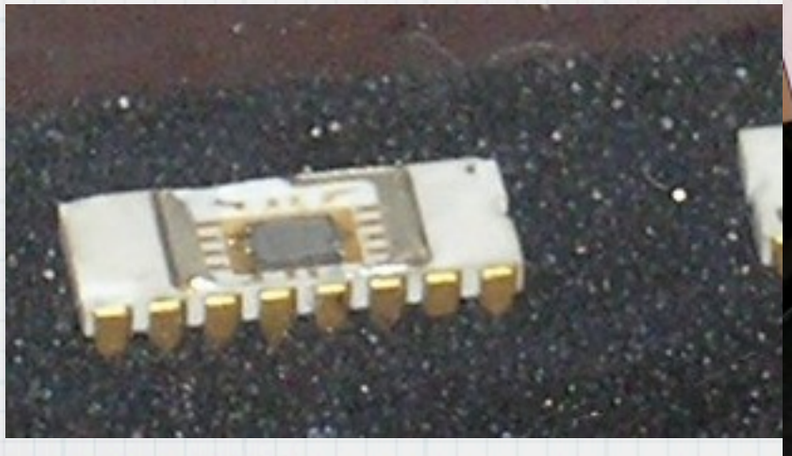


IVe génération

* Microprocesseur
* Premier : Intel 4004 (1971)
  2 300 transistors
  46 instructions
  60 000 Instructions/s
* Début des main-frame







Altair 8800 (1975)

RAM : 256 octets
à 64 Kio

C'est parti !

* Xerox Alto (1973)
  * Souris (1967)
    Douglas Engelbart
  * Interface graphique
  * Réseau
  * Smalltalk (programmation objet - Alan Key)

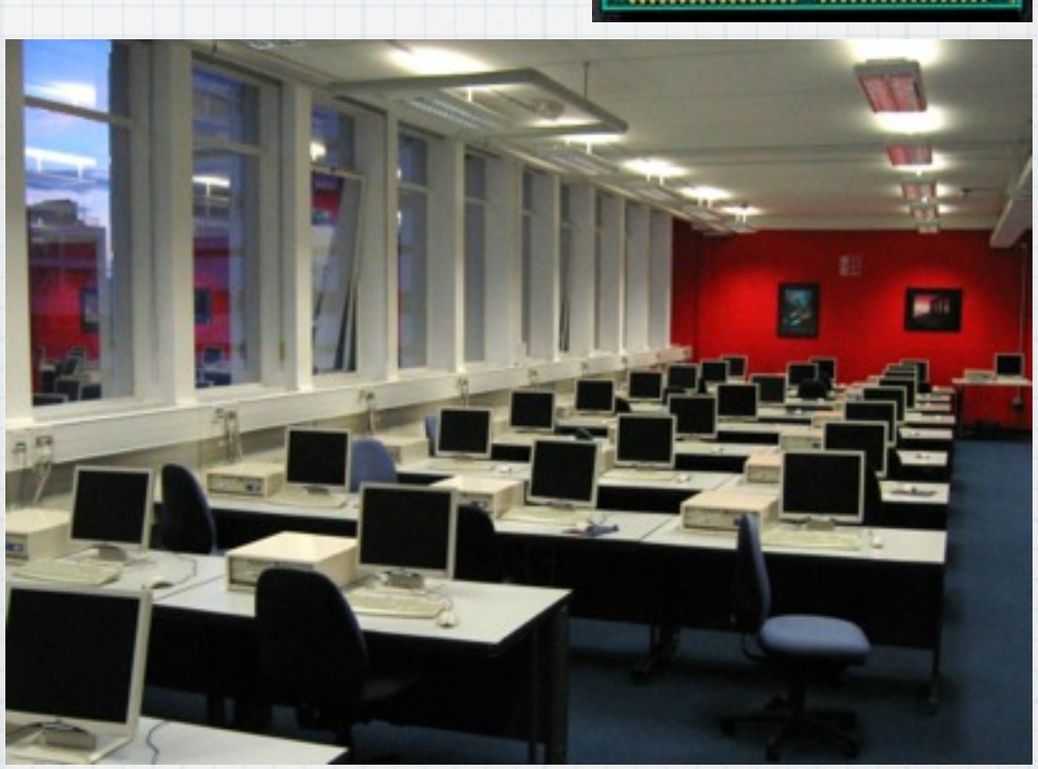
Micro Informatiqu

1978 1979 steve va chez alto

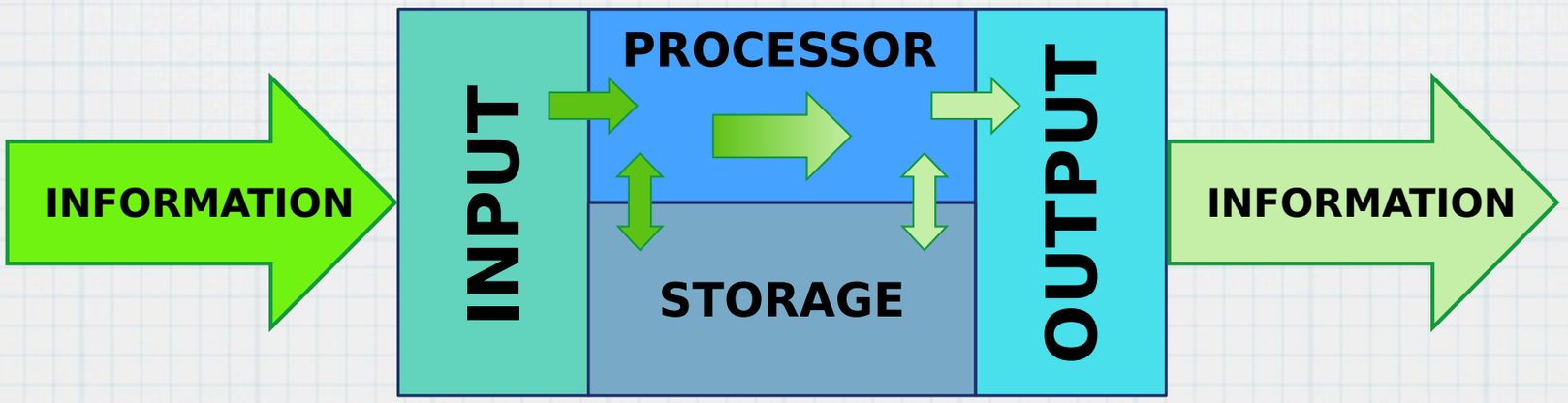
* IBM 5100 (1970) 19 975$
* Apple I,II (1976 -1977) 2000$
* IBM-PC et Dos (1981) 1 565$
* Apple Lisa (1983) 9 995$
* Windows 1 (1985)
* Carte son (1987)
* Linux (1991)

Maintenant

* Core-i7 Gulftown : $147{,}6*10^9$ inst./s
  $1{,}17*10^9$ transistors
* 1 To disque dur
* 4 Go RAM
* Vidéo, 3D, musique, etc.



Fonctionnement

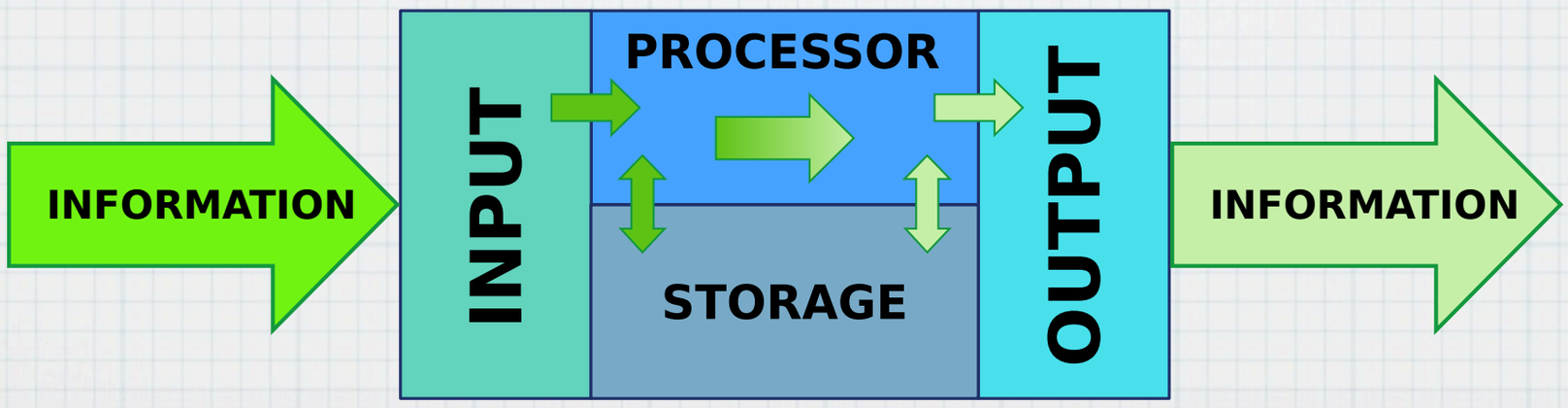


Discret vs continu

* Nombre fini d'éléments séparés :
  * Phrase
  * ADN
  * Entier naturel
* Le reste :
  * Nombre réel
  * Forme
  * Courbe
  * Photographie

Binaire

* Le binaire : 0 ou 1
* Comment coder l'information
* Comment faire calculer



Codage de l'information

0=0
1=1
2=10
3=11
4=100
5=101
6=110
7=111
8=1000
9=1001
10=1010
11=1011

* Vrai /Faux -> 1/0
* Nombre entier
* Nombre relatif (le premier bit donne le signe)
* Nombre rationnel (couple de deux nombres entiers)

Déduire la mémoire nécessaires

* Calcul combinatoire
  * 2 combinaisons possibles par élément combien de combinaisons avec n de ces éléments ?

Déduire la mémoire nécessaires

* Calcul combinatoire
  * 2 combinaisons possibles par élément combien de combinaisons avec n de ces éléments ?

$$2^n$$

Déduire la mémoire nécessaires

* Calcul combinatoire
  * 2 combinaisons possibles par élément combien de combinaisons avec n de ces éléments ?

$$2^n$$

* On veut compter n' éléments avec des objets n'ayant que deux états.

Déduire la mémoire nécessaires

* Calcul combinatoire
  * 2 combinaisons possibles par élément combien de combinaisons avec n de ces éléments ?

$$2^n$$

* On veut compter n' éléments avec des objets n'ayant que deux états.

$$log_2(n')$$

Déduire la mémoire nécessaires

* Calcul combinatoire
  * 2 combinaisons possibles par élément combien de combinaisons avec n de ces éléments ?

$$2^n$$

* On veut compter n' éléments avec des objets n'ayant que deux états.

$$log_2(n') \qquad log_b(x) = \frac{log'_b(x)}{log'_b(b)}$$

Déduire la mémoire nécessaires

* Calcul combinatoire
  * 2 combinaisons possibles par élément combien de combinaisons avec n de ces éléments ?

$$2^n$$

* On veut compter n' éléments avec des objets n'ayant que deux états.

$log_2(n')$ $\qquad$ $log_2(n') = \frac{ln(n')}{ln(2)} = \frac{log_{10}(n')}{log_{10}(2)}$

Tailles et capacités

* Représentations
* 1 octet (byte) = 8 bits
* 1 Ko = 1000 o
* 1 Mo = 1000 Ko
* 1 Go = 1000 Mo
* 1 Kio (kibi) = 1024 o
* 1 Mio (mébi) = 1024 Kio
* 1 Gio (gibi) = 1024 Mio

* P peta, E exa, Z zetta, Y yotta

ASCII

American Standard Code for Information Interchange

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Texte formaté

* Doc

```
ÐÏ à¡± á    >   þÿ !  #       þÿÿÿ  ÿÿ%    ð ¿      a     bjbj%ç%ç
Cours d’initiation à l’informatique
  i      8  @ñÿ  8     N o r m a l
 CJ  _H  aJ  mH  sH  tH  N  @    N     T i t r e   1
ÿ
[… beaucoup d'informations binaires supprimées ]
ÿ
 ÿÿÿÿ       À  F    Document Microsoft Word  MSWordDoc
Word.Document.8 ô9²q
```

* Rtf

```
{\rtf1\ansi
{\fonttbl
   {\f0\fnil\fcharset0\fprq0\fttruetype Helvetica;}
   {\f1\fnil\fcharset0\fprq0\fttruetype Bitstream Charter;}}
{\f1\fs24  Ceci est un texte accentué  }
\par
{\f0\fs24 avec des caractères {\b gras},}
\par
{\f1 des {\fs18 petits} et des {\fs32 gros}.
}
```

* Pdf

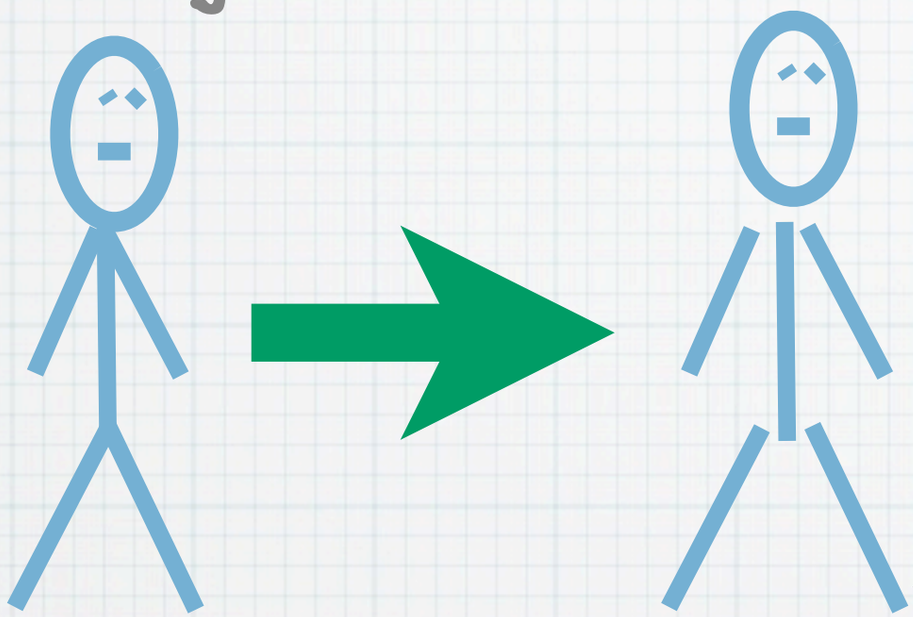
* Html

```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html>
 <head>
  <title>
   Exemple de HTML
  </title>
 </head>
 <body>
  Ceci est une phrase avec un <a href="cible.html">hyperlien</a>.
  <p>
   Ceci est un paragraphe où il n’y a pas <em>d’hyperlien</em>.
  </p>
 </body>
</html>
```

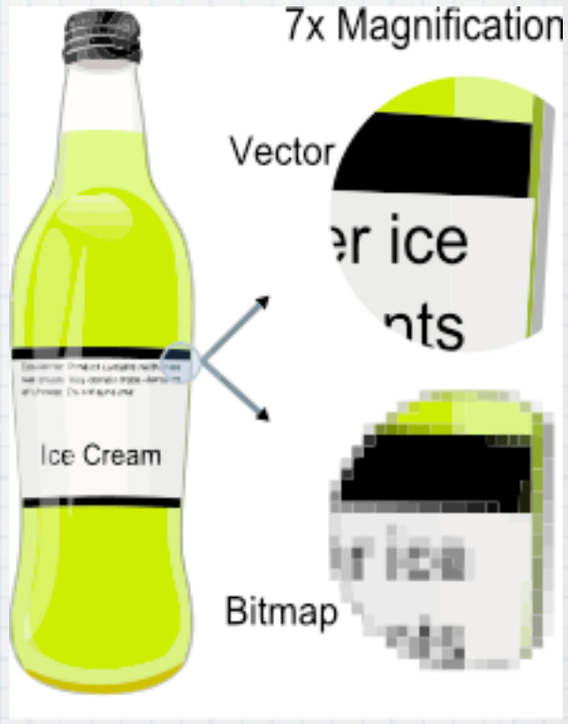
* ....

Numérisation

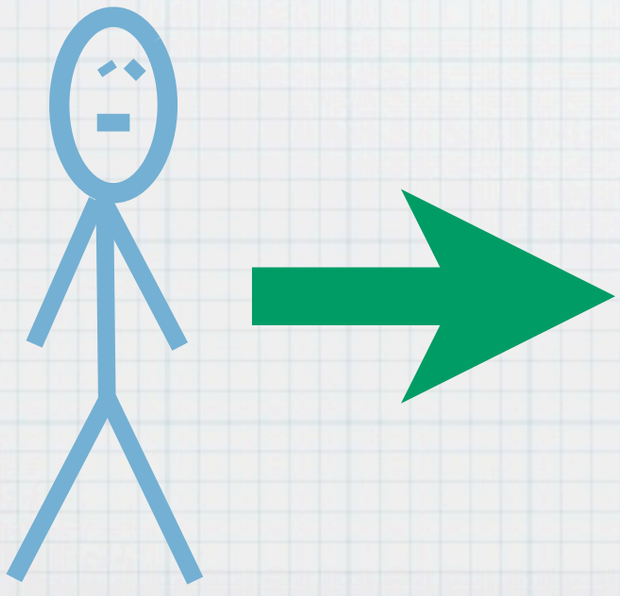
* Image vectorielle



```
<?xml version="1.0" encoding="utf-8"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   version="1.1"
   width="300"
   height="200">
  <title>Exemple simple de figure SVG</title>
  <desc>
    Cette figure est constituée
    de segment de droite et d'un cercle.
  </desc>
   <!-- ceci est un commentaire-->
  <line
     x1="5" y1="5"
     x2="250" y2="95"
     stroke="red" />
  <circle
     cx="90" cy="80"
     r="50"
     fill="blue" />
   <text x="180" y="60">
     Un texte
   </text>
</svg>
```



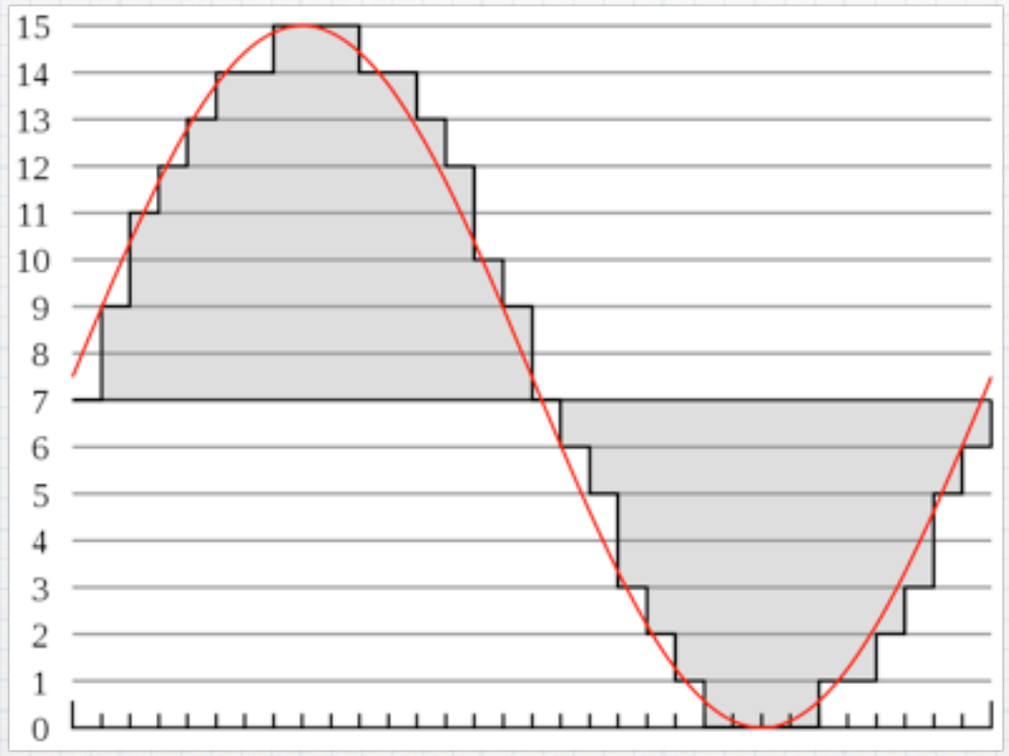
* Image bitmap



```
00000000000000000000000000
00000000000011111110000000
00000000000111000111000000
00000000000110101011000000
00000000000110000011000000
00000000000110111011000000
00000000000110000011000000
00000000000011111110000000
00000000000000011000000000
00000000000001111000000000
00000000000011111100000000
00000000000110110110000000
00000000001100110011000000
00000000011000110001100000
00000000110000110000110000
00000000000000011000000000
00000000000000111100000000
00000000000001111110000000
00000000000110000110000000
00000000001100000011000000
00000000011000000001100000
00000000111000000000111000
00000000110000000000110000
```

Numérisation

* Son (Pulse Code Modulation)



* 8 bit
* 8 KHz
* 1 piste

* 8 bit
* 16 bit

* 8 KHz
* 11 KHz
* 22 KHz
* 44,1 KHz
* 48 KHz
* 96 KHz

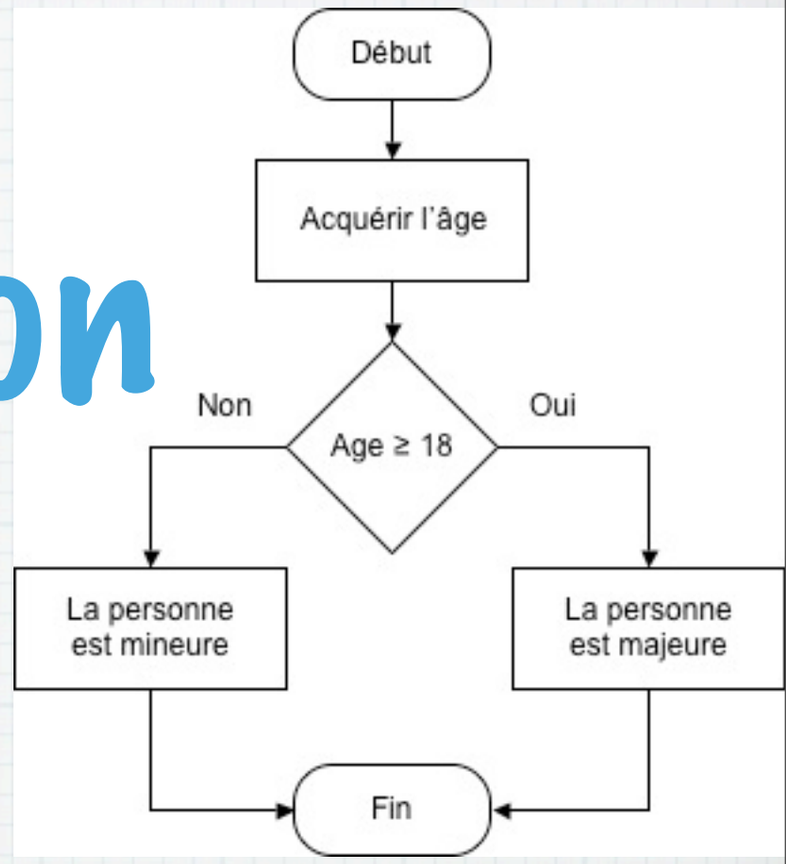
* 16 bit
* 44,1 KHz
* 2 pistes (stéréo)

Compression

* Sans perte
  * Fichier (zip, rar, tbz2...)
  * Son (Ape)
* Avec perte
  * Sons (MP3, OGG)
  * Image (JPG)
  * Vidéo (Divx,Xvid, MPEG...)

Langage de programmation

```
# define N 16

        .global _start

        .comm   BUFF    , N

_start: mov     $3      , %eax
        mov     $0      , %ebx
        mov     $BUFF   , %ecx
        mov     $N      , %edx
        int     $0x80

        mov     %eax    , %edx
        mov     $4      , %eax
        mov     $1      , %ebx
        mov     $BUFF   , %ecx
        int     $0x80
```



* Binaire
* Assembleur
* Basic (1964)
* Pascal (1970)
* C (1972)
* Python (1991)

```c
/* En C (norme ISO) */
int factorielle(int n)
{
    if (n > 1) return n * factorielle(n - 1);
    else return 1;
}
```

```
{ En Pascal }
function factorielle(n: integer) : integer
begin
    if n > 1 then factorielle := n * factorielle(n - 1)
    else factorielle := 1
end.
```

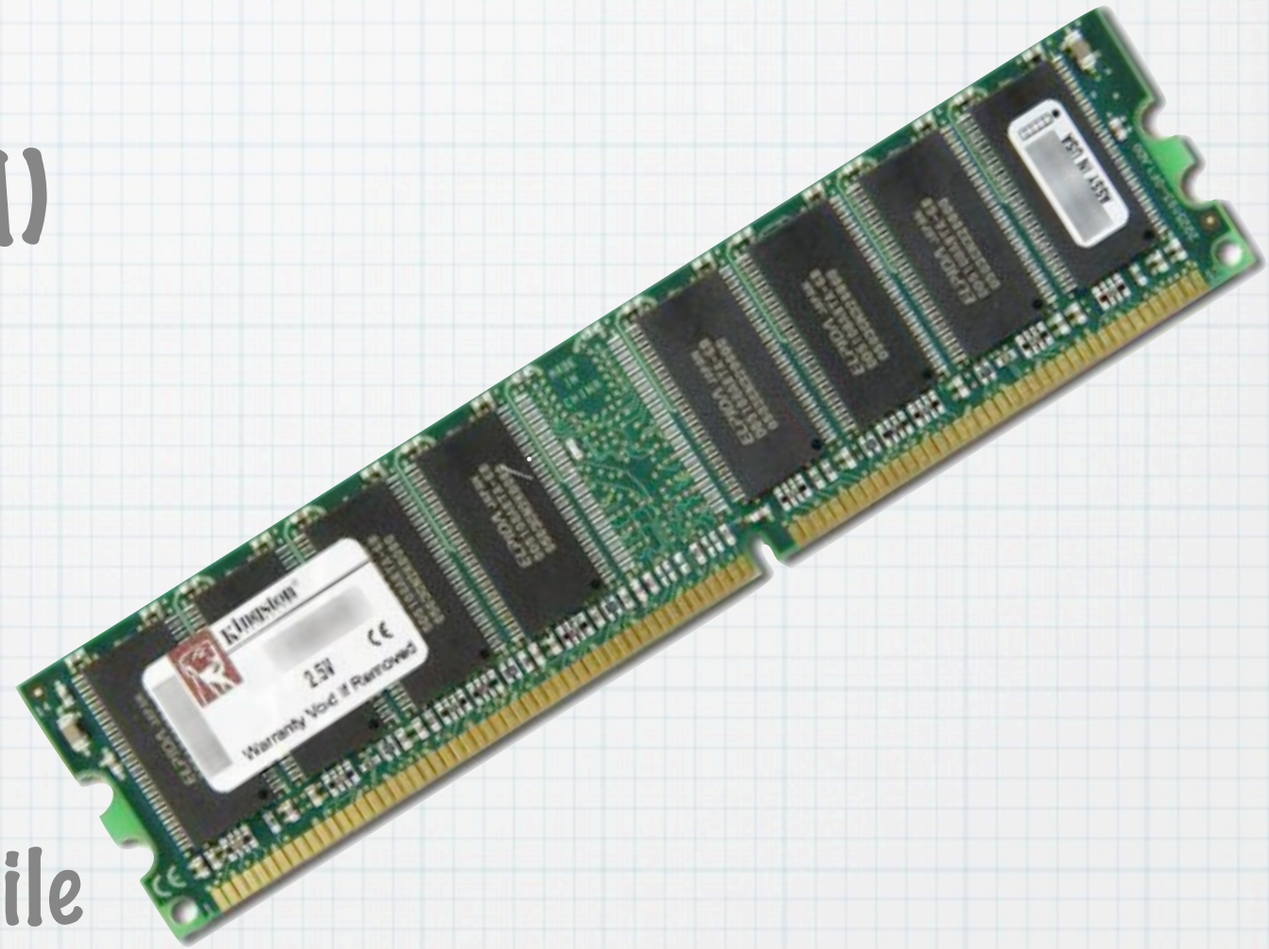
```
def add5(x):
    return x+5

def dotwrite(ast):
    nodename = getNodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print '    %s [label="%s' % (nodename,label),
    if isinstance(ast[1], str):
        if ast[1].strip():
            print '= %s"];' % ast[1]
        else:
            print '"]'
    else:
        print '"];'
        children = []
        for n, child in enumerate(ast[1:]):
            children.append(dotwrite(child))
        print '    %s -> {' % nodename,
        for name in children:
            print '%s' % name,
```

Différents langages de programmation

* Compilé ou interprété
* Rapidité d'exécution
* Méthode d'allocation de la mémoire
* Méthode libération mémoire
* Typage des variables
* Lisibilité par l'être humain
* Conception

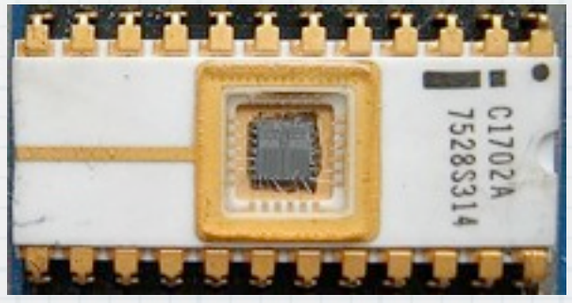
Type de mémoire

* Mémoire vive (RAM)
  * Mémoire volatile
  * Rapide : nanoseconde
  * Stocke les résultats intermédiaires afin des les réutiliser
  * Information non volatile
  * Quelques Go



Type de mémoire

* Mémoire morte (ROM)
  * Gravée dans la roche non modifiable
  * Généralement rapide mais pas toujours
* PROM EPROM EEPROM...



Type de mémoire

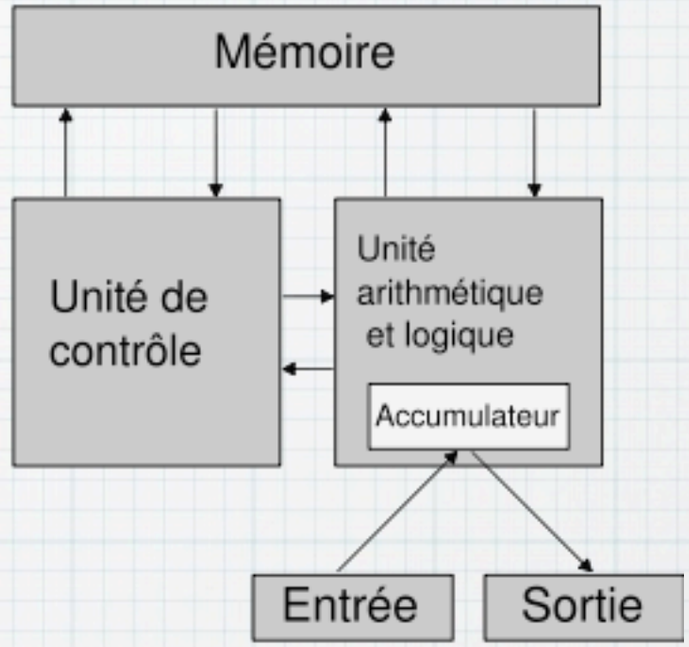
* Mémoire de masse (mass storage)
  * Écriture et lecture avec beaucoup de place et persistante
  * Lente : milliseconde
  * Ex. :
    * Disque dur 60 Go -> 2 To
    * Clef USB 500 Mo -> 32 Go
    * Cartes Flash " "
    * CD-ROM 700 Mo, DVD 4,7 Go, 8 Go
      Blu-ray 25 Go, 50 Go, 128 Go

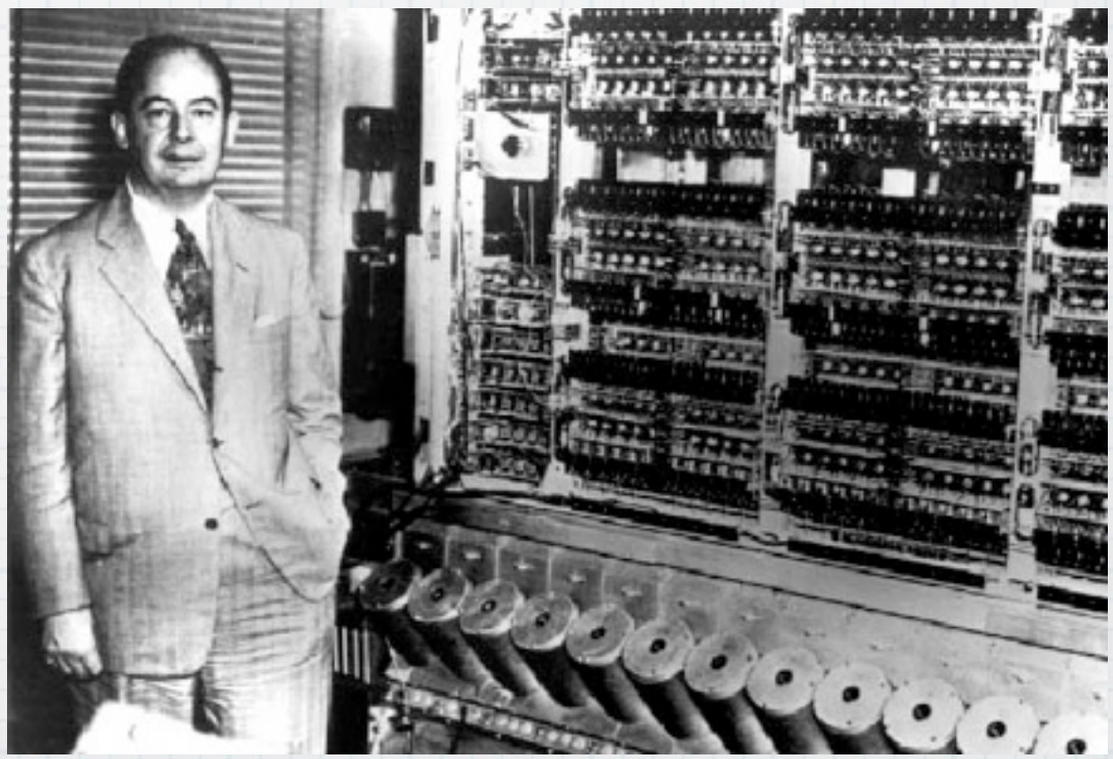
Périphériques

* Clavier
* Souris
* Scanner
* WebCam
* Lecteur
* Ecran
* Imprimante
* Graveur
* Disque dur
* Disquette
* Réseau
* Lecteur enregistreur de carte
* Clef USB

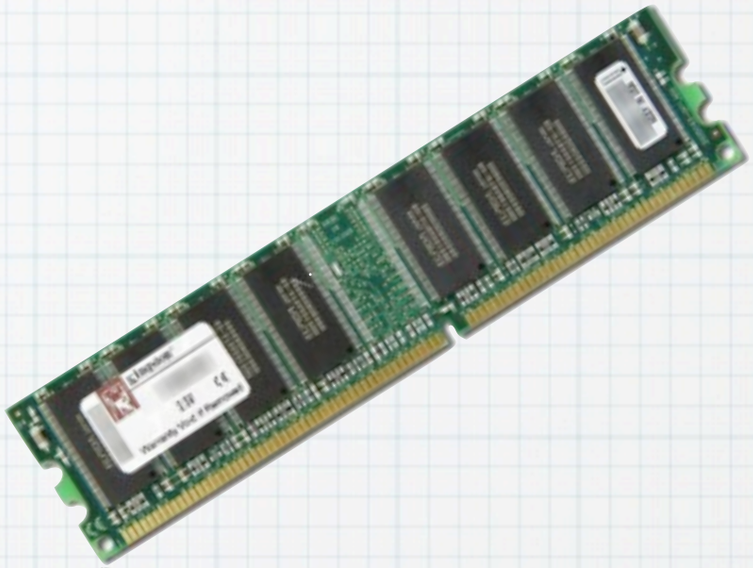
Rappel de von neumann

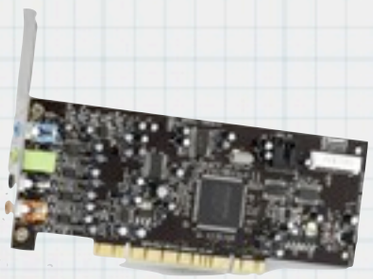
* Turing-complet
* Architecture actuellement utilisée
* EDVAC (1948)
  * 7T
  * 6 000 tubes



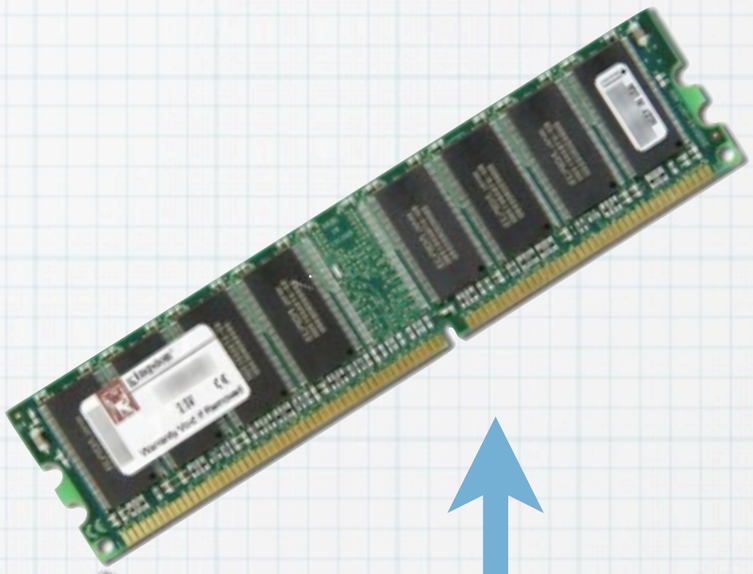


Comment relier ?





Comment relier ?

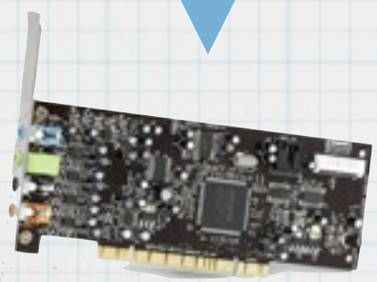


PCI-E

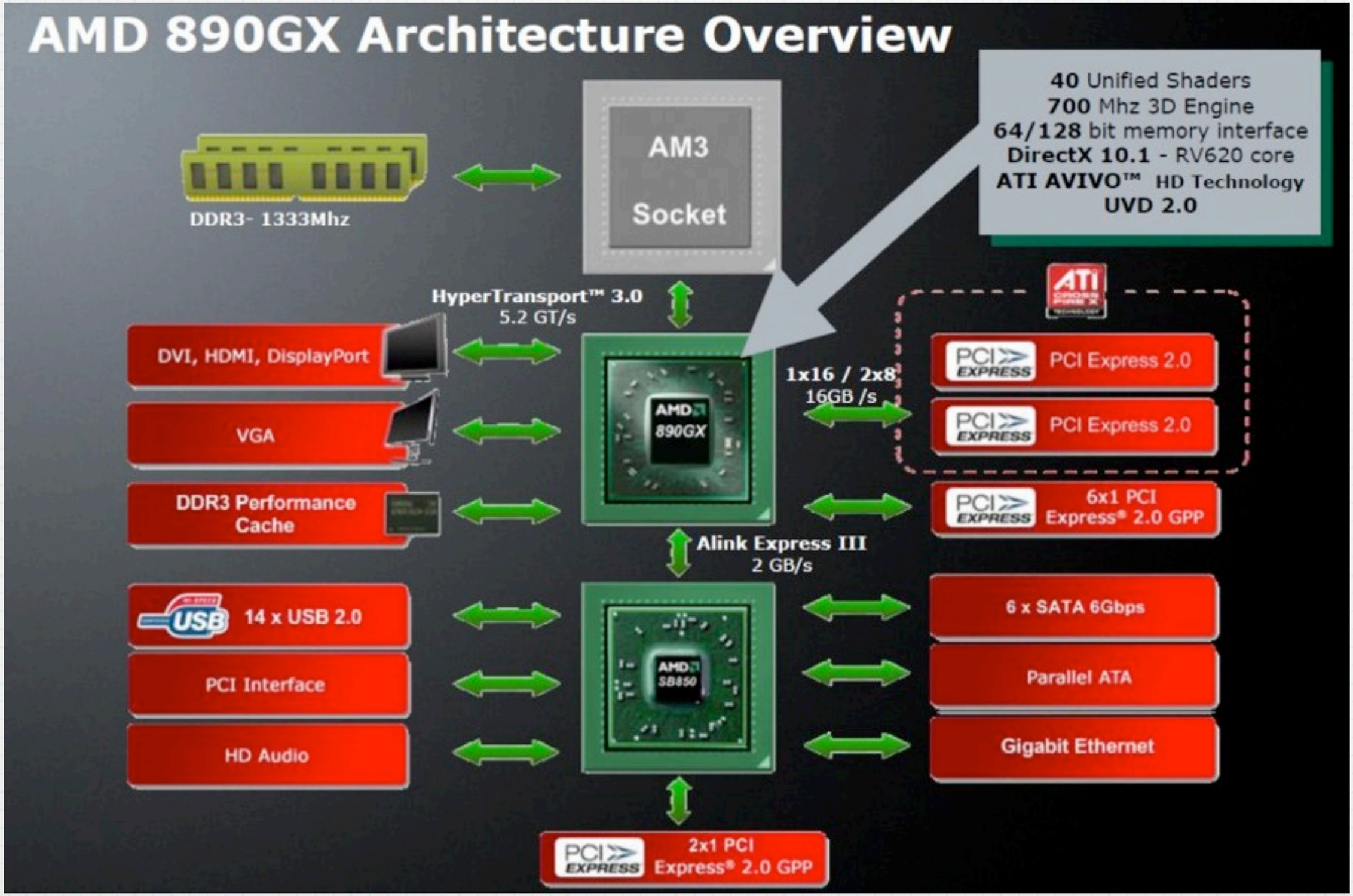
BUS

PCI

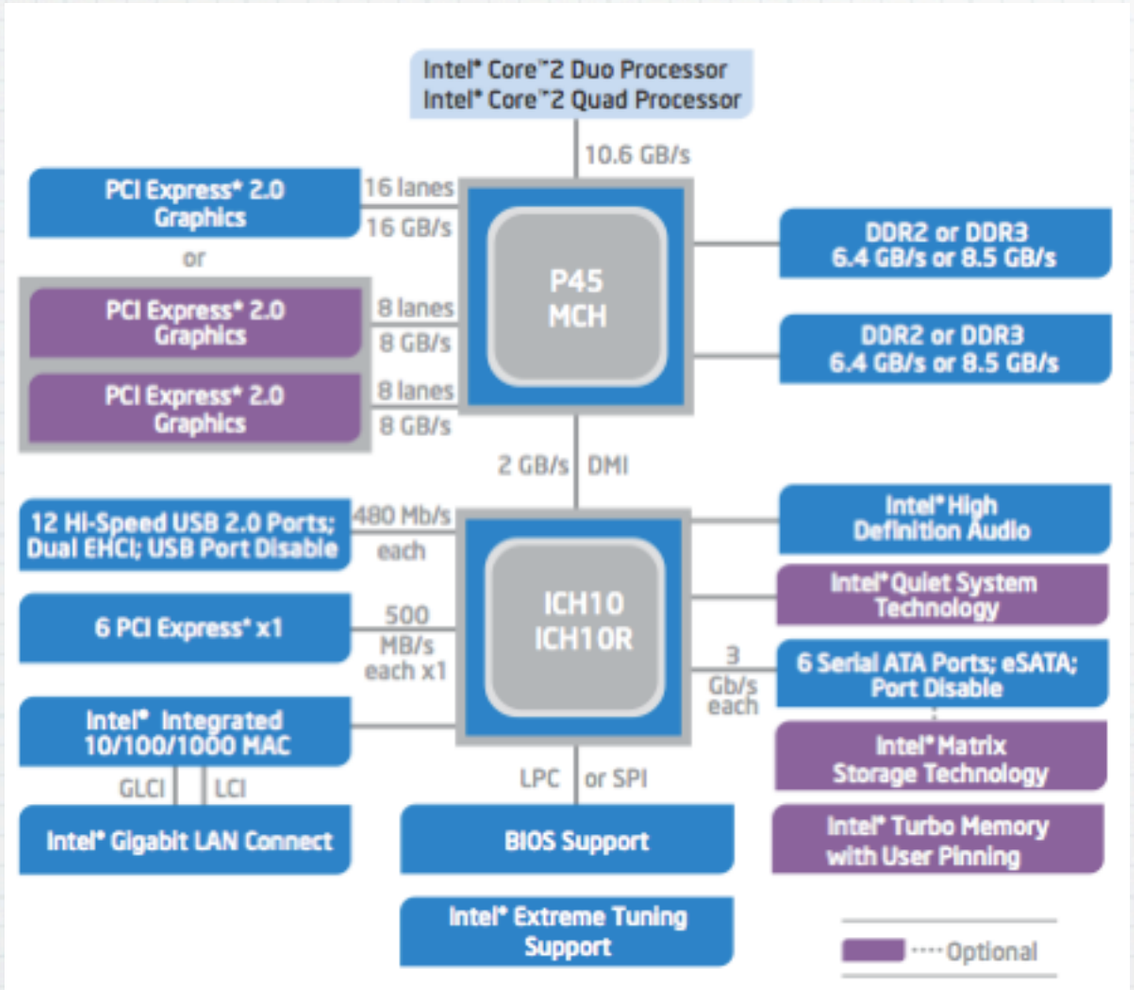
SATA

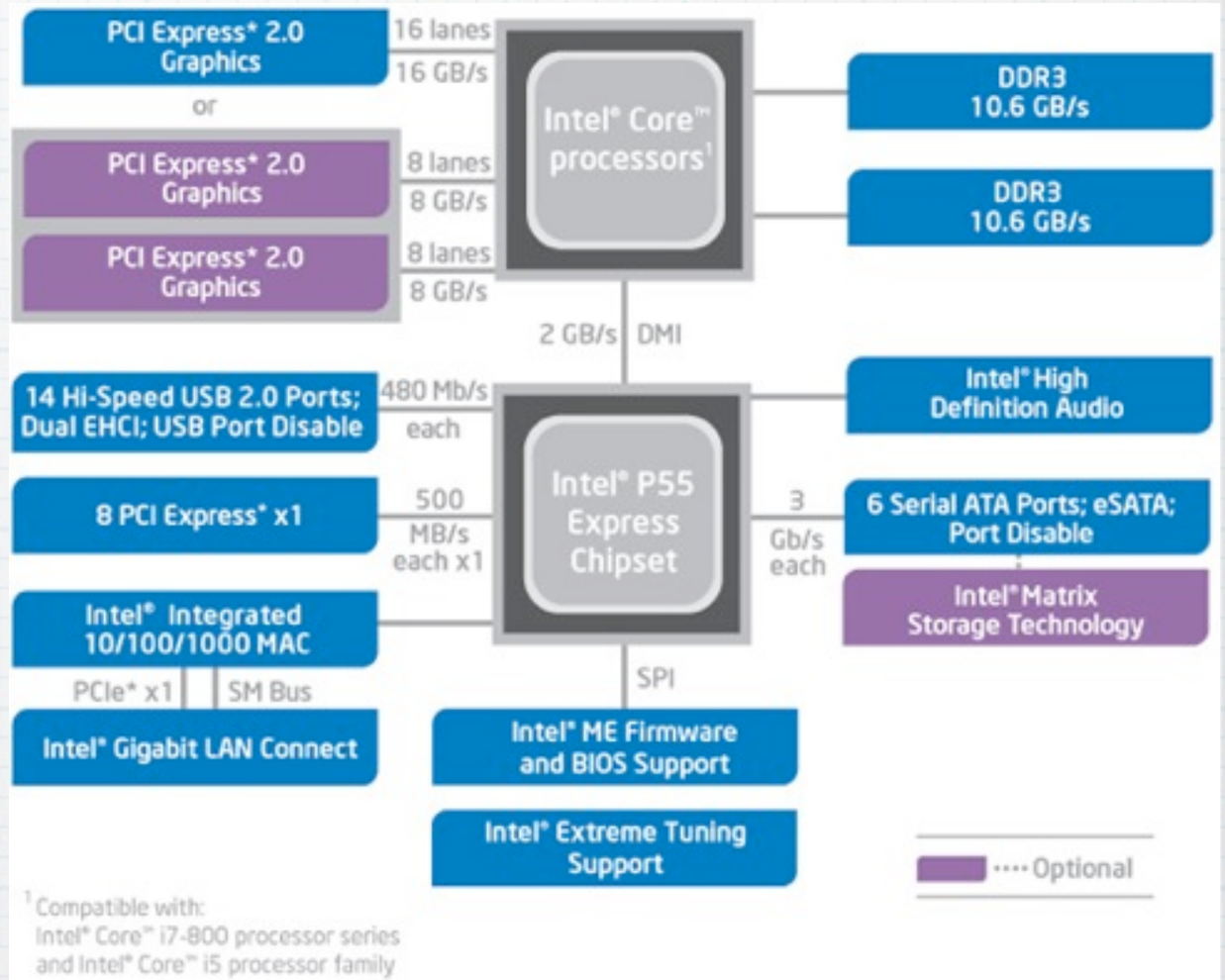


Plus précisément

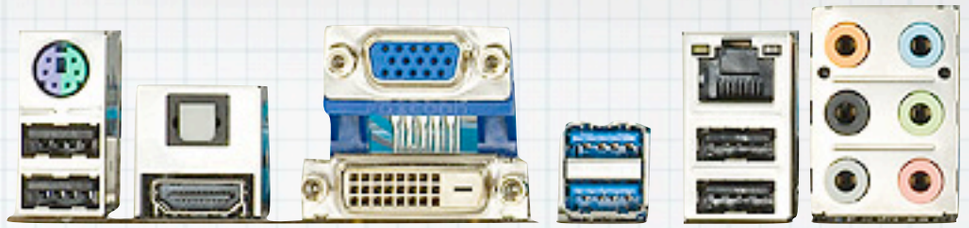


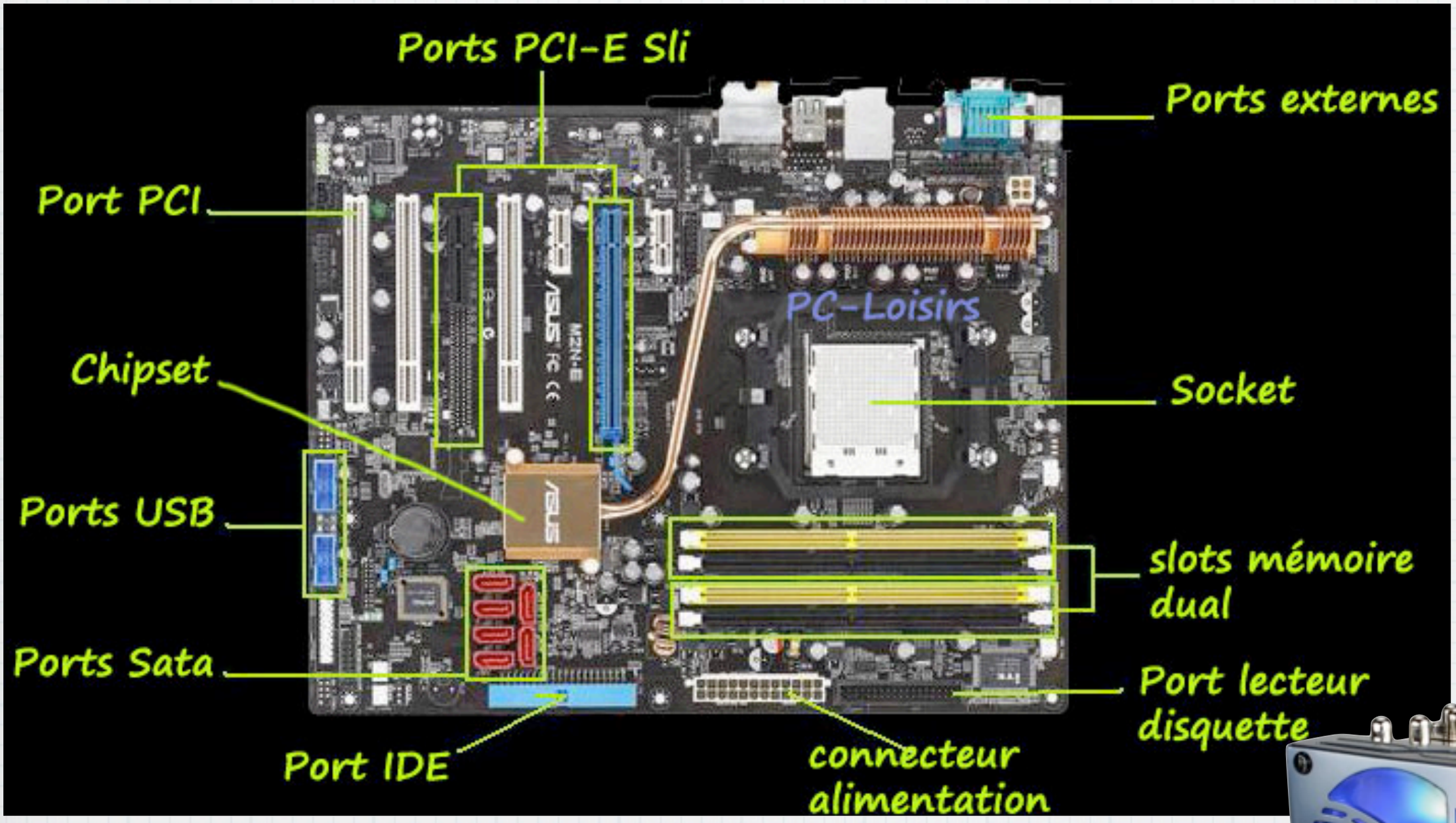
Tout dans une puce

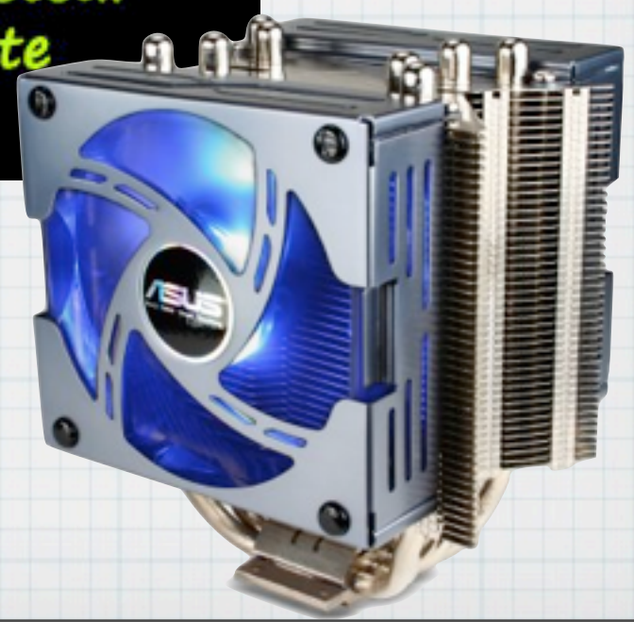




Carte mère

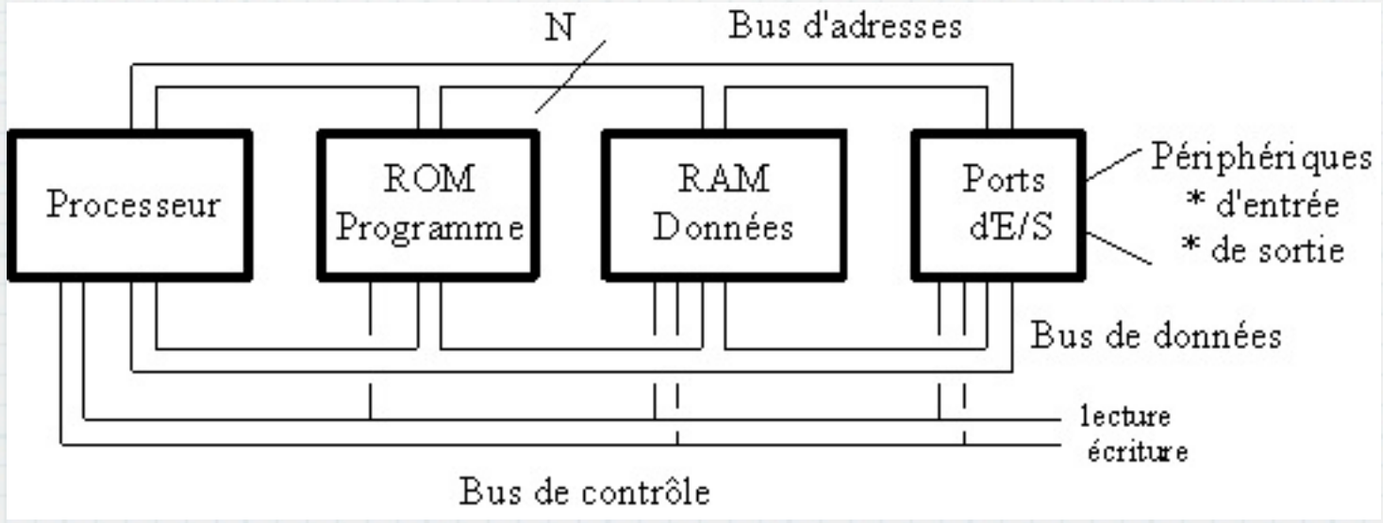



Ports PCI-E Sli
Ports externes
Port PCI
PC-Loisirs
Chipset
Socket
Ports USB
slots mémoire dual
Ports Sata
Port lecteur disquette
Port IDE
connecteur alimentation



Comment cela fonctionne-t-il ?

* Bus d'adressage (donné par le CPU)
* Bus de données
* Parfois 1 bit Lecture/Ecriture (Half-duplex)



Mémoire

* Tableau contenant des données

Adresse

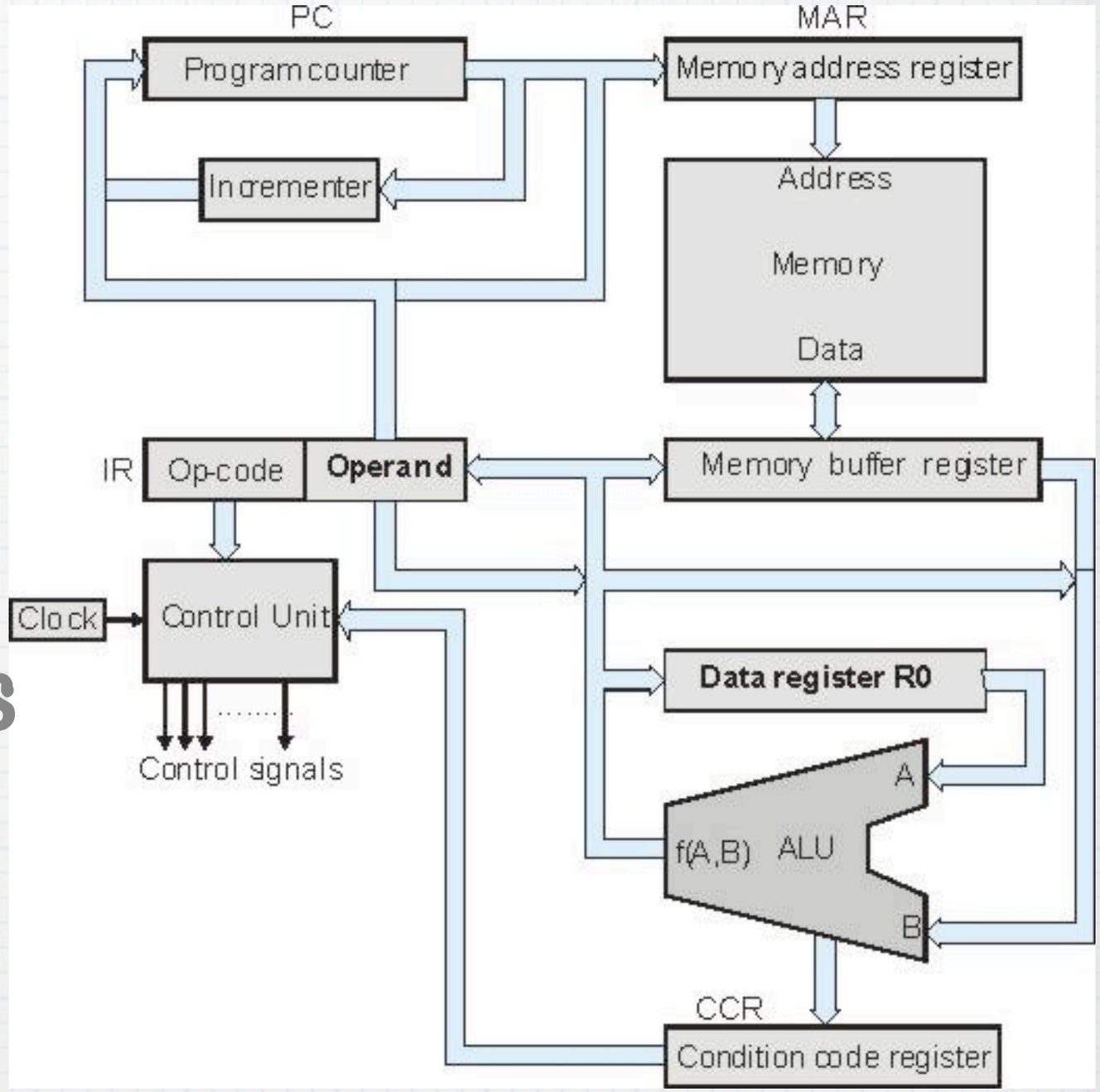
On écrit ou on lit ?

Données

0	2432
1	435
2	543
3	0
4	0
5	0
6	0

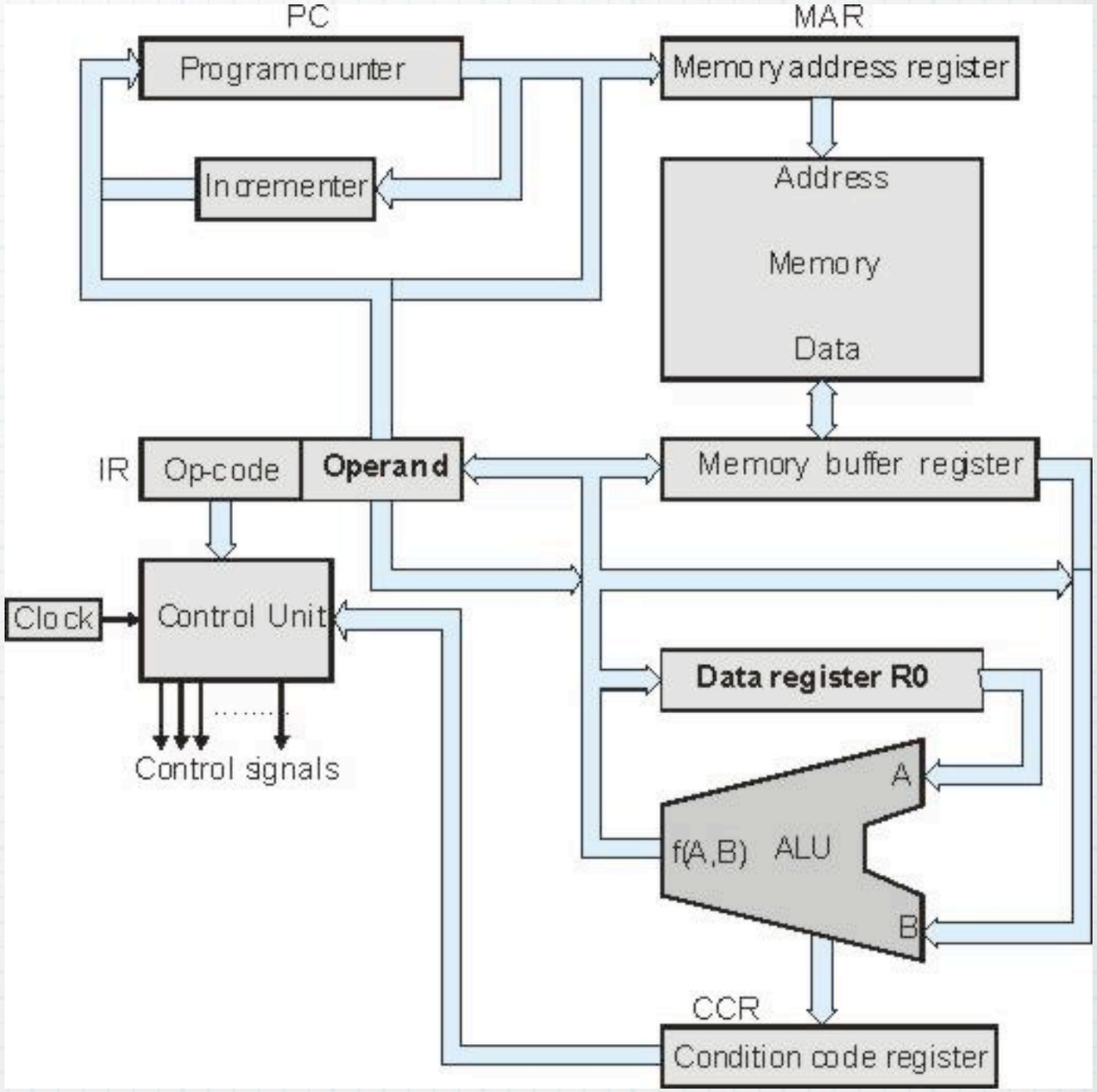
Le processeur (UC ou CPU)

* Compteur ordinal
* Accumulateur
* Registre d'adresses
* Registre d'instructions
* Registre d'état
* ULA



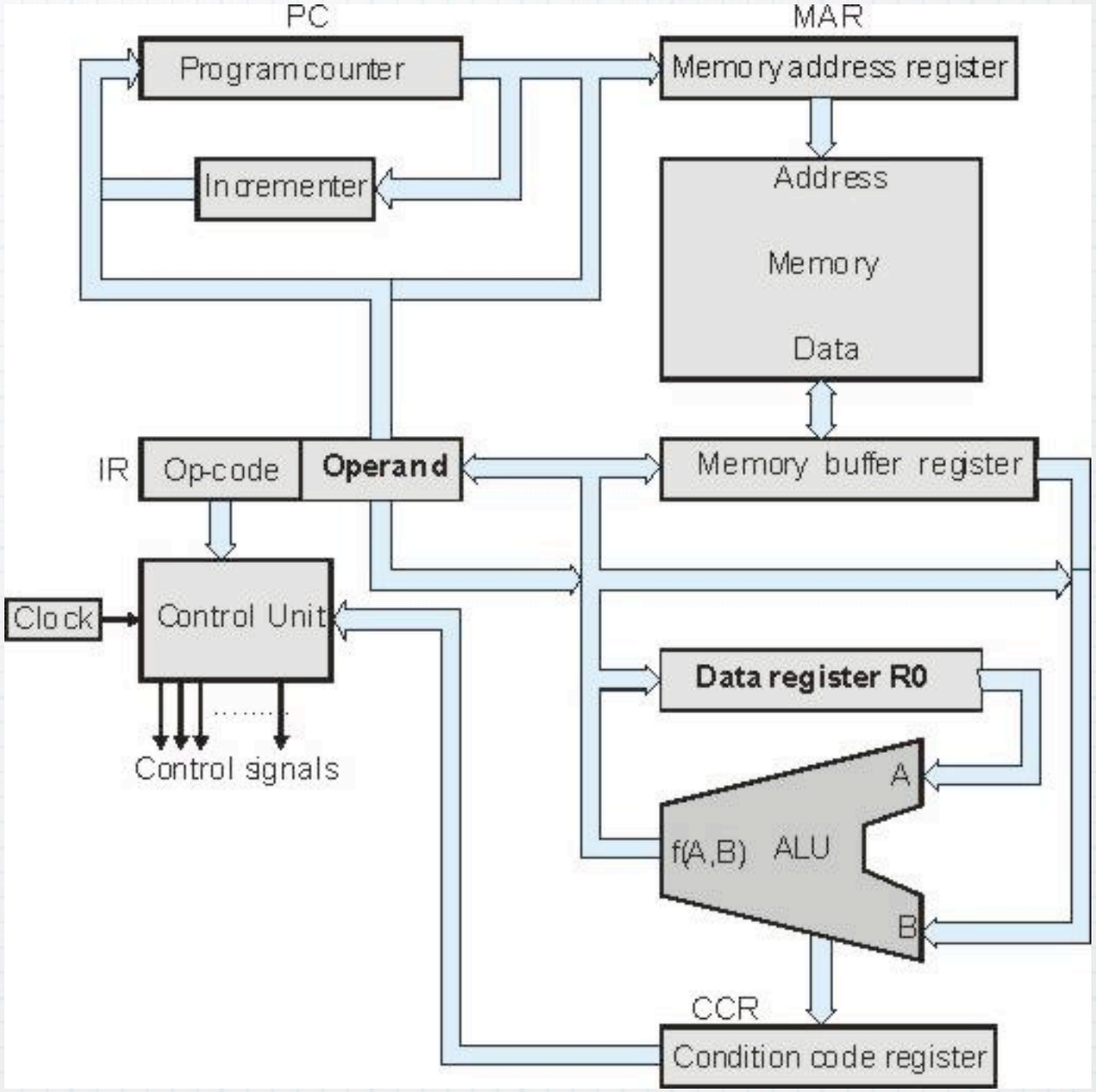
Le processeur (UC ou CPU)

* Chargement (inst.)
* Décodage (inst.)
* Chargement (données)
* Calcul
* Ecriture résultat
* Incrémentation (co)



Assembleur

* Add a
* Add #a
* JMP
* JNZ
* SHL
* LOAD a
* LOAD #a
* STO a
* STOP



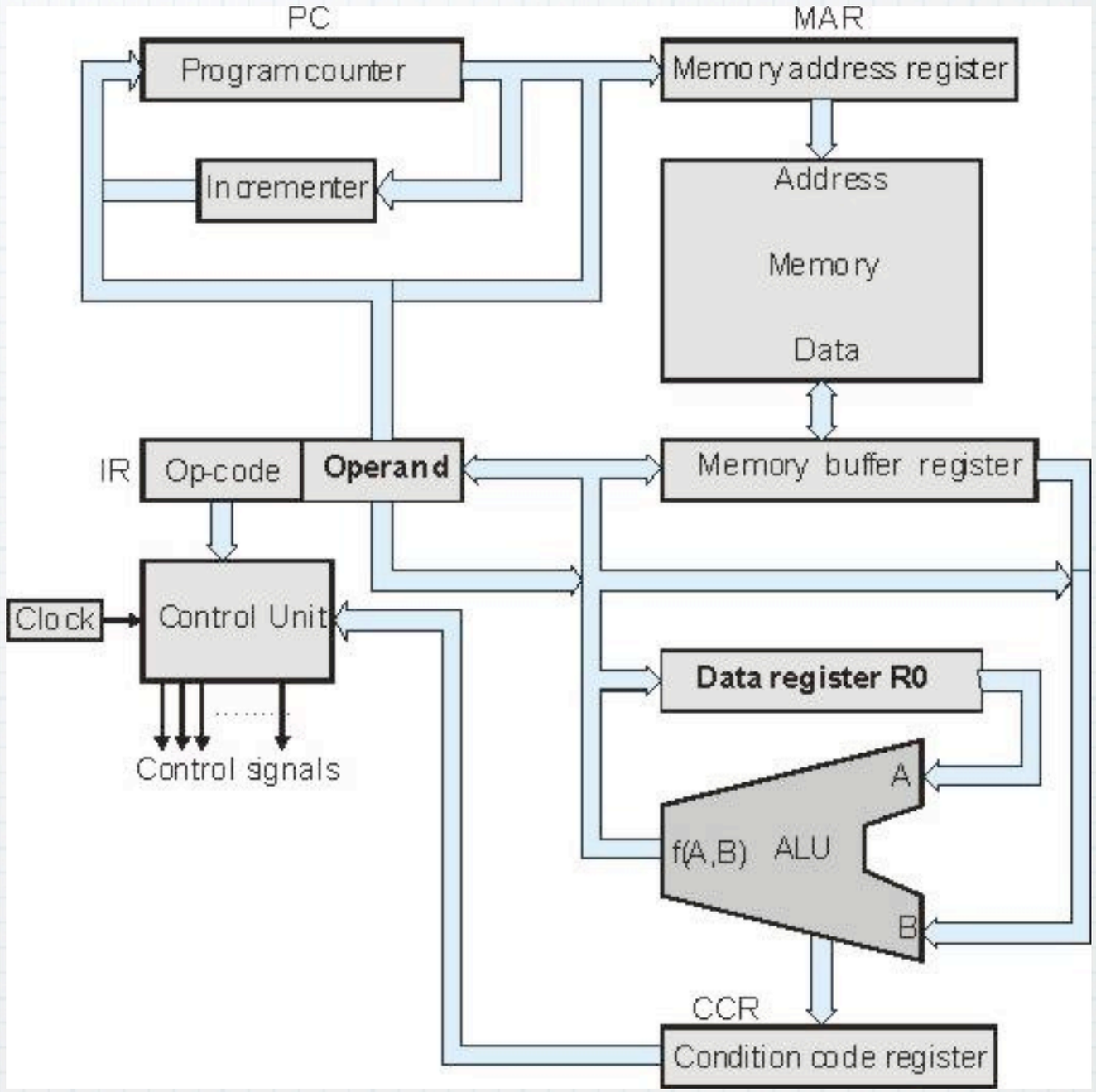
MIPS32 Add Immediate Instruction

001000	00001	00010	0000000101011110
OP Code	Addr 1	Addr 2	Immediate value

Assembleur

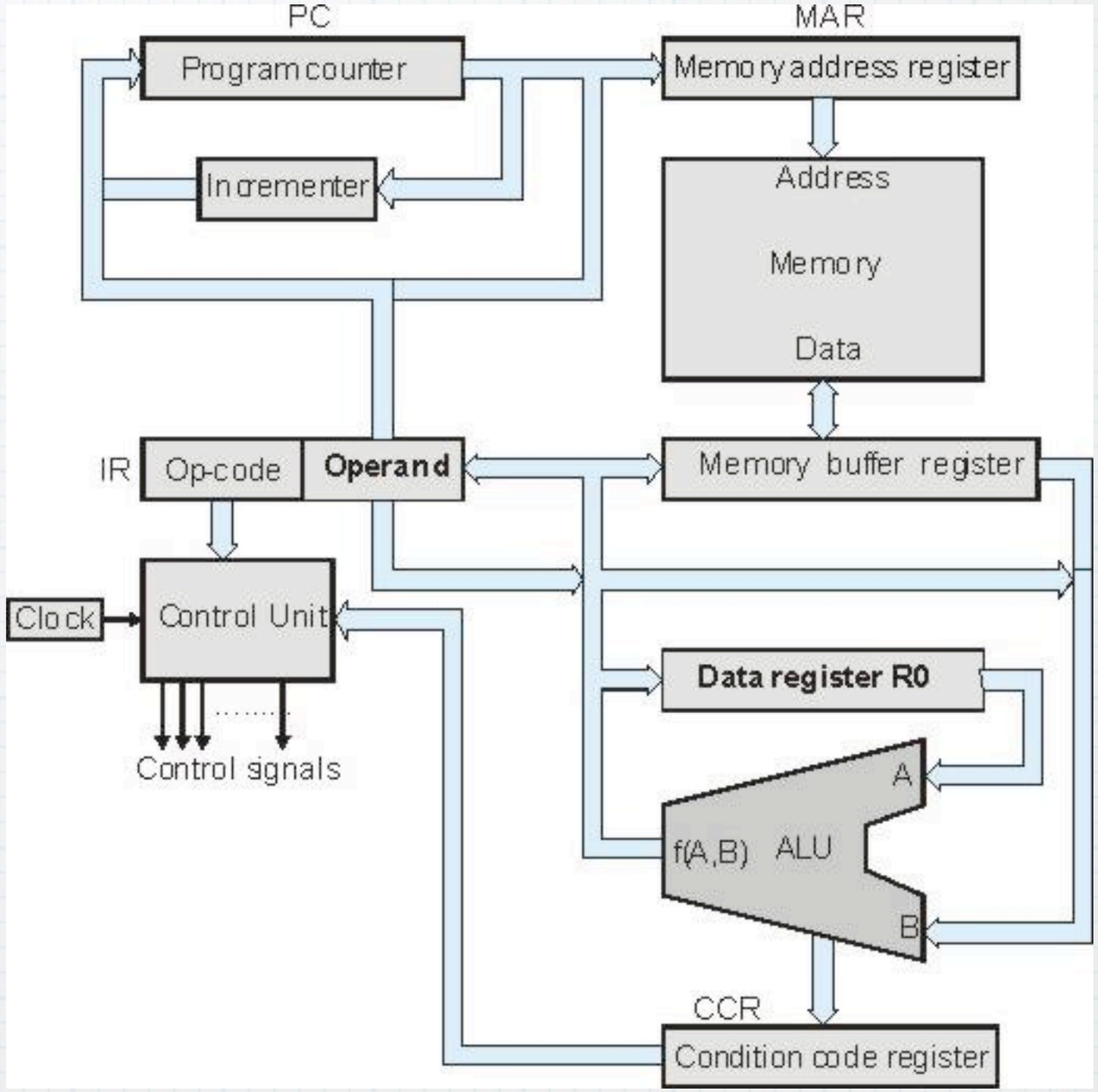
adresse	contenu
0000	01010111
0001	00010001
0010	00110100
0011	00000110
0100	01110110
0101	11110000
0110	00011110
0111	00010100

adresse	contenu
0	LOAD 7
1	ADD #1
2	JNZ 4
3	ADD 6
4	STO 6
5	STOP
6	30
7	20



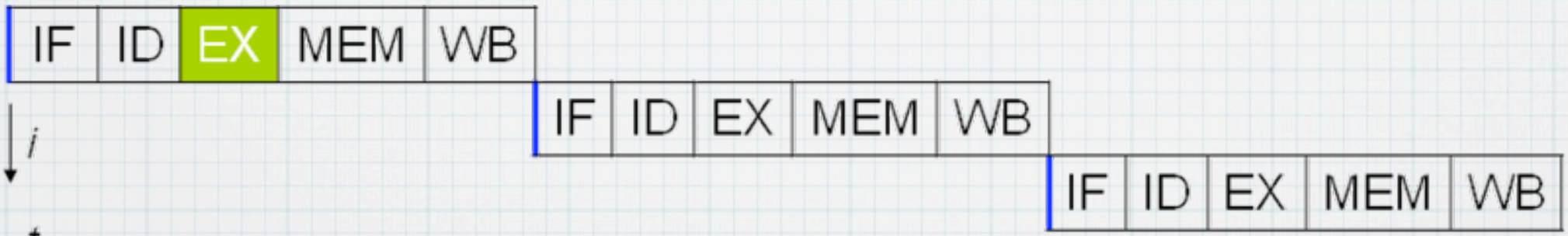
Assembleur

0	Load 13
1	Sto 10
2	Load 11
3	Add 12
4	Sto 11
5	Load 10
6	Add #-1
7	Sto 10
8	Jnz 2
9	Stop
10	0
11	0
12	2
13	3



Pipeline

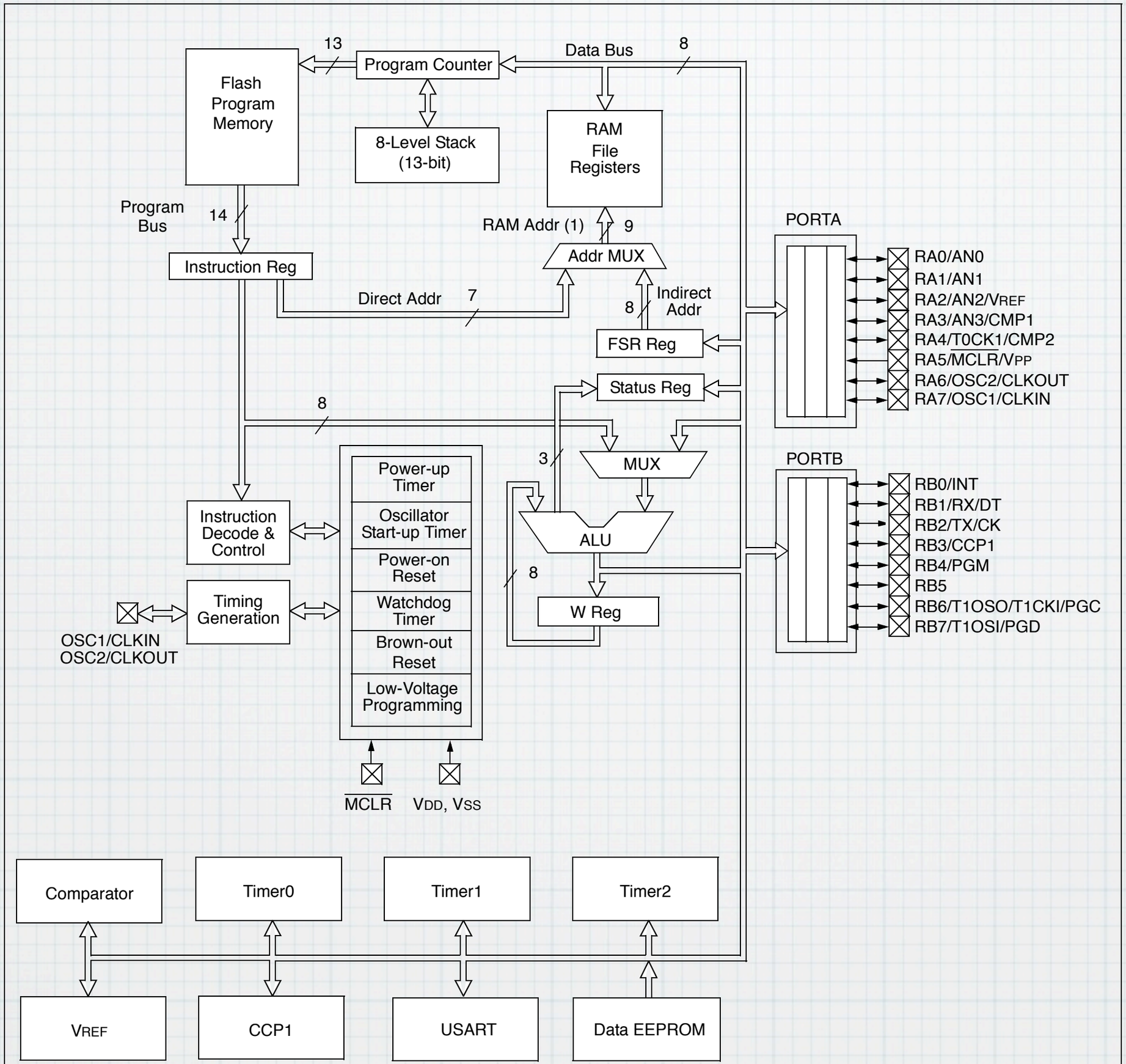
* Ajoute un peut de parallélisme
* Pré-décode les instructions suivantes.
* Pré-charge les valeurs.
* Fait gagner du temps



Caractéristiques

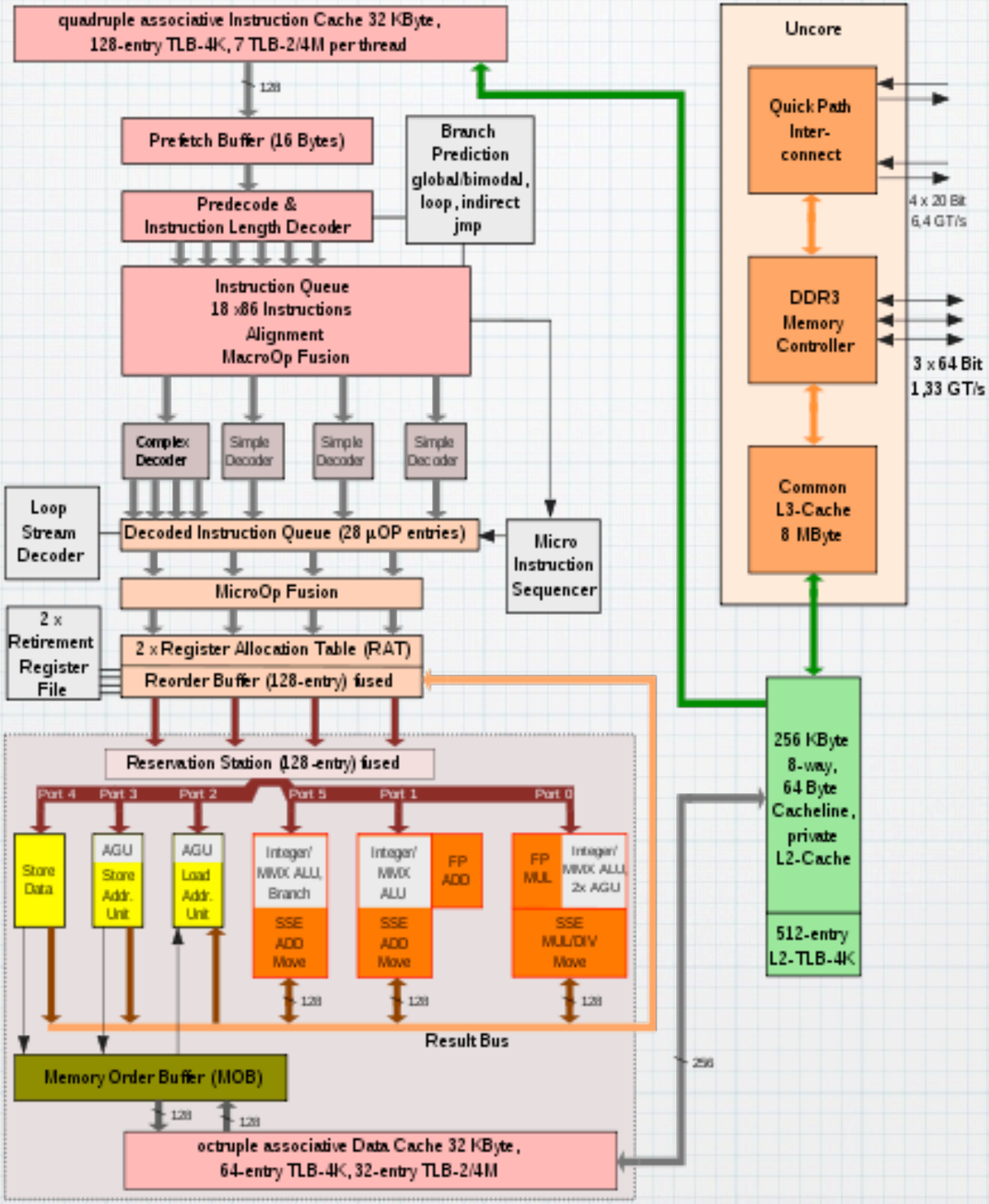
* la largeur de ses registres internes de manipulation de données (8, 16, 32, 64, 128) bits
* la cadence de son horloge exprimée en MHz (mega hertz) ou GHz (giga hertz)
* le nombre de noyaux de calcul (core)
* son jeu d'instructions
* sa finesse de gravure exprimée en nm (nanomètres) et sa microarchitecture.

FIGURE 3-1: BLOCK DIAGRAM



Note 1: Higher order bits are from the Status register.

Intel Nehalem microarchitecture

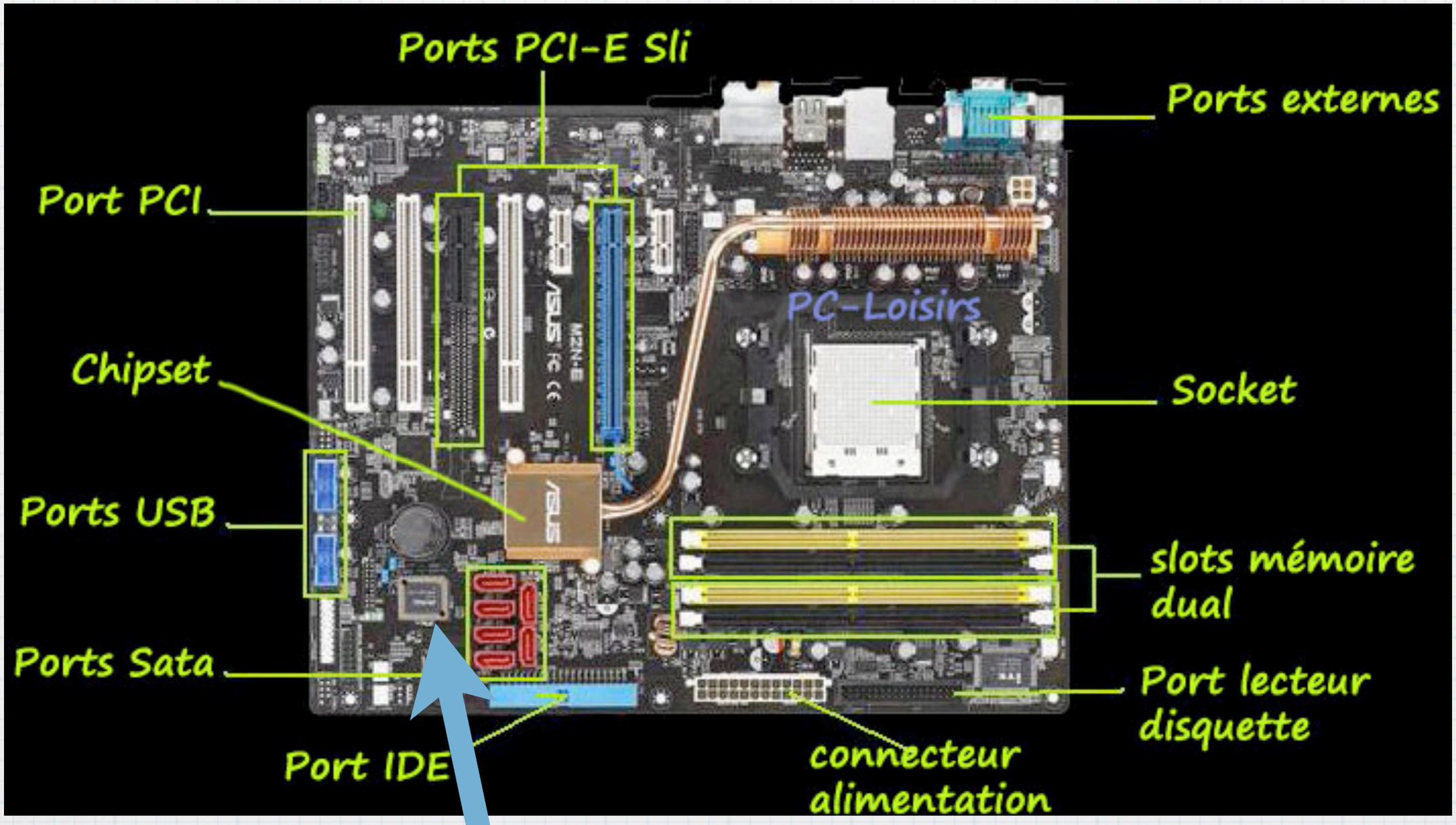


GT/s: gigatransfers per second

Séquence d'amorçage

* Iniatialisation -> Compteur de programme à une valeur fixe indépendant des périphériques installés.
* Correspond à un petit programme dans une ROM, EEPROM, ou flash sur la carte mère.
* Il fait quelques vérifications :
    * Alimentation
    * Processeur
    * Mémoire
    * Horloge, Contrôleur des interruptions
    * Clavier, Carte graphique
    * Recherche de secteur de boot dans l'ordre configuré

Carte mère



EEPROM

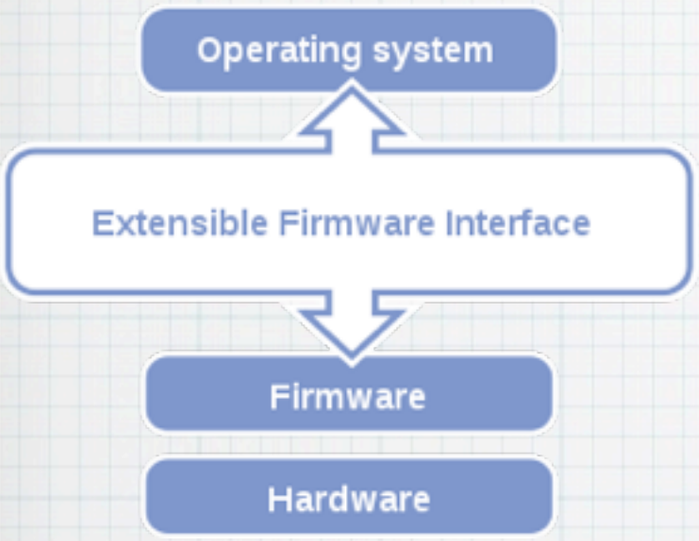
Séquence de boot

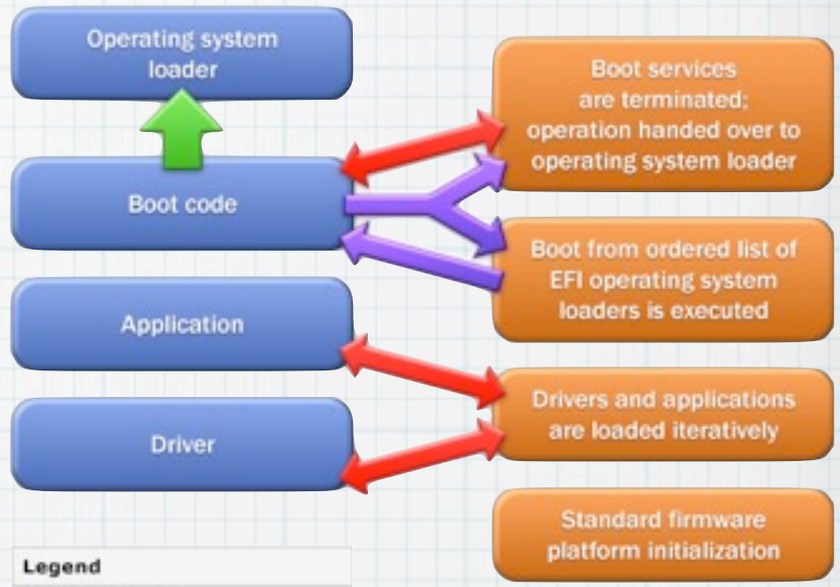
* Basic Input/Output System (BIOS - 1981)
  * Inventaire du matériel (CPU, RAM, vidéo, HDD...)
  * Vérification de l'état de fonctionnement (POST)
  * Si aucune erreur recherche d'un périphérique contenant un System d'Exploitation et lancement de celui-ci

Très Limité

* Ecrit pour du 16bit, possède 1Mio adressable.
* MBR (4 partition actives, 2,2 Tio par partition)
* Trop basique

Extensible Firmware Interface





* UEFI (2005) U= Unified
* Existe en 32 bit ou 64 bit
* Capable d'héberger des applications et des pilotes.
* Prise en charge du GPT. 9,4 Zio ($1\,0^{21}$)
* Devient compliqué
* Itanium, mac intel ...