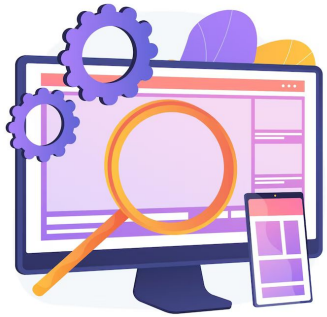
Du process au transistor

le dessous des programmes

15/02/2023 - Julien Rollin

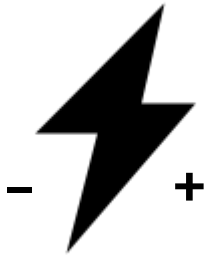
source : freepik.com



Voyage au centre de l'ordinateur !

“

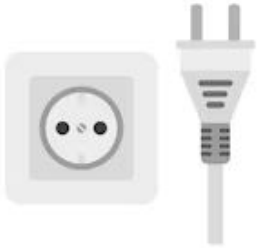
Image by freepik.com




```c
#include <stdio.h>
int main() {
   printf("Hello, World!");
   return 0;
}
```

```
$ Hello, World!
```

Quel est le lien ?

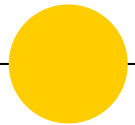


Tout commence par un signal électrique !

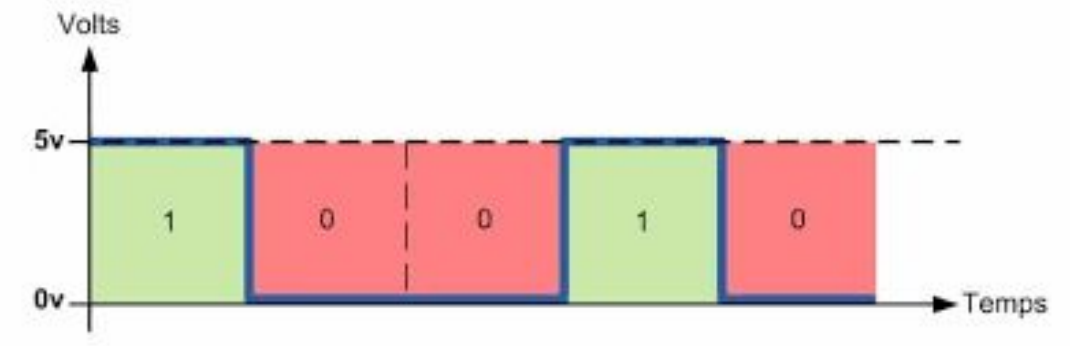
“

Image by freepik.com

Volts et bits



Conversion du signal en binaire



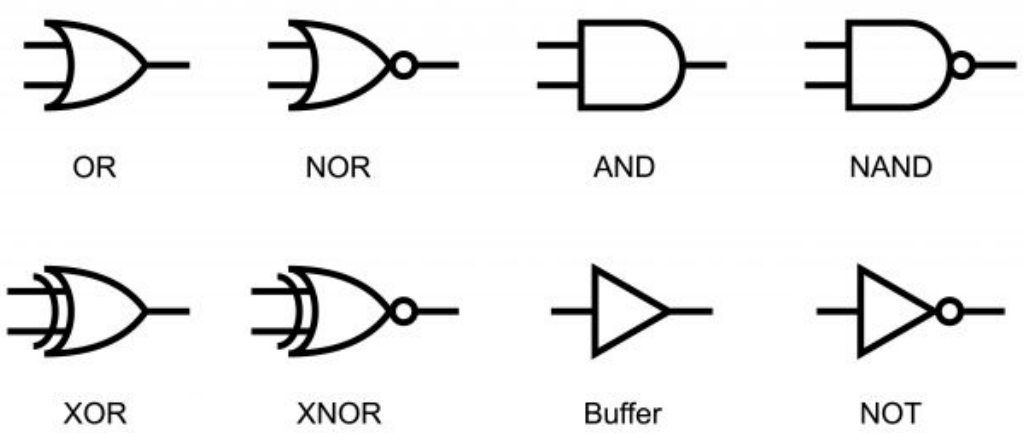
Nb: La tension varie constamment

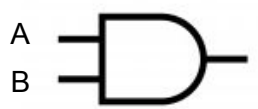
source: fingerinthenet

Révolution du transistor

- créé en 1947
- Trois physiciens à son origine, John Bardeen, Walter Brattain et William Shockley : prix Nobel de la physique de 1956
- Permet la construction simple de portes logiques, ils ont permis d'intégrer la logique binaire

Portes logiques





AND

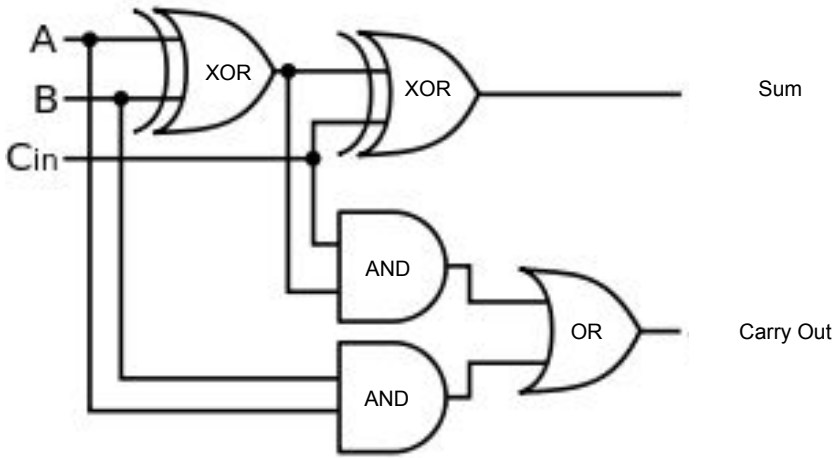
A	B	Sortie
0	0	0
1	0	0
0	1	0
1	1	1

Addition binaire

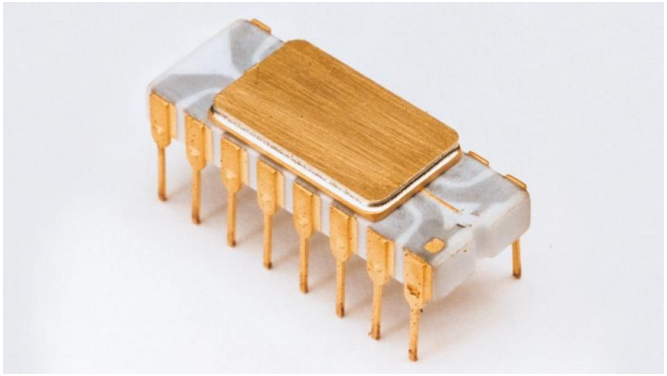
	Binary Representation				Decimal
Carry	1	1	0		1
1st number	0	1	1	0	6
2nd number	0	1	1	1	7
Sum	1	1	0	1	13

source : watson.latech.edu

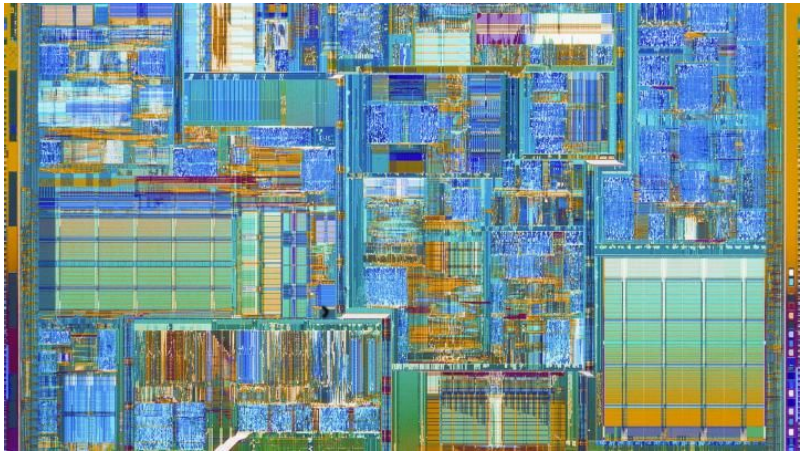
Addition avec des portes logique



Miniaturisation des circuits



1971, the Intel® 4004 processor, 2,300 transistor



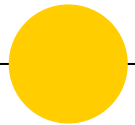
2010, Intel® Core™ processor 32 nm, 560 million transistors

C’est parfait pour un ordinateur !

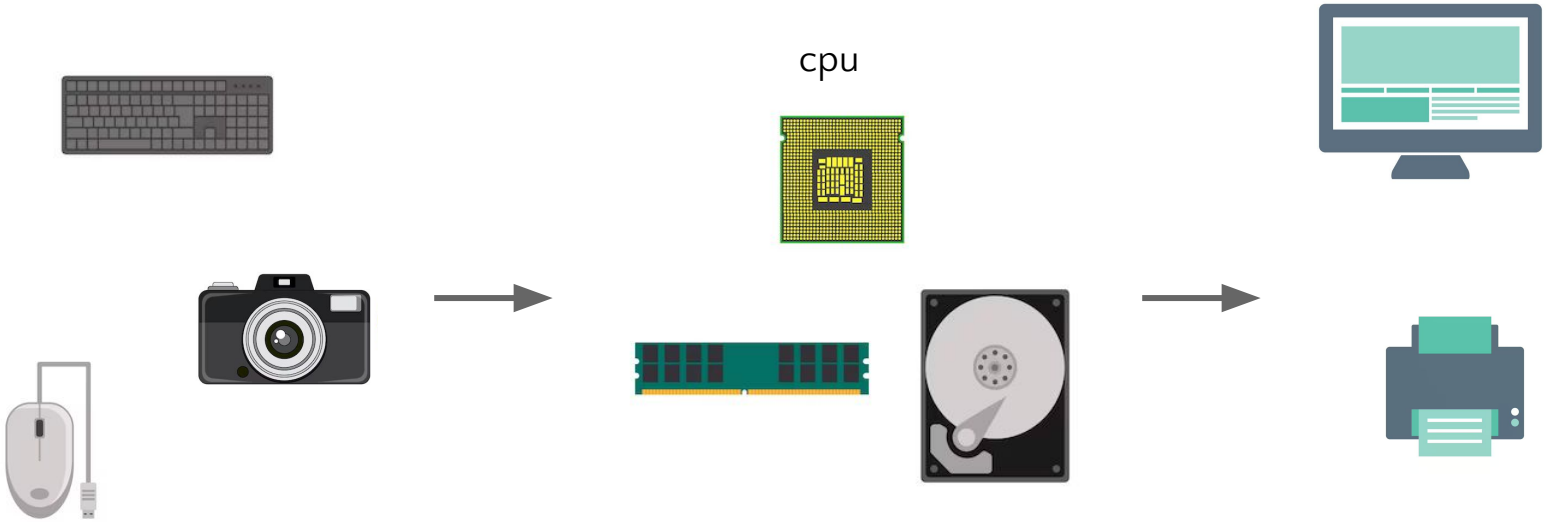
“

Image by freepik.com

Composition d’un Ordinateur



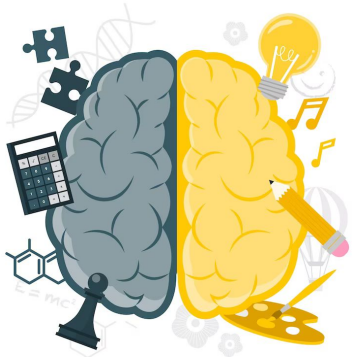
Les composants d'un ordinateur



Périphériques d'entrée

mémoire
(volatile ou pas)

Périphériques de sortie

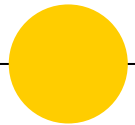


Qui est le cerveau des opérations ?

“

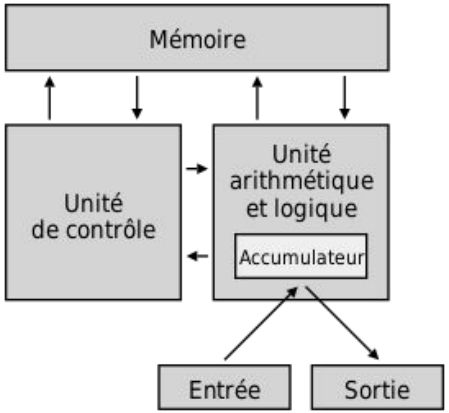
Image by freepik.com

Le processeur



Origine : Architecture Von Neumann

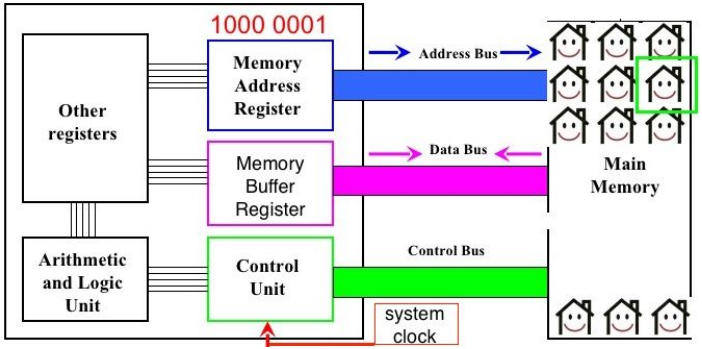
- Référence au mathématicien John von Neumann (juin 1945)
- L'unité de contrôle
  - ordonnancer les instructions
  - envoyer tous les calculs à effectuer à l'unité arithmétique et logique (ALU)
- La mémoire
  - stocke à la fois les instructions et les données.
    - utilisée par l'unité de contrôle pour stocker les séquences d'instructions
    - utilisé par l'ALU pour stocker les données d'entrée d'un calcul et le résultat.
- Entrées/Sorties
  - Il s'agit de toutes les interfaces permettant d'interagir avec l'ordinateur. Classiquement un clavier peut être vu comme une entrée et un écran comme une sortie.



la première description d'un ordinateur dont le programme est stocké dans sa mémoire.

Composants d'un CPU

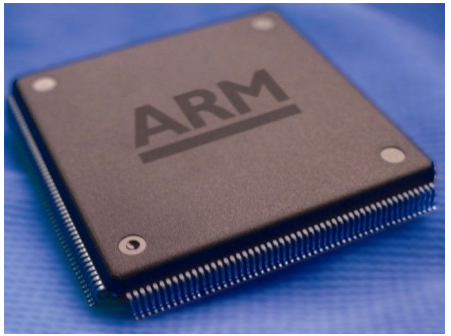
- Bus :
  - nappes transportant l'info
- Registres :
  - emplacements de stockage à accès rapide
- UAL :
  - unité de calcul (entiers et booléens)
- Unité de contrôle :
  - interprète les instructions
- Horloge :
  - cadence les instructions



source : bournetocode.com

Différentes Architectures

- x86 (Intel, AMD) : PC et compatibles
- ARM : beaucoup de téléphone
- PowerPC (IBM) : Wii, PS3, XBox 360
- MIPS, etc.



Instruction machine

- Instruction Set Architecture (ISA): modèle qui définit comment le logiciel contrôle le hardware (code binaire)
- chaque processeur possède son propre jeu d'instructions machine
  - ARM is RISC (Reduced Instruction Set Computing)
  - x86 is CISC (Complex Instruction Set Computing)
- chaque instruction possède un identifiant numérique : opcode
- La programmation directe du processeur avec les instructions machines est difficile, fastidieuse et quasi-impossible à comprendre

IBM System/360 Reference Data

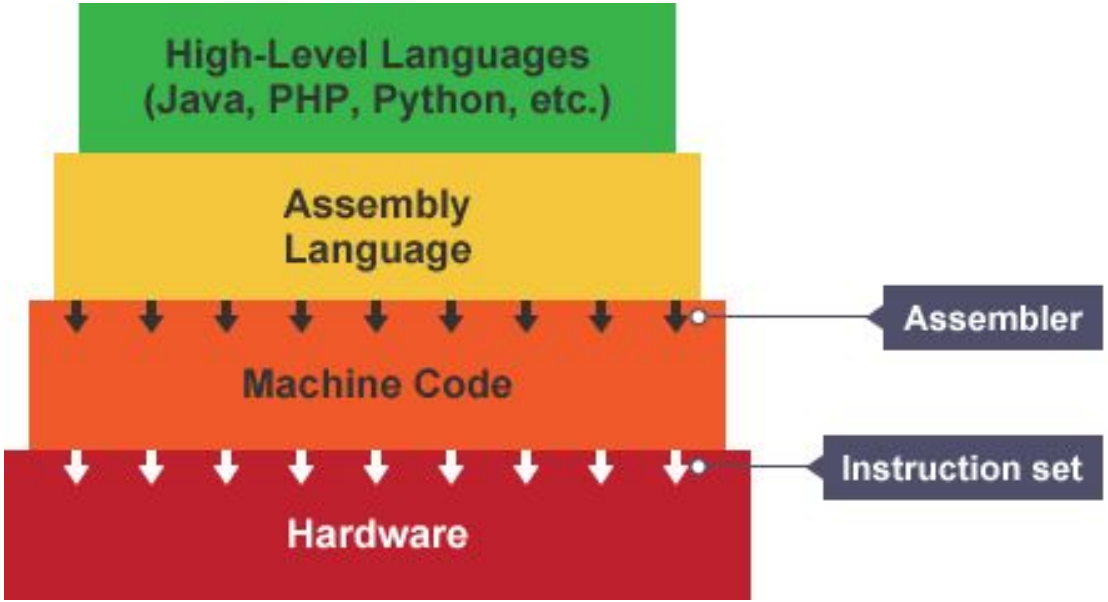
MACHINE INSTRUCTIONS

NAME	MNEMONIC	OP CODE	FOR-MAT	OPERANDS
Add (c)	AR	1A	RR	R1,R2
Add (c)	A	5A	RX	R1,D2(X2,B2)
Add Decimal (c,d)	AP	FA	SS	D1(L1,B1),D2(L2,B2)
Add Halfword (c)	AH	4A	RX	R1,D2(X2,B2)
Add Logical (c)	ALR	1E	RR	R1,R2
Add Logical (c)	AL	5E	RX	R1,D2(X2,B2)
AND (c)	NR	14	RR	R1,R2
AND (c)	N	54	RX	R1,D2(X2,B2)
AND (c)	NI	94	SI	D1(B1),I2
AND (c)	NC	D4	SS	D1(L,B1),D2(B2)
Branch and Link	BALR	05	RR	R1,R2
Branch and Link	BAL	45	RX	R1,D2(X2,B2)
Branch and Store (e)	BASR	0D	RR	R1,R2
Branch and Store (e)	BAS	4D	RX	R1,D2(X2,B2)
Branch on Condition	BCR	07	RR	M1,R2
Branch on Condition	BC	47	RX	M1,D2(X2,B2)
Branch on Count	BCTR	06	RR	R1,R2
Branch on Count	BCT	46	RX	R1,D2(X2,B2)
Branch on Index High	BXH	86	RS	R1,R3,D2(B2)
Branch on Index Low or Equal	BXLE	87	RS	R1,R3,D2(B2)
Compare (c)	CR	19	RR	R1,R2
Compare (c)	C	59	RX	R1,D2(X2,B2)

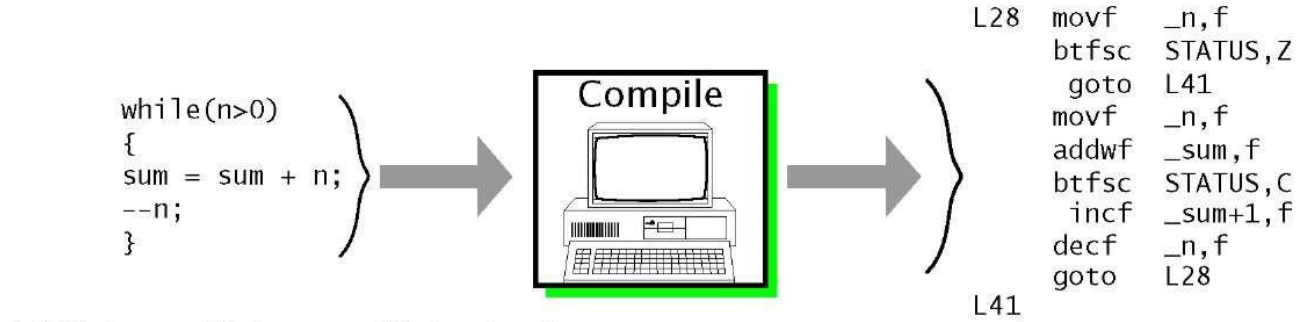
Exemple d'instruction machine pour processeur IBM 360 - Main frame

Le Langage assembleur

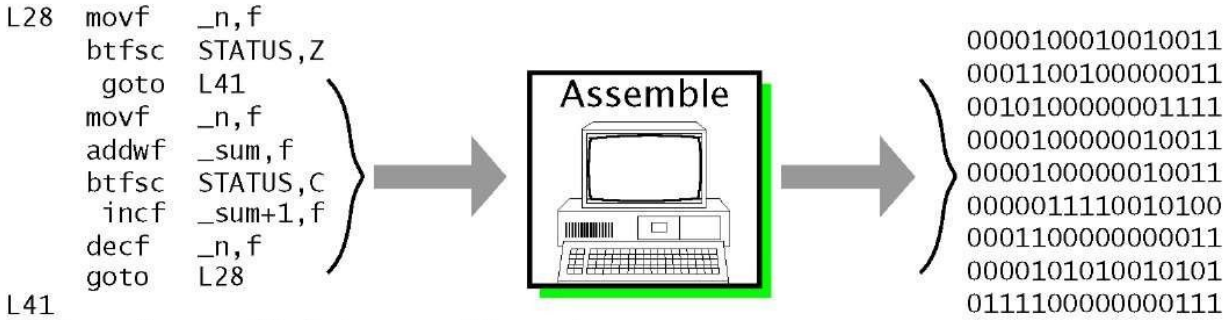
- langage machine compréhensible par l'humain
- seulement quelques instructions très simples (déplacer une donnée, multiplier, additionner, etc.)
- Les registres sont des petites zones mémoire dans le processeur où sont stockées les variables d'entrée ou de sortie d'un calcul.
  - seulement quelques registres nommés AX, BX, CX, DX, etc.
  - taille étant extrêmement limitée,
  - leur tâche est de stocker une donnée uniquement le temps du calcul (quelques fractions de secondes) et d'envoyer le résultat en mémoire vive par la suite, quitte à récupérer le résultat dans la mémoire vive pour un calcul ultérieur.



On va utiliser un langage intermédiaire, l'assembleur !



(a) First, compile to assembly-level code.

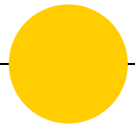


(b) Second, assemble-link to machine code.

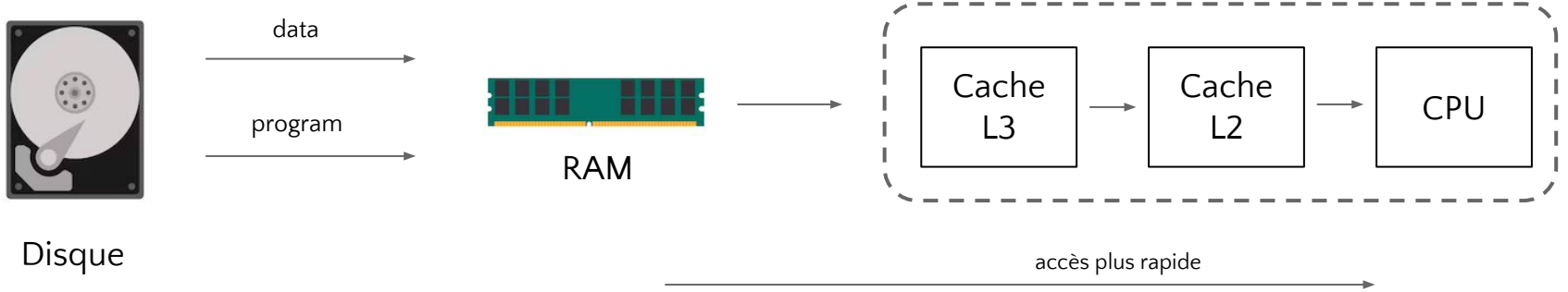
Workflow

source: what-when-how.com

Processeur et mémoire

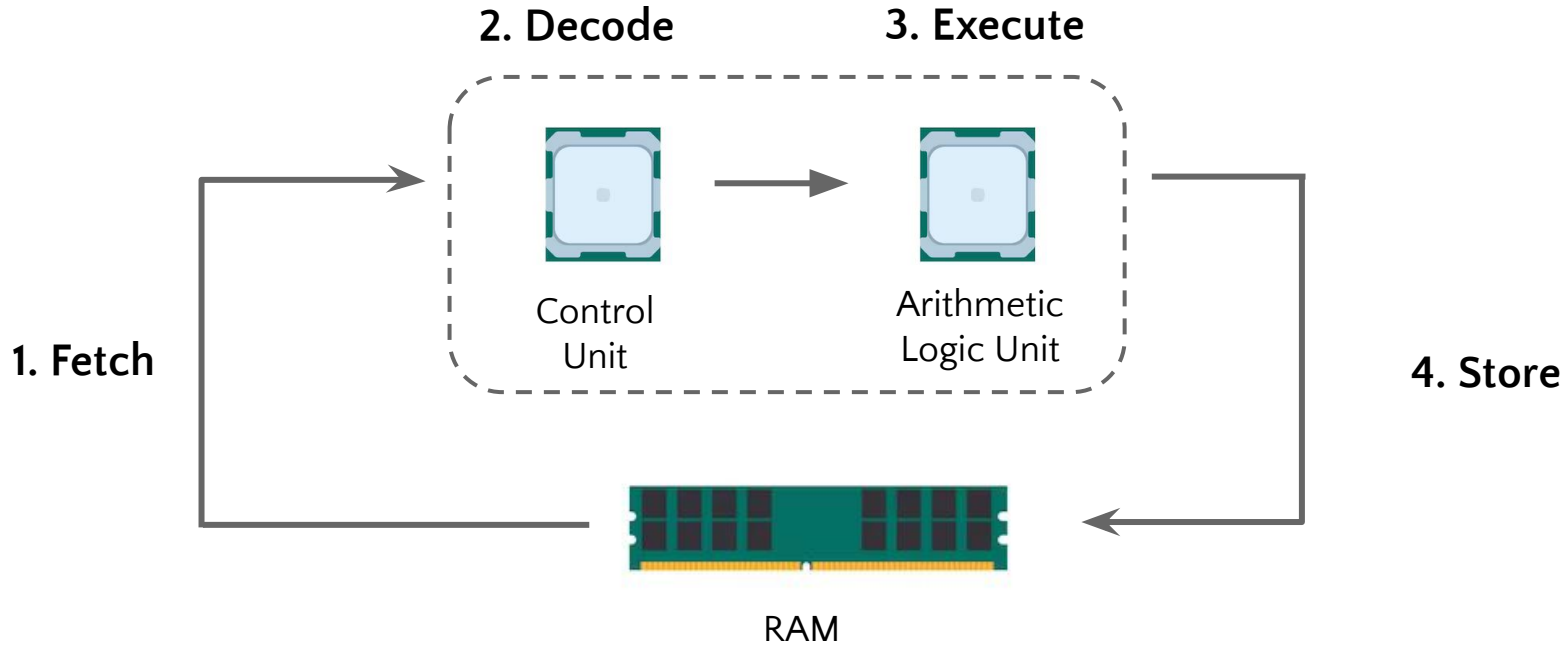


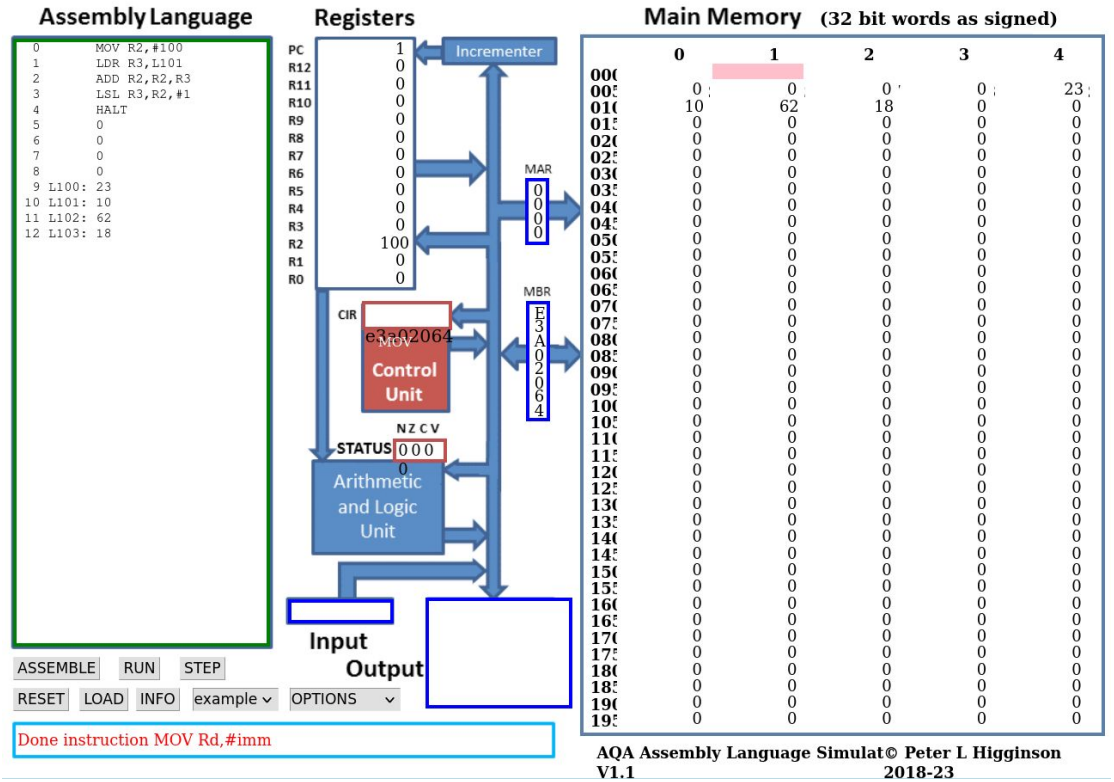
Les données sont stockées dans la RAM



Le cpu met les données et programmes
au plus près de lui pour être le plus performant possible

Instruction Cycle





[Simulateur d'instructions]

On peut manipuler la mémoire depuis nos programmes ?

```
(gdb) run

Starting program: /tmp/binary.bin



Program received signal SIGSEGV, Segmentation fault.

0x0000000000400557 in main (argc=1, argv=0x7fffffffe458) at file.c:6

6           printf("hello %d",*boom);

(gdb)
```

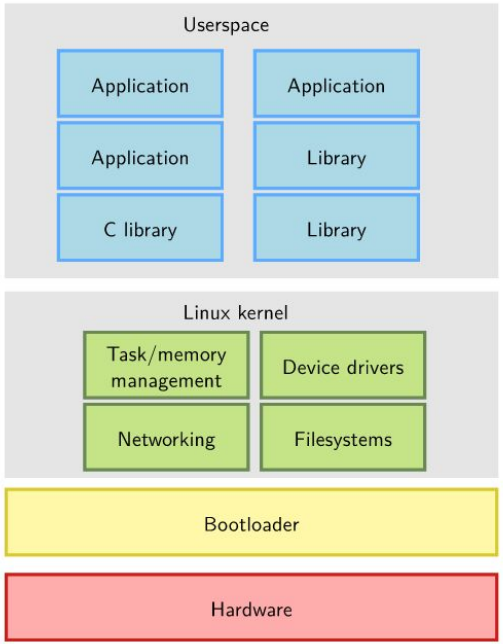
Segmentation Fault

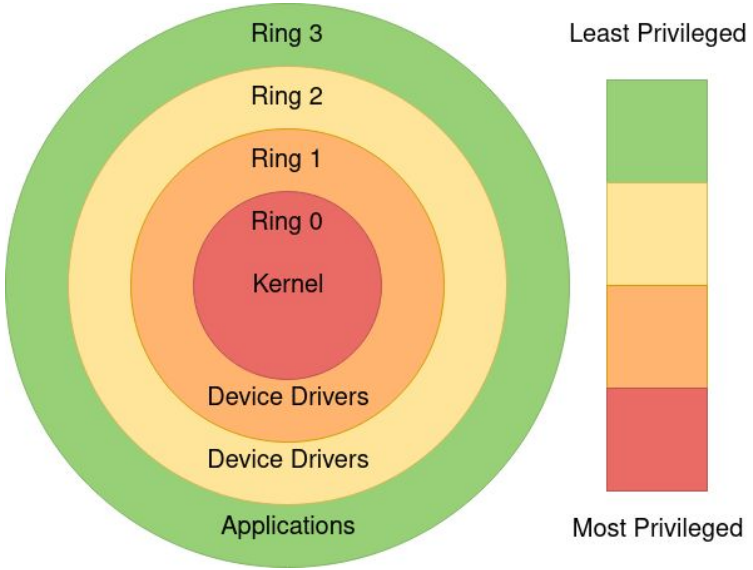
Système

Démarrage (Aka Boot)

- charge la configuration matérielle (cpu, memory, disques, etc)
- charge le kernel
- exécute le kernel

User land vs Kernel Land





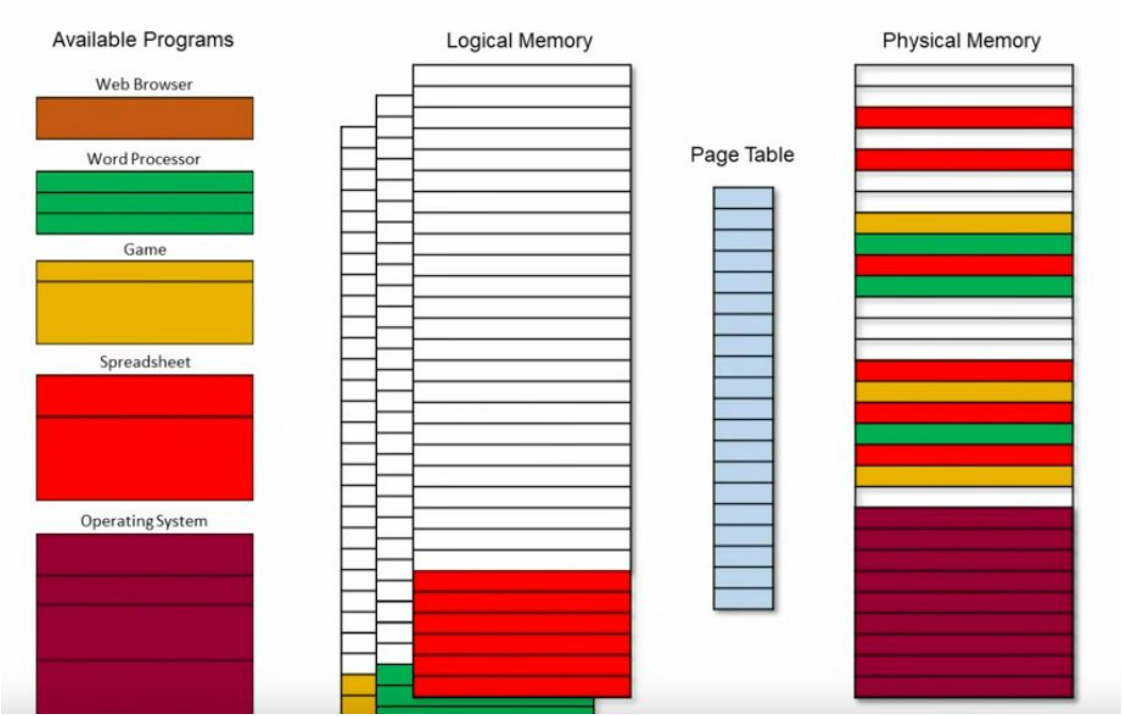
Pourquoi séparer Kernel Land et User land ?

Responsabilités du Kernel

- Process :
  - le kernel détermine quel process est autorisé à utiliser le CPU
- Memoire
  - le kernel assure le suivi de toutes les allocations mémoire par process, ce qui devrait être partagé entre les process, ce qui est disponible
- Drivers
  - le kernel assure l'interface entre le matériel et les process
- System calls
  - les process utilisent les sysCall pour communiquer avec le kernel

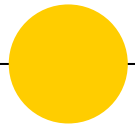
Memory Management Unit (MMU)

- les process n'ont pas accès à l'endroit où sont stockées physiquement les données
- le kernel fait en sorte que chaque process ait l'impression d'avoir la machine pour lui
- le MMU intercepte les accès mémoire et utile une table de correspondance (pageTable) entre la mémoire virtuelle et la mémoire physique



Process, mémoire virtuelle et mémoire physique

Les éléments clés



Langage Haut niveau

```
fn virt_device_io<T: GuestMemory>(mem: &T) {
    let sample_buf = &[1, 2, 3, 4, 5];
    mem.write(sample_buf, GuestAddress(0xffc));
}
```

Langage Assembleur

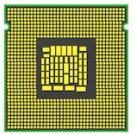
```
mov $0x3f8, %dx
add %bl, %al
```

Langage machine

```
0xba, 0xf8, 0x03
0x00, 0xd8
```

Matériel

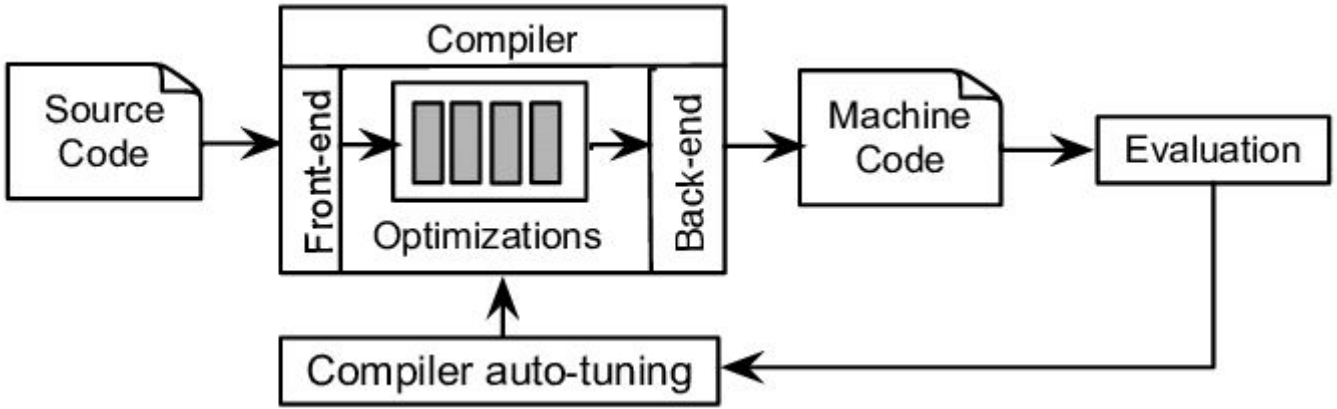
```
microcode
```



Bas vs Haut niveau

Le choix du langage et compilateur a un impact sur les performances

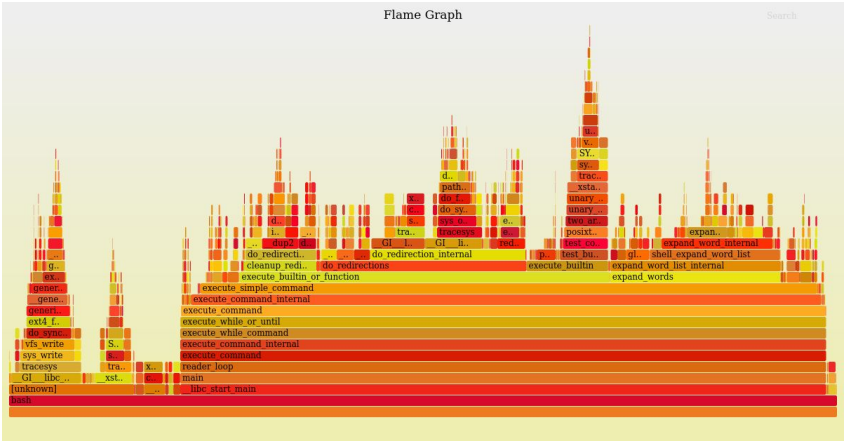
L'optimisation du compilateur



La façon d'interagir avec son système a des impacts sur les performances

Debug des process avec Strace

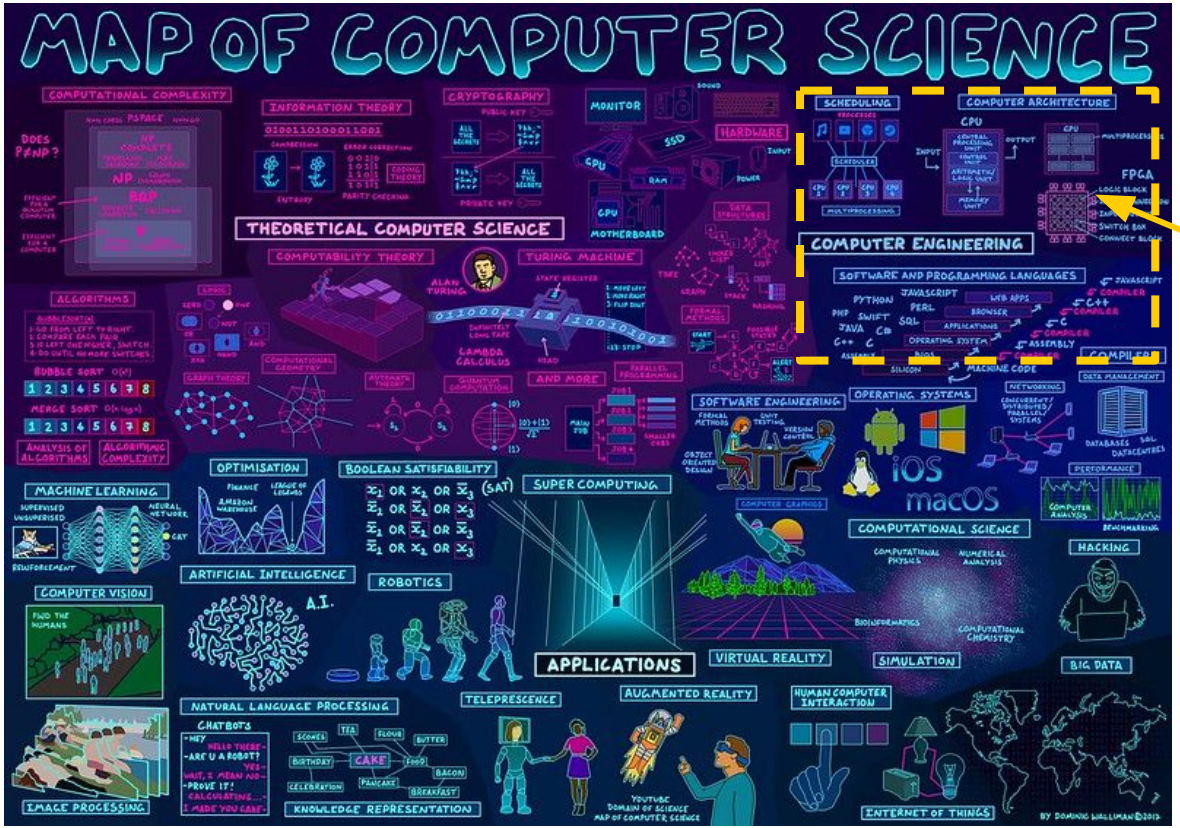
Desktop % time	Documents seconds	Downloads usecs/call	Music calls	Pictures errors	Public syscall	Templates	Videos	vscode
0.00	0.000000	0	7		read			
0.00	0.000000	0	1		write			
0.00	0.000000	0	11		close			
0.00	0.000000	0	10		fstat			
0.00	0.000000	0	27		mmap			
0.00	0.000000	0	8		mprotect			
0.00	0.000000	0	1		munmap			
0.00	0.000000	0	3		brk			
0.00	0.000000	0	2		rt_sigaction			
0.00	0.000000	0	1		rt_sigprocmask			
0.00	0.000000	0	2		ioctl			
0.00	0.000000	0	2	2	access			
0.00	0.000000	0	1		execve			
0.00	0.000000	0	2	2	statfs			
0.00	0.000000	0	1		arch_prctl			
0.00	0.000000	0	2		getdents64			
0.00	0.000000	0	1		set_tid_address			
0.00	0.000000	0	9		openat			
0.00	0.000000	0	1		set_robust_list			
0.00	0.000000	0	1		prlimit64			
100.00	0.000000		93	4	total			



Operation Time	in ns Time	in ms Time	Memo
L1 cache reference	1		
Branch misprediction	3		
L2 cache reference	4		
Mutex lock/unlock	17		
Main memory reference	100		~100GB/sec, 100x L1 cache
Send 2 kB over 10 Gbps network	1,600	0.0016	
Compress 1 kB with Zippy	2,000	0.002	
Read 1 MB sequentially from memory	**10,000**	**0.010**	
Round trip within same datacenter	500,000	0.5	
Read 1 MB sequentially from SSD	**1,000,000**	**1**	**~1GB/sec SSD, 100x main memory**
Read 1 MB sequentially from disk	5,000,000	5	~200MB/sec HDD, 500x main memory
Read 1 MB sequentially from 1Gbps network	10,000,000	10	
Disk seek	**10,000,000**	**10**	**20x datacenter round trip**
TCP packet round trip between continents	150,000,000	150	

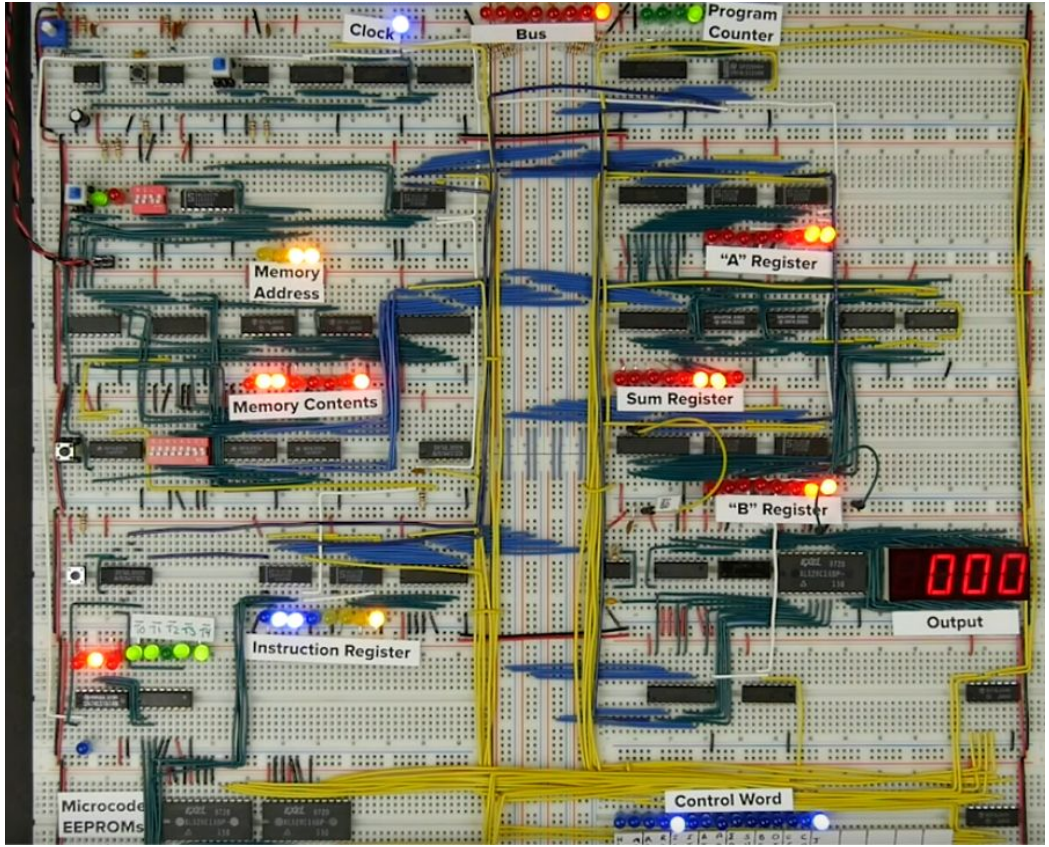
Ordre de grandeur en temps

Le sujet est une science à part entière !



Computer Architecture

Computer Science



[Build an 8-bit computer from scratch](#) - Ben eater

“Il n'y a pas de bonne ou de mauvaise architecture vous savez..”

Du process au transistor

Merci ! ❤️

N'hésitez pas à laisser
des commentaires !

 jrollin  julienrollin  julienrollin.com