Voyage au coeur d'un programme exécutable

EPISODE 1

BTS SN-IR LaSalle Avignon (tvaira)

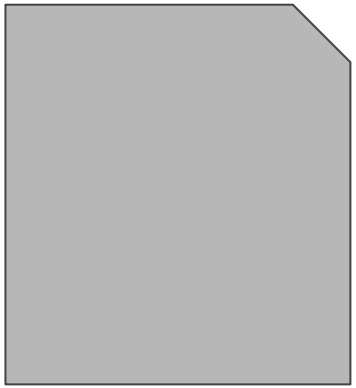
Fabrication

Code source

```
int main()
{
   int a, b, c;
   a = 1;
   b = 2;
   c = a + b;
   return 0;
}
```

Fichier “texte”

Code objet
code “machine”

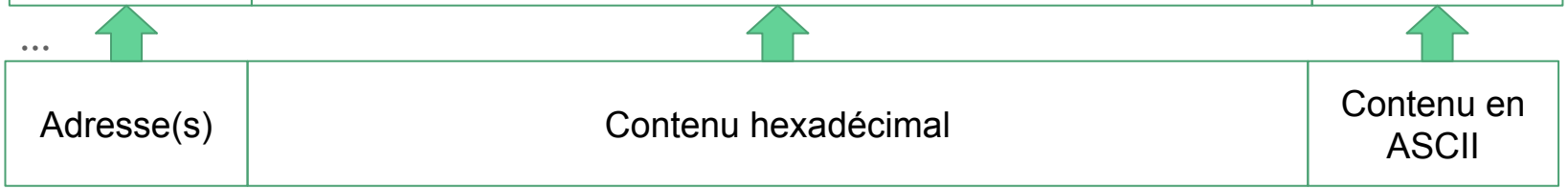


Fichier binaire

Affichage du contenu d'un fichier binaire

$ hexdump -C a.out

```
00000000  7f 45 4c 46 02 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  03 00 3e 00 01 00 00 00  f0 04 00 00 00 00 00 00  |..>.............|
...
```



Affichage du contenu “texte” d’un fichier binaire

$ strings a.out

/lib64/ld-linux-x86-64.so.2

libc.so.6

...

GLIBC_2.2.5

...

GCC: (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0

...

Réduire la taille d’un exécutable

$ ls -l a.out

-rwxr-xr-x 1 tv tv 8168 sept. 17 12:54 a.out

$ strip a.out

$ ls -l a.out

-rwxr-xr-x 1 tv tv 6056 sept. 17 14:28 a.out

Détails sur le type d’un fichier

$ file a.out

a.out: **ELF 64-bit** LSB shared object, x86-64, version 1 (SYSV), **dynamically linked**, interpreter /lib64/l, for **GNU/Linux 3.2.0**, BuildID[sha1]=f5d56e3099eeeaf40bd9572ddaba0d63e0fef7ec, not stripped

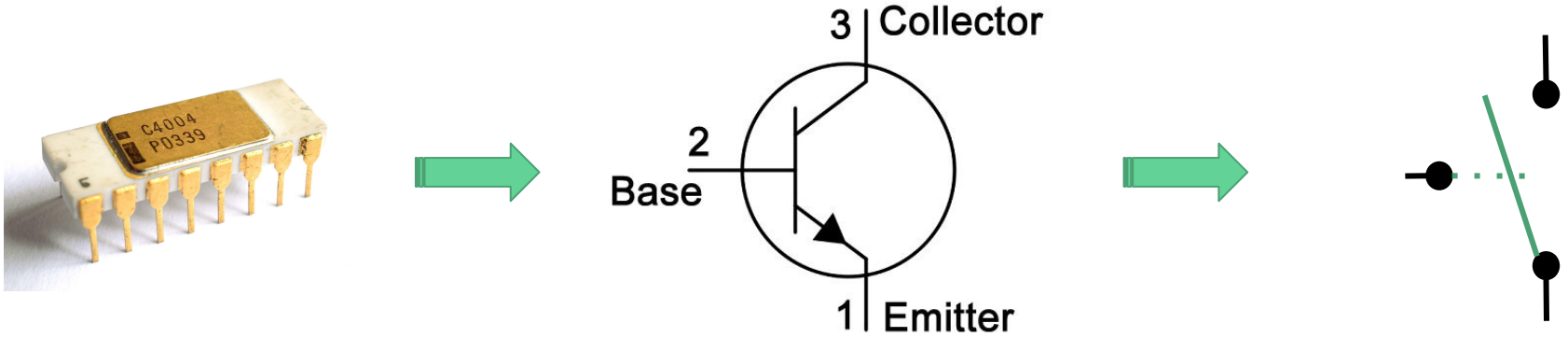
Liste des bibliothèques dynamiques

$ ldd a.out

```
linux-vdso.so.1 (0x00007ffc1d743000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f94d4cfe000)
/lib64/ld-linux-x86-64.so.2 (0x00007f94d52f1000)
```

Processeur (I)

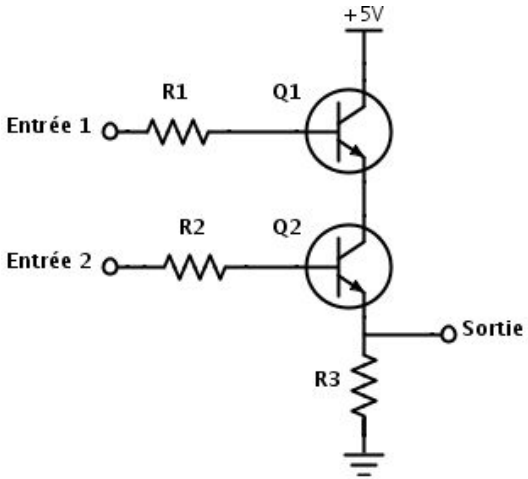
En électronique, un processeur est un ensemble de plusieurs milliers, millions ou milliards de transistors utilisés comme un interrupteur “commandé” à deux états.

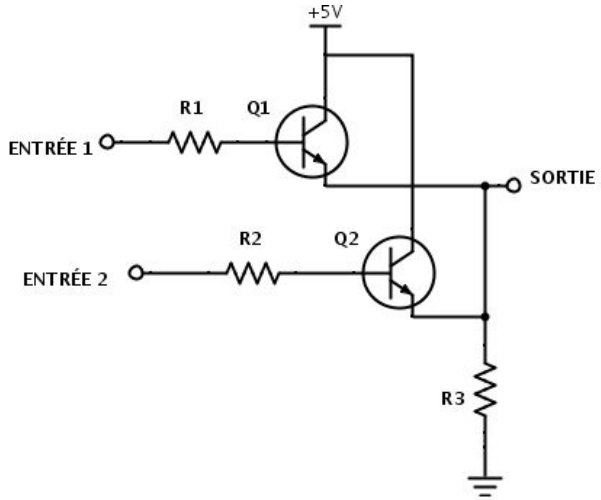


Le **4004** d'Intel est le premier microprocesseur commercialisé. Il intègre **2 300 transistors**. Avec une puissance d'exécution de **92 600 opérations par seconde** à une fréquence maximale de **740 kHz**, il est comparable à l'ENIAC, le premier ordinateur moderne dévoilé en 1946, qui occupait 167 m2 pour un poids total de 30 tonnes.

Processeur (II)

Les transistors sont “câblés” afin de réaliser des **opérations** (arithmétiques et logiques). Par exemple :

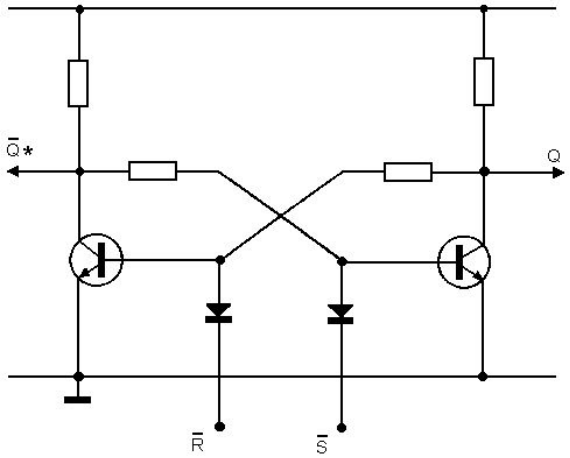




Cette partie du processeur se nomme l’unité arithmétique et logique (**UAL** ou ALU).

Processeur (III)

Les transistors sont “câblés” afin de réaliser des bascules (RS, D) pour **mémoriser** des bits. Par exemple :



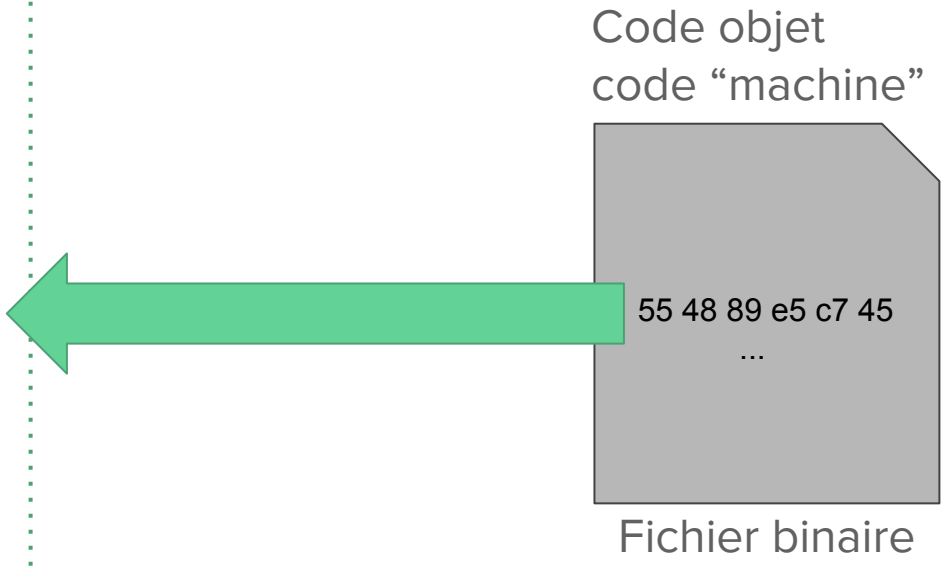
Cette partie du processeur constitue des **registres de *n* bits**.

Langage machine

- C’est le seul langage que le processeur puisse traiter.
- C’est une suite de bits interprétée par le processeur.
- Il est composé d’instructions et de données à traiter codées en binaire.
- Un processeur possède nativement un jeu d’instructions composé d’opcode (code opération) et d’opérandes. Il existe des opcodes pour faire des opérations arithmétiques, logiques, etc ...
- Chaque instruction nécessite un certain temps (généralement un nombre de cycles d’horloge) pour s’exécuter.

Code objet (code “machine”)

```
55
48 89 e5
c7 45 f4 01 00 00 00
c7 45 f8 02 00 00 00
8b 55 f4
8b 45 f8
01 d0
89 45 fc
b8 00 00 00 00
5d
c3
```



Langage assembleur

- C’est le langage informatique le plus proche du langage machine
- C’est un langage de bas niveau qui représente le langage machine sous une forme lisible par un humain.
- Les combinaisons de bits du langage machine sont représentées par des mnémoniques (des symboles) faciles à retenir : **ADD** pour l’addition, **MOV** pour la copie de valeurs, etc ...

Déassemblage d'un code objet (code "machine")

$ objdump -d add.o

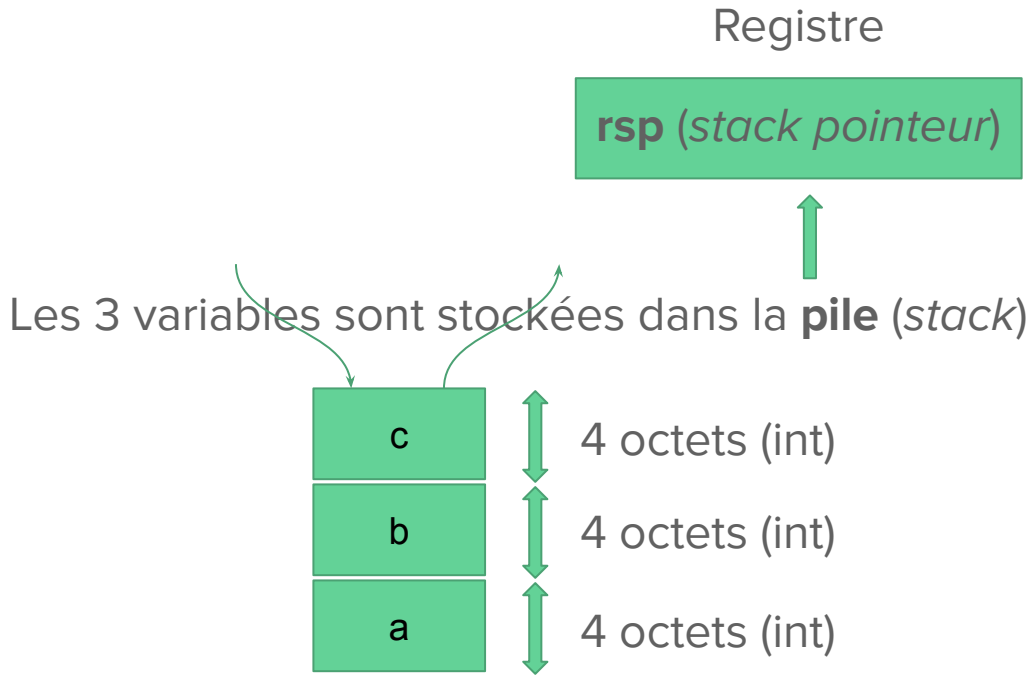
0000000000000000 <main>:

```
 0: 55                    push   %rbp
 1: 48 89 e5              mov    %rsp,%rbp
 4: c7 45 f4 01 00 00 00  movl   $0x1,-0xc(%rbp)
 b: c7 45 f8 02 00 00 00  movl   $0x2,-0x8(%rbp)
12: 8b 55 f4              mov    -0xc(%rbp),%edx
15: 8b 45 f8              mov    -0x8(%rbp),%eax
18: 01 d0                 add    %edx,%eax
1a: 89 45 fc              mov    %eax,-0x4(%rbp)
1d: b8 00 00 00 00        mov    $0x0,%eax
22: 5d                    pop    %rbp
23: c3                    retq
```

Explications (I)

Code source

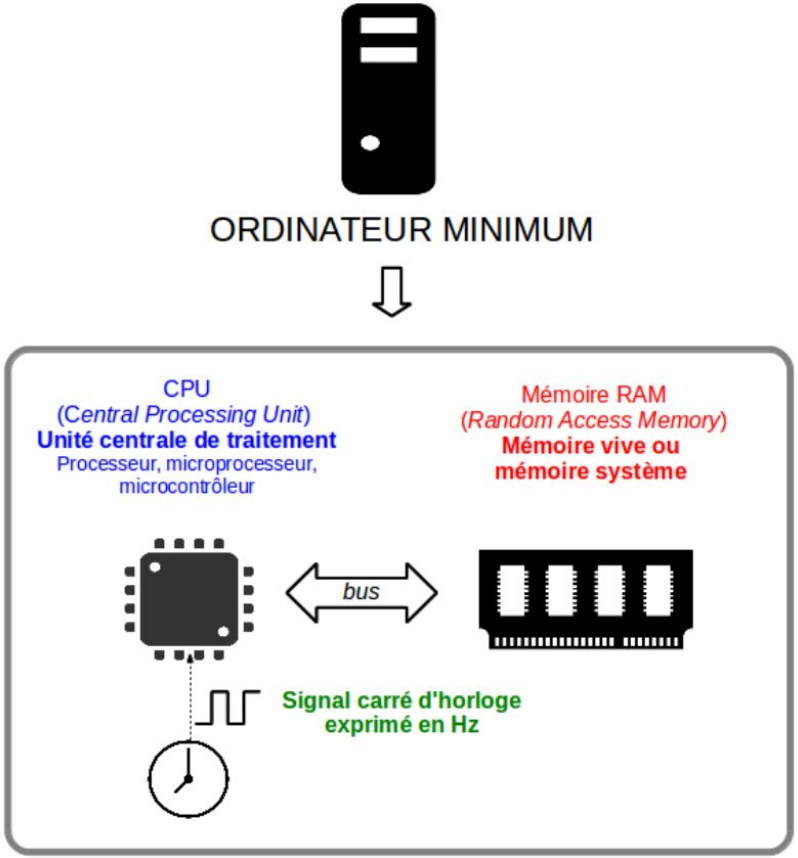
```
int main()
{
   int a, b, c;
   a = 1;
   b = 2;
   c = a + b;
   return 0;
}
```



Explications (II)

push %rbp	➜ empile (sauvegarde) le contenu de rbp (base pointer)
mov %rsp,%rbp	➜ copie le contenu de rsp (stack pointeur) vers rbp
movl $0x1,-0xc(%rbp)	➜ copie la valeur 1 dans la pile
movl $0x2,-0x8(%rbp)	➜ copie la valeur 2 dans la pile
mov -0x8(%rbp),%eax	➜ copie la valeur 2 de la pile vers le registre EAX
mov -0xc(%rbp),%edx	➜ copie la valeur 1 de la pile vers le registre EDX
add %edx,%eax	➜ additionne EDX et EAX et met le résultat dans EAX
mov %eax,-0x4(%rbp)	➜ copie le résultat contenu dans EAX dans la pile
mov $0x0,%eax	➜ copie la valeur 0 (valeur de retour du programme) dans EAX
pop %rbp	➜ dépile (restaure) l'ancienne valeur de rbp à partir de la pile
retq	

Ordinateur minimum



Définitions

- **Microprocesseur** (µP) : processeur construit en un seul circuit intégré
- **Microcontrôleur** (µC) : circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires, E/S, ...
- **Bus** : ensemble de liaisons (*n* fils)

  On distingue trois bus : le bus de **données**, le bus d'**adresse** et le bus de **commande** (qui gère les deux autres)

Micral (1973) Fr

Altair (1975) US

La suite au prochain épisode ...

Voyage au coeur d'un programme exécutable

EPISODE 2