Architecture matérielle

Table des matières

1 Du transistor au CPU

Histoire

À la base de la plupart des composants d'un ordinateur, on retrouve le transistor. Ce composant électronique a été inventé fin 1947par les Américains John Bardeen, William Shockley et Walter Brattain.
on ne trouve plus, depuis quelques temps déjà, des transistors en tant que composant électronique discret. Dans un ordinateur, les transistors sont regroupés au sein de ce que l'on appelle des circuits intégrés. Dans un circuit intégré, les transistors sont gravés sur des plaques de silicium, les connexions entre les millions de transistors qui composent un circuit intégré sont, elles aussi, gravées directement dans le silicium.

1.1 Les circuits logiques

Le transistor est l'élément de base des circuits logiques. Un circuit logique permet de réaliser une opération booléenne. Ces opérations booléennes sont directement liées à l'algèbre de Boole. Un circuit logique prend en entrée un ou des signaux électriques(chaque entrée est dans un état "haut" (symbolisé par un "1") ou à un état "bas" (symbolisé par un "0")) et donne en sortie un ou des signaux électriques (chaque sortie est aussi dans un état "haut" ou à un état "bas"). Il existe deux catégories de circuit logique :

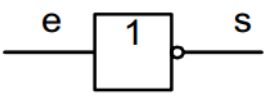
- les circuits combinatoires (les états en sortie dépendent uniquement des états en entrée)
- les circuits séquentiels (les états en sortie dépendent des états en entrée ainsi que du temps et des états antérieurs)

Dans la suite nous nous intéresserons principalement aux circuits combinatoires.

1.1.1 La porte NON

E	S
1	0
0	1

La porte NON

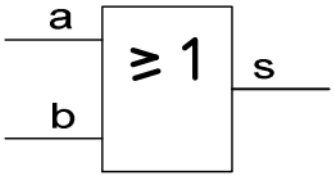


Symbole européen de la porte NON

1.1.2 La porte OU

E_1	E_2	S
0	0	0
0	1	1
1	0	1
1	1	1

La porte OU

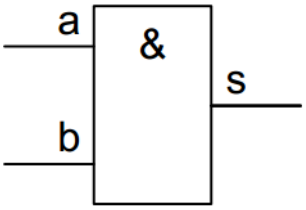


Symbole européen de la porte OU

1.1.3 La porte ET

E_1	E_2	S
0	0	0
0	1	0
1	0	0
1	1	1

La porte ET



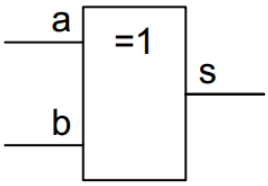
Symbole européen de la porte ET

1.1.4 La porte OU EXCLUSIF

En combinant ces circuits, il est possible d'en obtenir de plus complexes, comme celui-ci :

E_1	E_2	S
0	0	0
0	1	1
1	0	1
1	1	0

La porte OU EXCLUSIF (XOR)



Symbole européen de la porte OU EXCLUSIF (XOR)

1.1.5 Additionneur de bits

On peut aussi créer par exemple un additionneur de bits :

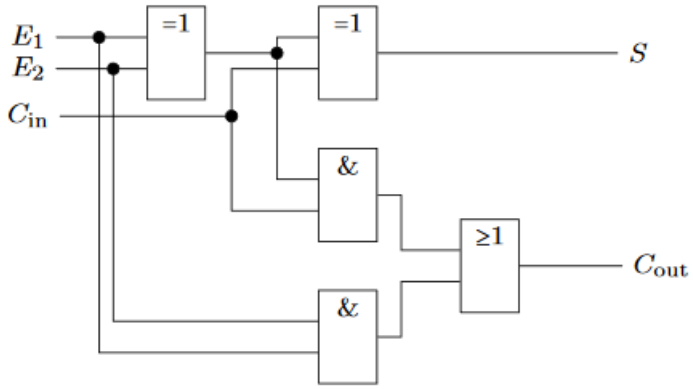


Schéma d'un additionneur de bits

Exercice 5.1

En utilisant le schéma précédent, compléter la table suivante :

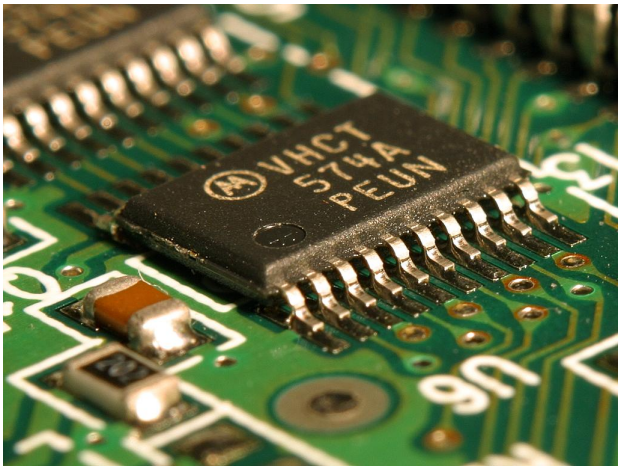
E_1	E_2	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

1.2 Les circuits intégrés

1.2.1 Définition

Définition 5.1

Le circuit intégré (CI), aussi appelé puce électronique, est un composant électronique, basé sur un semi-conducteur, reproduisant une, ou plusieurs, fonction(s) électronique(s) plus ou moins complexe(s), intégrant souvent plusieurs types de composants électroniques de base dans un volume réduit (sur une petite plaque), rendant le circuit facile à mettre en œuvre.



Un circuit intégré

Les circuits intégrés numériques les plus simples sont des portes logiques (et, ou et non), les plus complexes sont les microprocesseurs et les plus denses sont les mémoires.

Le motif de base est le transistor, et ce sont ensuite les interconnexions métalliques entre les transistors qui réalisent la fonction particulière du circuit.

1.2.2 Fabrication

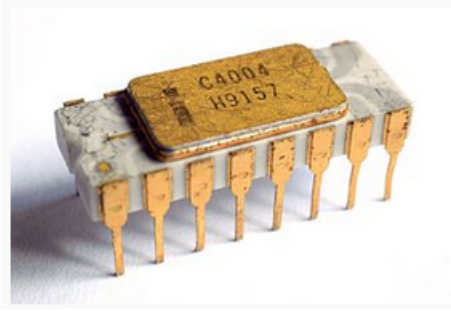
Le motif de base est le transistor, et ce sont ensuite les interconnexions métalliques entre les transistors qui réalisent la fonction particulière du circuit.
La matière première de base habituellement utilisée pour fabriquer les circuits intégrés est le silicium.

1.2.3 Ecologie

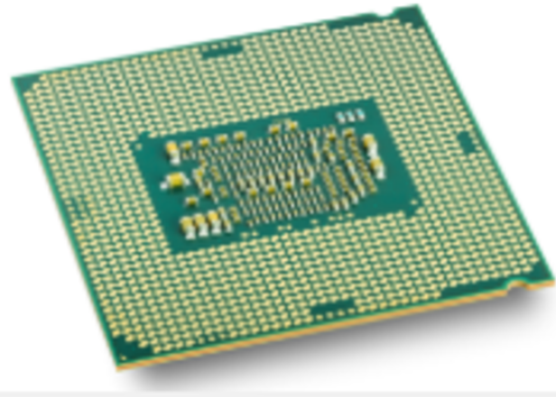
⚠ Il faut une quantité considérable de matière première pour fabriquer une puce électronique : le sac à dos écologique (qui représente la quantité de matières premières nécessaires à la fabrication du produit) d'une puce électronique de 0,09 g est de 20 kg.

2 Le microprocesseur

Le microprocesseur (unité centrale de traitement, UCT, en anglais Central Processing Unit, CPU) est un composant essentiel qui exécute les instructions machine des programmes informatiques.



Intel 4004, 740 KHz, années 70-80



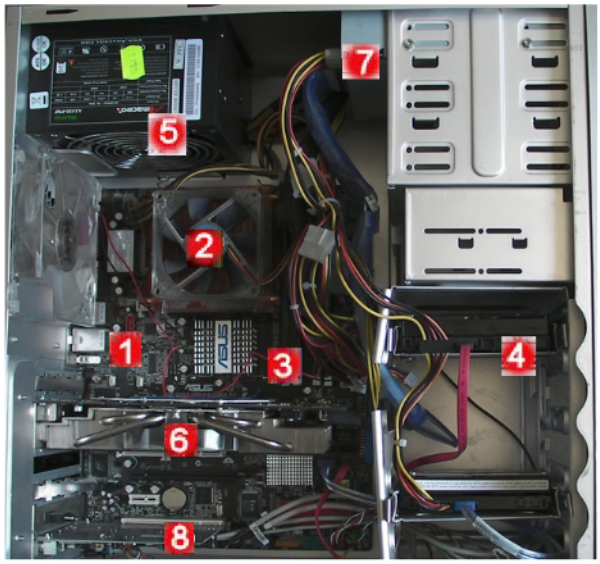
Intel Core I7 6700K

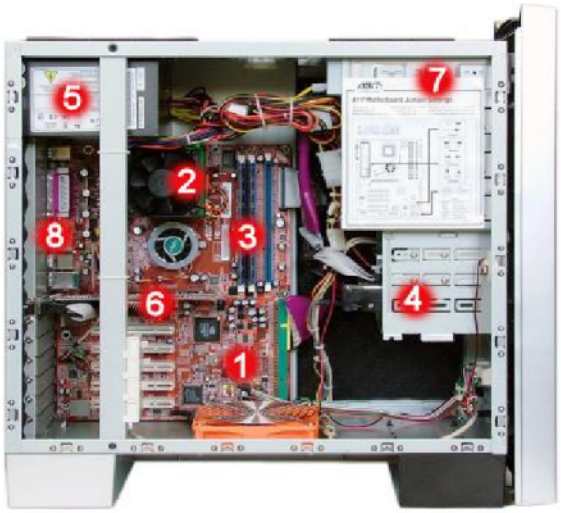
Toutes les activités du microprocesseur sont cadencées par une horloge.
On caractérise le microprocesseur par :

- sa fréquence d'horloge : en MHz ou GHz.
- le nombre d'instructions par seconde qu'il est capable d'exécuter : en MIPS (million d'instructions par seconde).
- la taille des données qu'il est capable de traiter : en bits.

Il est schématiquement composé de 3 parties :

- l'unité arithmétique et logique(UAL) est chargée de l'exécution de tous les calculs que peut réaliser le microprocesseur





A l'intérieur d'une unité centrale...

- les registres permettent de mémoriser de l'information (donnée ou instruction) au sein même du CPU, en très petite quantité
- l'unité de contrôle permet d'exécuter les instructions (les programmes)

☞ L'accumulateur aux côtés de l'UAL sert à conserver le résultat d'un calcul qui peut être immédiatement réutilisé par l'instruction suivante sans passer par le registre. C'est encore un gain de temps !

3 A l'intérieur d'une unité centrale....

Si on ouvre l'unité centrale, on découvre de nombreux composants :

Exercice 5.2

Essayez d'identifier ces principaux composants d'un ordinateur.
Pour vous aider, voici la liste de composants à trouver :

- La carte mère : Le rôle de la carte mère est de centraliser et traiter les données échangées dans un ordinateur à l'aide du processeur, qui est fixé dessus. La carte mère gère donc le disque dur, le clavier et la souris, le réseau, les ports USB. La carte mère est en quelque sortes le support sur lequel tout vient se brancher dans votre ordinateur.
- Le processeur (CPU : Central Processing Unit). Le processeur est le cerveau de l'ordinateur. C'est lui qui organise les échanges de données entre les différents composants (disque dur, mémoire RAM, carte graphique) et qui effectueles opérations arithmétiques et logiques.
- Mémoire RAM (Random Access Memory). La mémoire vive (RAM) est la mémoire informatique dans laquelle un ordinateur place les données lors de leur traitement par le processeur.
- Disque dur (mémoire de masse). Le disque dur est l'un des principaux composants d'un ordinateur. Son rôle est de stocker des données informatiques : c'est donc la mémoire de celui-ci. Le disque dur contient le système d'exploitation, vos programmes installés ainsi que vos données personnelles.
- Alimentation :Le bloc d'alimentation (power supply uniten anglais, souvent abrégé PSU), ou simplement l'alimentation, d'un PC est le matériel informatique l'alimentant.
- Carte graphique. La carte graphique est un composant de l'unité centrale chargé de l'affichage sur l'écran : l'interface graphique du système d'exploitation, les fenêtres, lebureau...

- Lecteur Graveur CD/DVD. Un lecteur graveur de CD/DVD est un lecteur de disque capable de lire , d'écrire et de réécrire des données vers et à partir de CD/DVD. Ils sont de moins enmoins courants car ils sont remplacés par des disque dur externe, des clés USB ou encoredes espaces numériques de stockage (Cloud)
- Carte d'interfaces (réseau, USB, son...) .Les cartes d'interfacessont des périphériques permettant de connecter son ordinateur à un réseau (Ethernet, WIFI ...), aux ports USB, aux enceintes ... etc.

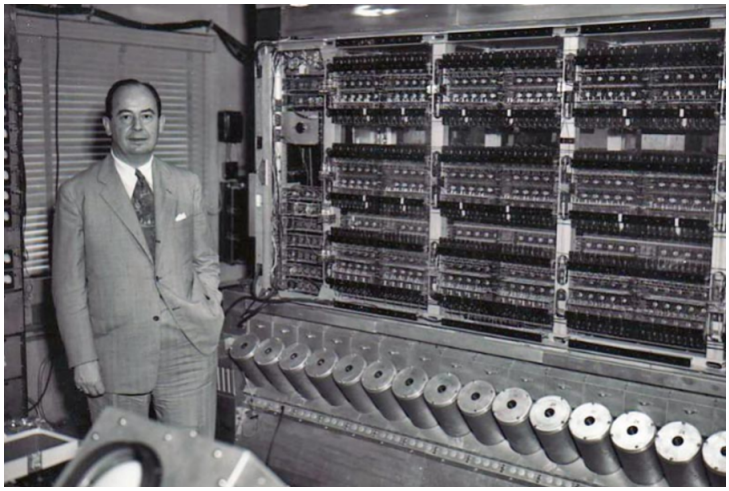
4 Le modèle de Von Newmann

Afin d'exécuter des algorithmes, tous les ordinateurs actuels sont bâtis autour du même modèle architectural théorique : l'***L'architecture de Von Newmann***

Histoire

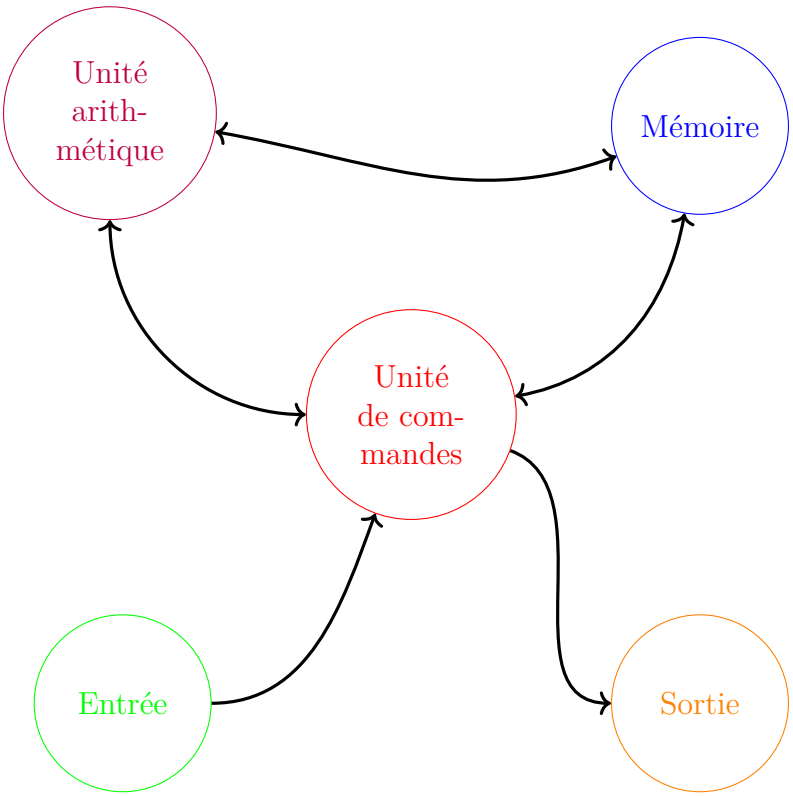
C'est en 1945, dans le cadre de recherche pour l'US ARMY que Von Neumann et son équipe ont conçu une machine révolutionnaire :

Cet ordinateur est capable d'additionner, soustraire, multiplier en binaire.
Sa capacité mémoire est l'équivalent actuel de 5,5ko.
Pourtant, il pèse 7850 kg, occupe une surface 45.5m^2 et nécessite des dizaines de personnes à son chevet en permanence....



Von Newmann et sa machine

Le schéma de référence est celui-ci :



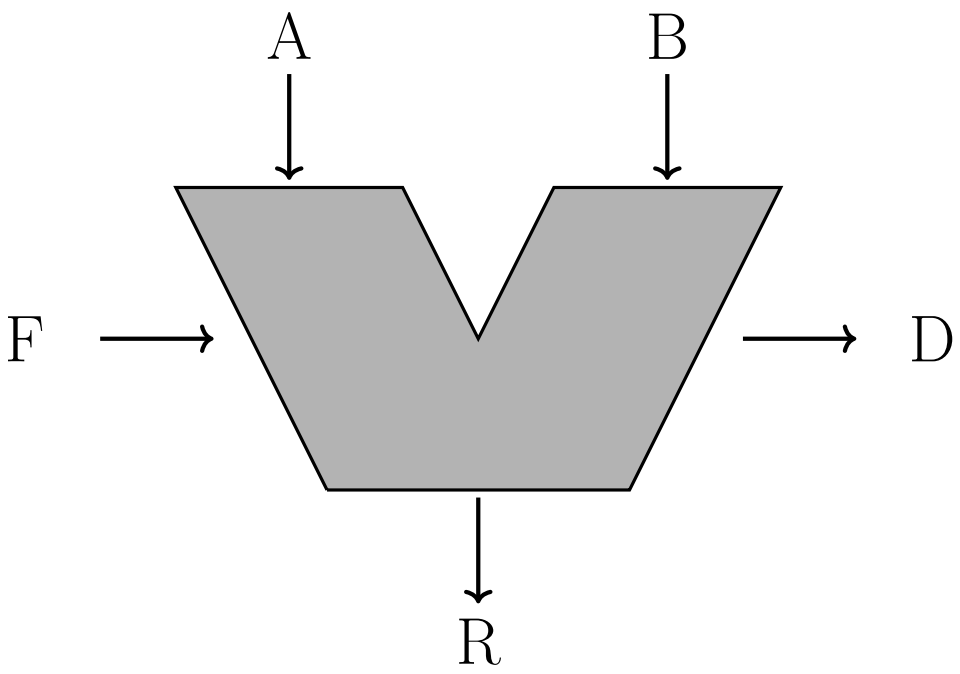
4.1 L'unité arithmétique et logique

l'unité ***arithmétique et logique***, appelée ***UAL*** ou ***ALU*** en anglais, ou ***unité de traitement*** a pour rôle d'effectuer les opérations de base.

☞ l'UAL est situé dans le microprocesseur.

l'ULA est un circuit imprimé qui permet de réaliser les opérations logiques et arithmétiques de bases sur des données numériques : -ET logique, -OU logique, -NON logique, -OU exclusif logique, addition, soustraction, division, multiplication, comparaison...

L'UAL est habituellement représentée comme ceci :



A et B : ce sont des ***opérandes*** (les entrées)

R : Le résultats

F : Fonction binaire (l'opération à effectuer)

D : Un drapeau indiquant un résultat secondaire (signe, erreur, division par zéro...)

Exercice 5.3
Supposons une UAL fonctionnant en 8 bits (comme par exemple sur les anciennes Gameboy !).

Prenant comme opérandes en binaire : A=10010001 et B=10000011.

Supposons que l'on demande à cette UAL d'effectuer l'addition de ces deux nombres : cette information correspond à F.
Quel sera le résultat R ? Quelle information comportera le drapeau D ?
***

4.2 L'unité de contrôle

l'***unité de contrôle*** est chargée de la partie séquentielle des opérations.
☞ L'unité de contrôle se trouve dans la microprocesseur.
L'unité de contrôle utilise une horloge pour cadencer le rythme de travail du microprocesseur. Plus la vitesse de l'horloge est grande, plus le microprocesseur effectue d'instructions en une seconde.

La combinaison de la vitesse de cadencement de l'horloge et l'architecture du microprocesseur détermine la puissance du microprocesseur qui s'exprime en «millions d'instructions par seconde» (MIPS). A leur naissance dans les années 1970, les micro-processeurs effectuaient moins d'un million d'instructions par seconde sur 4 bits, puis en 2007, naissance des microprocesseurs sur 64 bits, ils pouvaient effectuer plus de 10 milliards d'instructions par seconde.

C'est l'unité de contrôle qui donne les ordres à toutes les autres parties de l'ordinateur.

Elle est composé de registres :

CO : Le ***compteur ordinal***, qui contient l'adresse de la prochaine instruction à exécuter

RI : Le ***registre d'instruction*** qui contient l'instruction en cours d'éxecution

AC : L'***accumulateur*** chargé de stocker des opérandes intérmédiaires

CC : Le ***code condition***, utilisé pour les instruction de ruptures conditionnelles (saut).

4.3 La mémoire

Une mémoire peut être représentée comme une armoire de rangement constituée de différents tiroirs. Chaque tiroir représente alors une case mémoire qui peut contenir un seul élément : des données. Le nombre de cases mémoires pouvant être très élevé, il est alors nécessaire de pouvoir les identifier par un numéro. Ce numéro est appelé adresse. Chaque donnée devient alors accessible grâce à son adresse.

La ***mémoire*** contient à la fois les données et le programme qui indiquera à l'unité de contrôle quels sont les calculs à faire sur ces données.
Il existe deux type de mémoires :

- La ***mémoire volatile*** : programme et données en cours de fonctionnement
- La ***mémoire permanente*** : programme et données de base de la machine.

4.3.1 La RAM

La ***mémoire vive*** est appelé ***RAM*** pour ***Random Acces Memory***. Elle contient les programmes et les données en cours d'utilisation dans le processeur.

☛ Le contenu de la RAM disparaît lors de la mise hors tension de l'ordinateur.
☛ La RAM est organisée en cellules qui contiennent chacune une donnée ou une instruction ;
Ces cellules sont repérées par un entier : leur ***adresse mémoire***.
Le temps d'accès à chaque cellule est la même, et on parle donc de mémoire à accès aléatoire(RAM)

Voici quelques ordres de grandeur :

Capacité : jusqu'à 32 Go

vitesse : jusqu'à 2 Go/s

4.3.2 La ROM

La ***mémoire morte***, appelée ***ROM*** pour ***"read only memory"*** contient les programmes et les données de base de la machine.
Ils sont stockés en lecture seule dans l'EEPROM (stockage du BIOS qui est un programme nécessaire au démarrage de la machine)

4.3.3 La mémoire externe ou mémoire de masse

Ce sont les disques durs, les clefs USB, etc...

Elles visent à obtenir une capacité de stockage élevée à faible coût et ont généralement une vitesse inférieure aux autres mémoires.

Voici quelques ordres de grandeur :

capacité : jusqu'à 10 To (HDD)

vitesse : jusqu'à 500 Mo/s (SSD)

4.3.4 Le registre

☞ Intégré au processeur.

Ce type de mémoire est très rapide mais aussi très cher et est donc réservé à une très faible quantité de données.

Voici quelques ordres de grandeur :

capacité : quelques dizaines d'octets

Vitesse : jusqu'à 30 Go/s

4.3.5 La mémoire cache

⚠ La mémoire cache permet de regler les problèmes qui se posent devant la différence de vitesse entre le CPU et la mémoire vive.
La mémoire cache sert à conserver un court instant des informations fréquemment consultées.

☞ On part du principe qu'une instruction ou donnée en cours d'utilisation a plus de probabailité d'être réutilisées que les autres.
Les technologies des mémoires caches visent à accélérer la vitesse des opérations de consultation.

Elles ont une très grande vitesse, et un coût élevé pour une faible capacité de stockage.

Ordres de grandeur :

capacité : quelques ko à quelques Mo

vitesse : jusqu'à 5 Go/s

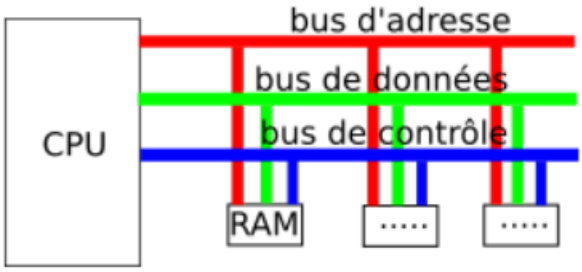
4.4 Les dispositifs d'entrée et de sortie

Les ***dispositifs d'entrée et de sortie*** permettent de communiquer avec la machine.

4.5 Les bus

Il existe trois type de bus :

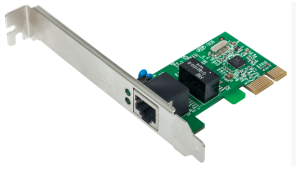
- Le ***bus de données***, ou les données circulent à double sens, notamment entre le processeur et la mémoire.
- Le ***bus d'adresse***, c'est par là que passe le numéro de la case dont le processeur demande le contenu. Il est à sens unique, du processeur vers la mémoire.
- Le ***bus de commande***, c'est là que passent les actions à réaliser, le tout dans une certaine synchronisation



bus d'adresse, bus de données et bus de contrôle

Exercice 5.4
Voici différents composants.
Identifier ces composants et préciser leur rôle suivant le modèle de Von Newmann


1


2


3

5 Les différents langages

5.1 Le langage machine

Programmer en langage machine est extrêmement difficile (très longue suite de 0 et de 1).

5.2 Langage assembleur

Pour pallier cette difficulté, les informaticiens ont remplacé les codes binaires abscons par des symboles mnémoniques (plus facile à retenir qu'une suite de "1" et de "0"),cela donne l'assembleur. Par exemple un "ADD R1,R2,#125" sera équivalent à "11100010100000100001000001111101".

Le processeur est uniquement capable d'interpréter le langage machine, un programme appelé "assembleur" assure donc le passage de "ADD R1,R2, # 125" à "11100010100000100001000001111101". Par extension, on dit que l'on programme en assembleur quand on écrit des programmes avec ces symboles mnémoniques à la place de suite de "0" et de "1"

Voici quelques instructions :

```
LDR R1,78
```

Place la valeur stockée à l'adresse mémoire 78 dans le registre R1 (par souci de simplification, nous continuons à utiliser des adresses mémoire codées en base 10)

```
STR R3,125
```

Place la valeur stockée dans le registre R3 en mémoire vive à l'adresse 125

```
ADD R1,R0,#128
```

Additionne le nombre 128 (une valeur immédiate est identifiée grâce au symbole #) et la valeur stockée dans le registre R0, place le résultat dans le registre R1

```
ADD R0,R1,R2
```

Additionnela valeur stockée dans le registre R1 et la valeur stockée dans le registre R2, place le résultat dans le registre R0

```
SUB R1,R0,#128
```

Soustrait le nombre 128 de la valeur stockée dans le registre R0, place le résultat dans le registre R1

```
 SUB R0,R1,R2
```

Soustrait la valeur stockée dans le registre R2 de la valeur stockée dans le registre R1, place le résultat dans le registre R0

```
MOV R1,#23
```

Place le nombre 23 dans le registre R1

```
MOV R0,R3
```

Place la valeur stockée dans le registre R3 dans le registre R0

```
B 45
```

Nous avons une structure de rupture de séquence, la prochaine instruction à exécuter se situe en mémoire vive à l'adresse 45

```
CMP R0,#23
```

Compare la valeur stockée dans le registre R0 et le nombre 23. Cette instruction CMP doit précéder une instruction de branchement conditionnel BEQ, BNE, BGT, BLT

```
CMP R0,R1
```

Compare la valeur stockée dans le registre R0 et la valeur stockée dans le registre R1.

```
CMP R0,#23
BEQ 78
```

La prochaine instruction à exécuter se situe à l'adresse mémoire 78 si la valeur stockée dans le registre R0 est égale à 23

```
CMP R0,#23
BNE 78
```

La prochaine instruction à exécuter se situe à l'adresse mémoire 78 si la valeur stockée dans le registre R0 n'est pas égale à 23

```
CMP R0,#23
BGT 78
```

La prochaine instruction à exécuter se situe à l'adresse mémoire 78 si la valeur stockée dans le registre R0 est plus grand que 23

```
CMP R0,#23
BLT 78
```

Laprochaine instruction à exécuter se situe à l'adresse mémoire 78 si la valeur stockée dans le registre R0 est plus petit que 23

```
HALT
```

Arrête l'exécutiondu programme

Exercice 5.5
Ecrire en langage assembleur :
Additionne la valeur stockée dans le registre R0 et la valeur stockée dans le registre R1, le résultat est stocké dans le registre R5.

Exercice 5.6
Ecrire en langage assembleur :
la prochaine instruction à exécuter se situe en mémoire vive à l'adresse 478. Si la valeur stockée dans le registre R0 est égale 42 alors la prochaine instruction à exécuter se situe à l'adresse mémoire 85.

Exercice 5.7
Ecrire en langage assembleur le programme suivant :

```
x = 4
y = 8
if x == 10:
    y =9
else:
    x = x+1
z = 6
```

***

Exercice 5.8
Ecrire en langage assembleur le programme suivant :

```
x = 4
y = 8
if x == y:
    y = y + x
else:
    y = y - x
```

Exercice 5.9
Ecrire en langage assembleur le programme suivant :

```
x = 4
y = 5
p = 0
while x > 0:
    p = p + y
    x = x - 1
```

☛ A la fin du programme, on a p = 20, ce qui correspond à $p = 5 + 5 + 5 + 5 = 5 \times 4 = x \times y$

Exercice 5.10
Ecrire en langage assembleur un programme qui permet de calcule 2^5.
☛ Il n'y a pas d'instruction "multiplier", il va falloir se débrouiller avec des additions...
☛ Essayez de n'utiliser qu'un seul registre, R1 par exemple !

5.3 Langage de haut niveau

Même si coder en assembleur est plus rapide qu'en binaire, cela reste une étape fastidieuse : manipuler les données et les placer octet par octet dans la mémoire ou dans les registres, c'est long !

De plus que chaque processeur possède des noms et des numéros d'instructions différents ! Ainsi, un code en assembleur qui fonctionne sur un processeur Pentium ne marchera pas forcément sur un AMD 64 ou sur un processeur ARM. En gros, si vous écrivez un programme sur votre ordinateur, il ne marchera pas sur le mien ! C'est embêtant...

Si le programme ne marche que sur un processeur et non sur un autre, on dit que le programme n'est pas ***portable***.

Dès les débuts de l'informatique moderne on a donc inventé un moyen pour palier à ces deux inconvénients que sont la lenteur à coder et la non portabilité : les langages de haut niveau.

Parmi les langages de haut niveau, il y a une multitude de langages différents. Le C est un exemple, mais peut-être avez vous entendu parler des langages Java, Python, PHP ? Ce sont tous des langages de haut niveau ! Il en va des préférences du programmeur d'utiliser l'un ou l'autre des langages pour faire ses programmes.

Chaque langage possède sa syntaxe et ses objectifs. Par exemple, le langage Perl est très bon pour chercher et analyser dans des fichiers textes, le langage PHP est utilisé pour générer des sites web.

Il n'est pas rare non plus de voir un programme qui soit écrit en différents langages : on utilise le langage le plus convenable pour chaque tâche.

☞ Les instructions d'un programme écrit dans un langage de haut niveau sont traduite pour être comprise par la machine. Avec le module **dis**, nous pouvons avoir une idée des instructions passées à la machine lorsque l'on écrit un code Python :

```
import dis
dis.dis('x=1;x=x+2')
```

```
  0 LOAD_CONST               0 (1)
  2 STORE_NAME               0 (x)
  4 LOAD_NAME                0 (x)
  6 LOAD_CONST               1 (2)
  8 BINARY_ADD
 10 STORE_NAME               0 (x)
 12 LOAD_CONST               2 (None)
 14 RETURN_VALUE
```

Affichage du code machine désassemblée

- Le nombre 1 est copié dans le registre
- Le contenu du registe est copié en mémoire à l'adresse x
- la valeur de x est copié dans le registre
- Le nombre 2 est copié dans le registre
- On effectue l'addition binaire
- Le résultat est copié dans la mémoire à la case adressé par x
- La valeur None est copiée dans le registre

- Puis elle est renvoyée.

Exercice 5.11
utilisez et intérpréter le résulats du module dis avec le code Python suivant :

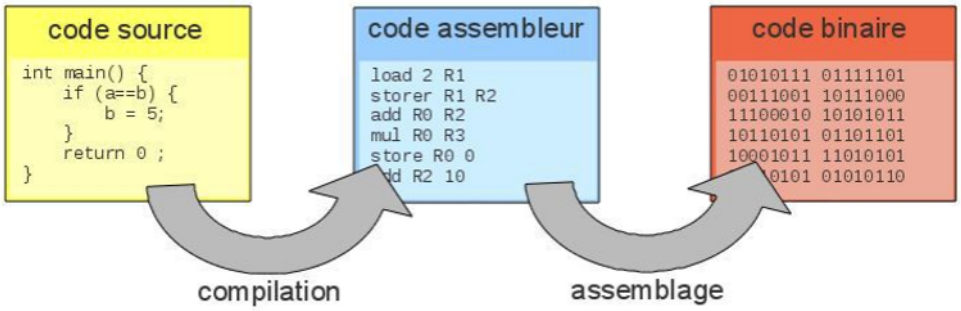
```
x = 3
if x < 0:
    y = -x
else:
    y = x
```

5.4 La compilation et l'assemblage

Le problème des langages de haut niveau c'est qu'ils ne sont absolument pas compréhensibles par le processeur ! Avant de pouvoir être utilisé, le code doit être traduit en langage assembleur.

Cette traduction, c'est ce qu'on appelle la compilation.

Une fois en langage assembleur, il faut ensuite l'assembler en langage binaire :



La compilation et l'assemblage

6 Simulateur

C'est ici !

Exercice 5.12

Prendre en main le simulateur avec :

```
MOV R0, #4
STR R0,30
MOV R0, #8
STR R0,75
LDR R0,30
LDR R1,75
ADD R2,R0,R1
STR R2,40
HALT
```

Repérer dans le simulateur, les différents éléments spécifiques de l'architecture de Von Neumann.

- Repérer notamment dans le simulateur les éléments suivants :
  - Les registres :

        - PC
        - CIR (instruction en cours)
    - Les entrées
    - Les sorties
    - l'UAL
    - L'unité de contrôle
    - La mémoire
- Donner le code hexadécimal qui correspond à l'instruction "LDR R0,30"

Exercice 5.13

On considère la suite d'instruction :

```
INP R0,2
INP R1,2
CMP R1,R0
BGT VRAI
OUT R0,4
B DONE
VRAI:
OUT R1,4
DONE:
HALT
```

1. Devinez ce que fait ce programme
2. Tester le programme en langage assembleur en utilisant le simulateur avec comme saisie 4 puis 8
3. Tester le programme en langage assembleur en utilisant le simulateur avec comme saisie 7 puis 2
4. Que fait ce programme en langage assembleur ?

Exercice 5.14

Ecrire en langage assembleur ce programme :

```
x=1
while x<6:
   x=x+2
print(x)
```