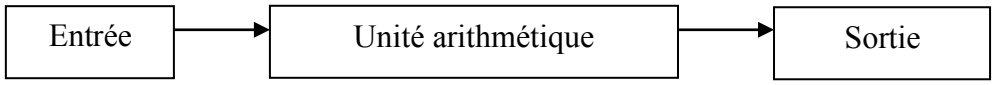
Architecture des ordinateurs

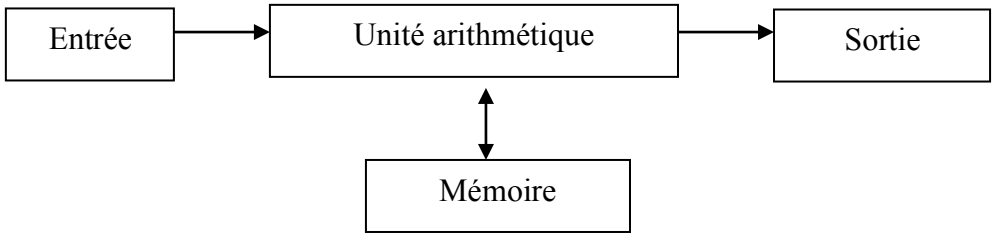
1. Genèse et histoire.

Entre le XIe et le XVIIe siècle, l'ordinateur désigne celui qui est chargé de « régler les affaires publiques ».
Le sens actuel a été proposé en 1955 en réponse à une demande du service publicitaire d'IBM France.
La définition a beaucoup évolué en 60 ans. Celle de Wikipedia est aboutie : « machine électronique qui fonctionne par la lecture séquentielle d'un ensemble d'instructions qui lui font exécuter des opérations logiques et arithmétiques sur des chiffres binaires ». Nous avons déjà vu l'aspect « lecture séquentielle d'instructions », en écrivant des programmes, et l'aspect binaire dans le cours sur le codage. Nous allons dans ce chapitre voir comment l'architecture permet d'effectuer les « opérations arithmétiques et logiques ».

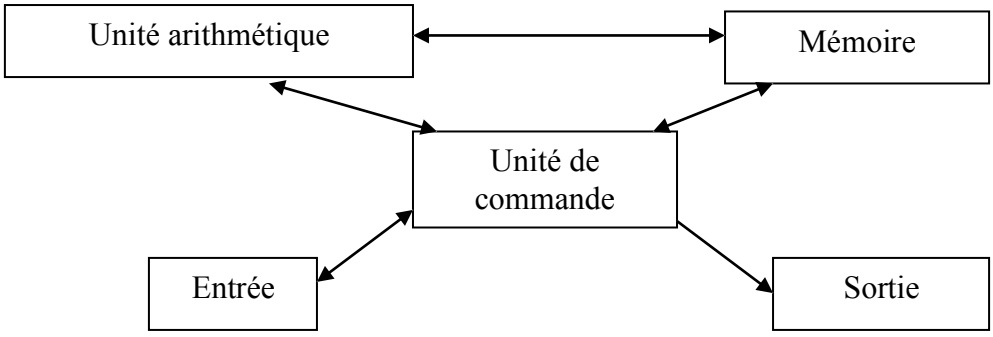
Un peu d'histoire.
Les premières machines à calculer datent du XVIIe siècle. La pascaline est la plus connue, c'est en fait la seule dont on soit sûr de l'existence puisqu'il en existe toujours une dizaine. Elle ne pouvait faire que des additions, mais Blaise Pascal avait trouvé la méthode du complément à 10 (sur le principe du complément à 2 vu dans le dernier chapitre) pour faire des soustractions.
On a le schéma suivant :



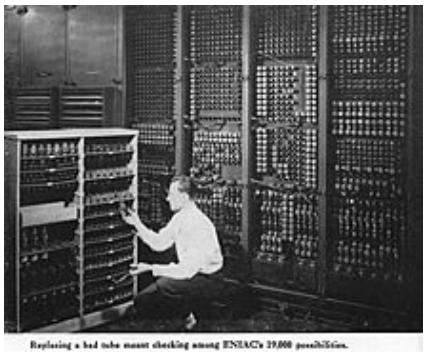
Leibniz, en 1673, rajoute la mémorisation des résultats intermédiaires : sa machine pouvait faire ainsi des multiplications et des divisions.
D'où le schéma :



Leibniz évoque de plus la représentation binaire grâce… à la Chine. En effet, il connaît le « Yi-King », traité de divination, dont les tirages divinatoires se font en fait en binaire. Il communique sur le binaire à l'Académie des Sciences en 1703.
150 ans plus tard, en 1847, Georges Boole fonde l'algèbre de Boole, que l'on a vue dans le chapitre du même nom. Cette algèbre est basée sur les valeurs de vérité Vrai ou Faux, et permet notamment de fabriquer des circuits électroniques compacts.
À peine plus tôt, en 1834, Charles Babbage à l'idée d'incorporer dans la machine à calculer des cartes perforées, pour donner la suite d'instructions à exécuter. Il ne put jamais réaliser sa machine la plus performante, faute de fonds suffisants. Ada Lovelace, qui travailla avec Babbage, écrivit le premier véritable programme informatique de l'histoire. Son nom a été donné à un langage de programmation, et son portrait figure sur les hologrammes d'authentification des produits Microsoft.
Depuis cette époque, le schéma qui va servir de référence est le schéma ci-contre.



C'est Claude Shannon, spécialiste de la théorie de l'information, qui expose en 1938 comment utiliser l'algèbre de Boole pour construire des machines à calculer basées sur des commutateurs et des relais.
Le premier calculateur utilisant l'électricité et le binaire fut le Z1, conçu en 1936 par l'ingénieur allemand Konrad Zuse, qui crée aussi le premier langage de haut niveau. Il a été construit dans la salle à manger des parents Zuse et utilisait des vieilles pellicules de cinéma 35mm en guise de bandes… Le langage de Zuse fut oublié, faute de machine capable de le supporter, jusqu'en 1972. Il n'a été implémenté qu'en 2000, à titre historique. Quant à l'ordinateur, dont la troisième version, le Z3, fut détruite par les alliés à la fin de la guerre, il semblerait qu'il était bien plus avancé que ses concurrents directs, dont l'ENIAC (cf. ci-dessous). Le Z3 était composé de 2600 relais, d'un lecteur de bandes et d'une console pour l'opérateur, sa mémoire pouvait contenir 64 nombres de 22 chiffres exprimés en virgule flottante. Il réalisait une multiplication en trois à cinq secondes. Il était bien plus performant que toute machine équivalente de l'époque, voire que des machines plus connues comme l'ENIAC ci-dessous.


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

Le premier ordinateur « moderne », l'ENIAC, est mis en service en 1946. Il ne travaillait pas en binaire, mais en décimal.Einstein et Gödel (vous connaissez le premier, le deuxième est un génie de la logique) estimaient que cette coûteuse réalisation n'apporterait aucune contribution à la science… Cet ordinateur de première génération (1946-1956) fonctionnait avec des lampes à vide (toujours utilisées dans les amplificateurs audio de grande qualité), de durée de vie très limitée. La durée moyenne entre deux pannes était de quelques heures. Une fausse origine du terme « bug » est l'une des causes de pannes : un insecte (bug) qui, se posant sur un tube, brûle et cause la rupture du tube. En fait ce terme est bien plus ancien, il désigne depuis le XIXe siècle les disfonctionnements dans des éléments mécaniques.
En 1947 apparaît le transistor, petit, fiable, et peu coûteux. Il remplace les lampes à vide dans les ordinateurs de deuxième génération (1956-1963).
Enfin les circuits intégrés formés de quelques dizaines à plusieurs centaines de millions de transistors voient le jour en 1958. Ils sont au cœur des ordinateurs de troisième génération (1963-1971).

Nous sommes dans la quatrième génération d'ordinateurs, celle qui utilise des microprocesseurs, qui ont permis la naissance des micro-ordinateurs. Selon certaines sources, le premier micro-ordinateur, serait français : le Micral, crée en 1972. Les américains disent bien sûr que le premier micro-ordinateur a été fabriqué aux USA, mais on sait bien qu'ils se vantent ☺.
Selon la loi de Moore, la densité des composants électroniques sur une puce suit une croissance exponentielle (suite géométrique), en doublant tous les deux ans. Le premier microprocesseur, un Intel était une unité de calcul sur 4 bits, tournant à 108 KHz et intégrant 2300 transistors. Un Intel actuel (un des i7) comporte 3,2 milliards de transistors, tourne à 3,7 GHz, et calcule sur 64 bits. La loi de Moore s'est vérifiée jusqu'en 2010, mais la miniaturisation devient telle que des limites physiques infranchissables se dressent. La fréquence des microprocesseurs correspond au nombre d'opérations élémentaires effectuées par seconde (108 KHz = 108 000 opérations par seconde, 3,7 GHz = 3 700 000 000 opérations par seconde).
Les recherches se portent actuellement sur le long terme, avec des ordinateurs quantiques, où un bit peut superposer en même temps les valeurs 0 et 1…

2. <u>Les dates et les concepts clés à retenir.</u>
Du moins que vous êtes censés retenir. Essayez au minimum d'avoir une vision de l'évolution par décennie.
Algorithmes : présent dès l'antiquité, voir par exemple l'algorithme d'Ahmès dans la 1er chapitre.

-3000 : le binaire en Chine
XVIIe siècle : premières machines à calculer.
XIXe siècle : formalisation et création des sciences de l'information, notamment
1843 : théorie de la programmation par Ada Lovelace
1854 : logique binaire de Georges Boole
1936 : Alan Turing : concept de machine universelle. Travail s'étendant sur de longues années
1936 : création du Z1, premier calculateur binaire fonctionnant à l'électricité
1943 : invention du transistor, se généralise plus tard.
1946 : ENIAC, premier ordinateur moderne, à tubes
1947 : langage Assembleur
1968 : premier microprocesseur (Intel 4004, 2300 transistors, 4 bits, puissance identique à l'ENIAC… mais tient dans le creux de la main)
1969 : ARPANET (Advanced Research Projects Agency Network) ancêtre d'internet, relie deux machines, puis 4 sites.
1969 : Unix, système d'exploitation
1972 : premier micro-ordinateur, le Micral (français, cocorico !)
1972 : langage C (successeur du… B)
1972 : le jeu Pong
1974 : premier super ordinateur le Cray I
1976 : premier ordinateur personnel (Apple I, 50 exemplaires vendus). L'année d'après, en 1977, le Apple II aura un grand succès.
1978 : protocole TCP/IP pour la communication entre réseaux hétérogènes
1981 : premier PC (personal computer) d'IBM, avec le système d'exploitation Microsoft MS-DOS
1983 : Apparition des virus. Le premier virus pour PC date de 1986, il n'était pas malicieux puisque les auteurs, deux frères pakistanais, avaient mis leur noms et numéro de téléphone dans le code.
1983 : réelle naissance d'Internet avec la séparation entre réseau militaire et réseau civil.
1985 : GUI (interface graphique utilisateur) Windows (immense succès, copié sur l'interface des Apple Macintosh sortis l'année précédente)
1985 : le jeu Tétris
1990 : langage HTML et mise au point du WWW (world wide web) par Tim Berners-Lee
1991 : Système d'exploitationLinux
1992 : premier smartphone (IBM Simon)
1994 : le jeu Doom
1995 : Internet décolle
1998 : notion de logiciel libre
2003 : premier microprocesseur multi-cœur
2006 : la téléphonie mobile décolle
2007 : décollage des smartphones avec l'iPhone
2019 : suprématie quantique (contesté). Première fois qu'un ordinateur quantique exécute un programme qu'un ordinateur classique ne sait pas faire.
Loi de Moore : le nombre de composants sur une puce double tous les deux ans. En 1971, le Intel 4004 comporte 2300 transistors et a une fréquence de fonctionnement de 0,108 MHz. En 2019, le Intel Core i9-7980xe comporte approximativement 7 000 000 000 de transistors (estimation car Intel ne communique plus sur ces données) et a une fréquence de 4200 MHz. Il semble que l'on atteint les limites possibles de la miniaturisation des transistors, la fin de la loi de Moore a été annoncée en 2016 par une association d'experts de l'industrie électronique.

3. Structure générale des ordinateurs.
Un ordinateur est structuré en trois couches :
   - La plus « visible » est celle des logiciels : traitement de texte, navigateur internet, jeux…
   - Transparente et pourtant indispensable pour simplifier l'accès à la machine : le système d'exploitation. On citera Windows, Unix, Linux, Mac OsX (basé sur Linux).

- La partie matérielle : le processeur, la mémoire interne ou externe, les périphériques (souris, écran, imprimante…)

Les deux premières couches sont du « software », ce qui signifie matériel mou, tandis que la dernière est du « hardware », soit matériel dur.

4. L'architecture de la Machine de Von Neumann

L'architecture générale des ordinateurs a été fournie par John Von Neumann en 1945-1946. Cette architecture est identique du nano-processeur qui équipe la machine à laver jusqu'au superordinateur.

On retiendra les principes suivants :

- La fonction de calcul est réalisée par *l'UAL* (*unité arithmétique et logique*) ;
- La fonction d'enregistrement est réalisée par la *mémoire* ;
- Le déroulement séquentiel est réalisé par *l'UC* (*unité de commande*) ;
- Les *registres* stockent temporairement des données ;
- Les dispositifs d'entrée-sortie permettent la communication avec le monde extérieur, qu'il soit humain ou machine.
- La mémoire utilisée pour stocker les données fonctionne de la même manière que la mémoire utilisée pour faire tourner/stocker les programmes.

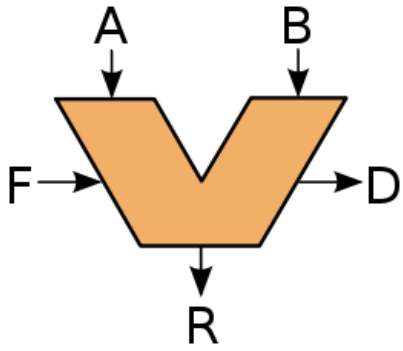
L'UAL et l'unité de commande sont regroupés dans le processeur, qui communique avec la mémoire par un bus. Historiquement, ces deux parties ont pu être séparées.

Ces composants constituent l'unité centrale, située sur la carte mère de votre pc. L'ensemble est rythmé par une horloge, c'est la fréquence du microprocesseur. Exemple : 3,4 gHz, soit 3 milliards 400 000 000 cycles d'horloge par seconde pour un Core i7-2600 K.

a. UAL

L'unité arithmétique et logique est le cœur de l'ordinateur.

On la représente habituellement par le schéma ci-contre :



- A et B sont les opérandes (les entrées)
- R est le résultat
- F est une fonction binaire (l'opération à effectuer)
- D est un drapeau indiquant un résultat secondaire de la fonction (signe, erreur, division par zéro, « supérieur »,…)

Les opérations que peuvent réaliser une UAL de base sont :

- Les opérations arithmétiques usuelles (+, - , × , ÷) ;
- Les opérations logiques (ET, OU, NON) ;
- Les opérations de comparaison (> , < , =)

Certaines UAL sont spécialisées (calcul sur les nombres en virgule flottante, cartes graphiques, cartes sons, …). Un même processeur peut comporter plusieurs UAL. Dans certaines architectures, les UAL sont spécialisées et ne fonctionnent pas simultanément, dans d'autres les UAL fonctionnent en parallèle (architectures multi-cœur). Les architectures multi-cœur permettent d'augmenter la puissance de calcul sans se heurter aux problèmes de miniaturisation (cf. limites sur la loi de Moore).

Toues les fonctions réalisées par les UAL le sont grâce à des circuits électroniques effectuant des fonctions binaires, nous en verrons quelques exemples dans le chapitre sur le calcul booléen.

b. Mémoire.

Les mémoires sont de deux types : ROM ou RAM, ce qui signifie Read Only Memory pour la première, et Read And write Memory pour la deuxième. On dit mémoire morte (ROM) et mémoire vive (RAM) en français. La ROM contient le BIOS, ce qui permet de faire démarrer l'ordinateur et de mettre en route le système d'exploitation. Mais d'autres appareils, comme une télécommande ou un lave-vaisselle, ne contiennent que de la ROM.
Plus où s'éloigne du processeur, plus les mémoires seront lentes et moins elles seront chères. Comme on l'a vu ci-dessus, il y a au plus près les registres, puis la mémoire cache (SRAM pour Static RAM) qui contient les mots de données les plus utilisés, ensuite la RAM usuelle (les barrettes de mémoire que l'on rajoute à l'intérieur de l'ordinateur), et enfin les mémoires externes, de masse : disques durs, cartes flash des appareils photo (qui ne sont réinscriptibles que 100 000 fois au plus, ce sont un type particulier de ROM) , clefs USB, etc…

Les **registres** sont des unités de mémoire situées au plus près de l'UAL et de l'UC. Ce sont des circuits logiques, d'accès extrêmement rapide, très coûteux et donc peu nombreux. Ils servent soit au calcul, soit à des tâches spécialisées. Pour ces dernières, on trouve :

- Le compteur ordinal (CO), nommé aussi pointeur d'instruction ou compteur de programme. Il indique où se trouve la prochaine instruction à devoir être exécutée.
- Le registre d'état du processeur (PSW (processor status word) qui contient des indications sur l'effet de la dernière instruction. Il est interprété bit à bit, sous forme de drapeaux. Exemple : le bit n° *xx* pourra indiquer que le résultat d'une opération est 0, le bit n° *xy* pourra indiquer le signe +/- = 0/1 du résultat, le bit n° *xz* pourra indiquer un dépassement de capacité (overflow), ce qui se produit lorsque la somme de deux nombres positifs donne un nombre négatif (cf. cours codage)…
- Le registre d'instruction RI contient l'instruction en cours.

La **mémoire cache** est très proche du CPU, un peu moins que les registres cependant. Elle sert à stocker les mots (données et/ou instructions) les plus utilisés par l'ordinateur. Cela permet au processeur de les chercher très rapidement. Vu la vitesse du processeur, si cette mémoire n'était pas là, le processeur ne ferait qu'attendre les données provenant de la **mémoire centrale**, c'est-à-dire des barrettes de mémoires qui contiennent les programmes en cours d'exécution, y compris le système.
La **mémoire de masse**, disques durs internes ou externes, clés USB, contient toutes les données et programmes « dormants », qui ne sont pas en cours d'exécution, ou bien qui sont trop lourds pour être chargés en entier en mémoire centrale.

Pour la suite du cours, on considère que la mémoire utilisée est la RAM.
Dans sa forme de base, la mémoire est un ensemble de cellules organisées en tableau.
Chaque cellule possède :

- Une adresse, qui est l'indice de la cellule dans le tableau : X
- Un contenu, qui est la valeur de l'élément dans le tableau : M[X]

Adresse		Contenu (exemple)
Décimal	Binaire	
0	0	00101100
1	1	11001011
2	10	…
3	11	…

Les cellules servent à la fois à stocker les données (ou variables), et le programme.
Remarque : un virus, c'est un programme qui se camoufle en données pour pouvoir s'exécuter en douce…

c. <u>Le langage machine.</u>
Les langages de programmation les plus proches du codage de « bas niveau », en 0 et 1, sont appelés « Assembleurs ». Il y a presque autant d'Assembleurs que de microprocesseurs. Un programme écrit en Assembleur ne tournera que sur une seule machine, contrairement à un programme écrit dans un langage de haut niveau.

En Assembleur, dans sa forme la plus simple, une instruction est stockée dans une cellule de mémoire désignée par une adresse.
Elle est constituée de 2 parties :

- Un code opération (CO) qui indique quelle opération doit être effectuée
- Une adresse (AD) désignant l'opérande de cette opération

Ainsi, par exemple, un algorithme de calcul d'une addition décrit par l'expression arithmétique $z = x + y$ donnera naissance a` une suite de 3 instructions telles que :

- LDA X Chargement (load) du contenu de la cellule mémoire X dans une cellule appelée accumulateur (c'est un des *registres*). On note ce registre ACC ici.
- ADD Y Addition du contenu de la cellule mémoire Y avec le contenu de l'accumulateur
- STO Z Stockage (pour store) du contenu de l'accumulateur dans la cellule mémoire Z

LDA, ADD, STO étant des codes opération ; X, Y, Z les adresses des opérandes.

Toute expression arithmétique ou logique, quelle que soit sa complexité, va être décomposée en une suite d'instructions de cette nature. L'unité de commande va alors se charger d'enchaîner séquentiellement la suite de ces instructions calculant cette expression.

Cependant il est bien évident que ce seul mécanisme est insuffisant et ne peut réaliser les instructions itératives ou conditionnelles des langages de haut niveau.
Pour cela, il est nécessaire d'introduire des instructions spécifiques appelées des « ruptures de séquence ». Celles si peuvent être simples ou conditionnelles.
Une instruction de rupture de séquence simple est de la forme :

- BR X Saut Inconditionnel (jump/branch) indique que le programme se poursuit à partir de l'adresse X

Une instruction de rupture de séquence conditionnelle est de la forme :

- BRc X Saut Si *c* indique que le programme se poursuit à partir de l'adresse X si la condition *c* est réalisée.

Si la mémoire d'adresse X contient 10, MUN est l'adresse d'une cellule contenant la valeur -1, et « P » derrière BR signifie le résultat est strictement positif, que va faire le programme suivant ?

```
        LDA   X
ADR     ADD   MUN
        BRP   ADR
        STO   X
```

Remarque : il est particulièrement simple de tester si un entier est positif. Comme nous l'avons vu dans le chapitre précédent sur le codage, son premier bit vaut 0. Les opérations réalisables par les premières UAL étaient des opérations particulièrement simples. Rappelez-vous comment on fait pour multiplier par 2/diviser par 2 par exemple…

d. <u>L'unité de commande (approfondissement).</u>

Si l'UAL est le cœur de l'ordinateur, l'unité de commande en est le cerveau : comme son nom l'indique, c'est elle qui donne les ordres à toutes les autres parties de l'ordinateur.
Elle est principalement de quatre registres :

- CO : le compteur ordinal, qui contient l'adresse de la prochaine instruction à executer ;
- RI : le registre d'instruction qui contient l'instruction en cours d'execution ;
- ACC : l'accumulateur chargé de stocker des opérandes intermédiaires ;
- CC : le code condition, utilisé pour les instructions de rupture conditionnelle (saut).

Son algorithme de fonctionnement est le suivant :

Répéter (indéfiniment)

L'instruction désignée par le compteur ordinal est chargée dans le registre d'instructions, soit :

RI ← M[CO]

Le compteur ordinal est incrémenté pour désigner l'instruction suivante, soit :

CO ← CO + 1

L'opération chargée dans RI se décompose en deux parties : la fonction proprement dite désignée par OP, et l'opérande sur lequel cette fonction est exécutée, désignée par AD. L'opération désignée par OP est exécutée avec l'opérande AD, soit :

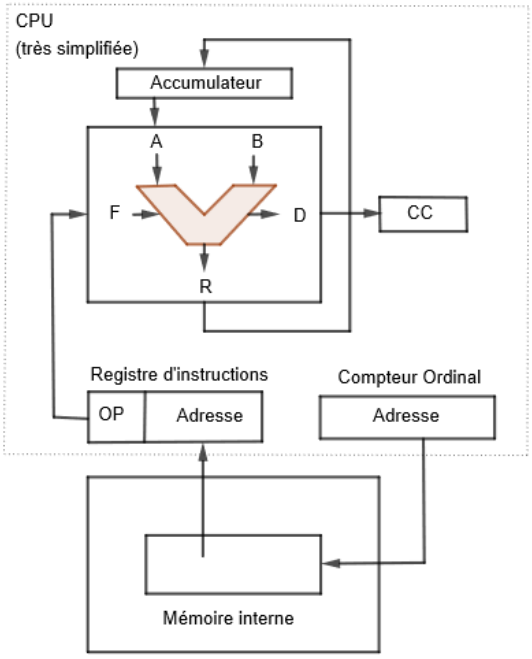
RI.OP (RI.AD)

En cas de rupture conditionnelle, le contenu du compteur ordinal CO est changé par l'adresse du saut.

On peut avoir aussi ECOUTER les périphériques *dans le fonctionnement*

Fin répéter

Le schéma de fonctionnement en est le suivant, vous pouvez constater que l'entrée A de l'UAL correspond au registre accumulateur ACC.



5. Périphériques

On distingue :

- Les organes d'*entrée/sortie* permettant l'échange d'informations entre le système et le milieu extérieur (son environnement). Les principaux organes d'entrée sont le clavier et la souris ; les principaux organes de sortie sont l'écran et l'imprimante. Les périphériques d'entrée peuvent être lus régulièrement, ou bien communiquer via une *interruption* avec l'UC.
- Les *mémoires auxiliaires* qui sont des supports de stockage de grande capacité.

  Les principaux supports de stockage sont les disquettes, les disques durs et les CDROM.

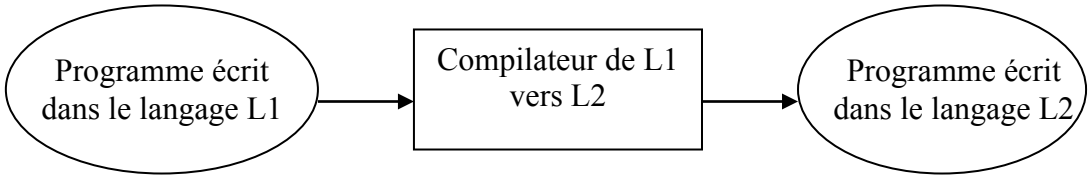
  L'information est stockée dans ces différents supports sous forme de fichiers.

6. Compilation (pour la culture générale).

Programmer en Assembleur n'est ni très simple ni très agréable. Aussi, de nouveaux langages de programmation, dit de haut niveau, ont peu à peu été définis pour faciliter la tâche des programmeurs. En inventer un nouveau, cela signifie :

- Définir un langage permettant d'exprimer n'importe quel algorithme ;
- Définir une correspondance, une méthode de traduction (exprimable par un algorithme !) entre ce nouveau langage et un langage d'assemblage.

La difficulté vient plutôt de la seconde condition que de la première. En effet, toutes les langues humaines sont en principe capables d'exprimer toute méthode de calcul, tout algorithme, de façon plus ou moins informelle. Mais les ordinateurs ne comprennent rien aux langues humaines. De plus, les langues humaines sont beaucoup trop imprécises et ambiguës pour être traduisibles directement en Assembleur. Le problème est donc de définir une langue qui sera traduisible en Assembleur, de telle sorte que cette traduction elle-même puisse s'exprimer par un algorithme.
Ce rôle de traduction est joué par ce qu'on appelle les *compilateurs*. Un compilateur est un logiciel capable de transformer un programme écrit dans un langage de programmation donné L1, appelé code *source*, en un programme réalisant le même traitement mais écrit dans un autre langage L2, appelé code *compilé*, en général un Assembleur, comme le montre le schéma de la figure suivante :



Les compilateurs étant des programmes, ils sont eux-mêmes écrits dans un certain langage de programmation (et s'ils ne sont pas écrits en Assembleur, ils doivent eux-mêmes être compilés...).
La manière la plus simple de compiler un programme est de traduire les instructions une par une. Cependant, les compilateurs optimisent la traduction afin de rendre les programmes plus efficaces. Par exemple, conserver une donnée très souvent utilisée dans un registre, plutôt que d'aller la rechercher en mémoire systématiquement, ce qui peut prendre cent fois plus de temps.

Exercices Architecture

Les exercices qui suivent s'appuient sur une structure ultra simplifiée d'un ordinateur totalement imaginaire, le Little Man Computer (LMC), dont voici les caractéristiques :

- On le trouve à l'adresse https://peterhigginson.co.uk/lmc/
- Ce qui permet de tester les programmes, et de voir le fonctionnement simulé du microprocesseur (de manière très simplifiée)
- La mémoire centrale est constituée d'un tableau de 99 cellules
- Le jeu d'instructions est constitué en partie des instructions suivantes (il y en a d'autres qui ne nous serviront pas).

Instruction	Nom	Effet (version simple)	Effet (version moins simple)	Code opération
ADD	Additionner	Ajoute à l'accumulateur (entrée A de l'UAL) le contenu situé à l'adresse xx	ACC ← ACC+M [xx]	1xx
SUB	Soustraire	Soustrait à l'accumulateur (entrée A de l'UAL) le contenu situé à l'adresse xx	ACC ← ACC-M [xx]	2xx
STO	Stocker	Stocke à l'adresse xx le contenu de l'accumulateur (le résultat du calcul précédent)	M [xx] ← ACC	3xx
LDA	Charger	Charge dans l'accumulateur (l'entrée A de l'UAL) le	ACC ← M [xx]	5xx

		contenu situé à l'adresse xx		
BRA	Saut	Saut à l'adresse xx. On peut donner un nom à l'adresse pour plus de commodité	CO ← xx	6xx
BRZ	Saut si = 0	Idem, si résultat du calcul à la ligne d'avant (stocké dans l'accumulateur) est égal à 0	si CC = 0 alors CO ← xx	7xx
BRP	Saut si > 0	Idem, si résultat du calcul dans l'accumulateur est supérieur à 0	si CC > 0 alors CO ← xx	8xx
INP	Entrée de données	Lit le contenu du champ INPUT et le charge dans l'accumulateur	ACC ← INPUT	901
OUT	Sortie de données	Lit le contenu de l'accumulateur et l'écrit dans le champ OUTPUT	OUTPUT ← ACC	
HLT	Fin de programme			000
DAT	Data	Traite le contenu d'une adresse comme une donnée. S'utilise en donnant un nom à l'adresse.	M[xx] ← *valeur*	Pas de code

- Exemples : copier les instructions dans le LMC dans la partie « assembly langage », puis cliquer sur submit », et enfin sur RUN pour faire tourner le programme. Les chevrons <<< et >>> permettent de ralentir/d'accélérer le fonctionnement. Compléter ci-dessous pour expliquer ce que fait le programme (on peut écrire un algorithme pour le 3ème programme, dans la colonne de droite). Pour relancer un programme, cliquer sur « assemble into RAM » avant, afin de nettoyer la mémoire.

Programme 1	Programme 2	Programme 3	Algorithme programme 3
INP STA 20 OUT HLT	INP STA A ADD UN OUT HLT A DAT UN DAT 001	INP STA 25 INP STA 26 SUB 25 BRP ADR OUT HLT ADR LDA 25 OUT HLT	
Explication des programmes 1 et 2			

Ex 1 : Écrire un programme qui ajoute deux nombres rentés au clavier.
Ex 2 : Écrire un programme vérifiant les conditions suivantes :

- Rentrer deux nombres aux adresses ADR1 et ADR2

- Si le nombre dans ADR1 est plus grand ou égal à ADR2, remplacer le contenu de la case ADR2 par le nombre dans ADR1.

Ex 3 : Même exercice que le précédent, mais maintenant, on écrit dans la case mémoire ADR3 :

- 0 si le contenu de la case ADR1 est strictement inférieur à celui de la case mémoire ADR2
- 1 sinon

Ex4 :

1. Écrire un programme qui fait un minuteur qui débute à un nombre donné est atteint, et s'arrête à 0, en diminuant un par un. Les nombres seront affichés en sortie au fur et à mesure.
2. Écrire un programme qui fait un minuteur pour partir de 0 et s'arrêter lorsque qu'un nombre donné est atteint, en augmentant un par un. Les nombres seront affichés en sortie au fur et à mesure.
3. Écrire un programme qui fait un minuteur pour partir de 0 et s'arrêter lorsque qu'un nombre donné est atteint, en augmentant un par un. Les nombres seront rangés dans des mémoires d'adresse successive.

Ex 5 : Écrire un programme qui effectue la multiplication de deux entiers positifs ou nuls entrés au clavier.

Ex 6 : En déduire un programme qui élève un nombre au carré.

Ex 7 : Écrire un programme qui effectue la division euclidienne d'un entier *a* par un entier *b*. On stockera le quotient et le reste.

Ex 8 : Écrire un programme qui donne les termes de la suite de Fibonacci, définie par :

$u_0 = 1$, $u_1 = 1$ et $u_{n+2} = u_n + u_{n+1}$

On obtient un terme en ajoutant les deux précédents, d'où les termes successifs : 1 1 2 3 5 8 13 21 34 55…

Ex 9 : Les nombres triangulaires sont les nombres 1, 3, 6, 10, 15 etc… On les obtient en calculant :

$1 = 1$

$3 = 1 + 2$

$6 = 1 + 2 + 3$

$10 = 1 + 2 + 3 + 4$

$15 = 1 + 2 + 3 + 4 + 5$

Ainsi 6 est le $3^{ème}$ nombre triangulaire, 10 est le $4^{ème}$, etc.

Écrire un programme qui, étant donné un entier rentré au clavier, renvoie :

- 0 si l'entier n'est pas un nombre triangulaire
- le rang si l'entier est un nombre triangulaire.

Ainsi, 7 renverra 0 et 15 renverra 5.

Ex 10 (difficile) : Suite de Syracuse.

Écrire un programme qui, étant donné un nombre entier positif en entrée, s'arrête lorsque la suite définie par $u_0 =$ ce nombre et $\begin{cases} u_{n+1} = u_n/2 & \text{si } u_n \text{ est pair} \\ u_{n+1} = 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$ donne 1 (la propriété n'est pas prouvée, mais on conjecture que c'est vrai).

On peut rajouter :

- la sauvegarde dans un registre du « temps de vol », c'est-à-dire de *n* pour lequel $u_n = 1$;
- la sauvegarde de « l'altitude maximale » atteinte, c'est-à-dire le maximum atteint par u_n .