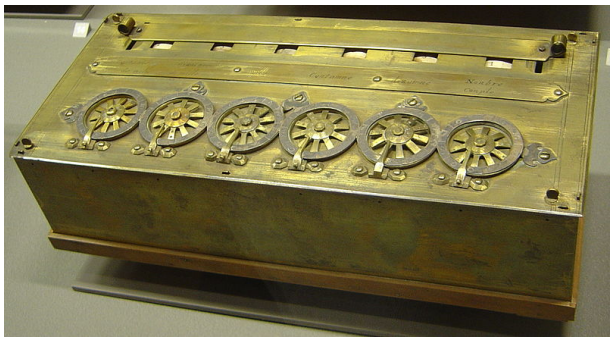
Architecture

1 Un peu d'Histoire

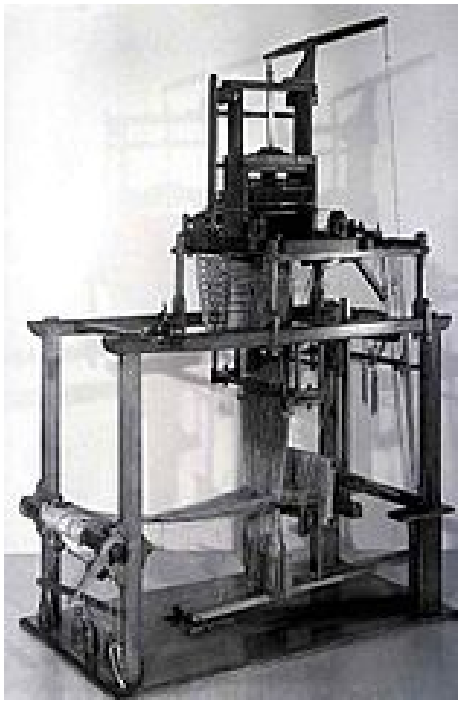
La Pascaline de Pascal (1650)

Il s'agit de l'une des premières calculatrices construites avec des roues dentées. Un procédé astucieux permet de réaliser des retenues*(Visible au musée des arts et métiers à Paris)*



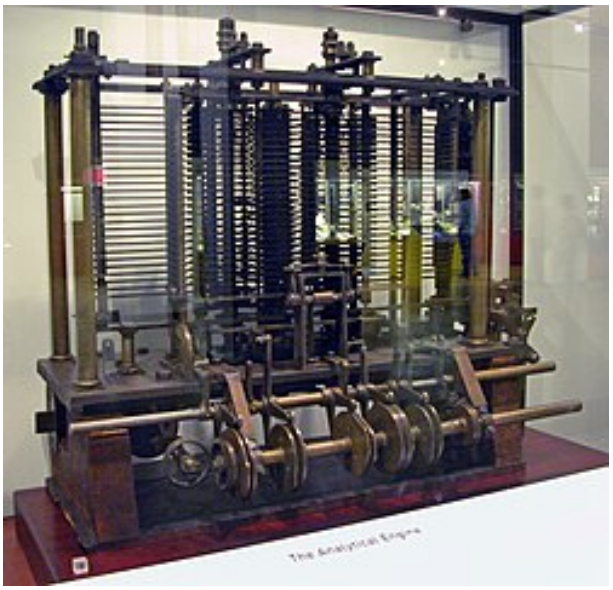
Métier à tisser de Jacquard (1801)

Il s'agit d'un automate capable de créer un tissu à partir d'une sorte de programme fait avec des cartes perforées
(Visible au musée des arts et métiers à Paris)



La machine analytique de Charles Babbage(1830, jamais achevée)

D'une certaine manière la machine analytique de Babbage est une version du métier à tisser de Jacquard destiné aux calculs.



Ada Lovelace et Babbage mettent aux points les premiers programmes écrits pour une machine *(Visible au Science Museum à Londres)*

Algèbre de Boole (1847)

Par analogie avec le calcul numérique, Boole introduit un calcul sur des grandeurs (les booléens) n'ayant que deux valeurs possibles (vraie ou faux) de telle sorte que le **ou** correspond à l'addition et le **et** correspond à la multiplication
Ainsi (A ou B)et$C \iff A$ et C ou A et B correspond à $(A+B)\times C = A\times C + B\times C$

Problème de décidabilité de Hilbert (1900)

Peut -on trouver un ***algorithme*** *permettant de décider, pour n'importe quelle équation diophantienne (c'est-à-dire équation polynomiale à coefficients entiers), si cette équation possède des solutions entières ?*
Par exemple $3x+2y=1$ est une équation diophantienne ayant une infinité de solutions les couples d'entiers $(-1+2k, 2-3k)|k \in \mathbb{Z}$ alors que l'équation $2x^2+2y^2=1$ n'a pas de solutions entières

Machine de Turing (1935)

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO
THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing.

[Received 28 May, 1936.—Read 12 November, 1936.]

Pour donner une réponse au problème de Hilbert, Turing introduit un objet théorique, appelé depuis lors *machine de Turing*

Voir ici une animation `https://interstices.info/comment-fonctionne-une-machine-de-turin`

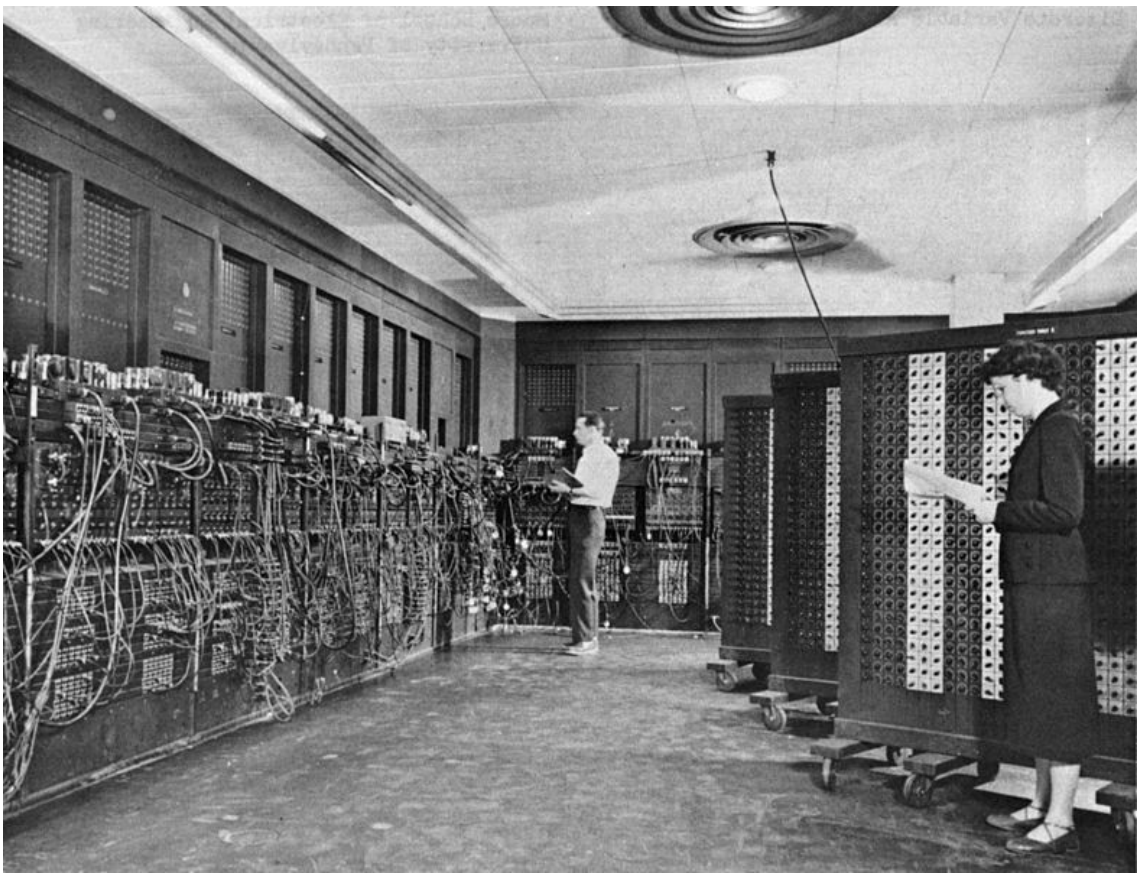
Le calculateur Harvard Mark 1 (1937)

Le calculateur mesurait environ 16 mètres de long et 2,6 mètres de hauteur et pesait 5 tonnes. Ce calculateur a été construit à partir des plans de la machine de Babbage. La technologie utilisée est celle des relais électromécaniques.

Le calculateur ENIAC(1943)

(Electronic Numerical Integrator And Computer) Ce calculateur a été construit en collaboration avec l'armée américaine et l'université de Pennsylvanie

La technologie est celle des tubes à vides électroniques (17468 tubes). La machine dans son ensemble pesait environ 30 tonnes pour une surface au sol de 160 mètres carrés



Machine de Von Neumann

Eckert et Mauchly, les deux principaux concepteurs de l'ENIAC avec Von Neumann définissent la machine de Von Neumann **(le programme et les données se trouvent dans la même mémoire)** et mettent au point l'EDVAC (Electronic Discrete Variable Automatic Computer) considéré comme le premier ordinateur

Invention de l'assembleur (1954)

Les programmes de l'EDVAC étaient écrits en **binaire**, le langage machine. Progressivement la nécessité de créer un langage intermédiaire entre le langage des humains et celui de la machine s'est fait sentir. Le langage assembleur est apparu en 1954

Invention du transistor (1947-> 1953)

Une des révolutions du vingtième siècle est l'invention du **transistor** qui remplacera les tubes à vides dans les ordinateurs et permettra leur miniaturisation

Fortran (Formula Translator) (1957)

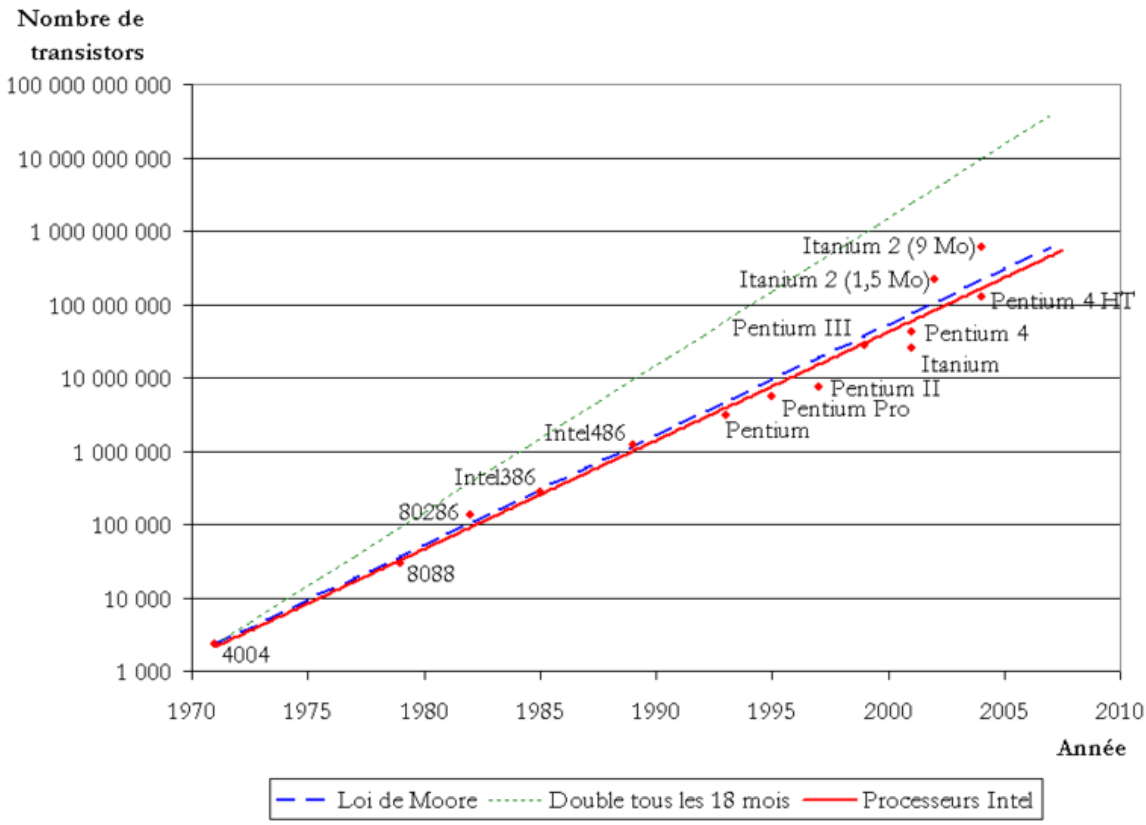
Dans le prolongement de l'assembleur apparaît le premier langage de programmation plus proche de l'Humain que de la machine. Destiné aux calculs scientifiques le FORTRAN sera étudié dans les universités jusqu'à la fin du vingtième siècle

Compilateur (1957)

Un compilateur est un programme qui traduit un programme écrit dans un langage évolué comme Fortran , C, Java, C++ en langage machine

Loi de Moore (1965)

"Le nombre de transistors dans un microprocesseur double à peu près tous les deux ans"



Le langage C (1972)

Le langage C est un langage de programmation dit de bas niveau car il permet de gérer des mécanismes liés aux tâches de bas niveau des ordinateurs (comme la gestion de la mémoire, ou les interruptions par le processeur)

Processeur 8080 (Intel) 1974

Microprocesseur 8 bits comportant 6000 transistors et une horloge de 2MHz

Le langage Python (1991)

Le langage Python est un langage de programmation dit de haut niveau dans le sens où il se rapproche de l'Humain, ainsi quand on parcourt une liste `L` afin de faire un traitement sur chaque élément de la liste `L`

En pseudo-code on écrit

```
Pour chaque élément de la liste L faire ...
```

En Python

```
for element in L:
    #...traitement ...
```

En 2022 ?

Les processeurs ont une dizaine de milliards de transistors, une fréquence d'horloge autour de 3 GHz et travaillent sur des mots de 64 bits

La mémoire vive des ordinateurs est environ d'une dizaine de Go

Microcontrôleurs

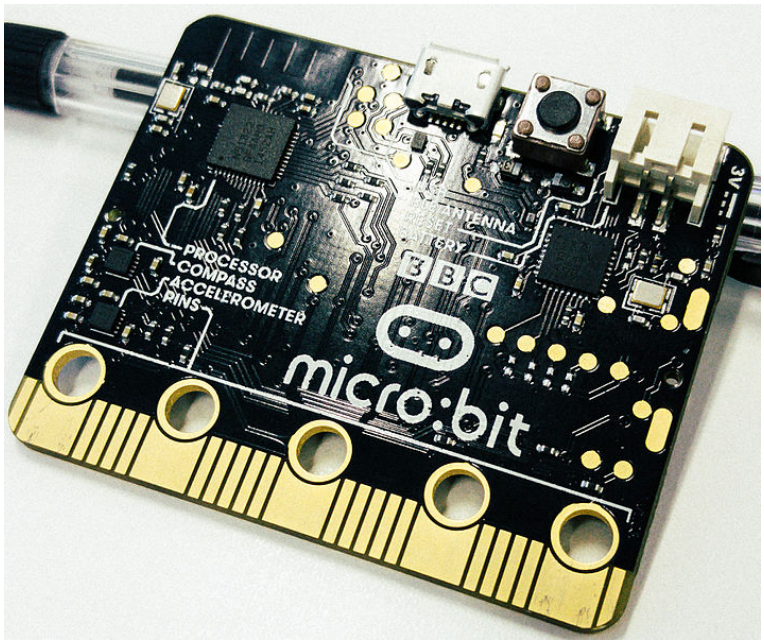
Il existe aussi une **informatique embarquée** (voiture, avion, machine à laver, plaque de cuisson , etc...) possible grâce aux microcontrôleurs

Un microcontrôleur est constitué :

1. d'un **processeur** aux performances moindres que celles des processeurs actuels équipant les ordinateurs, les tablettes et les smartphones, mais consommant peu d'énergie
2. de **capteurs** divers (accéléromètre, boussole, pression, température, etc...)
3. d'**actionneurs** (leds, speakers, etc...)

Exemples

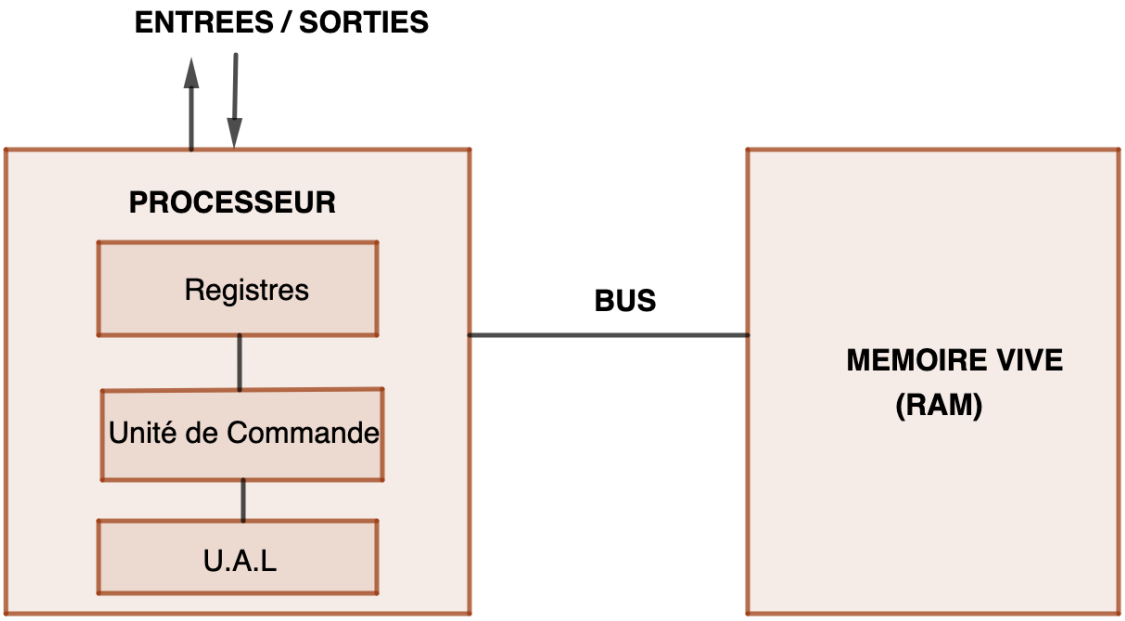
1. En appuyant sur une région bien précise d'une plaque de cuisson vitrocéramique, vous pouvez régler l'intensité de chauffage d'une plaque en particulier
2. Un son de plus en plus strident retentit dans l'habitacle d'une voiture lorsque celle-ci se rapproche d'un obstacle



Le microbit utilisé en TP a pour caractéristiques :

1. Microprocesseur : CPU ARM Cortex M0 32 bits à 16 MHz
2. Mémoire flash : 256 ko
3. Mémoire RAM : 16 ko
4. Dimensions : 40 x 50 mm
5. Poids : 8g

2 Modèle d'architecture séquentielle (Machine de Von Neumann)



L'originalité de la machine de Von Neumann est que **le programme et les données** sont dans la mémoire (vive) de l'ordinateur. Les éléments principaux sont :

1. un **processeur** constitué d'une :
   (a) **Unité arithmétique et logique (UAL)**
   (b) Des **registres** (des mémoires peu nombreuses et accessibles à l'unité arithmétique et logique pour y stocker des résultats de calculs)
   (c) Une **unité de commande (UC)**
2. Un **Bus** (fils permettant la circulation des données, des adresses mémoire ou des instructions entre la mémoire vive et le processeur)
3. **La mémoire Vive**
4. **Divers périphériques (écran, clavier,etc... :**

Voici le **cycle principal (en boucle) du processeur**

1. **(phase lecture)** Lit une instruction dans la mémoire à l'adresse n spécifiée par le registre PC (program counter) et la stocke dans un registre particulier appelé registre d'instruction
2. **(phase décodage et exécution)** L'instruction est découpée en bits significatifs, puis les bits significatifs sont distribués à tous les éléments du processeur (par exemple à l'UAL) pour exécuter des tâches précises (addition , opération logique etc...)

   Parmi ces bits significatifs dans certaines instructions peuvent figurer des indications de saut d'instruction ce qui signifie que la prochaine instruction n'est pas à l'adresse $n + 1$ (pour faire un test si ...alors... par exemple) Le résultat de l'exécution peut intervenir aussi dans un saut (boucle)

   Autrement dit l'actuelle instruction contient des indications avant et/ou après exécution décidant l'adresse de la prochaine instruction

Nous allons maintenant entrer plus en détail à travers quelques exemples "simples" sur le langage assembleur

3 Langage machine vs Assembleur

Pour visualiser le fonctionnement du processeur nous allons utiliser le simulateur en ligne `http://www.peterhigginson.co.uk/AQA/` mis au point par Peter Higginson d'un processeur ARM

ARM est un fabricant de processeurs intervenant dans les téléphones portables et les tablettes. Il existe d'autres fabricants de processeurs comme Intel, AMD pour les ordinateurs personnels et les ordinateurs portables
Il existe deux grandes familles de processeurs :

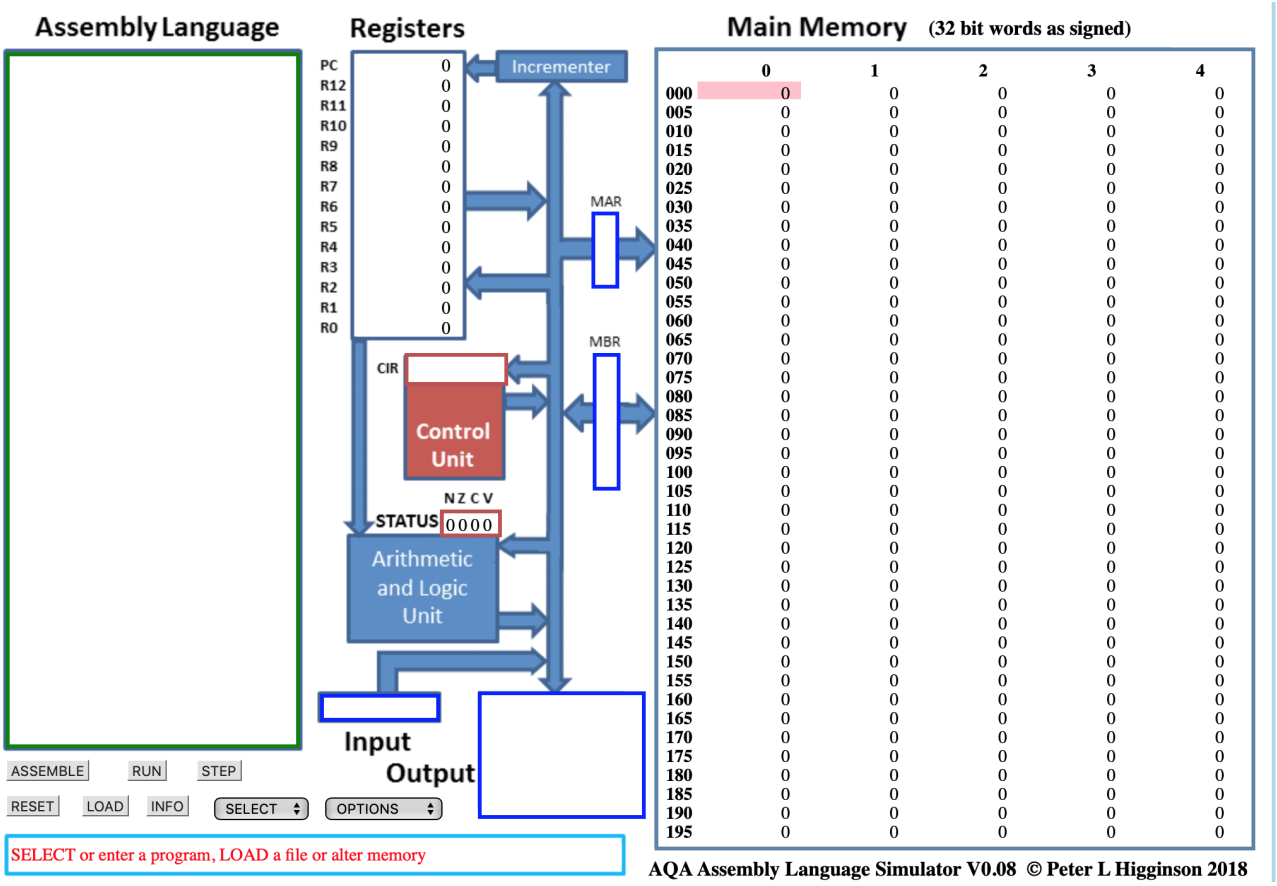
1. Les processeurs RISC (Reduced Integration Set Computing) : ARM, ...
   Le processeur n'a qu'un nombre limité d'instructions de base

2. Les processeurs CISC (Complex Integration Set Computing) : Intel, AMD
   Le processeur a un grand nombre d'instructions

Le langage machine ARM est constitué de mots de 32 bits (4 octets) dont le sens est donné ci-dessous par une copie d'écran d'un manuel d'un processeur ARM

l'opCode se trouve sur les 4 bits de 21 à 24 ainsi par exemple **1101** signifie mettre une valeur numérique dans un registre alors que **0100** signifie ajouter une valeur ou le contenu d'un registre au contenu d'un autre registre et mettre le résultat dans un registre de destination

Ainsi 11100011 10100000 00000000 00000100 signifie en langage machine "Mettre la valeur numérique 4 dans le registre R0" ce qui est codée dans le langage **assembleur**



MOV R0, #4
Voici "quelques phrases" du langage assembleur :
Les calculs ou les tests logiques se font avec l'UAL et les registres

1. **MOV R1,#4** signifie "Mettre la valeur décimale 4 dans le registre R1"
2. **MOV R1,R0** signifie "Mettre la valeur contenue dans le registre R0 dans le registre R1"
3. **ADD R1,R0,#4** signifie "Ajouter 4 à la valeur contenue dans R0 et mettre le résultat dans R1"
4. **ADD R2,R1,R0** signifie "Ajouter le contenu de R0 à celui de R1 et mettre le résultat dans R2"
5. **SUB R1,R0,#2** signifie "Soustraire 2 à la valeur contenue dans le registre R0 et mettre le résultat dans le registre R1"

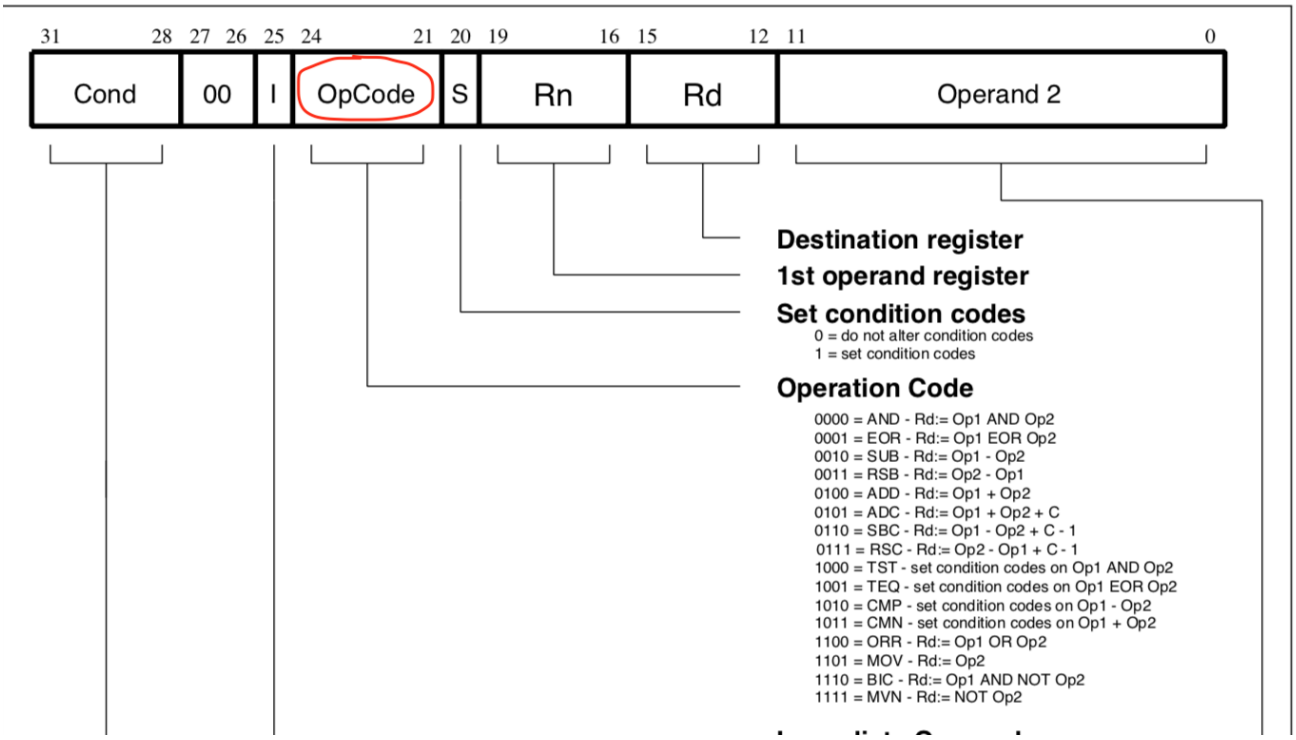
Le lien entre les registres et la mémoire. On ne peut pas utiliser des valeurs numériques avec LDR et STR

1. **STR R0,1** signifie "Enregistrer (Store Register) la valeur contenue dans le registre R0 à la place en mémoire repérée par l'adresse 1"
2. **LDR R1,3** signifie "Load Register = Mettre la valeur enregistrée dans la mémoire à l'adresse 3 dans le registre R1"

Le rôle de la commande HALT est d'arrêter l'exécution du programme et de séparer la zone du programme en mémoire de la zone des données.

The data processing instruction is only executed if the condition is true. The conditions are defined in ***Table 4-2: Condition code summary*** on page 4-5.

The instruction encoding is shown in ***Figure 4-4: Data processing instructions*** below.



Par exemple on peut utiliser des étiquettes (labels) comme des noms de variables pour stocker des valeurs numériques en mémoire, ainsi dans le programme suivant `nombre1` et `nombre2` sont des noms de variables apparaissant après HALT

```
//le programme
LDR R0,nombre1
LDR R1,nombre2
ADD R1,R0,R1
HALT
//les données
nombre1 : 3
nombre2 : 4
```

Ce programme en assembleur équivaut au programme Python suivant :

```
nombre1 = 3
nombre2 = 4
nombre2 = nombre1 + nombre2
```

Branchements ou sauts

Pour faire des tests et des boucles on peut faire des branchements après des comparaisons

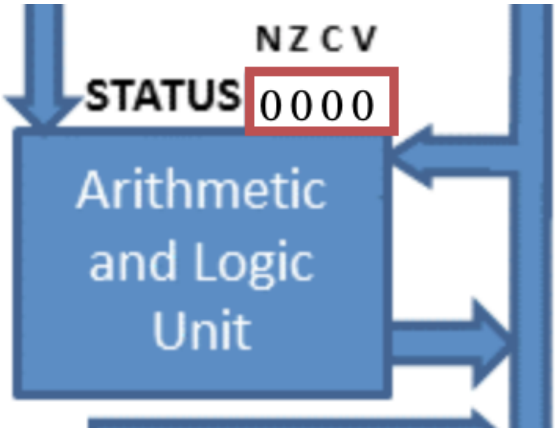
B 4 est une instruction qui "fait sauter" le programme à l'instruction se trouvant à l'adresse mémoire 4

B `findesi` est une instruction qui "fait sauter" le programme à l'instruction se trouvant à l'adresse étiquetée par le mot findesi

On peut "conditionner " un saut à une comparaison dans ce cas on fait succéder dans l'ordre d'abord une comparaison avec le contenu d'un registre, puis un saut conditionné par le résultat de la comparaison précédente

CMP R0,#val permet de comparer la valeur val au contenu du registre

Concrètement cette comparaison est une soustraction effectuée au niveau de l' Unité Arithmétique et Logique (UAL) du contenu de R0 par #val



1. l'UAL calcule R0 - #val
2. Si R0 - #val $<$ 0 le bit N (pour Negative) passe à 1 (pour Vrai)
3. Si R0 - #val $=$ 0 le bit Z (pour Zero) passe à 1 (pour Vrai)

4. Si R0 - #val $>$ 0 l'UAL ne fait rien et laisse les bits N et Z à 0

En fonction du résultat de la comparaison on peut faire différents sauts :

Regardons cela sur un exemple où on entre un nombre entier si ce nombre est égal à 4 on lui ajoute 1 sinon on lui retranche 1

En Python :

```
nombre = int(input("Entrez un entier "))
if nombre == 4:
    nombre = nombre + 1
else:
    nombre = nombre - 1
```

ce qui en Assembleur (ARM) donne :

Dans le code on a mis des commentaires avec des // pour expliquer les mots du code

```
//mot 0 : le nombre entré au clavier est stocké dans le registre R0
INP R0,2
//mot 1 : le contenu de R0 est stocké en mémoire vive à l'adresse étique
STR R0, nombre
//mot 2 : on calcule R0 - 4
CMP R0,#4
//mot 3 : si N = 0 le registre PC deviendra égal à 4
// si N = 1 le registre PC deviendra égal à  6 (saut)
BNE sinon
//mot 4 : on ajoute 1 au contenu de R0 et
//le résultat est mis dans R0
ADD R0,R0,#1
//mot 5 : on saute (le registre PC
B findesi
//mot 6 : c'est l'instruction à exécuter si R0 est différent de 4
sinon: SUB R0,R0,#1
//mot 7: il ne faut pas exécuter le mot 6 si R0 = 4
findesi: OUT R0,4
//mot 8 Fin
HALT
//la donnée
nombre: 0
```

Le programme et la donnée en code

```
0         INP R0,2
1         STR R0, nombre
2         CMP R0,#4
3         BNE sinon
4         ADD R0,R0,#1
5         B findesi
6 sinon:  SUB R0,R0,#1
7 findesi: OUT R0,4
8         HALT
9 nombre:0
```

Le programme et la donnée en mémoire sous forme de mots de 32 bits en hexadécimal

	0	1	2	3	4
000	xef010002	xe58f0018	xe3500004	x1a000001	xe2800001
005	xea000000	xe2400001	xef020004	xef000000	x00000001

Boucles

On entre un nombre entier au clavier et on répète 4 fois on ajoute 5 à ce nombre : Puis on affiche le résultat

En Python :

```
nombre = int(input("Entrez un entier "))
for i in range(4):
    nombre = nombre + 5
print(nombre)
```

En Assembleur

```
INP R0,2
//le registre R1 joue le rôle du compteur de boucle i
MOV R1,#0
//on calcule R1 - 3
boucle: CMP R1,#3
//si R1 - 3 > 0 on saute à finDeBoucle
//sinon on fait les instructions suivantes
BGT finDeBoucle
ADD R0,R0,#5
ADD R1,R1,#1
//on saute à boucle
B boucle
finDeBoucle: OUT R0,4
HALT
```

4 Exercices : Assembleur

Ex 1

Au musée des Arts et Métiers à Paris sont exposés des ordinateurs anciens comme l'ordinateur IBM 7030 et le CRAY 2 . En allant sur le Web rechercher les informations suivantes

1. Taille de l'ordinateur
2. Prix à l'époque de l'ordinateur
3. Nombre de transistors
4. Fréquence de l'horloge
5. Comparer avec les caractéristiques de votre smartphone et de votre ordinateur portable (processeur,fréquence d'horloge)

Ex 2

Aller sur le site d'Intel et relever les caractéristiques des processeurs INTEL de 2020 . Que signifie le terme lithographie ?

Ex 3

A l'aide du simulateur vérifier les opCodes de ADD et SUB

Ex 4

1. Que signifie l'instruction **ADD** R0,R0, #10 ?
2. L'instruction suivante est elle valide **ADD** R0,R0, R0 ?
3. Ecrire en assembleur successivement "La prochaine instruction à exécuter se situe en mémoire vive à l'adresse 100. Si la valeur contenue dans le registre R0 est égale à 1 alors la prochaine instruction à exécuter se situe à l'adresse 90"

Ex 5

Faire un programme en Assembleur pour obtenir le nombre 84

1. à partir de 50, 25, 8, 7, 3 et 1
2. à partir de 100 , 25, 8, 7, 3 et 1

Ex 6

1. Comprendre en Python l'instruction 5 » 2 et 5 « 2 (à la console)
2. Lire dans la documentation le sens des mnémoniques LSR et LSL

Ex 7

Ecrire en assembleur un programme qui affiche le plus grand de deux entiers entrés au clavier

Ex 8

Ecrire en assembleur : "Si un nombre entré est divisible par 2 afficher la moitié de ce nombre sinon afficher le triple de ce nombre plus 1"

Ex 9

Ecrire en assembleur un programme qui calcule la somme $1 + 2 + 3 + ... + n$ où n est un entier entré au clavier

Ex 10

Aller voir sur le Web l'algorithme de la multiplication égyptienne. Coder cet algorithme en Python, en Javascript et en assembleur