

Langage assembleur

Fiche de synthèse

① DONNÉES ET INSTRUCTIONS

Dans le modèle de von Neumann, la mémoire contient à la fois les données et le programme. Un même octet, 01000001, vaut 65 lu comme un entier, « A » lu en ASCII, ou un ordre lu par le processeur comme une instruction : c'est l'usage qui décide. Le processeur exécute le mot rangé à l'adresse que contient son **compteur de programme**.

② LANGAGE MACHINE ET ASSEMBLEUR

Définition 1 (Instruction machine) Une **instruction machine** est un mot binaire formé d'un **code d'opération**, qui dit quoi faire, et d'**opérandes**, qui disent sur quoi agir : un registre, une valeur ou une adresse. L'ensemble des opérations d'un processeur est son **jeu d'instructions**.

- Le **langage machine** est la suite de bits que le processeur exécute directement.
- Le **langage assembleur** écrit les mêmes instructions avec des noms courts, des **mnémoniques** : **ADD**, **SUB**, **MOV**, **LDR**.
- Le programme **assembleur** traduit le langage assembleur en langage machine, presque ligne pour ligne.

Chaque famille de processeurs a son jeu d'instructions : un programme en langage machine ne fonctionne que sur la sienne. L'assembleur est un langage de **bas niveau** : chaque ligne est une opération élémentaire du processeur. Les instructions se rangent en quatre familles : **transfert**, **calcul**, **comparaison** et **saut**.

③ INSTRUCTIONS DU SIMULATEUR AQA

Les opérandes s'écrivent **R_n** pour le registre numéro *n*, **#*n*** pour la valeur *n*, et **n** pour le mot de mémoire d'adresse *n*. Il n'y a pas de variables : le processeur ne calcule qu'avec ses registres, R0 à R12.

Instruction	Effet
MOV R1, #25	met 25 dans R1
LDR R1, 20	copie dans R1 le mot d'adresse 20
STR R1, 22	écrit R1 dans le mot d'adresse 22
ADD R3, R1, R2	range R1 + R2 dans R3
SUB R3, R1, #1	range R1 - 1 dans R3
CMP R1, #0	compare R1 à 0
BGT <i>etiquette</i>	saute si « strictement plus grand »
INP R1, 2	lit un nombre au clavier dans R1
OUT R1, 4	affiche R1
HALT	arrête le programme

④ UNE BOUCLE

```

MOV R0, #5
boucle: OUT R0, 4
        SUB R0, R0, #1
        CMP R0, #0
        BGT boucle
        HALT

```

Ce programme affiche 5, 4, 3, 2, 1. L'assembleur n'a ni **if**, ni **while** : on les fabrique avec des **sauts conditionnels**, vers une ligne repérée par une **étiquette**, ici **boucle**. Après une comparaison, des indicateurs signalent un résultat nul, négatif, une retenue ou un dépassement.

⑤ COMPILER OU INTERPRÉTER

Un langage de **haut niveau**, comme C ou Python, manipule des variables, des listes et des fonctions, loin des registres et des adresses. Il faut le traduire en langage machine :

- un **compilateur** traduit une fois le programme entier en un fichier exécutable : c'est le cas de C ;
- un **interpréteur** traduit et exécute les instructions une à une, à chaque lancement ;
- en méthode mixte, le programme est traduit en **byte-code**, exécuté par une **machine virtuelle** : c'est le cas de Python et de Java.

⑥ À QUOI SERT L'ASSEMBLEUR

On n'écrit presque plus de logiciel entier en assembleur : c'est long, difficile à relire, et lié à un seul type de processeur. Il reste utile pour quelques parties du système d'exploitation, pour les calculs les plus exigeants, pour les petits micro-contrôleurs, et pour comprendre ce que fait réellement la machine.

À retenir.

- Une instruction machine est un code d'opération suivi d'opérandes.
- L'assembleur écrit le langage machine avec des mnémoniques, propres à chaque famille de processeurs.
- Boucles et conditions se programment avec des comparaisons et des sauts.
- Un compilateur traduit tout le programme ; Python passe par un bytecode et une machine virtuelle.