1 Introduction

Dans le film Terminator de James Cameron sorti en 1984, les lignes de code du champ de vision du Terminator défilent parfois à l'écran. Elles sont écrites en "**langage machine**" : une suite de bits qui est interprétée par le processeur d'un ordinateur. C'est le langage natif d'un processeur, c'est-à-dire le seul qu'il puisse traiter. Il est composé d'instructions et de données codées en binaire. Le **langage assembleur** possède les mêmes instructions que le langage machine, mais ces instructions sont écrites en lettres. A chaque instruction en langage machine correspond une instruction en langage assembleur.

Chaque processeur possède son propre langage assembleur, pas nécessairement compatible avec d'autres processeurs. Par exemple, pour Terminator, c'est le langage de programmation assembleur 6502 d'un microprocesseur 8 bits nommé MOS Technology 6502 qui est utilisé.

Les **langages évolués** (comme Python, Java ou C) sont indépendants des processeurs utilisés. Les programmes sont traduits en langage machine par un compilateur ou un interpréteur.

Exemple pour une même instruction :

Langage Machine	Langage assembleur	Langage évolué
10100010 00000000	LDX #0	X = 0

2 Principe de fonctionnement

Chaque instruction est composée de 2 parties :

- le "code opération"" qui indique au processeur le type de traitement à réaliser. Par exemple, le code "10100010" donne l'ordre au CPU d'effectuer la commande LDX (en assembleur).
- les "opérandes" indiquent la nature des données sur lesquelles l'opération désignée par le "code opération" doit être effectuée. Par exemple, la valeur 0. Noter que pour désigner un nombre, le # est nécessaire, sinon c'est une adresse de la mémoire.

Dans le schéma ci-contre, le langage machine est en hexadécimal pour plus de clarté, et le code opération est séparé des opérandes éventuelles.
Pour chaque instruction, voici le suite des actions effectuées par le processeur :

1. Lire dans la mémoire l'instruction située à l'adresse stockée dans le pointeur d'instructions.
2. Décoder l'instruction lue.
3. Charger les opérandes éventuelles de l'instruction.
4. Incrémenter le pointeur d'instructions qui passe à l'adresse de l'opération suivante.
5. Exécuter l'opération décrite par l'instruction.
6. Reprendre à l'étape 1.

Ce fonctionnement est caractéristique de l'architecture de Von Neumann (les instructions et les données sont stockées dans la même mémoire).

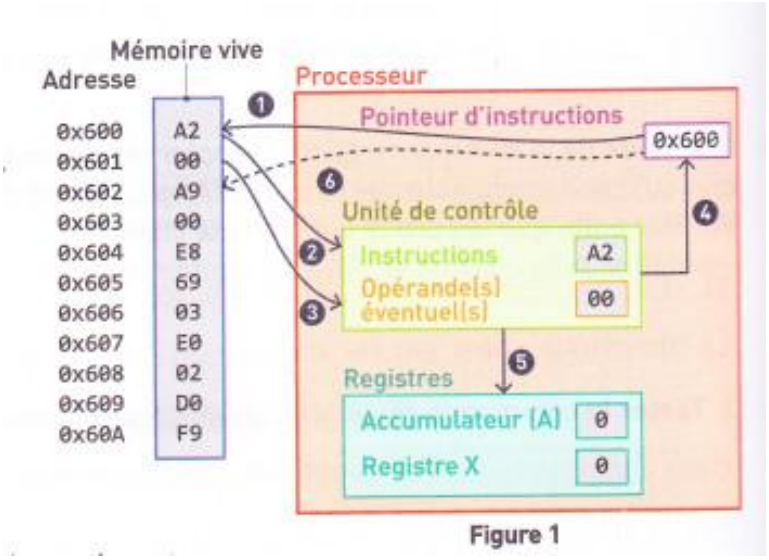

Figure 1

3 Langage Assembleur

Instruction de **déplacements** et de transfert de données :

Exemple	Description	*code machine en hexadécimal*
LDX #14	**Charge** la valeur 14 dans le registre X	**A2**
LDA #5	**Charge** la valeur 5 dans le registre A	**A9**
STA 123	**Stocke** la valeur du registre A à l'emplacement mémoire 123	
STX 325	**Stocke** la valeur du registre X à l'emplacement mémoire 325	
TAX	**Copie** la valeur du registre A dans le registre X	
TXA	**Copie** la valeur du registre X dans le registre A	

Opérations arithmétiques :

Exemple	Description	*code machine en hexadécimal*
ADC #18	**Ajoute** 18 à la valeur de A	**69**
ADC 605	**Ajoute** la valeur située à l'emplacement mémoire 605 à la valeur de A	
INX	**Incrémente** le registre X de 1 (c'est-dire ajoute 1)	**E8**
DEX	**Décrémente** le registre X de 1 (c'est-dire soustrait 1)	

Instruction de rupture de séquence :

Les instructions machines sont situées en mémoire vive. Au cours de l'exécution, le CPU passe d'une instruction à une autre dans l'autre. Une rupture de séquence (ou saut, ou branchement) consiste à interrompre l'ordre initial sous certaines conditions en passant à une instruction située à une adresse mémoire donnée.

Exemple	Description	*code machine en hexadécimal*
JMP 45	**Saut inconditionnel** : la prochaine instruction se situe en mémoire à l'adresse 45	
Pour faire un branchement conditionnel, on commence par effectuer un test :		
CPX #18	**Compare** la valeur du registre X avec le nombre 18	**E0**
CMP 45	**Compare** la valeur du registre A avec la valeur située à l'emplacement mémoire 605	
On effectue ensuite des actions en fonction du résultat du test qui précède :		
B 45	**Saut inconditionnel** : la prochaine instruction se situe en mémoire à l'adresse 45	
BEQ 78	Si le dernier test est ***égal***, on **saute** à l'adresse 78 (pour la prochaine instruction)	
BNE 125	Si le dernier test est ***différent***, **saute** à l'adresse 125	
BNE #12	Si le dernier test est ***différent***, **ajoute** la valeur 12 à la valeur du pointeur d'instructions *(attention, la valeur est codée en complément à 2 et peut donc être négative)*	**D0**

4 A vous de jouer

EXERCICE 0 : Rappel :

Convertir la valeur hexadécimale F9 en binaire puis en décimal sachant qu'elle est codée en complément à 2.

EXERCICE 1 :

1. A l'aide de la documentation précédente, décoder les instructions de la figure 1 en complétant le tableau suivant :

Adresse mémoire	Instruction avec opérande(s) éventuel(s) en hexadécimal	Traduction en langage assembleur	Instruction équivalente en Python si possible
0x600	A2 00	LDX #0	X = 0
0x602			
0x604			
0x605			
0x607			
0x609	D0 F9		Revenir en arrière de ..... adresses si la comparaison vaut False. Attention, le pointeur d'instruction a déjà été incrémenté, on passe donc de 0x60B à .....

2. Dérouler l'exécution du programme à la main en complétant le tableau ci-contre :

3. A quoi correspond l'instruction BNE en langage algorithmique ?

4. Quelle est l'opération mathématique réalisée au cours de ce programme ? Combien d'étapes ont été nécessaires pour la réaliser ?

5. Quelle instruction faudrait-il modifier afin que le programme effectue la multiplication de 3 par 4 ?

Etape	Pointeur d'instructions	Accumulateur (registre A)	Registre X	Comparaison
1	0x600	Inconnu	0	Inconnu
2	0x602	0	0	Inconnu
3	0x604	0	1	Inconnu
4				
5				
6				
7				
8				
9				
10				
11	0x60B	Suite du programme		

EXERCICE 2 :

Écrire un programme permettant de calculer 3 + 2 .

EXERCICE 3 :

Voici une suite d'instructions en langage Assembleur avec l'adresse mémoire où commence chaque instruction.

1. Expliquer pourquoi les adresses en mémoire ne sont pas toujours consécutives.

2. Y-a-t-il une boucle ? Justifier.

3. Que calcule cette suite d'instructions ? Détailler son déroulement.

Adresse	Instructions
600	LDX #01
602	LDA #02
604	STA 0
606	ADC 0
608	INX
609	CPX #5
60A	BNE 604
60C	STA 0