Langage machine

1. Les opérations du processeur

Les programmes stockés dans la mémoire centrale d'un ordinateurs sont constituées d'instructions de bas niveau, directement compréhensibles par le CPU. Il s'agit des instructions du **langage machine**. Lorsqu'elles sont stockées dans la mémoire, ces instructions ne sont ni plus ni moins que de simples nombres binaires, comme les données manipulées par les programmes.

Pour progresser dans l'exécution d'un programme, l'unité de contrôle de l'ordinateur réalise de manière continue, à un rythme imposé par l'horloge globale, la boucle suivante, appelée cycle d'exécution d'une instruction :

- Chargement de l'instruction (FETCH);
- Décodage de l'instruction : opérations et opérandes (DECODE)
- Exécution des opérations (EXECUTE)
- Ecriture du résultat (WRITEBACK)

La figure 1 illustre ce principe de traitement.

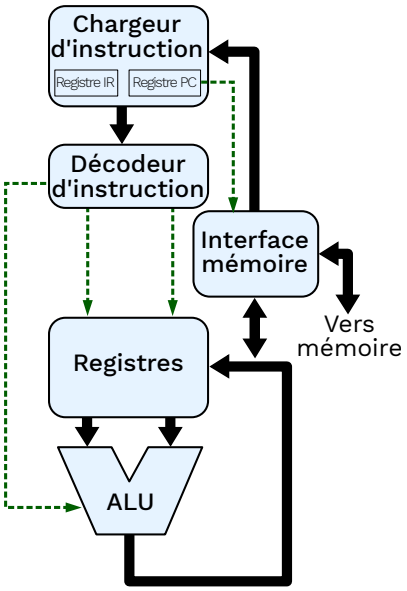

Figure 1: Principe de traitement d'une instruction machine

2. Description du cycle d'exécution

Chargement de l'instruction : l'unité de contrôle récupère à l'adresse mémoire indiquée par son **registre PC** (compteur programme), le mot binaire qui contient la prochaine instruction à exécuter et le stocke dans son **registre IR** (registre d'instruction). Le registre PC est ensuite incrémenté afin de pointer sur l'adresse mémoire de l'instruction suivante qui sera prise en compte lors de l'opération suivante.

Décodage de l'instruction : La suite de bits contenue dans le registre IR est décodée afin de déduire quelle instruction doit être exécutée et sur quelles données. Cette étape peut nécessiter de lire d'autres mots binaires depuis la mémoire si le format de l'instruction en cours de décodage le nécessite. C'est également à cette étape que sont chargées les données (on dit aussi opérandes sur lesquelles l'opération va porter ; ces données pouvant être dans des registres où en mémoire).

Exécution des opérations : L'instruction est exécutée, soit par l'ALU, s'il s'agit d'une opération arithmétique ou logique, soit par l'unité de contrôle, s'il s'agit d'une opération de branchement qui va donc modifier la valeur du registre PC

Ecriture du résultat : Le résultat de l'opération est tout simplement stocké en mémoire. Très souvent, les résultats sont écrits dans un registre interne au processeur pour bénéficier de temps d'accès très courts pour les instructions suivantes. Dans d'autres cas, les résultats sont écrits plus lentement dans des mémoires RAM.
Certains types d'instructions manipulent le compteur de programme (registre PC) , ces instructions sont appelées des sauts (*jumps*) et permettent de réaliser des boucles (*loops*), des programmes à exécution conditionnelle ou des fonctions (sous-programmes) dans des programmes.

Après l'exécution de l'instruction et l'écriture des résultats, tout le processus se répète, le prochain cycle d'instruction recherche la séquence d'instruction suivante puisque le compteur de programme avait été incrémenté. Si l'instruction précédente était un saut, c'est l'adresse de destination du saut qui est enregistrée dans le compteur de programme.

3. Les langages de programmation

Langage machine (bas niveau)

Le langage machine, ou code machine, est la suite de bits qui est interprétée par le processeur d'un ordinateur exécutant un programme informatique. 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 à traiter codées en binaire (affiché ci-contre en hexadécimal)
Chaque processeur possède son propre langage machine donc un code machine ne peut s'exécuter que sur lamachine pour laquelle il a été préparé.
Le code machine est aujourd'hui généré automatiquement, généralement par la compilation d'un langage de programmation.

```
0000: 2A21 3FFF 3FFF 3FFF 00FF 0E03 1283 1303
0008: 00A0 0E0A 00A1 0E04 00A2 118A 120A 2A24
0010: 0000 0000 0000 0000 0000 0000 0000 0BB4
0018: 2810 0008 08A6 1D03 281E 0008 30F9 3EFF
0020: 1D03 281F 0000 0BA6 281E 0008 01AF 01B0
0028: 01B2 01B3 01B1 1A31 0008 1003 0DB2 0DB3
```

Figure 2: Exemple de code machine en mémoire

Un langage d'assemblage ou langage assembleur ou simplement assembleur par abus de langage, abrégé ASM est, en programmation informatique, un langage de bas niveau qui représente le langage machine sous une forme lisible par un humain. Les combinaisons de bits du langage machine sont représentées par des symboles dits « mnémoniques ». Le programme assembleur convertit ces mnémoniques en langage machine en vue de créer un fichier exécutable.

```
ORG    0x0000

GOTO   Label_0001
ORG    0x0004
MOVWF  0x7F
SWAPF  STATUS ,W
BCF    STATUS ,RP0
BCF    STATUS ,RP1
MOVWF  0x20
SWAPF  PCLATH ,W
MOVWF  0x21
SWAPF  FSR    ,W
MOVWF  0x22
BCF    PCLATH ,03
BCF    PCLATH ,04
GOTO   Label_0002
```

Figure 3: Exemple de programme assembleur

Un langage de programmation (haut niveau)

Un langage de programmation est un langage informatique, permettant à un être humain d'écrire un code source qui sera analysé par une machine, généralement un ordinateur.
Le code source subit ensuite une compilation ou une évaluation dans une forme exploitable par la machine, ce qui permet d'obtenir un programme.

4. Jeu d'instructions simplifié des processeurs ARM

Dotés d'une architecture simple et bénéficiant d'une faible consommation électrique, les processeurs ARM sont devenus dominants dans le domaine de l'informatique embarquée, en particulier la téléphonie mobile et les tablettes.
Les processeurs ARM ont une architecture de type RISC (Reduced instruction set computer) qui se caractérise par un nombre d'instructions de base aisées à décoder, uniquement composé d'instructions simples.
Les tableaux suivants présentent un sous ensemble du langage des processeurs ARM. Pour la suite, et sauf mention contraire, dest et op1 désignent des registres et op2 un registre ou une valeur immédiate.
Les valeurs immédiates peuvent être :

- un entier en base 10 : #92
- un entier en binaire précédé par 0b : #0b00101
- un entier en hexadécimal précédé par 0x : #0xF3

Les registres

Ce langage comporte 13 registres R0 à R12. A ces registres s'ajoutent ceux pour manipuler les segments mémoires :

- CIR : registre d'instruction
- PC : compteur programme

Opérations arithmétiques et logiques

Syntaxe	Explications	Commentaires
MOV dest, op1	dest = op1	op1 peut être une valeur immédiate
ADD dest, op1, op2	dest = op1 + op2	
SUB dest, op1, op2	dest = op1 – op2	
AND dest, op1, op2	dest = op1 et op2	Le ET est fait bit à bit
OR dest, op1, op2	dest = op1 ou op2	Le OU est fait bit à bit
EOR dest, op1, op2	dest = op1 xor op2	Le XOR est fait bit à bit

Tests et sauts

Afin de stocker les résultats des tests, quatre bits spéciaux sont mis à jour lors des opérations suivantes :

- N vaut 1 si le résultat est négatif et 0 sinon ;
- Z vaut 1 si le résultat est nul et 0 sinon ;
- C vaut 1 s'il y a une retenue et 0 sinon ;
- V vaut 1 s'il y a un dépassement (overflow) et 0 sinon.

Ces valeurs sont modifiées par les instructions suivantes :

Syntaxe	Explications	Commentaires
CMP op1, op2	op1 = op2?	Met à jour NZCV en faisant op1 – op2
CMN op1, op2	op1 = –op2?	Met à jour NZCV en faisant op1 + op2
TST op1, op2		Met à jour NZ en faisant op1 et op2
TEQ op1, op2		Met à jour NZ en faisant op1 xor op2

Ces informations sont utilisées pour effectuer des sauts dans le programme. Pour cela, il est possible d'insérer des labels dans le code du programme. Lors d'une instruction de saut, si les conditions sont vérifiées, le programme saute à l'instruction indiquée par le label donné, sinon, il continue à l'instruction suivante. Les instructions de saut sont :

Syntaxe	Explications	Commentaires
B label	Saut inconditionnel	Le saut est toujours exécuté
BEQ label	Saut si Z = 1	Cas d'égalité
BNE label	Saut si Z = 0	Cas d'inégalité
BPL label	Saut si N = 0	Valeur positive ou nulle
BGE label	Saut si N = V	Cas supérieur ou égal
BGT label	Saut si N , V et Z = 0	Cas strictement supérieur
BLE label	Saut si N = V et Z = 1	Cas inférieur ou égal
BLT label	Saut si N , V	Cas strictement inférieur

Un exemple de programme

Le programme suivant permet de réaliser la multiplication des valeurs se trouvant dans les registres R0 et R1. Le résultat se trouve dans R2 à la fin de l'exécution

```
init    MOV R0, #4      ; R0 = 4
        MOV R1, #2      ; R1 = 2
        MOV R2, #0      ; R2 = 0 (résultat)
boucle  CMP R1, #0      ; Compare R1 = 0
        BEQ fin         ; Si R1 = 0 sauter à fin
        ADD R2, R0, R2  ; Sinon, R2 = R2 + R0
        SUB R1, R1, #1  ; R1 = R1 – 1
        B boucle        ; Recommencer
fin END                 ; Fin du programme
```

Ce programme effectue l'opération R2 = R0 + R0 + . . . + R0 autant de fois que nécessaire. Pour cela, R1 est décrémenté de 1 à chaque fois que R0 est ajouté à R2 jusqu'à R1 = 0.