```
Cours numéro 2 : notion d'instruction, d'interprète et de langage machine

I) Interprétation des instructions

  Les programmes, écrits dans un langage, sont des suites d'instructions
exécutées par un interprète.

  Algorithme d'interprétation = une boucle sans fin :
    * lire l'instruction courante en mémoire
    * décoder l'instruction : de quelle instruction s'agit-il ?
    * exécuter les actions spécifiées par l'instruction
    * passer à l'instruction suivante.

  Pour exécuter un programme écrit dans un langage L1, il faut soit :
    * traduire une seule fois tout le programme en langage L2 : compilation.
      chaque instruction L1 est décodée et traduite en L2 une seule fois.
    * écrire en langage L2 un programme interprète de langage L1 (algo
ci-dessus) : interprétation
      chaque instruction L1 est décodée autant de fois qu'elle est exécutée


  Compilation :    + efficace pour exécutions répétées/boucles
  Interprétation : + souple mise au point

  Comment exécuter un programme compilé ou un interprète écrit en
    L2 : compiler du L2 en L3 ou écrire en L3 un interprète de L2
    L3 : compiler du L3 en L4 ou écrire en L4 un interprète de L3
    ...
    Lj : compiler du Lj en LM ou écrire en LM un interprète de Lj

==> LM : langage machine binaire dont l'algo d'interprétation est câblé
         dans un circuit séquentiel : le processeur

   Pincipaux cas de figure :

   * Compilation C -> asm/machine X
        + interprétation directe matérielle par X  (C -> SPARC ou AMD64)
        + interprète programmé langage machine X exécuté sur processeur Y <> X
             code 68000 exécuté sur PowerPC,
             code ARM ou POWERPC exécuté sur AMD64 ...
   * Interprétation directe (BASIC)
   * Compilation en XX-code (d'une machine virtuelle portable) interprété.
        ==> P-code de PASCAL, bytecode de JAVA
   * compilation LL vers C + compilation C vers machine
        (cas de C++ à ses débuts).

II)  Modèle hérité de Von Neumann, compteur ordinal, vitesse, démarrage

  Instrutions executées en séquence dans l'ordre du programme stocké en memoire.
  Instruction machine courante pointée par registre compteur ordinal
ou compteur programme : PC (program counter) du processeur.
  Déplacé sur instruction suivante après chaque instruction.

  Rien ne distingue données et instructions en mémoire.
  Contenu interprété comme instruction quand pointé par PC du processeur,
  comme donnée sinon.

  Cadencé par horloge périodique.
  1 ou plusieurs cycles / instruction
        Mhertz (MHz) = Million /sec => 1 us (10^-6).
        Mhertz (GHz) = Milliard/sec => 1 ns (10^-9).
```



```
  Démarrage : signal "Reset" a la mise sous tension
  PC <- adresse (no emplacement) 1ère instruction du programme (ex: PC <- 0)
  Boucle sans fin jusqu'a coupure courant.
     lire instruction dans Mem[PC]
     décoder instruction et effectuer actions associées
     PC ++  ; passer à l'instruction suivante

III) Notion de langage machine et d'assemblage

Instruction/langage machine processeur : format binaire, seul compris par le
processeur.

Langage d'assemblage : idem  mais sous forme lisible (textuelle)

L'assembleur est le programmme traducteur texte -> binaire
Abus de langage : programmer "en assembleur" au lieu de "en langage
d'assemblage". (Presque) tout ce que permet le langage machine peut être
décrit en langage d'assemblage.

Gros inconvenient : fastidieux (opérations élémentaires) et non portable
Chaque famille de processeurs a son propre langage machine et
un plusieurs langages d'assemblage.

==> langage de programmation portable + traduction

Cas simple : 1 instruction ADA,C,PASCAL... simple =
             1 instruction processeur  ~= 1 cycle d'horloge


                 --------- processeur ARM (PDA, applis embarquées)
C                Assemblage              Binaire machine
                                         ... ... ... ... ... ... ... ...
r1  = r2 - r3    sub     r1,r2,r3        11100000010000100001000000000011
                                                    <2 ><1 >        <3 >
                                        0xE0421003
                                            ^^  ^
Cas - simple : il faut une séquence de plusieurs instructions machine pour
               faire l'equivalent d'une instruction ADA


r3  = r1 +  r2/4 + 7             add     r3, r1, r2, lsr #2
                                 add     r3, r3, #7

IV) Objectifs

A quoi ca sert de savoir programmer en

Lge machine bin/hexa : * comprendre le déroulement d'une instruction dans le
                         processeur, écrire un assembleur (traducteur)

Langage d'assemblage : * nécessité de manipuler des ressources spécifiques
                         du processeur (registres spéciaux) dans le noyau
                         d'un système d'exploitation
                       * écriture de bibliothèques, de compilateurs
                       * mieux comprendre ce qu'on manipule en C (pointeurs)
                       * observer le résultat d'un compilateur optimisant et
                        des erreurs éventuelles
                       * optimisation fine du code généré  (1 procedure)

IV) Notion de jeu d'instructions

Ce qui est connu du programmeur/compris par le processeur
```

```
Ensemble de registres   généraux / état<ZNCV> / PC

Instructions :

arithmétiques et logiques : addition / soustractions/ décalages etc

accès mémoire :

                                    load (chargement/lecture)
                                    <------------------------
                                        ldr reg, [ad]
                           registre                              Mem[ad]
                                        str reg, [ad]
                                    ------------------------>
                                   store (rangement/écriture)


controle et divers:     sauts/branchements
```