Pr: Ismaili Alaoui El Mehdi

Filière: SMI-4

Architecture des ordinateurs

Chapitre 3: Programmation en assembleur

1 Langage machine

Un langage machine est un langage compris directement par le processeur en vue d'une exécution. C'est un langage binaire : ses seules lettres sont les bits 0 et 1.
Chaque modèle de processeur possède son propre langage machine. Étant donnés deux modèles de processeurs P1 et P2 , on dit que P1 est compatible avec P2 si toute instruction formulée pour P2 peut être comprise et exécutée par P1. L'inverse n'est pas forcément vrai, P1 peut avoir des instructions supplémentaires que P2 ne connaît pas.
Dans la plupart des langages machine, une instruction commence par un opcode , une suite de bits qui porte la nature de l'instruction. Celui-ci est suivi des suites de bits codant les opérandes de l'instruction.

❒ Exemple:
La suite 10101011 00000000 00000010 (AB 00 02 en hexa)
est une instruction dont le opcode est 10101011 et l'opérande est 00000000 00000010.
Elle affecte la valeur 2 dans le registre AX.

Ce langage est très difficile à maîtriser puisque chaque instruction est codée par une séquence propre de bits. Afin de faciliter la tâche du programmeur, on a créé un langage assembleur.

2 Langage assembleur

Historiquement, le langage assembleur a été le premier langage de programmation non binaire accessible au programmeur. C'est un langage permet de décrire les instructions élémentaires qui manipulent dans le processeur les registres, l'accès à la mémoire. Le langage assembleur permet au programmeur:

➥ D'utiliser des codes symboliques (MOV, ADD, DIV, etc.) au lieu des codes numériques des opérations.

❒ Exemple:
Additionner 2 et 3 produit le code suivant en langage assembleur:
MOV AX,2
ADD AX,3
Traduit en langage binaire, il donnera :
10101011 00000000 00000010 (AB 00 02 en hexa)
11011011 00000000 00000011 (DB 00 03 en hexa)

➥ D'utiliser les adresses symboliques (étiquettes) au lieu des adresses numériques des emplacements mémoire.

Le langage assembleur est la représentation symbolique du langage machine. Autrement dit, le code en langage assembleur est la version lisible du code machine.
Le langage assembleur est donc un langage de très bas niveau. Cela signifie qu'il est plus proche du langage machine. Cette caractéristique offre des avantages et des inconvénients divers.

- Elle permet, entre autres,
  - ➥ d'exploiter au maximum les ressources matérielles. Il ne cache rien au programmeur, contrairement aux langages évolués (C++, Java, etc).
  - ➥ d'avoir un contrôle très fin sur la mémoire ;
  - ➥ souvent, de produire du code dont l'exécution est très rapide;
  - ➥ d'écrire les compilateurs des langages évolués.
- En revanche, elle ne permet pas
  - ➥ d'utiliser des techniques de programmation abstraites;
  - ➥ de programmer rapidement et facilement (le temps de programmation plus long).

Le langage assembleur est propre à chaque type du processeur. On dit que le langage assembleur est un langage "orienté machine" car il nécessite de penser d'abord à la machine avant de penser au problème à résoudre.

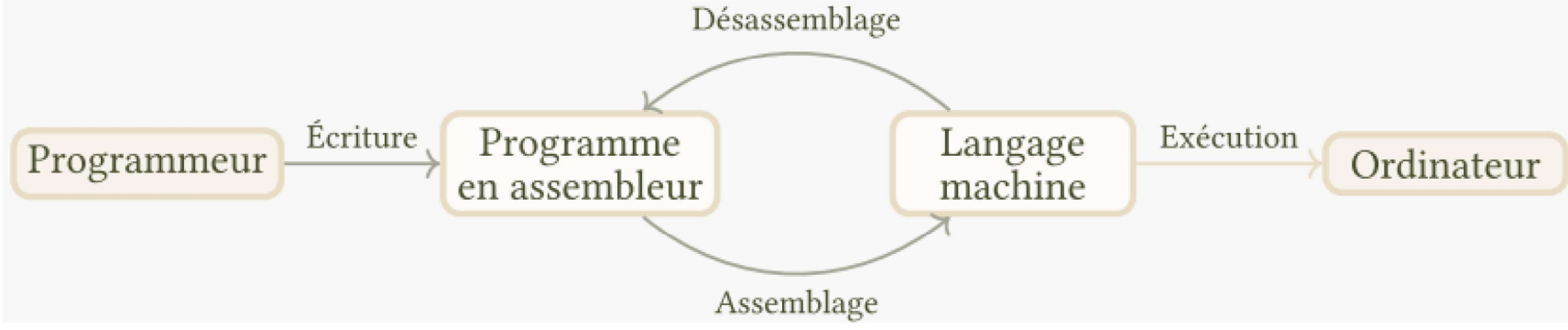
- ➥ Absence de portabilité (y compris sur même machine avec système d'exploitation différent)
- ➥ Il n'existe pas un langage assembleur, mais un langage assembleur par type de processeur.

Enfin, l'apprentissage de la programmation en langage assembleur permet de mieux comprendre le fonctionnement des autres langages et ainsi d'écrire des programmes plus performants.

3 Langages d'assemblage vs assembleur

L'assemblage est l'action d'un programme nommé assembleur qui consiste à traduire un programme en assembleur vers du langage machine.
Le désassemblage , réalisé par un désassembleur , est l'opération inverse. Elle permet de re-

trouver, à partir d'un programme en langage machine, un programme en assembleur qui lui correspond.
Voici le schéma opérationnel liant tout ceci :



4 De l'écriture du programme à son exécution

L'exécution de programme source se déroule en qutre étapes:

1. L'édition: La saisie du programme source au clavier nécessite un éditeur de texte. Ce programme doit sauvegardé avec un nom suivi de l'extension .asm.
2. La compilation (l'assemblage): Consiste à traduit chaque instruction du programme source en une instruction machine. Le résultat de l'assemblage est enregistré dans un fichier avec l'extension .obj.
3. L'édition de liens: Combine plusieurs fichiers objets afin de former un seul fichier complet et exécutable, avec l'extension .exe.
4. Exécution du programme : Un utilitaire spécial du système d'exploitation, *le chargeur* est responsable de la lecture du fichier exécutable, de son implantation en mémoire principale, puis du lancement du programme.

5 Outils de programmation

5. 1 Le compilateur

Il existe plusieurs compilateurs pour programmer en langage assembleur. Les plus connus sont :

— Masm (Microsoft Assembler): Pour MSDOS, Windows.
— Nasm (Netwide Assembler): Pour MSDOS, Windows et Linux.
— Fasm (Flat Assembler): Pour MSDOS, Windows et Linux.
— Tasm (Turbo Assembler) : Pour MSDOS et Windows.
— NBAsm (New Basic Assembler) : Pour MSDOS seulement.
— GoAsm : Pour Windows et sa programmation.
— ...

Pour des raisons pédagogiques, nous choisissons de travailler sur une architecture x86 en 16 bits. C'est une architecture dont le modèle de processeur est compatible avec le modèle Intel 8086. Nous utiliserons l'assembleur Fasm.

5. 2 Le débogueur

Le débogueur (debugger) est un programme qui facilite la mise au point de programmes et la correction des erreurs. Il permet d'examiner le contenu de la mémoire ainsi que celui des différents registres du processeur. Il permet également de créer des points d'arrêt (breakpoints) lors de l'exécution du programme à mettre au point, et à partir de ces points d'arrêt, d'exécuter le programme pas à pas, i.e. instruction par instruction tout en observant le contenu des registres, des variables et de la mémoire.