Langage assembleur

- ? Qu'est-ce ... ?
- ? Pourquoi utiliser l'assembleur...
- ? Inconvénients de l'assembleur...
- ? Pourquoi connaître l'assembleur...
- ? Les assembleurs...
- ? Déboguer un programme assembleur...

Le langage assembleur

Travailler en code machine n'est pas très aisé pour l'être humain. Il y a deux raisons essentiellement :

- EB, A1,.. Ces codes machines ne sont pas très parlants et donc difficiles à mémoriser ;
- Le calcul d'adresse est fastidieux : EB FB. Pourquoi FB ? De plus, il doit être recalculé à chaque modification du programme.

L'assembleur, tout en travaillant au même niveau que le langage machine, offre

- Des mnémoniques comme MOV AX,valeur ;
- Un calcul d'adresse à l'aide des labels comme JMP ici.

De plus, il connaît le format standardisé des fichiers exécutables tels que elf.

Pourquoi l'assembleur

Un processeur ne sait exécuter que du code machine. Tout ce qui se fait sur un ordinateur est du code machine. L'assembleur, qui est une vue plus humaine de ce code machine, permet de comprendre ce qui se se passe sur un ordinateur.
À chaque fois que l'on utilise un langage de plus haut niveau, on perd des possibilités et des performances. Les problèmes spécifiques (virus, pilote...) ou qui sont sensibles à la performance peuvent être réglés par l'utilisation de l'assembleur.

Pourquoi jamais l'assembleur

Écrire et maintenir un programme en assembleur est pénible. Comme il est le reflet du jeu d'instruction du processeur, il y a autant d'assembleurs que de processeurs. Quand on change de processeur, on change de jeu d'instruction et donc il faut tout réécrire. L'alternative devient (caricaturé !)

- développer pendant un an un programme en assembleur qui va s'exécuter pendant une seconde ;
- développer pendant un jour un programme dans un langage évolué qui va s'exécuter pendant une heure.

Ainsi, un programme 'one shot' est écrit dans un langage de très haut niveau, une routine du S.E. est en c (proche de l'assembleur) et une partie de routine du S.E. très utilisée est en assembleur.

Performances et assembleur

Un problème choisi a été écrit dans différents langages.
Voici les performances :

- en assembleur : 0.67 secondes.
- en c : 7.8 secondes.
- en java : 28.3 secondes

Il s'agit de trouver l'exposant de 2 le plus grand qui soit inférieur à un nombre donné.
Par exemple, si on donne 17, le programme cherche le plus grand x tel que $2^x < 17$, soit x=4.
Pour des questions de lisibilité, le temps est donné sur 25.000.000 recherches.

Pourquoi étudier un assembleur

Étudier l'assembleur du x86, c'est comprendre le fonctionnement du processeur x86.
Ce processeur est à la base de tous les PC et est représentatif des autres processeurs.
Le S.E. se base sur le fonctionnement du processeur pour implémenter la multiprogrammation, la pagination, la segmentation, le temps réel ...
Sans comprendre le fonctionnement d'un processeur, on ne peut pas comprendre les choix qui sont faits au niveau du S.E. ni comprendre convenablement le fonctionnement de ce S.E.
Les limitations des processeurs se retrouvent forcément au niveau de tous les langages.

Les assembleurs

Un compilateur assembleur propose souvent le choix de la cible (8086, 80286, 80386...) et adapte les messages d'erreurs et son travail en conséquence.
En linux, nasm permet de choisir la cible ainsi que le format de l'exécutable (elf pour nous).
Un compilateur génère un fichier objet qui doit être éventuellement lié avec d'autres objets pour former un exécutable à l'aide d'un éditeur de lien tel que ld.
Lors de l'édition de liens, il faut préciser l'adresse de la première instruction à exécuter. Cette instruction est appelée point d'entrée. Son adresse sera placée dans IP à la fin du chargement de l'exécutable en mémoire.

Déboguer un programme

Votre programme ne donne pas les résultats espérés. Il faut trouver l'erreur.

- Avez-vous écrit votre programme soigneusement ? Plus que pour tout autre langage, vous devez écrire votre programme seulement après avoir bien réfléchi au problème.
- Vous utilisez un debugger tel que ald ou gdb qui permettent d'exécuter votre programme instruction par instruction et de visualiser les registres et mémoires.
- Vous ajoutez des impressions de valeurs intermédiaires que vous vérifiez. La procédure printd a été écrite dans ce but.