Assembleur : Cours 2

I. Rappel sur Les différentes phases de la compilation

- Édition du fichier source : fichier texte contenant le programme — nécessite un éditeur de texte.
- Traitement par le préprocesseur : le fichier source est traité par un préprocesseur (un programme) qui fait des transformations purement textuelles (remplacement de chaînes de caractères, inclusion d'autres fichiers source, etc).
- La compilation : le fichier engendré par le préprocesseur est traduit en assembleur i.e. en une suite d'instructions associées aux fonctionnalités du microprocesseur (faire une addition, etc).
- L'assemblage : transforme le code assembleur en un fichier objet i.e. compréhensible par le processeur.
- L'édition de liens : afin d'utiliser des librairies de fonctions déjà écrites, un programme est séparé en plusieurs fichiers source. Une fois le code source assemblé, il faut lier entre eux les fichiers objets. L'édition de liens produit un fichier exécutable.

II. Rappel sur l'architecture des ordinateurs

➢ Un ordinateur est composé principalement :

- Des unités périphériques : Les unités périphériques sont des dispositifs permettant l'entrée (clavier, souris, CD-ROM,...) ou la sortie de données (écrans, imprimantes) ou encore le stockage permanent de ces données (disques, disquettes, bandes, ...).
- D'une unité centrale : L'unité centrale est composée de trois éléments indispensables :

a. L'unité de commande,
b. L'unité de traitement
c. La mémoire principale (ou centrale), cette dernière se partageant entre mémoire vive (à lecture/écriture) et mémoire morte (à lecture seule).

➢ Le processeur est l'ensemble constitué de

- L'unité de commande
- L'unité de traitement

Le processeur lit et écrit des informations en mémoire

- Il peut de plus effectuer des opérations arithmétiques et logiques
- Chaque action qu'il peut effectuer est appelée instruction
- Les instructions effectuées par le processeur sont stockées dans la mémoire
- Il dispose d'un petit nombre d'emplacements mémoire d'accès plus rapide, les registres.
- Un registre spécial nommé pc (program counter) (ou ip (instruction pointer)) contient à tout moment l'adresse de la prochaine instruction à exécuter

De façon répétée le processeur :

- lit l'instruction stockée à l'adresse contenue dans pc
- l'interprète, ce qui peut modifier certains registres (dont pc) et la mémoire

L’unité centrale d'ordinateur est donc l'ensemble processeur-mémoire
toutefois, dans le langage courant, le terme processeur désigne souvent la partie "exécutive" de l'unité centrale..
Les unités périphériques sont des dispositifs permettant l'entrée (clavier, souris, CD-ROM,...) ou la sortie de données (écrans, imprimantes) ou encore le stockage permanent de ces données (disques, disquettes, bandes, ...).

- Les familles des microprocesseurs

- L’histoire de la famille 80x86 d’Intel commence dans les années 70 avec le 8080, un processeur de 8 bits avec un bus d’adresses de 16 bits, qui pouvait adresser un total de 64 Ko.
- Vers 1980, le 8086 et le 8088 font leur apparition, ce dernier avec le premier PC d’IBM. Ce sont des processeurs de 16 bits avec un bus d’adresses de 20 bits, qui avaient une capacité d’adressage de 1 Mo. Le 8088 diffère du 8086 par la largeur du bus de données externe qui est de 16 bits dans le 8086 et de 8 bits dans le 8088.
- Toutefois, même si le bus d’adresses était de 20 bits, les registres internes d’adresses étaient toujours de 16 bits pour assurer la compatibilité avec le 8080. Comment donc accéder au reste de la mémoire? Toute la complexité des processeurs Intel vient de la solution adoptée à cette époque pour régler ce problème. On décida que l’adresse serait constituée des 16 bit des registres internes ajoutée à 16 fois le contenu d’un de quatre registres appelés registres de segment. Ces quatre registres étaient CS (Code Segment), DS (Data Segment), SS (Stack Segment) et ES (Extra Segment).
  On remarque que chaque segment a une taille de 64 Ko (offset 16 bits et 216), et que la distance entre chaque segment peut aller de 16 octets à 64 Ko. La capacité totale d’adressage est 0xFFFF0 + 0xFFFF = 0x10FFEF, qui dépasse légèrement 1 Mo (0xFFFFF).
- Le 80286 fait son apparition quelques années plus tard avec un bus d’adresses de 24 bits (capacité de 16 Mo). C’est là que les choses se compliquent.

- Mode réel et mode protégé

Jusqu’alors, les processeurs fonctionnaient en ce qu’Intel appelle le « mode réel ». Les systèmes d’exploitation utilisés avec ces processeurs étaient mono-tâches et mono-usagers. Les registres de segment contenaient de vraies adresses, et l’utilisateur pouvait accéder sans limite à toutes les ressources du système : les périphériques, les interruptions, etc.
Toutefois, les registres de segment demeuraient de 16 bits. Comment donc accéder aux 16 Mo que permettait le bus d’adresses de 24 bits du 80286?
Pour permettre cet adressage sur une plus grande plage de mémoire ainsi que l’avènement de systèmes d’exploitation plus performants, Intel introduisit avec le 80286 le « mode protégé ». Mais comme la plupart des applications roulant sous MS-DOS, qui dominait le marché, étaient incompatibles avec le mode protégé, on continua pendant des années à fonctionner en mode réel avec une capacité de mémoire de 1 Mo.
Le 80286 fut donc longtemps considéré comme un 8086 rapide parce que personne ne savait comment utiliser le mode protégé. Pourtant, ce processeur offrait la mémoire virtuelle, des droits d’accès pour la sécurité, des niveaux de privilège d’exécution, etc.
Pendant ce temps, Motorola mettait en marché la famille 68000, qui offrait des registres de 32 bits et, à partir de 1985 avec le 68020, une capacité d’adressage de 4 Go.

En 1987, Intel met au point le 80386, puis le 80486, ensuite, le Pentium et, finalement, en 1997, le Pentium II que nous retrouvons dans les ordinateurs que nous utilisons pour ce cours. Ils fonctionnent tous sous Windows NT 4.0.

III. Systèmes de numérations

Chiffre : Un chiffre est un symbole utilisé pour l'écriture des nombres.
Nombre : Un nombre est un moyen de représenter des grandeurs (des quantités).

- Le système décimal

Les nombres que nous utilisons habituellement sont ceux de la base 10 (système décimal).
Nous disposons de dix chiffres différents de 0 à 9 pour écrire tous les nombres.
D'une manière générale, toute base N est composée de N chiffre de 0 à N-1.

- Le binaire

Dans les domaines de l'automatisme, de l'électronique et de l'informatique, nous utilisons la base 2.
Tous les nombres s'écrivent avec deux chiffres uniquement (0 et 1). De même que nous utilisons le système décimal parce que nous avons commencé à compter avec nos dix doigts, nous utilisons le binaire car les systèmes technologiques ont souvent deux états stables.
A chaque état du système technologique, on associe un état logique binaire.
La présence d'une tension sera par exemple notée 1 et l'absence 0.
Le chiffre binaire qui peut prendre ces deux états est nommé "Bit" (Binary digit)
A chaque nouveau bit, le nombre de combinaisons possibles est doublé.
Ce nombre est égal à 2 puissance N (N étant le nombre de bits).
Un groupe de bits est appelé un mot, un mot de huit bits est nommé un octet (byte).

- Conversion d'un nombre binaire en décimal.

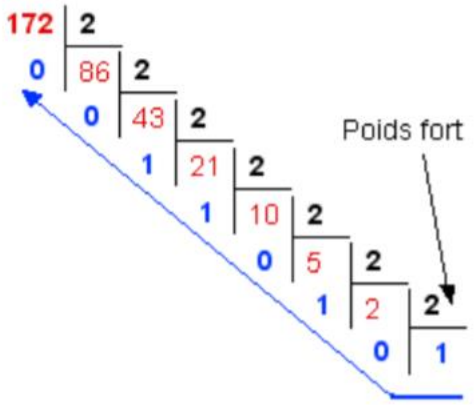
Il suffit de faire la somme des poids de chaque bit à 1
Le nombre ci-dessus est égal à 64 + 4 + 1 = 69
Conversion d'un nombre décimal en binaire (exemple : N = 172).
Méthode par soustractions.

172	44	12	4
- 128	- 32	- 8	- 4
44	12	4	0

$172 = 128 + 32 + 8 + 4$

$172_{(10)} = 10101100_{(2)}$

Méthode par divisions



- L'hexadécimal

La manipulation des nombres écrits en binaire est difficile pour l'être humain et la conversion en décimal n'est pas simple. C'est pourquoi nous utilisons de préférence le système hexadécimal (base 16).
Pour écrire les nombres en base 16 nous devons disposer de 16 chiffres, pour les dix premiers, nous utilisons les chiffres de la base 10, pour les suivant nous utiliserons des lettres de l'alphabet.

Décimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadécimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Les règles sont ici aussi les mêmes que pour le décimal.

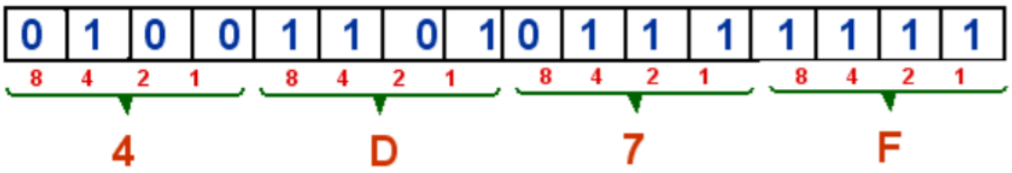
$A3F_{(16)} = (A \times 16^2) + (3 \times 16^1) + (F \times 16^0)$

$A3F_{(16)} = (10 \times 256) + (3 \times 16) + (15 \times 1)$

$A3F_{(16)} = 2560 + 48 + 15 = 2623_{(10)}$

Correspondance entre binaire et hexadécimal.
La conversion du binaire en hexadécimal est très simple, c'est d'ailleurs la raison pour laquelle nous utilisons cette base.
Il suffit de faire correspondre un mot de quatre bits (quartet) à chaque chiffre hexadécimal.

Conversion d'un mot de 16 bits entre binaire et hexadécimal



$4D7F_{(16)} = 0100110101111111_{(2)}$

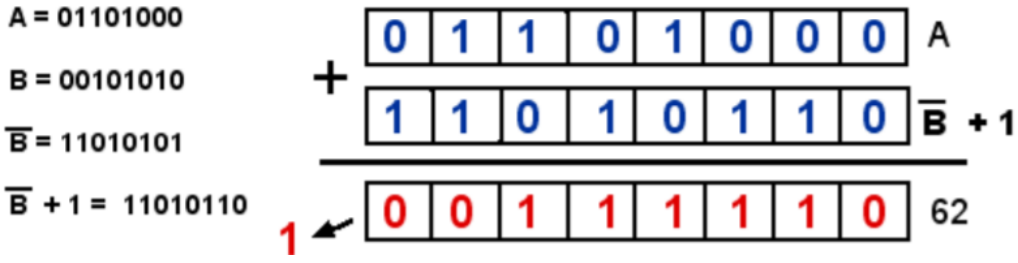
Correspondance entre décimal et hexadécimal.

La méthodes par divisions s'applique comme en binaire (exemple : N = 2623).

2623 / 16 = 163, il reste 15...

Les nombres signés
En binaire, le négatif d'un nombre est son complément à 2, c'est à dire son complément + 1.

Soient deux nombres **A = 104** et **B = 42.** A - B = A + (- B)



- Le codage ASCII

Le binaire permet de coder les nombres que les systèmes informatiques peuvent manipuler. Cependant, l'ordinateur doit aussi utiliser des caractères alphanumériques pour mémoriser et transmettre des textes. Pour coder ces caractères, on associe à chacun d'entre eux un code binaire, c'est le codage ASCII (American Standard Code for Information Interchange).
Le caractère A par exemple à pour code 65 soit 01000001 en binaire.
Le caractère f : 102
Le point d'interrogation ? : 63
Le chiffre 2 : 50

- L'Unicode

Développé par le Consortium Unicode, une organisation privée sans but lucratif, l'Unicode peut se résumer à un codage de caractères sur 16 bits. Il permet de coder l'ensemble des

caractères couramment utilisés dans les différentes langues de la planète en spécifiant un nombre unique pour chacun de ces caractères.
Au total, l'Unicode regroupe plus de 120 000 caractères parmi lesquels figurent les lettres des différents alphabets, mais aussi des symboles mathématiques, chinois, etc. Il couvre environ une centaine d'écritures.

- Jeu d'instructions

Le jeu d'instructions est l'ensemble des instructions machines qu'un processeur d'ordinateur peut exécuter. Ces instructions machines permettent d'effectuer des opérations élémentaires (addition, ET logique…) ou plus complexes (division, passage en mode basse consommation…). Le jeu d'instructions définit quelles sont les instructions supportées par le processeur. Le jeu d'instructions précise aussi quels sont les registres du processeur manipulables par le programmeur (les registres architecturaux).

- jeu d'instructions architecture intel x86

X86 est un jeu d'instruction commun à plusieurs processeurs. Le nom X86 provient des processeurs Intel utilisant ce jeu d'instructions (8086, 80186, 80286, 80386, 80486).
C'est le jeu d'instruction des processeurs équipant les ordinateurs personnels.
Les processeurs X86 (a partir du 386) ont huit registres de quatre octets chacun

eax	ax	ah	al
ebx	bx	bh	bl
ecx	cx	ch	cl
edx	dx	dh	dl
esi			
edi			
esp			
ebp			

Les registres eax, ebx, ecx et edx peuvent être découpés en registres plus petits
eax peut être découpé en trois registres : un registre de deux octets ; ax et deux registres d'un octet ; ah et al.
esp pointe sur le sommet de la pile.
ebp pointe sur l'adresse de base de l'espace local.

- Segmentation de la mémoire

La mémoire est divisée en segments indépendants.
L'adresse de début de chaque segment est stockée dans un registre.
Chaque segment contient un type particulier de données.

- Le segment de données permet de stocker les variables globales et les constantes. La taille de ce segment n'évolue pas au cours de l'exécution du programme (il est statique).
- Le segment de code permet de stocker les instructions qui composent le programme
- La pile permet de stocker les variables locales, paramètres de ` fonctions et certains résultats intermédiaires de calcul

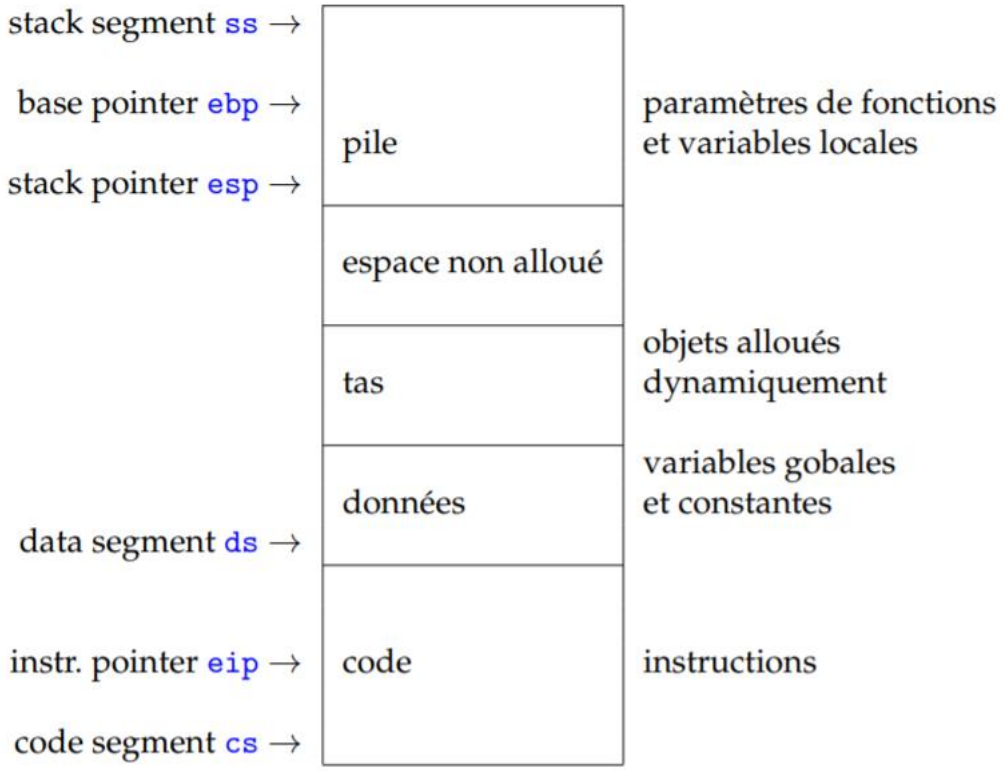
- Registres liés aux segments

Segment de code

- cs (Code Segment) adresse de début du segment de code

- eip (Instruction Pointer) adresse relative de la prochaine instruction à effectuer
- cs + eip est l'adresse absolue de la prochaine instruction à effectuer
  Segment de données
- ds (Data Segment) adresse de début du segment de données
  Pile
- ss (Stack Segment) adresse de la base de la pile
- esp (Stack Pointer) adresse relative du sommet de pile
- ss + esp est l'adresse absolue du sommet de pile
- ebp (Base Pointer) registre utilise pour le calcul d'adresses de variables locales et de paramètres

Segmentation de la mémoire



➢ Flags

Les flags sont des variables booléennes (stockées sur un bit) qui donnent des informations sur le déroulement d'une opération et sur l'état du processeur.
32 flags sont définis, ils sont stockés dans le registre eflags, appelé registre d'état.
Valeur de quelques flags après une opération :

- CF : Carry Flag, indique une retenue (CF=1) sur les entiers non signes. ´
- PF : Parity Flag, indique que le résultat est pair (PF=1) ou impair (PF=0).
- ZF : Zero Flag, indique si le résultat est nul (ZF=1) ou non nul (ZF=0).
- SF : Sign Flag, indique si le résultat est positif (SF=0) ou négatif (SF=1).
- OF : Overflow Flag, indique un débordement (OF=1) sur les entiers signes.