Partie 1: Généralités

Pourquoi Assembleur

- Langage plus proche de l’ordinateur (processeur).
- la vitesse d'exécution des programmes est nettement supérieure. Pas besoin de compiler!
- Tout programme, écrit dans n'importe quel langage, est, finalement, traduit en langage machine pour être exécuté.
- Programmation des **périphériques** (carte graphique, carte son, ...) est **accessible** pour les applications telles que systèmes d'exploitation, jeux vidéo.
- Permet de mieux comprendre comment fonctionne la machine et les autres langages.

Exemple:

En R.A.M. -> 0100 : 2D913AB80BAF484BBB66BD

CODE MACHINE	A**SSEMBLEUR**
0100 **:** 2D913A	SUB AX, 3A91H
0103 **:** B80BAF	MOV AX, 0AF0BH
0106 **:** 48	DEC AX
0107 **:** 4B	DEC BX
0108 **:** BB66BD	MOV BX, 0BD66H

Fonctionnement de l’assembleur

- Le travail d'un ordinateur est d'exécuter des programmes qui manipuleront des données numériques ou des caractères, il faut donc représenter **:**
  - des opérations;
  - des données;

Données:

- Soit de type numérique:
  - Des séquences binaires représentant des nombres positifs et négatifs
  - Codification: Binaire, Décimal, Hexadécimal…
- Soit de type caractère: *chaque caractère est composé de 8 bits:*
  - La table ASCII (Sur P.C., en D.O.S.)
  - La table ANSI (Sur P.C., en Windows)
  - La table EBCDIC (Sur main frame I.B.M.)

Les données d’un programmes

Un programme pourra travailler avec trois type de données:

- Des valeurs immédiates sont des nombres comme 100, 24, FFFFh.

- Des constantes ou des variables repérées par un ***identificateur***.
  - Le programme devra définir l'identificateur et le ***type*** de la variable
  - ou la ***valeur*** de la constante associée.

- Des zones de la mémoire. Le programme devra alors spécifier l'adresse et la longueur de la zone mémoire qu'il désire manipuler.

Exemples de données

Donnée		Valeur décimale
014BCh	constante exprimée en base hexadécimale	5308
742d	constante exprimée en base décimale	742
1101b	constante exprimée en base binaire	13
1101	constante exprimée en base décimale	1101
"A"	caractère	65
"OK"	chaîne de caractères	79 et 75

Registres du processeur

- Les registres sont des emplacements de mémoire situés à l'intérieur du microprocesseur,
- Ces registres ont chacun une fonction bien précise**:**
  - traiter les données en provenance de la mémoire
  - contenir les adresses de début de programme, de début des données, ...
  - indiquer le résultat des opérations arithmétiques
  - **...**

Registres du processeur

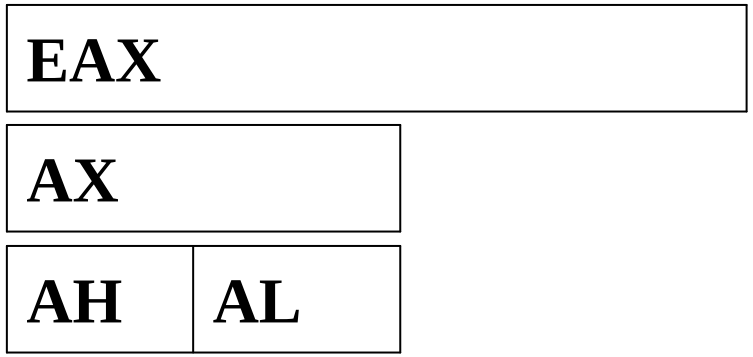
- les registres sont repartis en quatre catégories**:**

  - les registres de travail ou généraux;
  - les registres de segment;
  - les registres d'offset;
  - le registre FLAGS.

Les différents registres

Ils servent comme zone de stockage de résultats intermédiaires. Appelés souvent « Accumulateur » mais AX reste le privilégié. Les autres ont d'autres spécificité

AX = ACCUMULATEUR (16 bits)
BX = BASE
CX = COMPTEUR
DX = DATA
Ajouter **E** (extended) → étendu à 32 bits



RQ: Pour utiliser les Registres Extended il faudra rajouter "**.**486" au début du programme (à revoir)

Les registres de segments

Ils sont utilisés pour stocker des adresses: de début d'un segment.
Ex: adresse du début des instructions du programme, du début des données ou du début de la pile.

- **CS** = CODE SEGMENT

indique l'adresse du début des instructions d'un programme ou d'une sous-routine.

- **DS** = DATA SEGMENT

adresse du début des données de vos programmes. Si le programme utilise plusieurs segments de données, cette valeur devra être modifiée durant son exécution.

- **ES** = EXTRA SEGMENT

utilisé, par défaut, par certaines instructions de copie de bloc. En dehors de ces instructions, le programmeur est libre de l'utiliser comme il l'entend.

- **SS** = STACK SEGMENT

Il pointe sur une zone appelée la pile.

Les registres d'offset

- **SI** = SOURCE INDEX (16 bits): principalement utilisé lors d'opérations sur des chaînes de caractères; il est associé au registre de segment DS.

- **DI** = DESTINATION INDEX (16 bits): Il est normalement associé au registre de segment DS; dans le cas de manipulation de chaînes de caractères, il sera associé à ES.

- **IP** = INSTRUCTION POINTER (16 bits): Associé au registre de segment CS (CS**:**IP) pour indiquer la prochaine instruction à exécuter. **JAMAIS** modifié directement; il sera modifié indirectement par les instructions de **saut** , par les **sous programmes** et par les **interruptions** .

- **BP** = BASE POINTER (16 bits): Associé au registre de segment SS (SS**:**BP) pour accéder aux données de la pile lors d'appels de sous-programmes (*CALL*)

- **SP** = STACK POINTER (16 bits): associé au registre de segment SS (SS**:**SP) pour indiquer le dernier élément de la pile.

Le registre FLAG.

15	14	13 12	11	10	9	8	7	6	5	4	3	2	1	0
0	NT	IOPL	OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

- **CF**= Carry Flag (il y aune retenue retenue)
  PF = Parity Flag (le nombre des 1 est pair)
  AF = Auxiliary Flag (retenue auxiliaire)
  ZF = Zero Flag (le résultat est un zéro)
  SF = Sign Flag (signe) du résultat
  TF = Trap Flag (exécution pas à pas)
  IF = Interrupt Flag (interruption)
  DF = Direction Flag (direction de l'incrémentation)
- **OF** = Overflow Flag (débordement ou dépassement de capacité)
  IOPL = Input/Output privilege level (286 et plus)
  NT = Nested Task Flag (386 et plus)

Le registre FLAG

- La valeur représentée par ce nombre de 16 bits n'a **aucune** signification en tant qu'ensemble**:** ce registre est manipulé **bit par bit**.

- **Les Flags d'état :** (CF, PF, AF, ZF ,SF et OF)

  seront modifiés par certaines instructions. Il s'agit des instructions arithmétiques, logiques et des instructions de comparaison. Les instructions de saut conditionnel testent par la suite l'état des Flags en vue d'effectuer, ou non, un branchement.

- **Les Flags de contrôle :** (TF, IF, DF)

  donnent au processeur des indications quant au déroulement du programme. Ils peuvent être activés ou désactivés par le programme en cours.

Quelques liens utiles

- ftp://ftp-developpez.com/asm/cours/noteworthy/pas-a-pas-vers-l-assembleur-par-lord-noteworthy.pdf
- http://asm.developpez.com/cours/#initiation
- http://www.iprezo.org/index.php?page=asm
- http://luce.yves.pagesperso-orange.fr/plan.htm