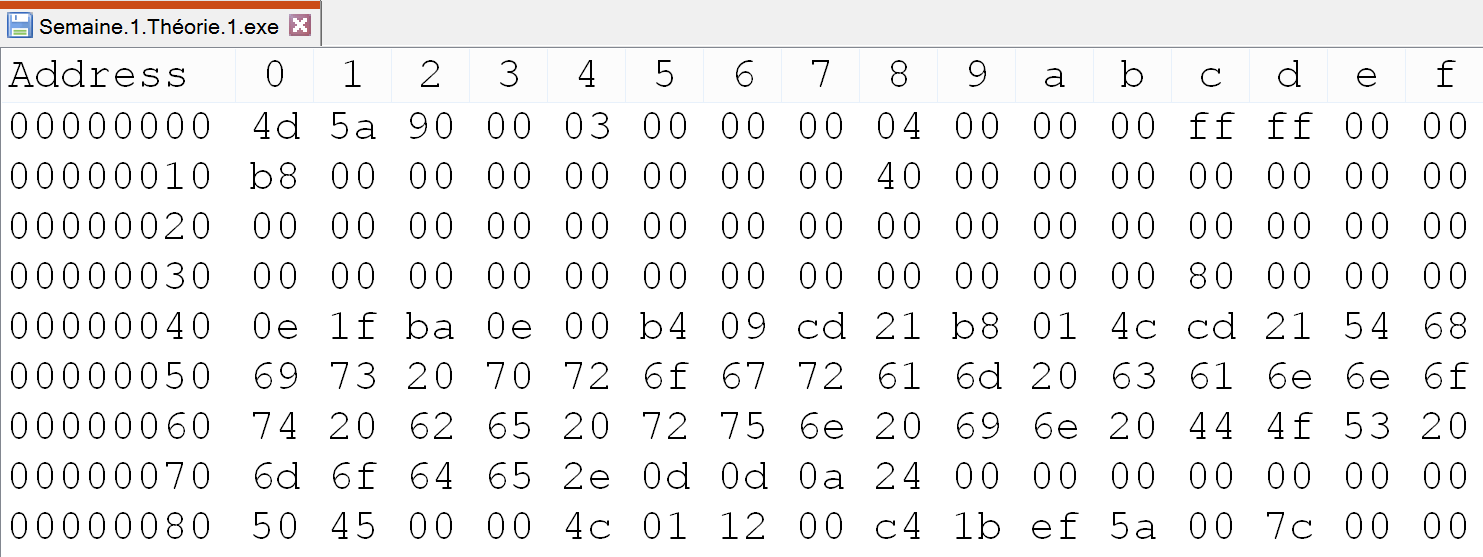
# Semaine 3 – Langage machine – Exercices

## Exercice n°3.1 – Adressage absolu de la mémoire



En visualisant le *dump* mémoire de l’exécutable Semaine.1.Théorie.1.exe, écrivez, pour chaque adresse absolue suivante, quelle valeur (sur 1 octet) se trouve en mémoire :

|  |  |  |
| --- | --- | --- |
| L’adresse absolue | pointe sur la | valeur en mémoire |
| 00000000 | 🡪 | 4d |
| 00000001 | 🡪 | 5a |
| 00000004 | 🡪 | 03 |
| 0000000D | 🡪 | ff |
| 00000018 | 🡪 | 40 |
| 0000004B | 🡪 | 4c |
| 00000060 | 🡪 | 74 |
| 00000077 | 🡪 | 0a |
| 0000008D | 🡪 | 7c |
| 0000008F | 🡪 | 00 |

## Exercice n°3.2 – Instruction MOV point de vue Architecture

Dessinez un schéma simple contenant les éléments suivants :

* Registre EAX initialisé à la valeur D3
* CPU
* Bus d’adresse
* Bus de données
* Mémoire RAM

Adaptez votre dessin pour montrer l’impact d’une instruction telle que MOV [1A2C],AL.

Précisez quelle est la valeur du bus d’adresse ?

1A2C

Et quelle est la valeur du bus de données ?

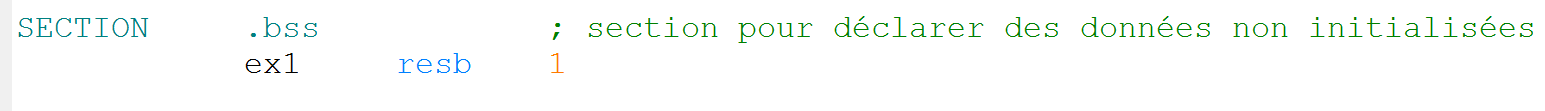
D3

## Exercice n°3.3 – Instruction MOV point de vue Langage (NASM)

Dans le langage NASM, les variables en mémoire sont nommées par des noms de variables qui seront transformés lors de la compilation en adresses. On peut dire qu’un nom de variable correspond à une adresse. On parle aussi de pointeur (vers la mémoire).

Nous ne l’avons pas vu en théorie mais il est possible de déclarer des variables en mémoire sans les initialiser. Cela s’écrit dans la section .bss de notre programme assembleur.

Déclarez comme suit une variable « ex1 » **non initialisée** :



resb 1 permet de réserver en mémoire 1 octet.

Remarque : il est possible de réserver en mémoire plusieurs octets avec resb, par exemple resb 5 permet de réserver 5 *bytes* (octets).

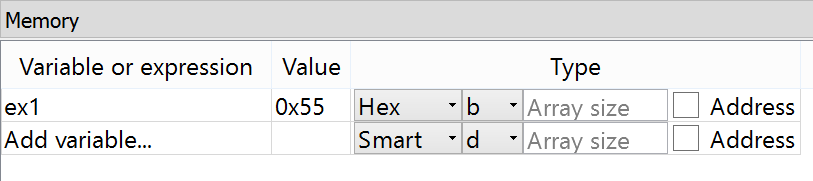
Dans le code, nous écrivons pour initialiser la variable « ex1 » en mémoire sur 1 *byte* à la valeur 0x55 :



Remarquez ci-dessus que la taille des deux opérandes de l’instruction MOV est fixée explicitement ! Comment est-elle précisée ?

Par le mot clé byte

Exécutez le mode *Debug* pour visualiser la variable « ex1 » en mémoire, on appelle cela inspecter, après exécution de l’instruction de copie :



Ecrivez ensuite l’instruction pour accueillir la valeur de la variable « ex1 » dans le registre EAX (situé dans le processeur) :

Attention : pour être précis, les deux opérandes doivent toujours être de même taille. Un registre dans une instruction fixe normalement la taille des deux opérandes.   
Écrire mov eax,[ex1] n’est donc pas correct !

A l’aide de l’IDE SASM, dans la section .data écrivez **les directives utiles pour** **initialiser** en mémoire les variables suivantes aux valeurs précisées à droite :

|  |  |  |
| --- | --- | --- |
| Nom de la variable | Valeur ou suite de valeurs | Remarque |
| ex2 | 0x55,0x56,0x57 | suite de 3 octets |
| ex3 | 'a',0x58 | le caractère ‘a’ suivi d’un octet |
| ex4 | 'hello',0 | la chaine de caractères « hello » suivie de la valeur 0 |
| ex5 | 0x1234 | le mot de 2 octets 0x1234, attention en mémoire il doit y avoir d’abord 0x34 et puis 0x12 |
| ex6 | 0x12345678 | le double mot de 4 octets 0x12345678, attention en mémoire il doit y avoir dans l’ordre : 0x78, 0x56, 0x34 et enfin 0x12 |
| ex7 | 0x4D5E6C7F | le double mot de 4 octets 0x4D5E6C7F, attention en mémoire il doit y avoir dans l’ordre : 0x7F, 0x6C, 0x5E et enfin 0x4D |

**Inspectez ensuite toutes ces variables à l’aide du mode *Debug* de SASM**.

Ensuite, accueillons la valeur de ces variables déjà initialisées dans des registres, écrivez les instructions utiles pour :

Copier le 1er octet de la variable « ex2 » dans AL, le 2ème octet dans BL, et le 3ème octet dans CL.

Vérifiez à l’aide du mode Debug de SASM que vous obtenez bien les valeurs suivantes :

EAX 0x00000055

EBX 0x00000056

ECX 0x00000057

Affichez le message « ex4 » à l’écran. Pour ce faire, vous avez besoin d’utiliser la macro de SASM intitulée PRINT\_STRING …

Copiez la valeur de la variable en mémoire ex6 dans EDX. Vérifiez que vous obtenez bien 0x12345678 dans EDX.

Expliquez à votre avis pourquoi les octets des valeurs de plus d’un octet sont stockés en mémoire dans l’ordre inverse ?

Pour pouvoir les déplacer plus facilement grâce aux sous-adresses.

## Exercice n°3.4 – Codage et décodage ASCII

Codez, selon le code ASCII, votre prénom suivi d’un espace suivi de votre nom :

436F72656E74696E

442768616579657265

Faites décoder votre codage par votre voisin.

Décodez, selon le code en ASCII, le codage suivant écrit en binaire :

0100 0010 0110 1001 0110 0101 0110 1110 0111 0110 0110 0101 0110 1110 0111 0101 0110 0101

42 69 65 6E 76 65 6E 75 65

Bienvenue

Décodez, selon le code en ASCII, le codage suivant écrit en hexadécimal :

4C4D

LM

C’est plus simple d’écrire en hexadécimal n’est-ce pas 😉