|  |  |  |
| --- | --- | --- |
| **Logo_Lycée** | **Entiers en base 2, 10, 16**  **[CORRECTION]** | **logo%20ac%20orl%E9ans%20tours** |
|  |

**Activité 5 – Microprocesseurs et langage machine**



*Motorola MC6800 (1975)*

**

*2017*

*« Le langage machine, ou****code machine****, est la* ***suite de***[***bits***](https://fr.wikipedia.org/wiki/Bit_(informatique))***qui est interprétée par le*** [***processeur***](https://fr.wikipedia.org/wiki/Processeur) *d'un*[*ordinateur*](https://fr.wikipedia.org/wiki/Ordinateur)*exécutant un*[*programme informatique*](https://fr.wikipedia.org/wiki/Programme_informatique)*. C'est le* ***langage natif d'un*** [***processeur***](https://fr.wikipedia.org/wiki/Processeur)*, c'est-à-dire le seul qu'il puisse traiter. Il est composé d'*[***instructions***](https://fr.wikipedia.org/wiki/Instruction_informatique)*et de* ***données*** *à traiter codées en*[*binaire*](https://fr.wikipedia.org/wiki/Syst%C3%A8me_binaire) *et situées en* ***mémoire*** *principale*. »[Wikipédia](https://fr.wikipedia.org/wiki/Langage_machine)

|  |  |
| --- | --- |
| **Mémoire**  **Adresses2  : Données2**  0000000000000000 : 11001110  0000000000000001 :00000000  0000000000000010 : 00011101  0000000000000011 : 11111111  | : |  0000000000101001 : 00000000  0000000000101010 : 00000000  0000000000101011 : 11111011  0000000000101100 : 00000000  | : |  ***Figure 1 : langage machine MC6800*** | La mémoire principale contient, sous forme binaire, le(s) programme(s) exécuté(s) par le processeur.  Elle peut être représentée comme une **table** dont les **cases contiennent les instructions et les données**. Chaque case est accessible par une **adresse**. Sauf rupture de séquence, le processeur exécute le programme en suivant l’ordre des adresses.  0 → 1 → 2 → etc.  Dans les années 70, la mémoire était organisée en **octets** comme ci-contre. L’**adresse** était composée de **16 bits**.  Le binaire étant difficile à manipuler pour un humain, les instructions et les adresses en machine sont représentées en **hexadécimal.** |

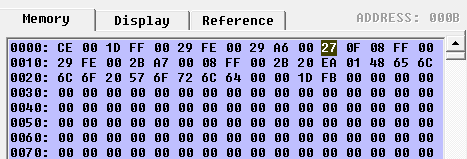
***1. Convertissez, en hexadécimal, la valeur située à l’adresse 4310 dans la mémoire ci-dessus. Quelle est la valeur hexadécimale de***

***cette adresse ?***

*Réponse :* 4310 = 32 + 8 + 2 + 1 = 1010112. A cette adresse on trouve la valeur 111110112 = **FB16** de plus 4310 = **2B16 soit** 002B **FB**

L’organisation de la mémoire ci-dessus est adaptée à l’étude de petites portions de code. Lorsque ce code est plus conséquent, les logiciels la présentent plutôt sous la forme ci-dessous.

Exemple : Mémoire (partiel) d’un micro-ordinateur à processeur MC6800



***Figure 2 : langage machine MC6800***

La première colonne représente les **adresses**. **Toutes les valeurs sont représentées en hexadécimal (base 16)**.

***2. À quelle adresse hexadécimale se situe la valeur EA16?***

*Réponse :* EA16 se situe à l’adresse **001B16** 001B **EA**

***3. EA16 est un entier relatif (complément à 2). Il représente un déplacement dans la mémoire. Donnez la valeur de ce déplacement en décimal.***

*Réponse :* EA16 = 111010102. b7 = 1 donc le nombre est négatif. |EA16| = + 1 = 000101012 + 1 = 000101102 = 2210 donc **EA16 = -22**

Pour ce type de déplacement, l’adresse de la prochaine instruction exécutée par le processeur se calcule avec la relation : [PC](https://webge.fr/6800.html#PC-reg) + [disp](https://webge.fr/6800.html" \l "disp-desc) + 2,

En complément à 2, le calcul se fait avec des nombres présentant le même format !

Ici 16bits. PC + 2 = 001C16  EA16 = FFEA16 (extension du bit de signe)

***4. Calculez cette adresse si PC = 001A16 et disp = EA16 ? Quelle est la valeur mémorisée à cette adresse ?***

Remarque : -12810 ≤ disp ≤ 12710 et il faut étendre le bit de signe pour effectuer l’opération.

001C16 0000 0000 0001 11002

+ FFEA16 1111 1111 1110 10102

= ~~1~~000616 ~~1~~ 0000 0000 0000 01102

*Réponse :* 001C16 + FFEA16 = ~~1~~0006. 1 est le débordement (17eme bit). On le supprime du résultat, car on calcule en complément à 2. La prochaine instruction se situe à l’adresse 000616 où est mémorisée la valeur **FE**

*Vérification :* EA16 = -2210. En reculant de (-22 + 2) octets à partir de 001A16, on atteint l’adresse 000616

Si l’adresse de l’instruction pointée par le microprocesseur est plus petite après le déplacement, on dit que le "saut" s’est fait "en arrière" dans le programme. Si elle est plus grande, le saut s’est fait "en avant".

***5. Comment peut-on qualifier le saut de la question 4 ?***

Le processeur "saute" de l’adresse 001A16 à l’adresse 000616 => **en arrière.**

**Activité 6 – Calcul d’un checksum**



Lors du transfert d’un programme à un microcontrôleur, un mécanisme permet de vérifier que les octets reçus par le microcontrôleur sont bien ceux que le PC a envoyés. Le logiciel de transfert divise le programme en paquets d’octets et joint un octet de checksum.

Exemple : paquet à transmettre au microcontrôleur (se termine par le checksum CA16)



Le checksum est le **complément à deux** de la somme des valeurs binaires des octets transmis (sauf lui). (Les calculs sont faits sur 8 bits, en ignorant les retenues.)

***6. Calculer le*** ***cheksum de la suite des valeurs hexadécimales 10 00 00 00 0C 94 5C.***

*Réponse : 1016 + C16 + 9416 + 5C16 = ~~1~~0C16 donc* ***checksum = 0C16***