**TP Architecture des ordinateurs**

**3 TP3 : Structuration des traitements**

* Structure de contrôle "if else"
* affichage d'un octet en hexadécimal.

**Mise en route**

* Lancez Turbo-C,
* Ouvrez le fichier AFF.C qui illustre l'affichage d'un caractère à l'écran au moyen de l'INT 21 du DOS.
* Compilez le et exécutez le en pas-à-pas sous Turbo-C (visualisez en même temps la fenêtre des registres : menu Window / Registers)

Pour **afficher un caractère à l'écran**, on utilise la fonction 2 de l'interruption 21 hexa fournie par le DOS. Il faut mettre 2 dans AH, le code ascii dans DL et exécuter INT 0x21.

* Ouvrez le fichier HEX.C.

**Le fichier Hex.c**

Méthode : codez **sous** chaque ligne de commentaire les quelques instructions assembleur correspondantes.

#include <stdio.h>

void aff\_hex( unsigned char car ) {

asm {

// -- affichage poids forts

// si <= 9

//   ajouter code ascii de '0'

// sinon -- si > 9

//   retirer 10 et ajouter le code ascii de 'A'

// fsi

// afficher

// -- afficher poids faibles

// si <= 9

//   ajouter code ascii de '0'

// sinon -- si > 9

//   retirer 10 et ajouter le code ascii de 'A'

// fsi

// afficher

}

}

int i, lignes;

int main() {

  clrscr();

  lignes = 0;

  for (i=0; i<=255; i++) {

        aff\_hex( i );

        lignes++;

        if (lignes == 25) {

          getch();

          lignes=0;

        }

        printf( "\n" );;

  }

  return 0;

}

**Indications**

En général, ce sujet trouble un peu les étudiants. En effet, la plupart du temps, on se contente d'utiliser printf pour afficher un nombre. Cela paraît tellement naturel qu'on ne se rend pas compte qu'un traitement est nécessaire pour passer du nombre lui-même tel qu'il est stocké dans la variable à une représentation affichable à l'écran, dans une base quelconque. Cela me fait penser au tableau de Magritte *"Ceci n'est pas une pipe"* qui représente bien sûr une pipe mais n'en est pas une. La représentation d'un nombre à l'écran est une chaîne de caractères, mais ce n'est pas le nombre lui-même.

Pour afficher un nombre en hexadécimal, il faut afficher deux caractères. Le premier représente le quartet de poids fort et le second, le quartet de poids faible. Le code ascii du 0 est 48 décimal (ou 30 hexa). Le code ascii du A majuscule est 65 décimal (ou 41 hexa).

**Pour caler le quartet de poids fort en poids faible**

4 décalages à droite (instructions SHR AL,1). Dans l'exemple 4 décalages sur 5A donnent 05.

**Pour isoler le quartet de poids faible**

Faire un et "bit à bit" avec la constante hexa 0F. Par exemple un "et" entre 5A et 0F donne 0A.

**Pour obtenir le code ascii**

Il faut envisager deux cas :   
Si le quartet est un chiffre compris entre 0 et 9 bornes incluses, il suffit d'ajouter 30 (hexa)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| caractère | '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' |
| code ascii | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |

Si le quartet est un chiffre compris entre A et F bornes incluses, il suffit d'ajouter (41 - 0A) hexa, 41 hexa étant le code ascii du caractère 'A'. 

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| caractère | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' |
| code ascii | 41 | 42 | 43 | 44 | 45 | 46 |

**Exemple avec car = 5A**

On peut résumer l'algorithme par le schéma suivant :

