Qroc A23 : Architecture des ordinateurs

Telecom Lille 1. Epreuve de 2 heures sans documents.

Matériel

On considère un processeur compatible 8086, mais muni d'un bus de données 16 bits et d'un bus d'adresses 16 bits. Pour ce processeur, on considère l'instruction `PUSH [var1]` qui empile la variable entière `var1`. On suppose que le pointeur de pile a la valeur `FFEE`, que la variable `var1` est située à l'adresse `2000` et contient la valeur `5`.

```
Addr   Opcode      Assembler
0100   FF360020    PUSH    [2000]
```

i) Quand on représente la mémoire octets par octets on voit :

```
0100   FF 36 00 20 00 00 00 00-00 00 00 00 00 00 00 00
0110   00 00 00 00 00 00 00 00-00 00 00 00 34 00 11 15
```

Compte tenu de l'ordre de stockage little-endian de ce processeur, que voit-on quand on représente la mémoire mots de 16 bits par mots de 16 bits ?

```
0100
```

ii) Quelle est la capacité d'adressage de ce processeur ?

iii) Représentez à l'aide d'un chronogramme l'activité du bus d'adresses, du bus de données et des signaux `RD#` et `WR#`, lors de l'exécution de cette instruction (lecture du code machine + exécution proprement dite). Annotez avec les valeurs hexadécimales véhiculées par les bus d'adresses et de données.

```
A0-A15

D0-D15

RD#

WR#
```

Modes d'adressage

On considère maintenant la plate-forme 16 bits TurboC / 8086. Tous les pointeurs et toutes les adresses sont supposées intra-segment.
On considère les variables `var1` (adresse `2000`) et `var2` (adresse `2002`).

```
mov  ax,0
mov  [var1],ax
lea  bx,[var1]
mov  [var2],bx
mov  [bx],1
```

Quelle est la valeur des registres et des variables suivantes ?

```
AX=

BX=

var1=

var2=
```

Représentation des données

En langage C, on considère les variables `i` et `p` d'adresse respective `2000` et `2002` :

```
int i = 0;
int *p = &i;
```

Donnez la valeur des expressions suivantes.

```
&p :


p :


*p :
```

Accès aux données

On considère les déclarations suivantes :

```
char v1 = 'A';
int v2 = ...; // valeur inconnue
int v3[10];
struct s1 {
  char *ch1;
  char *ch2;
};
struct s1 v4;
struct s1 v5[10];
struct s1 *v6 = &v4;
```

Codez en assembleur 8086 les instructions C suivantes :

```
v3[v2] = 0;
```

```
v4.ch1 = &v1;
```

```
v4.ch2 = 0;
```

```
v5[v2].ch2 = &v1;
```

```
v6->ch2 = v4.ch2;
```

Structures de contrôle

On suppose que `i` et `j` sont des variables entières globales. Codez en assembleur :

```
i=0;
while (i<9) {
  j=i+1;
  while (j<10) {
    ...
    j++;
  }
  i++;
}
```

En procédant, avec méthode, codez **sous** chaque instruction sur fond gris.

```
i=0;

while (i<9) {



  j=i+1;





  while (j<10) {



    ...
    j++;

  }



  i++;

}
```

Appel de fonction

Codez en assembleur un appel à la fonction `malloc`.

Elle admet un paramètre entier et fournit en retour une adresse.

Ci-dessous, la variable `p` est supposée globale :

```
char *p = (char *) malloc( 30 );
```

Fonction

On considère la fonction suivante :

```
int fonc( int tab[10], int from, int to, int init ) {
  int i;
  ...
  return from;
}
```

Quelles expressions assembleur faisant appel à l'adressage basé sur `BP` doit-on utiliser pour accéder aux variables suivantes :

```
from

to
```

Codez en assembleur :

```
tab[i] = init;
```