Université de Bordeaux
Licence STS

université de BORDEAUX

ARCHITECTURE DES ORDINATEURS

TD : **02**

Boucles et Pointeurs en y86

Exercice 1 : Boucle simple

Réalisez l'équivalent du code C suivant :

```
long sum = 0, i;
for (i = 1; i <= 10; i ++)
  sum += i;
```

Exercice 2 : Comptage de bits à 1

On veut réaliser un programme de « comptage de population », c'est-à-dire qui fournit le nombre de bits à 1 dans le codage binaire d'un nombre entier (son bit de signe y compris).

Question 1

Comment savoir si un bit spécifique d'un registre (par exemple, le bit codant 2^3) est à `1` ?

Question 2

Comment faire en sorte pour savoir si chaque bit d'un registre est à `1` ? Comment passer de bit en bit ?

Question 3

Réalisez le programme correspondant, qui lit la valeur d'un nombre en mémoire à une adresse d'étiquette `n`, et écrit le nombre de bits à 1 que contient ce nombre en mémoire à une adresse d'étiquette `c`.

Rappels

- Le processeur y86 dispose des trois modes d'adressage suivants :
  - immédiat : l'adresse mémoire à laquelle lire ou écrire est définie dans l'instruction elle-même. C'est donc une constante ;
  - indirect : l'adresse mémoire est contenue dans un registre, dont le numéro est défini dans l'instruction. On la note « `(%reg)` » ;
  - indexé : l'adresse mémoire est construite à la volée en additionnant une constante immédiate à l'adresse contenue dans un registre dont le numéro est lui aussi défini dans l'instruction. On la note « `cste(%reg)` ».
- Toute étiquette représente l'adresse, constante, de l'instruction ou de la directive située juste après cette étiquette.
- La différence entre les adresses de deux cellules `.long` consécutives en mémoire est égale à `sizeof(long)`, soit 4 octets pour des entiers stockés sur 32 bits.

Exercice 3 : Pointeur

Réalisez l'équivalent du code C suivant :

```
long a, b = 2, x = 3, *p;
p = &x;
...
a = *p;
*p = b;
p = &b;
...
(*p) ++;
```

Pour des raisons d'efficacité, ne définissez pas d'emplacement mémoire pour `p`, mais utilisez le registre `esi` à la place.

Exercice 4 : Parcours de tableau

Réalisez l'équivalent du code C suivant :

```
long t[4] = { 1, 2, 3, 4 };
long n = 4;

for (p = &t[0]; n > 0; p ++, n --)
  (*p) ++;
```

Pour des raisons d'efficacité, ne définissez pas d'emplacement mémoire pour `p` et `n`, mais utilisez les registres `esi` et `ecx` à la place.

Pour aller plus loin...

Exercice 5 : Écriture binaire

Soit un tableau de 32 entiers p_{31} , p_{30} , p_{29} , ..., p_1 , p_0.

Question 1

Écrivez un programme qui calcule dans un registre l'entier p dont les chiffres de sa représentation binaire sont les p_i, c.-à-d. qui convertit depuis la représentation binaire.

Conseil : chaque fois qu'on lit un nouvel élément p_i, remplacez p par $2 * p + p_i$.

Question 2

Comment traiter le cas où les entiers seraient plutôt dans l'ordre p_0 , p_1 , ..., p_{30} , p_{31} ?