Université de Bordeaux
Licence STS

université de BORDEAUX

ARCHITECTURE DES ORDINATEURS

TD : **03**

Utilisation de la pile en y86

Rappels

- La pile est une zone spécifique de la mémoire, gérée par un registre particulier : le pointeur de pile (« *stack pointer* »). En y86, le pointeur de pile est le registre `esp`.
- Les deux opérations élémentaires que l'on peut effectuer sur une pile sont :
  - l'empilage (« *push* »), qui ajoute une valeur au sommet de la pile,
  - le dépilage (« *pop* »), qui retire la valeur située au sommet de la pile ;
- On peut toujours représenter une expression mathématique sous forme d'un arbre, dont les nœuds sont les opérations et les feuilles, les valeurs. Cette expression peut alors être calculée de façon post-fixe. Pour cela :
  - on évalue l'expression en partant des feuilles, de gauche à droite ;
  - lorsqu'un sous-arbre a été évalué, on remonte d'un niveau, et on évalue le sous-arbre correspondant à la branche droite avant d'effectuer l'opération portée par le nœud ;
  - si l'on ne dispose pas d'assez de registres pour effectuer une opération, on peut empiler le résultat de la branche gauche jusqu'à ce qu'on revienne à son niveau.

Exercice 1 : Calculs avec deux registres et la pile

En n'utilisant que les deux registres `eax` et `edx`, ainsi que la pile comme stockage temporaire, réalisez l'équivalent du code C suivant :

```
a = (b | c) & (d | e);
```

Exercice 2 : Calculs avec deux puis trois registres et la pile

Question 1

En n'utilisant que les deux registres `eax` et `edx`, ainsi que la pile comme stockage temporaire, réalisez l'équivalent du code C suivant :

```
a = ((((b | c) & (d | e)) + ((f | g) & (h | i)));
```

Question 2

Si l'on s'autorise maintenant à utiliser les trois registres `eax`, `ecx` et `edx`, en plus de la pile, quelle partie du code peut-on optimiser ?

Rappels

L'appel de fonctions en y86, comme en x86, repose sur un certain nombre de conventions. Notamment :

- La valeur de retour d'une fonction est toujours contenue dans le registre `eax`.

Exercice 3 : Appel de fonction sans paramètres

On veut simuler le fonctionnement d'un distributeur de tickets à l'entrée d'un guichet, qui donnera un numéro de ticket correspondant à deux files : une file « normale » et une file « prioritaire ». La fonction `ticket()` doit donc prendre en entrée un numéro de file (`0` ou `1`), et renverra un numéro dans la file.

Question 1

Écrivez en C le code de la fonction `ticket ()`. D'habitude, on n'aime pas les variables globales, mais ici, pour mémoriser la position courante de chaque file, il faudra bien en avoir !

Question 2

Programmez en `y86` cette fonction. On lui passera dans `eax` le numéro de file, et elle retournera, également dans `eax`, le numéro courant dans la file correspondante.

Attention : pour prendre de bonnes habitudes, on demande de n'utiliser que les registres `eax`, `ecx` et `edx`.

Question 3

Créer un programme principal qui appelle cette fonction plusieurs fois, pour la tester.

Attention : pensez bien que la fonction peut modifier les registres `eax`, `ecx` et `edx`. Si vous les utilisez vous aussi, il faudra les sauvegarder.