TP Compilation : Génération de code pour un sous ensemble du langage λ-ada.

À partir de l'arbre abstrait construit lors du dernier TP, avec les outils JFlex et CUP, l'objectif consiste à générer du code pour la machine à registres décrite dans le cours, afin d'être en mesure d'exécuter les programmes reconnus par l'analyseur sur la machine à registres.

Exercice 1 :

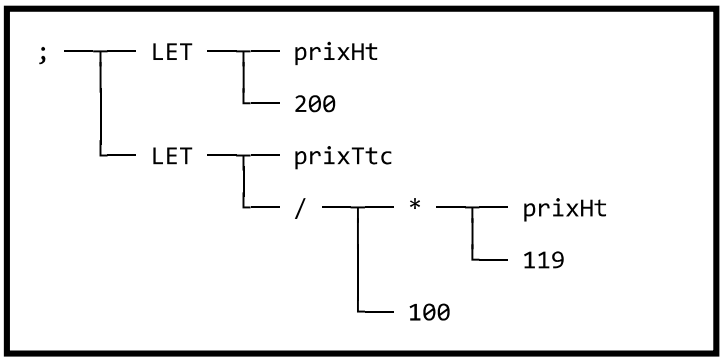
Dans la première partie du tp on pourra se limiter à la génération de code pour les expressions arithmétiques sur les nombres entiers.

Ainsi, l'expression

```
let prixHt = 200;
let prixTtc =  prixHt * 119 / 100 .
```

correspondant, par exemple, à l'arbre ci-dessous pourrait amener à la production du code suivant :

```
DATA SEGMENT
    prixHt DD
    prixTtc DD
DATA ENDS
CODE SEGMENT
    mov eax, 200
    mov prixHt, eax
    mov eax, prixHt
    push eax
    mov eax, 119
    pop ebx
    mul eax, ebx
    push eax
    mov eax, 100
    pop ebx
    div ebx, eax
    mov eax, ebx
    mov prixTtc, eax
CODE ENDS
```



Exercice 2 :

Étendre la génération de code aux opérateurs booléens, de comparaison, aux boucles et aux conditionnelles, correspondant au sous-ensemble du langage **λ-ada** utilisé pour le TP précédent.

Exemple de code source pour le compilateur : calcul de PGCD.

```
let a = input;
let b = input;
while (0 < b)
do (let aux=(a mod b); let a=b; let b=aux );
output a
.
```

Et un exemple de code qui pourrait être produit :

```
DATA SEGMENT
    b DD
    a DD
    aux DD
DATA ENDS
CODE SEGMENT
    in eax
    mov a, eax
    in eax
    mov b, eax
debut_while_1:
    mov eax, 0
    push eax
    mov eax, b
    pop ebx
    sub eax,ebx
    jle faux_gt_1
    mov eax,1
    jmp sortie_gt_1
faux_gt_1:
    mov eax,0
sortie_gt_1:
    jz sortie_while_1
    mov eax, b
    push eax
    mov eax, a
    pop ebx
    mov ecx,eax
    div ecx,ebx
    mul ecx,ebx
    sub eax,ecx
    mov aux, eax
    mov eax, b
    mov a, eax
    mov eax, aux
    mov b, eax
    jmp debut_while_1
sortie_while_1:
    mov eax, a
    out eax
CODE ENDS
```

Émulateur pour la machine à pile

Vous trouverez un émulateur pour la machine à registres ici : vm-0.8.jar.

Il s'utilise de la façon suivante :

```
java -jar vm-0.8.jar pgcd.asm
```

ou

```
java -jar vm-0.8.jar pgcd.asm --debug
```