Exercice 1. Une expression arithmétique telle que $((1+2)\times4)+3$ (forme infixe) peut s'écrire sous la forme suivante: $12+4\times3+$ ou même $3412+\times+$ (forme post-fixée). Donner le rapport entre cette notation et la notion de pile.

Exercice 2.Le langage PostScript, à l'instar de certaines calculatrices scientifiques Hewlet-Packard, utilise une notation postfixée, qui, de plus, peut se passer de parenthèses, les opérateurs ayant un nombre fixe d'opérandes. L'expression $3\times(4+5+6)$ s'écrit alors 3 4 5 add 6 add mul.

Lorsque l'interpréteur rencontre un opérande, il l'empile. Lorsqu'il rencontre un opérateur, il l'exécute sur les opérandes empilés et met le résultat de l'opération sur la pile. Ainsi, quand il rencontre l'opérateur add, il prend les deux nombres du haut de la pile et les remplace par leur somme. Il fait de même pour la multiplication avec l'opérateur mul. La pile aura donc successivement le contenu suivant :

()> push 3
(3)> push 4
(3 4)> push 5
(3 4 5)> add
(3 9)> push 6
(3 9 6)> add
(3 15)> mul
(45)>

Utilisez ces notations pour expliciter la version post-fixée et le comportement de la pile avec les expressions suivantes:

a. (1-2)+4+4+(3*2)
b. (3*4)+2+(3*4)+5+(3*4)+(3*8*3)

Exercice 3. Si on se donne les opérations sur la pile *dup* et *swap* peut-on réduire la taille de la seconde expression post-fixée de l'exercice précédent sans changer l'ordre des valeurs entières à entrer (de gauche à droite) mais en changeant l'ordre des opérations, en s'arrangeant pour rentrer moins de valeurs et en ne faisant aucun pré-calcul?

dup :duplique le sommet de la pile

swap :permute les deux valeurs au sommet de la pile.

Dans toute la feuille de td, on considère que les variables locales i, j, k sont respectivement les variables numérotées 1, 2, 3 dans la pile.

Exercice 4.Quel est le code assembleur pour le code JAVA suivant :

```
i=1;
k=2;
i=i+k ;
```

Exercice 5. Donner deux version de code assembleur correspondant au code JAVA suivant :

```
k=5;
j=5;
i=j+k+4 ;
```

Exercice 6.Quel est le code assembleur pour le code JAVA suivant :

```
i=1;
k=2;
aux=i;
i=k;
k=aux;
```

Peut-on faire mieux que le compilateur si on écrit cette permutation directement en assembleur?

Exercice 7.Quel est le code assembleur pour les 3 codes JAVA suivant :

```
j=3;
k=5;
i=j+k;
if (i==0)
     i=0 ;
     k=0 ;
else
     j=j-1 ;
```

```
j=3;
k=5;
i=j+k;
i=-1;
if (i<0)
     i=0 ;
     k=0 ;
else
     j=j-1 ;
```

```
j=3;
k=5;
i=j+k;
if (i<j)
     i=0 ;
     k=0 ;
else
     j=j-1 ;
```

Exercice 8.Quel est le code assembleur pour les 3 codes JAVA suivant :

```
j=0;
k=2;
if (j>=0)
     i=0 ;
else
     j=-1 ;
```

```
j=0;
k=2;
if (j>0)
     i=0 ;
else
     j=-1 ;
```

```
j=0;
k=2;
if (j>=k)
     i=0 ;
else
     j=-1 ;
```

Exercice 9.Voici un code assembleur IJVM :

```
.constant
a 10
.end-constant
.main
.var
i
j
k
.end-var
LDCW a
BIPUSH    6
ISTORE j
BIPUSH    4
ISTORE k
ILOAD     j
ILOAD     k
ISUB
ISTORE i
IINC i 6
.end-main
```

a. Donner sa traduction en code JAVA. Est-ce exactement équivalent?
b. Donner sa traduction en code hexadécimal ISA (byte-code).
c. A votre avis, peut-on savoir le temps d'exécution du programme sur une machine cadencée à 500Mhz ?

Exercice 10.Quel est le code assembleur, puis le byte-code , pour le code JAVA suivant :

```
i=3 ;
j=0;
if (i< j)
     j=j+1 ;
     k=i ;
else
```

```
j=i-1 ;
```

Exercice 11.

a. Quel est le code **assembleur IJVM**, <u>et aussi</u> **le code hexadécimal exécutable** des instructions correspondant, pour le code de haut niveau suivant :

```
x=-10 ;
y=0;
while (x=<0)
    {x=x+1 ;
    y=2*y+1;}
```

b. Donnez la valeur de y à la fin de l'exécution

Dans cet exercice, on considère que les variables x et y sont placées en mémoire respectivement aux adresses 1et 2 calculées à partir de la position stockée dans le registre LV. On considère que le compilateur n'est pas trop capable d'optimiser le code.

Exercice 12.Donner le code assembleur IJVM correspondant au code hexadécimal ci-dessous.

Method Area

Addr	Content
0x40000	0xb6 0x00 0x01 0x00
0x40004	0x01 0x00 0x03 0x10
0x40008	0x0a 0x36 0x02 0x15
0x4000c	0x02 0x9b 0x00 0x0d
0x40010	0x15 0x02 0x10 0x01
0x40014	0x64 0x36 0x02 0xa7
0x40018	0xff 0xf4 0x10 0x03
0x4001c	0x36 0x01 0x00 0x00

Constant Pool

Addr	Content
0x0	0x0
0x1	0x40003

Exercice 13.Donner le code assembleur IJVM correspondant au code hexadécimal ci-dessous. Donnez ensuite le code JAVA correspondant.

Method Area

Addr	Content
0x40000	0xb6 0x00 0x01 0x00
0x40004	0x01 0x00 0x04 0x10
0x40008	0xfd 0x36 0x01 0x10
0x4000c	0x05 0x36 0x02 0x10
0x40010	0x08 0x15 0x01 0x60
0x40014	0x36 0x03 0x15 0x02
0x40018	0x15 0x03 0x64 0x99
0x4001c	0x00 0x0a 0x15 0x01
0x40020	0x36 0x04 0xa7 0x00
0x40024	0x07 0x10 0x00 0x36
0x40028	0x04 0x10 0x00 0x36
0x4002c	0x01 0x00 0x00 0x00

Constant Pool

Addr	Content
0x0	Ox0
0x1	Ox40003