TD 7 - Code MVàP pour des fonctions

On se propose d'ajouter à notre calculette la possibilité de déclarer et ensuite d'appeler des fonctions.

Soit le code suivant

```
  JUMP Start
LABEL f
  PUSHL -3
  PUSHI 2
  MUL
  STOREL -4
  RETURN
RETURN
LABEL Start
  PUSHI 0
  PUSHI 2
  CALL  f
  POP
  WRITE
  POP
  HALT
```

et le résultat de son assemblage

```
 Adr |  Instruction
-----+---------------
   0 | JUMP       11
   2 | PUSHL      -3
   4 | PUSHI       2
   6 | MUL
   7 | STOREL     -4
   9 | RETURN
  10 | RETURN
  11 | PUSHI       0
  13 | PUSHI       2
  15 | CALL        2
  17 | POP
  18 | WRITE
  19 | POP
  20 | HALT
```

Qu 1. Commenter et compléter le début de trace suivant. Commenter le code original. Que réalise-t-il ?

```
  pc |              |    fp   pile
=====================================================
   0 | JUMP       11 |     0 [ ] 0
  11 | PUSHI       0 |     0 [ ] 0
  13 | PUSHI       2 |     0 [ 0 ] 1
  15 | CALL        2 |     0 [ 0 2 ] 2
   2 | PUSHL      -3 |     4 [ 0 2 17 0 ] 4
     |               |
     |               |
     |               |
     |               |
```

Soit le programme suivant ainsi que le résultat de son assemblage :

```
  JUMP START                          Adr |  Instruction
LABEL f                              -----+---------------
  PUSHL -3                             0 | JUMP       38
  PUSHI 2                              2 | PUSHL      -3
  MUL                                  4 | PUSHI       2
  STOREL -4                            6 | MUL
  RETURN                               7 | STOREL     -4
RETURN                                 9 | RETURN
LABEL g                               10 | RETURN
LABEL B1                              11 | PUSHL      -4
  PUSHL -4                            13 | PUSHL      -3
  PUSHL -3                            15 | PUSHI       1
  PUSHI 1                             17 | ADD
  ADD                                 18 | INF
  INF                                 19 | JUMPF      32
  JUMPF B2                            21 | PUSHI       0
  PUSHI 0                             23 | PUSHL      -3
  PUSHL -3                            25 | CALL        2
  CALL f                              27 | POP
  POP                                 28 | STOREL     -4
  STOREL -4                           30 | JUMP       11
  JUMP B1                             32 | PUSHL      -4
LABEL B2                              34 | STOREL     -5
  PUSHL -4                            36 | RETURN
  STOREL -5                           37 | RETURN
  RETURN                              38 | PUSHI       0
RETURN                                40 | PUSHI       3
LABEL START                           42 | PUSHI       5
  PUSHI 0                             44 | CALL       11
  PUSHI 3                             46 | POP
  PUSHI 5                             47 | POP
  CALL g                              48 | WRITE
  POP                                 49 | POP
  POP                                 50 | HALT
  WRITE
  POP
  HALT
```

Qu 2. Compléter les traces d'exécution suivantes. Identifier les appels de fonctions. Repérer les blocs d'activation. Que réalise le code ?

```
  pc |               |    fp   pile
=================================================
   0 | JUMP       38 |     0 [ ] 0
  38 | PUSHI       0 |     0 [ ] 0
  40 | PUSHI       3 |     0 [ 0 ] 1
  42 | PUSHI       5 |     0 [ 0 3 ] 2
  44 | CALL       11 |     0 [ 0 3 5 ] 3
  11 | PUSHL      -4 |
     |               |
     |               |
     |               |
  18 | INF           |     5 [ 0 3 5 46 0 3 6 ] 7
  19 | JUMPF      32 |     5 [ 0 3 5 46 0 1 ] 6
  21 | PUSHI       0 |     5 [ 0 3 5 46 0 ] 5
  23 | PUSHL      -3 |     5 [ 0 3 5 46 0 0 ] 6
  25 | CALL        2 |     5 [ 0 3 5 46 0 0 5 ] 7
   2 | PUSHL      -3 |     9 [ 0 3 5 46 0 0 5 27 5 ] 9
   4 | PUSHI       2 |     9 [ 0 3 5 46 0 0 5 27 5 5 ] 10
   6 | MUL           |     9 [ 0 3 5 46 0 0 5 27 5 5 2 ] 11
   7 | STOREL     -4 |     9 [ 0 3 5 46 0 0 5 27 5 10 ] 10
   9 | RETURN        |
     |               |
     |               |
     |               |
  11 | PUSHL      -4 |     5 [ 0 10 5 46 0 ] 5
  13 | PUSHL      -3 |     5 [ 0 10 5 46 0 10 ] 6
  15 | PUSHI       1 |     5 [ 0 10 5 46 0 10 5 ] 7
  17 | ADD           |     5 [ 0 10 5 46 0 10 5 1 ] 8
  18 | INF           |     5 [ 0 10 5 46 0 10 6 ] 7
  19 | JUMPF      32 |     5 [ 0 10 5 46 0 0 ] 6
  32 | PUSHL      -4 |     5 [ 0 10 5 46 0 ] 5
  34 | STOREL     -5 |     5 [ 0 10 5 46 0 10 ] 6
  36 | RETURN        |     5 [ 10 10 5 46 0 ] 5
  46 | POP           |     0 [ 10 10 5 ] 3
  47 | POP           |     0 [ 10 10 ] 2
  48 | WRITE         |     0 [ 10 ] 1
10
  49 | POP           |     0 [ 10 ] 1
  50 | HALT          |     0 [ ] 0
```

Qu 3. Quel code MVàP doit être produit par le compilateur pour effectuer le calcul suivant ?

```
int x = 1
int foo(int i) {
    return x+i
}
foo(3)
```

Qu 4. Même question avec ces calculs :

```
int abs(int x) {
    if (x<0) then return -x;
    return x
}
abs(1)+abs(-1)
```

```
int x = 1
int y = 2
int add(int i, int j) {
    return i+j
}
add(x,y)
```