Compilation — TD 5
Génération de code intermédiaire

J. Chroboczek et P. Letouzey

15 Novembre 2006

0. Construisez le code intermédiaire correspondant à l'instruction suivante.

```
2 + (3 * 5) - 8
```

1. Construisez le code intermédiaire correspondant au code suivant. Mettez-le ensuite sous forme canonique, et linéarisez-le. On supposera que l'on se trouve au niveau 1.

```
printint(2 + (3 * 5) - 8)
```

2. Pour chacun des fragments de code suivants, construisez le code intermédiaire correspondant, mettez-le sous forme canonique et linéarisez-le. On supposera que l'on se trouve au niveau (*level*) 2, que la fonction `f` est définie au niveau 3, que la variable `x` est définie au niveau 2 et à l'offset 2, que la variable `y` est définie au niveau 2 et à l'offset 3, que la variable `b` est définie au niveau 2 et à l'offset 4, et que la variable `z` au niveau 1 et à l'offset 2.

```
x := 2 + 3 * f(5)
b := (x >= (y + 1))
2 * x + z
while(x < 3) x := x - 1
```

3. On suppose maintenant que l'on se trouve au niveau 4, que la variable `x` est au niveau 4 offset 2, et que la fonction `f` a été définie au niveau 2. Construisez le code intermédiaire correspondant à l'expression

```
x := f(3, 5)
```

4. Construisez le code intermédiaire correspondant à la définition de fonction

```
let f(x : int, y : int) = x * y + 1
```

5. Construisez le code intermédiaire correspondant à la définition de fonctions

```
let pair(x:int) = if x = 0 then 1 else impair(x-1)
and impair(x:int) = if x = 0 then 0 else pair(x-1)
in pair(3)
```

6. Le code intermédiaire sera généré par une fonction

```
translate :
  Frame.frame Symbol.table ->
  int Symbol.table ->
  int ->
  Absyntax.exp ->
  Frame.frag list
```

où le premier paramètre est une table associant à chaque nom de fonction une structure de données contenant en particulier le nom de l'étiquette associée, et où le resultat final est une liste de fragments contenant chacun un `Ir.stm`. Expliquez comment le code de `pair` et `impair` sera généré.

7. On suppose maintenant qu'on se trouve au niveau 2, que la variable `a` est de type `arraint = array of int` et est définie au niveau courant, offset 2. Construisez le code intermédiaire correspondant à l'expression

```
a := arrint [10] of 0;
printint(a[5])
```