Examen

16/05/2013

Durée: 3h. Tous documents autorisés.

Exercice 1 (Thème). Compiler en bytecode OCaml les expressions suivantes[1]:

1. `(2 + 2) = (1 + 3)`
2. `1 :: 2 :: []`
3. `let f x = x * x in f 2`
4. `let x = (true, false) in snd x && fst x`
   (*indice*: `fst` et `snd` sont traduits directement en primitives sur les blocs, et la conjonction en un saut conditionnel)
5. `match [1;2] with [] -> 0 | x :: xs -> x`
6. `match N (N (L, L), L) with L -> L | N(L, y) -> y | N(x, y) -> x`
   (avec la définition `type tree = N of tree * tree | L`)
7. `let x = {contents = None} in x.contents <- Some (); x`
   (on rappelle la définition `type 'a ref = {mutable contents: 'a}`)
8. `let x = 2+2 in fun y -> x :: y`

Exercice 2 (Version). Soit le bytecode OCaml suivant:

```
        closurerec 1, 0
        const 10
        push
        acc 1
        appterm 1, 3
L1:     acc 0
        push
        const 1
        eqint
        branchif L3
        acc 0
        push
        const 2
        eqint
        branchifnot L2
L3:     const 1
        return 1
L2:     acc 0
        offsetint -2
        push
ICI:    offsetclosure 0
        apply 1
        push
        acc 1
        offsetint -1
        push
        offsetclosure 0
        apply 1
        addint
        return 1
```

1. On exécute ce bytecode jusqu'au premier passage par l'instruction d'étiquette ICI.
   Décrire alors l'état de la machine virtuelle OCaml après cette instruction: contenu de

2. Retrouver une expression OCaml dont la compilation produit ce bytecode. Associer à chaque sous-expression la partie de bytecode qui correspond.
3. Quelle est la valeur de cette expression?

Exercice 3 (Bytecode Java)**.** Étant donné le bytecode Java suivant de la fonction `func`[2]:

```
public static int[] func(int[]);
  Code:
   0:   aload_0
   1:   arraylength
   2:   newarray int
   4:   astore_1
   5:   iconst_0
   6:   istore_2
   7:   iload_2
   8:   aload_0
   9:   arraylength
   10:  if_icmpge       33
   13:  aload_1
   14:  iload_2
   15:  aload_0
   16:  iload_2
   17:  iaload
   18:  aload_1
   19:  iload_2
   20:  iconst_1
   21:  isub
   22:  iaload
   23:  iadd
   24:  iconst_2
   25:  imul
   26:  iastore
   27:  iinc    2, 1
   30:  goto    7
   33:  aload_1
   34:  areturn
```

1. On appelle la fonction `func` avec comme argument `{1, 2, 3}`. Décrire l'état de la machine après l'exécution des instructions 4, 8, et après le premier passage à l'instruction 20.
2. Retrouver la fonction Java dont la compilation produit ce bytecode.
3. On échange les instructions 13 et 14, et on assemble le résultat en un fichier `prog.class`. Que produit l'exécution de `java prog`? Décrire précisément où se situe alors le problème.

[redacted]

1. Soit e l'expression $((2 + 3) = 5) \wedge ((8 + 1) = 9)$. Dessiner son arbre de syntaxe abstraite.
2. Proposer une liste d'instructions permettant de calculer e. Associer à chaque sous-