Université Paris 7 - L3 Informatique Année 2008–2009

Introduction aux machines virtuelles

Examen du 25 juin 2009

Durée : 3h. Documents autorisés

I) Caml (∼6 points)

Exercice 1 Traduire en bytecode Caml les phrases suivantes :

- `8*6/4`
- `let double x = x+x in double 6`
- `let a = 10 in let decuple x = a * x in decuple 5`
- `let a = (1,2) in fst a + snd a`
  Indice : `fst` et `snd` sont traduit directement en primitives sur les blocs.
  Citer une autre phrase Caml qui pourrait donner le même bytecode.

Exercice 2 Étant donné la portion de bytecode Caml suivant :

```
        closurerec 1, 0
        const [0: 10 [0: 20 0a]]
        push
        acc 1
        apply 1
        push
        const 0
        addint
        return 2
L1:     acc 0
        branchifnot L2
        acc 0
        getfield 1
        push
        offsetclosure 0
        apply 1
        push
        const 1
        addint
        return 1
L2:     const 0
        return 1
```

- On exécute ce bytecode jusqu'au premier passage par l'instruction `offsetclosure`. Décrire alors l'état de la machine virtuelle Caml après cette instruction : contenu de l'accumulateur, de la pile et du tas. Justifier brièvement.
- Retrouver une phrase Caml dont la compilation produit ce bytecode. Associer à chaque élément de cette phrase la partie bytecode qui correspond. Quel est le résultat de l'exécution ?

II) Java (∼6 points)

Exercice 3 Traduire en bytecode Java les instructions suivantes (en sachant que `x` et `y` sont les variables locales 0 et 1).

- `y=4+3*(5+x));`
- `y=(x+5)*3+4;`
- `y=15; for (x=0; x*x<y; x++);`

Dans chacun des cas, simuler l'évolution de la pile lors de l'exécution (on suppose que `x` est nul initialement) et donner la taille de pile nécessaire pour le bon déroulement du calcul.

Exercice 4 Étant donné la portion de bytecode Java suivant :

```
.method public static f([I)I
        .limit locals 3
        .limit stack 3
        iconst_0
        istore_1
        iconst_0
        istore_2
L1:
        iload_2
        aload_0
        arraylength
        if_icmpge L2
        iload_1
        aload_0
        iload_2
        iaload
        iadd
        istore_1
        iinc 2 1
        goto L1
L2:
        iload_1
        ireturn
.end method
```

- Proposer une portion de code Java dont la compilation produit ce bytecode. Associer à chaque élément de code Java la partie bytecode qui correspond.
- Proposer une variante du bytecode ci-dessus permettant de minimiser l'usage de l'instruction `arraylength`.

III) Logo (∼8 points)

On souhaite créer une machine virtuelle pour un mini-Logo. Pour mémoire, un Logo est un langage permettant de réaliser des dessins en déplaçant un curseur nommé "tortue". Ce curseur possède en permanence une position exprimée par une paire de coordonnées entières, et une orientation exprimée par un angle entier en degré.

a) Le langage Logo

Voici les instructions de notre langage mini-Logo. Nous ne les utiliserons pas directement, elles ne sont décrites ici que pour aider à la compréhension des instructions de bytecode de la section suivante.

- `Avance n` : déplace la tortue de `n` dans sa direction courante, sans changer son orientation.
- `Recule n` : déplace la tortue de `n` dans la direction opposée à son orientation actuelle. L'orientation est préservée lors du déplacement.
- `Gauche n` : l'orientation est augmentée de `n` degrés.
- `Droite n` : l'orientation est diminuée de `n` degrés.
- `LeveCrayon` : suspend le tracé lors des déplacements à venir de la tortue.
- `BaisseCrayon` : enclenche le tracé lors des déplacements à venir de la tortue.
- `Repete n [ ... ]` : itère `n` fois les instructions présentes entre les crochets.
- `Pour p ... Fin` : associe à la chaîne de caractères `p` la procédure constituée de la suite d'instructions allant jusqu'au `Fin`
- `p` : si la chaîne de caractères `p` correspond à une procédure `Pour p ... Fin` , exécute les instructions correspondantes à cette procédure

Par exemple, un programme Logo dessinant un carré peut alors s'écrire :

```
Avance 10 Gauche 90 Avance 10 Gauche 90 Avance 10 Gauche 90 Avance 10 Gauche 90
```

ou aussi :

```
Repete 4 [ Avance 10 Gauche 90 ]
```

ou encore :

```
Pour CARRE
 Repete 4 [ Avance 10 Gauche 90 ]
Fin
CARRE
```

b) Un bytecode Logo

Au lieu de travailler avec les instructions "haut niveau" précédentes, on va utiliser une version "bytecode", dans laquelle chaque instruction correspond à un nombre entier ou deux, le premier nombre étant toujours le code de l'instruction :

- `STOP` (code=0) : arrête le programme.
- `AVANCE n` (code=1) : avance la tortue de `n`, cette quantité pouvant être négative.
- `TOURNE n` (code=2) : ajoute `n` degrés à l'orientation de la tortue (`n` peut être négatif).
- `LEVECRAYON` (code=3)
- `BAISSECRAYON` (code=4)
- `REPETE n` (code=5) : instruction marquant le début d'un bloc à répéter `n` fois
- `FINREPETE o` (code=6) : fin d'une zone à répéter. L'entier `o` est un décalage (offset) permettant de revenir juste après l'instruction `REPETE` correspondant (cf. exemple ci-dessous).
- `FINPOUR` (code=7) : instruction marquant la fin d'une procédure et le retour à l'endroit de l'appel.

- `LANCE o` (code=8) : lancement de la procédure situé `o` entiers plus loin dans le bytecode.

Ainsi les trois exemples précédents correspondent aux trois portions de bytecode suivants :

```
1 10 2 90 1 10 2 90 1 10 2 90 1 10 0
```

```
5 4 1 10 2 90 6 -6 0
```

```
8 1 0 5 4 1 10 2 90 6 -6 7
```

Exercice 5

1. On considère d'abord une Machine Virtuelle Logo (MVL) ne traitant que les instructions de bytecode 0 à 4. Quels sont les registres nécessaires au fonctionnement de cette MVL ? L'usage d'une pile est-il nécessaire ?
2. Que faut-il en plus pour obtenir une MVL traitant aussi les instructions 5 et 6 ? Votre solution permet-elle d'imbriquer des répétitions ? Comment traiter ce cas ?
3. Que faut-il en plus pour obtenir une MVL traitant aussi les instructions 7 et 8 ? Votre solution permet-elle d'imbriquer des appels de procédures dans d'autres procédures ? Comment traiter ce cas ?
4. L'argument `o` de l'instruction 6 (`FINREPETE`) est-il indispensable ? Si ce n'est pas le cas, que faut-il en plus comme structure pour s'en passer ?

Exercice 6 En utilisant le langage de votre choix (parmi Ocaml, Java et C), écrire une Machine Virtuelle Logo, c'est-à-dire une fonction recevant un tableau de nombres correspondant à du bytecode Logo et réalisant le dessin correspondant.

On pourra supposer l'existence d'une fonction `dessine_ligne` qui attend quatre entiers x_1 y_1 x_2 y_2 et trace le segment entre (x_1, y_1) et (x_2, y_2).

Indication : Dans un premier temps, vous pouvez vous concentrer sur la réalisation d'une machine virtuelle traitant seulement les instructions 0 à 4, puis 0 à 6. Évidemment, à vaincre sans péril, on triomphe sans gloire, et on a moins de points...