# Cours de Compilation-Exercices

# Génération de code

Master d'Informatique M1 2011–2012

17 novembre 2010

## 6 Compilation langages objets

### 6.1 Passage de paramètres en Java

Soit le programme Java suivant:

```
class A {
    int x;
    A (int v) { x=v;}
    static void m (A a) {a = new A(2);}
    public static void main (String argv[]) {
        A b = new A(0); m (b);
        System.out.println(b.x); }
}
```

1. Quel est le mode de passage de paramètres de Java ?
2. Quelle est le résultat de l'évaluation de ce programme ?
3. Justifier votre réponse en donnant les étapes de l'évaluation du programme (indiquer les variables allouées dans la pile et leur valeur en terme d'adresses des objets alloués dans le tas).
4. On considère maintenant le programme :

```
class B {
    A a;
    B (int v) { a=new A(0); }
    static void m (B b) {b.a = new A(2);}
    public static void main (String argv[]) {
        B c = new B(0); m (c);
        System.out.println(c.a.x); }
}
```

Donner le résultat de ce programme et expliquer les étapes de son évaluation.

### 6.2 Compilation de code Java

Soit le programme Java suivant:

```
class Vehicule
{ int pos;
    Vehicule () {pos = 0;}
    void move (int x) { pos = pos+x; }
}

class Voiture extends Vehicule
{ int places;
    Voiture () {places = 1;}
    void rejoint (Vehicule v)
    { if (v.pos < pos) v.move(pos-v.pos); else this.move(v.pos-pos);}
}

class Camion extends Vehicule
{int charge;
    Camion () {charge = 10; }
    void move (int x) { if (x<10) pos = pos+x; else pos = pos+10; }
}

class Main {
    static Camion c = new Camion ();
    static Voiture v = new Voiture ();
    static void main(String [] argv) {
        v.places = 4; v.move(30); c.move(10); v.rejoint(c);
        System.out.println(v.pos);
        System.out.println(c.pos); }
}
```

1. Quel est le résultat de ce programme ?
2. On décide de représenter le descripteur d'une classe $C$ avec un pointeur sur le descripteur de la classe que $C$ étend et les adresses des codes des méthodes non statiques appliquables aux objets de la classe $C$. Proposer un squelette de code MIPS contenant les déclarations des descripteurs de classe dans la zone statique.
3. Expliquer la représentation des objets pour chacune des classes non statiques. Proposer un code qui traduise l'effet des instructions `new` sur chacune de ces classes. On utilisera les instructions MIPS vues en cours, notamment l'appel `sbrk` (`syscall` avec $\$v0{=}9$) qui prend en argument une taille $n$ dans le registre $\$a0$, alloue un bloc de taille $n$ dans le tas et retourne l'adresse obtenue dans le registre $\$v0$.
4. Proposer un code pour la fonction `instanceof`: `instanceof(o,C)` renvoie `true` si et seulement si l'objet `o` appartient à une sous-classe de `C`. Pour cela on précisera l'algorithme utilisé ainsi que le tableau d'activation de la fonction `instanceof`.
5. On veut maintenant produire du code pour les méthodes dynamiques du programme Java. Proposer un format de tableau d'activation simple exploitant le fait que ces méthodes n'ont jamais plus d'un argument, et donner le code MIPS pour chaque méthode.
6. Donner le codage des variables et méthodes statiques de la classe `Main`.
7. Préciser l'organisation de la table des symboles qui vous sera utile pour engendrer ce code de manière générale. Donner le schéma général de compilation pour les instructions d'affectation à un champ d'un objet : $e_1.c = e_2$, l'appel de méthode $e.m(e_1, \ldots, e_n)$.

8. Dans cette question, nous nous intéressons a l'optimisation d'un appel de méthode. Notons que même dans le cas d'un langage avec héritage simple, un appel de méthode est plus coûteux qu'un appel de fonction. Il est nécessaire de récupérer l'adresse du descripteur de classe de l'objet puis l'adresse de la méthode dans cette classe avant d'effectuer l'appel.

   Pour un appel de méthode *o.m()* ou $o$ est de classe $C$, l'analyse de la hiérarchie des classes (donnée par la relation d'héritage) peut être utilisée pour déterminer les sous-classes de $C$ redéfinissant $C\_m$. S'il n'existe aucune redéfinition de $C\_m$ alors l'instance de la méthode est nécessairement $C\_m$. Donner le code de la compilation de l'appel de méthode dans ce cas.