Examen réparti d'ILP

Christian Queinnec

7 novembre 2013

Conditions générales

Cet examen est formé d'un unique problème en plusieurs questions auxquelles vous pouvez répondre dans l'ordre qui vous plait.

Le barème est fixé à 20 ; la durée de l'épreuve est de 3 heures. Tous les documents sont autorisés et notamment ceux du cours.

Votre copie sera formée de fichiers textuels que vous laisserez aux endroits spécifiés dans votre espace de travail pour Eclipse. L'espace de travail pour Eclipse sera obligatoirement nommé `workspace` et devra être un sous-répertoire direct de votre répertoire personnel.

À l'exception des clés USB en lecture seule, tous les appareils électroniques communiquants sont prohibés (et donc notamment les téléphones portables). Vos oreilles ne doivent pas être reliées à ces appareils.

L'examen sera corrigé à la main, il est donc absolument inutile de s'acharner sur un problème de compilation ou sur des méthodes à contenu informatif faible. Il est beaucoup plus important de rendre aisé, voire plaisant, le travail du correcteur et de lui indiquer, par tout moyen à votre convenance, de manière claire, compréhensible et terminologiquement précise, comment vous surmontez cette épreuve. À ce sujet, vos fichiers n'auront que des lignes de moins de 80 caractères, n'utiliseront que le codage ASCII ou UTF-8 enfin, s'abstiendront de tout caractère de tabulation.

Le langage à étendre est ILP3. Le paquetage Java correspondant sera nommé `fr.upmc.ilp.ilp3mem`. Sera ramassé, à partir de votre *workspace* (situé sous ce nom directement dans votre HOME), le seul répertoire `ILP/Java/src/fr/upmc/ilp/ilp3mem/` et tout ce qu'il contient.

Pour vous éviter de la taper à nouveau, voici l'url du site du master (mais il faut, pour y accéder, ne plus passer par le proxy) et celle de l'ARI où se trouvent de nombreuses documentations dont celle de Java :

```
http://www-master.ufr-info-p6.jussieu.fr/site-annuel-courant/
http://www-ari.ufr-info-p6.jussieu.fr/
```

Introduction

On souhaite d'abord être capable de mesurer la consommation en mémoire des programmes ILP3 compilés en C pour ensuite améliorer l'implantation des constantes.

Dans un premier temps, on supposera que la bibliothèque d'exécution en C maintient le compte d'octets qu'alloue la fonction `ILP_malloc`. La fonction `memoryGet` permet alors de lire ce compte tandis que la fonction `memoryReset` remet à zéro ce compte. Ainsi

```
{ m0 = memoryGet();
  1 + 2;
  print(memoryGet() - m0); // imprime 64 (sur certaines machines)
  memoryReset();
  print(memoryGet());      // imprime 0
  print(memoryGet());      // imprime 16 (sur certaines machines)
}
```

Question 1 – Tests (4 points)

Écrire, en pseudo-code lisible, une fonction unaire, que l'on nommera `alloue`, prenant un entier naturel en argument et allouant au moins autant d'octets. Cette fonction pourra être utilisée dans les programmes de tests que vous aurez aussi à écrire (toujours en pseudo-code) et qui éprouveront les fonctions `memoryGet` et `memoryReset`.

On n'oubliera pas d'expliciter ce que souhaitent vérifier ces programmes afin de faciliter la lecture et la compréhension de ces tests.

Livraison

- le fichier `workspace/ILP/Java/src/fr/upmc/ilp/ilp3mem/tests.txt` qui contiendra la fonction `alloue` suivie des tests des fonctions `memoryGet` et `memoryReset`.

Question 2 – Bibliothèque d'exécution (6 points)

Écrire, en C, ce qu'il faut ajouter ou modifer à la bibliothèque d'exécution pour implanter les fonctions `memoryGet` et `memoryReset`.

Livraison

- le fichier `workspace/ILP/Java/src/fr/upmc/ilp/ilp3mem/CEASTprogram.java`

Question 3 – Optimisation (8 points)

La compilation actuelle d'une constante numérique fait qu'elle est à nouveau allouée chaque fois que mentionnée. Ainsi, dans la boucle :

```
{ int n = 0;
  while (n < 10) {
    n = n + 1;
  }
  print(memoryGet());  // imprime 512
}
```

Alors que seulement 10 additions sont effectuées, 32 allocations d'entiers sont en fait réalisées ! Expliquez pourquoi et proposez de nouveaux schémas de compilation permettant de n'allouer que seulement 14 nombres (les constantes 0, 10 et 1, dix résultats d'addition et l'entier rendu par la fonction `memoryGet`).

Vous placerez vos réflexions, les nouveaux schémas de compilation et toute explication supplémentaire dans le fichier `strategie.txt` indiqué ci-dessous.

Livraison

- le fichier `workspace/ILP/Java/src/fr/upmc/ilp/ilp3mem/strategie.txt`

Question 4 – Optimisation suite (2 points)

Une autre technique complémentaire consiste à préallouer les nombres les plus courants. Parmi les nombres les plus courants, on trouve les nombres entre 0 et 15, les puissances de deux, les puissances de deux moins un. Ici, afin de ne pas perturber les tests précédents, on supposera que les seuls nombres courants sont 42 et 43. Proposez une modification de la bibliothèque d'exécution afin que ces deux nombres soient préalloués.

Livraison

- le fichier `workspace/ILP/Java/src/fr/upmc/ilp/ilp3mem/CEASTprogram.java`