IFT 1227 automne 2005
Architecture des ordinateurs I

Jean Pierre DAVID

Devoir 1

Utilisation des outils Mic-1 (voir site du cours IFT1227)

1) Soient les nouvelles instructions (en bytecode IJVM) d'accès à la mémoire suivantes:

MREAD i (0x21, i) : Fait une lecture de la mémoire à l'adresse contenue dans la variable locale numéro i et dépose la valeur sur la pile.

MWRITE i (0x22, i) : Retire la valeur qui se trouve au sommet de la pile et l'écrit dans la mémoire à l'adresse contenue dans la variable locale numéro i.

Implémentez le micro-code de ces deux instructions et modifiez les fichiers ***mic1ijvm.mal*** et ***ijvm.conf*** en conséquence.

2) Soit l'algorithme de tri quickSort suivant [1]:

```
public void quickSort(int[] a, int left, int right) {
   if (right > left) {
      int pivotNewIndex = partition(a, left, right);
      quicksort(a, left, pivotNewIndex-1);
      quicksort(a, pivotNewIndex+1, right);
   }
}

public int partition(int[] a, int left, int right) {
   int pivotValue = a[right];
   int storeIndex = left;
   for (int i=left;i<right;i++) {
      if (a[i] <= pivotValue) {
         swap(a, storeIndex, i);
         storeIndex = storeIndex + 1;
      }
   }
   swap(a, right, storeIndex);
   return storeIndex;
}

public void swap(int[] a, int index1, int index2) {
   /* Cette méthode échange les deux données du tableau a situées aux
      positions index1 et index2. Elle doit être implémentée
      directement en bytecode en utilisant le point 1) */}
```

[1] Cette implémentation est inspirée de http://en.wikipedia.org/wiki/Quicksort

Écrivez le bytecode IJVM de chacune des trois méthodes sans faire aucune modification à l'implémentation de l'algorithme. Pour vérifier votre code, utilisez le squelette « quickSort.jas ». Écrivez enfin le nombre total de cycles d'exécution de votre algorithme (sans compter la méthode de test).

3) En modifiant le microcode (fichiers *mic1ijvm.mal*. et *ijvm.conf*), optimisez l'algorithme pour obtenir une réduction sensible du temps d'exécution. Écrivez le nombre total de cycles d'exécution de votre algorithme.

Bon travail

Critères d'évaluation :

- Pour chacun des programmes demandés :
  - Il fonctionne : oui – le plus souvent – rarement ou pas du tout.
  - Le code est bien structuré et bien documenté : oui – quelques lacunes – non.
- Pour le point 3), la réduction est de : plus de 50% - entre 50% et 25% - moins de 25%

Consignes :

- Remettez une seule version des fichiers ***mic1ijvm.mal*** et ***ijvm.conf*** sans modifier leurs noms.
- Pour les points 2) et 3), nommez vos fichiers ***QuickSort2.jas*** *et* ***QuickSort3.jas***.
- Vérifiez toutes vos simulations sous LINUX au DIRO avant d'envoyer vos fichiers. Si un programme ne compile pas, il ne sera même pas regardé.
- Chaque fichier doit commencer par un en-tête reprenant les noms et codes permanents des deux auteurs.
- Les travaux se font obligatoirement par équipes de deux. En cas d'empêchement majeur, prendre rendez-vous **immédiatement** pour expliquer la situation. N'avoir trouvé personne n'est pas un empêchement majeur. Toutes les personnes qui craignent de se retrouver dans cette situation doivent contacter le démonstrateur suffisamment tôt pour qu'il puisse lui-même constituer des équipes avec les personnes concernées.
- **Chaque auteur doit** comprendre l'entièreté du travail et en **remettre sa propre copie** de manière électronique. Ce sera la preuve qu'il assume la responsabilité du contenu du travail. Une pénalité majeure sera appliquée si les deux copies remises sont différentes.
- Si dans une équipe, une personne se retrouve forcée à réaliser la majeure partie du travail (parce que l'autre ne fait rien), elle doit en faire part avant la correction et la participation des deux personnes sera vérifiée.
- Une équipe sera de toutes façons tirée au sort pour vérifier que chacun des auteurs maîtrise le travail.
- Aucune source d'information extérieure n'est justifiable pour ce travail. Tout cas de plagiat détecté sera rapporté à la faculté.