ECOLE POLYTECHNIQUE DE MONTREAL

Département de génie informatique et génie logiciel

Cours INF8601: Systèmes informatiques parallèles (Hiver 2013)
3 crédits (3-1.5-4.5)

EXAMEN FINAL

DATE: Mercredi le 5 décembre 2012

HEURE: 13h30 à 16h00

DUREE: 2H30

NOTE: Toute documentation permise, calculatrice non programmable permise

Ce questionnaire comprend 4 questions pour 20 points

Question 1 (5 points)

a) Un programme effectue pour chaque point la moyenne avec ses deux voisins. Convertissez ce programme en code efficace écrit en assembleur Vectoriel MIPS. Tous les éléments de a (a[0] à a[n + 1]) sont accédés mais seulement les éléments b[1] à b[n] sont écrits. **(3 points)**

```
double a[n + 2], b[n + 2];

for(i = 1 ; i <= n ; i++) b[i] = (a[i] + a[i + 1] + a[i - 1]) / 3;
```

b) Une unité centrale de traitement vectorielle est telle que décrite dans le livre et contient une de chacune des unités suivantes: addition et soustraction point flottant, multiplication point flottant, division point flottant, opérations entières, opérations logiques, et rangement et chargement. Ces unités en pipeline permettent de produire une nouvelle valeur à chaque cycle. Les instructions scalaires (e.g. L.D) prennent un cycle d'exécution chacune. Dans ces unités en pipeline, les additions et soustractions ont une profondeur de pipeline de 8, les multiplications de 10, les divisions de 20 et les chargements et rangements de 12. L'instruction de comparaison (SGTV) est traitée par l'unité d'addition et soustraction point flottant, et prend le même nombre de cycles qu'une addition ou soustraction. Calculez le temps requis pour l'exécution de chacune des deux séquences d'instructions suivantes. **(2 points)**

```
/* Version inconditionnelle */
L.D      F1, #10
L.D      F2, #2
LV       V1, R1
MULVS.D  V1, V1, F2
SV       V1, R1

/* Version conditionnelle */
L.D      F1, #10
L.D      F2, #2
LV       V1, R1
SGTVS.D  V1, F1
MULVS.D  V1, V1, F2
CVM
SV       V1, R1
```

Question 2 (5 points)

a) Il faut calculer une version réduite d'une image. Pour ce faire, la moyenne d'une région de 16x16 pixels est calculée et devient le pixel correspondant de l'image réduite. Le

code en C pour effectuer ce travail est fourni. Proposez une implémentation efficace en OpenCL qui effectue ce travail. Fournissez le code pour la fonction de type kernel correspondante. Expliquez par ailleurs les arguments fournis pour l'appel de cette fonction et donnez l'énoncé clEnqueueNDRangeKernel avec ses arguments qui serait appelé pour exécuter votre fonction. **(3 points)**

```
int i, j;
float image_in[1024][1024];
float image_out[64][64]

for(i = 0; i < 1024; i++) {
  for(j = 0; j < 1024; j++) {
    image_out[i/16][j/16] += image_in[i][j] / 256;
  }
}
```

b) Dans la mesure où les processeurs sur les GPGPU sont de type SIMD, qu'arrive-t-il lorsque le code OpenCL d'une fonction de type kernel, exécutée sur ces processeurs, contient des conditions (if then else) de sorte que le traitement requis peut différer d'un élément à l'autre du groupe de traitement (work group). Comment cela est-il exécuté par le processeur SIMD? Le temps de traitement est-il le plus court entre la branche if et else, le plus long, ou la somme des deux? **(1 point)**

c) En OpenCL, est-il possible d'appeler la fonction malloc (ou new en C++) à l'intérieur d'une fonction de type kernel? Pourquoi? **(1 point)**

Question 3 (5 points)

a) Dans un cours de sécurité informatique, le professeur a donné un énoncé de devoir encrypté. Heureusement, il a fourni la fonction de décryption (sans la clé) et spécifié que la clé de décryption est un entier de 64 bits dont seulement les 6 octets les moins significatifs sont utilisés. Il faut donc essayer toutes les valeurs qu'il est possible de représenter avec 6 octets. Voici le programme sériel pour ce faire mais qui prendrait plus de temps que l'échéance pour la remise du devoir. Convertissez ce programme en programme MPI efficace. Les ordinateurs utilisés ont des entiers (int) de 64 bits et la taille du message chiffré est connue statiquement (LEN). Pour simplifier le programme MPI, il suffit d'imprimer le résultat trouvé sur le noeud qui trouve la bonne clé, comme dans le programme sériel. **(3 points)**

```
int key, end;
char message_chiffre[LEN];
char *message_clair;
```

```
end = 1 << (6 * 8);
for(key = 0 ; key < end; key++) {
  decrypt(message_chiffre, LEN, key, message_clair);
  if(strncmp("Devoir 1", message_clair, 8) == 0)
      printf("Devoir decode:\n%s", message_clair);
}
```

b) Dans le cadre du travail pratique 3, le nombre des instances du programme MPI était augmenté de quelques instances jusqu'au nombre total de coeurs par ordinateur multiplié par le nombre d'ordinateurs dans la grappe. Est-ce que le délai de communication varie beaucoup entre le cas de deux instances sur le même ordinateur versus deux instances sur deux ordinateurs différents? Le délai de communication signifie-t-il nécessairement un délai équivalent perdu pour le programme parallèle? **(1 point)**

c) La fonction suivante, rencontrée dans le cadre du travail pratique 2 contient un problème de performance sérieux. Lequel? Suggérez un correctif. **(1 point)**

```
int encode_slow_c(struct chunk *chunk)
{ int i, j, checksum = 0;
  int width = chunk->width, height = chunk->height;
  int key = chunk->key;
  char *data = chunk->data;

  #pragma omp parallel for private(i,j) reduction(+:checksum)
  for (i = 0; i < width; i++) {
    for (j = 0; j < height; j++) {
      int index = i + j * width;
      data[index] = data[index] + key;
      checksum += data[index];
    }
  }
  chunk->checksum = checksum;
  return 0;
}
```

Question 4 (5 points)

a) Les plus récents processeurs Intel bénéficient du support pour les tables de pages étendues (EPT) afin de mieux supporter la virtualisation. Comment cela fonctionne-t-il? Expliquez quelles opérations sur une machine virtuelle seront plus rapides ou plus lentes avec EPT? **(2 points)**

b) Un GPGPU récent de NVidia, le GTX580, contient 16 processeurs SIMD de 32 éléments de traitement (avec un ALU) chacun, pour un total de 512 éléments de traitement. Par comparaison, le Intel Xeon Phi qui vient d'être annoncé contient 60 processeurs avec vecteur. Sur les forums, certains amateurs prétendent qu'avec 60 processeurs plutôt que 512, le Xeon Phi n'est pas compétitif du tout. Qu'en est-il? Comment peut-on comparer leur performance pour des tâches de calcul scientifique parallèle?

(1 point)

c) L'entreprise pour laquelle vous travaillez aimerait pouvoir tirer profit des cycles inutilisés de tous les ordinateurs de bureau des employés, particulièrement lorsque ceux-ci sont absents, en réunion ou le soir. Quel logiciel pourrait faire ce travail? Est-ce que ceci constitue une grille ou un nuage? **(1 point)**

d) Dans quelles circonstances utiliseriez-vous OpenMP ou OpenCL? Donnez un exemple. **(1 point)**

Le professeur: Michel Dagenais