ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Ακ. έτος 2007-2008, 8ο εξάμηνο, Σχολή ΗΜ&ΜΥ

4η ΕΡΓΑΣΙΑ

Τελική Ημερομηνία Παράδοσης: 29-6-2008 *(δεν θα δοθεί παράταση)*

Cache optimizations

Αντικείμενο της άσκησης αυτής είναι η μελέτη της επίδρασης που έχουν στην απόδοση των προγραμμάτων διάφορες τεχνικές βελτιστοποίησης κώδικα που στοχεύουν στην αξιοποίηση της cache.

Για τους σκοπούς της άσκησης θα χρησιμοποιήσετε τον προσομοιωτή SESC με τον ίδιο τρόπο όπως και στις 2 πρώτες ασκήσεις. Ο κώδικας που θα αξιολογήσετε είναι ο πολλαπλασιασμός δύο τετραγωνικών πινάκων A και B, τις διάφορες εκδόσεις του οποίου θα πρέπει να γράψετε σε γλώσσα C, και να τις μεταγλωττίσετε ώστε να μπορούν να προσομοιωθούν στον SESC. Οδηγίες για το πώς γίνεται αυτή η διαδικασία του “cross compilation” μπορείτε να βρείτε εδώ: http://www.cslab.ntua.gr/courses/advcomparch/sesc_cross_compile.go .

Η L1 Data cache και η L2 cache του συστήματος, για όλες τις διαφορετικές εκδόσεις που θα αξιολογήσετε, θα πρέπει να είναι παραμετροποιημένες ως εξής (οι υπόλοιπες παράμετροι είναι οι default):

```
[DataL1]
...
size = 32*1024
assoc =  4
bsize = 64
...

[L2Cache]
...
size = 256*1024
assoc =  8
bsize = 64
...
```

Όσον αφορά τον αλγόριθμο, θεωρείστε για όλες τις εκδόσεις που θα δοκιμάσετε ότι οι πίνακες έχουν διάσταση 256x256 και ότι περιέχουν στοιχεία κινητής υποδιαστολής μονής ακρίβειας (τύπου float). Επιπλέον, σε όλες τις εκδόσεις φροντίστε να αρχικοποιήσετε τους πίνακες A,B και C στις ίδιες τιμές (με όποιον τρόπο θέλετε αρκεί οι τιμές να είναι μη μηδενικές), ώστε να μπορείτε να ελέγχετε την ορθότητα της κάθε έκδοσης σε σχέση με την αρχική. Τέλος, όλες οι διαφορετικές εκδόσεις του αλγορίθμου θα πρέπει να μεταγλωττιστούν με optimizations επιπέδου 2 στον gcc (χρησιμοποιώντας το flag -O2).

1. Αρχική έκδοση

Αρχικά, υλοποιείστε μια απλοϊκή, μη-βελτιστοποιημένη έκδοση του αλγορίθμου όπως αυτή που παρουσιάζεται στη συνέχεια:

```
for(i=0; i<N; i++)
      for(j=0; j<N; j++)
            for(k=0; k<N; k++)
                  C[i][j] += A[i][k]*B[k][j];
```

Προκειμένου να περιορίσετε την (αναλυτική) προσομοίωση μόνο στο τμήμα του προγράμματος που εκτελεί τον αλγόριθμο, εξαιρώντας π.χ. τις όποιες αρχικοποιήσεις που γίνονται στους πίνακες και οι οποίες θα οδηγήσουν πιθανώς σε αλλοίωση των αποτελεσμάτων, χρησιμοποιήστε simulation marks για να περιβάλετε και να απομονώσετε το τμήμα αυτό. Αναλυτικές οδηγίες για τη χρήση simulation marks υπάρχουν εδώ: http://www.cslab.ntua.gr/courses/advcomparch/sesc_simulation_marks.go .

1.1. Προσομοιώστε την παραπάνω απλοϊκή έκδοση του αλγορίθμου. Καταγράψτε τον απαιτούμενο χρόνο εκτέλεσης (αριθμός κύκλων), καθώς και τα miss rates στην L1 data cache και στην L2 cache.

2. Loop interchange

Ο αρχικός αλγόριθμος δεν είναι βελτιστοποιημένος ως προς την χωρική τοπικότητα των αναφορών, και συνεπώς δεν κάνει την καλύτερη δυνατή αξιοποίηση της cache. Για αυτό το λόγο, καλείστε να εφαρμόσετε την τεχνική της αναδιάταξης βρόχων προκειμένου να πετύχετε καλύτερη τοπικότητα στην cache, η οποία ευελπιστείτε να οδηγήσει τελικά και σε καλύτερους χρόνους εκτέλεσης.

2.1. Δοκιμάστε όλες τις διαφορετικές αναδιατάξεις που μπορείτε να κάνετε στον αρχικό κώδικα. Καταγράψτε τη συμπεριφορά της κάθε μιας, ως προς το χρόνο εκτέλεσης και τα miss rates στις L1D και L2 caches.

2.2. Υπάρχει κάποια συσχέτιση ανάμεσα στους παρατηρούμενους χρόνους εκτέλεσης για τις διάφορες εκδόσεις και τα αντίστοιχα miss rates που μετρήσατε; Σχολιάστε σχετικά.

2.3. Πώς αλλάζει η κάθε αναδιάταξη την απόδοση του απλοϊκού αλγορίθμου; Πώς εξηγείται αυτό σε σχέση με τα διαφορετικά access patterns που συνεπάγεται η κάθε αναδιάταξη, και την τοπικότητα που επιτυγχάνει θεωρητικά το κάθε access pattern;

2.4. Τι speedup δίνει η καλύτερη αναδιάταξη σε σχέση με την απλοϊκή έκδοση;

3. Cache blocking

Βασικός στόχος της τεχνικής αυτής είναι η βελτίωση της χρονικής τοπικότητας των αναφορών. Η γενική ιδέα του cache blocking έγκειται στον διαχωρισμό του χώρου επαναλήψεων ενός loop (loop iteration space) σε μικρότερους υποχώρους, έτσι ώστε το σύνολο δεδομένων (working set) που επεξεργάζεται ο κάθε υποχώρος να χωρά σε κάποιο επίπεδο κρυφής μνήμης, και να μπορεί συνεπώς να επαναχρησιμοποιηθεί εκεί στο μέγιστο δυνατό βαθμό, προτού εκτοπιστεί.

Στην περίπτωση που έχουμε να κάνουμε με πίνακες, ο διαχωρισμός του χώρου επαναλήψεων ενός ή περισσότερων loops οδηγεί στο διαχωρισμό των πινάκων σε μικρότερους υποπίνακες (ή blocks). Έτσι μπορούμε να εφαρμόσουμε τον αρχικό αλγόριθμο διαδοχικά πάνω στους επιμέρους υποπίνακες, και αν αυτοί συναθροιστικά χωράνε σε κάποιο επίπεδο κρυφής μνήμης, τότε μπορούμε να επιτύχουμε υψηλότερα επίπεδα επαναχρησιμοποίησης των στοιχείων τους και των στοιχείων του προβλήματος συνολικά.

3.1. Σε αυτό το ερώτημα καλείστε να υλοποιήσετε μια έκδοση του πολλαπλασιασμού πινάκων χρησιμοποιώντας την τεχνική του blocking. Συγκεκριμένα, χρησιμοποιήστε την έκδοση από το

ερώτημα 2 που σας έδωσε τα καλύτερα αποτελέσματα, και εφαρμόστε σε αυτήν blocking ώστε ο πολλαπλασιασμός να πραγματοποιείται πάνω σε υποπίνακες των A, B, C. Δοκιμάστε μεγέθη τετραγωνικών blocks από 8 έως 160 με βήμα 8. Παρουσιάστε σε διαγράμματα τη μεταβολή του χρόνου εκτέλεσης, καθώς και των miss rates για τις L1D και L2 caches, σε σχέση με το μέγεθος του block.

3.2. Ποιο είναι το overhead της εφαρμογής του blocking; Παρουσιάστε σε ένα διάγραμμα, για όλα τα μεγέθη blocks, τη μεταβολή του συνολικού αριθμού εντολών της blocked έκδοσης σε σχέση με τον αριθμό εντολών της αρχικής interchanged.

3.3. Πώς εξηγείται η μεταβολή των miss rates καθώς μεταβάλλεται το μέγεθος του block, για τα δεδομένα μεγέθη των L1D και L2 caches που έχουμε θεωρήσει; Αποτυπώνεται σε μεταβολή στον χρόνο εκτέλεσης, και αν ναι πώς;

3.4. Τι speedup δίνει η καλύτερη cache-blocked έκδοση σε σχέση με την απλοϊκή έκδοση;

4. Register blocking (“Unroll & jam”)

Όπως μπορούμε να χρησιμοποιήσουμε την τεχνική του blocking για να πετύχουμε μεγαλύτερα ποσοστά επαναχρησιμοποίησης σε κάποιο επίπεδο κρυφής μνήμης, μειώνοντας έτσι τα misses και την επικοινωνία με το αμέσως επόμενο επίπεδο στην ιεραρχία, με τον ίδιο ακριβώς τρόπο μπορούμε να εφαρμόσουμε την τεχνική του blocking και στους registers, προκειμένου να γλιτώσουμε κάποιο μέρος της επικοινωνίας με την L1 data cache.

Εφαρμόζοντας register blocking με blocks διάστασης RxR, χρησιμοποιούμε στην ουσία R^2 floating point registers όπου θα αποθηκεύσουμε ισάριθμα στοιχεία κάποιου πίνακα, τα οποία θα επαναχρησιμοποιούνται στο εσωτερικότερο loop του κώδικα.
Ο πίνακας απ' όπου θα πάρουμε στοιχεία για να τα αποθηκεύσουμε στους registers θα πρέπει να είναι “loop-invariant” ως προς το εσωτερικότερο loop, δηλαδή θα πρέπει να επιλεγεί ώστε η μεταβλητή του εσωτερικότερου loop να μην είναι κάποια από τις 2 μεταβλητές που χρησιμοποιούνται για την αναφορά στα στοιχεία του. Έτσι, αμέσως πριν μπούμε στο εσωτερικότερο loop οι registers μπορούν να αρχικοποιηθούν με τα στοιχεία του, και μέσα στο εσωτερικότερο loop οι αρχικές αναφορές μνήμης στα στοιχεία αυτά μπορούν να αντικατασταθούν από αναφορές στους αντίστοιχους registers, γεγονός που μειώνει την επικοινωνία με την L1 cache.

Στις διαφάνειες του μαθήματος παρουσιάζεται αναλυτικότερα ο τρόπος με τον οποίον εφαρμόζεται register blocking στον αρχικό, απλοϊκό αλγόριθμο.

4.1. Στο ερώτημα αυτό καλείστε να χρησιμοποιήσετε την blocked έκδοση που υλοποιήσατε στο προηγούμενο ερώτημα, για να εφαρμόσετε πάνω σε αυτή register blocking με blocks διάστασης 2x2. Στο πρόγραμμά σας, χρησιμοποιήσετε τον specifier `register` της C για μια μεταβλητή, προκειμένου να υποδείξετε στον compiler ότι επιθυμείτε να αποθηκευτεί η μεταβλητή αυτή σε κάποιον register του επεξεργαστή[1].

Όπως και στο ερώτημα 3, για το πρώτο επίπεδο blocking (cache blocking) πειραματιστείτε κι εδώ με μεγέθη blocks από 8 έως 160 με βήμα 8 (η διάσταση των register blocks θα παραμένει σταθερή και ίση με 2x2). Παρουσιάστε σε διαγράμματα (τα οποία μπορείτε να ενσωματώσετε, για λόγους σύγκρισης, στα αντίστοιχα διαγράμματα του ερωτήματος 3.1) τη μεταβολή του χρόνου εκτέλεσης και των miss rates για τις L1D και L2 caches, σε σχέση με το μέγεθος του block. Σχολιάστε σύντομα πάνω στα αποτελέσματα.

4.2. Ποιο το συνολικό overhead της εφαρμογής του blocking σε αυτή την περίπτωση; Παρουσιάστε σε ένα διάγραμμα, για όλα τα μεγέθη blocks, τη μεταβολή του συνολικού αριθμού εντολών της register-blocked έκδοσης του παρόντος ερωτήματος σε σχέση με τον αριθμό εντολών της καλύτερης

1 Το αν γίνει αυτό τελικά ή όχι, εξαρτάται από τον compiler, και κατά κύριο λόγο από την ανάθεση μεταβλητών στους registers που έχει κάνει μέχρι εκείνη τη στιγμή.

interchanged έκδοσης του ερωτήματος 2.

4.3. Τι speedup δίνει η καλύτερη register-blocked έκδοση σε σχέση με την απλοϊκή έκδοση;

4.4. Από τους 3 πίνακες A,B,C, αυτός που ενδείκνυται πιο πολύ για επαναχρησιμοποίηση στα στοιχεία του είναι ο C, γιατί σε κάθε επανάληψη του εσωτερικού loop πραγματοποιούνται 2 προσπελάσεις στην cache για αυτόν (1 για ανάγνωση + 1 για εγγραφή), σε αντίθεση μιας προσπέλασης για τον A ή B. Σε περίπτωση που η register-blocked έκδοση που υλοποιήσατε στο ερώτημα 4.1 κάνει επαναχρησιμοποίηση σε κάποιον από τους A ή B, ξαναγράψτε την ώστε να κάνει επαναχρησιμοποίηση στα στοιχεία του C, ελπίζοντας ότι θα πετύχετε ακόμα μεγαλύτερη απόδοση. Σημειώστε ότι σε αυτή την περίπτωση θα χρειαστεί να αναδιατάξετε κάποιο από τα εσωτερικά loops.

Πειραματιστείτε με μεγέθη blocks από 8 έως 160 με βήμα 8 (η διάσταση των register blocks είναι η ίδια όπως και στο 4.1 και ίση με 2x2), και παρουσιάστε όπως και πριν σε διαγράμματα (τα οποία μπορείτε να ενσωματώσετε στα αντίστοιχα διαγράμματα του ερωτήματος 3.1) τη μεταβολή του χρόνου εκτέλεσης και των miss rates για τις L1D και L2 caches, σε σχέση με το μέγεθος του block. Σχολιάστε σύντομα πάνω στα αποτελέσματα.

4.5. Τι speedup δίνει σε αυτή την περίπτωση η καλύτερη register-blocked έκδοση σε σχέση με την απλοϊκή έκδοση;

5. Συνολική εκτίμηση

Για όλες τις παραπάνω διαδοχικές βελτιστοποιήσεις, κάντε μια ποσοτική εκτίμηση της συνεισφοράς κάθε τεχνικής στην τελική απόδοση του αλγορίθμου, καθώς και μια σύντομη ποιοτική εκτίμηση της συνεισφοράς τους συνοψίζοντας τα βασικότερα συμπεράσματα των προηγούμενων ερωτημάτων.

Παραδοτέο της άσκησης θα είναι ένα ηλεκτρονικό κείμενο (pdf, doc ή odt). Στο ηλεκτρονικό κείμενο να αναφέρετε στην αρχή τα στοιχεία σας (Όνομα, Επώνυμο, ΑΜ).

Η άσκηση θα παραδοθεί μόνο ηλεκτρονικά στην ιστοσελίδα:
http://www.cslab.ece.ntua.gr/courses/advcomparch/submit.

Δουλέψτε ατομικά. Έχει ιδιαίτερη αξία για την κατανόηση του μαθήματος να κάνετε μόνοι σας την εργασία. Μην προσπαθήσετε να την αντιγράψετε απλά από άλλους συμφοιτητές σας.

Μην αφήσετε την εργασία για το τελευταίο Σαββατοκύριακο, απαιτεί αρκετό χρόνο για την εκτέλεση των benchmarks, ξεκινήστε αμέσως!