1 εργαστηριακή άσκηση

Compilers

23/11/2012

Παπoυτσής Γεώργιος Μεταπτυχιακός ΟΣΥΛ 193

gpap

Σκοπός

Στην παρούσα εργασία επιχειρούμε να βελτιώσουμε το χρόνο εκτέλεσης γινομένου δύο πινάκων ,διαστάσεων 100\*100 και στοιχείων κινητής υποδιαστολής δεδομένης αρχιτεκτονικής υπολογιστή .

Ως μέτρο σύγκρισης θα πάρουμε την διάρκεια που έκανε να τρέξει το πρόγραμμα ,τους κύκλους ρολογιού ,τα misses στις κρυφές μνήμες εντολών και δεδομένων. Το πρόγραμμα είναι αρχικά γραμμένο απλά χωρίς ιδιαίτερες μετατροπές για να το έχουμε ως μέτρο σύγκρισης χειρότερης περίπτωσης.

Αρχιτεκτονική 1

Η αρχιτεκτονική αυτή ως προς τα σημεία ενδιαφέροντος περιλαμβάνει :

Μια κρυφή μνήμη δεδομένων πρώτου επιπέδου, την DL1 μεγέθους 1M ,πολιτικής αντικατάστασης (Last Release Use) LRU ,πολιτικής προσκόμισης 8-way associetivity και διάρκεια στο hit 1 κύκλο ρολογιού. Επίσης περιλαμβάνει κρυφή μνήμη εντολών πρώτου επιπέδου IL1 μεγέθους 64 Κ με πολιτική αντικατάστασης LRU και κρυφή μνήμη εντολών δευτέρου επιπέδου IL2 μεγέθους 512 Μ με πολιτική αντικατάστασης LRU και διάρκεια στο hit 6 κύκλων ρολογιού.

Κύρια μνήμη

Κρυφή Μνήμη DL1 = 1M

/usr/local/SimpleScalar/simplesim-3.0/sim-outorder -res:ialu 4 -res:imult 2 -issue:width 16 -decode:width 16 -fetch:ifqsize 16 -lsq:size 32 -ruu:size 32 -bpred perfect -cache:dl1lat 1 -cache:dl1 dl1:1024:128:8:l -cache:dl2 none -mem:width 128 -mem:lat 180 2 -cache:il1lat 1 -tlb:lat 30 -res:memport 4 -res:fpalu 2 -res:fpmult 2 -cache:il1 il1:256:64:4:l -cache:il2 il2:2048:64:8:l -commit:width 8 -cache:il2lat 6 ex

Αρχιτεκτονική 2

Η αρχιτεκτονική αυτή ως προς τα σημεία ενδιαφέροντος περιλαμβάνει :

Μια κρυφή μνήμη δεδομένων πρώτου επιπέδου, την DL1 μεγέθους 8 Κ ,πολιτικής αντικατάστασης (Last Release Use) LRU ,πολιτικής προσκόμισης 4-way associetivity και διάρκεια στο hit 1 κύκλο ρολογιού.Επίσης περιλαμβάνει κρυφή μνήμη δεδομένων δευτέρου επιπέδου DL2 μεγέθους 128 Κ με πολιτική προσκόμισης 8-way associetivity και με πολιτική αντικατάστασης LRU.Η αρχιτεκτονική αυτή περιλαμβάνει κρυφή μνήμη εντολών πρώτου επιπέδου IL1 μεγέθους 64 Κ με πολιτική αντικατάστασης LRU και διάρκεια στο hit 1 κύκλων ρολογιού. Ακόμη περιλαμβάνει κρυφή μνήμη εντολών δευτέρου επιπέδου IL2 μεγέθους 256 Μ με πολιτική αντικατάστασης LRU και διάρκεια στο hit 6 κύκλων ρολογιού.

Κύρια Μνήμη

Κρυφή Μνήμη DL2 = 128 K

Κρυφή Μνήμη DL1 = 8K

local/SimpleScalar/simplesim-3.0/sim-outorder -res:ialu 4 -res:imult 2 -issue:width 16 -decode:width 16 -fetch:ifqsize 16 -lsq:size 32 -ruu:size 32 -bpred perfect -cache:dl1lat 1 -cache:dl1 dl1:128:16:4:l -cache:dl2lat 7 -cache:dl2 dl2:512:32:8:l -mem:width 32 -mem:lat 180 2 -cache:il1lat 1 -tlb:lat 30 -res:memport 4 -res:fpalu 2 -res:fpmult 2 -cache:il1 il1:256:64:4:l -cache:il2 il2:2048:64:8:l -commit:width 8 -cache:il2lat 6 ex

Από τις παραπάνω αρχιτεκτονικές για την σύγκριση των αποτελεσμάτων θεωρούμε ότι όλα τα δεδομένα είναι στην κύρια μνήμη οπότε δεν μας ενδιαφέρει η καθυστέρησή της μιας και είναι πέρα από τα ζητούμενα αυτής της εργασίας.

Πειραματική Διαδικασία με βάση την απλή μορφή του προγράμματός μου

Για την αρχιτεκτονική 1 έχω :

Κύκλους ρολογιού 3594881

Χρόνος εκτέλεσης 15 sec

DL1 misses 1047

IL1 misses 212

IL2 misses 212

Για την αρχιτεκτονική 2 έχω :

Κύκλους ρολογιού 4063760

Χρόνος εκτέλεσης 16 sec

DL1 misses 263339

DL2 misses 4163

IL1 misses 212

IL2 misses 212

Πειραματική Διαδικασία με βάση την σύνθετη μορφή του προγράμματός μου

Για την αρχιτεκτονική 1 έχω :

Κύκλους ρολογιού 1775722

Χρόνος εκτέλεσης 6 sec

DL1 misses 948

IL1 misses 156

IL2 misses 156

Για να πετύχουμε τα παραπάνω αποτελέσματα τα οποία είναι καλύτερα από τα προηγούμενα της ίδιας αρχιτεκτονικής κάναμε unrolling κατά 4 στους πίνακες και μαζί με το unrolling φορτώσαμε σε κάθε βήμα 16 στοιχεία Cmatrix (πίνακας αποτελέσματος) σε 16 καταχωρητές .Αυτό έχει σαν αποτέλεσμα να μειώσουμε τα misses κατά 99 και τον χρόνο εκτέλεσης κατά 9 sec.

Για την αρχιτεκτονική 2 έχω :

Κύκλους ρολογιού 1873391

Χρόνος εκτέλεσης 6 sec

DL1 misses 73994

DL2 misses 4166

IL1 misses 236

IL2 misses 236

Mε τον ίδιο κώδικα έχουμε τα παραπάνω αποτελέσματα τα οποία είναι καλύτερα από τα προηγούμενα της ίδιας αρχιτεκτονικής. Σαν αποτέλεσμα έχουμε να μειώσουμε τα misses κατά 189345 στην κρυφή μνήμη του δεύτερου επιπέδου .Τα misses στο πρώτο επίπεδο δεν καταφέραμε να τα βελτιώσουμε καθώς το μεγεθός της είναι πολύ μικρό 8K πράγμα που δεν βοηθά στο να φορώσουμε αποτελεσματικά εκεί κάποιο πίνακα.Μείωση έχουμε επίσης και στον χρόνο εκτέλεσης κατά 9 sec.

Παράρτημα

Ο κώδικας της άσκησης

#include <stdio.h>

main(){

float Amatrix[100][100];

float Bmatrix[100][100];

float Cmatrix[100][100];

register float temp,temp1,temp2,temp3;

register float temp4,temp5,temp6,temp7,temp8,temp9,temp10,temp11,temp12;

register float temp13,temp14,temp15,r,r1,r2,r3,r4,r5,r6,r7,r8;

int i,ii,j,jj,k,kk=0;

for(i=0;i!=100;i++){

for(j=0;j!=100;j++){

Cmatrix[i][j]=0.0;

}

}

/\*testing code\*/

for(i=0;i!=100;i++){

for(j=0;j!=100;j++){

Amatrix[i][j]=1.0;

Bmatrix[i][j]=1.0;

}}

for(i=0;i!=100;i+=4){

for(j=0;j!=100;j+=4){

temp = Cmatrix[ii][jj];

temp1 = Cmatrix[ii][jj+1];

temp2 = Cmatrix[ii][jj+2];

temp3 = Cmatrix[ii][jj+3];

temp4 = Cmatrix[ii+1][jj];

temp5 = Cmatrix[ii+1][jj+1];

temp6 = Cmatrix[ii+1][jj+2];

temp7 = Cmatrix[ii+1][jj+3];

temp8 = Cmatrix[ii+2][jj];

temp9 = Cmatrix[ii+2][jj+1];

temp10 = Cmatrix[ii+2][jj+2];

temp11 = Cmatrix[ii+2][jj+3];

temp12 = Cmatrix[ii+3][jj];

temp13 = Cmatrix[ii+3][jj+1];

temp14 = Cmatrix[ii+3][jj+2];

temp15 = Cmatrix[ii+3][jj+3];

for(k=0;k!=100;k+=4){

for(ii=i;ii!=i+4;ii+=4){

for(jj=j;jj!=j+4;jj+=4){

for(kk=k;kk!=k+4;kk++){

Cmatrix[ii][jj] += Amatrix[ii][kk]\*Bmatrix[kk][jj];

Cmatrix[ii][jj+1] += Amatrix[ii][kk]\*Bmatrix[kk][jj+1];

Cmatrix[ii][jj+2] += Amatrix[ii][kk]\*Bmatrix[kk][jj+2];

Cmatrix[ii][jj+3] += Amatrix[ii][kk]\*Bmatrix[kk][jj+3];

Cmatrix[ii+1][jj] += Amatrix[ii+1][kk]\*Bmatrix[kk][jj];

Cmatrix[ii+1][jj+1] += Amatrix[ii+1][kk]\*Bmatrix[kk][jj+1];

Cmatrix[ii+1][jj+2] += Amatrix[ii+1][kk]\*Bmatrix[kk][jj+2];

Cmatrix[ii+1][jj+3] += Amatrix[ii+1][kk]\*Bmatrix[kk][jj+3];

Cmatrix[ii+2][jj] += Amatrix[ii+2][kk]\*Bmatrix[kk][jj];

Cmatrix[ii+2][jj+1] += Amatrix[ii+2][kk]\*Bmatrix[kk][jj+1];

Cmatrix[ii+2][jj+2] += Amatrix[ii+2][kk]\*Bmatrix[kk][jj+2];

Cmatrix[ii+2][jj+3] += Amatrix[ii+2][kk]\*Bmatrix[kk][jj+3];

Cmatrix[ii+3][jj] += Amatrix[ii+3][kk]\*Bmatrix[kk][jj];

Cmatrix[ii+3][jj+1] += Amatrix[ii+3][kk]\*Bmatrix[kk][jj+1];

Cmatrix[ii+3][jj+2] += Amatrix[ii+3][kk]\*Bmatrix[kk][jj+2];

Cmatrix[ii+3][jj+3] += Amatrix[ii+3][kk]\*Bmatrix[kk][jj+3];

}

Cmatrix[ii][jj] = temp;

Cmatrix[ii][jj+1] = temp1;

Cmatrix[ii][jj+2] = temp2;

Cmatrix[ii][jj+3] = temp3;

Cmatrix[ii+1][jj] = temp4;

Cmatrix[ii+1][jj+1] = temp5;

Cmatrix[ii+1][jj+2] = temp6;

Cmatrix[ii+1][jj+3] = temp7;

Cmatrix[ii+2][jj] = temp8;

Cmatrix[ii+2][jj+1] = temp9;

Cmatrix[ii+2][jj+2] = temp10;

Cmatrix[ii+2][jj+3] = temp11;

Cmatrix[ii+3][jj] = temp12;

Cmatrix[ii+3][jj+1] = temp13;

Cmatrix[ii+3][jj+2] = temp14;

Cmatrix[ii+3][jj+3] = temp15;

}

}

}

}

}

/\*testing code\*/

/\*

for(i=0;i!=100;i++){

for(j=0;j!=100;j++){

printf("%f",Cmatrix[i][j]);

}

}

\*/

}