ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΜ&ΜΥ

Προηγμένα Θέματα
Αρχιτεκτονικής Υπολογιστών

4η Άσκηση
Ακ. έτος 2011-2012

Γρηγόρης Λύρας Α.Μ.: 03109687

19 Ιουλίου 2012

Εισαγωγή

I) Προσομοίωση

Ο κώδικας που μας δόθηκε ήταν ο ακόλουθος:

```
/* -.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
 * File Name : partA.c
 * Creation Date : 16-07-2012
 * Last Modified : Mon 16 Jul 2012 01:46:39 PM EEST
 * Created By : Greg Liras <gregliras@gmail.com>
 _.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-.*/

#include <stdio.h>
#include <stdlib.h>
#define __MAGIC_CASSERT(p) do {                         \
    typedef int __check_magic_argument[(p) ? 1 : -1];   \
} while (0)

#define MAGIC(n) do {                       \
    __MAGIC_CASSERT(!(n));                  \
    __asm__ __volatile__ ("xchg %bx,%bx");  \
} while (0)

#define MAGIC_BREAKPOINT MAGIC(0)



inline int min(int a, int b)
{
    if(a<=b)return a;
    else return b;
}
void init_matrix(float **mat, int n)
{
    unsigned int i,j;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            mat[i][j] = (float)(i+j);
}
int main(int argc, char **argv)
{
    float **A,**B,**C;
    int i,j,k,N;
    N=atoi(argv[1]);
    A=(float**)malloc(N*sizeof(float*));

    for(i=0; i<N; i++)
        A[i]=(float*)malloc(N*sizeof(float));

    B=(float**)malloc(N*sizeof(float*));

    for(i=0; i<N; i++)
        B[i]=(float*)malloc(N*sizeof(float));

    C=(float**)malloc(N*sizeof(float*));

    for(i=0; i<N; i++)
        C[i]=(float*)malloc(N*sizeof(float));

    fprintf(stderr, "Initializing matrices...\n");
    init_matrix(A, N);
    init_matrix(B, N);
    init_matrix(C, N);
    MAGIC_BREAKPOINT;
    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];
    }
    MAGIC_BREAKPOINT;
    return 0;
}
```

II) Ιεραρχία μνήμης και μοντέλο απόδοσης

Χρησιμοποιήσαμε την ιεραρχία μνήμης όπως μας δίνεται στο Παράρτημα.

	assoc	line size	lines	size
L1 instruction cache	2	64	512	32768 = 32 KB
L1 data cache	2	64	512	32768 = 32 KB
L2 cache	4	128	1024	131072 = 128 KB

Πίνακας 1: Cache Hierarchy

Cycles	1885656236
L1 miss ratio	0.0117
L2 miss ratio	0.0183

Πίνακας 2: Μετρικές πρώτης εκτέλεσης

Τεχνικές Βελτιστοποίησης

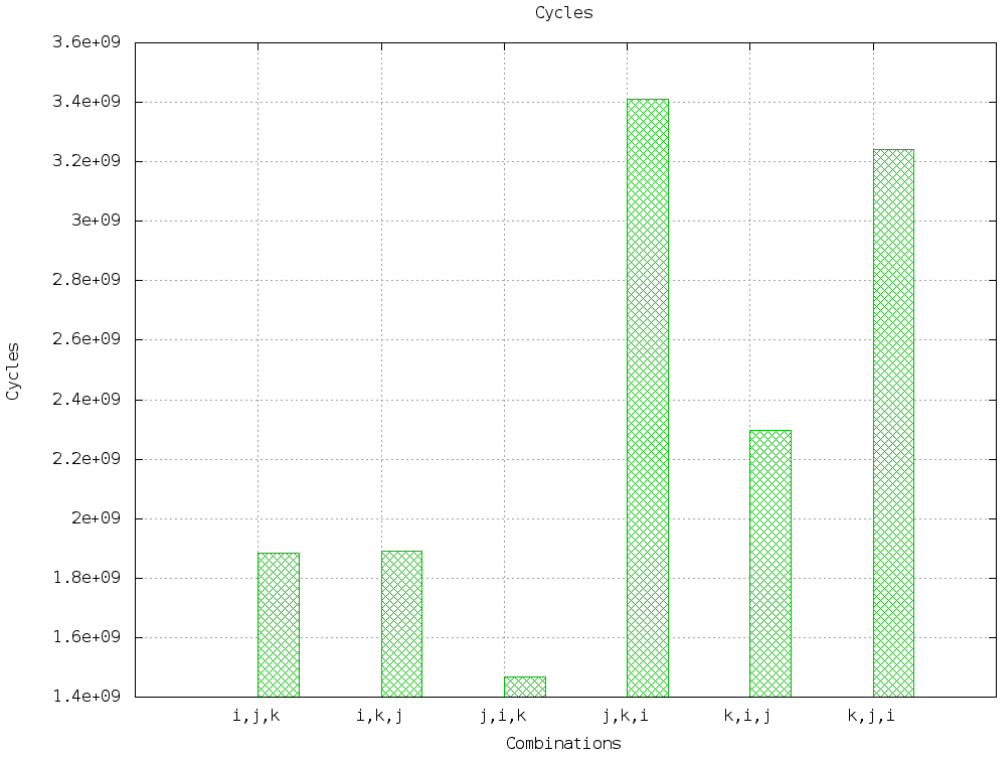
III) Loop Interchange

Για την προσομοίωση κάναμε 6 εκτελέσεις για κάθε διάταξη των i,j,k όπως φαίνεται στον πίνακα 3.

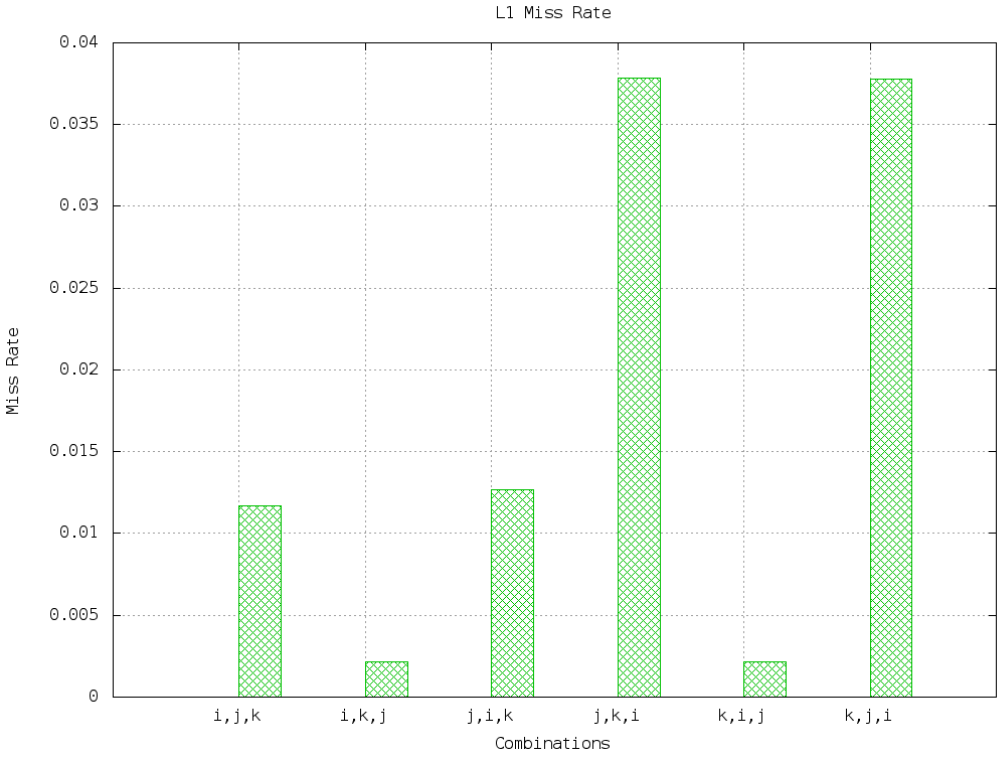
Σειρά προσομοίωσης	Διάταξη
1	i,j,k
2	i,k,j
3	j,i,k
4	j,k,i
5	k,i,j
6	k,j,i

Πίνακας 3: Δυνατές Διατάξεις

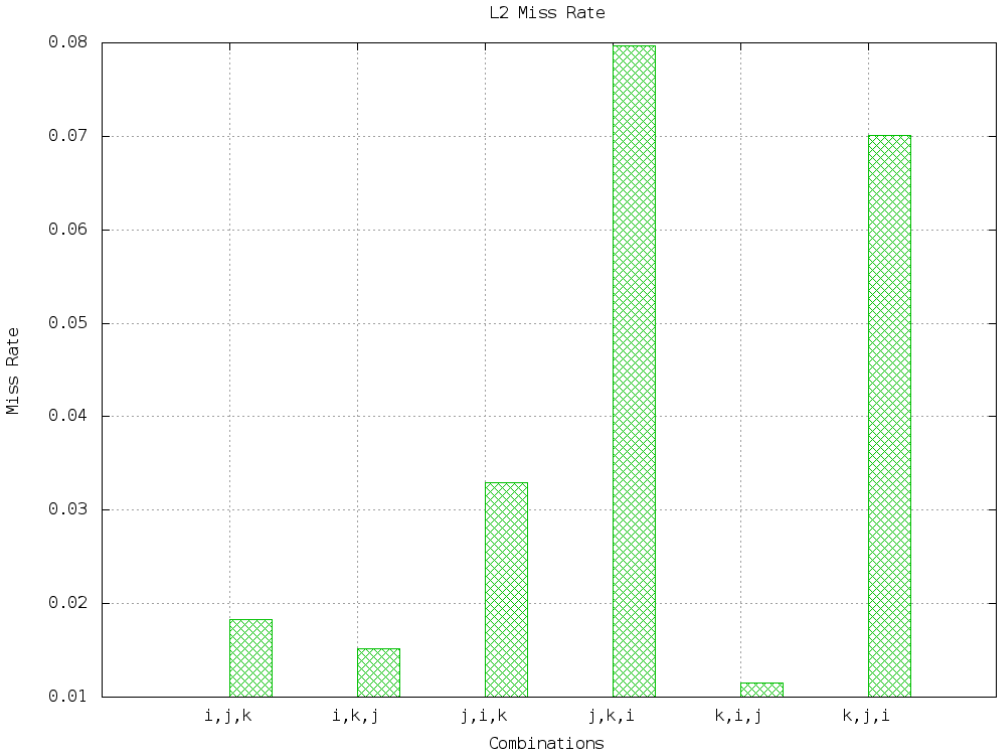
Τα αποτελέσματα φαίνονται στα ακόλουθα σχήματα.



Σχήμα 1: Cycles



Σχήμα 2: L1 Miss Rate



Σχήμα 3: L2 Miss Rate

IV) Cache Blocking

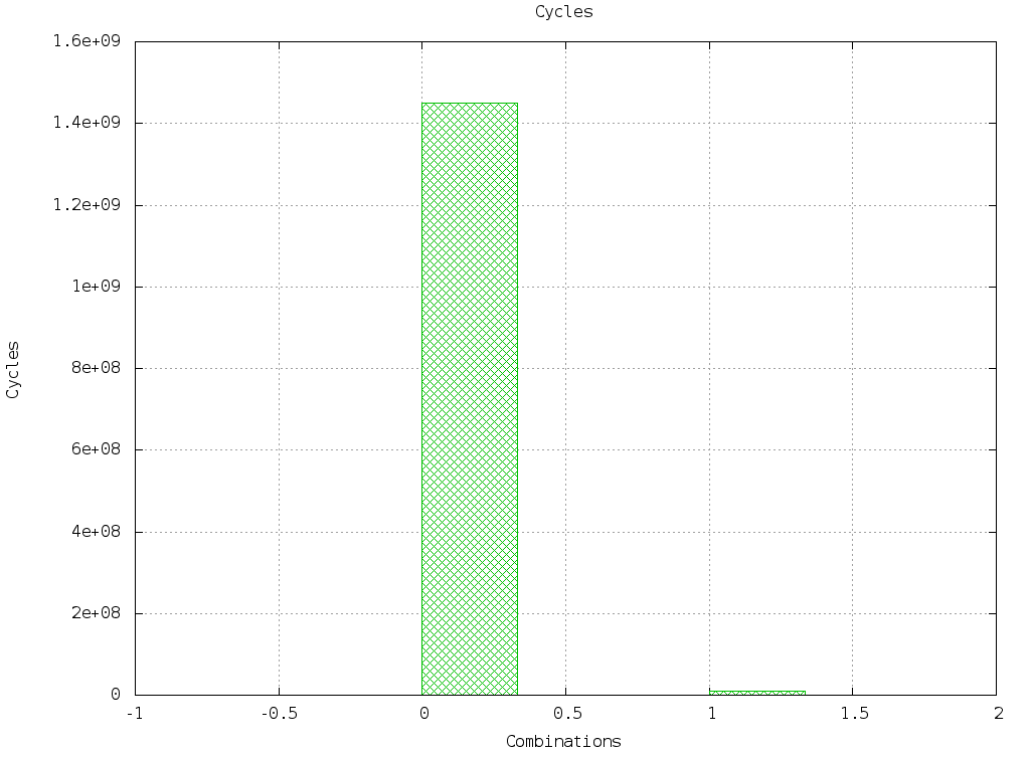
Η L1 cache έχει μέγεθος 32KB, και line size 64 bytes. Κάνουμε πράξεις μεταξύ αριθμών κινητής υποδιαστολής σε 32bit σύστημα συνεπώς κάθε ένας έχει μέγεθος 4bytes. Κάθε cache line χωράει 16 αριθμούς συνεπώς για να εκμεταλλευτούμε καλύτερα την τοπικότητα των αναφορών θα εκτελούμε κάθε loop σε 16δες. Έτσι θα χρησιμοποιούμε δεδομένα που έχουν ήδη έρθει στην cache.

```c
/* -.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
 * File Name : partA.c
 * Creation Date : 16-07-2012
 * Last Modified : Tue 17 Jul 2012 06:07:55 PM EEST
 * Created By : Greg Liras <gregliras@gmail.com>
 _.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-.*/

#include <stdio.h>
#include <stdlib.h>
#define __MAGIC_CASSERT(p) do {                         \
    typedef int __check_magic_argument[(p) ? 1 : -1];   \
} while (0)

#define MAGIC(n) do {                       \
    __MAGIC_CASSERT(!(n));                  \
    __asm__ __volatile__ ("xchg %bx,%bx");  \
} while (0)

#define MAGIC_BREAKPOINT MAGIC(0)



inline int min(int a, int b)
{
    if(a<=b)return a;
    else return b;
}
void init_matrix(float **mat, int n)
{
    unsigned int i,j;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            mat[i][j] = (float)(i+j);
}
int main(int argc, char **argv)
{
    float **A,**B,**C;
    int i,j,k,N;
    int start,stop;
    N=atoi(argv[1]);
    A=(float**)malloc(N*sizeof(float*));

    for(i=0; i<N; i++)
        A[i]=(float*)malloc(N*sizeof(float));

    B=(float**)malloc(N*sizeof(float*));

    for(i=0; i<N; i++)
        B[i]=(float*)malloc(N*sizeof(float));

    C=(float**)malloc(N*sizeof(float*));

    for(i=0; i<N; i++)
        C[i]=(float*)malloc(N*sizeof(float));

    fprintf(stderr, "Initializing matrices...\n");
    init_matrix(A, N);
    init_matrix(B, N);
    init_matrix(C, N);
    MAGIC_BREAKPOINT;
    for(start=0; start<N; start+=16) {
        stop = start + 16;
        stop = stop <= N ? stop : N;
        for(j=start; j<stop; j++)
            for(i=start; i<stop; i++)
```

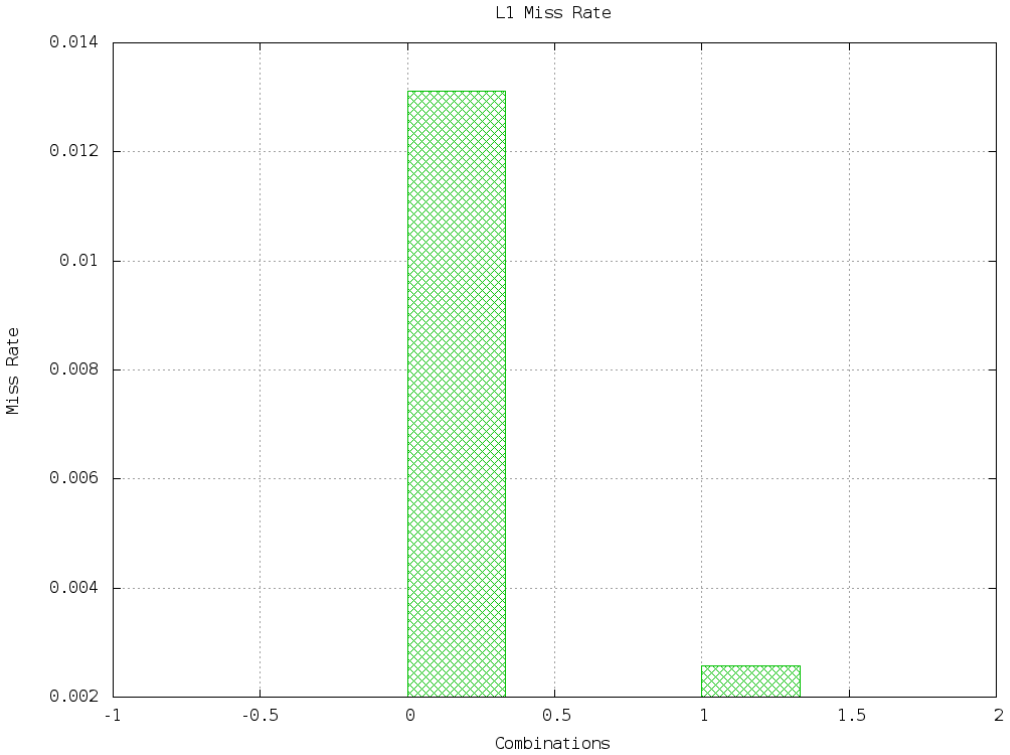
```
66                  for(k=start; k<stop; k++)
67                      C[i][j] += A[i][k]*B[k][j];
68      }
69      MAGIC_BREAKPOINT;
70      return 0;
71  }
```

Cycles	1448776923
Cycles	9385880
L1 miss ratio	0.0131
L1 miss ratio	0.0026
L2 miss ratio	0.0292
L2 miss ratio	0.0065

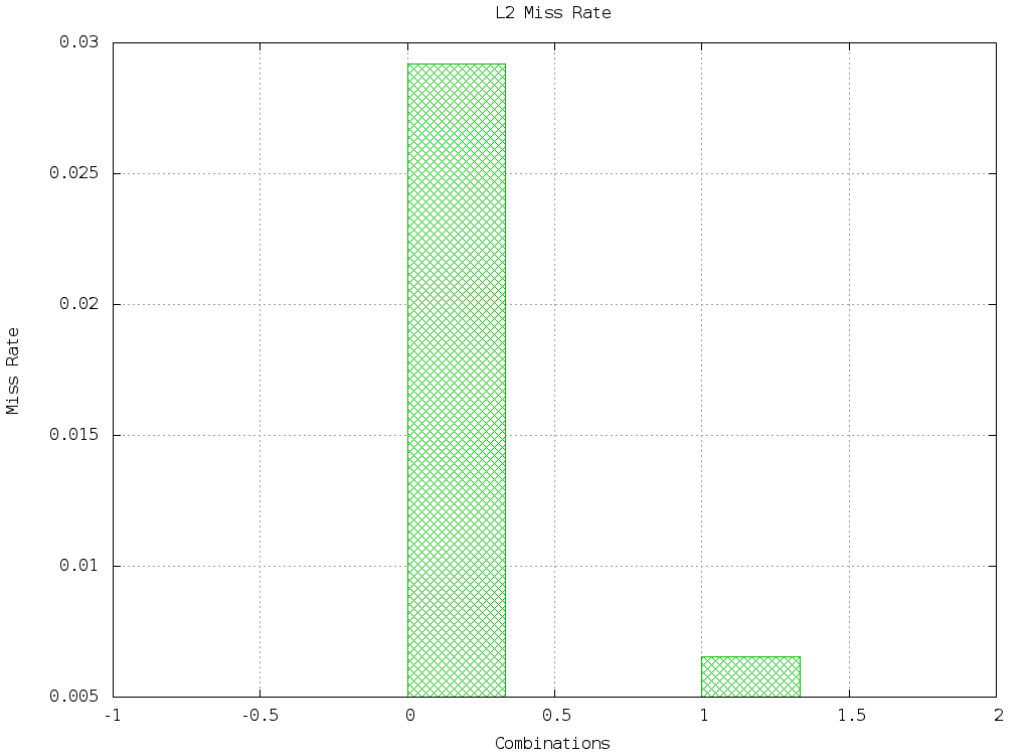
Πίνακας 4: Διαφορά των εκτελέσεων



Σχήμα 4: Cycles



Σχήμα 5: L1 Miss Rate



Σχήμα 6: L3 Miss Rate