Αρχιτεκτονική Υπολογιστών

3η σειρά ασκήσεων

Ονοματεπώνυμο: Λιαροκάπης Αλέξανδρος

Αριθμός Μητρώου: 03114860

Άσκηση 1

Παρατίθεται ο πίνακας διευθύνσεων σε δεκαεξαδική και δυαδική μορφή.

No	Hex Address	Binary Address	Cache Hit/Miss
1	0x044	001000100	Miss
2	0x042	001000010	Hit
3	0x048	001001000	Miss
4	0x1af	110101111	Miss
5	0x04a	001001010	Miss

- Βλέπουμε πως στην 2 γίνεται hit, επομένως οι 1 και 2 έχουν ίδιο tag και index. Αφού το 3ο απο το τέλος bit είναι διαφορετικο, σίγουρα το offset θα είναι τουλάχιστον τα 3 τελευταία bits.
- Στην 3 γίνεται miss άρα έχει διαφορετικό tag ή index απο την 2. Αφού μόνο το 4ο bit διαφέρει και μας δίνει κάποια καινούργια πληροφορία, εξασφαλίζουμε πως το offset είναι τα 3 τελευταία bits, και πως το index περιέχει τουλάχιστον το 4ο.
- Τα 3 και 5 είναι σχεδόν πανομοιότυπα με μία μικρή διαφορά στο offset. Χωρίς κάποιο άλλο cache query μεταξύ τους θα έπρεπε να γίνει hit. Το γεγονός πως γίνεται miss σημαίνει πως το 4 έχει ίδιο index αφού επηρρεάζει το ίδιο block με τα 3,5 αλλα διαφορετικό tag. Έτσι το index μπορεί να είναι είτε μόνο το 4ο απο το τέλος bit είτε τα bits 4 και 5 απο το τέλος. Το tag είναι όλα τα υπόλοιπα.

Σε κάθε περίπτωση το offset είναι 3 bits και αφού η ελάχιστη μονάδα διευθυνσιοδότησης είναι το byte κάθε block έχει μέγεθος 8 bytes.

Στην περίπτωση που το index είναι 1 bit έχουμε 2 blocks και επομένως το μέγεθος του cache είναι 16 bytes.

Αντίστοιχα στην περίπτωση που το index είναι 2 bits, έχουμε 4 blocks και το μέγεθος του cache είναι 32 bytes

Άσκηση 2

Δίνεται ο κώδικας:

```
int i, j;
double a[8][8], b[8][8], c[64];

for (i = 0; i < 8; ++i) {
    for (j = 0; j < 8; ++j) {
        if (i % 2 == 0) {
            a[i][j] = a[i][j] + b[i][j] + c[8*i + j];
        } else {
            a[i][j] = a[i][j] + b[i][j];
        }
    }
}
```

Παίρνοντας το αποτέλεσμα ακέραιας διαίρεσης μίας διεύθυνσης με το μέγεθος του cache παίρνουμε τα index και offset fields. Και οι 3 πίνακες έχουν 64 στοιχεία των 8 bytes και επομένως μέγεθος ίσο με 512 bytes. Όπως είναι εμφανές απο την πρώτη πρόταση, τα $a[i][j]$, $b[i][j]$ και $c[8*i+j]$ έχουν ίδιο index και offset και το πρώτο στοιχείο τους έχει διεύθυνση που αντιστοιχεί στο offset 0 του block 0. Τα στοιχεία των πινάκων στη μνήμη διευθυνσιοδοτούνται διαδοχικά και μπορούμε να προβλέψουμε πως ανα 4 στοιχεία θα γεμίζει ένα block. Επίσης προβλέπουμε πως δεν θα υπάρξουν capacity misses αφού ένας πίνακας χωράει στο cache.

Παρατίθεται ο πίνακας διευθυνσιοδοτήσεων για τις δύο πρώτες επαναλήψεις με direct-mapped cache.

$a[0][0]$	$b[0][0]$	$c[0]$	$a[0][0]$	com-m	m	m	m
$a[0][1]$	$b[0][1]$		$a[0][1]$	h	m		m
$a[0][2]$	$b[0][2]$	$c[2]$	$a[0][2]$	h	m	m	m
$a[0][3]$	$b[0][3]$		$a[0][3]$	h	m		m
$a[0][4]$	$b[0][4]$	$c[4]$	$a[0][4]$	com-m	m	m	m
$a[0][5]$	$b[0][5]$		$a[0][5]$	h	m		m
$a[0][6]$	$b[0][6]$	$c[6]$	$a[0][6]$	h	m	m	m
$a[0][7]$	$b[0][7]$		$a[0][7]$	h	m		m
$a[1][0]$	$b[1][0]$	$c[8]$	$a[1][0]$	com-m	m	m	m
$a[1][1]$	$b[1][1]$		$a[1][1]$	h	m		m
$a[1][2]$	$b[1][2]$	$c[10]$	$a[1][2]$	h	m	m	m
$a[1][3]$	$b[1][3]$		$a[1][3]$	h	m		m
$a[1][4]$	$b[1][4]$	$c[12]$	$a[1][4]$	com-m	m	m	m
$a[1][5]$	$b[1][5]$		$a[1][5]$	h	m		m
$a[1][6]$	$b[1][6]$	$c[14]$	$a[1][6]$	h	m	m	m
$a[1][7]$	$b[1][7]$		$a[1][7]$	h	m		m

Συνολικά έχουμε 48 hits και 176 misses απο τις οποίες οι 16 είναι compulsory και οι 160 είναι conflict misses

Παρατίθεται ο πίνακας διευθυνσιοδοτήσεων για τις δύο πρώτες επαναλήψεις με 2-way set-associative cache

$a[0][0]$	$b[0][0]$	$c[0]$	$a[0][0]$	com-m	m	m	m
$a[0][1]$	$b[0][1]$		$a[0][1]$	h	m		h
$a[0][2]$	$b[0][2]$	$c[2]$	$a[0][2]$	h	h	m	m
$a[0][3]$	$b[0][3]$		$a[0][3]$	h	m		h
$a[0][4]$	$b[0][4]$	$c[4]$	$a[0][4]$	com-m	m	m	m
$a[0][5]$	$b[0][5]$		$a[0][5]$	h	m		h
$a[0][6]$	$b[0][6]$	$c[6]$	$a[0][6]$	h	h	m	m
$a[0][7]$	$b[0][7]$		$a[0][7]$	h	m		h
$a[1][0]$	$b[1][0]$	$c[8]$	$a[1][0]$	com-m	m	m	m
$a[1][1]$	$b[1][1]$		$a[1][1]$	h	m		h
$a[1][2]$	$b[1][2]$	$c[10]$	$a[1][2]$	h	h	m	m
$a[1][3]$	$b[1][3]$		$a[1][3]$	h	m		h
$a[1][4]$	$b[1][4]$	$c[12]$	$a[1][4]$	com-m	m	m	m
$a[1][5]$	$b[1][5]$		$a[1][5]$	h	m		h
$a[1][6]$	$b[1][6]$	$c[14]$	$a[1][6]$	h	h	m	m
$a[1][7]$	$b[1][7]$		$a[1][7]$	h	m		h

Συνολικά έχουμε 96 hits και 128 misses από τις οποίες οι 16 είναι compulsory και οι 112 είναι conflict misses

Χωρίς να αλλάξουμε τις δηλώσεις, μπορούμε να αλλάξουμε τον κώδικα έτσι ώστε να μην γίνονται fetch 3 διαφορετικοί πίνακες σε κάθε επανάληψη και έτσι να αποφύγουμε τα περισσότερα conflict misses:

```
int i, j;
double a[8][8], b[8][8], c[64];

for (i = 0; i < 8; ++i) {
    for (j = 0; j < 8; ++j) {
        a[i][j] = a[i][j] + b[i][j];
    }
}

for (i = 0; i < 8 ++i) {
    for (j = 0; j < 8; j += 2) {
        a[i][j] = a[i][j] + c[8*i+j];
    }
}
```

Τροποποιώντας τον παραπάνω έτσι ώστε τα δύο loops να τρέχουν για κάθε block του cache ξεχωριστά, αποφεύγουμε να κάνουμε δύο φορές fetch τον πίνακα a:

```
int i, j;
double a[8][8], b[8][8], c[64];

for (i = 0; i < 8; ++i) {
    for (j = 0; j < 4; ++j)
        a[i][j] = a[i][j] + b[i][j];
    for (j = 0; j < 4; j += 2)
        a[i][j] = a[i][j] + c[8*i+j];
    for (j = 4; j < 8; ++j)
        a[i][j] = a[i][j] + b[i][j];
    for (j = 4; j < 8; j += 2)
        a[i][j] = a[i][j] + c[8*i+j];
}
```

Παρατίθεται ο πίνακας διευθυνσιοδοτήσεων για την πρώτη επανάληψη για τον παραπάνω κώδικα.

a[0][0]	b[0][0]	a[0][0]	com-m	m	h
a[0][1]	b[0][1]	a[0][1]	h	h	h
a[0][2]	b[0][2]	a[0][2]	h	h	h
a[0][3]	b[0][3]	a[0][3]	h	h	h
a[0][0]	c[0]	a[0][0]	h	m	h
a[0][2]	c[2]	a[0][2]	h	h	h
a[0][4]	b[0][4]	a[0][4]	com-m	m	h
a[0][5]	b[0][5]	a[0][5]	h	h	h
a[0][6]	b[0][6]	a[0][6]	h	h	h
a[0][7]	b[0][7]	a[0][7]	h	h	h
a[0][4]	c[4]	a[0][4]	h	m	h
a[0][6]	c[6]	a[0][6]	h	h	h

Συνολικά έχουμε 240 hits και 48 misses από τις οποίες οι 16 είναι compulsory και οι 32 είναι conflict misses.