ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ 4η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ

Όνομα: Μιχάλης Παπαδόπουλλος
Α.Μ: 03114702
Ημερ/νια παράδοσης: 12/02/2017

[**ΜΕΡΟΣ Α**]

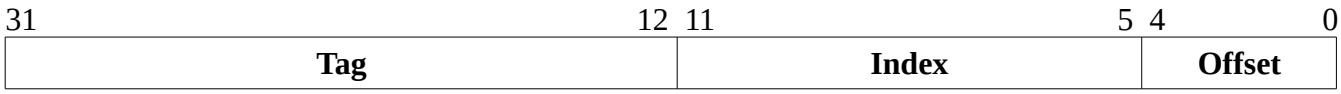
- sizeof (double) = 8 bytes = 2 words

L1 Cache:

- Direct mapped (1-way associative)
- Write allocate (update on write miss)
- Write back
- 128 blocks
- block size: 32 bytes = 8 words

Each address in MIPS is 32 bits long.

- 128 blocks = 2^7 => Tag: 7 bits
- Block size = 8 words = 32 bytes = 2^5 => Offset: 5 bits
- Tag = sizeof (ADDRESS) – sizeof (index) – sizeof (offset) = 32 – 7 – 5 = 20 bits



Δηλαδή:

- Offset = 5 bits
- Index = 7 bits
- Tag = 20 bits

[**ΜΕΡΟΣ Β**]

```
#define N 16
int i,j;
double a[N*N], b[N][N], c[N][N];

for (i = 0; i < N; i++)
   for (j = 0; j < N; j++)
        a[i * N + j] = b[j][i] + c[i][j];
```

Σε κάθε block της cache αποθηκεύονται 4 συνεχόμενα στοιχεία του κάθε πίνακα

- Ο πίνακας **a** γίνεται map στα block 0..63
- Ο πίνακας **b** γίνεται map στα block 64..127
- Ο πίνακας **c** γίνεται map στα block 0..63

Ο πίνακας b προσπελαύνεται κατα στήλες.

- Όταν **i % 4 = 0** (4 φορές: i = 0, i = 4, i = 8, i = 12), δηλαδή στην αρχή μιας διεύθυνσης, προκύπτει **compulsory miss**, και φορτώνονται στη cache τα στοιχεία b[j][i], b[j][i + 1], b[j][i + 2], b[j][i + 3], δηλαδή
  compulsory misses = 4 * N * 1= 64
- Όταν **i % 4 ≠ 0** (12 φορές: i = {1, 2, 3}, {5, 6, 7}, {9, 10, 11}, {13, 14, 15}), τότε το στοιχείο υπάρχει ήδη από προηγούμενη αναφορά, οπότε προκύπτει
  hits = (4 * 3) * N * 1 = 192

Για τους πίνακες **a, c** επειδή γίνονται map στα ίδια block της cache:

- **conflict miss** σε κάθε επανάληψη όπου **j % 4 ≠ 0,** δηλαδή
  conflict misses = 2 * N * 12 = 384
- ενώ έχουμε **compulsory miss** σε κάθε επαναληψη όπου **j % 4 = 0**, δηλαδή
  compulsory misses = 2 * N * 4 = 128

Σύνολικά: 192 (compulsory misses) + 384 (conflict misses) + 192 (hits) = 576 (misses) + 192 (hits) = 768 (accesses)

$$Hit_{ratio} = \frac{192_{(hits)}}{768_{(mem.acceses)}} = 25\%$$

[**ΜΕΡΟΣ Γ**]

- 4-way set-associative
- 128 blocks
- Block size = 32 bytes = 8 words
- Write back
- Write allocate

Με τα νέα δεδομένα προκύπτουν:

- index = $\log_2$ (128 / 4) = 5 bits
- offset = 5 bits
- tag = 32 – 10 = 22 bits

Οι πίνακες **a, b, c** γίνονται map στα set 0..3 της αντίστοιχης γραμμής που ορίζει το index. Ακολουθόντας την ίδια συλλογιστική:

- Όταν **i % 4 = 0** (4 φορές: i = 0, i = 4, i = 8, i = 12), δηλαδή στην αρχή μιας διεύθυνσης, προκύπτει **compulsory miss**. Δηλαδή
  compulsory misses = 3 * 4 * 16 = 192
- Όταν **j % 4 ≠ 0** έχουμε για τους πίνακες **a, c**
  hits = 2 * N * 12 = 384
- Για τον πίνακα **b** όταν **i % 4 ≠ 0** έχουμε
  hits = 12 * N * 1 = 192

Συνολικά: 576 **hits** + 192 **misses** = 768 (accesses)

$$Hit_{ratio} = \frac{576_{(hits)}}{768_{(mem.acceses)}} = 75\%$$

Παρατηρούμε ότι με την αλλαγή στο associativity της cache απο 1 σε 4, πετυγχαίνουμε **τριπλάσιο** hit rate.

[**ΜΕΡΟΣ Δ**]

Γενικά: $(t_{total}) = (acceses) \times [(t_{hit}) + (miss_{rate}) \times (t_{miss})]$

Στην περίπτωση της **Direct Mapped**:
$(t_{total})_{DM} = 768 \times (2 + 0.75 \times 10)ns = 768 \times 9.5ns$

ενώ στην περίπτωση όπου έχουμε **4-way set-associative**:
$(t_{total})_{4-way} = 768 \times (k + 0.25 \times 10)ns = 768 \times (k + 2.5)ns$

Θέλουμε: $(t_{total})_{4-way} \leq (t_{total})_{DM}$ οπότε προκύπτει $k \leq 7ns$