**Assigment Cache (Nhóm 2)**

**Exercise 1:**

Design an L1 cache (number of bits for tag, entry, …) for a CPU with 32-bit address in 3 following types. The cache size is 32KB, block (line) size is 32 byte

• Direct mapped

• Fully Associative

• 4 way associative

- Số block offset bits:

Block size = 32 byte = 25 byte

=> Block offset bits = 5

**Direct mapped cache**

[ tag bits][index bits][ block offset bits]

- Xác định số index bits:

Number of blocks in cache = Cache size/Block size = 32KB/32B = 210

=> Index bits = 10

- Xác định số tag bits

CPU address is 32 bits

Number of bits in Tag = Total bits - Index bits - Block offset bits = 32 - 10 - 5 = 17

**Fully associative cache**

[ tag bits][ block offset bits]

- Xác định số tag bits

CPU address is 32 bits

Number of bits in Tag = Total bits - Block offset bits = 32 - 5 = 27

**4 way associative**

[ tag bits][index bits][ block offset bits]

- Xác định số index bits:

Number of blocks in cache = Cache size/Block size = 32KB/32B = 210

4-way set associative 210/4= 28

=> Index bits = 8

- Xác định số tag bits

CPU address is 32 bits

Number of bits in Tag = Total bits - Index bits - Block offset bits = 32 - 8 - 5 = 19

**Exercise 2:**

For the cache in Exercise 1, assumed the cache is 4-way associative, how many cache hit, miss occur if the CPU execute the following memory access sequence in case of:

* Write through no write allocation
* Write back with write allocation.

RD 0x00000000

WR 0x01000000

RD 0x01000010

WR 0x02000050

RD 0x02000058

***Write through no write allocation*.**

- RD 0x0000 0000 : Read miss => cache miss \*

Put to cache value of address form: 0x0000 0000 - 0x0000 001F

- WR 0x0100 0000 : Write miss => Data didn’t have in cache => cache miss \*

- RD 0x0100 0010 : Read miss => cache miss \*

Put to cache value of address form:: 0x0100 0010 -> 0x0100 002F

- WR 0x0200 0050: Write miss => Data didn’t have in cache => cache miss \*

- RD 0x0200 0058: Read miss => cache miss \*

Put to cache value of address form: 0x0200 0058 - 0x0200 0077

=> Total we have 5 times cache miss.

***Write back with write allocation*.**

- RD 0x0000 0000 : Read miss in entry 0 with tag 0x00000 => cache miss\*

Put to cache in entry 0 with value of address from: 0x00000000 – 0x0000001F, dirty = 0.

- WR 0x0100 0000 : Write miss in entry 0 with tag 0x01000 => cache miss \*

Put to cache in entry 0 with value of address from: 0x01000000 – 0x0100001F, dirty = 0

- RD 0x0100 0010 : Read hit in entry 0 with tag 0x01000 => cache hit \*\*

- WR 0x0200 0050: 50(Hex) = 01010000 (Bin) => index = 2, block offser 10 (Hex) => Write miss in entry 2 with tag 0x02000 => cache miss\*

Put to cache in entry 2 with value of address from: 0x02000050 – 0x0200005F

- RD 0x0200 0058: Read hit in entry 2 with tag 0x0200, offset 18 (Hex). => cache hit \*\*

58 (Hex) = 01011000

=> Total we have 3 cache miss and 2 cache hit.

**Exercise 3:**

How much faster/slower is a unified 32KB cache than a separated 16KB I/16KB D cache if the miss rate is ones in the following slide table, and there are 70% instructions are LD/ST. Assumed that unified cache has 1-port only. The hit time is 1 cycle and miss penalty is 50 cycles

Size I-Cache D-Cache U-Cache

16KB 3.82 40.9 51.0

32KB 1.36 38.4 43.3

Misses per 1000 instructions

AMAT= hit time + miss rate X miss penalty = hit time + (misses per instruction/ mem access per instruction) x miss penalty

AMAT-I = 1 + (0.00382 / 1.0) x 50 = 1.191

AMAT-D = 1 + (0.0409/ 0.7) x 50 = 3.921

AMAT Separate = %Inst x AMAT-I + %Data x AMAT-D

AMATSeparate = (100%/170%) x 1.191 + (70%/170%) x 3.921 = 0.59x1.191 + 0.41x3.921 = 2.3103

Unified cache have only 1 port, so cannot access instruction and data simultaneously ->1 extra stall cycle

AMATUnified = (100%/170%) x (1 + (0.0433/1.7)x50) + (70%/170%) x (1 + 1 + (0.0433/1.7)x50) = 2.6852

AMATUnified /AMATUnified = 2.6852 / 2.3103 = 1.1623

A separated 16KB I/16KB D cache is faster than a unified 32KB cache 1.1623 times