CSE/EEE 230 Fall 2016 Assignment 7
Due December 5 11:59PM

1. A cache holds 64 words where each word is 4 bytes. Assuming a 32-bit address, give

    a. A direct-mapped cache with block size = 8 words
    b. 2-way set-associative cache with block size = 8 words
    c. 4-way set-associative cache with block size = 4 words
    d. A fully associative cache with block size = 16 words.

Complete the table for each cache.

	Cache a	Cache b	Cache c	Cache d
total # bits for word & byte displacement	3 & 5	3 & 5	2 & 4	4 & 6
# bits in tag	26 & 24	27 & 25	29 & 27	28 & 26
# sets in cache	8	4	2	1
# bits (total) per set (including valid and dirty bits)	28 & 26	29 & 27	31 & 29	30 & 28

Ans – 64-words each word is 4 bytes
Address bits – 32-bits
a) Direct mapped cache
Block size = 8 words
bits for word displacement = 3-bits
bits for byte displacement = 5-bits
number of sets = 64/8 = 8
Index bits = 3
Tag bits = 32 – (3 + 3) = 26

b) 2-way set associative cache
Block size = 8 words
bits for word displacement = 3-bits
bits for byte displacement = 5-bits
number of sets = 64/(8 * 2) = 4
Index bits = 2
Tag bits = 32 – (3 + 2) = 27

c) 4-way set associative cache
Block size = 4 words
bits for word displacement = 2-bits
bits for byte displacement = 4-bits
number of sets = 64/(8 * 4) = 2
Index bits = 1
Tag bits = 32 – (2 + 1) = 29

c) Fully associative cache
Block size = 16 words
bits for word displacement = 4-bits
bits for byte displacement = 6-bits
Number of ways = 64/16 = 4
Tag bits = 32 – (4) = 28

2. Here is a series of addresses in hexadecimal:

   20, 3C, 10, 16, 20, 04, 28, 60, 10, 17

   Assume a LRU replacement algorithm. For the four caches in problem 1, draw each cache as it would appear at the end of the series of references. Include the valid bit, dirty bit, and tag. Note that these are addresses, not block numbers. Show the memory contents with the range of byte addresses such as M[20-23] for the address 22 (if the block size is 1 word).

Ans – Addresses -
20 – 0010_0000
3C – 0011_1100
10 – 0001_0000
16 – 0001_0110
20 – 0010_0000
04 – 0000_0100
28 – 0010_1000
60 – 0110_0000
10 – 0001_0000
17 – 0001_0111

Final stage of the word displacement caches -
a)

Address Range	Set	Tag	Valid	Dirty
00-1F	0	0	1	1
20-3F	1	x	0	0
40-5F	2	0	1	1
60-7F	3	x	0	0
80-9F	4	1	1	0
A0-BF	5	0	1	0
C0-DF	6	x	0	0
E0-FF	7	0	1	0

b)
Way 0

Addresse Range	Set	Tag	Valid	Dirty
00-1F	0	3	1	0
20-3F	1	1	1	0
40-5F	2	1	1	1
60-7F	3	1	1	0

Way 1

Addresse Range	Set	Tag	Valid	Dirty
00-1F	0	0	1	1
20-3F	1	x	0	0
40-5F	2	0	1	1
60-7F	3	x	0	0

c)

Addresse Range	Set	Tag	Valid	Dirty
00-0C	0	4	1	0
10-1F	1	7	1	0

Addresse Range	Set	Tag	Valid	Dirty
00-0F	0	2	1	1
10-1F	1	2	1	1

Addresse Range	Set	Tag	Valid	Dirty
00-0F	0	5	1	0
10-1F	1	0	1	0

Addresse Range	Set	Tag	Valid	Dirty
00-0F	0	C	1	0
10-1F	1	x	0	0

d) In fully associative the data can be placed in any of the ways!

Addresse Range	Set	Tag	Valid	Dirty
00-0F	0	2	1	1

Addresse Range	Set	Tag	Valid	Dirty
00-0F	0	6	1	0

Addresse Range	Set	Tag	Valid	Dirty
00-0F	0	1	1	1

Addresse Range	Set	Tag	Valid	Dirty
00-0F	0	0	1	0

3. A processor has a base CPI of 1.6 when all references hit the cache and a clock rate of 4 GHz. The time to access main memory is 50ns including all miss handling. The miss rate in the instruction and data cache is 3%. On the average, 30% of the instructions in the program include a memory read/write. What is the effective CPI?

Ans – CPI = 1.6
Freq = 4 GHz
Main mem = 50 ns
Miss rate = 3%
30% instruction include memory read write
Effective CPI = Base CPI + latency due to cache misses
Total memory accesses = CPI * (data mem + instr mem)
= 1.6 * 0.30 + 1
= 1.48
Cycles per instruction spent on cache miss
1.48 memory access/instr * 3% miss rate * (200 cycles per miss)
= 8.88
Eff CPI = 1.6 + 8.88 = 10.48

43. The memory hierarchy contains a single cache with a miss rate of 2% that holds both instructions and data. The miss penalty to access main memory is 100 cycles. 20% of the instructions are jumps, 20% are stores, 20% are loads (30% have values used in the next instruction), 10% are branches (taken 20% of the time) and 30% are ALU instructions. Jumps and branches are determined in the ID stage.

a. What is the base CPI?

Ans – Base CPI is the one without taking cache misses into account -

J – 20%
Stores – 20%
Loads – 20%
Branches – 10%
ALU – 30%

For the base CPI -
Jump/branches would take 2 cycles
Load would take – 5 cycles
Stores would take – 4 cycles
ALU – 3 cycles

Base CPI = (30 * 2 + 20 * 4 + 20 * 5 + 3 * 30)/ 100 = 3.3

b. What is the effective CPI?

For the effective CPI, we would consider the cache misses -
Total memory accesses = 3.3 * 40% + 1 = 2.32
Cycles per instruction spent on cache miss
2.32 memory access/instr * (1.4% + 0.02%) miss rate * (100 cycles per miss) = 3.2
Effective CPI = 3.3 + 3.2 = 6.5