CS224

LAB 6

PRELIMINARY WORK

SECTION 3

MUSTAFA MERT GÜLHAN

22201895

09/12/2024

Part 1:

No.	Cache Size KB	N Way Cache	Word Size (bits)	Block size (no. of words)	No. of Sets	Tag size in Bits	Index Size (Set No.) in bits	Word Block Offset Size in bits	Byte Offset Size in bits	Block Replacement Policy Needed (Yes/No)
1	64	1	32	4	4096	16	12	2	2	No
2	64	2	32	4	2048	17	11	2	2	Yes
3	64	4	32	8	512	18	9	3	2	Yes
4	64	Full	32	8	1	27	0	3	2	Yes
9	128	1	16	4	16384	-1	14	2	1	No
10	128	2	16	4	8192	0	13	2	1	Yes
11	128	4	16	16	1024	1	10	4	1	Yes
12	128	Full	16	16	1	11	0	4	1	Yes

No 9. is not possible because we exceed the word size 16 bits.

Part 2:

a)

Instruction	Iteration No.				
	1	2	3	4	5
lw $t1, 0x4($0)	Compulsory	Hit	Hit	Hit	Hit
lw $t2, 0xC($0)	Compulsory	Hit	Hit	Hit	Hit
lw $t3, 0x8($0)	Hit	Hit	Hit	Hit	Hit

Note: I assumed the last 2 bits are byte offset and cache uses spatial locality.

b)

Byte offset = 2 bits
Block offset = 1 bit
Index = 2 bits
So Tag is 32-2-1-2 = 27 bits

So 1 block of cache is 1+27+32+32 = 92 bits

There are total of 4 blocks, 4*92 = 368 bits

Thus, total cache memory is 368 bits.

c)

1 AND Gate
1 Equality Comparator
1 2:1 Mux

is enough to implement this cache.

Part 3:

a)

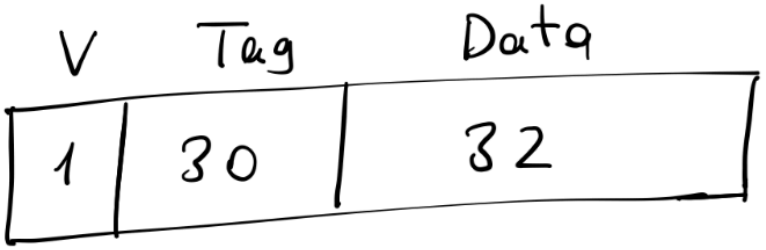
Instruction	Iteration No.				
	1	2	3	4	5
lw $t1, 0x4($0)	Compulsory	Capacity	Capacity	Capacity	Capacity
lw $t2, 0xC($0)	Compulsory	Capacity	Capacity	Capacity	Capacity
lw $t3, 0x8($0)	Capacity	Capacity	Capacity	Capacity	Capacity

Note: In the first iteration for 0x8 I think it can also be compulsory because of accessing it for the first time but also the cache is full so I believe both capacity and compulsory would work.

b)

Byte offset = 2 bits
Block offset = 0 bits because there's only 1 data in each block.
Index = 0 bits because there's only 1 set.

So Tag is 32-2-0-0 = 30 bits



So 1 block of cache is 1+30+32 = 63 bits

There are 2 blocks so 2*63 = 126 bits

We also need a single bit for LRU because there's 2 blocks in a set. Thus total cache size in bits is 126 + 1 = 127 bits.

c)

2 AND Gate
2 Equality Comparator
1 OR Gate
1 2:1 Mux

is enough to implement this cache.

Part 4:

L1: 1 cycle, 20% miss
L2: 4 cycles, 5% miss
MM: 40 cycles

AMAT = (L1 Cycle) + (MissRate L1) * (L2 Cycle + MissRate L2 * MM Cycle)
AMAT = 1 + 20% * (4+(5% * 40))
AMAT = 1 + 20% * (4 + 2)
AMAT = 1 + 1.2
AMAT = 2.2 Cycles

4GHz = 4 * 10^9 cycles per sec

Execution Time =
(Instruction Count) * AMAT / Clock Rate

10^12 * 2.2 / (4 * 10^9) = 550 seconds

Thus, results are:

AMAT = 2.2 Cycles
Execution Time = 550 seconds