CSA FINAL EXAM
Sahil Makwane – sm9127

Q1.
(i) CPI for the benchmark

Assume there are 100 instructions. Then, **30** are load & store, **50** are arithmetic instructions and **20** are all others.

CPI = (30 * 6) + (50 * 4) + (20 * 3) = 440 cycles (per 100 instructions) = **4.4** cycles per instruction

CPI = **4.4**

(ii) CPU execution time = 11 seconds

MIPS = Clock rate / CPI * 10^6
MIPS = 200 * 10^6 / 4.4 * 10^6
MIPS = **45.4546**

Instruction throughput processor performance is **45.4546**

(iii) Clock Rate = 1 / Cycle Time

Cycle Time = 1 / Clock Rate
Cycle Time = 1 / (200 * 10^6) = 5 * 10^{-9}

There is a 20% increase in the cycle time.

Thus, New Cycle Time = 1.2 * (5 * 10^{-9}) = 6 * 10^{-9}

New Clock Rate = 1 / 6 * 10^{-9} = 166.667 * 10^6
New Clock Rate = **166.6667 MHz**

(iv) We reduce the number of load & store by half, now we have **15** load & store, **50** arithmetic instructions and **20** all others. Total **85** instructions.

Therefore,
CPI = [(15 * 6) + (50 * 4) + (20 * 3)] / 85 = 350 cycles / 85 instructions = 4.12 cycles per instruction

CPI = **4.12**

(v) CPU Time = No. of instructions * CPI / Clock Rate

Calculating the number of instructions which execute in 11 seconds,
$(45.4546 * 10^6) * 11 = 500 * 10^6$ instructions (in 11 seconds)

30% are load & store, so
30% of $500 * 10^6 = 150 * 10^6$ load & store instructions
Since load & store are half, we have only $75 * 10^6$ load & store instructions
Also, total instructions are now $425 * 10^6$

Thus,

CPU Time = $((425 * 10^6) * 4.12) / (166.6667 * 10^6) = 10.5060$ seconds

CPU Time = **10.5060**

Q2.
(i)

1. `add x7, x6, x8`
   There are no dependencies as there is no previous instruction

2. `slt x9, x10, x9`
   There are no dependencies with respect to (1.)

3. `beq x9, x0, next`
   RAW on x9 with respect to (2.)

4. `lw x6, 80(x10)`
   WAR on x6 with respect to (1.) and RAR on x10 with respect to (2.)

(ii)

	cc1	cc2	cc3	cc4	cc5	cc6	cc7	cc8	cc9	cc10
`add x7, x6, x8`	IF	ID	EX	MEM	WB					
`slt x9, x10, x9`		IF	ID	EX	MEM	WB				
`beq x9, x0, next`			IF	ID	EX	MEM	WB			
`lw x6, 80(x10)`				IF	ID	EX	MEM	WB		

CPI = 8 cycles per 4 instructions = 2 cycles per instruction
CPI = **2**

Q3.

(i) 8 bits identifying the set, thus 2^8 = 256 sets
There are 2 lines per set, thus 2 * 256 = **512 lines**

(ii) 24 bits are there in the address from which 2 are for word id. Thus, 24-2 = 22.
So, there are **2^{22} blocks** in the memory space.

(iii) Converting to binary

$7121C51_6$ = $011100010010000111000101_2$

$011100010010000111000101_2$

01 indicates the 2nd column of data
01110001 indicates 4th row from the bottom

4th row from the bottom contains the matching tag 0111000100100.
Thus, the data is in the 2nd column in the 4th row from bottom, which is $A5_{16}$.
Data is in **$A5_{16}$**.

Q4.

	Compulsory Misses	**Conflict Misses**	**Capacity Misses**
Double the associativity	There is NO effect reason – When the associativity is doubled, there is no effect on the cache as it was not already present in the cache	There is a DECREASE reason – There is a reduction in the conflict misses as doubled associativity means, now there are more places to place the same element	There is NO effect reason – capacity is a constant, thus, no effect on capacity misses
Halve the line size	There is an INCREASE reason – Halving the line size leads to less number of adjacent elements being brought in with the first access to a line	There is an INCREASE reason – There will be an increase in the total caches line thus enabling more conflict misses	There is an INCREASE reason – There is now an increase in the capacity misses as the capacity has been cut in half
Double the number of sets	There is NO effect reason – When the associativity is halved, there is no effect on the cache as it was not already present in the cache	There is an INCREASE reason – Halving the associativity leads to increase in conflict misses as now there are less places to place the same element	There is NO effect reason – capacity is a constant, thus, no effect on capacity misses

Q5.

	Hit Time	**Miss Rate**	**Miss Penalty**
Double the associativity	There is an INCREASE reason – There is a decrease in the number of sets, tags become larger. Thus, there is an increase in the hit time since more tags need to be checked	There is a DECREASE reason – There is a decrease in the miss rate as there are less conflict misses	There is NO effect reason – Miss penalty has no effect as outer memory hierarchy is responsible for influencing the miss penalty
Halve the line size	There is a DECREASE reason – Due to halving the line size the cache is smaller in size now, thus dominating the time to check tag.	There is an INCREASE reason – There is an increase in the miss rate as smaller capacity leads to lesser ability to utilize the spatial locality in a single cache	There is a DECREASE reason – There is a decrease in miss penalty as smaller lines can be brought in faster
Double the number of sets	There is a DECREASE reason – There is a decrease in the hit time as there is an increase in the number of sets, thus, the tags become smaller. Thus, less tags need to be checked	There is a INCREASE reason – There is an increase in the miss rate as doubling the number of sets leads to associativity getting halved resulting in more conflict misses.	There is NO effect reason – Miss penalty has no effect as outer memory hierarchy is responsible for influencing the miss penalty