Homework 2

Q1) Fill in the blanks

a) Instruction Level Parallelism
b) control
c) hardware
d) array / vector
e) vector
f) threads
g) distributed shared memory
h) test top
i) 35 $(2^5 \times 2^{10} \times 2^{10} \times 2^{10})$
j) registers
k) temporal
l) valid

Q2) a) Hit rates (1 cycle clock)

$I_{cache} = 98\%$. $D_{cache} = 92\%$. Penalty = 10 cycles

$$AMAT_{Icache} = 1 + (1 - 0.98) \times 10 = 1 + 0.02 \times 10 = \underline{1.2 \text{ cycles}}$$

↳ Ans

$$AMAT_{Dcache} = 1 + (1 - 0.92) \times 10 = 1 + 0.08 \times 10 = \underline{1.8 \text{ cycles}}$$

↳ Ans

Runtime

Instruct^{n} = $800000 \times 0.98 = 784000$ hits (1 cycle) = 784 000 cycles

$(800000 - 784,000) \times (10+1) = 160000 \times 11 = 176000$ cycles

For Data cache, since we have already counted cache access previously so need to count only penalties for miss.

$(100000 - (100000 \times 0.92)) \times 10 = 8000 \times 10 = 80,000$ cycles

Total runtime = $784,000 + 176000 + 80000 = \underline{1040\,000 \text{ cycles}}$ ↳ Ans

c) $IPC = \frac{\text{No of Instruct}^{n}}{\text{No of cycles}} = \frac{800000}{1040000} = 0.769$ Instructions Per Cycle

↳ Ans :=

For ideal pipeline, (with no stalls or delays), the number of cycles is equal to the number of instructions. If we have 'n' instructions, it takes 'n' cycles

$IPC = \frac{n}{n} = 1$ Instruction Per Cycle

↳ Ans

Q3)

a) Machine = 16-bit

256 entry cache

Tag (4-bit)	Line (8-bit)	Offset (4-bit)

$\therefore$ No of line-bits = $\log_2 256$ = 8 bits (00 to FF)

Cache line Size = 16 bytes

1 Memory address = 1 byte

So we bring in 16 memory addresses in each line

Offset bits required = $\log_2 16$ = 4 bits

Tag bits = $16 - (8+4) = 4$ bits ⟶ Ans

b)

1) 6F8E

Tag Line Offset

(Cache miss)

	Tag	Data	Valid	Dirty
F8	6	—	1	-

	Tag	Data	Valid	Dirty
2) 4F8E → F8 (cache miss)	4	—	1	—
3) 4F8D → F8 (cache hit)	4	—	1	—
4) 4FC2 → FC (cache miss)	4	—	1	—
5) 0000 → 00 (cache miss)	0	—	1	—
6) 0002 → 00 (cache hit)	0	—	1	—
7) 2F01 → F0 (cache miss)	2	—	1	—
8) 4F8E → F8 (cache hit)	4	—	1	—
9) 2F67 → F6 (cache miss)	2	—	1	—
10) 2E38 → E3 (cache miss)	2	—	1	—
11) 3F69 → F6 (cache miss)	3	—	1	—
12) 2E32 (cache hit)	2	—	1	—

Final Cache Entries

	Tag	Data	Valid	Dirty
00	0	—	1	—
⋮			0	
			⋮	
E3	2	—	1	—
F0	2	—	1	—
F6	3	—	1	—
F8	4	—	1	—
⋮				
FC	4	—	1	—
FF			⋮	
			0	

↳ Ans

c) Time in no-cache machine = $30 \times 12 = 360$ cycles

Cache hit = 1 cycle

Cache miss penalty = 30 cycle

8 cache misses & 4 cache hits

$\Rightarrow 31 \times 8 + 4 \times 1 = 248 + 4 = 252$ cycles

↳ (30 penalty + 1 cache access)

Time saved = $360 - 252 =$ 108 cycles saved

↳ Ans

d) There are 3 instances of spatial locality cache hits in this program

The 2nd instruction is 4F8E which brings the whole line (4F80 - 4F8F) into the cache memory since

the cache is of 16 byte line size.
The 3rd instruction accesses 4F8D which is now already present in the cache at line number F8. Similar case is for 0000,0002 and 2E38 & 2E32.

For temporal locality cache hit, we see that memory address 4F8E is accessed in the 2nd as well as the 8th instruction. This is an example of temporal locality because the same address is accessed in a short time span twice.

Q4) Since, we cannot jump to a 28-bit displacement constant using BEQZ, we can jump to another label where we can use the unconditional jump J to jump to our actual destination.

```
.text
main:
        # Some code here
        LI   $x, 0          # To perform the jump
        BEQZ $x, my_label

my_label:
        J    long_disp      # Assuming long_disp works like B (i.e., not handling the PC relative counter)

        # Some Code Here
        LI   $v0, 10
        syscall

.data
    long_disp  .word  0xffffffff   # 28 bit constant
```