Problem 1)
Consider the 5 stage integer pipeline with forwarding. Assume the branch penalty is 3 cycles (branch condition computed in ID). Now assume we have pipelined the memory system to three stages (rather than 1 stage) for both instruction fetch and data fetch. Branches are resolved at the end of the EX stage. We use a static branch-not-taken prediction strategy, i.e., if branches are taken we incur the branch penalty. Assume conditional branches occur with a frequency of 14%.

a. If branches are taken 62% of the time, what is the increase in CPI due to this prediction strategy?

Ans. The branch penalty is 3 cycles and incurred only when the branch is taken.
Therefore we can calculate the Increase in CPI =>
= 0.62 * 0.14 * 3 = 0.2604

b. Alternatively, if we modify the pipeline and implement a delayed branch with a single delay slot, and we are able to successfully fill 65% of the slots, what is the increase in CPI?

Ans 35% of the time we are unable to fill these slots with a penalty of 1 cycle (as now branch delay slots are implemented).
Hence increase in CPI would be
= 0.35 * 1 = .35

Problem 2)
Consider the dynamically scheduled execution of the following code sequence where a ROB buffer is used. Assume register F10 is initialized with the value 1.0 and memory locations 0(R1) and 0(R2) are initialized with 6 and 7 respectively. All other registers are initialized to 0. Consider the first iteration through the loop.

```
1. LOOP: L.D F2, 0(R1)
2. L.D F4, 0(R2)
3. MUL.D F6, F2, F4
4. SUB.D F4, F6, F10
5. DIV.D F6, F12, F2
6. S.D F6, 0(R1)
7. ADDD F8, F8, F4
8. DADDIU R1, R1, #-8
9. DADDIU R2, R2, #-8
10. BNE R1, R4 LOOP
```

a. Show a valid state of a 4 entry ROB when instruction 7 issued. Identify the head and tail of the ROB.
Ans. The following table shows the 4 entries present in the ROB. As we know that ROB holds the destination registers of the instructions and commits them in-order. The following was used to come up with the final table -

Destination Register	Value	Status
F6	-	Pending
R1	-	Pending
F8	-	Pending (TAIL)
F4	41	Done (HEAD)

b. Register re-naming is used where architecture registers are renamed to a physical registers (PR). F6 in instruction 3 is renamed to issue as PR 9. When the DIV instruction reaches the head of the ROB can PR9 be freed (i.e. can it be again used by another instruction)? Justify your answer.

Ans. Register renaming is used in Out-of-order cores to have instructions easily issue in out-of-order fashion and to avoid any register dependencies. It also helps in restoring the architecture state if the later but older instruction takes an exception. In this case, yes, PR9 can be freed once DIV instruction reaches the head of ROB. It simply means that all instructions prior to the DIV.D have committed since DIV is at the head of the ROB therefore all instructions that used the mapped register for F6 have completed. Hence, it can be freed.

3) You are studying a program that runs on a vector computer with the following latencies for various instructions: • VLD and VST: 50 cycles for each vector element; fully interleaved and pipelined. • VADD: 4 cycles for each vector element (fully pipelined). • VMUL: 16 cycles for each vector element (fully pipelined). • VDIV: 32 cycles for each vector element (fully pipelined). • VRSHF (right shift): 1 cycle for each vector element (fully pipelined).

Assume that:
• The machine has an in-order pipeline.
• The machine supports chaining between vector functional units.
• In order to support 1-cycle memory access after the first element in a vector, the machine interleaves vector elements across memory banks. All vectors are stored in memory with the first element mapped to bank 0, the second element mapped to bank 1, and so on.
• Each memory bank has an 8 KB row buffer.
• Vector elements are 64 bits in size.
• Each memory bank has two ports (so that two loads/stores can be active simultaneously), and there are two load/store functional units available.

a) What is the minimum power-of-two number of banks required in order for memory accesses to never stall? (Assume a vector stride of 1.)
Ans – 64, because memory latency is 50 cycles and the next power of two is 64

b) The machine (with as many banks as you found in part a) executes the following program (assume that the vector stride is set to 1):
VLD V1 $\leftarrow$ A
VLD V2 $\leftarrow$ B
VADD V3 $\leftarrow$ V1, V2
VMUL V4 $\leftarrow$ V3, V1
VRSHF V5 $\leftarrow$ V4, 2
It takes 111 cycles to execute this program. What is the vector length?

Ans.

```
VLD |-----50-----|---(VLEN-1)---|
VLD |1|-----50-----|---(VLEN-1)---|
VADD                              |-4-|--(VLEN-1)---|
VMUL                                                |-16-|--(VLEN-1)---|
VRSHF                                                                  |1|--(VLEN-1)--|
```

$1 + 50 + 4 + 16 + 1 + (VLEN - 1) = 71 + VLEN = 111$
=> VLEN = 40 elements

c) If the machine did not support chaining (but could still pipeline independent operations), how many cycles would be required to execute the same program?

Ans) $50 + 1 + 4 + 16 + 1 + 4 * (VLEN - 1) = 68 + 4 * VLEN = 228$ cycles

4) A five instruction sequence executes according to Tomasulo's algorithm. Each instruction is of the form ADD DR,SR1,SR2 or MUL DR,SR1,SR2. ADDs are pipelined and take 9 cycles (F-D-E1-E2-E3-E4-E5-E6-WB). MULs are also pipelined and take 11 cycles (two extra execute stages). An instruction must wait until a result is in a register before it sources it (reads it as a source operand). For instance, if instruction 2 has a read-after-write dependence on instruction 1, instruction 2 can start executing in the next cycle after instruction 1 writes back (shown below). instruction 1 |F|D|E1|E2|E3|..... |WB| instruction 2 |F|D |- |- |..... |- |E1| The machine can fetch one instruction per cycle, and can decode one instruction per cycle.

Given the following code, show the final state of the Register File, state of reservation station and register alias table after 8 cycles –

```
ADD    R7, R6, R7
ADD    R3, R6, R7
MUL    R0, R3, R6
MUL    R2, R6, R6
ADD    R2, R0, R2
```

Assume the initial state of register file is the following –

	Valid	Tag	Value
R0	1	-	4
R1	1	-	5
R2	1	-	6
R3	1	-	7
R4	1	-	8
R5	1	-	9
R6	1	-	10
R7	1	-	11

Ans. After executing the program, the final state of register file is –

	Valid	Tag	Value
R0	1	-	310
R1	1	-	5
R2	1	-	410
R3	1	-	31
R4	1	-	8
R5	1	-	9
R6	1	-	10
R7	1	-	21

The state of reservation station after 8 cycles would be –

Rename	Tag	Val	Tag	Val
A	-	10	-	11
B	-	10	A	-
C	X	-	Y	-
Adder				

Rename	Tag	Val	Tag	Val
X	b	-	-	10
Y	-	10	-	10
Z				
Mult				

The final state of register alias table after 8 cycles –

	Valid	Tag	Value
R0	0	X	4
R1	1	-	5
R2	0	C	6
R3	0	B	7
R4	1	-	8
R5	1	-	9
R6	1	-	10
R7	0	a	11

5) A 4-way write-back cache using the MESI (Modified Exclusive Shared Invalid) algorithm for cache coherency is implemented in a 4 processor core. Assume that location 1E0 is not in any cache at the start of the following sequence. Show the state (M, E, S or I) for the line containing location 1E0 in each processor cache and the state in main memory after each operation. Also note any transfers to/from memory if any occur. For example, if Px has a snoop hit of Py reading a line while it is holding a dirty copy of the line, indicate “Px writes line back, Py reads line” under the Memory Transfers column.

Sequence:
a) Processor 0 reads from location 1E0.
b) Processor 0 writes to location 1E0.
c) Processor 1 reads from location 1E0.

d) Processor 2 reads from location 1E0.
e) Processor 1 writes to location 1E0.
f) Processor 3 writes to location 1E0.

Ans. The following table gives the required information –

Action	P0	P1	P2	P3	Mem	Memory Transfer
P0 reads	E	I	I	I	Valid	P0 reads line
P0 writes	M	I	I	I	Invalid	
P1 reads	S	S	I	I	Valid	P0 writes back, P1 reads line
P2 reads	S	S	S	I	Valid	P2 reads line
P1 writes	I	M	I	I	Invalid	
P3 writes	I	I	I	M	Invalid	P1 writes back, P3 reads line