KIIT Deemed to be University
Online End Semester Examination(Autumn Semester-2020)

Subject Name & Code: HPC(CS-3010)Regular
Applicable to Courses:B.Tech/ M.Tech End Semester Examination 2020.

Full Marks=50 **Time:2 Hours**

SECTION-A(Answer All Questions. Each question carries 2 Marks)

Time:30 Minutes **(7×2=14 Marks)**

HPC (CS-3010) Solution

<table>
<tr><th>Question No</th><th>Question</th></tr>
<tr><td>Q.No:1</td><td>Question -1

Differentiate between computer architecture and organization.</td></tr>
<tr><td>ANS.</td><td>• Architecture:
– Also known as Instruction Set Architecture (ISA)
– Programmer visible part of a processor: instruction set, registers, addressing modes, etc.
• Organization:
– High-level design: how many caches?
– how many arithmetic and logic units? What type of pipelining, control design, etc.</td></tr>
<tr><td></td><td>Question -2

Differentiate between RISC and CISC processor.</td></tr>
<tr><td>ANS.</td><td>
<table>
<tr><th>CISC Processor</th><th>RISC Processor</th></tr>
<tr><td>Rich instruction set:
✓ Some simple, some very complex
Complex addressing modes:
✓ Orthogonal addressing (Every possible addressing mode for every instruction).
Many instructions take multiple cycles:
✓ Large variation in CPI
Instructions are of variable sizes
Small number of registers
Microcode control
No (or inefficient) pipelining</td><td>Small number of instructions
Small number of addressing modes
Large number of registers (>32)
Instructions execute in one or two clock cycles
Uniformed length instructions and fixed instruction format.
Register-Register Architecture:
Separate memory instructions (load/store)
Separate instruction/data cache
Hardwired control
Pipelining (Why CISC are not pipelined?)</td></tr>
</table>
</td></tr>
<tr><td></td><td>Question -3

Differentiate between true and false dependency with an example.</td></tr>
<tr><td>ANS.</td><td>True dependency :- may occur between two instructions I & J, when j attempts to read some data object that has been modified by I

Example:-</td></tr>
</table>

ADD R_1, R_2,R_3
SUB R_4,R_1,R_5

False dependency :- will be of two types anti and output dependency

Anti dependency :- It occurs between two instructions i and j, if
- j writes to a register or memory location that i reads.

Example:
ADD F_0,F_6,F_8
SUB F_8,F_4,F_5

Output dependency :- It occurs between two instructions i and j, if
- The two instructions write to the same memory location.

Example:- ADD F_6, F_0, F_8
MUL F_6, F_{10}, F_8

Question -4

Differentiate between in order and out order execution with an example.

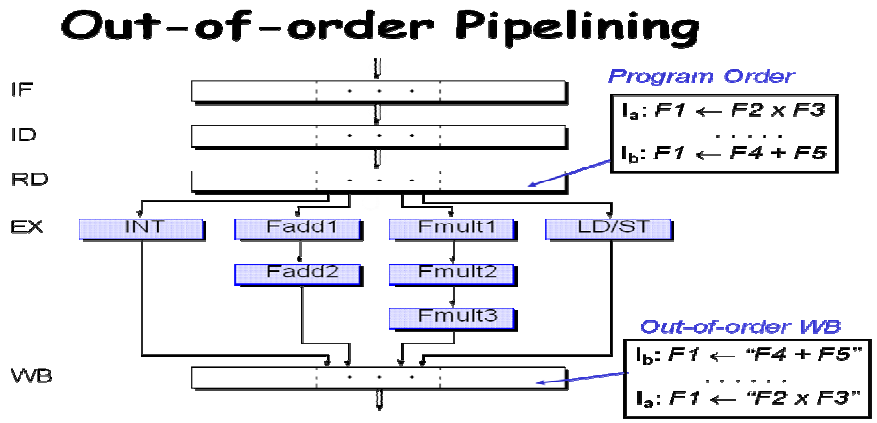
ANS.

In order execution instructions are executed in a particular sequence or order

1. If input operands are available (in processor registers, for instance), the instruction is dispatched to the appropriate functional unit. If one or more operands are unavailable during the current clock cycle (generally because they are being fetched from memory), the processor stalls until they are available.
2. The instruction is executed by the appropriate functional unit.
3. The functional unit writes the results back to the register file.

In out of order execution instruction do not maintain any sequence.

1. The instruction waits in the queue until its input operands are available. The instruction is then allowed to leave the queue before earlier, than older instructions.
2. The instruction is issued to the appropriate functional unit and executed by that unit.
3. The results are queued.
4. Only after all older instructions have their results written back to the register file, and then this result is written back to the register file.



Q.No:2

Question -1

What is pipeline interlock? Explain with examples?

ANS.

Pipeline interlock:- mechanism to detect data hazard even if operand forwarding is used
For example:- LOAD R1, 10(R2)

	SUB R4, R1,R5 **LOAD instruction has a latency that can't be eliminated by operand forwarding. This is called as pipeline inter-lock to preserve the correct execution.**
	Question -2 Consider the following MIPS assembly code: LOAD R_4, 10(R_2) LOAD R_5, 0(R_4) MUL R_7, R_5, R_4 DIV R_3, R_7, R_6 Identify each data hazard by type and list the two instructions involved in it.
ANS.	RAW :- 1st LOAD - 2nd LOAD (due to R_4) RAW :- 1st LOAD - MUL (due to R_4) RAW :- 2nd LOAD - MUL (due to R_5) RAW :- MUL – DIV (due to R_7)
	Question -3 Assume a pipeline processor has shared a single memory pipeline for data and instruction. What type of hazard it may leads to and why?
ANS.	It leads to Structural Hazard due to shared resources like data and instruction memory. If same resource is required by two (or more) concurrently executing instructions at the same time then Resource Conflict in the pipeline. (A resource conflict is a situation when more than one instruction tries to access the same resource in the same cycle.) A resource can be a register, memory, or ALU. In this case it is memory.

	Question -4
	
	Consider the following MIPS assembly code:
	DIV R_2, R_3, R_1
	SUB R_3, R_2, R_4
	LOAD R_3, 8(R_2)
	MUL R_6, R_3, R_5
	Identify each dependency by type and list the two instructions involved in it.

ANS.	True dependency :- DIV – SUB (due to R_2) Anti dependency :- DIV – SUB (due to R_3) Anti dependency :- DIV – LOAD (due to R_3) True dependency :- DIV – LOAD (due to R_2) Output dependency :- SUB – LOAD (due to R_3) True dependency :- LOAD – MUL (due to R_3)
Q.No:3	Question -1 A 7 stage pipeline separated by a clock 6ns along with a latch delay of 1ns. If the non-pipeline clock is also having the same duration and the pipeline efficiency is 50% then calculate the speed up factor. (A) 2 (B) 4 (C) 6 (D) 3
ANS.	D
	Question -2

Consider two processors P1 and P2 executing the same instruction set. Assume that under identical conditions, for the same input, a program running on P2 takes 25% less time but incurs 20% more CPI (clock cycles per instruction) as compared to the program running on P1.
If the clock frequency of P1 is 1GHz, then the clock frequency of P2 (in GHz) is _____.
(A) 1.2
(B) 3.2
(C) 1.6
(D) 0.8

ANS. C

Question -3

We have 2 designs D1 and D2 for a pipeline processor. D1 has 5 stage pipelines with execution time of 3 ns, 2 ns, 4 ns, 2 ns and 3 ns. While the design D2 has 8 pipeline stages each with 2 ns execution time. How much time can be saved using design D2 over design D1 for executing 100 instructions?(hints: assume latch delay time is 0).

A.214 ns
B .202 ns
C. 86 ns
D. 200 ns

ANS. B

Question -4

Consider an instruction pipeline with 5 stages that take 7 nsec, 4 nsc, 3 nsec, 8 nsec, and 5 nsec respectively. The delay of an inter-stage register stage of the pipeline is 2 nsec. What is the approximate speedup of the pipeline in the steady state under ideal conditions as compared to the corresponding non-pipelined implementation?

(A) 2.7
(B) 3.5
(C) 2.5
(D) 3.7

ANS. A

Q.No:4

Question -1

Differentiate between static scheduling and dynamic scheduling?

ANS.

Static scheduling

- Optimized by compiler
- When there is a stall (hazard) no further issue of instructions
- This does not allow the prediction scheme to adapt to program behavior that changes over time.

Dynamic scheduling

- enforced by hardware
– Instructions following the one that stalls can issue if they do not produce structural hazards or dependencies
- Examples Scoreboarding/Tomasulos Technique

Question -2

Explain the advantages of 1 bit prediction over 2-bit prediction method?

ANS.

1 bit prediction	2-bit prediction
Change prediction only if *once* mispredicted	Change prediction only if *twice* mispredicted
if branch alternates between taken, not	if branch alternates between taken, not

	taken:- We get 0% accuracy | taken:- We get 50% accuracy Accuracy normally depending on whether branch prediction starts with Taken (T) or Not-taken (NT)
	Question -3 What is the advantage of using VLIW instruction in Intel IA-64 processor?
ANS.	VLIW processors rely on compile time analysis to identify and bundle together instructions that can be executed concurrently. This concept is employed in the Intel IA64 processors. Advantages:- VLIW processors deploy multiple independent functional units. VLIW processors have static instruction issue capability Issue hardware is simpler. Compiler has a bigger context from which to select co-scheduled instructions. The group of instructions that could be issued in a single cycle is called: - An issue packet or a Bundle. Each "instruction" is very large – Bundles multiple operations that are independent. Complier detects hazard, and determines scheduling. There is no (or only partial) hardware hazard detection: – No dependence check logic for instructions issued at the same cycle.
	Question -4 What is the difference between SIMD and MIMD? Give examples
ANS.	SIMD (Single Instruction Multiple data) | MIMD (Multiple Instruction Multiple data) Single Control unit controls multiple processing elements | Dedicated Control unit for every processing elements Only Shared memory | May be shared (UMA) or distributed memory (NUMA) Works as Array processor | Works as multiprocessor Uses Inter-PE connection network | Uses Inter-Processor Memory connection Network
Q.No:5	Question -1 A memory system has a cache, main & virtual memory. If the hit rate of cache is 90% & the hit rate of main memory is 95%. Find out average memory access time, if it takes 10 cycles to access cache memory, 20 cycles to access main memory and 100 cycles to access virtual memory?
ANS.	$T_{avg} = H_C * T_C + (1 - H_C) * H_{MM} * (T_C + T_{MM}) + (1 - H_C) * (1 - H_{MM}) (T_C + T_{MM} + T_{VM})$ $= 0.9 * 10 + 0.1 * 0.95 * 30 + 0.1 * 0.05 * 130 = 9 + 2.85 + 0.65 = 12.5$
	Question -2 A memory system has a cache, main & virtual memory. If the hit rate of cache is 70% & the hit rate of main memory is 75%. Find out average memory access time, if it takes 5 cycles to access cache memory, 10 cycles to access main memory and 50 cycles to access virtual memory?
ANS.	$T_{avg} = H_C * T_C + (1 - H_C) * H_{MM} * (T_C + T_{MM}) + (1 - H_C) * (1 - H_{MM}) (T_C + T_{MM} + T_{VM})$ $= 0.7 * 5 + 0.3 * 0.75 * 15 + 0.3 * 0.25 * 65 = 3.5 + 3.375 + 4.875 = 11.75$
	Question -3

	A memory system has a cache, main & virtual memory. If the hit rate of cache is 80% & the hit rate of main memory is 85%. Find out average memory access time, if it takes 15 cycles to access cache memory, 25 cycles to access main memory and 80 cycles to access virtual memory?
ANS.	$T_{avg} = H_C * T_C + (1 - H_C) * H_{MM} * (T_C + T_{MM}) + (1 - H_C) * (1 - H_{MM}) (T_C + T_{MM} + T_{VM})$ $= 0.8 * 15 + 0.2 * 0.85 * 40 + 0.2 * 0.15 * 120 = 12 + 6.8 + 3.6 = 22.4$
	Question -4 A memory system has a cache, main & virtual memory. If the hit rate of cache is 75% & the hit rate of main memory is 90%. Find out average memory access time, if it takes 10 cycles to access cache memory, 15 cycles to access main memory and 60 cycles to access virtual memory?
ANS.	$T_{avg} = H_C * T_C + (1 - H_C) * H_{MM} * (T_C + T_{MM}) + (1 - H_C) * (1 - H_{MM}) (T_C + T_{MM} + T_{VM})$ $= 0.75 * 10 + 0.25 * 0.90 * 25 + 0.25 * 0.10 * 85 = 7.5 + 5.625 + 2.125 = 15.25$
Q.No:6	Question -1 Hazards are eliminated through register remaining by renaming all (A) Data Registers (B) Destination Registers (C) Source Registers (D) Memory Address registers (E) All of these
ANS.	
	Question -2 In which stage of the tomasulo approach, the output dependency has been resolved. (A) Issue Stage (B) Read operand Stage (C) Execution Stage (D) Write result stage
ANS.	A
	Question -3 Who determine the parallel schedule of the instructions in VLIW? (A) Task Scheduler (B) Interpreter (C) Data Dependency (D) Compiler (E) Job Scheduler
ANS.	D
	Question -4 Find which statement is true in case of Scoreboard approach. (A) In order issue and in order execution of the instructions. (B) Out order issue and in order execution of the instructions (C) In order issue and out order execution of the instructions. (D) Out order issue and out order execution of the instructions
ANS.	C
Q.No:7	Question -1 A pipeline system executes a program with 30% LOAD / STORE instructions. The system has

	two level memory hierarchies like cache and main memory. The cache memory hit time is 2 cycles, cache miss rate is 5% and the cache miss penalty is 50 cycles. Assume CPI without memory stalls is 1, and then calculate the improvement in CPI of the system due to the cache memory.
ANS.	30 % Load / Store instructions, memory ref/ inst= 1.3 Hit time=2 Miss rate = 5% =0.05 Miss penalty=50 CPI=1 CPU time with cache= IC×[1+0.05×1.3×50]×CT= IC×4.25×CT CPU time without cache= IC×[1.3×50]×CT= IC×65×CT So, the CPI is decreased a factor of 65/4.25=15.3 due to cache memory.
	Question -2 Consider a system with two level cache having 100 memory references and there are 10 misses in the L1 cache and 30 misses in the L2 cache. Calculate the global and local miss rates for both the caches?
ANS.	The miss rate given for L1 and L2 is logically wrong. So if student attempted the question then only give 2 marks or else not. If attempted also you can check the answer as given bellow:- The miss rate (either local or global) for the (first-level cache) L1 is 10/100 or 10%. The global miss rate of (second-level) L2 cache is 30/100 or 30%. The local miss rate of (second-level) L2 cache is 30/10.
	Question -3 A pipeline system executes a program with 40% LOAD / STORE instructions. The system has two level memory hierarchies like cache and main memory. The cache memory hit time is 5 cycles, cache miss rate is 10% and the cache miss penalty is 100 cycles. Assume CPI without memory stalls is 1, and then calculate the improvement in CPI of the system due to the cache memory.
ANS.	40 % Load / Store instructions, memory ref/ inst= 1.4 Hit time=5 Miss rate = 5% =0.1 Miss penalty=100 CPI=1 CPU time with cache= IC×[1+0.01×1.4×100]×CT= IC×2.4×CT CPU time without cache= IC×[1.4×100]×CT= IC×140×CT So, the CPI is decreased a factor of 140/2.4=58.3 due to cache memory.
	Question -4 Consider a system with two level cache having 400 memory references and there are 40 misses in the L1 cache and 80 misses in the L2 cache. Calculate the global and local miss rates for both the caches?
ANS.	The miss rate given for L1 and L2 is logically wrong. So if student attempted the question then only give 2 marks or else not. If attempted also you can check the answer as given bellow:- The miss rate (either local or global) for the (first-level cache) L1 is 40/400 or 10%. The global miss rate of (second-level) L2 cache is 80/400 or 20%.

	The local miss rate of (second-level) L2 cache is 80/40.

SECTION-B(Answer Any Three Questions. Each Question carries 12 Marks)

Time: 1 Hour and 30 Minutes **(3×12=36 Marks)**

Question No	Question
Q.No:8	**Q.No:8-1st question** a) "Amdahl's Law Quantifies overall performance gain due to improve in a part of a computation." -: Justify & prove the statement for speed up overall. Considered an enhancement to a system that improves some mode of execution by a factor of 30. Enhanced mode is used 70% of the time, measured as a percentage of execution time when the enhanced mode is in use. What is the speedup we have obtained from the first mode? What % of original execution time has been converted to fast mode? [6 Mark] Ans:- Amdahl's Law: (3 Marks) – Performance improvement gained from using some faster mode of execution is limited by the amount of time the enhancement is actually used – **FRACTION** $_{\text{ENHANCED}}$:-Fraction of the computation time in the original machine that can use the enhancement – It is always less than or equal to 1 – **SPEEDUP** $_{\text{ENHANCED}}$:-Improvement gained by enhancement that is how much faster the task would run if the enhanced mode is used for entire program. – It is always greater than 1 Speed up derivation: Ex-old= time taken without enhancement Fr-en= fraction of Ex-old where enhancement is possible. 1-Fr-en= enhancement is not possible. Ex-new=(1-Fr-en) Ex-old+ Fr-en × Ex-old/ Sp-e Overall speed up gained= Ex-old / Ex-new =1/(1- Fr-en)+ (Fr-en/ Sp-e) $$\text{Speedup}_{\text{overall}} = \frac{\text{Execution Time}_{\text{old}}}{\text{Execution Time}_{\text{new}}} \quad \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$ **Solution to the Associated Numerical** (3 Marks) Speed up gained= 1/(1-07)+(0.7/30)=1/0.323=3.09 3.09=1/(1- Fr-en)+ Fr-en/30 Fr-en=0.6999, 69.9% of original has converted.

b) A five stage pipeline processor has IF, ID, EXE, MEM, WB. The IF, ID, WB stages takes 1 clock cycles each for any instruction. The execution of different instructions takes more than ideal time as given:- 3 clock cycle for an ADD instruction, 2 clock cycles for SUB instruction, 3 clock cycles for MUL instructions, 3 clock cycles for DIV instructions and 1 clock cycles for LOAD instruction respectively.

Consider the following instructions:-

ADD R_2, R_3, R_4
LOAD $R_4, (02)R_1$
SUB R_6, R_4, R_2
LOAD $R_4, (02)R_6$
ADD R_1, R_6, R_4
MUL R_1, R_1, R_4

For the above sequence of instructions draw time and space diagram in order to find out total number of clock cycles required to complete their execution without using operand forwarding?

[6 Mark]

Ans:-

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
ADD	IF	ID	EX	EX	EX	M	WB												
LD		IF	ID	S	S	EX	M	WB											
SUB			IF			ID	S	EX	EX	M	WB								
LD						IF		ID	S	EX	M	WB							
ADD								IF		ID	S	EX	EX	EX	M	WB			
MUL										IF		ID	S	S	EX	EX	EX	M	WB

Q. No:8-2nd question

a) Derive the overall speed up gained by Amdahl's law.

Assume that we make an enhancement to a computer that improves some mode of execution by a factor of 40. Enhanced mode is used 75% of the time. Measured as a percentage of the execution time when the enhanced mode is in use. Recall that Amdahl's law depends on the fraction of the original unenhanced execution time that could make use of enhanced mode. What is the speed up we have obtained from the fast mode? What percentage of original the original execution time has been converted to fast mode?

[6 Mark]

Ans:-

Amdahl's Law: (3 Marks)

- Performance improvement gained from using some faster mode of execution is limited by the amount of time the enhancement is actually used
- **FRACTION** $_{\text{ENHANCED}}$:-Fraction of the computation time in the original machine that can use the enhancement
- It is always less than or equal to 1
- **SPEEDUP** $_{\text{ENHANCED}}$:-Improvement gained by enhancement that is how much faster the task would run if the enhanced mode is used for entire program.
- It is always greater than 1

Speed up derivation:

Ex-old= time taken without enhancement

Fr-en= fraction of Ex-old where enhancement is possible.

1-Fr-en= enhancement is not possible.

Ex-new=(1-Fr-en) Ex-old+ Fr-en × Ex-old/ Sp-e

Overall speed up gained= Ex-old / Ex-new

=1/(1- Fr-en)+ (Fr-en/ Sp-e)

$$Speedup_{overall} = \frac{Execution\ Time_{old}}{Execution\ Time_{new}} \qquad \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

Solution to the Associated Numerical (3 Marks)

Speed up gained= 1/(1-0.75)+(0.75/40)=1/0.26=3.84
3.84=1/(1- Fr-en)+ Fr-en/40
Fr-en=0.75, 75%of original has converted

b) A five stage pipeline processor has IF, ID, EXE, MEM, WB. The IF, ID, WB stages takes 1 clock cycles each for any instruction. The execution of different instructions takes more than ideal time as given:- 2 clock cycle for an ADD instruction, 2 clock cycles for SUB instruction, 4 clock cycles for MUL instructions, 4 clock cycles for DIV instructions and 2 clock cycles for LOAD instruction respectively.

Consider the following instructions:-

SUB R_1, R_2, R_4
LOAD $R_2, (02)R_1$
ADD R_7, R_1, R_2
LOAD $R_4, (02)R_7$
DIV R_1, R_7, R_4
MUL R_1, R_7, R_4

For the above sequence of instructions draw time and space diagram in order to find out total number of clock cycles required to complete their execution using operand forwarding?

[6 Mark]

Ans.:-

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
SUB	IF	ID	EX	EX	M	W														
LD		IF	ID	S	EX	EX	M	W												
ADD			IF		ID	S	EX	EX	M	W										
LD					IF		ID	EX	EX	M	W									
DIV							IF	ID	S	S	EX	EX	EX	EX	M	W				
MUL								IF			ID	S	S	S	EX	EX	EX	EX	M	W

Q. No:8-3rd question

a) Derive the overall speed up gained by Amdahl's law. Assume that we make an enhancement to a computer that improves some mode of execution by a factor of 40. Enhanced mode is used 75% of the time. Measured as a percentage of the execution time when the enhanced mode is in use. Recall that Amdahl's law depends on the fraction of the original unenhanced execution time that could make use of enhanced mode. What is the speed up we have obtained from the fast mode? What percentage of original the original execution time has been converted to fast mode?

[6 Mark]

Ans:-

Amdahl's Law: (3 Marks)

- Performance improvement gained from using some faster mode of execution is limited by the amount of time the enhancement is actually used
- **FRACTION** $_{\text{ENHANCED}}$:-Fraction of the computation time in the original machine that can use the enhancement
- It is always less than or equal to 1
- **SPEEDUP** $_{\text{ENHANCED}}$:-Improvement gained by enhancement that is how much faster the task would run if the enhanced mode is used for entire program.
- It is always greater than 1

Speed up derivation:
Ex-old= time taken without enhancement
Fr-en= fraction of Ex-old where enhancement is possible.
1-Fr-en= enhancement is not possible.
Ex-new=(1-Fr-en) Ex-old+ Fr-en × Ex-old/ Sp-e
Overall speed up gained= Ex-old / Ex-new
=1/(1- Fr-en)+ (Fr-en/ Sp-e)

$$Speedup_{overall} = \frac{Execution\ Time_{old}}{Execution\ Time_{new}} \qquad \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$$

Solution to the Associated Numerical (3 Marks)

Speed up gained= 1/(1-0.75)+(0.75/40)=1/0.26=3.84

3.84=1/(1- Fr-en)+ Fr-en/40

Fr-en=0.75, 75%of original has converted

B) Find out the total no of clock cycles required to execute the following instructions without and with operand forwarding?

LD R_1, 0(R_2)
LD R_1, 0(R_3)
DADDIU R_1,R_1,#1
SD R_1, 0(R_2)
DADDIU R_2,R_2,#4
DSUB R_4,R_3,R_2

[6 Mark]

Ans:-

Without operand forwarding=16

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LD	IF	ID	EX	M	W											
LD		IF	ID	EX	M	W										
DADD			IF	ID	S	S	EX	M	W							
SD				IF			ID	S	S	EX	M	W				
DADD							IF			ID	EX	M	W			
DSUB										IF	ID	S	S	EX	M	W

With operand forwarding=11

	1	2	3	4	5	6	7	8	9	10	11			
LD	IF	ID	EX	M	W									
LD		IF	ID	EX	M	W								
DADD			IF	ID	S	EX	M	W						
SD				IF		ID	EX	M	W					
DADD						IF	ID	EX	M	W				
DSUB							IF	ID	EX	M	W			

Q.No:9

Q.No:9-1st question

a) Draw and explain the different operational steps to execute instructions with Scoreboard approach to achieve higher ILP.

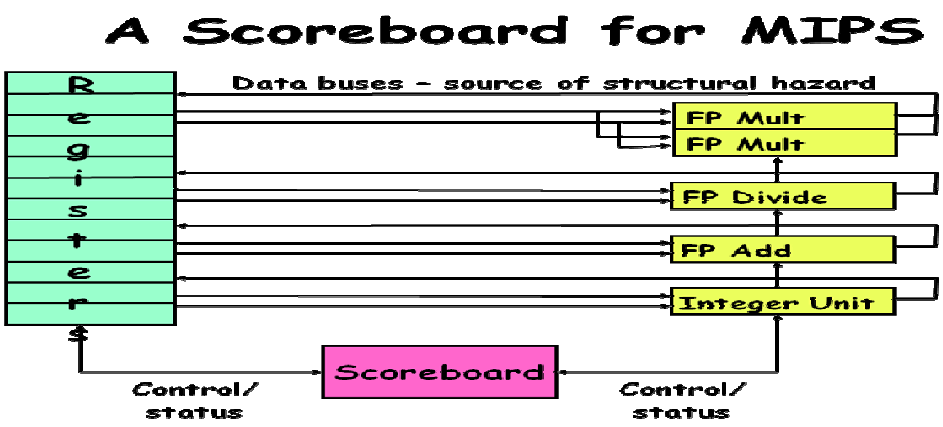
[6 Mark]

Ans:-

4 Steps of Execution with Scoreboarding **(2 marks)**

1. Issue: when a functional unit for an instruction is free and no other active instruction has the same destination register:
   - Avoids structural and WAW hazards.
2. Read operands: when all source operands are available:
   - Operand Forwarding not used.
   - A source operand is available if no earlier issued active instruction is going to write it.
   - Thus resolves RAW hazards dynamically.
3. **Execution:** begins when the functional unit receives its operands; scoreboard notified when execution completes.
4. **Write Result:** after WAR hazards have been resolved.

Diagram (2 marks)



Different parts of Scoreboard (2 marks)

- **Instruction Status Unit :-**
  - **Indicates which of 4 phases the instruction is in**
- **Functional Status Unit :-**
  - **Indicates the states of functional unit & each functional unit has 9 different fields.**
    1. **BUSY:- Indicates weather the functional unit is busy or free.**
    2. **OP :- Indicates the type of operation to be perform in the unit**
    3. **F_i :- Indicates the destination register**
    4. **F_j, F_k :- Indicates the source registers**
    5. **Q_j, Q_k :- Indicates the functional units producing source registers value**
    6. **R_j, R_k :- Flags indicating when are ready & not yet read**
       1. **IF operand is ready & not yet read then à YES**
       2. **IF operand is not ready then à NO**
       3. **IF operand is already read then à NO**
- **Register Result Status Unit :-**
  - **Indicates which functional unit will write each register, if an active instruction has the register as its destination.**
  - **The field is set to blank whenever there are no pending instruction that will write that register.**

b) Consider the following instructions:-

ADD R_1, R_9, R_3
LOAD $R_2, 0(R_1)$
MUL R_3, R_2, R_4
LOAD $R_5, 3(R_6)$
ADD R_6, R_3, R_5
DIV R_7, R_3, R_6
SUB R_1, R_6, R_3

Consider following execution status for above instruction set:-

- First ADD instruction is ready to go for its write result phase.
- Other remaining instructions are in their issue phase.

Show the status of instruction, functional unit & register using scoreboard approach (with two integers, three multipliers, one divider & three adder units).

[6 Mark]

Ans.

Instruction Status Table:

Instruction	Issue	Read operands	Execution	Write result
ADD R1,R9,R3	✓	✓	✓	
LOAD R2, O(R1)	✓			

MUL R3,R2,R4	✓			
LOAD R5,3(R6)	✓			
ADD R6,R3,R5	✓			
DIV R7,R3,R6	✓			
SUB R1,R6,R3	✓			

FUNCTIONAL UNIT STATU TABLE

Name	Busy	Operation	F_i	F_j	F_k	Q_j	Q_k	R_j	R_k
Integer1	Yes	Load	R2	R1		ADDER1		NO	
Integer2	Yes	Load	R5	R6				YES	
Mul1	Yes	Multiplication	R3	R2	R4	INTEGER1		NO	YES
Mul2	No								
Mul3	No								
Div1	Yes	Division	R7	R3	R6	MUL1	ADDER2	NO	NO
Adder1	Yes	Addition	R1	R9	R3			NO	NO
Adder2	Yes	Addition	R6	R3	R5	MUL1	INTIGER2	NO	NO
Adder3	Yes	Subtraction	R1	R6	R3	ADDER2	MUL1	NO	NO

REGISTER RESULT STATUS TABLE:

R1	R2	R3	R5	R6	R7
ADDER1, ADDER3	INTEGER1	MUL1	INTEGER2	ADDER2	DIVIDER1

Q. No:9-2nd question

a) Draw and explain different operational steps execute instructions with tomasulo approach to achieve higher ILP.

[6 Mark]

Ans.

Discuss about reservation station, load and store buffer, CDB. **(2 marks)**

- Reservation stations:
  - Single entry buffer at the head of each functional unit has been replaced by a multiple entry buffer.
- Common Data Bus (CDB):
  - Connects the output of the functional units to the reservation stations as well as registers.

Three Stages of Tomasulo Algorithm

Issue: Get instruction from Instruction Queue

- Issue instruction only if a matching reservation station is free (no structural hazard).
- Send registers or the functional unit that would produce the result (achieves renaming).
- If there is a matching reservation station that is empty, issue the instruction to the station (otherwise structural hazard) with the operand values, if they are currently in the registers (otherwise keep track of FU that will produce the operand & WAR, WAW hazard).

Execute: Operate on operands (EX)

- If one or more of the operands is not yet available, monitor the common data bus while waiting for it to be computed.
- When both operands ready then execute;
  if not ready, watch Common Data Bus for result
- RAW hazard are avoided.

Write result: Finish execution (WB)

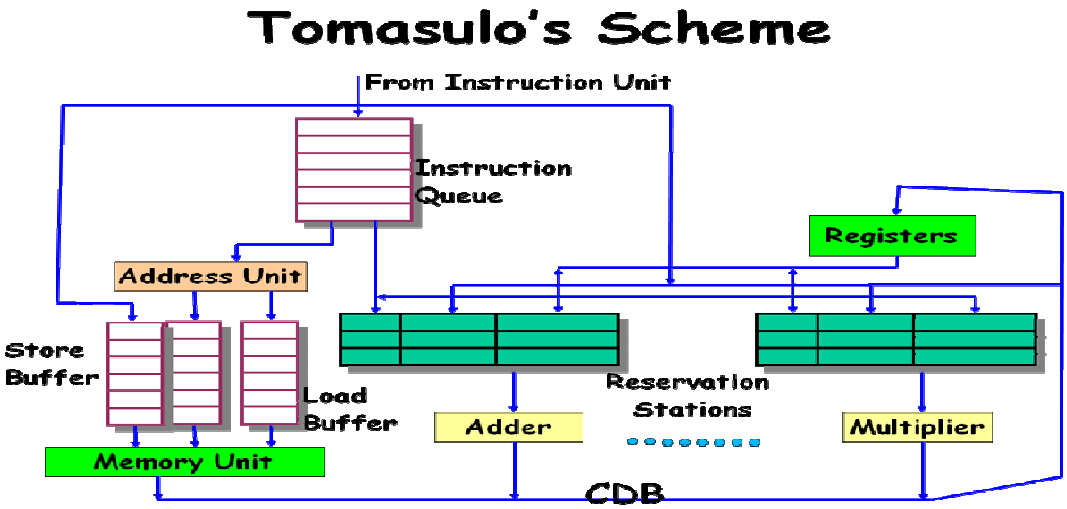
- When result is available, write it on the CDB and from there into the registers and into the reservation stations waiting for this result.
- Write on CDB to all awaiting units; mark reservation station available.

Different parts of Tomasulo (2 marks)

- **Instruction Status Unit :-**
  - **Indicates which of 3 phases the instruction is in**
- **Reservation station Status Unit :-**
  - **Indicates the states of reservation station & each reservation station has 7 different fields.**

1) **BUSY:- Indicates weather the reservation station and its accompanying functional unit is busy or free.**
2) **OP :- Indicates the type of operation to be perform in the reservation station.**
3) **V_j, V_k :- Indicates the value of source operands (hold the offset field).**
4) **Q_j, Q_k :- Indicates the reservation station that will produce the source operand.**
5) **A:- Use the hold information for the memory address calculation for the load and store.**

Diagram (2 marks)



b) Consider the following MIPS instructions:-

SUB R_2, R_1, R_3
ADD R_4, R_2, R_5
LOAD $R_1, 10(R_4)$
LOAD $R_4, 8(R_6)$
MUL R_7, R_5, R_4
DIV R_6, R_7, R_4

Consider following execution status for above instruction set:-

- SUB instruction is ready to go for its write result phase.
- ADD instruction is to go for its execution phase.
- Other remaining instructions are in their issue phase.

Show the status of instruction, reservation station, & register result status using dynamic scheduling with tomasulo approach (Assume we have 3 multipliers, 2 adders & 3 loader units).

[6 Mark]

Ans.:- (Each table have 2 marks)

Instruction Status Table:

Instruction	Issue	Execute	Write result
SUB R2,R1,R3	✓	✓	

ADD R4, R2, R5	✓		
LOAD R1, 10(R4)	✓		
LOAD R4,8(R6)	✓		
MUL R7, R5, R4	✓		
DIV R6,R7, R4	✓		

RESERVATION STATION

NAME	BUSY	OPERATION	V_J	V_K	Q_J	Q_K	A
MUL1	YES	MUL	R5			LOAD2	
MUL2	YES	DIV			MUL1	LOAD2	
MUL3	NO						
ADDER1	YES	SUB	Reg[R1]	Reg[R3]			
ADDER2	YES	ADD		Reg[R5]	ADDER1		
LOAD1	YES	LOAD					10+REG[R4]
LOAD2	YES	LOAD					8+REG[R6]
LOAD3	NO						

REGISTER STATUS TABLE

FIELD	R1	R2	R4	R6	R7
Qi	LOAD1	ADDER1	LOAD2	MUL2	MUL1

Q. No:9-3rd question

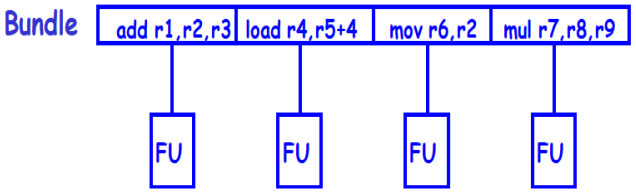
a) Compare between VLIW processor and superscalar processor.

[6 Mark]

Ans.;-

VLIW architecture, basic principle and diagram. (3 marks)

- VLIW:
  - Let complier take the complexity.
  - Simple hardware, smart compiler.
  - The compiler has complete responsibility of selecting a set of instructions:-These can be concurrently be executed.
  - VLIW processors have static instruction issue capability
  - One single VLIW instruction:- separately targets differently functional units.
  - VLIW - Compiler finds parallelism
  - Give example



Schematic Explanation for a VLIW Instruction

Super scalar Processor architecture, basic principle and diagram. (3 marks)

- Superscalar processors:
  - Multiple issue, dynamically scheduled, speculative execution, branch prediction
  - More hardware functionalities and **complexities.**
  - superscalar processors have dynamic issue capability
  - Superscalar – hardware finds parallelism
  - Give example with diagram

b) The following set of MIPS instruction is going to be executed in a pipelined system.

ADD R_2, R_1, R_3

MUL R_4, R_2, R_5
LOAD $R_1, 10(R_4)$
LOAD $R_4, 8(R_6)$
SUB R_7, R_5, R_4
DIV R_6, R_2, R_4
ADD R_4, R_6, R_7

Consider following execution status for above instruction set:-

- ADD instruction is completed its write result phase.
- MUL instruction is in its execution phase.
- DIV instruction is ready to go for its write result phase.
- Other remaining instructions are in their issue phase.

Show the status of instruction, reservation station, & register result status using dynamic scheduling with tomasulo approach (Assume we have 4 multipliers, 4 adders & 3 loader units).

[6 Mark]

Ans:- (Each table have 2 marks)

INSTRUCTION STATUS TABLE :-

Instruction	Issue	Execute	Write result
ADD R2,R1,R3	✓	✓	✓
MUL R4,R2,R5	✓	✓	
LOAD R1,10(R4)	✓		
LOAD R4,8(R6)	✓		
SUB R7,R5,R4	✓		
DIV R6,R2, R4	✓		
ADD R4,R6,R7	✓		

RESERVATION STATION:-

NAME	BUSY	OPERATION	V_J	V_K	Q_J	Q_K	A
MUL1	YES	MUL	Reg[R2]	Reg[R5]			
MUL2	YES	DIV	R2			LOAD2	
MUL3							
MUL4							
ADD1	NO						
ADD2	YES	SUB	Reg[R5]			LOAD2	
ADD3	YES	ADD			MUL2	ADD2	
ADD4							
LOAD1	YES	LOAD					10+REG[R4]
LOAD2	YES	LOAD					8+REG[R6]
LOAD3							

REGISTER RESULT STATUS TABLE:-

FIELD	R1	R4	R6	R7
Qi	LOAD1	ADD3	MUL2	ADD2

Q.No:10

Q.No:10-1st question

a) Derive the CPU performance equation? How clock-rate based performance measurement measures -"despite of a lower MIPS rate a processor is faster"- Explain With example.

[6 Mark]

Ans:-

MIPS mean millions of instructions per second. Unlike clock rate, MIPS provide some idea of the work actually performed. However, even MIPS is not a good indication of performance because not all instructions perform the same amount of computation. A higher MIPS rating in many cases may not mean higher performance or better execution time. i.e. due to compiler design variations.

CPU Performance Equation (3 Marks)

- Computers using a clock running at a constant rate.
- CPU time = CPU clock cycles for a program X
  clock cycle time
  = CPU clock cycles for a program / Clock Rate

CPI can be defined in terms of No. of clock cycles & instruction count.
(IPC α CPI)

$$CPI = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}}$$

- CPU time = Instruction count X clock cycle time X cycle per instruction.

$$\text{CPU time} = \frac{\text{Instruction}}{\text{Program}} \times \frac{\text{Clock cycle}}{\text{Instruction}} \times \frac{\text{Second}}{\text{Clock cycle}}$$

$$\text{CPU clock cycle} = \sum_{i=1}^{n} IC_i \times CPI_i$$

Where :-
IC_I → No. of times instructions i is executed in a program.

CPI_I → Average No. of clock pre instructions.

$$CPI_{overall} = \frac{\sum_{i=1}^{n} IC_i \times CPI_i}{\text{Instruction Count}}$$

Example:- (3 Marks)
Example can be taken for a program which is mix of different class of Instructions with different CPI with a common clock rate. Two different compilers can be chosen which will show different instruction counts for each class of instructions for the same program .Then CPU time can be calculated deriving the MPIS for each compiler which will finally show that Higher MIPS rating is always not high performance.

Let's take the following example:

- Consider the following computer:

Instruction counts (in millions) for each instruction class

Code type-	A (1 cycle)	B (2 cycle)	C (3 cycle)
Compiler 1	5	1	1
Compiler 2	10	1	1

The machine runs at 100MHz.

Instruction A requires 1 clock cycle, Instruction B requires 2 clock cycles, Instruction C requires 3 clock cycles.

$$CPI = \frac{\text{CPU Clock Cycles}}{\text{Instruction Count}} = \frac{\sum_{i=1}^{n} CPI_i \times N_i}{\text{Instruction Count}}$$

Solution:-

$$CPI_1 = \frac{[(5\times1) + (1\times2) + (1\times3)] \times 10^6}{(5 + 1 + 1) \times 10^6} = 10/7 = 1.43$$

(count, cycles)

$$MIPS_1 = \frac{100\ MHz}{1.43} = 69.9$$

$$CPI_2 = \frac{[(10\times1) + (1\times2) + (1\times3)] \times 10^6}{(10 + 1 + 1) \times 10^6} = 15/12 = 1.25$$

$$MIPS_2 = \frac{100\ MHz}{1.25} = 80.0$$

So, compiler 2 has a higher MIPS rating and should be faster?

- Now let's compare CPU time:

Note important formula!

$$\text{CPU Time} = \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

$$\text{CPU Time}_1 = \frac{7 \times 10^6 \times 1.43}{100 \times 10^6} = 0.10 \text{ seconds}$$

$$\text{CPU Time}_2 = \frac{12 \times 10^6 \times 1.25}{100 \times 10^6} = 0.15 \text{ seconds}$$

Therefore program 1 is faster despite a lower MIPS!

Hence, it can be proved that with higher MIPS even the CPU time is high, thus Performance is less.

b) Explain loop unrolling with an example.

[6 Mark]

Ans:-

Loop unrolling:- (2 Marks)

Loop unrolling is a loop transformation technique that helps to optimize the execution time of a program. We basically remove or reduce iterations. Loop unrolling increases the program's speed by eliminating loop control instruction and loop test instructions.

Example: - **(4 Marks)**

Adding a scalar to a vector (loop is parallel since the body of each iteration is independent)

```
for(i=1000;i>0;i--)
{
  a[i]=a[i]+c;
}
```

```
Loop:  L.D     F0,0(R1)
       ADD.D   F4,F0,F2
       S.D     F4,0(R1)
       DADDUI  R1,R1,#-8
       BNE     R1,R2,Loop
```

Assume the functional unit latencies as shown in the bellow given table.

Instruction Producing Result	Instruction Using Result	Latency
LOAD	FP ALU	1
FP ALU	FP ALU	3
FP ALU	STORE	2
INT. ALU	Branch Instruction	1
LOAD	STORE / Int. ALU	0

STORE	Any / FP ALU / INT. ALU	0

After Introducing Latency

```
LOOP :  LOAD    F0, 0(R1)
        STALL
        ADD     F4, F0, F2
        STALL
        STALL
        STORE   F4, 0(R1)
        ADD     R1, R1, # -8
        STALL
        BNE     R1, R2, LOOP
        NOP
```

Total of 10 clock cycles used

After Re-ordering Instruction

```
LOOP :  LOAD    F0, 0(R1)
        ADD     R1, R1, # -8
        ADD     F4, F0, F2
        STALL
        BNE     R1, R2, LOOP
        STORE   F4, 8(R1)
```

Total of 6 clock cycles used for each iteration

Assuming $R_1 - R_2$ (that is, the size of the array) is initially a multiple of 32.

After using **Loop Unrolling**

```
Loop:  L.D      F0,0(R1)
       ADD.D    F4,F0,F2
       S.D      F4,0(R1)
       L.D      F6,-8(R1)
       ADD.D    F8,F6,F2
       S.D      F8,-8(R1)
       L.D      F10,-16(R1)
       ADD.D    F12,F10,F2
       S.D      F12,-16(R1)
       L.D      F14,-24(R1)
       ADD.D    F16,F14,F2
       S.D      F16,-24(R1)
       DADDUI   R1,R1,#-32
       BNE      R1,R2,Loop
```

After Re-ordering Instruction as per the latency table

```
LOOP :-  LOAD    F0, 0(R1)
         LOAD    F6, -8(R1)
         LOAD    F10, -16(R1)
         LOAD    F10, -24(R1)

         ADD     F4, F0, F2
         ADD     F8, F6, F2
         ADD     F12, F10, F2
         ADD     F16, F14, F2

         STORE   F4, 0(R1)
         STORE   F8, -8(R1)
         ADD     R1, R1, # -32
         STORE   F4, 24(R1)
         BNE     R1, R2, LOOP
         STORE   F4, 8(R1)
```

Total of 3.5 clock cycles used for each iteration

Q.No:10-2nd question

a) Draw the state transition to explain the implementation of snooping protocol considering both CPU requests & bus requests for each cache block access.

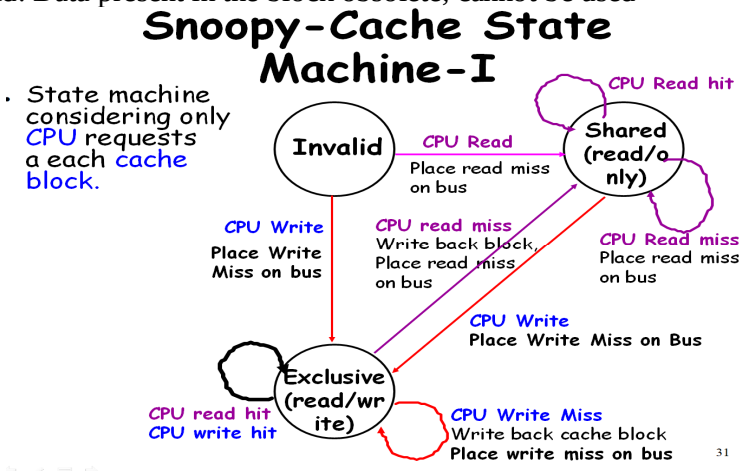
[6 Mark]

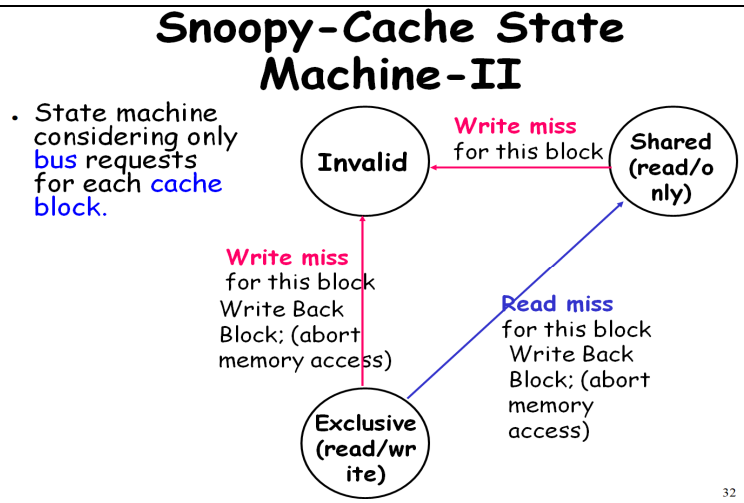
Ans:-

Snoopy Protocols

- Essentially two types:
  - Write Invalidate Protocol
  - Write Broadcast Protocol
- Write invalidate protocol:
  - When one processor writes to its cache, all other processors having a copy of that data block invalidate that block.
- Write broadcast:
  - When one processor writes to its cache, all other processors having a copy of that data block update that block with the recent written value.
- Assume:
  - Invalidation protocol, write-back cache.
- Each block of memory is in one of the following states:
  - Shared: Clean in all caches and up-to-date in memory, block can be read.
  - Exclusive: cache has the only copy, it is writeable, and dirty.

  Invalid: Data present in the block obsolete, cannot be used





b) What are the approaches to get higher ILP? Explain, how higher ILP can be achieved during dynamic scheduling? Explain loop unrolling with an example.

[6 Mark]

Ans:-

Two basic approaches: (1Mark)

1) I rely on hardware to discover and exploit parallelism dynamically,
2) I rely on software to restructure programs to statically facilitate parallelism.

Dynamic Scheduling:- (2 Marks)

1) some basics of scoreboard (Refer to the answer of Question No. 9 1st Question [a]
2) some basics of Tomasulo (Refer to the answer of Question No. 9 2nd Question [a]

Loop unrolling:- (3 marks) Refer to the answer of Question No. 10 1st Question [b]

Q.No:10-3rd question

a) What is cache coherence problem? Explain the state transition diagram for the CPU in the directory based method used to handle the cache coherence problem.

[6 Mark]

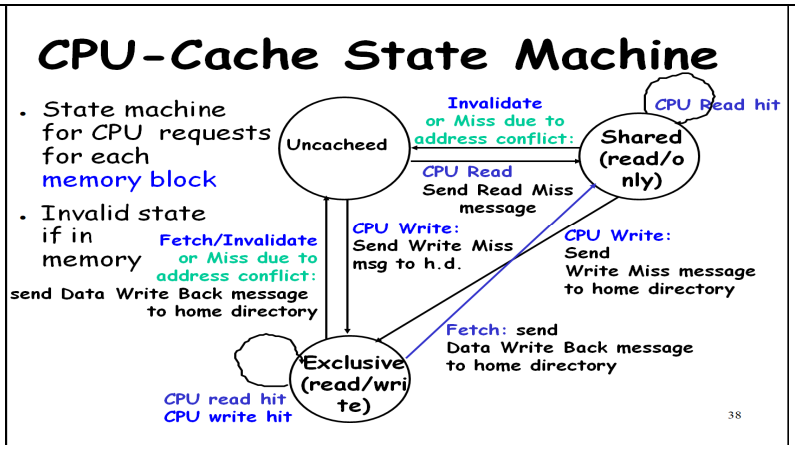
Ans:- Cache coherence (1 mark)

Cache coherence is a concern raised in a multi-core system distributed L1 and L2 caches. The Cache Coherence Problem is the challenge of keeping multiple local caches synchronized when one of the processors updates its local copy of data which is shared among multiple caches. Such as

- When shared data are cached:
  - These are replicated in multiple caches.
  - The data in the caches of different processors may become inconsistent.

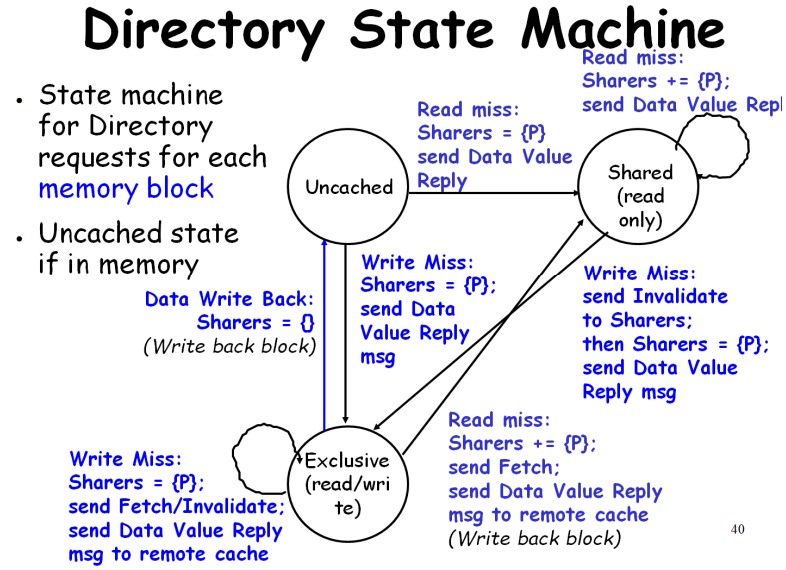
Directory Protocol (5 mark)

- Three states as in Snoopy Protocol
  - Shared: 1 or more processors have data, memory is up-to-date.
  - Uncached: No processor has the block.
  - Exclusive: 1 processor (owner) has the block.
- In addition to cache state,
  - Must track which processors have data when in the shared state.
  - Usually implemented using bit vector, 1 if processor has copy.



State Transition Diagram for the Directory

- Tracks all copies of memory block.
- Same states as the transition diagram for an individual cache.
- Memory controller actions:
  - Update of directory state
  - Send messages to satisfy requests.
  - Also indicates an action that updates the sharing set, Sharers, as well as sending a message.



b) What is static loop unrolling? Explain loop unrolling with an example. Write different merits and demerits of loop unrolling.

[6 Mark]

Ans:-

Loop unrolling:- (4 marks) Refer to the answer of Question No. 10 1st Question [b]

Merits and demerits of loop unrolling (2 marks)

- Unrolling Merits
  - It improves the performance by eliminating overhead instructions.
- Unrolling Limitations
  - Decrease in the amount of overhead amortized with each unroll.
  - Growth in cache size due to loop unrolling may increase cache miss rates.
  - Shortfall of registers created by aggressive unrolling register renaming and scheduling.
  - Significant increase in complexity of compilers.

Q.No:11

Q.No:11-1st question

a) What is cache miss penalty time? Explain how to improve cache memory performance by reducing cache miss penalty time.

[6 Mark]

Ans:-

Cache miss penalty time :- (1 mark)

Refers to the extra time required to bring the data into cache from the Main memory whenever there is a "miss" in cache.

Explain Technique for Reducing Miss Penalty time :- (5 Marks)

- Multilevel caches
- Priority to Read Misses over Writes
- Victim caches
- Critical word first

b) Differences between no-write allocate and write allocate method during cache write miss.

Assume a fully associative write back cache with many cache entries that starts empty. Below is a sequence of memory operations (the address is in square brackets).

WriteMem [500]
ReadMem [400]
WriteMem [100]
ReadMem [200]
ReadMem [400]
WriteMem [500]
WriteMem [200]
ReadMem [300]
WriteMem [200]

What are the numbers of hits and misses when using no-write allocate Vs write allocate?

[6 Mark]

Ans:-

Write Allocate Vs No-write Allocate :- (3 mark)

- Write allocate: - A block allocated in cache.
- No-write allocate:- no block allocation, but just written to in main memory.

- In no-write allocate,
  - Only blocks that are read from can be in cache.
  - Write-only blocks are never in cache.
- But typically:
  - write-allocate used with write-back
  - no-write allocate used with write-through

Associated Numerical:- (3 mark)

Memory Reference	No-write Allocate	Write Allocate
WriteMem [500]	MISS	MISS
ReadMem [400]	MISS	MISS
WriteMem [100]	MISS	MISS
ReadMem [200]	MISS	MISS
ReadMem [400]	HIT	HIT
WriteMem [500]	MISS	HIT
WriteMem [200]	HIT	HIT
ReadMem [300]	MISS	MISS
WriteMem [200]	HIT	HIT
	Total of 3 HIT & 6 MISS	Total of 4 HIT & 5 MISS

Q.No:11-2nd question

a) What is average memory access time (ATMT)? Explain how to improve cache memory performance by reducing cache hit time.

[6 Mark]

Ans:-

Average Memory Access Time (AMAT) (2 Marks)

- AMAT:- The average time it takes for the processor to get a data item it requests.
- The time it takes to get requested data to the processor can vary
  - due to the memory hierarchy.

$AMAT = Hit\ Time_{L1} + Miss\ Rate_{L1} \times Miss\ Penalty_{L1}$

Explain Techniques for Reducing Cache Hit Time (4 Marks)

- Small & Simple Caches
- Avoiding Address Translation
- Pipelining Writes

b) A computer system consists of three level memory organizations. The capacity of direct mapped L1 cache memory is of 4096 bytes where as the 8-way set associative L2 cache memory is of 512 Kbytes. The access time of L1 and L2 cache is 10 nsec and 20 nsec respectively. The Main memory access time is 200 nsec. The hit rate of L1 cache is 75%. Calculate the hit ratio for L2 cache memory, if the average memory access time is 50 nsec?

[6 Mark]

Ans:-

$T_{avg} = H_{L1} * T_{L1} + (1 - H_{L1}) * H_{L2} * (T_{L1} + T_{L2}) + (1 - H_{L1}) * (1 - H_{L2}) (T_{L1} + T_{L2} + 8*T_{MM})$

8-way set

- ⇨ $50 = 0.75 * 10 + 0.25 * H_{L2} * 20 + 0.25 * (1 - H_{L2}) * 1630$
- ⇨ $H_{L2} = 0.9068$
- ⇨ $H_{L2} = 90\ \%$ (Approx)

Q. No:11-3rd question

A) Differentiate between message passing models and distributed shared memory architecture? Explain cache coherence in symmetric shared memory multiprocessor with an example.

[6 Mark]

Ans:- Difference between message passing models and DSM (3 Marks)

Message Passing Model

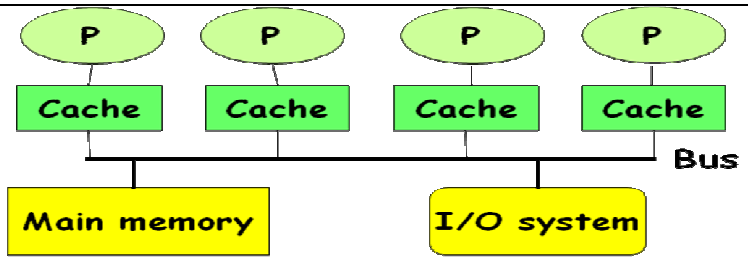
- Explicit message send and receive instructions have to be written by the programmer.
  - Send: specifies local buffer + receiving process (id) on remote computer (address).
  - Receive: specifies sending process on remote computer + local buffer to place data.

Distributed Shared-Memory Architecture (DSM)

- Physically separate memories are accessed as one logical address space.
- Processors running on a multi-computer system share their memory.
  - Implemented by operating system.
- DSM multiprocessors are NUMA:
  - Access time depends on the exact location of the data.

Symmetric Multiprocessors (SMPs)

- SMPs are a popular shared memory multiprocessor architecture:
  - Processors share Memory and I/O
  - Bus based: access time for all memory locations is equal --- “Symmetric MP”



Cache Coherence (3 Mark)

- When shared data are cached:
  - These are replicated in multiple caches.
  - The data in the caches of different processors may become inconsistent.

Explain Cache Coherence with example

b) Assume two cache memory specifications. A split cache which consists of 64KB instruction cache [with 60% of instruction reference access] and 64KB Data cache [with 40% of data reference access]) or a 128KB unified cache. The cache memory hit time is of 2 clock cycles, miss penalty time is of 50 clock cycles.

For the above cache memory specifications a Simulator showed the following conditions like:-

1) 60 miss per 1000 instructions for data cache.
2) 10 miss per 1000 instructions for instruction cache.
3) 70 miss per 1000 instructions for the unified cache.

If there are 30% Load/Store instructions then calculate the average memory access time (AMAT) for split and unified cache memory.

[6 Mark]

Ans:-

Miss rate = (misses/instructions)/(mem accesses/instruction)

Instruction cache miss rate= (10/1000)/1.0=0.01

Data cache miss rate = (60/1000)/0.3 =0.2

Unified cache miss rate = (70/1000)/1.3 =0.05

AMAT (split cache) = 0.6 * (1 + 0.01 * 50) + 0.4 (1 + 0.2 * 50) = 0.9 + 4.4 = 5.3

AMAT (Unified) = 0.6 * (1 + 0.05 * 50) + 0.4 * (1 + 1 + 0.05 * 50)

= 2.1 + 1.8 = 3.9