	Course Name:	Computer Architecture	Course Code:	EE204
	Program:	BS (Computer Science)	Semester:	Fall2018
	Duration:	3 hours	Total Marks:	70
	Paper Date:	31-12-2018	Weight	45
	Exam Type:	Final	Page(s):	7

Student : Name:_____ Roll No._____ Section: _____

Instruction:
1. Attempt all questions in the provided space. You can use rough sheets but it should not be attached.
2. If you think some information is missing, write assumption and solve accordingly.

Question 1 [10 marks] (Multiple correct options can be selected in each question)

1. A decoder having 64 outputs will have _____ inputs.
   A. 4
   B. 6
   C. 8
   D. 64
2. A single cycle machine has separate instruction and data memory in order to avoid _____.
   A. Branch Hazard
   B. Data Hazard
   C. Structural Hazard
   D. Memory Hazard
3. The total number of bits needed for a cache is a function of
   A. Data bits, Tag bits and index bits
   B. Data bits, Block size, and valid bit
   C. Tag bits, index bits and Data Block size
   D. Data bits, Tag bits, and valid bit
4. What are the side effects of increasing the block size to a much larger value
   A. Use of spatial locality principle decreases
   B. The number of blocks in cache decreases
   C. Cost of cache miss increases
   D. All of the above
5. In a write-back policy, the updated data is written to
   A. Cache only
   B. Cache and main memory simultaneously
   C. Memory only
   D. Cache and Buffer

6. Suppose 10% of the instructions are stores and write through policy is used to handle writes. If the CPI without cache misses was 1.0, What will be the CPI if we spend 100 extra cycles on every write to the memory?
   Answer: ____CPI of 1.0 + 100 x 10 % = 11_____
7. The type of hazard that occurs when executing instructions cannot access hardware simultaneously are called ____structurual hazards_____
8. Cause register in case of exception records _____cause of the exception_____

9. If we have compiler that can produce code by rearranging instructions and inserting NOPs, we don't need underlying hazard detection and forwarding hardware. True/False?
10. In virtual memory, size of the physical address is always greater than the virtual address because more space can be accessed through main memory and the disk collectively. True/False?

Question 2

Part a: [1 + 1 + 1 + 1 marks]

Consider the following code snippet's execution on a 5-stage pipelined processor with hazard detection and forwarding fully implemented.

Instruction 1: Or R1,R1,R3
Instruction 2: Lw R2, 20(R2)
Instruction 3: Lw R3, 0(R1)
Instruction 4: sw R2,40(R1)
Instruction 5: Add R4,R5,t4
Instruction 6: Add R4,R5,R5
Instruction 7: Add R4,R6,R6

Suppose the execution of the code starts in the first clock cycle.

1. Is the following condition true in 5th clock cycle?
   MEM/WB.RegisterRd = ID/EX.RegisterRs True/False
   Reason: R1=R1

2. Is the following condition true in 5th clock cycle?
   MEM/WB.RegisterRd = ID/EX.RegisterRt True/False
   Reason: R1 !=R3

   Is the following condition true in 6th clock cycle?
   MEM/WB.RegisterRd = ID/EX.RegisterRs True/False
   Reason: R2 != R1

   Is the following condition true in 6th clock cycle?
   MEM/WB.RegisterRd = ID/EX.RegisterRt True/False
   Reason: R2 = R2

Part b: [2+2 marks]

Consider the following facts for the page table and answer the questions below.

a) Each page table entry consists of a physical page number, 1 valid bit, 1 dirty bit
b) Virtual addresses are 32 bits - Physical addresses are 26 bits - The page size is 8 Kbytes

1. How many pages a process can have? 2^19 pages

2. What is the size of the page table? 2^19 * 15 bits

Part c: [5 marks]

Considering two code sequences that require the following instruction counts

	Code Sequence 1	**Code Sequence 2**
Load Instructions Count	5	4
Branch Instructions Count	2	2
ALU Instructions Count	10	15

Following details are provided for the processor

1. Full forwarding and hazard detection unit is implemented
2. Branch decision hardware is implemented in the decode stage and the forwarding unit has been modified for forwarding data from earlier instructions to the decode stage
3. Branch prediction hardware always predicts the branch as taken. Penalty for misprediction is 1 clock cycle. Branch prediction accuracy is 80%
4. All load instructions are immediately followed by the ALU instructions that use the loaded data
   Example: lw R1, 0(R2)
   add R2, R1, R3

Compute average CPI (including stalls) for Code sequence 1 and 2. Show all steps of the calculations.

CPI for load instructions = 2
CPI for branch instructions = 2 * 20% + 1 * 80% = 1.2
CPI for ALU instructions = 1

For code Sequence 1: CPI = 2*5 + 1.2*2+1*10 = 22.4/17 = 1.32
For code Sequence 1: CPI = 2*4 + 1.2*2+1*15 = 25.4/21 = 1.20

Question 3 [8+11+8]

Code	Instructions

FOR: ld R1,100(R6)
ld R2,200(R6)
add R5,R2,R1
ld R4,0(R5)
addi R6,R6,4
and R4,R5,R4
st R4,0(R5)
beq R6,R7, FOR
	o We have 5-stage MIPS pipelined processor where Memory and Branch instructions use all 5 stages however ALU instructions use 4 stages: no cycle in memory and 1 cycle in all other 4 stages. Branch instructions consumes 1 cycle in each stage. Memory type instructions take 2 cycles in memory and 1 cycle in all other stages.
	o Instructions can be completed out of order. Instructions are fetched and decoded in order however execution, memory and writeback can be out of order.
	o An instruction will only enter the execution stage if it does not cause a ReadAfterWrite, WriteAfterRead or WriteAfterWrite hazard.
	o There is only one unit in each stage so only one instruction can enter a stage at a time, and in case multiple instructions are ready, the oldest one will go first.
	o Full Forwarding is implemented
	o There is no register renaming

a. Fill in the following table pointing for each instruction, the pipeline stages activated in each clock cycle. For example instruction 1 is fetched in 1st cycle, decoded in 2nd as shown below

Cycle / Inst	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
(1)	F	D																												
(2)																														
(3)																														
(4)																														
(5)																														
(6)																														
(7)																														
(8)																														

b. Unroll the loop for level 2 (two iterations only), add stalls and then reschedule to remove as many stalls as possible. For this part ignore structural hazards. You have to consider only data and control hazards.

Code after loop unrolling	**Code with added stalls**	**Optimized schedule of instruction**

c. Consider the given program (used in part a) to be executed on 2-issue superscalar processor with following specification.

- There are two generic arithmetic execution units
- Memory type instructions take 2 cycles in memory and arithmetic instruction does not use memory stage.
- Full Forwarding is implemented
- In case of false dependencies, write back should be in order. No need to wait in decode stage

Show single iteration of the given code

Cycle No.	IF		ID		EX		MEM		WB	
1										
2										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										

a. Fill in the following table pointing for each instruction, the pi
cycle. For example instruction 1 is fetched in 1st cycle, decod

Cycle / Inst	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
(1)	F	D	e	m	M	W												
(2)		F	D	e	-	m	m	W										
(3)			F	D	-	-	-	e	W									
(4)				F	D	-	-	-	e	m	m	W						
(5)					F	D	e	-	-	W								
(6)						F	D	-	-	-	-	e	W					
(7)							F	D	e	-	-	-	m	m	W			
(8)								F	D	e	M	-	-	W				

. Unroll the loop for level 2 (two iterations only), add stalls and then reschedule to remove as many stalls as possible. For this part ignore structural hazards. You have to consider only data and control hazards.

2 ll 4 5

Code after loop unrolling	Code with added stalls	Optimized schedule of instruction
ld-1 ld-1 add-1 ld-1 and-1 St-1 beq ld-2 ld-2 add-2 ld-2 and-2 St-2 addi R6, R6, 8 beq R6, R7, FOR	ld-1 ld-1 ∞ ∞ add-1 ld-1 ∞ ∞ and-1 St-1 ld-2 ld-2 ∞ ∞ add-2 ld-2 ∞ ∞ and-2 St-2 addi beq ∞	ld-1 ld-1 ld-2 ld-2 add-1 ld-1 add-2 ld-2 and-1 St-1 addi- and-2 beq- St-2
2 marks for this column.	4 marks for this column.	5 marks for this column

7/8 stalls

e. Consider the given program (used in part a) to be executed on 2-issue superscalar processor wit following specification.

- There are two generic arithmetic execution units
- Memory type instructions take 2 cycles in memory and arithmetic instruction does not memory stage.
- Full Forwarding is implemented
- In case of false dependencies, write back should be in order. No need to wait in decode

Show single iteration of the given code

Cycle No.	IF		ID		EX		MEM		WB	
1	1	2								
2	3	4	1	2						
.	5	6	3	4	1	2				
.	5	6	3	4	-	-	1	2		
.	5	6	3	4	-	-	1	2		
.	7	6	5	4	3	-	-	-	1	2
.		8	7	6	4	5	-	-	3	
.			7	6	-	-	4	-	5	
.			7	6	-	-	4	-	-	
.			7	8	6	-	-	-	4	
.					7	8	-	-	6	
.							7	8	8	
.							7		7	
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										
.										

Q4: Consider a memory sub-system—with 16-bit word-size and addresses—that has two levels of cache (L1 and L2). L1 is a direct-mapped cache with 16 bytes block size and cache capacity of 4 KB. L2 is an 8-way set associative cache with 256 bytes block size and cache capacity of 32 KB. (20 marks)

a. Calculate the size of tag, index, and offset fields for the L1 cache (3 marks)

Number of tag bits	4 bits
Number of index bits	8 bits
Number of offset (word + byte) bits	4 bits (3 + 1)

b. Calculate the size of tag, index, and offset fields for the L2 cache (3 marks)

Number of tag bits	4 bits
Number of index bits	4 bits
Number of offset (word + byte) bits	8 bits (7 + 1)

c. A test application attempts to read the following memory address (16-bit addresses). List all generated events that might take place for each read access. Possible events include L1-Hit, L1-Miss, L2-Hit, L2-Miss, and Mem (for memory read). Both (10 marks)

Note that memory blocks are only transferred between adjacent memory levels. For eviction of blocks, Least Recently Used (LRU) approach is used by L1 and L2 caches. The caches are initially empty.

The following cases (see below) generate the following events (see below). Use this information to fill the table in the right-box.

Cases	Generated Events
L1 Hit	L1-Hit
L2-Hit	L1-Miss, L2-Hit
Memory Access	L1-Miss, L2-Miss, Mem

Memory Address (to read from)	Generated Events (L1-Hit, L1-Miss, L2-Hit, L2-Miss, and Mem)
0x1562	L1-Miss, L2-Miss, Mem
0x1588	L1-Miss, L2-Hit
0x1581	L1-Hit
0x1562	L1-Miss, L2-Hit
0x4562	L1-Miss, L2-Miss, Mem
0x45BC	L1-Miss, L2-Hit
0x4562	L1-Miss, L2-Hit
0x1562	L1-Miss, L2-Hit
0x45BC	L1-Hit
0x45B2	L1-Hit

Assume that the:
Clock rate is 2 GHz,
L1 access time is 1 cycle,
L2 access time is 10 cycles,
Memory access time is 100 cycles,
L1 hit rate is 60%,
L2 hit rate is 70%.

What is the average memory access time? (4 marks)

AMAT = Time for a hit + Miss Rate x Miss Penalty

$AMAT_1 = HitTime_1 + MissRate_1 \times (MissPenalty_1)$
$MissPenalty_1 = HitTime_2 + MissRate_2 \times (MissPenatly_2)$

Since the clock rate is 2 Ghz, clock cycle time is 0.5 ns.
$AMAT_1 = 5\ ns + 0.4\ (MissPenalty_1)$

$MissPenalty_1 = 5\ ns + .3(50ns) = 20\ ns$

$AMAT_1 = 5ns + .4\ (20ns) = 8.5\ ns$