


National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Computer Architecture	Course Code:	EE204
	Program:	BS(Computer Science)	Semester:	Fall 2019
			Total Marks:	50
	Due Date:	02-12-2019	Weight	~3.3
	Evaluation Type:	Assignment 3	Page(s):	4

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

Question 1[8+8+9]

Code	Instructions
1. Loop: ld R4, 0(R1) 2. mul R4, R4, R0 3. ld R5, 0(R2) 4. add R4, R4, R5 5. sw R4, 0(R2) 6. addi R1, R1, 4 7. addi R2, R2, 4 8. bne R2, 0, Loop	<ul style="list-style-type: none"> ○ 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 3 cycles in memory and 1 cycle in all other stages. ○ Instructions can be completed out of order. Instructions are fetched and decoded in order however execution, memory and writeback can be out of order. ○ An instruction will only enter the execution stage if it does not cause a ReadAfterWrite ○ Write back should be in order in case of false dependency. There is no register renaming ○ 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. ○ Full Forwarding is implemented

- 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.
- o There are two generic arithmetic execution units
 - o Memory type instructions take 3 cycles in memory and arithmetic instruction does not use memory stage.
 - o Full Forwarding is implemented
 - o In case of false dependencies, write back should be in order. No need to wait in decode stage

Show two iterations of the given code

[illegible]

Question 2 [10]

Suppose we have a cache that has the following division of 32-bit address:

bits 0 - 3 = offset
bits 4 - 14 = index
bits 15 - 31 = tag

- What would be the total data size of the cache if above figures are considered for a direct mapped cache?
- What would be the total data size of the cache if above figures are considered for 4-way set associative cache?
- Calculate the number of sets for 2-way set associate cache.
- How many total bits of storage (tag+data +valid) are required if it is considered as an 8-way set associate cache
- In case of an 8-way set associate cache, Into what set would bytes with each of the following addresses be stored

Set number (decimal)

0001 1011 1111 0001 0001 0001 1011 1111

0001 1100 1111 0011 1111 0011 0100 1011

Question 3 [15]

You need to design a memory system that has 64 bit address bus and 32 bit data bus. The memory system will have 2 cache levels.

- L1 cache is a direct-mapped cache containing 128 MB data and has a line/block size of 256 bits
- L2 cache is an 8-way set associative cache containing 512 MB data and has a line/block size of 256 bits.

Design the memory system clearly showing the inter connection between the 2 caches and memory. You should label the tag, index and offset bits clearly with their relevant sizes. Also calculate the size of tag array for each cache.