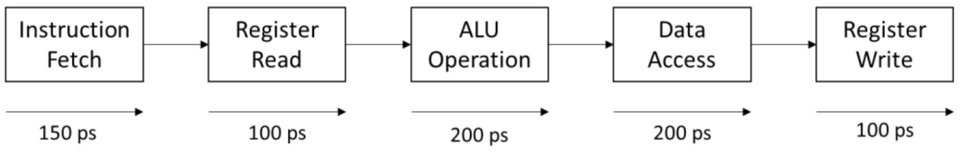
CSC 137 SP22 FINAL EXAM

Show your work clearly to earn points. Work not shown will not earn any points.
Date: May 19, 2022 Instructor: Dr. Ilkan Çokgör Total: 35 points

Student Name: Student Number:

1) Instruction Set Architecture:
Consider the datapath and the execution timing given below.

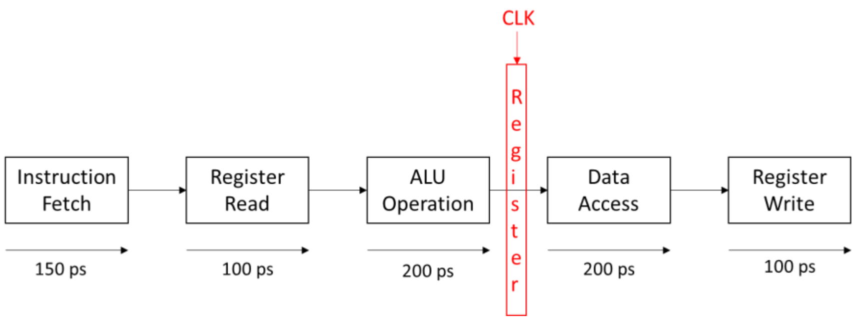


a) What is the overall latency (execution time)? (1 point)
b) What is the throughput (instructions per second)? (1 point)
c) Turn this datapath into a <u>two-stage</u> pipelined execution so that the throughput is maximized (assume there is no execution stage conflicts; i.e. no need to consider the hazards.) Show on the drawing above how you turn it into a pipelined datapath. Justify your answer by calculating the latency and throughput of the pipelined execution. (2 points)

1.a) Latency: 150 + 100 + 200 + 200 + 100 = 750 ps

1.b) Throughput: 1/750ps = 1.3GHz

1.c)



Two stage pipeline:
Smallest achievable latency: 150 + 100 + 200 = 450 ps.
The clock period: 450ps
Throughput: 2.2GHz

2) Instruction Set Architecture:
Consider the instruction execution for a five-stage pipelined processor for all instructions below where the 'register read' and 'register write' do not conflict. The computer is a von Neumann architecture (only one data and address bus for memory access). Each stage takes 300 ps.

Instruction Fetch	Register Read	ALU Operation	Data Access	Register Write

You are given the following instructions:

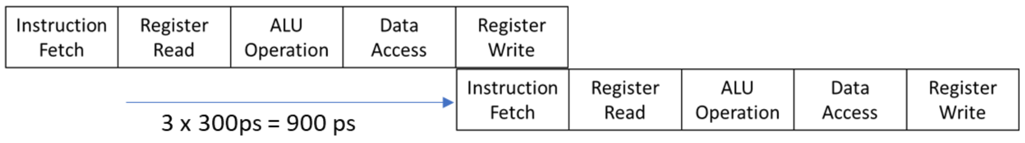
ADD R_d, R_1, R_2	// $R_1 + R_2 \rightarrow R_d$;
LDR R_d, [R_a, #m]	//Load data to R_d from the memory address $R_a + m$
STR R_d, [R_a, #m]	//Store the contents of R_d to the memory address $R_a + m$

R_d, R_1 and R_2 can be any register between R0 to R8.

a) Write a program segment so that a data hazard occurs in the pipeline. Show how this hazard would occur by drawing the pipeline. How long do you have to stall the pipeline as a result of this hazard? (2 points)
b) Write a program segment so that a structural hazard occurs in the pipeline. Show how this hazard would occur by drawing the pipeline. How long do you have to stall the pipeline as a result of this hazard? (2 points)

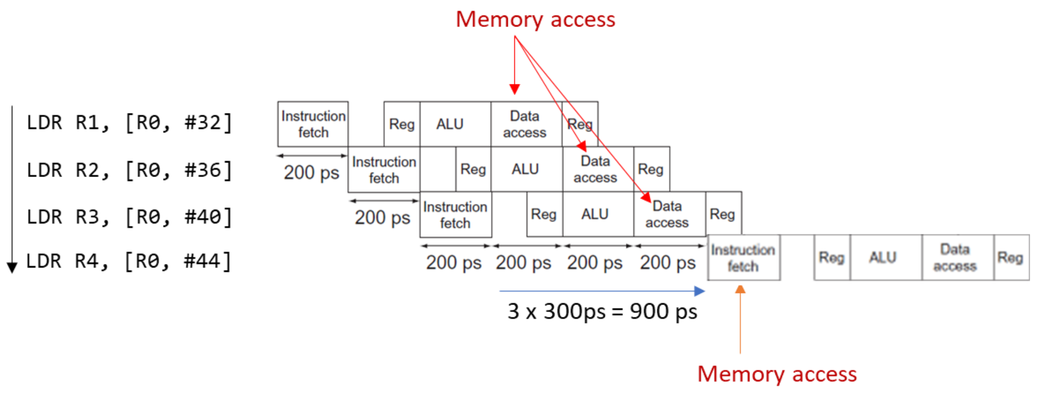
2.a) There can be more than one correct answer. One possible answer is:
Data hazard:

```
ADD R9, R0, R1
ADD R2, R9, R3
```



(This is a simplified picture. The pipeline would be stalled after instruction fetch. The stall time is the same).

2.b)) There can be more than one correct answer. One possible answer is:
Structural hazard:



```
LDR R1, [R0, #32]
LDR R2, [R0, #36]
LDR R3, [R0, #40]
LDR R4, [R0, #44]
```

3) Cache Memory:
You are working with an 8-bit processor and an 8-bit memory.

a) Create a 4-block direct mapped instruction cache where the block size is a single word. Show all cache columns as well as the index for each block. (2 points)
b) What is the size of this cache in bytes (round up to the nearest integer byte)? (2 points)
c) The following program segment is located at memory address 0x40 (i.e. the first instruction is at address 0x40). Each instruction takes up a single byte in memory. What will be the miss rate of this cache when the program runs? Assume the cache is initially empty. (2 points)

```
      MOV R0, #2        // 2 → R0
loop: ADD R2, R2, 1     // R2 + 1 → R2
      STR R1, [R2, #1]  // Store contents of R1 to memory address R2+1
      SUB R0, R0, 1     // R0 – 1 → R0; i.e. decrement R0
      BNE loop          // Go to loop if R0 not zero; i.e. instructions
                        // that form the loop run total of 2 times
```

d) What will the cache contents look like after the above program has finished running? (2 points)
e) If each instruction takes 2 clock cycles to run, and each main memory access takes 100 clock cycles, what will be the effective clock cycles per instruction after the program above runs the first time? (2 points)
f) What would be the miss rate after the program above has run if the block size were 2 words? Assume the cache is initially empty. (2 points)
g) What would the contents of this 2-word block size cache look like after the above program has run? (2 points)

3.a) 8-bit address with last 2 bits used for index; hence, 6 bits are used for the Tag field.

	1 bit	6 bit	8 bit
Index	V	Tag	Data
00			
01			
10			
11			

3.b) Cache size:
1+6+8 = 15 bits per cache word. 15 bits x 4 = 60 bits. 60/8 = 7.5 Bytes -> 8 bytes

3.c) The memory looks like as follows: 0x40 = 0100 0000

Instruction	Memory Address
MOV R0, #2	0100 0000
ADD R4, R4, 1	0100 0001
STR R1, [R4, #1]	0100 0010
SUB R0, R0, 1	0100 0011
BNE loop	0100 0100

As the program runs there will be 5 misses (note that the BNE instruction replaces the MOV instruction in cache). See 3.d below for the cache contents. There are total of 9 instructions that run.
Miss rate: 5 / 9 = 55.5%

3.d)

	1 bit	6 bit	8 bit
Index	V	Tag	Data
00	1	0100 01	BNE loop
01	1	0100 00	ADD R4, R4, 1
10	1	0100 00	STR R1, [R4, #1]
11	1	0100 00	SUB R0, R0, 1

3.e)

(5 x 100) + (9 x 2) = 518 total clock cycles
518/9 = 57.5 clock cycles per instruction

OR, for N number of total instructions:
(N x 55.5% x 100) + (N x 2) = 57.5 N total clock cycles
57.5 N / N = 57.5 clock cycles per instruction

3.f and g) Block size 2 words => 1 bit block offset, 2 bit index. # of misses: 3. Miss rate: 3 / 9 = 33.3%

	1 bit	5 bit	8 bit 0	8 bit 1
Index	V	Tag	Data	Data
00	1	0100 0	MOV R0, #2	ADD R4, R4, 1
01	1	0100 0	STR R1, [R4, #1]	SUB R0, R0, 1
10	1	0100 0	BNE loop	
11	0			

4) Virtual Memory:
   a) What is the total memory capacity of a 16-bit processor in kilobytes? (1 point)
   b) Assume that you only have 1KB of DRAM in your system. If the page size is 32 Bytes, how many virtual pages do you have available? (2 point)
   c) How many pages can be stored in DRAM? (1 point)
   d) Assuming that you are running a 16-bit proprietary operating system, will your programs run faster if you add another 1KB memory? Justify your answer. (2 points)

4.a) Virtual Memory capacity: 2^{16} = 64Kbyte

4.b) Page size: 32 Bytes =2^5 => 5 bits for page offset

16 bits

Virtual page number: 11 bits	Page offset: 5 bits

# of Virtual pages: 2^{11} = 2048

4.c) Physical Memory capacity: 1Kbyte = 2^{10}

10 bits

Physical page number: 5 bits	Page offset: 5 bits

# of Physical pages: 2^5 = 32

4.d) If the physical memory is doubled, the number of physical pages will be 2^6 = 64. More physical memory will result in fewer number of page faults. The programs will run faster.

5) System Interconnection:

   a) You are connecting five water sensors to an embedded computer. These sensors alert the computer when the water level they are monitoring reaches a certain level. You would like to make sure that the processor can dedicate its full time on running other programs when it is not alerted by the sensors. What type of I/O (polled, interrupt or DMA) would be the most efficient to implement? Justify your answer. (2 points)

   b) Can a DRAM and an Ethernet port have the same address in memory mapped I/O? Why or why not? (1 point)

   c) What is an interrupt vector? (1 point)

   d) Where is the interrupt vector table located in a computer? (1 point)

   e) You need to refresh a full HD video screen (1920 x 1080 pixels) periodically. The microprocessor sends a word to the video screen that corresponds to each pixel on the screen. What type of I/O (polled, interrupt or DMA) would be the most efficient to implement? Justify your answer. (2 points)

5.a) Interrupt I/O is the most efficient I/O type to implement in this example. The processor can dedicate its full time on running other programs when it is not interrupted by the sensors. Polled I/O would take up processor time since the processor needs to check each sensor periodically and very frequently. DMA would not be appropriate (would be too costly and too slow) since there isn't a great amount of data being transferred.

5.b) No, a DRAM and an Ethernet port cannot have the same address in memory mapped I/O. Since each peripheral would have a memory address in memory mapped I/O, memory locations and peripherals would have their own unique addresses.

5.c) An interrupt vector contains the address of the Interrupt Service Routine.

5.d) The interrupt vector table is located in the main memory.

5.e) Direct Memory Address (DMA) would be the most efficient type of I/O to implement in this example. The processor needs to transfer 1920 X 1080 words each time the video screen is refreshed. Since the amount of data transfer is quite large, the computer can utilize a DMA controller to execute this task independent of the processor.