Department of Electrical and Computer Engineering
The University of Texas at Austin

ECE 460N Fall 2024
Instructor: Yale N. Patt
TAs: Anna Guo, Nadia Houston, Logan Liberty, Luke Mason, Abhay Mittal, Asher Nederveld, Edgar Turcotte
Exam 1
October 9, 2024

Name: ____

Problem 1 (15 points): _______

Problem 2 (25 points): _______

Problem 3 (30 points): _______

Problem 4 (30 points): _______

Total (100 points): ______________

Note: Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Note: Please be sure your name is recorded on each sheet of the exam.

Please read the following sentence, and if you agree, sign where requested:
I have not given nor received any unauthorized help on this exam.

Signature: ____

GOOD LUCK!

Question 1 (15 points): Two five-stage pipelined processors are shown in the table below. The amount of time needed to process the work in each stage is shown in the table.

	Fetch	Decode	Execute	Memory	Writeback
Processor A	200ns	250ns	200ns	500ns	100ns
Processor B	400ns	150ns	100ns	300ns	150ns

Part a (2 points): What is the minimum cycle time for each processor? How long does it take for a single instruction to be carried out by each processor?

	Cycle time	Time to process an instruction
Processor A	**500ns**	**5 * 500ns = 2500ns**
Processor B	**400ns**	**5 * 400ns = 2000ns**

Part b (4 points): A program that has no control instructions and no stalls due to dependencies is executed by each processor. The number of instructions in the program is large. What is the average number of instructions executed per ns for each processor?

Processor A	**1/500 instr/ns**	Processor B	**1/400 instr/ns**

Part c (5 points): Suppose Processor B needs to execute 10% more instructions than Processor A for a particular program. Assuming the answers from (b), which processor provides higher performance? Please show your work.

Using the performance equation from class:
Performance = 1 / (length * CPI * cycle time)
Processor A: Performance = 1 / (100 instr * 1 CPI * 500ns) = 1/50000
Processor B: Performance = 1 / (110 instr * 1 CPI * 400ns) = 1/44000
Since 1/44000 > 1/50000, Processor B provides higher performance.

PROBLEM CONTINUES ON NEXT PAGE

Part d (2 points): Suppose you can split one of the pipeline stages of each processor into 2 stages, each requiring half of the time of the full stage. Which pipeline stage would you select for each processor and why?

Processor A stage:	**Memory**	Processor B stage:	**Fetch**

Explain:

These are the longest stages for each processor. Splitting these stages can reduce the cycle time for each processor.

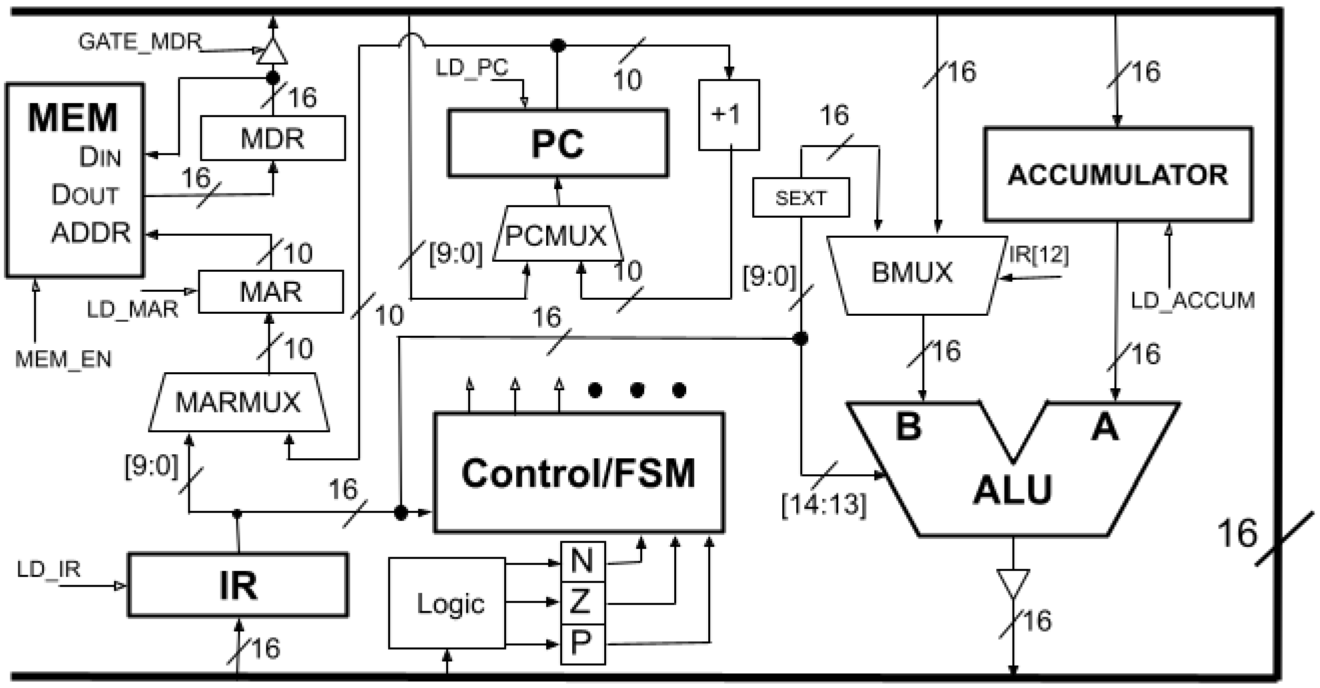
Part e (2 points): For your selections in (d), what is the minimum cycle time and how long does it take for a single instruction to be carried out by each processor?

	Cycle time	Time to process an instruction
Processor A	**250ns**	**6 * 250ns = 1500ns**
Processor B	**300ns**	**6 * 300ns = 1800ns**

Question 2 (25 points) A Texas A&M student is designing his first computer for his graduate thesis. Unfortunately, he was never taught how to make a datapath or use microcode, so he made many mistakes when creating the datapath based on the ISA below.

Instruction	Opcode [15:13]	[12]	[11]	[10]	[9:0]	Description
ADD [MEM]/Imm*	000	Steer	0	0	Addr/Imm10	Accum = Accum + M[Addr]/Imm10
AND [MEM]/Imm*	001	Steer	0	0	Addr/Imm10	Accum = Accum **&** M[Addr]/Imm10
OR [MEM]/Imm*	010	Steer	0	0	Addr/Imm10	Accum = Accum | M[Addr]/Imm10
XOR [MEM]/Imm*	011	Steer	0	0	Addr/Imm10	Accum = Accum ^ M[Addr]/Imm10
JMP Addr	100	0	0	0	Addr	PC = Addr
LD [MEM]*	101	0	0	0	Addr/~~Imm10~~	Accum = MEM[Addr]
ST [MEM]	110	0	0	0	Addr	MEM[Addr] = Accum
BR cc Addr	111	N	Z	P	Addr	If condition met, PC = Addr, else PC+=1

*These instructions set condition codes.



Answer the questions on the next page. Here are some important details of the ISA and the datapath:

- He made no mistakes in specifying the ISA.
- The steering bit, if present, selects between a value from memory at the specified address and a sign-extended immediate value.
- The memory uses a 10 bit address, and can complete an access in one clock cycle.
- The ALU supports these operations: ADD, AND, OR, XOR.

Given 2kb memory size, 10 bit address, what is Addressability?
How many architectural registers are there? Is this machine 0-address, 1-address, 2-address, or 3-address?

Addressability: **2KB / (2^10) = 2 bytes or 16 bits addressability**	**Number of architectural registers:** **1 (The accumulator)**	**#-Address machine:** **1**

Part b (5 points): This is the student's current list of control signals generated by the control store: LD_ACCUM, LD_IR, LD_PC, LD_BEN, LD_MAR, GATE_MDR, MEM_EN, IRD, COND.
Given the data path, list 5 of the missing control signals.

1 pt per correct answer: MARMUX, PCMUX, MemR_W, Set_CC, GateALU, LD_MDR, J

These will generally detract from score:
Incorrect: ALUK, BMUX, R (ALU already has select lines from the opcode)

Some exceptions are made for examples such as this:

- **Add a control signal specifically to allow the ALU to PASSA**

Part c (15 points): Other than missing control signals, there are some major issues in the data path. Find 3 different mistakes that cause ISA features to not be properly supported. Keep each explanation to fewer than 20 words, and identify the instruction(s) that are affected by each mistake.

3 correct issues get full credit, 5 pts each. There were more than 3 possible issues, and issues not listed here will also get credit if reasonable.	
When using memory addressed operands, RAM and ALU have to drive the BUS at the same time.	**Instruction(s) affected:** **ADD, AND, OR, XOR**
There is no way for Addr to get to PC	**Instruction(s) affected:** **BR, JMP**
You cannot write to memory because you can't load from bus to MDR	**Instruction(s) affected:** **ST**
You can't output accumulator to the bus, because there is no PASS A or gate to BUS. **There was not meant to be an immediate addressing mode on LD [MEM] in the ISA, but that is also not supported here.**	**Instruction(s) affected:** **ST**

Question 3 (30 points): The C standard library has a string compare function, strcmp(), that compares two null-terminated ASCII strings in memory. Your job is to implement STRCMP as an instruction in the LC-3b.

Recall that a string is an array of one-byte characters, ending in a null terminator, which has the value 0x00.

The STRCMP instruction is to operate as follows: Starting with the first character in each string (N=0), if the Nth character of string A is equal to the Nth character of string B, and both characters are not the null terminator, set N to N+1 and repeat. If the Nth character of string A is not equal to the Nth character of string B, STRCMP outputs the difference between their ASCII values. If the Nth character of both strings are the null terminator, then STRCMP outputs 0.

Additionally, here is a C implementation of strcmp() to help you:

```
int strcmp(char *ptr_A, char *ptr_B) {
     char c1, c2;
     int diff;
     int N = 0;
     do {
          c1 = ptr_A[N];
          c2 = ptr_B[N];
          diff = c1 - c2;
          N++;
     }
     while(diff == 0 && c1 != 0);
     return diff;
}
```

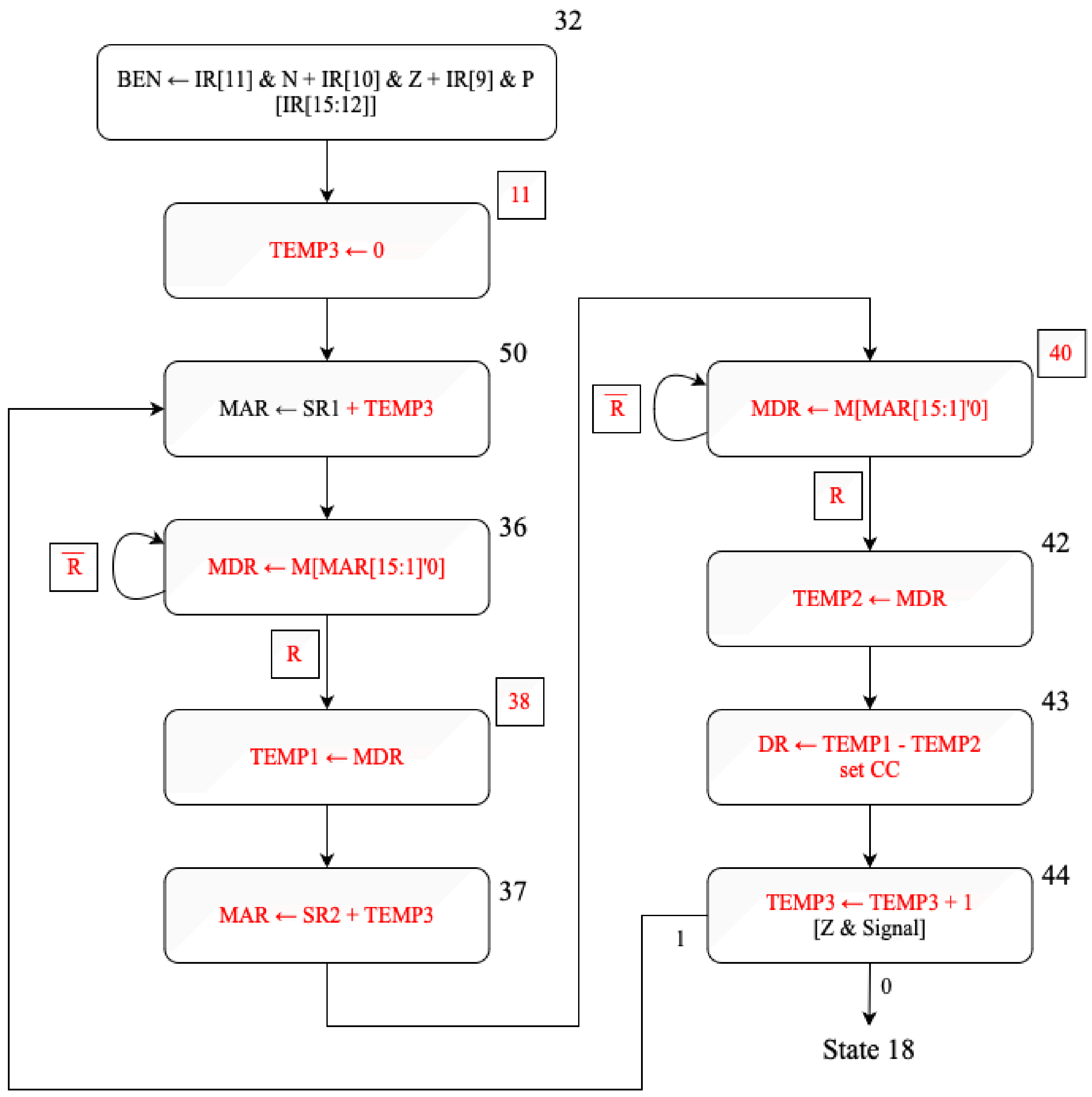
The instruction encoding is shown below. We will use the unused opcode 1011.

1	0	1	1	DR	SR1	000	SR2

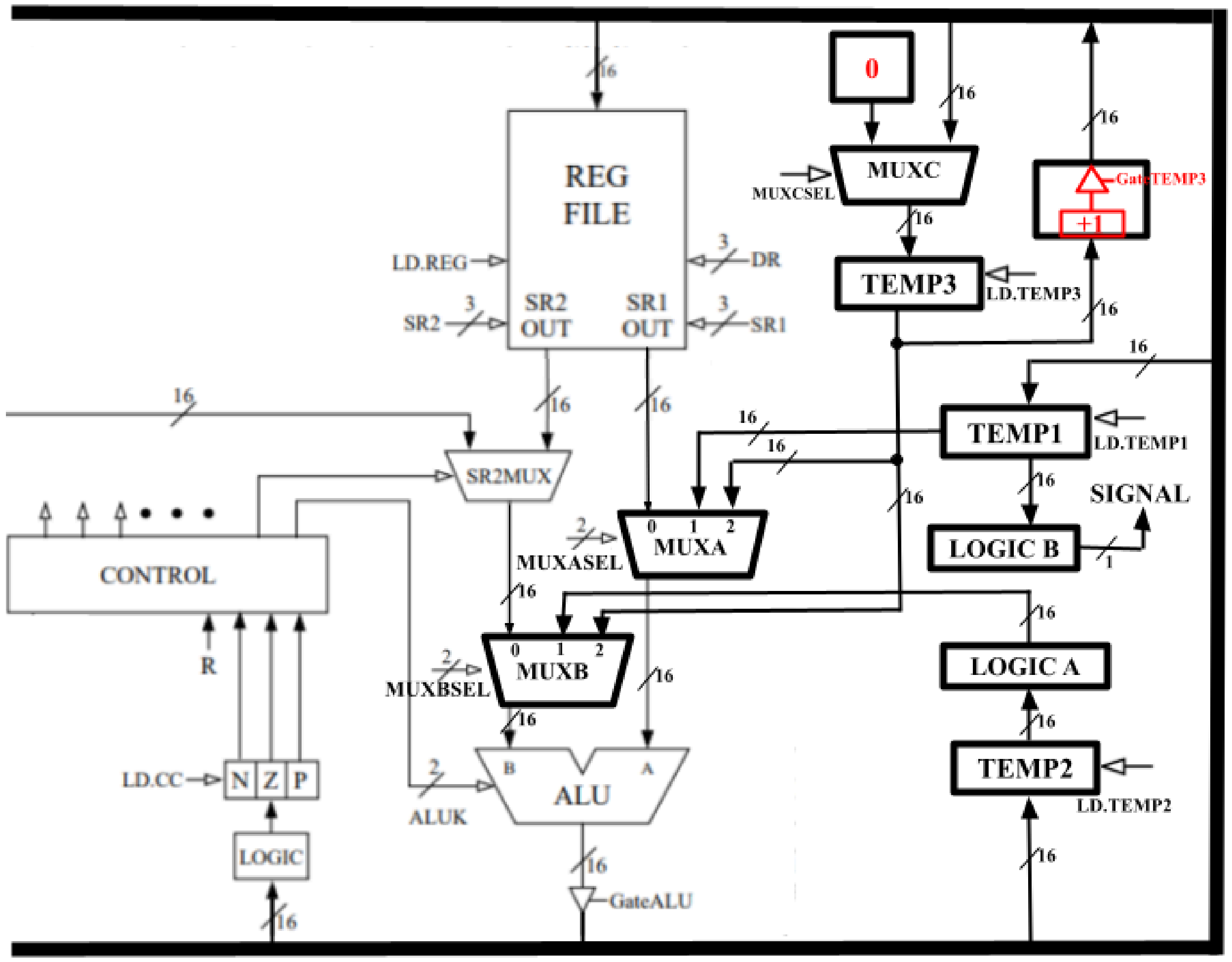
SR1 contains the starting address of string A in memory. SR2 contains the starting address of string B in memory. The execution of this instruction does not destroy the contents of SR1 or SR2. **The output is written to DR. This instruction updates condition codes based on the value in DR.**

Your job: Implement STRCMP. There are 5 parts: State machine, datapath, microsequencer, control store and analysis. **Note:** Examine all parts of the problem before completing your implementation, as that may help you understand how to solve it.

Part a (10 points): Complete the state machine below:

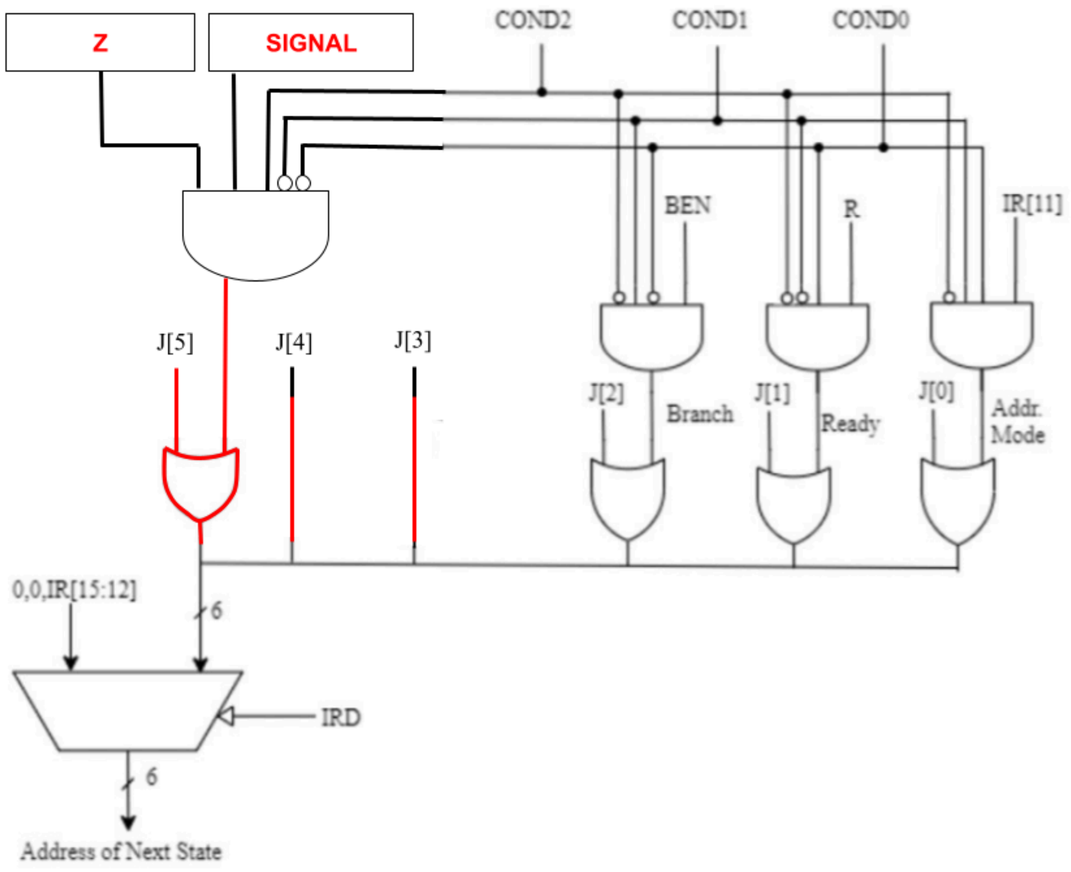


Part b (8 points): The additions to the datapath are shown in boldface below.
Fill in the 2 boxes in the datapath and answer the questions below.
Points will not be deducted for missing the buffer and GateTEMP3.



What is the function of LOGIC A?	What is the function of LOGIC B?
~TEMP2 + 1 **(Negate TEMP2)**	**TEMP1 != 0**

Part c (4 points): Modify the microsequencer below with the correct control logic:



Part d (5 points): Shown below are six of the control signals. Fill in the missing entries and the missing state number. If the value of a control signal does not matter, label that entry as 'X':

State	COND[2:0]	J	LD.CC	ALUK	MUXASEL	MUXBSEL
37	000	**40**	0	**ADD**	**2**	0
43	000	**44**	**1**	ADD	1	**1**
44	**100**	**18**	**0**	X	X	X

Part e (3 points): Does a programmer using the STRCMP instruction need to know the length of register TEMP3? Explain in 20 words or fewer.

Yes. Executing the STRCMP instruction on strings with greater length than what TEMP3 supports results in an infinite loop because TEMP3 overflows.

Question 4 (30 points): A microarchitecture using Tomasulo's algorithm is executing a program. Specifications are as follows:

- The instruction cycle is 4 stages: FETCH, DECODE, EXECUTE, and WRITEBACK.
- FETCH, DECODE, and WRITEBACK take one cycle each.
- ADD takes 3 cycles to execute, MUL takes 4 cycles to execute.
- The reservation stations for ADD and MUL have 3 entries each.
- The tags are α, β, and γ for ADD, and π, ρ, and σ for MUL
- There is one pipelined adder and one pipelined multiplier.
- Registers are renamed and allocated to reservation stations at the end of DECODE in a top-to-bottom manner.
- Instructions with no dependencies can start execution directly after DECODE.
- There is no data forwarding. Dependent instructions can begin execution in the cycle after the source value is written back.
- Reservation station entries are deallocated at the end of WRITEBACK. An instruction that cannot enter a reservation station must stall in DECODE and can enter the reservation station in the clock cycle following the WRITEBACK of a previous instruction.
- Only one instruction can be in WRITEBACK in each clock cycle. If multiple instructions are ready to write in the same clock cycle, the oldest instruction is written back while the others stall.
- Tags remain in the register alias table after a value is written back.

Your job:

Part a (10 points): Fill in the missing entries in the program.

Part b (10 points): Complete the pipeline timing diagram for the execution of the program.

Part c (10 points): Determine and fill in the missing entries in the register alias tables.

PROBLEM CONTINUES ON NEXT PAGE

	OP	DR	SR1	SR2
Instruction 1	MUL	R1	R7	R2
Instruction 2	MUL	R5	R1	R6
Instruction 3	ADD	R2	R3	R1
Instruction 4	ADD	R4	R4	R1
Instruction 5	ADD	R5	R3	R7
Instruction 6	ADD	R6	R3	R6

•	Stall
A	ADD execute
M	MUL execute
R#	Writeback

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
I1	F	D	M	M	M	M	R1									
I2		F	D	•	•	•	•	M	M	M	M	R5				
I3			F	D	•	•	•	A	A	A	R2					
I4				F	D	•	•	•	A	A	A	•	R4			
I5					F	D	A	A	A	R5						
I6						F	D	•	•	•	A	A	A	R6		

	V	Tag	Value
R0	1	α	0
R1	1	β	1
R2	1	γ	2
R3	1	π	3
R4	1	ρ	4
R5	1	σ	5
R6	1	α	6
R7	1	β	7

Before cycle 1

	V	Tag	Value
R0	1	α	0
R1	1	π	14
R2	0	α	2
R3	1	-	3
R4	0	β	4
R5	0	γ	5
R6	1	-	6
R7	1	-	7

After cycle 9

	V	Tag	Value
R0	1	α	0
R1	1	π	14
R2	1	α	17
R3	1	-	3
R4	1	β	18
R5	1	γ	10
R6	0	γ	6
R7	1	-	7

After cycle 13

Below are blank reservation stations and a timing diagram to use for scratch work. Nothing on this page will be graded

	V	Tag	Value	V	Tag	Value
α						
β						
γ						

	V	Tag	Value	V	Tag	Value
π						
ρ						
σ						

	V	Tag	Value	V	Tag	Value
α						
β						
γ						

	V	Tag	Value	V	Tag	Value
π						
ρ						
σ						

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
I1																
I2																
I3																
I4																
I5																
I6																