ESE 356 Digital System Specification and Modeling
Project 1: MINI RISC Processor
Phase 2 (Final Version) Requirement

Due on 10/10/2019

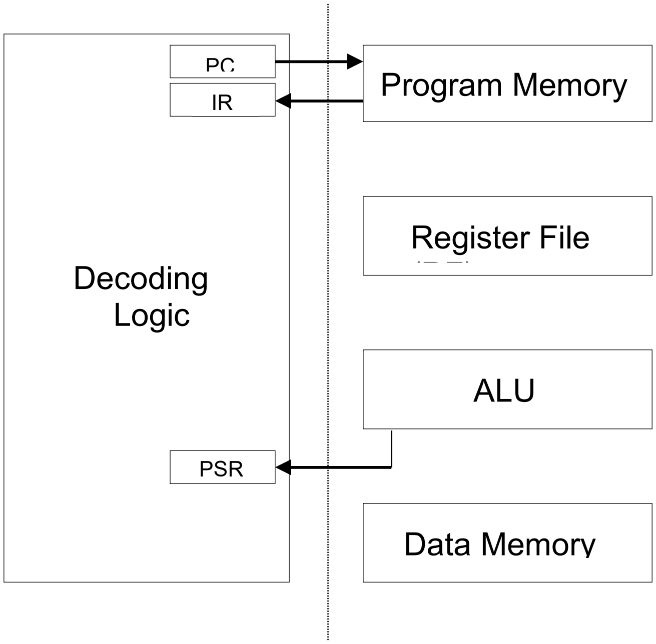
Total Points (25): No late submission (submit the file by midnight of the due date)

1. Phase Specification

Revise your data-path if necessary.

Based on the control signals defined in the control signal table, create a controller module. All input and output ports must match the ports defined in your data-path modules.

For PSR, Z, N, C bits will be implemented (no F bit).



Controller design strategy will be discussed in the lecture.

Obtain a table for indicating all control signals necessary to control the data-path.

Table 1: Control Signals from Controller to Data-path (Revised from Phase 1)

Instruction	MUX1	MUX2	ALU	…..
ADD	010	11	111100	
ADDI				
….				

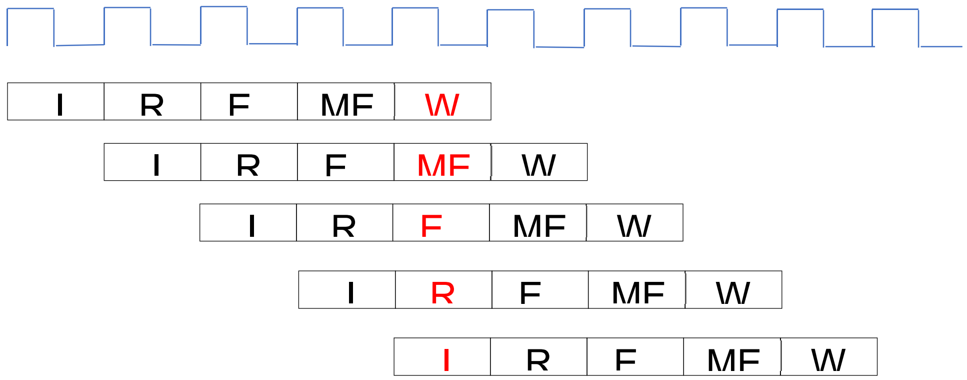
Obtain a table for indicating all control signals necessary to control the data-path.

Table 2: Status Signals from Data-path to Controller

Data-path	MUX1	MUX2	ALU	…..
UNIT1	010	11	111100	
….				

5-stage pipeline execution is assumed in this project.

IF: Instruction Fetch – At the rising edge of the clock cycle, an instruction is read out from the program memory based on the address located by the program counter (PC).
At the next rising edge, this instruction is latched by the instruction register (IR).



RD: At the rising edge of the clock cycle, the IR content and the values in the PSR are used to decode an instruction. All control signals are generated, and some will be stored in the shift registers to be used later.

EXE: Execution – ALU performs the operation. The ALU also updates PSR content (You need to revise your current data-path modules)

MEM: Memory access

WB: At the falling edge of the clock cycle, the result will be written

Obtain a table for indicating all control signals necessary to control the data-path. Revise the table for control signals including the stage index (Indicate when specific control signals are used in the pipeline stage)

Table 3: Pipeline Stage Dependent Control Signal Generation

Instruction	Stage	UNIT1	UNIT2	…..
ADD	Overall	11	111100	
	RD			
	EXE			
	MEM			
	WB			
ADDI	Overall			
	RD			
	EXE			
	…			

Once the controller modules are completed, integrate with the data-path modules.

2. Verification and Simulation

Testing:

Write a simple test program to test all of your instructions including conditional instructions. Make sure to insert NOP instruction right after the memory access instruction.

Will provide a sample structure of the test program soon!!

Program 1:

```
// DM[10] = 36;
// DM[12] = 12;
// DM[14] = -24;
// All other locations are initialized to ZEROS
// RF[R3] = 10;
// RF[R5] = 12;
// RF[R7] = 14;
// RF[R8] = 12;
// All other locations are initialized to ZEROS

LOAD   R2, R3;               DM[R3] -> RF[R2]               // R2 has Integer 36
LOAD   R4, R5;               DM[R5] -> RF[R4]               // R4 has Integer 12
ADDI   17, R8;               RF[R8] + 17 -> RF[R8]          // R8 is changed from 12 to 29
ADD    R2, R4;               RF[R4] + RF[R2] -> RF[R4]      // R4 has integer 48
LOAD   R6, R7;               DM[R7] -> RF[R6]               // R6 has Integer -24
SUBI   15, R8;               RF[R8] - 15 -> RF[R8]          // R8 has Integer 14
LOAD   R2, R5;               DM[R5] -> RF[R2]               // R2 has Integer 12
ADD    R8, R2;               RF[R2] + RF[R8] -> RF[R2]      // R2 has Integer 26
STOR   R8, R3;               RF[R8] -> DM[R3]               // DM[10] has Integer 14
STOR   R4, R5;               RF[R4] -> DM[R5]               // DM[12] has Integer 48
```

Program 2:

```
// RF must be initialized so that following integers are stored
// RF[R1] = 1;
// RF[R2] = 2;
// RF[R3] = 3;
// RF[R4] = 4;
// RF[R5] = 5;
// RF[R6] = 6;
// RF[R7] = 7;
// RF[R8] = 8;
// RF[R10] = 26;
// Program must start from address 10
ADD    R2, R1;          RF[R1] + RF[R2] -> RF[R1]           // Address 10, R1 has Integer 3
ADD    R3, R1;     RF[R1] + RF[R3] -> RF[R1]      // R1 has Integer 6
ADD    R4, R1;     RF[R1] + RF[R4] -> RF[R1]      // R1 has Integer 10
ADD    R5, R1;     RF[R1] + RF[R5] -> RF[R1]      // R1 has Integer 15
CMP    10, R1;                                    // Address 18
JNE    R10;                                       // Address 20
ADD    R6, R1;     RF[R1] + RF[R6] -> RF[R1]
ADD    R7, R1;     RF[R1] + RF[R7] -> RF[R1]
ADD    R8, R1;     RF[R1] + RF[R8] -> RF[R1]
// Repeat the same program by replacing CMP 10, R1 to CMP 15, R1
```

3. Submission Requirements

- Source codes for controller, data-path, top main and necessary test-bench codes
- Data-path diagram and control signal table
- Verification/Simulation results: Before and after of the content of data memory, program memory
- Summary report (1-2 pages)

Submission through electronic files (zip version)
The report grading will be based on 1. Clarity of the report, 2. Completeness of the results.