CSC / CpE 142 Term Project
Fall 2019, Dr. Arad
Phase 1: 100 Points due 11/4/2019 100 Pts Phase 2: 900 Pts 12/02/2019
10% Late Penalty for Phase 2
No late submission for Phase 2 after 11:59 pm 12/04/2019

The objective of this project is to design and simulate the datapath and control unit for a pipelined system, which can execute the following assembly instruction set:

Function	Syntax	opcode	op1	op2	funct. Code	type	Operation
Signed addition	add op1, op2	0000	reg	reg	0000	A	op1 = op1 + op2
Signed subtraction	sub op1, op2	0000	reg	reg	0001	A	op1 = op1 - op2
Signed multiplication	mul op1, op2	0000	reg	reg	0100	A	op1 = op1 * op2 op1: Product (lower half) R15: Product (upper half)
Signed division	div op1, op2	0000	reg	reg	0101	A	op1: 16-bit quotient R15: 16-bit remainder
AND immediate	andi op1, op2	0001	reg	Imm d.	N/A	C	op1 = op1 & {8'b0, constant}
OR immediate	or op1, op2	0010	reg	Imm d.	N/A	C	op1 = op1 | {8'b0, constant}
Load byte unsigned	lbu op1, immd (op2)	1010	reg	reg	N/A	B	op1 = {8'b0, Mem [immd + op2] } (sign extend immd)
Store byte	sb op1, immd (op2)	1011	reg	reg	N/A	B	Mem [immd + op2](7:0) = op1(7:0) (sign extend immd)
Load	lw op1, immd (op2)	1100	reg	reg	N/A	B	op1 = Mem [immd + op2] (sign extend immd)
Store	sw op1, immd (op2)	1101	reg	reg	N/A	B	Mem [immd + op2] = op1 (sign extend immd)
Branch on less than	blt op1, op2	0101	reg	immd.	N/A	C	if (op1 < R15) then PC = PC + op2 (sign extend op2 & shift left)
Branch on greater than	bgt op1, op2	0100	reg	immd.	N/A	C	if (op1 > R15) then PC = PC + op2 (sign extend op2 & shift left)
Branch on equal	beq op1, op2	0110	reg	immd.	N/A	C	if (op1 == R15) then PC = PC + op2 (sign extend op2 & shift left)
jump	jmp op1	0111	off-set	-------	N/A	D	pc = pc + op1 (sign extend op1& shift left)
halt	Halt	1111	-----	------	N/A	D	halt program execution

Each addressable memory location is a byte. The memory module can store 2^{15} words. Each instruction is 16-bit long (1 word). Each register is 16-bit long (1 word). There are 16 registers denoted by R0 through R15. Register R15 is used implicitly in branch instructions. There are 4 possible instruction formats:

Type A:

4- bit opcode	4-bit operand 1	4-bit operand 2	4-bit funct code

Type B:

4- bit opcode	4-bit operand 1	4-bit operand 2	4-bit offset

Type C:

4- bit opcode	4-bit operand 1	8-bit offset/constant

Type D:

4- bit opcode	12-bit offset in jump -- unused in halt

You need to follow the design criteria discussed in Chapter 4 of the textbook. In particular, you must add enough logic to deal with hazards. Your system should also handle exceptions. In case of any exception (for example wrong opcode, an arithmetic overflow, etc.) you need to halt the execution of the program and display a message indicating the type of exception, which has occurred.

PHASE 1

First, you need to design the datapath and control unit that meet the above requirements and validate your design.

On the due date for Phase I you must submit a typed report including the items listed below through as one PDF file to Canvas. Each teammate must submit a copy of the team report.

- *i) Cover sheet including CSC/CpE 142, Term Project, Phase 1, names, percentage of contribution from each partner.*
- *ii) A professionally drawn figure showing the complete datapath.*
- *iii) Specifications of datapath components.*
- *iv) Representation of each control logic as a truth table or logical expressions.*

PHASE 2:

Once the design is complete, model and simulate each component of the system in Verilog. Verify the functionality of the individual components using stimulus files written in Verilog. After modeling and validation of individual components, simulate the complete system in Verilog. Use a machine-code program to test the simulated system. You must use the assembly code that will be provided in future lectures. You need to store the machine code version of the assembly program provided in the memory unit before testing the design.

Use a **stimulus file** to illustrate the functionality of the simulated system. In the stimulus file you must: i) instantiate the simulated system, ii) provide a clock generator block, iii) initialize all necessary parameters (such as reset) and provide any other necessary input, and iv) display input & output ports of major components such as PC, ADDERS, MEMORY, REGISTER FILE, ALU, and pipeline buffers (inputs and output) on every negative edge of the clock. These values will be used to test your system. It will also show the number of clock cycles required for each instruction.

The Verilog code you provide must the original work of the team. You must indicate if any Verilog code that you provide is not your original work and must list the reference(s) used to get the code. You only get credit for the code you designed.

On the due date for Phase II you must submit a typed report including the items listed below through as one PDF file to Canvas. Each teammate must submit a copy of the team report.

- *A Cover sheet (Include CSC/CpE 142, Term, Term Project, your name(s), percentage of overall contribution by each partner).*
- *Table of contents (with page number).*
- *Status report (A template for this will be provided).*
- *A professionally drawn diagram which shows the datapath. Properly label each component.*
- *The high representation of any control logic used in the design including the main control unit, ALU control unit, and any hazard detection, register forwarding used.(truth table or other forms)*
- *Verilog source code and test file for each component of the datapath.*
- *The test assembly program used to test the simulated system with expected results.*
- *The stimulus module used for testing the system.*
- *Simulation Results for the top-level*

In addition to the report you must submit the complete design electronically through Canvas. You will be given instruction on how to submit electronic copy through email later.

You must work in groups of 2 for this project. It is your responsibility to find partners who are willing to work with you. Groups must work independently. No collaboration is allowed among groups at any level. If you have any question regarding the project you need to contact the instructor.

Tentative Grading policy for the project:

Phase I	10%
Phase II	
Project Report	20 %
Functionality of the individual components	40%
Functionality of the overall design	25%
Design approach	5%

Test Assembly Code:

The register file and the memory module for your term project must be initialized based on the given immediate values or instructions. All immediate values and memory addresses are given as hexadecimal numbers. Registers are denoted by R0 - R15. In case of instructions, you need to First convert them into machine code and then store them in the memory. The memory and register file should be re-initialized with the given values on each reset.

Register	***Content***
======	======
R1	*7B18*
R2	*245B*
R3	*FF0F*
R4	*F0FF*
R5	*0051*
R6	*6666*
R7	*00FF*
R8	*FF88*
R9	*0000*
R10	*0000*
R11	*3099*
R12	*CCCC*
R13	*0002*
R14	*0011*
R15	*0000*

Instruction Address	***Memory Content***
======	=========
00	*ADD R14, R2*
02	*SUB R11, R2*
04	*ORi R3, 0088*
06	*ANDi R4, 9A*
08	*MUL R5, R6*
0A	*DIV R1, R6*
0C	*SW R5, A(R9)*
0E	*ORi R8, 2*

10 LW R14, A(R9)
12 SUB R15, R15
14 ADD R1, R2
16 SUB R1, R2
18 ANDi R8, 2
1A LBU R6, 4(R9)
1C SB R6, 6(R9)
1E LW R6, 6(R9)
20 SUB R7, R13
22 BEQ R7, 4
24 ADD R11, R1
26 BLT R7, 5
28 ADD R11, R2
2A BGT R7, 2
2C ADD R1, R1
2E ADD R1, R1
30 LW R8, 0(R9)
32 ADD R8, R8
34 SW R8, 2 (R9)
36 LW R10, 2 (R9)
38 ADD R12, R12
3A SUB R13, R13
3C ADD R12, R13
3E HALT
Other memory locations = 0000

Data Memory (data and address in hex)
00 3142
02 0000
04 5678
06 DEAD
08 BEEF
Other memory locations = 0000