CSC/CPE 142 Term Project Phase 2

Daniel Komac 50%

Scott O'Hair 50%

Table of Contents

CSc/CPE 142
Term Project Status Report

Complete this form by typing the requested information and include the completed form in your report after TOC. Gray cells will be filled by the instructor.

Name	*% Contribution*	*Grade*
Daniel Komac	*50*	
Scott O'Hair	*50*	

Please do not write in the first table

Project Report/Presentation 20%	/200
Functionality of the individual components 40%	/400
Functionality of the overall design 25%	/250
Design Approach 5%	/50
Total points	/900

A: List all the instructions that were implemented correctly and verified by the assembly program on your system:

None were verified to be working or implemented correctly in the final CPU instance, however with an additional day of debugging all would be working fine

Instructions	State any issue regarding the instruction.
Signed addition	See above.
Signed subtraction	See above.
Signed multiplication	See above.
Signed division	See above.
Move	See above.
SWAP	See above.
AND immediate	See above.
OR immediate	See above.
Load byte unsigned	See above.
Store byte	See above.

Instructions	State any issue regarding the instruction.
Load	See above.
Store	See above.
Branch on less than	See above.
Branch on greater than	See above.
Branch on equal	See above.
Jump	See above.
halt	See above.

B: Fill out the next table:

Individual Components	Does your system have this component?	List the student who designed and verified the block	Does it work ?	List problems with the component, if any.
ALU	Yes	Daniel	Yes	Assuring the each instruction could be handled
ALU control unit	Yes	Daniel	Yes	Syncing with control unit
Memory Unit	Yes	Scott	Yes	None
Register File	Yes	Scott	Yes	None
PC	Yes	Scott	Yes	None
IR	Yes	Scott	Yes	None
Other registers	Yes	Both	Yes	None
Multiplexors	Yes	Both	Yes	None
exception handler 1. Unknown opcode 2. Arith. Overflow …..	N/A			
Control Units 1. main 2. forwarding 3. lw hazard	1.yes 2.yes 3.yes	1.Daniel 2.Scott 3.Daniel	Yes	Aligning correct bit outputs to each controlled unit.

Individual Components	Does your system have this component?	List the student who designed and verified the block	Does it work ?	List problems with the component, if any.
detection				

How many stages do you have in your pipeline? 5 Stages

C: **State any issue regarding the overall operation of the datapath? Be Specific.**

Our datapath would be fully operational given another day of debugging. We were able to build and test each individual component. We started to piece these together and were able to get IF, ID, and parts of EX, MEM and WB to be functional. Stages in our pipeline work individually as designed.

We ran out of time while debugging the control unit and its operation when integrated into the datapath. We are both confident that given a little more time to debug that we would have had the system fully functional.

Formatting and debugging were the most time consuming and difficult aspects of the project. The design and programming of individual components was significantly less difficult and less time consuming.

Data Path and Control

Hazard Detection Unit
Control
IF/ID
[15-12]
4
Branch Taken
+
2
Shift Left 1
PC
Address
Instruction Memory
Instruction
16
[11-8]
[7-4]
Read Register 1
Read Register 2
Registers
Write Register 2
Write Register 1
Write Data 2
Write Data 1
Op1
Op2
Sign Extend
8
IF Flush
<, >, =
ID/EX
WB
M
EX
0
Op1
Op2
Data Memory or Earlier Result Forward
ALU Result Forward
ALU
32
ALUControl
4
ID/EX Write Register
Reg2
Reg1
Forwarding Unit
EX/MEM
MEM/WB
Address
Data Memory
Write Data
Write Data
EX/MEM Write Register 1
EX/MEM Write Register 2
SWAP
MEM/WB Write Register
ALU Result Forward

Specifications of Datapath Components

Note: multiplexors are numbered in ascending order from top to bottom in their respective stage

Instruction Fetch:

Multiplexor: When branch is taken, the address is taken from the branch line. Otherwise the program counter address is taken from the increment by two block

Program Counter: Keeps track of 16 bit program address

Instruction Memory: Uses 16 bit instruction address input to fetch 16 bit instruction from corresponding memory location, passes the instruction to IF/ID buffer

Instruction Decode:

Mux: Muxes control with 0, if flushed then 0 is written, else passes control forward

Registers:

Read Register 1: Takes 4 bit [11..8] read reg address to determing read location

Read Register 2: Takes 4 bit [7..4] read reg address to determing read location

Write Register 1: 4 bit write register location

Write Register 2: 4 bit write register location, used in SWAP

Write Data 1: Data to be written into Write Register 1 location

Write Data 2: Data to be written into Write Register 2 location

Sign Extend: Sign extends 8 bits to 16 bit

Shift Left: Shifts left by 1, used to determine branch address when branch is taken

Comparator: Comparator is used to determine if branch will need to be taken

Execute:

ALU: Performs the following arithematic operations on two inputs: signed addition, signed subtraction, signed multiplication, signed division, AND, OR

ALUControl: 4 bit function code input from instruction is used to determine ALU operation to be performed, outputs selection to ALU

Mux1: Muxes Write Back control with 0, if flushed then 0 is written, else passes WB. control forward

Mux2: Muxes Memory control with 0, if flushed then 0 is written, else passes M. control forward

Mux3: Muxes Operand 1, Data Memory/Earlier Result Forward and ALU result forward based on forwarding unit selection

Mux4: Muxes sign extended operand and Mux5 output and passes result to ALU. Controlled by Execute Control signal

Mux5: Muxes Operand 2, Data Memory/Earlier Result Forward and ALU result forward based on forwarding unit selection

Memory:

Data Memory: 16 bits of ALU result is used to determine write address if memory write is occuring. 16 bit write data is buffered in through EX/MEM buffer based on decisions in previous stages.

Mux 1: Muxed ALUControl switches between 16 bit ALU result or EX/MEM buffer

Write Back:

Mux: Muxes two data inputs from buffer and passes result to Write Data 1 in the Instruction Decode stage

Control:

Control Unit: Decodes 4 bit function code input and determines control bit outputs based on control logic truth table

AND: AND's the register comparator and branch control line to determine if a branch needs to be taken, result is used to control the Instruction Memory Multiplexor

Buffers:

IF/ID:

-16 bit instruction to ID stage
-program counter increment
-can be flushed with IF Flush

ID/EX:

-control lines
-16 bit operand 1
-16 bit operand 2
-operand address 1
-operand address 2
-sign extend result

EX/MEM:

-Write Back and Memory Control
-32 bit ALU result
-16 bit EX Mux 5 result
-ALUControl signals
-operand address 1
-operand address 2

MEM/WB:

-Memory read value
-Memory stage mux result
-ALU result
-operand address 1

-operand address 2

Forwarding Unit:

-Forwarding Unit decides whether forwarding needs to occur based on its reg address inputs, if forwarding is needed it will switch the Execute multiplexors to use forwarded data as input

-Reg 1 address input
-Reg 2 address input
-Control output to ALU Mux 3 and ALU mux 5 if forward occurs/doesn't occur
-Reg 1 address input (Mem stage)
-Reg 1 address input (Write Back stage)

Hazard Detection Unit:

Hazard Detection Unit: detects hazard conditions based on inputs and inserts bubbles into data path

OR: Hazard Detection Unit out and Control to control mux to insert bubbles into datapath

Mux: Muxes control with 0, if flushed then 0 is written, else passes control forward

	Truth Table		Output								
	Op 4'b	Func Cd	ALU CNTRL	ALUSrc	MemWrt	RegWrt	MemReg	MemRd	RegDst	Branch	JMP	DIV/MUL
add	0000	0000	0000	0	0	1	1	0	1	0	0	0
sub		0001	0001	0	0	1	1	0	1	0	0	0
mult		0100	0010	0	1	1	1	0	1	0	0	1
div		1000	0011	0	1	1	1	0	1	0	0	1
move		1110	0100	0	0	0	1	0	1	0	0	0
swap		1111	0101	0	0	0	1	0	1	0	0	0
AND	1	----	0110	1	0	1	0	0	1	0	0	0
OR	10	----	0111	1	0	1	0	0	1	0	0	0
Load B	1000		1000	10	0	1	1	1	0	1	0	0
Store B	1001		1000	10	0	0	0	0	0	1	0	0
Load	1010		1000	11	0	1	1	1	0	1	0	0
Store	1011		1000	11	0	0	0	0	0	1	0	0
												0
BLE	0100		1001	100	-	0	-	1	-	1	0	0
BGE	0101		1001	100	-	0	-	1	-	1	0	0
BE	0110		1001	100	-	0	-	1	-	1	0	0
JUIMP	1100			0	0	0	0	0	0	0	1	0
HALT	1111			---	-	-	-	-	-	-	1	-

Forwarding Unit	Forward Mux A	Forward Mux B
0	0	0
1	1	0
10	0	1
11	1	1

Results only happen when Register Add is equal which is comapred in forwarding unit

ALU Control		MUX Control
0	add	0
1	sub	0
10	div	1
11	mult	1
100	MV	0
101	SWAP	0
110	AND	0
111	OR	0

Hazard Detection Unit

READBIT	Op1	Op2	Future Op	IFBLK	PCBLK	BUBMUX
0	1	1	1	0	0	0
1	1	0	1	1	1	1
1	0	1	1	1	1	1

Whenever registers op1 or 2 are equal
to future op then the flag is set
Note: Operations are 4 bit reg addresses