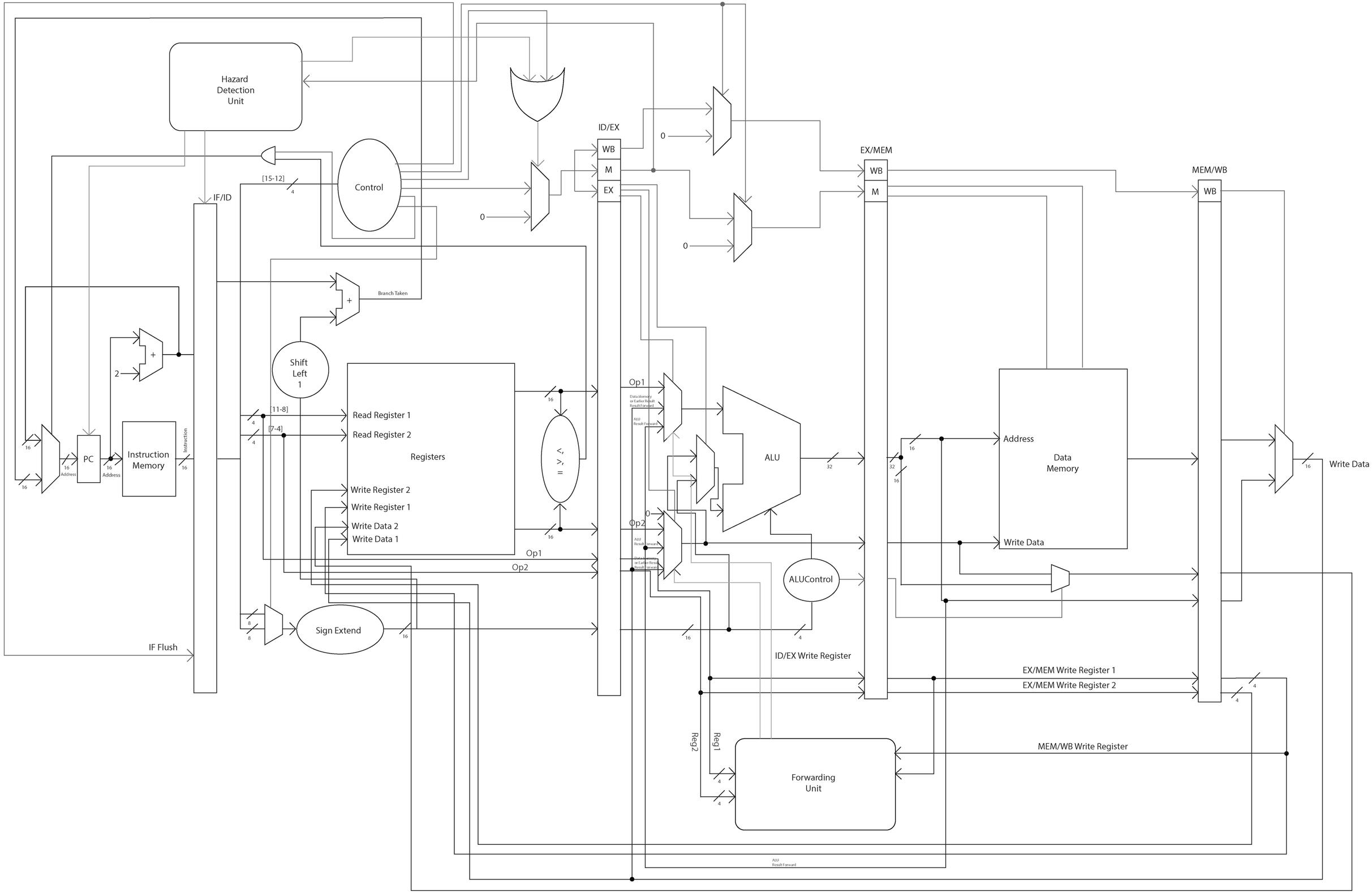
CSC/CPE 142 Term Project Phase 1

Daniel Komac 50%
Scott O'Hair 50%

Data Path and Control



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

Control Logic Truth Table

Function	Syntax	opcode	op1	op2	funct. Code	type	Operation		Truth Table Op 4'b	Func Cd	Output 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
Signed addition	add op1, op2	0	reg	reg		0 A	op1 = op1 + op2	mult		0100	0010	0	1	1	1	0	1	0	0	1
								div		1000	0011	0	1	1	1	0	1	0	0	1
Signed subtraction	sub op1, op2	0	reg	reg		1 A	op1 = op1 - op2	move												
										1110	0100	0	0	0	1	0	1	0	0	0
								swap		1111	0101	0	0	0	1	0	1	0	0	0
Signed multiplication	mul op1, op2	0	reg	reg		100 A	op1 = op1 * op2													
							op1: Product (lower half) R15: Product (upper half)	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
Signed division	div op1, op2	0	reg	reg		1000 A	op1: 16-bit quotient R15: 16-bit remainder	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
Move	mv op1, op2	0	reg	reg.		1110 A	op1 <= op2													0
SWAP	swp op1, op2	0	reg	reg		1111 A	op1 <= op2 op2 <= op1	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
AND immediate	andi op1, op2	1	reg	Immd.	N/A	C	op1 = op1 & {8'b0, constant}	JUIMP	1100			0	0	0	0	0	0	0	1	0
OR immediate	or op1, op2	10	reg	Immd.	N/A	C	op1 = op1 | {8'b0, constant}	HALT	1111			---	-	-	-	-	-	-	1	-
Load byte unsigned	lbu op1, immd (op2)	1000	reg	reg	N/A	B	op1 = {8'b0, Mem [immd + op2] } (sign extend immd)	Forwarding Unit		Forward Mux A	Forward Mux B									
									0	0	0									
									1	1	0									
Store byte	sb op1, immd (op2)	1001	reg	reg	N/A	B	Mem [immd + op2](7:0) = op1(7:0)		10	0	1									

						(sign extend immmd)	11	1	1
Load	lw op1, immd (op2)	1010	reg	reg	N/A	B	op1 = Mem [immd + op2] (sign extend immmd)	Results only happen when Register Add is equal which is comapred in forwarding unit	
Store	sw op1, immd (op2)	1011	reg	reg	N/A	B	Mem [immd + op2] = op1 (sign extend immmd)		
Branch on less than	blt op1, op2	100	reg	immd.	N/A	C	if (op1 < R15) then PC = PC + op2 (sign extend op2 & shift left)		
Branch on greater than	bgt op1, op2	101	reg	immd.	N/A	C	if (op1 > R15) then PC = PC + op2 (sign extend op2 & shift left)		
Branch on equal	beq op1, op2	110	reg	immd.	N/A	C	if (op1 = R15) then PC = PC + op2 (sign extend op2 & shift left)		
jump	jmp op1	1100	off-set	-------	N/A	D	pc = pc + op1 (sign extend op1& shift left)		
halt	Halt	1111	-----	------	N/A	D	halt program execution		

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

Hazard Detection Unit

On Incorrect sequence through error detection outputs signals 1 for all.

Verilog Modules to be Developed

Test Bench Scott & Daniel

Main CPU Instance

- **IF Mux S**
- **Program Counter S**
- **Instruction Memory S**

- **IF/ID Buffer S**

- **Register Module S**
- **Sign Extend S**
- **Shift Left S**
- **Branch Adder S**
- **Comparator S**
- **ID Mux1 S**
- **ID Mux2 S**
- **OR Hazard Detection Unit/Control S**

- **ID/EX Buffer S**

- **ALU Module D**
- **EX Mux1 D**
- **EX Mux2 D**
- **EX Mux3 D**
- **EX Mux4 D**
- **EX Mux5 D**
- **ALU Control D**

- **EX/MEM Buffer D**

- **Forwarding Unit D**

- **MEM/WB Buffer D**

- **Data Memory D**
- **Mem Mux D**

- **Write Back Mux D**

- **Control Unit S**

- **Hazard Detection Unit D**