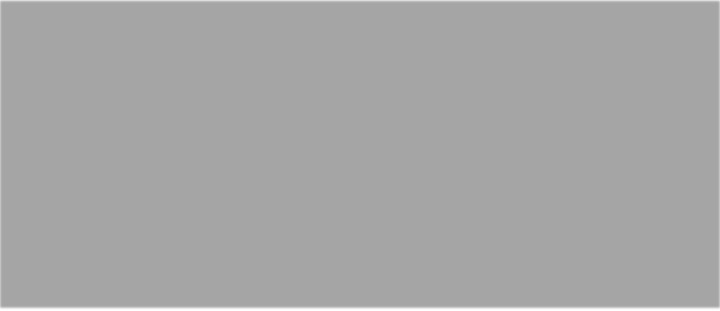
**CprE 381: Computer** 

**Organization and Assembly Level Programming**

****Processor Design

Henry Duwe

Electrical and Computer Engineering

Iowa State University

**Review: Fetching Instructions**

• Fetching instructions involves

– reading the instruction from the Instruction Memory

– updating the PC value to be the address of the next (sequential) instruction

clock

4 Fetch PC=PC+4

Add

Instruction Memory

Exec Decode

PC

Read

Address Instruction

– PC is updated every clock cycle, so it does not need an explicit write control signal – just a clock signal

– Reading from the Instruction Memory is a combinational activity

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.2 

**Review: Decoding Instructions**

• Decoding instructions involves

– Sending the fetched instruction’s opcode and function field bits to the control unit

Control

Unit

Fetch

PC=PC+4

Exec Decode

Instruction

Read Addr 1

Register

Read Addr 2 File

Write Addr Write Data

Read

Data 1

Read

Data 2

– Reading two values from the Register File

• Register File addresses are contained in the instruction

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.3 

***Review*: Executing R Format Operations** • R format operations (**add, sub, slt, and, or**) 31 25 20 15 5 0

10

R-type:

op rs rt rd shamt funct

– Perform operation (op and funct) on values in rs and rt – Store the result back into the Register File (into location rd)

RegWrite ALU control

Fetch

PC=PC+4

Exec Decode

Instruction

rs rt

rd

Read Addr 1

Register

Read Addr 2 File

Write Addr Write Data

Read

Data 1

Read

Data 2

ALU

overflow zero

– Note that Register File is not written every cycle (e.g. sw or jr), so we need an explicit write control signal for the Register File

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.4 

**Consider the slt Instruction** • Remember the R format instruction **slt**

**slt $t0, $s0, $s1 # if $s0 < $s1**

**# then $t0 = 1**

**# else $t0 = 0**

– Where does the 1 (or 0) come from to store into $t0 in the Register File at the end of the execute cycle?

RegWrite ALU control

Instruction

rs rt

rd

Read Addr 1

Register

Read Addr 2 File

Write Addr Write Data

Read

Data 1

Read

Data 2

ALU

overflow zero

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.5 

***Review*: Executing Load and Store Operations** • Load and store operations have to:

31 25 20 15 0

I-Type: op rs rt address offset

– Compute a memory address by adding the base register (in rs) to the 16-bit signed offset field in the instruction

• Base register was read from the Register File during

decode

• Offset value in the low order 16 bits of the instruction

must be sign extended to create a 32-bit signed value

– Store value, read from the Register File during decode, must be written to the Data Memory

– Load value, read from the Data Memory, must be stored in the Register File

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.6 

**Executing Load / Store Operations (cont.)**

rs

RegWrite ALU control overflow

MemWrite

Instruction

rt rt

Read Addr 1

Register

Read Addr 2 File

Write Addr

Read Data 1

Read

zero

ALU

Address

Data

Memory

Read Data

address offset

Write Data

Data 2

Sign

Extend

Write Data

16 32

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.7 

**Executing Branch Operations**

• Branch operations have to

31 25 20 15 0

I-Type: op rs rt address offset

– Compare the operands read from the Register File during decode (rs and rt values) for equality (**zero** ALU output)

– Compute the branch target address by adding the updated PC to the sign extended16-bit signed offset field in the

instruction

• The “base register” is the updated PC

• Offset value in the low order 16 bits of the instruction

must be sign extended to create a 32-bit signed value

and then shifted left 2 bits to turn it into a word address

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.8 

**Executing Branch Operations (cont.)**

Add

4 Add Shift

left 2

ALU control

PC

Branch target

address

rs

rt

Instruction

Read Addr 1

Register

Read Addr 2 File

Write Addr Write Data

Read

Data 1

Read

Data 2

zero

ALU

(to branch control logic)

Sign

Extend 16 32

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.9 

**Executing Jump Operations** • Jump operations have to

31 25 0

J-Type: op

jump target address

– Replace the lower 28 bits of the PC with the lower 26 bits of the fetched instruction shifted left by 2 bits

Add

PC

4

Shift

left 2

Read

Instruction Memory

4

28

Jump

address

Address Instruction

26

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.10 

**Creating a Single Datapath from the Parts**

• Assemble the datapath elements, add control lines as needed, and design the control path • Fetch, decode and execute each instructions in one clock cycle – single cycle design

– One instruction can’t use same resource/structure twice (ergo Harvard split memory architecture)

– Two different instructions need multiplexors at the input of the shared elements with control lines to do the selection

• Cycle time is determined by length of the longest path

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.11 

**Fetch, R, and Memory Access Portions**

Add

4

RegWrite ALU control ovf

MemWrite

PC

Read

Instruction Memory

Read Addr 1

Register

Read Addr 2 File

Read Data 1

zero

Address Data

Address Instruction

Write Addr

ALU

Memory

Read Data

Write Data

Read

Data 2 Sign

Write Data

MemRead

Extend 16 32

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.12 

**Multiplexor Insertion**

Add

4

RegWrite ALU control ALUSrc

ovf

MemWrite

MemtoReg

PC

Read

Instruction Memory

Read Addr 1

Register

Read Addr 2 File

Read Data 1

zero

Address Data

Address Instruction

Write Addr

ALU

Memory

Read Data

Write Data

Read

Data 2 Sign

Write Data

MemRead

Extend 16 32

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.13 

**Clock Distribution** System Clock

clock cycle

Add

RegWrite MemWrite

PC

4

Read

Instruction Memory

Read Addr 1

Register

Read Addr 2 File

Read Data 1

ALUSrc

ALU control

ovf

zero

Address Data

MemtoReg

Address Instruction

Write Addr

ALU

Memory

Read Data

Write Data

Read

Data 2 Sign

Write Data

MemRead

Extend 16 32

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.14 

**(Almost) Complete Single Cycle Processor**

0

Add

4

Shift left 2

Add

1

PCSrc

RegDst

Instr[25-21]

RegWrite

Read Addr 1

ALUSrc MemtoReg MemWrite

ovfzero

PC

Read

Instruction Memory

Instr[20-16] 0

Register

Read Addr 2 File

Read Data 1

Address Data

1

Address Instr[31-0]

Write Addr ALU

Memory

Read Data

1

Instr[15 -11]

Instr[15-0]

Write Data

Read

Data 2 Sign

0 1

Write Data

MemRead

0

Extend 16 32

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.15 

**Adding the Control**

• Selecting the operations to perform (ALU, Register File and Memory read/write)

• Controlling the flow of data (multiplexor inputs) • Information comes from the 32 bits of the instruction 31 25 20 15 5 0

10

R-type:

op = 0 rs rt rd shamt funct 31 25 20 15 0

I-Type: op rs rt address offset

• Observations

– op field always in bits 31-26

– When op field is 0, funct field used from bits 5-0

– Operand locations regular – rs always 25-20, rt always 20-15, etc.

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.16 

**(Almost) Complete Single Cycle Processor** 0

Add

4

Instr[31-26] Instr[5-0]

Control Unit

Shift left 2

Add

1

PCSrc

RegDst

Instr[25-21]

RegWrite

Read Addr 1

ALUSrc MemtoReg MemWrite

ovfzero

PC

Read

Instruction Memory

Instr[20-16] 0

Register

Read Addr 2 File

Read Data 1

Address Data

1

Address Instr[31-0]

Write Addr ALU

Memory

Read Data

1

Instr[15 -11]

Instr[15-0]

Write Data

Read

Data 2 Sign

0

1

ALUcontrol

Write Data

MemRead

0

Extend 16 32

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.17 

**ALU Control**

• ALU's operation based on instruction type and function

code:

ALU control input

Function

0000 and

0001 or

0010 xor

0011 nor

0110 add

1110 subtract

1111 set on less than

• Notice that we are using different encodings than in the book (and different than you have chosen for your project)

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.18 

**ALU Control (cont.)** • So describe with a truth table

Four truth tables

Instr op Opcode funct action ALUcontrol

lw 100011 xxxxxx add 0110

sw 101011 xxxxxx add 0110

beq 000100 xxxxxx subtact 1110

add 000000 100000 add 0110

subt 000000 100010 subtract 1110

and 000000 100100 and 0000

or 000000 100101 or 0001

xor 000000 100110 xor 0010

nor 000000 100111 nor 0011

slt 000000 101010 slt 1111

12 inputs

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.19 

**(Almost) Complete Datapath w/ Control** 0

Add

4

1

ALUControl

Shift left 2

Add

PCSrc

/4

Instr[31-26] Instr[5-0]

Control Unit

Branch

ALUSrc

MemRead

~~MemtoReg~~

~~MemWrite~~

RegDst

Instr[25-21]

RegWrite

ovf

PC

Instruction

Memory

Read

Address Instr[31-0]

Instr[20-16] 0

Read Addr 1

Register

Read Addr 2 File

Read Data 1

zero ALU

Address

Data

Memory

Read Data

1

1

Instr[1

5 -11]

Instr[15-0]

Write Addr Write Data

Read

Data 2 Sign

00

Write Data

1

Extend 16 32

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.20 

**Main Control Unit**

**Instr ALU control**

**R-type**

000000

**lw**

100011

**sw**

101011

**beq**

000100

**RegDst ALUSrc MemReg RegWr MemRd MemWr Branch**

• Note that a multiplexor whose control input is 0 has a definite action, even if it is not used in performing the operation

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.21 

**R-Type Instruction Data/Control Flow** 0

Add

4

1

ALUControl

Shift left 2

Add

PCSrc

/4

Instr[31-26] Instr[5-0]

Control Unit

Branch

ALUSrc

MemRead

~~MemtoReg~~

~~MemWrite~~

RegDst

Instr[25-21]

RegWrite

ovf

PC

Instruction

Memory

Read

Address Instr[31-0]

Instr[20-16] 0

Read Addr 1

Register

Read Addr 2 File

Read Data 1

zero ALU

Address

Data

Memory

Read Data

1

1

Instr[1

5 -11]

Instr[15-0]

Write Addr Write Data

Read

Data 2 Sign

00

Write Data

1

Extend 16 32

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.22 

**lw Instruction Data/Control Flow** 0

Add

4

1

ALUControl

Shift left 2

Add

PCSrc

/4

Instr[31-26] Instr[5-0]

Control Unit

Branch

ALUSrc

MemRead

~~MemtoReg~~

~~MemWrite~~

RegDst

Instr[25-21]

RegWrite

ovf

PC

Instruction

Memory

Read

Address Instr[31-0]

Instr[20-16] 0

Read Addr 1

Register

Read Addr 2 File

Read Data 1

zero ALU

Address

Data

Memory

Read Data

1

1

Instr[1

5 -11]

Instr[15-0]

Write Addr Write Data

Read

Data 2 Sign

00

Write Data

1

Extend 16 32

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.23 

**sw Instruction Data/Control Flow** 0

Add

4

1

ALUControl

Shift left 2

Add

PCSrc

/4

Instr[31-26] Instr[5-0]

Control Unit

Branch

ALUSrc

MemRead

~~MemtoReg~~

~~MemWrite~~

RegDst

Instr[25-21]

RegWrite

ovf

PC

Instruction

Memory

Read

Address Instr[31-0]

Instr[20-16] 0

Read Addr 1

Register

Read Addr 2 File

Read Data 1

zero ALU

Address

Data

Memory

Read Data

1

1

Instr[1

5 -11]

Instr[15-0]

Write Addr Write Data

Read

Data 2 Sign

00

Write Data

1

Extend 16 32

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.24 

**beq Instruction Data/Control Flow** 0

Add

4

1

ALUControl

Shift left 2

Add

PCSrc

/4

Instr[31-26] Instr[5-0]

Control Unit

Branch

ALUSrc

MemRead

~~MemtoReg~~

~~MemWrite~~

RegDst

Instr[25-21]

RegWrite

ovf

PC

Instruction

Memory

Read

Address Instr[31-0]

Instr[20-16] 0

Read Addr 1

Register

Read Addr 2 File

Read Data 1

zero ALU

Address

Data

Memory

Read Data

1

1

Instr[1

5 -11]

Instr[15-0]

Write Addr Write Data

Read

Data 2 Sign

00

Write Data

1

Extend 16 32

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.25 

**Main Control Unit (cont.)**

**Instr ALU control**

**RegDst ALUSrc MemReg RegWr MemRd MemWr Branch**

**R-type** 000000 **lw**

100011

**sw**

101011

**beq**

000100

Depends

on Funct 1 0 0 1 0 0 0 0110 0 1 1 1 1 0 0 0110 X 1 X 0 0 1 0 1110 X 0 X 0 0 0 1

• Setting of the MemRd signal (for R-type, sw, beq) depends on the memory design (could have to be 0 or could be a X (don’t care))

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.26 

**Control Unit Logic**

• From the truth table can design the Main Control logic

Instr[31]

Instr[30]

Instr[29]

Instr[28]

Instr[27]

Instr[26]

R-type lw sw beq RegDst

ALUSrc

MemtoReg

RegWrite

MemRead

MemWrite

Branch

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.27 

**Review: Executing Jump Operations** • Jump operations have to

31 25 0

J-Type: op

jump target address

– Replace the lower 28 bits of the PC with the lower 26 bits of the fetched instruction shifted left by 2 bits

Add

PC

4

Shift

left 2

Read

Instruction Memory

4

28

Jump

address

Address Instruction

26

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.28 

**Adding the Jump Instruction**

Instr[25-0]

Shift

28

32

1

Add

26PC+4[31-28]

0

left 2

0

4

ALUControl

Branch

Jump

Shift left 2

Add

PCSrc

1

MemRead

Instr[31-26] Instr[5-0]

Control Unit

ALUSrc RegWrite

~~MemtoReg~~

~~MemWrite~~

Instruction

RegDst

Instr[25-21]

Read Addr 1

ovf

Read

Address

Memory

Instr[20-16]

Register

Data 1

zero

PC

Read

Address Instr[31-0]

0

Read Addr 2 File

ALU

Data

Memory

Read Data

1

Instr[1

1

Write Addr Write Data

Read Data 2

00 Write Data

5 -11]

Instr[15-0]

1

Sign

16 Extend 32

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.29 

**Adding the Jump Instruction**

Instr[25-0]

Shift

28

32

1

Add

26PC+4[31-28]

0

left 2

0

4

ALUControl

Branch

Jump

Shift left 2

Add

PCSrc

1

MemRead

Instr[31-26] Instr[5-0]

Control Unit

ALUSrc RegWrite

~~MemtoReg~~

~~MemWrite~~

Instruction

RegDst

Instr[25-21]

Read Addr 1

ovf

Read

Address

Memory

Instr[20-16]

Register

Data 1

zero

PC

Read

Address Instr[31-0]

0

Read Addr 2 File

ALU

Data

Memory

Read Data

1

Instr[1

1

Write Addr Write Data

Read Data 2

00 Write Data

5 -11]

1

**I~~n-class~~ As~~s~~essment!**

Instr[15-0]

Sign

16 Extend 32

**A~~ccess Code: He~~igh-ho** 

**Note: sharing access code to those outside of classroom or using access while outside of classroom is considered cheating** 

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.30

**Main Control Unit (cont.)**

**Instr ALU control**

**RegDst ALUSrc MemReg RegWr MemRd MemWr Branch Jump**

**R-type** 000000 **lw**

100011

**sw**

101011

**beq**

000100 **j**

000010

on Funct 1 0 0 1 0 0 0 0 Depends

0110 0 1 1 1 1 0 0 0 0110 X 1 X 0 0 1 0 0 1110 X 0 X 0 0 0 1 0 X X X X 0 0 0 X 1

• Setting of the MemRd signal (for R-type, sw, beq) depends on the memory design

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.31 

**Model Truth Table in VHDL**

Use selected signal statement (data flow)

**-- input : 5-bit addr; output: 32-bit sel**

**with addr select**

**sel <= x”00000001” when b”00000”,**

**x”00000002” when b”00001”,**

**… -- more cases**

**x”80000000” when b”11111”;**

****Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.32 

**Clock Distribution System Clock**

clock cycle

Add

4

Shift

Add

0

1

PCSrc

ALUOp

Instr[31-26]

Branch

Control

Unit

left 2

MemRead

MemtoReg

~~ALUSrc~~

MemWrite

RegDst

Instr[25-21]

RegWrite

ovf

Instruction

Memory

Read

Instr[20-16]

Read Addr 1 Register

Read Addr 2

Read Data 1

zero

Address Data

00 0

PC

Address Instr[31-0]

1

File

Write Addr

Read

ALU

Memory

Read Data

1

Instr[1

5 -11]

Instr[15-0]

Write Data

Data 2 Sign

1

ALU

Write Data

16 Extend 32 Instr[5-0]

control

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.33 

**Operation**

• We wait for everything to settle down

– ALU might not produce “final answer” right away

– Memory and RegFile reads are combinational (as are ALU, adders, muxes, shifter, signextender)

– Use write signals along with the clock edge to determine when to write to the sequential elements (to the PC, to the Register File and to the Data Memory)

• The clock cycle time is determined by the logic delay through the longest path

We are ignoring some details like register

setup and hold times

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.34 

**Worst Case Timing (Load Instruction)System Clock**

clock cycle

0

Add

4

ALUOp

Instr[31-26]

Branch

Control

Unit

Shift

left 2

MemRead

MemtoReg

~~ALUSrc~~

Add

1

PCSrc

MemWrite

RegDst

Instr[25-21]

RegWrite

ovf

Instruction

Memory

Read

Instr[20-16]

Read Addr 1 Register

Read Addr 2

Read Data 1

zero

Address Data

00 0

PC

Address Instr[31-0]

1

File

Write Addr

Read

ALU

Memory

Read Data

1

Instr[1

5 -11]

Instr[15-0]

Write Data

Data 2 Sign

1

ALU

Write Data

16 Extend 32 Instr[5-0]

control

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.35 

**Worst Case Timing (Load Instruction)**

Clk

Clk-to-Q

PC

Old Value New Value

Rs, Rt, Rd, Op, Func

ALUctr

Instruction Memory Access Time

Old Value New Value

Delay through Control Logic

Old Value New Value

ExtOp Old Value New Value ALUSrc Old Value New Value MemtoReg Old Value New Value RegWr Old Value New Value

Register

Write Occurs

busA busB

Register File Access Time

Old Value New Value Delay through Extender & Mux

Old Value New Value ALU Delay

Address Old Value New Value Data Memory Access Time

busW Old Value New Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.36 

Design Conception **Tools: ASMs**

No

No

Design Entry

Compile & Elaborate Simulate (functional)

Design

correct?

Yes

Physical Design

Simulate (timing)

Timing

requirements

met?

Yes

Chip Configuration (Program Device)

**HDLs (VHDL)**

****

****[ MODIFIED Figure 2.35 from the 281 textbook ]

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.37

**Automated Testing Framework** tb\_SimplifiedMIPS

MIPS Assembly File

Processor.vhd MIPSProcessor.vhd Other VHDL files

Other VHDL files

your implemented

your implemented Other VHDL files

your implemented

**MARS vcom**

Imem.hex

(instruction binary)

**vsim**

mars\_dump.out modelsim\_dump.out

Success!

or

Failed here…

**Dump**

**Compare**

**Quartus** timing.txt

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.38 

**Acknowledgments**

• These slides contain material developed and copyright by:

– Joe Zambreno (Iowa State)

– David Patterson (UC Berkeley)

– Mary Jane Irwin (Penn State)

– Christos Kozyrakis (Stanford)

– Onur Mutlu (Carnegie Mellon)

– Krste Asanović (UC Berkeley)

Duwe, Spring 2018 © ISU CprE 381: Processor Design Lec06.2.39 