EEL4713
Lab 2
Spring 2018

Overview: This EEL-4713 assignment will build up to finishing a functional single cycle MIPS.

Please use the ENTIRE 2 weeks for this labs as it is long and tedious.

Please help us by reporting inconsistencies or bugs you find in the lab to TAs so we can adjust and improve!

In this assignment, you will perform the following tasks:

1. ALU Controller
2. Controller
3. Memories
4. Single Cycle MIPS

EEL4713
Lab 2
Spring 2018

Read chapter 4.4 ALU Control / Main Control Unit section of your book. Your book is pretty much the Bible of Computer Architecture so it’s worth your time. If you ask me questions, I will ask you what the book said before answering your question.

EEL4713
Lab 2
Spring 2018

1. ALU Controller

4.4 ALU Controller section will walk you through how to make a basic ALU controller. Before continuing, fully understand figures 14.12, 14.13, and 14.14. Start by making this basic ALU and then extend it as more instructions are added.

Remember that the mips you make will have slight differences than the one in the book. We might have some more muxing or some more control signals which aren't shown. The book diagrams are simplifications.

It is important for you to understand that the ALU gets a control line "ctrl" which selects its configuration/operation. The signal "ALUop" is sent from the main controller (which is next) to the alu controller. This is a really important distinction. Having two controller **is required.** Having two controllers is a key optimization mips designers created to reduce the amount of decode logic.

Start with an ALUop of two and grow it as needed. The exact translation of instructions to ALUop is not standardized (similar to the ALU control not being standardized).

Make a table that extends 4.12 as more instructions are added. Make this digitally since you will be using it for the rest of the semester as we add complexity to the mips.

Note: Try not to pass the full opcode to the ALU controller, but this will get increasingly difficult. This will not be penalized this, but if it is used the decode logic for it must be very minimal.

2. Controller

Once again, the book is your best resource here. Signal names might vary slightly. The controller essentially does the decode logic. Do your best to keep the complexity of the decode logic to a minimum.

Start with your table. Add/Look at columns "instruction name", "opcode", "alu func" to see how each instruction configures the ALU. Use localparam to define each instruction opcode and each alu configuration. Now make the controller correctly configure the ALU via ALUop. Use an always block and a case. You will likely need to go back and add more cases to your alu controller.

Always include defaults. You should understand that the don't cares allow for much simpler logic equations which makes for faster clock cycles. For this lab, don't actually use don't cares, rather just assign to 0 or whatever the major of signals require. By doing this you only need to specify if an opcode explicitly needs a control signal set rather than the default.

At this point you have aluop mapped to the instruction opcode. Now do this for the other control signals. Start table 4.18. Use separate case for the control signals. This will allow you to debug much easier.

Q: Elaborate on the logical difference between case and if else. How does it affect synthesis/hardware implementation? Show screenshots of RTL viewers.

3. Memories

So making a single cycle mips is tricky because we have many synchronous components. I.e. program counter reg, memories, reg file, hi and lo, etc… A decent solution to this is set certain synchronous values on the clk posedge and others on the negative edge. Additionally we can clock divide and use faster clocks for memories.

Create a RAM for data mem and a rom for instruction mem. 32 bit by 64 words, don't register outputs, don't have rden (we will just ignore output), MK9 embedded bram, First Word Fall Through (new data when reading and writing to memory), .mif (edit to make 64 words) (only for instruction mem), check box for instantiation template file

4. Single Cycle Mips

Support the following instructions. Assume I will provide you an assembly file. You will use your assembler to create a mif, upload that mif to the instruction memory, and then execute the code.

//ADD
ADD
ADDI
ADDIU
ADDU

//AND
AND
ANDI

//BRANCH
BEQ
BNE

//JUMP
J

JAL
JR

//LOAD
LBU
LHU
LUI
LW

//OR
NOR
OR
ORI

//SET
SLT
SLTI
SLTIU
SLTU

//SHIFT
SLL
SLLV
SRA
SRL
SRLV

//STORE
SB
SH
SW

//SUB
SUB
SUBU

//XOR
XOR
XORI

Your mips must work in hardware. Use a 50MHz clock. Create and sdc file to set timing constraints for I/O.

EEL4713
Lab 2
Spring 2018

Q. tba
Extra Credit (): tba
Extra Credit (): tba
Extra Credit (): tba

DELIVERABLES:

For this lab, you will submit a zipped file of your verilog code and a document with your answers to lab questions on them. Be prepared to talk about code and lab questions in the lab section.
I.e. zip up src/ and sim/

Deliverables	Grade
Questions throughout lab doc	10%
ALU Controller, Controller	5%
Memories	5%
Single Cycle Mips	80%

Extra Credit	15%