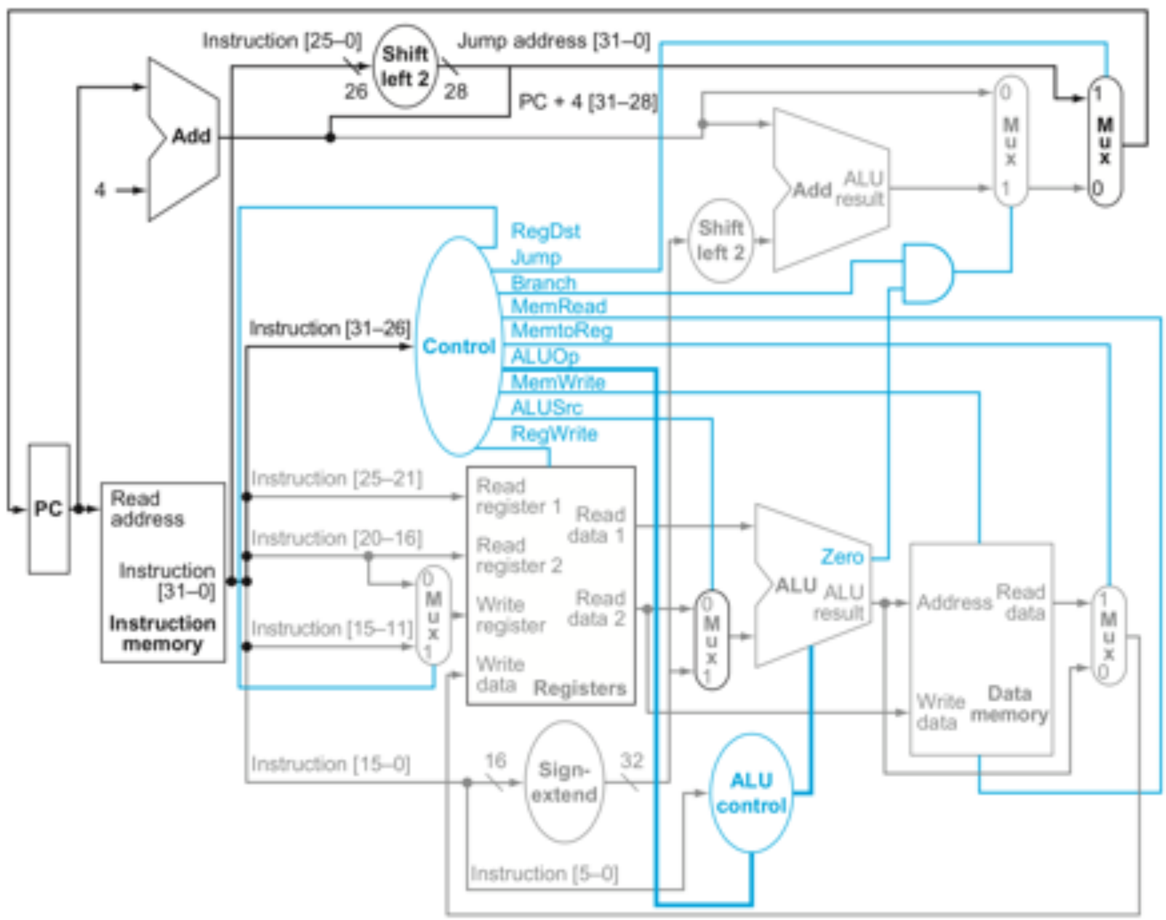
Project Description

This project will build the single-cycle version of the MIPS processor as documented in the book on page 271 and in the following copy of the diagram. The project will build the processor utilizing the Register File and the ALU previously completed in Projects 1 & 2. The Controller will be supplied by the instructor.



Modifications Required

1) There will be a modification required by the Register File. As noted in class the R0 of the MIPs is always 32'b0. The simple modification is to change the read of the register file to:

assign ReadData = (ReadAddress != 5'b0) ? Register[ReadAddress] : 32'b0;

Since the read case always returns a 0 for register 0 there is no need to adjust the write circuitry.

2) Since the Sign Extension on Instruction[15:0] is now performed in front of the ALU we no longer need to perform sign extension in the ALU.

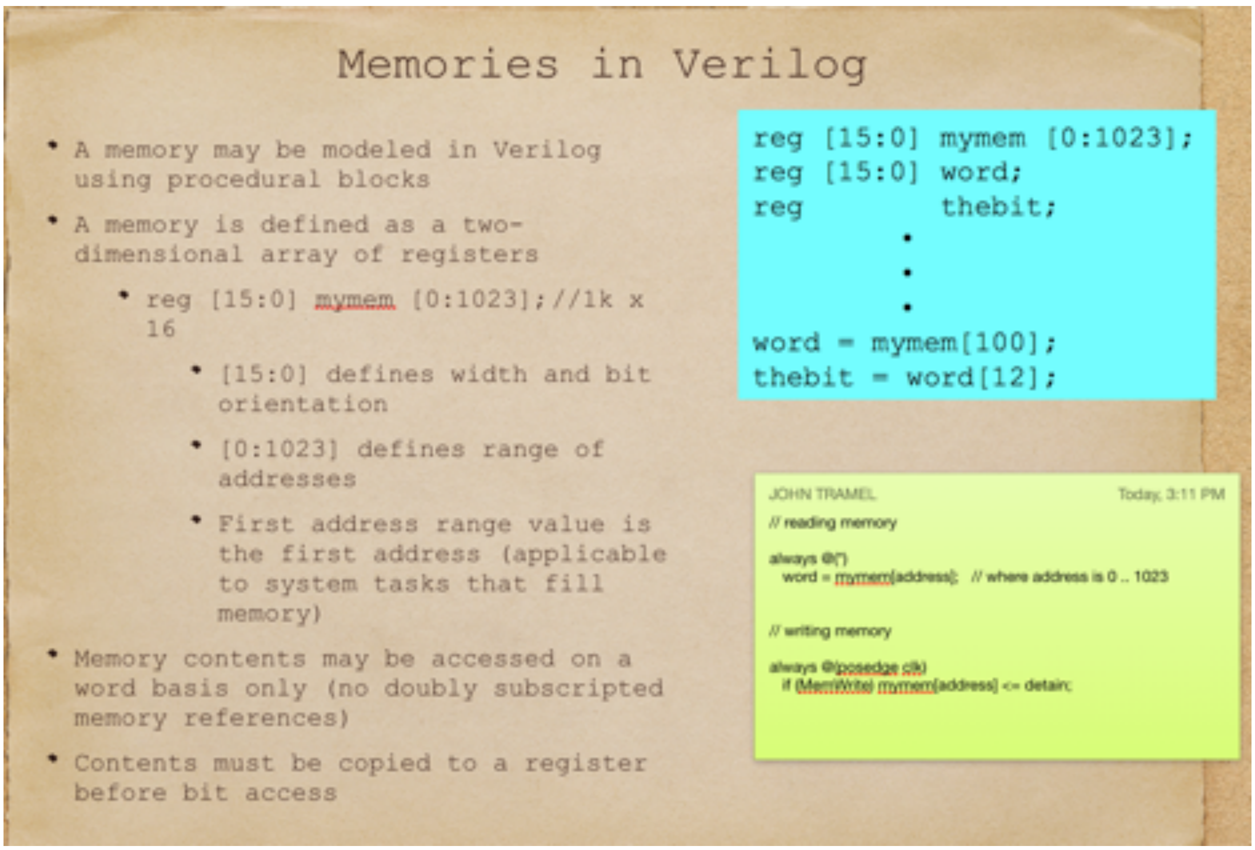
3) We want to execute the instruction LUI so the ALU will need to have an instruction added that will assign Y = {B[15:0],16'b0}; The new ALU select for this will be 0x12.

4) Since the instruction memory we are utilizing is 1024 x 32 we will need to shift the output of the PC >> 2 when providing an address to the instruction memory.

5) The current design in the block diagram above is able to execute a BEQ (branch if equal to zero) instruction. That is why the term (Zero & Branch) are used as the mux select in the upper right of the diagram. We will have a BNE (branch if not equal to zero) instruction. We will then need to generate the term (BranchNE & ~Zero). We will also change the name of Branch to BranchE. The resultant term will now be MuxSel = (BranchE & Zero) | (BranchNE & ~Zero);

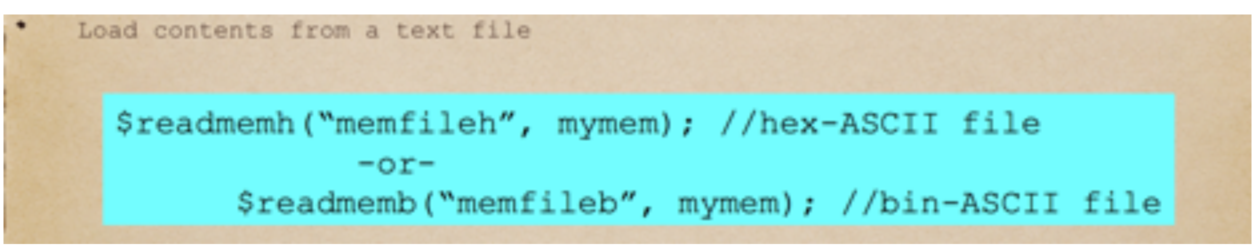
Memory Implementation

The implementation of the memory will first be using a Verilog model as described in the following slide. After that an approach will be introduced that will allow a synthesizable implementation.



Loading Memory

The instruction memory will be loaded at reset and will use the following technique to load the instructions into the memory. This should be in the test bench and will be demoed in class.



Instructions

Our machine will execute a subset of the MIPS instructions. The following documents all of the instructions that our machine will be able to decode and execute. This work is being done for you right now in the Controller. You will responsible in the future for generating your own machine code to be executed by the memory.

Single Cycle Instructions

Instruction	31:26	25:21	20:16	15:11	10:6	5:0
add rd, rs, rt	0	rs	rt	rd	0	0x20
addu rd, rs, rt	0	rs	rt	rd	0	0x21
addi rt, rs, immed	8	rs	rt	Immediate		
addiu rt, rs, immed	9	rs	rt	Immediate		
and rd, rs, rt	0	rs	rt	rd	0	0x24
andi rt, rs, immed	0xC	rs	rt	Immediate		
nor dr, rs, rt	0	rs	rt	rd	0	0x27
or rd, rs, rt	0	rs	rt	rd	0	0x25
ori rt, rs, immed	0xD	rs	rt	Immediate		
sub rd, rs, rt	0	rs	rt	rd	0	0x22
subu rd, rs, rt	0	rs	rt	rd	0	0x23
xor rd, rs, rt	0	rs	rt	rd	0	0x26
xori rt, rs, immed	0xE	rs	rt	Immediate		
lui rt, immed	0xF	0	rt	Immediate		
slt rd, rs, rt	0	rs	rt	rd	0	0x2A
sltu rd, rs, rt	0	rs	rt	rd	0	0x2B
slti rt, rs, immed	0xA	rs	rt	Immediate		
sltiu rs, rt, immed	0xB	rs	rt	Immediate		
beq rs, rt, immed	4	rs	rt	Offset		
bne rs, rt, immed	5	rs	rt	Offset		
lw rt, address	0x23	rs	rt	Offset		
sw rt, address	0x2B	rs	rt	Offset		
nop	0	0	0	0	0	0

Deliverables
3/10 - Diagram of Single Cycle Processor with modifications and every signal/bus named
3/17 - Verilog implementation of Single Cycle Processor
3/24 - Debugged Single Cycle Processor running in simulation

3/24 - Report documenting design and what you have accomplished