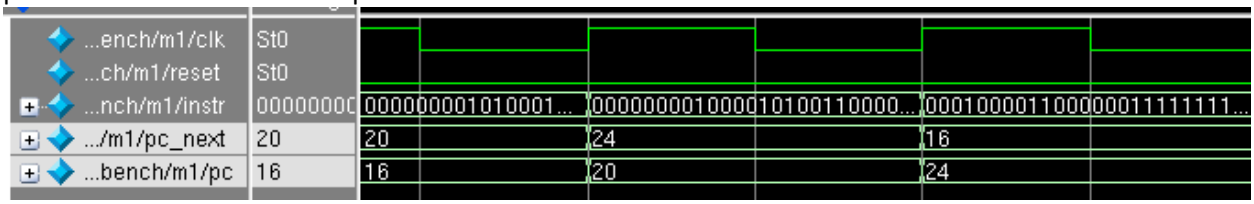
Modeling the Single-Cycle MIPS Processor in Verilog
By Ashtyn Moehlenhoff & Torben Rasmussen
ECE472 – Fall 2010

Part1 – Building and Simulating the Processor Model

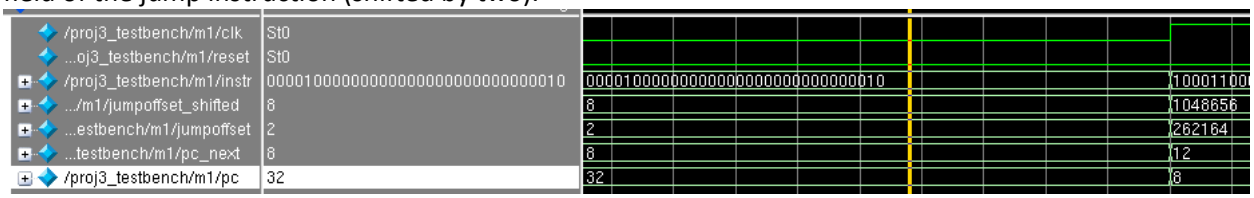
Building the processor is very straight-forward. We created a test bench in order to simulate the processor with a 100ns clock period.



You can see in the above figure that the pc_next = pc+4 for each of these instructions. Also note that these instructions are not jump/branch instructions.

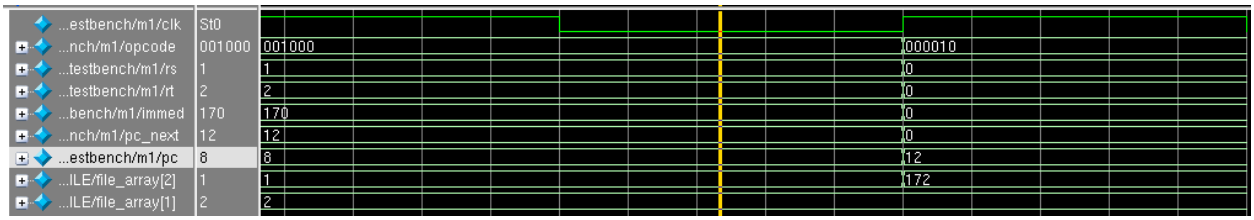
Part2 – Extending the Processor Design

Jump: For jump we created a Jump Mux to choose from the output of the PCMUX and or the immediate field of the jump instruction (shifted by two).



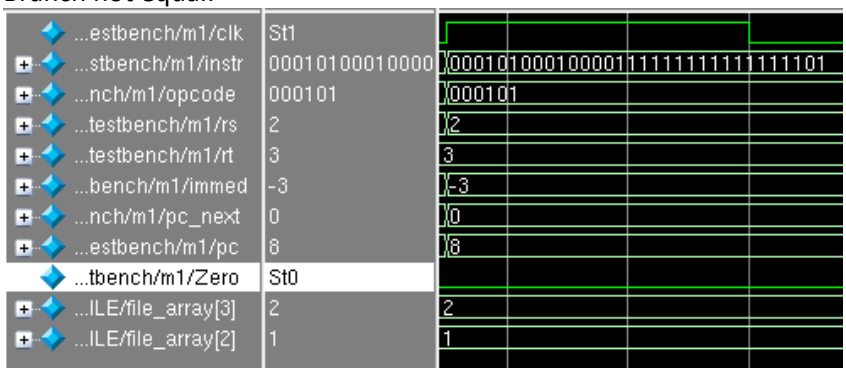
This instruction represents PC = Jump Address = IMM << 2. In our case, pc_next = 2 << 2 = 8. The PC in the cycle following the jump instruction is 8.

Add Immediate:



The opcode 0b001000 is for the add immediate instruction: R[rt] = R[rs] + IMMEDIATE. So in our case, R2 = R1 + IMMEDIATE which equals 2 + 170 = 172. Register 2 is updated in the next cycle to be 172.

Branch not equal:



Our instruction was bne r2, r3, -3. This subtracts r2 and r3, and when the zero flag is NOT set (they are not equal), pc = pc_next + immediate. In our example pc is set to (8 + 4) – (3<<2) = 0 because r3 != r2.

Part3 – Simulating the modified processor

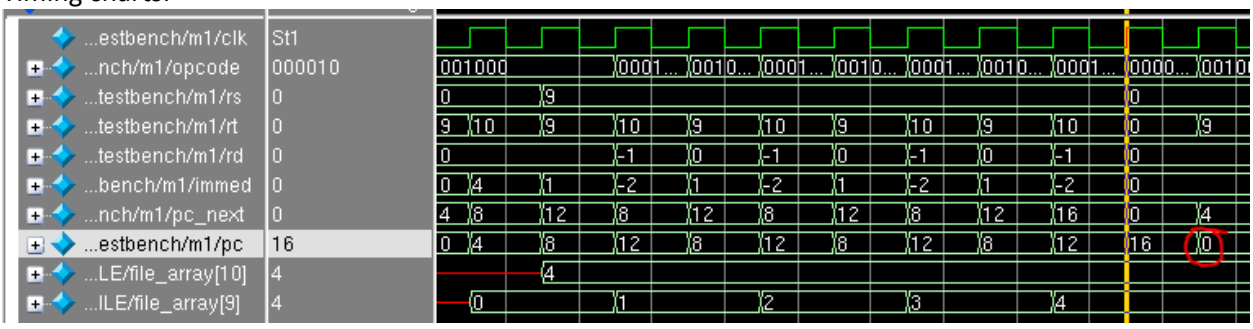
Assembly code (from MARS):

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20090000	addi $9,$0,0x0000	2: addi $t1,$zero,0
	0x00400004	0x200a0004	addi $10,$0,0x0004	3: addi $t2,$zero,4
	0x00400008	0x21290001	addi $9,$9,0x0001	5: addi $t1,$t1,1
	0x0040000c	0x152afffe	bne $9,$10,0xfffe	6: bne $t1, $t2,LOOP
	0x00400010	0x08100000	j 0x00400000	7: j START

Edited Rom32.v:

```
5'd0 : data_out = { 32'h20090000 };
5'd1 : data_out = { 32'h200a0004 };
5'd2 : data_out = { 32'h21290001 };
5'd3 : data_out = { 32'h152afffe };
5'd4 : data_out = { 32'h08000000 };
```

Timing charts:



Above you can see our program being run. First, registers 9 and 10 are initialized. Next, register 9 is incremented and register 9 and 10 are compared. If these registers are equal, it loops back to increment register 9. Finally, when register 9 and 10 are equal, the program jumps to the start (circled in red in the above diagram).