Microprocessor Control Logic - 2
Load and Jump Instructions

DDCO Assignment 6

October 19, 2019

Building upon assignment 5 (in which arithmetic and logic instructions were implemented) the intent of this assignment is to enhance the control logic to implement a load and a jump instruction.

The datapath has been augmented as follows. For the load instruction, data out from RAM has been connected to register file input. In order to enable above input to register file (instead of usual input from ALU) when a load instruction is being executed, the sel input of the register file is used, which is supplied by the control logic. For the jump instruction, bits in the instruction (IR) that represent the jump offset are connected to the offset input of the PC module, whose sub input is derived from the jump output supplied by the control logic. Only the sub input is used (and not the add input) since the jump is backwards only (offset is subtracted from PC). Also, the jump occurs only if the cout output of the ALU is high, ensured by AND of cout with jump in mproc module to generate sub.

The format of the instructions is shown in figure 1.

15	14	13				9	8 6	5					0
1	0	X	X	X	X	X	wr addr	X	X	X	X	X	X

15 0
data to be loaded into register

Load instruction

15	14	13					8	7 0
0	1	X	X	X	X	X	X	jump offset

Jump instruction

Figure 1: Format of load and jump instructions.

The load instruction occupies two 16-bit words, the first of which specifies the address of the register (wr_addr) into which the data is to be loaded, while the second contains the 16-bit data to be loaded[1]. The jump instruction

[1]Note that wr_addr field occupies the same bit positions as in arithmetic and logic instructions, in order to simplify wiring.

specifies an 8-bit offset. If cout is 1, then the offset is subtracted from the PC, else the next instruction is executed. Note that while ALU and jump instructions execute in two clock cycles, the load instruction (which loads a 16-bit word from memory) would require three clock cycles to execute. As a result, the FSM in the control logic would have three states.

Only contents of control_logic module need to be modified (from what you did in assignment 5) in this assignment. As mentioned above, the control logic now needs to supply two more outputs (compared to assignment 5 control logic) sel and jump, which are to be asserted (for one or more clock cycles) during the load and jump instructions respectively. Also note that the logic used to generate pc_inc inside the control logic needs to change. Previously, pc_inc was asserted for only one of the three clock cycles required to execute an instruction, but for the load instruction pc_inc needs to be asserted twice in three clock cycles. Note that even while executing a jump instruction, PC increment should be allowed to happen followed by offset subtract from PC (if cout is 1). Also, the inc and sub inputs to PC cannot be high at the same time, so the jump output of control logic (used to generate sub input to PC) must be accordingly generated.

1 Design and Simulation

As in previous assignment, the commands to simulate:

```
iverilog -o tb_mproc_mem lib.v pc.v alu.v reg_alu.v mproc.v mproc_mem.v tb_mproc_mem.v
vvp tb_mproc_mem
```

followed by waveform observation with the command:

```
gtkwave tb_mproc_mem.vcd
```