Microprocessor Control Logic - 1
ALU Instructions

DDCO Assignment 5

October 14, 2019

Just as a microprocessor can be thought of as the control center of a computer, a microprocessor's control logic can be thought of as the control center of the microprocessor. The task in this assignment is to design the control logic, which is essentially a finite state machine.

The control logic is to be designed for a 16-bit microprocessor whose other parts, an ALU, register file, and PC were designed in previous assignments. For the current assignment above modules have been instantiated inside a bigger module called mproc, which represents the microprocessor. In addition to above, another module called ir which implements the instruction register is also instantiated inside mproc. Finally, also instantiated within mproc is a module called control_logic, contents of which have to be written to complete this assignment. Modules instantiated inside mproc have been connected together as discussed in class for the microprocessor to function.

The input to the control_logic is cur_ins (the instruction stored in the ir) and its outputs are rd_addr_a, rd_addr_b, wr_addr (read and write port addresses to the reg_file), op (alu operation) pc_inc, load_ir and wr_reg (to pc, ir and reg_file respectively). The latter three signals need to come from the FSM to be implemented within control_logic, while the former signals need to be derived (in a straightforward manner) from the cur_ins input.

Currently, the control logic needs to support no load/store or jump instructions but only arithmetic and logic instructions, each of which requires three clock cycles to execute. In the first cycle (fetch) the instruction address is supplied to external memory and the instruction is fetched from memory. In the second cycle (decode/execute), the instruction is stored in ir, decoded and executed by the alu, and its output stored in reg_file.

In order to do so, a memory module called ram_128_16 has been provided that implements 128 memory words each of 16-bit length. The memory is connected to the microprocessor in the module mproc_mem. Interestingly enough (but not surprising if you think about it) mproc_mem only has two inputs (clk and reset) and no outputs so when it is instantiated inside the supplied testbench tb_mproc_mem, there is no way to supply test vectors. So to enable testing, the first few locations of ram_128_16 have been initialized with a few instructions, which the microprocessor (with cor-

rect control_logic) will start fetching and executing, enabling testing of the implemented logic. For further testing, more instructions can be added to initial block of the ram_128_16 module.

Each instruction is 16-bits and has the format shown in figure 1.

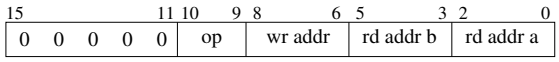

Figure 1: Format of arithmetic and logic instructions.

Since only the four arithmetic and logic instructions are being implemented, the registers in the register file, which are all initialized to 0, cannot be loaded with arbitrary values. So a peculiar problem arises because each above instruction will output 0 if its inputs are 0. So in order to have non zero register values (to test proper instruction execution) register 0 is initialized to 16'hffff.

1 Design and Simulation

Similar to previous assignments, 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
```