CS220: Assignment#6

1. [100 points] Implement a register file having 32 registers where each register is 16-bit wide. The register file interface needs to be designed such that it has two read ports and one write port. You need to write an interface module to access the register file. The module should have the following inputs: two read addresses, one write address, one write data, and three bits that specify which of the three ports have valid address inputs. The outputs of the module are the two read data values. Initialize the registers with zero values. Write a top-level instruction processor module to demonstrate that the register file works using the test program given at the end of this document. This module uses eight instructions to interact with the register file access module. These instructions are defined below along with the binary encoding of them in parentheses. Note that implementing these instructions may require additional modules to carry out computation that takes values read from the register file as inputs and produces the value to be written to the register file.

- No read, one write (000)
- One read, no write (001)
- Two reads, no write (010)
- One read, one write (011)
- Two reads, one write (100)
- Two reads, add two read values, write the result, ignore overflow (101)
- Two reads, subtract second read value from the first value, write the result, ignore overflow (110)
- One read, shift the read value left, write the result (111)

Note that the shift amount can be at most 15. For instructions 101, 110, and 111, the writing of the result should be done after the operation completes. Assume that the operation takes 16 cycles. Assume that register read takes two cycles and register write takes two cycles. After an instruction is completed, a "done" signal is generated which tells the top module to start the next instruction. Note that during the execution of an instruction such as 101, 110, 111, the register file access module will have to be invoked twice (once for reading and once for writing). When an instruction completes, the top module displays some output related to the completed instruction before starting the next instruction. For each completed instruction, what output needs to be displayed is defined in the following.

- If instruction is 000, nothing to display.
- If instruction is 001, display the address of the register and the 16-bit value read.
- If instruction is 010, display the two 16-bit values read along with the corresponding register addresses.
- If instruction is 011, display the address of the register read and the 16-bit value read.
- If instruction is 100, display the two 16-bit values read along with the corresponding register addresses.
- If instruction is 101 or 110 or 111, display the address of the register written and the 16-bit value written.

Test program: Notice that the designed hardware is a miniature processor with a register file and eight instructions. It has a 16-bit adder/subtracter and a left shifter. In the following, I list the C statements of a program along with translation into assembly language of this miniature processor. I use the binary command encoding as the instruction names. The registers are numbered \$0 to \$31. All numbers in the following are listed in decimal. Use 16-bit two's complement representation for binary conversion. The top module needs to show correct execution of this program.

```
   C statements                  Assembly language instructions
-------------------------------------------------------------------
short a, b, c, d;
a = 17;                           000  $1  17
printf("%h", a); b = -9;          011  $1  $2  -9
printf("%h %h", a, b); c = 65;    100  $1  $2  $3  65
printf("%h %h", b, c);            010  $2  $3
d = 8*c - 512*(a + b);            111  $3  $5   3      // 8*c
                                  101  $1  $2  $4      // a + b
                                  111  $4  $4   9      // 512*(a + b)
                                  110  $5  $4  $6      // d
printf("%h", d);                  001  $6
```

Submission: Email the files as attachment to cs220spring2021submit@gmail.com. The subject line of the email MUST be the following (replace N by your group number): Assignment#6 submission for Group#N