University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Science

EECS 150 R. H. Katz
Spring 2007

Problem Set # 7 (Assigned 6 March, Due 16 March)

In this assignment, you are to implement the datapath, to the register level, and a Moore Machine controller for a very simple processor. Your machine has separate instruction and data memories (Harvard architecture), each of which is 256 by 8 bits. The machine is a zero-address expression stack architecture, that is, most instructions reference the top elements of a stack organized in the data memory. The stack grows from data memory location 0 towards the top of the data memory. Program variables are allocated from the top of the data memory downwards. It is up to the compiler or the assembly language programmer to make sure that the expression stack and the variable memory don't overlap. No special hardware support is provided. Instructions are executed from instruction memory, starting from location 0 on reset.

Instruction Set

Your instruction set includes the following instructions (we do not specify every possible instruction here, just a representative subset: PUSH/POP, ADD/SUB, Conditional Branches, and Jump Subroutine/Return from Subroutine). Note that TOS stands for "Top of Stack", a special processor register that holds the address of the current top of the stack in the data memory. On reset, it is initialized to 255, just as the PC is initialized to 0.

PUSH <addr>	TOS ← TOS + 1; Data Memory[TOS] ← Data Memory[addr];
POP <addr>	Data Memory[addr] ← Data Memory[TOS]; TOS ← TOS – 1;
ADD	Data Memory[TOS – 1] ← Data Memory[TOS] + Data Memory[TOS – 1]; TOS ← TOS – 1;
SUB	Data Memory[TOS – 1] ← Data Memory[TOS] - Data Memory[TOS – 1]; TOS ← TOS – 1;
BRZ <addr>	IF Data Memory[TOS] = 0 THEN PC ← addr;
BRN <addr>	IF Data Memory[TOS]<7> = 1 THEN PC ← addr;
JSR <addr>	TOS ← TOS + 1; Data Memory[TOS] ← PC + 1; PC ← addr;
SRR	PC ← Data Memory[TOS]; TOS ← TOS – 1;

These instructions are encoded in 8 bit parcels as follows. The first parcel is the opcode. PUSH is encoded as 10000000_2, POP as 01000000_2, and so on to SRR as 00000001_2. For those instructions with an address field, like PUSH/POP and the branch instructions above, a second 8 bit parcel holds the target address. PUSH/POP refers to the data memory while BRZ/BRN and JSR refer to the instruction memory.

Memory Interface

The memory interface is kept simple. It is synchronous, running on the same clock as the processor. Assert Read and an address following the positive clock edge and data will be available for latching on the following positive edge. Assert Write and an address following the positive clock edge and the data will be written at the following positive edge.

What You Are To Do

1. Given the partial instruction set specification given above, design a datapath to the level of registers, arithmetic units (adders/subtractors), and busses to support the instruction set. Identify the control signals associated with each element, such as Load, Multiplexer/Select, Arithmetic Operations, Tri-state enables, and so on.
2. Write the controller Verilog for a "high level" register-transfer description of the instruction set subset described above. Your Verilog should describe the high level processing of instruction fetch, instruction decode, and instruction execution. Treat instruction and data memory as nothing more than arrays of 256 8-bit words.
3. Show the Moore Machine controller state diagram for Instruction Fetch, Instruction Decode, and Instruction Execution for the portion of the instruction set specified above. For the purposes of simplifying the decode step, assume that there are no other instructions in the instruction set. For each state, show the control signals that are to be asserted in that state. Remember to include an initial state, entered when an external Reset signal is asserted, that sets the PC to 0, TOS to 255 (11111111_2), and initializes any other registers to their initial values.
4. Write the controller Verilog that corresponds to the state diagram of your solution to Problem 3.
5. Go back to working on your project!