CSE 141L
Lab 1: 8-Bit Instruction Set Architecture

Due Wednesday, July 11.

PART 1
In this lab, you will design the instruction set for a processor. You will design the hardware for that processor in subsequent labs. This will be an 8-bit processor which you will optimize for 3 simple programs (described on the next page). For this lab, you will design the instruction set and instruction formats, and code three programs to run on your instruction set. Given the extreme limit on instruction bits, the target programs and their needs should be considered carefully. The best design will come from an iterative process of designing an ISA, then coding the programs, then redesigning the ISA.

Your instruction set architecture should feature fixed-length instructions 8 bits wide. Your instruction-set specification should describe:

- what operations it supports and what their opcodes are.
- how many instruction formats it supports and what they are (in detail – how many bits for each field, and where they’re found in the instruction). Your instruction format description should be detailed enough that someone could write an assembler (a program that creates machine code from assembly code) for it.
- number of registers, and how many; general-purpose or specialized.
- the size of main memory.
- addressing modes supported (this applies to both memory instructions and branch instructions).

In order to fit this all in 8 bits, the memory demands of these programs will have to be small. For example, you will have to be clever to have a conventional main memory even as big as 256 bytes. However, for the programs you are going to run, you may not need all that. You should consider how much data space you will need before you finalize your instruction format. You can assume that instructions are stored in a different memory, so that your data addresses need only be big enough to hold data. You should also notice that these programs probably don’t need procedure calls and stack pointers. This will be an 8-bit machine otherwise, meaning that memory is byte-addressable, and registers and all important data types are all 8 bits.

When implemented, this will be a single-cycle machine, so realize that there is a limit to what can be done in a single cycle. For example, assume you cannot read two memory locations (or read one and write one) in a single cycle. However, you will be able to design a register file that reads two and writes one register in a cycle.

To simplify the ISA design process, you need only optimize for the following three goals:
1. Minimize dynamic instruction count (i.e., the number of instructions executed during the running of a particular program).
2. Simplify your processor hardware design.
3. Use the least amount of necessary hardware as possible, in order to minimize costs.

You are welcome to also optimize for other things (e.g., cycle time, ease of pipelining), but if you do so, we will expect you to discuss that optimization intelligently, and these three goals should still have the highest priority.

Generic ISAs (that is, ISAs that will execute other programs just as efficiently as those shown here) will be seriously frowned upon. We really want you to optimize for these programs. There is only one mandatory instruction: HALT. When the CPU finds this instruction, it stops.

You will turn in a lab report no more than six pages long. The report will answer the following questions. In describing your architecture, keep in mind that the person grading it has much less experience with your ISA than you do. It is your responsibility to make everything clear.
For all the labs, your report will have two parts: a lab report (in this case, your ISA description) and the answers to all questions. Since, particularly in this case, a good report will already have the answers to many of the questions, it is okay to put something like “2. See section 3 of ISA description” as an answer. But for short answers, it is probably wiser to just repeat the answer.

Questions for lab 1:
1) What instruction formats are supported and what do they look like? Give an example of each.
2) What instructions are supported and what are their opcodes?
3) How many registers are supported? Is there anything special about the registers?
4) How are branches supported? What is the maximum branch distance supported?
5) What addressing modes are supported? How are addresses calculated (including instruction addresses for branches)? Give an example.
6) How large is the main memory?
7) In what ways did you optimize for dynamic instruction count?
8) In what ways did you optimize for ease of design?
9) In what ways have you used the hardware efficiently, without using too many gates to implement some infrequent instructions?
10) Your chief competitor just announced a load-store ISA with three operands, two registers (i.e., a 1-bit register specifier), and 32 instructions (5 opcode bits). Tell me why your ISA is better.
11) What do you think will be the bottleneck in your design (i.e., what “resource” (loosely defined) will you run out of most quickly for bigger, more complex programs)?
12(a) What would you have done differently if you had 2 more bits for instructions?
12(b) 2 fewer bits?
13. Can you classify your machine in any of the classical ways (e.g., stack machine, accumulator, register, load-store)? If so, which? If not, give me a name for your class of machine.
14. Give an example of an “assembly language” instruction in your machine, then translate it into machine code.

For 15-17, give assembly instructions. Make sure your assembly format is either very obvious or well described. If you also want to include machine code, the effort will not be wasted, since you will need it later. We will not correct/grade the machine code. State any assumptions you make.

15. Write a program to filter an array in main memory. The filter equation is \( y[n] = 0.5x[n-2] + x[n-1] + 0.5x[n] \). Array variable \( x \) is in positions 0 to 31 of Data Memory, and the output vector \( y \) is to be stored in positions 32 to 63. In order to compute \( y[0] \) and \( y[1] \), assume that \( x[-2] \) and \( x[-1] = 0 \). You must deal if SIGNED numbers, but you can choose any representation that suits you. What is the dynamic instruction count of this program?

16(a), Write a program to determine the integer square root of each byte in an array. The array is stored in locations 0 thru 31, and the results should be stored in positions 32 to 63.
16(b), What is the dynamic instruction count for your program (if it varies with the data, tell me the worst case count)?

17(a), Write a program which sorts an array composed of 16-bit constants. In positions 33 and 34 of main memory you have the value of a number (MSB is in position 33). All values in the array from position 0 to 31 that are smaller than the constant must be moved to positions 64 and up. The original position must be filled with ones (1). 17(b), What is the dynamic instruction count for your program (if it varies with the data, tell me the worst case count)?

**PART 2**

Take the problem of recognizing a sequence of three consecutive ones at the input. Build the FSM and simulate it in the Xilinx package. Answer the following questions:
1) What is the chip you are using to synthesize the hardware?
2) What is the operating frequency you achieve (make sure the answer comes from the timing analyzer tool).
3) Compile the chip to a different device. What is the new frequency?

NOTE: At least one of your compilations (to question 1 or 3) MUST use the Spartan family.