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

Due Friday, October 8.

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 a few simple programs (described on the last 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. You should have a “halt” instruction in the ISA.
- 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 internal storage will also be 8 bits wide.
- the size of main memory.
- addressing modes supported (this applies to both memory instructions and branch instructions). That is, how are addresses constructed/calculated.

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. You should also assume that memory is byte addressable, and that loads and stores read and write exactly 8 bits.

You will write and run three programs. You can assume each starts at location 0, or you can assume that the first starts at 0, and the other two are found in memory after the end of the first program. However, the specification of your branch instructions may depend on which approach you take (where in instruction memory particular programs reside).

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. In particular, you will assume single-ported memory (a maximum of one read or one write per cycle, not both). You will also assume a register file (or whatever internal storage you support) that can only write one register per cycle. But you can, of course, read more than one register per cycle.

To simplify the ISA design process, you need only optimize for the following two 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.

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 two goals should still take highest priority. You will be rewarded, in particular, for doing a good job with goal 1.

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 only.

You will turn in a lab report no more than eight pages long (excluding the program listings). 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 writeup (in this case, your ISA description) and the answers to some questions. The format of the report is given next.

Components of lab 1:

1. Introduction. This should include the name of the architecture, overall philosophy, specific goals strived for and achieved.
2. Instruction formats. Give all formats and an example of each.
3. Operations. Give all instructions supported and their opcodes/formats.
4. Internal operands. How many registers are supported? Is there anything special about the registers?

5. Control flow (branches). What types of branches are supported? How are the target addresses calculated? What is the maximum branch distance supported?

6. Addressing modes. What addressing modes are supported? How are addresses calculated (including instruction addresses for branches)? Give examples.

Additionally, please explicitly answer the following questions.

7. How large is the main memory?

8. In what ways did you optimize for dynamic instruction count?

9. In what ways did you optimize for ease of design?

10. If you optimized for anything else, what and how? (It’s OK if you didn’t)

11. 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.

12. What do you think will be the bottleneck in your design? That is, what don’t you have that you will miss the most if you were to have to write other, longer, programs.

13(a). What would you have done differently if you had 2 more bits for instructions?
13(b). 2 fewer bits?

14. 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.

15. Give an example of an “assembly language” instruction in your machine, then translate it into machine code.

for 16-18, give assembly instructions. Make sure your assembly format is either very obvious or well described, and that the code is well commented. 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. You cannot assume anything about the values in registers or data memory, other than those specifically given, when the program starts.

16. Write a program that takes the square of a number. The number is loaded from location 0, and you will store the result in locations 2 and 3 (high byte in 2, low byte in 3). Assume the value in 0 is unsigned. You will not have a multiply instruction in your ISA (or a square instruction for that matter). Trust me, you don’t want either.
16(b). What is the dynamic instruction count of this program if the value in location 0 is 117 (0x75)?

17. Write a program that counts the number of 1s in an array A at bit positions specified by a mask. The mask is in location 1. A starts at position 64 and is 32 bytes long. The result shall be written in location 96. For example, if the mask is 10011001, then your program will count the number of 1s in bit positions 0, 4, 5, or 7 of each element of the array (e.g., 00001111 in the first element would yield a count of 2).
17(b). What is the dynamic instruction count of this program? If it varies according to the mask, assume it is 11001101. If it varies according to the elements of the array, assume every element has value 01100111.

18. Write a program to find the most common integer found in an array of 30 integers. Assume all values are less than 64. The array of integers starts at location 32. Write the result in memory location 63. In case of a tie, use the lower integer.
18(b). What is the dynamic instruction count of this program?