Lab 1. 8-Bit Instruction Set Architecture.

Due Friday, January 22.

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 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 may 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 for it.
- number of registers, and how many; general-purpose or specialized.
- the size of main memory.
- addressing modes supported.

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 instruction and 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 all the way through, meaning that memory is byte-addressable, and registers and all important data types are also 8 bits.

To simplify the IS design process, you need only optimize for the following two goals:

1. Minimize dynamic instruction count (i.e., the number of instructions needed to execute particular programs).
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 turn in a lab report no more than five pages. 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.

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? Anything special about the registers?

4. What addressing modes are supported? How are addresses calculated? Give an example.

5. How large is the main memory?

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

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

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

9. 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.

10. 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)?

11a. What would you have done differently if you had 2 more bits for instructions?

11b. 4 more bits?

12. 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.

13. Give an example of an "assembly language" instruction in your machine, then translate it into machine code.

for 14-16, 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.

14. Write a program in your assembly to determine whether the total number of ones in an array is evenor odd (for example, the number of 1's in the single byte 10110011 is 5, an odd number). The array starts at location 1 and the length of the array is in location 0 (but not higher than 33). If the total number of ones is odd, write a 1 in location 45, otherwise leave location 45 *unchanged*.

14b. What is the dynamic instruction count of this program? (If it varies with the data, tell me how).

15. Write a program to divide A / B -- A is in memory location 5, B at address 7, the (integer, rounded down) result to be written to address 12, and the remainder to address 13. Assume all unsigned numbers. (By the way, you can't have a divide instruction.)

15b. What is the dynamic instruction count of this program if A = 237 and B = 9?

16. Write a program which finds the second-lowest of 32 signed numbers (memory locations 0 through 31), and writes the result in location 32.

16b. What is the dynamic instruction count for this program? If it depends on the data, give the worst-case count.