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 will come from an iterative process of designing an ISA, then coding the programs, redesigning the ISA, and so on.

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 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. The only exception to this rule is that you can have a single 1-bit condition register (e.g., carry out, or shift out, sign result) that can be written at the same time as an 8-bit register, if you want. You can, of course, read more than one register per cycle. Please restrict register file size to no more than 16 registers. Also, manual loop unrolling is not allowed.

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 two explicit operands (one source register same as destination), four registers (i.e., a 2-bit register specifier), and 16 instructions (4 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. That means, for example, that if you need zeroes in registers or memory, you need to put them there.

16. Write a program that finds the median value in an array of 31 integers. Assume the array starts at address 64, and write the result in address 0. Also assume that all integers are non-negative, and less than 32.

16(b). What is the dynamic instruction count of this program if the median value is 19 (and the numbers are randomly distributed)? If the count depends heavily on the numbers, give me the worst case result.

17. Write a program that calculates the square of an unsigned number. The 8-bit input is found in memory location 1, and the 16-bit answer is to be written in locations 2 (high byte) and 3 (low byte). You will NOT have a square or multiply instruction in your ISA.

17(b). What is the dynamic instruction count of this program if the input is 189?

18. Write a program to find the first “widest” integer in an array of 32 integers. The width of an integer is defined to be the distance between the least significant and the most significant “1” in the binary representation. For example, the width of 00110100 is four, the width of 10000000 is one, the width of 00000000 is zero, and 11010001 is eight. Write the width of the widest integer in memory location 4, and the first number to have that width in location 5. Thus, if the four values above were the inputs, you would write eight in location 4, and 11010001 in location 5. The array begins at location 32.

18(b). What is the dynamic instruction count of this program?