

**Fall 2003**  
**CSE 141L Projects in Computer Architecture**  
**Lab 1: Construct 8-bit Instruction Set Architecture**  
*(Last Updated 10/13/03)*

**Reports are due at the beginning of class on Friday, October 17.**

In this lab, you will design the Instruction Set Architecture (ISA) for an 8-bit “statistics processor.” The architecture of the processor will be optimized for coding and running three statistics related tasks specified below. You will design an instruction set simulator and hardware for this processor in subsequent labs.

**What you will turn in for this Lab**

- Lab report covering all the issues outlined below.
- Assembly code and hand-assembled machine code for three programs in your ISA.
- Instruction and data files for the three programs.

**Requirements**

The following are some of constraints and requirements your ISA must follow:

- Instruction format should be fixed-length 8-bit instructions.
- Instruction memory (I-Mem) and data memory (D-Mem) are separate.
- The memory operations must be explicit load and store instructions in your ISA.
- Memory is byte addressable. Loads/stores read or write exactly 8 bits.
- This is an 8-bit machine for every aspect. All registers and data types are maximum 8 bits.
- Single-ported instruction and data memory (a maximum of one read or one write per cycle, not both) must be used.
- A register file with only one write port. May have multiple read ports.
- When the processor is reset, execution starts at instruction memory location 0.
- A “halt” instruction that would halt the execution must be supported.

**General Guidelines**

8-bit fixed instruction format will put extreme limit on the number of bits available to encode various instruction components. For best design, you should go through an iterative process of designing ISA then coding the assembly programs and tuning the ISA based on your observations.

Your instruction-set specification should be detailed enough for someone else to write assembly programs for your processor or implement it. You should nicely arrange the ISA specification using tables and figures as you see fit. You will be turning in this tabular description of your ISA with every assignment in this class. ISA description should include the following.

- How many instruction formats are supported and what they are.
- Instructions supported and their opcodes. Include a brief description for each instruction, including the operation it performs and registers affected.
- Number of general purpose registers and any other internal state registers.
- Size of the instruction and data memory supported
- Addressing modes supported, including how the addresses are computed.

You will write and run three programs. You can assume each program starts at location 0 in I-Mem and that the three programs will be run one at a time.

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, smaller number of gates, etc.), but if you do so we will expect you to discuss that optimization intelligently, and above two goals should still take highest priority. The first goal will call for some specialized instructions to reduce the instruction count.

Keep in mind that in Lab 3 and 4 you will implement your architecture in logic and test it. Complex instructions (e.g. a general divide instruction) may be quite complex to implement and test in such a short amount of time. It would be a good idea to keep the design as simple as possible. Review ISA design principles covered in CSE141 theory course.

### **Lab Report**

This lab report will consist of description of your ISA, machine code and assembly code for the three programs. While describing your ISA, ensure that you answer the following items/questions. Also, keep in mind that the person grading your report has much less experience with your ISA than you do. **It is your responsibility to make everything clear.**

1. Describe your design goals.
2. Classify your processor in any one of the classical ways (e.g. stack machine, accumulator, register, load-store, etc.) If your architecture is different from classical ways, propose a name for you architecture.
3. What instruction formats are supported? Give all opcodes.
4. How many registers are supported? What are the special features of these registers?
5. What addressing modes are supported? How are the addresses calculated?
6. What types of branch and jump instructions are supported? How are the target addresses calculated? What maximum branch distance is supported?
7. Give a brief description of each instruction and its side effects.
8. How large are instruction and data memories?
9. In what ways did you optimize for dynamic instruction count? Are there any special instructions for each one of the three programs that significantly reduced the instruction count?
10. How did you optimize for ease of implementation?
11. What would you have done differently if you had 4 more bits for instructions?
12. What are the bottlenecks in your design, i.e. what resources will you run out of the most quickly for bigger, more complex programs?
13. For each one of the three programs specify static instruction count and dynamic instruction count. What is the difference between the two?

## Code for Optimizing ISA and Architecture

For the three programs described below, turn in assembly listing and hex machine code for the programs in your new ISA. Make sure your assembly format is well described and assembly code is heavily commented (each line should contain a comment). State any assumptions you make. Recommended assembly code file format is (optional fields are in []):

```
machine_code PC [label:] instruction_mnemonic [operand 1, 2, ...] comment
```

So in the program text file instruction lines will look like following.

```
0xf2          07          Add R0 R2    // Adds contents of R0 (mean) to R2 (num2)
```

The assembly code will not be graded. However, note that subsequent assignments will be graded based on the correct functionality of your programs on instruction set simulator and the logic implementation. All the incoming parameters for your program are specified in terms of the contents of the data memory. You cannot assume anything about the values in registers or other locations of the data memory (D-Mem) other than what is specified, when the program starts.

## Program Descriptions

While implementing following programs, you should not change the incoming parameters, i.e. the value of the number of array elements and the starting address(es) of the array(s). For temporary storage, you may use a small number of data memory locations other than those specified below. Specifically, D-Mem locations 2 – 11 may be used for temporary storage. However, keep in mind that the result for each program is expected to be in D-Mem location 2 when the processor halts.

We have supplied example input data below. TAs may test your design with other data consistent with the constraints outlined below.

1. D-Mem location 0 contains number of elements of an array of unsigned bytes. D-Mem location 1 contains the starting address of array a[]. The array values are guaranteed to be between 0 and 7. Find the mode, i.e. most frequent value and save it in D-Mem location 2. The data patterns given to you will have only one mode. The contents of data memory starting from location 0 are: 0x09, 0x0C, 0xDE, 0xAD, 0xF0, 0x0D, 0xDE, 0xAD, 0xF0, 0x0D, 0xDE, 0xAD, 0x04, 0x07, 0x04, 0x05, 0x07, 0x04, 0x05, 0x06, 0x04
2. D-Mem location 0 contains number of elements of an array a[] of unsigned bytes and location 1 contains the start address of the array. Find median value and save it in D-Mem location 2. Note that if there are even number of elements, median is average of the two values in the middle. You should implement “early out” bubble sort algorithm to sort the array in order to find the median. The contents of the data memory starting from location 0 are: 0x09, 0x0D, 0xDD, 0xDD, 0xDD, 0xDD, 0xDD, 0xDD, 0xDD, 0xDD, 0xDD, 0xDD, 0xDD, 0x78, 0xFF, 0x65, 0x55, 0xEA, 0x23, 0x9D, 0x09, 0x44.
3. D-Mem location 0 contains the address of array a[] of unsigned bytes. Memory location 1 contains 0x10 which specifies number of elements of array a[]. Find the mean (i.e. average)

for the 16 elements of array `a[]` and save it in location 2. If the mean value has a fractional part, you should round it to the nearest integer. Thus, if the fractional part is 0.5 or more, increment the mean value. The initial contents of the data memory starting from location 0 are: 0x0D, 0x10, 0xDE, 0xAD, 0xF0, 0x0D, 0xDD, 0xDD, 0xDD, 0xDD, 0xDD, 0xDD, 0xDD, 0xFF, 0x12, 0x80, 0x27, 0x34, 0x87, 0xCE, 0x46, 0x01, 0xDD, 0x5D, 0xD8, 0x76, 0x49, 0xBF, 0x06.

Generate a text file with machine code for each instruction on one line. This file will be loaded into the instruction memory of your processor. Also, generate a text file with hex code for each data memory location starting from 0 including all the valid data. Include both of these files in your report. They will be used in Labs 2 and 4.