

Least Significant Zero Locator

Introduction: For this machine problem you will be writing a program in LC-3 machine code whose purpose is to find the location of the least significant zero in a number. You will create a code file (mp1.bin) that implements the functionality described below.

Program: Given a 16-bit number stored in memory at address x4000, your job is to find the position of the least significant zero. Your program will need to load the number from memory, and analyze it to detect the location of the least significant zero. You must use a loop to perform this analysis. Solutions not using a loop will receive a zero for the MP. As illustrated below, the rightmost bit is the LSB and its location is 0. The leftmost bit is the MSB and its location is 15.

	MSB														LSB	
Location	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	1	1	1	0	1	1	0	1	1	0	1	1	1	0

After locating the least significant zero, your program needs to store the value of its position at memory location x5000 and then halt.

Examples:

Original Number	Result
1111 1111 1111 1110	0
1001 1101 0010 1111	4
0111 1111 1111 1111	15

Hint 1: Figure out how to detect the value of a single bit.

Hint 2: Shifting a number is an important concept.

You are strongly encouraged to create a flowchart of your program before writing any actual code.

Specifics:

- Your program must be named “mp1.bin”
- Your program must use less than thirty (30) lines of code and must use a loop
- Your code must be written in LC-3 machine language. Use spaces to separate instruction fields
- Your code must begin at memory location x3000
- Your program must expect the number that it will analyze to be placed at memory location x4000.
- You may assume that the number will contain at least one zero
- The rightmost bit (the LSB) is at location 0, the leftmost bit (the MSB) is at location 15
- Your program must store the value of the location of the least significant zero at memory location x5000
- Your code may only modify the memory location x5000; no other memory locations may be modified
- Your code must be well-commented and you must provide an introductory paragraph and a register table at the top of your file (see Style Requirements)

Testing: You should test your program thoroughly before handing in your final version. Just because it runs does not mean it functions correctly. Remember, when testing your program, you need to set the relevant memory contents appropriately in the simulator. When we grade your project, we will initialize the memory for you. You should use several test cases and hand-check the results. We will grade your program by observing its behavior and the final value at location x5000 using several different input

ECE190: Introduction to Computing Systems Fall 2009
Program Due: 10 p.m., Wednesday, September 23

values at location x4000. *Hint: use the simulator's disassembly (.list.) capability to double-check your encodings before running your program.*

Style Requirements: MP1 must be written in LC-3 machine language for any credit to be received (i.e. each instruction is a 16-bit binary word). Your code should contain less than thirty instructions and you must use a loop in your solution to receive credit. Your code must be well-commented (every line for this MP because it is in binary). Comments begin with a semicolon. They should explain why something is done instead of restating what the ASM/RTL already says. In general, you should use the commenting style provided in the Student Manual (see Page 35 for an example). For readability reasons, you should not have more than 80 characters on a single line. If you are writing a long comment and exceed 80 characters, continue your comment on the next line. In addition, you must use spaces to separate instruction fields in your binary encoding (e.g. 0010 001 000000100, not 0010001000000100). This will improve the readability of your code. Include a paragraph at the top of your code explaining the purpose of your program and your approach. Below that paragraph, create a table explaining the role of each register used by your program. Be sure to make the paragraph and the table a comment using semicolons.

Tools: You will use the “lc3convert” command to convert your machine code to an “obj” file, and the LC-3 simulator in order to execute and test the program that you write for this MP. You should use a text editor on a Linux machine (vi, emacs, pico, etc.) in order to code your program. Your code must work in the command line LC-3 simulator on the Linux EWS machines in Everitt lab to receive credit. ***Important: The LC-3 tools have a bug that prevents them from loading the last line of machine program correctly. To bypass this, create a blank line at the end of your mp1.bin file.***

Hand In: Turning in your program is done electronically via the ECE 190 handin script. You must name your file `mp1.bin`; we will NOT grade files with any other name. To hand in the code, enter the ECE 190 work area by typing `ece190` from a Linux EWS machine. Next change to the directory containing your code and type: “handin --MP 1 mp1.bin”. You may hand in as many times as you like, but only the last submission will be graded.

Grading:

Functionality (55%)

45% Program properly detects the location of the least significant zero for different test cases

5% Number read from and result stored to memory correctly

5% Program Halts

Style (15%)

5% Program does not store intermediate results in memory

10% Program contains no more than thirty (30) lines of code

Comments, clarity, and write-up (30%)

10% Introductory paragraph explaining program's purpose and approach used

10% Code includes table of registers that clearly explains their meaning and contents as used by code

10% Code is clear, well-commented, and formatting follows the style guidelines

Notes: Be sure to consult the online FAQ or the Student manual if you have any questions. It has information on LC-3 simulator tools, UNIX help, SSH/sFTP programs, handin, and many other things. Check the class web board frequently for clarifications on the project.