Introduction to Computer Science, IDC Herzliya, 2016

Low-Level Programming

This is a self-study exercise. You don't have to submit a solution. For your own benefit though, you must complete the exercise. Understanding low-level programming is one of the objectives of the course, and the only way to satisfy this objective is write and debug low-level programs.

Guidelines

- Unless instructed otherwise, write Vic programs using the symbolic Vic language (not the numeric 3-digit command language).
- Translate each program using the Vic Assembler. Next, load and execute the resulting code in the Vic simulator.
- The Vic Assembler is part of the Vic Simulator, available in www1.idc.ac.il/vic. To start the Vic Assembler, click the "Vic Assembler" link at the bottom right of the simulator window.
- When writing each program, assume that the program's input follows the assumptions described in the program's description. Assume that cells 98 and 99 contain the numbers 0 and 1, respectively. You may not assume that the memory contains any other numbers.

The Programs

1. Write a Vic program (`readWritePos`) that reads a series of numbers that ends with a 0. For each number in the series, if the number is greater than 0, the program writes the number. For example, When applied to the input -3,2,5,-1,7,-2,8,0, the program produces the output 2,5,7,8.

2. Write a Vic program (`printIndex`) that reads a non-zero number (say x), followed by a series of numbers that ends with a zero. If x appears in the series, the program writes its location in the series; otherwise, the program writes -1. For example, if the input is 4,2,1,7,4,8,0, the program writes 3 (the first location is considerd 0). If the input is 5,3,1,-3,15,7,0, the program writes -1.

3. Write a program (`odd`) that reads a single positive number from the input. If the number is odd, the program prints 1; if the number is even, the program prints 0. For example, if the number is 5, the program prints 1; if the number is 8, the program prints 0.

4. Write a program (`multiply`) that reads two non-negative numbers, and prints the product. For example, if the input is 3,12, the program prints 36.

5. Write a program (`printMem`) that reads two numbers: the constant 300, followed by a number (say x) between 0 and 99. The program writes the value of the memory register whose address is x. For example, if x = 90, the program writes the value of M[90], i.e. the value of the memory cell whose address is 90. Tips: (i) this is a tricky program; (ii) the program must be written in Vic's native machine language: each command must be written as a 3-digit number; (iii) the first input, the constanst 300, is needed in order to write this program.