Homework 6: Vic Programming

Guidelines

Unless instructed otherwise, write Vic programs using the symbolic (assembly) Vic language, not the numeric 3-digit command language. Write the programs using a simple text editor.

Translate each program into numeric code using the Vic Assembler. Next, load and execute the numeric code in the Vic simulator. The Vic simulator and assembler are 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 always contain the numbers 0 and 1, respectively. You may not assume that the memory contains any other values.

Relevant resources: lecture 7-1 and code..

Programs

1. Write a Vic program (`readWritePos.asm`) 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 outputs 2, 5, 7, 8.

2. Write a Vic program (`find.asm`) that reads a nonzero number (say *x*), followed by a sequence of one or more numbers that ends with a zero. If *x* appears in the series (once or more), the program writes its first location in the series; otherwise, the program writes -1. For example, if the input is 7, 2, 1, 7, 2, 5, 0, the program outputs 3 (in this program, the first element in the sequence is considered 1). If the input is 5, 3, 1, 3, 15, 7, 0, the program outputs -1.

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

4. Write a program (`multiply.asm`) that reads two nonnegative numbers, and writes their product. For example, if the input is 3, 12, the program outputs 36.

5. Write a program (`getMem.vic`) 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, *not in assembly*: each command must be written as a 3-digit number; (iii) the first input, the constant 300, is needed in order to write this program.

Submission: This is a self-study exercise. You don't have to submit it, and it won't be graded. We ask you to do this exercise because it's important to experience low-level programming hands-on, as part of your CS education. For this reason, in the final exam of this course you may be asked to write and / or explain a Vic program, similar to those given in this homework.