**Md Towhedur Rahman-u2151946**

**Practical (Week 3)**

**Programming with the Visible Virtual Machine**

The purpose of this practical is to help you develop your understanding of the CPU and memory components of a computer and, in particular, instruction sets and the fetch execute cycle. To do so, you will write a number of programs using a simulator called the Visible Virtual Machine (VVM). The Visible Virtual Machine (VVM) software, based on the Little Man Computer model, was designed and written by Dr Stu Westin of the University of Rhode Island. Although it is a simulation, the operation of the VVM (and the Little Man Computer upon which it is based) does not differ fundamentally from that of a real computer.

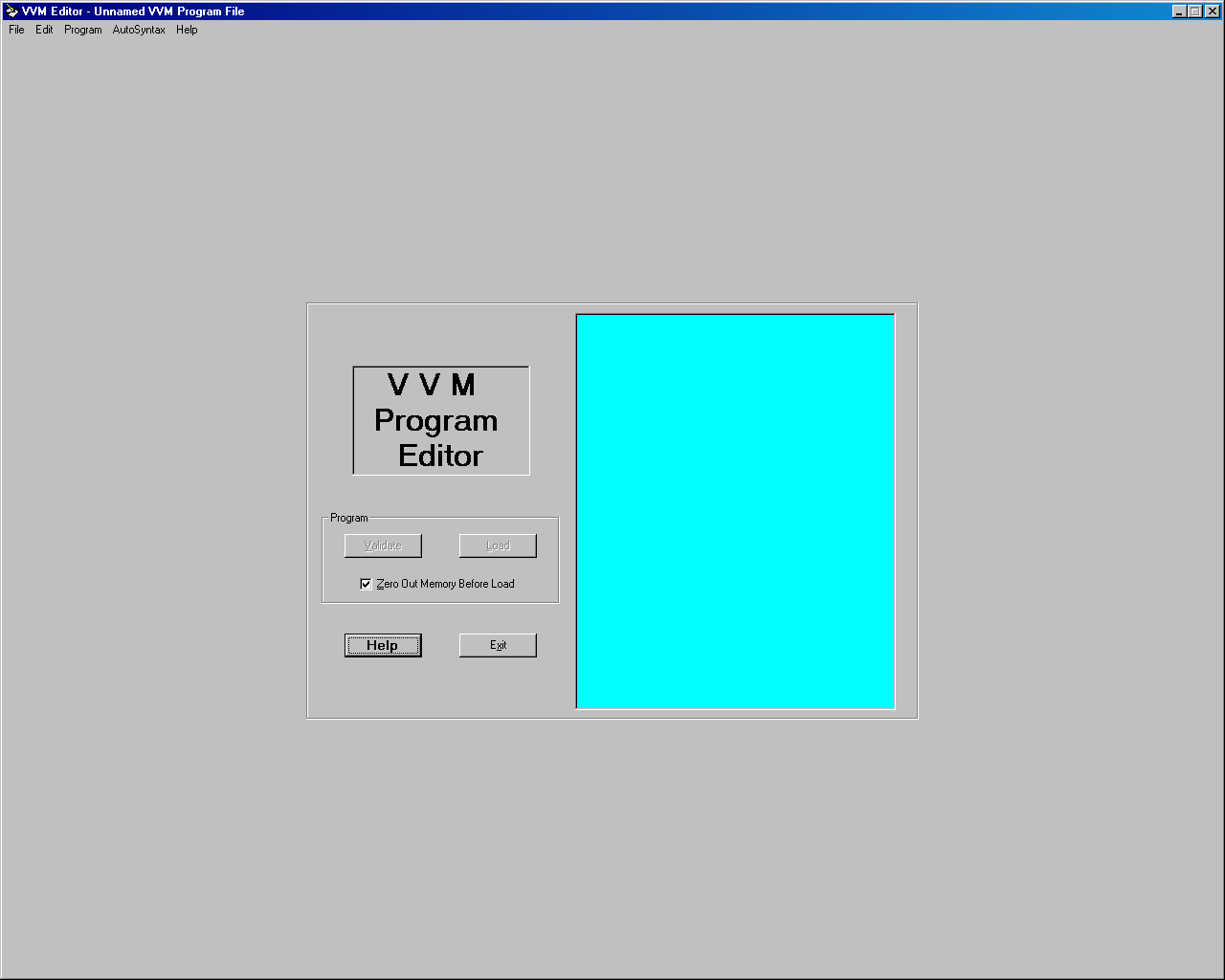
Detailed information about the VVM software can be found online at

<http://visible-virtual-machine.software.informer.com/download/>. You can also download a free copy of the software from this site.

**Introduction to Tasks**

The VVM software has already been installed on the computers in your laboratory. To start the software from within Windows 10, go to

Start → Programs → Visible Virtual Machine 5.0.5 → Visible Virtual Machine



Carefully read through the notes available from the Help option in the VVM, paying particular attention to the VVM instruction set i.e. LDA, ADD, SUB, STO, IN, OUT, HLT etc.

Note also that the instructions you write for the VVM do not explicitly include the addresses of the mailboxes into which they will be loaded. Instead, your program is simply loaded into the mailboxes in sequence starting at 00. This is not very useful if you want to store data rather than instructions, because it is likely that you will want the data to be stored in high-numbered mailboxes well away from the program instructions themselves.

To force the VVM to use a mailbox other than the next one in sequence, you use an asterisk followed by the mailbox address, for example:

\*98

The VVM then continues to use mailboxes in sequence, starting from the address you have specified.

You can add comments to your programs using a double slash at the front of the line – the VVM will ignore these lines rather than attempt to interpret them as instructions, for example:

// This is a comment

You can add further comments to your programs on each instruction line – everything after the instruction itself is ignored by the VVM. For example:

LDA 96 Put contents of box 96 in calculator

**Tasks**

You are now ready to write Little Man Computer programs using the Visible Virtual Machine.

You should save each of your programs as a file with a .vvm extension. Remember to add comments to your programs.

Use the 'Step' facility rather than 'Run' so that you can follow the progress of your programs one step at a time, paying close attention to the changes in the PC, IR and accumulator at each step. The 'Trace View' option creates a table tracking these values.

You may find it useful to run through your program on paper before running it in the VVM. You can then compare the way you think your program will behave with the table generated in the VVM's Trace View to check your understanding of Little Man Computer programs.

1. Using the VVM, write the following LMC program (adapted from fig. 6.2, p.184 of Englander) which inputs two numbers, adds them together and outputs the result.

|  |  |  |  |
| --- | --- | --- | --- |
| **Mailbox** | **Code** | **Instruction** | **Description** |
| 00 | 901 | IN | Input first number (***xxx***, say) into calculator |
| 01 | 399 | STO 99 | Store number from calculator (***xxx***) in mailbox 99 |
| 02 | 901 | IN | Input second number (***yyy***, say) into calculator |
| 03 | 199 | ADD 99 | Add value in mailbox 99 (***yyy***) to number from calculator (***xxx***). Calculator now holds the sum. |
| 04 | 902 | OUT | Output number from calculator (***xxx*** + ***yyy***) |
| 05 | 000 | HLT | Stop. |

Choose some example inputs and work through each of the stages to check that your VVM program does indeed add two numbers together. Be aware that the VVM cannot handle large numbers i.e. greater than 999.

Now compare your VVM program with the following :

// Program to input two numbers, add

// them together, and output the result.

//

in Input first number

sto 99 Store in box 99

in Input second number

add 99 Add value from box 99

out Output result

hlt End program

Your VVM program may not be exactly the same as the above, but should be similar. One likely difference is that you may have used other locations for data storage – the program above used the highest-valued box available (i.e. 99). A more important difference may be in the length of your program. If your version has more instructions that the answer above, check through your program and see if you can improve its efficiency by using fewer instructions to complete the task.

1. Write a VVM program which inputs two numbers, subtracts the second number from the first and outputs the result.

|  |
| --- |
| In  Sto 98  In  Sto 99  Lda 98  Sub 99  out  HLT |

1. Write a VVM program which adds the numbers 157 and 13 together and outputs the result.

|  |
| --- |
|  |

1. Write VVM programs to do the following:
   1. Input a number, add 100 to it and output the result. The number 100 should be placed in a memory location prior to running the program.

|  |
| --- |
|  |

* 1. Input a number, double it, and output the result.

|  |
| --- |
|  |

* 1. Input a number, double it, subtract 1, and output the result.

|  |
| --- |
|  |

* 1. Input three numbers and output the sum.

|  |
| --- |
|  |

* 1. Input three numbers, add the first two together, subtract the third from the sum, and output the result.

|  |
| --- |
|  |