

Fundamentals of Computing

(4CS015)

Report Writing

Workshop Portfolio Task - 3

# 

Student ID : 2357550

Student Name : Shubham Neupane

College ID : NP03CS4S230067

Group : L4CG1

Lecturer : Mr. Uttam Acharya

Tutor : Mr. Rukesh Shrestha

Word Count :

Submitted on : <dd-mm-yy>

**Table of Contents**

[Workshop tasks: 1](#_Toc133436731)

[Von Neumann Simulator. This program simulates a very simple computer with the von Neumann architecture. 1](#_Toc133436732)

[Qno. A 1](#_Toc133436733)

[Q no. B 2](#_Toc133436734)

[Qno. C 4](#_Toc133436735)

[Qno. D 6](#_Toc133436736)

[Qno. E 8](#_Toc133436737)

[Qno. F 10](#_Toc133436738)

[Qno. G 12](#_Toc133436739)

[Qno. H 14](#_Toc133436740)

[Qno. I 16](#_Toc133436741)

[Conclusion 18](#_Toc133436742)

**Table of Figures**

[Figure 1 VonNeumann Interface (Provided in Question) 1](#_Toc133436743)

[Figure 2 Qno. B Implementation 3](#_Toc133436744)

[Figure 3 Displaying Entered Number in VonNeumann.exe 5](#_Toc133436745)

[Figure 4 Qno. D Implementation in VonNeumann.exe 7](#_Toc133436746)

[Figure 5 Program to add 3 numbers in VonNeumann.exe 9](#_Toc133436747)

[Figure 6 Program to subtract in VonNeumann.exe 11](#_Toc133436748)

[Figure 7 Implementation of Qno. G 13](#_Toc133436749)

[Figure 8 A program to let add multiple numbers as per input in VonNeumann.exe 15](#_Toc133436750)

[Figure 9 Program to Multiply 2 numbers in VonNeumann.exe 17](#_Toc133436751)

# Workshop tasks:

## Von Neumann Simulator. This program simulates a very simple computer with the von Neumann architecture.

### Qno. A

Download the von Neumann Simulator (VonNeumann.exe) program from WOLF in the Week 5 folder. Save it in your Documents folder and run it. You will see a window similar to this:

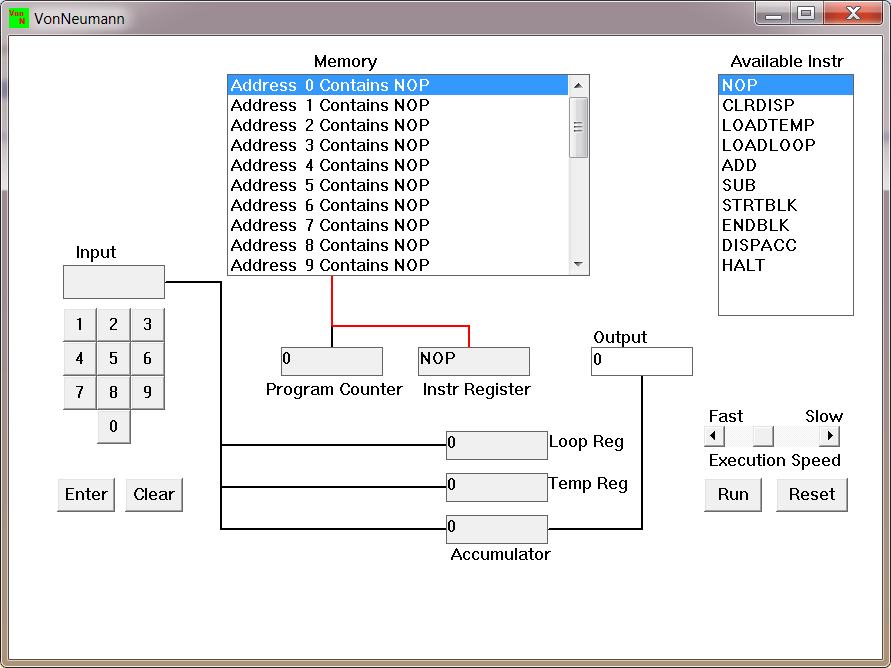


Figure 1 VonNeumann Interface (Provided in Question)

The simulator has a small program memory area which is available for programming. To enter your program instructions simply click on the “Available” instruction on the list on the right and then click on the “Memory” location you wish to put it in.

This simulator understands only the following ten instructions:

|  |  |
| --- | --- |
| NOP | No Operation, i.e. do nothing. |
| LOADTEMP | Get a number from the keypad, completed by the Enter key, into the Temporary Register. |
| LOADLOOP | Get a number from the keypad, completed by the Enter key, into the Loop Register. |
| CLRDISP | Clear the Display. |
| ADD | Add the Temporary Register to the Accumulator |
| SUB | Subtract the Temporary Register from the Accumulator |
| DISPACC | Display the contents of the Accumulator |
| STRTBLK | Start of Loop Block |
| ENDBLK | End of Loop Block |
| HALT | Halt. Stop Program |

### Q no. B

Load the following program into the memory:  
LOADTEMP  
ADD  
DISPACC   
HALT  
  
To do this, first click on the “LOADTEMP” in the list of instructions on the right of simulator window. Then click on Memory location with “Address 0 Contains NOP”. This will then change into “Address 0 Contains LOADTEMP”. Repeat the process with “Address 1” and so on until the whole program is loaded.

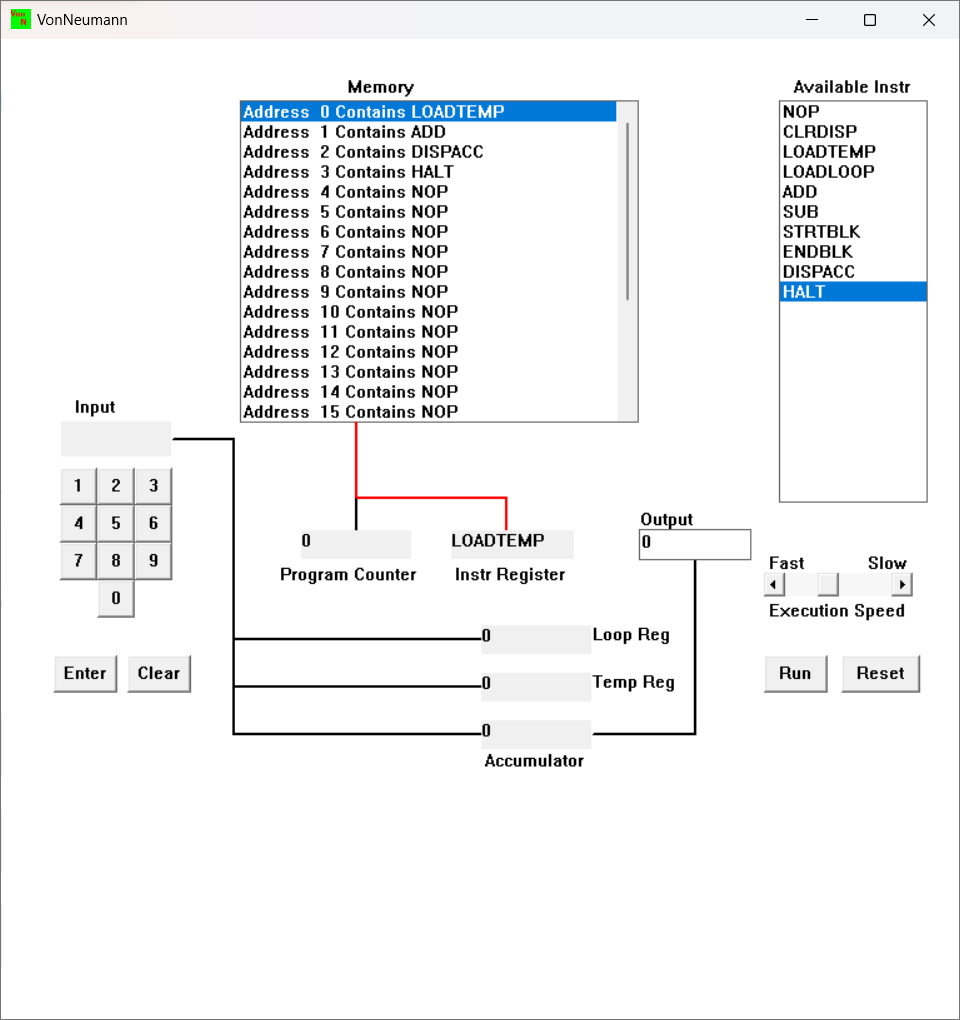


Figure 2 Qno. B Implementation

-: In this figure, the program is loaded in memory according to the question’s instructions. At first, Address 0 is set at LOADTEMP which is for loading input in Tem reg. Then ADD instruction is set to address 2 which adds temp reg value to accumulator value. Then DISPACC instruction displays output. And HALT instruction in address line 3 in memory terminates the program.

### Qno. C

Run the program by clicking on the “Run” button. The simulator would highlight the Address 0 location and then pause. It is executing the instruction “LOADTEMP” which requires you to input a number into the keypad.   
  
Click 2 or 3 numbers on the keypad and then click the “Enter” button. The simulator will then resume running the program and execute the instruction “ADD”. This adds the number that you just entered, to the zero in the accumulator.   
  
The next instruction is “DISPACC” which stands for “Display Accumulator”, and it does exactly that. After than the simulator stops running the program when it executes the instruction “HALT”.

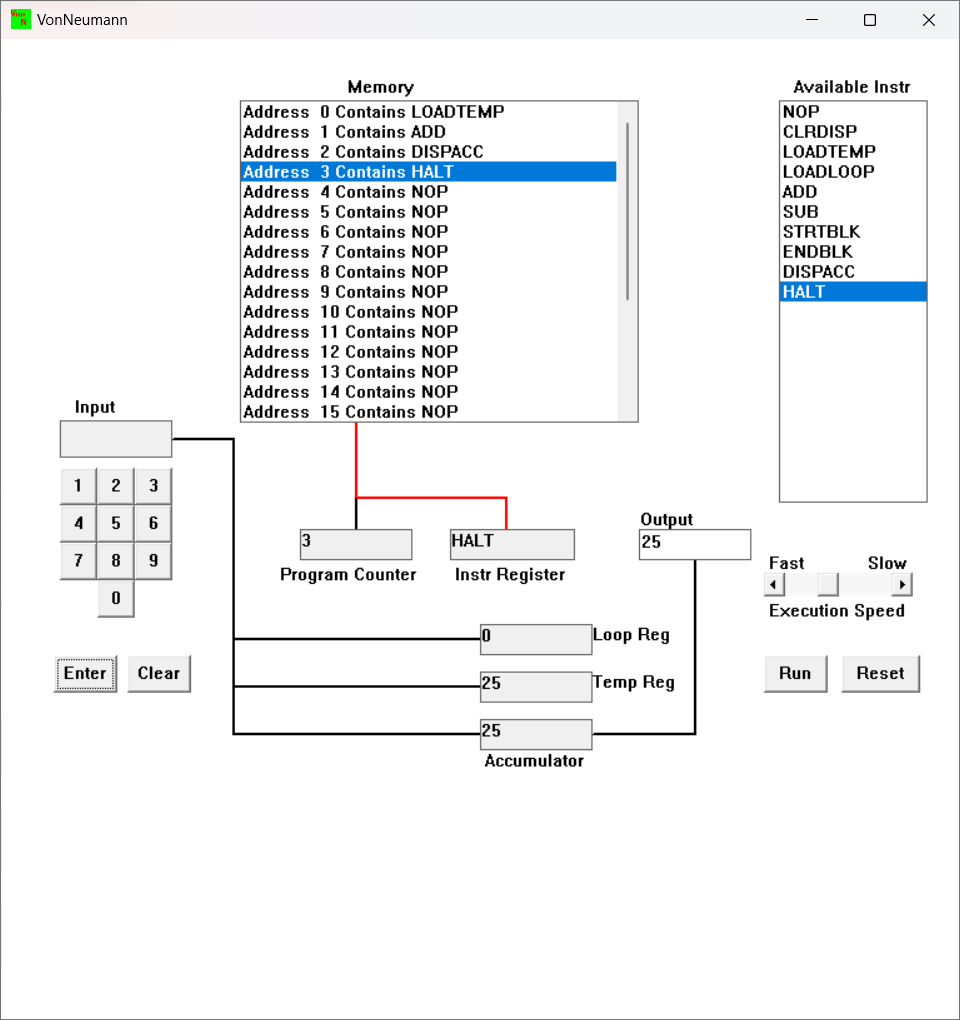


Figure 3 Displaying Entered Number in VonNeumann.exe

-: In this figure of VonNeumann program, the set of instruction in memory are executed. The LOADTEMP instruction asks for imput from user and for this case the input was 25. And ADD added the value to the accumulator value which was initially 0. And the new sum 25 was stored in the accumulator which is then displayed by DISPACC instruction.

### Qno. D

Load the following program into the simulator:  
LOADTEMP  
ADD  
LOADTEMP  
ADD  
DISPACC  
HALT  
  
What do you think it does? Write your answer below (10 marks)

-: When the program is loaded into the simulator, It checks address 0 at memory which is LOADTEMP and ask for user input, then address 1 is at ADD instruction which adds the temp register value to accumulator value and stores it in accumulator as new value. Address line 2 is Set at LOADTEMP which asks for user input and adds the value to the accumulator value stored and the value is stored in the accumulator. The DISPACC instruction in memory address 4 displays the output and HALT terminates the program which is in address line 5.

To conclude the program provides the sum of two numbers entered.

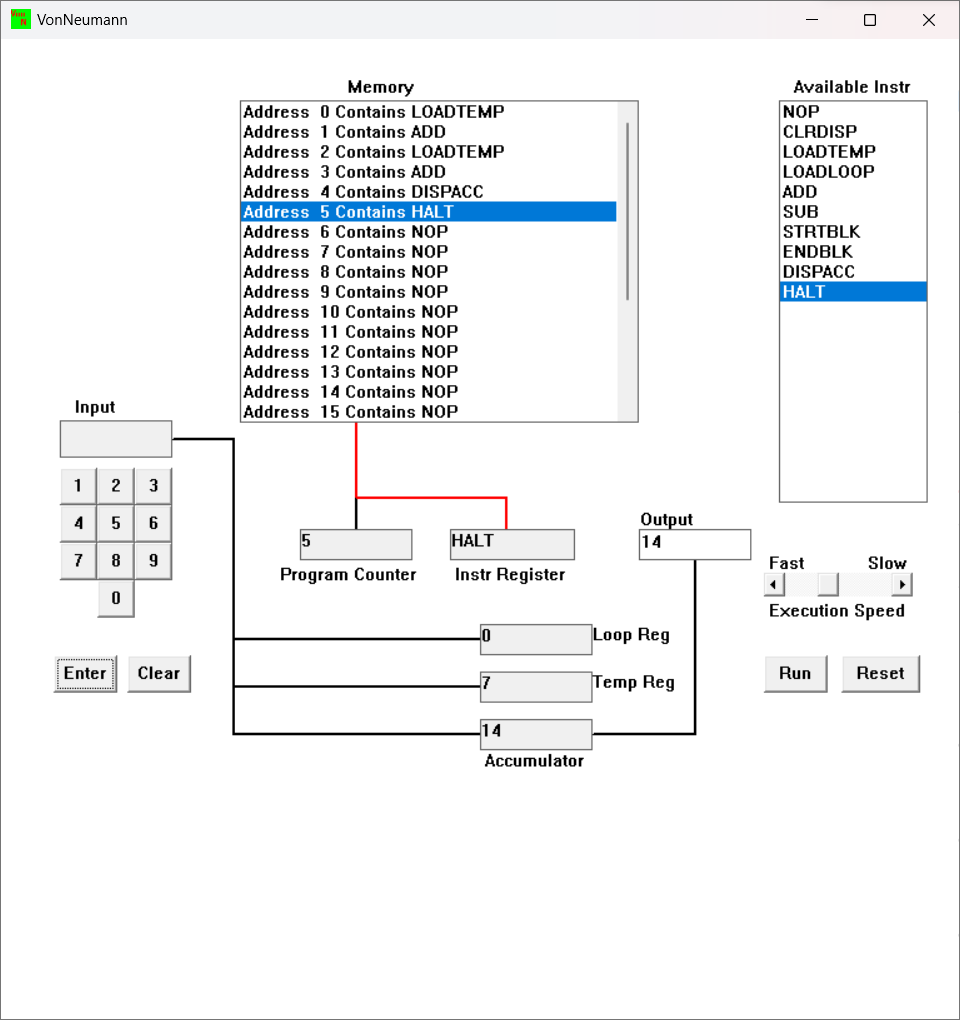


Figure 4 Qno. D Implementation in VonNeumann.exe

-: In this figure, Sum of two numbers is generated. At first the program counter is at 0 and asks for input and the input was given 7, it was stored in temporary register and it was added to the value in accumulator (initially 0) and stored in accumulator. The other input was asked and 7 was provided as another input. Which was stored in temp register and was added to the previous value in accumulator and stored as new value. The DISPACC instruction displayed the output stored in accumulator and HALT terminated the program.

### Qno. E

Write a program to add 3 numbers together. List your program below (10 marks)

-: The program to add 3 numbers together is as :

- LOADTEMP

- ADD

- LOADTEMP

- ADD

- LOADTEMP

- ADD

- DISPACC

- HALT

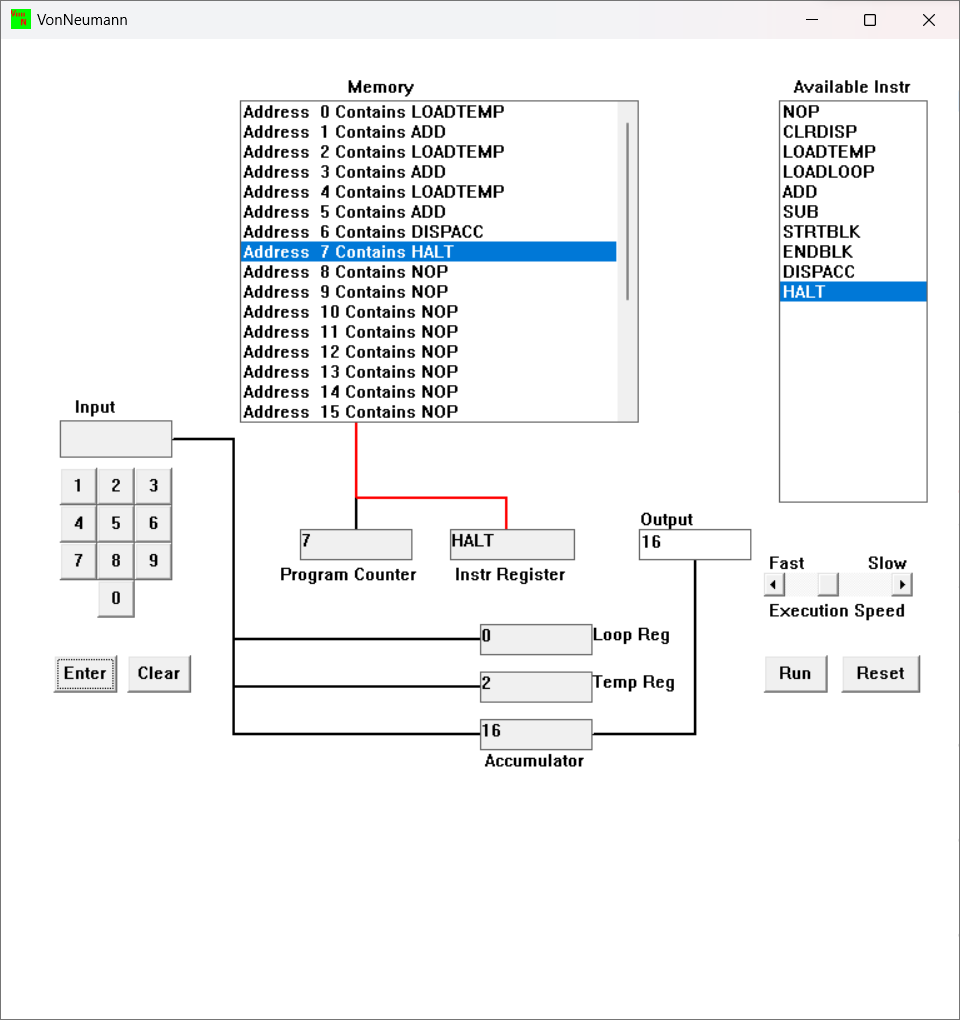


Figure 5 Program to add 3 numbers in VonNeumann.exe

In this figure, the program run serially from address line 0 to address line 7, it starts from address 0 which is set for LOADTEMP, it asks for user input to temp register and adds it to accumulator value and stores it as new value to the accumulator which is 7(input), again the process repeats as program counter goes to LOADTEMP (7) and then ADD again LOADTEMP (2)and then ADD. The DISPACC instruction displays output and HALT instruction terminates the program.

### Qno. F

Write a program to subtract a number from another. List your program below (10 marks)  
  
-: The program to subtract a number is as follows:

* LOADTEMP
* ADD
* LOADTEMP
* SUB
* DISPACC
* HALT

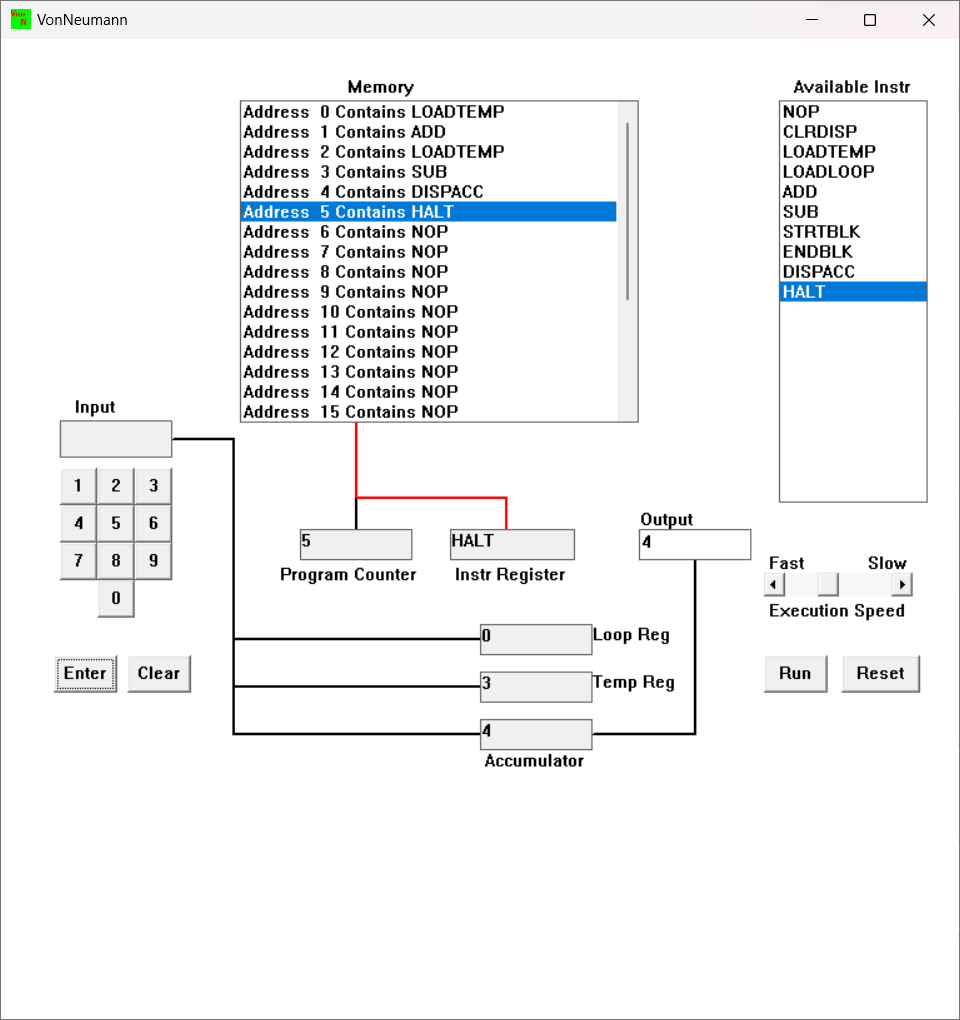


Figure 6 Program to subtract in VonNeumann.exe

-: In this figure, initially program counter is at 0 so, it reads LOADTEMP and asks for input which was given 7(for this figure implementation), which is stored in temp register and ADD instruction passes the value to accumulator to be added and stored as new value. Then LOADTEMP asks for another value (3), and SUB instruction subtract the value in temp register (3) to the value in acculumator (7) and new value is stored in the accumulator. DISPACC displays the output(4) and HALT terminates the program.

### Qno. G

Load the following program into the simulator:  
LOADTEMP  
ADD  
LOADLOOP  
STRTBLK  
ADD  
DISPACC  
ENDBLK  
HALT  
  
Run it and when it reach the LOADTEMP instruction, enter 5 on the keypad and click the “Enter” button. When it reaches the LOADLOOP instruction, enter 6. What do you think the program does? Write your answer below in the form of an equation (10 marks)

-: The equation for the program is: (5 + (5 \* 6)) i.e (LOADTEMP + (LOADTEMP \* LOADLOOP)

This program returns multiplication for 5 and 7. This program takes 5 as input stores it at accumulator and it iterate the ADD instruction process until the LOADLOOP value (6) and stores it in accumulator. And 5 was already stored in accumulator before Iteration so the final ouput comes as the multiplication for 5 and 7. As 5 is being added 7 times.

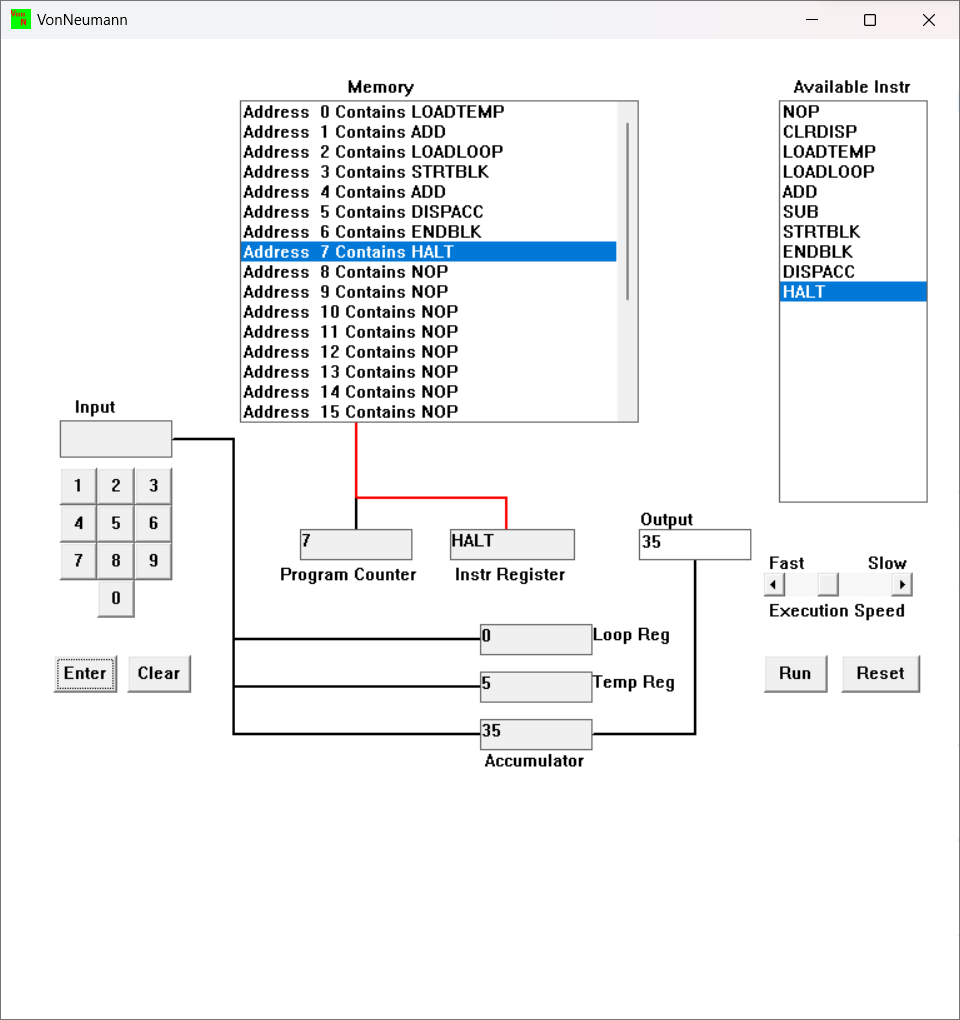


Figure 7 Implementation of Qno. G

-: In this program, the program counter is at 0 which is set as LOADTEMP. It asks user for input to be loaded at temporary register which is 5. In every step program counter increments. Then ADD instruction adds the temp register value to initial value of accumulator and stores as new value. The LOADLOOP instruction asks for the total number of iterations that is to be done for certain block of code inside loop block, as per question 6, STRTBLK instruction starts the loop block (which will repeat 6 times). Inside loop block ADD instruction adds temporary register value to accumulator value and stores it as new value . DISPACC instruction displays the accumulator value. The process repeats 6 times and ENDBLK instruction ends the loop and program counter increases and HALT instruction terminates the program.

### Qno. H

Write a program that will let you add 5, or 10 or 20 numbers together. List your program below and explain how it works (25 marks)

-: The program to let user add 5 or 10 or 20 numbers:

* LOADLOOP
* STRTBLK
* LOADTEMP
* ADD
* ENDBLK
* DISPACC
* HALT

The working mechanism of the program is as follows:

At first, memory address line is at 0 and is set to LOADLOOP that asks user for the amount of number that they want to add and stores it to Loop register. Program Counter (PC) increases and memory address line 1 is executed, STRTBLK starts the block of code to be executed in loop. LOADTEMP asks input and stores it in temporary register. ADD instruction adds the value stored in the accumulator and the temp register and stores the output to accumulator. Then loop register value is decreased by 1 and the loop is re-executed until the value gets 0. After the loop register value is 0, PC increases and ENDBLK instruction ends the loop block. PC counter increases and DISPACC instruction in memory address 5 displays the output that is stored in accumulator. And at last increment in PC executes HALT instruction that terminates the program.

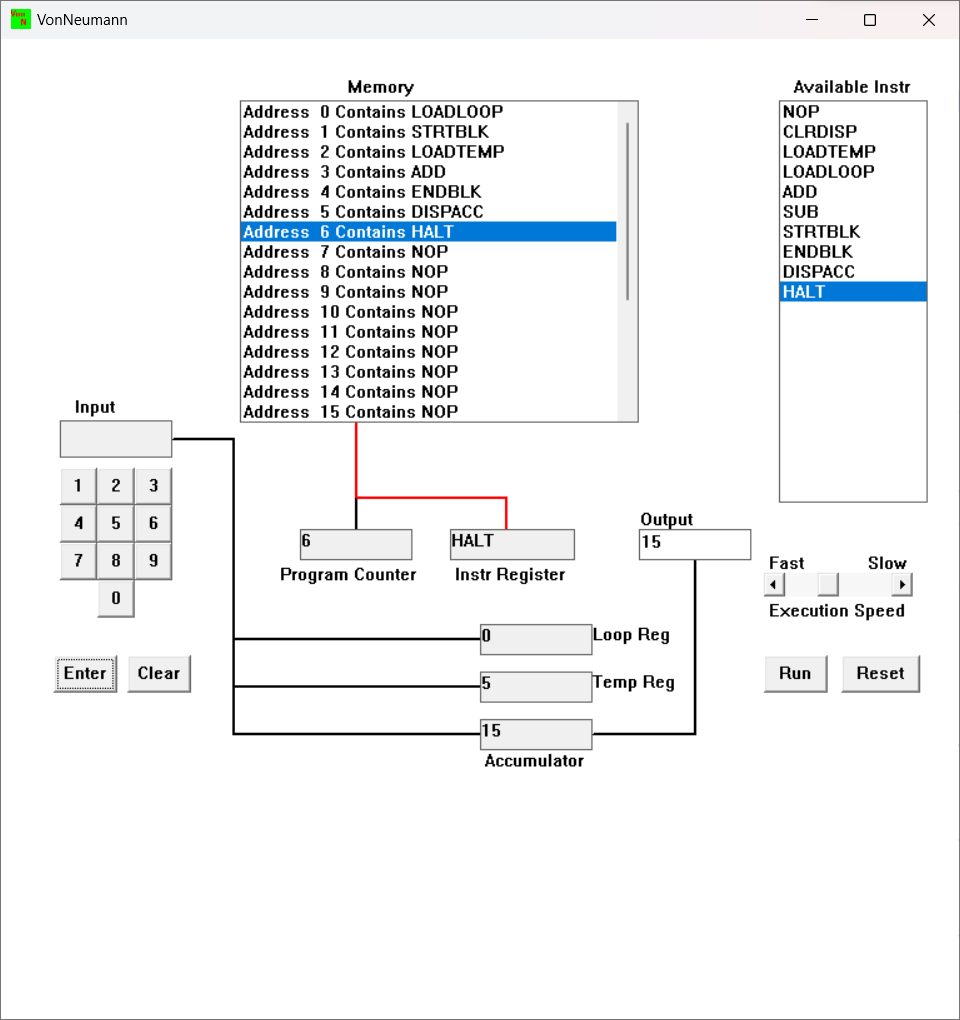


Figure 8 A program to let add multiple numbers as per input in VonNeumann.exe

In this figure, 5 is entered for loop register value. 1,2,3,4,5 input is simultaneously entered to the temporary register and the sum of 1,2,3,4,5 is displayed as output.

### Qno. I

Write a program that will let you multiply 2 numbers together. List your program below and explain how it works (35 marks)

-: the program to multiply 2 numbers together:

- LOADTEMP

- LOADLOOP

- STRTBLK

- ADD

- ENDBLK

- DISPACC

- HALT

And the working mechanism of the program can be explained as:

At first, memory address line is at 0 as Program Counter (PC) is at 0, it executes the instruction LOADTEMP set at address line 0. LOADTEMP ask for user input and stores it to temporary address register. PC increases, LOADLOOP asks for input for number of times the loop block to be executes and stores it to the loop register. PC increases, STRTBLK instruction initiates the block of code that needs executed in loop. The ADD instruction adds the temp register value to the accumulator value and stores it as new value, the loop register values decreases by 1and loop is re-executed form STRTBLK again until the loop register value is 0. When loop register value is 0, PC counter value increases and DISPACC instruction displays output stored in accumulator and HALT instruction terminates the program.

The program performs multiplication by adding the temporary register value for loop register value which is simply multiplication. For instance: 2 times 3 is 2+2+2.

The LOADTEMP value acts as first value and LOADLOOP value acts as the number to be multiplied.

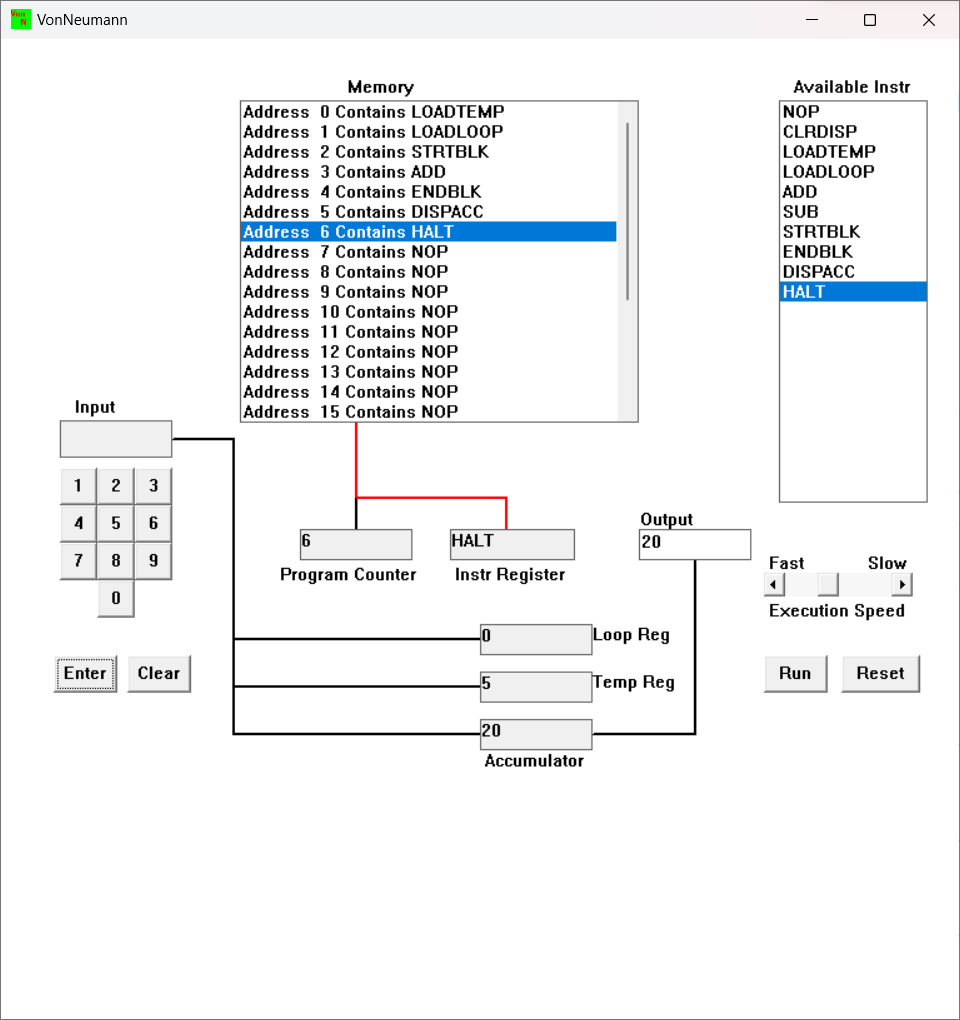


Figure 9 Program to Multiply 2 numbers in VonNeumann.exe

-: In this figure, LOADTEMP instruction asked for first value for multiplication, entered 4. Then LOADLOOP asked second value for multiplication, entered 5, The Loops iterated 5 times which executed the instruction ADD to Temp register value (4) and which simultaneously added the prior and current value and stored the sum as new value. At last, DISPACC instruction displayed the output. HALT instruction terminated the program.

# Conclusion

The portfolio task was a great opportunity for learning and enhancing the Vonneumann architecture. The task contained different challenges for logic development and it was a great way to visualize the process of execution of program and instructions in processor level. The coursework consisted different program that were meant to perform different mathematical tasks such as multiplication between 2 numbers, addition of 3 numbers, addition of numbers as per user need, subtraction of number and others. It used different concepts for looping, loading values and displaying message along with terminating message. As a whole, the portfolio has played a significant role for understanding processes in CPU.