James Salzer

tinyvm readme file

Date: 07/26/2023

-------------------------BRIEF OVERVIEW-------------------------

This program emulates the design of a Virtual Tiny Machine known as the Tiny-Harvard Machine Architecture, coded in C programming language. This VM has the following architecture; Instruction Register(IR), Program Counter(PC), Memory Address Register(MAR), Data Memory(DM), Memory Data Register(MDR), Instruction Memory(IM), and Accumulator(A). The IM will be capable of holding 129 instruction lines, and the capacity to hold 10 data memory slots.

-------------------------RUNNING A PROGRAM ------------------------

-When running a program the user must have a compiler that will operate using “C” Programming Language.

-The user must place both the “elf.txt” and “tinyvm.c” in the same file path location.

-User will run the “tinyvm.c” program using acceptable compiler.

-------------------------WRITING A PROGRAM----------------------

Program will execute by reading the input of a "elf.txt".

-The "elf.txt" file must be written in columns of 2 using decimal numbers.

-The first column will represent the Op-Code

-Op-Code must be in-between [1-11].

-The second column will represent the Address

-Data Memory Address must be between [0-9]

\* 1 is hard coded to be stored in Memory Address [10]

-The files’ last line must end with |7 0| to execute HALT

Programs can be executed with ONLY the following instructions:

Opcode Mnemonic Meaning

1.) LOAD <X> A <- DM[X]

2.) ADD <X> A <- A + DM[X]

3.) STORE <X> MD[X] <- A

4.) SUB <X> A <- A - DM[X]

5.) IN <5> A <- Read from keyboard

6.) OUT <7> A -> Write to screen

7.) HALT End of program

8.) JMP <X> PC <- A

9.) SKIPZ If A = 0; PC <- PC + 2

10.) SKIPG If A > 0; PC <- PC + 2

11.) SKIPL If A < 0; PC <- PC + 2

-------------------------EXAMPLE PROGRAM------------------------

-In this example two numbers inputted by the user will be added together.

-A + B = C

5 5 //IN <5>

6 7 //OUT <7>

3 0 //STORE IN DM[0]

5 5 //IN <5>

6 7 //OUT <7>

3 1 //STORE IN DM[1]

2 0 //ADD DM[0]

3 2 //STORE IN DM[2]

6 7 //OUT <7>

7 0 //HALT (End of program!)

-Result of this program running:

Would you like to see the state of the VM when executing the program? (y/n): y

PC = 0 | A = 0 | DM = [0,0,0,0,0,0,0,0,0,0]

IN <5>

Input a value: 6

PC = 2 | A = 6 | DM = [0,0,0,0,0,0,0,0,0,0]

OUT <7>

Result is: 6

PC = 4 | A = 6 | DM = [0,0,0,0,0,0,0,0,0,0]

STORE <0>

PC = 6 | A = 6 | DM = [6,0,0,0,0,0,0,0,0,0]

IN <5>

Input a value: 4

PC = 8 | A = 4 | DM = [6,0,0,0,0,0,0,0,0,0]

OUT <7>

Result is: 4

PC = 10 | A = 4 | DM = [6,0,0,0,0,0,0,0,0,0]

STORE <1>

PC = 12 | A = 4 | DM = [6,4,0,0,0,0,0,0,0,0]

ADD <0>

PC = 14 | A = 10 | DM = [6,4,0,0,0,0,0,0,0,0]

STORE <2>

PC = 16 | A = 10 | DM = [6,4,10,0,0,0,0,0,0,0]

OUT <7>

Result is: 10

PC = 18 | A = 10 | DM = [6,4,10,0,0,0,0,0,0,0]

End of Program!

PC = 18 | A = 10 | DM = [6,4,10,0,0,0,0,0,0,0]

-------------------------PROGRAM EXECUTION----------------------

The program executes by using a FETCH CYCLE to retrieve OP-CODE/ADDRESS in the Instruction Memory and store it in the Instruction Register. After each FETCH CYCLE, Program Counter will increase by 2, then the stored instruction will be executed. This will continue until the Instruction Op-Code 7 (HALT) has been read/executed. When the program is run, it will ask the user if they would like to display updated VM information. If selected 'n', the program will only display imperative information.

During/After each execution cycle the following information will be updated and displayed in the following format:

-Before

- Program Counter

- Accumulator

- Data Memory[0-9]

-Example: PC = X | A = X | DM = [X,X,X,X,X,X,X,X,X,X]

-During

-Op-Code

-Address

-Examples: STORE <X>

LOAD <X>

JUMP <X>

-------------------------PROGRAM PROVIDED------------------------

The program provided is designed to multiply two decimal numbers together. Multiplication is not hard coded into the machine. Instead, multiplication will be achieved by using loops to add the same number together by the desired number of times, i.e. 2 x 3 can be written as 2 + 2 + 2.

-The program will ask the user to input decimal numbers after the message "Input a value: ", this will happen twice.

-The program will run with any two decimal numbers given by the user greater than 0.

-If user inputs a number smaller than 1, the program will display given number and then end.

-The user will be notified when the program has finished running with the message "End of Program!"