VARIANCE CALCULATION

Team Members :

1. Arnav Oruganty (IMT2023078)
2. Aaryan Antala (IMT2023548)
3. Rakshith Srinivasan (IMT2023544)

Explanation Of Code :

We have implemented a program to calculate the variance of the first n natural numbers (where n is odd).

For the same, we have created 5 program files:

1) VarianceCalculation.c
2) Assembly.txt
3) Assembler.py
4) Machine.txt
5) processor.py

1) **VarianceCalculation.c** is the main program code written in c language in order to find the variance of the first n natural numbers (where n is odd). To do the same we have first calculated the mean of these n natural numbers and then used the formula of variance to return our desired result.

2) **Assembly.txt** is the assembly code for our processor which includes the instructions to be followed by the processor to execute the program.

3) **Assembler.py** is the python file which reads the Assembly.txt file line by line and converts the instructions in the Assembly.txt file into 40 bit binary numbers which can be understood by the computer and know which instruction has to be executed. These 40 bit binary numbers are stored and returned in the Machine.txt file.

4) **Machine.txt** is the file which contains the 40 bits binary numbers, that stores the instructions and is imported by the processor.py file in order to execute the instructions and get the desired output.

5) **Processor.py :** It reads instructions from a file called "Machine.txt" and stores them in memory. It sets up specific memory locations to store values like the mean, variance, and

loop counters.Next, it creates a class called IAS_Machine that simulates a basic IAS processor. This processor repeatedly fetches instructions from memory, decodes them, and executes them using various registers and operations like LOAD, STOR, ADD, SUB, INC, DIV, SQR, and JUMP. The specific instructions, when executed, perform the steps to calculate the variance of the first 5 natural numbers (1, 2, 3, 4, 5). After the calculation is complete and the program has exited, it prints the final variance stored in memory.

Extra Instructions :

Optcode	Symbol	Function
11000000	INC	The contents of the memory location is increased by 1. Steps to implement this instruction: (Assume that we have to increment the contents in memory location 501 and the memory location of 500 contains 1) LOAD 501 ADD 500 STOR 501 NOP 0
10000000	SQR	The data stored in memory location M(X) is first loaded into the MBR, then it is squared in the ALU and the least significant bits are stored in AC.
11100000	NOP	Implies to do nothing and just to move on to the next instruction.
10101010	EXIT	It sends a signal to the processor to stop its operations.

Terminal Output :

We have displayed few screenshots of the output generated by the processor :

LOAD M(500) SUB M(506)
JUMP+ M(6,0:19) LOAD M(501)

```
After fetching the instructions, the registers contain as follows:
PC : 1
MAR <-- PC : 1
MBR <-- M[MAR] : 00000001000111110100000011000011111010
IBR <-- MBR[20:39] :  00000110000111111010
IR <-- MBR[0:7] :  00000001
MAR <-- MBR[8:19] :  000111110100
--------------------------------------------------------------------------------
LOAD :
MBR <-- M(500 :) 00000000000000000000000000000000000000
AC <-- MBR :    00000000000000000000000000000000000000
--------------------------------------------------------------------------------
After fetching the right instructions, the registers contain as follows:
IR <-- IBR[0:7] :  00000110
MAR <-- IBR[8:19] :  000111111010
PC <-- PC + 1 : 2
--------------------------------------------------------------------------------
SUB :
ALU <-- AC :  00000000000000000000000000000000000000
MBR <-- M(506) :  00000000000000000000000000000000000110
AC <-- MBR :  00000000000000000000000000000000000110
ALU <-- ALU - AC :  11111111111111111111111111111111111010
AC <-- ALU :  11111111111111111111111111111111111010
--------------------------------------------------------------------------------
After fetching the instructions, the registers contain as follows:
PC : 2
MAR <-- PC : 2
MBR <-- M[MAR] : 00001111000000000110000000001000111110101
IBR <-- MBR[20:39] :  00000001000111110101
IR <-- MBR[0:7] :  00001111
MAR <-- MBR[8:19] :  000000000110
--------------------------------------------------------------------------------
JUMP + :
Condition not satisfied, so will continue:
--------------------------------------------------------------------------------
After fetching the right instructions, the registers contain as follows:
IR <-- IBR[0:7] :  00000001
MAR <-- IBR[8:19] :  000111110101
PC <-- PC + 1 : 3
--------------------------------------------------------------------------------
LOAD :
MBR <-- M(501 :) 00000000000000000000000000000000000000
AC <-- MBR :    00000000000000000000000000000000000000
--------------------------------------------------------------------------------
```

SQR M(505) ADD M(504)
STOR M(504) INC M(503)

```
After fetching the instructions, the registers contain as follows:
PC : 11
MAR <-- PC : 11
MBR <-- M[MAR] : 10000000000111111001000001010001111111000
IBR <-- MBR[20:39] :  00000101000111111000
IR <-- MBR[0:7] :  10000000
MAR <-- MBR[8:19] :  000111111001
--------------------------------------------------------------------------
SQR :
MBR <-- M(505) :  00000000000000000000000000000000000000000
AC <-- MBR :  00000000000000000000000000000000000000000
ALU <-- AC :  00000000000000000000000000000000000000000
ALU <-- ALU * ALU :  00000000000000000000000000000000000000000
AC <-- ALU :  00000000000000000000000000000000000000000
--------------------------------------------------------------------------
After fetching the right instructions, the registers contain as follows:
IR <-- IBR[0:7] :  00000101
MAR <-- IBR[8:19] :  000111111000
PC <-- PC + 1 : 12
--------------------------------------------------------------------------
ADD :
ALU <-- AC :  00000000000000000000000000000000000000000
MBR <-- M(504) :  00000000000000000000000000000000000000101
AC <-- MBR :  00000000000000000000000000000000000000101
ALU <-- ALU + AC :  00000000000000000000000000000000000000101
AC <-- ALU :  00000000000000000000000000000000000000101
--------------------------------------------------------------------------
After fetching the instructions, the registers contain as follows:
PC : 12
MAR <-- PC : 12
MBR <-- M[MAR] : 00100001000111111000110000000000111110111
IBR <-- MBR[20:39] :  11000000000111110111
IR <-- MBR[0:7] :  00100001
MAR <-- MBR[8:19] :  000111111000
--------------------------------------------------------------------------
STOR :
MBR <-- AC :  00000000000000000000000000000000000000101
M(504) <-- MBR :  00000000000000000000000000000000000000101
--------------------------------------------------------------------------
After fetching the right instructions, the registers contain as follows:
IR <-- IBR[0:7] :  11000000
MAR <-- IBR[8:19] :  000111110111
PC <-- PC + 1 : 13
--------------------------------------------------------------------------
INC :
MBR <-- M(503) :  00000000000000000000000000000000000000011
AC <-- MBR :  00000000000000000000000000000000000000011
ALU <-- AC :  00000000000000000000000000000000000000011
ALU <-- ALU + 1 :  00000000000000000000000000000000000000100
AC <-- ALU :  00000000000000000000000000000000000000100
--------------------------------------------------------------------------
```

LOAD M(501) DIV M(502)
LOAD MQ STOR M(501)

```
After fetching the instructions, the registers contain as follows:
PC : 6
MAR <-- PC : 6
MBR <-- M[MAR] : 0000000100011111010100001100000111110110
IBR <-- MBR[20:39] :  00001100000111110110
IR <-- MBR[0:7] :  00000001
MAR <-- MBR[8:19] :  000111110101
--------------------------------------------------------------------------
LOAD :
MBR <-- M(501 :) 00000000000000000000000000000001111
AC <-- MBR :    00000000000000000000000000000001111
--------------------------------------------------------------------------
After fetching the right instructions, the registers contain as follows:
IR <-- IBR[0:7] :  00001100
MAR <-- IBR[8:19] :  000111110110
PC <-- PC + 1 : 7
--------------------------------------------------------------------------
DIV :
MBR <-- M(502) :  00000000000000000000000000000000101
ALU <-- AC : 15
ALU <-- MBR :  00000000000000000000000000000000101
MQ <-- AC / M(X)  :  00000000000000000000000000000000011
--------------------------------------------------------------------------
After fetching the instructions, the registers contain as follows:
PC : 7
MAR <-- PC : 7
MBR <-- M[MAR] : 0000101000000000000010000100011111010 1
IBR <-- MBR[20:39] :  00100001000111110101
IR <-- MBR[0:7] :  00001010
MAR <-- MBR[8:19] :  000000000000
--------------------------------------------------------------------------
LOAD MQ :
AC <-- MQ :  00000000000000000000000000000000011
--------------------------------------------------------------------------
After fetching the right instructions, the registers contain as follows:
IR <-- IBR[0:7] :  00100001
MAR <-- IBR[8:19] :  000111110101
PC <-- PC + 1 : 8
--------------------------------------------------------------------------
STOR :
MBR <-- AC :  00000000000000000000000000000000011
M(501) <-- MBR :  00000000000000000000000000000000011
```

LOAD M(504) DIV M(502)
LOAD MQ STOR M(504)
EXIT 0 NOP 0

```
After fetching the instructions, the registers contain as follows:
PC : 14
MAR <-- PC : 14
MBR <-- M[MAR] : 00000001000111111000000011000001111110110
IBR <-- MBR[20:39] :  00001100000111110110
IR <-- MBR[0:7] :  00000001
MAR <-- MBR[8:19] :  000111111000
--------------------------------------------------------------------------
LOAD :
MBR <-- M(504 :) 00000000000000000000000000000001010
AC <-- MBR :    00000000000000000000000000000001010
--------------------------------------------------------------------------
After fetching the right instructions, the registers contain as follows:
IR <-- IBR[0:7] :  00001100
MAR <-- IBR[8:19] :  000111110110
PC <-- PC + 1 : 15
--------------------------------------------------------------------------
DIV :
MBR <-- M(502) :  00000000000000000000000000000000101
ALU <-- AC : 10
ALU <-- MBR :  00000000000000000000000000000000101
MQ <-- AC / M(X)  :  00000000000000000000000000000000010
--------------------------------------------------------------------------
After fetching the instructions, the registers contain as follows:
PC : 15
MAR <-- PC : 15
MBR <-- M[MAR] : 00001010000000000000010000100011111000
IBR <-- MBR[20:39] :  00100001000111111000
IR <-- MBR[0:7] :  00001010
MAR <-- MBR[8:19] :  000000000000
--------------------------------------------------------------------------
LOAD MQ :
AC <-- MQ :  00000000000000000000000000000000010
--------------------------------------------------------------------------
After fetching the right instructions, the registers contain as follows:
IR <-- IBR[0:7] :  00100001
MAR <-- IBR[8:19] :  000111111000
PC <-- PC + 1 : 16
--------------------------------------------------------------------------
STOR :
MBR <-- AC :  00000000000000000000000000000000010
M(504) <-- MBR :  00000000000000000000000000000000010
--------------------------------------------------------------------------
After fetching the instructions, the registers contain as follows:
PC : 16
MAR <-- PC : 16
MBR <-- M[MAR] : 10101010000000000001110000000000000000
IBR <-- MBR[20:39] :  11100000000000000000
IR <-- MBR[0:7] :  10101010
MAR <-- MBR[8:19] :  000000000000
--------------------------------------------------------------------------
EXIT :
The program will now exit.
--------------------------------------------------------------------------
The variance of the first 5 natural numbers is: 2
```