Name: JAHANVI AGRAWAL
Roll No.: IMT2019506

PROGRAM IMPLEMENTED:
The program I have implemented using the IAS architecture is to Find the sum of digits of a positive number.
For example: 345
3+4+5=12

C implementation:

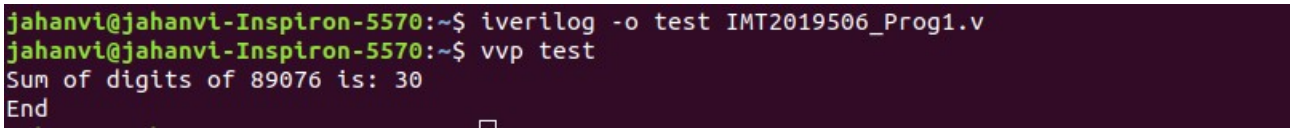
```
a= N;
sum=0;
while( a!=0 )
{
        sum = sum + a%10;
        a=a/10;
}
return sum;
```

Corresponding implementation using ISA intructions in the memory:

```
LOAD M(17)
STOR M(19)
LOAD -M(16)
JUMP + M(5,20:39)
LOAD M(16)
DIV M(18)
ADD M(19)
STOR M(19)
LOAD MQ
STOR M(16)
JUMP M(1,0:19)
LOAD M(19)
HALT()
```

Memory Location	**LHS Instruction**	**RHS Instruction**
0	LOAD M(17)	STOR M(19)
1	LOAD -M(16)	JUMP + M(5,20:39)
2	LOAD M(16)	DIV M(18)
3	ADD M(19)	STOR M(19)
4	LOAD MQ	STOR M(16)
5	JUMP M(1,0:19)	LOAD M(19)
6	XXXXXXXXXX	HALT()
16	89076	
17	0	
18	10	
20	89076	

OUTPUT:
This is the snapshot of the terminal.



Memory Allocations and Assumptions:

- Initially PC is set to 0.
- In the program above, I have calculated the sum of digits of 89076 but the value of any positive number N can be entered in location 16 and location 20 as a 40bit binary to calculate sum of it's digits.
- At location 16, "a" is stored, which is divided by 10 (which is stored at location 18) in each iteration.
- At location 17, 0 is stored to initialize the value of sum.
- The final result is stored at location 19 in each iteration.
- The N must be stored at location 16 and 20 in the start as the value in location 16 is changed to a/10 in each iteration. Thus, we need N at location 20 for future use after loop ends.
- Memory locations from 0 to 15 are used for writing the instructions and the next locations are for data storage (this can be changed).
- If the LHS or RHS instruction is 20'bX, it means that there is no instruction there.

Explaination:
Initially 0 is loaded into AC from location 17 and then stored to location 19 (where the sum is stored in each iteration). At location 16, num=N is stored.
Now, to check the loop condition we load -M(16) into the accumulator because when the content of the accumulator is non-negative (i.e. -M(16) >=0) the loop stops. And when the content of AC is negative (i.e. -M(16) <0 i.e. M(16) >0) the loop should continue. This loop checking is achieved through the JUMP + M(X,20:39) instruction.
In each loop, the value of "a" is loaded into AC from location 16. Then we use DIV M(X) instruction to divide content at AC by 10 (which is at location 18). The quotient is stored in MQ and the remainder is stored in AC. So we add sum with the remainder in AC using ADD M(19) instruction and then this sum is stored back to location 19 (i.e. here we achieve sum=sum+a%10). After this, we load AC with content of MQ (which holds a/10) using LOAD MQ instruction. Then we store this back to location 16 (i.e. here we achieve a=a/10). After this, we JUMP to the part of checking the loop condition (i.e. we JUMP to location 1 to check this).
When the loop ends, the sum is in location 19 which is loaded back to AC and the program halts.