Name: JAHANVI AGRAWAL
Roll No.: IMT2019506

PROGRAM IMPLEMENTED:

The program I have implemented using the IAS architecture is to Find the sum of first N natural numbers.

C implementation:

```
res=0;
i=1;
while( i<(N+1) )
{
        res=res+i;
        i=i+1;
}
return res;
```

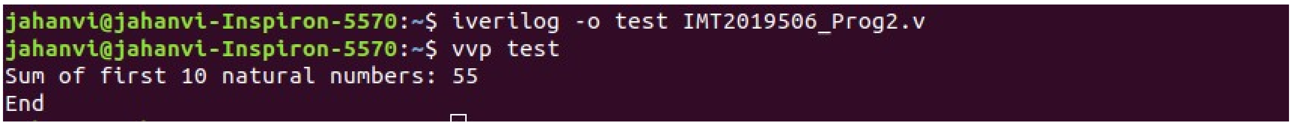
Corresponding implementation using ISA intructions in the memory:

```
LOAD M(20)
STOR M(19)
LOAD M(18)
SUB M(17)
JUMP + M(6,0:19)
LOAD M(19)
ADD M(18)
STOR M(19)
LOAD M(18)
ADD M(16)
STOR M(18)
JUMP M(1,0:19)
LOAD M(19)
HALT()
```

Memory Location	LHS Instruction	RHS Instruction
0	LOAD M(20)	STOR M(19)
1	LOAD M(18)	SUB M(17)
2	JUMP + M(6,0:19)	LOAD M(19)
3	ADD M(18)	STOR M(19)
4	LOAD M(18)	ADD M(16)
5	STOR M(18)	JUMP M(1,0:19)
6	LOAD M(19)	XXXXXXXXXX
7	XXXXXXXXXX	HALT()
16	1	
17	11	
18	1	
20	0	

OUTPUT:

This is the snapshot of the terminal.



```
jahanvi@jahanvi-Inspiron-5570:~$ iverilog -o test IMT2019506_Prog2.v
jahanvi@jahanvi-Inspiron-5570:~$ vvp test
Sum of first 10 natural numbers: 55
End
```

Memory Allocations and Assumptions:

- Initially PC is set to 0.
- I have calculated the sum of first 10 natural numbers here, but the value of N can be changed by entering N+1 as a 40 bit binary at memory location 17 (So to calculate the sum of first 15 natural numbers you must write 16 at location 17)
- Memory locations from 0 to 15 are used for storing the instructions and the next locations are for data storage (this can be changed).
- At location 16, 1 is stored as a 40 bit binary
- At location 19, the result is stored after each interation.
- At location 18, the counter i is stored and incremented in each iteration.
- If the LHS or RHS instruction is 20’bX, it means that there is no instruction there.

Explaination:

Initially 0 is loaded into the accumulator from location 20 and then stored to location 19 (where the result is stored after each iteration). Then AC is loaded with 1 from location 18 (where the counter is stored and is incremented after each iteration). 11 is subtracted from the AC to get i-11 in it for checking the loop condition.

Now, to check the loop condition we have loaded i-11 in the accumulator (where i=1 initially) because when the content of AC is non-negative (i.e. i-11>=0) the loop stops. And when the content of AC is negative (i.e. i-11<0) the loop should continue. This loop checking is achieved through the JUMP + M(X,0:19) instruction.

In each loop, the res value is loaded into AC from location 19, the counter i (stored at location 18) is added to content of AC (i.e. here we achieve res=res+i) and the result is now stored back to location 19. Also, now the counter is incremented by loading it from the location 18 and adding 1 (stored at location 16) to it (i.e. here we do i=i+1). This is stored back to location 18. 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 result is in location 19 which is loaded back to AC and the program halts.