My processor does 5 tasks-

1. Swap two numbers at memory location without using temporary variable
2. Add two numbers (a+b)
3. Subtract two numbers (a-b)
4. Finding perimeter of rectangle 2*(a+b) using LSH operation
5. Finding 5th element of Fibonacci series (indexing starts from zero)

>>If the user inputs 1 then numbers will be swapped, if 2 then the sum of two numbers will be implemented and 3 then subtraction of two numbers, if 4 then perimeter of rectangle and if 5 then 5th element of Fibonacci series

For the above implementation, the user will change the assembly language

>>My code is written in c++.

<u>The assembly language for my processor and the equivalent "40 bits" instructions code is here: -</u>

1. Assembly language for swapping two numbers without using temporary variable :-

```
LOAD M(X) 200 ADD M(X) 201- 0000000100001100100000000101000011001001
STOR M(X)200- 0000000000000000000000100001000011001000
LOAD M(X) 200 SUB M(X) 201- 0000000100001100100000000110000011001001
STOR M(X) 201 - 0000000000000000000000100001000011001001
LOAD M(X) 200 SUB M(X) 201- 0000000100001100100000000110000011001001
STOR M(X) 200- 0000000000000000000000100001000011001000
HALT- 0000000000000000000000000000000000000000
```

2. Assembly language for adding two numbers:-

```
LOAD M(X) 200 ADD M(X) 201- 0000000100001100100000000101000011001001
STOR M(X) 200- 0000000000000000000000100001000011001000
HALT - 0000000000000000000000000000000000000000
```

3. Assembly language for subtracting two numbers(a-b):-

```
LOAD M(X) 200 SUB M(X) 201 - 0000000100001100100000000110000011001001
STOR M(X)200- 0000000000000000000000100001000011001000
HALT- 0000000000000000000000000000000000000000
```

4. Assembly language for finding perimeter of rectangle using LSH(2*(A+B)(A>0 AND B>0)

```
LOAD M(X) 200 ADD M(X) 201 - 0000000100001100100000000101000011001001
STOR M(X) 200 LSH 1 - 0010000100001100100000010100000000000001
STOR M(X) 200 - 0000000000000000000000100001000011001000
HALT - 0000000000000000000000000000000000000000
```

5. Assembly language for finding 5th element of Fibonacci series- user will have input 0 and 1 in the start

```
LOAD M(X) 200 ADD M(X) 201- 0000000100001100100000000101000011001001

STOR M(X) 202- 0000000000000000000000100001000011001010

LOAD M(X) 201 STOR M(X) 200- 0000000100001100100100100001000011001000

LOAD M(X) 202 STOR M(X) 201- 0000000100001100101000100001000011001001

JUMP M(X,0:19) 0- 0000000000000000000000001101000000000000

HALT- 0000000000000000000000000000000000000000
```

>> I have implemented an assembler which converts English instructions to 40 bit binary

INPUT

The user has to type the Assembly language instructions as it is. The user can give two numbers of its choice

>>When the program is made to run the following options user gets

Enter '1' to Swap two Numbers

Enter '2' to Add two Numbers

Enter '3' to Subtract two numbers(a-b)

Enter ‘4’ to find perimeter of Rectangle

Enter ‘5’ to find 5th element of Fibonacci series

- If user inputs 1 then user has to input two numbers separated by a space and then user will input the Assembly language instructions as it is

So the entire input will look like

```
1
7 5
LOAD M(X) 0 ADD M(X) 1
STOR M(X) 0
LOAD M(X) 0 SUB M(X) 1
STOR M(X) 1
LOAD M(X) 0 SUB M(X) 1
STOR M(X)
0 HALT
```

- If user inputs 2 then user has to input two numbers separated by a space and then user will input the Assembly language instructions as it is

So the entire input will look like

```
2
7 5
LOAD M(X) 0 ADD M(X) 1
STOR M(X)
0 HALT
```

- If user inputs 3 then user has to input two numbers separated by a space and then user will input the Assembly language instructions as it is

So the entire input will look like

```
2
7 5
LOAD M(X) 0 SUB M(X) 1
STOR M(X)
0 HALT
```

- If user inputs 4 then user has to input two numbers separated by a space and then user will input the Assembly language instructions as it is

```
4
7 5
LOAD M(X) 0 ADD M(X) 1
STOR M(X) 0 LSH 1
STOR M(X)
0 HALT
```

- If user inputs 5 then user has to input 0 and 1 separated by a space and then user will input the Assembly language instructions as it is

(using JUMP)

So the entire input will look like

```
LOAD M(X) 200 ADD M(X) 201
STOR M(X) 202
LOAD M(X) 201 STOR M(X) 200
LOAD M(X) 202 STOR M(X) 201
JUMP M(X,0:19)
0 HALT
```

>> My memory location starts from 200 and PC from 0