# Computer Architecture Lab 02

**Yihang Qiu, 520030910155, 2021/12**

## Explanation of "lab2.asm"

For **TASK 2**, I made some minor adjustments such as changing "`RET`" in `OpAdd, OpClear,` etc. to "`BRnzp NewCommand`". The change can maintain the expected functions while avoiding the problem caused by the nesting of `RET` command.

Also, to ensure the stack can be visited by any part of the program using PC + Offset Addressing Mode, the stack is moved to the middle part of the whole program.

Moreover, it seemed that the code of `PushValue` given (10.23) failed to deal with the cases when the input is in the form of "Dx3", **i.e.** a 3-bit input with non-number characters. The given code regard it as a legal input and using its ascii minus 48 as the "number". Meanwhile, the given `PushValue` failed to deal with the cases when the input is a pure carriage, which will be regarded as "0".

I modified the given code to ensure the input is legal. View **`lab2.asm`** for specific details.

For **TASK 3**, `OpMod` and `OpXor` is added for operation extension.

In module `OpMod`, we regard the first element popping out of the stack as the modulus and the second one as the number to be modulared.

Use R1 and R0 to store the modulus and the modulared number respectively. Check whether `POP` is successful while popping the stack. Then we check whether R1 (the modulus) is legal, **i.e.** positive. If R1 is non-positive, raise an exception with message "`Error: Modulus is a Non-positive Number.`"

After validity check, we continue subtract R0 by R1 (**i.e.** add -R1 to R0) until R0 is negative. Then we continue adding R1 to R0 until R0 is non-negative. At this time, R0 is the result.

Since the result is non-negative and smaller than the modulus, we know the result is for sure within the range. Thus, `RangeCheck` is not needed in this process.

Eventually, PUSH R0 onto the stack.

In module `OpXor`, the operands are always valid.

Use R1 and R2 to store the two operands, **i.e.** the top two elements popped out of the stack. Check whether `POP` is successful while popping the stack.

Use R3 as a mask. Initialize R3 as `0000 0001`. We discuss one bit at a time. When we need to discuss the next bit, we double R3.

We find that (R1 AND R3 + R2 AND R3) AND R3 (where $R3 = 2^x$) will produce the result of R1 XOR R2 masked by R3. Therefore, we accumulate the result of (R1 AND R3 + R2 AND R3) AND R3 for all R3s and get the final result.

Check whether the result (stored in R0) is within the valid range.

Eventually, PUSH R0 onto the stack.

## Runtime Screenshots of "lab2.obj"

Illegal inputs and several special inputs are tested on the code. Different legal cases are also tested.

The runtime screenshots are as follows.

The commands boxed by blue test MOD.

LC3 Console

```
Enter a command:D
Error: Too Few Values on the Stack.
Enter a command:a
Too many digits or Illegal Input.
Enter a command:899

Enter a command:+
Error: Too Few Values on the Stack.
Enter a command:12

Enter a command:_
Too many digits or Illegal Input.
Enter a command:+
Enter a command:D
+911
Enter a command:-
Enter a command:D
-911
Enter a command:C
Enter a command:-
Error: Too Few Values on the Stack.
Enter a command:

Too many digits or Illegal Input.
Enter a command:899

Enter a command:12

Enter a command:+
Enter a command:D
+911
Enter a command:90

Enter a command:+
Error: Number is out of range.
Enter a command:%
Enter a command:D
+011
Enter a command:-
Enter a command:23

Enter a command:%
Enter a command:D
+012
Enter a command:12

Enter a command:9

Enter a command:*
Enter a command:D
+108
Enter a command:8

Enter a command:*
Enter a command:D
+864
Enter a command:2*
Too many digits or Illegal Input.
Enter a command:2

Enter a command:*
Error: Number is out of range.
Enter a command:32

Enter a command:-
Enter a command:%
Error: Modulus is a Non-positive Number.
```

The commands boxed by red test XOR.

```
Enter a command:C
Enter a command:999

Enter a command:887

Enter a command:@
Enter a command:D
+144
Enter a command:1

Enter a command:-
Enter a command:+
Enter a command:887

Enter a command:@
Error: Number is out of range.
Enter a command:C
Enter a command:D
Error: Too Few Values on the Stack.
Enter a command:X
----- Halting the processor -----
```