# Chimera-2018-A Emulator Assignment

## Practical 6 - Rotate

CANS Tech INC

# Implementing the RLCA Instruction

Once again inside the Group_1 function switch add

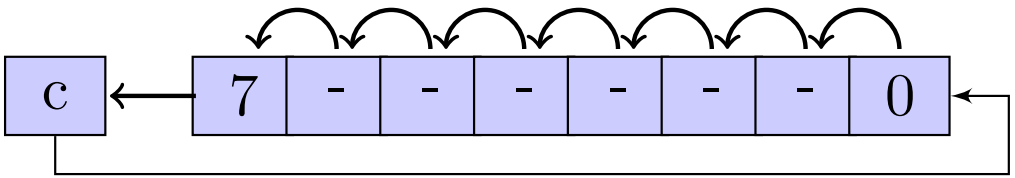
```
case 0x68: // RLCA
    CODE HERE
    break;
```

| RLCA | | Addressing | Opcode |
|---|---|---|---|
| Rotate left through carry Memory or Accumulator | | A | 0x68 |
| Flags: | T - - - - - T T | | |
| notes | | | |

RLCA instructions rotate the Register A one bit towards left through the carry bit.

If Register A contains 0x7A (01111010) and the carry bit is 1, after ROLA instruction is executed, Register A contains 0xF5 (11110101) and the carry bit is reset to 0.

If you look at the RLCA op-code in detail you will see what it does...



Firstly, save the current flags...

```
saved_flags = Flags;
```

Next, set the carry based on the MSB of Register A...

```
if ((Registers[REGISTER_A] & 0x80) == 0x80) {
    Flags = Flags | FLAG_C;
}
else{
    Flags = Flags & (0xFF - FLAG_C);
}
```

Next, do the shift...

```
Registers[REGISTER_A] = (Registers[REGISTER_A] «1) & 0xFE;
```

Remember the old carry goes into the RLCA of Register A...

```
if ((saved_flags & FLAG_C) == FLAG_C){
    Registers[REGISTER_A] = Registers[REGISTER_A] | 0x01;
}
```

# Don't forget the other flags!

**Compile and run your code to see how many marks you have!**

# Implementing the ASLA Instruction

Once again inside the Group_1 function switch add

```
case 0x78: // ASLA
    CODE HERE
    break;
```

| ASLA | | Addressing | Opcode |
|---|---|---|---|
| Arithmetic shift left Memory or Accumulator | | A | 0x78 |
| Flags: | T - - - - - T T | | |
| notes | | | |

If you look at the ASLA op-code in detail you will see what it does...

It is the same as RLCA except we don't set the LSB to 1 if the carry was set prior to the op-code being execute...

...Good Luck

# Implementing the ASRA Instruction

Once again inside the Group_1 function switch add

```
case 0x88: // ASRA
    CODE HERE
    break;
```

| ASRA | | Addressing | Opcode |
|---|---|---|---|
| Arithmetic shift right Memory or Accumulator | | A | 0x88 |
| Flags: | T - - - - - T T | | |
| notes | | | |

If you look at the ASRA op-code in detail you will see what it does...

Firstly, pre-set the Carry

```
if ((Registers[REGISTER_A] & 0x01) == 0x01) {
      Flags = Flags | FLAG_C;
}
else {
      Flags = Flags & (0xFF - FLAG_C);
}
```

Do the shift...
Registers[REGISTER_A] = (Registers[REGISTER_A] »1) & 0x7F

But there is more, remember that ASRA has sign extention...

Add...

```
if((Flags & FLAG_N) == FLAG_N) {
        Registers[REGISTER_A] = Registers[REGISTER_A] | 0x80
}
```

Remember to test any remaining flags!

**Compile and run your code to see how many marks you have!**

**You should now be able to do all remaining Op_codes, good luck!**

# Questions?