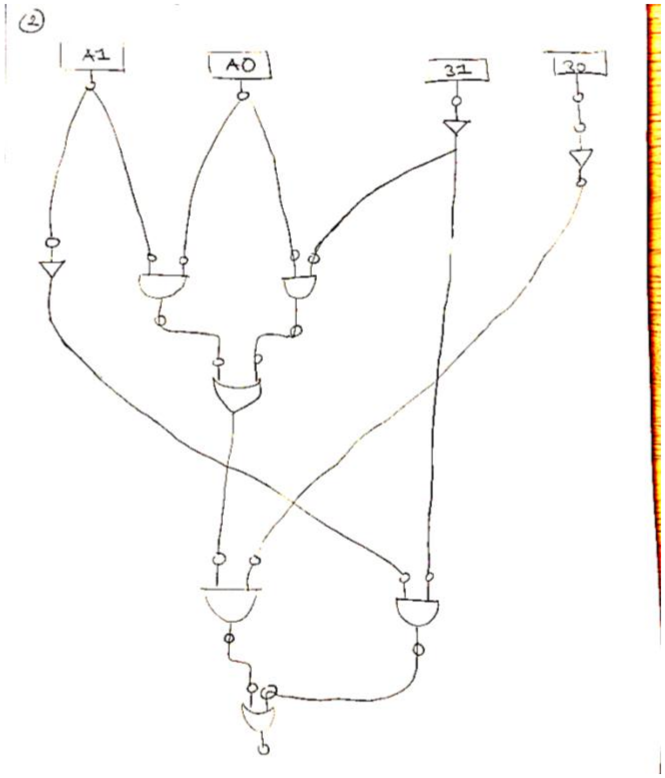
Dev Raiyani & Dawit Solomon – HW 4

1.

(a) : X = (-notA && B && C) || (A && -notB && -notC) || (A && -notB && C) || (A && B && -notC) || (A && B && C)

(b): X = ((A NAND B) NAND C) NAND (A NAND B)

2.



3.

(a) AND 0xAAAAAAAA
(b) OR 0x0000007
(c) AND 0x00000007
(d) OR 0xFFFFFFFF
(e) XOR 0xC0000000
(f) AND 0xFFFFFFF8

4.

```
                            JMP start

value: 0x00

top           LOAD          value

              WRITE         0x08

               ADD          0x01

               STORE        value

               SUB          0x100

               JGZ          top

end:           JMP          end
```

4.

```
C0000003    JMP     start              ; jump over the data area
 00000001   num:      1                ; increasing value stored here
 000000FF   limit:   256               ; count to this amount
 00000001   start:  LOAD    num        ; bringing value in accumulator
 30000008           WRITE   0x8       ; current number output
 40000001           ADD     num       ; counts the next number after adding to itself
 10000001           STORE   num       ; storing it
 50000002            SUB    limit      ; to compare with the limit
 E0000003            JLZ    start     ; keep going, if limit not passed
 C0000009   end:     JMP     end      ; program stops
```

6.

```
Var01: a                              JMP   start
Var02: b
 GCD: 0

  start:        LOAD     var01
                JLZ       loop
                JGZ      loop
                LOAD     var02
                STORE   GCD
                LOAD     var02
                JLZ       loop
                JGZ      loop
                LOAD     var01
                STORE   GCD

  loop:         LOAD     var01
                MOD      var02
                STORE   remainder
                LOAD     var02
                STORE    var01
                LOAD     remainder
                STORE    var02
                JLZ       loop
                JZ         loop
                LOAD     var02
                STORE    GCD
  print:        WRITE    0x200
   end:         JMP       end
```

7.

In mem: y
In accumulator: x

Temp01:0
Temp02:0

```
start:  STORE   Temp01     Storing the accumulator to a temporary variable
        LOAD    0x30AA     Loading the accumulator from the memory address
        STORE   Temp02     storing the value in a new variable: temp02
        LOAD    Temp01     Loading the accumulator with the value stored in temp01
        STORE   0x30AA     storing that value in this memory address
End:    LOAD    Temp02     loading tep02 in the accumulator
```

8.

```
        JMP     start

start:  JGZ     0x837BBE1

        JZ      0x837BBE1

end:    JMP     end
```

9.

(a) After running the code, the data stored for r8 will be switched with the data in r9.

(b) XOR the values r8 and r9 into a value that is r8, XOR the value again and store it into r9, XOR the value again and store it into r8. This swaps the data between r8 and r9.