CSCI 3130 – 32-bit RSC Microcode

```
        t0: AR ← PC
        t1: DR ← M, PC ← PC + 1
        t2: IR ← DR, AR ← PC

HALT    d00_t3: S ← 1

LDAC    d01_t3: DR ← M, PC ← PC + 1
        d01_t4: AR ← DR
        d01_t5: DR ← M
        d01_t6: ACC ← DR
        d01_t7: SC ← 0

STAC    d02_t3: DR ← M, PC ← PC + 1
        d02_t4: AR ← DR
        d02_t5: DR ← ACC
        d02_t6: M ← DR
        d02_t7: SC ← 0

MVAC    d03_t3: R ← ACC
        d03_t4: SC ← 0

MOVR    d04_t3: ACC ← R
        d04_t4: SC ← 0

JUMP    d05_t3: DR ← M
        d05_t4: PC ← DR
        d05_t5: SC ← 0

JMPZ    * if Z=1
        z_d06_t3: DR ← M
        z_d06_t4: PC ← DR
        z_d06_t5: SC ← 0
        * if Z=0
        z'_d06_t3: PC ← PC + 1
        z'_d06_t4: SC ← 0

OUT     d07_t3: OUTR ← ACC
        d07_t4: SC ← 0

SUB     d08_t3: ACC ← ACC - R
        d08_t4: SC ← 0

ADD     d09_t3: ACC ← ACC + R
        d09_t4: SC ← 0

INC     d10_t3: ACC ← ACC + 1
        d10_t4: SC ← 0

CLAC    d11_t3: ACC ← 0
        d11_t4: SC ← 0

AND     d12_t3: ACC ← ACC AND R
        d12_t4: SC ← 0

OR      d13_t3: ACC ← ACC OR R
        d13_t4: SC ← 0

ASHR    d14_t3: ACC ← ASHR ACC
        d14_t4: SC ← 0

NOT     d15_t3: ACC ← ~ACC
        d15_t4: SC ← 0
```

dXX = Instruction, where XX is the **opcode**

```
d00 = HALT = Halt RSC Execution
d01 = LDAC = Load ACC from memory
d02 = STAC = Store ACC to memory
d03 = MVAC = Move ACC to R
d04 = MOVR = Move R to ACC
d05 = JMP  = Jump (unconditional)
d06 = JMPZ = Jump if Z=1
d07 = OUT  = Move ACC to OUTR (Output)
d08 = SUB  = SUB ACC and R (ACC = ACC - R)
d09 = ADD  = ADD ACC and R (ACC = ACC + R)
d10 = INC  = Increment ACC (ACC = ACC + 1)
d11 = CLAC = Clear ACC     (ACC = 0)
d12 = AND  = ACC AND R     (ACC = ACC & R)
d13 = OR   = ACC OR R      (ACC = ACC | R)
d14 = ASHR = ASHR ACC      (ACC = ACC>>1)
d15 = NOT  = NOT ACC       (ACC = ~ACC)
```

Operations Table

```
Data Transfer:
      LDAC address/label
      STAC address/label
      MVAC
      MOVR
      OUT
Data Operation :
      ADD
      SUB
      INC
      CLAC
      AND
      OR
      ASHR
      NOT
Program Control :
      HALT
      JMP  address/label
      JMPZ address/label
```

Sequence Counter (SC)

3-bit Register, with asynchronous **Clear**

SC → t signals by the **Timer (T)**
IR → d signals by the **Decoder (D)**

Registers on the **Bus**

```
AR   (Register) Output-Memory
IR   (Register) Output-Decoder
OUTR (Register) Output-7SegDisplay
DR   (BusRegister←Register) Input-Memory
R    (BusRegister←Register)
ACC  (ClearRegister←BusRegister←Register)
PC   (CountRegister←BusRegister←Register)
```

Registers **NOT** on the **Bus**

```
S (1-bit Register) (S=0 → GO, S=1 → STOP)
Z (1-bit Register) (ACC=0: Z=1 else Z=0)
```