1) Given a one-byte number, store the power of 2 value for each non-zero bit of the number. Store zero for zero-bits of the number. The result will be stored in eight consecutive memory locations. Each location will store either of the power of 2 value or zero.

Example: data: 192

memory:
0000: 192
0001: 0
0002: 0
0003: 0
0004: 0
0005: 0
0006: 0
0007: 64
0008: 128

```
MOVE R1, 00
LOAD R2, 0(R1)
MOVE R3, R2
ANI R2, R2, 1
STORE R2, 1(R1)
MOVE R2, R3
ANI R2, R2, 2
STORE R2, 2(R1)
MOVE R2, R3
ANI R2, R2, 4
STORE R2, 3(R1)
MOVE R2, R3
```

```
ANI R2, R2, 8
STORE R2, 4(R1)
MOVE R2, R3
ANI R2, R2, 16
STORE R2, 5(R1)
MOVE R2, R3
ANI R2, R2, 32
STORE R2, 6(R1)
MOVE R2, R3
ANI R2, R2, 64
STORE R2, 7(R1)
MOVE R2, R3
ANI R2, R2, 128
STORE R2, 8(R1)
HLT
```

2) There are 7 one-byte elements: e0, e1, e2, e3, e4, e5, e6. The second element of each triplet (non-overlapping) should store a value one less than the sum of the triplet's first and third elements. The last element, e6, should store the OR value of these second elements of the triplets, i.e., e1 and e4. The following code performs the below operations:

e1=e0+e2-1

e4=e3+e5-1

e6=e1 or e4

ASSEMBLY CODE:

```
MOVE R1, 00
```

```
LOAD R2, 0(R1)
LOAD R3, 2(R1)
ADD R4, R2, R3
SUI R4, R4, 1
STORE R4, 1(R1)
ADI R1, R1, 3
LOAD R2, 0(R1)
LOAD R3, 2(R1)
ADD R4, R2, R3
SUI R4, R4, 1
STORE R4, 1(R1)
MOVE R1, 00
LOAD R2, 1(R1)
LOAD R3, 4(R1)
OR R4, R2, R3
STORE R4, 6(R1)
HLT
```