1.

a. If this instruction were implemented in the MIPS instruction set, the most appropriate format would be R-type. This is because it does not require any immediate values to be input and only needs to specify two registers, which can be done using R-type.

```
# $t1 is the counter
# $t0 stores the address
# Bits 7:0 are checked one by one to ensure they are >= 0
# If they are not, the program stops and returns
loop:
lw $at, 0($t0)          # load value into $at
sll $at, $at, 24        # doing this to only keep last 8 bits
srl $at, $at 24         # now only rightmost 8 bits [Bits 7:0]
slti $t2, $at, 0        # if value in $at < 0, then $t0 set to 1
beq $t2, 1, exit        # if $t2 is 1, that means $at < 0, therefore exit
addiu $t0, $t0, 1       # R[rt] = R[rt] + 1
addiu $t1, $t1, 1       # R[rd] = R[rd] + 1
j loop
```

b. begin: starts at address 0x00400abc
= 4,197,052 (Decimal) / 2^2 (Because last 2 bits are 0)
= 1,049,263 (Address in)

nor – R type instruction
opcode: 0, rs: 16, rt: 4, rd: 8, shamt: 0, function: 39
binary encoding: 0000 0010 0000 0100 0100 0000 0010 0111
hex encoding: 0x02044027

addiu – I type instruction
opcode: 9, rs: 0, rt: 9, immediate: 127
binary encoding: 0010 0100 0000 1001 0000 0000 0111 1111
hex encoding: 0x2409007F

lw – I type instruction
opcode: 35, rs: 8, rt: 10, immediate: 64
binary encoding: 1000 1101 0000 1010 0000 0000 0100 0000
hex encoding: 0x8D0A0040

sltu – R type instruction
opcode: 0, rs: 10, rt: 9, rd: 11, shamt: 0, function: 43
binary encoding: 0000 0001 0100 1001 0101 1000 0010 1011
hex encoding: 0x0149582B

beq – I type instruction
opcode: 4, rs: 0, rt: 11, immediate: -3
binary encoding: 0001 0000 0000 1011 1111 1111 1111 1101
hex encoding: 0x100BFFFD
j – J type instruction
opcode: 2, address: 1,049,263
binary encoding: 0000 1000 0001 0000 0000 0010 1010 1111
hex encoding: 0x081002AF

c. **Two-instruction sequence using lui:**
lui $t0, 0xFEED
ori $t0, $t0, 0x2050
Machine code (lui):
lui – I type instruction
opcode: 15, rs: 8, rt: 0, immediate: 65,261
binary encoding: 0011 1100 0000 1000 1111 1110 1110 1101
hex encoding: 0x3C08FEED
Machine code (ori):
ori – I type instruction
opcode: 13, rs: 8, rt: 8, immediate: 8,272
binary encoding: 0011 0100 0000 0000 1000 0000 0101 0000
hex encoding: 0x34008090

Three-instruction sequence if lui is not supported:
addiu $t0, $zero, 0xFEED
sll $t0, $t0, 4
addiu $t0, $t0, 0x2050
Machine code (addiu):
addiu – I type instruction
opcode: 9, rs: 0, rt: 8, immediate: 65,261
binary encoding: 0010 0100 0000 1000 1111 1110 1110 1101
hex encoding: 0x2408FEED
Machine code (sll):
sll – R type instruction
opcode: 0, rs: 0, rt: 8, rd: 8, shamt: 16, function: 0
binary encoding: 0000 0000 0000 1000 0100 0100 0000 0000
hex encoding: 0x00084400
Machine code (addiu):

addiu – I type instruction
opcode: 9, rs: 8, rt: 8, immediate: 8,272
binary encoding: 0010 0101 0000 1000 0010 0000 0101 0000
hex encoding: 0x25082050

2.

a. Submitted file
b. Submitted file
c. For 2.a. if N = 5, there are 13 base, 24 (6 instructions in loop * 4), and 6 exit instructions = 43 Total Instructions
For 2.b. if N = 4