Tuan Nguen

<u>Question 1:</u>

a) i) If r0 is zero:
```
cmp r0, #0
beq skip
```

ii) If f0 is not zero
```
cmp r0, #0
bne skip
```

iii) If r0 is odd, we can check the least significant bit
```
movs r3, #1
ands r3, r3, r0
cmp r3, #1
beq skip
```

iv) If r0 is even, then the least significant bit is going to be 0.
```
movs r3, #1
ands r3, r3, r0
cmp r3, #0
beq skip
```

b)
```
account(a: Array[Int], n: Int)
    num = 0
    for i = 0 until n
        num = num + count(a(i))
    return num
```

c)
```
account:
    push {r4, lr}
    movs r0, #0 // r0 = num
    ldr r3, =row // r3 is the base address of the array
    movs r2, #0 // r2 = i
again:
    ldr r4, [r3, r2]
    count r4, r4
    adds r0, r0, r4
    adds r2, r2, #4 // increment i
    cmp r2, r1 // assume r1 has the length of the array
    bne again
    pop {r4, pc}
    bx lr
```

d)
0x358 in binary is 0011 0101 1000.
0x357 in binary is 0011 0101 0111.
The AND of those two numbers is 0011 0101 0000. The number of set bits in the AND of x and x-1 is the same as the number of set bits in x, right until when a digit changes in x-1. The rest of the AND result is going to 0, because the bits in x and x-1 are either going to be 0 and 1, or 1 and 0.

e)
```
count(x: Int)
    num = 0 // number of 1s
    x1 = x
    while(x1 != 0)
        if(x1 & 1) num = num + 1
        x1 = x1 / 2
    return num

count:
    movs r1, #0 // r1 = num
again:
    cmp r0, 0 // assume r0 = x
    beq done
    movs r2, #1
    ands r2, r2, r0
    lsrs r0, r0, #1
    blo again // Check the shifted bit
    adds r1, r1, #1
    b again
done:
    movs r0, r1
    bx lr
```

Question 2:
a)
```
    mov r0, #0          @ i = 0
    mov r1, #0          @ m = 0
    b test
loop:
    ldr r2, =a          @ r2 will have the base address
    ldr r3, [r2, r0]
    adds r0, r0, #4
    cmp r3, r1
    bls test            @ if lower or same go to test
    mov r1, r3
test:
    cmp r0, #N
    blt loop
```

b)
If the address of a[i] is kept in a register, then we would not need to store i(r0 in the subroutine). We would not need to load the address of the base of the array every time we loop. After each loop we can just increment the address by 4, because each element in the array occupies 4 bytes of memory.

Question 3:
a) The n-type transistor has the source and drain at both ends. If the source is connected to a positive voltage, electricity would flow through the transistor. The p-type transistor is the opposite of a n-type transistor. If a p-type is connected to ground, then current flows through it, otherwise no electricity flow through it.

b) Suppose the p-type transistors are p1 and p2, and the n-type transistors are n1 and n2.
When both a and b are 0, the desired behavior is z to be 1. In this case, current will flow through both p1 and p2, making the output 1.
When only one of the input is 1, then current will flow through exactly one of the p-type transistors – either p1 or p2. In both cases, z will be connected to Vdd, so z will be 1.
When both inputs are 1, then both n1 and n2 will conduct electricity and none of the p-type transistors will have any current through them. In this case, z will be connect to ground, so z will be 0. *See pdf for diagram*

c) $z = \neg((a \wedge b) \vee (c \wedge d))$ can be simplified using De Morgan's Law.
$\neg(a \ \vee \ \ b) = \neg a \wedge \neg b$ => $z = \neg(a \wedge b) \wedge \neg(c \wedge d)$
See pdf for diagram

d) *See pdf for diagram*

Question 4:
See pdf for diagram

Question 3:
See pdf for diagram