```
[ 3 ]  0000 IJKL 0000 0000 0000 0000 0000 0000

[ 4 ]  aaaa aaaa aaaa aaab cdef ghij klmn pq00

[ 2 ]  0000 0000 0000 0000 0000 0000 abcd efgh

[ 5 ]  0000 0000 0000 0000 abcd efgh ijkl mnpq

[   ]  q000 0000 0000 0000 0000 0000 0000 000A
```

(a)

```
(1)  sll   $v1, $a0, 31
     srl   $t0, $a0, 31
     and   $v1, $t0, $v1

(2)  lui   $t0, 0xFF00
     ori   $t0, $t0, 0xFF00
     and   $v1, $a0, $t0
     srl   $t1, $v1, 8
     andi  $v1, $t1, 0xFF

(3)  sll   $v1, $a0, 8
     srl   $v1, $v1, 28
     sll   $v1, $v1, 24

(4)  sll   $v1, $a0, 16
     sra   $v1, $v1, 14

(5)  sll   $t0, $a0, 24
     srl   $t0, $t0, 24
     andi  $t1, $a0, 0xFF00
     or    $v1, $t0, $t1
```

(b)

Figure 1: Outcomes in `$v1` and Assembly Code Segments

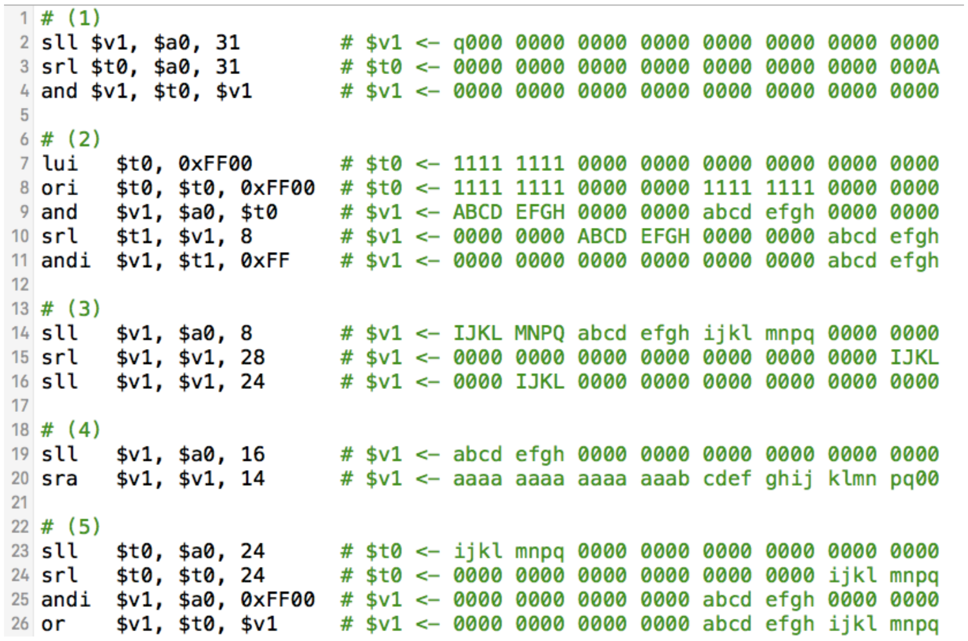
Question 1 (10 points): For this question assume that initially register `$a0` has the following binary number:

```
$a0 = ABCD EFGH IJKL MNPQ abcd efgh ijkl mnpq
```

Where each letter represents a bit that could have either the value `0` or the value `1`. We used both upper and lower case letters because we needed 32 distinct symbols, but there is no special meaning to the fact that the bit in a given position is represented by an upper or lower case letter. For instance, the value of the bit at position `A` is independent of the value of the bit at position `a`. Also, we avoided using the letters `O` and `o` so that they are not confused with the number `0`.

Figure 1(b) contains five separate segments of MIPS assembly code. Each code segment has an associated number from `(1)` to `(5)`. Figure 1(a) has values that are produced in `$v1`. You have to determine which code segments from Figure 1(b) produce the value of Figure 1(a) in `$v1`. Beware that some values may not be produced by any of the code segments and some code segments may not produce any of the values shown in Figure 1(a). Indicate the correspondence between the code segments and the resulting values by writing the numbers associated with the code segments into the brackets provided to the left of the values. Leave blank brackets for the cases where there is no code segment that produces that value.

Figure 2 shows detailed comments on the execution of the code segments for this question.



```
# (1)
sll $v1, $a0, 31        # $v1 <- q000 0000 0000 0000 0000 0000 0000 0000
srl $t0, $a0, 31        # $t0 <- 0000 0000 0000 0000 0000 0000 0000 000A
and $v1, $t0, $v1       # $v1 <- 0000 0000 0000 0000 0000 0000 0000 0000

# (2)
lui   $t0, 0xFF00       # $t0 <- 1111 1111 0000 0000 0000 0000 0000 0000
ori   $t0, $t0, 0xFF00  # $t0 <- 1111 1111 0000 0000 1111 1111 0000 0000
and   $v1, $a0, $t0     # $v1 <- ABCD EFGH 0000 0000 abcd efgh 0000 0000
srl   $t1, $v1, 8       # $v1 <- 0000 0000 ABCD EFGH 0000 0000 abcd efgh
andi  $v1, $t1, 0xFF    # $v1 <- 0000 0000 0000 0000 0000 0000 abcd efgh

# (3)
sll   $v1, $a0, 8       # $v1 <- IJKL MNPQ abcd efgh ijkl mnpq 0000 0000
srl   $v1, $v1, 28      # $v1 <- 0000 0000 0000 0000 0000 0000 0000 IJKL
sll   $v1, $v1, 24      # $v1 <- 0000 IJKL 0000 0000 0000 0000 0000 0000

# (4)
sll   $v1, $a0, 16      # $v1 <- abcd efgh 0000 0000 0000 0000 0000 0000
sra   $v1, $v1, 14      # $v1 <- aaaa aaaa aaaa aaab cdef ghij klmn pq00

# (5)
sll   $t0, $a0, 24      # $t0 <- ijkl mnpq 0000 0000 0000 0000 0000 0000
srl   $t0, $t0, 24      # $t0 <- 0000 0000 0000 0000 0000 0000 ijkl mnpq
andi  $v1, $a0, 0xFF00  # $v1 <- 0000 0000 0000 0000 abcd efgh 0000 0000
or    $v1, $t0, $v1     # $v1 <- 0000 0000 0000 0000 abcd efgh ijkl mnpq
```

Figure 2: Detailed Comments of code for Question #1.