Alexander Miller
Homework 2

1i.

```
# if (bnsf - atsf > csx - kcs), store (bnsf-atsf) in cbq
        # get bnsf
        la      $t0, bnsf
        lw      $t1, 0($t0)             # t1 = bnsf
        sub     $t1, $t1, $s2           # t1 = bnsf - atsf
        sub     $t2, $s1, $s0           # t2 = csx - kcs
        slt     $t3, $t2, $t1           # t3 = csx - kcs < bnsf - atsf
        beq     $t3, $zero, Done        # if (csx - kcs >= bnsf - atsf), skip op
        la      $t4, cbq                # t4 = address of cbq
        sw      $t1, 0($t4)             # write bnsf - atsf to cbq
Done:
```

1j.

```
# if (kcs < erie && erie < epsw), then store value of kcs into epsw
        la      $t0, erie
        lw      $t1, 0($t0)             # t1 = erie
        la      $t0, epsw               # t0 = address of epsw
        lw      $t2, 0($t0)             # t2 = epsw
        slt     $t3, $s0, $t1           # t3 = kcs < erie
        slt     $t4, $t1, $t2           # t4 = erie < epsw
        beq     $t3, $zero, Done        # if kcs >= erie, skip op
        beq     $t4, $zero, Done        # if erie >= epsw, skip op
        sw      $s0, 0($t0)             # store kcs at address of epsw
```

2e.

```
# Mask: 0101_0101_0101_0101_0101_0101_0101_0101
        addi    $s1, $zero, 0x5555      # s1 = 1st half of mask
        sll     $s1, $s1, 16            # s1 = 2nd half of mask
        addi    $s1, $zero, 0x5555      # s1 = full mask
        and     $s1, $s0, $s1           # s1 = s0 & mask
```

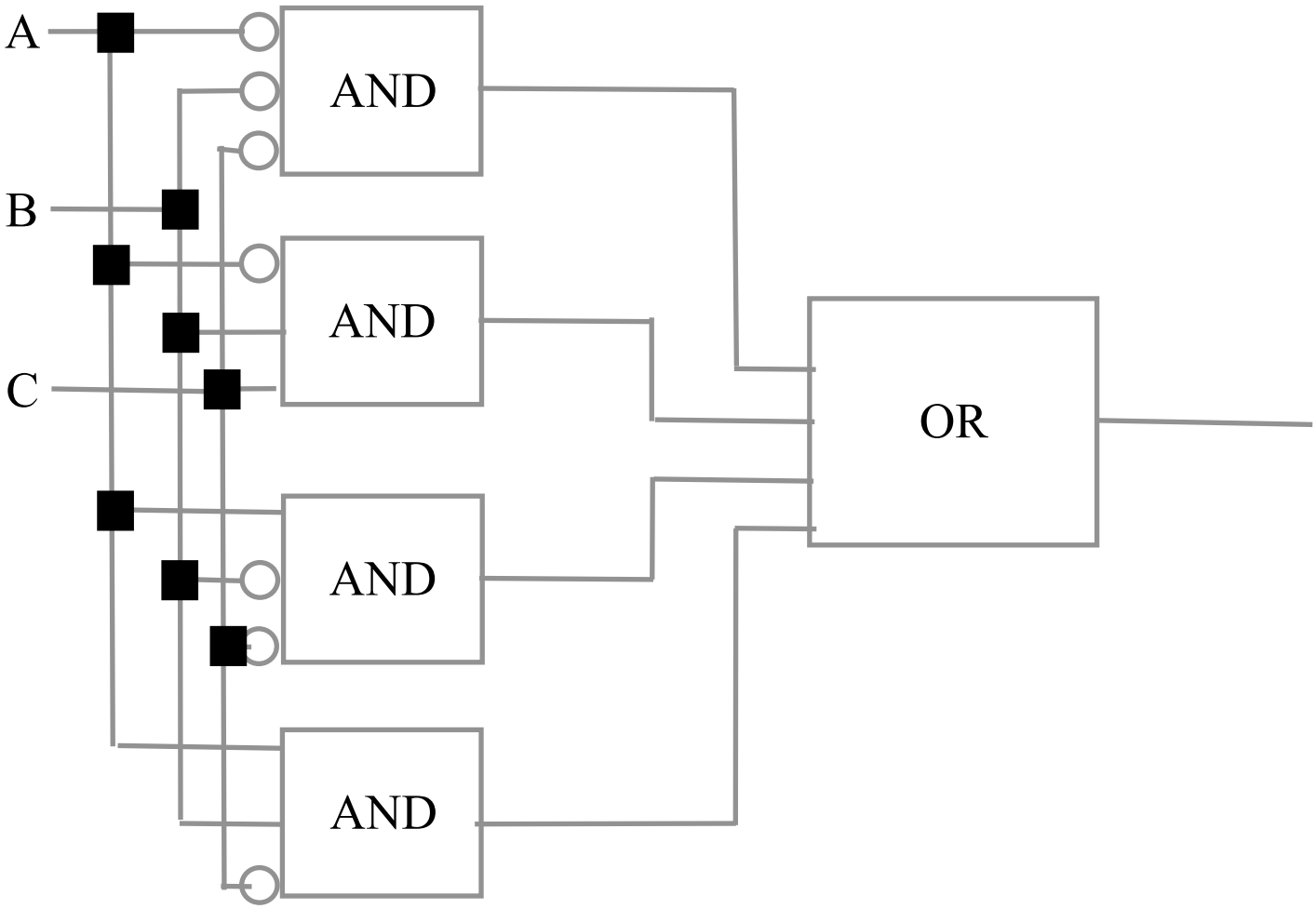
3d.

W: $A\overline{B}C + AB\overline{C} + ABC$

X: $\overline{ABC} + \overline{A}BC + A\overline{B}C + AB\overline{C}$

Y: $\overline{AB}C + \overline{A}B\overline{C} + ABC$

Z: $\overline{ABC} + \overline{A}BC + A\overline{B}\overline{C} + AB\overline{C}$



4d.

```
# int pow = … ; // s0 - this is set by previous code
# int prod = 1;          // allocate a register for this
# for (int i = 0; i<pow; i++) {
#         prod = prod*2;
# }
# REGISTERS:
# s0              pow
# s1              prod
# s2              i
        addi    $s1, $zero, 1               # s1 = prod = 1
        add     $s2, $zero, $zero           # s2 = i = 0
Loop:
        beq     $s2, $s0, Done              # skip loop if i == pow
        sll     $s1, $s1, 2                 # prod = prod * 2
        addi    $s2, $s2, 1                 # i++
        j       Loop                        # return to head of loop
Done:
```

5d.

Input:
a = 0100
b = 1001
aluOp = 1 (to perform OR)
bNegate = 0 (to perform OR)

Output:
4 AND bits: 0000
4 OR bits: 1101
4 ADD bits: 1101
4 Out Values: 1101 (since we selected OR)