Peter Pettino
CSE 3666
18 February, 2024

Homework 3

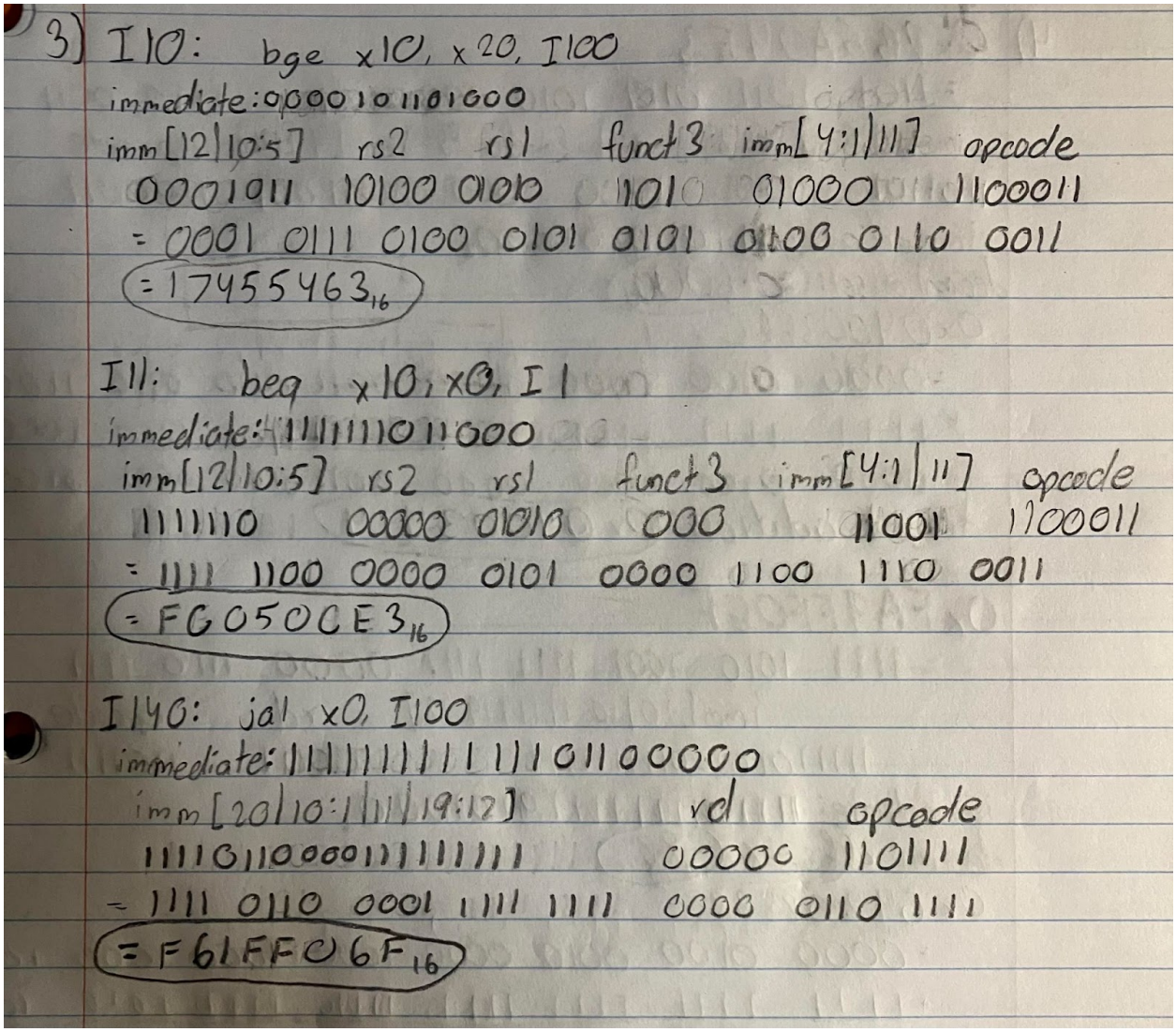
1)

```
# s1 = count
# s2 = i

foo:    addi    sp, sp, -20     # allocate 20 bytes of memory
        sw      ra, 0(sp)       # save return address to stack
        sw      a0, 4(sp)       # save a0 to stack
        sw      a1, 8(sp)       # save a1 to stack
        sw      s3, 12(sp)      # save &d[i] to stack
        sw      s2, 16(sp)      # save s2 to stack


        add     s1, x0, x0      # count = 0
        add     s2, x0, x0      # i = 0

loop:   beq     s2, a1, f_exit  # if (i == n) goto f_exit

        slli    t0, s2, 2       # t0 = s2 * 4
        add     s3, a0, t0
        add     a0, x0, s3      # &d[i]
        sub     a1, a1, s2      # n - i
        jal     ra, bar         # bar(&d[i], n-i)

        bge     x0, a0, skip    # if(0 >= t) goto skip
        addi    s1, s1, 1       # count += 1


skip:   addi    s2, s2, 1       # i += 1
        beq     x0, x0, loop    # goto loop

f_exit:
        lw      ra, 0(sp)       # load return address from stack
        lw      a0, 4(sp)       # load a0 from stack
        lw      a1, 8(sp)       # load a1 from stack
        lw      s3, 12(sp)      # load s3 from stack
        lw      s2, 16(sp)      # load s2 from stack
        addi    sp, sp, 20      # adjust stack pointer

        jalr    x0, ra, 0       # return

```

2)

```
msort:  addi    sp, sp, -12     # allocate 8 bytes of memory
        sw      ra, 8(sp)       # save return address to stack
        sw      s1, 4(sp)       # save n to stack
        sw      s0, 0(sp)       # save d to stack

        addi    sp, sp, -1028   # allocate 257 bytes
        sw      s3, 0(sp)       # save s3 to stack

        blt     x1, a1, skip    # if(1 < n) goto skip

        lw      ra, 8(sp)       # load ra from stack
        lw      s1, 4(sp)       # load n from stack
        lw      s0, 0(sp)       # load d from stack
        addi    sp, sp, 12      # adjust stack pointer

        lw      s3, 0(sp)       # load s3 from stack
        addi    sp, sp, 1028    # adjust stack pointer

        jalr    x0, ra, 0       # return

skip:   add     s0, x0, a0      # s0 = d
        add     s1, x0, a1      # s1 = n
        srli    s2, s1, 1       # n1 = n / 2
                                # s3 = c
        slli    s4, s2, 2       # &n1


        add     a0, x0, s0      # a0 = d
        add     a1, x0, s2      # a1 = n1
        jal     ra, msort       # msort(d, n1)

        add     a0, s0, s4      # a0 = &d[n1]
        sub     a1, s1, s2      # a1 = n - n1
        jal     ra, msort       # msort(&d[n1], n-n1)

        add     a0, x0, s3      # a0 = c
        add     a1, x0, s0      # a1 = d
        add     a2, x0, s2      # a2 = n1
        add     a3, s0, s4      # a3 = &d[n1]
        sub     a4, s1, s2      # a4 = n - n1
        jal     ra, merge       # merge(c, d, n1, &d[n1], n-n1)

        add     a0, x0, s0      # a0 = d
        add     a1, x0, s3      # a1 = c
        add     a2, x0, s1      # a2 = n
        jal     ra, copy        # copy(d, c, n)
```

3)



3) I10: bge x10, x20, I100

immediate: 0 0 0 0 1 0 1 1 0 1 0 0 0

imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode
0 0 0 1 0 1 1	1 0 1 0 0	0 1 0 1 0	1 1 0 1	0 1 0 0 0	1 1 0 0 0 1 1

= 0 0 0 1 0 1 1 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1 1 0 0 0 1 1

$= 17455463_{16}$

I11: beq x10, x0, I1

immediate: 1 1 1 1 1 1 1 0 1 1 0 0 0

imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode
1 1 1 1 1 1 0	0 0 0 0 0	0 1 0 1 0	0 0 0	1 1 0 0 1	1 1 0 0 0 1 1

= 1 1 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 0 1 1 1 0 0 0 1 1

$= FC050CE3_{16}$

I140: jal x0, I100

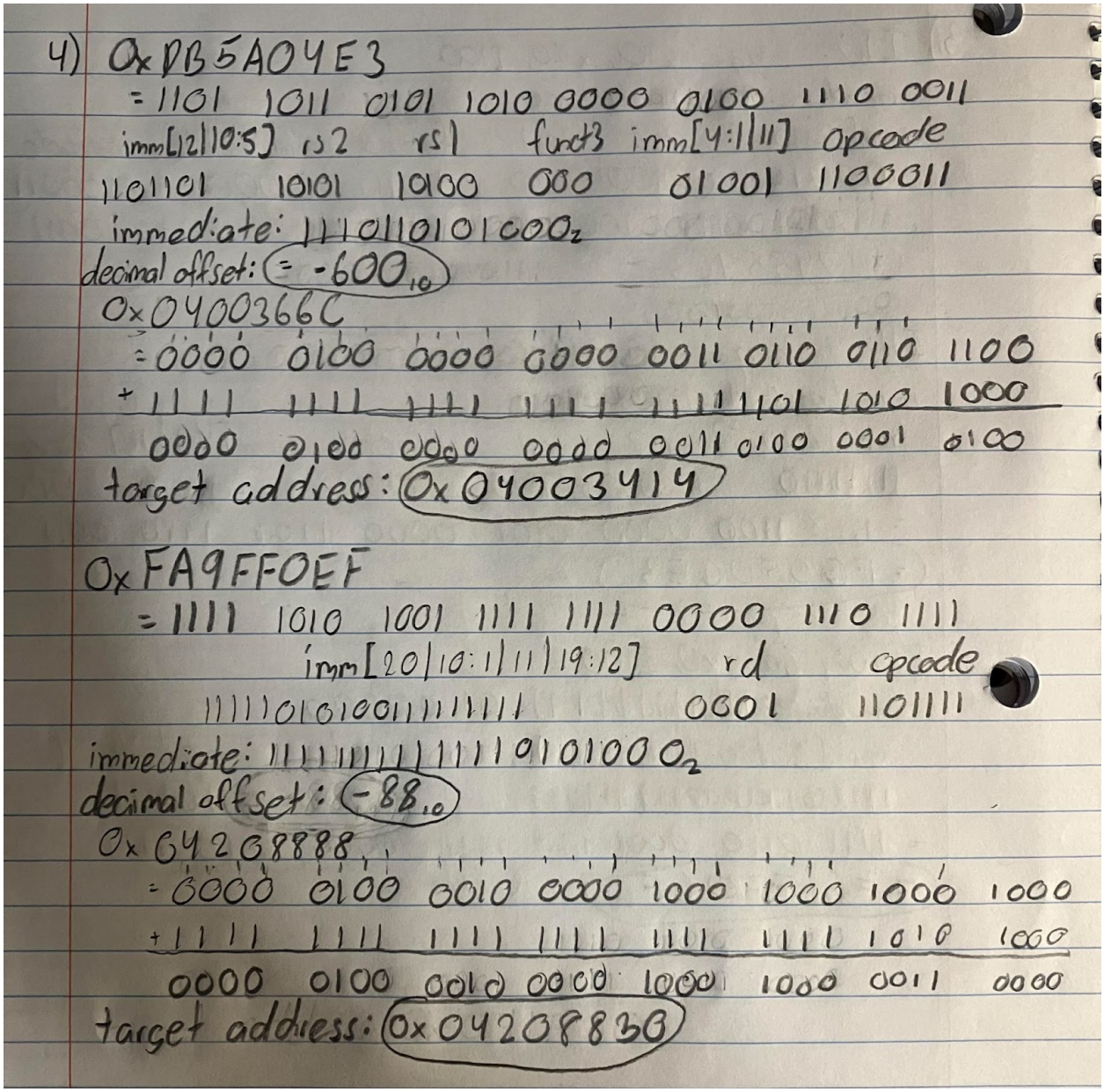
immediate: 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 0

imm[20|10:1|11|19:12]	rd	opcode
1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 1 1 1 1	0 0 0 0 0	1 1 0 1 1 1 1

= 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 0 1 1 1 1

$= F61FF06F_{16}$

4)


4) 0xDB5A04E3
= 1101 1011 0101 1010 0000 0100 1110 0011
imm[12|10:5] rs2 rs1 funct3 imm[4:1|11] opcode
1101101 10101 10100 000 01001 1100011
immediate: 1110110101002
decimal offset: = -600₁₀
0x0400366C
= 0000 0100 0000 0000 0011 0110 0110 1100
+ 1111 1111 1111 1111 1111 1101 1010 1000
0000 0100 0000 0000 0011 0100 0001 0100
target address: 0x04003414
0xFA9FF0EF
= 1111 1010 1001 1111 1111 0000 1110 1111
imm[20|10:1|11|19:12] rd opcode
11111010100111111111 00001 1101111
immediate: 11111111111110101000₂
decimal offset: -88₁₀
0x64208888
= 0000 0100 0010 0000 1000 1000 1000 1000
+ 1111 1111 1111 1111 1111 1111 1010 1000
0000 0100 0010 0000 1000 1000 0011 0000
target address: 0x04208830