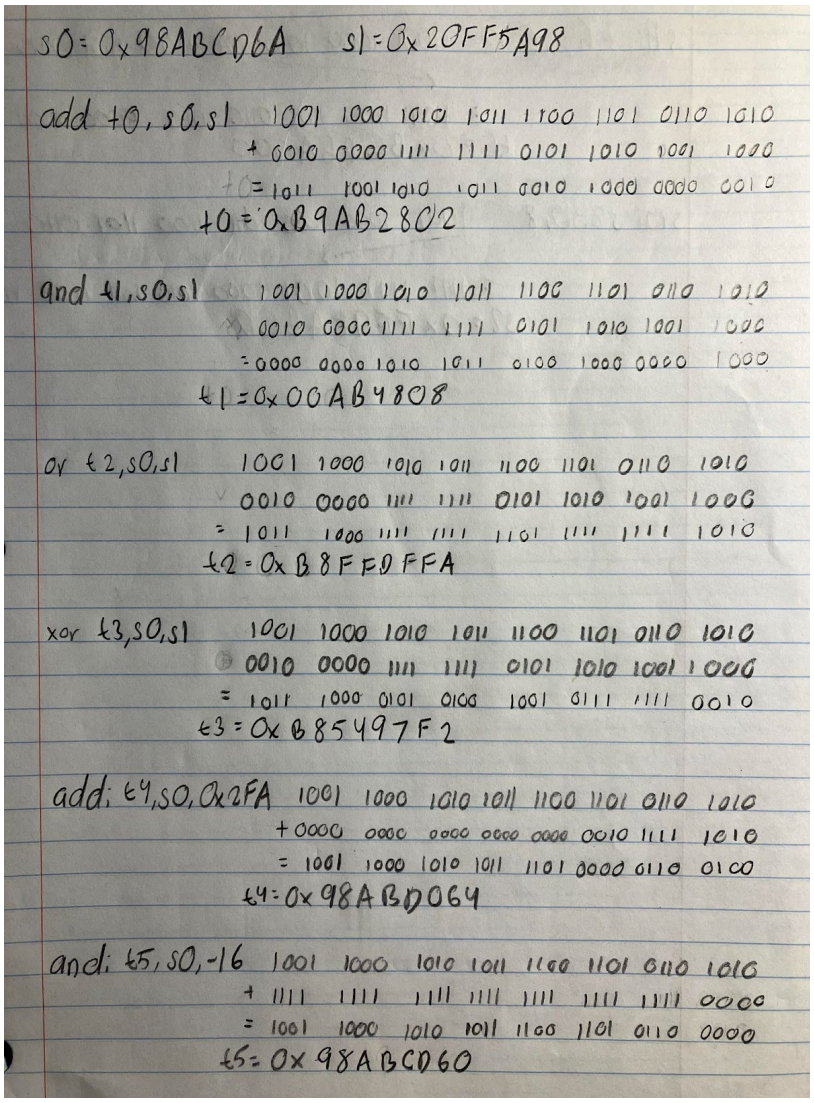
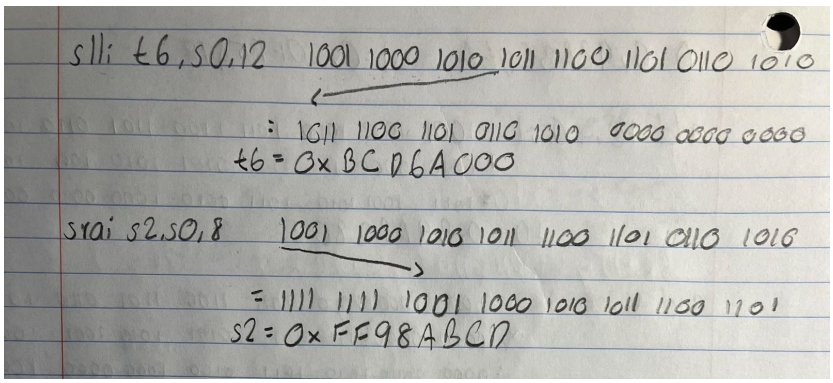
Peter Pettino
CSE 3666
3 February, 2024

Homework 1

1)





2)

```
        .globl  main

        .text
main:
        lui     s2, 0x12345     # load upper 20 bits into s2
        addi    s2, s2, 0x678   # add lower 12 bits to s2
        add     t0, x0, s2      # add the value from register s2 into t0

        addi    t2, x0, 24      # t2 = 24
        addi    t3, x0, 0       # t3 = 0

        addi    t4, x0, 0       # t4 = 0
        addi    t5, x0, 5       # t5 = 5

loop:   beq     t4, t5, exit

        srl     t1, t0, t2      # extract the appropriate byte
        andi    t1, t1, 0xFF    # clear the upper bits
        sll     t1, t1, t3      # shift the byte to the appropriate position
        or      s4, s4, t1      # combine with s4

        addi    t2, t2, -8      # update t2
        addi    t3, t3, 8       # update t3

        addi    t4, t4, 1       # t4 += 1
        beq     x0, x0, loop    # goto loop

        # exit
exit:   addi    a0, s4, 0       # print s4
        addi    a7, zero, 34
        ecall

        addi    a7, x0, 10
        ecall
```

The code extracts bytes from the value stored in t0 through the use of two counters, t2 and t3, in order to keep track of the byte positions during the byte extraction and shifting processes. By shifting and masking operations, it isolates each byte from the 32-bit value stored in t0. The loop iterates 4 times in order to properly extract and rearrange each byte of the value stored in t0 and stores it into s4 before the end of every iteration.

3)

a) If s0 is 0xFF00FF00 the number of instructions that are executed is 146. The number of executed instructions does depend on the number of 1's in s0, however it does not depend on the location of 1's. The number of instructions will always be:
$number\ of\ instructions = 2 + (4 * number\ of\ bits) + (number\ of\ 1's)$
Therefore, for 0xFF00FF00, converting to binary:
$11111111000000001111111100000000_2$, using the equation:
$number\ of\ instructions = 2 + (4 * 32) + 16$, thus the number of instructions is 146.

b)

```
6  main:   addi    s1, x0, 0       # s1 = 0
7          addi    t1, x0, 31      # t1 = 31
8
9  loop:   srl     t0, s0, t1      # extract the most significant bit
10         andi    t0, t0, 1       # AND the most significant bit and 1
11         beq     t0, x0, skip    # if 0 then goto skip
12         addi    s1, s1, 1       # s1 += 1
13
14 skip:   addi    t1, t1, -1      # t1 -= 1
15         bge     t1, x0, loop    # if(t1>=0) goto loop
```

The number of instructions when s0 = 0xFF00FF00 is 178. This is because 6 instructions are run when the bit is 1, and 5 instructions are run when the bit is 0. Since there are 16 bits of both 1's and 0's, $5 * 16 + 6 * 16 + 2 = 178$.

4)

```
1  # s1 = a
2  # s2 = i
3  # s3 = r
4
5  loop:   beq     s2, s1, exit    # if(i == a) goto exit
6
7          andi    s4, s2, 0xA5    # s4 = (s2 & 0xA5)
8          beq     s4, x0, else    # if(s4 == x0) goto else
9          slli    s5, s2, 8       # s5 = i << 8
10         xor     s3, s3, s5      # r ^= s5
11
12         addi    s2, s2, 1       # i += 1
13         beq     x0, x0, loop    # goto loop
14
15 else:   srli    s5, s2, 4       # s5 i >> 4
16         add     s3, s3, s5      # i += s5
17
18         addi    s2, s2, 1       # i += 1
19         beq     x0, x0, loop    # goto loop
20
21 exit:
```

5)

```
1
2           .globl  main
3
4           .text
5   main:
6           # use system call 5 to read integer
7           addi    a7, x0, 5
8           ecall
9           addi    s1, a0, 0       # copy to s1
10
11          addi    t0, x0, 1       # t0 = 1
12          addi    s2, x0, 0       # s2 = 0
13
14
15  loop:   beq     s1, t0, exit    # if(s1 == 1) goto exit
16
17          andi    t1, s1, 1       # t1 = (s1 && 1)
18          beq     t1, x0, even    # if(t1 == 0) goto even
19
20          add     t2, x0, s1      # temporarily store s1 to t2
21          slli    s1, s1, 1       # multiply by two (s1 = s1 << 1)
22          add     s1, s1, t2      # s1 += t2
23          addi    s1, s1, 1       # s1 += 1
24
25          addi    s2, s2, 1       # s2 += 1
26          beq     x0, x0, loop    # goto loop
27
28  even:   srli    s1, s1, 1       # divide by two (s1 = s1 >> 1)
29          addi    s2, s2, 1       # s2 += 1
30          beq     x0, x0, loop    # goto loop
31
32
33          # exit
34  exit:   add     a0, x0, s2      # print s2
35          addi    a7, x0, 1
36          ecall
37
38          addi    a7, x0, 10
39          ecall
```