

Peter Pettino
CSE 3666
3 February, 2024

Homework 1

1)

$s0 = 0x98ABCD6A \quad s1 = 0x20FF5A98$

add $t0, s0, s1$

$$\begin{array}{r} 1001\ 1000\ 1010\ 1011\ 1100\ 1101\ 0110\ 1010 \\ + 0010\ 0000\ 1111\ 1111\ 0101\ 1010\ 1001\ 1000 \\ \hline t0 = 1011\ 1001\ 1010\ 1011\ 0010\ 1000\ 0000\ 0010 \end{array}$$
$$t0 = 0xB9AB2802$$

and $t1, s0, s1$

$$\begin{array}{r} 1001\ 1000\ 1010\ 1011\ 1100\ 1101\ 0110\ 1010 \\ \times 0010\ 0000\ 1111\ 1111\ 0101\ 1010\ 1001\ 1000 \\ \hline t1 = 0000\ 0000\ 1010\ 1011\ 0100\ 1000\ 0000\ 1000 \end{array}$$
$$t1 = 0x00AB4808$$

or $t2, s0, s1$

$$\begin{array}{r} 1001\ 1000\ 1010\ 1011\ 1100\ 1101\ 0110\ 1010 \\ \vee 0010\ 0000\ 1111\ 1111\ 0101\ 1010\ 1001\ 1000 \\ \hline t2 = 1011\ 1000\ 1111\ 1111\ 1101\ 1111\ 1111\ 1010 \end{array}$$
$$t2 = 0xB8FFDFFA$$

xor $t3, s0, s1$

$$\begin{array}{r} 1001\ 1000\ 1010\ 1011\ 1100\ 1101\ 0110\ 1010 \\ \oplus 0010\ 0000\ 1111\ 1111\ 0101\ 1010\ 1001\ 1000 \\ \hline t3 = 1011\ 1000\ 0101\ 0100\ 1001\ 0111\ 1111\ 0010 \end{array}$$
$$t3 = 0xB85497F2$$

add: $t4, s0, 0x2FA$

$$\begin{array}{r} 1001\ 1000\ 1010\ 1011\ 1100\ 1101\ 0110\ 1010 \\ + 0000\ 0000\ 0000\ 0000\ 0010\ 1111\ 1010 \\ \hline t4 = 1001\ 1000\ 1010\ 1011\ 1101\ 0000\ 0110\ 0100 \end{array}$$
$$t4 = 0x98ABD064$$

andi: $t5, s0, -16$

$$\begin{array}{r} 1001\ 1000\ 1010\ 1011\ 1100\ 1101\ 0110\ 1010 \\ + 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 0000 \\ \hline t5 = 1001\ 1000\ 1010\ 1011\ 1100\ 1101\ 0110\ 0000 \end{array}$$
$$t5 = 0x98ABCD60$$

$s1 \leftarrow t6, s0, 12$

$$\begin{array}{r} 1001\ 1000\ 1010\ 1011\ 1100\ 1101\ 0110\ 1010 \\ \leftarrow \\ : 1011\ 1100\ 1101\ 0110\ 1010\ 0000\ 0000\ 0000 \\ t6 = 0xBCD6A000 \end{array}$$

srai $s2, s0, 8$

$$\begin{array}{r} 1001\ 1000\ 1010\ 1011\ 1100\ 1101\ 0110\ 1010 \\ \searrow \\ = 1111\ 1111\ 1001\ 1000\ 1010\ 1011\ 1100\ 1101 \\ s2 = 0xFF98ABCD \end{array}$$

2)

```
1 |     .globl  main
2 |
3 |     .text
4 |main:
5 |     lui    s2, 0x12345      # load upper 20 bits into s2
6 |     addi   s2, s2, 0x678    # add lower 12 bits to s2
7 |     add    t0, x0, s2       # add the value from register s2 into t0
8 |
9 |     addi   t2, x0, 24      # t2 = 24
10 |    addi   t3, x0, 0        # t3 = 0
11 |
12 |    addi   t4, x0, 0        # t4 = 0
13 |    addi   t5, x0, 5        # t5 = 5
14 |
15 |loop:  beq    t4, t5, exit
16 |
17 |    srl    t1, t0, t2      # extract the appropriate byte
18 |    andi   t1, t1, 0xFF     # clear the upper bits
19 |    sll    t1, t1, t3       # shift the byte to the appropriate position
20 |    or     s4, s4, t1       # combine with s4
21 |
22 |    addi   t2, t2, -8       # update t2
23 |    addi   t3, t3, 8        # update t3
24 |
25 |    addi   t4, t4, 1        # t4 += 1
26 |    beq    x0, x0, loop     # goto loop
27 |
28 |# exit
29 |exit:  addi   a0, s4, 0      # print s4
30 |       addi   a7, zero, 34
31 |       ecall
32 |
33 |       addi   a7, x0, 10
34 |       ecall
35 |
```

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:

$$\text{number of instructions} = 2 + (4 * \text{number of bits}) + (\text{number of 1's})$$

Therefore, for 0xFF00FF00, converting to binary:

111111100000000111111100000000₂, using the equation:

$$\text{number of instructions} = 2 + (4 * 32) + 16, \text{ thus the number of instructions is 146.}$$

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      .globl  main
2
3      .text
4
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
```