CSCI 206 Problem Set 2

Yifan Ge

Exercise 2.6

2.6.1

a. f = -g + h + B[1]; (f: $s0; g: $s1; h: $s2; i: $s3; j: $s4; A[0]: $s6; B[0]: $s7)

MIPS code:

```
sub   $t0, $zero, $s1
add   $t0, $t0, $s2
lw    $t1, 4($s7)
add   $s0, $t0, $t1
```

b. f = A[B[g]+1];

MIPS code:

```
add   $t0, $s7, $s1
lw    $t1, 0($t0)
addi  $t1, $t1, 1
add   $t0, $t1, $s6
lw    $s0, 0($t0)
```

2.6.2

a. 4

b. 5

2.6.3

a. $s1, $s2, $s0, $s7, $t0, $t1, ➔ 6

b. $s0, $s1, $s6, $s7, $t0, $t1, ➔ 6

2.6.4

a.
```
add   $s0, $s0, $s1   ➔   f = f + g
add   $s0, $s3, $s2   ➔   f = h + i
add   $s0, $s0, $s3   ➔   f = f + i
```

b.
```
addi  $s6, $s6, -20   ➔   &A[0] = &A[0] - 20
add   $s6, $s6, $s1   ➔   &A[0] = &A[0] + g
lw    $s0, 8($s6)     ➔   f = A[2]
```

2.6.5

a. $s0 = $s3 + $s2 + $s3 = 110

b. &A[0] = 256 – 20 + 20 = 256

$s0 = 200

2.6.6

a.
```
add   $s0, $s0, $s1   ➔   R; op: 0; rs: $s0; rt: $s1; rd: $s0
add   $s0, $s3, $s2   ➔   R; op: 0; rs: $s3; rt: $s2; rd: $s0
add   $s0, $s0, $s3   ➔   R; op: 0; rs: $s0; rt: $s3; rd: $s0
```

b.
```
addi  $s6, $s6, -20   ➔   I; op: 8; rs: $s6; rt: 0; immediate: -20
add   $s6, $s6, $s1   ➔   R; op: 0; rs: $s6; rt: $s1; rd: $s6
lw    $s0, 8($s6)     ➔   I; op: 23; rs: $s0; rt: $s6; immediate: 8
```

Exercise 2.13

2.13.1

a. $t2 = 0x55555550 | 0x12345678 = 0x57755678
b. $t2 = 0xEADFEED0 | 0xDEADFADE = 0xFEEDFADE

2.13.2

a. $t2 = 0x55555550 – 1 = 0x5554FFFE
b. $t2 = 0xEADFEED0 – 1 = 0xFEECFFFE

2.13.3

a. $t2 = 0x0AAAAAAA + 0xFFEF = 0x0AABAA99
b. $t2 = 0x17D5BFDD + 0xFFEF = 0x17D6DFCC

2.13.4

a. $t2 = 0x00014B4A | 0x00005A5A = 0x0001EFEF
b. $t2 = 0x000052D2 + 0x00F0 = 0x000053C2

2.13.5

a. $t2 = 0x4B4A0000 | 0xA5A50000 = 0xEFEF0000
b. $t2 = 0x52D28000 + 0xA5A50000 = 0xF8778000

2.13.6

a. $t2 = 0x4B4BFFFE | 0xA5A5FFFF = 0xEFEFFFFE
b. $t2 = 0x52D2FFFF + 0xA5A5FFFF = 0xF878FFFE

Exercise 2.14

2.14.1

a.
```
sll   $t0, $t0, 9
srl   $t1, $t0, 14
```
b.
```
sll   $t1, $t0, 9
```

2.14.2

a.
```
sll   $t0, $t0, 27
srl   $t1, $t0, 27
```
b.
```
sll   $t0, $t0, 27
srl   $t1, $t0, 12
```

2.14.3

a.
```
sll   $t1, $t0, 28
```
b.
```
sll   $t0, $t0, 28
srl   $t1, $t0, 14
```

2.14.4

a.
```
addi  $t2, $zero, 0x7F
nor   $t2, $t2, $zero
sll   $t0, $t0, 14
srl   $t0, $t0, 25
and   $t1, $t1, $t2
and   $t1, $t1, $t0
```

b.
```
sll   $t0, $t0, 14
srl   $t0, $t0, 25
sll   $t0, $t0, 14
addi  $t2, $zero, 0x7F
sll   $t2, $t2, 14
nor   $t2, $t2, $zero
and   $t1, $t1, $t2
and   $t1, $t1, $t0
```

2.14.5

a.
```
sll   $t0, $t0, 26
srl   $t0, $t0, 26
addi  $t2, $zero, 0x2F
nor   $t2, $t2, $zero
and   $t1, $t1, $t2
and   $t1, $t1, $t0
```
b.
```
sll   $t0, $t0, 26
srl   $t0, $t0, 12
addi  $t2, $zero, 0x2F
sll   $t2, $t2, 14
nor   $t2, $t2, $zero
and   $t1, $t1, $t2
and   $t1, $t1, $t0
```

2.14.6

a.
```
srl   $t0, $t0, 29
addi  $t2, $zero, 0x7
nor   $t2, $t2, $zero
and   $t1, $t1, $t2
and   $t1, $t1, $t0
```
b.
```
srl   $t0, $t0, 29
sll   $t0, $t0, 14
addi  $t2, $zero, 0x7
nor   $t2, $t2, $zero
and   $t1, $t1, $t2
and   $t1, $t1, $t0
```

Exercise 2.17

2.17.1

These are pseudo instructions, which are coded by one or multiple MIPS instruction.

2.17.2

Branching.

2.17.3

a.
```
bge   $t3, $zero, 8
nor   $t3, $t3, $zero
addi  $t3, $t3, 1
```

```
        add     $t2, $zero, $t3
b.      add     $t0, $zero, $zero
        ble     $t2, $t3, 4
        addi    $t0, $t0, 1
        add     $t2, $zero, $t0
```

2.17.4

a. $s2 = 2 * 10 = 20

b. $s2 = 2 * 10 * 10 = 200

2.17.5

a.
```
while (i > 0){
        B = B + 2;
        i = i – 1;
}
```

b.
```
while (i > 0){
        temp = 10;
        while (temp > 0){
                B = B + 2;
                temp = temp -1;
        }
        i = i – 1;
}
```

2.17.6

a. n = N * 5 + 3 = 5N + 3

b. n = (3 * 11 + 3) * N = 36N

Exercise 2.27

2.27.1

```
a.      jr      $ra
b.      bne     $s1, $s2, Loop
```

2.27.2

a. R

b. I

2.27.3

a. Register Addressing can reach a very large range of addresses. However, it take an extra register to store the address.

b. PC-relative is more efficient. It doesn't need memory. But it can only reach a much limited range of addresses.

2.27.4

a. 0b 001111 10000 00000 0000000001110100 = 0x3E000064
0b 001101 10000 10000 0000000000101000 = 0x36100028

b. 0b 001000 01000 00000 0000000000000000 = 0x21000000
0b 100011 01001 00000 0100000000000000 = 0x8D208000

2.27.5

a. lw $s0, 100
   ori $s0, $s0, 40

b. add $t0, $zero, 0x00
   lw $t1, 0x40
   sll $t1, $t1, 8

2.27.6

One more.

Exercise 2.30.1

a. add $t1, $zero, $t2

b. addi $t0, $zero, small
   beq $t1, small, LOOP