Solutions for Exercise 2.4, 2.6, 2.7, 2.11, 2.14, 2.18, 2.23, 2.24, 2.26, 2.34, 2.40, 2.40, 2.41, and 2.42

Disclaimer:

1. The solution is just for your reference. They may contain some mistakes. DO TRY to solve the problems by yourself. Please also pay attentions to the course website for the updates.

2. Try not to use pseudoinstructions for any exercises that ask you to produce MIPS code. Your goal should be to learn the real MIPS instruction set, and if you are asked to count instructions, your count should reflect the actual instructions that will be executed and not the pseudoinstructions.

**2.4** B[g] = A[f] + A[1+f];

**2.6.1** temp = Array[0];
temp2 = Array[1];
Array[0] = Array[4];
Array[1] = temp;
Array[4] = Array[3];
Array[3] = temp2;

**2.6.2**
```
lw $t0, 0($s6)
lw $t1, 4($s6)
lw $t2, 16($s6)
sw $t2, 0($s6)
sw $t0, 4($s6)
lw $t0, 12($s6)
sw $t0, 16($s6)
sw $t1, 12($s6)
```

**2.7**

| **Little-Endian** | | **Big-Endian** | |
|---|---|---|---|
| **Address** | **Data** | **Address** | **Data** |

|  |  |  |  |
|---|---|---|---|
| **12** | **ab** | **12** | **12** |
| **8** | **cd** | **8** | **Ef** |
| **4** | **ef** | **4** | **Cd** |
| **0** | **12** | **0** | **Ab** |

**2.11**

|  | **type** | **opcode** | **rs** | **rt** | **rd** | **immed** |
|---|---|---|---|---|---|---|
| addi $t0, $s6, 4 | I-type | **8** | **22** | **8** |  | **4** |
| add $t1, $s6, $0 | R-type | **0** | **22** | **0** | **9** |  |
| sw $t1, 0($t0) | I-type | **43** | **8** | **9** |  | **0** |
| lw $t0, 0($t0) | I-type | **35** | **8** | **8** |  | **0** |
| add $s0, $t1, $t0 | R-type | **0** | **9** | **8** | **16** |  |

**2.14** r-type, add $s0, $s0, $s0

**2.18.1** opcode would be 8 bits, rs, rt, rd fields would be 7 bits each
**2.18.2** opcode would be 8 bits, rs and rt fields would be 7 bits each
**2.18.3** more registers → more bits per instruction → could increase code size
more registers → less register spills → less instructions
more instructions → more appropriate instruction → decrease code size
more instructions → larger opcodes → larger code size

**2.23** $t2 = 3

**2.24** jump: no, beq: no

**2.26.1** 20

**2.26.2** i = 10;

```
do {
B += 2;
i = i - 1;
} while ( i > 0)
```

**2.26.3** 5*N

**2.34** f: addi $sp,$sp,-12

```
sw $ra,8($sp)
sw $s1,4($sp)
sw $s0,0($sp)
move $s1,$a2
move $s0,$a3
jal func
move $a0,$v0
add $a1,$s0,$s1
jal func
lw $ra,8($sp)
lw $s1,4($sp)
lw $s0,0($sp)
addi $sp,$sp,12
        jr $ra
```

**2.40** No, jump can go up to 0×0FFFFFFC.

**2.41** No, range is 0×604 + 0×1FFFC = 0×0002 0600 to 0×604 – 0×20000 = 0×FFFE 0604.

**2.42** Yes, range is 0×1FFFF004 + 0×1FFFC = 0×2001F000 to 0×1FFFF004 - 0×20000 = 1FFDF004