Chapter 6: Architecture

Exercise 6.4 **Consider** memory storage of a 32-bit word stored at memory **word 15** in a byte-addressable memory.

(a) What is the **byte address** of memory word 15?

(b) What are the **byte addresses** that memory word 15 spans?

(c) **Draw** the number 0xFF223344 stored at word 15 in both big-endian and little-endian machines. Clearly label the byte address corresponding to each data byte value.

Exercise 6.5 **Explain** how the following program can be used to determine whether a computer is big-endian or little-endian:

```
li $t0, 0xABCD9876
sw $t0, 100($0)
lb $s5, 101($0)
```

Exercise 6.10 **Convert** the following MIPS assembly code into machine language.

Write the instructions in hexadecimal.

```
add $t0, $s0, $s1
lw $t0, 0x20($t7)
addi $s0, $0, -10
```

Exercise 6.12 **Consider** I-type instructions.

(a) Which instructions from Exercise 6.10 are **I-type instructions**?

(b) **Sign-extend** the 16-bit immediate of each instruction from **part (a)** so that it becomes a 32-bit number.

Exercise 6.16 The `nori` instruction is not part of the MIPS instruction set, because the same functionality can be implemented using existing instructions.

Write a short assembly code snippet that has the following functionality:

`$t0 = $t1 NOR 0xF234`.

Use as few instructions as possible.

Exercise 6.20 **Convert** the high-level function from Exercise 6.18 into MIPS assembly code.

```
int find42( int array[], int size)
{
  int i;     // index into array

  for (i = 0; i < size; i = i+1)
    if (array[i] == 42)
      return i;

  return -1;
}
```

Hint: `size` specifies the **number of elements in array**, and `array` specifies the **base address** of the array. The function **returns the index number** of the first array entry that holds the value 42. If no array entry is 42, the function **returns the value –1**.

Exercise 6.25 **Convert** the following `beq` assembly instructions into machine code. Instruction addresses are given to the left of each instruction.

(a)

```
0x00401000          beq $t0, $s1, Loop
0x00401004           . . .
0x00401008           . . .
0x0040100C   Loop:   . . .
```

(b)

```
0x00401000          beq $t7, $s4, done
 . . .              . . .
0x00402040   done:  . . .
```

(c)

```
0x0040310C   back:  . . .
 . . .              . . .
0x00405000          beq $t9, $s7, back
```