Computer Organization(H)
Southern University of Science and Technology
Mengxuan Wu
12212006

Theory Assignment 1

Mengxuan Wu

Problem 1

```
addi x5, x7, -5  // f = h - 5
add  x5, x5, x6  // f = f + g
```

Problem 2

```
slli x5, x28, 2   // x5 = i * 4
add  x5, x5,  x10 // x5 = x5 + A
lw   x6, 0(x5)    // x6 = A[i]

slli x5, x29, 2   // x5 = j * 4
add  x5, x5,  x10 // x5 = x5 + A
lw   x7, 0(x5)    // x7 = A[j]

add  x5, x6, x7   // x5 = x6 + x7
sw   x5, 32(x11)  // B[8] = x5
```

Problem 3

1)

```
// Array[] is a int array with 5 elements.
// The following code uses bubble sort to sort the array in ascending order.
for (int i = 0; i < 5; i++) {
        for (int j = 4; j > i; j--) {
                if (Array[j] < Array[j - 1]) {
                        int temp = Array[j];
                        Array[j] = Array[j - 1];
                        Array[j - 1] = temp;
                }
        }
}
```

2)

```
// The following code uses bubble sort to sort the array in ascending order.
// x22 is the base address of the array.
// x5 is the index i.
// x6 is the index j.
// x7, x28, x29 store temporary values.

li x5, 0 // i = 0
loop1:
        li x6, 4 // j = 4
        loop2:
                slli x7, x6, 2   // x7 = j * 4
                add  x7, x7, x22 // x7 = x7 + Array
                lw   x28, 0(x7)  // x28 = Array[j]
                lw   x29, -4(x7) // x29 = Array[j - 1]

                blt  x28, x29, swap // if (Array[j] < Array[j - 1]) goto swap
                j    end_swap       // else goto end_swap
        swap:
                sw   x28, -4(x7) // Array[j - 1] = x28
                sw   x29, 0(x7)  // Array[j] = x29
        end_swap:
                addi x6, x6, -1    // j = j - 1
                blt  x5, x6, loop2 // if (j > i) goto loop2
        addi x5, x5, 1 // i = i + 1
        slti x7, x5, 5 // if (i < 5) x7 = 1 else x7 = 0
        bne  x7, x0, loop1 // if (x7 != 0) goto loop1
```

Problem 4

Segmentation	funct7	rs2	rs1	funct3	rd	opcode
Content	0000000	00001	00001	000	00001	0110011
Meaning	ADD	x1	x1	ADD	x1	R-format

Assembly language instruction: `add x1, x1, x1`

Problem 5

Since `opcode` is `0x3`, this is an `I-format` instruction. Then we can arrange the binary code as follows:

Segmentation	imm[11:0]	rs1	funct3	rd	opcode
Content	000000000100	11011	010	00011	0000011
Meaning	4	x27	LW	x3	I-format

Assembly language instruction: `lw x3, 4(x27)`
Binary representation: `00000000_01001101_10100001_10000011`.