JOINT INSTITUTE
交大密西根学院

## Ve370 Introduction to Computer Organization

# Homework 2

1. Convert the following assembly code fragment into machine code, assuming the memory location of the first instruction is 0x10000400: (15 points)

```
LOOP:   slt  $t2, $zero, $t0
        bne  $t2, $zero, ELSE
        j    DONE
ELSE:   addi $s2, $s2, 2
        addiu $t0, $t0, 1
        j    LOOP
DONE:
```

   …

2. Following memory location has address 0x10000000 and content 0x15478933.

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0x10000000 | 15 | 47 | 89 | 33 |

   Write MIPS assembly instructions to load the byte 47 to register $s1, then show the content of $s1 after the operations. (5 points)

3. Exercise 2.19.1 (15 points)
4. Exercise 2.19.2 (10 points)
5. Exercise 2.19.3 (10 points)

## Exercise 2.19

For the following problems, the table holds C code functions. Assume that the first function listed in the table is called first. You will be asked to translate these C code routines into MIPS assembly.

**a.**

```
int fib(int n){
    if (n==0)
       return 0;
    else if (n == 1)
       return 1;
    else
       fib(n-1) + fib(n-2);
```

**b.**

```
int positive(int a, int b) {
    if (addit(a, b) > 0)
       return 1;
    else
       return 0;
    }
int addit(int a, int b) {
    return a+b;
    }
```

**2.19.1** [15] <2.8> Implement the C code in the table in MIPS assembly. What is the total number of MIPS instructions needed to execute the function?

**2.19.2** [5] <2.8> Functions can often be implemented by compilers "in-line." An in-line function is when the body of the function is copied into the program space, allowing the overhead of the function call to be eliminated. Implement an "in-line" version of the the C code in the table in MIPS assembly. What is the reduction in the total number of MIPS assembly instructions needed to complete the function? Assume that the C variable `n` is initialized to 5.

**2.19.3** [5] <2.8> For each function call, show the contents of the stack after the function call is made. Assume the stack pointer is originally at address 0x7ffffffc, and follow the register conventions as specified in Figure 2.11.

The following three problems in this Exercise refer to a function `f` that calls another function `func`. The code for C function `func` is already compiled in another module

6. Exercise 2.23.1 (15 points)

## Exercise 2.23

In this exercise, you will be asked to write an MIPS assembly program that converts strings into the number format as specified in the table.

| | |
|---|---|
| **a.** | positive and negative integer decimal strings |
| **b.** | positive hexadecimal integers |

**2.23.1** [10] <2.9> Write a program in MIPS assembly language to convert an ASCII number string with the conditions listed in the table above, to an integer. Your program should expect register `$a0` to hold the address of a null-terminated string containing some combination of the digits 0 through 9. Your program should compute the integer value equivalent to this string of digits, then place the number in register `$v0`. If a non-digit character appears anywhere in the string, your program should stop with the value –1 in register `$v0`. For example, if register `$a0` points to a sequence of three bytes $50_{ten}$, $52_{ten}$, $0_{ten}$ (the null-terminated string "24"), then when the program stops, register `$v0` should contain the value $24_{ten}$.

7. Exercise 2.24.3 (5 points)

## Exercise 2.24

Assume that the register `$t1` contains the address 0x1000 0000 and the register `$t2` contains the address 0x1000 0010. Note the MIPS architecture utilizes big-endian addressing.

| | |
|---|---|
| **a.** | `lbu $t0, 0($t1)`<br>`sw $t0, 0($t2)` |
| **b.** | `lb $t0, 0($t1)`<br>`sh $t0, 0($t2)` |

**2.24.3** [5] <2.9> Assume that the data (in hexadecimal) at address 0x1000 0000 is:

| | | | | |
|---|---|---|---|---|
| 1000 0000 | 11 | 00 | 00 | FF |

What value is stored at the address pointed to by register `$t2`? Assume that the memory location pointed to `$t2` is initialized to 0x5555 5555.

8. Exercise 2.26.1 (5 points)
9. Exercise 2.26.2 & 2.26.3 (5 points)

## Exercise 2.26

For this exercise, you will explore the range of branch and jump instructions in MIPS. For the following problems, use the hexadecimal data in the table below.

| | |
|---|---|
| **a.** | 0x00020000 |
| **b.** | 0xFFFFFF00 |

**2.26.1** [10] <2.6, 2.10> If the PC is at address 0x00000000, how many branch (no jump instructions) do you need to get to the address in the table above?

**2.26.2** [10] <2.6, 2.10> If the PC is at address 0x00000000, how many jump instructions (no jump register instructions or branch instructions) are required to get to the target address in the table above?

**2.26.3** [10] <2.6, 2.10> In order to reduce the size of MIPS programs, MIPS designers have decided to cut the immediate field of I-type instructions from 16 bits to 8 bits. If the PC is at address 0x0000000, how many branch instructions are needed to set the PC to the address in the table above?

10. Exercise 2.26.4 (10 points)

For the following problems, you will be using making modifications to the MIPS instruction set architecture.

| | |
|---|---|
| **a.** | 128 registers |
| **b.** | Four times as many different operations |

**2.26.4** [10] <2.6, 2.10> If the instruction set of the MIPS processor is modified, the instruction format must also be changed. For each of the suggested changes above, what is the impact on the range of addresses for a beq instruction? Assume that all instructions remain 32 bits long and any changes made to the instruction format of i-type instructions only increase/decrease the immediate field of the beq instruction.

11. Exercise 2.27.1 & 2.27.2 (5 points)

## Exercise 2.27

In the following problems, you will be exploring different addressing modes in the MIPS instruction set architecture. These different addressing modes are listed in the table below.

| | |
|---|---|
| **a.** | Base or Displacement Addressing |
| **b.** | Pseudodirect Addressing |

**2.27.1** [5] <2.10> In the table above are different addressing modes of the MIPS instruction set. Give an example MIPS instructios that shows the MIPS addressing mode.

**2.27.2** [5] <2.10> For the instructions in 2.27.1, what is the instruction format type used for the given instruction?