CDA 4205 Computer Architecture

Assignment 3: MIPS Assembly Procedure

1. (10 pts) Implement the following C code in MIPS assembly. Show the contents of the stack after the function call to the function “compare” is made. Assume that the stack pointer is originally at address 0x7FFFFFFc.

**int compare(int a, int b) {**

**if (sub(a, b) >= 0) return 1;**

**else return 0;**

**}**

**int sub(int a, int b) {**

**return a – b;**

**}**

**Compare:**

**addi $sp, $sp, -4 // allocates 4 bytes moves stack pointerup**

**sw $ra, 0($sp) //**

**jal sub // jumps to sub**

**bltz $v0, end // if (sub(a, b) >= 0 continu**

**li $t0, 1 //returns 1**

**j skip //jump to skip**

**end:**

**li $t0, 0 //returns 0**

**skip:**

**move $v0, $t0 //moves return to final resul**

**lw $ra, 0($sp) //loads word from stored add**

**addi $sp, $sp, 4 // frees allocated bytes**

**jr $ra //jumps back stored register**

**sub: sub $v0, $a0, $al**

**jr $ra //jumps back stored register**

**0x7FFFFFFc -4 = 0x7FFFFFF8**

1. (10 pts) Implement the following C code in the table in MIPS assembly. Suppose that fib\_iter was called with n = 4, show the contents of the stack after the function call to the function “fib\_iter” is made. Assume that the stack pointer is originally at address 0x7ffffffc.

**int fib\_iter(int a, int b, int n) {**

**if (n == 0) return b;**

**else return fib\_iter(a+b, a, n-1);**

**}**

**Fib\_iter:**

**Addi $a3, $a3, 0**

**bne $a2, $0, else // if(n == 0) continue**

**Move $v0, $al // return b**

**Beq $a3, $0, Woops**

**Lw $ra, 0($sp)**

**Addi $sp, $sp, 4**

**Woops:**

**Jr $ra //jumps back stored register**

**Else: addi $sp, $sp, -4 //moves stack pointer up 4**

**Sw $ra, 0($sp) //**

**Move $t0, $a0 // moves a away**

**Addu $a0, $a0, $a1 // a + b**

**Move $a1, $t0 // puts a into a1**

**Addi $a2, $a2, -1 // n-1**

**Jal fib\_iter //jumps to beginning**

**0x7FFFFFFc – 4 -4 -4 -4??? = 0x7FFFFFF0**

1. (15 points) The following problems refer to a function f that calls another function func. The function declaration for func is “int func(int a, int b);”. The code for function f is as follows:

**int f(int a, int b, int c) {**

**return func(func(a, b), c);**

**}**

1. Translate function f into MIPS assembly code, using the MIPS calling convention. If you need to use register $t0 through $t7, use the lower-numbered registers first.
2. Right before your function f of Problem 3 returns, what do you know about contents of registers $ra, and $sp? Keep in mind that we know what the entire function f looks like, but for function func we only know its declaration.

F:

Addi $sp, $sp, -4 // moves stack pointer up 4

Sw $ra, 0($sp) //

Jal func //jumps to func while storing the address thing

Move $a0, $v0 // replace a with func answer

Move $a1, $a2 // replace b with c

Jal func // second use of func compared with c

lw $ra, 0($sp)

addi $sp, $sp, 4

jr $ra

b. I’m not too sure at what the contents of the registers are… $ra should be return address of what called f()?

1. (15 points) The following problems refer to a function f that calls another function func. The function declaration for func is “int func(int a, int b);”. The code for function f is as follows:

**int f(int a, int b, int c) {**

**return func(a, b) + func(b, c);**

**}**

1. Translate function f into MIPS assembly code, using the MIPS calling convention. If you need to use register $t0 through $t7, use the lower-numbered registers first.
2. Right before your function f of Problem 4 returns, what do you know about contents of registers $ra, and $sp? Keep in mind that we know what the entire function f looks like, but for function func we only know its declaration.

F:

Addi $sp, $sp, -4 // moves stack pointer up 4

Sw $ra, 0($sp) //

Jal func //jumps to func while storing the address thing

Move $a4, $v0 // stores func answer 1

Move $a0, $a1 // replaces b with a

Move $a1, $a2 // replaces old b with c

Jal func // second use of func compared with c

Move $a5, $v0 // stores func answer 2

Add $v0, $a4, $a5 // returns answer

lw $ra, 0($sp)

addi $sp, $sp, 4

jr $ra

$ra should be return address of what called f()?

1. (20 points) Write a program in MIPS assembly language to **convert a positive integer decimal string 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 though 9. Your program should compute the integer value equivalent to this string of digits, then place the number in register $v0. If a nondigit character appears anywhere in the string, your program should stop with the value ‐1 in register $v0.

Convert:

Move $t0, $a0 //moves string to t0

Lb $t1, ($t0) //loads the digit

Li $v0, 0 // set 0 here so it doesn’t loop

Loop:

Li $t2, 0x30

Li $t3, 0x39

Blt $t1, $t2, nondigit // if($1 < 0) jump nondigit

Bgt $t1, $t3, nondigit // if($1 > 9) jump nondigit

sub $t1, $t1, $t2 // converts to int I think

Mul $v0, $v0, 10 // int \* 10 = int0 like 5 \* 10 = 50, which allows more ints 51 55 etc

Add $v0, $v0, $t1 // adds on character to $v0

Addiu $t0, $t0, 1 // moves character selection

Lb $t1, ($t0) // loads the digit

Bne $t1, $0, loop // branches if both not equal, so it’s saying when it ends string

Jr $ra

Nondigit:

Li $v0, -1 // returns -1

Jr $ra

1. (20 points) Repeat problem 5 with **convert a string of hexadecimal digits to an integer**.

Convert:

Move $t0, $a0 //moves string to t0

Lb $t1, ($t0) //loads the digit

Li $v0, 0 // set 0 here so it doesn’t loop

Loop:

Li $t2, 0x30 //0

Li $t3, 0x39 //9

Li$t4, 0x41 //a

Li $t5, 0x46 //f

Blt $t1, $t2, nondigit // if($1 < 0) jump nondigit

Bgt $t1, $t3, check // if($1 > 9) jump nondigit

sub $t1, $t1, $t2 // converts to int I think

J skip

Check:

Blt $t1, $t4, nondigit // if($1 < a) jump nondigit

Bgt $t1, $t5, nondigit // if ($1 > f) jump nondigit

Addiu $t1, $t1, -48 // converts shifts

Mul $v0, $v0, 10

Skip:

Mul $v0, $v0, 10 //allocates 2 slots if hex, and 1 slot of int

Add $v0, $v0, $t1

Addiu $t0, $t0, 1 // moves character selection

Lb $t1, ($t0) // loads the digit

Bne $t1, $0, loop // branches if both not equal, so it’s saying when it ends string

Jr $ra

Nondigit:

Li $v0, -1 // returns -1

Jr $ra

* **Submission Requirements**
* Your solutions must be in a single file with a file name yourname-hw1.
* If scanned from hand-written copies, then the writing must be legible, or loss of credits may occur.
* Only submissions via the link on Canvas where this description is downloaded are graded. Submissions to any other locations on Canvas will be ignored.
* Late submissions are accepted for a maximum of 3 late days with 20% assignment credit off as late penalization. Assignments submitted after 3 late days will not be accepted.