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;**

**}**

**Solution:**

**compare: addi $sp, $sp, –4 # allocate frame = 4 bytes**

**sw $ra, 0($sp) # save return address**

**jal sub # call sub**

**li $t0, 0 # result = 0**

**bltz $v0, exit # if sub(a,b)<0 goto exit**

**li $t0, 1 # result = 1**

**exit: move $v0, $t0 # $v0 = result**

**lw $ra, 0($sp) # restore return address**

**addi $sp, $sp, 4 # free stack frame**

**jr $ra # return to caller**

**sub: sub $v0, $a0, $a1 # result = a - b**

**jr $ra # return to caller**

**after calling function compare:**

**$sp = $sp – 4 = 0x7ffffff8**

**0x7ffffff8: return address of compare**

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);**

**}**

**Solution:**

**fib\_iter: bne $a2, $0, else # if (n != 0) goto else**

**move $v0, $a1 # result = b**

**jr $ra # return to caller**

**else: addiu $sp, $sp, –4 # allocate frame = 4 bytes**

**sw $ra, 0($sp) # save return address**

**move $t0, $a0**

**addu $a0, $a0, $a1 # $a0 = a+b**

**move $a1, $t0 # $a1 = a**

**addiu $a2, $a2, -1 # $a2 = n-1**

**jal fib\_iter # recursive call**

**lw $ra, 0($sp) # restore return address**

**addiu $sp, $sp, 4 # free stack frame**

**jr $ra # return to caller**

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.

**Solution:**

**f: addiu $sp, $sp, -8 # allocate frame = 8 bytes**

**sw $ra, 0($sp) # save return address**

**sw $a2, 4($sp) # save c**

**jal func # call func(a,b)**

**move $a0, $v0 # $a0 = result of func(a,b)**

**lw $a1, 4($sp) # $a1 = c**

**jal func # call func(func(a,b),c)**

**lw $ra, 0($sp) # restore return address**

**addiu $sp, $sp, 8 # free stack frame**

**jr $ra # return to caller**

Register $ra is equal to the return address in the caller function, registers $sp has the same value they had when function f was called,

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 in the over part a) 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: addiu $sp, $sp, -12 # allocate frame = 12 bytes**

**sw $ra, 0($sp) # save return address**

**sw $a1, 4($sp) # save b**

**sw $a2, 8($sp) # save c**

**jal func # call func(a,b)**

**lw $a0, 4($sp) # $a0 = b**

**lw $a1, 8($sp) # $a1 = c**

**sw $v0, 4($sp) # save result of func(a,b)**

**jal func # call func(b,c)**

**lw $t0, 4($sp) # $t0 = result of func(a,b)**

**addu $v0, $t0, $v0 # $v0 = func(a,b)+func(b,c)**

**lw $ra, 0($sp) # restore return address**

**addiu $sp, $sp, 12 # free stack frame**

**jr $ra # return to caller**

Register $ra is equal to the return address in the caller function, registers $sp has the same value they had when function f was called,

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.

**Solution:**

**str2int: # convert string to integer**

**li $t6, 0x30 # $t6 = '0'**

**li $t7, 0x39 # $t7 = '9'**

**li $v0, 0 # initialize $v0 = 0**

**move $t0, $a0 # $t0 = pointer to string**

**lb $t1, ($t0) # load $t1 = digit character**

**LOOP:**

**blt $t1, $t6, NoDigit # char < ‘0’**

**bgt $t1, $t7, NoDigit # char > ‘9’**

**subu $t1, $t1, $t6 # convert char to integer**

**mul $v0, $v0, 10 # multiply by 10**

**add $v0, $v0, $t1 # $v0 = $v0 \* 10 + digit**

**addiu $t0, $t0, 1 # point to next char**

**lb $t1, ($t0) # load $t1 = next digit**

**bne $t1, $0, LOOP # branch if not end of string**

**jr $ra # return integer value**

**NoDigit:**

**li $v0, -1 # return -1 in $v0**

**jr $ra**

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

**Solution:**

**hexstr2int: # convert hex string to int**

**li $t4, 0x41 # $t4 = 'A'**

**li $t5, 0x46 # $t7 = 'F'**

**li $t6, 0x30 # $t6 = '0'**

**li $t7, 0x39 # $t7 = '9'**

**li $v0, 0 # initialize $v0 = 0**

**move $t0, $a0 # $t0 = pointer to string**

**lb $t1, ($t0) # load $t1 = digit character**

**LOOP:**

**blt $t1, $t6, NoDigit # char < ‘0’**

**bgt $t1, $t7, HEX # check if hex digit**

**subu $t1, $t1, $t6 # convert to integer**

**j Compute # jump to Compute integer**

**HEX:**

**blt $t1, $t4, NoDigit # char < ‘A’**

**bgt $t1, $t5, NoDigit # char > ‘F’**

**addiu $t1, $t1, -55 # convert: ‘A’=10,‘B’=11,etc**

**sll $v0, $v0, 4 # multiply by 16**

**add $v0, $v0, $t1 # $v0 = $v0 \* 16 + digit**

**addiu $t0, $t0, 1 # point to next char**

**lb $t1, ($t0) # load $t1 = next digit**

**bne $t1, $0, LOOP # branch if not end of string**

**jr $ra # return integer value**

**NoDigit:**

**li $v0, -1 # return -1 in $v0**

**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.