CPE 315
Professor Retz

Lab 2 (r2): MIPS operation, Creation of Functions Using the Stack, Function Calls

Purpose: Creation of Functions (MIPS assembler subroutines) that assume arguments (parameters) in registers. Define your calling convention for the stack. Functions called should save the return address and any other necessary data to the stack, restoring it before returning. This involves defining the stack frame you are using, including saved $fp, $ra, and any saved registers.

Create MIPS assembler subroutines that are called as functions performing the following:

1. Create a function 'bintohex' to Convert a Binary number to a Hexadecimal null-terminated string

This function will accept an argument in a $a0 register, with a pointer to a storage area in $a1. (Said storage area should be at least 10 bytes in length.) It will generate a null-terminated string of hexadecimal characters corresponding to the value in $a0. Each hexadecimal digit (in ASCII) represents a 4-bit value in the 32-bit number passed in $a0.

2. Create a recursive function that computes the Fibonacci value.

If necessary, use the stack for temporary storage.
Fibonacci sequence:
"By definition, the first two numbers in the Fibonacci sequence are either 1 and 1, or 0 and 1, depending on the chosen starting point of the sequence, and each subsequent number is the sum of the previous two."

3. Double fixed-point arithmetic add double_add(ahi, alo, bhi, blo)

(ahi, alo) and (bhi, blo) are provided in $a0 through $a3, respectively, and each are 32-bit values that comprise signed 64-bit values a and b. This returns a 64 bit value in the return registers $v0 and $v1 consisting of the signed 64-bit sum of a and b. The high-order portion of the value is returned in $v0.

Allow input of an arbitrary value for A and B, produce a 64 bit result, then print the result after converting to hex using bintohex, printing the result with a syscall.

4. Use of Shift instructions to select fields.

Make a subroutine that assumes a 32-bit binary value formatted as bit fields as follows: fff0 0nn0 0000 x000 yyyy 0000 0000 0000

Subroutine returns a 32-bit value having the following format:

0000 0000 0000 0000 yyyy 000x 0fff 00nn

Demonstrate with the following test data:

6a. 0x6608C000
6b. 0xC2008000
6c. Also detect and demonstrate that any invalid input is detected, i.e., specified 0 fields are non-zero.

Each function must make use of the stack for temporary storage. Parameters may be passed in registers ($a0, $a1, $a2, etc.) and returned in registers ($v0, $v1). Each function should initialize a stack frame upon entry, and should save the previous stack frame pointer ($fp) in the current stack frame.

In all three cases, there should be a test routine to verify that the desired operation is working correctly (i.e., a test driver program).

What to Hand In: Completed and fully documented listing of your program subroutines and the calling program in each case. Documentation MUST include calling conventions (registers assumed and returned). Calling program must display result by converting to hex using bintohex, then print using a syscall.