David J Tinley

09/24/2023

Lab 1 Report

Computer Organization

For this lab assignment I was able to utilize macro’s that I had learned about online. They are essentially functions that you can write instead of having to write each line of assembly code over and over for things like print or read in the terminal. The second concept I used was a loop. The MIPS loops were like using for loops in C/C++. A big difference being you must declare your loop iterator and maximum loop counter outside of the loop. You also must explicitly tell the program where to branch to after the loop is finished.

*#*

*# Lab 1 - Computer Organization*

*# David J Tinley*

*# 09/22/2023*

*# Write a MIPS program to compute f = g - (f + 5)*

*#*

*# MACROS #########################################################*

.macro print\_string(%string) *# macro for printing string parameter*

li $v0, 4 *# load syscall for print string in $v0*

la $a0, %string *# load address of string to be printed*

syscall

.end\_macro

.macro input(%num) *# macro for inputting integer*

li $v0, 5 *# load syscall for reading integer*

la $a0, %num *# load address of integer to be input*

syscall *# value stored in $v0*

.end\_macro

.macro print\_result(%num) *# macro for printing result*

li $v0, 1 *# load syscall for printing integer*

la $a0, (%num) *# load address of integer to be printed*

syscall

.end\_macro

*##################################################################*

*# DATA ###########################################################*

.data

.align 2 *# align memory to 2^2, so 4 for word*

*# alignment must be declared before .word???*

*# declaration must also be literally aligned*

*# with .word???*

\_f: .word 0 *# 32 bit integer for f*

\_g: .word 0 *# 32 bit integer for g*

\_new\_line: .asciiz "\n"

\_f\_prompt: .asciiz "Enter a value for f: "

\_g\_prompt: .asciiz "Enter a value for g: "

\_answer: .asciiz "Answer for f = g - (f + 5): "

*##################################################################*

*# TEXT ###########################################################*

.text

li $t2, 0 *# load loop counter (i = 0)*

li $t3, 3 *# load max loop iterations (i < 3)*

loop: *# loop through 3 times total*

beq $t2, $t3, exit *# branch to exit when $t2 == $t3*

li $t1, 5 *# load immediate value 5 into $t1*

*# used for equation (\_f + 5)*

print\_string(\_f\_prompt) *# print prompt for \_f input*

input(\_f) *# input value for f variable*

la $s1, ($v0) *# store value of \_f into $s1*

print\_string(\_g\_prompt) *# print prompt for \_g input*

input(\_g) *# input value for g*

la $t0, ($v0) *# store value of \_g into $t0*

add $s1, $t1, $s1 *# $s1 = (\_f + 5)*

sub $s1, $t0, $s1 *# $s1 = \_g - \_f*

print\_string(\_answer) *# print string "Answer for f = g - (f + 5):"*

print\_result($s1) *# print result*

print\_string(\_new\_line) *# print new line character*

addi $t2, $t2, 1 *# increment $t2 by 1 (++i)*

j loop *# jump back to top of loop*

*# EXIT SYSCALL ###############################################*

exit:

li $v0, 10 *# load syscall for program exit*

syscall

*##################################################################*

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

In conclusion, I think the project went well. The biggest issue I faced was understanding how memory alignment works within the code. I was eventually able to figure how the memory must be aligned differently for words, half-words, and so on. I do not fully understand why the alignment call had to be positioned exactly where I had put it in the code though.