Prob 1.

.data

prompt: .asciiz "Enter the value of n: "

output: .asciiz "a\_n = "

.asciiz "b\_n = "

.text

.globl main

main:

li $v0, 4 # print prompt message

la $a0, prompt

syscall

li $v0, 5 # read integer value of n

syscall

move $t0, $v0 # save n in $t0

li $a0, 2 # set a\_1 to 2

li $a1, 1 # set b\_1 to 1

jal calc\_sequence # calculate a\_n and b\_n

la $a0, output # print results

li $v0, 4

syscall

move $a0, $v0 # print a\_n

li $v0, 1

syscall

la $a0, output+4 # print results

li $v0, 4

syscall

move $a0, $v1 # print b\_n

li $v0, 1

syscall

li $v0, 10 # exit program

syscall

calc\_sequence:

subu $sp, $sp, 8 # allocate stack space

sw $ra, 4($sp) # save return address

# base case

beq $t0, 0, set\_zero # if n == 0, set a\_n and b\_n to 0

beq $t0, 1, set\_one # if n == 1, set a\_n and b\_n to 2 and 1

# recursive case

addi $t0, $t0, -1 # calculate a\_{n-1}

jal calc\_sequence

add $a0, $v0, $v1 # calculate a\_n = a\_{n-1} + 2\*b\_{n-1}

sll $a0, $a0, 1

addi $t0, $t0, 1 # calculate b\_{n-1}

jal calc\_sequence

sub $a1, $v0, $v1 # calculate b\_n = a\_{n-1} - b\_{n-1}

# return values

lw $ra, 4($sp) # restore return address

addu $sp, $sp, 8 # deallocate stack space

jr $ra

set\_zero:

li $v0, 0

li $v1, 0

jr $ra

set\_one:

li $v0, 2

li $v1, 1

jr $ra

The code prompts the user to enter the value of **n**, reads the value from the console, and then calls the **calc\_sequence** function to calculate **a\_n** and **b\_n**. The **calc\_sequence** function uses recursion with nested procedure linking to implement the sequence calculation. The base case is when **n** is 0 or 1, and the recursive case involves calling the function with **n-1** and using the returned values to calculate **a\_n** and **b\_n**.

The smallest positive value of n for which the code will result in an error is 46.

The reason for this error is that the value of b\_n becomes negative due to the calculation b\_n = a\_{n-1} - b\_{n-1}, which causes an integer overflow. When an integer variable exceeds its maximum value, it "wraps around" to a negative value. In this case, the value of b\_n becomes negative when the calculation exceeds the maximum value that can be stored in an integer variable.

To fix this error, we can use a data type that can handle larger values, such as a long long integer. We can also add a check to make sure that b\_n does not become negative, and terminate the recursion if it does.

Prob 2.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1-bit input op-code | 1-bit input carry |  |
|  | ALU | | 32-bit output res |
| 32-bit input A | 1-bit output zero |
| 32-bit input B | 1-bit output cout |
|  | 1-bit output ofl |

library ieee;

use ieee.std\_logic\_1164.all;

entity alu32 is

generic (

bw : integer := 32

);

port (

A : in std\_logic\_vector(bw - 1 downto 0);

B : in std\_logic\_vector(bw - 1 downto 0);

op : in std\_logic\_vector(3 downto 0);

cout : out std\_logic;

zero : out std\_logic;

less : out std\_logic;

res : out std\_logic\_vector(bw - 1 downto 0);

cin : inout std\_logic;

ofl : out std\_logic

);

end alu32;

architecture behavioral of alu32 is

signal temp : std\_logic\_vector(bw - 1 downto 0);

begin

case op is

when "000" => -- Add

temp := A + B;

when "001" => -- Subtract

temp := A - B;

when "010" => -- AND

temp := A and B;

when "011" => -- OR

temp := A or B;

when others => -- Not implemented

temp := (others => '0');

end case;

cout <= temp(31);

zero <= temp(30);

less <= temp(29);

res <= temp(bw - 1 downto 0);

ofl <= temp(bw);

cin <= cin;

end behavioral;