COMPUTER SYSTEM ARCHITECTURE

LAB REPORT #04

Translating a C Program to RISC-V Assembly

Prepared by:
SHAZIL 532263

Presented To:
LE USAMA SHAUKAT

NUST CEME
DEPT. COMPUTER ENGINEERING

TABLE OF CONTENT

Lab#4 Translating a C Program to RISC-V Assembly

Introduction:

In this lab, we will learn how to convert a simple C program into its RISC-V assembly equivalent and writing it in Venus. Our chosen example is a program that calculates and prints the nth Fibonacci number.

Step 1: The C Program

Let's begin with the C version of the code:

```
#include <stdio.h> int n = 12;

int main(void) {
   int curr_fib = 0, next_fib = 1;     int new_fib;
   for (int i = n; i > 0; i--) {          new_fib = curr_fib +
next_fib;       curr_fib = next_fib;       next_fib = new_fib;
   }
   printf("%d\n", curr_fib);     return 0; }
```

This program simply loops **n** times to generate the nth Fibonacci number.

Objective:

To understand the process of translating a simple C program into RISC-V assembly. The goal

is to practice writing, running, and debugging RISC-V code in Venus by converting a Fibonacci number generator from C to assembly.

Lab Task:

Quiz Question:
Now, your task is to combine all these steps into a single complete RISC-V Assembly program **that prints the** entire Fibonacci series up to n terms**.**

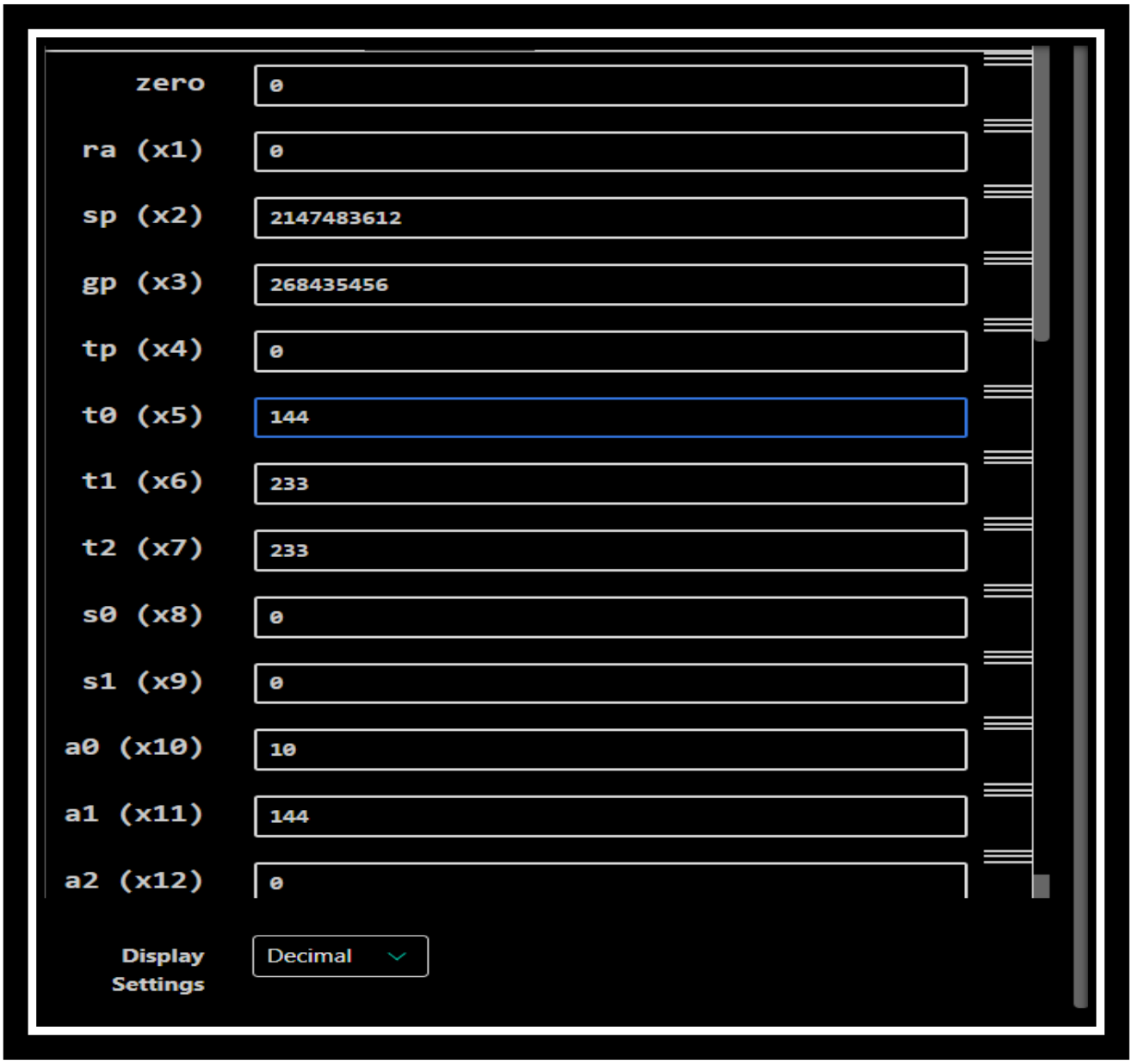
```
.data
n: .word 12

.text
main:
    addi t0 , x0 ,0
    addi t1 , x0 , 1
    la t3 , n
    lw t3 , 0(t3)

loop:
    beq t3 , x0 , exit
    add t2 , t1 , t0
    add t0 , t1 , x0
    add t1 , t2, x0
    addi t3 , t3 , -1
    beq x0 , x0 , loop

exit:
    addi a0 , x0 , 1
    addi a1 , t0 , 0
    ecall
    addi a0 , x0 , 10
    ecall
```

PC	Machine Code	Basic Code	Original Code
0x0	0x00000293	addi x5 x0 0	addi t0 , x0 ,0
0x4	0x00100313	addi x6 x0 1	addi t1 , x0 , 1
0x8	0x10000E17	auipc x28 65536	la t3 , n
0xc	0xFF8E0E13	addi x28 x28 -8	la t3 , n
0x10	0x000E2E03	lw x28 0(x28)	lw t3 , 0(t3)
0x14	0x000E0C63	beq x28 x0 24	beq t3 , x0 , exit
0x18	0x005303B3	add x7 x6 x5	add t2 , t1 , t0
0x1c	0x000302B3	add x5 x6 x0	add t0 , t1 , x0
0x20	0x00038333	add x6 x7 x0	add t1 , t2, x0
0x24	0xFFFE0E13	addi x28 x28 -1	addi t3 , t3 , -1

0x14	0x000E0C63	beq x28 x0 24	beq t3 , x0 , exit
0x18	0x005303B3	add x7 x6 x5	add t2 , t1 , t0
0x1c	0x000302B3	add x5 x6 x0	add t0 , t1 , x0
0x20	0x00038333	add x6 x7 x0	add t1 , t2, x0
0x24	0xFFFE0E13	addi x28 x28 -1	addi t3 , t3 , -1
0x28	0xFE0006E3	beq x0 x0 -20	beq x0 , x0 , loop
0x2c	0x00100513	addi x10 x0 1	addi a0 , x0 , 1
0x30	0x00028593	addi x11 x5 0	addi a1 , t0 , 0
0x34	0x00000073	ecall	ecall
0x38	0x00A00513	addi x10 x0 10	addi a0 , x0 , 10
0x3c	0x00000073	ecall	ecall

OUTPUT:


0 1 1 2 3 5 8 13 21 34 55 89
Sum = 232


zero 0
ra (x1) 0
sp (x2) 2147483612
gp (x3) 268435456
tp (x4) 0
t0 (x5) 144
t1 (x6) 233
t2 (x7) 233
s0 (x8) 0
s1 (x9) 0
a0 (x10) 10
a1 (x11) 144
a2 (x12) 0
Display Settings
Decimal

Exercise 2: Exploring the Venus Debugger

Question 1: What is the machine code of the first highlighted instruction?
0x00000293

Question 2: What is the machine code of the instruction stored at memory address 0x34?
0x00000073

Question 3: What is the value of the sp register?
0x7fffffdc

Question 4: What is the new value stored in t1? (32-bit hex, with 0x)
0x00000001

Question 5: What is the machine code of the current instruction? (32-bit hex with 0x)
0x10000e17

Question 6: What is the current value of the t3 register? (32-bit hex with 0x)
0x00000001

Question 7: What single byte (8-bit value) is stored at the address pointed to by t3? (Hex with 0x)
C

Question 8: What is the value of t0 at this point? (32-bit hex with 0x)
0x00000001

Question 9: What new value does t0 hold now? (32-bit hex with 0x)
0x00000090

Question 10: What is the value of t0 now, in decimal (no prefix)?
144

Question 11: What is the program's final output? (Write the number in decimal, no prefix).
144

TASK 02:

An array array1 contains the sequence:
-1 22 8 35 5 4 11 2 1 78 each
element of which is word.
Rearrange the element order in this array such that,
• All the elements smaller than the 3rd element (i.e. 8) are on the left of it, •
All the elements bigger than the 3rd element (i.e. 8) are on the right of it.

RISC-V CODE:

```
.data
array1: .word -1, 22, 8, 35, 5, 4, 11, 2, 1, 78
n:      .word 10
pivot:  .word 8

.text
.globl main
main:
    la   s0, array1
    lw   s1, n
    lw   s2, pivot

    addi t0, x0, 0
    addi t1, s1, -1

partition_loop:
    bge  t0, t1, print_array

    slli t3, t0, 2
    add  t4, s0, t3
    lw   t2, 0(t4)

    blt  t2, s2, inc_i

```

```
    blt  t2, s2, inc_i

    slli t6, t1, 2
    add  s3, s0, t6
    lw   t5, 0(s3)

    blt  s2, t5, dec_j

    sw   t5, 0(t4)
    sw   t2, 0(s3)

    addi t0, t0, 1
    addi t1, t1, -1
    j    partition_loop

inc_i:
    addi t0, t0, 1
    j    partition_loop

dec_j:
    addi t1, t1, -1
    j    partition_loop

```

```
    addi t0, t0, 1
    j    partition_loop

dec_j:
    addi t1, t1, -1
    j    partition_loop

print_array:
    addi t0, x0, 0

```

```
print_loop:
    blt  t0, s1, print_continue
    j    done

print_continue:
    slli t3, t0, 2
    add  t4, s0, t3
    lw   a1, 0(t4)

    addi a0, x0, 1
    ecall

    addi a1, x0, 32
    addi a0, x0, 11
    ecall

    addi t0, t0, 1
    j print_loop

done:
    addi a1, x0, 10
    addi a0, x0, 11
    ecall
```

PC	Machine Code	Basic Code	Original Code
0x0	0x10000417	auipc x8 65536	la s0, array1
0x4	0x00040413	addi x8 x8 0	la s0, array1
0x8	0x10000497	auipc x9 65536	lw s1, n
0xc	0x0204A483	lw x9 32(x9)	lw s1, n
0x10	0x10000917	auipc x18 65536	lw s2, pivot
0x14	0x01C92903	lw x18 28(x18)	lw s2, pivot
0x18	0x00000293	addi x5 x0 0	addi t0, x0, 0
0x1c	0xFFF48313	addi x6 x9 -1	addi t1, s1, -1
0x20	0x0462D463	bge x5 x6 72	bge t0, t1, print_array
0x24	0x00229E13	slli x28 x5 2	slli t3, t0, 2

0x28	0x01C40EB3	add x29 x8 x28	add t4, s0, t3
0x2c	0x000EA383	lw x7 0(x29)	lw t2, 0(t4)
0x30	0x0323C463	blt x7 x18 40	blt t2, s2, inc_i
0x34	0x00231F93	slli x31 x6 2	slli t6, t1, 2
0x38	0x01F409B3	add x19 x8 x31	add s3, s0, t6
0x3c	0x0009AF03	lw x30 0(x19)	lw t5, 0(s3)
0x40	0x03E94063	blt x18 x30 32	blt s2, t5, dec_j
0x44	0x01EEA023	sw x30 0(x29)	sw t5, 0(t4)
0x48	0x0079A023	sw x7 0(x19)	sw t2, 0(s3)
0x4c	0x00128293	addi x5 x5 1	addi t0, t0, 1
0x50	0xFFF30313	addi x6 x6 -1	addi t1, t1, -1

0x54	0xFCDFF06F	jal x0 -52	j partition_loop
0x58	0x00128293	addi x5 x5 1	addi t0, t0, 1
0x5c	0xFC5FF06F	jal x0 -60	j partition_loop
0x60	0xFFF30313	addi x6 x6 -1	addi t1, t1, -1

0x84	0x00000073	ecall	ecall
0x88	0x02000593	addi x11 x0 32	addi a1, x0, 32
0x8c	0x00B00513	addi x10 x0 11	addi a0, x0, 11
0x90	0x00000073	ecall	ecall
0x94	0x00128293	addi x5 x5 1	addi t0, t0, 1
0x98	0xFD5FF06F	jal x0 -44	j print_loop
0x9c	0x00A00593	addi x11 x0 10	addi a1, x0, 10
0xa0	0x00B00513	addi x10 x0 11	addi a0, x0, 11
0xa4	0x00000073	ecall	ecall
0xa8	0x00A00513	addi x10 x0 10	addi a0, x0, 10
0xac	0x00000073	ecall	ecall

OUTPUT:

Copy! Download! Clear!

```
-1 5 4 2 1 8 22 35 11 78
```

CONCLUSION:

Through this lab, we successfully translated a C program into its RISC-V assembly equivalent. The exercise reinforced how high-level constructs such as loops, variables, and arithmetic operations map to assembly instructions. It also improved familiarity with the Venus simulator and gave practical experience in understanding the low-level execution of a program.