ECE9047: Laboratory 1 Report

Ojective

This lab is about programming an ARM®Cortex-A9 on a DE1 SoC board in the ARMv7 assembly language. The goal of this lab is to implement the following system: Write the first 10 Fibonacci numbers to a seven-segment display. Show each number on the display for one second (1 s) before showing the next. The program should endlessly loop, cycling through displaying these 10 numbers.

Explanation

The implementation involves three main components:

Generating Fibonacci Numbers

A precomputed table (SEGMENT_MAP) is used to store the seven-segment display representations of the Fibonacci numbers.

An index (R8, range from 0-10) iterates through this table to display the numbers sequentially.

Displaying the Numbers on a Seven-Segment Display

The program loads the corresponding segment pattern from SEGMENT_MAP and writes it to the display register at 0xFF200020.

Implementing a 1-Second Delay

The program utilizes a timer to count down from a predefined interval (timer_T = 1000000000 cycles) to achieve a 1-second delay.

The timer is monitored to determine when 1 second has elapsed before proceeding to the next Fibonacci number.

Timer Implementation

The first thing was to initialize the timer. The timer base address (0xFF202000) is loaded, and an initial countdown value (timer_T) is set and stored in the timer registers. Once the timer is configured, the main loop execution starts, during which the timer begins counting down while the program continuously checks if it has reached zero. When one second has elapsed, the next Fibonacci number is retrieved from SEGMENT_MAP and displayed on the output. If all ten numbers have been displayed, the index is reset to restart the sequence. This process loops indefinitely, with the program continuously cycling through the ten Fibonacci numbers. After completing a full cycle, it resets the timer and restarts the display sequence, ensuring

a continuous and automated operation.

Description

In the practical implementation, I met many difficulties, but finally solved them all. I calculated the first 10 Fibonacci numbers in advanced manually, which is quite easy. To display them on the 7-segment monitor, they should be converted into corresponding 7-bit data based on the patterns. And since it requires two displays to display 13,21 and 34, the data for 0xff200020 is 16-bit or 4-bitin hexadecimal. Then I created a loop to read the numbers in the array. At first, I am not sure how to update the number in the array as the loop went on, luckily, I found that an offset of LSL #2 for index R8 can solve this. It can shift 2 bits duration, which equals to the length of the word type, leading to the next number in the array. I did not encounter any more problems getting the number displayed.

As for the timer, I originally decided to write a subroutine(delay_1s) to achieve it. I then found that I cannot get the counter of the timer, i.e., 0xff202010 and 0xff202014, which made my codes cannot know when to stop and exit the loop. I tried to delete judging condition afterwards. It turned out that the interval setting of the timer did not work somehow, the 7-segment display was just keeping flashing. Finally, I gave up writing the subroutine and referred to the code in the textbook, which only use a main loop to fulfill the timer function, as mentioned above in part 2. In the end, I succeeded in achieving all the requirements

Cost-Benefit Analysis

For convenience, I did not design an algorithm to compute specific Fibonacci numbers dynamically. Instead, I created a lookup table where each number is precomputed in advance. On the one hand, it is highly efficient because using a lookup table (SEGMENT_MAP) allows quick retrieval of Fibonacci values without the need for runtime computation, reducing processing overhead. The code is also simple and easy to understand, as it follows a straightforward looping mechanism, making debugging and maintenance more manageable. Additionally, the use of a hardware timer ensures accurate 1-second delays, enhancing the reliability of the system by eliminating potential timing inaccuracies that could arise from software-based delays.

However, the approach also has some limitations. It lacks scalability since it is hardcoded to display only ten Fibonacci numbers, meaning that if the requirement changes to a larger set, the SEGMENT_MAP must be manually updated, making it less flexible. The code also suffers from limited reusability, as the timer and display functions are embedded within the main loop, reducing modularity and making it harder to adapt for different applications. Furthermore, the use of fixed memory addresses (0xFF202000 and

0xFF200020) restricts portability, as the program is specifically designed for a particular hardware platform and would require modifications to function on a different system.

Code

```
.global _start

.data

timer_T: .word 1000000000

timer_sT: .word 100000000

SEGMENT_MAP:

    .word 0x3F3F, 0x3F06, 0x3F06, 0x3F5B, 0x3F4F, 0x3F6D, 0x3F7F, 0x064F, 0x5B06, 0x4F66

.text

_start:

@ get the addresses

ldr r4 , =0xff202000

ldr r5 , =0xff200020

ldr r6 , adr_T

ldr r7 , adr_sT

@ initialize the 5 s count


ldr r0 , [ r6 ]

str r0 , [ r4 , #8 ]

mov r1 , r0 , lsr #16

str r1 , [ r4 , #12 ]

mov r1 , #6

str r1 , [ r4 , #4 ]

ldr r1 , [ r7 ]
```

```
MOV R8, #0                  @ index（starting from 0）

main_loop:

str r1 , [ r4 , #16 ]
ldr r2 , [ r4 , #16 ]
ldr r3 , [ r4 , #20 ]

add r2 , r3 , lsl #16
@ check if current count has
@ passed 1 s
cmp r2 , r0
BHI main_loop               @ If higher (unsigned comparison), branch to main_loop
LDR R9, =SEGMENT_MAP    @ Load the address of the digit mapping table
LDR R9, [R9, R8, LSL #2]    @ Load the value corresponding to the index R8
STR R9, [R5]                @ Store the value into the 7-segment display
ADD R8, R8, #1              @ Increment the index


cmp R8, #11
@ if not , subtract 1 s from interval
@ and loop back
sublo r0 , r1
blo main_loop
@ if we passed 5 s , reset interval
ldr r0 , [ r6 ]
MOV R8, #0
```

```
b main_loop

adr_T: .word timer_T

adr_sT: .word timer_sT
```