California State University, Northridge

Department of Electrical & Computer Engineering

Lab 7

Loops and Branches

October 27, 2022

ECE 425L

Assigned by: Neda Khavari

Written By: Cristian Robles & Pathum Addarapathirana

Introduction:

In lab 7 we further our understanding of loops and branches and are introduced to using strings in assembly

Procedure:

Equipment Used

- Keil uVision4
- Keil Debugger
- LPC2148 Education Board

Description of Procedure

1. Set up startup code and constants. This is the code needed to start the lab that includes the my startup code from lab 3. Here we also set the constants used in the pin configuration we will use later to control the LEDs.

```
1         GLOBAL  user_code
2         area    mycode, CODE, READONLY
3  user_code   ;this label is neccessary
4
5  IO0BASE     EQU 0xE0028000
6  IO0PIN      EQU 0
7  IO0DIR      EQU 0x8
8  IO0SET      EQU 0x4
9  IO0CLR      EQU 0xC
10
11 pointer     RN      r2
12 next        RN      r6
13 counter     RN      r3
```

2. In the next part of our code we set the pins we will be working with as GPIO and set the direction of the pins to be outputs

```
15 PINSEL0     EQU     0xE002C000      ;pin function for port 0, equate symb
16
17                                     ;;Selecting funcion as GPIO by writin
18             MOV     r0,#0           ;moves #0 into register r0
19             LDR     r1,=PINSEL0     ;puts what is stored in register 0xE0
20             STR     r0,[r1]         ;copies value stored in r0 to memory
21
22                                         ;;Selecting signal direction of e
23             MOV     r0, #0x0000FF00     ;moves 0x0000FF00 (binary 1111111
24             LDR     r1, =IO0BASE        ;puts 0xE0028008 or IO0DIR to r1
25             STR     r0, [r1,#IO0DIR]    ;copies value stored in r0 (0x000
26
27             MOV     r5,#128             ;initial substract
28             MOV     r4, #0x0000FF00 ;moves #0x00000000 into register r0 (
```

3. In this part of the code we use a loop to look for the value 0x6F000000 which represents the character ‘o’ inside the string. We do this by using a pointer and a next register to compare 2 values at once. To align the values we want when comparing we use LSL. Once the program encounters the end of the string which was a 0 we move onto the next part of the code

```
31
32              LDR     r0, =IO0BASE
33              LDR     r1,[r0,#IO0PIN]
34  ;Task1 start
35
36          MOV     counter,#0
37          LDR     pointer, =mydata
38  loop    LDR     next, [pointer]
39          LDR     r7,=0x6F000000              ;6F is the thing
40          LSL     next,#24
41          CMP     next, r7
42          BEQ     found
43  jump    ADD     pointer, pointer, #1
44          CMP     next,#0                     ;ending, checking for 0
45          BEQ     part2
46          B       loop
47
```

4. Once the ‘o’ characters are found we start to turn on each LED one by one for the number of times the character was counted. We also use a delay in between each LED turning on so that we can see it happen as the code would be moving so fast that it would appear that all the LEDs were turned on at once.

```
48  found   ADD     counter,counter,#1
49              ;MOV        r4, #0x0000FF00 ;moves #0x00000000 into register
50              MOV     r5,r5, LSL#1
51              SUB     r4,r4,r5
52              LDR     r1, =IO0BASE        ;puts 0xE0028008 or IO0DIR to r1
53              STR     r4, [r1,#IO0PIN]
54
55              ;Delay
56              LDR     r9,=500000
57  delay_1     SUBS    r9,r9,#1
58              BNE     delay_1
59
60          B       jump
61
62
63  part2   ADD     r0,r0,#1
```

5. In this part of the code we used DCB to allocate memory for the string. We also used the ,0 at the end to make it easier to detect the end of the string.

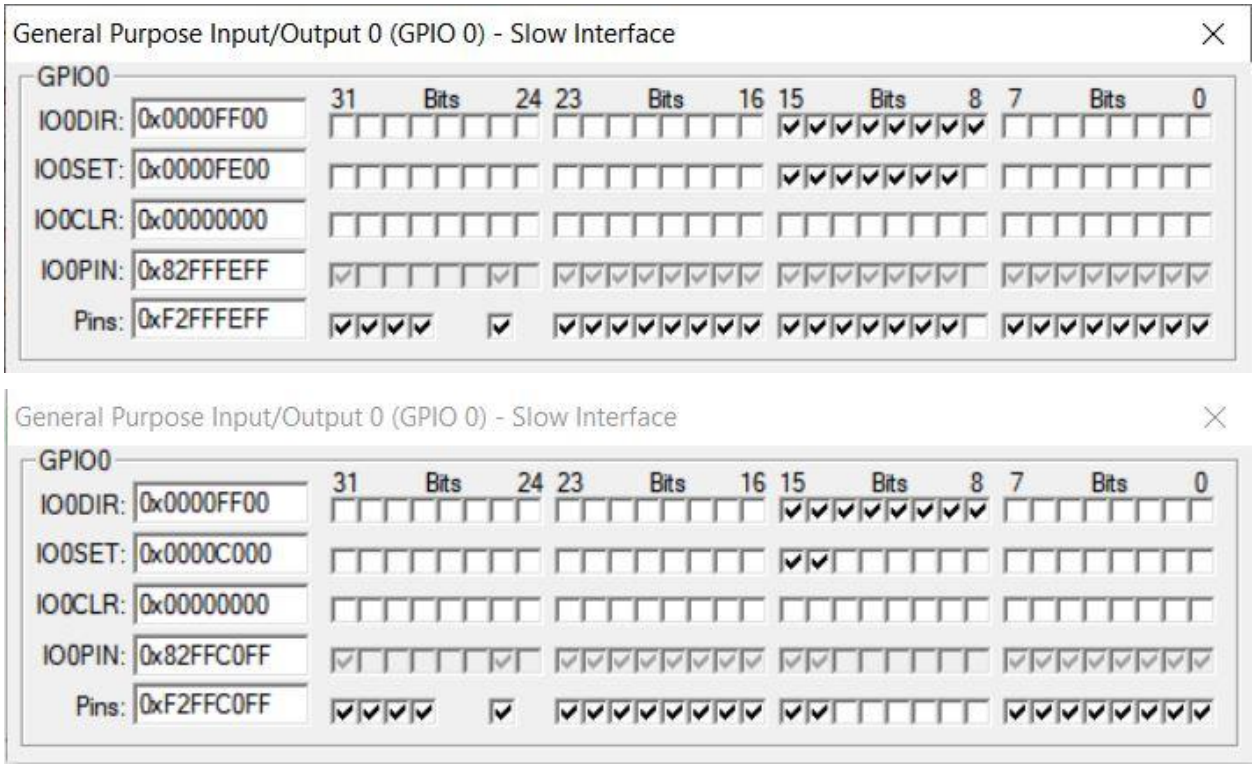
```
65  stop        B           stop
66  mydata       DCB     "If you are going to pass, you gotta love this class",0
67                  ALIGN
```

6. In task 2 since we already had the pins set to as GPIO we could begin the loop to alternate them to be on or off right away. We used a counter variable stored in R10 that would be initialized at five and would subtract once everytime it went through the loop. For the delay of five seconds we calculated the value 1,724,137 and subtracted once from it every time it looped. Finally we checked if the counter was 0 and went to the end of the code ending the program

```
107             LDR     r5, =IO0CLR     ;puts 0XE002800C or IO0CLR to r1 register
108             STR     r2,[r3,r5]          ;copies value stored in r2 to memory address
109             MOV     r10,#5
110 LOOP3
111             ;LOOP begin
112             LDR     r9,=1724137
113 LOOP        SUBS    r9,r9,#1
114             BNE     LOOP
115             ;SUBS   r10,r10,#1
116                                 ;turn them ON again
117             ;MOV        r0, #0x0000FF00 ;moves 0x0000FF00 (binary 1111111100000000) i
118             ;LDR        r1, =IO0SET     ;puts 0XE0028004 or IO0SET to r1 register
119             STR     r2,[r3,r5]          ;copies value stored in r2 to memory address
120
121             LDR     r8,=1724137
122 LOOP2       SUBS    r8,r8,#1
123             BNE     LOOP2
124                                 ;turn them OFF again
125             ;MOV        r0, #0x0000FF00 ;moves 0x0000FF00 (binary 1111111100000000) i
126             ;LDR        r1, =IO0CLR     ;puts 0XE002800C or IO0CLR to r1 register
127             STR     r2,[r3,r4]          ;copies value stored in r0 to memory address
128
129             SUBS    r10,r10,#1
130             CMP     r10,#0
131             BEQ     stop
132             B       LOOP3
```

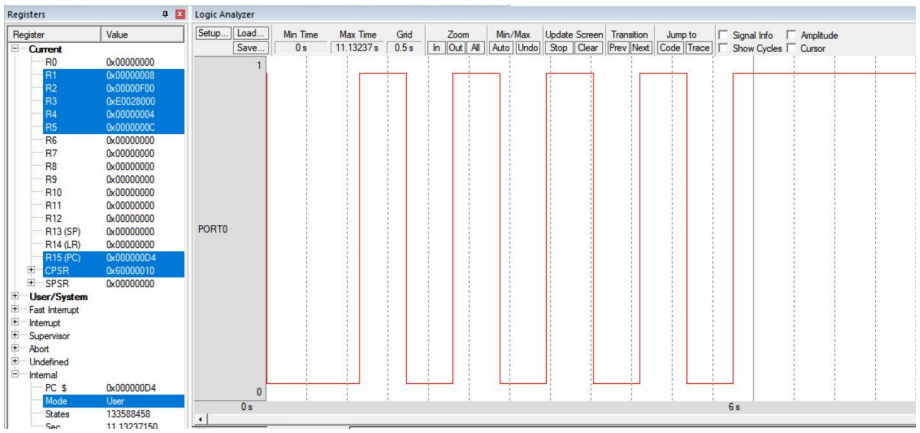
Results:

Task 1



In the simulator we were able to observe the lights turning off one by one by setting a delay after each light was turned on after the character ‘o’ was detected in the string. On the board we also observed the same thing

Task 2



In the logic analyzer we observed that pin 8 would successfully flash on and off five times with a delay in between. The board also did the flash five times successfully but the delay time differed between simulation and on the board

Questions:

```
myprogram   ⋮
            ⋮
stop        B       stop
mystring    DCB     "This is an example of strings",0
message     DCB     "Attention please!",0
            ALIGN
            END
```

Why is ALIGN necessary to be placed after DCBs?

The align is necessary to make the machine code placed after the characters aligned within 4 bit boundaries

Conclusion:

In conclusion, we were able to successfully detect the 'o' character and use the detection as a condition to flash or turn on the LEDs. Using the branch instructions allowed us to make loops for delays and also change the flow of the program by jumping to other points of the code. We observed that the code ran successfully on both the simulation and on the board although the delay times differed due to the simulator not being exactly the same as the hardware. A way this lab could be improved is to change the simulator to be the same as the board and get more accurate simulation results. This project furthered our knowledge of loops, branches, and using conditions.