MEF UNIVERSITY

YOUR FREEDOM IN LEARNING

EE306 - Microprocessors

Laboratory Exercise 4
I/O

April 10, 2020

Alp Gokcek
#041701014

1 Part I - Display HEX on 7-Segment Display

In this experiment, I've converted each hexadecimal number from 0-a into their 7-segment equivalent and stored in memory. We have 4 buttons which are for display 0, increment displayed number, decrement displayed number and blank displayed number respectively. I have created a pointer and on each key click, I change this pointer according to the key clicked.

Listing 1: Assembly Code for displaying hex numbers on 7-segment display

```
.global _start
_start:
        LDR R0, DISPLAY_BASE // 7-segment display base address
        LDR R1, =HEX_TO_7_SEG //points base address to hex of 7 segment display
        LDR R4, BUTTON_BASE //base address of buttons
        MOV R8, #0 //zero register
        LDR R2, [R1, #4] //current displayed element's value
        MOV R3, R1 // current displayed element's address
        ADD R3,R3, #4 // update address to show 0
MAIN:
        LDRB R5, [R4] //load state of button base
        CMP R5, #1 //check if KEY0 pressed
        BLEQ ZERO //if yes, branch
        CMP R5, #2 //check if KEY1 pressed
        BLEQ INCR //if yes, branch
        CMP R5, #4 //check if KEY2 pressed
        BLEQ DECR //if yes, branch
        CMP R5, #8 //check if KEY3 pressed
        BLEQ BLANK //if yes, branch
        B MAIN //continue on loop
ZERO:
        MOV R2, #63 //move 0 in hex to r2
        MOV R3, R1 //update pointer to show the first item in list which is blank
        ADD R3, R3, #4 //update pointer to show second item in list which is 0 in hex
        STR R2, [R0] //update display
        BX LR //return to main loop
INCR:
        CMP R2, #113 //check if displayed value is max
        BXEQ LR //if yes, return without updating
        LDRH R2, [R3, #4]! //update pointer
        STR R2, [R0] //update display
        BX LR //return to main loop
DECR:
        CMP R2, #63 //check if currently displayed item is 0
        BXEQ LR // if yes, return without updating
        CMP R2, #0 //check if currently blank
        BEQ ZERO //set to 0 if blank
        LDRH R2, [R3, #-4]! //update pointer
        STR R2, [R0] //update display
        BX LR
BLANK:
        MOV R2, #0 //blank value
        MOV R3, R1 //set pointer to begining of the list
        STR R2, [R0] //update display
        BX LR //return to main loop
END: B END
```

```
DISPLAY_BASE: .word 0xff200020 //display base address
BUTTON_BASE: .word 0xff200050 //buttons base address
//converted versions of hexadecimal numbers to display on 7-segment display
HEX_TO_7_SEG: .word 0,63,6,91,79,102,109,125,7,127,111,119,124,57,94,121,113

.end
```

2 Part II - Display Group ID on 7-Segment Display

In this experiment, I've stored "GrouP0" on the memory in a form that to be shown on the 7-segment display. Since it has 6 digits, we had to split into 2 rows cause it exceeds 32 bits. First row consists HEX_3 to HEX_0 and second row consists HEX_5 and HEX_4. Therefore, in first row I have stored "ouP0" and in second row I have stored "Gr". In execution, I summed first row with the class id to update our class id and show on 7-segment display.

Listing 2: Assembly code for displaying group id on 7-segment display

```
.global _start
_start:
       LDR R0, DISPLAY_BASE // 7-segment display base address
       LDR R1, FIRST_ROW // holds first 4 7-segment displays
       LDR R2, SECOND_ROW // holds last 2 7-segment displays
       LDR R3, CLASSID //R3 holds CLASSID
       ADD R1, R1, R3 //add class id to be shown on the 7-segment display
       STR R1, [R0], #16 //update first 4 7-segment displays
       STR R2, [R0] // update last 2 7-segment displays
END: B END
DISPLAY_BASE: .word 0xff200020 //display base address
CLASSID: .word 125 //class id is 6
FIRST_ROW: .word 0x5C1C7300 // "ouP0" on 7-segment display
SECOND_ROW: .word 0x3D50 // "Gr" on 7-segment display
.end
```

3 Part III - Display 2 Words with Delay

In this experiment, I have converted words assigned to me which are "SEt" and "tıme" to 7-segment form and change between these numbers after some delay. I have created a flag to check which word is currently showed on 7-segment display. If any key is pressed, then it stops switching words.

Listing 3: Assembly code for displaying 2 words with delay

```
.global _start
_start:
       LDR R0, DISPLAY_BASE // 7-segment display base address
       LDR R1, BUTTON_BASE //base address of buttons
       LDR R2, FIRST_WORD_1 //first word first row
       LDR R3, FIRST_WORD_2 //first word second row
       LDR R4, SECOND_WORD_1 //second word first row
       LDR R5, SECOND_WORD_2 //second word second row
       MOV R8, #0 //flag for current word showing
       MOV R9, #1 //value for XOR operation
MAIN:
       LDRB R6, [R1] //load state of button base
```

```
        CMP R6, #0 //check if any key is pressed
        BNE STOP_DELAY //if yes, stop delay
        BL UPDATE_DISPLAY //update display subroutine
        B DO_DELAY //else do delay

DO_DELAY: LDR R7, =2000000 // delay counter
SUB_LOOP:
        SUBS R7, R7, #1 //decrement r7
        BNE SUB_LOOP //branchif if delay is not completed
        B MAIN //branch after delay is completed

STOP_DELAY:
        LDRB R6, [R1] //load state of button base
        CMP R6, #0 // check if button is still clicked
        BNE STOP_DELAY //if still clicked continue loop
        B MAIN //else continue on main loop

UPDATE_DISPLAY:
        CMP R8, #0 //check if the displayed word is the first word
        STRNE R2, [R0] //if not load first row of first word
        STRNE R3, [R0, #16] //if not load second row of first word
        STREQ R4, [R0] //if yes, load first row of second word
        STREQ R5, [R0, #16] //if yes, load second row of second word
        EOR R8, R8, R9 //update flag
        BX LR //end subroutine
END: B END
DISPLAY_BASE: .word 0xff200020 //display base address
BUTTON_BASE: .word 0xff200050 //buttons base address
FIRST_WORD_1: .word 0x78000000
FIRST_WORD_2: .word 0x6D79
SECOND_WORD_1: .word 0x37377B00
SECOND_WORD_2: .word 0x7804
.end
```