ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 351E

MICROCOMPUTER LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	2
EXPERIMENT DATE	:	13.11.2024
LAB SESSION	:	WEDNESDAY - 12.30
GROUP NO	:	G8

GROUP MEMBERS:

150200009	:	Vedat Akgöz
150200097	:	Mustafa Can Çalışkan
150200016	:	Yusuf Şahin

SPRING 2024

Contents

1 INTRODUCTION

In this experiment, we aimed to deepen our understanding of the MSP430 Education Board and the MSP430G2553 microcontroller. By working with assembly language, we explored how to control hardware components such as LEDs and buttons. This hands-on experience allowed us, as a group, to enhance our skills in low-level programming and better understand concepts like switch debouncing.

2 MATERIALS AND METHODS

2.1 Part 1

We successfully completed Part 1, which required implementing a simple assembly program. The program waits for the user to press the button at `P2.4`, turns on the LED connected to `P1.4`, and then enters an infinite loop. Below is the assembly code we developed and tested:

```
INIT:
    bis.b #11111111b, &P1DIR    ; Set P1 as output
    bic.b #11111111b, &P1OUT    ; Turn off all LEDs on P1
    mov.b #00000000b, &P2DIR    ; Set P2 as input

START:
    bit.b #10000000b, &P2IN     ; Check if P2.4 button is pressed
    jz START                    ; Wait until button is pressed
    bis.b #00010000b, &P1OUT    ; Turn on LED at P1.4

END_LOOP:
    jmp END_LOOP                ; Infinite loop
```

2.2 Part 2

Due to time constraints during the lab session, we were unable to fully implement Part 2. However, we later developed the necessary assembly code to achieve the task. The program toggles between two LEDs connected to `P2.2` and `P2.3` using a button connected to `P1.5`. The state of the LEDs changes only when the button is pressed and released, implementing debounce logic. Below is the assembly code:

```
; SETUP PORTS
```

```
SETUP_P1            mov.b #00000000b, &P1DIR  ; P1 as input
                    mov.b #11111111b, &P2DIR  ; P2 as output
                    mov.b #00000000b, &P1OUT  ; Clear P1
                    mov.b #00000010b, &P2OUT  ; Set P2 to 0x02
                    mov.b #00000000b, &P1IN   ; Clear P1 input

; MAIN LOOP
MAIN_LOOP           bit.b #00010000b, &P1IN   ; Check P1.4 button
                    jnz SWITCH_OUTPUT         ; If pressed, switch

CONTINUE_MAIN_LOOP  jmp MAIN_LOOP             ; Repeat

; TOGGLE OUTPUT
SWITCH_OUTPUT       cmp #00000010b, &P2OUT    ; Compare P2
                    jeq SET_OUTPUT_TO_3       ; If 0x02, set to 0x04
                    jmp SET_OUTPUT_TO_2       ; Else, set to 0x02

SET_OUTPUT_TO_2     mov.b #00000010b, &P2OUT  ; Set P2 to 0x02
                    jmp WAIT_FOR_RELEASE

SET_OUTPUT_TO_3     mov.b #00000100b, &P2OUT  ; Set P2 to 0x04
                    jmp WAIT_FOR_RELEASE

; DEBOUNCE
WAIT_FOR_RELEASE    mov.w #00050000, R15      ; Delay
DEBOUNCE_LOOP       dec.w R15
                    jnz DEBOUNCE_LOOP
                    bit.b #00010000b, &P1IN   ; Check release
                    jnz WAIT_FOR_RELEASE
                    jmp CONTINUE_MAIN_LOOP
```

2.3 Part 3

Unfortunately, we were unable to complete Part 3 during the lab session due to time constraints. However, we later wrote the required assembly code. This part focuses on counting how many times the button at `P2.1` is pressed. The count is stored in a 4-bit variable and displayed on `Port 1`. The counter resets after reaching 15 (0xF), and debounce logic is implemented to prevent false counts. Below is the assembly code:

```
; PORT SETUP
SETUP_P1            mov.b #11111111b, &P2DIR   ; P2: Output
                    mov.b #11111110b, &P1DIR   ; P1.0: Input
                    mov.b #00000000b, &P2OUT   ; Clear P2
                    mov.b #00000000b, &P1IN    ; Clear P1 input

; MAIN LOOP
MAIN_LOOP           bit.b #00000001b, &P1IN    ; Check P1.0
                    jnz INCREMENT_COUNTER      ; If pressed, increment

CONTINUE_MAIN_LOOP  jmp MAIN_LOOP              ; Repeat

; COUNTER HANDLING
INCREMENT_COUNTER   mov.b counter, r4
                    cmp #00001111b, r4         ; Check if counter = 15
                    jeq RESET_COUNTER          ; If yes, reset
                    jmp INCREMENT_VALUE        ; Else, increment

RESET_COUNTER       mov.b #00000000b, r4       ; Reset counter
                    mov.b r4, counter
                    mov.b r4, &P2OUT
                    jmp DEBOUNCE

INCREMENT_VALUE     inc r4                     ; Increment counter
                    mov.b r4, counter
                    mov.b r4, &P2OUT
                    jmp DEBOUNCE

; DEBOUNCE HANDLING
DEBOUNCE            mov.w #00050000, R15
DEBOUNCE_LOOP       dec.w R15
                    jnz DEBOUNCE_LOOP          ; Wait for debounce

                    bit.b #00000001b, &P1IN    ; Check if still pressed
                    jnz DEBOUNCE               ; Wait for release

                    jmp CONTINUE_MAIN_LOOP     ; Return to main loop
```

```
; DATA SECTION
DATA_SECTION        .data
counter             .word 0x00                ; Counter variable
```

3 RESULTS

Overall, the experiment provided valuable hands-on experience with the MSP430 microcontroller. In Part 1, we successfully implemented and tested a program that controlled an LED based on button input. Although we ran out of time during the lab for Parts 2 and 3, we later completed the assembly codes for both tasks. These codes demonstrated proper toggling between LEDs and button press counting with debounce logic, ensuring reliable operation. Through this experiment, we deepened our understanding of low-level hardware control and assembly programming.

4 DISCUSSION

The primary challenge we faced during this experiment was time management. Although we successfully completed Part 1 within the allocated lab session, we were unable to finish Parts 2 and 3 due to time constraints. This limited our ability to test and refine the assembly code during the lab, requiring us to complete these sections afterward. Future sessions could benefit from better time allocation or additional lab hours to ensure all parts are completed and tested thoroughly.

5 CONCLUSION

This experiment provided us with a deeper understanding of the MSP430 microcontroller and its GPIO functionalities. Despite the time constraints, we successfully implemented all required assembly programs and gained valuable insights into low-level hardware control, including LED manipulation and button debouncing. The experience highlighted the importance of careful planning and time management in completing all tasks during the lab session. Overall, the experiment was a significant step in enhancing our proficiency in assembly language programming and embedded systems.

REFERENCES