Justin Mancinelli 42094353 Prac 7

Prep

Task 1 — Accumulator

① Write a C program to be run in the windows console which will model an accumulator

② Run — The program runs successfully without error

③ Test — Input 5 returns 5 = 0+5
Input 10 returns 15 = 5+10
Input -3 returns 12 = -3+15
Input 0 exits

④ Note: This program will not run for values other than integers.

Task 2 - Combination Lock

① Write a C program to be run in the windows console which will model a combination lock.

② Run - The program runs successfully without error

③ Test - Input 5 asks for another digit
Input 8 returns 58, closed
Input 8 asks for another digit
Input 3 returns 83, opened

④ Note: This program will not run for values other than integers and will work for any integer rather than single digit numbers.

prac7-1.c

```c
/* FILE: prac7-1.c */

#include <stdio.h>

int main(void)
{
        int input = 0;
        int total = 0;

        /* print preamble to the screen */
        printf("CSSE1000 PRAC 7 PREP TASK 1 - ACCUMULATOR\n");
        printf("Justin Mancinelli - 42094353\n");

        /* start an infinite loop */
        while(1){
                /* add the input to the total */
                total += input;
                /* show the user the total */
                printf("the accumulated total is %d\n", total);

                /* as the user for a new number */
                printf("enter a number: ");
                scanf("%d", &input);

                /* check if user wants to exit */
                if(input == 0)
                        /* break out of infinite loop */
                        break;
        }

        /* return success to the OS */
        return 0;
}
```

prac7-2.c

```c
/* FILE: prac7-2.c */

#include <stdio.h>

int main(void)
{
        int digit1, digit2;

        /* print preamble to the screen */
        printf("CSSE1000 PRAC 7 PREP TASK 1 - COMBINATION LOCK\n");
        printf("Justin Mancinelli - 42094353\n");

        /* start an infinite loop */
        while(1) {
                /* check whether the combination is correct */
                if(digit1 == 8 && digit2 == 3)
                        printf("\nthe lock is currently open\n\n");
                else
                        printf("\nthe lock is currently closed\n\n");

                /* allow user to enter digts */
                printf("enter the first digit: ");
                scanf("%d", &digit1);
                printf("enter the second digit: ");
                scanf("%d", &digit2);

                /* show the user their code */
                printf("\nthe code you entered is %d%d\n", digit1, digit2);

                /* check if the user wants to exit */
                if(digit1 == 0 && digit2 == 0)
                        /* break out of infinite loop */
                        break;
        }

        /* return success to the OS */
        return 0;
}
```

Prac 7 4209453

Procedure

Task 1 — Bit Inversion

① Write C program to be used with AT90S8515 which will take 8 bits of data from PORTC, invert the number then output the result to PORTB. (Invert ↔ 2's comp)

② Simulate — Build and debug in AVR Studio.
Build is a success and the simulation is as expected.

③ Test — PINC = 0x00, PORTB = 0x00
PINC = 0x01, PORTB = 0xFF
PINC = 0x7F, PORTB = 0x81
PINC = 0xFF, PORTB = 0x01
} correct

④ Download to board, wire SWs to PINC, LEDs to PORTB
Result — Success

⑤ Test — Same sequence as simulation test ③
Result — SW = 00000000, LED = 00000000 | SW = 01111111, LED = 10000001
SW = 00000001, LED = 11111111 | SW = 11111111, LED = 00000001

Task 2 — Accumulator

① Write a C program to be used with AT90S8515 which will take 8 bits of data from PORTC and add that input to a running total starting at 0 after the user presses a button wired to PINA[0]. The running total will be output to PORTB

② Simulate — Build and debug in AVR Studio
Build is a success as is the simulation.

③ Test — PORTB = 0x00, PINC = 0x01, Push Button
PORTB = 0x01, PINC = 0x0F, Push Button
PORTB = 0x10, PINC = 0x15, Push Button
PORTB = 0x25
} correct

④ Download to board, wire as described in ①
result — Success

⑤ Test — Same sequence as simulation test ③
Result — SW = 00000001, LED = 00000001 | SW = 00001111, LED = 00010000 | SW = 00010101, LED = 00100101

Prac 7 4209435 3

Task 3 – Combination Lock

① Write a C program to be used with AT90S8515 which will take 4 bits of data from PINC[3-0] and output to PORTA[3-0] when the pushbutton on PINC[4] is pressed. When the button is pressed again, PORTA[3-0] → PORTB[3-0] and PINC[3-0] → PORTA[3-0]. If the displayed code is correct, PORTD[0] = 1, if incorrect, PORTD[1] = 1.

② Simulate – Build and debug in AVR Studio
Build is a success as well as simulation

③ Test – At initialisation, PORTA = 0x00, PORTB = 0x00, LED closed
Set PORTC = 0x03 then push button
PORTA = 0x03, PORTB = 0x00, LED closed
Set PORTC = 0x08 then push button
PORTA = 0x08, PORTB = 0x03, LED open

④ Download to board, wire as described in ① with PINC[3-0] to switches, PINC[4] to a button, PORTA[3-0] to the left Hex display and PORTB[3-0] to the right Hex display and PORTD[0,1] to LEDs respectively.
Result – Successful write and initialization.

⑤ Test – Same sequence as simulation test ③
Result – Hex display shows correct sequence of numbers, "83" and with the open LED active at the final step. Each step before yeilded the expected output corresponding exactly to the simulation.

Table 1

Inputs	Expected	Actual
PORTC = 0x03 Button Pushed	PORTA = 0x03 PORTB = 0x00 PORTD = 0x02	Hex = '30', LED closed
PORTC = 0x08 Button Pushed	PORTA = 0x08 PORTB = 0x03 PORTD = 0x01	Hex = '83', LED opened
PORTC = 0x00 Button Pushed	PORTA = 0x00 PORTB = 0x08 PORTD = 0x02	Hex = '08', LED closed
PORTC = 0x00 Button Pushed	PORTA = 0x00 PORTB = 0x00 PORTD = 0x01	Hex = '00', LED closed

Prac 7

Tutor Task

① Modify the combination lock such that
PORTA[3-0] = LH Hex, PORTA[7-4]=RH Hex
PORTC[6-3] = Toggle Switch input, PORTC[0] = PB input
PORTC[1] = open LED, PORTC[2] = closed LED

② Test

Input	Expected	Actual
Initial switch = 0x03 PUSH Button switch: 0x09 Push Button switch: 0x0f Push Button	PORTA = 0x00 PORTC[1]=0, PORTC[2]=1 PORTA = 0x03 PORTC[1]=0, PORTC[2]=1 PORTA = 0x38 PORTC[1]=1, PORTC[2]=0 PORTA = 0x3F PORTC[1]=0, PORTC[2]=1	Hex = '00', LED: closed Hex = '30', LED: closed Hex = '83', LED: open Hex = 'F8', LED: closed

③ Conclusion

It is much easier to modify code in C than it is to do in assembly. The C compiler is able to choose appropriate registers so the programmer can define suitable variables for ease of programming logic.

Our simulation was a success on the second try (first try had semicolons missing). As always, it is best to simulate the program before wiring.

task1.asm

; FILE: task1.asm