ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 351E
MICROCOMPUTER LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	3
EXPERIMENT DATE	:	29.11.2020
LAB SESSION	:	WEDNESDAY - 13.30
GROUP NO	:	3

GROUP MEMBERS:

150170013	:	ALİ KEREM YILDIZ
150170082	:	ARDA CÜCE
150170908	:	MUHAMMAD RIZA FAIRUZZABADI

FALL 2020-2021

Contents

1 INTRODUCTION [10 points]

In this experiment, we were asked to implement a code to a board design that can generate various led patterns and seven-segment display output with push-buttons. Unlike the previous experiments, this time we were prohibited to use the Arduino Programming Language (which is a framework built on C/C++) instead it is the first time we start to program it in Assembly Language. Using the Atmel ATmega328p datasheet and other source such as Atmel AVR-8 bit Instruction Set, we managed to implement the necessary operations and manipulations in Assembly Language to accomplish the tasks given. In addition to LEDs and pushbuttons, this time we also have 7-Segment displays in our circuit, connected through the CD4511B latch decoder driver. Lastly, the board used is Arduino Uno R3, designed and tinkered digitally with use of Autodesk Tinkercad Design Tool.

2 MATERIALS AND METHODS [40 points]

2.1 MATERIALS

Tools Used[1]

- Autodesk Tinkercad
- Latex (overleaf.com)
- Arduino Uno

This experiment is done via Autodesk Tinkercad Design Tool as well as the previous experiments. We used the 'Microcomputer Lab Intro' Lecture Notes as a reference regarding the overall structure of the board, alongside 'Atmel ATmega328P', CMOS BCD-to-7-Segment Latch Decoder Drivers' datasheets and 'Segment Digit LED Display' User Manual as additional references regarding the new concepts implemented in the experiment such as 7-Segment Display, and programming Arduino in Assembly Language.

2.2 METHODS

2.3 PART 1

In the first part of experiment, we were tasked to write a code that generates a specific sequence over LEDs (1 LED lit starting from the rightmost bit to the leftmost), similar with the one we did in the previous Part, with the only significant difference being the code's written in Assembly Language.

To achieve this task, first we needed to define ports as input or output in setup() function. LEDs are connected to Arduino digital pins 0 to 7, therefore it is mapped with PORTD. To set the input/output pins, we need DDRD - The Port D Data Direction Register. Since we couldn't directly implement the Port Manipulation in normal way, we coded it in the form of its specified address in the Microcontrollers, which is 0x0A. First we used instruction IN, that Loads data from I/O Space (Which is the DDRD), into register R16. Then, we used instruction ORI that performs a logical OR operation between Rd and the '0b11111111', setting all digital pins as outputs (OR allows us to implement this without changing the bits value, safer approach). Lastly, we used instruction OUT to Store the data from register R16 to I/O Space (0x0A/DDRD), finalizing the set up. In the loop part of the code, to accomplish the sequence given, we first used LDI instruction or 'Load Immediate' that loads an 8-bit constant directly to specified registers between 16 to 31. With that function we loaded two 8-bit constants that which bit is HIGH/1 in one leftmost, and rightmost in the other one, to 16 and 17 registers (r16. r17). Then we again used OUT instruction to Store the data from register R17 to PORTD (0x0B). Then we used CALL instrucftion to call to subroutine delay. Then, the sequence is done with shifting. The instructions is written within 'Incrementation'. First we used LSL (Logical Shift Left) instruction to shift all bits in r17. Then we used OUT instruction to store the new data of r17 to PORTD (0x0B), calling delay afterwards. Lastly, using CPSE instruction, a comparison is performed between r16 and r17. If there is no equality, it goes to the next instruction which is JMP, returning to the beginning of 'Incrementation'. This instruction is skipped if the previous CPSE is true (there is equality).

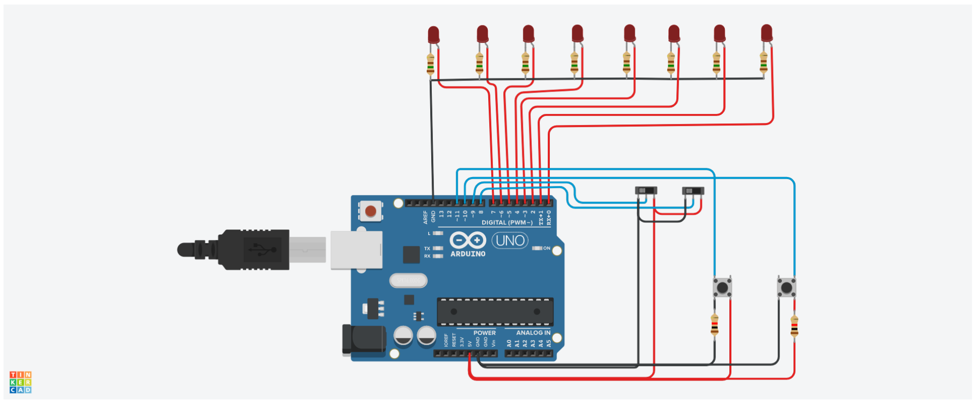

Figure 1: ...

2.4 PART 2

In this part of experiment we will code two-digit decimal counter using two seven segment display and push-buttons. One of the button should increase on every push, other one is for decreasing. We must implement this design in assembly format. First in start label we declare DDRC and DDRD ports's necessary pins as outputs by using IN(in to port) and OUT out to port) instructions. After that in loop label, we need to read PINB and PINC whose are showing that buttons are clicked or not. Then according to their value we must branch to proper function to do the asked operation. If first button is clicked PINB's 5. pin is going to HIGH and also if second button is clicked PINB's 6.pin is going to HIGH. Via using CP and BREQ instructions we can detect that situations and branch to related function. In countUp function first we look for ones digit. If this digit is 9, pattern should change , we need another function. By using LDI (loading a value to register) CP (comparing two register) and BREQ (if condition satisfy) system branchs to countUp2 to function. In countUp2 function we counting up numbers which have 9 value on their ones digit. Else via using BRNE function which means any value other than 9 in ones digit, we should do increment operation in this function. CountDown functions are as the same as countUp function design.

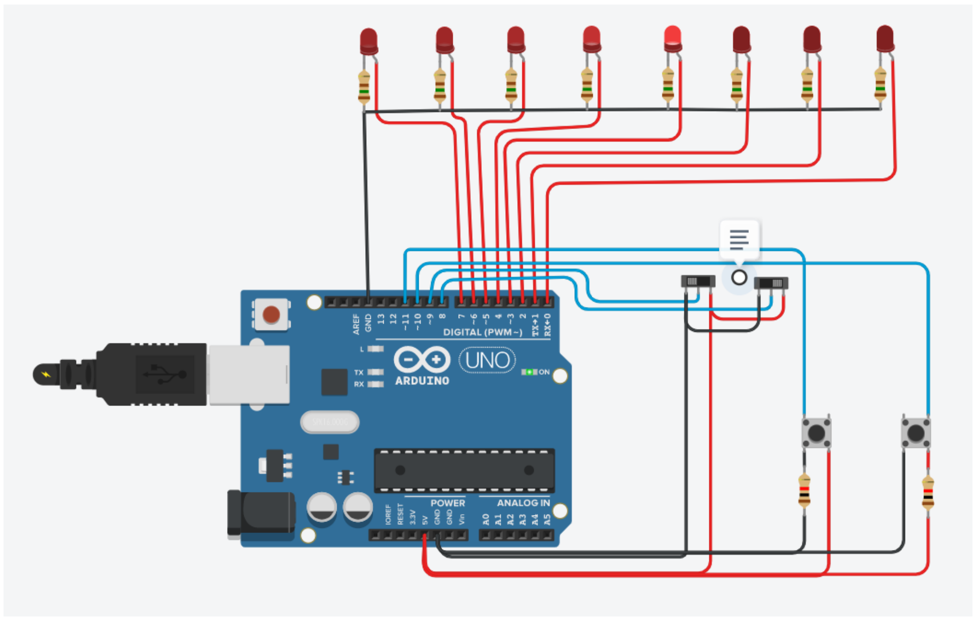

Figure 2: ...

3 RESULTS [15 points]

The first two tasks given in the experiment are accomplished, however there are factors that differed it greatly with the previous experiments. We are used to accomplish the task with using only registers, so implementing it in Assembly Language became slightly clearer. Using the datasheets and manual as a reference, we managed to find instructions for each functionalities we have in mind to accomplish the task. For the first time we got to tinker with 7-Segment Digit LED Display, alongside its decoder.

4 DISCUSSION [25 points]

In this experiment, we learnt to program Arduino with Assembly Language, and we got a clearer idea on how to implement the functionalities in form of instructions. We also for the first time got to experiment with 7-Segment Displays. In part 2 we were supposed to not increment or decrement when button is held down however when we tried to implement this functionality it gave compiler error, so we turned those lines responsible into comments to be discussed with the instructor.

5 CONCLUSION [10 points]

Tasks given in this experiment would normally be really simple to implement using c language, however when using assembly even simple tasks are really confusing to implement. Compiler gave hard to understand and fix errors very often and our code behaved very unexpected from time to time. We have managed to complete only the first two tasks however even that proved to be extremely time consuming.

REFERENCES

[1] Istanbul Technical University Department of Computer Engineering. Blg 351e micro-computer laboratory experiments booklet, Fall 2020-2021.