

Class:	CPE301L		Semester:	Spring 2024
Points		Document author:	Yanai Avila	
		Author's email:	avilay1@unlv.nevada.edu	
		Document topic:	Postlab 1: C and Assembly Programming	
Instructor's comments:				

1. Introduction / Theory of Operation

This lab allows one to work with both the C language and assembly. One uses an Arduino online environment to review the basics of C programming, and OnlineGDB to review the basics of assembly. It tests one's abilities by implementing 6 different circuits on Tinkercad using variables, loops, switch statements, and other conditional statements. Additionally, it allows one to learn how to work with registers in assembly using the debug tools on OnlineGDB.

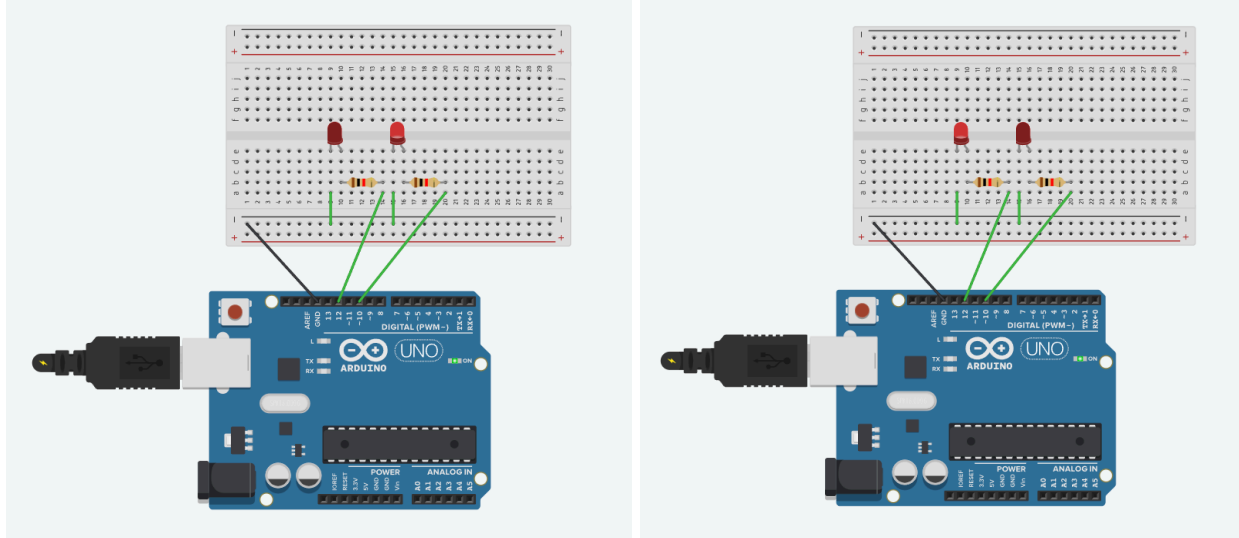
2. Prelab

No prelab

3. Results of Experiments

Experiment 1: LEDs inverse blinking ([Link to Tinkercad circuit and code](#))

Screenshot of the circuit:

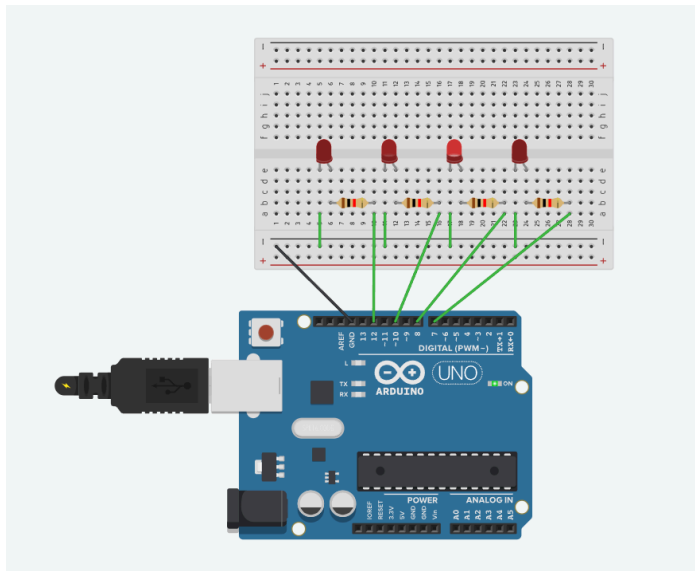


Code:

```
1 // Experiment 1: LEDs blink in inverse
2
3 int LED1 = 10;
4 int LED2 = 12;
5
6
7 void setup()
8 {
9   pinMode(LED1, OUTPUT);
10  pinMode(LED2, OUTPUT);
11 }
12
13 void loop()
14 {
15   // LED1 on, LED2 off
16   digitalWrite(LED1, HIGH);
17   digitalWrite(LED2, LOW);
18   // Wait for 1000 millisecond(s)
19   delay(1000);
20   // LED 1 off, LED 2 on
21   digitalWrite(LED1, LOW);
22   digitalWrite(LED2, HIGH);
23   // Wait for 1000 millisecond(s)
24   delay(1000);
25 }
26
```

Experiment 2: LEDs blinking in sequence ([Link to Tinkercad circuit and code](#))

Screenshot of the circuit:



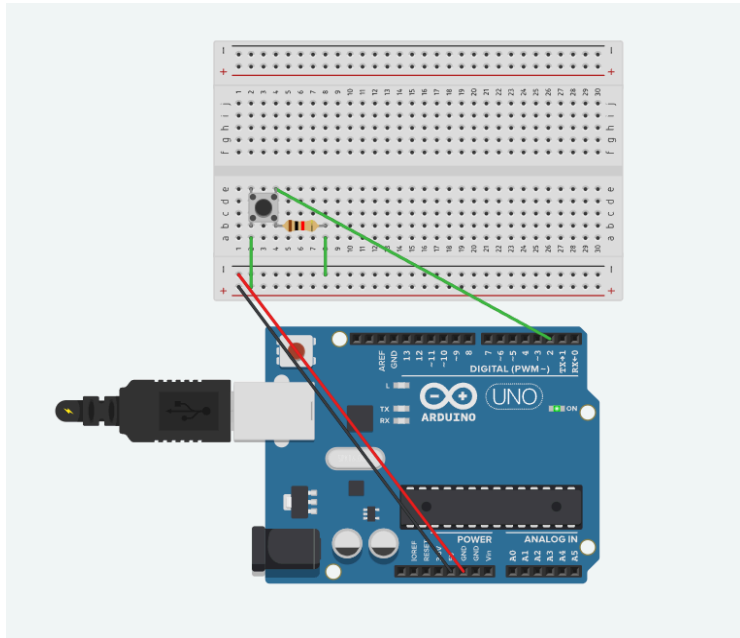
Code:

```
1 // Experiment 2: One of many LED operation
2
3 int LED0 = 12;
4 int LED1 = 10;
5 int LED2 = 8;
6 int LED3 = 7;
7
8 // Will indicate which of LEDs is on
9 int pos = 0;
10
11 void setup()
12 {
13   pinMode(LED0, OUTPUT);
14   pinMode(LED1, OUTPUT);
15   pinMode(LED2, OUTPUT);
16   pinMode(LED3, OUTPUT);
17 }
18
19 void loop()
20 {
21   // Switch statement to make LEDs light up one by one
22   switch(pos++ % 4)
23   {
24     case 0:
25       digitalWrite(LED0, HIGH);
26       digitalWrite(LED1, LOW);
27       digitalWrite(LED2, LOW);
28       digitalWrite(LED3, LOW);
29       break;
30     case 1:
31       digitalWrite(LED0, LOW);
32       digitalWrite(LED1, HIGH);
33       digitalWrite(LED2, LOW);
34       digitalWrite(LED3, LOW);
35       break;
36     case 2:
37       digitalWrite(LED0, LOW);
38       digitalWrite(LED1, LOW);
39       digitalWrite(LED2, HIGH);
40       digitalWrite(LED3, LOW);
41       break;
```

```
42     case 3:
43         digitalWrite(LED0, LOW);
44         digitalWrite(LED1, LOW);
45         digitalWrite(LED2, LOW);
46         digitalWrite(LED3, HIGH);
47         break;
48     default:
49         digitalWrite(LED0, LOW);
50         digitalWrite(LED1, LOW);
51         digitalWrite(LED2, LOW);
52         digitalWrite(LED3, LOW);
53         break;
54     }
55
56     delay(500); // Wait|
57 }
58
```

Experiment 3: Pushbutton ([Link to Tinkercad circuit and code](#))

Screenshot of the circuit:



Code:

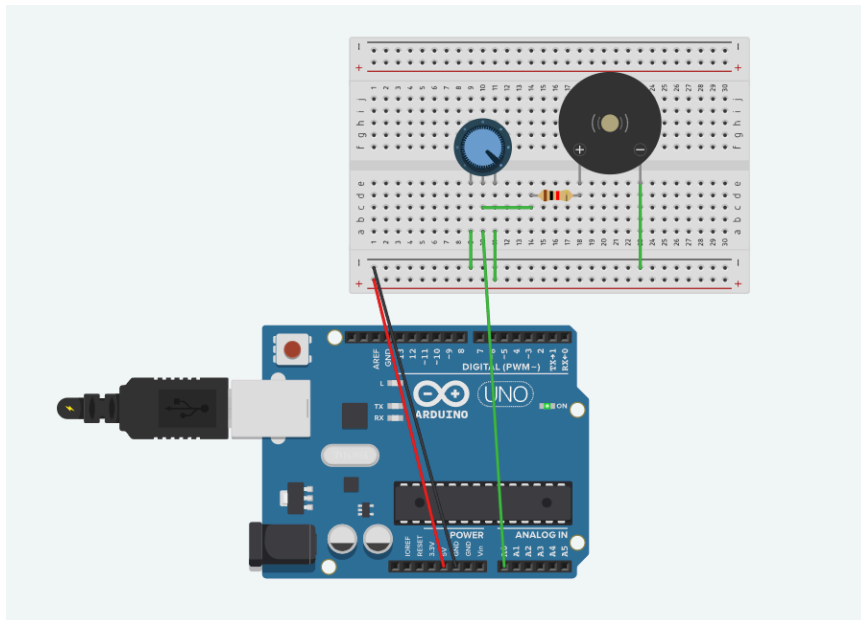
```
1 //Experiment 3: Pushbuttons
2
3 void setup()
4 {
5     pinMode(2, INPUT);
6     Serial.begin(9600);
7 }
8
9 void loop()
10 {
11     // If button is pressed, then display
12     // "Button pressed" in serial monitor
13     if(digitalRead(2) == HIGH){
14         Serial.println("Button pressed");
15         delay(100);
16     }
17 }
18
```

Serial monitor output:



Experiment 4: Changing buzzer with potentiometer ([Link to Tinkercad circuit and code](#))

Screenshot of the circuit:



Code:

```
1 // Experiment 4: Analog read
2
3 void setup()
4 {
5   pinMode(A0, INPUT);
6   Serial.begin(9600);
7 }
8
9 // To store previous/current values of potentiometer
10 int pot_prev = 0;
11 int pot_cur;
12
13 void loop()
14 {
15   pot_cur = analogRead(A0);
16
17   // Display potentiometer value in the serial monitor,
18   // but only if the potentiometer value changes
19   if(pot_cur != pot_prev){
20     Serial.println(pot_cur);
21     pot_prev = pot_cur;
22   }
23 }
```

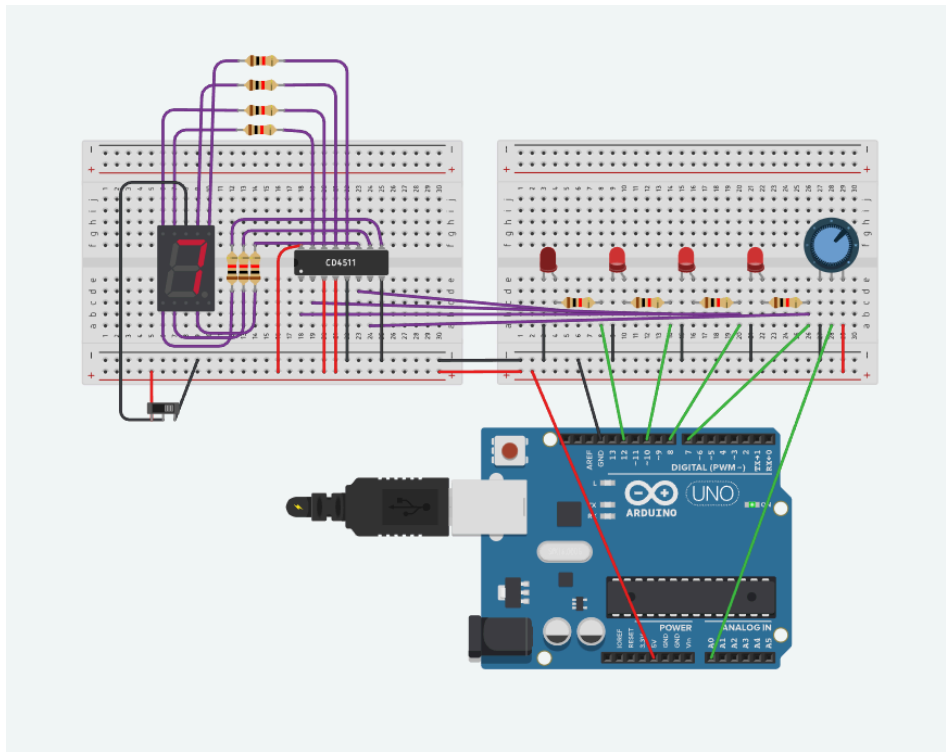
Serial monitor output:



NOTE: Serial monitor shows highest adc value. This is where the buzzer was the loudest.

Experiment 5: 7-segment display ([Link to Tinkercad circuit and code](#))

Screenshot of the circuit:



Code:

```
1 // Experiment 5: 7-segment
2
3 int LED0 = 12;
4 int LED1 = 10;
5 int LED2 = 8;
6 int LED3 = 7;
7
8 void setup()
9 {
10   pinMode(LED0, OUTPUT);
11   pinMode(LED1, OUTPUT);
12   pinMode(LED2, OUTPUT);
13   pinMode(LED3, OUTPUT);
14   pinMode(A0, INPUT);
15   Serial.begin(9600);
16 }
17
18 // digit variable
19 int digit = 0;
20
21 //// To store previous/current values of potentiometer
22 int pot_prev = 0;
23 int pot_cur;
24
```

```

25 void loop()
26 {
27     // Divide the range of potentiometer into 10 ranges
28
29     if (analogRead(A0) > 0 && analogRead(A0) < 110) {
30         digit = 1;
31     }
32     else if (analogRead(A0) > 109 && analogRead(A0) < 220) {
33         digit = 2;
34     }
35     else if (analogRead(A0) > 219 && analogRead(A0) < 330) {
36         digit = 3;
37     }
38     else if (analogRead(A0) > 329 && analogRead(A0) < 440) {
39         digit = 4;
40     }
41     else if (analogRead(A0) > 439 && analogRead(A0) < 550) {
42         digit = 5;
43     }
44     else if (analogRead(A0) > 549 && analogRead(A0) < 660) {
45         digit = 6;
46     }
47     else if (analogRead(A0) > 659 && analogRead(A0) < 770) {
48         digit = 7;
49     }
50     else if (analogRead(A0) > 769 && analogRead(A0) < 880) {
51         digit = 8;
52     }
53     else{digit = 9;}
54
55     pot_cur = analogRead(A0);
56
57     if(pot_cur != pot_prev){
58         Serial.println(digit);
59         // Passes digit into 7 seg function
60         control7SEG(digit);
61         pot_prev = pot_cur;
62     }
63     delay(100);
64 }
65

```

```

66 void control7SEG(int num){
67     // Set LED0-LED3 with the value of current number in BCD
68     digitalWrite(LED0, bitRead(num, 3));
69     digitalWrite(LED1, bitRead(num, 2));
70     digitalWrite(LED2, bitRead(num, 1));
71     digitalWrite(LED3, bitRead(num, 0));
72 }
73

```


Experiment 6: Assembly

Code and debugging:

The screenshot displays a debugger interface with the following components:

- Assembly Code (main.S):**

```
1 # Experiment 6: Assembly
2
3 .data
4 .text
5 .global main
6
7 main:
8     xor %rax, %rax # initialize ax with zero
9     add $6, %rax    # add 6 to ax
10    xor %rcx, %rcx  # initialize cx with 0
11
12
13 outer:
14     add $5, %rbx    # add 5 to bx - set
15
16 inner:
17     inc %rcx        # increment cx
18     dec %rbx        # decrement bx
19     jnz inner       # jump to back up to inner if not 0
20
21     dec %rax        # decrement ax
22     jnz outer       # jump to outer if not 0
23     ret
24
```
- Register List:**

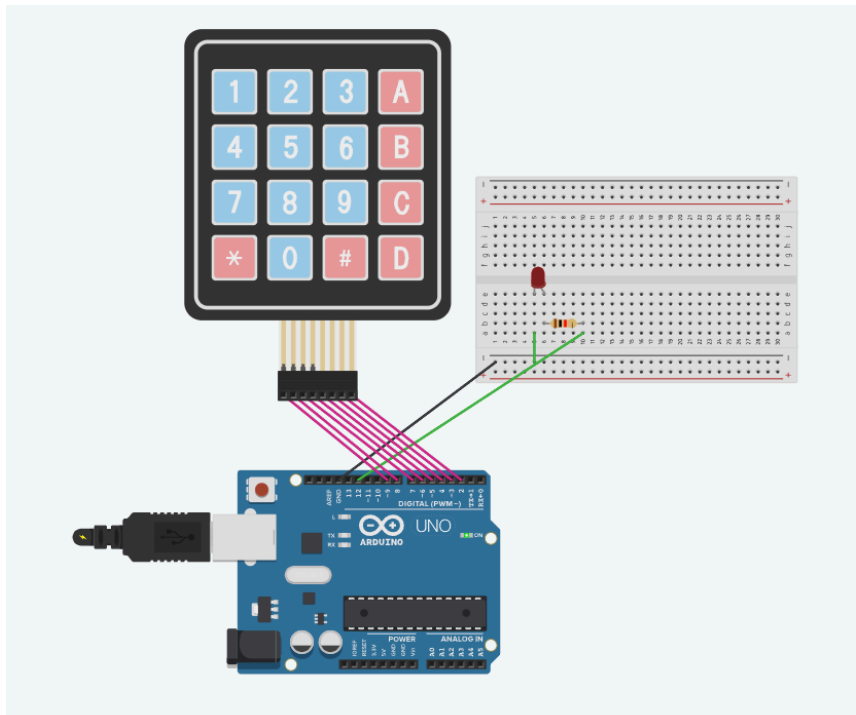
Register	Value
rbx	0x1 1
rcx	0x1e 30
rdx	0x7ffffffce8 140737488350440
rsi	0x7ffffffcd8 140737488350424
rdi	0x1 1
rbp	0x1 0x1
rsp	0x7ffffffebc8 0x7ffffffebc8
r8	0x7ffff7a2f10 140737353756432
r9	0x7ffff7fc9040 140737353912384
r10	0x7ffff7fc3908 140737353890056
r11	0x7ffff7fde680 140737354000000
r12	0x7ffffffecd8 140737488350424
r13	0x55555555129 93824992235817
r14	0x555555557df8 93824992247288
r15	0x7ffff7ffd040 140737354125376
rip	0x5555555513a 0x5555555513a <inner+3>
eflags	0x206 [PF IF]
- Debug Console:**

```
not 0
(gdb) next
Breakpoint 1, inner () at main.S:17
17     inc %rcx        # increment cx
(gdb) next
18     dec %rbx        # decrement bx
(gdb)
```

NOTE: RCX register contains the value of RAX*RBX right before program termination. In the list of registers, RCX is 30, which is RAX(6) x RBX(5).

Experiment 7: Keypad ([Link to Tinkercad circuit and code](#))

Screenshot of the circuit:



Code:

```
1 //Experiment 7: Keypad
2
3 #include <Keypad.h>
4
5 // # of rows and columns for keypad
6 const byte r = 4;
7 const byte c = 4;
8
9 // Keypad connections -> arduino terminals
10 byte rpins[r] = {9,8,7,6};
11 byte cpins[c] = {5,4,3,2};
12
13 // map
14 char keys[r][c]=
15 {
16   {'1', '2', '3', 'A'},
17   {'4', '5', '6', 'B'},
18   {'7', '8', '9', 'C'},
19   {'*', '0', '#', 'D'}
20 };
21
22 // Instance of keypad
23 Keypad myKeypad = Keypad(makeKeymap(keys), rpins, cpins, r, c);
24
25
26 void setup()
27 {
28   Serial.begin(9600);
29 }
30
```

```

31 void loop()
32 {
33     char keypressed = myKeypad.getKey();
34     KeyState state = myKeypad.getState();
35
36     // when key is pressed, display its value in
37     // the serial monitor.
38     if (keypressed != NO_KEY && keypressed != 'D')
39     {
40         Serial.println(keypressed);
41     }
42
43     // When D is pressed, apply new line to the serial monitor
44     if (keypressed == 'D')
45     {
46         Serial.println("\n");
47     }
48
49
50     // light up LED for the time the key is pressed,
51     // off when not
52     if(state == HOLD)
53     {
54         digitalWrite(12, HIGH);
55     }
56     else{
57         digitalWrite(12, LOW);
58     }
59 }
60

```

Serial monitor output:



NOTE: The space in the serial monitor is due to the newline when pressing “D”

4. Answers to questions

1. What is the technical difference between analog and digital input of Arduino board?

An analog input can be any number of values within a range (0V to 5V in most cases) while a digital signal can only be two values (HIGH and LOW). Arduino has a built-in analog-to-digital converter (ADC) that can convert analog voltage into digital values between 0 and 1023.

2. What is a bitwise operation?

A bitwise operation is an operation that involves working with and manipulating individual bits. Examples include bitwise AND, bitwise OR, bitwise XOR, left shift, right shift, and binary ones complement.

5. Conclusions & Summary

This lab was a needed review of the C language and a good introduction to assembly. I was able to implement 6 circuits on Tinkercad which allowed me to practice using variables, loops, and conditional statements in C. It allowed me to work with LEDs, pushbuttons, potentiometers, and even a keypad, all while learning about analog and digital signals. I was also able to use OnlineGDB to practice my assembly and get a feel for how the debugging process works.

The only experiment I had problems with was Experiment 5 with the 7-segment display. At one point, my potentiometer and LEDs were working fine, but the 7-segment display was not. I knew my connections with the display were correct because it displayed a 0, but it would not change with the potentiometer. I did some troubleshooting by messing around with the 4 wires connected to the inputs of the CD4511 decoder. I realized that I had the wires on the wrong side of the resistor, so it was not reading the values. Once I made that switch, the 7-segment display worked as desired.