นาย ชญานิน ยอดใย 5535512010

SUBJECT:

NO: DATE: / /

LAB 3HB05

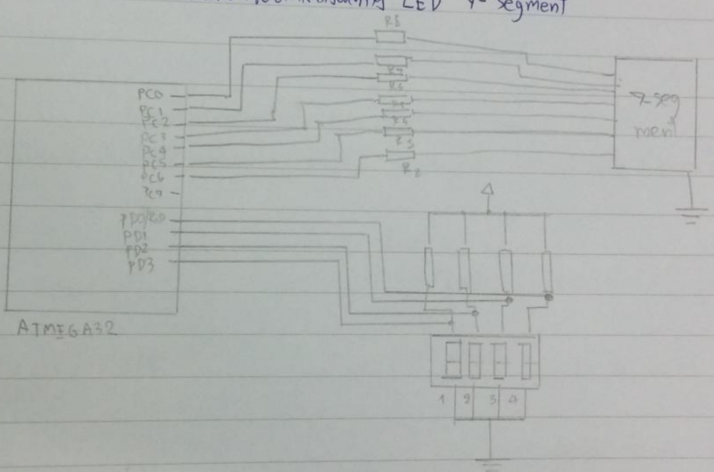
1. วัตถุประสงค์

1.1 เพื่อให้นักศึกษาได้เรียนรู้วิธีการเขียนโปรแกรมประยุกต์ใช้งานไมโครคอนโทรลเลอร์ AVR ด้วยภาษา แอสเซมบลี้และภาษาซี

1.2 เพื่อให้นักศึกษาได้เรียนรู้เทคนิคการดีบักโปรแกรมภาษาแอสเซมบลี้และภาษาซีก่อนที่จะนำโปรแกรมลงฮาร์ดแวร์จริง

1.3 เพื่อให้นักศึกษาทำการทดลองต่อวงจรเชื่อมต่อภายนอกกับบอร์ดไมโครคอนโทรลเลอร์ AVR และทดสอบการทำงานของโปรแกรมที่เขียนขึ้นบนฮาร์ดแวร์จริงได้

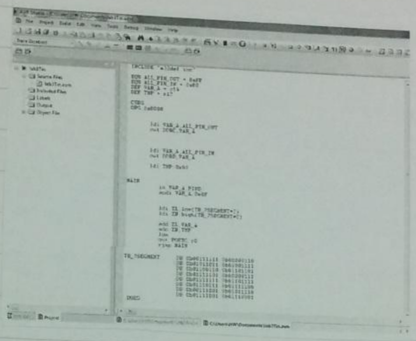
2. การทดลองที่ 1 กดสวิตช์ออกแสดงผลทาง LED 7-Segment



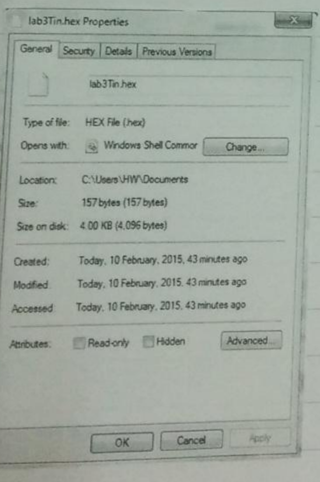
ต่อวงจรตามรูป

SUBJECT: NO: DATE: / /

code ภาษา C



อธิบายโค้ด โดยจะนำข้อมูลที่เข้ามาตรง PIND มา AND กับ 0x0F
และนำข้อมูลที่ได้ไปเปรียบเทียบใน ฟังก์ชัน LOOKUPTB ว่าตรงกับตัวใด ส่งข้อมูลออกให้ 7-segment
checkpoint 1.1



ตัวอย่าง

PIND = 0011

SWITCH_V = 0F & 0011

= 0000 1111 & 0011

= 0011

ส่งค่าไปเช็คใน LOOKUPTB

ตรงกับตัวที่ 3 ∴

จะส่งค่า 01001111

ซึ่งเมื่อนำมาแปลง แล้วจะได้เลข 3

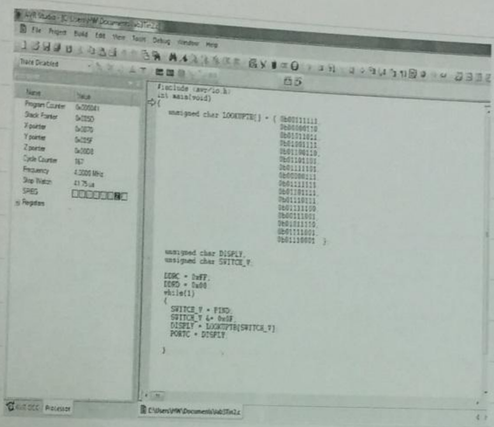
โดยข้อมูลจะเรียงจาก G→A ของขา 7-segment

นาย ชญานันท์ ยอดใย 5535512010

SUBJECT:

NO: DATE: / /

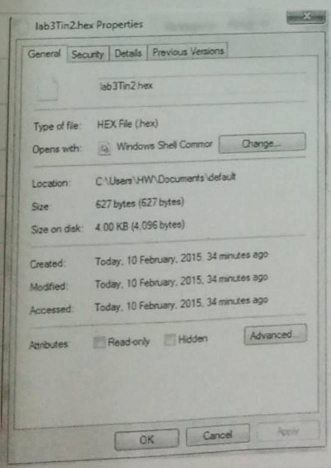
Code ภาษา Asm



อธิบายโค้ด

1. ประกาศ Input / output
   โดย input อยู่ที่ DDRD
   otput อยู่ที่ DDRC
2. นำ ข้อมูลที่ได้จาก Swith มา And กับ OxOF (คือ เอาแค่ค่าของ Switch 4 ตัว)
3. นำค่าที่ได้ไป บวกกับ Adress TB_Segment ที่เก็บไว้ใน Register z low และ hight
4. นำไปเขียนลง Ro ด้วยคำสั่ง lpm
5. นำออก ~~ทาง~~ PORTC

Checkpoint 1.2

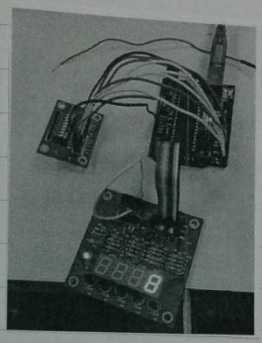


SUBJECT:

NO: DATE: / /

Check 2.3

ผลการทดลอง



จากรูป ใส่ input เป็น 0 ทั้ง 8 บิต ค่าที่แสดงจะเท่ากับ 8 ∴ เป็นไปตามที่โจทย์ต้องการ

คำถามท้ายการทดลอง

1. เปรียบเทียบ ข้อดี ข้อเสีย ของ code Asm และ ภาษา C ในรูป 1.10

Asm ข้อดี
1. ไฟล์มีขนาดเล็ก
2. ทำงาน compile เร็วกว่าเนื่องจากเป็นภาษาพื้นฐานในการคุม microcontroller

ข้อเสีย
1. โค้ดจะยาวกว่า และมีคำสั่งบางคำสั่งที่เข้าใจยาก

C ข้อดี
1. โค้ดเข้าใจง่าย
2. โค้ดจะสั้นกว่ามาก

ข้อเสีย
1. ไฟล์มีขนาดใหญ่
2. ความไวช้ากว่า เนื่องจากต้องทำการแปลงเป็น Asm ก่อนจะเขียนลง Board

Double A

การทดลองที่ 2 เขียนโปรแกรมหลังจำนวน Logic ต่ำ Check 2.1

Code ภาษา C

```
#include <avr/io.h>
int main(void)
{
  unsigned char LOOKUPTB[] = { 0b00111111,
                               0b00000110,
                               0b01011011,
                               0b01001111,
                               0b01100110,
                               0b01101101,
                               0b01111101,
                               0b00000111,
                             };
  unsigned char DISPLY;
  unsigned char SWITCH_V;
  int i;
  char count = 0;
  DDRC = 0xFF;
  DDRD = 0x00;
  while(1)
  {
    SWITCH_V = PIND;
    for(i=0;i<8;i++)
{       if((SWITCH_V & 1) == 0)
          count++;
        SWITCH_V = SWITCH_V >> 1;
}
    DISPLY = LOOKUPTB[count];
    PORTC = DISPLY;
}
}
```

อธิบายโค้ด

1. สร้าง LOOKUPTB ตามต้องการ
2. กำหนด input/output
   โดย input PIND
   output portC
3. นำ input ที่ได้ มา And กับ 1 ทีละ บิต
   ถ้า บิตใด And แล้วได้ 0 ให้ ออก
   ค่านั้น 1 ให้ครบทั้ง 8 บิต
   ยกตัวอย่าง

   1110 1101
   &
   1
   1 count = count

   ขึ้นข้อมูลไปทางซ้าย

   0111 0110 &
   1
   0 count + 1

   ทำอย่างนี้ให้ครบ 8 บิต
4. นำ ผ้าที่ได้ไป เปรียบเทียบใน LOOKUPTB
5. นำ ข้อมูลที่ ได้จาก LOOKUPTB ออกทาง
   PORT C

Double A

SUBJECT:

NO: DATE: / /

Code ASM Check 2.2

```
INCLUDE "m32def.inc"

EQU ALL_PIN_OUT = 0xFF
EQU ALL_PIN_IN = 0x00
DEF VAR_A = r16
DEF TMP = r17
DEF A = r18

CSEG
ORG 0x0000

      ldi VAR_A,ALL_PIN_OUT
      out DDRC,VAR_A


      ldi VAR_A,ALL_PIN_IN
      out DDRD,VAR_A

      ldi TMP,0x00
      ldi A,8

MAIN
       in VAR_A,PIND
      rol VAR_A
      adc count,TMP
       rol VAR_A
      adc count,TMP
       rol VAR_A
      adc count,TMP
       rol VAR_A
      adc count,TMP
       rol VAR_A
      adc count,TMP
       rol VAR_A
      adc count,TMP
       rol VAR_A
      adc count,TMP

      sub A,count

      ldi ZL,low(TB_7SEGMENT*2)
      ldi ZH,high(TB_7SEGMENT*2)

      add ZL,A
      adc ZH,TMP
      lpm
      out PORTC,r0
      rjmp MAIN

TB_7SEGMENT :   DB 0b00111111,0b00000110
                DB 0b01011011,0b01001111
                DB 0b01100110,0b01101101
                DB 0b01111101,0b00000111
                DB 0b01111111,0b01101111
                DB 0b01110111,0b01111100
                DB 0b00111001,0b01011110
                DB 0b01111001,0b01110001

DSEG
```

อธิบายโค้ด

1) กำหนดขา input / output
   input = DDRD
   output = DDRC
2) ทำการ ชิฟ ข้อมูลที่เข้ามา ไป ทาง ขวา และ เก็บค่าที่ ชิฟ ไปไว้ใน บิต carry
3) นำ ค่าที่เก็บไว้ที่บิตcarry มาบวกกับ TMP ทำทั้งหมด 8 รอบ
4) นำ A ที่เก็บค่า 8 - TMP จะได้จำนวนของ ตัวที่เป็น 0
5) ค่าจำนวนที่ได้ไปเทียบ กับ add ใน TB - 7SEGMENT ที่เก็บอยู่ใน Zlow, Zhigh
6) นำข้อมูลที่ได้ ออกทาง PORTC

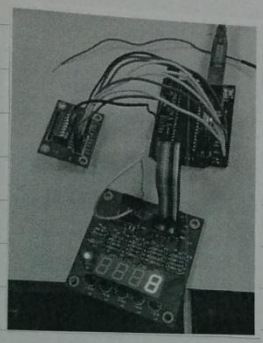
Double A

SUBJECT:

NO: DATE: / /

Check 2.3

ผลการทดลอง



จากรูป ใส่ input เป็น 0 ทั้ง 8 บิต ค่าที่ทดลองจะเท่ากับ 8 ∴ เป็นไปตามที่โจทย์ต้องการ

คำถามท้ายการทดลอง

1. เปรียบเทียบ ข้อดีข้อเสีย ของ code Asm และ ภาษา C ในรูป 1.10

Asm ข้อดี

1. ไฟล์มีขนาดเล็ก
2. ทำงาน compile เร็วกว่าเนื่องจากเป็นภาษาพื้นฐานในการคุม microcontroller

ข้อเสีย

1. โค้ดจะยาวกว่า และมีคำสั่งบางคำสั่งที่เข้าใจยาก

C ข้อดี

1. โค้ดเข้าใจง่าย
2. โค้ดจะสั้นกว่ามาก

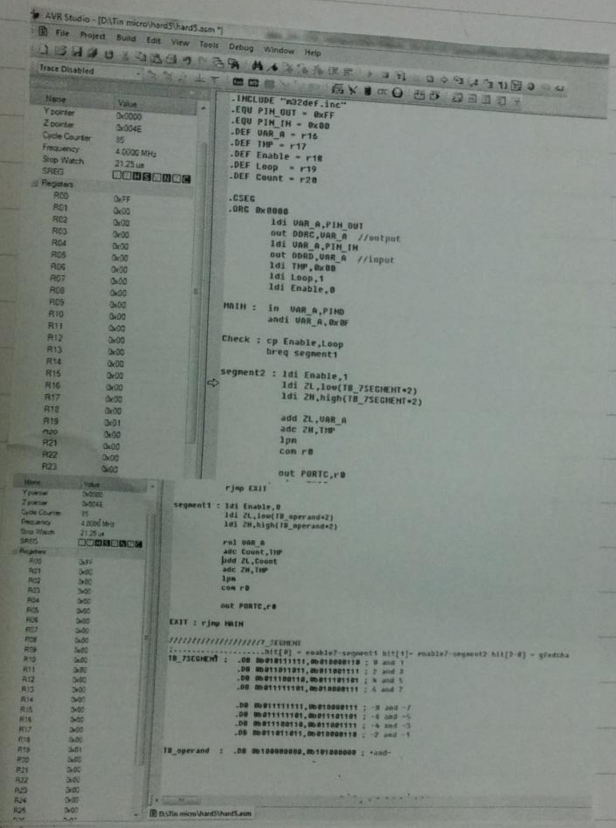
ข้อเสีย

1. ไฟล์มีขนาดใหญ่
2. คอมไพล์ช้ากว่า เนื่องจากต้องทำการแปลงเป็น Asm ก่อนจะเขียนลง Board

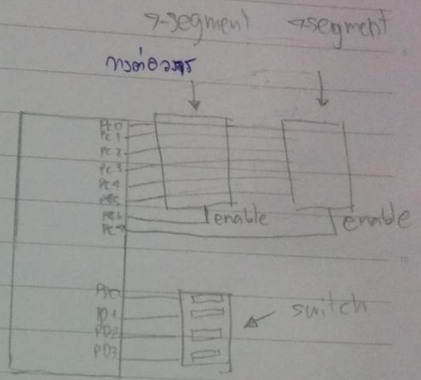
Double A

2) เขียนโค้ด ASM และภาษา C เพื่ออ่านค่าจาก switch และแสดงผ่าน 7-segment 2 หลัก โดยจะแสดงตัวเลข แบบมีเครื่องหมาย

Code ASM



```
.INCLUDE "m32def.inc"
.EQU PIN_OUT = 0xFF
.EQU PIN_IN = 0x00
.DEF VAR_A = r16
.DEF TMP = r17
.DEF Enable = r18
.DEF Loop = r19
.DEF Count = r20

.CSEG
.ORG 0x0000
        ldi VAR_A,PIN_OUT
        out DDRC,VAR_A  //output
        ldi VAR_A,PIN_IN
        out DDRD,VAR_A  //input
        ldi TMP,0x00
        ldi Loop,1
        ldi Enable,0

MAIN :  in  VAR_A,PIND
        andi VAR_A,0x0F

Check : cp Enable,Loop
        breq segment1

segment2 : ldi Enable,1
           ldi ZL,low(TB_7SEGMENT*2)
           ldi ZH,high(TB_7SEGMENT*2)

           add ZL,VAR_A
           adc ZH,TMP
           lpm
           com r0

           out PORTC,r0
        rjmp EXIT

segment1 : ldi Enable,0
        ldi ZL,low(TB_operand*2)
        ldi ZH,high(TB_operand*2)

        rol VAR_A
        adc Count,TMP
        add ZL,Count
        adc ZH,TMP
        lpm
        com r0

        out PORTC,r0

EXIT : rjmp MAIN
```



อธิบาย

1. โดย รับค่าจาก switch 4 ตัว
2. แยกการทำงาน ของ segment 2 ตัวด้วย ตัวแปร enable
3. ให้สลับกัน ทำงาน เร็วมากจนตา เรามองไม่ทัน
4. โดย ถ้าข้อมูล มากว่า 0111 จะให้การติดลบ ใน segment หน้า คือถ้า bit แรกเป็น 1 ให้ แสดงลบ ที่ segment แรก
5. ส่วนตัว segment หลังให้ แสดง ตามตัว TB_segment ไปเลย เพราะถ้า เกิด 8 ก็จะให้ แสดง 7-1 ตามลำดับ

NO: .... DATE: ..../..../....

12010

Code C

```
#include<avr/io.h>
int main(void)
{  unsigned char LOOKUPTB[] = {  0b010111111,//0
                                 0b010000110,//1
                                 0b011011011,//2
                                 0b011001111,//3
                                 0b011100110,//4
                                 0b011101101,//5
                                 0b011111101,//6
                                 0b010000111,//7

                                 0b011111111,//8
                                 0b010000111,//7
                                 0b011111101,//6
                                 0b011101101,//5
                                 0b011100110,//4
                                 0b011001111,//3
                                 0b011011011,//2
                                 0b010000110,//1
                              }

   unsigned char ENABLE_CH[] = { .0b010000000,//+
                                  0b011000000,//-
                               }

  unsigned char DISPLY;
  unsigned char SWITCH_V;
  unsigned char enable = 0;
  DDRC = 0xFF;
  DDRD = 0x00;
  while(1)
   {

     SWITCH_V = PIND;
     SWITCH_V &=0x0F;

     if(enable = 1)
         {.
          enable = 0;
          SWITCH_V =  SWITCH_V >> 3;
          SWITCH_V = SWITCH_V&1;

          DISPLY = ENABLE_CH[SWITCH_V];
         }
    else if(enable = 0)
        {
          enable = 1;
          DISPLY = LOOKUPTB[SWITCH];
          PORTC = DISPLY;
        }
   }

}
```

D:\Tin micro\hardc5\hardc5.c

การทำงาน

1. สร้าง ฟังก์ชั่น LOOKUPTB และ ENABLE เพื่อให้ สามารถกำหนดค่าไป หา 7-sement ตามที่เราต้องการได้
2. กำหนดให้ DDRC เป็น output DPRD เป็น input
3. ให้ while (1) ในการให้ segment 2 ตัว สลับกันแสดง ผล ตาม ฟังก์ชั่นนั้นๆ

Double A