PC____ ID_____ Name _____.

# การทดลองที่ 03: Addressing Mode and Arduino Digital IO

## 3.1 วัตถุประสงค์

1. เพื่อศึกษาการเข้าถึงข้อมูลแบบอ้างตำแหน่งโดยตรง (direct addressing mode)
2. เพื่อศึกษาการเข้าถึงข้อมูลแบบอ้างตำแหน่งโดยอ้อม (indirect addressing mode)
3. เพื่อศึกษาการเข้าถึงข้อมูลแบบอ้างตำแหน่งรีจิสเตอร์ (register addressing mode)
4. เพื่อศึกษาการเข้าถึงข้อมูลแบบอ้างตำแหน่งใช้ทันที (immediate addressing mode)
5. เพื่อศึกษาการใช้งานอินพุต เอาต์พุตแบบดิจิตอลของ Arduino ESP32

## 3.2 อุปกรณ์ทดลอง

1. เครื่องคอมพิวเตอร์ส่วนบุคคลสำหรับจำลองการทำงานของ MCS-51
2. ESP32 Board

## 3.3 เนื้อหา

1. MCS-51 Memory Map

| Add | D |
|---|---|
| FFFF | xx |
| .. | |
| 0000 | xx |
| 8002 | xx |
| 8001 | xx |
| 8000 | xx |
| 7FFF | xx |
| . | |
| . | |
| 0010 | 10 |
| 000F | 0F |
| 000E | 0E |
| 000D | 0D |
| 000C | 0C |
| 000B | 0B |
| 000A | 0A |
| 0009 | 09 |
| 0008 | 08 |
| 0007 | 07 |
| 0006 | 06 |
| 0005 | 05 |
| 0004 | 04 |
| 0003 | 03 |
| 0002 | 02 |
| 0001 | 01 |
| 0000 | 00 |

External Memory
Address Index = 2 Byte
Address 0000 to FFFF
ROM = Boot ,OS Area
RAM = Work Area
Port IO

Internal Memory
Address Index = 1 Byte
Address 00 to FF
SFR [80-FF]
RAM Indirect Access[80-FF]
RAM General Propose[30-7F]
RAM Bit Access[20-2F]
R0-R7 [00-07]

| Add | D |
|---|---|
| FF | xx |
| .. | |
| 85 | xx |
| 84 | xx |
| 83 | xx |
| 82 | xx |
| 81 | xx |
| 80 | xx |
| 7F | |
| .. | |
| 31 | |
| 30 | |
| 2F | |
| .. | |
| 21 | |
| 20 | |
| 1F | xx |
| .. | |
| 07 | xx |
| 06 | xx |
| 05 | xx |
| 04 | xx |
| 03 | xx |
| 02 | xx |
| 01 | xx |
| 00 | xx |

| Add | D |
|---|---|
| FF | yy |
| .. | |
| 85 | yy |
| 84 | yy |
| 83 | yy |
| 82 | yy |
| 81 | yy |
| 80 | yy |

2. MCS-51 Addressing Mode

| | Read | Write |
|---|---|---|
| External RAM, เลขห้อง 4 หลัก | **MOVX A,@DPTR** | **MOVX @DPTR,A** |
| Program Code | **MOVC A,@A+DPTR** | |
| Special RAM ,Address 80H-FFH | **MOV A,@R0**<br>**MOV A,@R1** | **MOV @R0,A**<br>**MOV @R1,A** |
| SFR, Address 80H-FFH | **MOV A, P0** | **MOV P0, A** |
| Address 30H – 7FH | **MOV A,40H**<br>**MOV A,@R0**<br>**MOV A,@R1** | **MOV 40H,A**<br>**MOV @R0,A**<br>**MOV @R1,A** |
| Address 20H – 2FH | **SETB 20H.1**<br>**MOV 20H.0,C**<br>**MOV 20H,A** | **CLR 20H.3**<br>**MOV C,21H.1**<br>**MOV A,21H** |
| Address 00H – 1FH, R0-R7 | | |

**<u>การเข้าถึงข้อมูลในหน่วยความจำ แบ่งได้ 5 แบบ</u>**

| | | | |
|---|---|---|---|
| - Immediate | แบบทันทีทันใด | **MOV R0,#40H** | **0 Step** |
| - Register | แบบผ่านรีจีสเตอร์ | **MOV A,R0** | **1 Step** |
| - Direct | แบบโดยตรง | **MOV A,40H** | **1 Step** |
| - Register indirect | แบบโดยอ้อม | **MOV A,@R0** | **2 Step** |
| - Indexed | แบบอินเด็กซ์ | **MOVC A,@A+DPTR** | **3 Step** |
| | หรือแบบโดยอ้อมผ่านค่าดัชนี | | |

**<u>การเข้าถึงข้อมูลแบบบิตสามารถทำได้กับ</u>**

- หน่วยความจำภายในตำแหน่ง 20H-2FH
- พอร์ตอินพุต-เอาต์พุต
- รีจีสเตอร์ที่สามารถเข้าถึงได้ระดับบิต
- การใช้งานอาจใช้คำสั่ง SETB, CLR เพื่อจัดการระดับบิตได้เลย

```
SETB  20H.7
CLR   EA
```

- หรือใช้การดำเนินการผ่าน C (Carry) ก็ได้

```
SETB  C
MOV   P0.3,C
```

**<u>การเข้าถึงหน่วยความจำภายในตำแหน่ง 80H - FFH ของ 8052</u>**

- หน่วยความจำภายในตำแหน่ง 80H-FFH จะมีใน 8032
- ยกตัวอย่างตำแหน่ง 90H จะทำหน้าที่เป็นพอร์ต 1 และเป็นหน่วยความจำภายในตำแหน่ง 90H

  การใช้คำสั่ง MOV 90H,#55H → หมายถึงทำให้ P1 = 55H

  หากต้องการให้หน่วยความจำภายในตำแหน่ง 90H เป็น 55H ต้องใช้ชุดคำสั่งดังนี้

```
MOV   R0,#90H     ; Point R0 to address 90H
MOV   A,#55H      ; Save Acc by Data 55H
MOV   @R0,A       ; Save Address @R0 with A
```

**<u>การเข้าถึงหน่วยความจำต่างๆ ในระบบไมโครคอนโทรเลอร์มีหลักการ</u>** ดังนี้

- ถ้าเป็นหน่วยความจำภายนอก ให้นึกถึงห้องพักที่มีเลขห้อง 4 หลัก เช่น **9000H**
- ถ้าเป็นหน่วยความจำภายใน ให้นึกถึงห้องพักที่มีเลขห้อง 2 หลัก เช่น **30H**
- กฎข้อที่ 1 ถ้าเป็นหน่วยความจำภายนอก (เลขห้อง 4 หลัก) ต้องเข้าถึงโดยอ้อมผ่านค่าดัชนี DPTR เท่านั้น

  เช่น Read Data **MOVX A,@DPTR**

  Write Data **MOVX @DPTR,A**

- กฎข้อที่ 2 ถ้าเป็นหน่วยความจำภายใน (เลขห้อง 2 หลัก) และห้องเบอร์ 80H-FFH ต้องเข้าถึงโดยอ้อมผ่านค่าดัชนี R0 หรือ R1 เท่านั้น

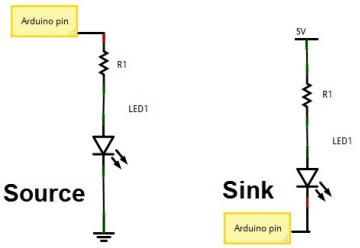
  เช่น Read Data **MOV A,@R0**

  Write Data **MOV @R0,A**

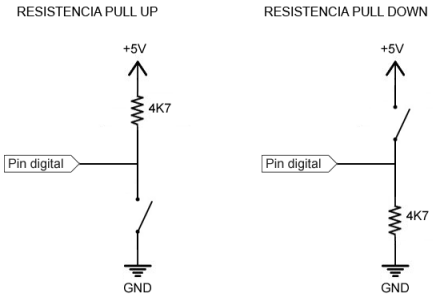
- กฎข้อที่ 3 ถ้าเป็นหน่วยความจำภายใน (เลขห้อง 2 หลัก) นอกจากห้องเบอร์ 80H-FFH สามารถเข้าถึงได้ตามแต่ความง่ายและความสะดวกในการเขียนโปรแกรม

## 3. Arduino Digital IO

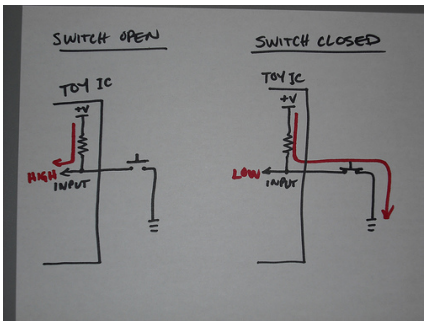
- Serial Monitor
  - Serial.begin(115200);
  - Serial.print(); or Serial.println();
- Digital Output
  - PinMode(**OUTPUT**);
  - digittalWrite(pin,**HIGH**);
  - digittalWrite(pin,**LOW**);




- Digital Input
  - PinMode(**INPUT**);
  - PinMode(**INPUT_PULLUP**);
  - Value = digitalRead(pin);




External Pull Up, External Pull Down




Internal Pull Up

PC____ ID_____ Name _____.

## 3.4 การทดลอง

1. เขียนโปรแกรม เพื่อย้ายค่าจากหน่วยความจำภายในตำแหน่ง 70H ถึงตำแหน่ง 7FH ไปเก็บไว้ที่หน่วยความจำภายนอก เริ่มจากตำแหน่ง 1000H เป็นต้นไป

```
;// ============== For  ==============
        ORG     8000H
        MOV     R0,#70H         ; Start Read
RAM
        MOV     R1,#10H         ; Stop-Start+1
        MOV     DPTR,#1000H     ; Save RAM
LOOP:   MOV     A,@R0
        MOCX    @DPTR,A
        INC     R0
        INC     DPTR
        DJNZ    R1,Loop
        JMP     $
```

```
;// ============== Repeat Until ==============
        ORG     8000H
        MOV     R0,#70H         ; Start Read
RAM
        MOV     DPTR,#1000H     ; Save RAM
LOOP:   MOV     A,@R0
        MOCX    @DPTR,A
        INC     R0
        INC     DPTR
        MOV     A,R0
        CJNE    A,#80H,Loop     ; Stop 7FH+1
        JMP     $
```

2. เขียนโปรแกรมเติมค่าในหน่วยความจำจากตำแหน่ง 1100H ถึงตำแหน่ง 111FH ให้มีค่าเริ่มจาก 0 และเพิ่มค่าขึ้นครั้งละ 2 จนครบ

3. เขียนโปรแกรมเพื่อเติมค่า(Fill) ในหน่วยความจำ

   แอดเดรส 1010H เก็บค่าข้อมูลที่จะนำไปเติม
   แอดเดรส 1020H, 1021H เก็บค่าแอดเดรสเริ่มต้น
   แอดเดรส 1030H, 1031H เก็บค่าแอดเดรสสุดท้าย

   เช่น เติมค่าจาก 9500H
   ก็ใส่ค่า 95H ที่ 1020H และ
   ใส่ค่า 00H ที่ 1021H ตามลำดับ

4. เขียนโปรแกรมเพื่อทำการย้ายข้อมูล 1 กลุ่ม(Block)

   แอดเดรส 1020H, 1021H เก็บค่าแอดเดรสเริ่มต้นของข้อมูลที่จะย้าย (Source)
   แอดเดรส 1030H, 1031H เก็บค่าแอดเดรสสุดท้ายของข้อมูลที่จะย้าย
   แอดเดรส 1040H, 1041H เก็บค่าแอดเดรสเริ่มต้นของข้อมูลปลายทาง (Destination)

5. เขียนโปรแกรมเพื่อตรวจค่าที่อยู่ใน Register B หากโดยเก็บผลการตรวจสอบไว้ที่ R0
   ถ้า A > #80H **ให้** B มีค่าเป็น #0AH **มิฉะนั้น** ให้ B มีค่าเป็น #0FH

```
        ORG     8000H
        CLR     C               ; Clear Carry Flag
        SUBB    A,#80H          ; A-80H -> Carry=1 if A>80 and Carry=0 if A <= 80H
        JC      A_More          ; Jump if carry set or Jump if Carry=1

A_Less: MOV     B,#0FH          ; Do if A <= 80H
        ;=======
        ;=======
        JMP     Connect

A_More: MOV     B,#0AH          ; Do if A > 80H
        ;=======
        ;=======
        JMP     Connect

Connect: NOP
        ;=======
        JMP     $
```

**If then Else**

6. เขียนโปรแกรมเพื่อหาค่าที่มากที่สุด(Maximum) กำหนดจุดเริ่มต้นโปรแกรมเอง

   แอดแดรส 1020H, 1021H เก็บค่าตำแหน่งเริ่มต้นของบล็อกในการค้นหา
   แอดแดรส 1030H, 1031H เก็บค่าตำแหน่งสุดท้ายของบล็อกในการค้นหา
   แอดแดรส 1042H เก็บค่ามากที่สุดที่พบ
   แอดแดรส 1040H, 1041H เก็บค่าตำแหน่งที่พบของค่าที่มากที่สุด

7. Arduino ESP32 - ทดสอบโปรแกรมไฟกระพริบ (Example→ Basic→Blink) เพิ่มเติมวงจรแล้วปรับแก้โปรแกรมให้วนรอบเพื่อแสดงไฟกระพริบ วนรอบ { RED→YLW→GRN→BLU→RED→YLW→GRN→BLU→...}

8. Arduino ESP32 - ทดสอบโปรแกรมส่งรหัส ASCII (Example Communication ASCII Table) แล้วปรับแก้โปรแกรมให้วนรอบแสดง "รหัสนักศึกษา, ชื่อ-นามสกุล, ขึ้นบรรทัดใหม่ ชิดซ้าย" จำนวน 5 บรรทัด แล้วแสดงข้อความ "My grade is A" แล้วหยุดการทำงานของโปรแกรม

9. Arduino ESP32 – ให้เขียนโปรแกรมรับค่าจากพอร์ตอนุกรมหากค่าที่รับมาเป็น 0 ให้ LED ที่ขา D0(LED Builin) ติดค้าง, หากเป็น 1 ให้ D2 กระพริบ 1 ครั้ง, หากเป็น 2 ให้ D2 กระพริบ 2 ครั้ง, ..., หากเป็น 9 ให้ D2กระพริบ 9 ครั้ง ค่าที่นอกจาก 0-9 โปรแกรมจะไม่สนใจ

10. Arduino ESP32 - ต่อสวิตช์โยกเข้ากับ Arduino เขียนโปรแกรมให้
    - โยกขึ้น ให้ D2 ติดกระพริบแบบปึ๊บ-ปึ๊บ
    - โยกลง ให้ D2 ติดค้าง

11. Arduino ESP32 - ต่อสวิตช์กด PB0 และ PB1 เข้ากับ ESP32 เขียนโปรแกรมให้
    - กด PB0 ให้ D2 ติดกระพริบแบบปึ๊บ-ปึ๊บ
    - กด PB1 ให้ ติดค้าง

12. Arduino ESP32 - ต่อสวิตช์โยก 4 ตัวให้ชื่อ SW3,SW2,SW1,SW0 เข้ากับ Arduino เขียนโปรแกรมตรวจสอบ
    - กำหนด SW3 คุมการกระพริบ {0=ค้าง 1=กระพริบ}
    - กำหนด SW2 คุมรูปแบบการกระพริบ {0=ติด,ดับ 1=ปึ๊บ-ปึ๊บ}
    - SW1, SW0 คุมความเร็ว
      - 00 ช้า
      - 01 เร็วกลาง
      - 10 เร็วมาก
      - 11 เร็วมากสุด

13. Arduino ESP32 ท้าทาย – แก้ไขโปรแกรมข้อ 11 ให้ใช้ปุ่มเดียวในการควบคุม โดยมีการทำงานคือ
    - กด PB0 ให้ D2 ติดกระพริบแบบ ปึ๊บ, ปึ๊บ, ปึ๊บ, ปึ๊บ, ...
    - กด PB0 ให้ ติดค้าง

14. Arduino ESP32 ท้าทาย – แก้ไขโปรแกรมข้อ 11 ให้ใช้ปุ่มเดียวในการควบคุม โดยมีการทำงานคือ
    - กด PB0 ให้ ดับ
    - กด PB0 ให้ ติดค้าง
    - กด PB0 ให้ D2 ติดกระพริบแบบ ปึ๊บ, ปึ๊บ, ปึ๊บ, ปึ๊บ, ...
    - กด PB0 ให้ D2 ติดกระพริบแบบ ปึ๊บ-ปึ๊บ, ปึ๊บ-ปึ๊บ, ปึ๊บ-ปึ๊บ, ปึ๊บ-ปึ๊บ, ...

## 3.5 คำถามท้ายการทดลอง

1. จงเขียนโปรแกรมภาษาแอสเซมบลีเพื่อดำเนินงานดังต่อไปนี้
    ก. กำหนด ORG ของโปรแกรมเป็น 8100H
    ข. เขียนข้อมูล 55H ไปยังหน่วยความจำภายในตำแหน่ง 44H
    ค. เขียนข้อมูล 66H ไปยังหน่วยความจำภายในตำแหน่ง 55H
    ง. ให้คัดลอกข้อมูลที่เก็บไว้ที่ตำแหน่ง 44H เพื่อเขียนลงสู่หน่วยความจำภายนอกที่ตำแหน่ง 8668H
    จ. ให้อ่านข้อมูลจากหน่วยความจำภายนอกที่ตำแหน่ง 8668H มาเก็บไว้ในรีจิสเตอร์ B
    ฉ. คัดลอกข้อมูลจากรีจิสเตอร์ B ไปยังรีจิสเตอร์ R0 R1 R2 และ R3
    ช. คัดลอกข้อมูลจากหน่วยความจำภายในที่อ้างอิงตำแหน่งด้วยข้อมูลในรีจิสเตอร์ R2 ไปเก็บไว้ในรีจิสเตอร์ R4 R5 R6 และ R7
    ซ. เมื่อดำเนินการรันโปรแกรมแล้ว ข้อมูลที่เก็บไว้ในรีจิสเตอร์ A B R0 R1 R2 R3 R4 R5 R6 R7 PC และ DPTR จะมีค่าเป็นเท่าใด

2. เขียนโปรแกรมเติมค่าในหน่วยความจำจากตำแหน่ง 9100H ถึงตำแหน่ง 910FH ให้มีค่าเริ่มจาก 0FFH และลดค่าลงครั้งละ 1 จนครบ

3. เขียนโปรแกรมเพื่อเติมค่า(Fill) ในหน่วยความจำภายในตำแหน่ง 70H ถึงตำแหน่ง 7FH ด้วยค่าที่เก็บในรีจีสเตอร์ B

4. เขียนโปรแกรมเพื่อย้ายค่าจากหน่วยความจำภายในตำแหน่ง 70H ถึงตำแหน่ง 7FH ไปเก็บไว้ที่หน่วยความจำภายนอก เริ่มจากตำแหน่ง 9000H เป็นต้นไป

5. เขียนโปรแกรมเพื่อย้ายค่าจากหน่วยความจำภายในตำแหน่ง 20H ถึงตำแหน่ง 4FH ไปเก็บไว้ที่หน่วยความจำภายใน เริ่มจากตำแหน่ง 50H เป็นต้นไป

6. เขียนโปรแกรมเพื่อย้ายค่าจากหน่วยความจำภายนอก ตำแหน่ง 1000H ถึงตำแหน่ง 111FH ไปเก็บไว้ที่หน่วยความจำภายนอก เริ่มจากตำแหน่ง 2300H เป็นต้นไป

7. เขียนโปรแกรมเพื่อย้ายค่าจากหน่วยความจำภายนอกตำแหน่ง 9000H ถึงตำแหน่ง 9007H ไปเก็บไว้ที่รีจิสเตอร์ R0, R1, ... R7

8. จากชุดคำสั่งแอสเซมบลีต่อไปนี้ ให้นักศึกษาตรวจสอบว่าคำสั่งในบรรทัดใดที่ไม่ถูกต้องตามหลักการเขียนโปรแกรมภาษาแอสเซมบลี ให้นักศึกษาทำการปรับแก้ให้ถูกต้องตามความเหมาะสม

```
ORG     8000H           ; 1
MOV     R1,#25H         ; 2
MOV     R2,#34H         ; 3
MOV     A,#F6H          ; 4
MOV     @R1,#24H        ; 5
MOV     26H,@R2         ; 6
MOV     DPTR,#8118H     ; 7
MOV     27H,A           ; 8
MOV     R3,#125         ; 9
MOV     A,R3            ; 10
MOVX    @DPTR,R3        ; 11
MOVX    A,@A+DPTR       ; 12
MOV     B,R3            ; 13
MOV     B,#11001010B    ; 14
MOV     DPL,#F4H        ; 15
MOV     DPH,#88H        ; 16
MOVX    @DPTR,A         ; 17
MOVC    A,@A+PC         ; 18
MOVC    @A+DPTR,A       ; 19
MOV     R5,#36H         ; 20
MOV     32H,43H         ; 21
MOV     @R5,#0A4        ; 22
MOV     #55H,60H        ; 23
SJMP    $               ; 24
END                     ; 25
```