| **การทดลองที่ 10: การใช้งานบัสข้อมูลและไดอะแกรมเวลา**<br>BUS and Digital Timing Diagram | | |
|---|---|---|
| Lab10_Prepair | Before อังคาร-20250121-0600 | https://forms.gle/pRhoUjvi6pGJ4pqw5 |
| Lab10_Full | Before ศุกร์-20250124-0600 | https://forms.gle/cped5yoYHSVnXu5J8 |
| **กลุ่มลงทะเบียน** | **กลุ่มการทดลอง** | **รหัส** **ชื่อ-สกุล** |

**ตารางตรวจ การทดลอง:** ให้พิมพ์ 2 หน้าแรกแบบหน้าหลัง มาพร้อมเข้าเรียน

| **การทดลอง** | **Score** | **ลายเซ็นต์** |
|---|---|---|
| 1. Start MCS51 with Mutisim14<br>จำลองการทำงาน ตามวงจรข้อ 5 | 3 | |
| 2. Start MCS51 with Mutisim14<br>ต่อวงจรเพื่อทดสอบ ตามวงจรข้อ 10 | 5 | |
| 3. Serial Shift Output<br>ต่อวงจรเพื่อทดสอบ ตามวงจรข้อ 13 | 5 | |
| 4. Serial Shift Input<br>ต่อวงจรเพื่อทดสอบ ตามวงจรข้อ 16 | 5 | |
| 5. Parallel Write/Read RAM<br>จำลองการทำงาน ตามวงจรข้อ 21 | 4 | |
| 6. Parallel Write/Read RAM<br>ต่อวงจรเพื่อทดสอบ ตามวงจรข้อ 22 | 8 | |
| 7. Advance (Top-up) ต่อวงจรเพื่อทดสอบ<br>☐ วงจร 8Bit Shift Input to 8Bit Shift Out<br>☐ วงจร Shift Input 16 Bit<br>☐ วงจร Shift Output 16 Bit | 10 | |

## สรุปผลการทดลอง

(1) ประเด็นขั้นตอนการทำ Serial Shift Input

(2) ประเด็นขั้นตอนการทำ Serial Shift Output

(3) ประเด็นขั้นตอนการอ่านและเขียน RAM

# การทดลองที่ 10: การใช้งานบัสข้อมูลและไดอะแกรมเวลา
# BUS and Digital Timing Diagram

## 1. วัตถุประสงค์เพื่อเรียนรู้เกี่ยวกับ

1. BUS → Address Bus, Data Bus, Control Bus
2. Digital Timing Diagram
3. Basic Assembly for 8051

## 2. อุปกรณ์การทดลอง

1. SUT-V5 MCS51 Board + Loader Accessory
2. TPIC6B595 board, 74165 Board, ETT-8LED, ETT-8SW, 8LED Board
3. RAM 6264
4. Breadboard + Jumper Wire

## 3. เนื้อหา

### 3.1 Microcontroller MCS-51

Micro-Controller คือ ตัวควบคุมการทำงานของอุปกรณ์หรือขบวนการต่างๆ ซึ่งอาจทำขึ้นมาจากวงจรไฟฟ้า, กลไก, PLC เป็นต้น Micro-Controller ก็คือ อุปกรณ์ประเภทสารกึ่งตัวนำที่รวบรวมฟังก์ชั่นการทำงานต่างๆ ไว้ภายในตัวของมันเอง มีขนาดเล็ก และสามารถเขียนโปรแกรมควบคุมการทำงานของอุปกรณ์ต่างๆ ที่เชื่อมต่อกับตัวมัน โดยเน้นความสมบูรณ์ภายในตัวของมันเองและง่ายต่อการนำไปใช้งานหรือแก้ไขดัดแปลง

Microcontroller ทั่วไป ประกอบด้วย

- CPU (Central Processing Unit)
- RAM (Random Access Memory)
- EPROM/PROM/ROM (Erasable Programmable Read Only Memory)
- I/O (Input/Output) - serial and parallel
- Timers
- Interrupt Controller
- และส่วนประกอบอื่นๆ เช่น Analog to Digital Convertor, Pluses Width Modulator ฯลฯ ซึ่งขึ้นกับผู้ผลิตที่จะใส่เข้าไป เพื่อเพิ่มความสามารถของ Microcontroller และจุดประสงค์ในการใช้งาน

ความแตกต่างของ Microcontroller และ Microcomputer คือ Microcomputer นั้นต้องการอุปกรณ์เชื่อมต่อภายนอก เช่น หน่วยความจำ I/O ฯลฯ ส่วน Microcontroller นั้นมีสมบูรณ์ภายในตัวของมันเอง

ภาษาของ Microcontroller

ภาษาที่ใช้กับ Microcontroller นั้นจะแตกต่างกันตาม Microcontroller ของแต่ละตระกูลแต่ประเภทของภาษาที่ใช้สามารถแบ่งออกเป็น

- ภาษาเครื่อง/ภาษา Assembly

ภาษาเครื่อง(Machine Language) คือโปรแกรมที่ Microcontroller สามารถเข้าใจมัน แต่มันไม่ง่ายสำหรับมนุษย์ที่จะอ่านได้ ภาษา Assembly คือ รูปแบบของภาษาเครื่องที่มนุษย์สามารถอ่านออกได้ ภาษา assembly เป็นโปรแกรมที่ทำหน้าที่ในการแปลงจากคำสั่งที่มนุษย์เอ่านออกได้ไปเป็นภาษาเครื่อง ซึ่งแปลงคำสั่ง/คำสั่ง โปรแกรมที่เขียนโดยภาพา assembly จะทำงานเร็วและมีขนาดเล็ก เพราะว่ามันสามารถเข้าถึง Hardware ได้โดยตรง แต่ทั้งนี้ขึ้นอยู่กับวิธีการเขียนของผู้เขียนด้วย

- Interpreters

interpreter คือ ภาษาระดับสูงซึ่งใกล้เคียงกับภาษาของมนุษย์ โดยจะฝังตัวอยู่ในหน่วยความจำ และทำหน้าที่อ่านคำสั่งจากโปรแกรมขึ้นมาทีละคำสั่งแล้วปฏิบัติตามคำสั่งนั้นๆ ตัวอย่างของ interpreter ที่รู้จักกันดีคือ ภาษา BASIC ข้อเสียของ interpreter คือ ทำงานได้ช้า เนื่องจากต้องแปลคำสั่งทีละคำสั่ง

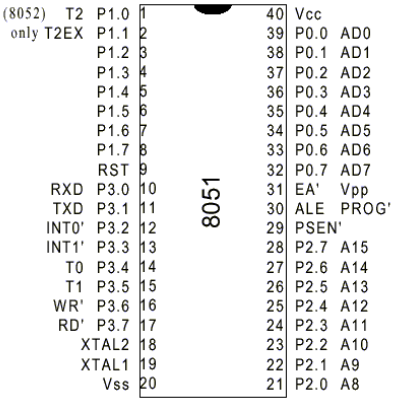
- Compilers

compiler คือ ภาษาระดับสูงซึ่งทำหน้าที่แปลโปรแกรมที่เขียนขึ้นให้เป็นภาษาเครื่อง จากนั้นจึงนำเอาโปรแกรมที่แปลเสร็จแล้วเข้าไปเก็บในหน่วยความจำ ทำให้การทำงานเร็วขึ้น ตัวอย่างเช่น ภาษา C เป็นต้น

โครงสร้าง 8051

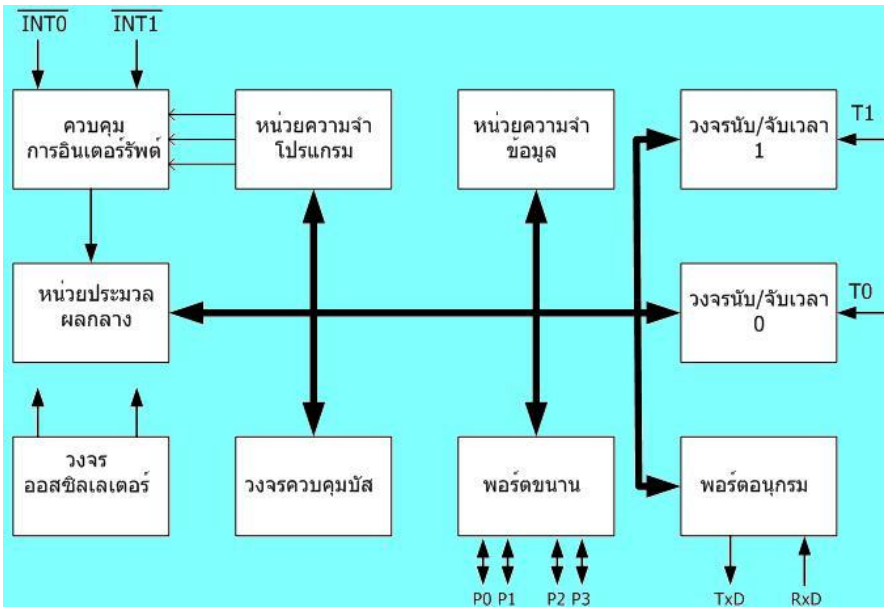
ลักษณะการจัดขาภายนอกของ MCS-51การจัดขาตามลักษณะภายนอกของชิป MCS-51 จะมีการแบ่งกลุ่มการจัดขาออกเป็น 4 กลุ่มด้วยกัน คือ

- กลุ่มขาแหล่งจ่ายไฟเลี้ยง และสัญญาณนาฬิกา
- กลุ่มขาสำหรับการอ้างแอดเดรสและรับส่งข้อมูล
- กลุ่มขาที่ใช้ในการควบคุม
- กลุ่มขาพอร์ตใช้งานแบบขนานและอนุกรม




โครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51

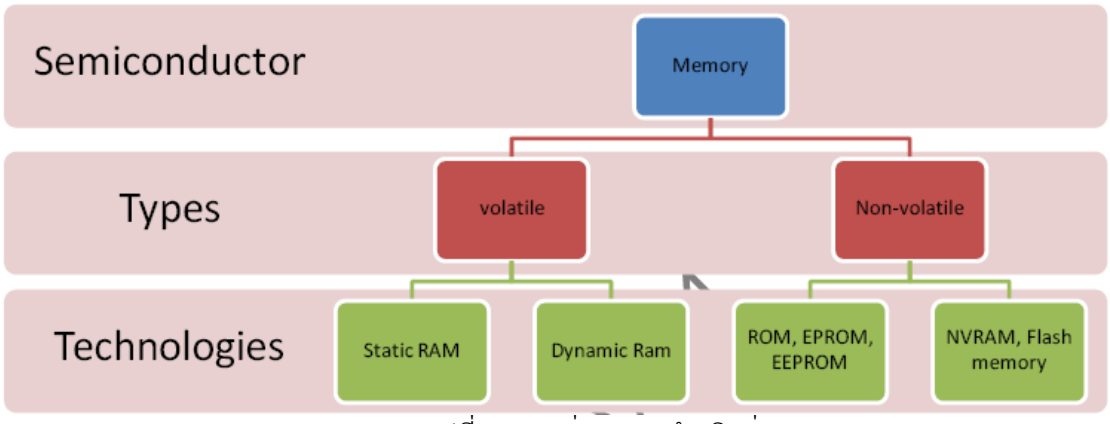
โครงสร้างภายในพื้นฐานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 เบอร์ 8051 ประกอบด้วยอุปกรณ์ต่างๆ ดังนี้ ส่วนของหน่วยความจำภายในสำหรับเก็บข้อมูลขนาด 128 ไบท์ (Internal Data Memory) ส่วนของหน่วยความจำภายในสำหรับเก็บโปรแกรมที่มีขนาด 4 กิโลไบท์ (Internal Program Memory) อุปกรณ์ควบคุมการอินเตอร์รัพท์ (Interrupt Control Unit) ตัวตั้งเวลาและตัวนับเวลาขนาด 16 บิท 2 ชุด (Timer/Counter 0 and Timer/Counter 1) พอร์ตควบคุมการสื่อสารอนุกรมแบบ Full Duplex ซึ่งสามารถรับส่งข้อมูลพร้อมกันได้ พอร์ตขนานสำหรับติดต่อกับอุปกรณ์ภายนอกจำนวน 4 พอร์ต พอร์ตละ 8 บิท วงจรผลิตสัญญาณนาฬิกาภายใน




รูปที่ 10.2 โครงสร้างภายในของไมโครคอนโทรลเลอร์ MCS-51

### 3.2 หน่วยความจำ

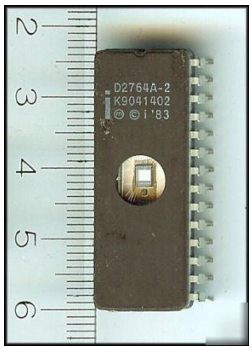
หน่วยความจำ (Memory) เป็นอุปกรณ์ที่ทำหน้าที่ในการเก็บข้อมูล สร้างจากสารกึ่งตัวนำ (Semiconductor devices) เราแบ่งลักษณะการจัดเก็บข้อมูลของหน่วยความจำได้ดังรูปที่ 10.3 โดยหน่วยความจำแบ่งออกได้เป็นสองชนิด คือ ข้อมูลคงอยู่แม้ว่าไม่มีไฟเลี้ยง (non-volatile) และ ข้อมูลไม่คงอยู่เมื่อไม่มีไฟเลี้ยง (volatile)

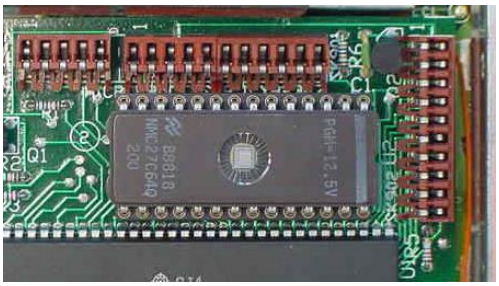



รูปที่ 10.3 หน่วยความจำชนิดต่างๆ

*หน่วยความจำในกลุ่ม Non-volatile แบ่งย่อยออกได้เป็น*

EPROM ย่อมาจาก Erasable Programmable Read Only Memory เป็นหน่วยความที่ใช้การโปรแกรมเพื่อให้เก็บข้อมูลด้วยไฟฟ้า และเมื่อหากต้องการลบข้อมู ลจำเป็นต้องใช้แสง ultraviolet ส่องที่ window ของ chip เพื่อให้สารกึ่งตัวนำที่อยู่ภายในกลับคืนสถานะว่าง ในการใช้งาน EPROM จะมีเทปกาวปิดเพื่อป้องกันแสง UV จากภายนอกเข้ามาลบข้อมูล



เบอร์ไอซี EPROM ขนาด 8bit ที่นิยมใน controller

2732 8bit 4KB

2764 8bit 64KB

27128 8Bit 128KB

รูปที่ 10.4 EPROM และขนาดความจุ

EEPROM หรือ E2PROM ย่อมาจาก Electrically Erasable Programmable Read Only Memory เป็นหน่วยความจำที่ใช้การโปรแกรมและการลบข้อมูลด้วยไฟฟ้า มีความเร็วในการลบข้อมูลและการเขียนข้อมูลค่อนข้างช้าเมื่อเทียบกับ Flash memory เพราะเนื่องจากข้อจำกัดทางด้านโครงสร้างที่การลบข้อมูลต้องทำที่ละบิต อายุในการลบและโปรแกรมอยู่ที่ประมาณ 10,000 ครั้ง

Flash memory พัฒนามาจาก EEPROM โดยสามารถลบข้อมูลได้ครั้งละหลายบิต ทำให้มีความเร็วในการลบข้อมูลที่สูงกว่า และมีการพัฒนาให้มีจำนวนครั้งในการเขียนข้อมูลสูงขึ้น โดยเทคโนโลยี NAND flash (chip: Samsung KFW4G16Q2M) สูงสุดประมาณ 100,000 ครั้ง NOR flash (chip: Spansion S29CD016J) สูงสุดที่ประมาณ 1,000,000 ครั้ง อุปกรณ์ที่ใช้ Flash memory ได้แก่ Multimedia card, Securer digital card, Compact flash, memory stick, USB flash drives, Solid-state drive เป็นต้น

*หน่วยความจำในกลุ่ม Volatile แบ่งออกได้เป็นสองเทคโนโลยีหลักๆ คือ*

Static RAM หรือ Static Random-Access Memory เป็นหน่วยความจำที่ใช้ไฟฟ้าในการรักษาข้อมูลให้คงอยู่ ภายออกแบบเป็นวงจรโลจิกที่เก็บสถานะของโลจิก ลักษณะคล้ายกับ latcher circuit เราสามารถสร้าง SRAM ได้จาก D flipflop

DRAM หรือ Dynamic Random-Access Memory เป็นหน่วยความจำที่ใช้ไฟฟ้าในการรักษาข้ อมูลให้คงอยู่และยังต้องมีการกระตุ้น (Refresh) เพื่อให้สถานะโลจิกคงอยู่ด้วย หลักการเก็บข้อมูลของหน่วยความจำแบบนี้ให้นึกถึงหลักการทำงานของตัวเก็บประจุ (Capacitor) ที่สามารถเก็บรักษาแรงดันไฟฟ้าไว้ได้ชั่วเวลาขณะหนึ่งจากนั้นแรงดันก็จะลดระดับลงมา ในทางโครงสร้างการทำงาน DRAM ใช้จำนวนอุปกรณ์ในการสร้างน้อยกว่า SRAM ทำให้ต้นทุนในการสร้างต่ำกว่า สามารถสร้างอุปกรณ์ที่มีความจุข้อมูลมากกว่าในขนาดวงจรที่เท่ากัน

**3.2 รายละเอียดของหน่วยความจำ และการเชื่อมต่อ**

ในหัวข้อที่ผ่านมาแสดงให้เราเห็นว่าหน่วยความจำมีด้วยกันหลายแบบ หน่วยความจำเหล่านั้นมีการเชื่อมต่อและใช้งานกับ Microcontroller และ Microcomputer ในหลายๆส่วนเช่น

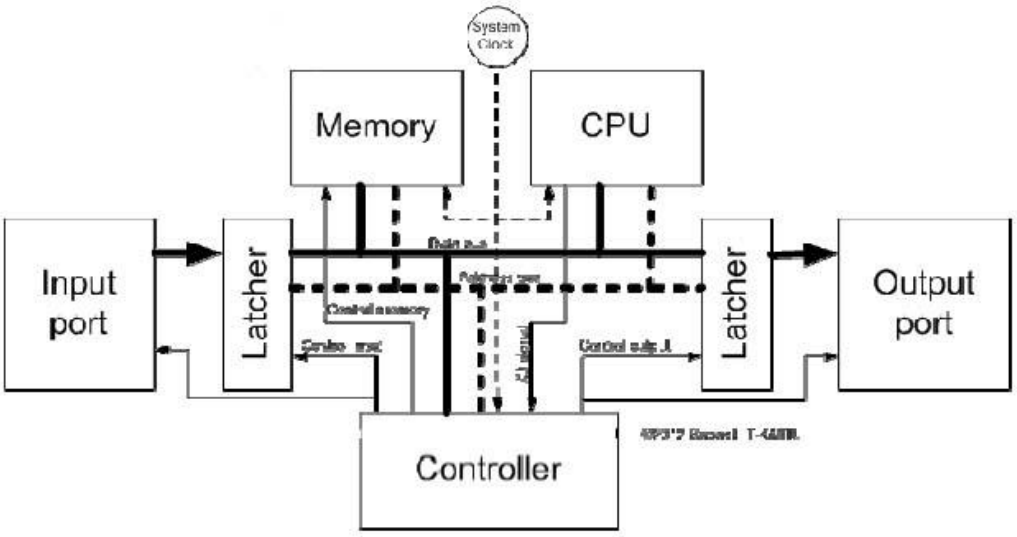
1) หน่วยความจำแบบ EEPROM, EPROM อาจนำมาใช้เก็บข้อมูลชุดคำสั่งเป็นโปรแกรมที่เขียนขึ้นมา
2) หน่วยความจำแบบ SRAM อาจนำมาประยุกต์ใช้เป็นพื้นที่ในการพักข้อมูลเพื่อใช้ในการประมวลผล
3) หน่วยความจำแบบ Flash อาจนำมาใช้ในการเก็บผลลัพธ์ที่ได้หลังจากประมวลผล และเพื่อใช้ในการทำงานของระบบต่อไป ลักษณะของการประยุกต์ใช้งานนั้นขึ้นอยู่กับลักษณะของงาน

3.2.1 สัญญาณต่างๆหน่วยความจำ

หน่วยความจำประกอบด้วยกลุ่มของสัญญาณสามส่วนคือ

- Data bus เป็นกลุ่มของสัญญาณข้อมูล
- Address bus เป็นกลุ่มของสัญญาณที่ใช้ชี้ตำแหน่งข้อมูลนั้น
- Control signal เป็นสัญญาณที่สั่งให้อ่านหรือเขียนข้อมูล

จากรูปที่ 10.5 ซึ่งเราจะเห็นได้ว่าหน่วยความจำมีสัญญาณ Data bus, address bus และ control signal ป้อนเข้าในหน่วยความจำ




รูปที่ 10.5 ส่วนประกอบของคอมพิวเตอร์และสัญญาณต่างๆ

3.2.2 ขนาดของหน่วยความจำ

หน่วยความจำเก็บข้อมูลในลักษณะ “0” และ “1” ซึ่งเป็น binary data ขนาดของหน่วยความจำมองได้สองลักษณะคือขนาดความจุ (capacity size) และ จำนวนของบิต (number of bit)

- ขนาดของความจุแบ่งระดับเป็น

  2^8 = 8bit (b) = 1Byte (B)

  2^8 x 2^8 = 16bit = 1Word

  2^8 x 2^8 x 2^8 = 1024B= 1Kilobyte (KB)

  2^8 x 2^8 x 2^8 x 2^8 = 1024KB = 1Megabyte (MB)

- จำนวนบิต คือขนาดของสายสัญญาณ data bus ซึ่งสัมพันธ์กับจำนวนระดับความแตกต่างของข้อมูล ซึ่งคำนวณได้จากสูตร $2^N$ โดยที่ N คือจำนวนบิต
- ตัวอย่าง ให้หา databus 16 bit มีขนาดของความแตกต่างเท่าใด

  การคำนวณใช้สูตร (2^16) = 65536 ระดับของความแตกต่าง

- ตัวอย่าง ให้หา databus 24 bit มีขนาดของความแตกต่างเท่าใด

  การคำนวณใช้สูตร (2^24) = 16,777,216 ระดับของความแตกต่าง

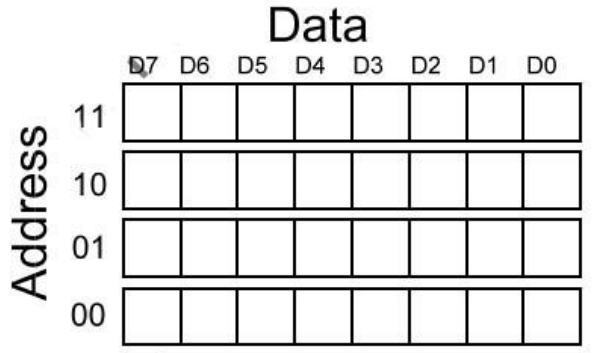
3.2.3 ตำแหน่งในเก็บข้อมูลในหน่วยความจำ

หน่วยความจำมีสัญญาณ Address bus เป็นสัญญาณที่ใช้ในการระบุตำแหน่งที่ใช้ในการเก็บข้อมูล ยกตัวอย่างง่ายๆ สิ่งที่ส่งมาใน address bus เปรียบเสมือนกับเลขที่บ้าน

ตัวอย่าง ให้หา Address but 16 เส้น มีตำแหน่งในการเก็บข้อมูลได้เท่าใด

การคำนวณใช้สูตร (2^16) = 65536 ตำแหน่ง

3.2.4 การเก็บข้อมูล




รูปที่ 10.6 ตัวอย่างหน่วยความจำขนาด 4x8

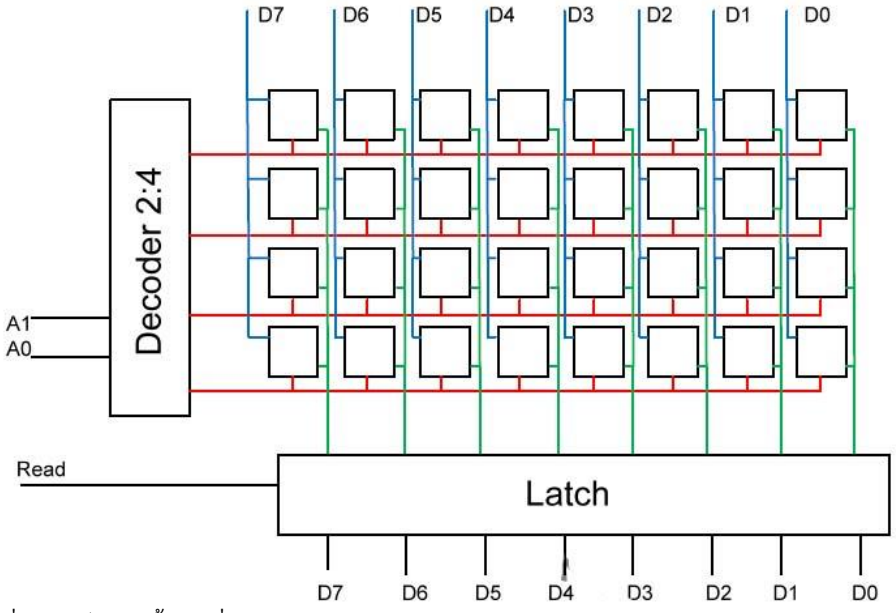
จากที่กล่าวมาในหัวข้อ 3.2.2 และ 3.2.3 นักศึกษาอาจจะยังสับสนว่า ทั้ง address และ data มีการคำนวณด้วย $2^N$ ทั้งหมด เพื่อใช้ในการคำนวณหาตำแหน่งและขนาด เพื่อให้เข้าใจได้ง่ายๆ หากระบบหน่วยความจำมี address 2 เส้น และมี data ขนาด 8 bit โครงสร้างของหน่วยความจำเขียนเป็นตารางได้ดังรูปที่ 10.6

อธิบายรูป 10.6 เรียกว่าหน่วยความจำขนาด 4 x 8 หมายถึงมี address 4 ตำแหน่ง และในแต่ละตำแหน่งเก็บข้อมูลได้ 8 bit

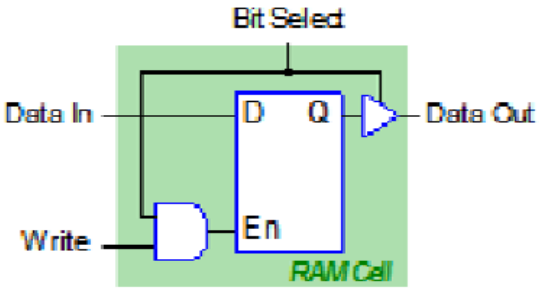
ตัวอย่าง หน่วยความจำตัวหนึ่งเขียนว่า 8K x 8 SRAM หมายความว่าอย่างไร

- หน่วยความจำเป็นชนิด SRAM ไม่ต้องมีการ Refresh และ
- หน่วยความจำมี Address 8K หรือ 8192 (2^13) มีจำนวนขา address ทั้งหมด 13ขา (A0 ถึง A12) และ
- หน่วยความจำมี Data bus ขนาด 8 บิต

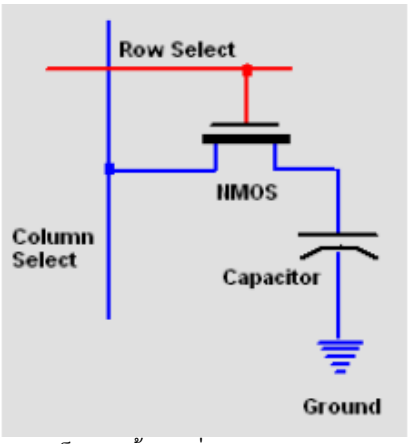
3.3 วงจรเสมือนแสดงการทำงานของหน่วยความจำ SRAM และ DRAM




รูปที่ 10.7 โครงสร้างหน่วยความจำ SRAM ขนาด 4x8 แบบ separate input/output data



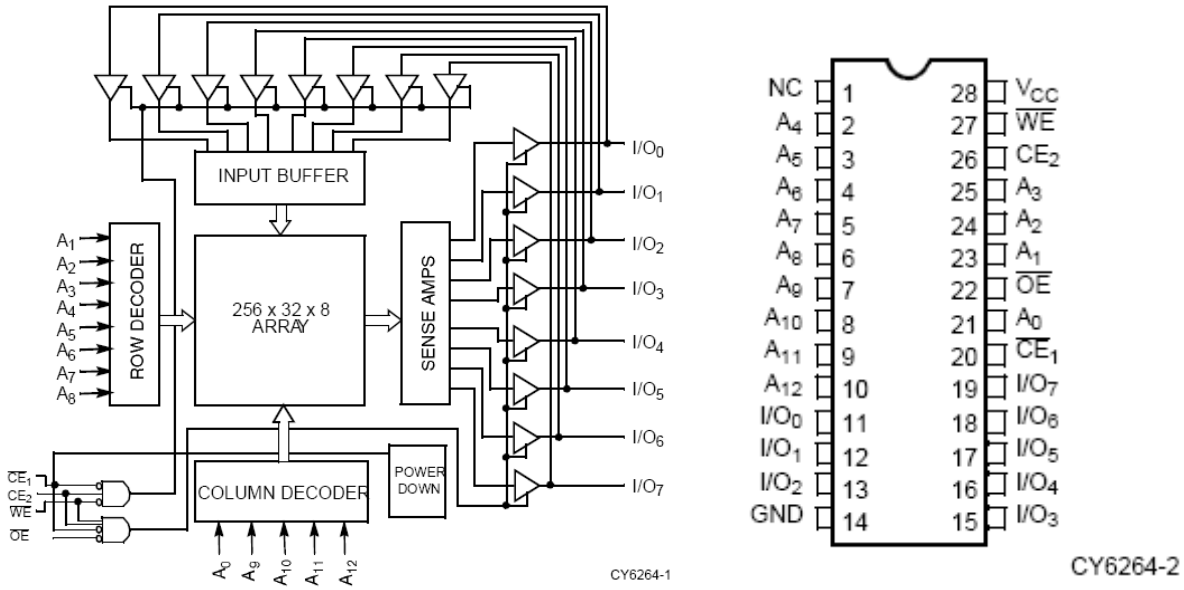

รูปที่ 10.8 โครงสร้างหน่วยความจำ SRAM 1 บิต



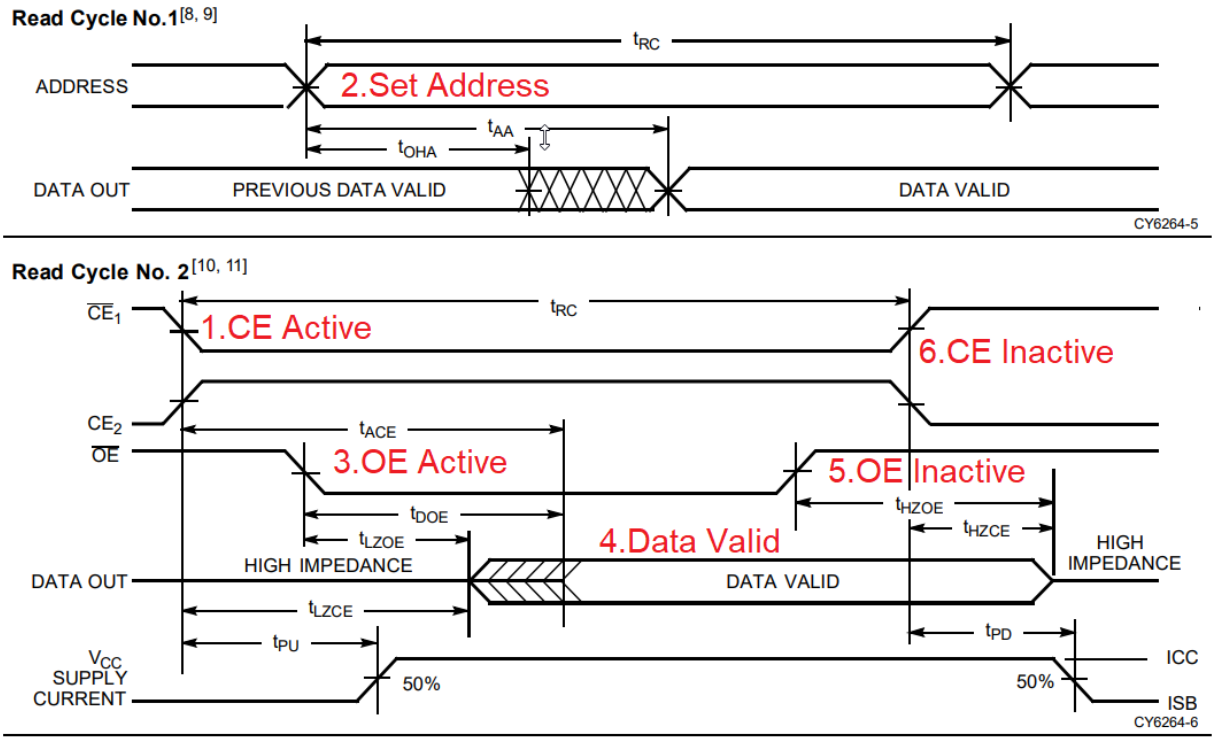

รูปที่ 10.9 โครงสร้างหน่วยความจำ DRAM 1 บิต

## 3.4 ไดอะแกรมเวลาการอ่านและเขียนข้อมูลจากแรม

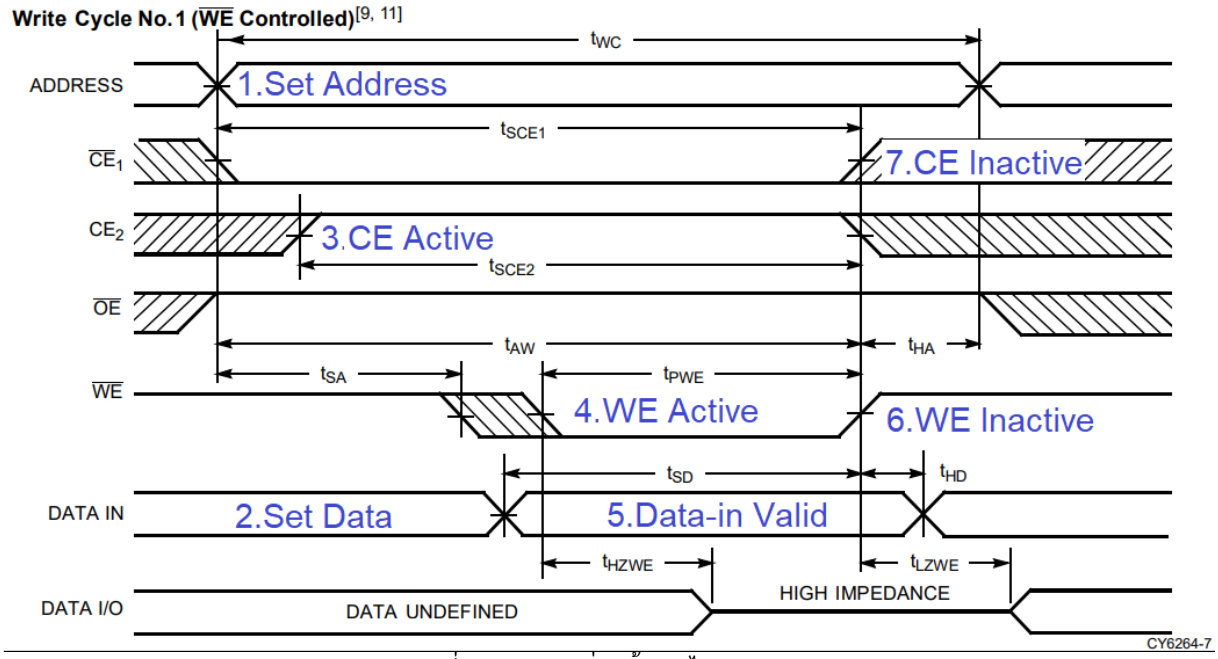
- http://www.cs.uml.edu/~fredm/courses/91.305/files/cy6264.pdf




รูปที่ 10.10 หน่วยความจำแรมเบอร์ 6264




รูปที่ 10.11 การเขียนข้อมูลไปยังแรม




รูปที่ 10.12 การอ่านข้อมูลไปจากแรม

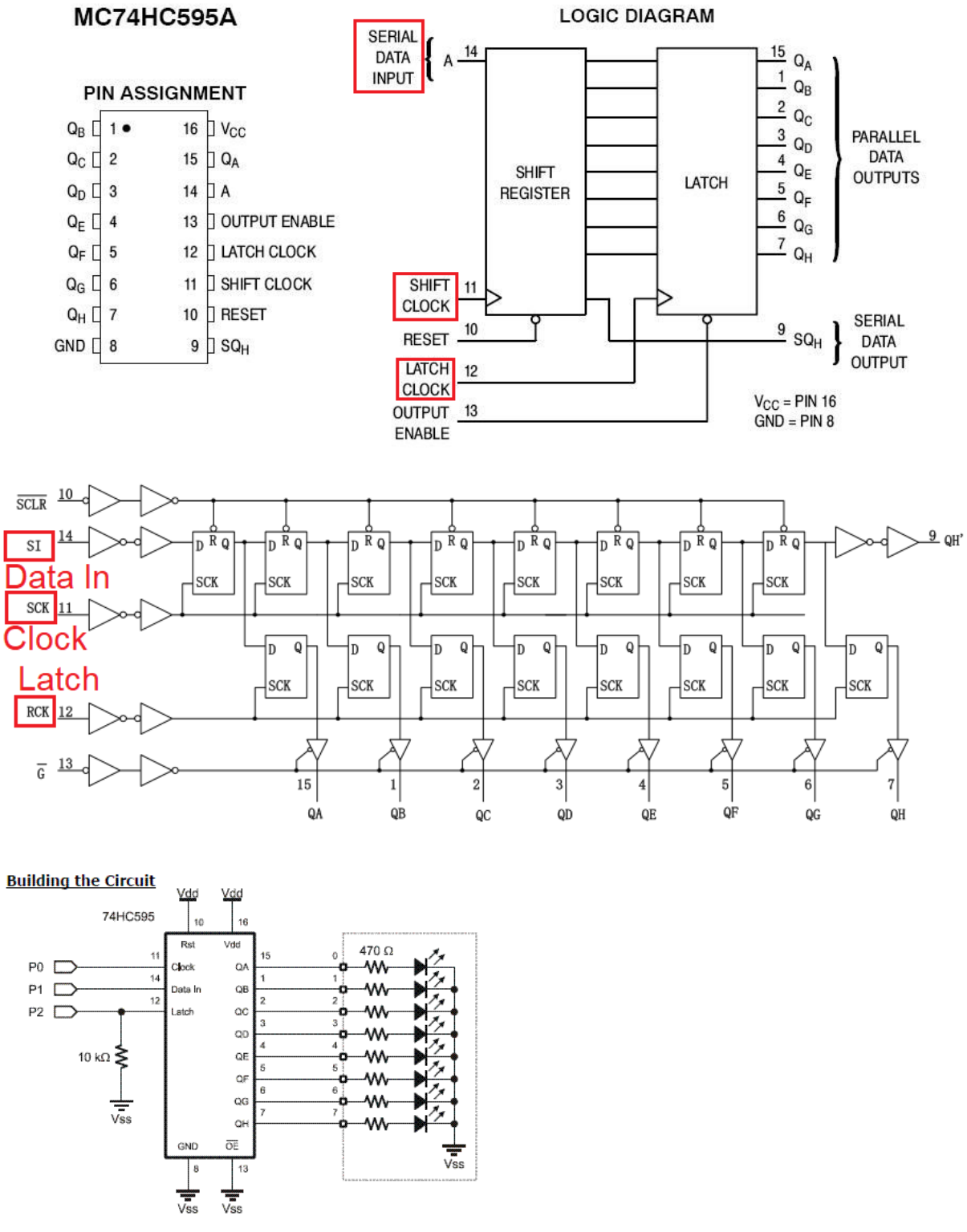
## 3.3 Serial to Parallel output with Shift Latch

- https://grace.bluegrass.kctcs.edu/~kdunn0001/files/Shift_Registers/Shift_Registers6.html
- https://eleceasy.com/t/output-arduino-12-pin-32-pin-ic-74hc595/2426
- https://www.programmersought.com/article/76774446243/

ชิฟต์เรจิสเตอร์ (shift register) เป็นอุปกรณ์ทางลอจิกที่สร้างมาจากฟลิปฟลอปใช้เก็บข้อมูลชั่วคราวการส่งข้อมูลเข้าเรจิสเตอร์ส่งได้ทีละบิต เรจิสเตอร์ที่เก็บข้อมูลสามารถเลื่อนข้อมูลได้เราเรียกว่า เรจิสเตอร์เลื่อนข้อมูล หรือ ชิฟต์เรจิสเตอร์ ถ้านำฟลิปฟล็อปมาต่อเรียงกันจะทำให้ฟลิปฟล็อปเอาข้อมูลจาก input ไปยัง output ได้เมื่อมีสัญญาณการกระตุ้นจะเกิดการเลื่อนข้อมูลที่ละบิต ตัวเลื่อนข้อมูลจัดเป็นการนำฟลิปฟล็อปไปใช้งานได้ การเลื่อนข้อมูลจะมีลักษณะต่างกันดังนี้

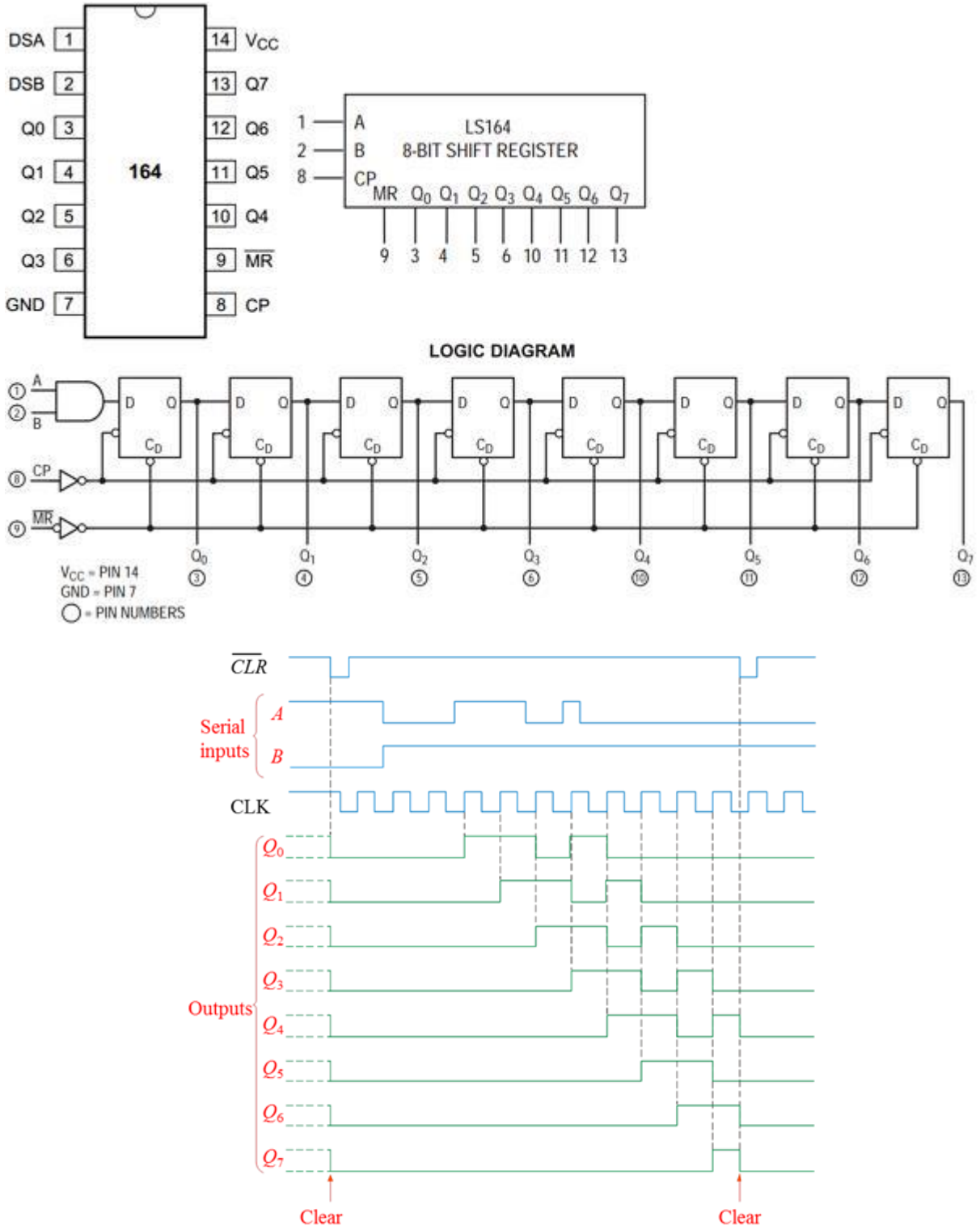
- ชิฟต์เรจิสเตอร์เข้าขนานออกขนาน
- ชิฟต์เรจิสเตอร์เข้าขนานออกอนุกรม
- ชิฟต์เรจิสเตอร์เข้าอนุกรมออกขนาน
- ชิฟต์เรจิสเตอร์เข้าอนุกรมออกอนุกรม

3.3.1 ไอซี 74HC595: 8 Bit Shift Register with 8-bit output register



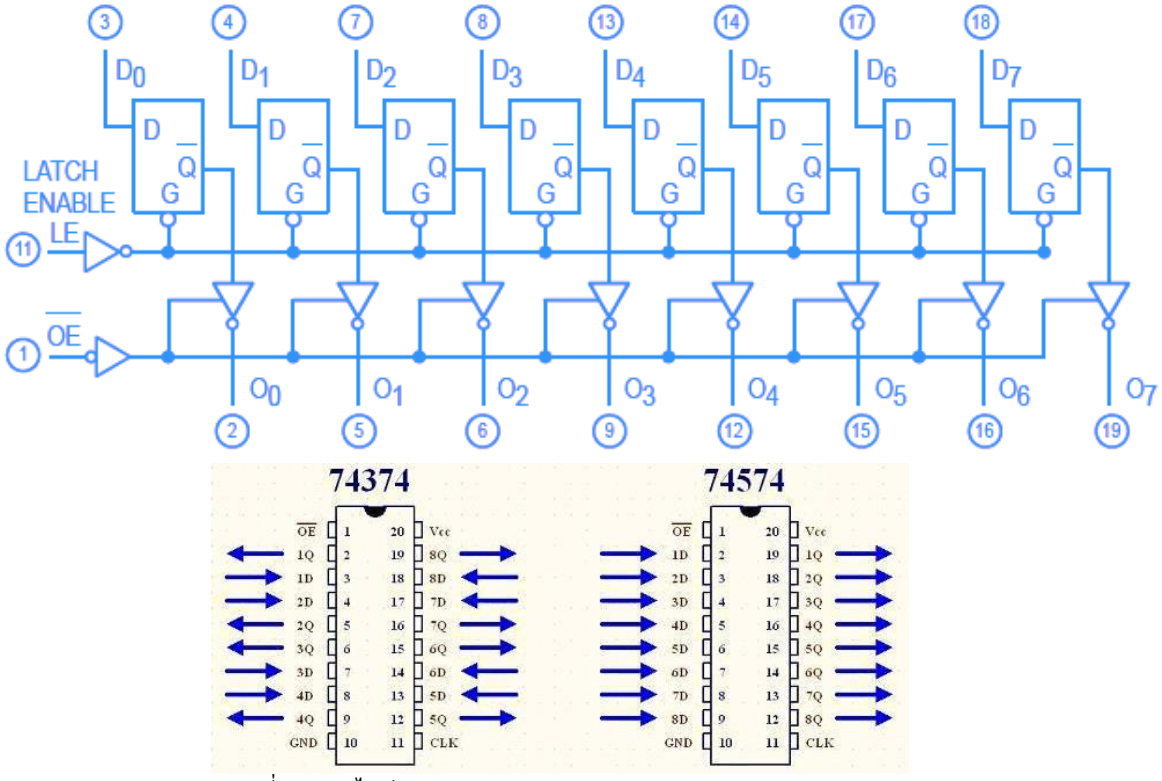

รูปที่ 10.13 ไอซี 74HC595: 8 Bit Shift Register with 8 bit output register

3.3.2 ไอซี 74164: 8 Bit Serial-in / Parallel-out Shift Register

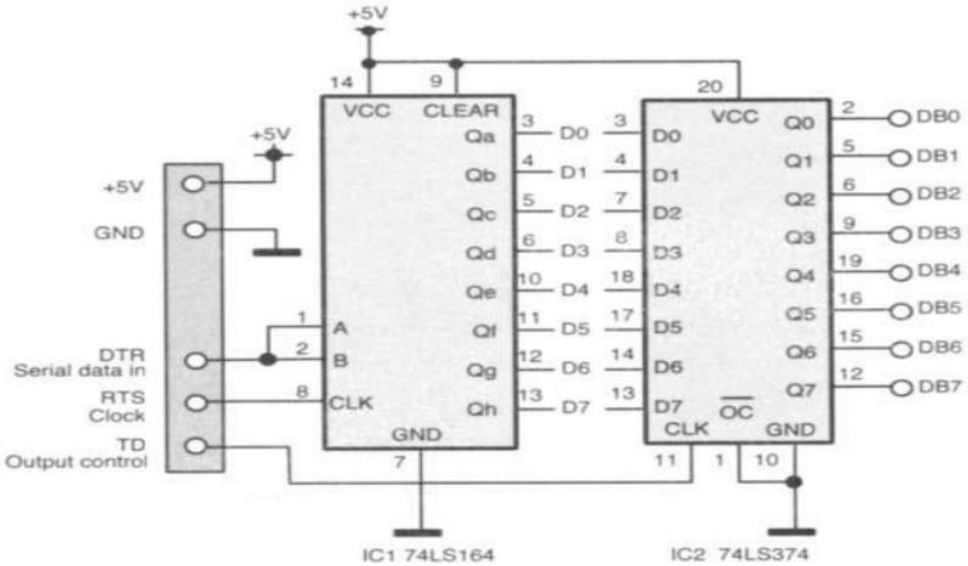



รูปที่ 10.14 ไอซี 74164: 8 Bit Serial-in / Parallel-out Shift Register

3.3.3 ไอซี 74LS374, 74LS574: Octal D-Type Flip-Flop Register




รูปที่ 10.15 ไอซี 74LS374: Octal D-Type Flip-Flop Register




รูปที่ 10.16 Serial to Parallel convertor circuits using 74LS164 and 74LS374

## 4. เตรียมการทดลอง

1. เตรียมการทดลองสำหรับข้อ 5 < ทดสอบวงจรใน Multisim แล้ว Capture หน้าจอคอมพิวเตอร์ตัวเองให้เห็น Desktop 50%, Mutisim 25%, MIDE_Code 25% >

2. เตรียมการทดลองสำหรับข้อ 11 < ทดสอบวงจรใน Multisim แล้ว Capture หน้าจอคอมพิวเตอร์ตัวเองให้เห็น Desktop 50%, Mutisim 25%, MIDE_Code 25% >

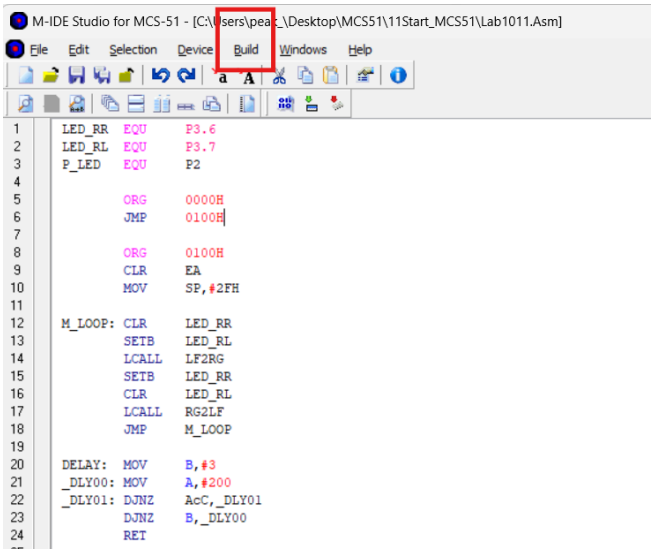
3. เตรียมการทดลองสำหรับข้อ 14 < ทดสอบวงจรใน Multisim แล้ว Capture หน้าจอคอมพิวเตอร์ตัวเองให้เห็น Desktop 50%, Mutisim 25%, MIDE_Code 25% >

4. เตรียมการทดลองสำหรับข้อ 21 < ทดสอบวงจรใน Multisim แล้ว Capture หน้าจอคอมพิวเตอร์ตัวเองให้เห็น Desktop 50%, Mutisim 25%, MIDE_Code 25% >

## 5. การทดลอง

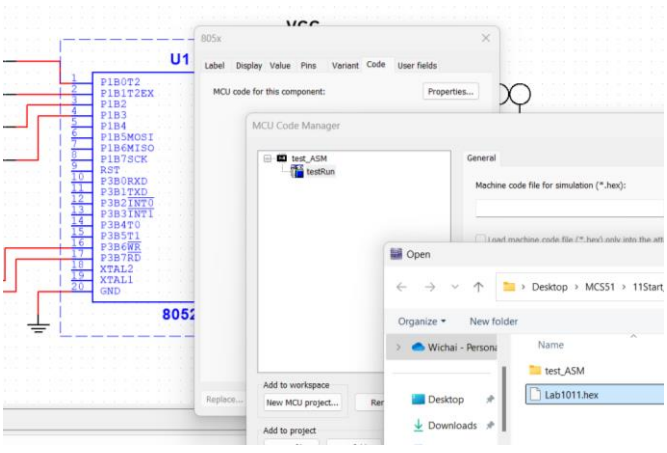
### 1. Start MCS51 with Mutisim14 for Parallel Output

1. Install MIDE51 >> https://drive.google.com/file/d/1t3MAFLE9mtpsSrT30UI0Rm6NZsubjEei/view?usp=drive_link
2. Open MIDE-51 → Edit Code → Save as "Lab1011.Asm"
3. Compile ด้วยการ Build → Build All จะได้ไฟล์ "Lab1011.Hex"





4. Run NI Mutisim 14, Open "Test1011_8LED.ms14"
5. Load "Lab1011.Hex" File and Run

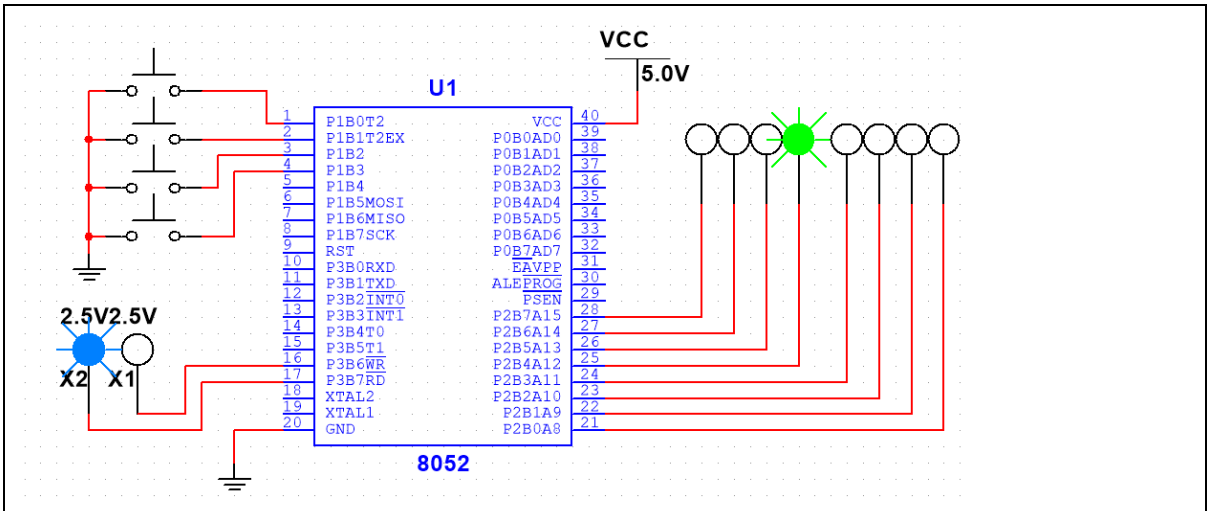



5.1 Click U1(8052) → Properties

5.2 Tab Code → Properties

- New MCU Project
- External Hex file
- Edit Project Name = Test1011
- Click Test1011
- Browse to Lab1011.Hex
- Ok

5.3 Ok

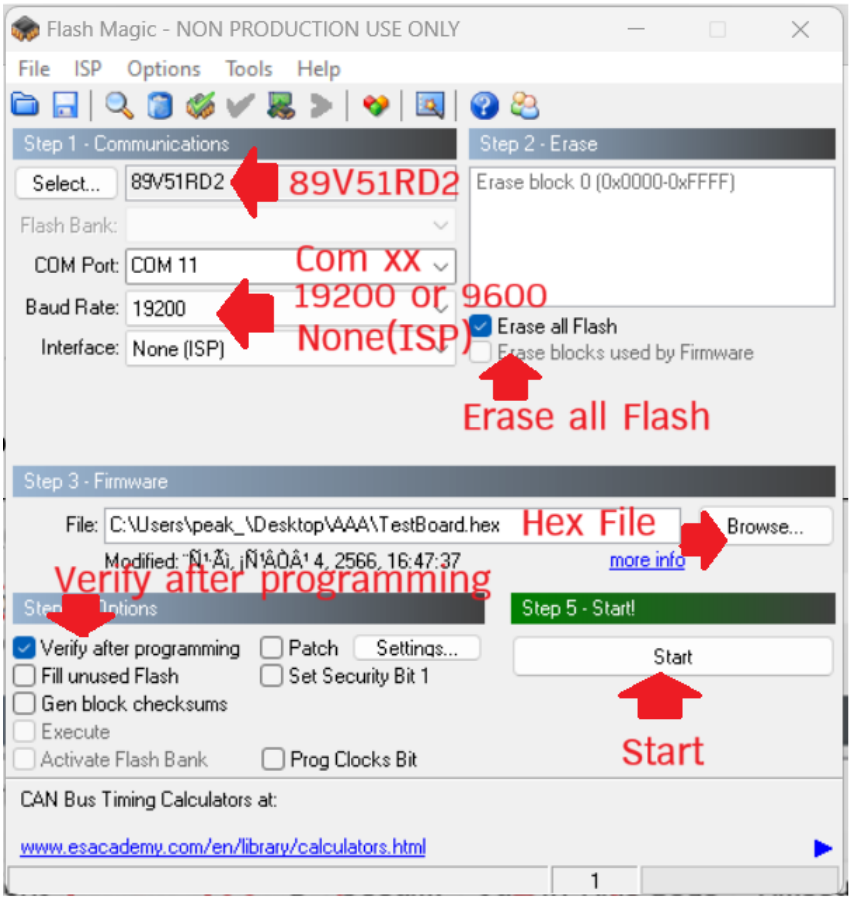



```
LED_RR      EQU     P3.6
LED_RL      EQU     P3.7
P_LED       EQU     P2

            ORG     0000H
            JMP     0100H

            ORG     0100H
            CLR     EA
            MOV     SP,#2FH

M_LOOP:     CLR     LED_RR
            SETB    LED_RL
            LCALL   LF2RG
            SETB    LED_RR
            CLR     LED_RL
            LCALL   RG2LF
            JMP     M_LOOP

DELAY:      MOV     B,#3
_DLY00:     MOV     A,#200
_DLY01:     DJNZ    AcC,_DLY01
            DJNZ    B,_DLY00
            RET

LF2RG:      MOV     P_LED,#10000000B
            LCALL   DELAY
            MOV     P_LED,#01000000B
            LCALL   DELAY
            MOV     P_LED,#00100000B
            LCALL   DELAY
            MOV     P_LED,#00010000B
            LCALL   DELAY
            MOV     P_LED,#00001000B
            LCALL   DELAY
            MOV     P_LED,#00000100B
            LCALL   DELAY
            MOV     P_LED,#00000010B
            LCALL   DELAY
            MOV     P_LED,#00000001B
            LCALL   DELAY
            RET

RG2LF:      MOV     P_LED,#00000001B
            LCALL   DELAY
            MOV     P_LED,#00000010B
            LCALL   DELAY
            MOV     P_LED,#00000100B
            LCALL   DELAY
            MOV     P_LED,#00001000B
            LCALL   DELAY
            MOV     P_LED,#00010000B
            LCALL   DELAY
            MOV     P_LED,#00100000B
            LCALL   DELAY
            MOV     P_LED,#01000000B
            LCALL   DELAY
            MOV     P_LED,#10000000B
            LCALL   DELAY
            RET

            END
```


```
1
2   LED_RR  EQU     P3.6
3   LED_RL  EQU     P3.7
4   P_LED   EQU     P2
5
6           ORG     0000H
7           JMP     0100H
8
9           ORG     0100H
10          CLR     EA
11          MOV     SP,#2FH
12
13  M_LOOP: CLR     LED_RR
14          SETB    LED_RL
15          LCALL   LF2RG
16          SETB    LED_RR
17          CLR     LED_RL
18          LCALL   RG2LF
19          JMP     M_LOOP
20
21  DELAY:  MOV     B,#3
22  _DLY00: MOV     A,#200
23  _DLY01: DJNZ    AcC,_DLY01
24          DJNZ    B,_DLY00
25          RET
26
27  LF2RG:  MOV     P_LED,#10000000B
28          LCALL   DELAY
29          MOV     P_LED,#01000000B
30          LCALL   DELAY
31          MOV     P_LED,#00100000B
32          LCALL   DELAY
33          MOV     P_LED,#00010000B
34          LCALL   DELAY
35          MOV     P_LED,#00001000B
36          LCALL   DELAY
37          MOV     P_LED,#00000100B
38          LCALL   DELAY
39          MOV     P_LED,#00000010B
40          LCALL   DELAY
41          MOV     P_LED,#00000001B
42          LCALL   DELAY
43          RET
44
45  RG2LF:  MOV     P_LED,#00000001B
46          LCALL   DELAY
47          MOV     P_LED,#00000010B
48          LCALL   DELAY
49          MOV     P_LED,#00000100B
50          LCALL   DELAY
51          MOV     P_LED,#00001000B
52          LCALL   DELAY
53          MOV     P_LED,#00010000B
54          LCALL   DELAY
55          MOV     P_LED,#00100000B
56          LCALL   DELAY
57          MOV     P_LED,#01000000B
58          LCALL   DELAY
59          MOV     P_LED,#10000000B
60          LCALL   DELAY
61          RET
62
63          END
```

6. ทดสอบการทำงานกับ SUT_MCS51-V5 Board
7. สำหรับ P89V51RD2 จะใช้ Flash Magic 11 ในการโหลด Hex File
   https://drive.google.com/file/d/1iBuzMsYA4epX0Hs5ggzyqscyvflVME-3/view?usp=sharing
8. ต่อวงจร, โหลดไฟล์ด้วย Flash Magic 11
   8.1 Open Flash Magic and Config


8.2 ให้กำหนดค่า CPU, ComX, Baud, None(ISP), Erase.., Hex File, Verify.. แต่ ยังไม่กด Start

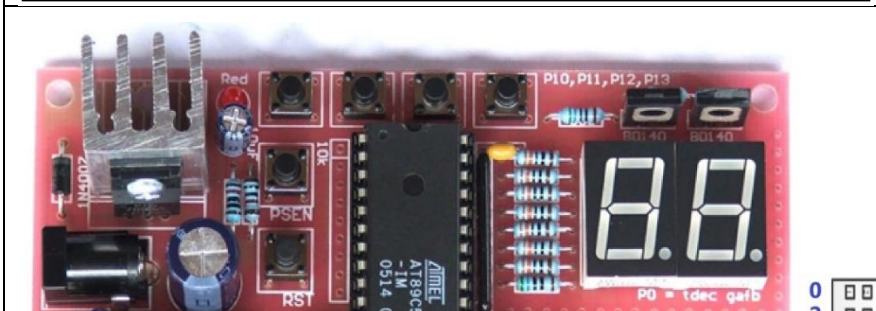


8.3 กดปุ่ม Reset ที่ MCS-51 Board ค้างไว้ แล้วกด Start ที่โปรแกรมรอจนกว่าขึ้น Reset the .. แล้วปล่อยปุ่ม Reset โปรแกรมจะทำการ Uplode Hex File ไปยังบอร์ด รอจนกว่าจะขึ้น Finished



8.4 กดและปล่อย ปุ่ม Reset ที่ MCS-51 Board อีกครั้งเพื่อเริ่มการทำงาน

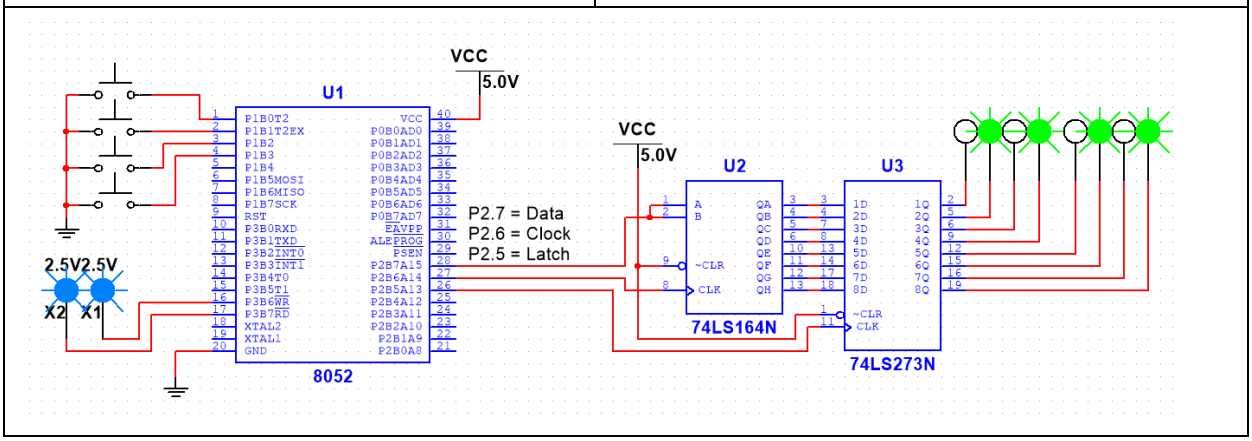
9. หากทำสำเร็จไฟ LED P3.6, P3.7 จะกระพริบสลับกัน
10. ต่อวงจรเพื่อทดสอบกับ 8 LED แบบขนาน



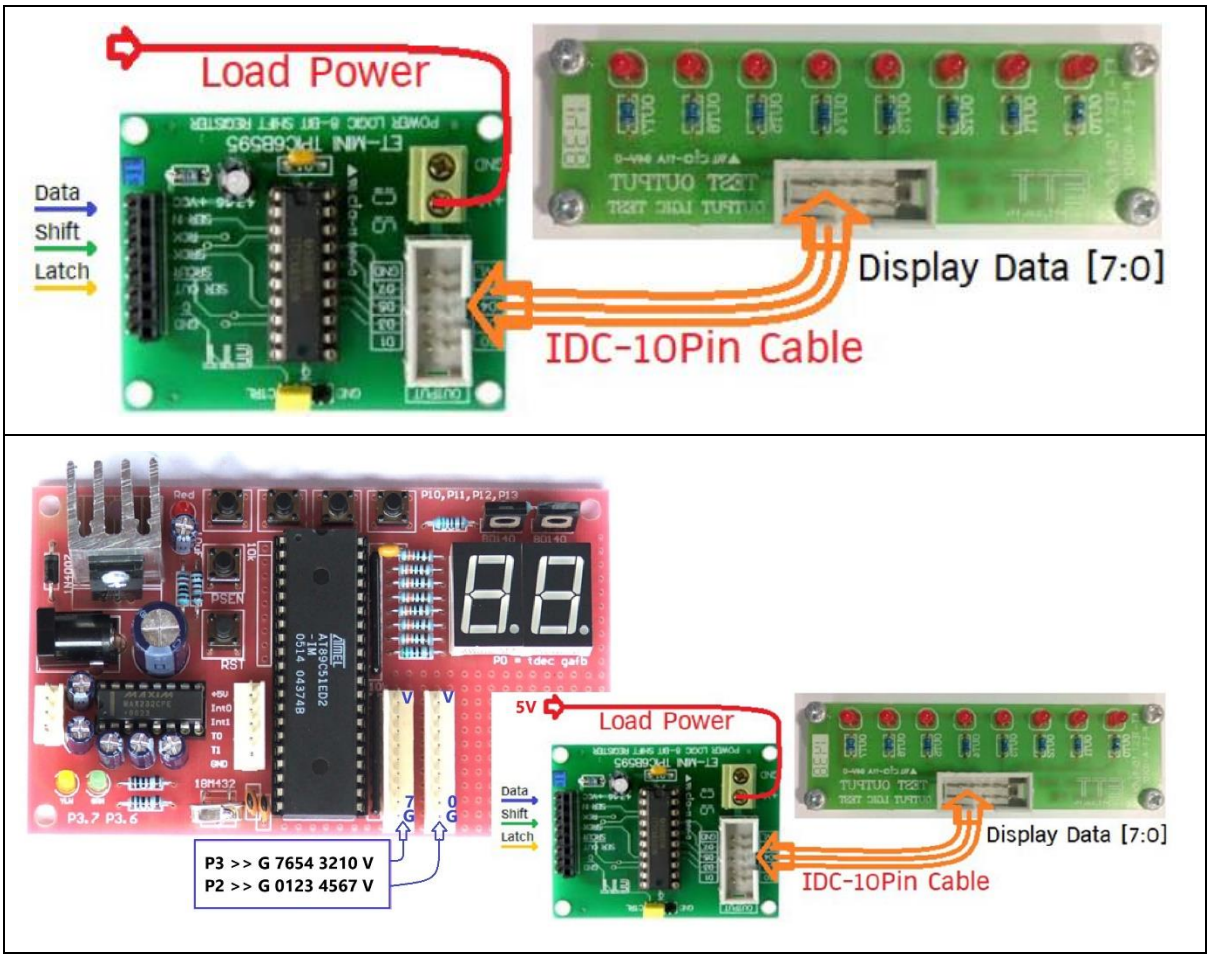
## 2. Serial Shift Output with 74595 (8-bit serial-input/serial or parallel-output shift register)

11. จำลองการทำงานของ Lab1021

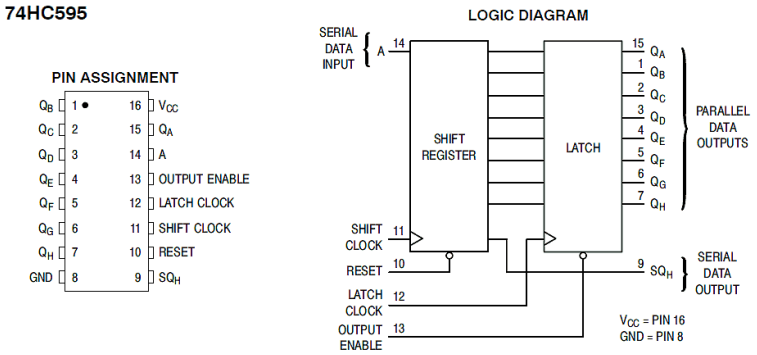
```
pLED_R      EQU     P3.6
pLED_G      EQU     P3.7
pData       EQU     P2.7
pClock      EQU     P2.6
pLatch      EQU     P2.5

            ORG     0000H
            JMP     0100H

            ORG     0100H
            CLR     EA
            MOV     SP,#3FH
            CLR     pClock

mLOOP:      SETB    pLED_R
            CLR     pLED_G
            MOV     A,#055H
            LCALL   XSend
            LCALL   xDelay

            CLR     pLED_R
            SETB    pLED_G
            MOV     A,#0AAH
            LCALL   XSend
            LCALL   xDelay
            JMP     mLOOP

xDelay:     MOV     B,#1
_Dly01:     MOV     A,#200
_Dly00:     DJNZ    Acc,_Dly00
            DJNZ    B,_Dly01
            RET

XSend:      MOV     R0,#8
_Shift:     RLC     A
            MOV     pData,C
            CLR     pClock
            SETB    pClock
            DJNZ    R0,_Shift
_Latch:     CLR     pLatch
            SETB    pLatch
            RET

            END
```

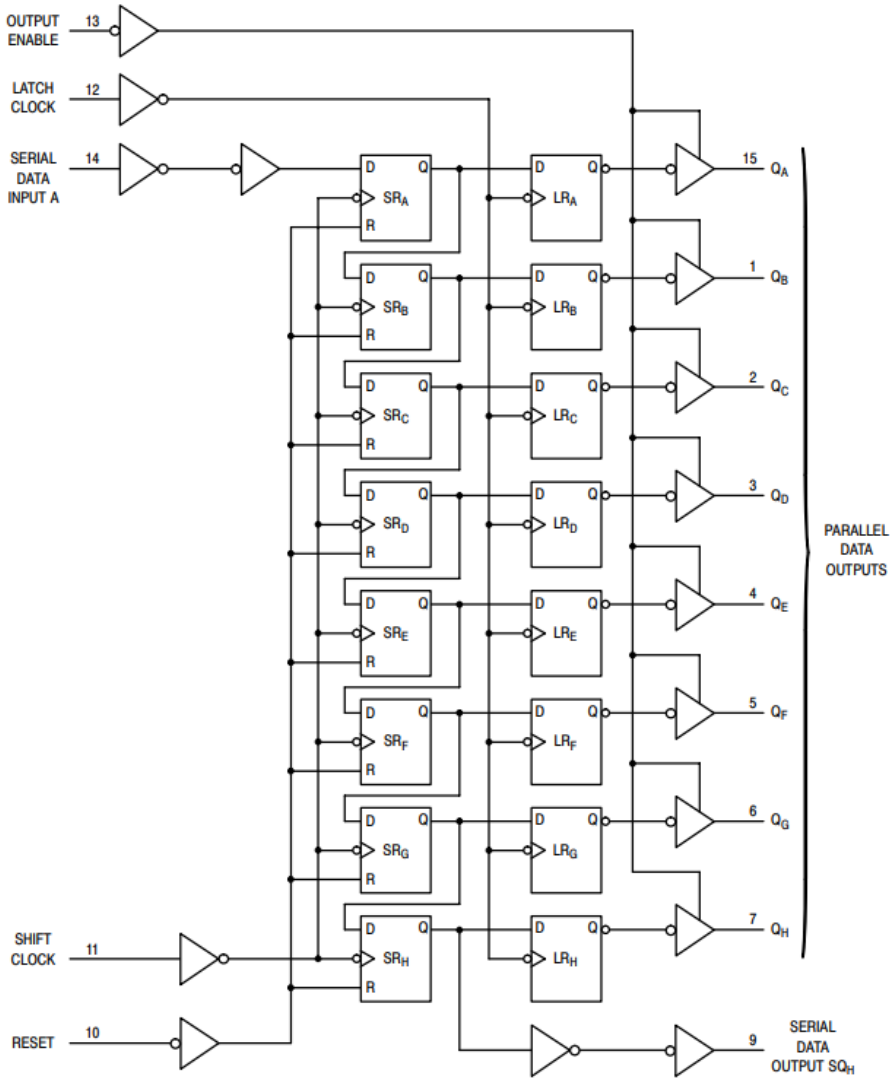


12. ปรับให้เป็นไฟวิ่ง 16 Bit จาก □ ซ้ายไปขวา, □ ขวาไปซ้าย, □ ซ้ายไปขวาและขวาไปซ้ายสลับไปมา (กำหนดให้ใช้สายสัญญาณควบคุมเพียง 3 เส้น คือ Data, Shift, Latch ต้องทำอย่างไร)
13. ต่อวงจรเพื่อทดสอบกับ 8 LED แบบอนุกรมโดยผ่าน ไอซี 74595 ซึ่งสามารถทำงานเป็น 74164 รวมกับ 74273 ได้
    - https://www.artronshop.co.th/article/30/การใช้งาน-7-segment-กับ-arduino-ตอนที่-1-7-segment-หลักเดียว
    - https://lastminuteengineers.com/74hc595-shift-register-arduino-tutorial/



## IC: 74595 8-bit serial-input/serial or parallel-output shift register

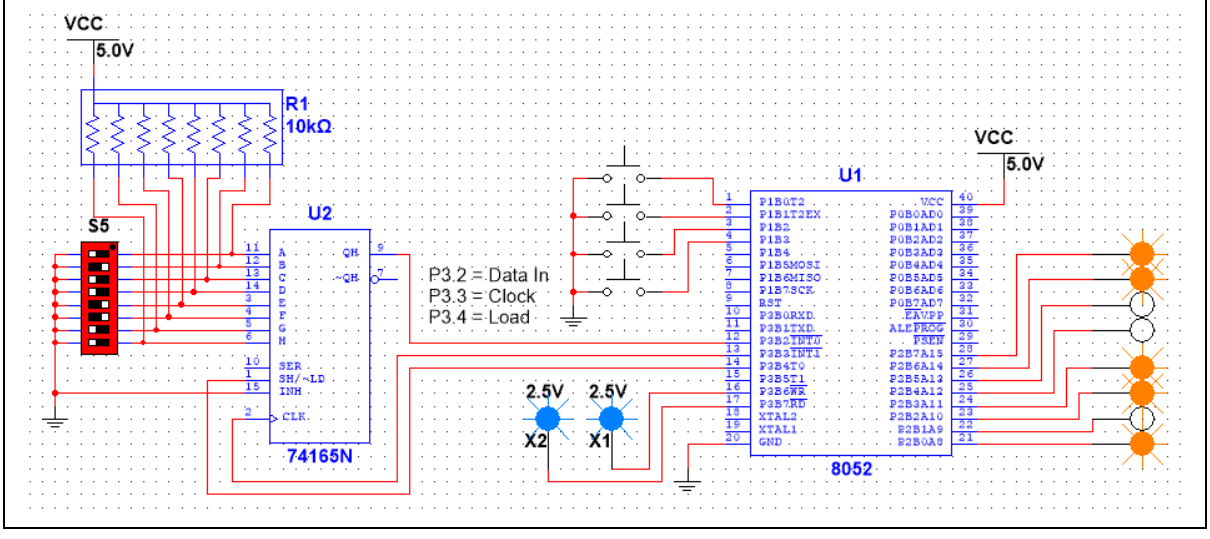


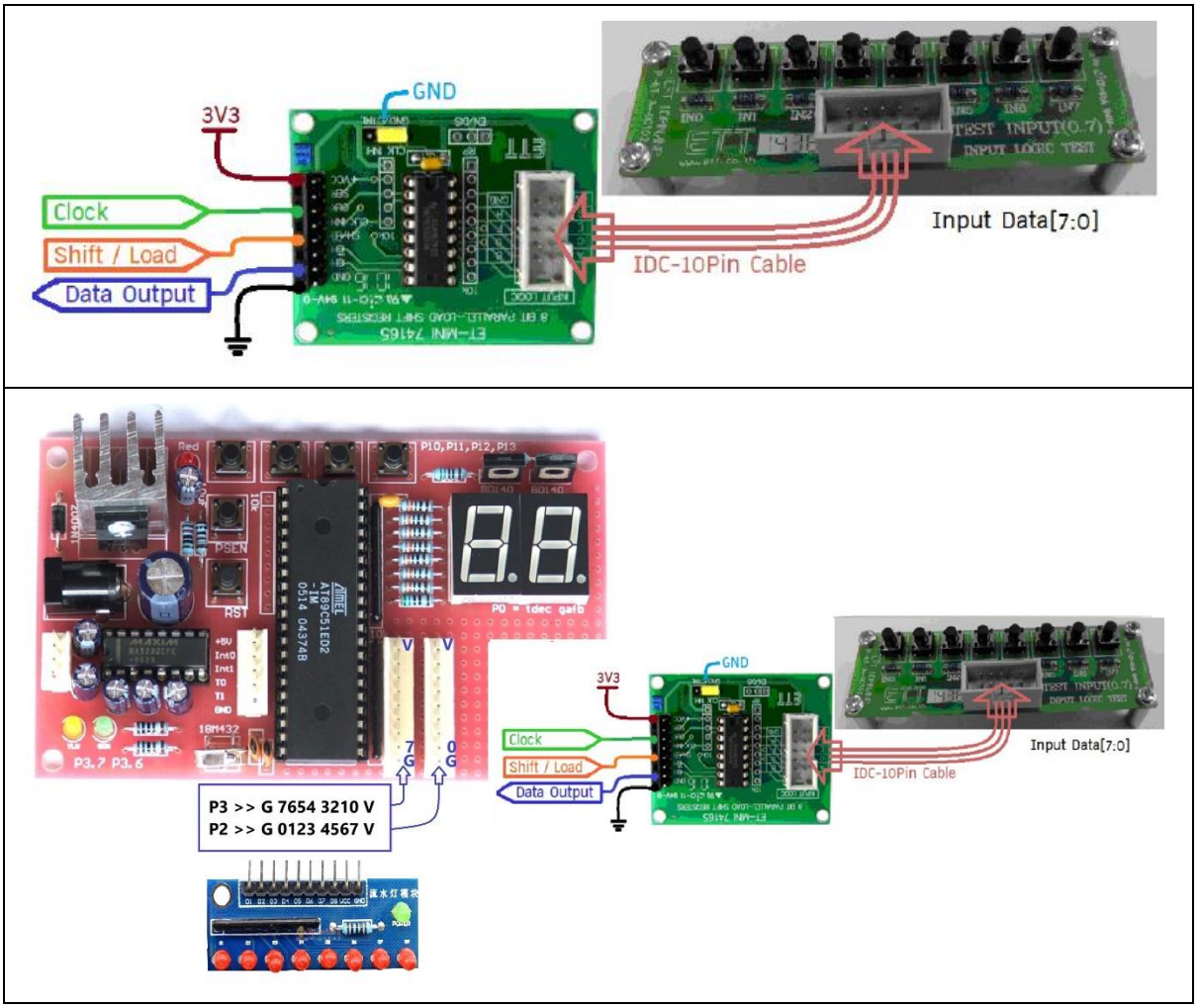
## 3. Serial Shift Input with 74165 (8-Bit Parallel In/Serial Out Shift Register)

14. จำลองการทำงานของ Lab1031

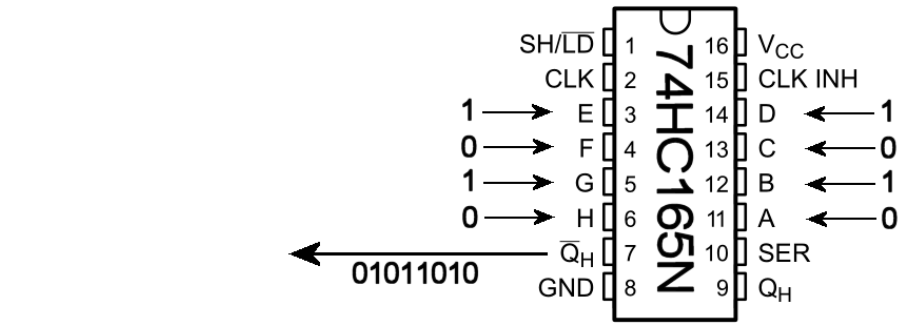
```
pLED_R      EQU     P3.6
pLED_G      EQU     P3.7
pData       EQU     P3.2
pClock      EQU     P3.3
pLoad       EQU     P3.4
Data01      EQU     20H

            ORG     0000H
            JMP     0100H

            ORG     0100H
            CLR     EA
            MOV     SP,#3FH
            SETB    pData

LOOP:       CALL    GET_Data
            MOV     A,Data01
            MOV     P2,A
            JMP     LOOP

GET_Data:
            CLR     pLoad
            SETB    pLoad
            MOV     R0,#8
_LP00:      MOV     C,pData
            RLC     A
            CLR     pClock
            SETB    pClock
            DJNZ    R0,_LP00
            MOV     Data01,A

            RET

            END
```

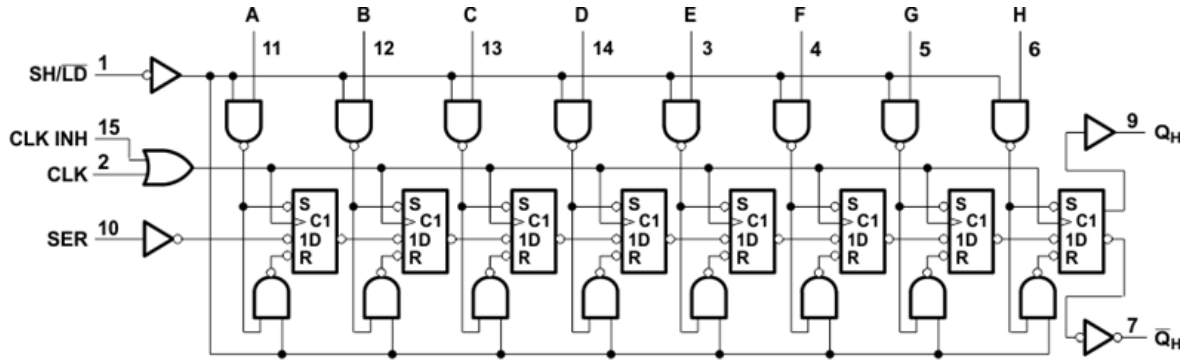


15. ปรับให้ปรับวงจรให้เป็นวงจรนำเข้าข้อมูลขนาด 16 บิต กำหนดให้ใช้สายสัญญาณควบคุมเพียง 3 เส้น คือ Load, Clock, Data ต้องทำอย่างไร
16. ต่อวงจรเพื่อทดสอบกับ 8 SW แบบอนุกรมโดยผ่าน ไอซี 74165



IC 74165 (8-Bit Parallel In/Serial Out Shift Register)



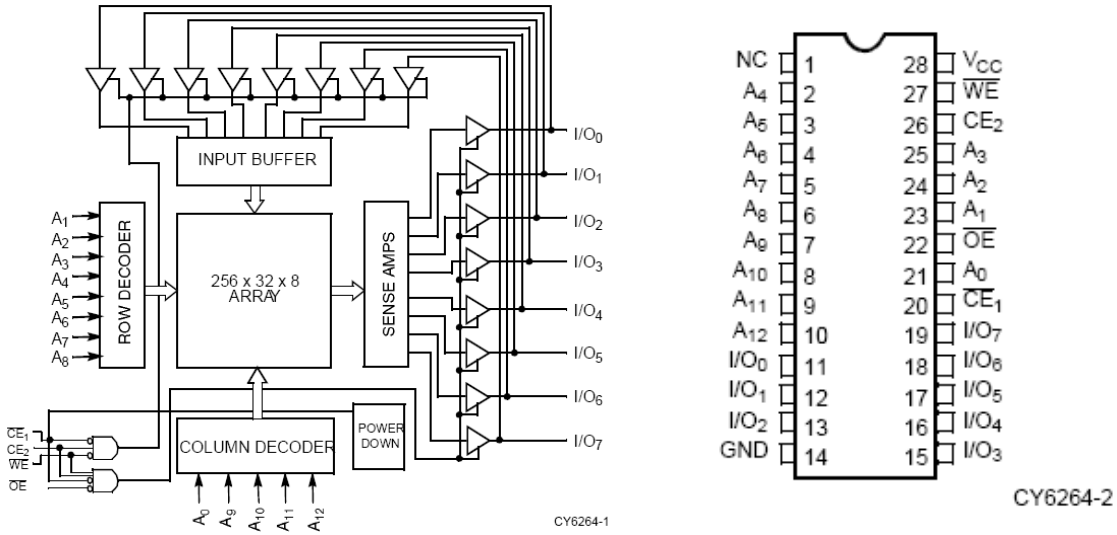




Pin numbers shown are for theD, DB, J, N, NS, PW and W packages.

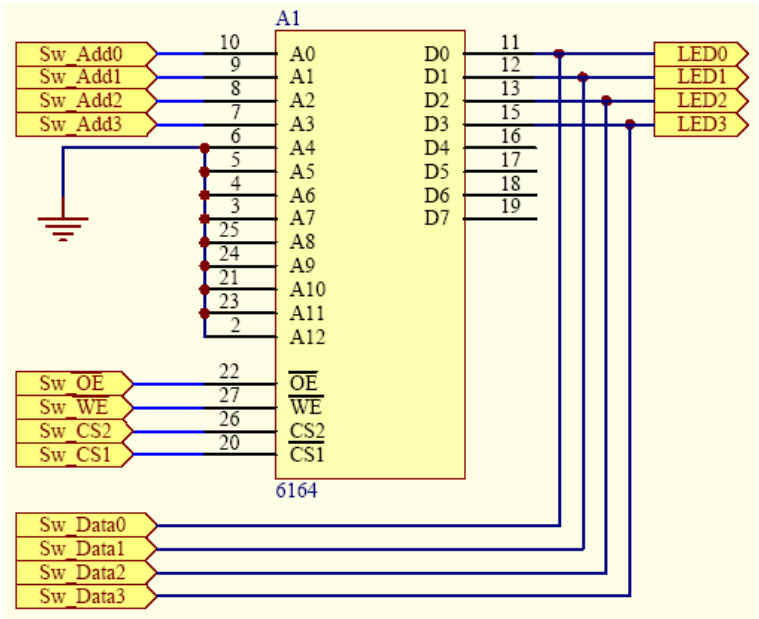
- https://microcontrollerslab.com/74ls166-shift-register-pinout-pinout-datasheet-examples-applications-features/
- https://cool-web.de/arduino/mit-dem-74hc165-schiebe-register-pins-und-leitungen-einsparen.htm

## 4. Parallel Write/Read RAM

17. การทดสอบนี้จะช่วยให้วิเคราะห์กระบวนการอ่านและเขียนข้อมูลใน RAM ตามไดอะแกรมเวลาได้อย่างละเอียด ซึ่งเป็นพื้นฐานสำคัญในการออกแบบวงจรควบคุม RAM ที่ใช้งานจริง โดยอาศัยขา Read, ขา Write และขาควบคุมอื่นๆ ของซีพียู
18. ประกอบวงจรตามรูปที่ 10.9




รูปที่ 10.8 หน่วยความจำแรมเบอร์ 6264

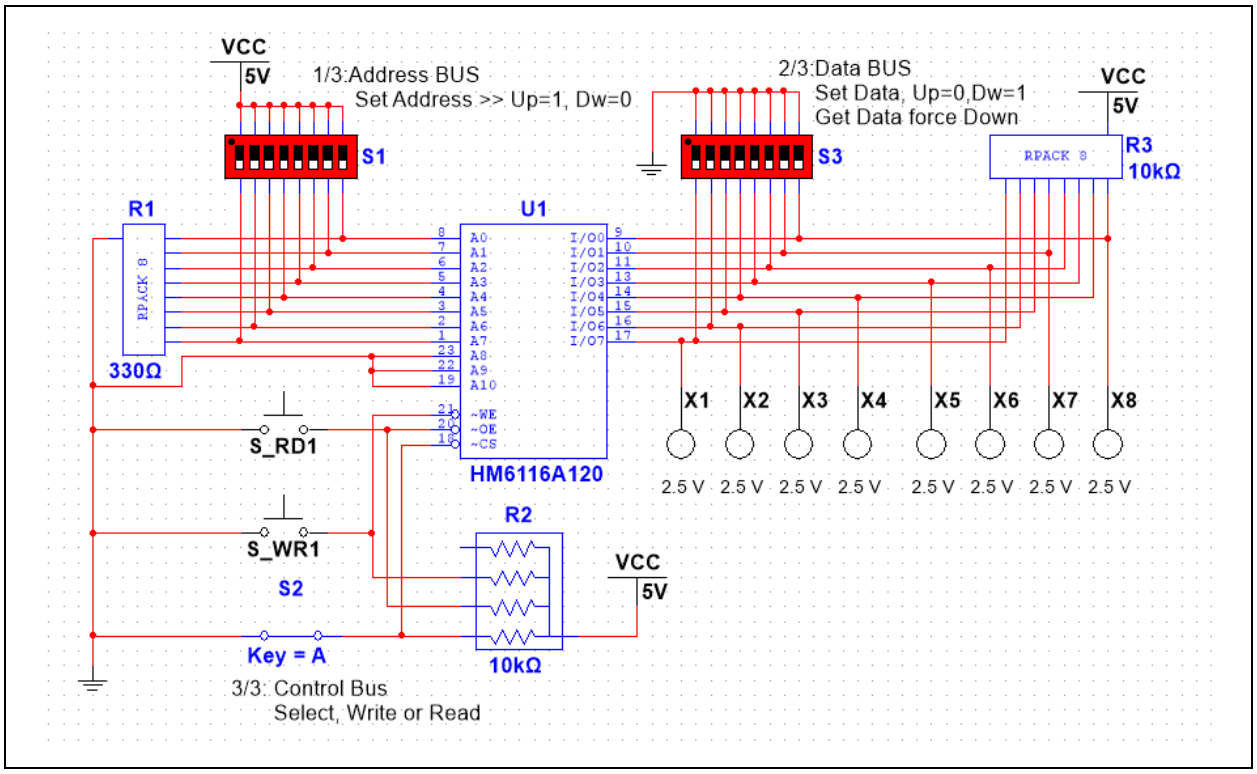



หมายเหตุ: ขาที่ระบุจะไม่ตรง ให้ดูรูปที่ 10.8 ประกอบในการต่อวงจร
อย่าลืม Vcc = 5Volt (Pin28) และ GND (Pin14)

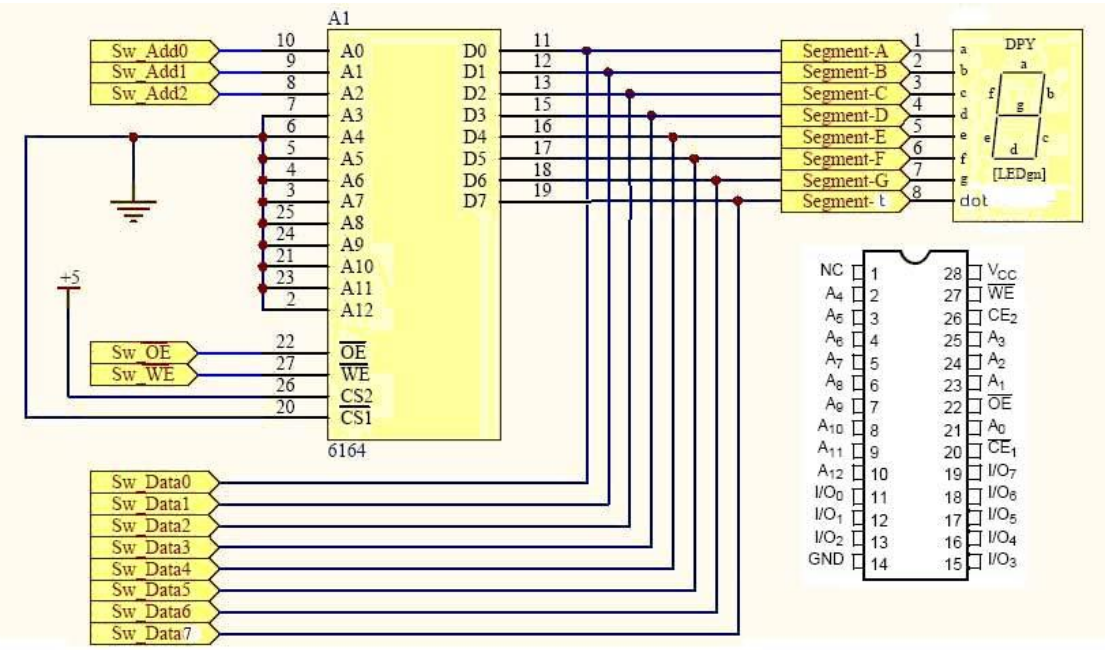
รูปที่ 10.9 การต่อหน่วยความจำแรมเบอร์ 6264

19. ป้อนข้อมูลเข้าไปเก็บไว้ในหน่วยความจำแรมตามลำดับ ดังนี้
    - 19.1 ก่อนเขียนข้อมูลที่แรมแต่ละชุด ต้องป้อนระดับลอจิกให้ขา nCS1=1, CS2=0, nWE=1 และ nOE=1
    - 19.2 กำหนดตำแหน่งแอดเดรส HGFE DCBA ด้วยระดับลอจิกเข้าที่ขา Sw_Add[7..0] ตามตารางที่ 1
    - 19.3 ป้อนข้อมูลที่ต้องการเก็บเข้าที่ ขา D1-D7 เข้าที่ขา Sw_Data[0..7] ตามตารางที่ 1
    - 19.4 การเขียนข้อมูลลงในแรมทีละแอดเดรส โดยป้อนลอจิกให้ขา nCS1=0, CS2=1 จากนั้นตามด้วยลอจิก 0 ให้ขา nWE เป็นสภาวะเขียนข้อมูล (Write Enable)
    - 19.5 ป้อนลอจิกให้ขา nCS1=1, CS2=0, nWE=1 และ nOE=1 เป็นการเสร็จสิ้นการเขียนข้อมูลใน 1 แอดเดรส
    - 19.6 ปฏิบัติตามขั้นตอนที่ 19.1 ถึง 19.5 ในทุกๆ แอดเดรส ตามตารางที่กำหนด

20. ทดลองอ่านข้อมูลจากหน่วยความจำแรม โดยใช้วงจรตามข้อ 1 ปฏิบัติตามขั้นตอนดังนี้
    - 20.1 จากรูปที่ 10.9 ให้ปลดขา Sw_Data[7..0] ออกให้เหลือเฉพาะด้านที่เป็น LED[7..0]
    - 20.2 ก่อนอ่านข้อมูลที่แรม ต้องป้อนระดับลอจิกให้ขา nCS1=1, CS2=0, nWE =1 และ nOE=1 ไว้ก่อน
    - 20.3 กำหนดตำแหน่งแอดเดรสที่ต้องการอ่านข้อมูลด้วยระดับลอจิกตามตารางที่กำหนด
    - 20.4 ป้อนลอจิกให้ขา nCS1=0 และ CS2=1 เพื่อเลือกให้ไอซีทำงาน(Chip Enable)
    - 20.5 ป้อนลอจิกให้ขา nOE=0 เพื่ออ่านข้อมูล(Output Enable)
    - 20.6 บันทึกค่า DO7 ถึง DO0 จากขา LED[7..0] ลงในตาราง
    - 20.7 ป้อนลอจิก 1 ให้ขา nOE เป็นการเสร็จสิ้นการอ่านข้อมูลใน 1 แอดเดรส
    - 20.8 เปลี่ยนค่าแอดเดรสตามตารางการทดลอง ปฏิบัติตามขั้นตอน 20.2 ถึง 20.7 ในทุก ๆ แอดเดรส

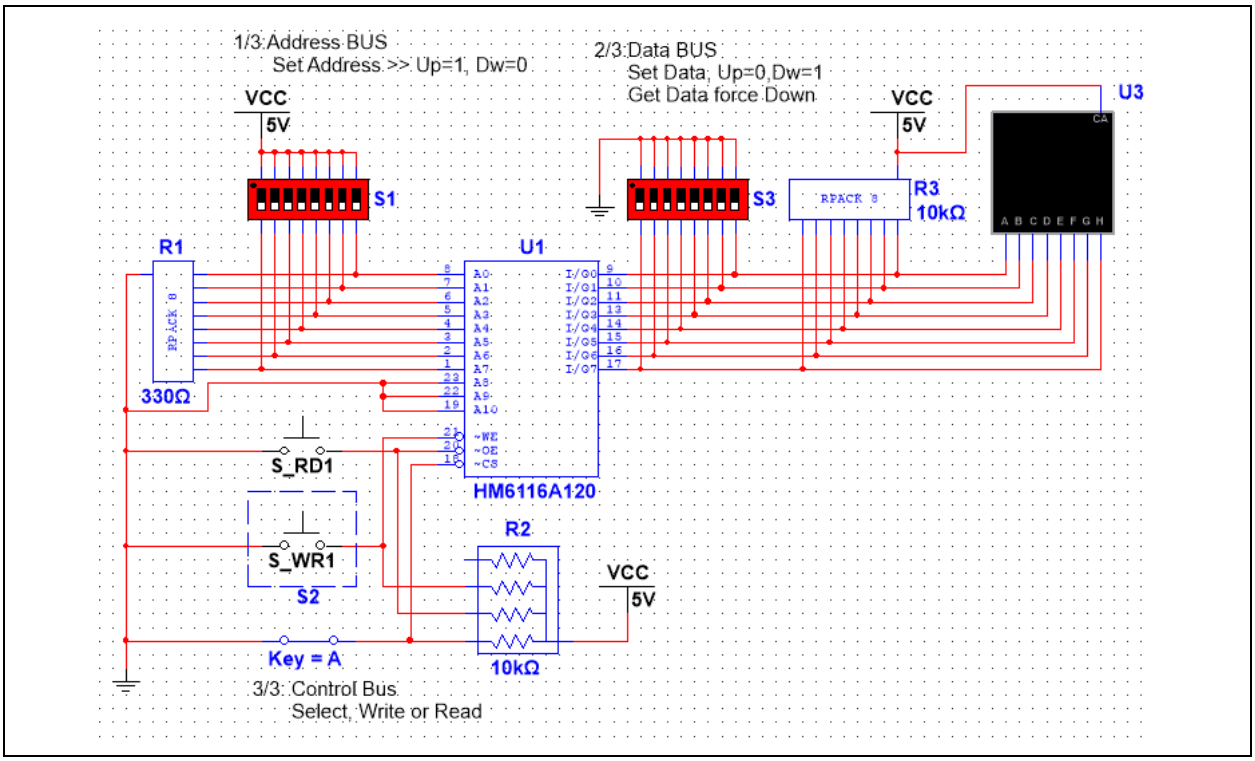


21. ให้ต่อวงจรดังรูปเพื่อป้อนค่าไปที่ Address ต่างๆ ตามตาราง โดยวิเคราะห์ข้อมูลที่จะป้อนให้เป็นไปตามที่ต้องการแสดงบน



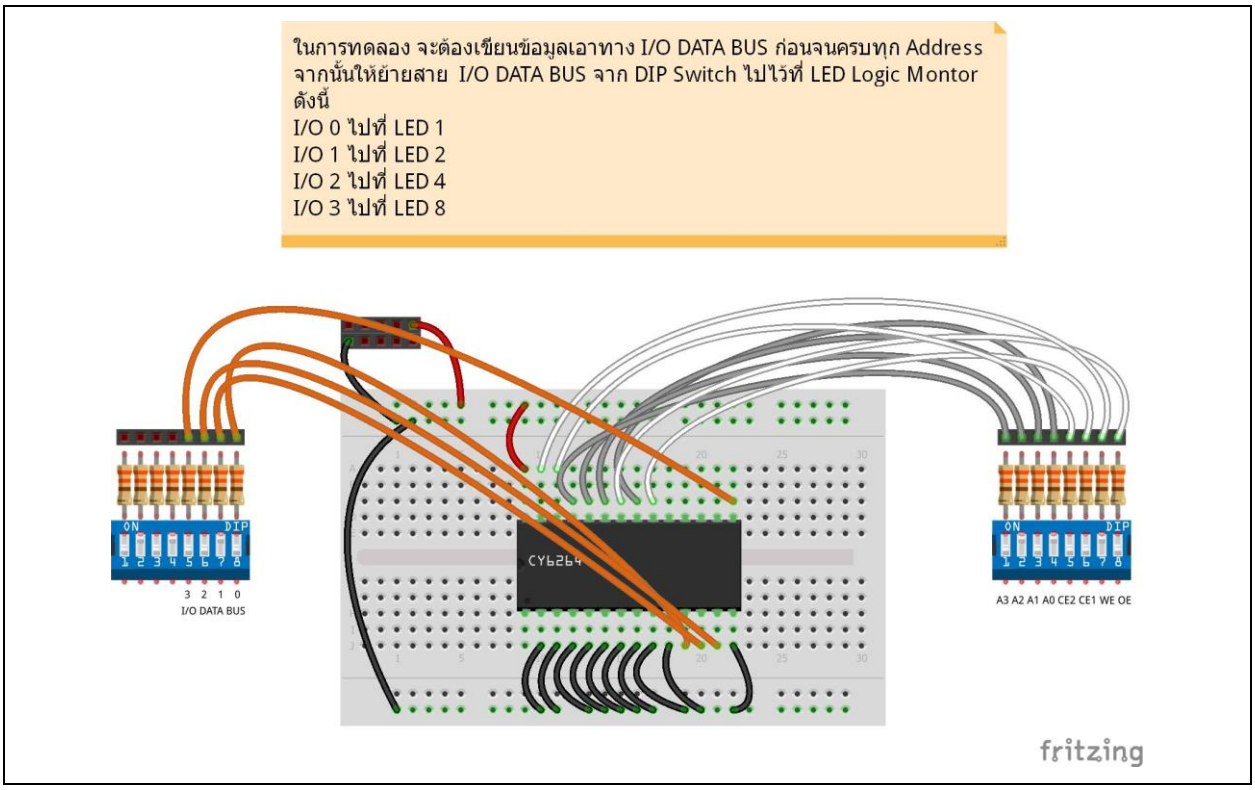

รูปที่ 10.10 การต่อหน่วยความจำแรมเบอร์ 6264 กับ 7_Segment

| Item | Address | Sw_Data D7,D6,D5,D4,D3,D2,D1,D0 | 7_Segment Display | Note |
|---|---|---|---|---|
| 0 | 0000 0000 | | C | |
| 1 | 0000 0001 | | P | |
| 2 | 0000 0010 | | E. | with dot |
| 3 | 0000 0011 | | _ | Under Score |
| 4 | 0000 0100 | | S | |
| 5 | 0000 0101 | | U | |
| 6 | 0000 0110 | | t. | with Dot |
| 7 | 0000 0111 | | | Blank |

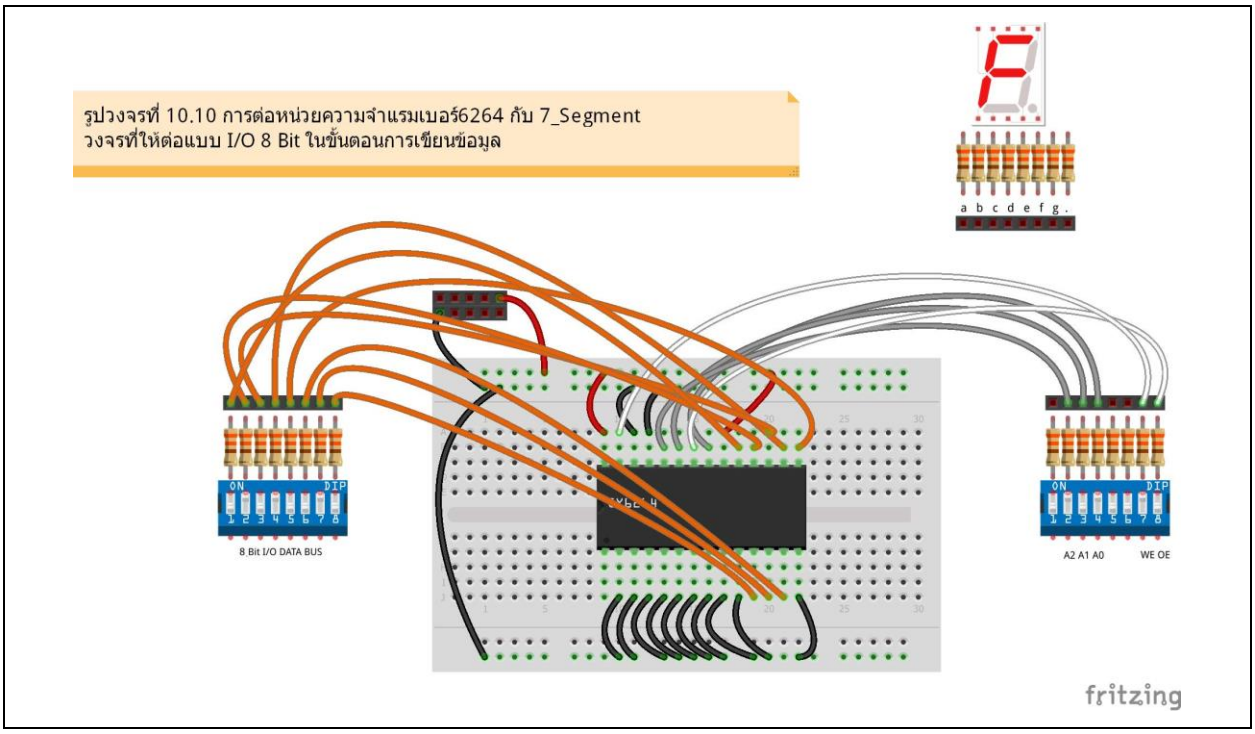



22. ต่อวงจรเพื่อทดสอบกับ RAM

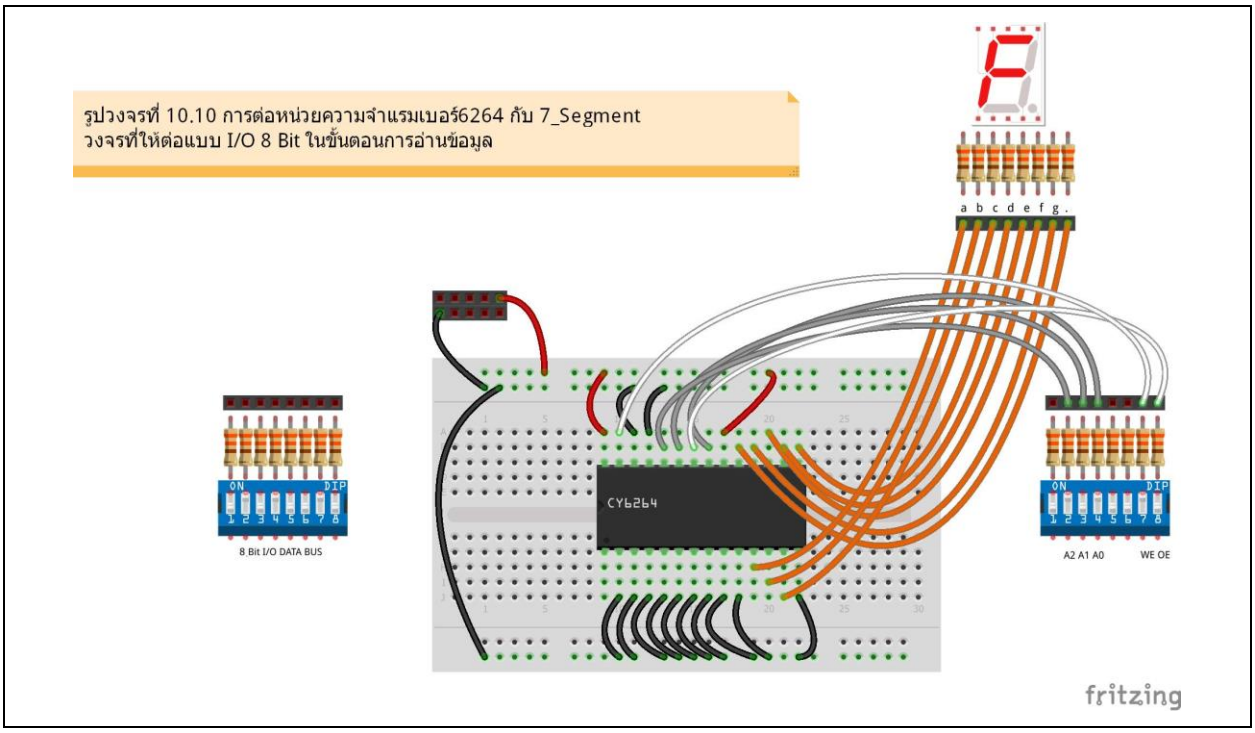
- เตรียมวงจร



- เวลาจะเขียนข้อมูลเก็บใน RAMจะต้องนำ I/O ทั้งหมดมาไว้ที่ Dip SW




- เมื่อเขียนข้อมูลลงทุก Address แล้ว เวลาจะอ่านให้ย้าย I/O ทั้งหมดไปที่7 Segment เพื่อแสดงผลข้อมูลของแต่ละ Address ที่ได้เขียนลงไปใน RAM



## 6. สรุปผลการทดลอง

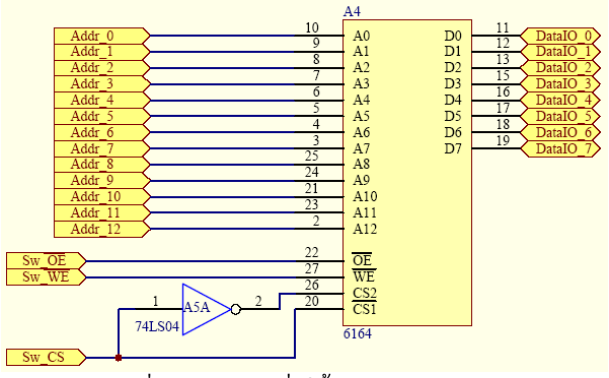
(1) ประเด็นขั้นตอนการทำ Serial Shift Input

(2) ประเด็นขั้นตอนการทำ Serial Shift Output

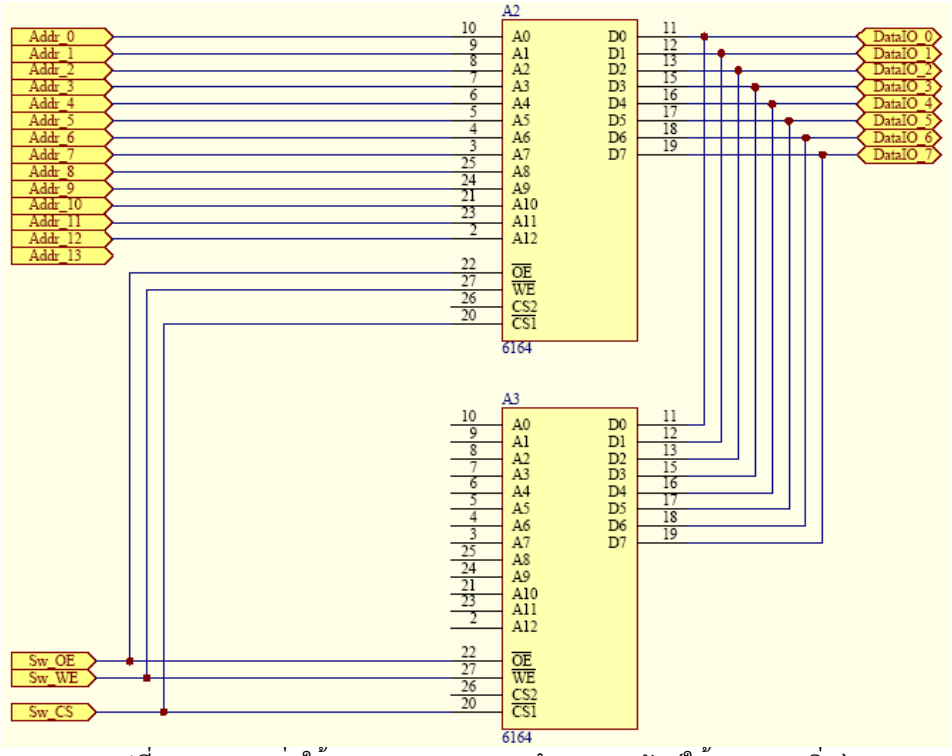
(3) ประเด็นขั้นตอนการอ่านและเขียน RAM

## 7. คำถามท้ายการทดลอง

1. SRAM เบอร์ 6264 1 ตัวมีขนาดความจุกี่กิโลไบต์ _____
2. SRAM เบอร์ 6264 1 ตัวมีขนาดความจุกี่กิโลบิต_____
3. SRAM เบอร์ 6264 1 ตัวอ้างอิง Address จาก A12–A0(จำนวน 13 เส้น) นั่นคือ 2^13 / 1024 = 8 กิโลไบต์ ดังรูปที่ 10.10 หากต้องการหน่วยความจำ SRAM ขนาด 16 กิโลไบต์ ต้องใช้ SRAM เบอร์ 6264 จำนวน 2 ตัว อ้างอิงแอดเดรส 2^n = 16*1024 → n=14 เส้น (จาก A13-A0) อยากทราบว่าจะต่อใช้งานอย่างไร




รูปที่ 10.10 การต่อใช้งาน SRAM 6364 จำนวน 1 ตัว




รูปที่ 10.11 การต่อใช้งาน SRAM 6364 จำนวน 2 ตัว (ให้วาดสายเพิ่ม)

4. การทดลองสำหรับข้อ 12 มีวงจรและโปรแกรมอย่างไร ( ข้อ 12 Serial Shift Output 16 Bit: ปรับให้เป็นไฟวิ่งจาก □ ซ้ายไปขวา, □ ขวาไปซ้าย, □ ซ้ายไปขวาและขวาไปซ้ายสลับไปมา กำหนดให้ใช้สายสัญญาณควบคุมเพียง 3 เส้น คือ Data, Shift, Latch ต้องทำอย่างไร)

5. การทดลองสำหรับข้อ 15 มีวงจรและโปรแกรมอย่างไร (ข้อ 15 Serial Shift Input 16 Bit: ปรับให้ปรับวงจรให้เป็นวงจรนำเข้าข้อมูลขนาด 16 บิต กำหนดให้ใช้สายสัญญาณควบคุมเพียง 3 เส้น คือ Load, Clock, Data ต้องทำอย่างไร )

6. ปรับโปรแกรมและวงจรให้รับค่าจาก Serial Shift Input 8 Bit แล้วส่งออกที่ Serial Shift Output 8 Bit

7. ปรับโปรแกรมและวงจรให้ส่งออกรหัสนักศึกษาที่ Serial Shift Output 32 Bit ดังรูป