CS231 ระบบคอมพิวเตอร์และภาษาแอสแซมบลี

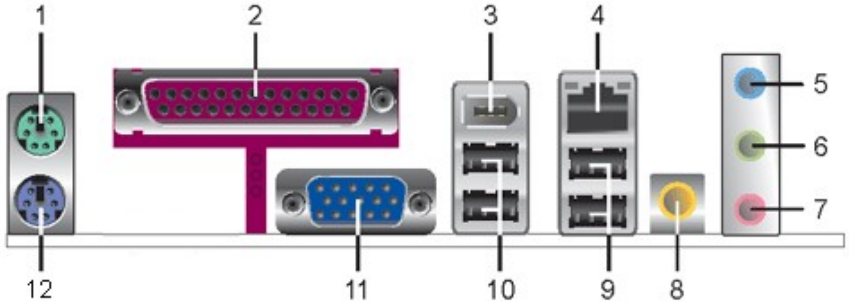
บทที่ 8: การรับเข้า/ส่งออก

ผู้ช่วยศาสตราจารย์ ดร. ปวีณ เขื่อนแก้ว

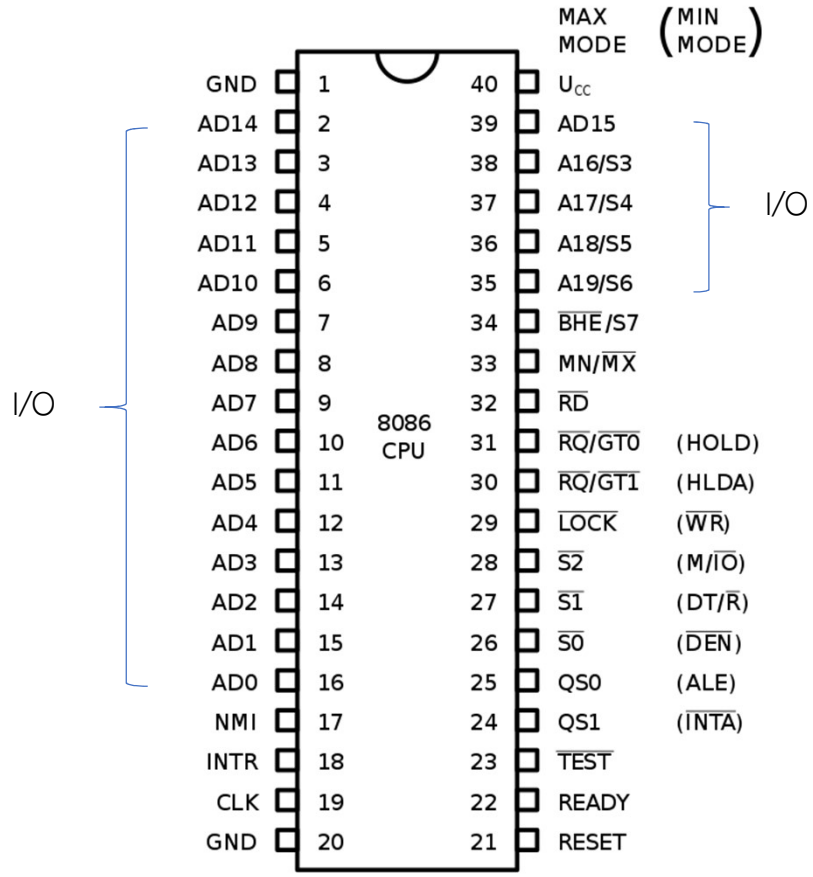
สาขาวิชาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์ มหาวิทยาลัยแม่โจ้

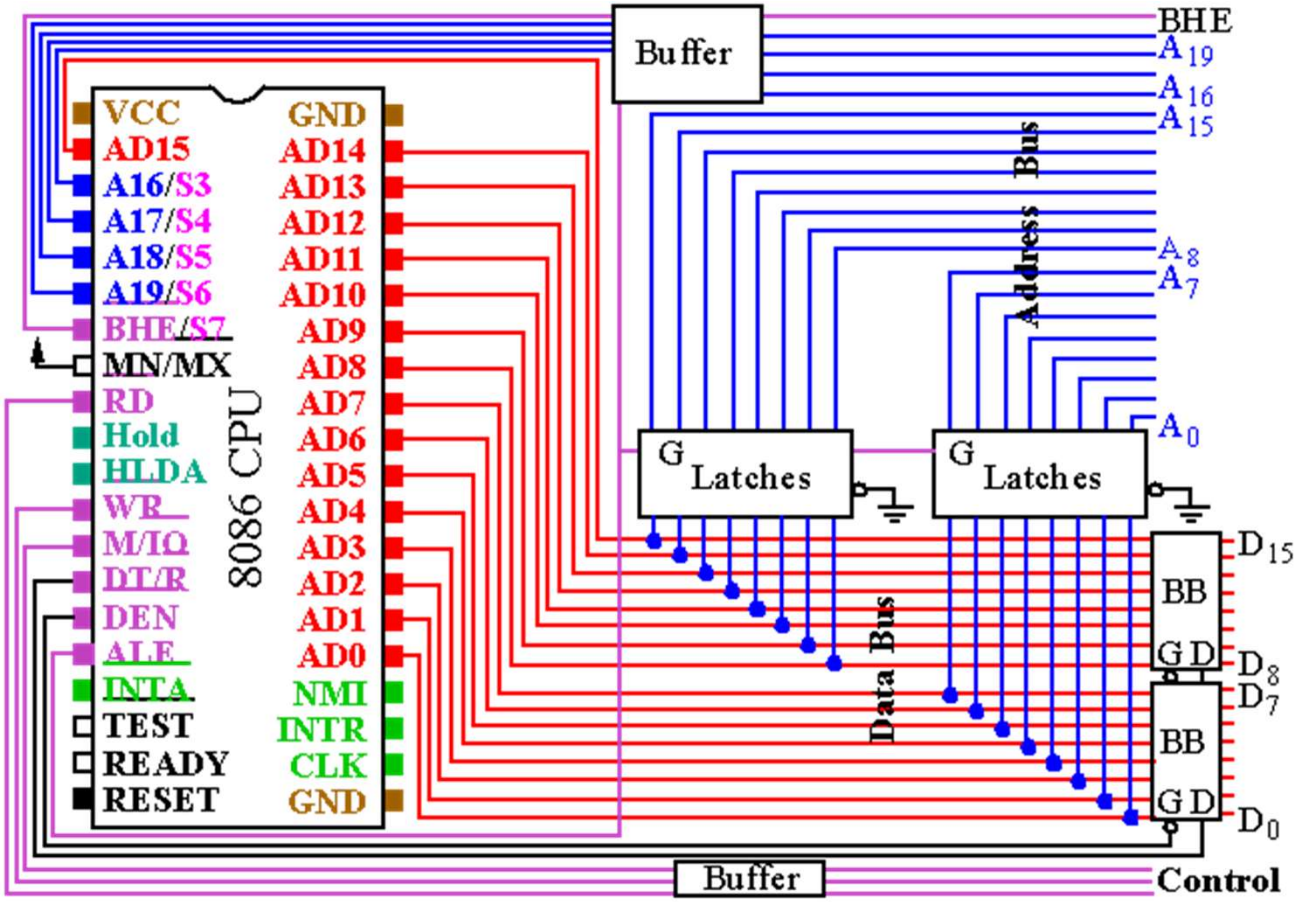
อินพุต/เอาต์พุต ย่อว่า ไอ/โอ (อังกฤษ: input/output: I/O) หรือภาษาไทยว่า **รับเข้า/ส่งออก** ในทางคอมพิวเตอร์ หมายถึงการสื่อสารระหว่างระบบประมวลผลสารสนเทศ (เช่นคอมพิวเตอร์) กับโลกภายนอก ซึ่งอาจเป็นมนุษย์หรือระบบประมวลผลสารสนเทศอีกระบบหนึ่ง อินพุตหรือสิ่งรับเข้าคือสัญญาณหรือข้อมูลที่ระบบรับเข้ามา และเอาต์พุตหรือสิ่งส่งออกคือสัญญาณหรือข้อมูลที่ระบบส่งออกไป



อินพุต/เอาต์พุต ของไมโครโพรเซสเซอร์ สำหรับเชื่อมต่อกับอุปกรณ์ภายนอก จะมี 2 ชุดคือ
Data pin ขนาด 16 บิต และ Address pin ขนาด 20 บิต

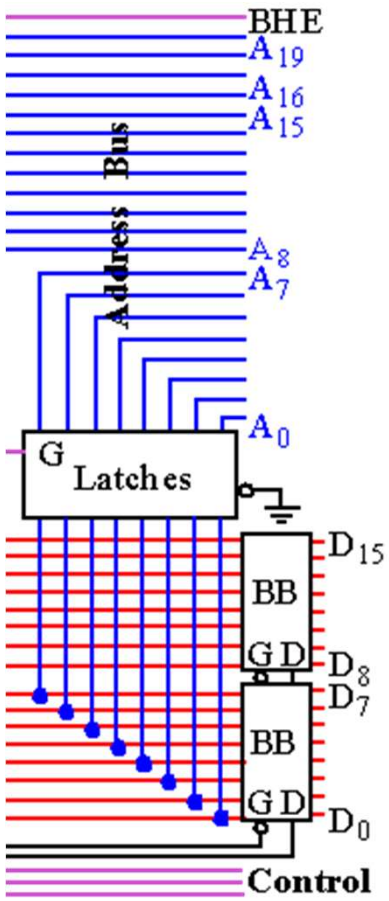


ในการต่อใช้งานจะมีวงจร คงข้อมูล (Latch) และพักข้อมูล(Buffer) เพื่อแยกสัญญาณ Address และ Data ออกจากกัน



http://ece-research.unm.edu/jimp/310/slides/8086_chipset.html

ไมโครโพรเซสเซอร์ใช้ ช่องสัญญาณ Address Bus และ Data Bus ในการสื่อสารกับอุปกรณ์ภายนอก เช่น หน่วยความจำ และ ชิพควบคุม Interrupt



Address Bus

Data Bus

การเชื่อมต่อกับอุปกรณ์ I/O ภายนอก ก็เชื่อมต่อผ่าน Address Bus และ Data Bus เช่นกัน

แต่การเชื่อมต่อ จะไม่สามารถกระทำได้โดยตรง เพราะ ช่องสัญญาณนี้ถูกใช้งานไปแล้วโดยหน่วยความจำ

ดังนั้นต้องทำการดัดแปลง ช่องสัญญาณนี้ให้กลายเป็น I/O สำหรับสื่อสารกับอุปกรณ์อื่นที่ไม่ใช่ หน่วยความจำ และ ชิพควบคุม Interrupt เรียกว่าการทำ I/O Mapping

8086 สามารถทำ I/O Mapping ได้ 2 วิธีคือ

- Memory-mapped IO
- Standard IO

Intel only

http://ece-research.unm.edu/jimp/310/slides/8086_chipset.html

8086 สามารถทำ I/O Mapping ได้ 2 วิธีคือ

- Memory-mapped IO
- Standard IO

ทั้ง 2 แบบจะใช้ Address Bus เป็นตัวกำหนดตำแหน่งของจุดเชื่อมต่อ
และใช้ Data Bus สำหรับรับ/ส่งข้อมูล

แต่จุดที่แตกต่างกันคือที่มาของสัญญาณควบคุมการ อ่าน/เขียน ข้อมูล

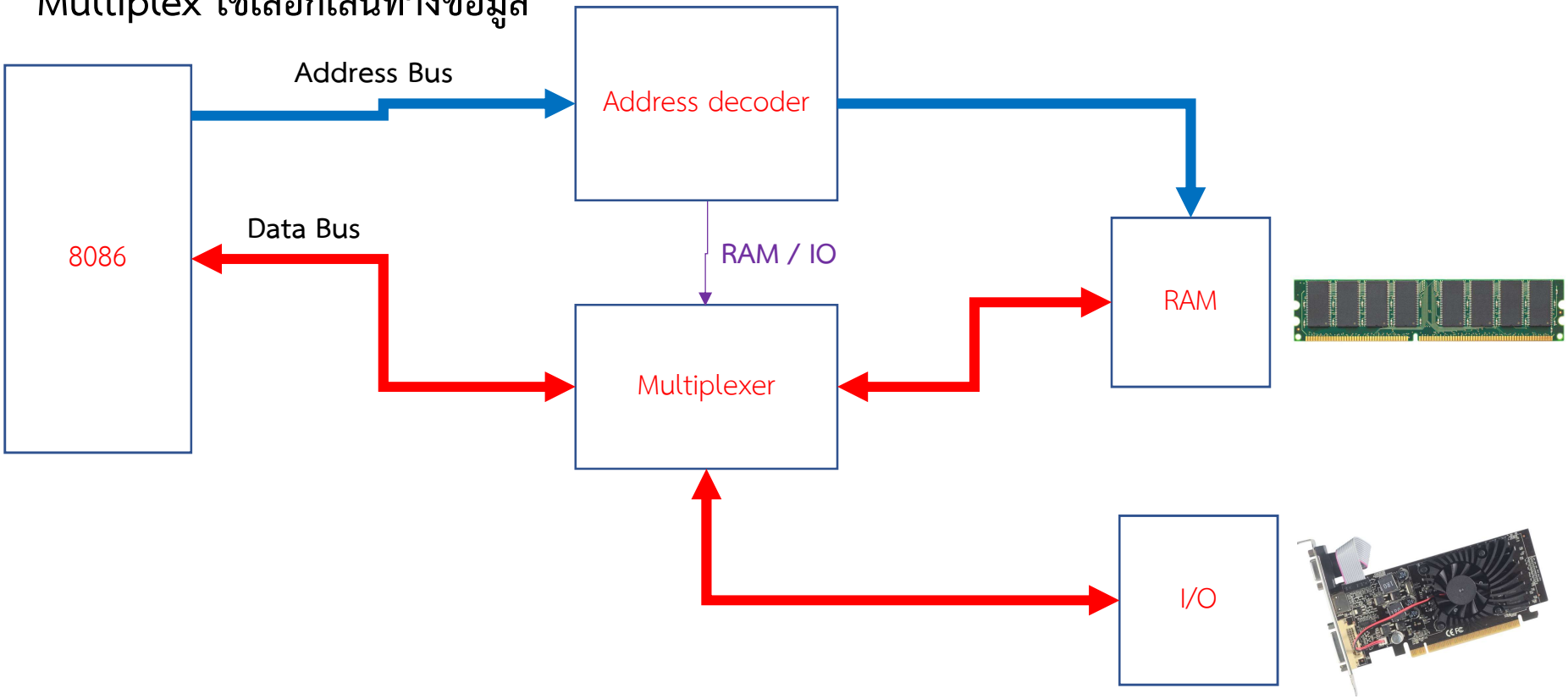
PC อาจมีทำ I/O Mapping ผสมกันได้ทั้ง 2 แบบ ทั้งนี้ขึ้นอยู่กับการออกแบบ Motherboard

Memory-mapped IO เป็นการเชื่อมต่อกับอุปกรณ์ภายนอก ใช้สัญญาณควบคุมหน่วยความจำเป็นตัวกำหนดการ อ่าน / เขียน ข้อมูล

โดยเลือกนำบางตำแหน่ง (Address) มาเป็น I/O หรือ เพื่อเชื่อมต่อกับอุปกรณ์ภายนอก

วงจร Decoder ใช้ในการตรวจหาตำแหน่งที่ต้องการอ่านหรือเขียน

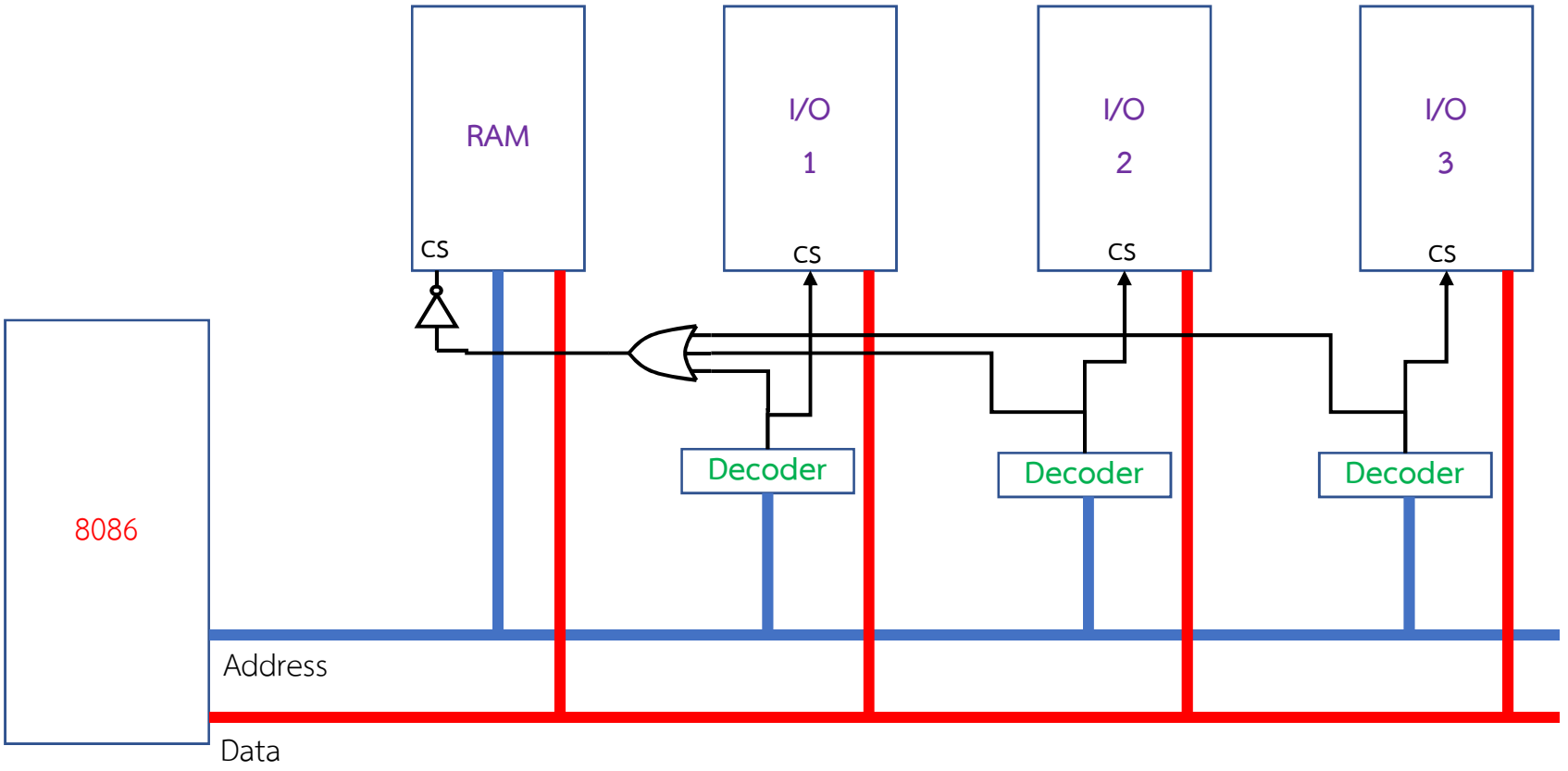
Multiplex ใช้เลือกเส้นทางข้อมูล



ไมโครชิพที่ออกแบบมาสำหรับเชื่อมต่อกับไมโครโพรเซสเซอร์ จะมี ขา CS (chip select) สำหรับเลือกให้ต่อ หรือตัดตัวเองออกจากวงจร ทำให้งานทำ I/O Mapping ทำได้ง่าย

การอ่าน และเขียนข้อมูล ใช้วิธีเดียวกับหน่วยความจำ

คือใช้คำสั่ง MOV



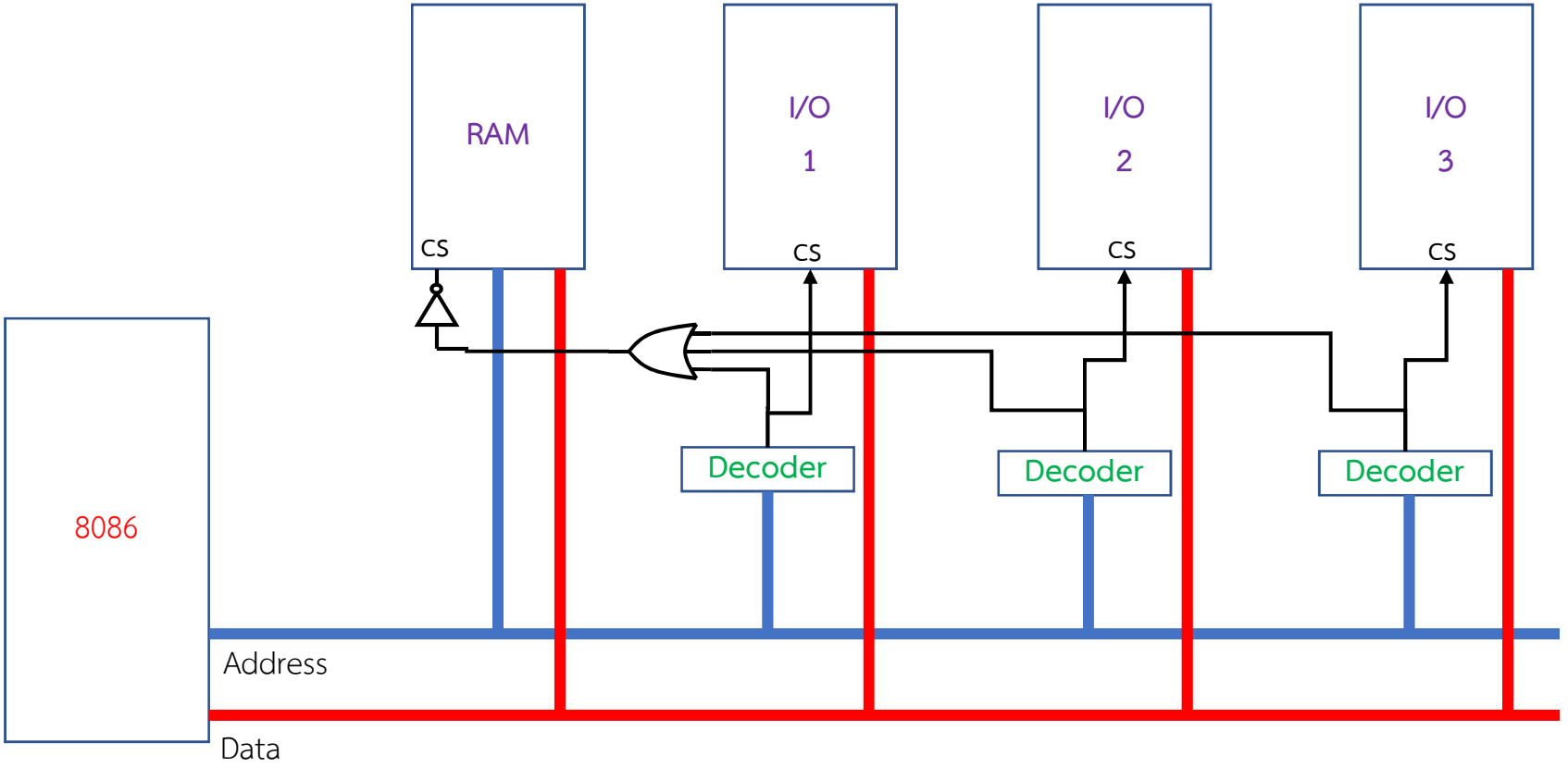
Memory-mapped I/O

ข้อดี

- ใช้ Instruction เดียวกับหน่วยความจำ เขียนโปรแกรมง่าย
- ไมโครโพรเซสเซอร์ทุกรุ่นที่รองรับการต่อหน่วยความจำภายนอก ใช้วิธีนี้ได้

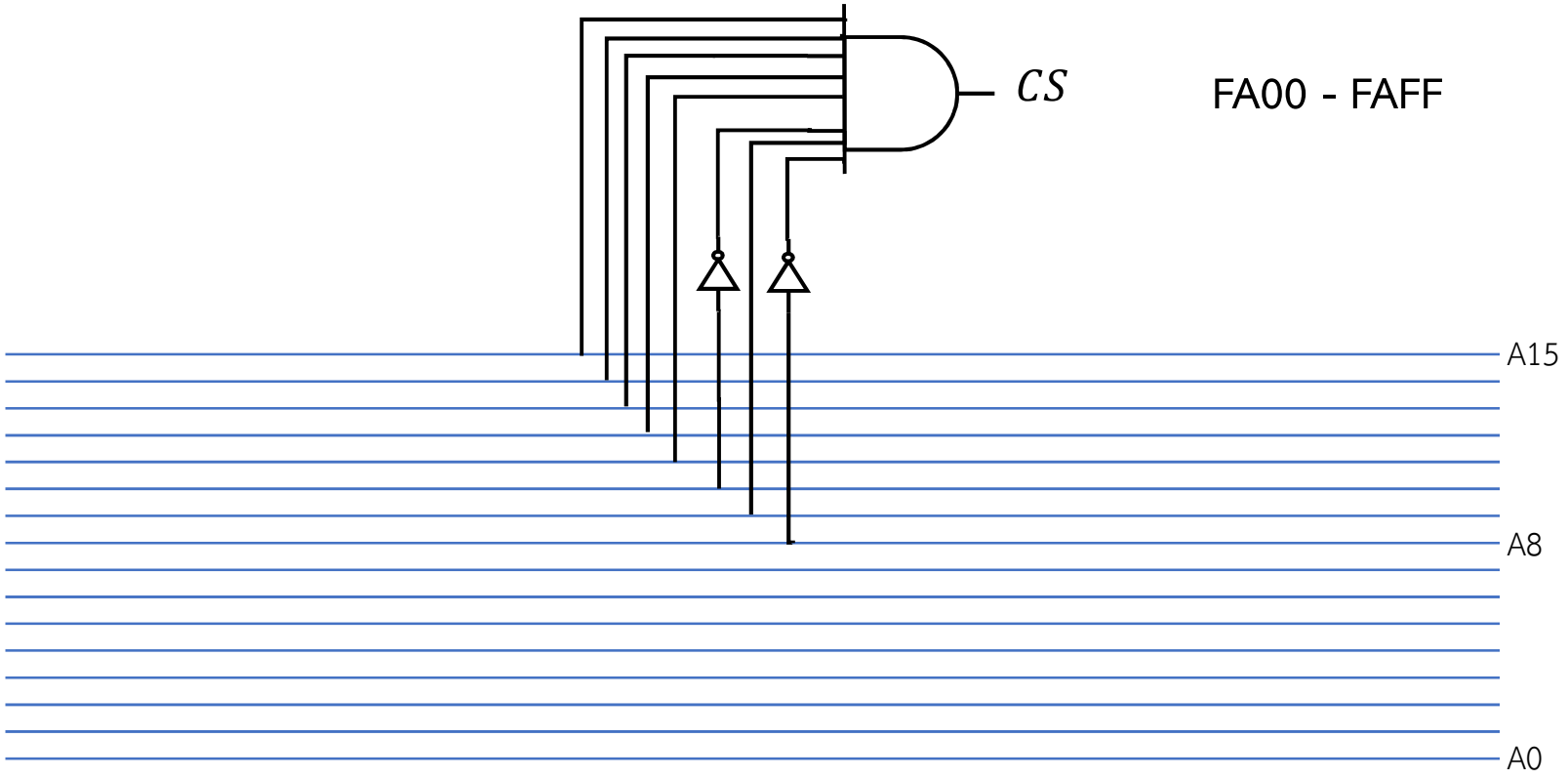
ข้อเสีย

- ตำแหน่งที่ใช้เป็น I/O จะใช้เป็นหน่วยความจำไม่ได้อีก ทำให้ใช้หน่วยความจำภายนอกได้ ไม่เต็มความจุ

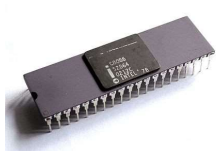


Memory-mapped IO มักไม่นิยมเลือกตำแหน่งทีละตำแหน่ง แต่จะใช้วิธีเลือกเป็นช่วง เพราะทำได้ง่ายกว่า
ตัวอย่าง ต้องการให้ FA00 ถึง FAFF สามารถสร้างสัญญาณ CS ได้ด้วยวงจรนี้

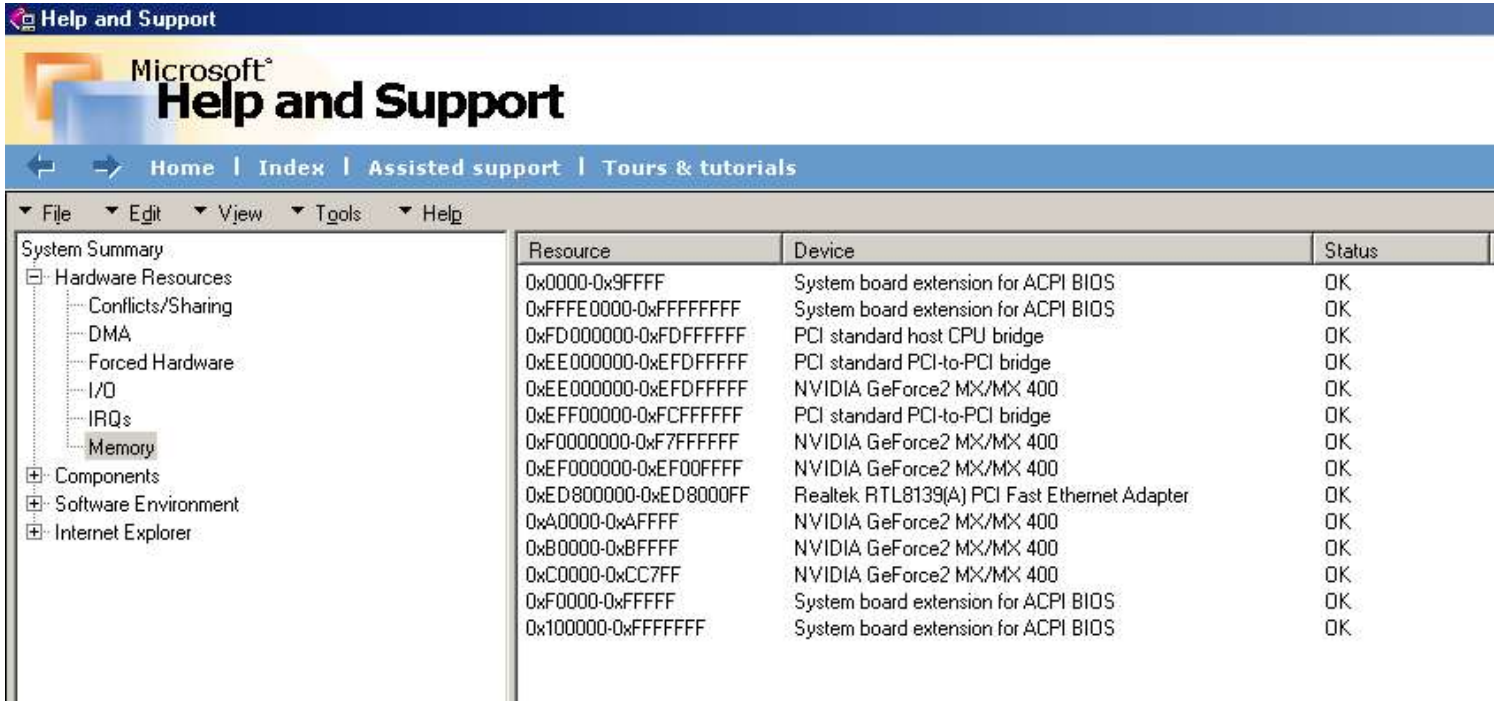
$$CS = A_{15} \wedge A_{14} \wedge A_{13} \wedge A_{12} \wedge A_{11} \wedge \overline{A_{10}} \wedge A_9 \wedge \overline{A_8}$$



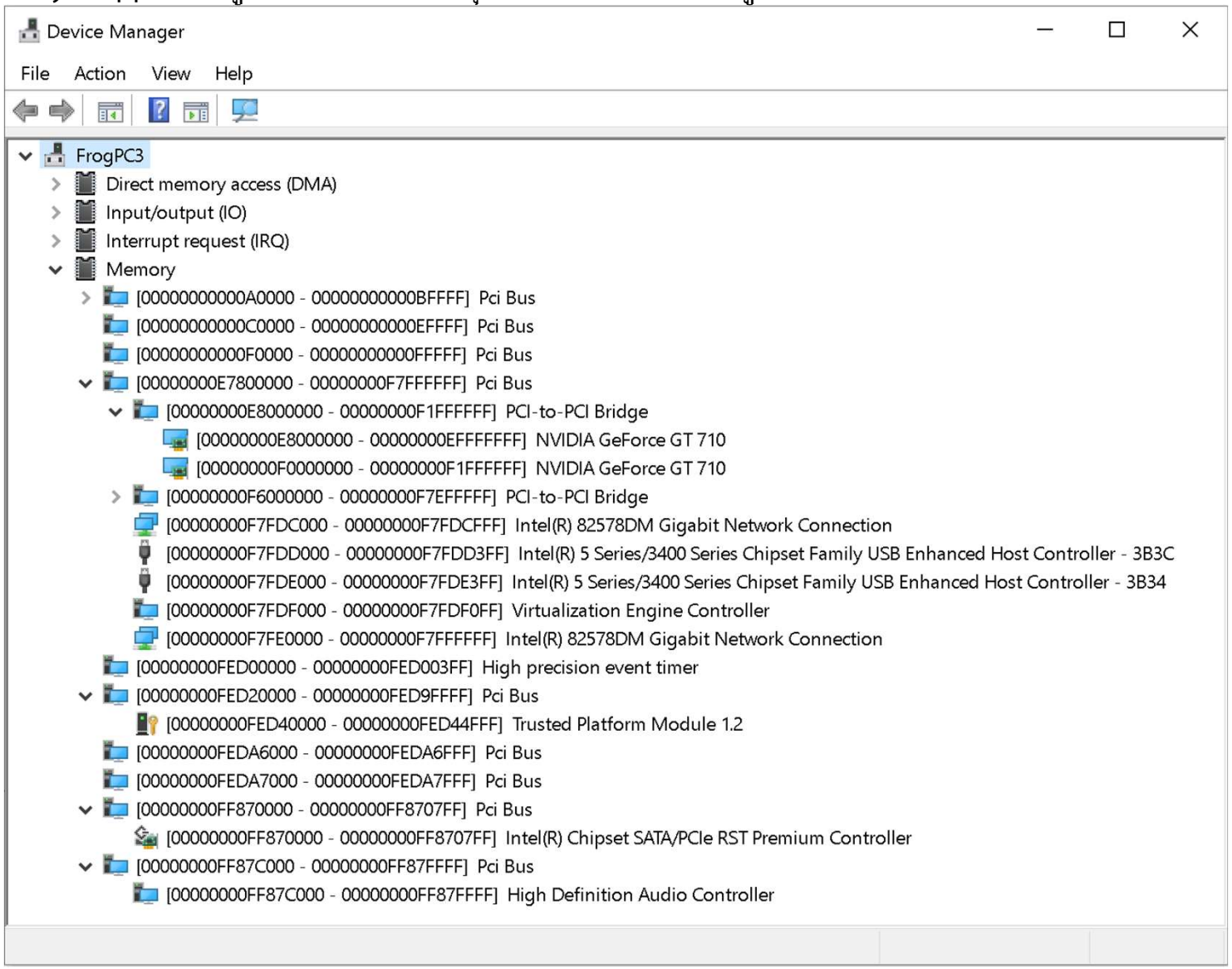
F0000	BIOS
E0000	BIOS
D0000	VRAM การ์ดจอ
C0000	VRAM การ์ดจอ
A0000	VRAM การ์ดจอ
90000	
80000	
70000	
60000	
50000	
40000	
30000	
20000	
10000	
00000	00000-003FF Interrupt Vector Table



ตัวอย่างการใช้งาน Memory-mapped I/O ของเครื่อง PC



ปัจจุบันนี้ Memory-mapped IO ถูกใช้ในการสื่อสารกับอุปกรณ์ที่มีการเชื่อมต่อในรูปแบบ PCI



8086 สามารถทำ I/O Mapping ได้ 2 วิธีคือ

- **Memory-mapped IO**
- **Standard IO**

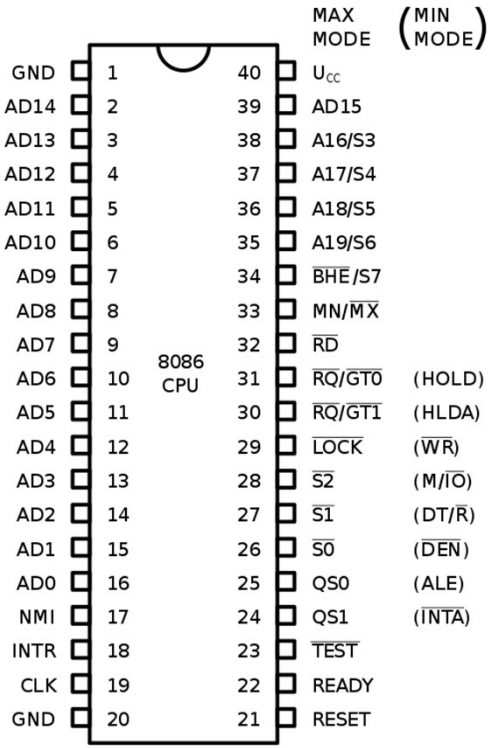
ทั้ง 2 แบบจะใช้ Address Bus เป็นตัวกำหนดตำแหน่งของจุดเชื่อมต่อ
และใช้ Data Bus สำหรับรับ/ส่งข้อมูล
แต่จุดที่แตกต่างกันคือที่มาของสัญญาณควบคุมการ อ่าน/เขียน ข้อมูล

Standard I/O จะมีการกำหนด Address แบบเดียวกับ Memory-mapped I/O

ใช้ Address bus และ Data bus เส้นเดียวกัน

แต่จะมี Address ไม่ทับกับซ้อนหน่วยความจำ เพราะใช้สัญญาณอ่านเขียน คนละเส้น กับหน่วยความจำ

การอ่าน / เขียน จะใช้คำสั่ง IN และ OUT



การอ่านข้อมูลจาก หน่วยความจำ M/IO จะสร้างสัญญาณ MREQ

การอ่านข้อมูลจาก Standard I/O M/IO จะสร้างสัญญาณ IOREQ

การต่อใช้งาน Standard I/O จะไม่ต้องตัดสัญญาณ CS ที่หน่วยความจำ

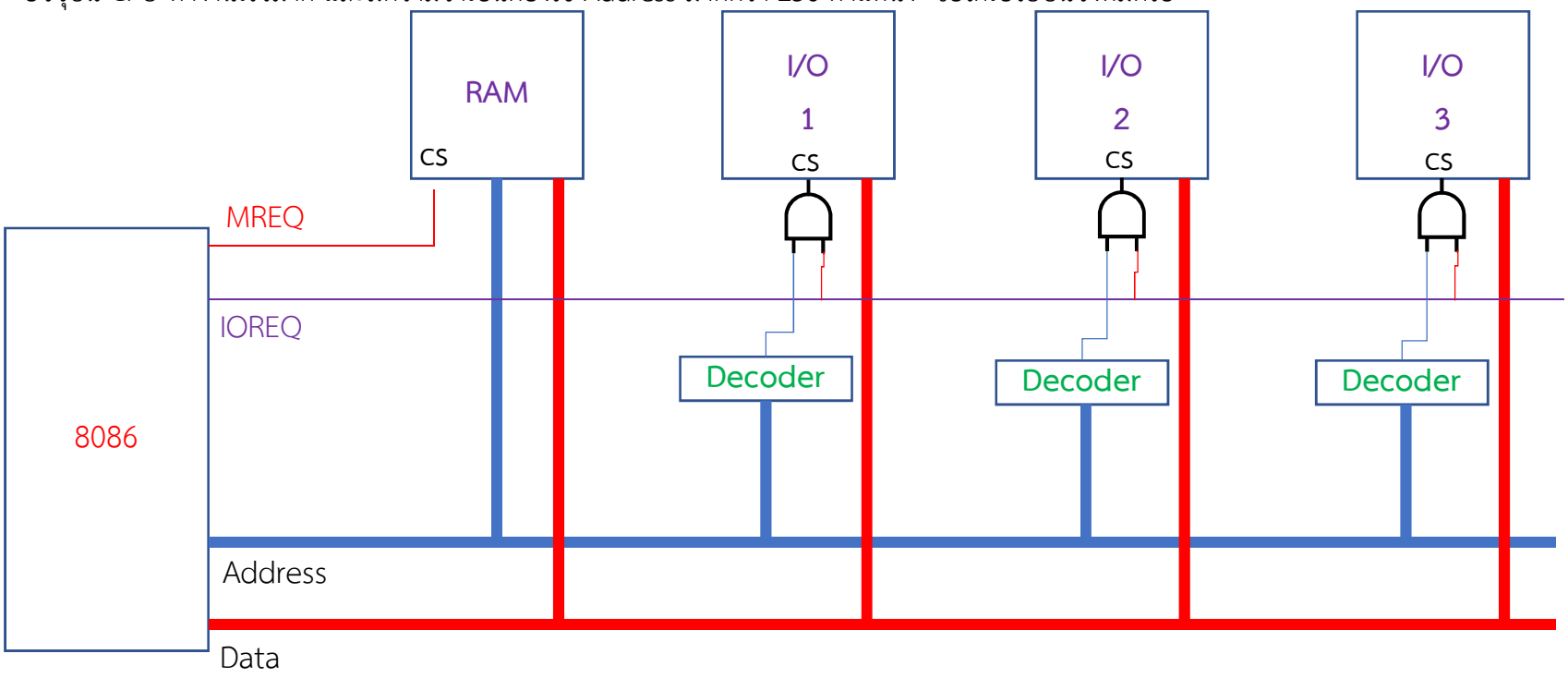
ข้อดี

- ตำแหน่งไม่ซ้อนทับกับหน่วยความจำ ทำให้สามารถใช้งานหน่วยความจำได้เต็มความจุ

ข้อเสีย

- ต้องใช้สายสัญญาณเพิ่มจากไมโครโพรเซสเซอร์ และต้องมี Instruction สำหรับ I/O

เดิม Standard I/O ได้เปรียบในด้านความเร็ว เพราะมี Instruction พิเศษสำหรับ Address ขนาด 8 บิต ที่สามารถทำงานได้ใน 1 Cycle เพราะเดิม CPU ทำงานช้า ปัจจุบัน CPU ทำงานเร็วมาก และมีความจำเป็นต้องใช้ Address มากกว่า 256 ตำแหน่ง ข้อได้เปรียบนี้จึงหมดไป



การต่อใช้งาน Standard I/O จะไม่ต้องตัดสัญญาณ CS ที่หน่วยความจำ

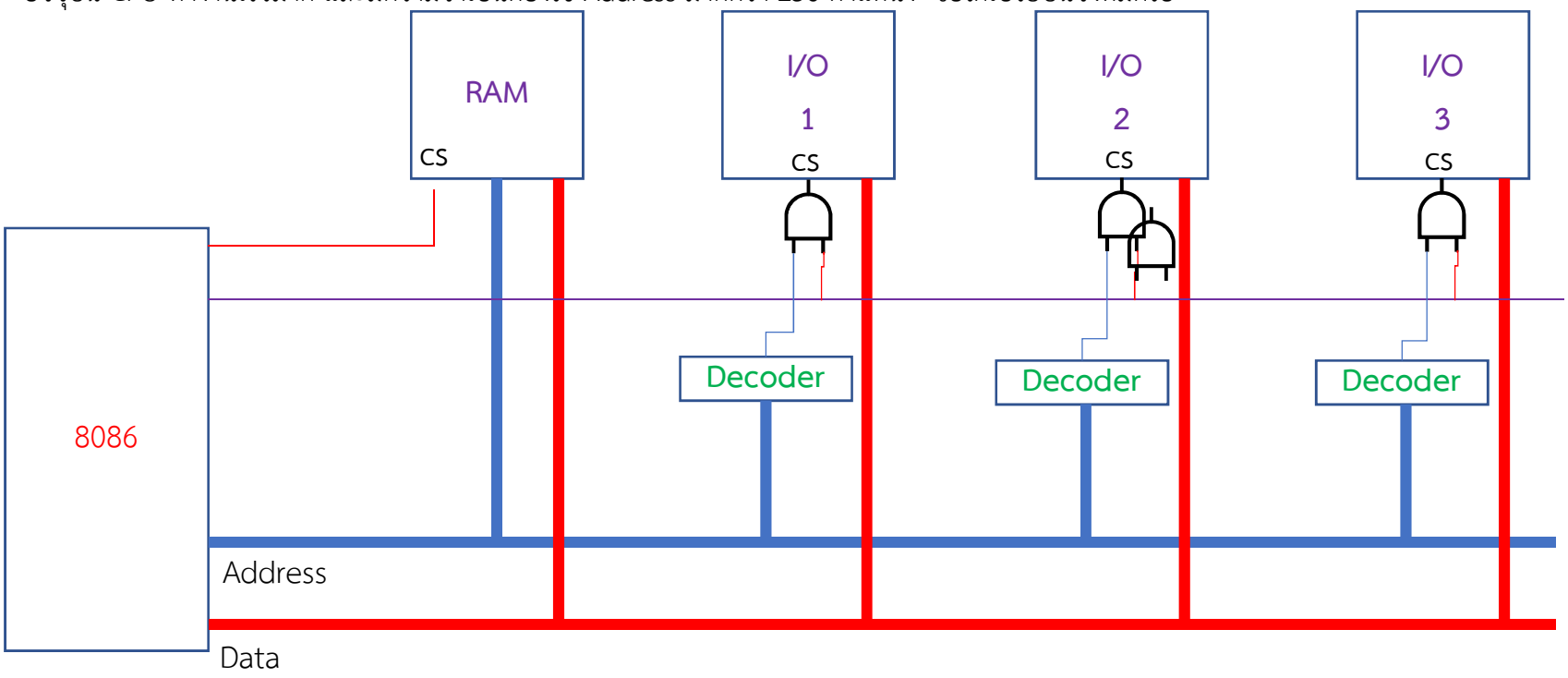
ข้อดี

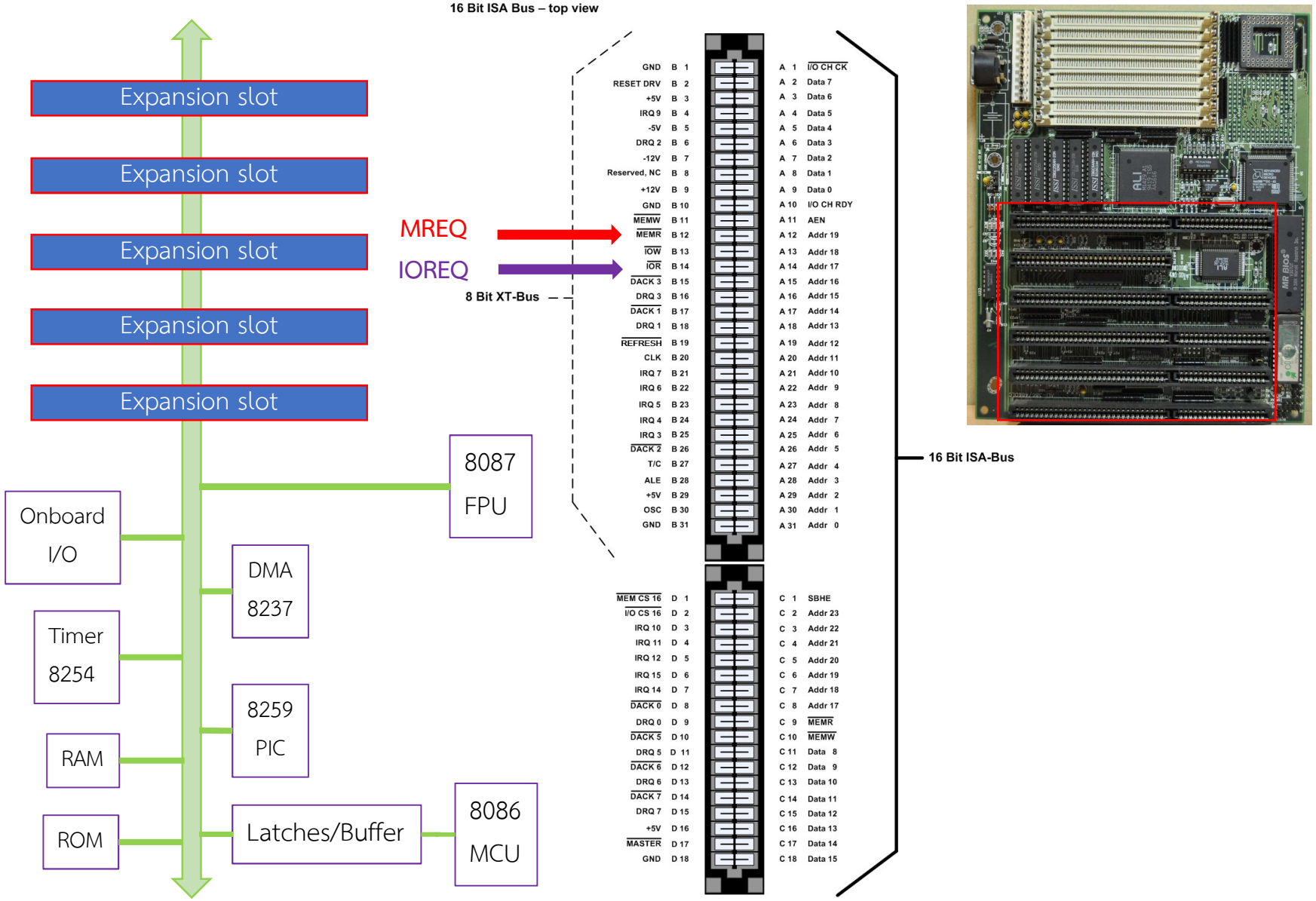
- ตำแหน่งไม่ซ้อนทับกับหน่วยความจำ ทำให้สามารถใช้งานหน่วยความจำได้เต็มความจุ

ข้อเสีย

- ต้องใช้สายสัญญาณเพิ่มจากไมโครโพรเซสเซอร์ และต้องมี Instruction สำหรับ I/O

เดิม Standard I/O ได้เปรียบในด้านความเร็ว เพราะมี Instruction พิเศษสำหรับ Address ขนาด 8 บิต ที่สามารถทำงานได้ใน 1 Cycle เพราะเดิม CPU ทำงานช้า ปัจจุบัน CPU ทำงานเร็วมาก และมีความจำเป็นต้องใช้ Address มากกว่า 256 ตำแหน่ง ข้อได้เปรียบนี้จึงหมดไป

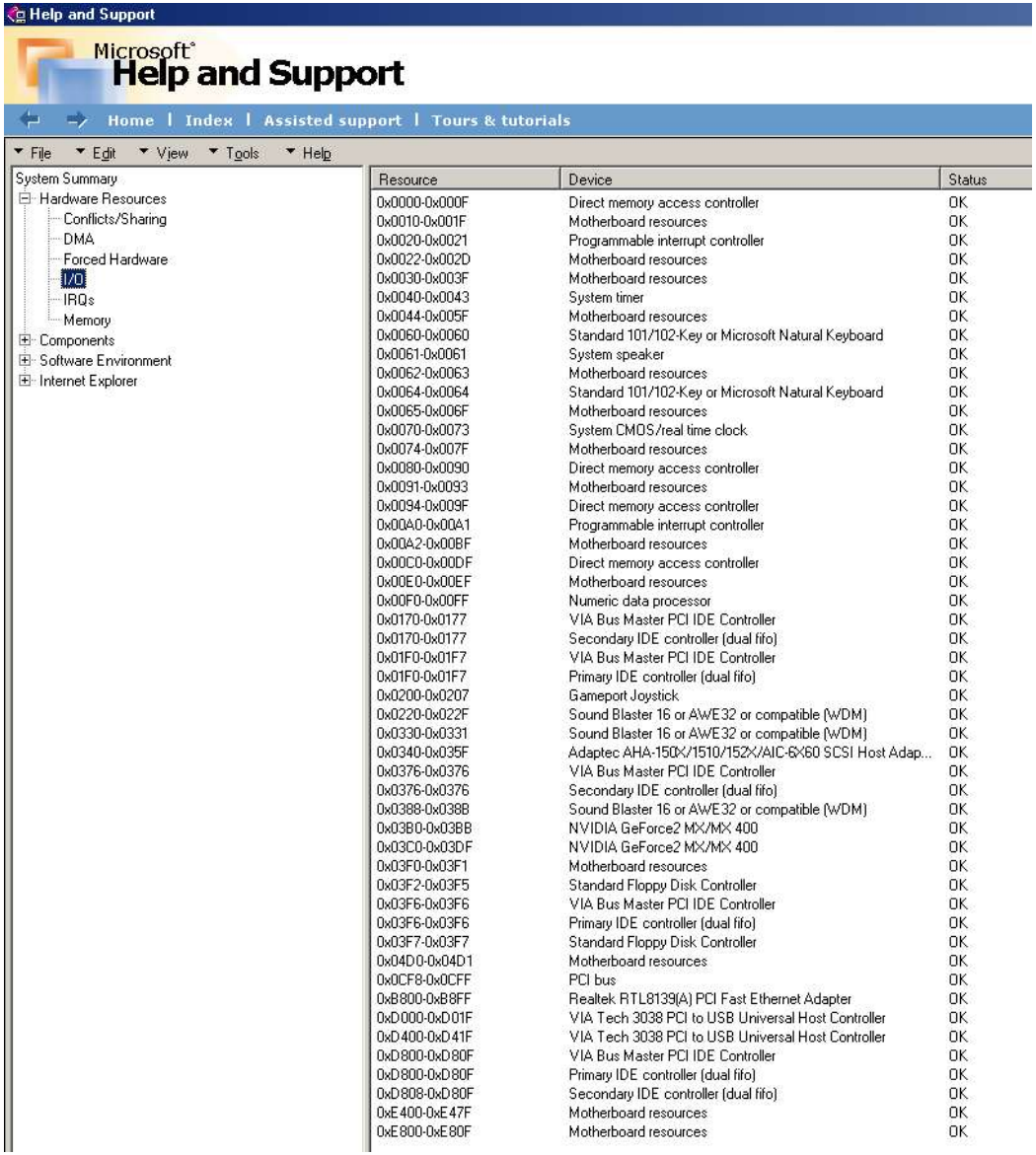



16 Bit ISA Bus – top view
Expansion slot
Expansion slot
Expansion slot
Expansion slot
Expansion slot
MREQ
IOREQ
8 Bit XT-Bus
16 Bit ISA-Bus
8087 FPU
Onboard I/O
DMA 8237
Timer 8254
8259 PIC
RAM
ROM
Latches/Buffer
8086 MCU
GND B 1
RESET DRV B 2
+5V B 3
IRQ 9 B 4
-5V B 5
DRQ 2 B 6
-12V B 7
Reserved, NC B 8
+12V B 9
GND B 10
MEMW B 11
MEMR B 12
IOW B 13
IOR B 14
DACK 3 B 15
DRQ 3 B 16
DACK 1 B 17
DRQ 1 B 18
REFRESH B 19
CLK B 20
IRQ 7 B 21
IRQ 6 B 22
IRQ 5 B 23
IRQ 4 B 24
IRQ 3 B 25
DACK 2 B 26
T/C B 27
ALE B 28
+5V B 29
OSC B 30
GND B 31
A 1 I/O CH CK
A 2 Data 7
A 3 Data 6
A 4 Data 5
A 5 Data 4
A 6 Data 3
A 7 Data 2
A 8 Data 1
A 9 Data 0
A 10 I/O CH RDY
A 11 AEN
A 12 Addr 19
A 13 Addr 18
A 14 Addr 17
A 15 Addr 16
A 16 Addr 15
A 17 Addr 14
A 18 Addr 13
A 19 Addr 12
A 20 Addr 11
A 21 Addr 10
A 22 Addr 9
A 23 Addr 8
A 24 Addr 7
A 25 Addr 6
A 26 Addr 5
A 27 Addr 4
A 28 Addr 3
A 29 Addr 2
A 30 Addr 1
A 31 Addr 0
MEM CS 16 D 1
I/O CS 16 D 2
IRQ 10 D 3
IRQ 11 D 4
IRQ 12 D 5
IRQ 15 D 6
IRQ 14 D 7
DACK 0 D 8
DRQ 0 D 9
DACK 5 D 10
DRQ 5 D 11
DACK 6 D 12
DRQ 6 D 13
DACK 7 D 14
DRQ 7 D 15
+5V D 16
MASTER D 17
GND D 18
C 1 SBHE
C 2 Addr 23
C 3 Addr 22
C 4 Addr 21
C 5 Addr 20
C 6 Addr 19
C 7 Addr 18
C 8 Addr 17
C 9 MEMR
C 10 MEMW
C 11 Data 8
C 12 Data 9
C 13 Data 10
C 14 Data 11
C 15 Data 12
C 16 Data 13
C 17 Data 14
C 18 Data 15

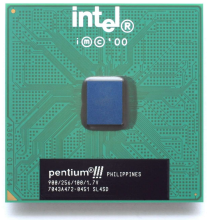
IBM PC Interrupts

การตั้งค่า Address decoder

ตัวอย่างการใช้งาน Standard I/O



Resource	Device	Status
0x0000-0x000F	Direct memory access controller	OK
0x0010-0x001F	Motherboard resources	OK
0x0020-0x0021	Programmable interrupt controller	OK
0x0022-0x002D	Motherboard resources	OK
0x0030-0x003F	Motherboard resources	OK
0x0040-0x0043	System timer	OK
0x0044-0x005F	Motherboard resources	OK
0x0060-0x0060	Standard 101/102-Key or Microsoft Natural Keyboard	OK
0x0061-0x0061	System speaker	OK
0x0062-0x0063	Motherboard resources	OK
0x0064-0x0064	Standard 101/102-Key or Microsoft Natural Keyboard	OK
0x0065-0x006F	Motherboard resources	OK
0x0070-0x0073	System CMOS/real time clock	OK
0x0074-0x007F	Motherboard resources	OK
0x0080-0x0090	Direct memory access controller	OK
0x0091-0x0093	Motherboard resources	OK
0x0094-0x009F	Direct memory access controller	OK
0x00A0-0x00A1	Programmable interrupt controller	OK
0x00A2-0x00BF	Motherboard resources	OK
0x00C0-0x00DF	Direct memory access controller	OK
0x00E0-0x00EF	Motherboard resources	OK
0x00F0-0x00FF	Numeric data processor	OK
0x0170-0x0177	VIA Bus Master PCI IDE Controller	OK
0x0170-0x0177	Secondary IDE controller (dual fifo)	OK
0x01F0-0x01F7	VIA Bus Master PCI IDE Controller	OK
0x01F0-0x01F7	Primary IDE controller (dual fifo)	OK
0x0200-0x0207	Gameport Joystick	OK
0x0220-0x022F	Sound Blaster 16 or AWE32 or compatible (WDM)	OK
0x0330-0x0331	Sound Blaster 16 or AWE32 or compatible (WDM)	OK
0x0340-0x035F	Adaptec AHA-150X/1510/152X/AIC-6X60 SCSI Host Adap...	OK
0x0376-0x0376	VIA Bus Master PCI IDE Controller	OK
0x0376-0x0376	Secondary IDE controller (dual fifo)	OK
0x0388-0x038B	Sound Blaster 16 or AWE32 or compatible (WDM)	OK
0x03B0-0x03BB	NVIDIA GeForce2 MX/MX 400	OK
0x03C0-0x03DF	NVIDIA GeForce2 MX/MX 400	OK
0x03F0-0x03F1	Motherboard resources	OK
0x03F2-0x03F5	Standard Floppy Disk Controller	OK
0x03F6-0x03F6	VIA Bus Master PCI IDE Controller	OK
0x03F6-0x03F6	Primary IDE controller (dual fifo)	OK
0x03F7-0x03F7	Standard Floppy Disk Controller	OK
0x04D0-0x04D1	Motherboard resources	OK
0x0CF8-0x0CFF	PCI bus	OK
0xB800-0xB8FF	Realtek RTL8139(A) PCI Fast Ethernet Adapter	OK
0xD000-0xD01F	VIA Tech 3038 PCI to USB Universal Host Controller	OK
0xD400-0xD41F	VIA Tech 3038 PCI to USB Universal Host Controller	OK
0xD800-0xD80F	VIA Bus Master PCI IDE Controller	OK
0xD800-0xD80F	Primary IDE controller (dual fifo)	OK
0xD808-0xD80F	Secondary IDE controller (dual fifo)	OK
0xE400-0xE47F	Motherboard resources	OK
0xE800-0xE80F	Motherboard resources	OK



จากนี้ไป จะเรียก

Standard I/O ว่า Port

Address ของ Standard I/O ว่า Port Address หรือ หมายเลข port

การเขียนข้อมูลออกไปยัง Port จะใช้คำสั่ง OUT operand1 ,operand2
คำสั่ง OUT มีการทำงาน 2 รูปแบบคือแบบ Direct mode และ Indirect mode

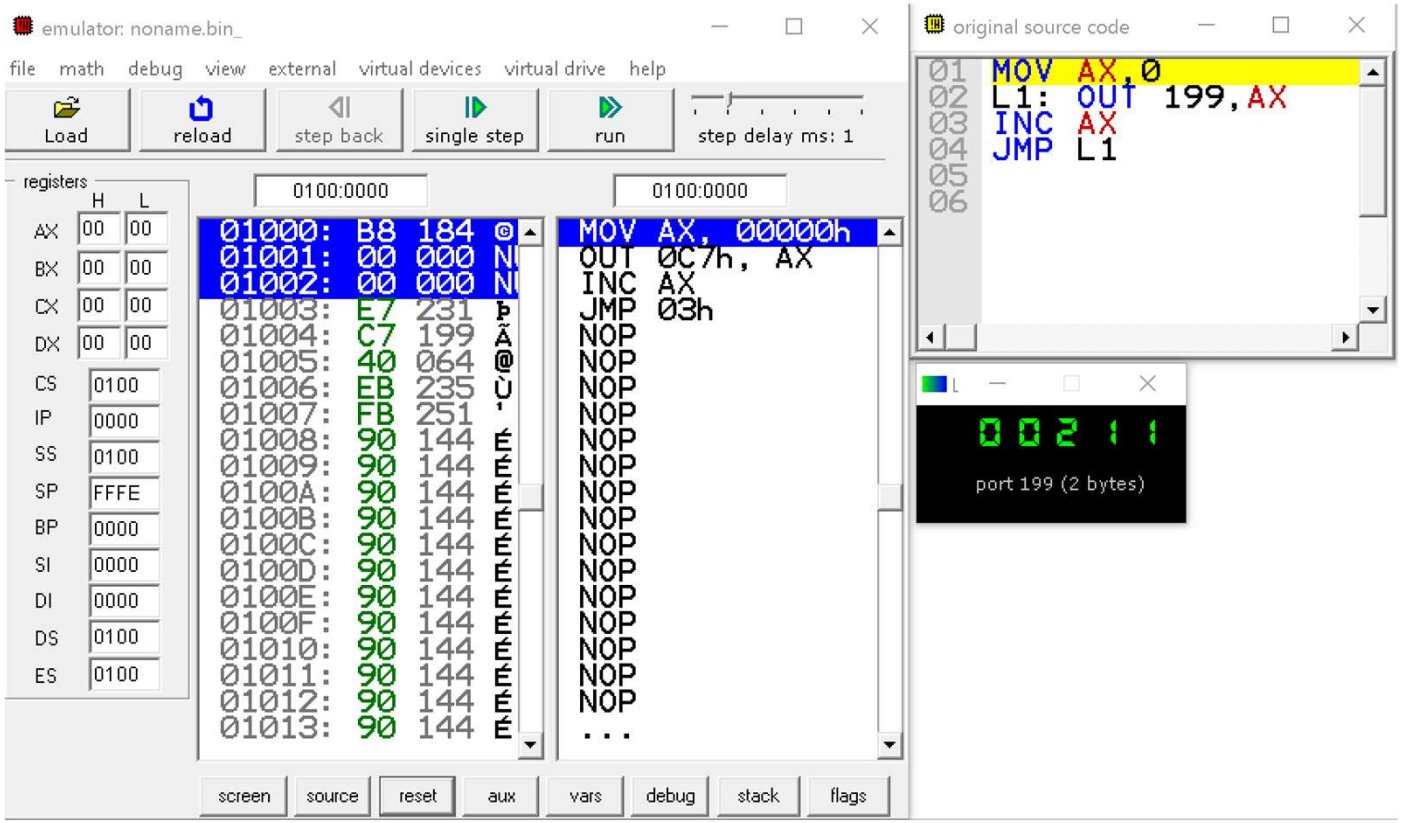
Direct mode:
Operand1 เป็น Immediate ค่า 0 – 255 หมายถึง Port Address
Operand2 เป็น รีจิสเตอร์ AX หรือ AL เท่านั้น

Indirect mode
Operand1 เป็นรีจิสเตอร์ DX เท่านั้น หมายถึง Port Address
Operand2 เป็น รีจิสเตอร์ AX หรือ AL เท่านั้น

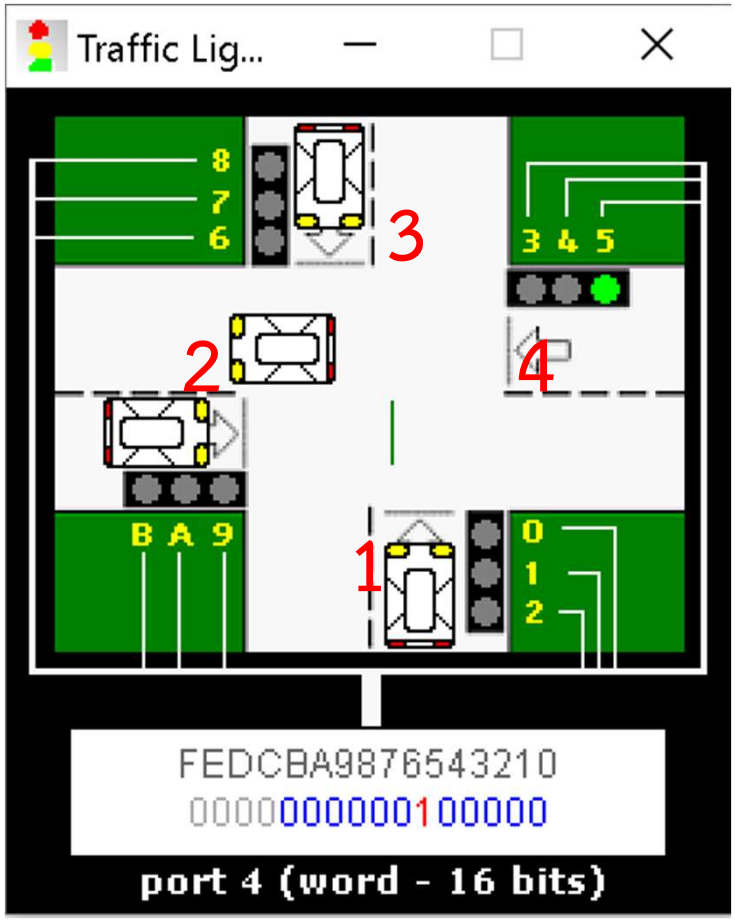
Direct mode หมายเลข port ระบุได้แค่ 0 – 255 เท่านั้น และแก้ไขขณะรันไม่ได้ แต่ใช้เวลาทำงาน 1 Cycle
Indirect mode หมายเลข port ระบุได้ 0 -65535 และแก้ไขขณะรันได้ แต่ใช้เวลาทำงาน 2 Cycle

LED Display ของโปรแกรมจำลอง ต่ออยู่ที่ Port หมายเลข 199

```
MOV AX,0
L1: OUT 199,AX
    INC AX
    JMP L1
```



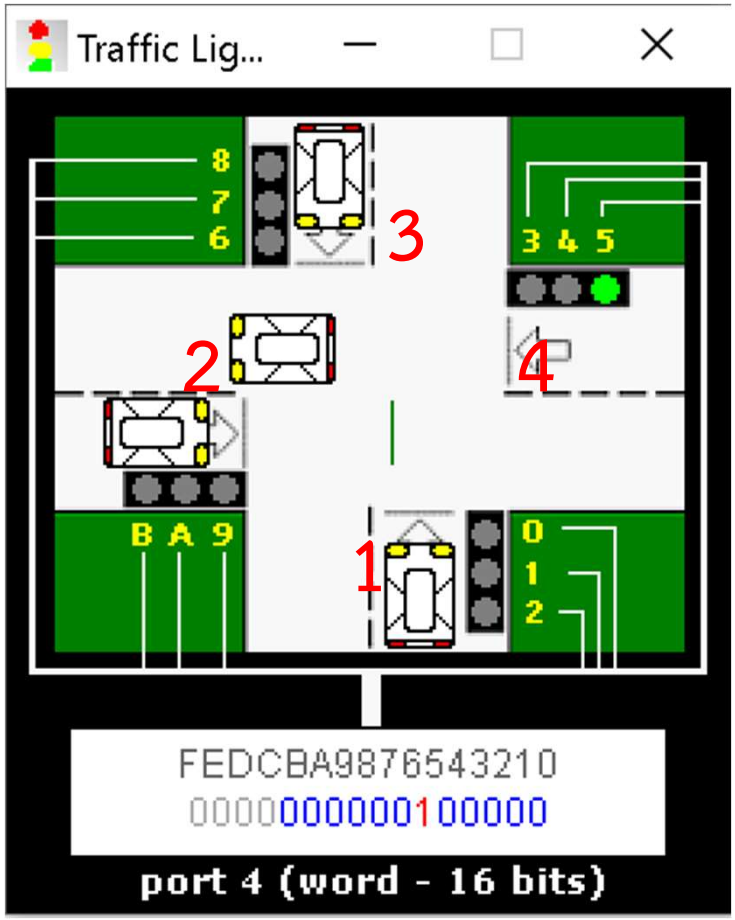
LED Display ของโปรแกรมจำลอง ต่ออยู่ที่ Port หมายเลข 4



แต่ละบิต ควบคุมแต่ละหลอดตามภาพ

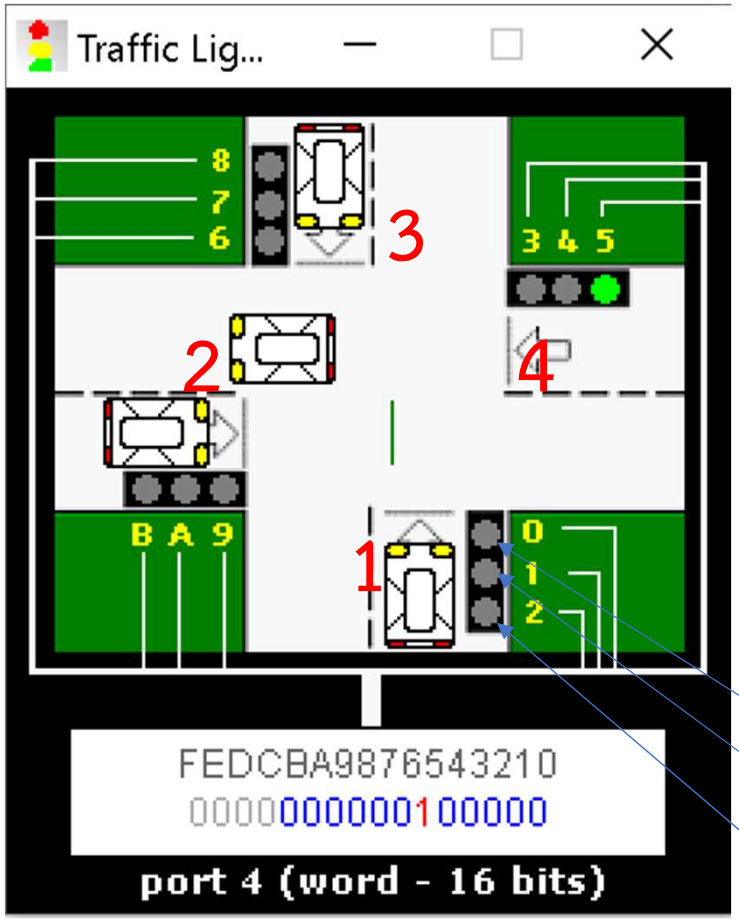
เขียนโปรแกรรมควบคุมไฟจาราจร โดย
ให้ปล่อยรถตามลำดับแยก
1 -> 2 -> 3 -> 4
ตามลำดับ

LED Display ของโปรแกรมจำลอง ต่ออยู่ที่ Port หมายเลข 4



ลำดับ	แยก 1	แยก 2	แยก 3	แยก 4
1	เขียว	แดง	แดง	แดง
2	ส้ม	แดง	แดง	แดง
3	แดง	เขียว	แดง	แดง
4	แดง	ส้ม	แดง	แดง
5	แดง	แดง	เขียว	แดว
6	แดง	แดง	ส้ม	แดง
7	แดง	แดง	แดง	เขียว
8	แดง	แดง	แดง	ส้ม

LED Display ของโปรแกรมจำลอง ต่ออยู่ที่ Port หมายเลข 4



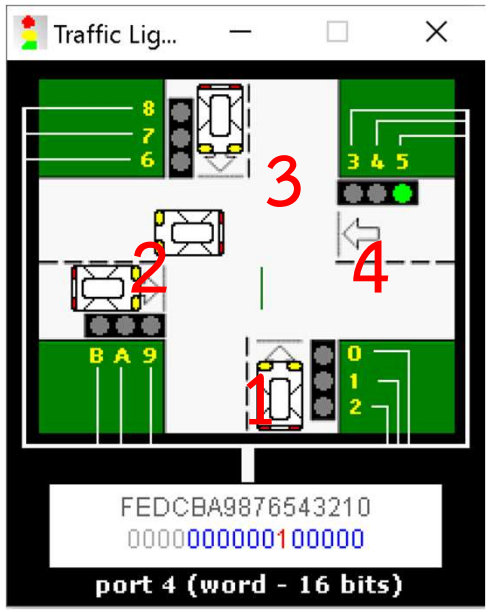
ลำดับ	แยก 1	แยก 2	แยก 3	แยก 4
1	เขียว	แดง	แดง	แดง
2	ส้ม	แดง	แดง	แดง
3	แดง	เขียว	แดง	แดง
4	แดง	ส้ม	แดง	แดง
5	แดง	แดง	เขียว	แดว
6	แดง	แดง	ส้ม	แดง
7	แดง	แดง	แดง	เขียว
8	แดง	แดง	แดง	ส้ม

แดง

เหลือง

เขียว

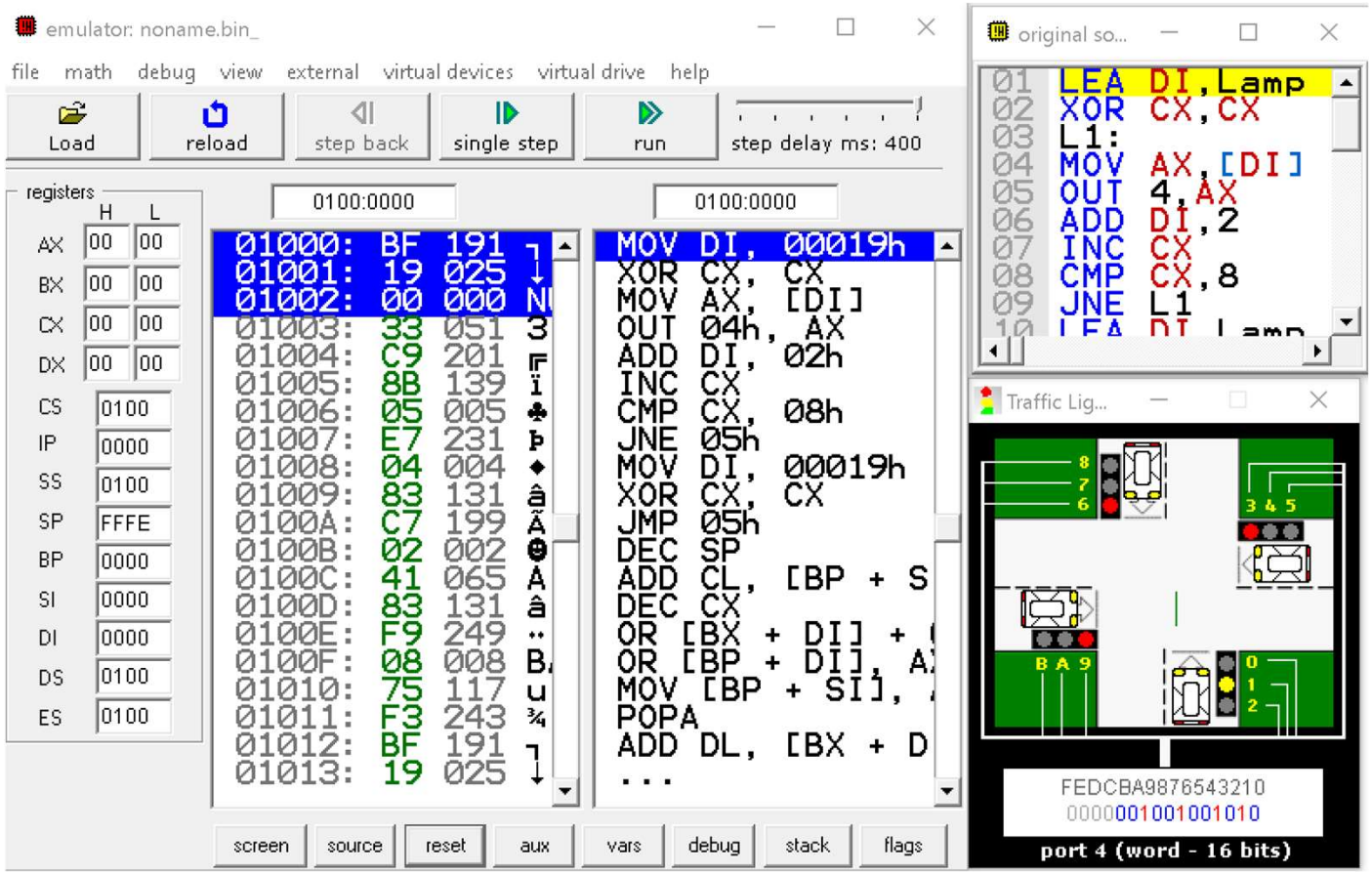
LED Display ของโปรแกรมจำลอง ต่ออยู่ที่ Port หมายเลข 4



	แยก				หลอดไฟ (Output)											
ลำดับ	1	2	3	4	B	A	9	8	7	6	5	4	3	2	1	0
1	เขียว	แดง	แดง	แดง	0	0	1	0	0	1	0	0	1	1	0	0
2	ส้ม	แดง	แดง	แดง	0	0	1	0	0	1	0	0	1	0	1	0
3	แดง	เขียว	แดง	แดง	1	0	0	0	0	1	0	0	1	0	0	1
4	แดง	ส้ม	แดง	แดง	0	1	0	0	0	1	0	0	1	0	0	1
5	แดง	แดง	เขียว	แดง	0	0	1	1	0	0	0	0	1	0	0	1
6	แดง	แดง	ส้ม	แดง	0	0	1	0	1	0	0	0	1	0	0	1
7	แดง	แดง	แดง	เขียว	0	0	1	0	0	1	1	0	0	0	0	1
8	แดง	แดง	แดง	ส้ม	0	0	1	0	0	1	0	1	0	0	0	1

LED Display ของโปรแกรมจำลอง ต่ออยู่ที่ Port หมายเลข 4

```
 LEA DI,Lamp
    XOR CX,CX
L1:
    MOV AX,[DI]
    OUT 4,AX
    ADD DI,2
    INC CX
    CMP CX,8
    JNE L1
    LEA DI,Lamp
    XOR CX,CX
    JMP L1
Lamp DW 0010010011008
        DW 0010010010108
        DW 1000010010018
        DW 0100010010018
        DW 0011000010018
        DW 0010100010018
        DW 0010011000018
        DW 0010010100018
```

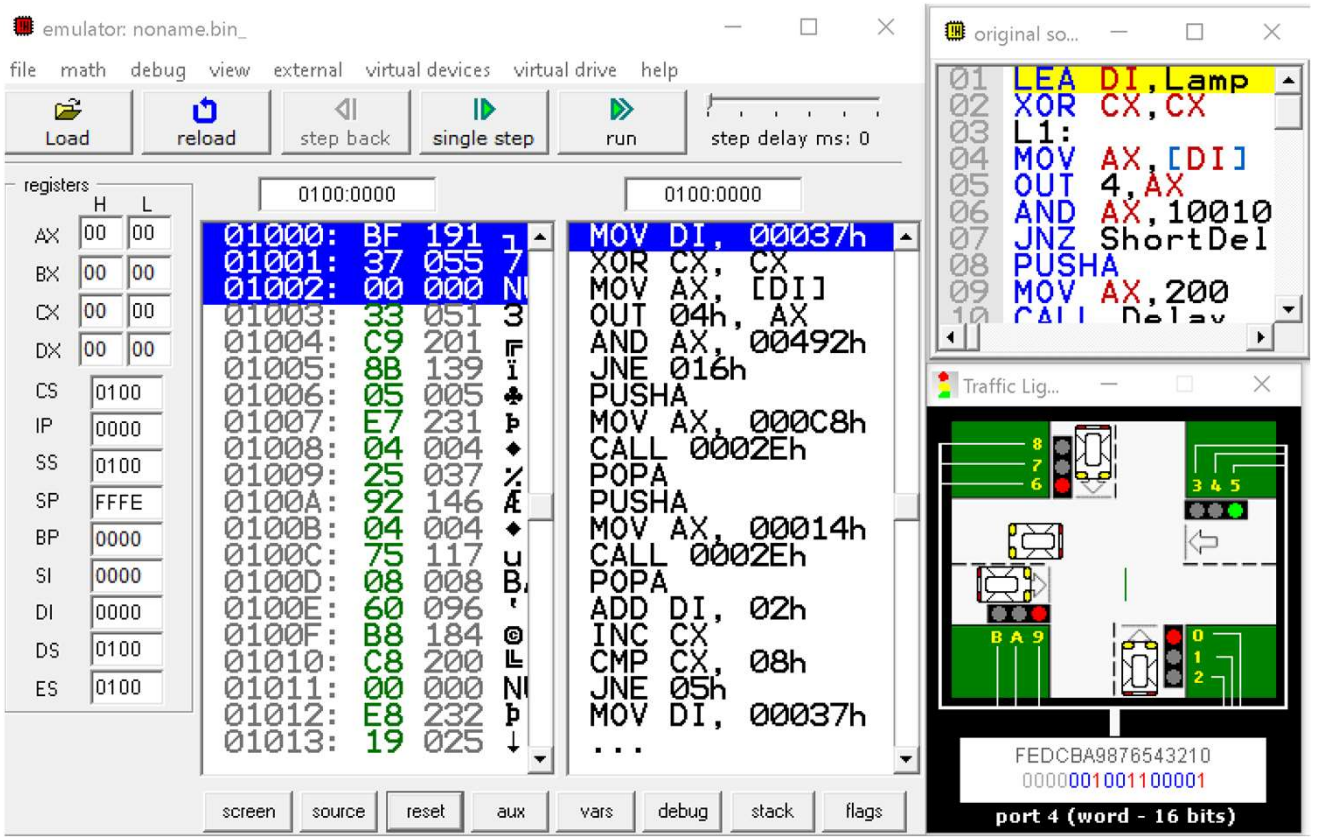


LED Display ของโปรแกรมจำลอง ต่ออยู่ที่ Port หมายเลข 4

```
 LEA DI,Lamp
    XOR CX,CX
L1:
    MOV AX,[DI]
    OUT 4,AX
    AND AX,100100100010B
    JNZ ShortDelay
    PUSHA
    MOV AX,200
    CALL Delay
    POPA
ShortDelay:
    PUSHA
    MOV AX,20
    CALL Delay
    POPA
    ADD DI,2
    INC CX
    CMP CX,8
    JNE L1
    LEA DI,Lamp
    XOR CX,CX
    JMP L1
```

```
Delay PROC
   MOV CX,0
   Start: INC CX
          CMP CX,AX
          JNG Start
          RET
Delay ENDP

Lamp DW 001001001100B
     DW 001001001010B
     DW 100001001001B
     DW 010001001001B
     DW 001100001001B
     DW 001010001001B
     DW 001001100001B
     DW 001001010001B
```



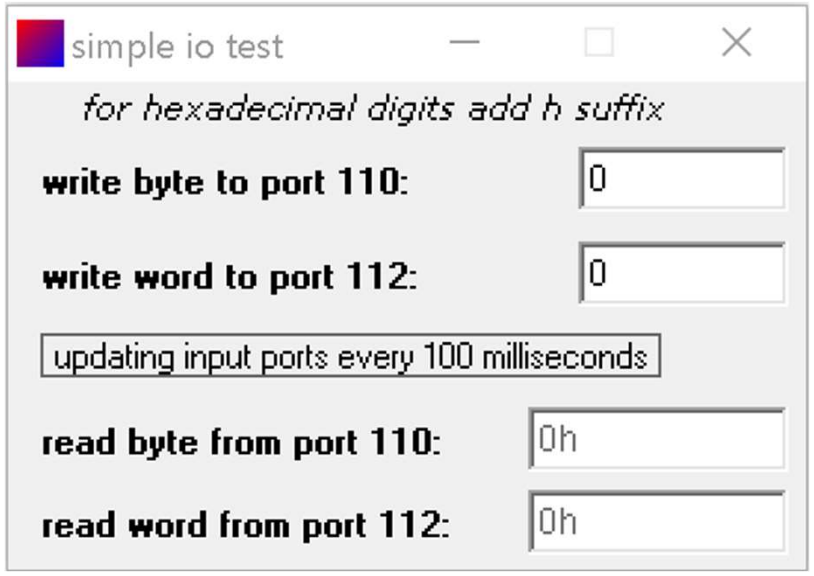
การอ่านข้อมูลจาก Port จะใช้คำสั่ง IN operand1 ,operand2
คำสั่ง IN มีการทำงาน 2 รูปแบบคือแบบ Direct mode และ Indirect mode

Direct mode:
Operand1 เป็น รีจิสเตอร์ AX หรือ AL เท่านั้น สำหรับรับข้อมูล
Operand2 เป็น Immediate ค่า 0 – 255 หมายถึง Port Address

Indirect mode:
Operand1 เป็น รีจิสเตอร์ AX หรือ AL เท่านั้น สำหรับรับข้อมูล
Operand2 เป็นรีจิสเตอร์ DX เท่านั้น หมายถึง Port Address

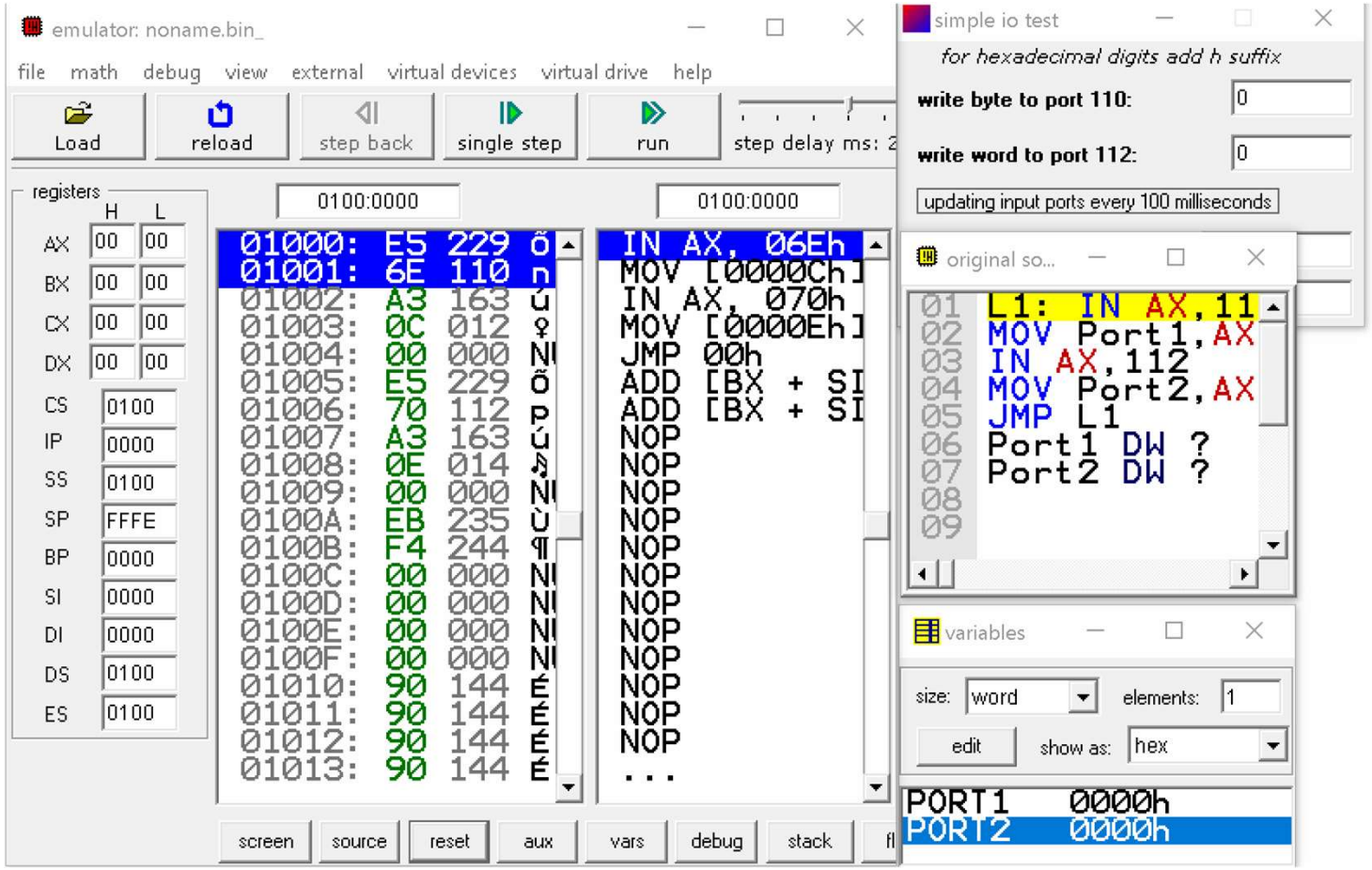
Direct mode หมายเลข port ระบุได้แค่ 0 – 255 เท่านั้น และแก้ไขขณะรันไม่ได้ แต่ใช้เวลาทำงาน 1 Cycle
Indirect mode หมายเลข port ระบุได้ 0 -65535 และแก้ไขขณะรันได้ แต่ใช้เวลาทำงาน 2 Cycle

Simple IO Test เชื่อมต่อกับ port 110 และ 112

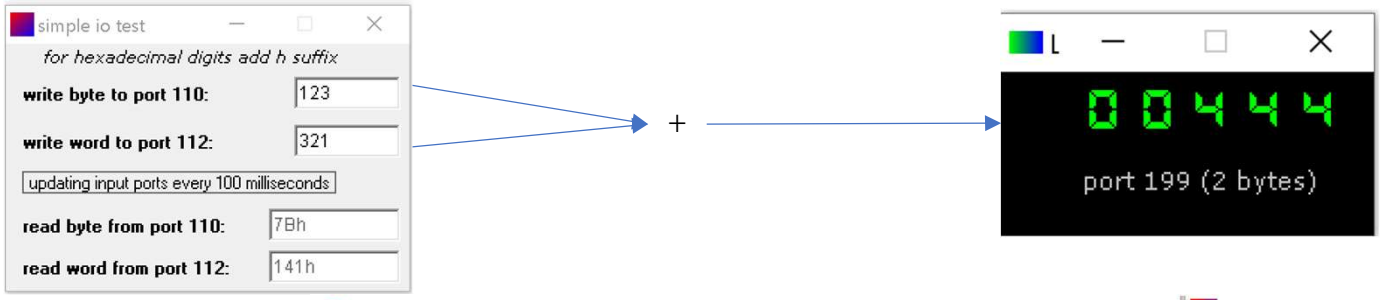


Simple IO Test เชื่อมต่อกับ port 110 และ 112

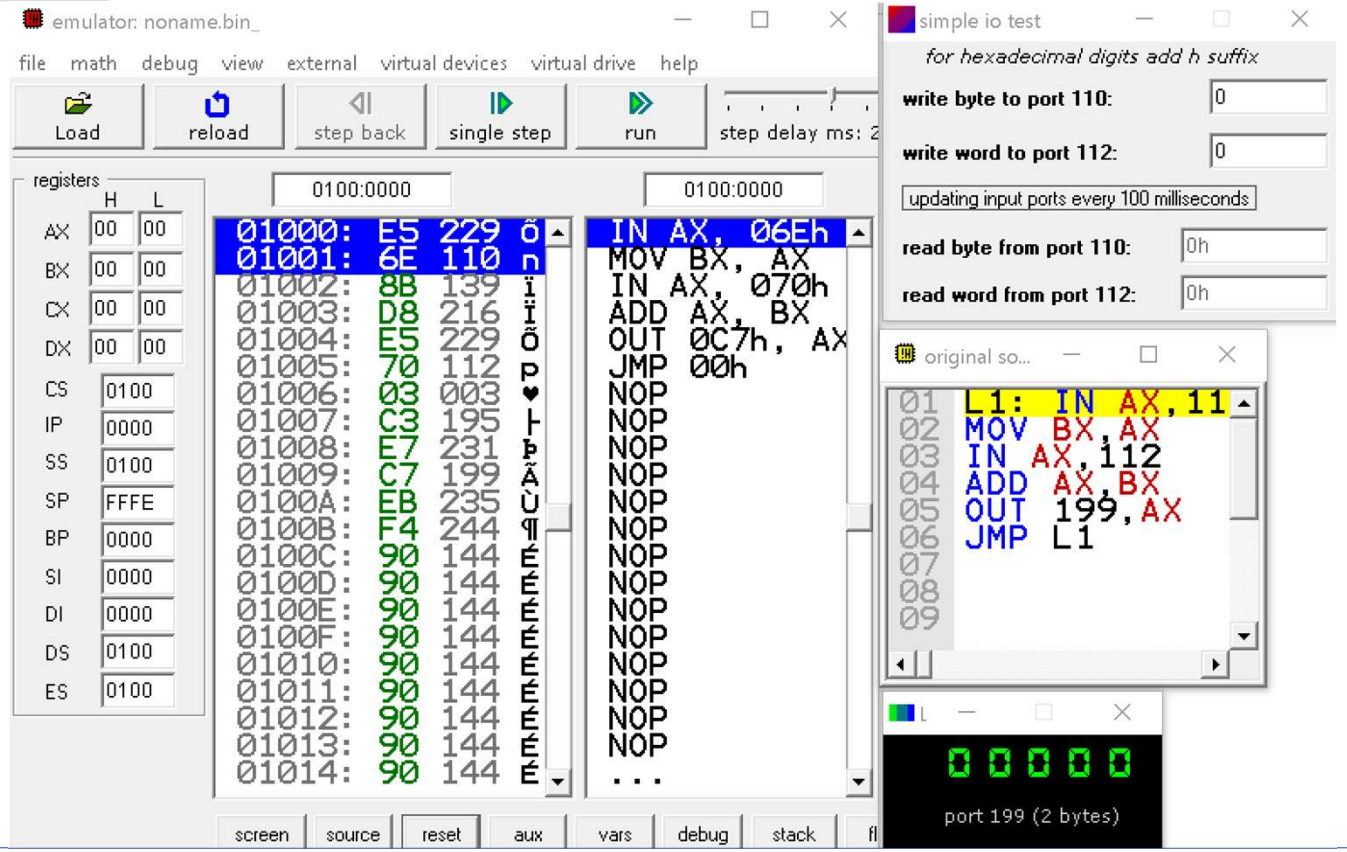
```
L1: IN AX,110
    MOV Port1,AX
    IN AX,112
    MOV Port2,AX
    JMP L1
Port1 DW ?
Port2 DW ?
```



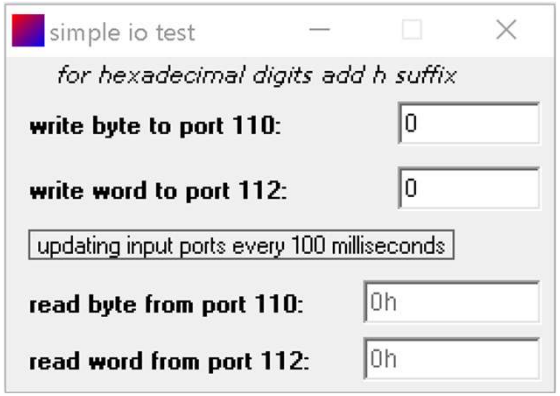
Simple IO Test เชื่อมต่อกับ port 110 และ 112

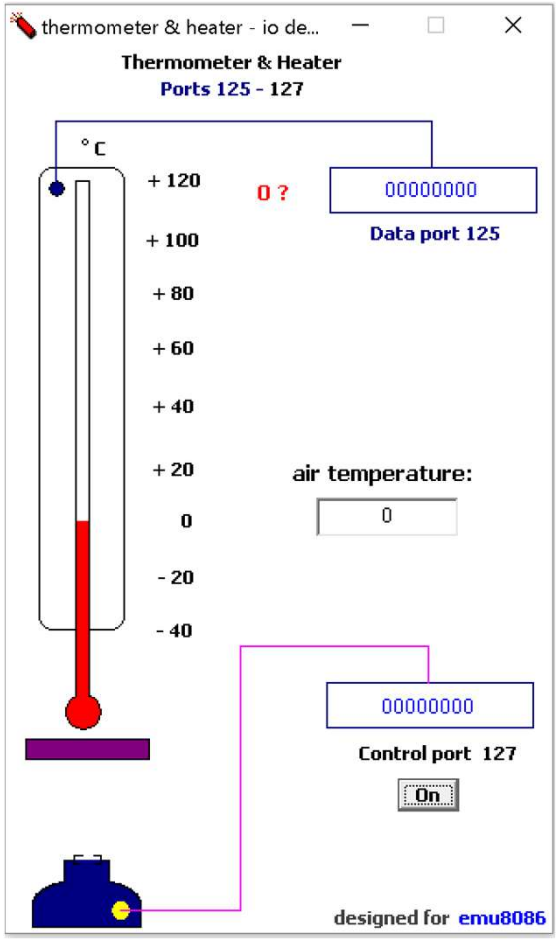



```
L1: IN AX,110
    MOV BX,AX
    IN AX,112
    ADD AX,BX
    OUT 199,AX
    JMP L1
```



Simple IO Test เชื่อมต่อกับ port 110 และ 112

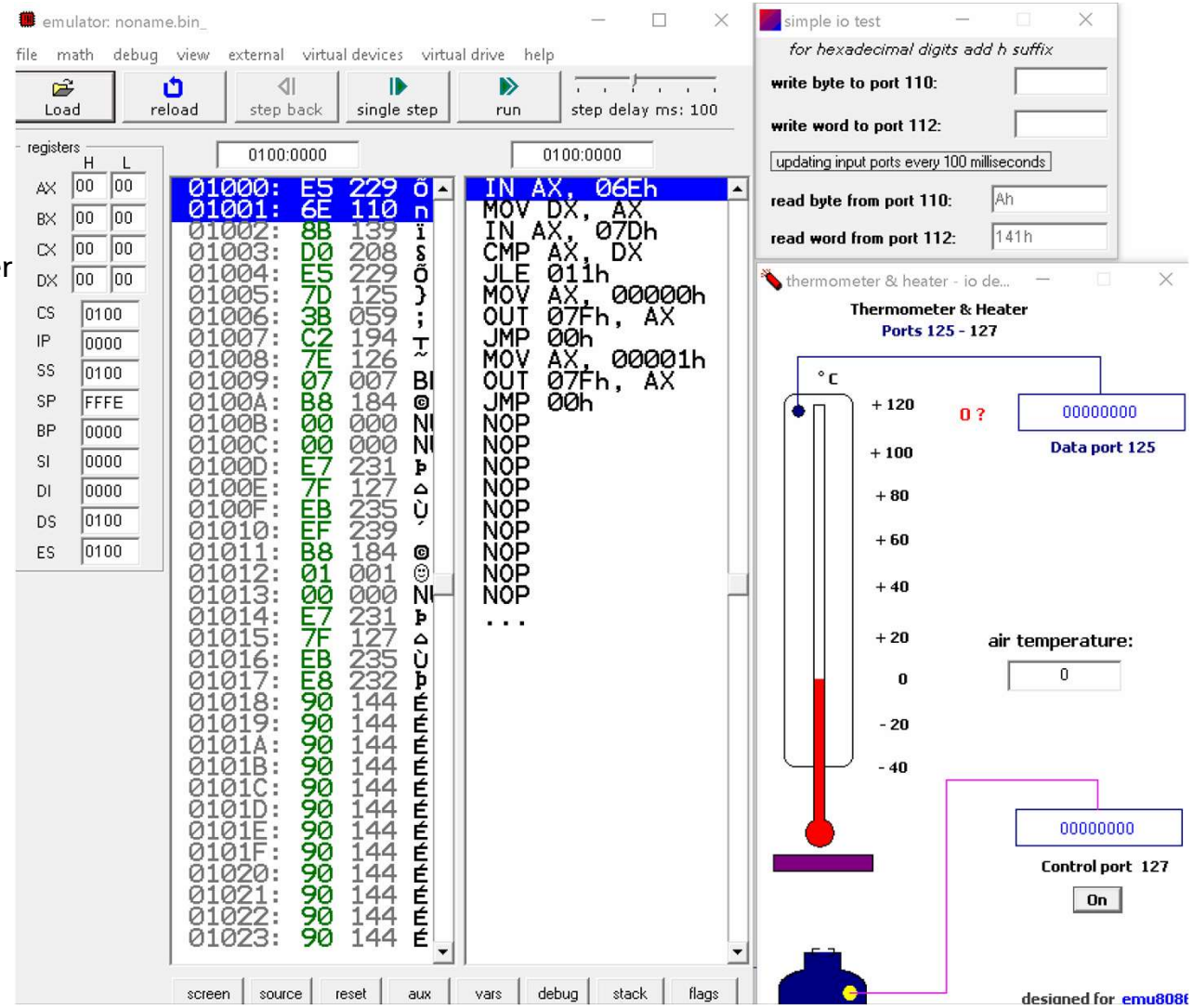




Thermometer ต่อกับ port 125 และวงจรควบคุมเตาเผาต่อกับ port 127 “0” ปิดเตา , “1” เปิดเตา

ให้เขียนโปรแกรมควบคุมอุณหภูมิ โดยกำหนดให้ป้อนค่าอุณหภูมิที่ port 110

```
Start:
    IN AX,110  ; Read Input
    MOV DX,AX
    IN AX,125  ; Read Thermometer
    CMP AX,DX
    JNG  TurnGasOn    ;AX < DX
    MOV AX,0    ; Gas Off
    OUT 127,AX  ;
    JMP Start
TurnGasOn:
    MOV AX,1    ; Gas On
    OUT 127,AX  ;
    JMP Start
```

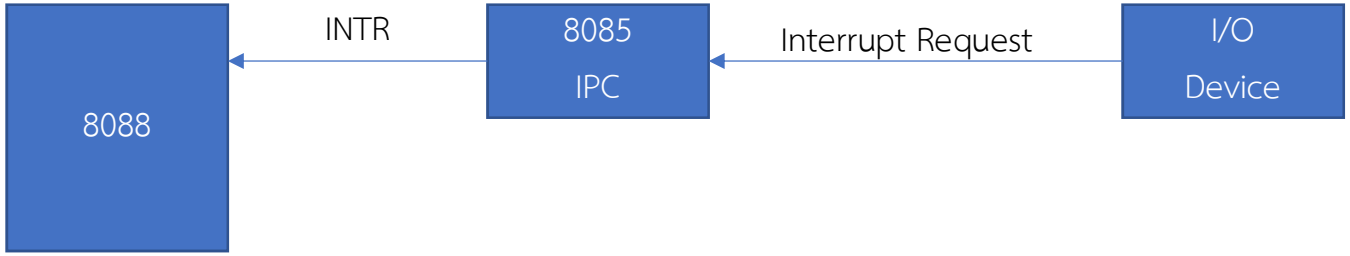


ไม่ว่าจะใช้วิธีไหนในการสร้าง I/O ขั้นตอนการสื่อสารก็จะไม่แตกต่างกัน

โดยทั่วไปจะมี 2 วิธี

- การขัดจังหวะ (Interrupt)
- การหยั่ง (Polling)

การขัดจังหวะ

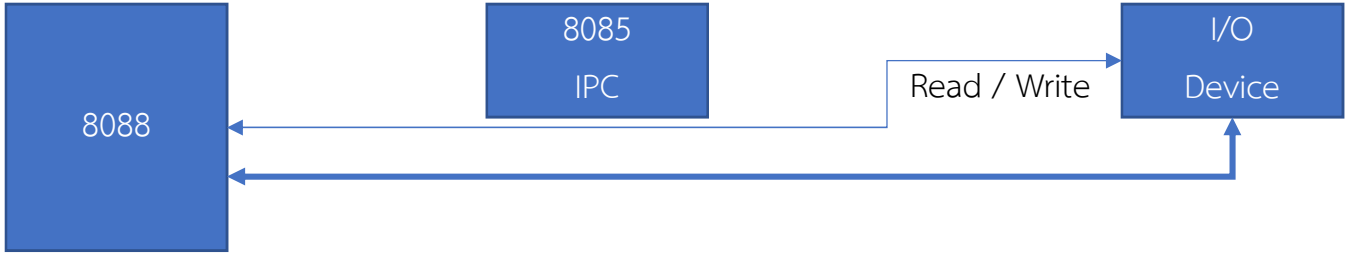


1 อุปกรณ์ ร้องขอใช้ CPU โดยสร้างสัญญาณ IRQ เข้ามาที่ชิพ IPC เมื่อถึงคิว IPC จะสร้างสัญญาณ INTR ไปยัง ไมโครโพรเซสเซอร์

1 เมื่อไมโครโพรเซสเซอร์พร้อมให้บริการ จะส่ง INTA กลับมาที่ IPC

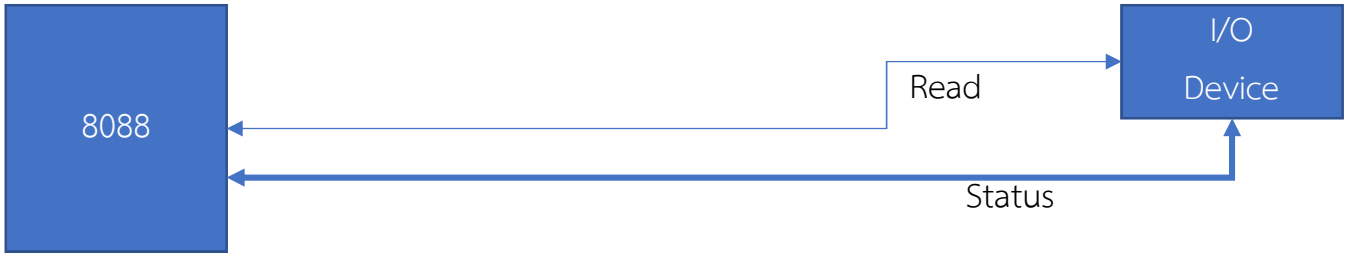
ไม่ว่าจะใช้วิธีไหนในการสร้าง I/O ขั้นตอนการสื่อสารก็จะไม่แตกต่างกัน

3 PIC ส่งชนิดของ Interrupt กลับไปให้ไมโครโพรเซสเซอร์ตามชนิดของ IRQ ที่โปรแกรมไว้

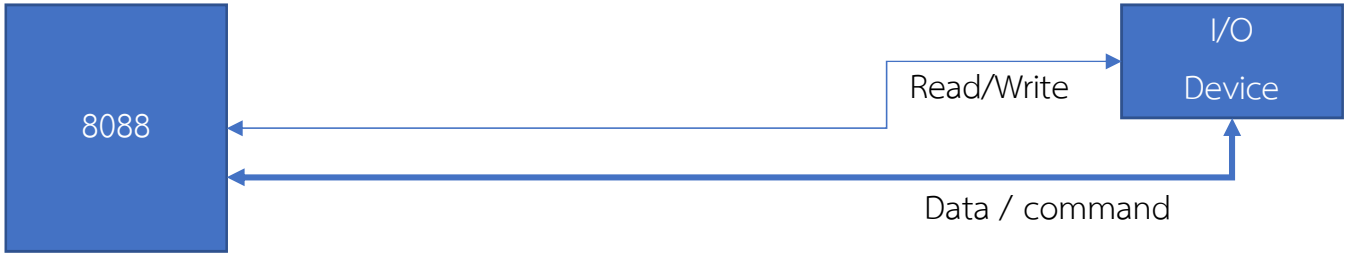


4 ไมโครโพสเซสเซอร์อ่านหรือเขียนข้อมูลไปยังอุปกรณ์ ตามวิธีการเชื่อมต่อ และ Address ที่ได้โปรแกรมไว้

การหยั่ง



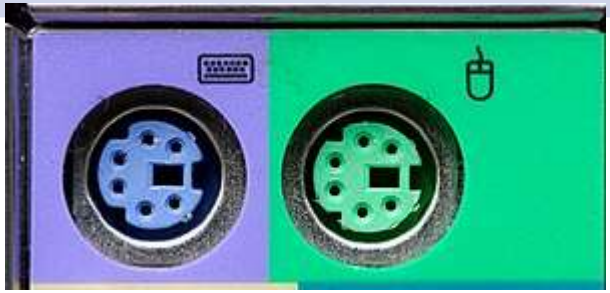
1 ไมโครโพสเซสเซอร์อ่านสถานะจากอุปกรณ์ ว่ามีข้อมูลต้องการส่งหรือไม่

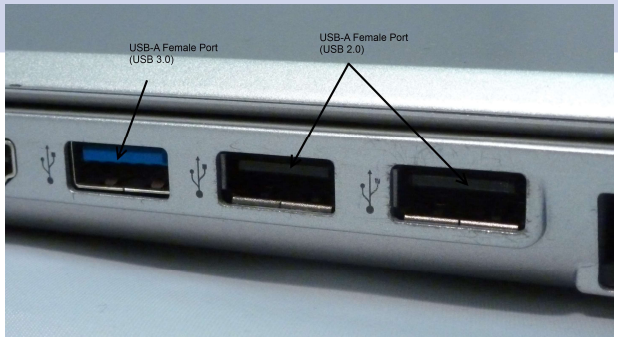


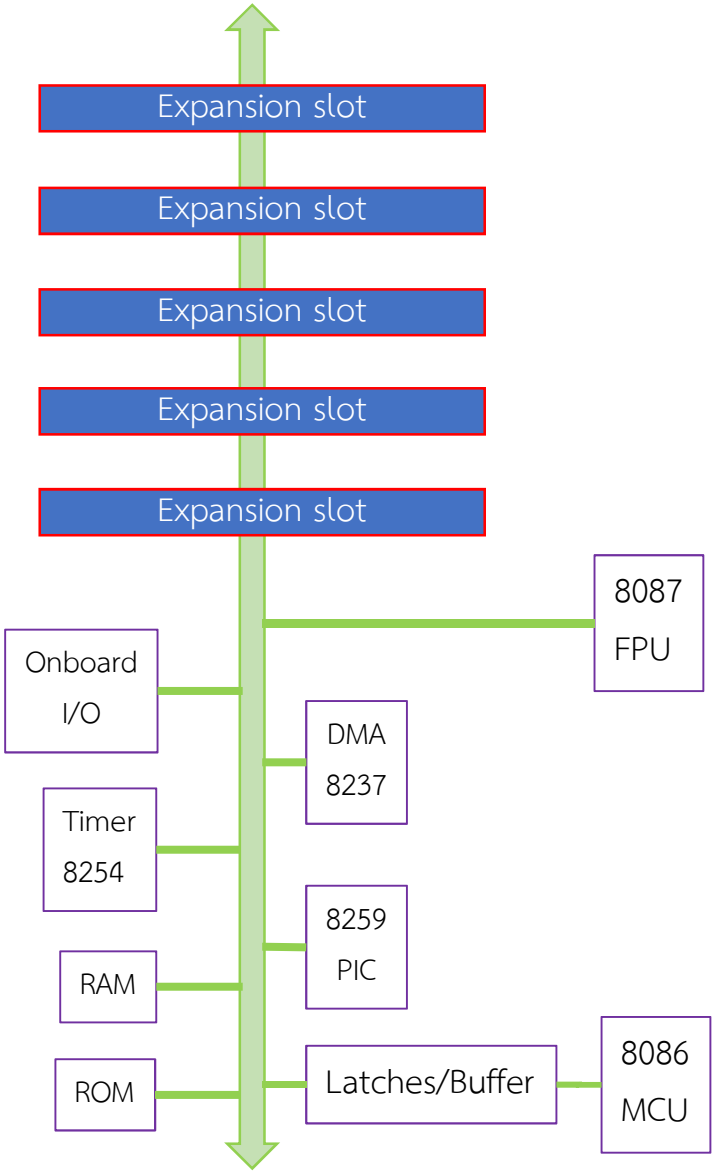
2 หากอุปรณ์ต้องการติดต่อกับไมโครโพสเซสเซอร์ จะส่งข้อมูลหรือคำสั่งกลับมา

รูปแบบการสื่อสาร

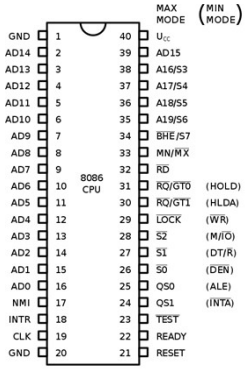
การขัดจังหวะ	การหยั่ง
- อุปกรณ์จะบอก CPU เมื่อต้องการใช้งาน CPU - สร้างด้วย hardware	- CPU ต้องคอยถาม อุปกรณ์ตลอดเวลา - สร้างด้วย software
ข้อดี - ส่งข้อมูลได้ตลอดเวลา - ตอบสนองเร็ว - ประมวลผลทำเท่าที่จำเป็น	ข้อดี - สร้างด้วย Software ไม่ต้องมี hardware ช่วย - มีความยืดหยุ่นสูง
ข้อเสีย - ต้องมีวงจรเพิ่ม สำหรับจัดการ IRQ - มีความยืดหยุ่นน้อย	ข้อเสีย - ต้องรอจังหวะส่งข้อมูล - ตอบสนองช้า - การประมวลผลจะสูญเปล่าไป หากอุปกรณ์ไม่มีการส่งข้อมูล

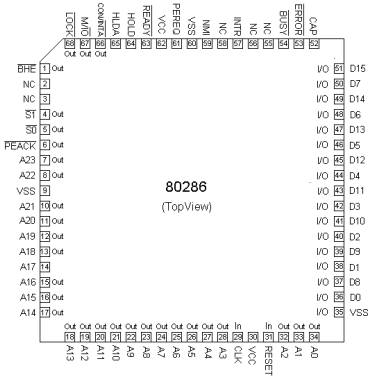




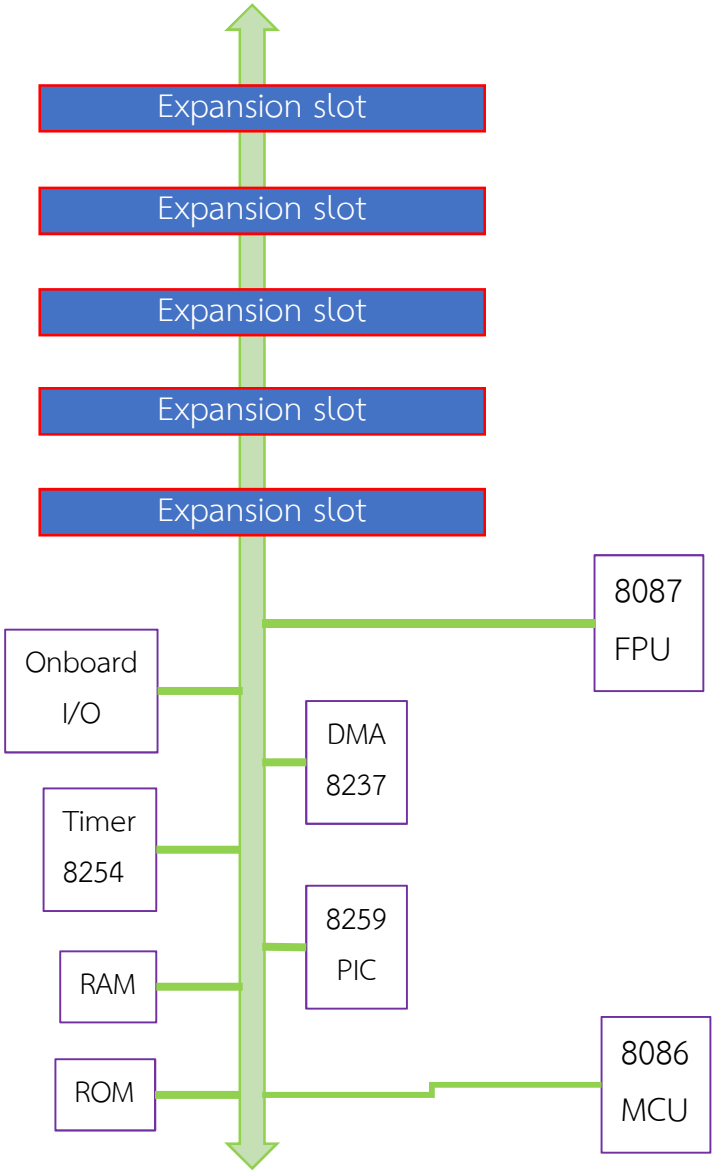


ปัจจุบันนี้ชิพเซตควบคุม I/O ถูกรวมกันเพื่อลดต้นทุนการผลิต และเพิ่มประสิทธิภาพในการสื่อสาร

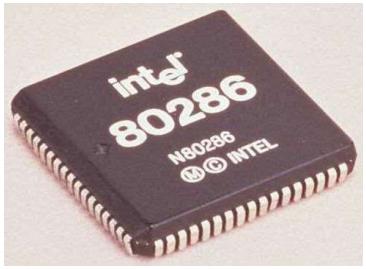


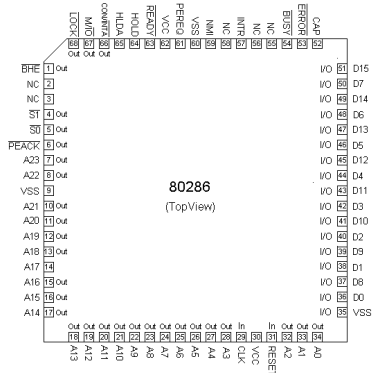


80286 มีการแยกสัญญาณ Address และ Bus ทำให้ไม่ต้องมี Latch / Buffer ในการแยกสัญญาณนี้ออกจากกันเหมือน 8086

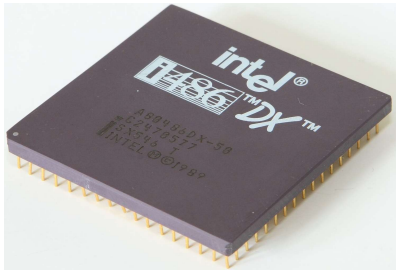


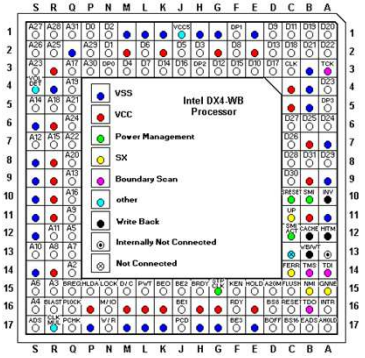
ปัจจุบันนี้ชิพเซตควบคุม I/O ถูกรวมกันเพื่อลดต้นทุนการผลิต และเพิ่มประสิทธิภาพในการสื่อสาร



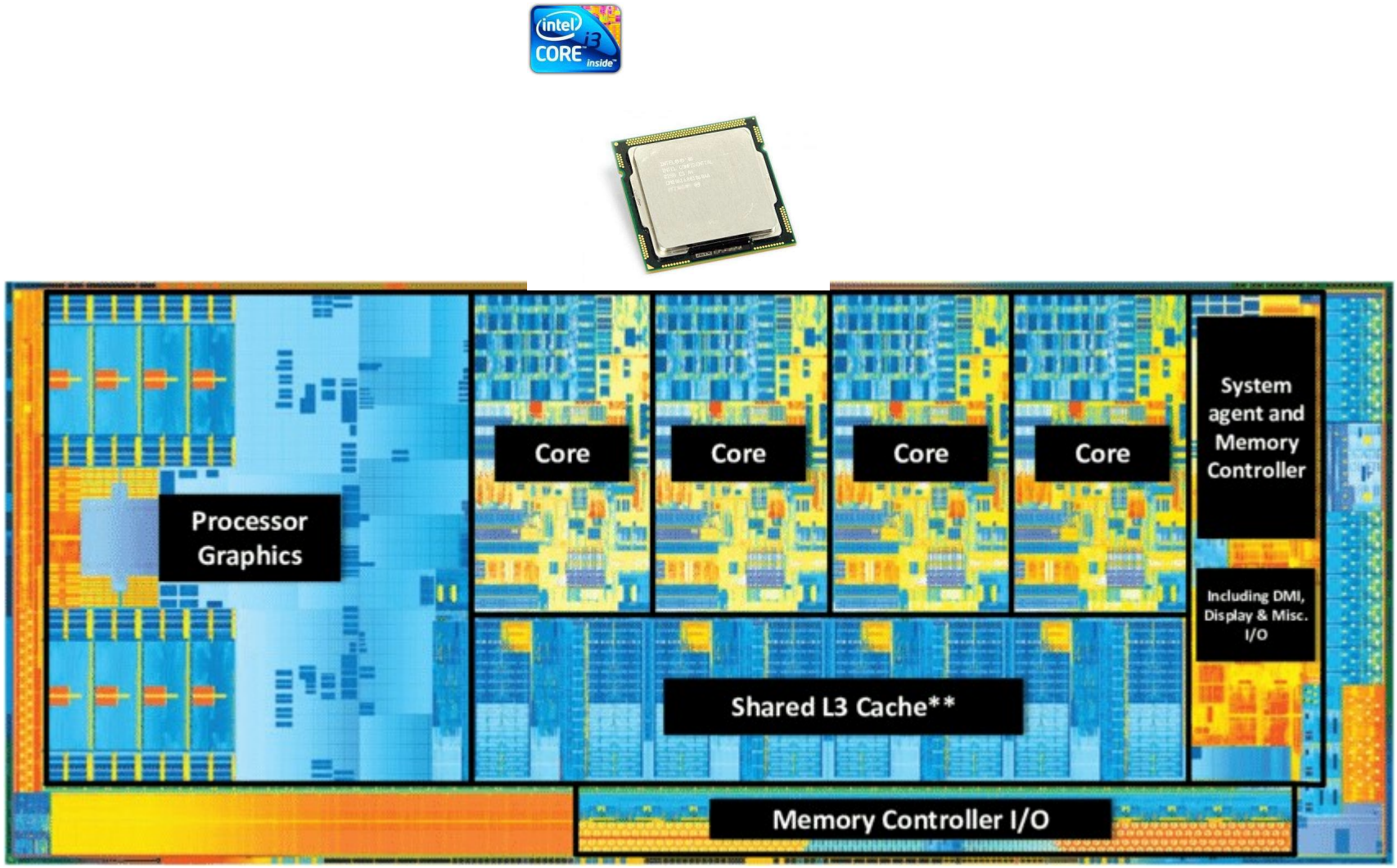


80286 มีการแยกสัญญาณ Address และ Bus ทำให้ไม่ต้องมี Latch / Buffer ในการแยกสัญญาณนี้ออกจากกันเหมือน 8086

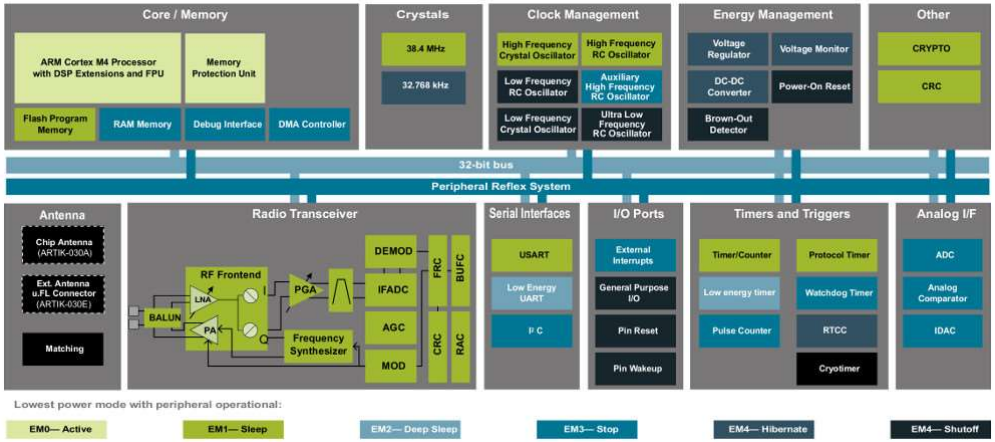


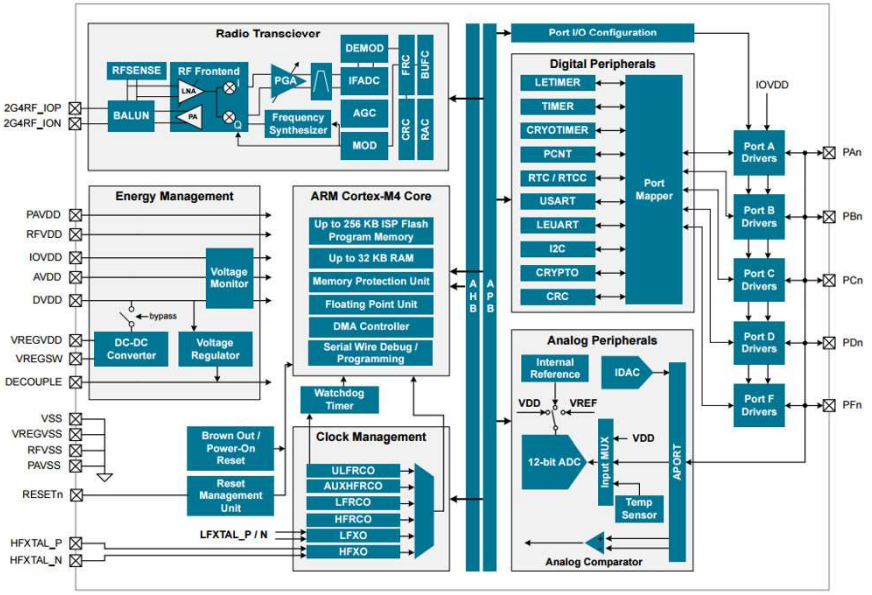


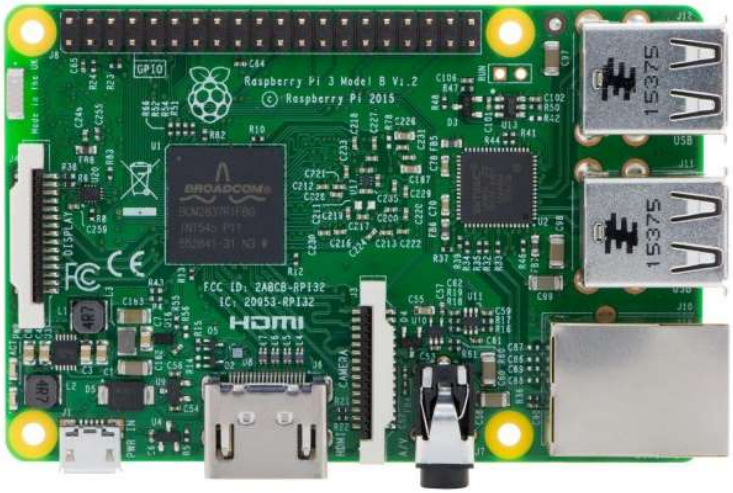
80486 ย้าย Co-processor 8087 และ Cache มาไว้ภายใน



Intel Core I ได้รวม ชิพเซ็ตควบคุมหน่วยความจำ และ I/O มาไว้ภายใน เหลือเพียง Bus controller อยู่ภายนอก







System on Chip (SoC) มีครับทั้งหมดในชิพเดียว

CS231: บทที่ 1 : ภาษาแอสเซมบลี

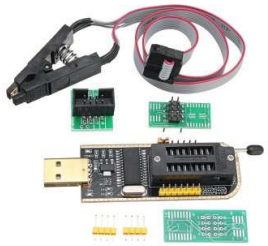
ADD Eb Gb 00	ADD Ev Gv 01	ADD Gb Eb 02	ADD Gv Ev 03	ADD AL Ib 04	ADD eAX Iv 05	PUSH ES 06	POP ES 07	OR Eb Gb 08	OR Ev Gv 09	OR Gb Eb 0A	OR Gv Ev 0B	OR AL Ib 0C	OR eAX Iv 0D	PUSH CS 0E	TWOBYTE 0F
ADC Eb Gb 10	ADC Ev Gv 11	ADC Gb Eb 12	ADC Gv Ev 13	ADC AL Ib 14	ADC eAX Iv 15	PUSH SS 16	POP SS 17	SBB Eb Gb 18	SBB Ev Gv 19	SBB Gb Eb 1A	SBB Gv Ev 1B	SBB AL Ib 1C	SBB eAX Iv 1D	PUSH DS 1E	POP DS 1F
AND Eb Gb 20	AND Ev Gv 21	AND Gb Eb 22	AND Gv Ev 23	AND AL Ib 24	AND eAX Iv 25	ES: 26	DAA 27	SUB Eb Gb 28	SUB Ev Gv 29	SUB Gb Eb 2A	SUB Gv Ev 2B	SUB AL Ib 2C	SUB eAX Iv 2D	CS: 2E	DAS 2F
XOR Eb Gb 30	XOR Ev Gv 31	XOR Gb Eb 32	XOR Gv Ev 33	XOR AL Ib 34	XOR eAX Iv 35	SS: 36	AAA 37	CMP Eb Gb 38	CMP Ev Gv 39	CMP Gb Eb 3A	CMP Gv Ev 3B	CMP AL Ib 3C	CMP eAX Iv 3D	DS: 3E	AAS 3F
INC eAX 40	INC eCX 41	INC eDX 42	INC eBX 43	INC eSP 44	INC eBP 45	INC eSI 46	INC eDI 47	DEC eAX 48	DEC eCX 49	DEC eDX 4A	DEC eBX 4B	DEC eSP 4C	DEC eBP 4D	DEC eSI 4E	DEC eDI 4F
PUSH eAX 50	PUSH eCX 51	PUSH eDX 52	PUSH eBX 53	PUSH eSP 54	PUSH eBP 55	PUSH eSI 56	PUSH eDI 57	POP eAX 58	POP eCX 59	POP eDX 5A	POP eBX 5B	POP eSP 5C	POP eBP 5D	POP eSI 5E	POP eDI 5F
PUSHA 60	POPA 61	BOUND Gv Ma 62	ARPL Ew Gw 63	FS: 64	GS: 65	OPSIZE: 66	ADSIZE: 67	PUSH Iv 68	IMUL Gv Ev Iv 69	PUSH Ib 6A	IMUL Gv Ev Ib 6B	INSB Yb DX 6C	INSW Yz DX 6D	OUTSB DX Xb 6E	OUTSW DX Xv 6F
JO Jb 70	JNO Jb 71	JB Jb 72	JNB Jb 73	JZ Jb 74	JNZ Jb 75	JBE Jb 76	JA Jb 77	JS Jb 78	JNS Jb 79	JP Jb 7A	JNP Jb 7B	JL Jb 7C	JNL Jb 7D	JLE Jb 7E	JNLE Jb 7F

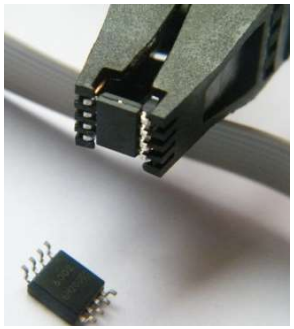
ADD Eb Ib 80	ADD Ev Iv 81	SUB Eb Ib 82	SUB Ev Ib 83	TEST Eb Gb 84	TEST Ev Gv 85	XCHG Eb Gb 86	XCHG Ev Gv 87	MOV Eb Gb 88	MOV Ev Gv 89	MOV Gb Eb 8A	MOV Gv Ev 8B	MOV Ew Sw 8C	LEA Gv M 8D	MOV Sw Ew 8E	POP Ev 8F
NOP 90	XCHG eAX eCX 91	XCHG eAX eDX 92	XCHG eAX eBX 93	XCHG eAX eSP 94	XCHG eAX eBP 95	XCHG eAX eSI 96	XCHG eAX eDI 97	CBW 98	CWD 99	CALL Ap 9A	WAIT 9B	PUSHF Fv 9C	POPF Fv 9D	SAHF 9E	LAHF 9F
MOV AL Ob A0	MOV eAX Ov A1	MOV Ob AL A2	MOV Ov eAX A3	MOVSB Xb Yb A4	MOVSW Xv Yv A5	CMPSB Xb Yb A6	CMPSW Xv Yv A7	TEST AL Ib A8	TEST eAX Iv A9	STOSB Yb AL AA	STOSW Yv eAX AB	LODSB AL Xb AC	LODSW eAX Xv AD	SCASB AL Yb AE	SCASW eAX Yv AF
MOV AL Ib B0	MOV CL Ib B1	MOV DL Ib B2	MOV BL Ib B3	MOV AH Ib B4	MOV CH Ib B5	MOV DH Ib B6	MOV BH Ib B7	MOV eAX Iv B8	MOV eCX Iv B9	MOV eDX Iv BA	MOV eBX Iv BB	MOV eSP Iv BC	MOV eBP Iv BD	MOV eSI Iv BE	MOV eDI Iv BF
#2 Eb Ib C0	#2 Ev Ib C1	RETN Iw C2	RETN C3	LES Gv Mp C4	LDS Gv Mp C5	MOV Eb Ib C6	MOV Ev Iv C7	ENTER Iw Ib C8	LEAVE C9	RETF Iw CA	RETF CB	INT3 CC	INT Ib CD	INTO CE	IRET CF
#2 Eb 1 D0	#2 Ev 1 D1	#2 Eb CL D2	#2 Ev CL D3	AAM Ib D4	AAD Ib D5	SALC D6	XLAT D7	ESC 0 D8	ESC 1 D9	ESC 2 DA	ESC 3 DB	ESC 4 DC	ESC 5 DD	ESC 6 DE	ESC 7 DF
LOOPNZ Jb E0	LOOPZ Jb E1	LOOP Jb E2	JCXZ Jb E3	IN AL Ib E4	IN eAX Ib E5	OUT Ib AL E6	OUT Ib eAX E7	CALL Jz E8	JMP Jz E9	JMP Ap EA	JMP Jb EB	IN AL DX EC	IN eAX DX ED	OUT DX AL EE	OUT DX eAX EF
LOCK: F0	INT1 F1	REPNE: F2	REP: F3	HLT F4	CMC F5	#3 Eb F6	#3 Ev F7	CLC F8	STC F9	CLI FA	STI FB	CLD FC	STD FD	#4 INC/DEC FE	#5 INC/DEC FF

AAA
AAD
AAM
AAS
ADC
ADD
AND
CALL
CBW
CLC
CLD
CLI
CMC
CMP
CMPSB
CMPSW
CWD
DAA
DAS
DEC
DIV
HLT
IDIV
IMUL
IN
INC
INT
INTO
IRET
JA
JAE
JB
JBE
JC
JCXZ
JE
JG
JGE
JL
JLE
JMP
JNA
JNAE
JNB
JNBE
JNC
JNE
JNG
JNGE
JNL
JNLE
JNO
JNP
JNS
JNZ
JO
JP
JPE
JPO
JS
JZ
LAHF
LDS
LEA
LES
LODSB
LODSW
LOOP
LOOPE
LOOPNE
LOOPNZ
LOOPZ
MOV
MOVSB
MOVSW
MUL
NEG
NOP
NOT
OR
OUT
POP
POPA
POPF
PUSH
PUSHA
PUSHF
RCL
RCR
REP
REPE
REPNE
REPNZ
REPZ
RET
RETF
ROL
ROR
SAHF
SAL
SAR
SBB
SCASB
SCASW
SHL
SHR
STC
STD
STI
STOSB
STOSW
SUB
TEST
XCHG
XLATB
XOR

คำสั่งบางตัวไม่เคยเขียนแต่ใช้ไปแล้ว เพราะ Assembler เลือกให้ตอนแปล ตัวย่างเช่น LEA

โปรแกรมทั้งหมดที่เราเขียนมา เป็นโปรแกรมแบบฝังตัว เมื่อเปิดเครื่องก็จะทำงานทันที
และ HALT เครื่องเมื่อหยุดทำงาน ไม่สามารถรันโปรแกรมอื่นต่อได้
และไม่เป็นมิตรกับระบบปฏิบัติการ





ในบทถัดไปจะศึกษาการสร้างโปรแกรมประยุกต์ ที่โหลดเข้าสู่หน่วยความจำและทำงาน
ภายใต้ระบบปฏิบัติการ MS-DOS โดยโปรแกรมสามารถเรียกใช้งาน และเมื่อจบการทำงาน
จะส่งการควบคุมกลับคืนสู่ระบบปฏิบัติการ เพื่อให้สามารถรันโปรแกรมอื่นได้ต่อไป