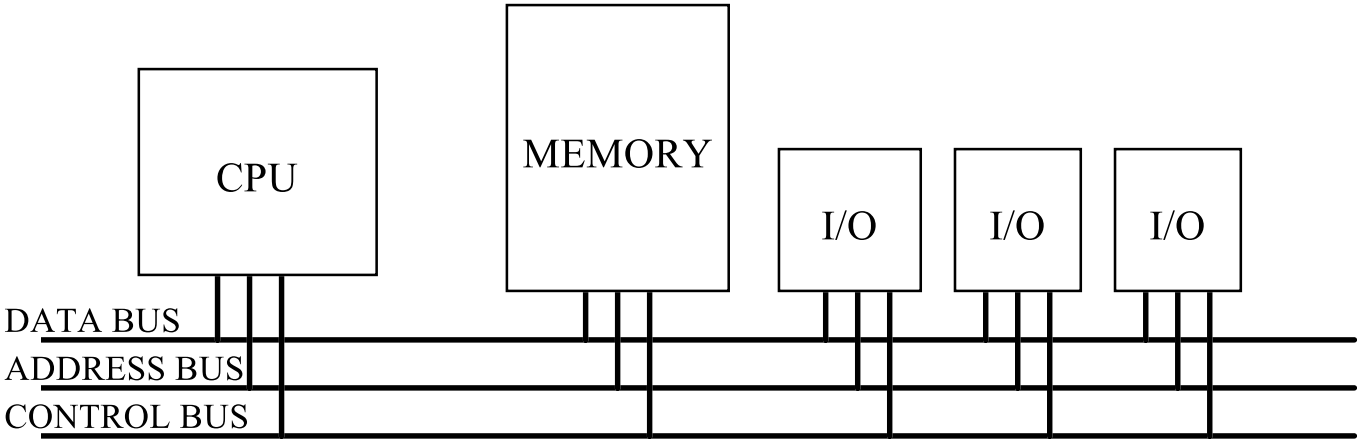
การโปรแกรมเชื่อมต่อกับอุปกรณ์รอบข้าง

- ระบบอินพุต-เอาท์พุต
- ระบบการขัดจังหวะใน 8086
- คำสั่งจัดการอินพุต-เอาท์พุต
  - ตัวอย่าง : การควบคุมลำโพง
- ระบบแสดงผลแบบตัวอักษร
  - หน่วยความจำของระบบแสดงผล
  - การควบคุมการแสดงผลโดยใช้ Video BIOS

สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

ระบบอินพุต-เอาท์พุต



- หน่วยประมวลผลจะติดต่อกับอุปกรณ์รอบข้าง (อุปกรณ์อินพุต-เอาท์พุต) โดยผ่านทางระบบบัส
- รูปแบบที่หน่วยประมวลผลจะติดต่อกับอุปกรณ์ได้นั้นมีอยู่ทั้งสิ้น 4 แบบ
  - Programmed I/O
  - Memory-mapped I/O
  - Interrupt I/O
  - Direct Memory Access I/O

การติดต่อกับอุปกรณ์

- Programmed I/O
  - หน่วยประมวลผลมีคำสั่งโดยเฉพาะสำหรับการส่ง-รับข้อมูลกับอุปกรณ์
- Memory-mapped I/O
  - การติดต่อแบบนี้ นิยมใช้ในอุปกรณ์ที่ไม่มีคำสั่งพิเศษในการติดต่อกับอุปกรณ์ I/O.
  - การติดต่อกับอุปกรณ์ I/O จะเหมือนกับการเขียนและอ่านกับหน่วยความจำ.
- Interrupt I/O
  - การที่ CPU ต้องตรวจสอบสถานะของ I/O ทำให้เสียเวลา
  - ให้ I/O รายงานแก่ CPU เองเมื่อมีการเปลี่ยนแปลงสถานะ เช่น มีข้อมูลเข้า ส่งข้อมูลเสร็จแล้ว
  - I/O สร้างสัญญาณ Interrupt ให้กับ CPU เพื่อให้ CPU ประมวลผลเหตุการณ์ที่เกิดขึ้น

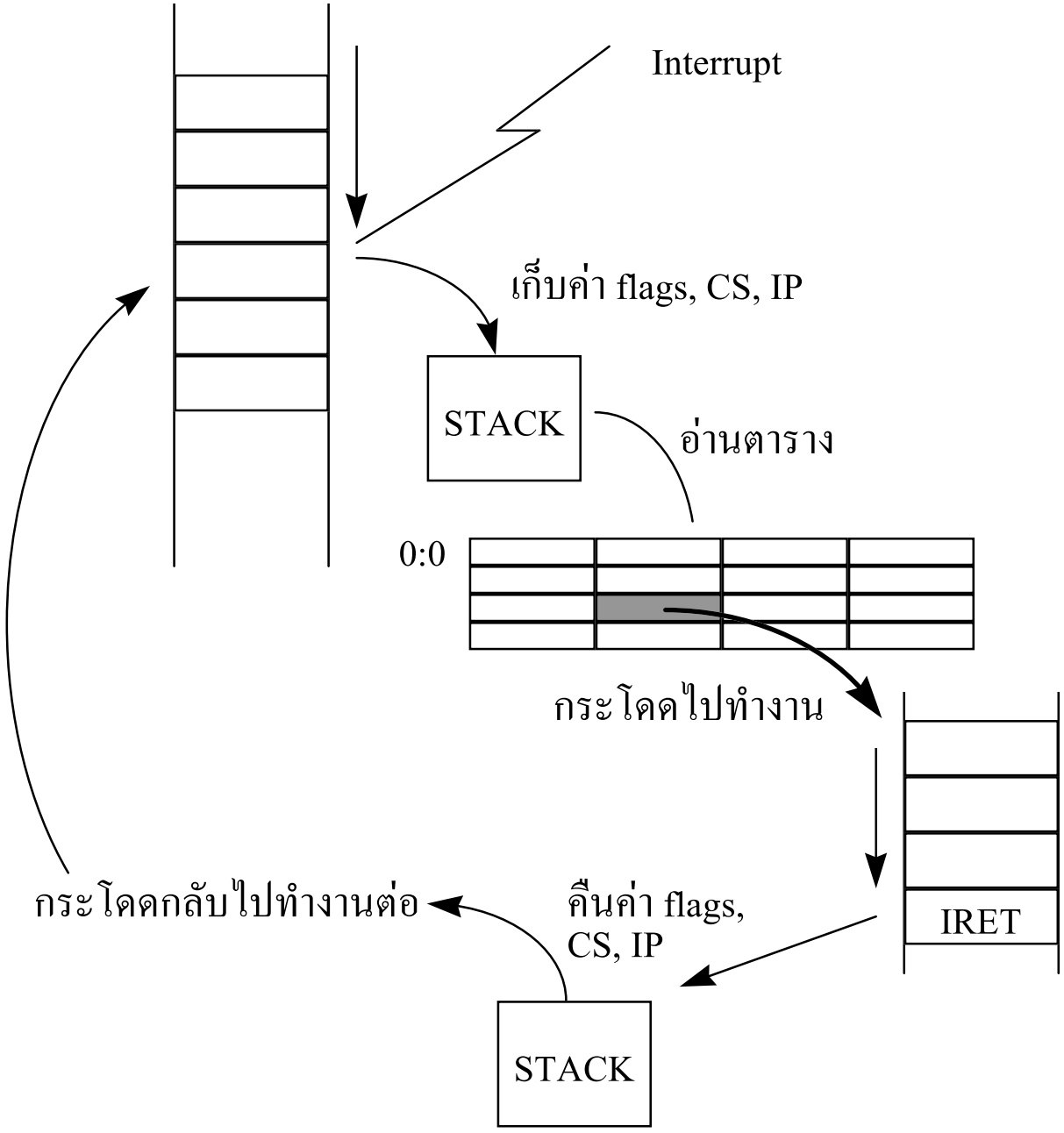
การติดต่อกับอุปกรณ์

- Direct Memory Access I/O
  - I/O ทั่วไปทำงานช้ามาเมื่อเทียบกับ CPU
  - การที่ CPU ต้องจัดการโอนย้ายข้อมูลเองจะเป็นการเสียเวลา
  - การทำงานโดยทั่วไปจึงมีขั้นตอนดังนี้ :
    - หน่วยประมวลผลสั่งงาน I/O และระบุตำแหน่งในหน่วยความจำที่จะให้ I/O เขียนผลลัพธ์ หรืออ่านข้อมูล จากนั้น CPU จะไปทำงานอื่น
    - I/O จะค่อยๆ เขียนผลลัพธ์ที่ได้ หรืออ่านข้อมูลจากหน่วยความจำโดยตรง**โดยไม่ผ่าน CPU**
    - เมื่อ I/O ทำงานเสร็จจะแจ้ง CPU โดยการสร้างการขัดจังหวะ

กระบวนการขัดจังหวะใน 8086

- เมื่อเกิดการขัดจังหวะการทำงานของหน่วยประมวลผลขึ้น
  - หน่วยประมวลผลจะเก็บสถานะการทำงานเดิมลงในแสต็ก
    - เก็บค่า flag, ค่าของรีจิสเตอร์ CS และ IP
  - การขัดจังหวะจะมีหมายเลขของการขัดจังหวะตามชนิดและประเภทของการขัดจังหวะ
  - จากนั้น CPU จะอ่านตำแหน่งของโปรแกรมจัดการการขัดจังหวะ (Interrupt Service Routine) หมายเลขที่เกิดขึ้น
  - ตำแหน่งของโปรแกรมที่จัดการการขัดจังหวะจะเก็บอยู่ในตาราง Interrupt Vecter ตารางนี้อยู่ที่หน่วยความจำตำแหน่ง 0:0

กระบวนการขัดจังหวะ



สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

ที่มาของการขัดจังหวะ

- การขัดจังหวะที่มาจากระบบฮาร์ดแวร์
  - อุปกรณ์ต่าง ๆ ที่ต้องการสร้างการขัดจังหวะจะส่งสัญญาณมาที่อุปกรณ์ควบคุมการขัดจังหวะ (Interrupt Controller) สัญญาณที่เข้ามาที่อุปกรณ์ควบคุมนี้มีทั้งสิ้น 16 สัญญาณ (IRQ0 - IRQ16)
    - มีบางสัญญาณที่ซ้ำซ้อนกัน
  - อุปกรณ์ควบคุมการขัดจังหวะจะส่งสัญญาณมาที่ CPU พร้อมทั้งส่งหมายเลขของการขัดจังหวะมาพร้อมกันด้วย
  - ถ้ามีการขัดจังหวะหลายอันที่ร้องขอพร้อมกัน อุปกรณ์นี้จะจัดลำดับความสำคัญของอุปกรณ์และส่งสัญญาณที่มีความสำคัญสูงสุดก่อน
  - อุปกรณ์ที่สร้างการขัดจังหวะกลุ่มนี้ได้แก่ : นาฬิกาของระบบ แป้นพิมพ์ ฮาร์ดดิสก์ การ์ดเสียง ฯลฯ
- การขัดจังหวะที่มาจากระบบซอฟต์แวร์
  - การเกิดความผิดพลาด (Exception) ในการทำงาน
  - การสั่งคำสั่ง INT

คำสั่งติดต่อกับอุปกรณ์ใน 8086

- หน่วยประมวลผล 8086 สามารถติดต่อกับอุปกรณ์ได้ทั้งแบบ Programmed I/O Interrupt I/O และ Direct Memory Access.
- คำสั่งเขียน/อ่านข้อมูลกับอุปกรณ์

  IN ***accumulator*****, DX**
  IN ***accumulator*****,** ***portnumber***

  OUT DX, ***accumulator***
  OUT ***portnumber*****,** ***accumulator***

  - accumulator คือรีจิสเตอร์ AX หรือ AL ขึ้นกับว่าเป็นการติดต่อแบบ 8 บิต หรือ 16 บิต
  - portnumber คือหมายเลขของอุปกรณ์ (หมายเลข I/O.) เราสามารถระบุค่านี้โดยผ่านทางรีจิสเตอร์ DX ได้. หมายเลขพอร์ตนี้มีค่าตั้งแต่ 0 ถึง 65535

ตัวอย่าง : การติดต่อกับลำโพง

- พอร์ตที่ 61h บิตที่ 1 คือค่าของสัญญาณที่ลำโพง
- เราสามารถสั่งให้ลำโพงมีเสียงได้โดยการกำหนดค่าลงในบิตนี้ โดยให้บิตนี้เปลี่ยนแปลงค่าไปมา จะทำให้เกิดการสั่นที่ลำโพงสร้างสัญญาณเสียงขึ้น
- เราจะต้องสั่งงานโดยไม่ให้กระทบกับบิตอื่น ๆ

```
in   al,61h
xor  al,02h
out  61h,al
```

- ในการสร้างเสียงเราจะต้องถ่วงเวลาในการเปลี่ยนค่าของบิตที่ลำโพง เพราะความถี่จะสูงเกินกว่าลำโพงจะเคลื่อนที่ทัน
- เราจะต้องถ่วงเวลาซักระยะหนึ่งเพื่อให้ผู้ใช้ได้ยินเสียงนั้นด้วย

โปรแกรมย่อยสำหรับสร้างเสียงอย่างง่าย

```
gensound  proc    near
;    CX  delay [for freq]
;    DX  duration
          push    ax
          push    dx
gensignal:
          in      al,61h
          xor     al,2
          out     61h,al
          push    cx
delayloop:
          nop     ;no operartion
          loop    delayloop
          pop     cx
          dec     dx
          or      dx,dx
          jnz     gensignal
          pop     dx
          pop     ax
          ret
gensound  endp
```

- การสร้างเสียงโดยทั่วไปมีวิธีการที่ซับซ้อนกว่านี้. โดยวิธีดังกล่าวจะใช้การตั้งเวลาให้ระบบสร้างสัญญาณเสียงให้

ระบบการแสดงผลแบบตัวอักษร

- ในการแสดงผลแบบตัวอักษรใน IBM/PC นั้น. ในแต่ละตัวอักษรที่แสดงจะมีค่าที่เกี่ยวข้องสองค่า คือ
  - รหัสแอสกีของตัวอักษรตัวนั้น
  - ค่าแอดตริบิวต์ (Attribute) ของตัวอักษร ซึ่งเป็นค่าที่บอกสีของตัวอักษรและลักษณะการแสดงผล
    - บิตที่ 0-3 แสดงสีของตัวอักษร (0-15)
    - บิตที่ 4-7 แสดงสีของพื้นหลัง
- เราสามารเขียนโปรแกรมจัดการการแสดงผลที่หน้าจอได้สองวิธี
  - ใช้บริการของ Video BIOS [Interrupt 10h]
  - ใช้การเขียนค่าลงในหน่วยความจำที่เก็บข้อมูลของหน้าจอโดยตรง

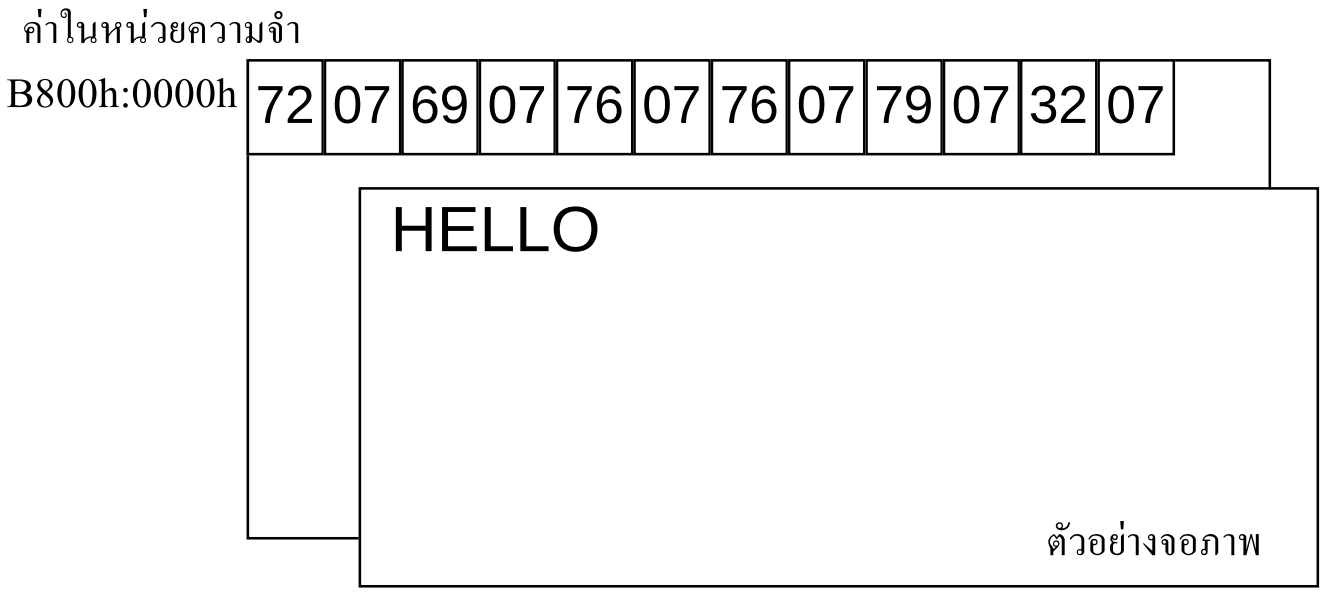
ระบบการแสดงผลแบบตัวอักษร

- Video BIOS [Interrupt 10h]
  - **Function 0** : Set video mode
  - **Function 1** : Set cursor size
  - **Function 2** : Set cursor position
  - **Function 3** : Get cursor position
  - **Function 9** : Write character and attribute - at current cursor position
- **การบ้าน#1**
  - ค้นข้อมูลของ Interrupt หมายเลข 10h
  - เขียนโปรแกรมย่อยเขียนข้อความลงบนจอภาพโดยมีข้อมูลป้อนเข้าดังนี้
    - DH และ DL : แถวและคอลัมน์ที่จะเริ่มพิมพ์ข้อความ
    - BX ออฟเซ็ตเริ่มต้นของข้อความที่ต้องการจะพิมพ์
    - CX ความยาวของข้อความ
  - ขั้นตอน
    - เลื่อนเคอร์เซอร์ไปที่ตำแหน่งที่ระบุ (function 2)
    - พิมพ์ข้อความจนครบ [ใช้ฟังก์ชั่นของ DOS ก็ได้]
  - ข้อควรระวัง :
    - video page ใช้เป็นค่า 0
    - function 9 ของ int 10h ไม่เลื่อนตำแหน่งเคอร์เซอร์

สัณฑิติ พัชรรุ่งเรือง
จิตร์ทัศน์ ฝักเจริญผล

หน่วยความจำของข้อมูลหน้าจอ

- ในหน่วยแสดงผลแบบ VGA หน่วยความจำตั้งแต่ตำแหน่งที่ 0B800h:0000h จะเป็นหน่วยความจำที่เก็บข้อมูลของหน้าจอ
- การเก็บข้อมูลจะเก็บทีละตัวอักษรเรียงกันไปตามลำดับ. การเก็บข้อมูลจะเก็บตัวอักษรละ 2 ไบต์ ไบต์แรก (ไบต์ต่ำ) จะเก็บค่ารหัสแอสกี ไบต์ที่สองจะเก็บค่าแอดตริบิวต์. การเก็บจะเก็บเรียงไปทีละบรรทัด บรรทัดละ 80 ตัวอักษร.
  - ข้อมูลของตัวอักษรที่บรรทัดที่ 10 ตัวที่ 3 อยู่ที่ออฟเซ็ต 10*(80*2) + 3*2 = 1606



ตัวอย่าง

- โปรแกรมย่อยเขียนตัวอักษรลงบนหน้าจอโดยตรง

```
writeonechar proc near
; (dh,dl) : (row,col)
; al : char   ah : attribute
            push    ds
            push    ax
            push    bx
            push    cx
            push    dx
            mov     ax,0B800h
            mov     ds,ax
            mov     cl,dh
            push    ax
            mov     al,160
            mul     cl  ;ax=row*160
            mov     bx,ax
            pop     ax
            mov     dh,0
            shl     dx,1
            add     bx,dx
            mov     [bx],ax
            pop     dx
            pop     cx
            pop     bx
            pop     ax
            pop     ds
            ret
writeonechar endp
```

- **การบ้าน#2** เขียนโปรแกรมย่อยทำงานเหมือนข้อ 1 แต่ใช้การเขียนค่าลงในหน่วยความจำโดยตรงแทนการใช้ Video BIOS. ให้ใช้ attribute เดิมของตัวอักษรที่อยู่บนจอภาพ