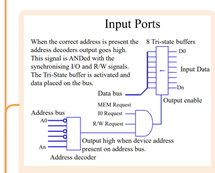
数字 IO 与中断

1. 8086 CPU 架构基础

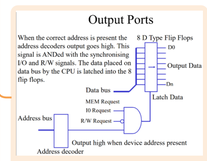
- 总线 (Bus)
  - 8086 使用 20 位的总线对内存进行读写。
- I/O 寻址空间
  - 8086 提供了 16 位的输入 / 输出地址空间。
  - 实现方式
  - I/O 空间中的位置被称为端口 (ports)。
  - 16 位的 I/O 地址空间可以被映射为 64K 个 8 位端口（字节）或 32K 个 16 位端口（字）。
- 内存映射 I/O (Memory-mapped I/O)
  - 部分硬件（如屏幕）被映射到内存空间而非 I/O 空间。

2. I/O 端口与操作

- 设备连接
  - 每个设备都共享地址总线和数据总线。
  - 每个设备都有一个唯一的地址。
  - 当一个设备的地址被放置在地址总线上时，该设备就会连接到数据总线。
- 输入端口 (Input Ports)



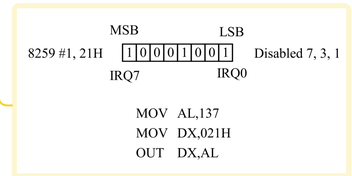
  - 'IN' 指令：用于从端口读取一个字节、字或双字到累加器。
    - 在立即数模式下，只能寻址 256 个端口。
    - 如果使用 `dx` 寄存器来指定端口，则可以访问 64K 个端口。
    - 示例：`IN AL,300h` 或 `IN AX,DX`。
  - 流程
    - 当正确的地址出现在地址总线上时，地址解码器的输出会变为高电平。
    - 这个高电平信号会与 同步 I/O 和读 / 写信号进行 “与” 运算 (ANDed)。
    - 三态缓冲器 (Tri-state buffers) 被激活，并将数据放置到总线上。
- 输出端口 (Output Ports)



  - 当正确的地址出现时，地址解码器输出高电平。
  - 这个信号与同步 I/O 和读 / 写信号进行 “与” 运算，CPU 放在数据总线上的数据被锁存到 8 个 D 型触发器 (D Type Flip Flops) 中。
  - 'OUT' 指令：用于将一个字节、字或双字从累加器写入端口。
    - 立即数模式下只能寻址 256 个端口。
    - 使用 `dx` 寄存器可以访问 64K 个端口。
    - 示例：`OUT AL,300h` 或 `OUT AX,DX`。
- I/O 权限 (8086)
  - 程序的当前权限级别 (CPL) 必须高于 I/O 权限级别 (IOPL) 才能执行 'IN' 或 'OUT' 指令。
  - 解决方案是将 IOPL 设置为 2 ('IOPL = 2')。
  - 还有一个 I/O 权限位图，大小为 8192 字节，为每个端口提供一个比特位的控制。
- 常用端口地址 4

3. 中断 (Interrupts)

- 中断基本概念
  - 中断允许正常的线性程序执行被外部信号或特殊指令打断。
- 中断流程：当中断发生时，微处理器会：
  1. 暂停主程序的执行。
  2. 调用一个服务于中断的程序（中断处理器或中断服务例程）。
  3. 将控制权返回给主程序。
- 8086 中断类型
  - **硬件中断 (Hardware Interrupts)**：由施加到处理器 INTR 或 NMI 引脚的外部信号引起。
  - **异常中断 (Exception Interrupts)**：由内部错误（如除零）触发。
  - **软件中断 (Software Interrupts)**：由执行 'INT' 汇编指令引起。
- 8259A 可编程中断控制器 (PIC)
  - 流程
    - 初始化地址：
      - 在 PC 上，主 8259A 的基地址为 '20H'。
      - 从 8259A 的基地址为 'A0H'。
    - 中断屏蔽：
      - 中断屏蔽寄存器位于 '21H' (主) 和 'A1H' (从)。



      - 设置该 8 位寄存器中的某一位会禁用对应的 IRQ 线。
    - 中断结束 (End of Interrupt, EOI)：
      - 在中断服务例程结束时，必须向控制器发送一个非特定的中断结束命令，告知中断已完成。
      - 'IRET' 指令是告诉微处理器中断结束。

      - EOI 命令示例：
  - 性质
    - PC 中通常使用两片 8259A，一片为主片 (Master)，一片为从片 (Slave)，共同提供 15 个硬件中断（IRQ）。
    - 8259 的硬件接口负责与 8086 的中断引脚对接。
    - 当 8259A 的一条 IRQ 线被拉低时，它会将中断号放到 PC 总线上。
- 中断向量表 (Interrupt Vector Table)
  - 存储位置
    - 位于 RAM 的 `0000:0000` 到 `0000:03FF`，即前 1024 字节。
  - 容量
    - 可以存储 256 个中断向量。
    - 每个中断向量占 4 个字节，两个字节用于代码段 (CS)，两个用于指令指针 (IP)。
  - 工作机制
    - 向量地址 = 4 * 中断号。
    - 当中断发生时，CPU 会查找此表以跳转到对应的服务程序地址。
- 中断处理流程
  1. 硬件通过拉低 IRQx 线请求服务。
  2. 8259A 通过 INTR 线向 CPU 发出信号。
  3. CPU 以中断确认信号 (INTA) 响应。
  4. 8259A 将中断号放置在数据总线上。
  5. CPU 读取中断号。
  6. CPU 将相关寄存器（CS:IP, SP, Flags）压入堆栈。
  7. CPU 将中断号乘以 4，在向量表中查找跳转地址。
  8. CS 和 IP 被改变，开始执行中断服务例程 (ISR)。
  9. ISR 执行直到遇到 'IRET' 指令。
  10. 'IRET' 恢复寄存器的值，主程序继续执行。
- 典型 IRQ 分配

4. 内存直接访问与屏幕操作

- 直接屏幕写入
  - 写入示例：

```
mov ax,0B800h
mov es,ax
mov byte ptr es:[0],'A'
mov byte ptr es:[1],1eh
```

  - 性质
    - 文本屏幕被内存映射到以地址 'B8000h' 开始的区域。
    - 这是一个非常快速但依赖于特定硬件的操作。
- 屏幕内存结构
  - 属性字节 (Attribute Byte)：
    - 位 7 (F)：闪烁 (Flash)。
    - 位 6-4：背景色 (Background)。
    - 位 3 (I)：强度 (Intensity)。
    - 位 2-0：前景色 (Foreground)。
  - 存储方式
    - 屏幕内存中，偶数地址存放属性字节 (Attribute)。
    - 奇数地址存放 ASCII 字符码 (Code)。

5. 调度与多任务

- 时间片调度 (Time Slice Scheduling)
  - 意义：这让用户感觉所有任务在同时运行。
  - 定义
    - 操作系统决定何时从一个任务切换到下一个任务。
    - 在这种模式下，CPU 实际上是在轮询每个硬件。
    - 每个任务被分配一定比例的 CPU 时间。
    - 在轮询 (round-robin) 实现中，每个任务可能每 20 毫秒被访问一次。
- 抢占式调度 (Pre-emptive Scheduling)
  - 低优先级的任务可以被高优先级的任务中断。
  - 意义

6. 终止并驻留程序 (TSR)

- TSR 概念
  - TSR 程序用于在基于 DOS 的应用中提供某种程度的多任务处理。
  - TSR 中的例程通常由中断调用。
  - 可以创建在后台运行的弹出式程序，如 Sidekick 或拼写检查器。
  - 由于它们保持驻留状态，应谨慎使用。
- TSR 编程要点
  - 重定向中断 (Re-vectoring Interrupts)：

```
push    es                ; Save existing value of ES.
mov     ax,0000h          ; Set ES (extra segment) to page 0
mov     es,ax             ; Note page 0 is vector address table
mov     bx,020h           ; Vector address for timer INT 8, IRQ0
mov     ax,es:[bx]        ; ax=[0000:0020], CS to jump to
mov     vtabip,ax         ; Store the CS, 1st two bytes of table
inc     bx
inc     bx
mov     ax,es:[bx]        ; ax=[0000:0020], IP to jump to
mov     vtabcs,ax         ; Store the IP, 2nd two bytes
pop     es                ; Restore the value of es
```

    1. **保存旧向量**：首先读取并保存中断向量表中现有的服务例程地址 (CS:IP)。

```
push    es                ; Change vector address to ISR routine
mov     ax,0000h          ; Vector address table segment
mov     es,ax
mov     bx,020h
mov     ax,OFFSET isr     ; Set IP of vector table to new ISR
mov     es:[bx],ax
inc     bx
inc     bx
mov     ax,cs             ; The ISR is in the same segment as the program.
mov     es:[bx],ax        ; New CS in vector table is the code segment of
pop     es                ; this program.
```

    2. **设置新向量**：将中断向量表中的地址修改为自定义 ISR 的地址。

```
Do something for a while and then
restore the vector table.

call delay              ; Kill time for a while, isr will occur a few
                        ; times during this delay.
push es
mov ax,0000h
mov es,ax
mov bx,020h
mov ax,vtabip           ; Put the old vector table values back
mov es:[bx],ax
inc bx
inc bx
                        Note: With a TSR the code on this
mov ax,vtabcs           page is not included. If this is done
mov es:[bx],ax          then the interrupt will call the code
pop es                  contained in the TSR.
```

    3. **恢复旧向量**：程序结束前（或在适当的时候），将原始的向量地址写回向量表。
  - 中断服务例程 (ISR)：
    - 注意点
      1. 手动保存通用寄存器
      2. 结束前发送 EOI
      3. 执行ISR后跳转回原址
      4. 结束时必须按相反顺序恢复所有保存的寄存器。
  - 驻留内存：
    - 使用 'INT 21h' 的 '31h' 功能号使程序终止并保持驻留。
    - 'DX' 寄存器指定需要保持驻留的字节数。