Microprocessors & Interfacing

INTERRUPTS

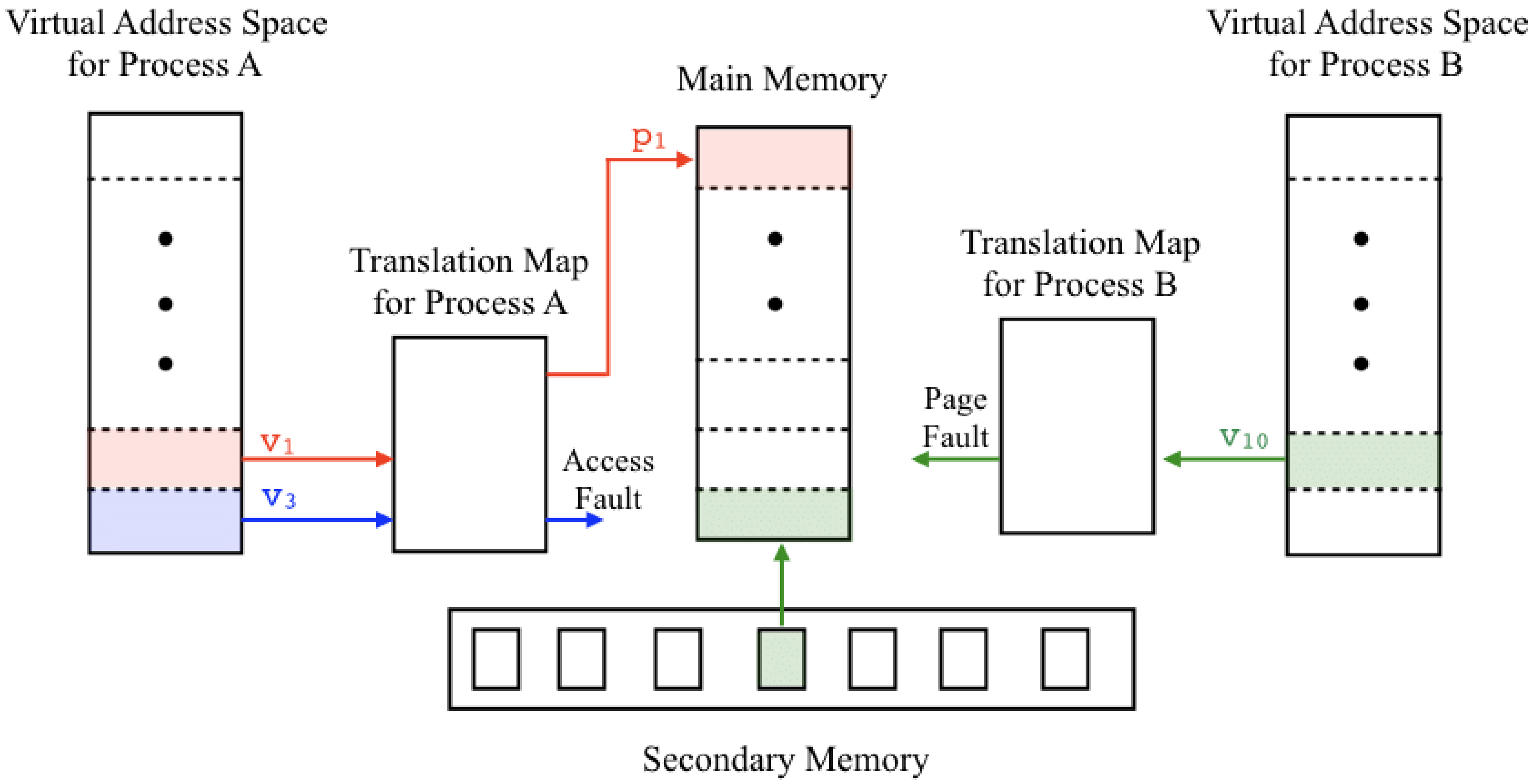
BITS Pilani

Dr. Gargi Prabhu
Department of CS & IS

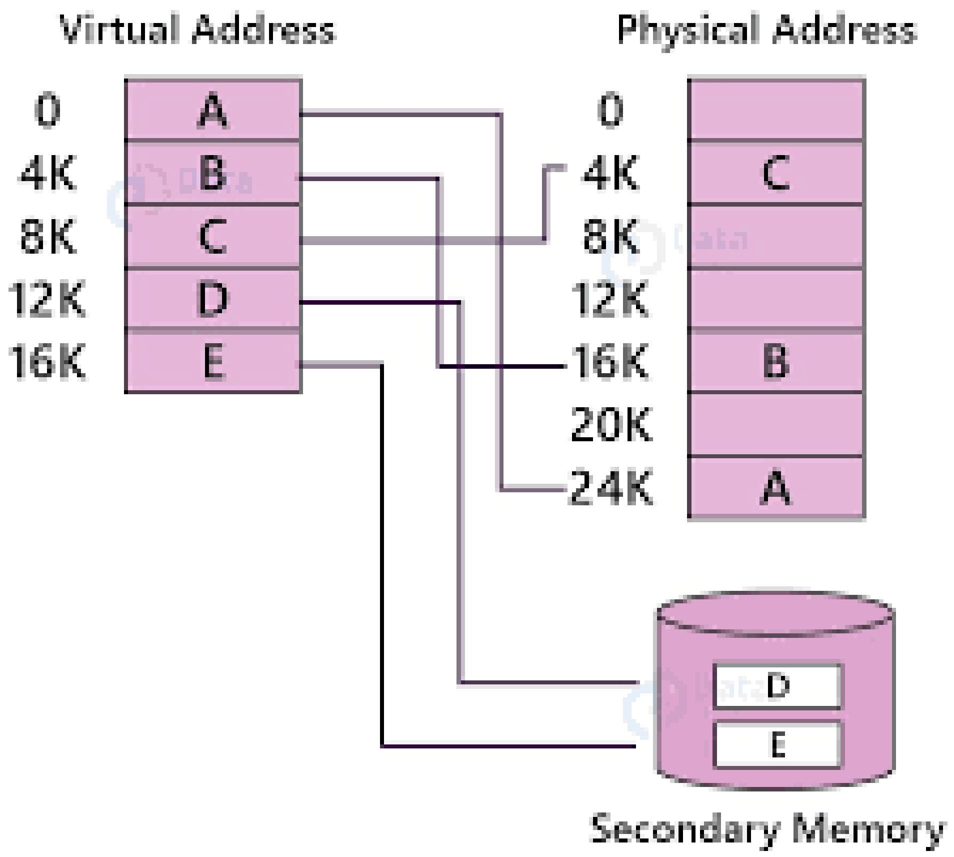
Virtual Memory

- Virtual memory is a memory management technique where secondary memory can be used as if it were a part of the main memory.
- Virtual memory uses both hardware and software to enable a computer to compensate for physical memory shortages, temporarily transferring data from random access memory (RAM) to disk storage.
- Mapping chunks of memory to disk files enables a computer to treat secondary memory as though it were main memory.

Virtual Memory


Virtual Address Space for Process A
Main Memory
Virtual Address Space for Process B
p1
Translation Map for Process A
Translation Map for Process B
Page Fault
v1
v3
Access Fault
v10
Secondary Memory

Virtual Memory



Virtual Memory

Two approaches:

a) Paging:

- Paging divides memory into sections or paging files.
- When a computer uses up its available RAM, pages not in use are transferred to the hard drive using a swap file.
- A swap file is a space set aside on the hard drive to be used as the virtual memory extension for the computer's RAM.
- When the swap file is needed, it is sent back to RAM using a process called page swapping.

Virtual Memory

Two approaches:

b) Segmentation:

- This approach divides virtual memory into segments of different lengths.
- Segments not in use in memory can be moved to virtual memory space on the hard drive.
- Segmented information or processes are tracked in a segment table, which shows if a segment is present in memory, whether it has been modified and what its physical address is

Benefits of Virtual Memory

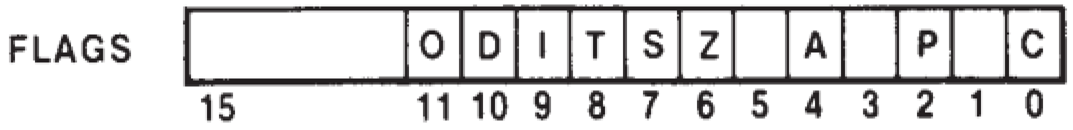
- It can handle twice as many addresses as main memory.
- It enables more applications to be used at once.
- It frees applications from managing shared memory and saves users from having to add memory modules when RAM space runs out.
- It has increased speed when only a segment of a program is needed for execution.
- It has increased security because of memory isolation.
- It enables multiple larger applications to run simultaneously.
- Allocating memory is relatively inexpensive.
- Data can be moved automatically.

Disadvantages

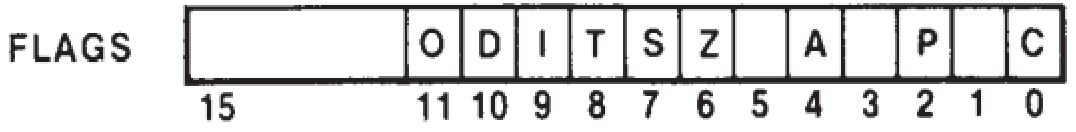
- Applications run slower if they are running from virtual memory.
- Data must be mapped between virtual and physical memory, which requires extra hardware support for address translations, slowing down a computer further.
- The size of virtual storage is limited by the amount of secondary storage, as well as the addressing scheme with the computer system.
- Thrashing can occur if there is not enough RAM, which will make the computer perform slower.
- It may take time to switch between applications using virtual memory.
- It lessens the amount of available hard drive space.

Interrupt Flag Bits

- The interrupt flag (IF) and the trap flag (TF) are both cleared after the contents of the flag register are stacked during an interrupt.



Interrupt Flag Bits



- When the IF bit is set, it allows the INTR pin to cause an interrupt; when the IF bit is cleared, it prevents the INTR pin from causing an interrupt.
- When TF = 1, it causes a trap interrupt (type number 1) to occur after each instruction executes.
- This is why we often call trap a single-step. When TF = 0, normal program execution occurs.

Interrupt Flag Bits

- The interrupt flag is set and cleared by the STI and CLI instructions, respectively.
- There are no similar special instructions that set or clear the trap flag.

Example

```
;A procedure that sets the TRAP flag bit to enable trapping

TRON   PROC    FAR USES AX BP

       MOV   BP,SP              ;get SP
       MOV   AX[BP+8]           ;retrieve flags from stack
       OR    AH,1               ;set trap flag
       MOV   [BP+8],AX
       IRET

TRON   ENDP
```

Example

```
;A procedure that clears the TRAP flag to disable trapping

TROFF PROC    FAR USES AX BP

      MOV   BP,SP             ;get SP
      MOV   AX,[BP+8]         ;retrieve flags from stack
      AND   AH,0FEH           ;clear trap flag
      MOV   [BP+8],AX
      IRET

TROFF ENDP
```

Storing an Interrupt Vector in the Vector Table

- INT40 function has an IRET instruction before ENDP.
- This is required because the assembler has no way of determining if the FAR procedure is an interrupt procedure.
- Normal FAR procedures do not need a return instruction, but an interrupt procedure does need an IRET.
- Interrupts must always be defined as FAR.

```
.MODEL TINY
.CODE
.STARTUP
        JMP     START
OLD     DD      ?       ;space for old vector

NEW40   PROC    FAR     ;must be FAR

;
;Interrupt software for INT 40H
;

        IRET            ;must have an IRET

NEW40   ENDP
```

```
;start installation

START:
        MOV     AX,0                    ;address segment 0000H
        MOV     DS,AX
        MOV     AX,DS:[100H]            ;get INT 40H offset
        MOV     WORD PTR CS:OLD,AX      ;save it
        MOV     AX,DS:[102H]            ;get INT 40H segment
        MOV     WORD PTR CS:OLD+2,AX    ;save it
        MOV     DS:[100H],OFFSET NEW40  ;save offset
        MOV     DS:[102H],CS            ;save segment
        MOV     DX,OFFSET START
        SHR     DX,4
        INC     DX
        MOV     AX,3100H                ;make NEW40 resident
        INT     21H

END
```

Storing an Interrupt Vector in the Vector Table

- The vector for INT 40H, for interrupt procedure NEW40, is installed in memory at real mode vector location 100H–103H
- First, old interrupt vector contents are saved in case we need to uninstall the vector.
- Function AX = 3100H for INT 21H, the DOS access function, installs the NEW40 procedure in memory until the computer is shut off.
- The number in DX is the length of the software in paragraphs (16-byte chunks).

```
.MODEL TINY
.CODE
.STARTUP
        JMP     START
OLD     DD      ?       ;space for old vector

NEW40   PROC    FAR     ;must be FAR

;
;Interrupt software for INT 40H
;

        IRET            ;must have an IRET

NEW40   ENDP
```

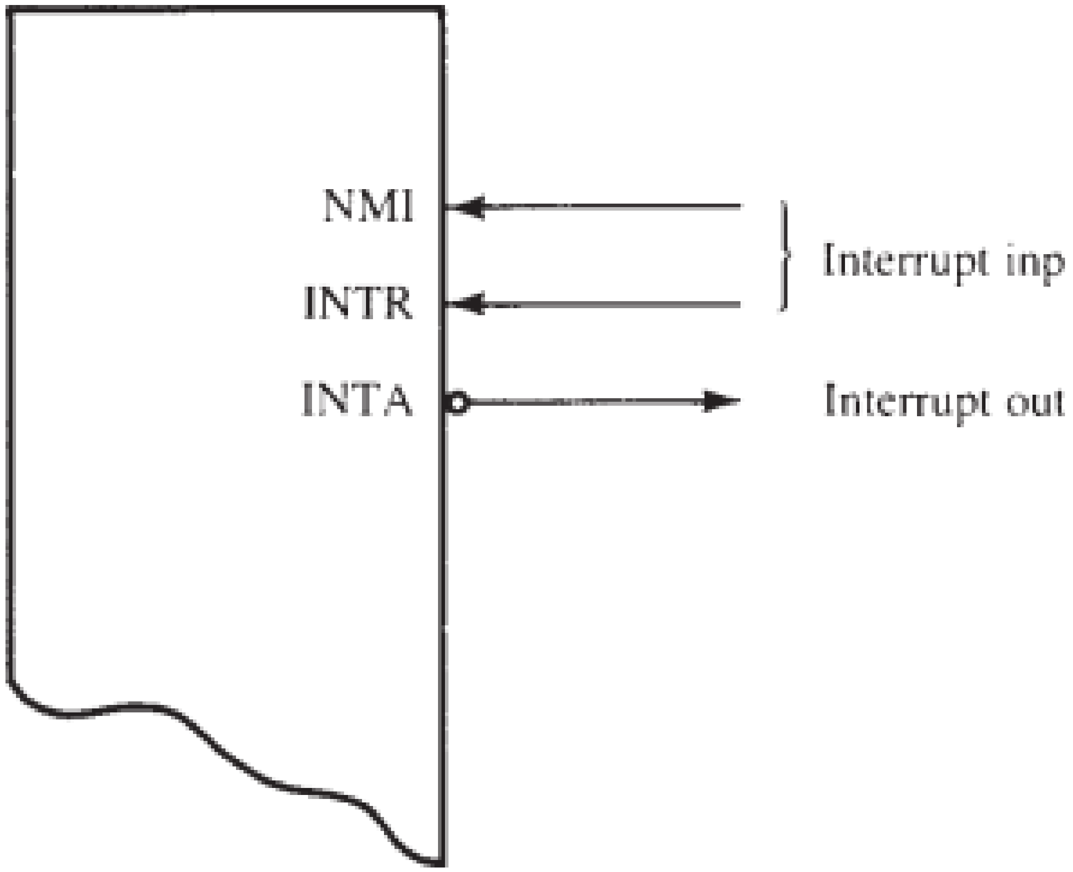
```
;start installation

START:
        MOV     AX,0                    ;address segment 0000H
        MOV     DS,AX
        MOV     AX,DS:[100H]            ;get INT 40H offset
        MOV     WORD PTR CS:OLD,AX      ;save it
        MOV     AX,DS:[102H]            ;get INT 40H segment
        MOV     WORD PTR CS:OLD+2,AX    ;save it
        MOV     DS:[100H],OFFSET NEW40  ;save offset
        MOV     DS:[102H],CS            ;save segment
        MOV     DX,OFFSET START
        SHR     DX,4
        INC     DX
        MOV     AX,3100H                ;make NEW40 resident
        INT     21H

END
```

Hardware Interrupts

The microprocessor has two hardware interrupt inputs:

- non-maskable interrupt (NMI)
- interrupt request (INTR)
- Whenever the NMI input is activated, a type 2 interrupt occurs because NMI is internally decoded.
- The INTR input must be externally decoded to select a vector.
- Any interrupt vector can be chosen for the INTR pin, but we usually use an interrupt type number between 20H and FFH.
- The INTA’ signal is also an interrupt pin on the microprocessor, but it is an output that is used in response to the INTR input to apply a vector type number to the data bus connections D7–D0.

Interrupt Pins

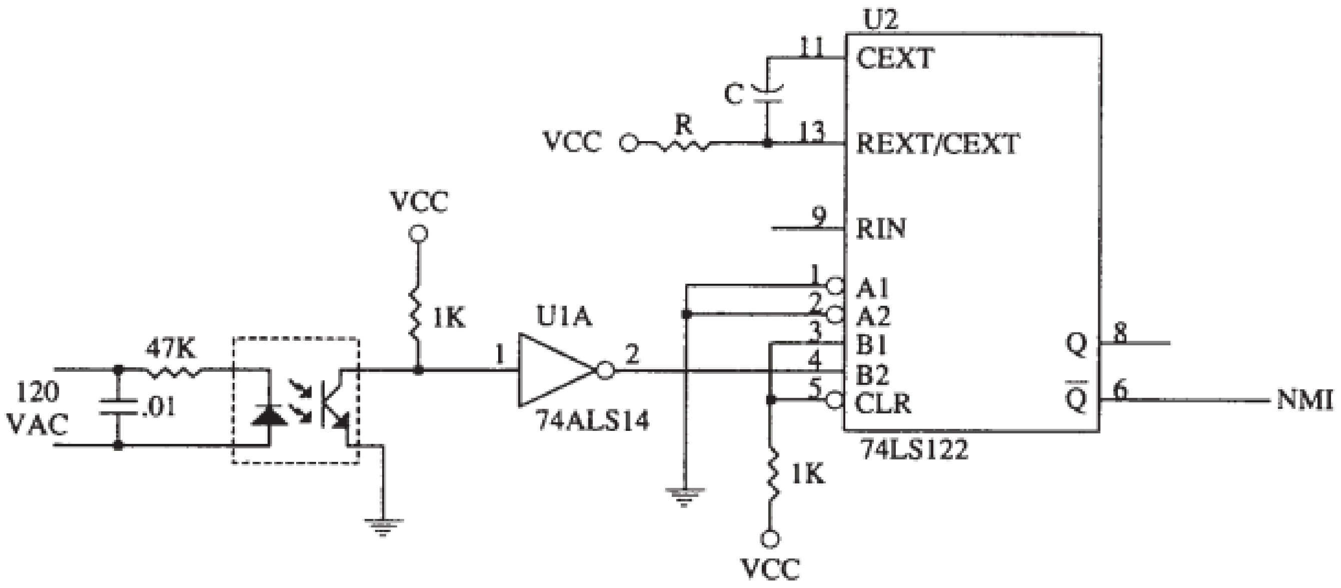


Non-Maskable Interrupts

- The non-maskable interrupt (NMI) is an edge-triggered input that requests an interrupt on the positive edge (0-to-1 transition).
- After a positive edge, the NMI pin must remain a logic 1 until it is recognized by the microprocessor.
- Note that before the positive edge is recognized, the NMI pin must be a logic 0 for at least two clocking periods.
- Used for **parity errors** and **other major system faults**, such as power failures.

Non-Maskable Interrupts

- Power failures are easily detected by monitoring the AC power line and causing an NMI interrupt whenever AC power drops out.
- In response to this type of interrupt, the microprocessor stores all of the internal register in a battery-backed-up memory or an EEPROM.

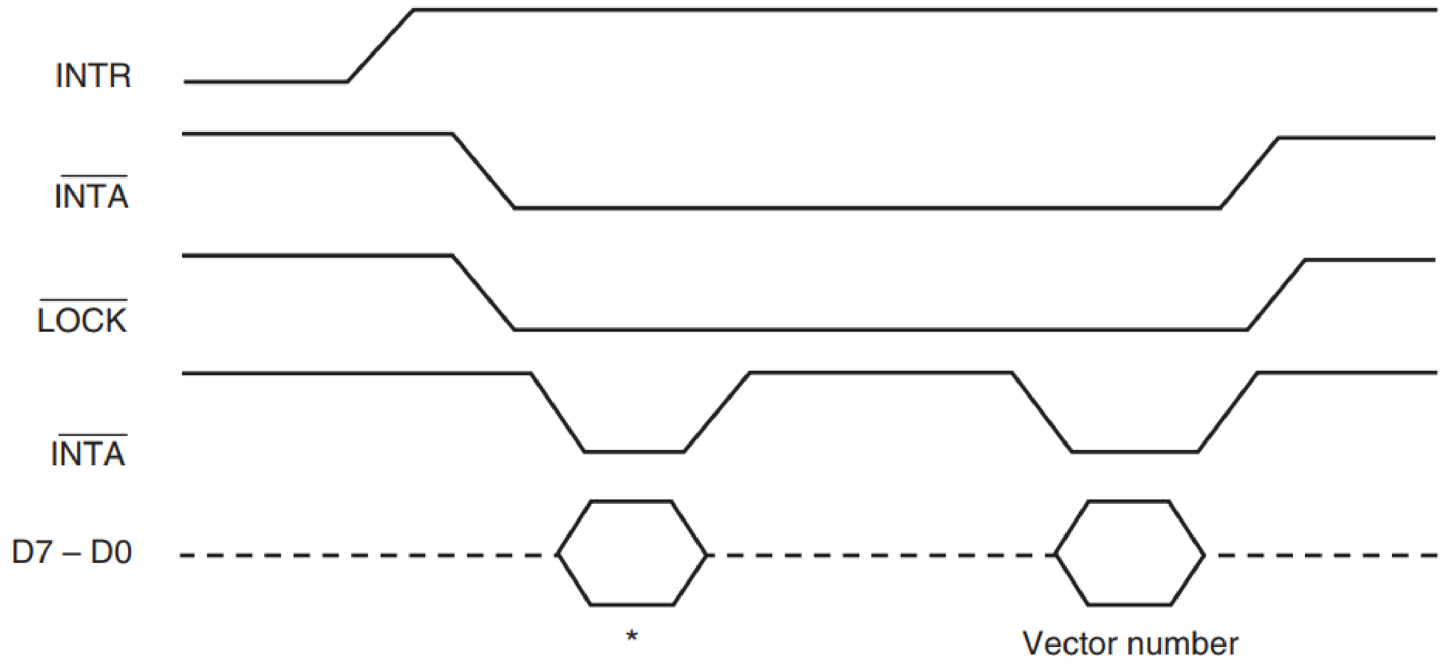


INTR and INTA’

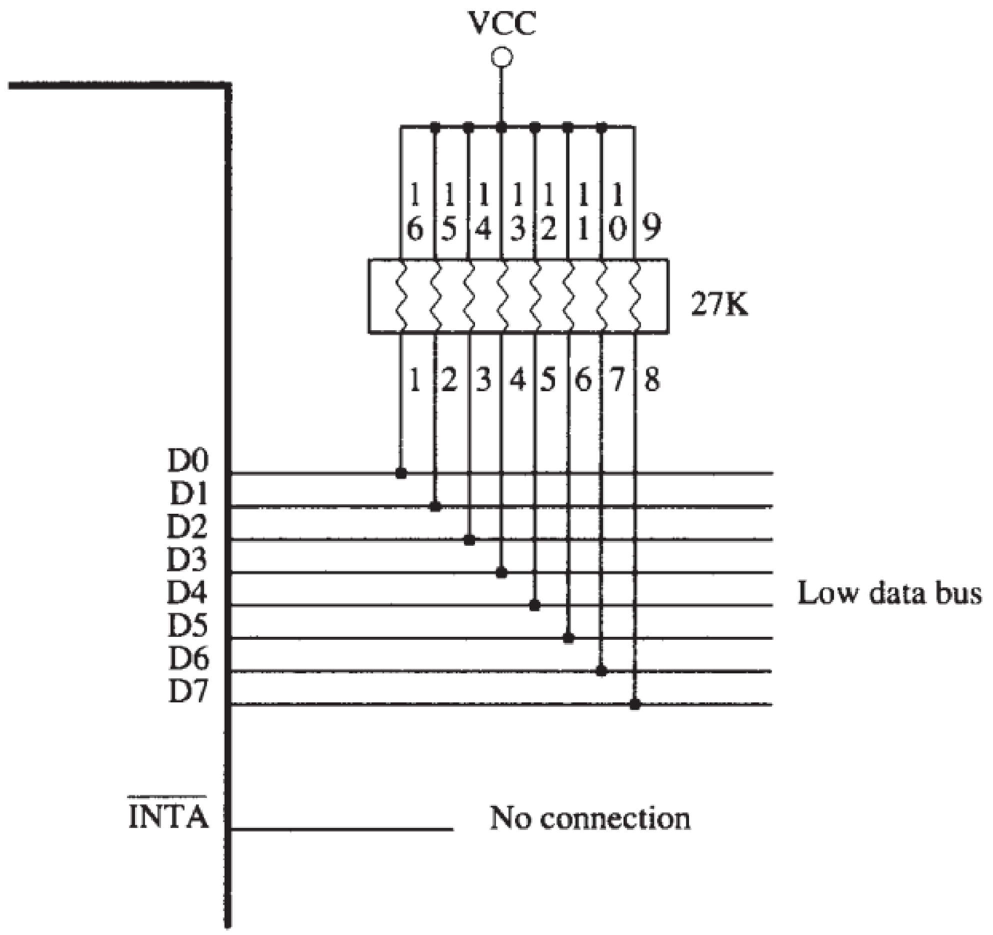
- The interrupt request input (INTR) is level-sensitive, which means that it must be held at a logic 1 level until it is recognized.
- The INTR pin is set by an external event and cleared inside the interrupt service procedure.
- This input is automatically disabled once it is accepted by the microprocessor and re-enabled by the IRET instruction at the end of the interrupt service procedure.
- The 80386–Core2 use the IRETD instruction in the protected mode of operation. In the 64-bit mode, an IRETQ is used in protected mode

INTR and INTA’

- The microprocessor responds to the INTR input by pulsing the output INTA’ in anticipation of receiving an interrupt vector type number on data bus connections D7–D0.
- There are two INTA’ pulses generated by the system that are used to insert the vector type number on the data bus.



INTR and INTA’



innovate achieve lead

BITS Pilani
Pilani Campus

Thank You