Microprocessors and Microcontrollers (EE-231)

Lecture-15

Main Objectives

- ADC and its interfacing to Microprocessor
- Interrupts
  - Basic Interrupt Processing
  - Hardware interfacing of Interrupts
  - Expansion of Interrupt Structure

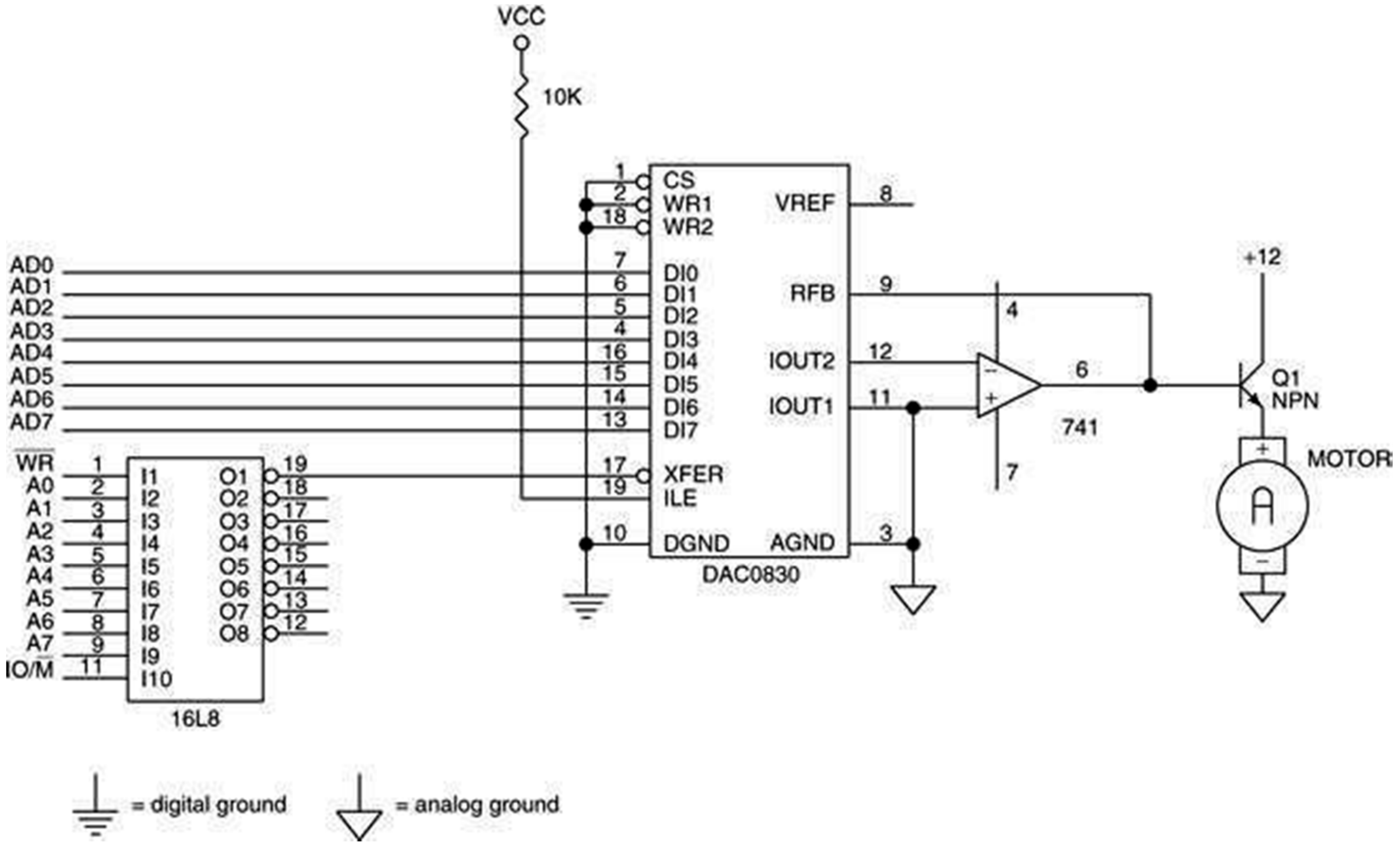
Connecting the DAC0830 to the Microprocessor.

- A PLD is used to decode the DAC0830 at I/O port address 20H.
  - when an OUT 20H,AL instruction is executed, contents of data bus connections AD_0–AD_7 are passed to the converter in the DAC0830
- The 741 operational amplifier, along with the –12 V zener reference voltage, causes the full-scale output voltage to equal +12 V.

Connecting the DAC0830 to the Microprocessor.

- A PLD is used to decode the DAC0830 at I/O port address 20H.
  - when an OUT 20H,AL instruction is executed, contents of data bus connections AD_0–AD_7 are passed to the converter in the DAC0830
- The 741 operational amplifier, along with the –12 V reference voltage, causes the full-scale output voltage to equal +12 V.

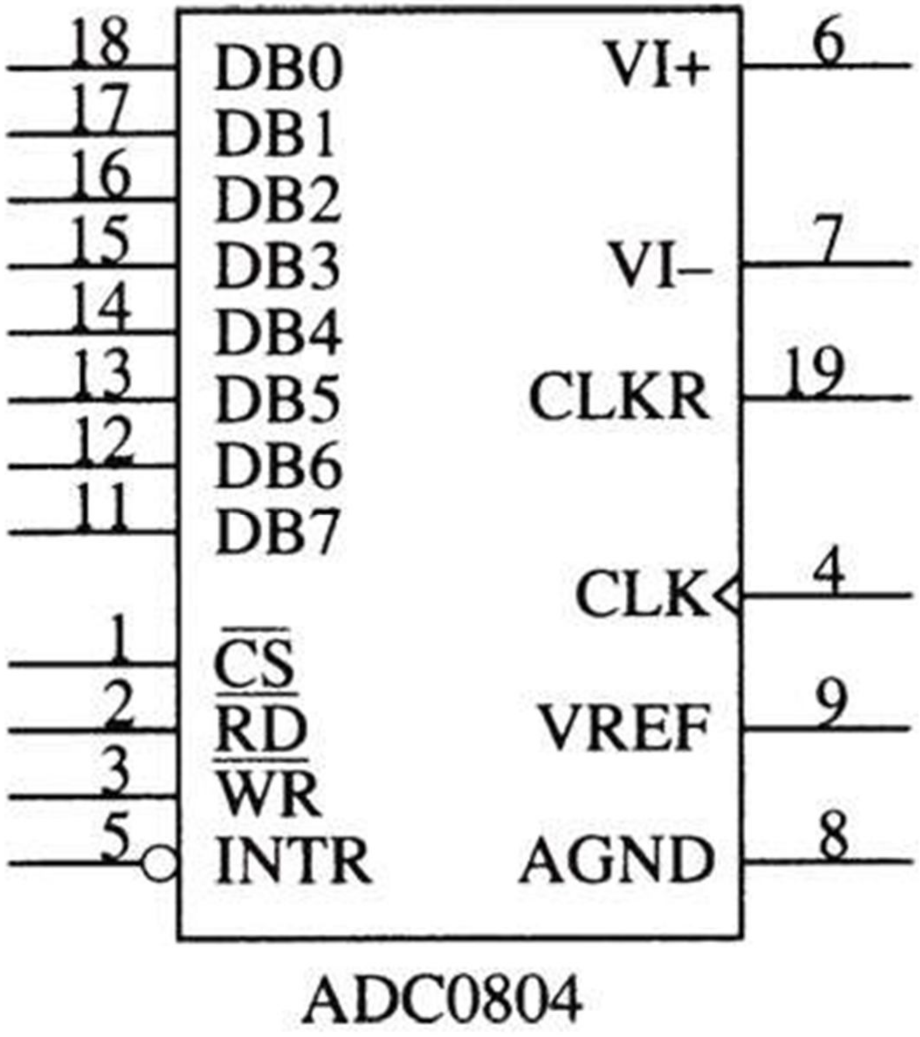
A DAC0830 interfaced to the 8086 microprocessor at 8-bit I/O location 20H.



The ADC080X Analog-to-Digital Converter

- A common, low-cost ADC, compatible with a wide range of microprocessors.
  - while there are faster ADCs available with more resolution, this device is ideal for applications
    that do not require a high degree of accuracy
- ADC080X requires up to 100 μs to convert an analog input voltage into a digital output code.

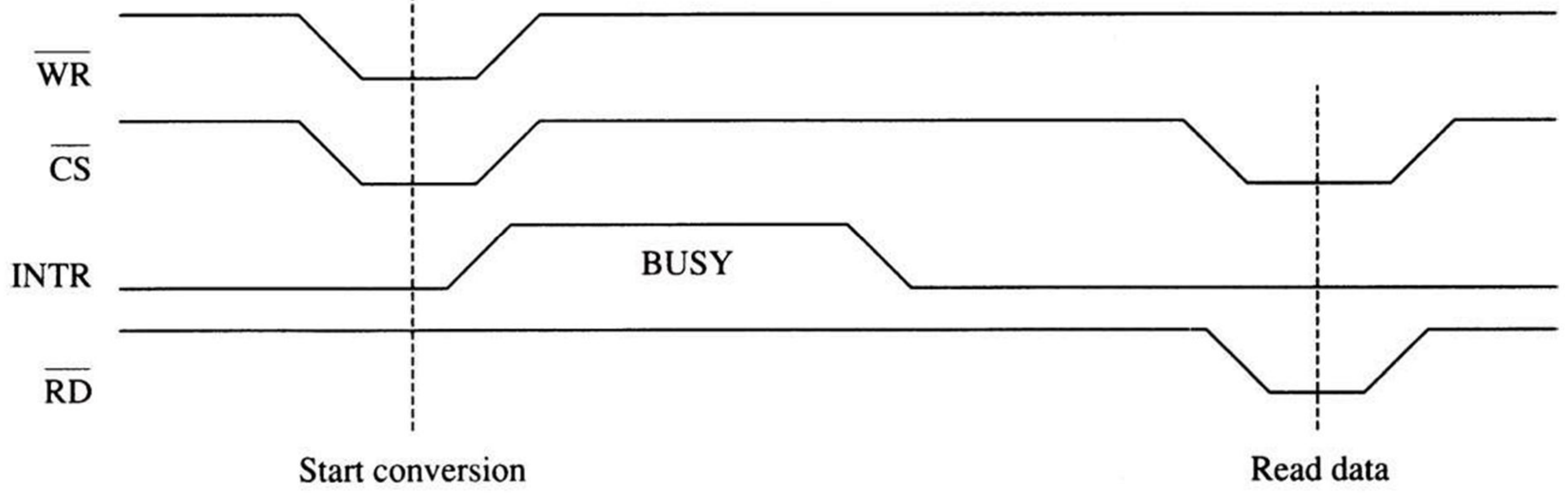
The pin-out of the ADC0804 analog-to-digital converter.



The ADC080X Analog-to-Digital Converter

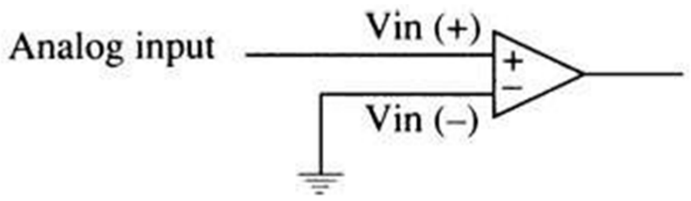
- To operate the converter, the WR pin is pulsed with CS grounded to start the conversion process.
- If a time delay is used that allows at least 100 μs of time, there is no need to test INTR pin.
- Another option is to connect the INTR pin to an interrupt input, so when the conversion is complete, an interrupt occurs.

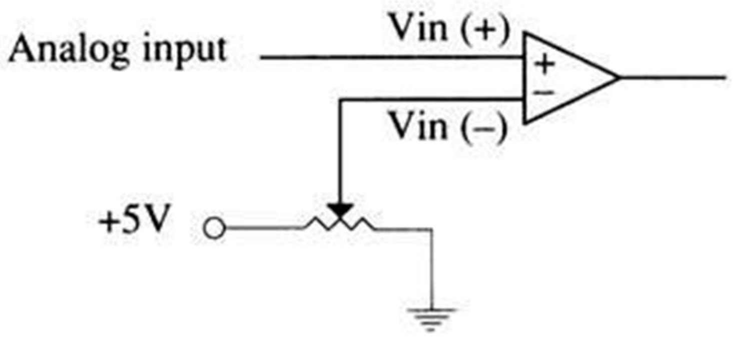
The timing diagram for the ADC0804 analog-to-digital converter



The Analog Input Signal

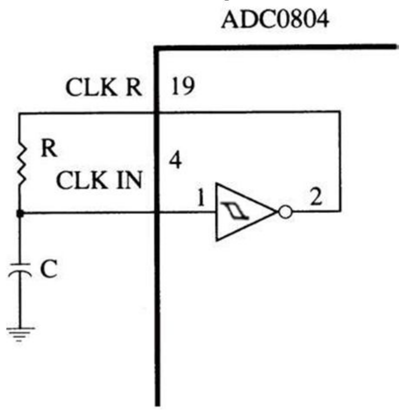
- Before ADC0804 can be connected, the two analog inputs must be understood:
  - VIN(+) and VIN(–)
- These differential inputs are summed by the operational amplifier to produce a signal for the internal analog-to-digital converter.
- These inputs are connected to an internal operational amplifier.



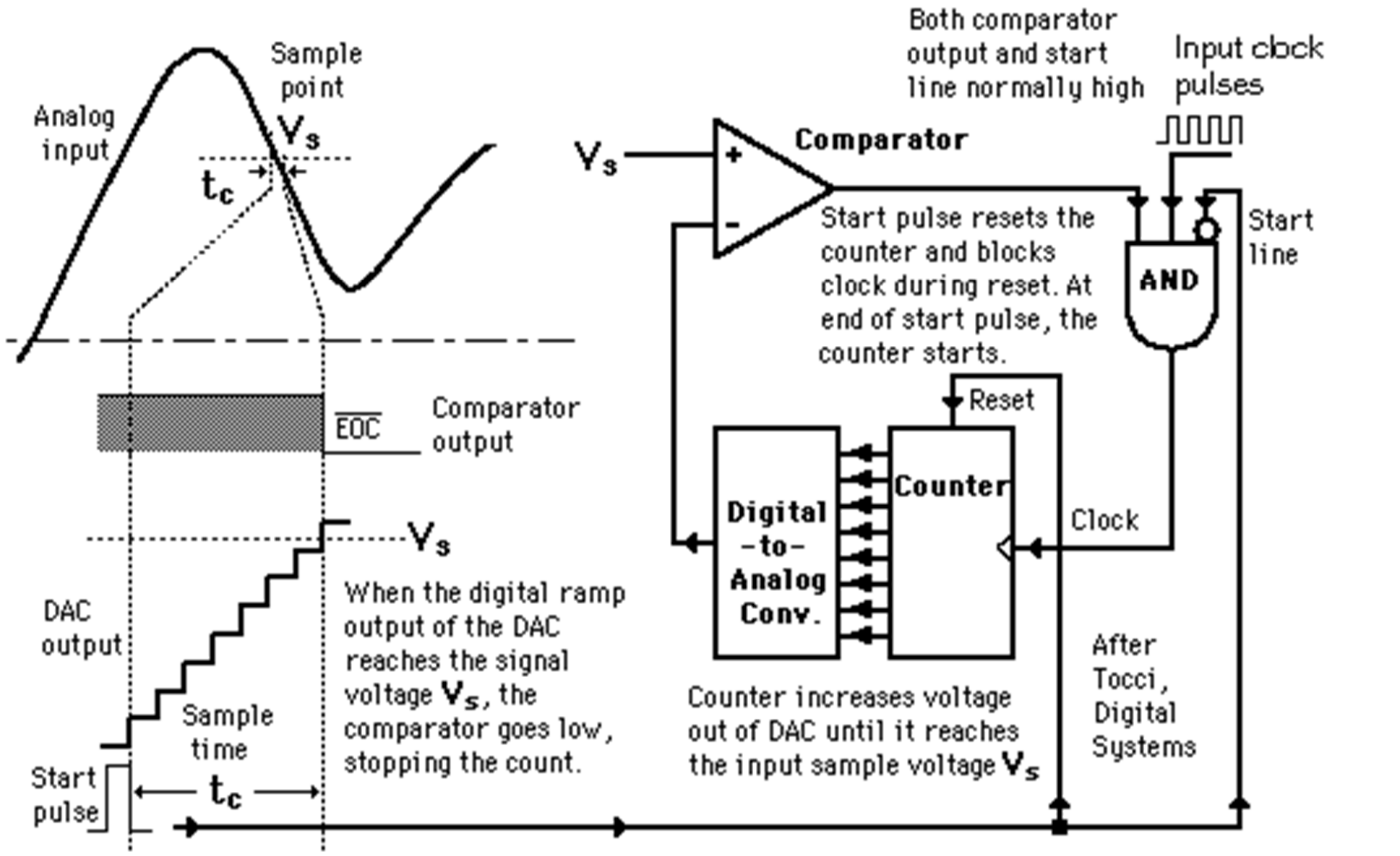


Generating the Clock Signal

- ADC0804 requires a clock source to operate.
- It can be an external clock applied to CLK IN pin or can be generated with an RC circuit.
  - permissible range of clock frequencies is 100 KHz - 1460 KHz.
  - desirable to use a frequency as close as possible to 1460 KHz so conversion time is minimized
  - here **$F_c = 1/(1.1 \times RC)$**
- If generated with an RC circuit, CLK IN and CLK R pins are connected to an RC circuit

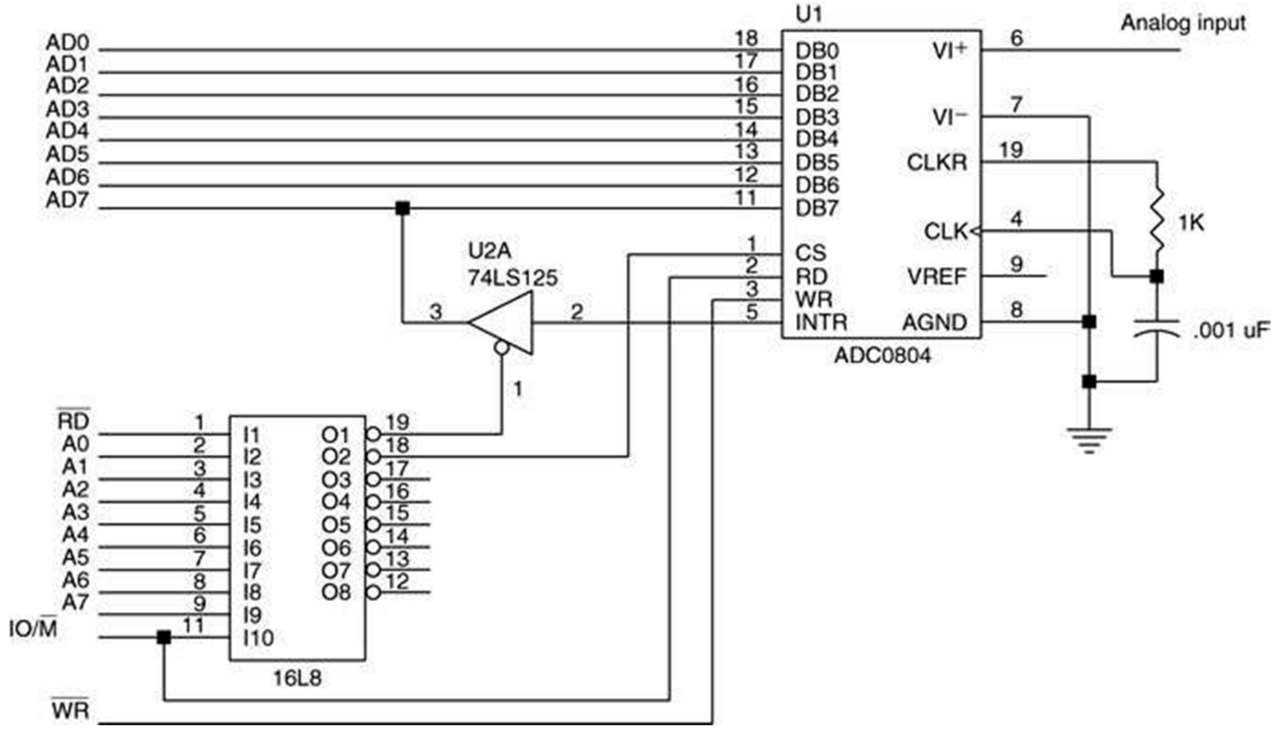


ADC from inside


Analog input
Sample point
V_s
t_c
$\overline{EOC}$
Comparator output
V_s
DAC output
Sample time
Start pulse
t_c
When the digital ramp output of the DAC reaches the signal voltage V_s, the comparator goes low, stopping the count.
Both comparator output and start line normally high
Input clock pulses
V_s
Comparator
+
-
Start pulse resets the counter and blocks clock during reset. At end of start pulse, the counter starts.
AND
Start line
Reset
Counter
Digital -to- Analog Conv.
Clock
Counter increases voltage out of DAC until it reaches the input sample voltage V_s
After Tocci, Digital Systems

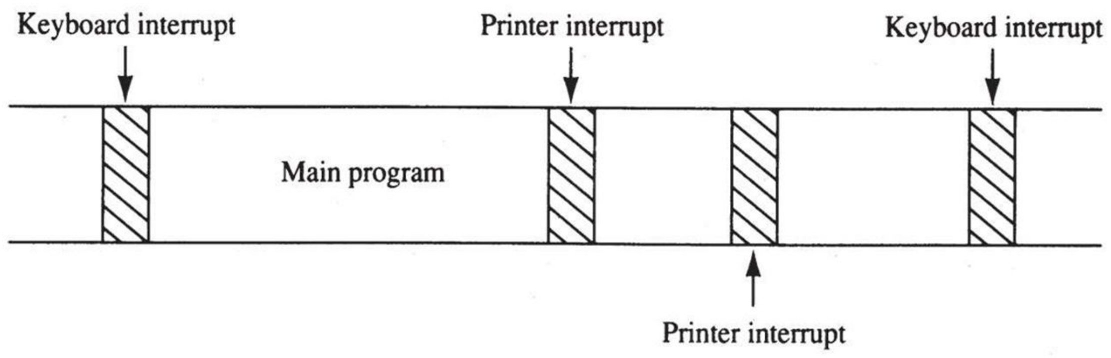
Connecting the ADC0804 to the Microprocessor

- ADC0804 interfaced to an 8086 is illustrated in Figure
- V_{REF} is not attached to anything, which is normal
- Suppose ADC0804 is decoded at I/O port address 40H for the data and address 42H for INTR.



The Purpose of Interrupts

- Interrupts are useful when interfacing I/O devices at relatively low data transfer rates, such as keyboard.
- Interrupt processing allows the processor to execute other software while the keyboard operator is thinking about what to type next.
- When a key is pressed, the keyboard puts out one pulse that interrupts the microprocessor.



- Diagram shows how the keyboard interrupt and the printer interrupt are called during a main program.

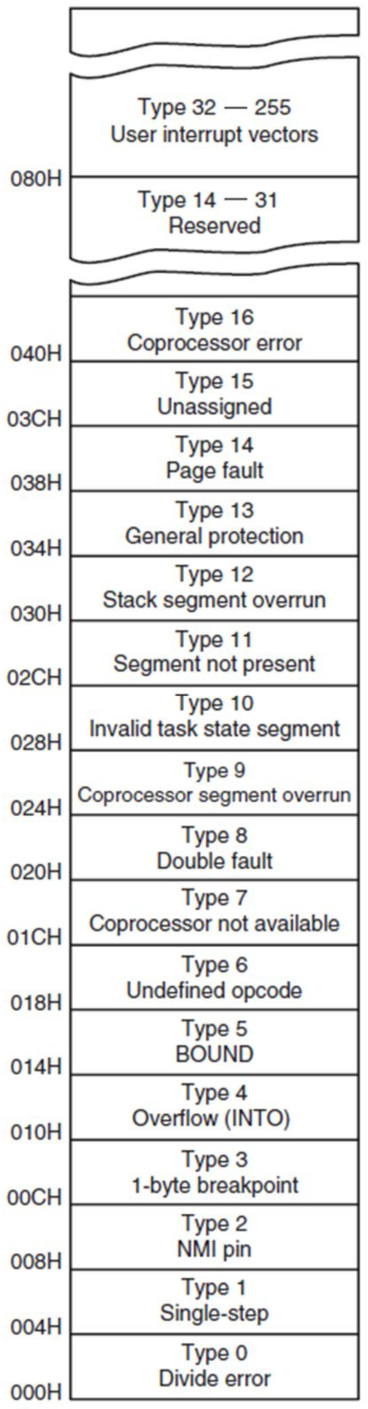
Interrupts

- Intel processors include two hardware pins (INTR and NMI) that request interrupts…
- And one hardware pin (INTA) to acknowledge the interrupt requested through INTR.
- The processor also has software interrupts INT, INTO, INT 3, and BOUND.
- Flag bits IF (interrupt flag) and TF (trap flag), are also used with the interrupt structure and special return instruction IRET
  - IRETD in the 80386, 80486, or Pentium

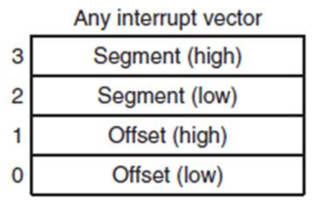
Interrupt Vectors

- Interrupt vectors and the vector table are crucial to an understanding of hardware
  and software interrupts.
- The **interrupt vector table** is located in
  the first 1024 bytes of memory at addresses 000000H–0003FFH.
  - It contains 256 different four-byte interrupt vectors
- An interrupt vector contains the address (segment and offset) of the interrupt service procedure.

The interrupt vector table for the microprocessor



- the first five interrupt vectors are identical in all Intel processors
- Intel reserves the first 32 interrupt vectors
- the last 224 vectors are user-available
- each is four bytes long in real mode and contains the starting address of the interrupt service procedure.
- the first two bytes contain the offset address
- the last two contain the segment address



Intel Dedicated Interrupts

- **Type 0**
  The **divide error** whenever the result from a division overflows or an attempt is made to divide by zero.
- **Type 1**
  Single-step or trap occurs after execution of each instruction if the trap (TF) flag bit is set.
- **Type 2**
  The non-maskable interrupt occurs when a logic 1 is placed on the NMI input pin to the microprocessor
- non-maskable means—it cannot be disabled

Intel Dedicated Interrupts

- **Type 3**
  A special one-byte instruction (INT 3) that uses this vector to access its interrupt-service procedure.
  - often used to store a breakpoint in a program for debugging
- **Type 4**
  Overflow is a special vector used with the INTO instruction. The INTO instruction interrupts the program if an overflow condition exists.
- As reflected by the overflow flag (OF)

Intel Dedicated Interrupts

- **Type 5**
  The **BOUND** instruction compares a register with boundaries stored in the memory.

  If the contents of the register are greater than or equal to the first word in memory and less than or equal to the second word, no interrupt occurs because the contents of the register are within bounds.
  - if the contents of the register are out of bounds,
    a type 5 interrupt ensues

BOUND, INTO, INT, INT 3, and IRET

These are 5 software instructions for interrupts.

- INT and INT 3 are very similar.
- BOUND and INTO are conditional.
- BOUND has two operands, and compares a register with two words of memory data.
- INTO checks or tests the overflow flag (O).
  - If O = 1, INTO calls the procedure whose address is stored in interrupt vector type 4
  - If O = 0, INTO performs no operation and the next sequential program instruction executes
- The INT n instruction calls the interrupt service procedure at the address represented in vector number n.

BOUND, INTO, INT, INT 3, and IRET

- INT 3 instruction (1-byte) is often used as a breakpoint-interrupt because it is easy to insert a one-byte instruction into a program.
  - breakpoints are often used to debug software
- The IRET instruction is a special return instruction used to return for both software and hardware interrupts.
  - much like a RET, it retrieves the return address from the stack

Operation of a Real Mode Interrupt

- When the processor completes executing the current instruction, it determines whether an interrupt is active by checking:
  - (1) instruction executions internal e.g., overflow, divide by 0
  - (2) single-step
  - (3) NMI
  - (4) coprocessor segment overrun
  - (5) INTR
  - (6) INT instructions in the order presented

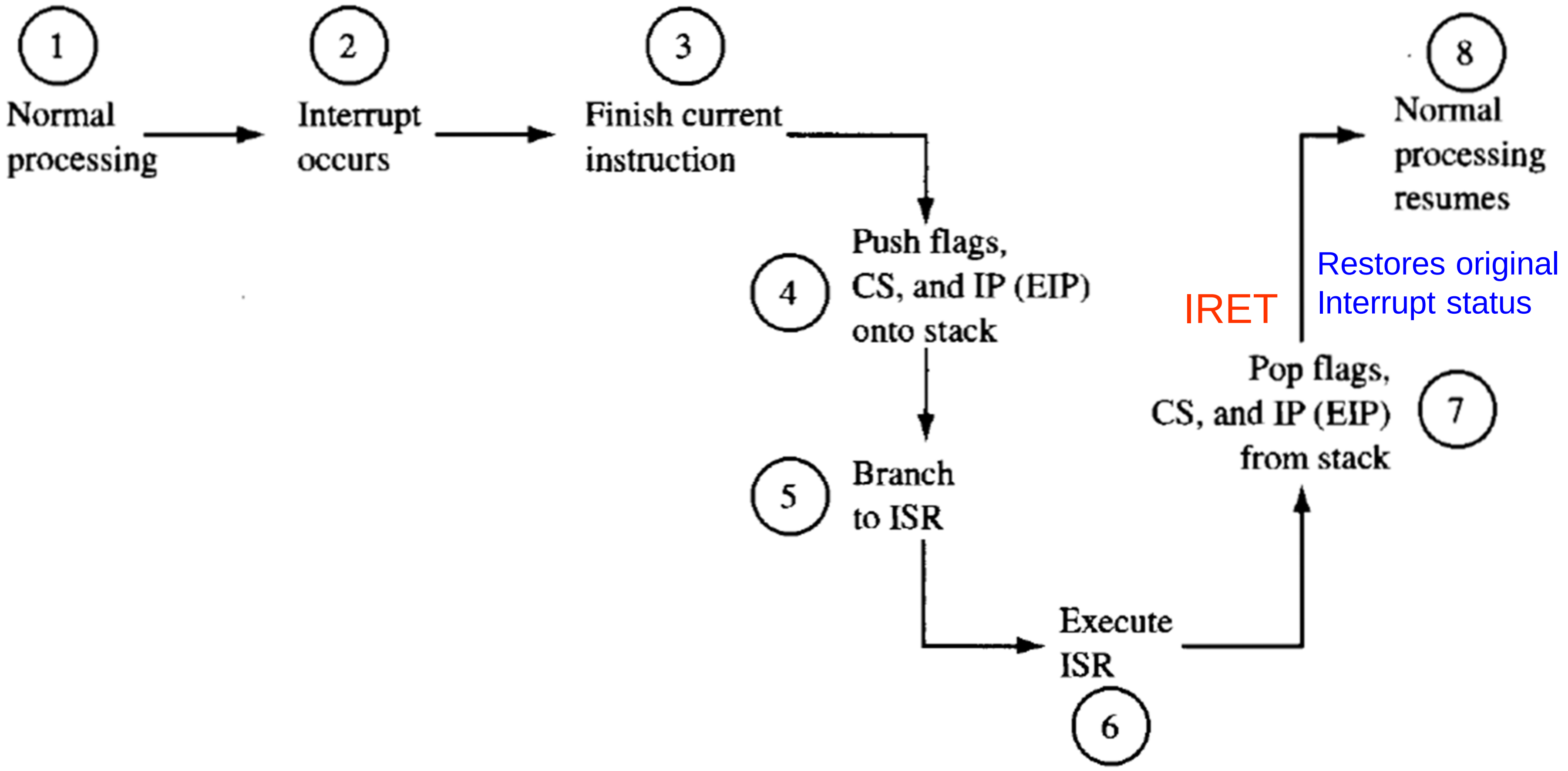
Operation of a Real Mode Interrupt

- If one or more are present:

  1. Flag register contents are pushed on the stack
  2. Interrupt (IF) & trap (TF) flags clear, disabling the INTR pin and trap or single-step feature
  3. Contents of the code segment register (CS) are pushed onto the stack
  4. Contents of the instruction pointer (IP) are pushed onto the stack
  5. Interrupt vector contents are fetched and placed into IP and CS so the next instruction executes at the interrupt service procedure addressed by the vector

Interrupt Processing Flow-Chart

Figure 9.1 When an interrupt occurs, normal processing is suspended while a special interrupt service routine (ISR) is executed. Normal processing resumes when this routine is completed.



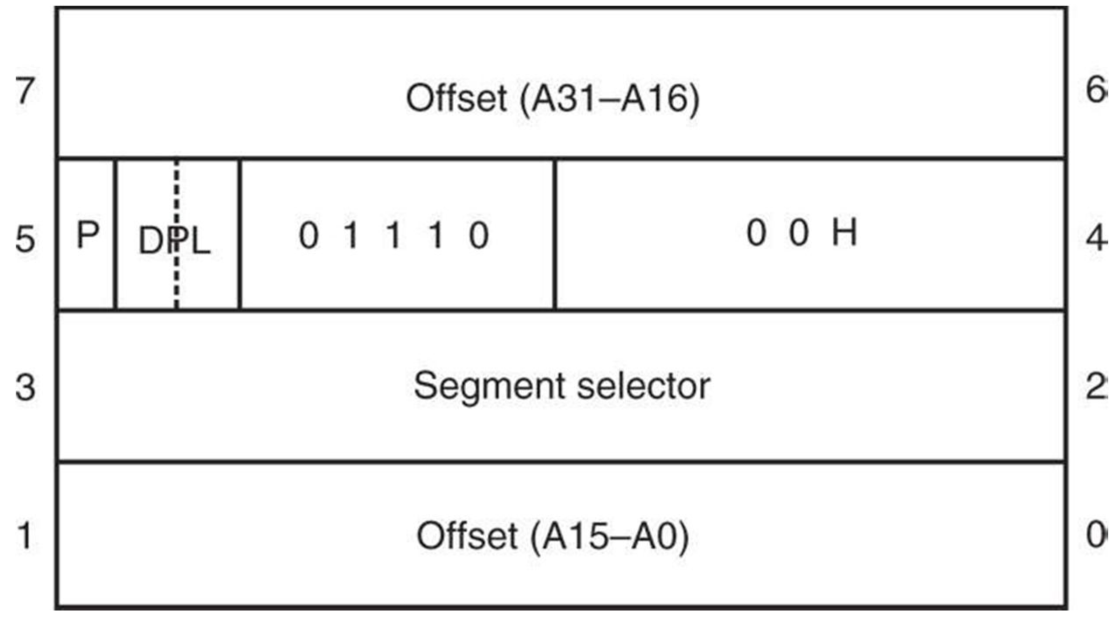
1 Normal processing
2 Interrupt occurs
3 Finish current instruction
4 Push flags, CS, and IP (EIP) onto stack
5 Branch to ISR
6 Execute ISR
7 Pop flags, CS, and IP (EIP) from stack
IRET
Restores original Interrupt status
8 Normal processing resumes

Operation of a Protected Mode Interrupt

- In protected mode, interrupts have the same assignments as real mode.
  - the interrupt vector table is different
- In place of interrupt vectors, protected mode uses a set of 256 interrupt descriptors stored in an interrupt descriptor table (IDT).
  - the table is 256 × 8 (2K) bytes long
  - each descriptor contains eight bytes
- The interrupt descriptor table is located at any memory location in the system by the interrupt descriptor table address register (IDTR).

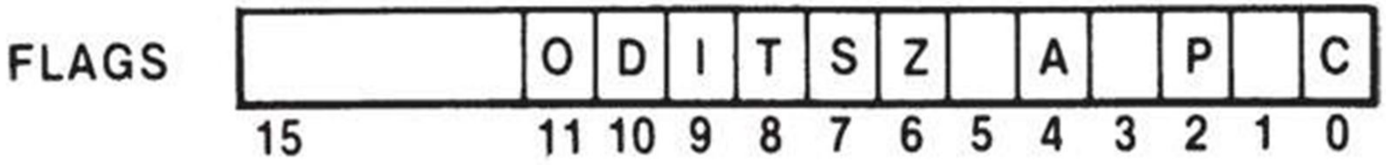
Operation of a Protected Mode Interrupt

- Each IDT entry contains the address of the interrupt service procedure
  - in the form of a segment selector and a 32-bit offset address
  - also contains the P bit (present) and DPL bits to describe the privilege level of the interrupt



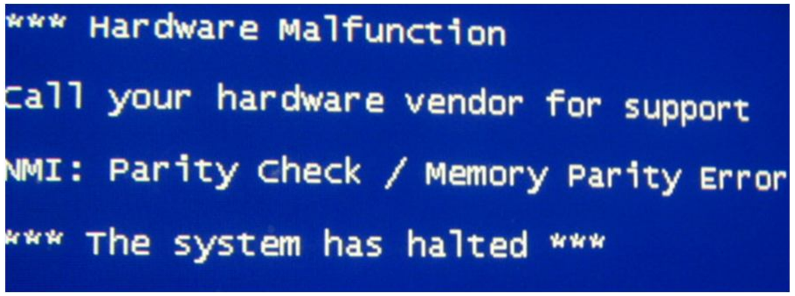
Interrupt Flag Bits

- The interrupt flag (IF) and the trap flag (TF) are both cleared after the contents of the flag register are stacked during an interrupt.
- the contents of the flag register and the location of IF and TF are shown here
  - when IF is set, it *allows* the INTR pin to cause an interrupt
  - when IF is cleared, it *prevents* the INTR pin from causing an interrupt
  - when TF = 1, it causes a trap interrupt (type 1) to occur after each instruction executes
  - Trap is often called a single-step
  - when TF = 0, normal program execution occurs



Hardware Interrupts

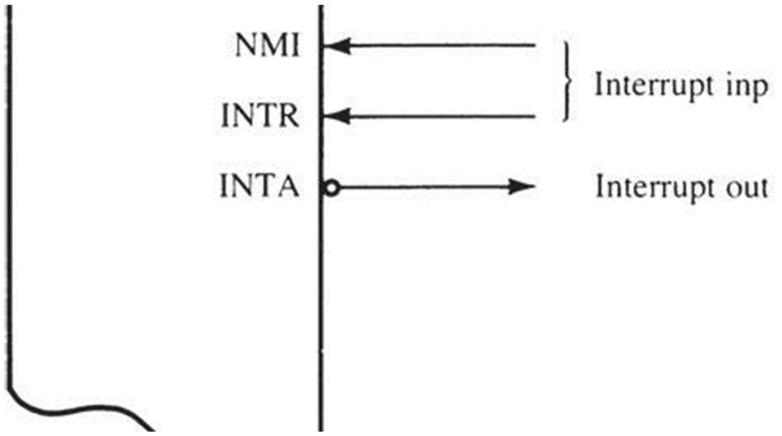
- The two processor hardware interrupt inputs:
  - non-maskable interrupt (NMI)
  - interrupt request (INTR)
- When NMI input is activated, a type 2 interrupt occurs
  - because NMI is internally decoded
- The **non-maskable interrupt** (NMI) is an edge-triggered input that requests an interrupt on the positive edge (0-to-1 transition).
- The NMI input is often used for parity errors and other major faults, such as power failures.



```
*** Hardware Malfunction

Call your hardware vendor for support

NMI: Parity Check / Memory Parity Error

*** The system has halted ***
```

Hardware Interrupts

- 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.
- Intel has reserved interrupts 00H - 1FH for internal and future expansion.
- INTA is also an interrupt pin on the processor.
  - it is an output used in response to INTR input to apply a vector type number to the data bus connections D_7–D_0

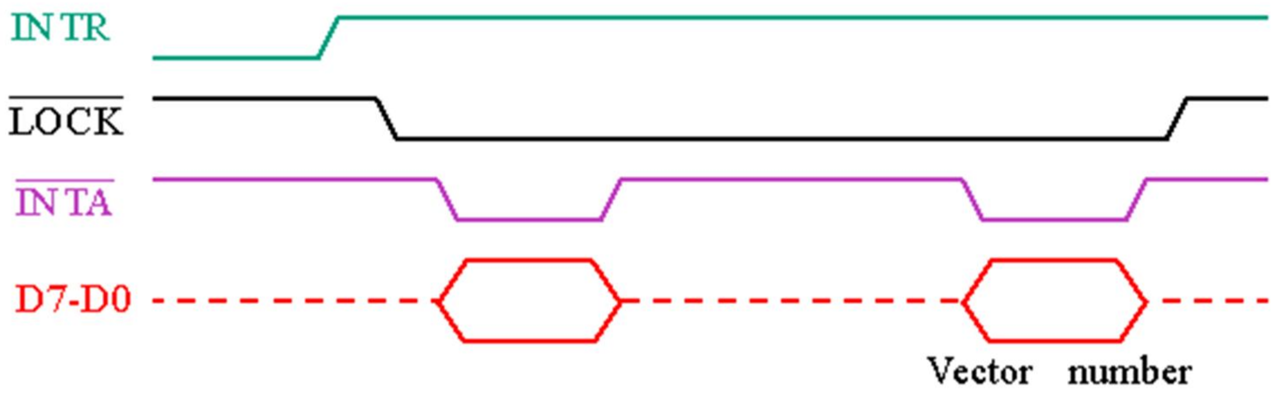


Hardware Interrupts

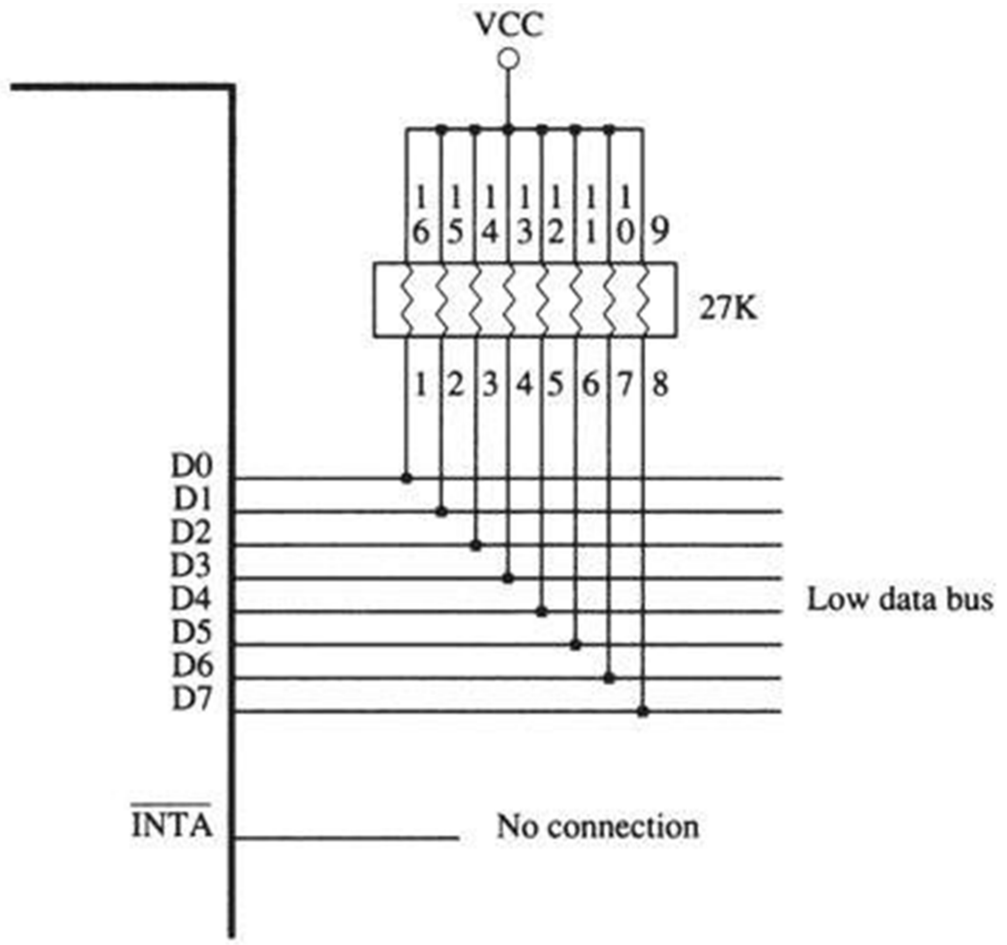
- The interrupt request input (INTR) is level-sensitive, which means that it must be held at a logic 1 level until it is recognized.
  - INTR is set by an external event and cleared inside the interrupt service procedure
- INTR is automatically disabled once accepted.
  - re-enabled by IRET at the end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

Hardware Interrupts

- The processor responds to INTR by pulsing INTA output in anticipation of receiving an interrupt vector type number on data bus connections D_7–D_0.
- Two INTA pulses generated by the system insert the vector type number on the data bus.

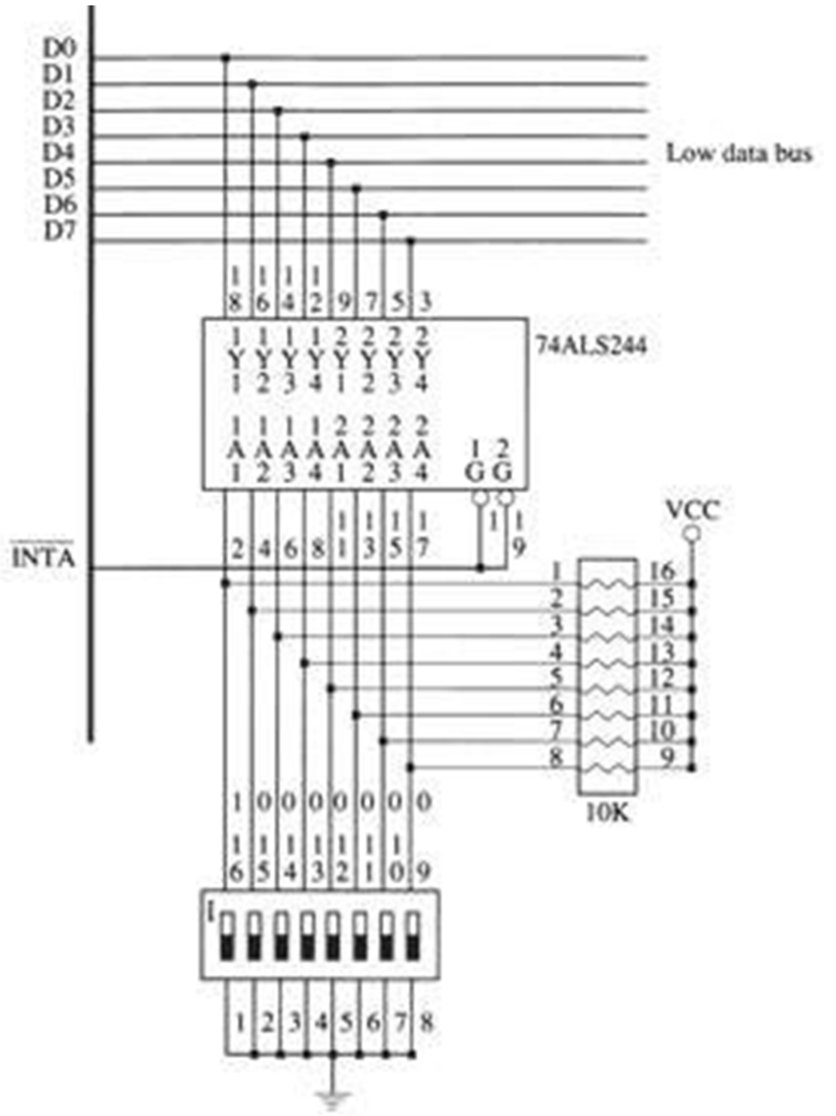


A simple method for generating interrupt vector type number FFH in response to INTR.



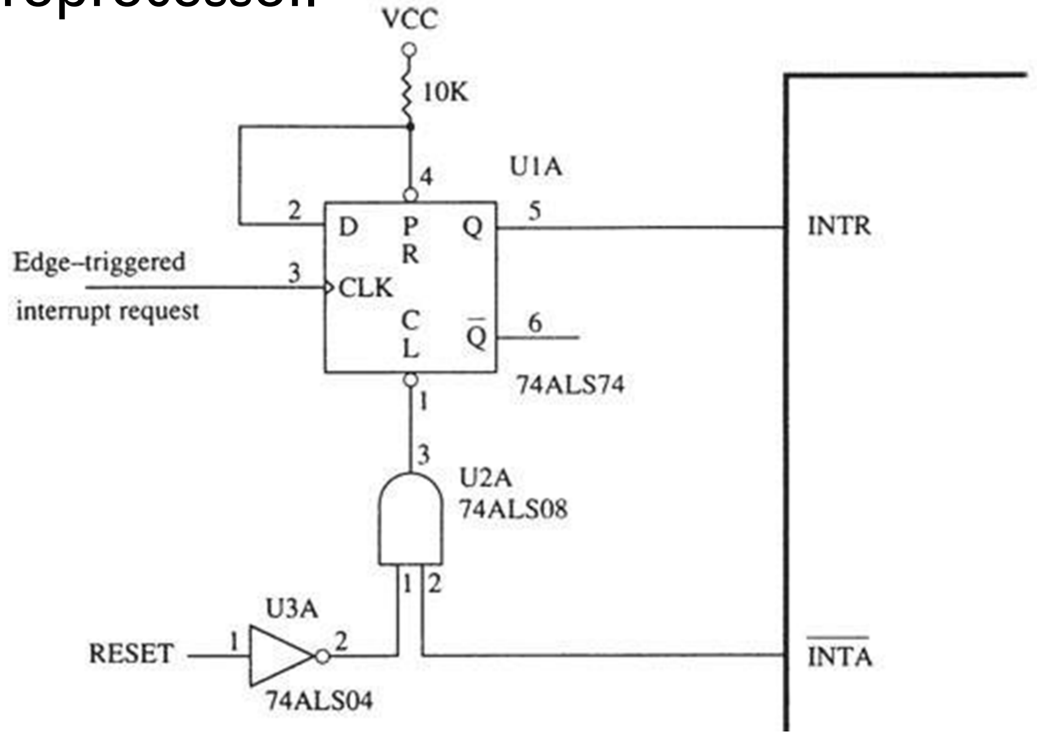
Using a Three-State Buffer for INTA

- In response to INTR, the processor outputs the INTA to enable a 74ALS244 three-state octal buffer.
- The octal buffer applies the interrupt vector type number to the data bus in response.
- The vector type number is easily changed with DIP switches shown in this illustration.



Making INTR Input Edge-Triggered

- INTR input can be converted to an edge-triggered input by using a D-type flip-flop.
- Clock input becomes an edge-triggered interrupt request input, and the clear input is used to clear the request when the INTA signal is output by the microprocessor.



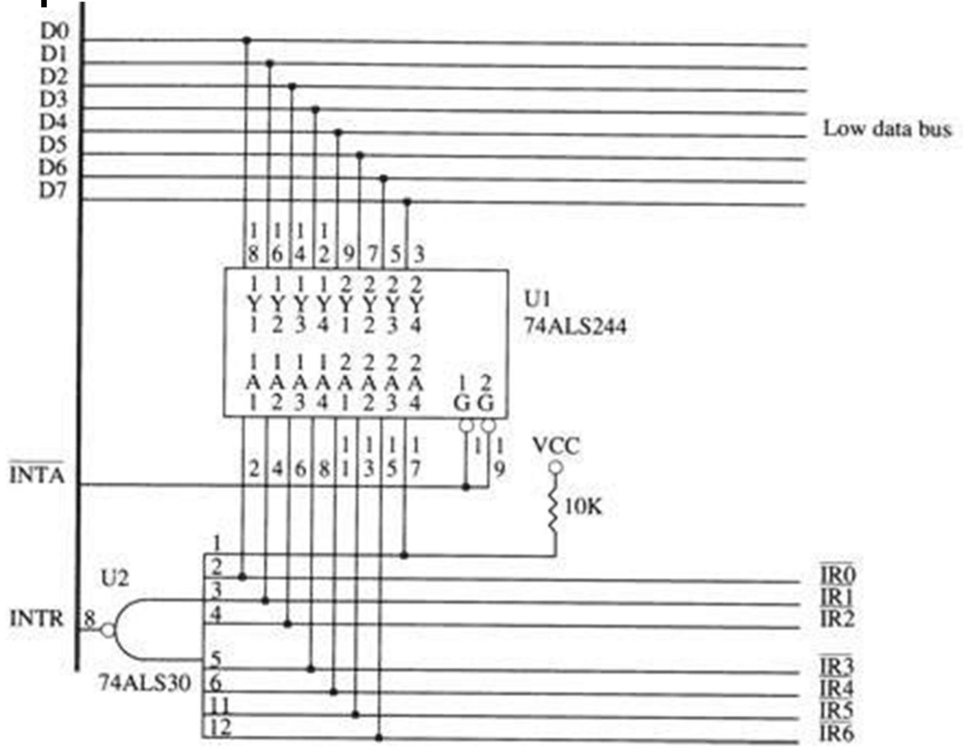
Expanding The Interrupt Structure

- This covers three common methods of expanding the interrupt structure of the processor.

1. Using 74LS244
2. Using “daisy-chain” method
3. Using 8259 Interrupt controller

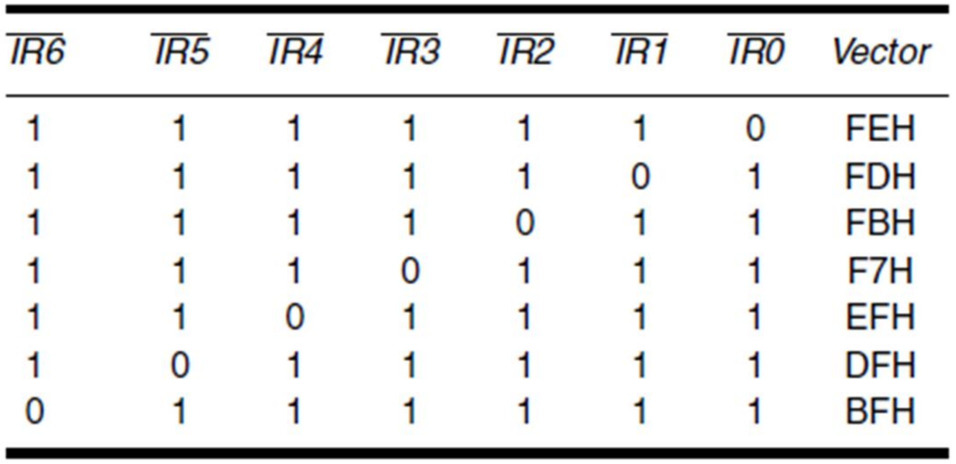
Using the 74ALS244 to Expand Interrupts

- The modification shown in Figure allows the circuit to accommodate up to seven additional interrupt inputs.
- The only hardware change is the addition of an eight-input NAND gate, which provides the INTR signal to the microprocessor when any of the IR inputs becomes active.



Using the 74ALS244 to Expand Interrupts

- If any of the IR inputs becomes logic 0, the output of the NAND gate goes to logic 1 and requests an interrupt through the INTR input.
- The interrupt vector that is fetched during the pulse depends on which interrupt request line becomes active.
  - Table shows the interrupt vectors used by a single interrupt request input
- If two or more interrupt requests are active, a new interrupt vector is generated. On that vector, we place the ISR of that interrupt which has higher priority.

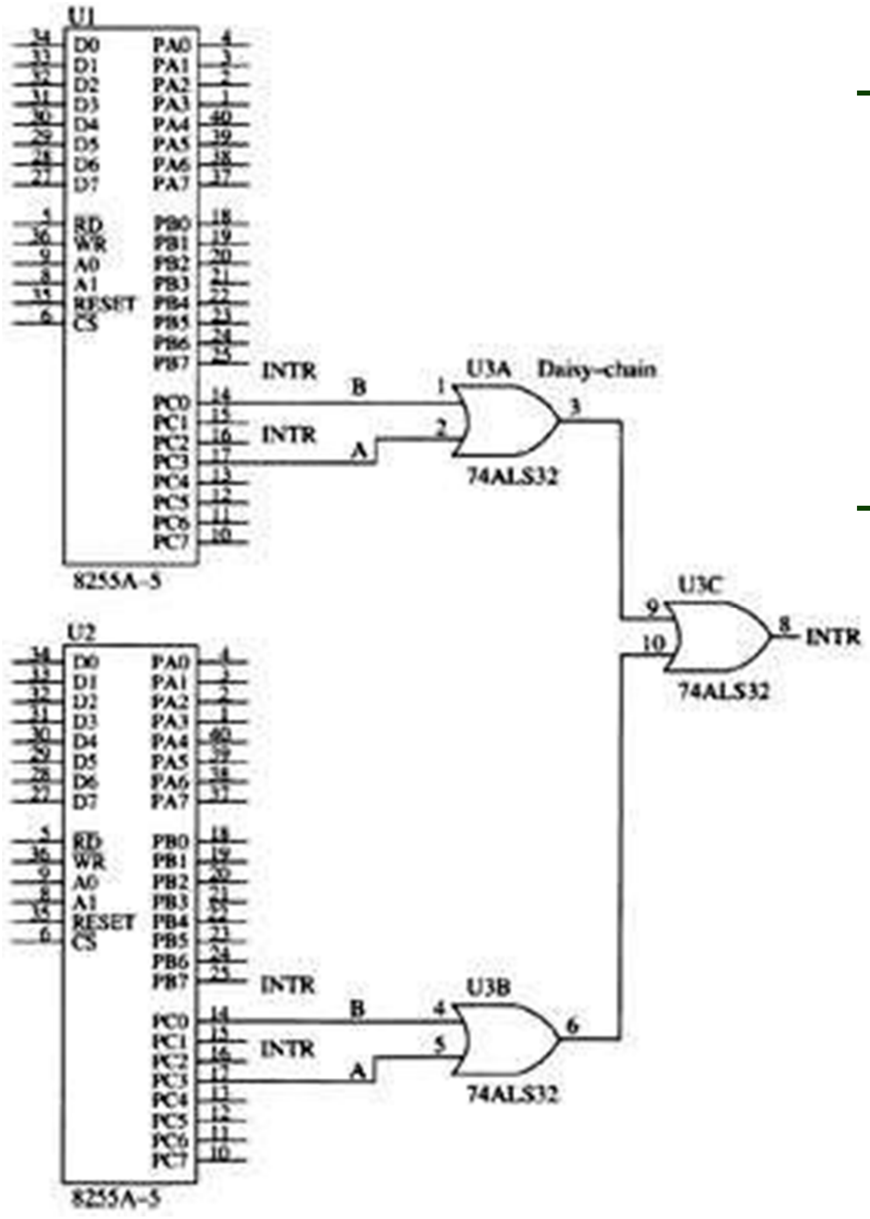


$\overline{IR6}$	$\overline{IR5}$	$\overline{IR4}$	$\overline{IR3}$	$\overline{IR2}$	$\overline{IR1}$	$\overline{IR0}$	Vector
1	1	1	1	1	1	0	FEH
1	1	1	1	1	0	1	FDH
1	1	1	1	0	1	1	FBH
1	1	1	0	1	1	1	F7H
1	1	0	1	1	1	1	EFH
1	0	1	1	1	1	1	DFH
0	1	1	1	1	1	1	BFH

Daisy-Chained Interrupt

- Expansion by a daisy-chained interrupt is in many ways better than using the 74ALS244.
  - because it requires only one interrupt vector
- Figure shows a two 82C55 peripheral interfaces with their four INTR outputs daisy-chained and connected to the single INTR input of the processor.
- If any interrupt output becomes logic 1, so does INTR input, causing an interrupt.

Two 82C55 connected to the INTR outputs (daisy-chained)

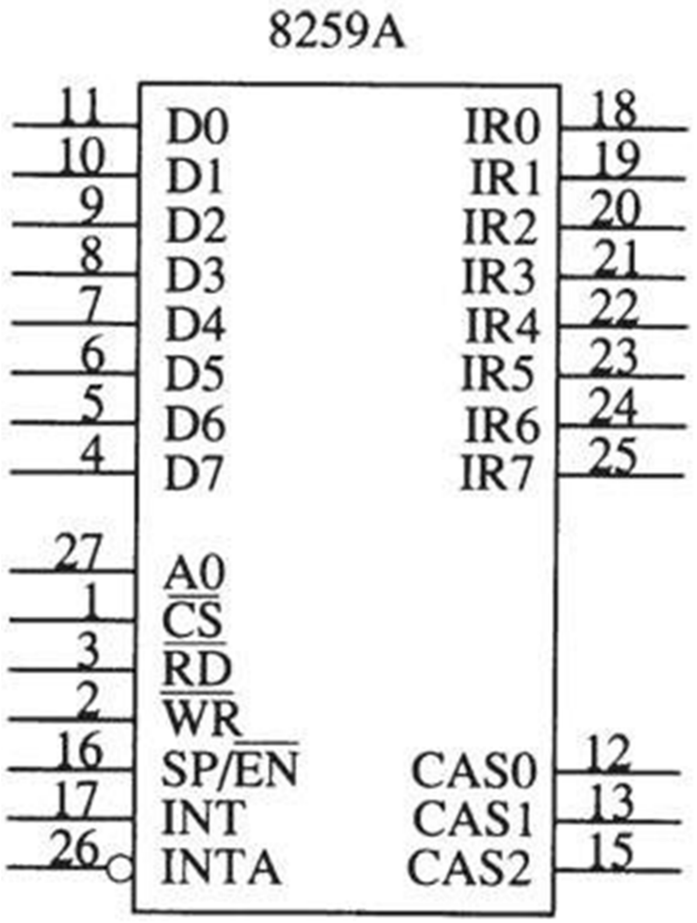


- any INTR output from the two 82C55s will cause the INTR pin on the processor to go high, requesting an interrupt
- The task of locating which INTR output became active is up to the interrupt service procedure, which must poll the 82C55s to determine which output caused the interrupt

8259A Programmable Interrupt Controller

- 8259A (PIC) adds eight vectored priority encoded interrupts to the microprocessor.
- Expandable, without additional hardware,
  to accept up to 64 interrupt requests.
  - requires a master 8259A & eight 8259A slaves
- A pair of these controllers still resides and is programmed in the latest chip sets from Intel and other manufacturers.

General Description of the 8259A



– 8259A is easy to connect to the microprocessor
– all of its pins are direct connections except the $\overline{CS}$ pin, which must be decoded, and the $\overline{WR}$ pin, which must have an I/O bank write pulse

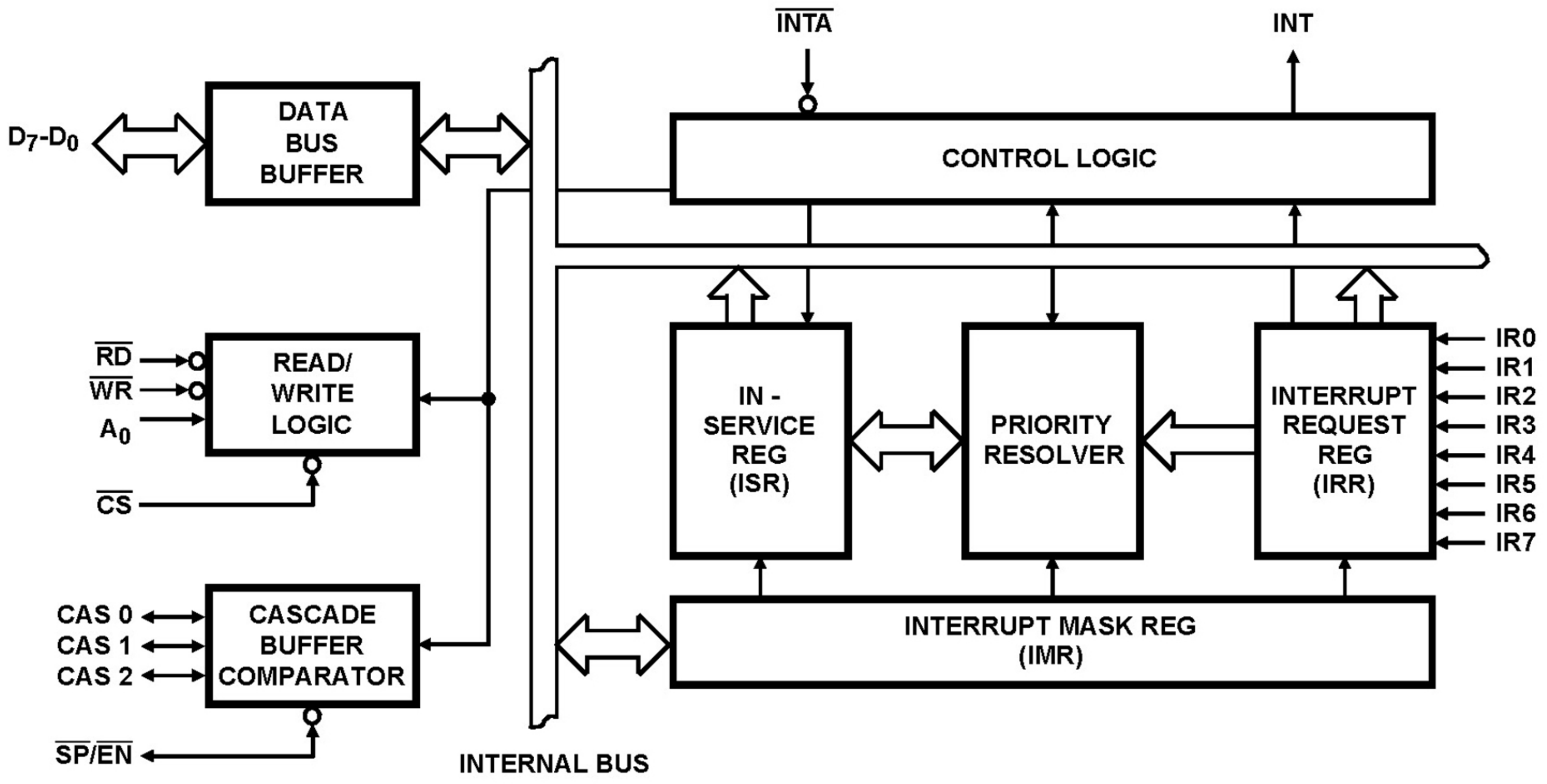
8259A Pin-Outs

- **IR_0–IR_7**
- **Interrupt request inputs** are used to request an interrupt and to connect to a slave in a system with multiple 8259As.
- **INTR**
- The **interrupt output** connects to the INTR pin on the processor from the master and is connected to a master IR pin on a slave.
- **INTA**
- Interrupt acknowledge is an input that connects to the INTA signal on the system.
  In a system with a master and slaves, only the master INTA signal is connected.

8259A Pin-Outs

- **CAS0–CAS2**
- The cascade lines are used as outputs from the master to the slaves for cascading multiple 8259As in a system.
- **$SP/\overline{EN}$**
- Slave program/enable buffer is a dual-function pin.
- when the 8259A is in buffered mode, this output controls the data bus transceivers in a large microprocessor-based system
- when the 8259A is not in the buffered mode, this pin programs the device as a master (1) or a slave (0)

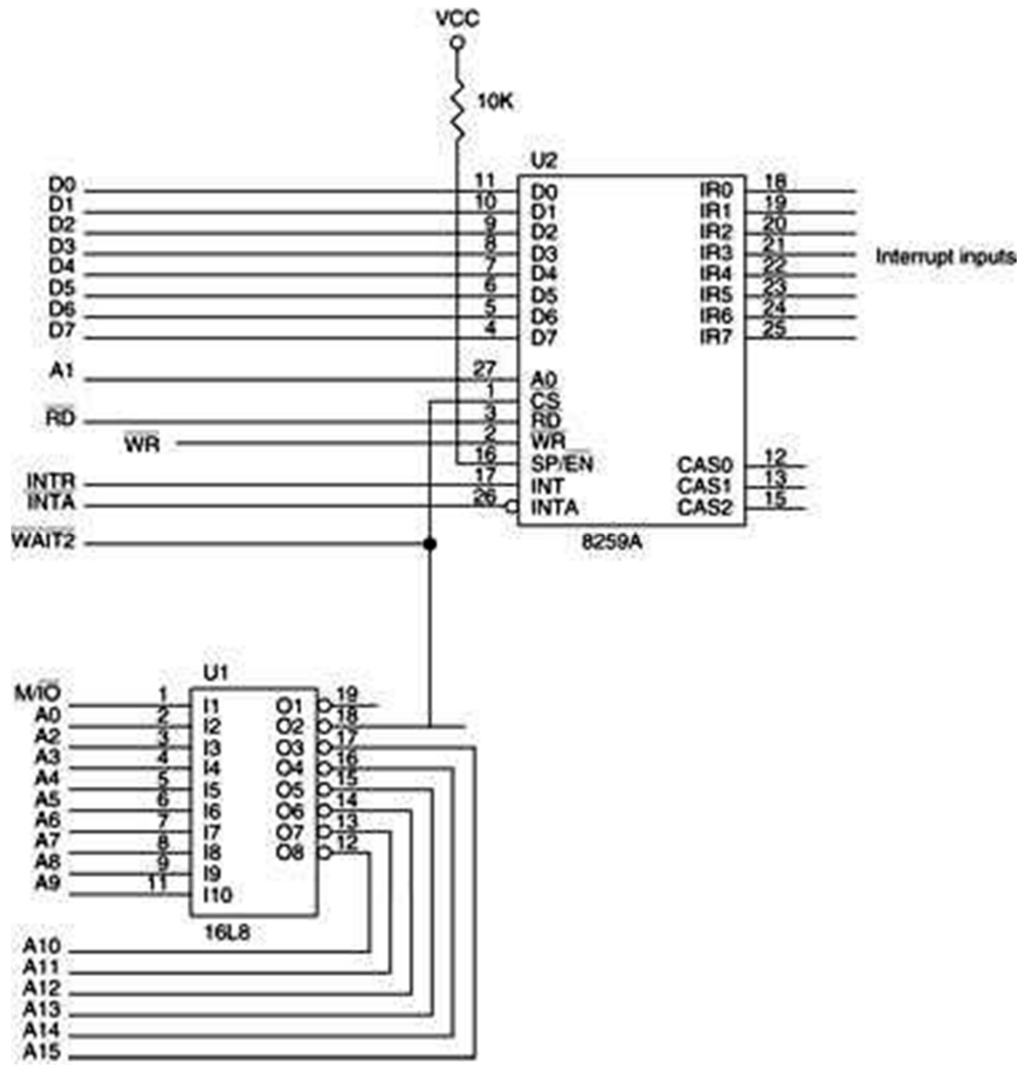
8259A internal Block Diagram



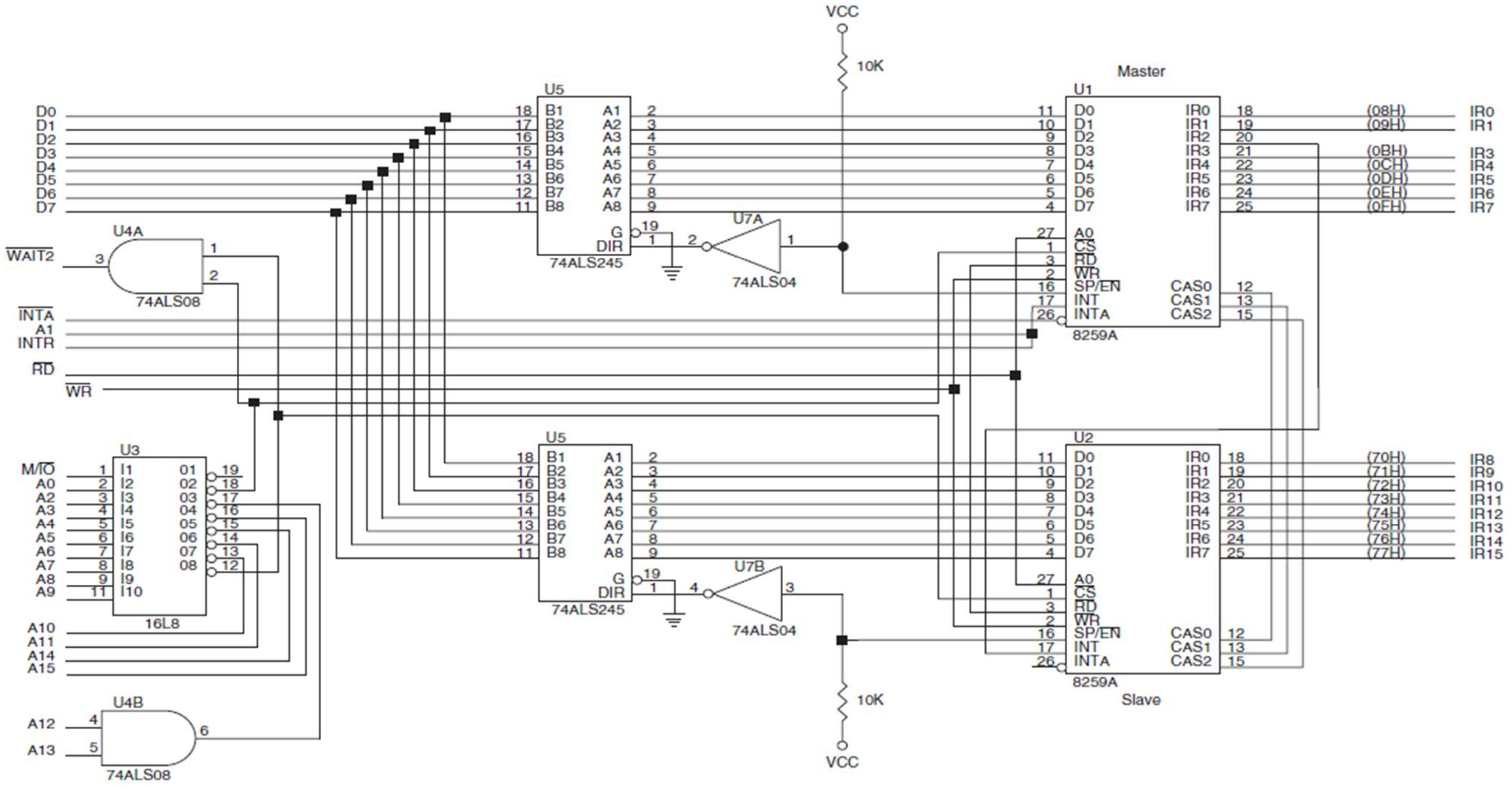
Connecting a Single 8259A

- The 8259A is decoded at I/O ports 0400H and 0401H by the PLD.

- The 8259A requires four wait states for it to function properly with a 16 MHz processor
  - more for some other versions of the Intel microprocessor family



Cascading Multiple 8259As



Cascading Multiple 8259As

- This circuit uses vectors 08H–0FH & I/O ports 0300H & 0302H for U1, the master
- Uses vectors 70H–77H & I/O ports 0304H & 0306H for U2, the slave
- Data bus buffers illustrate the use of the SP/EN pin on 8259A
- These buffers are used only in very large systems with many devices on their data bus connections