Microprocessors & Interfacing

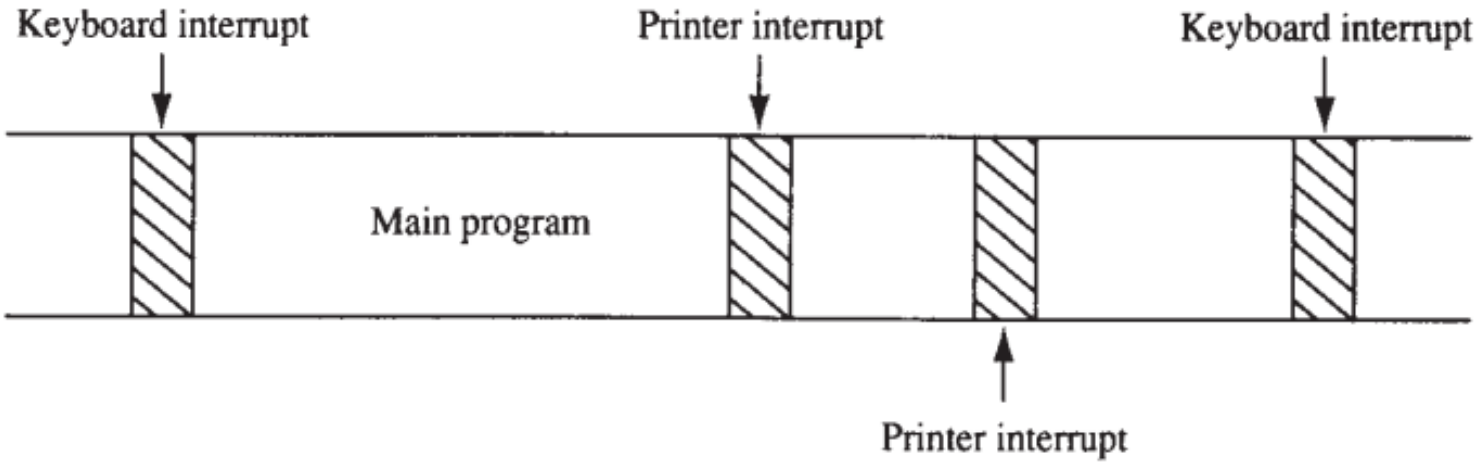
INTERRUPTS

BITS Pilani

Dr. Gargi Prabhu
Department of CS & IS

Basic Interrupt Processing

- Interrupts are particularly useful when interfacing I/O devices that provide or require data at relatively low data transfer rate.



Interrupts

- The interrupts of the entire Intel family of microprocessors include two hardware pins that request interrupts
  - INTR and NMI
- One hardware pin (INTA') that acknowledges the interrupt requested through
- Software Interrupts: INT, INTO, INT 3, and BOUND
- Two flag bits: IF (interrupt flag) and TF (trap flag), are also used with the interrupt structure
- Special return instruction, IRET

Interrupt Vectors

- The interrupt vectors and 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.

Any interrupt vector

3	Segment (high)
2	Segment (low)
1	Offset (high)
0	Offset (low)

Types of Interrupts

innovate achieve lead

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 the execution of each instruction if the trap (TF) flag bit is set. Upon accepting this interrupt, the TF bit is cleared so that the interrupt service procedure executes at full speed.

TYPE 2 The non-maskable interrupt occurs when a logic 1 is placed on the NMI input pin to the microprocessor. This input is non-maskable, which means that it cannot be disabled.

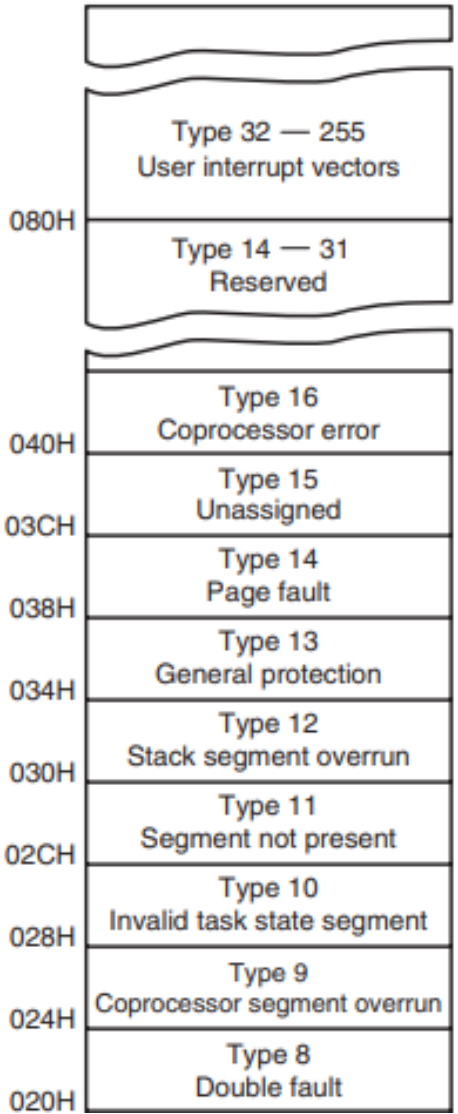
Types of Interrupts

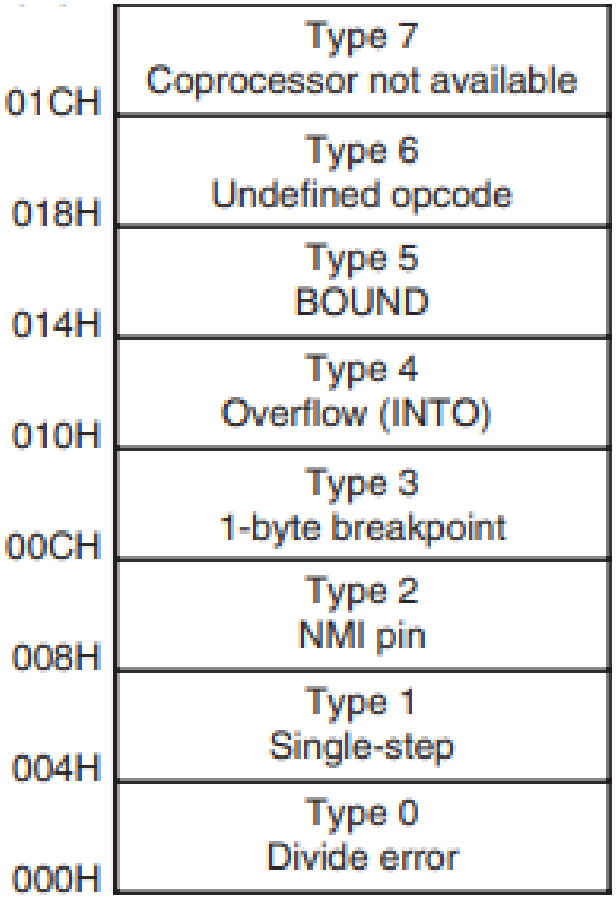
TYPE 3 A special one-byte instruction (INT 3) that uses this vector to access its interrupt service procedure. The INT 3 instruction is often used to store a breakpoint in a program for debugging.

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).

TYPE 9 The coprocessor segment overrun occurs if the ESC instruction (coprocessor opcode) memory operand extends beyond offset address FFFFH in real mode.

Types of Interrupts





Interrupt Instructions

- INT and INT 3 are very similar
- BOUND and INTO are conditional, and IRET is a special interrupt return instruction
- The BOUND instruction, which has two operands, compares a register with two words of memory data.

E.g.

BOUND AX,DATA

Interrupt Instructions

- The BOUND instruction, which has two operands, compares a register with two words of memory data.

E.g.

BOUND AX,DATA

- AX is compared with the contents of DATA and DATA+1 and also with DATA+2 and DATA+3.
- If AX is less than the contents of DATA and DATA+1, a type 5 interrupt occurs.
- If AX is greater than DATA+2 and DATA+3, a type 5 interrupt occurs.
- If AX is within the bounds of these two memory words, no interrupt occurs

Interrupt Instructions

- The INTO instruction checks or tests the overflow flag (O).
- If O = 1, the INTO instruction calls the procedure whose address is stored in interrupt vector type number 4.
- If O = 0, then the INTO instruction performs no operation and the next sequential instruction in the program executes.

Interrupt Instructions

- The INT n instruction calls the interrupt service procedure that begins at the address represented in vector number n.

  E.g. INT 80H or INT 128 calls the interrupt service procedure whose address is stored in vector type number 80H (000200H–00203H).

- To determine the vector address, just multiply the vector type number (n) by 4, which gives the beginning address of the four-byte long interrupt vector.

Interrupt Instructions

- For example, INT 5 = 4 × 5 or 20 (14H).

- The vector for INT 5 begins at address 0014H and continues to 0017H.
- Each INT instruction is stored in two bytes of memory: The first byte contains the opcode, and the second byte contains the interrupt type number.
- The only exception to this is the INT 3 instruction, a one-byte instruction. The INT 3 instruction 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 faulty software.

Interrupt Instructions

- The IRET instruction is a special return instruction used to return for both software and hardware interrupts.
- The IRET instruction is much like a far RET, because it retrieves the return address from the stack.
- It is unlike the near return because it also retrieves a copy of the flag register from the stack.
- An IRET instruction removes six bytes from the stack: two for the IP, two for the CS, and two for the flags.

Interrupt Instructions

innovate achieve lead

- In the 80386–Core2, there is also an IRETD instruction because these microprocessors can push the EFLAG register (32 bits) on the stack, as well as the 32-bit EIP in the protected mode and 16-bit code segment register.
- If operated in the real mode, we use the IRET instruction with the 80386–Core2 microprocessors.
- If the Pentium 4 operates in 64-bit mode, an IRETQ instruction is used to return from an interrupt.
- The IRETQ instruction pops the EFLAG register into RFLAGS and also the 64-bit return address is placed into the RIP register

Real Mode Vs Protected Mode

innovate achieve lead

Real Mode:

- Real Mode is the mode in which the x86 microprocessor starts up and initializes.
- It provides a compatibility mode for running software written for older x86 processors.
- In Real Mode, the processor can directly access only 1 MB of memory (20-bit address space).
- It lacks memory protection, which means that any program can access any part of memory, including the operating system's memory.
- Interrupts and exceptions are handled using a simple interrupt vector table.
- All memory addresses are interpreted as physical addresses.

Real Mode Vs Protected Mode

Protected Mode:

- Protected Mode is a more advanced operating mode that provides memory protection and virtual memory.
- It allows for multitasking and running multiple programs concurrently.
- Protected Mode provides a 32-bit address space, allowing access to up to 4 GB of memory (using segmentation).
- Memory protection mechanisms prevent programs from accessing memory outside their allocated regions.
- It supports multitasking by using a system of privilege levels (rings) to control access to system resources.
- Interrupt handling is more sophisticated, with separate interrupt descriptor tables for tasks and exceptions.
- Virtual memory support enables paging and allows for more efficient memory management.

Operation of a real mode interrupt

- When the microprocessor completes executing the current instruction, it determines whether an interrupt is active by checking

(1) instruction executions

(2) single-step or trap

(3) NMI

(4) coprocessor segment overrun

(5) INTR

(6) INT instructions

Operation of a real mode interrupt

- If one or more of these interrupt conditions are present, the following sequence of events occurs:

1. The contents of the flag register are pushed onto the stack.

2. Both the interrupt (IF) and trap (TF) flags are cleared. This disables the INTR pin and the trap or single-step feature.

3. The contents of the code segment register (CS) are pushed onto the stack

4. The contents of the instruction pointer (IP) are pushed onto the stack.

5. The interrupt vector contents are fetched, and then placed into both IP and CS so that the next instruction executes at the interrupt service procedure addressed by the vector.

Operation of a real mode interrupt

innovate achieve lead

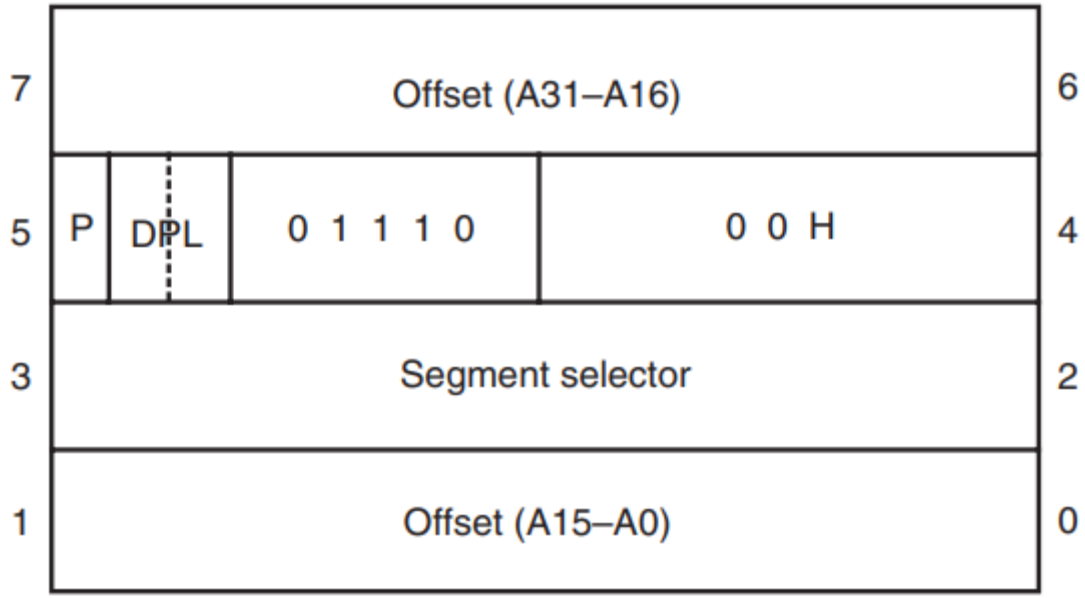
- Whenever an interrupt is accepted, the microprocessor stacks the contents of the flag register, CS and IP; clears both IF and TF; and jumps to the procedure addressed by the interrupt vector.
- After the flags are pushed onto the stack, IF and TF are cleared.
- These flags are returned to the state prior to the interrupt when the IRET instruction is encountered at the end of the interrupt service procedure.
- Therefore, if interrupts were enabled prior to the interrupt service procedure, they are automatically re-enabled by the IRET instruction at the end of the procedure.
- The return address (in CS and IP) is pushed onto the stack during the interrupt.

Operations of a Protected Mode Interrupt

- The interrupt vector table is different.
- In place of interrupt vectors, protected mode uses a set of 256 **interrupt descriptors** that are stored in an interrupt descriptor table (IDT).
- The interrupt descriptor table is 256 × 8 (2K) bytes long, with each descriptor containing eight bytes.
- The interrupt descriptor table is located at any memory location in the system by the interrupt descriptor table address register (IDTR).

Operations of a Protected Mode Interrupt

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



Operations of a Protected Mode Interrupt

- Real mode interrupt vectors can be converted into protected mode interrupts by copying the interrupt procedure addresses from the interrupt vector table and converting them to 32-bit offset addresses that are stored in the interrupt descriptors.
- A single selector and segment descriptor can be placed in the global descriptor table that identifies the first 1M byte of memory as the interrupt segment.
- Other than the IDT and interrupt descriptors, the protected mode interrupt functions like the real mode interrupt.
- We return from both interrupts by using the IRET or IRETD instruction.

Operations of a Protected Mode Interrupt

- The only difference is that in protected mode the microprocessor accesses the IDT instead of the interrupt vector table.
- In the 64-bit mode of the Pentium 4 and Core2, an IRETQ must be used to return from an interrupt.
- This is one reason why there are different drivers and operating systems for the 64-bit mode

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

- 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 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
```

innovate achieve lead

BITS Pilani

Pilani Campus

Thank You