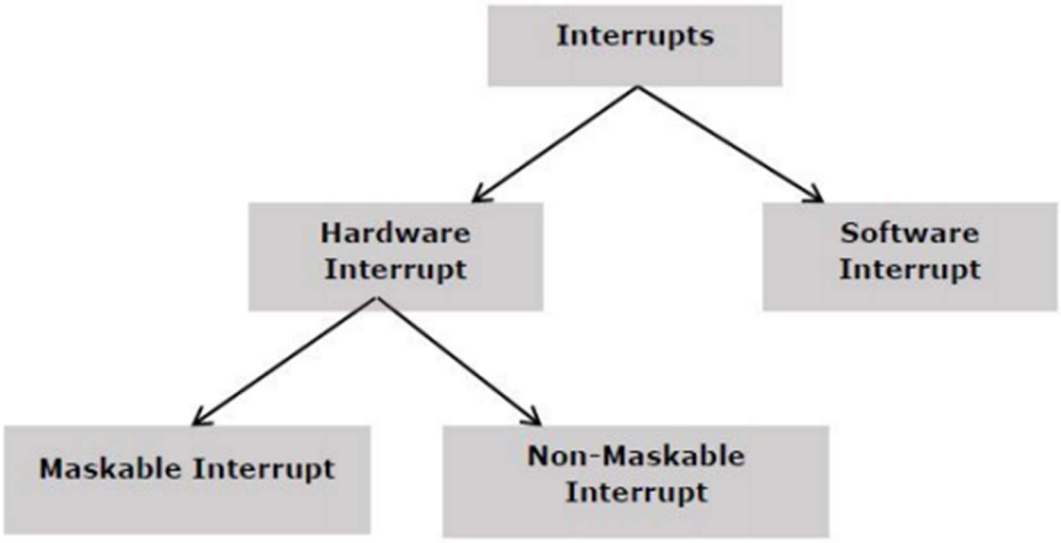
Timing diagrams and Interrupts

2 marks

1. **Which are the different types of interrupt supported by 8086?**



Hardware interrupts

Hardware interrupt is caused by any peripheral device by sending a signal through a specified pin to the microprocessor.
The 8086 has two hardware interrupt pins, i.e. **NMI** and **INTR**.
NMI is a non-maskable interrupt and INTR is a maskable interrupt having lower priority. One more interrupt pin associated is INTA called interrupt acknowledge.

Software interrupts

Some instructions are inserted at the desired position into the program to create interrupts. These interrupt instructions can be used to test the working of various interrupt handlers. It includes −

INT n (n is interrupt no.)

2. What is meant by interrupt?

Interrupt is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor. The microprocessor responds to that interrupt with an ISR (Interrupt Service Routine), which is a short program to instruct the microprocessor on how to handle the interrupt.

When an interrupt signal is received, MP sends an acknowledgement (INTA) to peripheral which is requesting for its service.

3. What is meant by polling?

Polling is the process where the computer or controlling device waits for an external device to check for its readiness or state, often with low-level hardware. For example, when a printer is connected via a parallel port, the computer waits until the printer has received the next character.

These processes can be as minute as only reading one bit. This is sometimes used synonymously with 'busy-wait' polling. In this situation, when an I/O operation is required, the computer does nothing other than check the status of the I/O device until it is ready, at which point the device is accessed. In other words, the computer waits until the device is ready.

Polling also refers to the situation where a device is repeatedly checked for readiness, and if it is not, the computer returns to a different task. Although not as wasteful of CPU cycles as busy waiting, this is generally not as efficient as the alternative to polling, interrupt-driven I/O.

4 marks

1. Explain interrupt structure of 8086.

Interrupt Structure of 8086:

An Interrupt Structure of 8086 can come from any one the three sources :

- **External signal**
- **Special Instruction in the program**
- **Condition produced by instruction**

External Signal (Hardware Interrupt):
An 8086 can get interrupt from an external signal applied to the nonmaskable interrupt (NMI) input pin; or the interrupt (INTR) input pin.

Special Instruction:
Interrupt Structure of 8086 supports a special instruction, INT to execute special program. At the end of the interrupt service routine, execution is usually returned to the interrupted program.

Condition Produced by Instruction:
An 8086 is interrupted by some condition produced in the 8086 by the execution of an instruction. For example divide by zero : Program execution will automatically be interrupted if you attempt to divide an operand by zero.

At the end of each instruction cycle 8086 Interrupts checks to see if there is any interrupt request. If so, 8086 responds to the interrupt by performing series of actions (Refer Fig. 9.1).

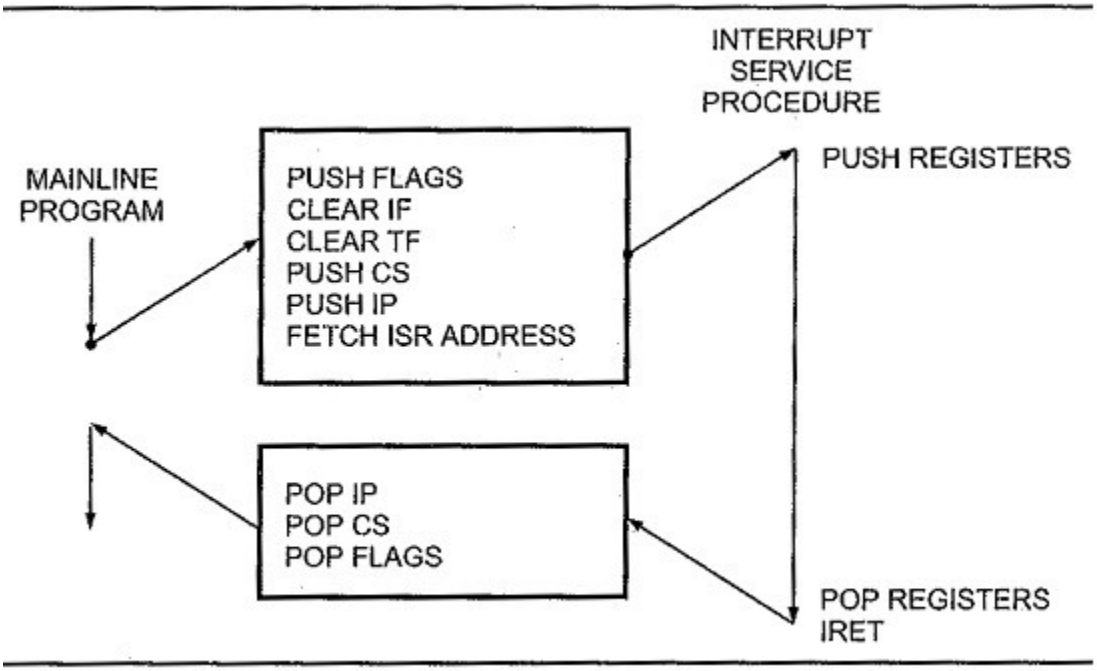


Fig. 9.1 8086 interrupt response

1. **It decrements stack pointer by 2 and pushes the flag register on the stack..**
2. **It disables the INTR interrupt input by clearing the interrupt flag in the flag**

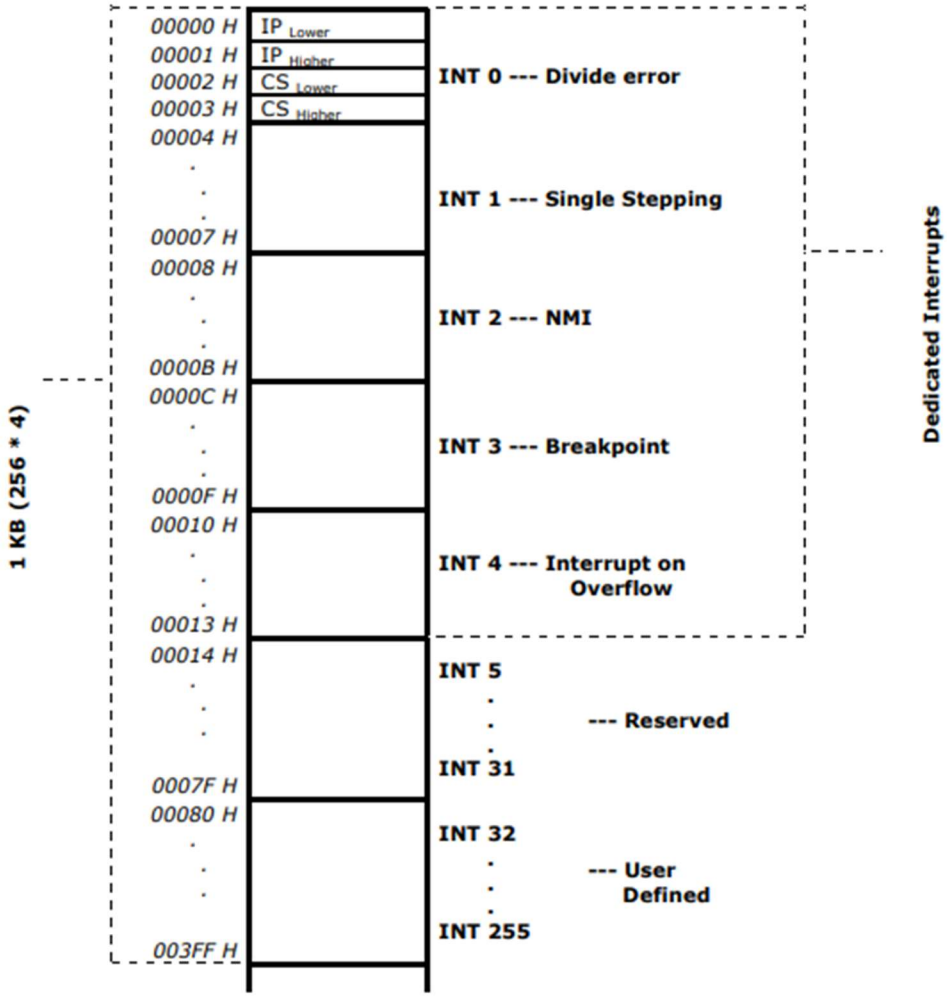
3. **It resets the trap flag in the flag register.**
4. **It decrements stack pointer by 2 and pushes the current code segment register contents on the stack.**
5. **It decrements stack pointer by 2 and pushes the current instruction pointer contents on the stack.**
6. **It does an indirect far jump at the start of the procedure by loading the CS and IP values for the start of the interrupt service routine (ISR).**

An IRET instruction at the end of the interrupt service procedure returns execution to the main program.

Interrupt Vector Table 8086:

Now the question is “How to get the values of CS and IP register ?” The 8086 gets the new values of CS and IP register from four memory addresses. When it responds to an interrupt, the 8686 goes to memory locations to get the CS and IP values for the start of the interrupt service routine. In an Interrupt Structure of 8086 system the first 1 Kbyte of memory from 00000H to 003FFH is reserved for storing the starting addresses of interrupt service routines. This block of memory is often called the **Interrupt Vector Table in 8086** or the **interrupt pointer table**. Since 4 bytes are required to store the CS and IP values for each interrupt service procedure, the table can hold the starting addresses for 256 interrupt service routines.

,,,,,,,,,,,,,,,,,,,,


1 KB (256 * 4)
00000 H
IP Lower
00001 H
IP Higher
00002 H
CS Lower
00003 H
CS Higher
INT 0 --- Divide error
00004 H
00007 H
INT 1 --- Single Stepping
00008 H
0000B H
INT 2 --- NMI
0000C H
0000F H
INT 3 --- Breakpoint
00010 H
00013 H
INT 4 --- Interrupt on Overflow
Dedicated Interrupts
00014 H
0007F H
INT 5
INT 31
--- Reserved
00080 H
003FF H
INT 32
INT 255
--- User Defined

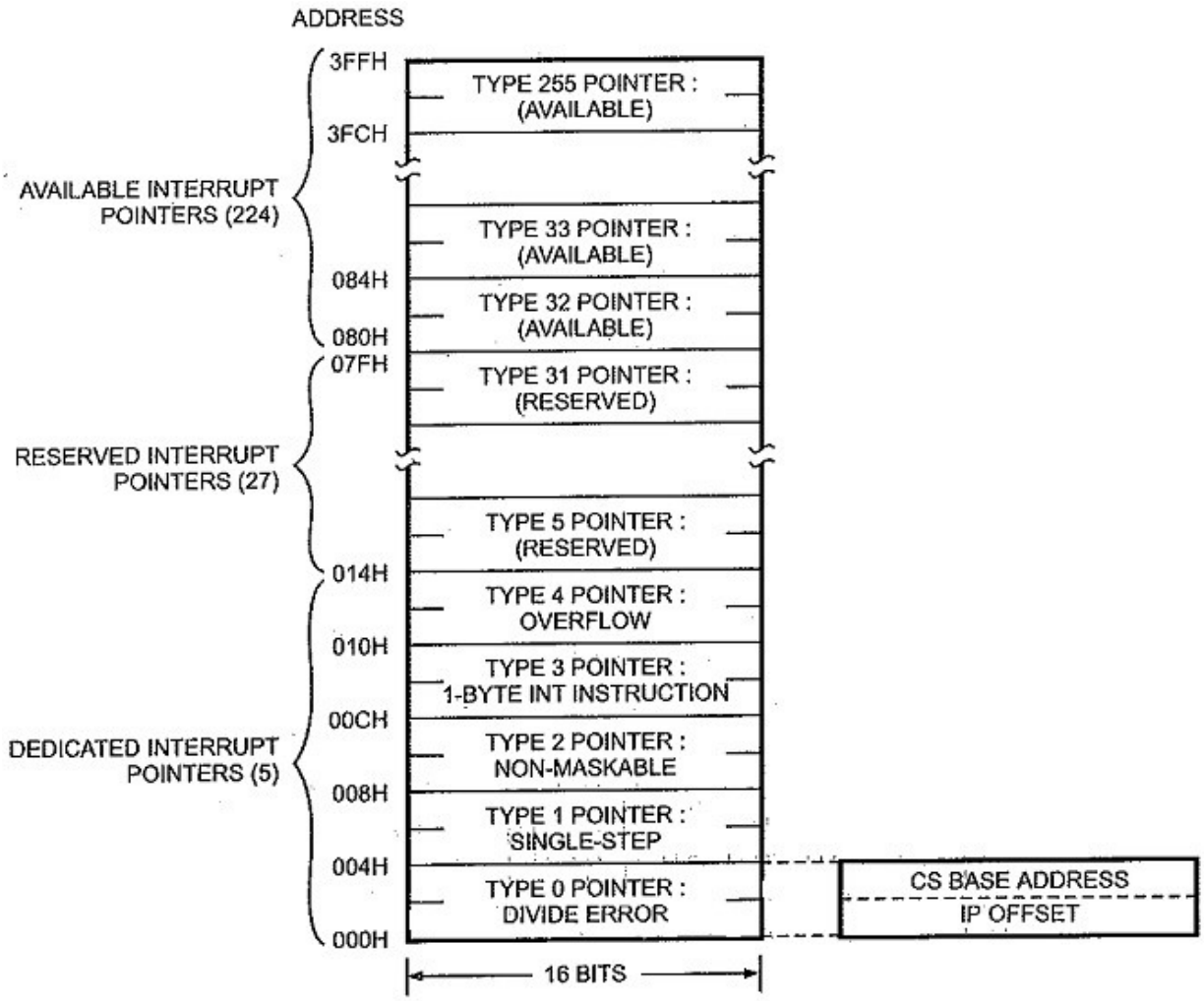


Fig. 9.2 8086 interrupt vector table

Each interrupt type is given a number between 0 to 255 and the address of each interrupt js found by multiplying the type by 4 e.g. for type 11, interrupt address is 11 x 4 = 44_{10}= 0002CH
Only first five types have explicit definitions such as divide by zero and non maskable interrupt. The next 27 interrupt types, from 5 to 31, are reserved by Intel for use in future microprocessors. The upper 224 interrupt types, from 32 to 255, are available for user for hardware or software interrupts.

When the 8086 responds to an interrupt, it automatically goes to the specified location in the Interrupt Vector Table in 8086 to get the starting address of interrupt service routine. So user has to load these starting addresses for different routines at the start of the program.

2. **Explain the first five dedicated interrupts of 8086**.

DEDICATED INTERRUPTS (INT 0 ... INT 4)

1) **INT O** (Divide Error)
This interrupt occurs whenever there is **division error**
i.e. when the result of a division is too large to be stored.
This condition normally occurs when the divisor is very small as compared to the dividend or the divisor is zero. #Refer example from Bharat Sir's lecture notes...
Its ISR address is stored at location 0 x 4 = 00000H in the IVT.

2) **INT 1** (Single Step)
The µP executes this interrupt **after every instruction if the TF is set**.
It puts µP in **Single Stepping** Mode i.e. the µP pauses after executing every instruction.
This is very useful during **debugging**. #Refer example from Bharat Sir's lecture notes...
Its ISR generally displays contents of all registers.
Its ISR address is stored at location 1 x 4 = 00004H in the IVT.

3) **INT 2** (Non Maskable Interrupt)
The µP executes this ISR in **response to** an interrupt on the **NMI** line.
Its ISR address is stored at location 2 x 4 = 00008H in the IVT.

4) **INT 3** (Breakpoint Interrupt)
This interrupt is used to cause **Breakpoints** in the program.
It is caused by writing the instruction INT 03H or simply INT.
It is useful in **debugging large programs** where Single Stepping is inefficient.
Its ISR is used to **display** the **contents of all registers** on the screen.
Its ISR address is stored at location 3 x 4 = 0000CH in the IVT.

5) **INT 4** (Overflow Interrupt)
This interrupt occurs if the **Overflow Flag is set AND** the µP executes the **INTO** instruction (Interrupt on overflow). #Show example from Bharat Sir's lecture notes...
It is used to detect overflow error in **signed arithmetic** operations.
Its ISR address is stored at location 4 x 4 = 00010H in the IVT.

Please Note: INT 0 ... INT 4 are called as dedicated interrupts as these interrupts are dedicated for the above-mentioned special conditions.

** Bharat ach6age no. 87 (Book), Pdf page no. 76

3. **Differentiate between memory mapped I/O and I/O mapped I/O**

   https://www.geeksforgeeks.org/difference-between-memory-mapped-io-and-io-mapped-io-with-reference-to-8085-microprocessor/

   OR

   Bharat acharya's

Differentiate between

	I/O Mapped I/O	Memory Mapped I/O
1	I/O device is **treated as an I/O device** and hence **given an I/O address**.	I/O device is **treated like a memory device** and hence **given a memory address**.
2	I/O device has an **8 or 16 bit I/O address**.	I/O device has a **20 bit Memory address.**
3	I/O device is given **IOR# and IOW#** control signals	I/O device is given **MEMR# and MEMW#** control signals
4	**Decoding is easier** due to lesser address lines	**Decoding is more complex** due to more address lines
5	Decoding is **cheaper**	Decoding is more **expensive**
6	Works **faster due to less delays**	More gates add more delays hence **slower**
7	Allows **max 2^{16} = 65536 I/O devices**	Allows **many more I/O devices** as I/O addresses are now 20 bits.
8	I/O devices can **only** be accessed by **IN and OUT** instructions.	I/O devices can now be accessed using **any memory instruction**.
9	ONLY **AL/ AH/ AX registers** can be used to transfer data with the I/O device.	**Any register** can be used to transfer data with the I/O device.
10	**Popular** technique in **Microprocessors**.	**Popular** technique in **Microcontrollers**.

4. **Why is it necessary to make even and odd banks while interfacing any kind of ROM to 8086**

   8086 has 20-bit addressing model for memory access. Each address represents a single byte - however, the natural word size of 8086 is 2 bytes, so you need a way to read two bytes at the same time - hence, two banks.

   The main benefit here is simplification - you need no memory controller, the CPU directly accessed data from the 8-bit modules.

6 marks

1. **Draw and explain the timing diagram of 8086 in minimum mode.**
2. **Draw and explain the timing diagram of 8086 in maximum mode.**

   1 & 2 in associated pdf

3. **What are exceptions, hardware interrupts and software interrupts of 8086. Explain their priority structure and interrupt vector table**

   - **Exceptions**
     An exception is a synchronous event that is generated when the processor detects one or more predefined conditions while executing an instruction. Exceptions occur when the processor detects an error condition while executing an instruction, such as division by zero. The processor detects a variety of error conditions including protection violations, page faults, and internal machine faults.

- **Hardware interrupts**

  Hardware interrupt is caused by any peripheral device by sending a signal through a specified pin to the microprocessor.
  The 8086 has two hardware interrupt pins, i.e. NMI and INTR.
  One more interrupt pin associated is INTA called interrupt acknowledge.

2 hardware interrupts in 8086 are:

NMI

It is a single non-maskable interrupt pin (NMI) having higher priority than the maskable interrupt request pin (INTR)and it is of type 2 interrupt.

When this interrupt is activated, these actions take place −

- Completes the current instruction that is in progress.
- Pushes the Flag register values on to the stack.
- Pushes the CS (code segment) value and IP (instruction pointer) value of the return address on to the stack.
- IP is loaded from the contents of the word location 00008H.
- CS is loaded from the contents of the next word location 0000AH.
- Interrupt flag and trap flag are reset to 0.

INTR

The INTR is a maskable interrupt because the microprocessor will be interrupted only if interrupts are enabled using set interrupt flag instruction. It should not be enabled using clear interrupt Flag instruction.

The INTR interrupt is activated by an I/O port. If the interrupt is enabled and NMI is disabled, then the microprocessor first completes the current execution and sends ‘0’ on INTA pin twice. The first ‘0’ means INTA informs the external device to get ready and during the second ‘0’ the microprocessor receives the 8 bit, say X, from the programmable interrupt controller.

These actions are taken by the microprocessor −

- First completes the current instruction.
- Activates INTA output and receives the interrupt type, say X.
- Flag register value, CS value of the return address and IP value of the return address are pushed on to the stack.

- IP value is loaded from the contents of word location X × 4
- CS is loaded from the contents of the next word location.
- Interrupt flag and trap flag is reset to 0

- **Software interrupts**

  These are instructions that are inserted within the program to generate interrupts. There are 256 software interrupts in 8086 microprocessor.

  The instructions are of the format INT type where type ranges from 00 to FF. The starting address ranges from 00000 H to 003FF H.
  These are 2 byte instructions. IP is loaded from type * 04 H and CS is loaded from the next address give by (type * 04) + 02 H.

  Format:

  INT n (Interrupt instruction with type number, n)

It is 2-byte instruction. First byte provides the op-code and the second byte provides the interrupt type number. There are 256 interrupt types under this group.

Its execution includes the following steps −

- Flag register value is pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of the word location 'type number' × 4
- CS is loaded from the contents of the next word location.
- Interrupt Flag and Trap Flag are reset to 0

The starting address for type0 interrupt is 000000H, for type1 interrupt is 00004H similarly for type2 is 00008H and ……so on. The first five pointers are dedicated interrupt pointers. i.e. −

- **TYPE 0** interrupt represents division by zero situation.
- **TYPE 1** interrupt represents single-step execution during the debugging of a program.
- **TYPE 2** interrupt represents non-maskable NMI interrupt.
- **TYPE 3** interrupt represents break-point interrupt.
- **TYPE 4** interrupt represents overflow interrupt.

Priority structure

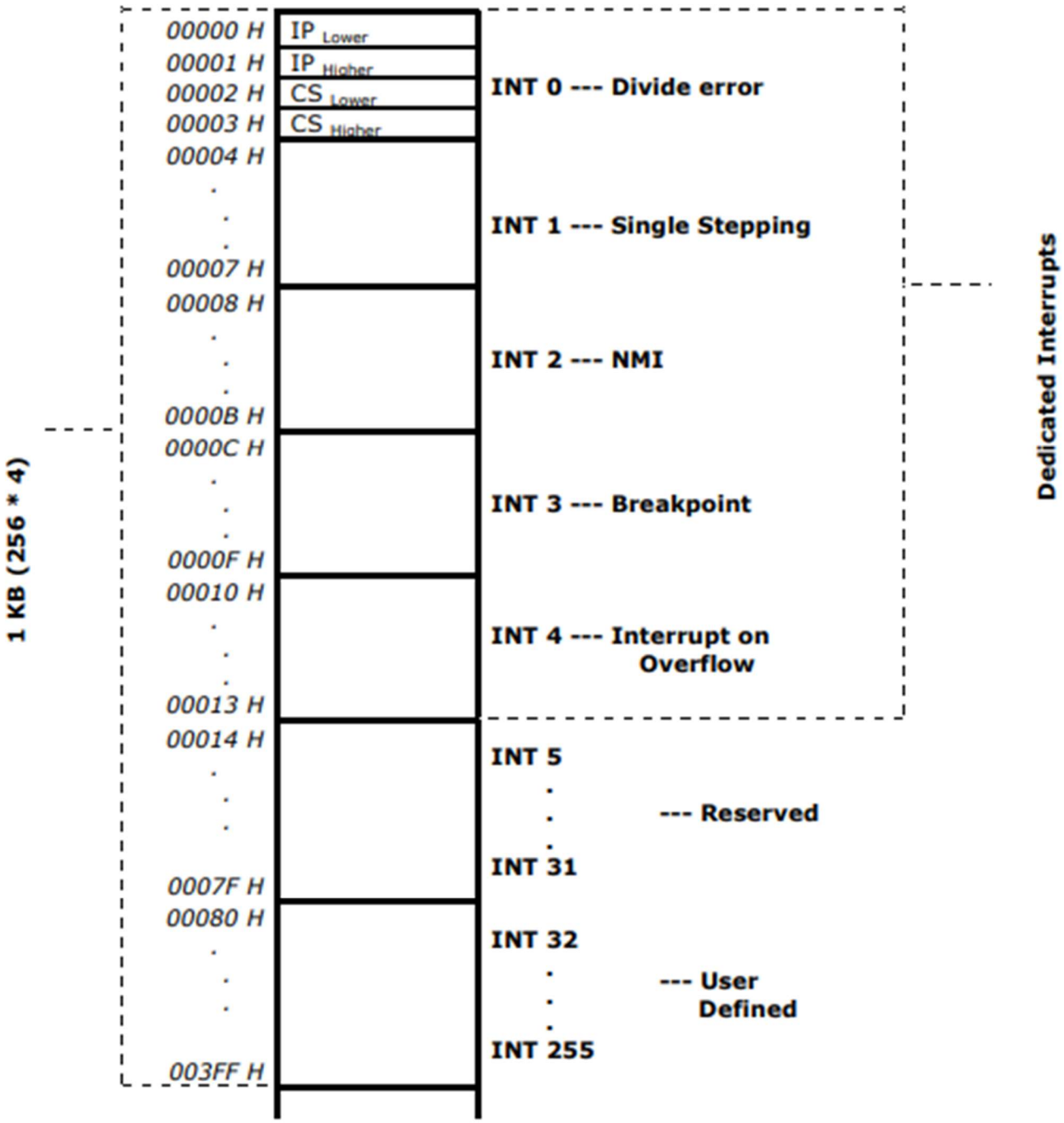
Software interrupts (All interrupts except single step, NMI and INTR interrupts) have the highest priority, followed by NMI followed by INTR. Single step has the least priority.

Interrupt	Priority
Divide Error, Int n, Int 0	HIGHEST
NMI	↓
INTR	↓
SINGLE - STEP	LOWEST

Examples:

- When divide-by-zero error interrupt and INTR interrupt comes simultaneously the 8086 will do a Divide error (type0) interrupt response first.
- When NMI interrupt and divide-by-zero error interrupt comes simultaneously the 8086 will do an NMI (type2) interrupt response first.

Interrupt Vector Table 8086:
Now the question is “How to get the values of CS and IP register ?”
The 8086 gets the new values of CS and IP register from four memory addresses. When it responds to an interrupt, the 8686 goes to memory locations to get the CS and IP values for the start of the interrupt service routine. In an Interrupt Structure of 8086 system the first 1 Kbyte of memory from 00000H to 003FFH is reserved for storing the starting addresses of interrupt service routines. This block of memory is often called the **Interrupt Vector Table in 8086** or the **interrupt pointer table**. Since 4 bytes are required to store the CS and IP values for each interrupt service procedure, the table can hold the starting addresses for 256 interrupt service routines.



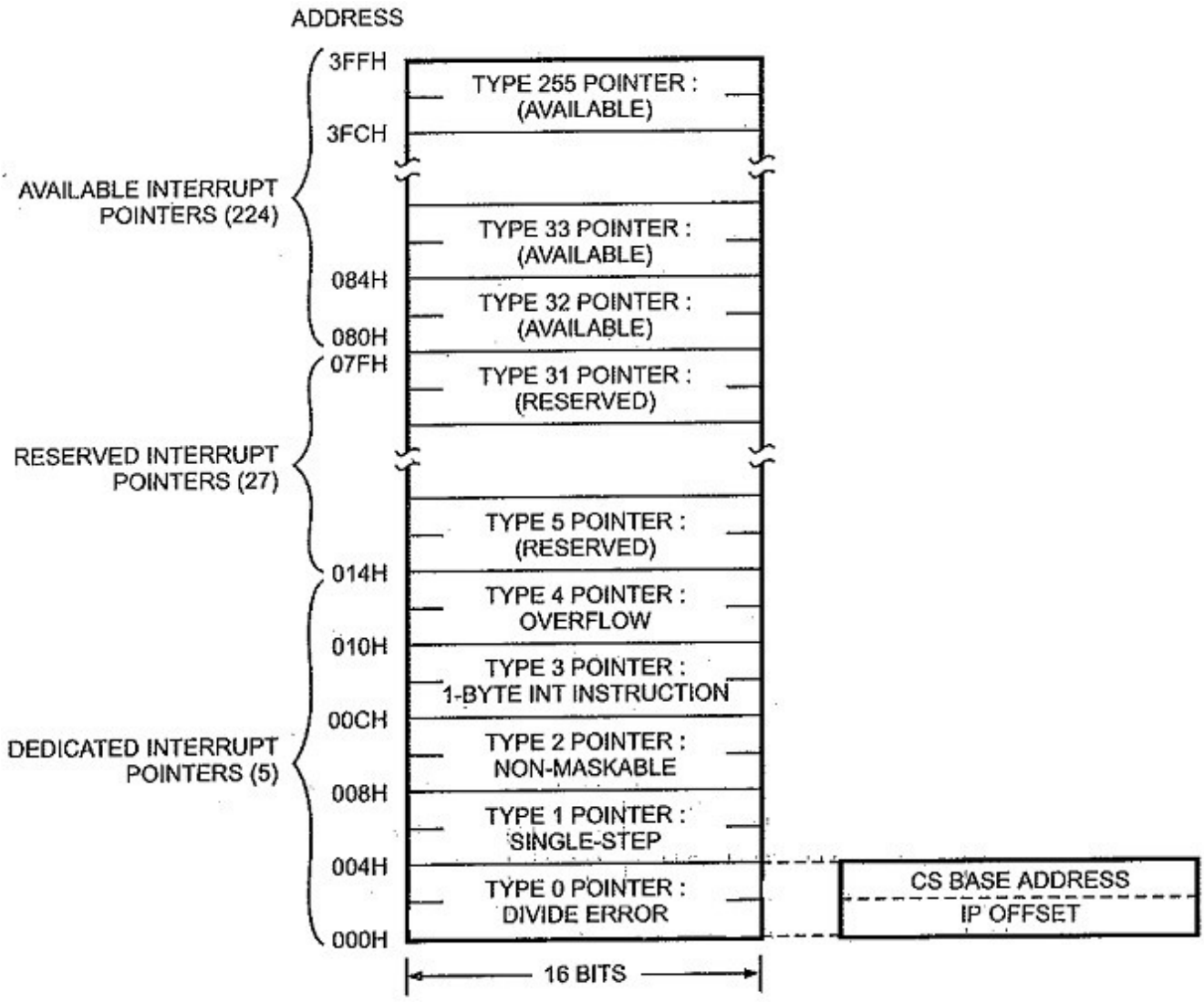


Fig. 9.2 8086 interrupt vector table