While 8086is executing a divide instruction which causes a divide-by- zero-error and a rising edge of NMI signal is received at the same time Explain whether a type-0 or type-2 interrupt service procedure will be executed.

Type-2 interrupt service procedure will be executed.

The 8086 has two types of interrupts: maskable and non-maskable. Maskable interrupts can be disabled by setting the interrupt flag (IF) in the flags register. Non-maskable interrupts cannot be disabled.

The divide-by-zero error is a maskable interrupt, while the NMI is a non-maskable interrupt. This means that if both interrupts occur at the same time, the NMI will be serviced first.

The NMI interrupt has the highest priority, so it will always be serviced before any other interrupt. This is because the NMI is used for emergency situations, such as power failures.

Once the NMI interrupt has been serviced, the divide-by-zero error will be serviced. The type-0 interrupt service procedure will be executed, and the appropriate action will be taken.

Explain the assembler directives EXTRN, EQU, EVEN.

- **EXTRN directive:** This directive is used to inform the assembler that a symbol or label is defined in another assembly language module. For example, the following directive tells the assembler that the symbol `SUM` is defined in another module:

  ```
  EXTRN SUM:DWORD
  ```

- **EQU directive:** This directive is used to assign a symbolic name to a constant value. For example, the following directive assigns the value 100 to the symbol `MAX`:

  ```
  MAX EQU 100
  ```

- **EVEN directive:** This directive is used to tell the assembler to align the next instruction or data item on an even address. For example, the following directive tells the assembler to align the next instruction on an even address:

  ```
  EVEN
  ```

Explain mode, command and status words of 8251 and their use in serial communication.

- **Mode word:** The mode word is used to configure the 8251 for a specific mode of operation. It can be used to set the baud rate, the number of stop bits, the parity bit, and the mode of operation (synchronous or asynchronous).
- **Command word:** The command word is used to control the 8251's operation. It can be used to start or stop transmission, to reset the 8251, and to read or write the 8251's registers.
- **Status word:** The status word is used to monitor the 8251's operation. It can be used to check the status of the transmitter and receiver, and to determine if any errors have occurred.

Explain the conditional jump instructions JAE, JPO, JLE and JGE of 8086 microprocessor.

- **JAE (Jump if Above or Equal):** This instruction will jump to the specified address if the carry flag (CF) is 0 or the zero flag (ZF) is 1.
- **JPO (Jump if Parity Odd):** This instruction will jump to the specified address if the parity flag (PF) is 0.
- **JLE (Jump if Less or Equal):** This instruction will jump to the specified address if the carry flag (CF) is 1 or the zero flag (ZF) is 1.
- **JGE (Jump if Greater or Equal):** This instruction will jump to the specified address if the carry flag (CF) is 0 or the zero flag (ZF) is 0.

Explain 8-bit and 16-bit signed division instructions in 8086.

The **DIV** instruction divides an 8-bit dividend by an 8-bit divisor. The dividend is stored in the AL register, and the divisor is stored in the BL register. The result of the division is stored in the AX register, with the quotient in the AL register and the remainder in the AH register.

The **IDIV** instruction divides a 16-bit dividend by an 8-bit divisor. The dividend is stored in the AX register, and the divisor is stored in the BL register. The result of the division is stored in the AX and DX registers, with the quotient in the AX register and the remainder in the DX register.

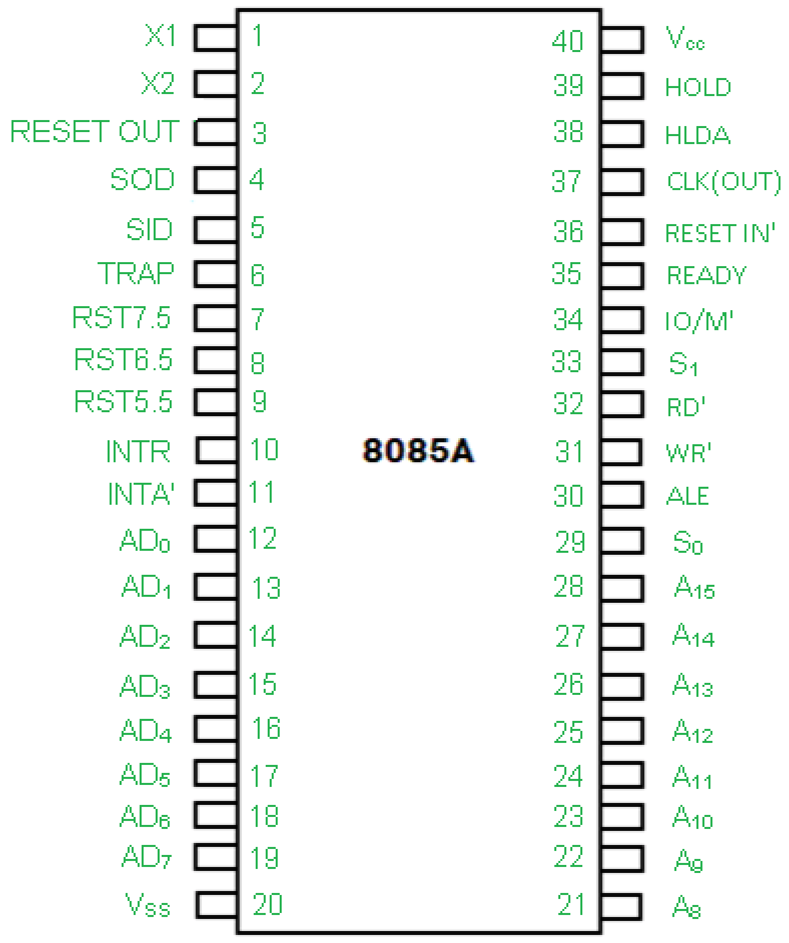
Example:

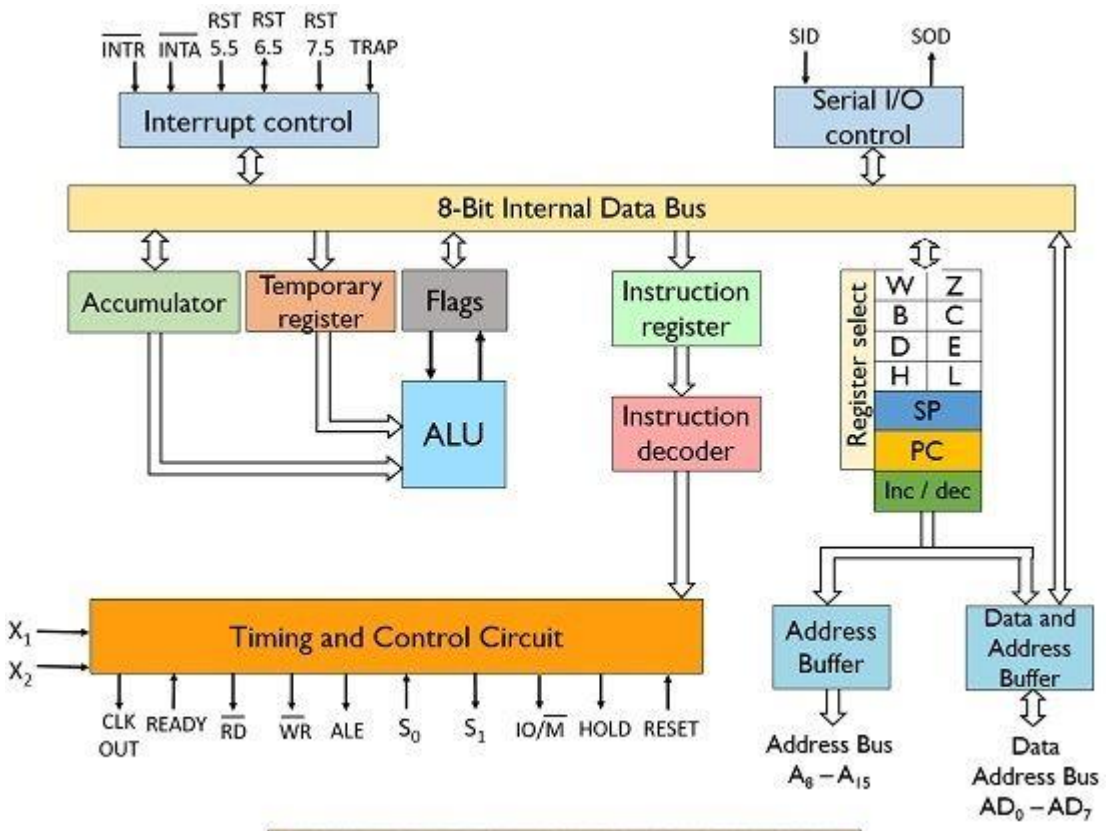
```
; Divide 8-bit dividend by 8-bit divisor

MOV AL, 10h

MOV BL, 2h

DIV BL

; The result of the division will be 5h in the AL register and 0h in the AH register


; Divide 16-bit dividend by 8-bit divisor

MOV AX, 100h

MOV BL, 2h

IDIV BL

; The result of the division will be 50h in the AX register and 0h in the DX register
```

Discuss evolution of 8086 to Pentium Processors in terms of register size, clock speeds, bus widths, memory access and instruction set capabilities and salient architectural features.

Processor	Register Size	Clock Speed	Bus Width	Memory Access	Instruction Set Capabilities	Salient Architectural Features
8086	16 bits	4.77 MHz	8 bits	1 MB	16-bit	16-bit registers, segmented addressing, real mode
8088	16 bits	4.77 MHz	8 bits	1 MB	16-bit	Same as 8086, but with an 8-bit data bus
80186	16 bits	6 MHz	16 bits	1 MB	16-bit	Added new instructions, improved addressing modes, and protected mode
80286	16 bits	10 MHz	16 bits	16 MB	16-bit	Added paging, virtual memory, and multitasking
80386	32 bits	16 MHz	32 bits	4 GB	32-bit	32-bit registers, flat addressing, and protected mode
80486	32 bits	25 MHz	32 bits	4 GB	32-bit	Added on-chip cache, pipelining, and SIMD instructions
Pentium	32 bits	60 MHz	64 bits	4 GB	32-bit	Added superscalar execution, floating-point unit, and MMX instructions

Draw the pin-out of 8085 microprocessor and explain the internal architecture and instructions of 8-bit microprocessor 8085.





Architecture of 8085 Microprocessor

Internal Architecture of 8085

- **Registers:** The 8085 microprocessor has 8 general-purpose registers, named B, C, D, E, H, L, A, and SP. These registers can be used to store data and address memory.
- **Accumulator:** The accumulator is an 8-bit register that is used to store arithmetic and logical results. It is the most commonly used register in the 8085 microprocessor.
- **Register pairs:** The 8 general-purpose registers can be paired to form 4 16-bit registers: BC, DE, HL, and SP. These register pairs are used to store larger data items.
- **Stack pointer:** The stack pointer is a 16-bit register that points to the top of the stack. The stack is a portion of memory that is used to store data temporarily.
- **Program counter:** The program counter is a 16-bit register that stores the address of the next instruction to be executed.
- **Instruction register:** The instruction register stores the current instruction that is being executed.
- **Memory address register:** The memory address register stores the address of the next memory location to be accessed.
- **Flag register:** The flag register is an 8-bit register that contains status flags that indicate the result of an arithmetic or logical operation.
- **System bus:** The system bus is a set of wires that connects the 8085 microprocessor to memory and I/O devices.

Instructions of 8085

- **Data transfer instructions:** These instructions are used to move data between registers, memory, and I/O devices.
- **Arithmetic instructions:** These instructions are used to perform arithmetic operations on data.
- **Logical instructions:** These instructions are used to perform logical operations on data.
- **Branch instructions:** These instructions are used to change the flow of execution of a program.
- **Stack I/O instructions:** These instructions are used to manipulate the stack.
- **I/O instructions:** These instructions are used to communicate with I/O devices.
- **Machine control instructions:** These instructions are used to control the operation of the 8085 microprocessor.

What is the purpose of Direction and Trap flag in 8086?

The **Direction Flag (DF)** controls the direction of string operations. When DF is set, string operations will increment the pointer to the string data. When DF is cleared, string operations will decrement the pointer to the string data.

The **Trap Flag (TF)** is used to enable single-step mode. When TF is set, the 8086 will execute one instruction and then stop. This allows the user to inspect the contents of the registers and memory before the next instruction is executed.

Explain the advantage of segment: Offset approach for accessing memory in 8086 microprocessor.

- **Flexibility:** The segment:offset approach allows for a wider range of addresses to be accessed. This is because the segment register can be used to select a different segment, which can then be combined with the offset to form a complete address.
- **Efficiency:** The segment:offset approach is more efficient than a direct addressing scheme because it allows the processor to access memory in larger chunks. This can improve the performance of programs that access large amounts of memory.
- **Simplicity:** The segment:offset approach is simpler to implement than a direct addressing scheme. This is because the processor only needs to decode the segment register and the offset, which is less complex than decoding a full 20-bit address.

Explain the function of following 8086 pins.

i) HOLD/HLDA ii) BHE iii) ALE iv) INTR/INTA v) Ready

- **HOLD/HLDA:** The HOLD pin indicates that another master(DMA) is requesting to take over the system bus. On receiving HOLD signal processor outputs HLDA signal HIGH as an acknowledgement.
- **BHE (Bus High Enable):** Low on this pin during first part of machine cycle indicates that at least one byte of current transfer is to be made on higher order byte AD_{15}-AD_8; otherwise transfer is made on lower order byte AD_7-AD_0.
- **ALE (Address Latch Enable):** This signal is provided by 8086 to demultiplex the AD_0-AD_{15} into A_0-A_{15} and D_0-D_{15} using external latches.
- **INTR:** It is a level triggered maskable interrupt request. It is sampled during last clock cycle of each instruction to determine if the processor should enter into an interrupt service routine.
- **INTA (Interrupt Acknowledge):** This indicates recognition of an interrupt request. It consists of two negative going pulses in two consecutive bus cycles.
- **READY:** If this signal is low the 8086 enters into wait state.

With suitable diagram explain how 8086 address and data buses are demultiplex or derived.

The **demultiplexing of the address bus** in the 8086 is done using the ALE (Address Latch Enable) signal. For demultiplexing twenty address lines one requires three latch chips like 74373. While demultiplexing the address bus, two of the three latch chips will be fully used and four latches of the third chip will be used.

The **demultiplexing of the data bus** in the 8086 is done using two bidirectional buffers 74245. The signals DNE and DT/R indicate the presence of data on the bus and the direction of the data.

If DNE is low it indicates that the data is available on the multiplexed bus and both the buffers are enabled to transfer data. When DIR pin goes high the data available at X pins of 74245 are transferred to Y pins. If DIR pin goes low the data available at Y pins of 74245 is transferred to X pins.

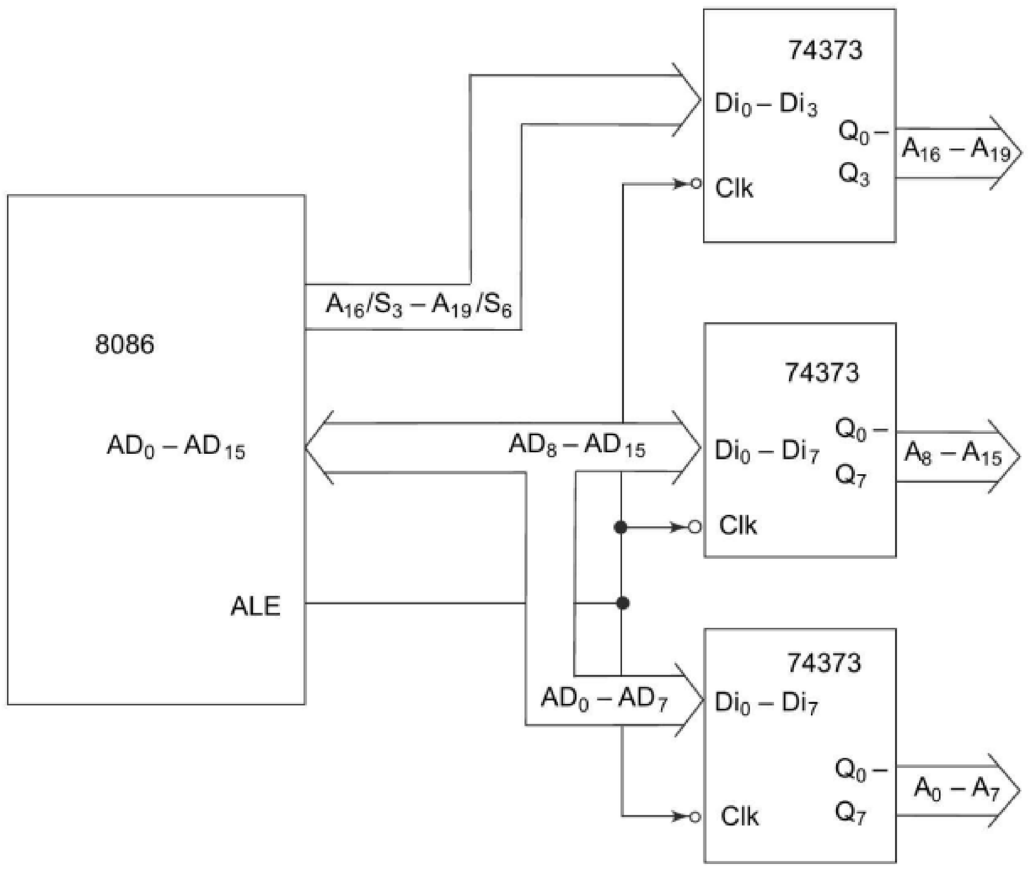


Fig. 1.10 *Latching 20-Bit Address of 8086*

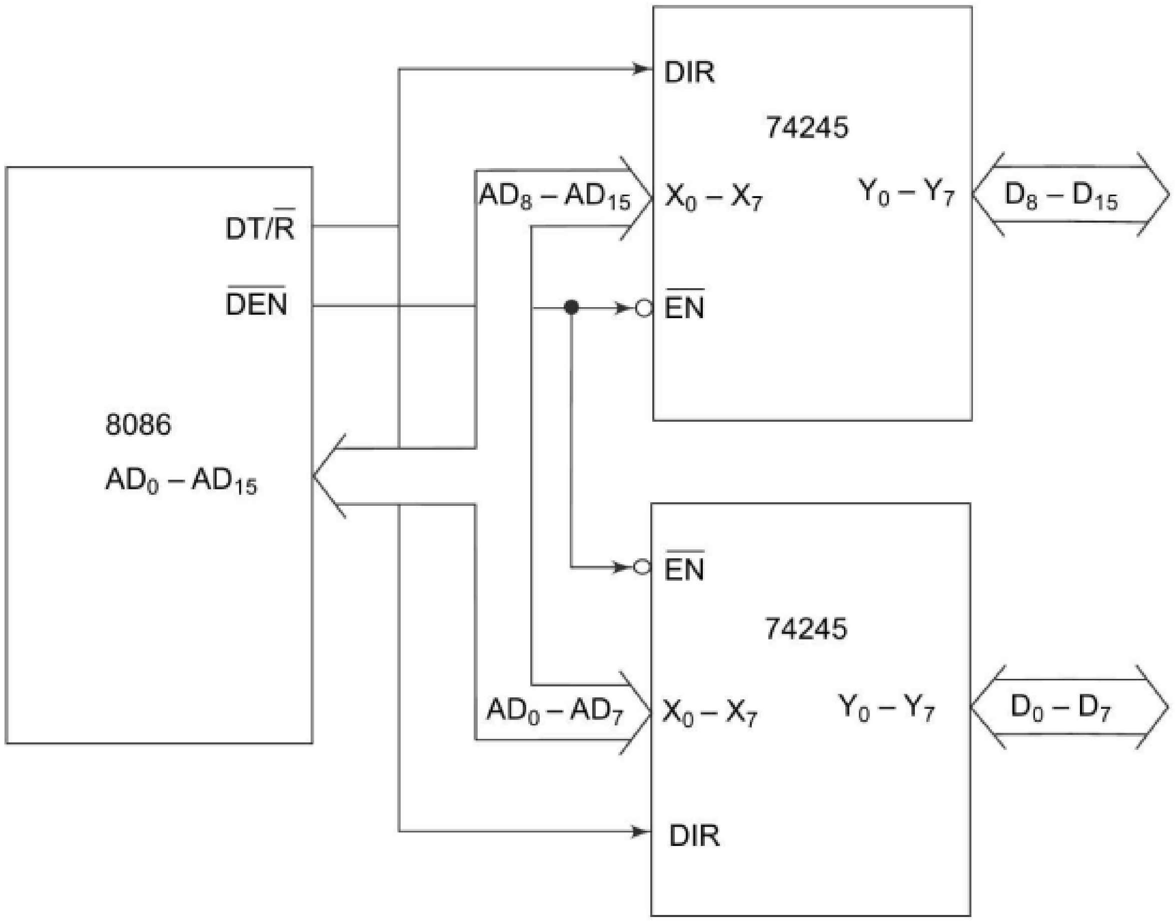


Fig. 1.11 *Buffering Data Bus of 8086*

Draw memory read and memory write cycle timing diagram for maximum mode operation of 8086.

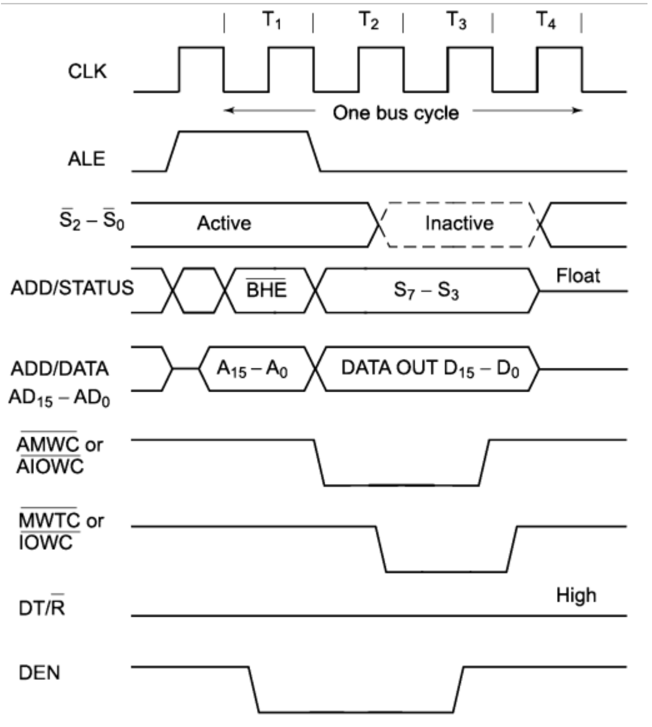


Fig. 1.16(b) *Memory Write Timing in Maximum Mode*

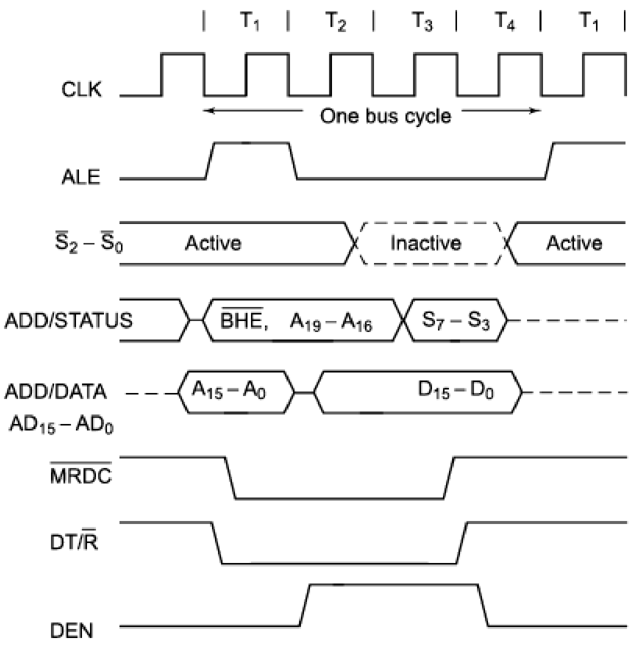
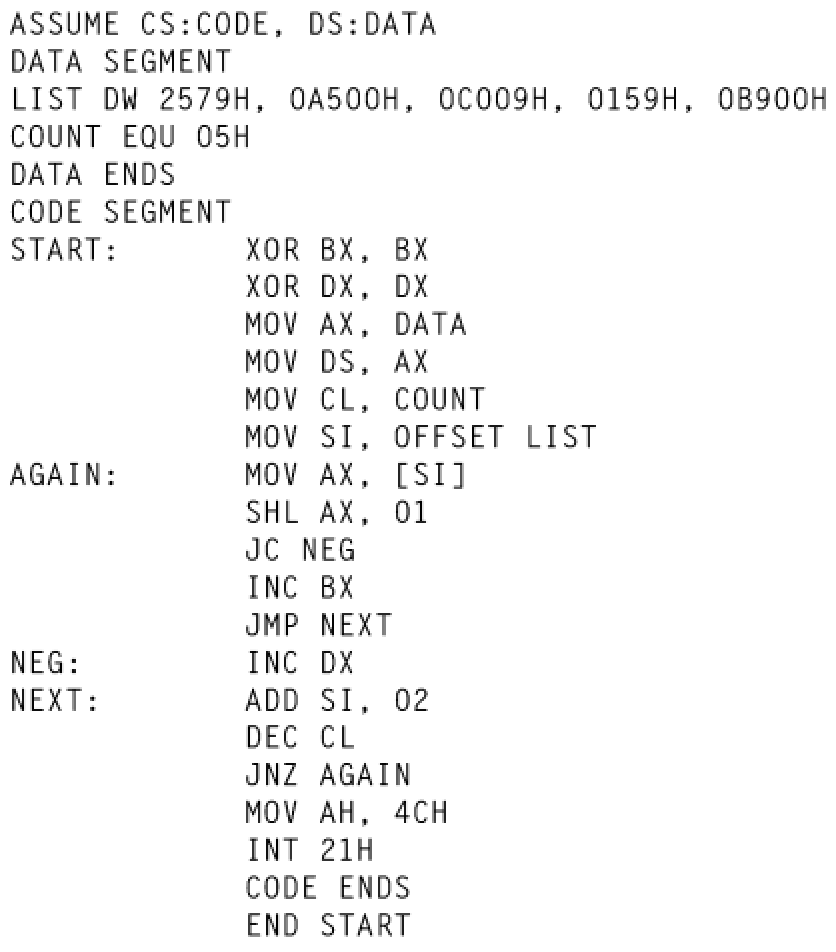


Fig. 1.16 (a) *Memory Read Timing in Maximum Mode*

Compare 8086 and 8088 microprocessors.

8086 Microprocessor	**8088 Microprocessor**
It has 16 bit data lines	It has 8 bit data lines
Memory space is organised as two 512 kB banks	Memory space is implemented as single 1MB memory bank
It has 6 bit instruction queue	It has 4 bit instruction queue
It has BHE	It has SSO status signal
It can read or write 8 bit or 16 bit data at a time	It can read or write 8 bit data at a time
I/O voltages level is measured at 2.5mA	I/O voltages level is measured at 2mA
Draws maximum supply current of 360mA	Draws maximum supply current of 340mA

Write an 8086 Assembly Language Program to find number of positive and negative integers from a list of signed integers.



```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
LIST DW 2579H, 0A500H, 0C009H, 0159H, 0B900H
COUNT EQU 05H
DATA ENDS
CODE SEGMENT
START:          XOR BX, BX
                XOR DX, DX
                MOV AX, DATA
                MOV DS, AX
                MOV CL, COUNT
                MOV SI, OFFSET LIST
AGAIN:          MOV AX, [SI]
                SHL AX, 01
                JC NEG
                INC BX
                JMP NEXT
NEG:            INC DX
NEXT:           ADD SI, 02
                DEC CL
                JNZ AGAIN
                MOV AH, 4CH
                INT 21H
                CODE ENDS
                END START
```

Write a sequence of 8086 assembly language instructions that clear that the leftmost three bits of DH register without changing the remainder of DH and store the result in BH register.

mov al, 00000111b ; Mask to clear the leftmost three bits

and dh, al ; Clear the leftmost three bits of DH

mov bh, dh ; Store the result in BH

What is the difference between AND and TEST instruction in 8086?

The **AND** instruction is used for supporting logical expressions by performing bitwise AND operation. The bitwise AND operation returns 1, if the matching bits from both the operands are 1, otherwise it returns 0.

```
e.g.    AND AL, 01H
        JZ EVEN_NUMBER
```

The **TEST** instruction works same as the AND operation, but unlike AND instruction, it does not change the first operand. So, if we need to check whether a number in a register is even or odd, we can also do this using the TEST instruction without changing the original number.

```
e.g.    TEST    AL, 01H
          JZ      EVEN_NUMBER
```

What condition(s) will terminate the repeated string instruction REPNE SCASB?

The REPNE SCASB instruction will terminate under the following conditions:

- The CX register reaches 0.
- The equal flag (ZF) is set.
- A hardware interrupt occurs.

With suitable examples illustrate the difference register relative base plus index addressing modes.

In **register relative addressing**, the effective address is calculated by adding the contents of a base register to a displacement. The displacement is a signed number that can be either positive or negative.

For example, the instruction MOV AX, [BP + 10] uses register relative addressing.

In **base plus index addressing**, the effective address is calculated by adding the contents of a base register, an index register, and a displacement. The displacement is a signed number that can be either positive or negative.

For example, the instruction MOV AX, [BX + SI + 10] uses base plus index addressing.

What is an emulator and debugger?

Feature	Emulator	Debugger
Purpose	To allow one computer system to behave like another computer system.	To help test and debug computer programs.
Functionality	Emulates the hardware and software of the target system.	Provides tools to step through code, set breakpoints, and inspect variables.
Use cases	Used to run software that is not compatible with the host system.	Used to find and fix bugs in software.

What is the purpose of tag register in numeric processor 8087?

The tag register contains several groups of 2 bits that can determine the state of the value of the eight 80-bit stack registers as valid, zero, special or empty. The tag word register presents the entire TAG field to the CPU. The tag values are 00 = valid, 01 = zero, 10 = special and 11 = empty.

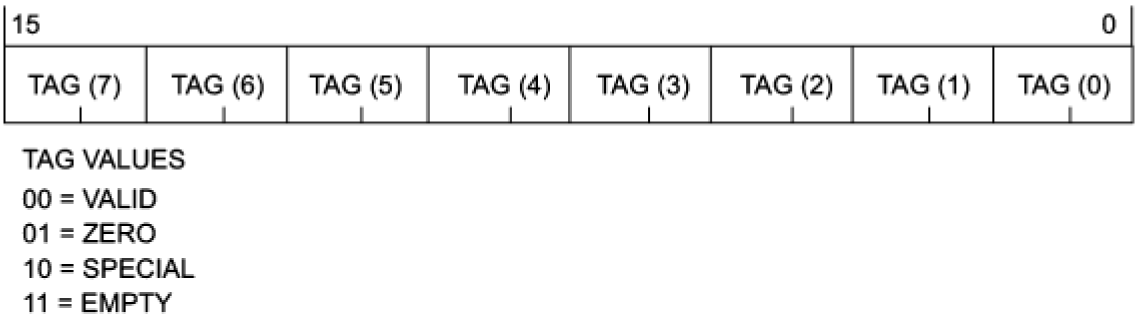


Fig. 8.13 *Tag Word of 8087*

Explain the various status bits of status register of numeric processor 8087.

- **C (Carry):** This bit is set if there is a carry out of the mantissa during an arithmetic operation.
- **Z (Zero):** This bit is set if the result of an arithmetic operation is 0.
- **S (Sign):** This bit is set if the result of an arithmetic operation is negative.
- **O (Overflow):** This bit is set if there is an overflow during an arithmetic operation.
- **P (Parity):** This bit is set if the result of an arithmetic operation has even parity.
- **A (Auxiliary Carry):** This bit is set if there is an auxiliary carry out of the mantissa during an arithmetic operation.
- **EM (Error Mask):** This bit is used to mask the error bits in the status register.
- **SF (Summary):** This bit is set if any of the error bits in the status register are set.

Explain following instructions of 8087 numeric processors.

i) FLD ii) FST iii) FSQRT iv) FCOM v) FABS

FLD (Load Real to Top of Stack): This instruction loads a real operand to the top of stack of the 80-bit registers, as shown:

```
FLD ST(7)     ; Stack Top ← [Reg 7]
FLD MEM       ; Stack Top ← [MEM]
```

FST (Store Top of Stack to the Operand): This instruction stores current content of the top of stack register to the specified operand, as shown:

```
FST ST (7)    ; Stack Top → [ST(7)]
FST MEM       ; Stack Top → [MEM]
```

FSQRT: This instruction finds out the square root of the content of the stack top and stores the result on the stack top again.

FCOM: This instruction compares real or integer operands specified by stack registers or memory.

FABS: This instruction replaces the content of the stack top by its absolute value (magnitude). The sign is neglected in operation.

Explain signal definitions for 8255 Mode 1 strobed input and Model strobed output operations and how these signals can be used to interface a keyboard and printer.

Input control signal definitions (mode 1)

✓ **STB (Strobe Input)-**If this line falls to logic low level, the data available at 8-bit input port is loaded into input latches

IBF (Input buffer full)-If this signal rises to logic 1, it indicates that data has been loaded into the latches, i.e. it works as an acknowledgement. IBF is set by a low on STB and is reset by the rising edge of RD input.

INTR (Interrupt request)- This active high output signal can be used to interrupt the CPU whenever an input device requests the service. INTR is set by a high at STB pin and a high at IBF pin. INTR is reset by a falling edge on RD input.

Output control signal definitions (mode 1)

OBF (Output buffer full)-This status signal, whenever falls to logic low, indicates that the CPU has written data to the specified output port. The OBF flip-flop will be set by a rising edge of WR signal and reset by a low going edge at the ACK input.

ACK (Acknowledge input)-ACK signal acts as an acknowledgement to be given by an output device. ACK signal, whenever low, informs the CPU that the data transferred by the CPU to the output device through the port is received by the output device.

INTR (Interrupt request)- Thus an output signal that can be used to interrupt the CPU when an output device acknowledges the data received from the CPU. INTR is set when ACK, OBF and INTE are '1'. It is reset by a falling edge on WR input.

✓ These signals can be used to interface a keyboard and printer like this:

- **STB (strobe) input:** This input is asserted to latch the data on Port A or B. When the STB input is asserted, the data on the corresponding port is latched and then transferred to the microprocessor or printer.
- **INTR (interrupt request) output:** This output is asserted when the data on Port A is latched. This can be used to notify the microprocessor that data is available on Port A.
- **D7-D0 (data lines):** These lines carry the data that is latched on Port A or B. These lines are used to transfer the data between the 8255 and the microprocessor or printer.