1. What is DMA and What are the uses of DMA?

DMA stands for Direct Memory Access. It is designed by Intel to transfer data at the fastest rate. It allows the device to transfer the data directly to/from memory without any interference of the CPU.

Using a DMA controller, the device requests the CPU to hold its data, address and control bus, so the device is free to transfer data directly to/from the memory. The DMA data transfer is initiated only after receiving HLDA signal from the CPU.

Direct Memory Access (DMA) is a process of data transfer between the memory and I/O devices. An external device takes over the control of system bus from the CPU and facilitates with the data transfer.

The basic idea of DMA is to transfer blocks of data directly between memory and peripherals. The data does not have to go through the microprocessor.

DMA controller can perform this task, relieving the CPU from the burden of transferring data. The CPU just issues the command for data transfer and shift its focus on actual data processing while the DMA handles the transfer of data

Can be used for high-speed data transfer between memory and peripherals such as HDD, magnetic tape etc.

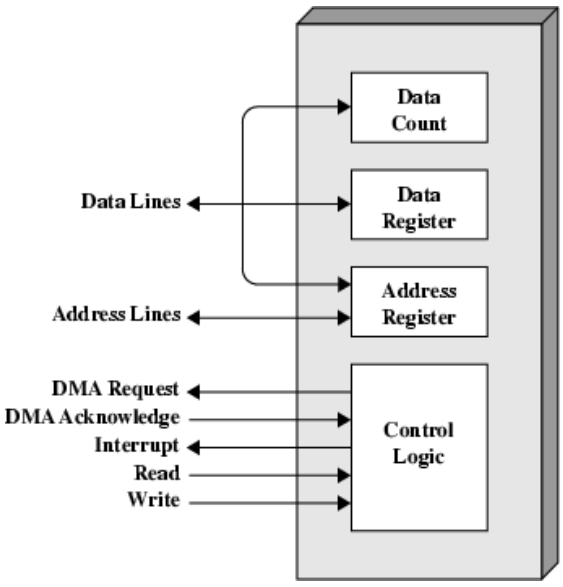
A DMA controller is connected with several peripherals that may request DMA.

The controller can decides the priority of the DMA requests, can communicate with the I/O devices and the CPU, and provides memory addresses for data transfer.

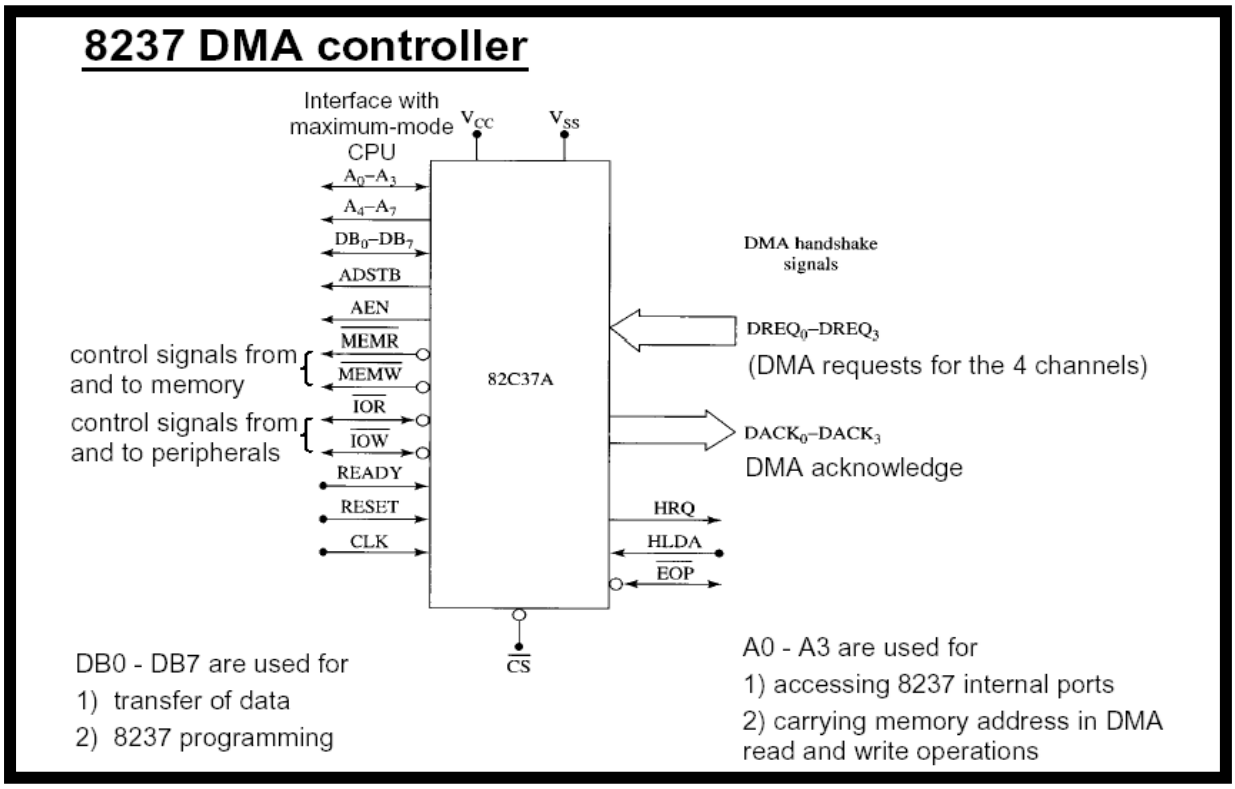
For e.g. the Intel 8237 is a DMA controller

It is a 4-channel device. Each channel is dedicated to a specific I/O device

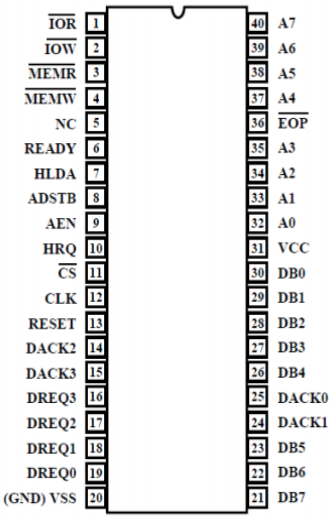
It is capable of addressing 64KB section of memory



2. Describe the functional Diagram of 8257 DMA Controller.



Pin Description:



- **DREQ3 – DREQ0 (DMA channel request):** Used to request a DMA transfer from a particular DMA channel.
- **DACK3 – DACK0 (DMA channel acknowledge):** Acknowledges a channel DMA request from a device.
- **HRQ (Hold request):** Requests control of the system bus to the CPU.
- **HLDA (Hold acknowledge):** Signals the 8237 that the microprocessor has acknowledged the hold request and relinquished control of the system bus.
- **AEN (Address enable):** Enables the DMA address latch connected to the 8237 (Use to take the control of the address bus from the microprocessor)
- **ADSTB (Address strobe):** It is used to latch the higher 8 bits of the address during DMA operation.
- **EOP (End of process):** It is a bi directional pin that signals the end of the DMA process.
- **IOR (I/O read):** A bi-directional pin which the CPU can use to read data from the internal registers of 8237 and can be used as an output to read data from a peripheral during a DMA write cycle.

- **IOW (I/O write** A bi-directional pin which the CPU can use to write data to the internal registers of 8237 and can be used as an output strobe to write data to a peripheral during a DMA read cycle.
- **MEMW (Memory write):** Used to write data to the selected memory during a DMA write cycle.
- **MEMR (Memory read):** Used to read data from the selected memory during a DMA read cycle.
- **A3 – A0:** They are bi directional pins. They can be used to select the internal registers of the 8237 by the CPU . Otherwise holds the lower nibble of the lower 8 bits of the memory address from where data is to be read / write during DMA operation.
- **A7 – A4:** Used to provide the higher nibble of the lower 8 bits of the memory address from where data is to be read / write during DMA operation.
- **DB0 – DB7:** Carries data during data transfer. Otherwise stores the higher 8 bits of the memory address from where data is to be read / write during DMA operation.

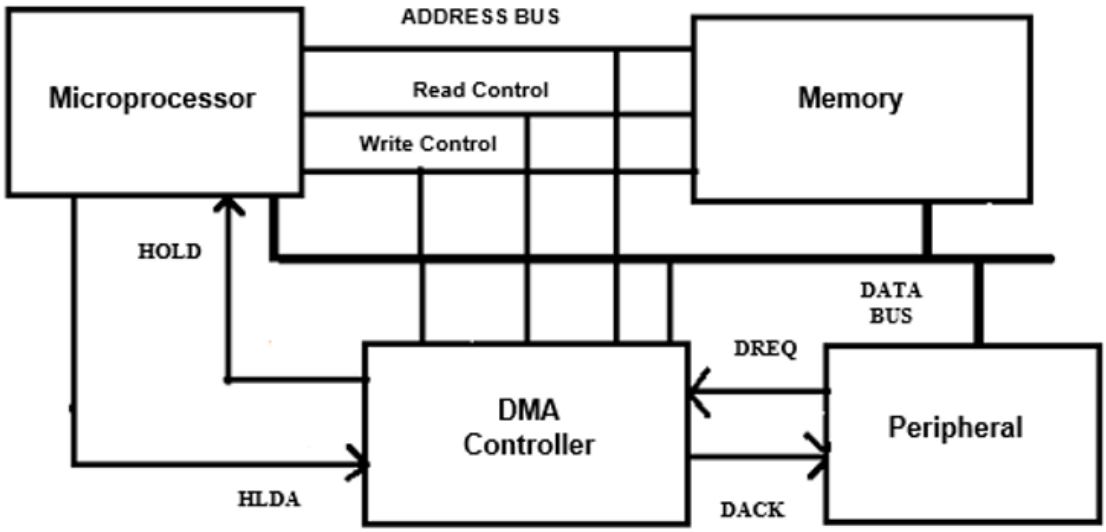
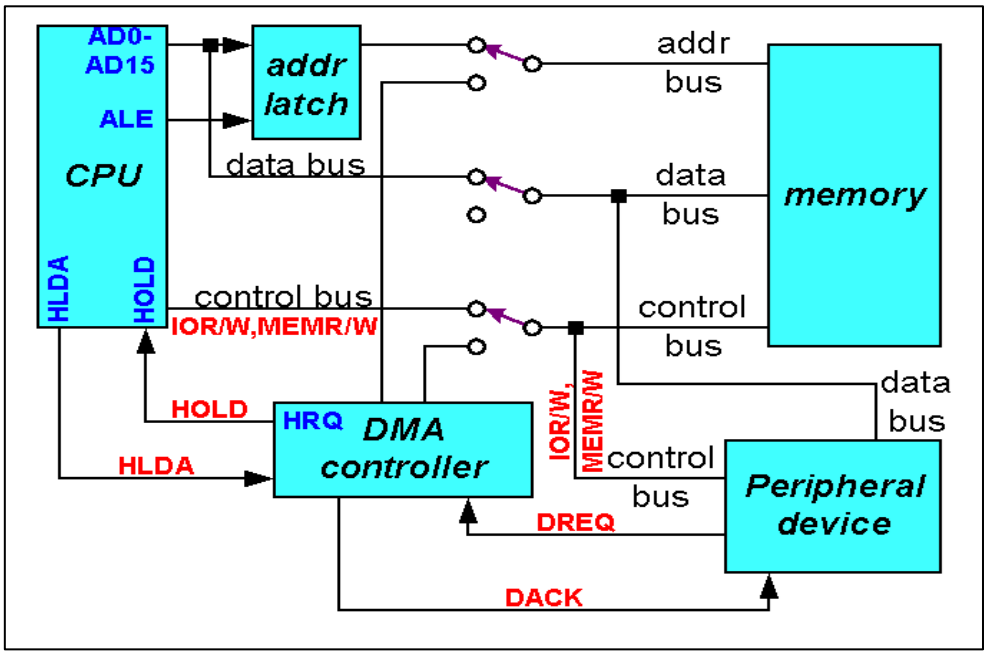


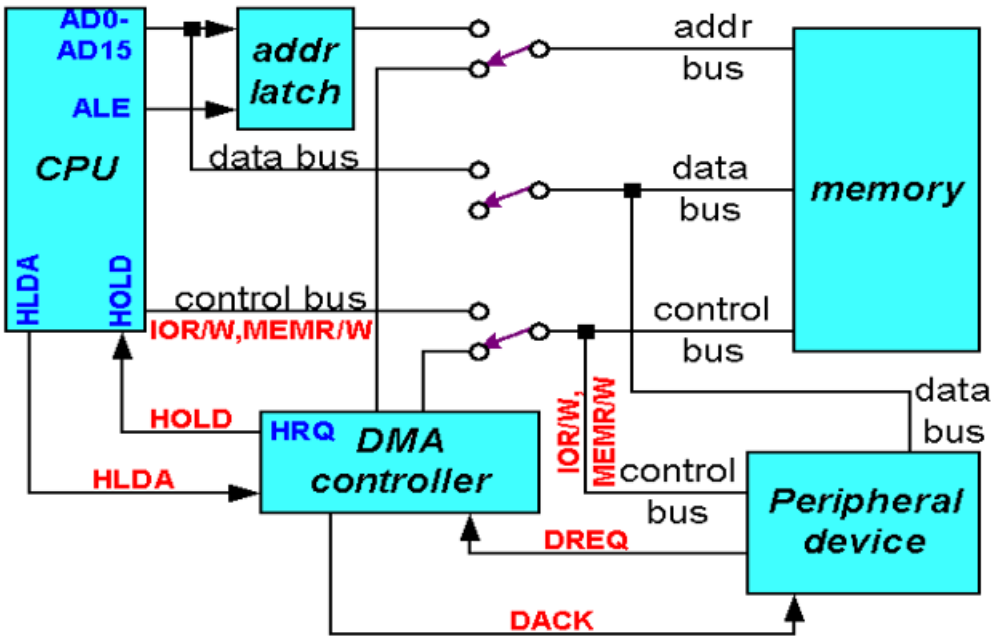
Fig : Simplified Block Diagram of a DMA Operation

- CPU invokes a read/write command to the peripherals
- The peripherals when ready sends a DMA request (DREQ) signal to the DMA controller
- The DMA controller sends a Hold Request (HRQ) signal to the CPU to gain control of the system bus
- The CPU replies with a Hold Acknowledgement signal to the DMA giving the DMA controller total control over the system bus
- The DMA controller then sends a DMA Acknowledgement signal to the peripherals in reply to the previous DMA request signal

- Data can now be transferred between Memory and I/O with the help of the DMA controller
- DMA controller sends interrupt signal to the CPU when finished with data transfer.
- The DMA disables the DMA Acknowledgement signal and the CPU disables the Hold Acknowledgement signal taking back control of the system bus.



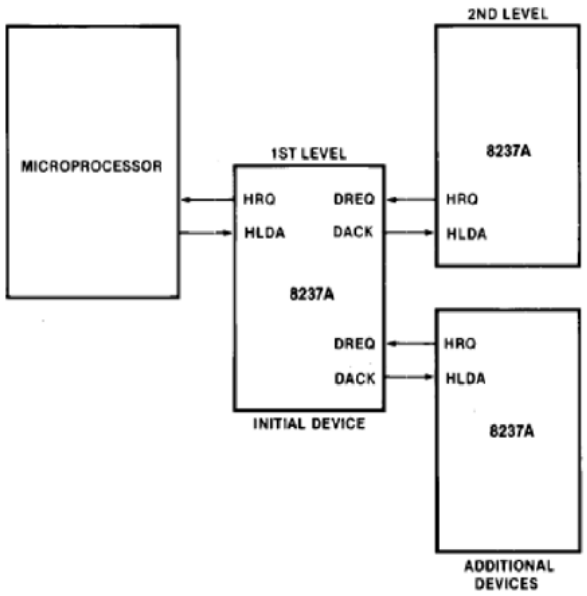
When the CPU is Bus Master



When DMA takes control

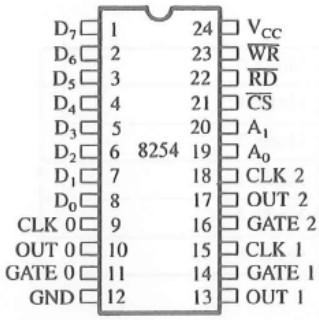
DMA Cascade Mode

- Using this method more than one 8237 DMA Controller can be connected.
- This can be adopted to get the utility of more than 4 channels
- The HRQ and HLDA signals from the additional controllers are connected with the DREQ and DACK pins of the primary or "Host" controller
- It is the responsibility of the host to prioritize and co ordinate the secondary controllers.



3. Draw and discuss the block diagram of 8254 programmable Interval Timer. (Book Page 494)

Pin Configuration



Pin Names

D_7–D_0	Data Bus (8 Bit)
CLK N	Counter Clock Inputs
GATE N	Counter Gate Inputs
OUT N	Counter Outputs
$\overline{RD}$	Read Counter
$\overline{WR}$	Write Command or Data
$\overline{CS}$	Chip Select
A_0–A_1	Counter Select
V_{CC}	+5 Volts
GND	Ground

Block Diagram

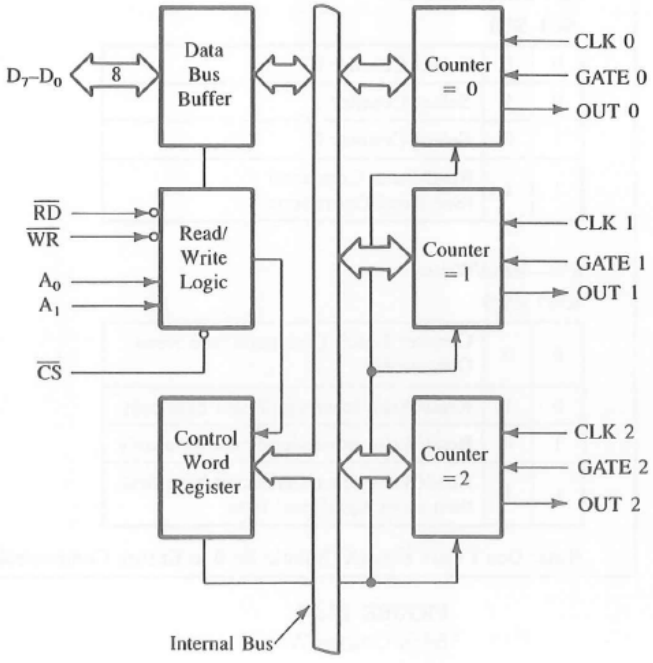


FIGURE 15.23
8254 Block Diagram

4. Discuss the role of RS232 interface in detail.

RS-232 is an interface for the interchange of serial binary data between two devices. It is a standard protocol used for serial communication; it is used for connecting a computer and its peripheral devices to allow serial data exchange between them. RS-232 was commonly used in computers to connect devices like a printer or a telephone modem. RS-232 port is often referred to as the serial port in computer terminology. RS-232 is standard defined by RETMA, a predecessor of the Electronic Industries Association (EIA). Hence the name RS (RETMA Standard). RS-232 was initially developed to standardize the connection of computers with telephone line modems.

Universal Asynchronous Data Receiver & Transmitter (UART) is used in connection with RS232 for transferring data between printer and computer. The micro controllers are not able to handle such kind of voltage levels, connectors are connected between RS232 signals. These connectors are known as the DB-9 Connector as a serial port and they are of two type Male connector (DTE) & Female connector (DCE). Common connectors in personal computers use either a 9 pins RS-232 connector or a 25 pins RS-232 connector. RS-232 provides a typical transmission speed of up to 9600 bps over a maximum distance of 15 meters.

5. Design an interfacing circuit to interface a 256-byte memory module externally using peripheral- mapped I/O. Also write the code to read and write data from this memory module.
   Watch from this link: https://youtu.be/tVRi0AEc6ao?si=EUdo514iyjzVg_3B
6. Why is Serial Data Transfer mostly preferred over parallel data transfer? Give reasons.

   Serial data transfer is often preferred over parallel data transfer in certain applications due to several reasons:

   1. Reduced Number of I/O Pins:
      - Serial communication requires fewer I/O (input/output) pins compared to parallel communication. This is particularly advantageous in scenarios where the number of available pins is limited, such as in integrated circuits or when interfacing with other devices.

   2. Simplified PCB Layout:

- The use of fewer wires simplifies the PCB (Printed Circuit Board) layout. This can lead to more straightforward designs, reduced manufacturing costs, and improved signal integrity, especially in high-speed applications.

3. Lower Cost:

- Serial communication typically requires less hardware (fewer wires, connectors, and pins), resulting in lower manufacturing costs. This is significant in mass production and cost-sensitive applications.

4. Easier Synchronization:

- Synchronizing data in serial communication is often simpler than in parallel communication, where precise alignment and synchronization of multiple data lines are required. Serial communication is inherently synchronized by a clock signal, making it more robust.

5. Extended Cable Lengths:

- Serial communication is generally more tolerant of longer cable lengths compared to parallel communication. This makes serial communication suitable for applications where devices are physically separated by significant distances.

6. Flexibility and Scalability:

- Serial communication is more flexible and scalable than parallel communication. It is easier to add or remove devices without significant changes to the communication infrastructure. This is particularly important in applications where the system may need to expand or contract.

7. Ease of Implementation:

- Serial communication is often easier to implement in terms of software and hardware. It requires less complex circuitry and is more straightforward to manage in software, especially for devices with limited resources.

8. Reduced Electromagnetic Interference (EMI):

- Serial communication typically involves fewer simultaneous switching signals, which can reduce electromagnetic interference. This is especially crucial in high-frequency or sensitive applications where EMI can affect signal quality.

9. Compatibility with Single-Wire Communication:

- Serial communication is well-suited for single-wire communication protocols like I2C and SPI. These protocols allow multiple devices to communicate over a single wire, simplifying the wiring and reducing the number of required connections.

Despite these advantages, it's essential to note that the choice between serial and parallel communication depends on the specific requirements of the application. Parallel communication may still be preferred in scenarios where high data transfer rates are critical, and the benefits of parallelism outweigh the drawbacks.

7. What is timer? Write the control word format of 8253 programmable interval timer.

Read From Book Page 494- The main Difference Between 8253 and 8254 is 8253 works in a clock Frequency 2.6MHz and 8254 works in a clock frequency 8MHz. 8253 doesn't Have read Back feature 8254 has read back feature.

8. Interface 8253 programmable interval timer with 8085 microprocessors. Suppose Pin CS of a given 8253 is activated by binary address:
A7-A2=100101.
Find the port address assigned to this 8253.
Find the configuration for this 8253 if the control register is programmed as follows:
MOV AL, 00110110
OUT 97h, Al

To find the port address assigned to the 8253 programmable interval timer (PIT), we need to use the given binary address A7-A2=100101 and convert it to its hexadecimal equivalent. The 8253 typically has three registers accessed through different port addresses. The base address for the 8253 is determined by the A7-A2 bits. The three registers are accessed by adding 0, 1, and 2 to the base address.

Given A7-A2=100101, the base address in hexadecimal is 0x94. Therefore, the port addresses for the three registers are as follows:

1. Counter 0 (Control Register): 0x94
2. Counter 1: 0x95
3. Counter 2: 0x96

Now, let's look at the given configuration for the 8253:

MOV AL, 00110110

OUT 97h, AL

Here, the value 00110110 is loaded into the AL register, and then it is output to the port address 97h.

The output instruction OUT 97h, AL writes the value in the AL register to the port address 97h. Since 97h corresponds to the base address (0x94) plus the offset for Counter 0, the configuration is being written to the control register of Counter 0.

Therefore, the configuration for Counter 0 (Control Register) of the 8253 is set to 00110110. The specific meaning of these bits (mode, binary/BCD, etc.) would depend on the datasheet of the 8253 and the desired configuration for your application.

For more information please read from the book Page 496

9. What do you mean by odd Address bank and even Address bank of 8086 processor.

   PDF Provided named Memory Bank

   Also you can watch: https://youtu.be/T_MdZzBANrw?si=IgtR9jIX3Brg7V05

10. Discuss how a byte and a word data is read from the odd boundary and even boundary address of 8086 processor.

In the context of the 8086 processor, reading a byte or a word from an odd or even boundary address refers to the alignment of the memory address with respect to the size of the data being accessed (byte or word).

Byte Read:

1. Even Boundary (Aligned):

- When reading a byte from an even boundary address (an address that is a multiple of 2), there is no issue. The 8086 processor can directly fetch the byte from the specified memory location.

Example:

MOV AL, [Even_Address]

2. Odd Boundary (Unaligned):
- Reading a byte from an odd boundary address (an address that is not a multiple of 2) is also straightforward. The 8086 processor can handle unaligned accesses for byte-sized data without any issues.

Example:

MOV AL, [Odd_Address]

Word Read:

1. Even Boundary (Aligned):
- When reading a word from an even boundary address (an address that is a multiple of 2), there is no issue. The 8086 processor can fetch the entire word directly from the specified memory location.

Example:

MOV AX, [Even_Address]

2. Odd Boundary (Unaligned):
- Reading a word from an odd boundary address (an address that is not a multiple of 2) requires special consideration. The 8086 processor expects word data to be aligned on even

boundaries, so if you attempt to read a word from an odd address, it will result in two memory accesses—one for the odd byte and one for the even byte.

Example:
assembly
MOV AX, [Odd_Address]

In this case, the processor performs two consecutive byte reads and then combines them to form the word value in the AX register.

It's important to note that while the 8086 processor can handle unaligned access for byte-sized data, accessing word-sized data from an odd address incurs a performance penalty due to the need for two memory accesses. Modern processors often have better performance for aligned memory access, and optimizing code for alignment can lead to more efficient execution.

Always refer to the processor's documentation or manuals for the most accurate and detailed information regarding memory access and alignment requirements.

Also You can watch: https://youtu.be/6VSObCrprOQ?si=UakxmmdOaHoS81oB