8251 USART: The 8251 is a USART for serial combination.

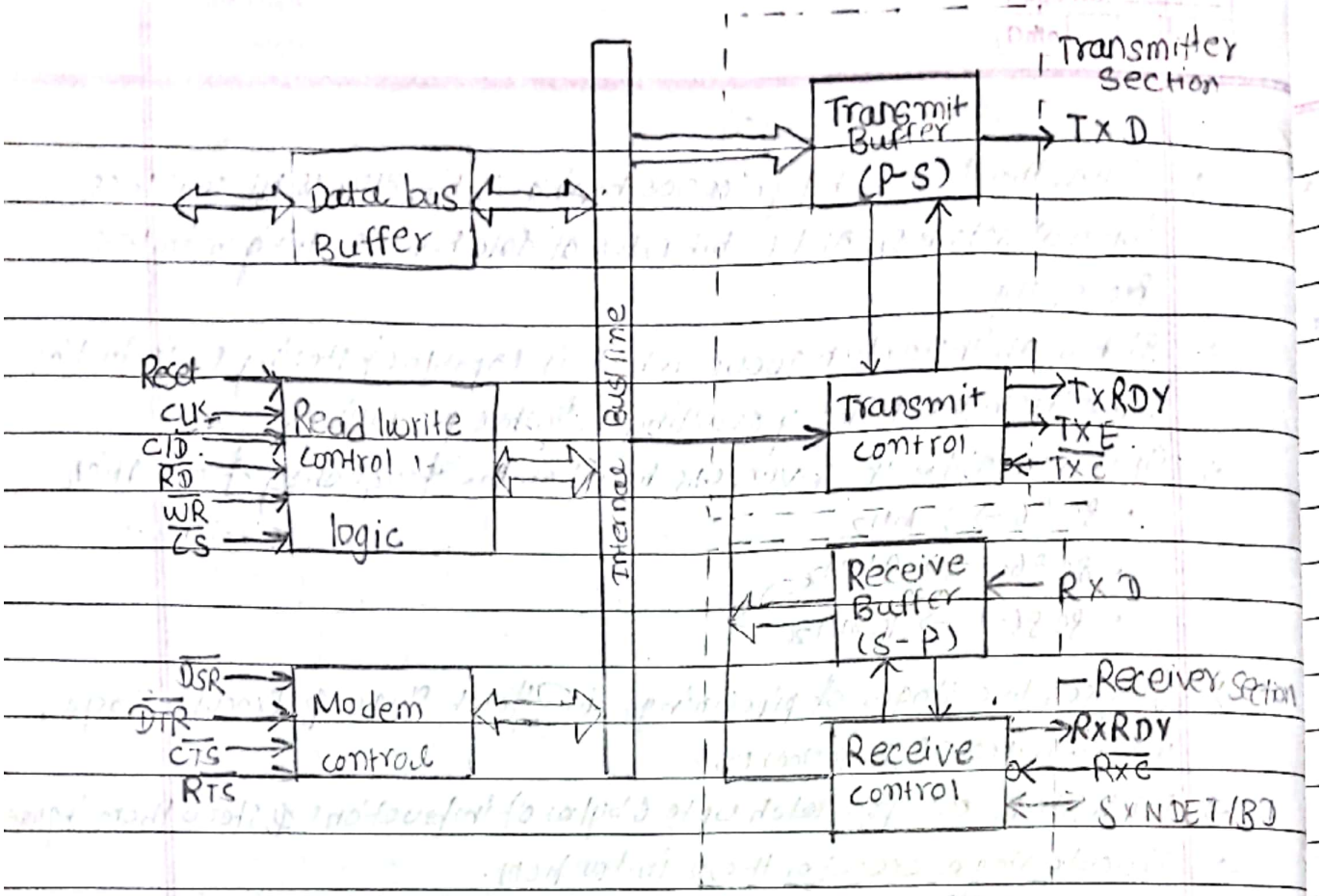


fig: Block diagram of 8251 USART

The control logic interface the Chip with the MPU. The transmitter section converts a parallel word received from MPU into serial bits and transmit them over the TxD line to a peripheral. The receiver section receives serial bits from a peripheral, converts them into a parallel word and transfers back to MPU. The MODEM control is used to establish data communication through modem over telephone lines.

The block diagram of 8251 consists of following blocks or parts or components.

1. Data Bus Buffer
2. Read Iwrite control signal
3. ~~Transmi~~ Modem control
4. Transmitter Section.
5. Receiver Section.

1. Data Bus Buffer:
The data bus buffer is a 8-bit data bus used to read or write status, command word or data from or to the 8251.

2. Read/write control logic:
This includes all the control signal or logic. It has 6 input signals and 3 buffer register.
The contrd logic interface the chip with mPu, delermines the function of chip according to the control word in the control register and monitors the data flow.

(a) 8 - input signals:

(i) $\overline{CS}$ (chip select):
when this signal is low, the 8251 is selected by mpu for communication.

(ii) C/$\overline{D}$ (control/Data):
when it is high, the control or status register is addressed.
when it is low, the data buffer is addressed.

(iii) $\overline{WR}$ (write):
when it is low, the mpu either writes into control register or sends output to the data bus buffer.

(iv) $\overline{RD}$ (Read):
when this signal is low, the mpu reads or accepts data from data bus buffer.

(v) RESET:
This signal makes 8251 idle.

(vi) CLK (Clock):
It is usually connected to system clock for communication with the µP.

(b) 3 Buffer register.

(i) control register:

It is a 16-bit register consisting of mode word and command word. Mode word specefies the general characteristics of operation, such as baud rate, parity bits, no. of parity bit, command word enables the data transmission and reception.

(i) ~~Status~~ Status register:
It checks the status of the peripheral.

(ii) Data register:
It is used as an input and output port when the C/$\overline{D}$ is low.

3. MODEM control:
It consists of 4-signals given by:

(a) $\overline{DSR}$ (Data Set Ready):
It checks whether the data set is ready when communicating with MODEM.

(b) $\overline{DTR}$ (Data terminal Ready):
This indicates device is ready to accept data when 8251 is communicating with MODEM.

(c) $\overline{CTS}$ (clear to send):
This is an input terminal for MODEM interface. Data is transmittable if the terminal is at low level.

(d) $\overline{RTS}$ (Request to send):
This is an output port for MODEM interface. The receiver is ready to receive the data when the terminal is at low level.

4. Transmitter Section:
The transmitter accepts parallel data from the mpu and converts into serial data. It has two register: a ~~buffer~~ bufer register to hold eight bits and an

output register to convert eight bits into a stream of serial bits. The mpu writes a byte in the buffer register.

whenever the output register is empty; the contents of buffer register are transferred to output register.

whenever a character is to be transmitted, that character must be placed inside the transmit buffer which is to be shifted to output register from where it transmit the data bit by bit using TXD pin.

Transmitter section consists of three output and one input signal.

* TXD (Transmitted Data output):
output signal to transmit the data to peripheral.
* $\overline{TXC}$ (transmitter clock @ input):
Input signal to control the rate of transmission.
* TXRDY (Transmitter Ready):
output signal to indicate that the 8251 is ready to accept the next data type.
* TXE (Transmitter empty):
output signal to indicate the output register is empty and the 8251 is ready to accept the next data byte.

5. Receiver section:
The Receiver accepts serial data on the RXD line from a peripheral and converts them into parallel data. This section has two registers the receiver input register and the buffer register.

Receiver section receives data bit by bit on RXD line in the input register.

The input register converts the serial data into parall-

el data and transfer to the receiver buffer register. when the data byte is transferred from the input register to receiver buffer register, the control logic generates a signal RxRDY to signal processor about the aviability of data byte to be read by processor.

RxRDY (Receiver Ready output):
This signal goes high when the USART has a characte in the buffer register and is ready to transfer it to the MPU.

RxD: (Receive Data input):
Clock signal that controls the rate at which bits are received by the USART.

SYNDET/BD:
It is taking Synchronous data indicating the da baud rate.

OR Short answer:

working of 8251

* Transmitter section
- Receives parallel data from mpu.
- Converts them into serial data.
- Transmits serially on TXD line.

* Receiver section.
- Receives serial data on RxD
- Converts serial data into parallel.
- Transfer parallel data to MPU.

methods of parallel Data transfer:

1. simple I/O:

To get digital data from a simple switch into a µP, the switch is connected to input port lines from which port can be read.

The data is always present and ready so that it can be read at any time.

To output data to an output device, the input of the device is connected to an output port pin. The output device is always present and ready so that data can be sent at any time.

fig: Timing diagram of simple IIo.

2. Simple Strobe IIo:

A strobe is a signal that is sent that validates data or other signals an adjacent parallel lines.

The sending devices outputs parallel data on the data lines and then outputs strobe signal to represent the valid data is present $\overline{STB}$.

fig: Timing diagram of Strobe IIo.

3. Single HandShake IIo Data transfer:

The peripheral outputs some parallel Data and sends $\overline{STB}$ signal to MPU. The MPU detects $\overline{STB}$ signal and reads the data. Then the mpu sends an Ack signal to the peripheral to indicate that the data has been read and the peripheral can send next byte of data.

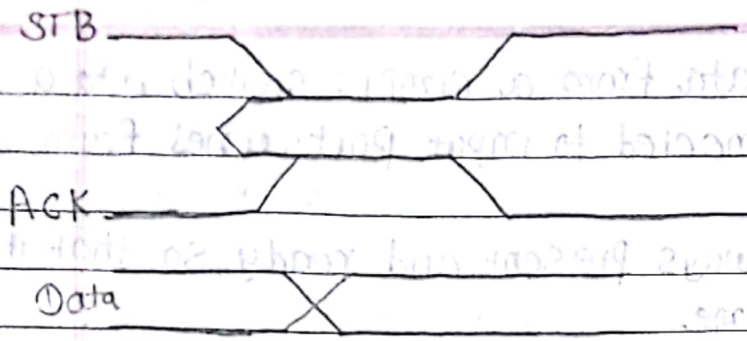


fig: Timing diagram of single handshake I/o.

4. Double Handshake I/o Data Transfer.

Input Handshake (peripheral to µP):

→ Peripheral asserts Strobe line low to ask the receiving device whether it is ready or not for data reception.

→ The receiving system raises PTS ACK line high indicating it is ready.

→ The peripheral device sends the byte of data and raises its strobe line high.

output Handshake (µP to Peripheral):

- µP Sends a strobe signal and data.
- Peripheral sends ACK signal.

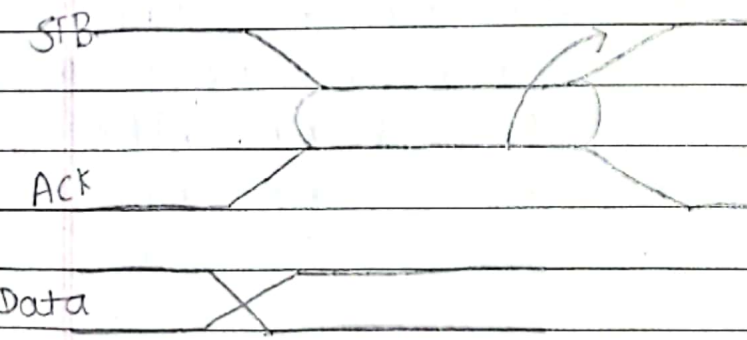


fig: Timing diagram of double Handshake I/o

I/O interface:

Parallel communication:

Parallel data transmission sends multiple data bits at the same time over multiple channels. The parallel transmission is used for shorter distance. parallel trans-

mission is faster as the data is transmitted using multiple lines. Parallel transmission is half duplex since the data is either sent or received. Parallel transmission is unreliable and complicated.

Parallel transmission is used when:

→ a large amount of data is being sent.

→ the data being sent is time sensitive and the data needs to be sent quickly.

A scenario where parallel transmission is used to send data is video streaming. when a video is streamed to a viewer, bits need to be received quickly to prevent a video pausing or buffering. video streaming also requires the transmission of large volume of data.

Serial communication:

Serial data transmission sents data bits one after another over a single channel. Serial transmission is normally used for long distance data transfer. It is also used in cases where the amount of data being sent is relatively small. It ensure that data integrity is maintained as it transmits the data bits in a specific order, one after another. In this way data bits are received in sync with one another. Serial communication is popular because most computers have one or more serial ports, so no extra hardware is needed other than a cable to connect the instrument to the computer or two computers together.

Difference Between Serial and parallel transmission.

	Serial transmission	Parallel transmission
1.	It requires a single line to communicate and transfer data.	It requires multiple lines to Communicate and transfer data.

2.	It is used for long distance communication.	It is used for short distance communication.
3.	Error and noise are least in Serial transmission.	Error and noise are more in parallel transmission.
4.	It is slower as data is transmitted through a single line or wire.	It is faster as data is transmitted over multiple lines.
5.	It is full duplex as the sender can send as well as receive the data.	It is half duplex since the data is either sent or received.
6.	Serial transmission cables are thinner, longer and economical.	Parallel transmission cables are thicker, smaller and non-economical.
7.	It is reliable and straight forward	It is unreliable and complicate.

Difference between Synchronous and Asynchronous data transmission.

	Synchronous data transmission	Asynchronous data transmission
1.	Two signals are used i.e clock pulse and data signals.	only one signal is used that is data signal.
2.	Sender and receiver should have Synchronized clock before transmission.	Doesnot required a clock but it adds a parity bit to the data before transmission.
3.	External clock is used.	Internal clock are used.
4.	Sender and Receiver should ~~be~~ have common clock.	sender and receiver have Separate clock.
5.	Data is transferred in the form of block or frames.	only one character is transmitted at a time.
6.	It is used for high speed transmission.	It is used for low speed transmission.
7.	It is expensive because more hardware is required.	It is cheaper because less hardware is needed.

Q. What is MODEM:

A modulator-demodulator is a hardware device that converts data from a digital format, intended for communication directly between devices with specialized wiring into one suitable for a transmission medium such as telephone lines or radio. A modem modulates one or more carrier wave signals to encode digital information for transmission, and demodulates signals to decode the transmitted information. The goal is to produce a signal that can be transmitted easily and decoded reliably to reproduce the original ~~data~~ digital data.

Modems can be used with almost any means of transmitting analog signals from light-emitting diodes to radio. A common type of modem is one that turns the digital data of a computer into a modulated electrical signal for transmission over telephone lines, to be demodulated by another modem at the receiver side to recover the digital data.

8255 TTL:

Features:

1. It is a programmable general purpose I/O device.
2. It has ~~three~~ 8 3 8-bit bidirectional I/O ports; Port A, Port B, Port C.
3. It provides 3 modes of data transfer; simple I/O, Handshake I/O and Bidirectional Handshake.
4. Additionally it also provides a ~~bit set~~ Bit SET RESET (BSR) mode to alter individual bits of Port C.

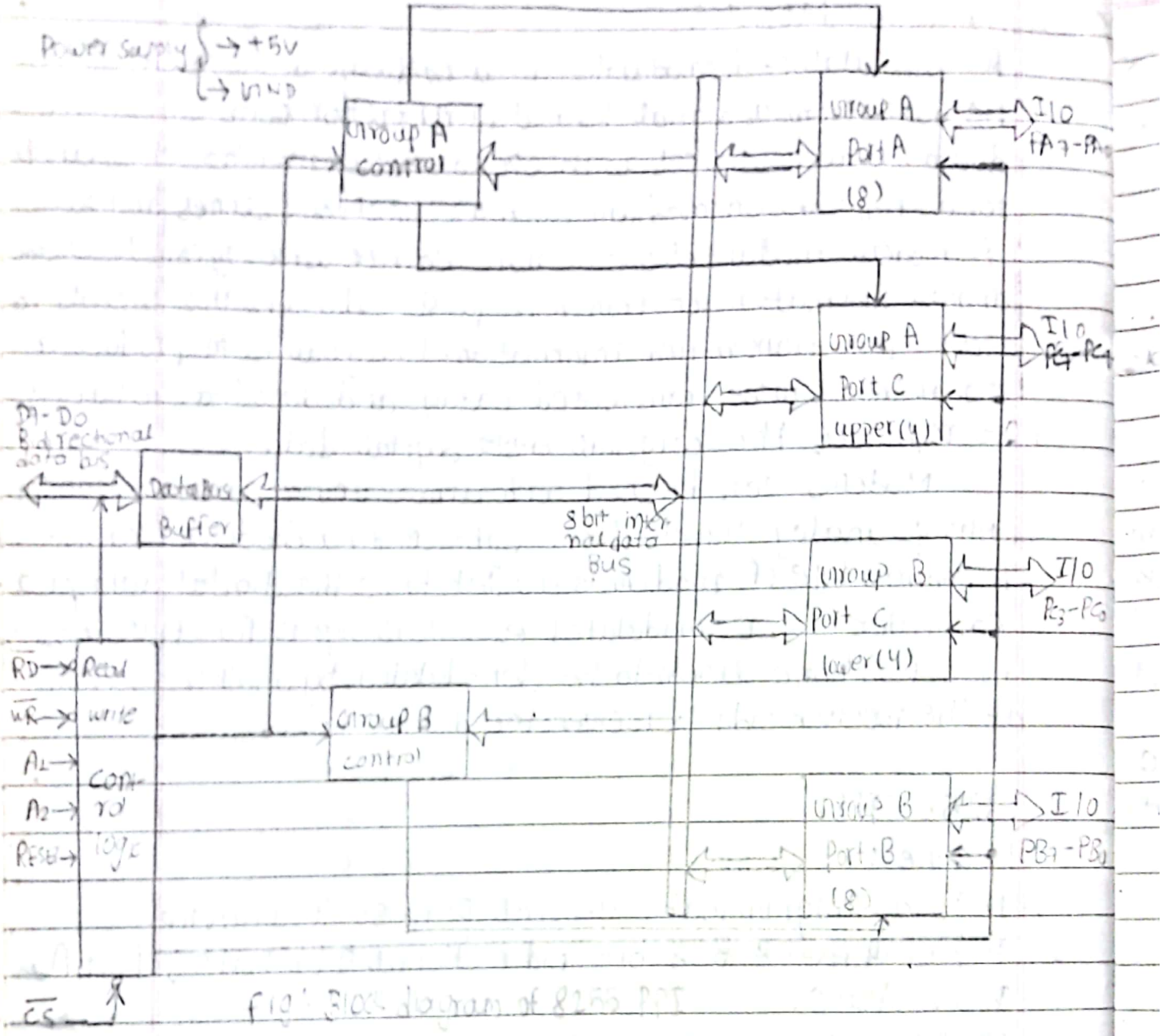


Fig: Block diagram of 8255 PIT

The architecture of 8255 can be divided into following parts

1. Data Bus Buffer:
This is a ~~bidire~~ 8-bit bidirectional buffer use to interface the ~~inte~~ internal data bus of 8255 with the external (system) data bus. The µP transfers data to and from the 8255 through this buffer.

2. Read write control logic:
It accepts address and control signals from µP the control

signals determine whether it is a ready or read or write operation and also selects or the 8255 chip.
The address bits (A_1, A_0) are used to select the port or the control word register

A_1	A_0	selection
0	0	Port A
0	1	Port B
1	0	Port C
1	1	control word

The ports are controlled by their respective group control registers.

3. Group A control:
This control block controls port A and port C upper i.e PC_7-PC_4. It accepts control signals from the control word and forwards them to the respective ports.

4. Group B control:
This control block controls port B and port C lower i.e PC_3-PC_0. It accepts control signals From the control word and forward them into respective ports.

5. Port A, Port B, Port C:
These are 8 bit bidirectional ports. They can be programm to work in the various modes as follows.

Port	Mode 0	Mode 1	Mode 2
Port A	Yes	Yes	Yes
Port B	Yes	Yes	No
Port C	Yes	No	No

Modes of operation:

1. Mode 0:
When program for mode 0, the PPI offers 3 simple I/O ports with no handshaking signals port A and port B use as ~~two~~ 2 simple 8-bit I/O ports., Port C is used as 2 simple 4 bit I/O ports. Each port can be programmed as input or output individually. Ports do not have handshake or interrupting capability. Hence slower devices can not be interfaced.

2. Mode 1:
In mode 1 handshake ~~sing~~ signals are exchange between the devices ~~&~~ before the data transfer takes place. Port A and Port B used as ~~2. 8 bit~~ two 8-bit I/O ports. That can be programmed in input or in output mode. Each port uses 3 lines from Port C for handshake. The remaining lines of Port C can be used for simple I/O. Interrupt ~~&~~ driven data transfer is possible hence slower devices can be interfaced.

3. Mode 2:
only Port A can be initialized in mode 2, Port A can be used as bidirectional handshake data transfer. This means that data can be outputed/inputed from same 8 lines

RS-232 C:
In RS232C RS stands for 'recommended Standard' RS232 is a serial ~~combination~~ communication ~~used for~~ Protocol used for transferring and receiving

the serial data between two devices. It defines the serial communication using DTE and DCE signals. Here DTE reffers to Data Terminal Equipment and DCE reffers to Data Communication Equipment. Example of DTE device is a computer and DCE is a MODEM. Formally it is specified as the interface between DTE and DCE ~~using ser~~ using serial binary data exchange.

RS232 cable is used to identify the difference of two signals level between logic 1 and logic 0. Logic 1 is represented by -12V and logic 0 is represented by +12V. The RS232 cable ~~worth~~ works at different baud rates like 9600 bits per second, 2400 bits per seconds, 4800 bits per seconds etc.

- 8086 microprocessor:

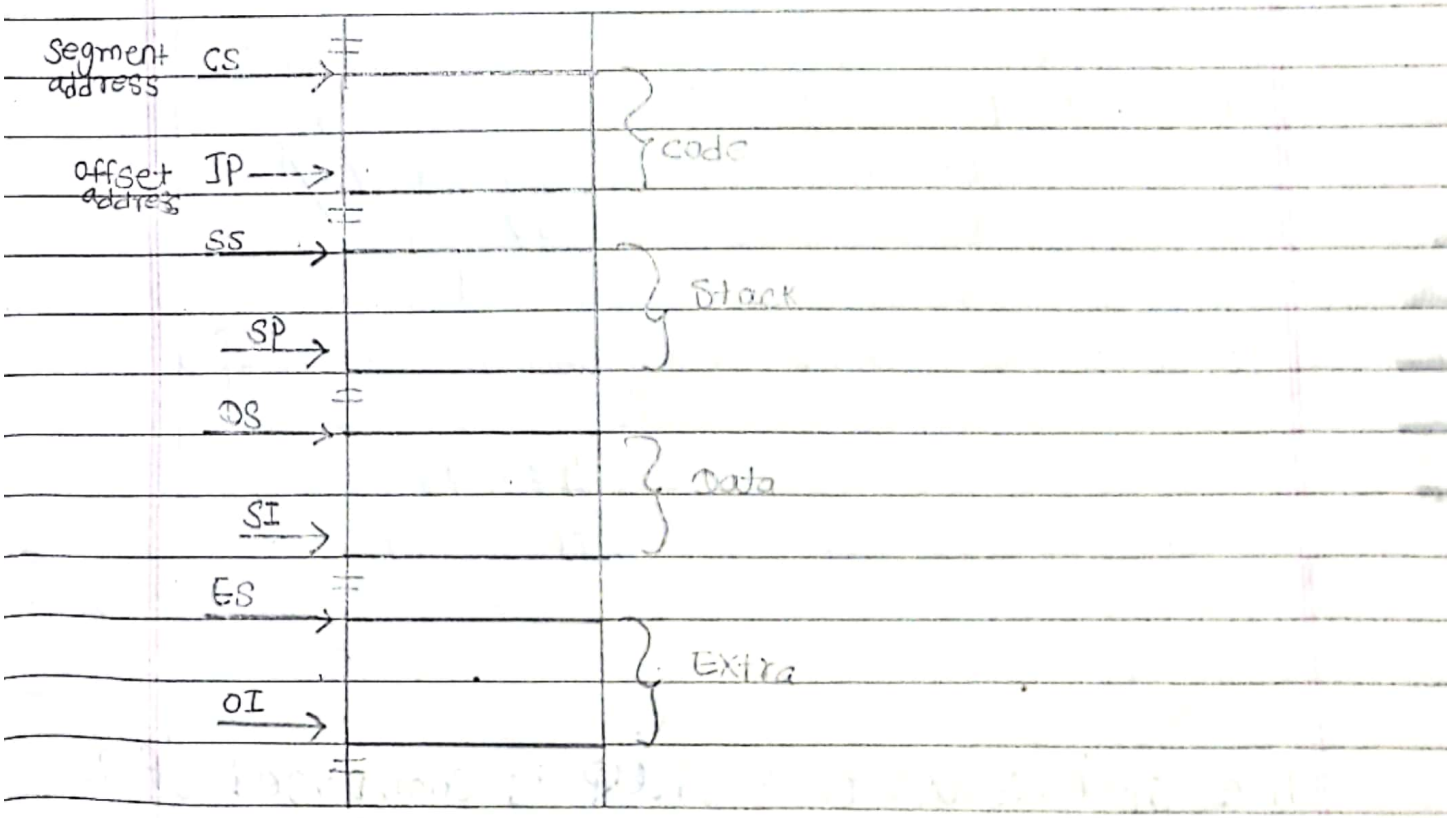


fig: Memory

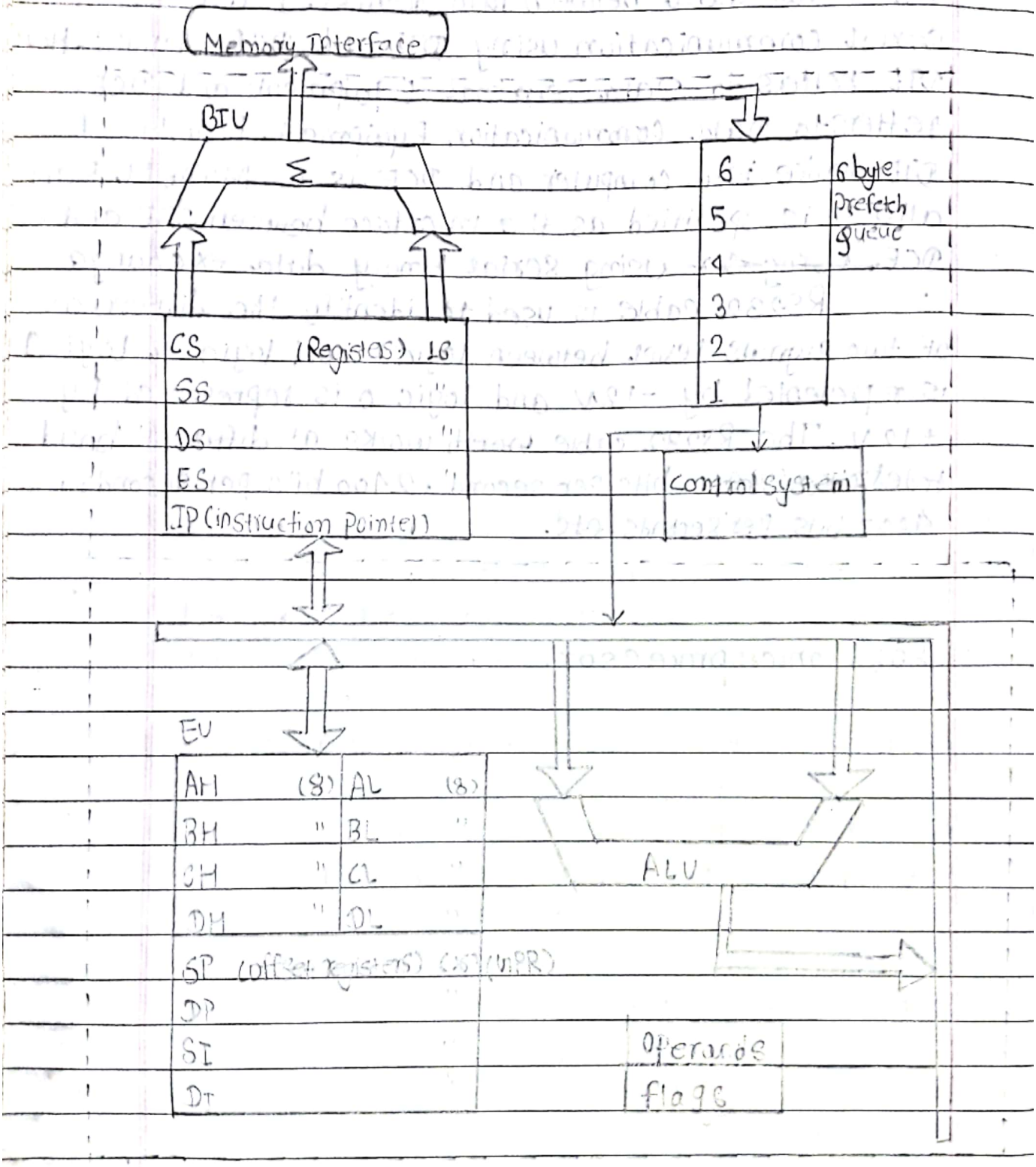


fig: 8086 microprocessor

The architecture of 8086 µP is composed of 2 major units.

1. Bus interface unit (BIU)
2. Execution unit (EU)

1. Bus interface unit (BIU):

The BIU manages the data, address and control buses. The BIU functions in such a way that it

* Fetches the sequenced instruction from the memory.
* Finds the physical address of that location in the memory where the instruction is stored
* Manage the 6-byte prefetch queue where the pipelined instructions are stored.

An 8086 µP exhibits a property of pipelining the instructions in a queue while performing decoding and execution of the previous instruction. That is saves the processor time of operation by a large amount. This pipelining is done in a 6 byte queue.

Also the BIU contains four segment registers. Each segment register is of 16-bit. The segment are present in the memory and these registers held the address of all the segments. These registers are as follows.

(a) Code segment register.
(b) Stack segment register.
(c) Data segment register
(d) Extra segment register

6-byte prefetch queue:

This queue is used in 8086 in order to perform pipelining. At the time of decoding an execution of the instruction in execution unit the BIU fetches the sequential upcoming instructions and store it in this queue. The size of this queue is 6 byte. This means at maximum 6-byte instruction can be stored in this queue.

2. Execution unit:
The execution unit performs the decoding and execution of the instructions that are being fetched from the desired memory location.

Control unit:
The control unit in 8086 µp produces control signal after decoding the opcode to inform the general purpose register to release the value stored in it. It also signals the the ALU to perform the desired operation

ALU:
The arithmetic and logic unit carries out the logical task according to the signal generated by the CU. The result of the operation is stored in the desired register.

Chapter 7: Advanced Microprocessor:

80286 Microprocessor:
The 80286 is an advanced high performance µp with special optimized capabilities for multiple users and multitasking systems. The 80286 has builtin memory protection that supports operating system and task isolation as well as program and data privacy within task.

In real address mode, the 80286 can address upto 1Mb of Physical memory. In virtual address mode, it can address upto 16Mb of physical memory address space and 1gb of virtual memory address space.

Features of 80286:
1. The intel 80286 is a 16-bit processor, it has the first CPU to incorporate the MMU (Memory management unit).

2. It can be operated at different clock speeds 4MHz, 6MHz and 8MHz.
3. It supports a pipelined Architecture: The pipelined architecture improves the performance of the 80286 processor.
4. The 80286 supports 2 operating modes: Real address mode and Protected Virtual Address Mode (PVAM).

Internal Block diagram of 80286:

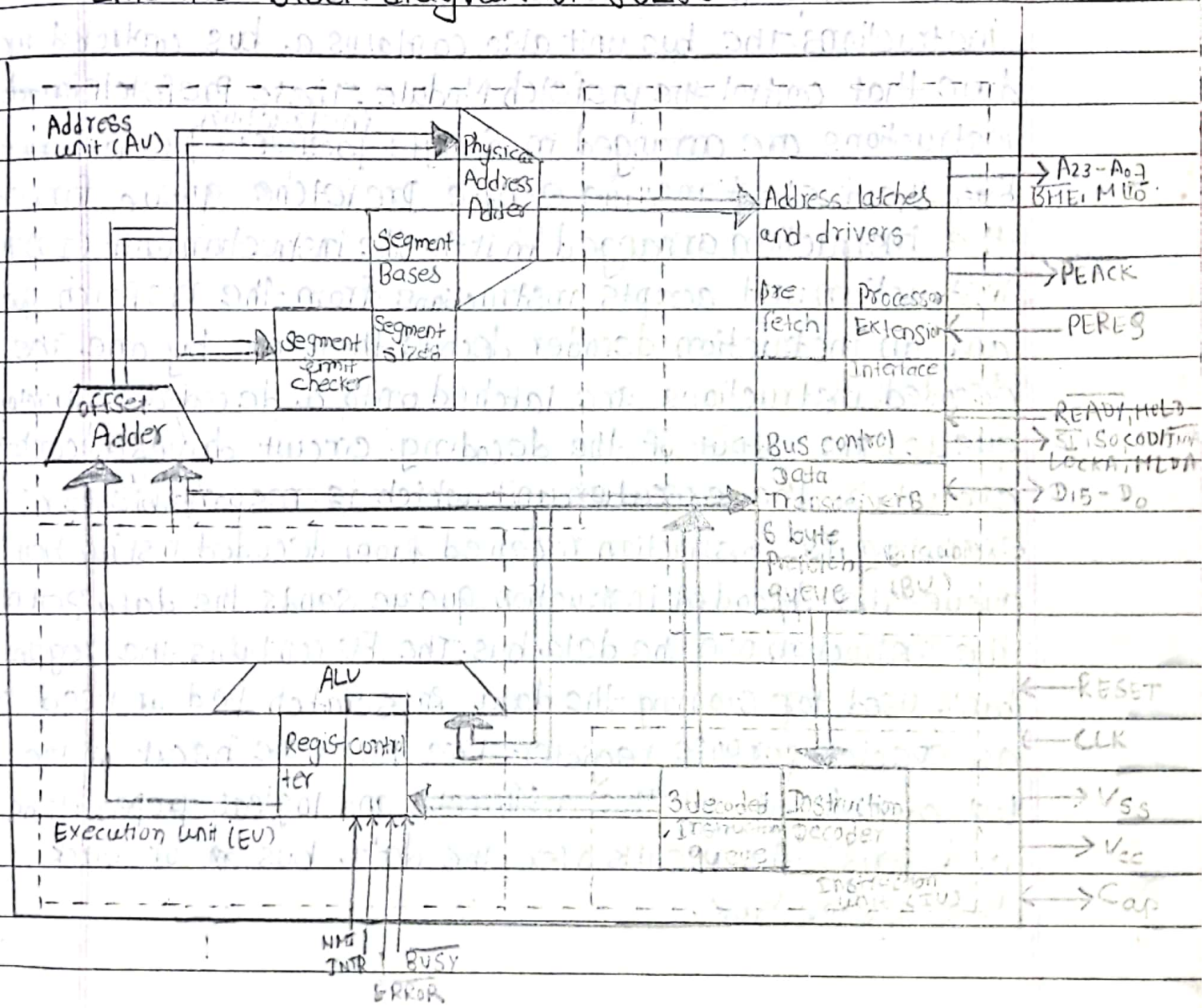


fig: Block diagram of 80286 microprocessor

The CPU contains four fundamental blocks. They are

1. Address unit (AU)
2. Bus unit (BU)
3. Instruction unit (IU)
4. Execution unit (EU)

The address unit is responsible for calculating the physical address of the data that the CPU wants to access. Also the address lines derived by this unit may be used to access different peripherals. The physical address is compuled by the address unit is handed over to the bus unit (BU) of th CPU. Major function of the bus unit is to fetch instruction bytes from the memory. Instructions are fetched in advance and stored in aqu queue to enable faster execution of the instructions. The bus unit also contains a bus controled module that control the prefetch Module. These prefetched instructions are arranged in 6-byte instruction prefetche queue. ~~forwards the instructions~~ The 6-byte prefetche queue forwards the instruction arranged in it to the instruction unit (IU). The instruction unit accepts instructions from the prefetch queue and an instruction decoder decode them one by one. The decoded instructions are latched onto a decoded instruction queue. The output of the decoding circuit drives a control circuit in the execution unit which is responsible for executing the instruction received from decoded instruction queue. The decoded instruction queue sends the data part of the instruction over the data bus. The EU contains the register bank used for storing the data & scratch pad or used as special purpose register. The ALU, the heart of the EU carries out all the arithmetic and logical operations and sends the results over the data bus or back to the register bank.

Real Address Mode:

80286 addresses only 1M bytes of physical memory using A_0-A_{19}. The lines A_{20} - A_{23} are not used by the internal circuit of 80286 in this mode. In real address mode, while addressing the physical memory, the 80286 uses BHE (Bus High Enable) (It indicates that there is a transfer on the higher byte of the data bus) along with A_0-A_{19}. The 20 bit Physical address is again form in the same way as that in 8086.

The content of segment registers are used as segment base address. The other registers depending upon the addressing mode contains the offset address. Because of extra pipelining and other circuit level improvements, the 80286 operats at a much faster rate than 8086 although functionally they work in an identical fashion.

An exception is generated if the segment size limit is exceeded by the instruction or data. In real mode the first 1K byte of memory starting from address 0000H to 003FFH is reserved for interrupt vector table. Also the address from FFFF0H to FFFFFH are reserved for System initialization.

The program execution starts from FFFFH after Reset and initialization. When the 80286 is Reset, it always starts the execution in real address mode.

The preparation for entering the protected virtual address mode occurs in real mode or Real address mode.

Protected virtual address Mode: (PVAM)

The 80286 is the first processor to support the corrept of virtual memory and memory management. The segment of the program or data required for actual execution at that instant is fetched from the secondary memory into Physical memory. After the execution of this fetchedsegment the next segment required for further execution is again fetched from the secondary memory, while the result of the executed segment are stored back into the secondary memory further references. This ~~continues~~ continues till the complete program is executed. The procedure of fetching the chosen program segment from the secondary storage into physical memory is called swaping.

The 80286 is able to address 1G byte of virtual memory per task. The complete virtual memory is mapped onto the 16M byte physical memory. If a program larger than 16M byte is stored on the harddisk and is to be executed, it is fetched in terms of data or program segments of less than 16M byte in size into the program memory by swaping sequentially as per sequence of execution.

The 80286 uses the 16 bit content of a segment register as a selector to address or discriptor stored in the physical memory. The discriptor is a Block of contiguous memory locations containing information of a segment like segment base address, segment limit, segment type, privilaged level, Segment availability in physical memory, discriptor type etc.

Multitasking: Multitasking has the same meaning of multiprogramming but in a more general sense as it refers to having multiple (programs, processes, tasks, threads) running at the same time. This term is used in modern operating system when multiple task share a common processing resource (e.g. CPU and Memory). At any time the CPU is executing one task only while other tasks waiting their turn. The illusion of parallelism is achieved when the CPU is reassigned to another task (i.e. process or thread context switching).

There are subtle difference between multitasking and multiprogramming. A task in a multitasking operating system is not a whole application program but it can also refer to a "thread of execution". when one process is divided into sub-tasks. Each smaller task does not hijack the CPU until it finishes like in the older multiprogramming but rather a fair share amount of the CPU time called quantum. Just to make it easy to remember, both multiprogramming and multitasking operating systems are (CPU) time sharing systems. However, while in multiprogramming (older OSs) one program as a whole keeps running until it blocks, in multitasking (modern OSS) time sharing is best manifested because each running process takes only a fair quantum of the CPU time.

~~Privilege level~~: Privilege level:

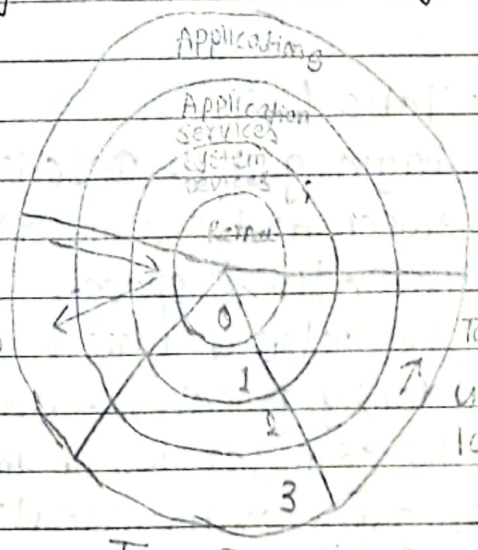


fig: Illustration of privilege level

There are four types of privilege levels

1. 00 - Kernel level (highest)
2. 01 - OS Services
3. 10 - OS extension (Operating System)
4. 11 - Applications (lowest).

→ Each task is assigned a privilege level which indicates the priority or privilege of that task.
→ It can only be changed by transferring ~~using~~ the control ~~by~~ using gate descriptors to a new segment.
→ A task executing at level zero most privileged level can access all the data segment defined in GDT and ~~GDT~~ LDT of the task.
→ A task executing at level three (the least privilege level) will have the most limited access to data and other descriptors.
→ The use of rings allows for System software to restrict task from accessing data.
→ In most enviroment the operating system and some device drivers run in ring zero and applications run in run 3.

GDT and LDT:

→ Global Descriptor Table (GDT):
The 80286 has a single global Descriptor Table (GDT) which is shared between all tasks and addresses upto 512 MB of the virtual address space.
→ The Global Descriptor Table or GDT is a data structure used by intel x86-family processor starting with the 80286 in order to define the characteristics of the various memory areas used during program execution including the base addres the size and access privileges like executeability and write-ability.

Local Descriptor Table (LTD):

→ Each task will have its own local Descriptor table (LTD) which is a private 512MB of address space.

→ LDT is essential to implement separate address spaces for multiple procels.

→ The operating system will switch the current LDT when Scheduling a new procels, using the LDT machine instruction

Descriptor Table:

* IDT used to store interrupt gates and task gates.
* LIDT instruction is used to load interrupt Descriptor table,

Difference between LDT and GDT:

→ LDT is actually defined by a descriptor inside the GDT, while the GDT is directly defined by a linear address.

→ The lack of symmetry between both tables is underlined by the fact that the current LDT can be automatically switched on the certain events, notably if TSS-based multitasking is used, while this is not possible for the GDT.

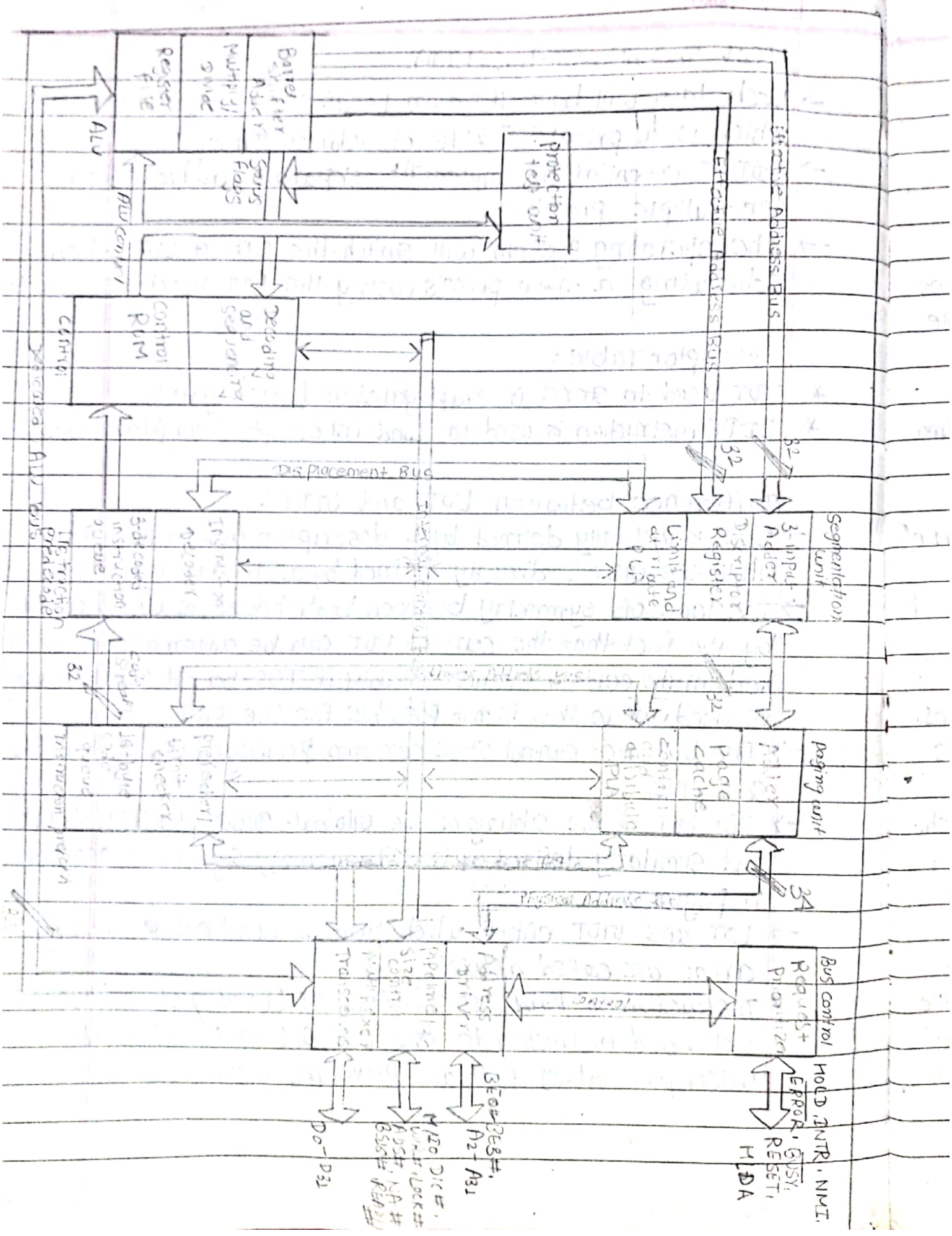
→ The LDT also cannot store certain privileged types of memory segments.

→ The LDT is the sibling of the Global Descriptor Table (GDT) and similarly defines upto 8191 memory segments accessible to programs.

→ LDT and GDT entries which point to identical @ memory areas are called aliases.

→ Instruction to load GDT is LGDT (Load global Descriptor Table) and instruction to bad LDT is LLDT (Load Global Descriptor table). Both are privileged instruction.

Fig: Block diagram of 80386 cpu



The internal architecture of 80386 is divided into three section.

1. central processing unit.
2. Memory management unit
3. Bus interface unit.

1. Central processing unit is further divided into Execution unit and instruction unit. Execution unit has 8 general purpose and 8 special purpose registers which are either used for handling data or calculating offsets addresses.

Instruction unit:

The instruction unit decodes the ~~off~~ opcode bytes receive from the 16 byte instruction code queue and arranges them in a three instruction decoded instruction queue. After decoding them passes it to the control section for deriving the necessary control signals. The barrel ~~seets~~ shifter increases the speed of all shift and rotate operations. The ~~multip~~ multiply / Divide ~~clos~~ logic implements the bits shift rotate algorithm to complete the operation in minimum time even 32 bit multiplication can be executed within 1 microseconds by the multiply / Divide logic. The protection test unit checks for segmentation ~~Voi~~ violation under the control of the micro code.

Execution unit:

The Execution unit read the instruction from the instruction queue and executes the instruction. It consist of 3 sub unit:

(i) Control unit
(ii) Data unit
(iii) Protection test unit.

2. Memory management unit:
The MMU consist of segmentation unit and a paging unit. Segmentation unit allows the use of two address components which are segment and offset for relocatable and sharing of code and data. Segmentation unit allows segments of size of 4GB at max.

The paging unit organizes the physical memory in terms of pages of 4KB size each. Paging unit works under the controls of segmentation unit that is each segment is further divided into pages. The virtual memory is also organized in term of segments and pages by the MMU.

3. Bus interface unit:
The Bus unit is the interface to the external devices. The BIU provides a 32 bit data Bus, A 32 bit address bus and the signals needed to control transfer over the bus. It accepts the internal request for code fetch, for data transfer from the code fetch unit and from the execution unit. It then prioritizes the request with the help of request prioritizer and generate Signal to perform bus cycle. The address driver drives the bus enable and address signal $A_0 - A_{31}$ and the transceiver interface the internal databus with the system bus. It controls the interface to the external bus masters and coprocessors.

Paging:

Paging is one of the memory management technique used for virtual memory multitasking operating system. The segmentation scheme may divide the physical memory into a variable size segments but the paging divides the memory into a fixed size pages. The segments are supposed to be the logical segment of the program, but the pages do not have any logical relation with the program. The pages are just fixed size portions of the program module or data.

The advantages of paging scheme is that the complete segment of a task need not be in the physical memory at any time. only a few pages of the segments, which are required currently for the execution need to be available in the physical memory. Thus the memory requirement of task is substantially reduced, relinquishing the available memory for other task. Whenever the other pages of task are required for execution, they may be fetched from the secondary storage. The Previous pages which are executed, need not be available in the memory, and Hence the space occupied by them may be relinquished for other tasks. Thus paging mechanism provides an effective technique to manage the physical memory for multitasking systems.

Pipeling:

pipeling is a technique of decomposing a sequential process into sub-operation with each sub process being

executed in a special dedicated segment that operates concorrently with all other segment. Each segment performs Partial processing detected by the way the task is Partition. in each segment is transferred to the next segment in the pipeling. The final result is obtained after the data have passed through all segment. The name "pipeline." implies a flow of in form a in analogous to an industrial assembly line. The overlapping of computation is made possible by associating a register with each segment in the pipeline. The registers provide isolation between each segment so that each can operate on distinct data simultaneously.

Pipelining is an implementation technique where multiple instruction are overlapped in execution. The computer pipelime is divide into stages. Stage computes a part of instruction is parallel. The stages are connected one to the next to form a pipe-instruction enter at one end progress through the stages and exist at other end

Advantages of Pipeline:

1. The execution unit always reads the next instruction byte from the queue in BIU. This is faster than sending out an address to the memory and waiting for the next instruction byte to come.
2. In short pipelining eliminates the waiting time of EU and sends to the processing.

Descriptor:

→ Descriptor is a identifier of a program segment or page.

→ A segment cannot be accessed if its descriptor does not exist in either LDT or GDT

→ Set of descriptor arranged in a proper sequence describe the complete program.

→ The descriptor is a block of contiguous memory location containing information of a segment like

(i) Segment base address
(ii) segment limit
(iii) segment type
(iv) privilege level - prevents unauthorized access
(v) segment availability in physical memory
(vi) descriptor type
(vii) segment use by another task.

Requirement of Descriptor table:

→ The descriptor describes the location, length and access rights of the segment of memory.

→ The selector, located in the segment register, selects one of descriptors from one of two tables of descriptor

Q. write an assembly language Program to sort an array in ascending order using 8 bit microprocessor. (Assume appropriate array data and address where minimum array size of 10 should be considered.

```
    LXI H, 5000H
    MOV C, M
    DCR C
REPEAT: MOV D, C
    LXI H, 5001H
LOOP: MOV A, M
    INX H
    CMP H
    JC SKIP
    MOV B, M
    MOV M, A
    DCX H
    MOV M, B
    INX H
SKIP: DCR D
    JNC LOOP
    DCR C
    JNZ REPEAT
    HLT
```