1. Explain operating modes of PIC 8259

The PIC 8259 is a Programmable Interrupt Controller used in computer systems to manage interrupt signals from various devices. It has several operating modes that allow it to handle interrupt requests in different ways.

The operating modes of PIC 8259 are as follows:

1. Interrupt Request (IRQ) Mode:

In this mode, the PIC 8259 waits for an interrupt request from a device. When an interrupt request is received, it sends an interrupt signal to the CPU to indicate that an interrupt has occurred.

2. Polling Mode:

In this mode, the CPU polls the PIC 8259 to check if any interrupt request has occurred. If an interrupt request has occurred, the CPU responds to it accordingly.

3. Automatic End of Interrupt (AEOI) Mode:

In this mode, the PIC 8259 automatically clears the interrupt request after it has been serviced by the CPU. This eliminates the need for the CPU to send an acknowledgement signal to the PIC 8259.

4. Cascade Mode:

In this mode, multiple PIC 8259 controllers can be cascaded together to handle more interrupt requests. One PIC 8259 acts as the master and the others act as slaves. The master PIC 8259 sends interrupt signals to the CPU while the slave PIC 8259 sends interrupt signals to the master PIC 8259.

Each of these operating modes provides a different way of managing interrupt requests from various devices in a computer system. The appropriate mode is selected based on the specific requirements of the system.

2. Give formats of initialisation command words (ICW's) of 8259 PIC

The 8259 Programmable Interrupt Controller (PIC) is initialized through a series of command words known as Initialization Command Words (ICWs). There are four ICWs that are sent to the PIC during the initialization process. The formats of the ICWs are as follows:

1. ICW1:

This command word initializes the PIC and specifies the following:

a. Whether the PIC is a master or a slave. b. Whether the initialization is for level or edge-triggered interrupts. c. Whether the PIC is to operate in a single or cascaded mode.

The format of ICW1 is:

0bxxxxxxx01

Where the first 6 bits are don't cares, and the last two bits are set to 01.

2. ICW2:

This command word sets the base address of the interrupt vectors. The format of ICW2 is:

0bxxxxxxxx

Where the 8 bits represent the base address of the interrupt vectors.

3. ICW3:

This command word is only sent to the slave PIC if the master PIC is operating in cascaded mode. It specifies which IRQ line on the master PIC is connected to the slave PIC. The format of ICW3 is:

0bxxxxxxyz

Where x are don't cares, y is the IRQ line number on the master PIC that is connected to the slave PIC, and z is set to 0 for the master PIC and set to the bit position of the slave

PIC's IRQ line on the master PIC.

4. ICW4:

This command word specifies additional information about the operation of the PIC, including whether the PIC is operating in buffered or non-buffered mode, whether the interrupts are being acknowledged automatically or manually, and whether the PIC is in special fully nested mode or not. The format of ICW4 is:

0bxxxxyzabc

Where x is a don't care, y is set to 1 for buffered mode or 0 for non-buffered mode, z is set to 1 for automatic end-of-interrupt mode or 0 for manual acknowledgement mode, a is set to 1 for special fully nested mode or 0 for not, b is set to 1 for not allowing the CAS signal from the 8086/8088 to be used for level/edge selection, and c is set to 1 for not allowing the 8086/8088 to read ISR registers as 0s.

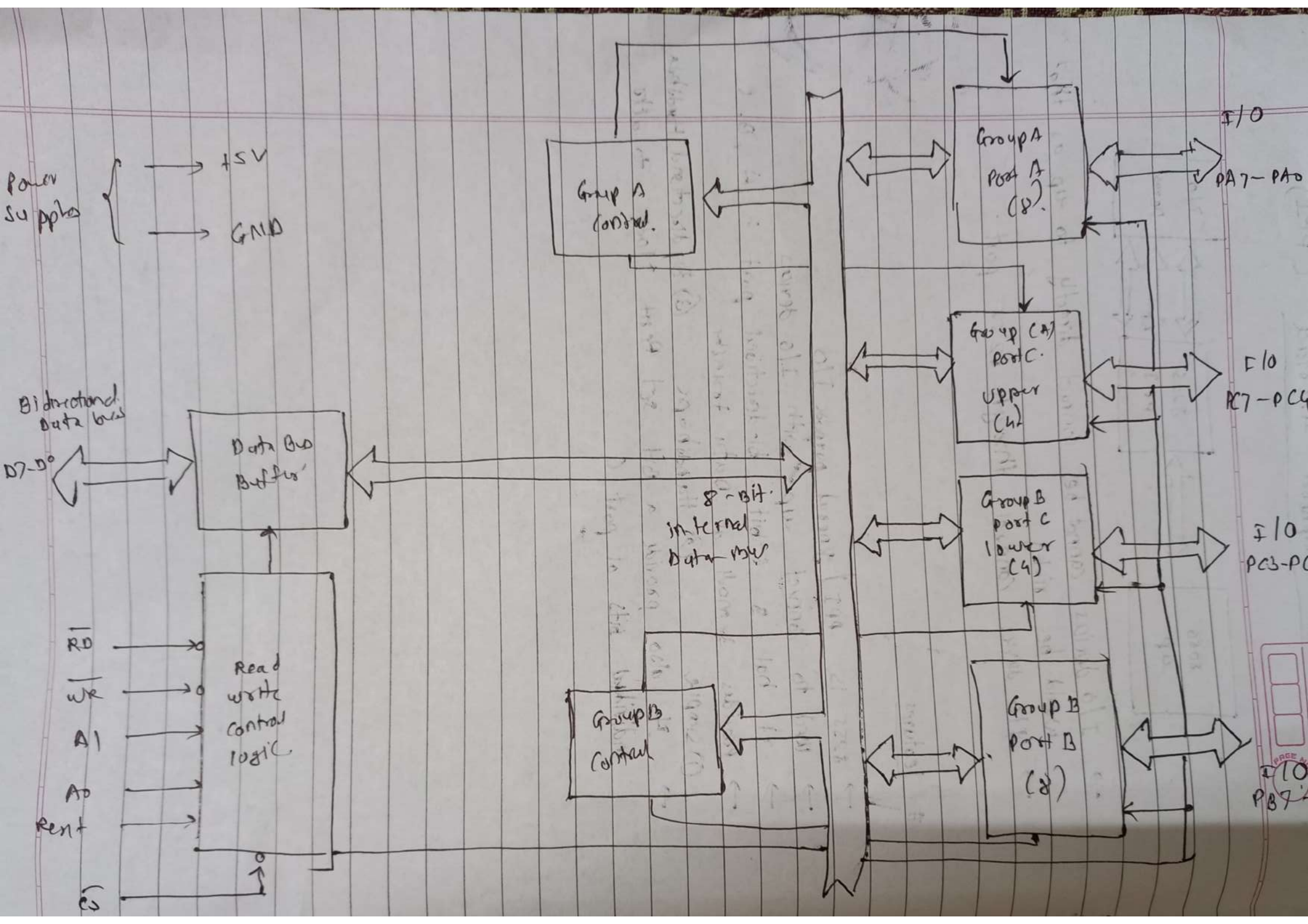
3. Draw and explain the block diagram of 8255 Programmable Peripheral Interface (PPI) with control word formats

The 8255 Programmable Peripheral Interface (PPI) is a versatile parallel I/O device that can be programmed to work in a variety of modes. It has three 8-bit I/O ports, Port A, Port B, and Port C, which can be configured as inputs or outputs.

The Control Logic block controls the operation of the 8255 PPI. It accepts commands from the CPU and generates control signals for the I/O ports. The control word is an 8-bit command that specifies the mode of operation for each port and the overall configuration of the device.

Here are the control word formats for each mode:

1. Mode 0 (Basic Input/Output):


Power Supplies
+5V
GND
Group A control
Group A Port A (8)
I/O
PA7 - PA0
Group (A) Port C upper (4)
I/O
PC7 - PC4
Bidirectional Data bus
D7 - D0
Data Bus Buffer
8-bit internal Data bus
Group B port C lower (4)
I/O
PC3 - PC
RD
WR
A1
A0
Reset
CS
Read write control logic
Group B control
Group B Port B (8)
I/O
PB7

This mode provides simple input and output operations on each port. The control word for this mode is:

0b0000000000

This sets Port A and Port B to be in input mode, and Port C to be in output mode.

2. Mode 1 (Strobed Input/Output):

This mode provides input and output operations for Port A and Port B, with Port C used for handshaking. The control word for this mode is:

0b0001000000

This sets Port A and Port B to be in input mode, and Port C to be in output mode with bit 0 acting as a strobe signal.

3. Mode 2 (Bidirectional):

This mode provides bidirectional data transfer between Port A and Port B, with Port C used for handshaking. The control word for this mode is:

0b0010000000

This sets Port A to be in input mode and Port B to be in output mode, with Port C to be in output mode with bit 0 acting as a strobe signal.

4. Mode 3 (Controlled Strobed Input/Output): This mode provides input and output operations for Port A and Port B, with Port C used for handshaking. The control word for this mode is:

0b0011000000

This sets Port A to be in input mode, Port B to be in output mode, and Port C to be in output mode with bit 0 acting as a strobe signal.

5. Mode 4 (Auto-Increment): This mode provides automatic incrementing of the address pointer for each read/write operation. The control word for this mode is:

0b0100000000

This sets Port A and Port B to be in input mode, and Port C to

be in output mode.

6. Mode 5 (Bi-Directional with Auto-Increment): This mode provides bidirectional data transfer between Port A and Port B, with Port C used for handshaking. The control word for this mode is:

0b0101010000

This sets Port A to be in input mode and Port B to be in output mode, with Port C to be in output mode with bit 0 acting as a strobe signal.

4. Explain the I/O mode control word format of 8255 PPI

The I/O mode control word format of 8255 PPI specifies the mode of operation for each of its three 8-bit I/O ports, Port A, Port B, and Port C. The control word is an 8-bit command that is loaded into the control register of the 8255 PPI to configure its operation.

The control word format is as follows:

bit 7: 0 (must be 0)

bit 6: 0 (must be 0)

bit 5: C3 mode selection for Port C bit 3 (1=output, 0=input)

bit 4: C2 mode selection for Port C bit 2 (1=output, 0=input)

bit 3: C1 mode selection for Port C bit 1 (1=output, 0=input)

bit 2: C0 mode selection for Port C bit 0 (1=output, 0=input)

bit 1: B mode selection (1=output, 0=input)

bit 0: A mode selection (1=output, 0=input)

The interpretation of the bits is as follows:

- Bits 5-2 specify the mode of operation for Port C. Each bit corresponds to a specific pin on Port C, with bit 5 corresponding to Pin 7, bit 4 corresponding to Pin 6, bit 3 corresponding to Pin 5, and bit 2 corresponding to Pin 4. A value of 1 sets the corresponding pin to output mode, and a value of 0 sets it to input mode.
- Bit 1 specifies the mode of operation for Port B. A value of 1

sets Port B to output mode, and a value of 0 sets it to input mode.

- Bit 0 specifies the mode of operation for Port A. A value of 1 sets Port A to output mode, and a value of 0 sets it to input mode.

By setting the appropriate bits in the control word, the user can configure the 8255 PPI for a variety of input and output operations. The 8255 PPI can be programmed to work in six different modes, each with its own set of control word settings.

5. Discuss control word format for Bit set Reset (BSR) mode of 8255 PPI

The Bit set Reset (BSR) mode of 8255 PPI allows the user to set or reset individual bits of Port C without affecting the other bits. The control word for BSR mode is an 8-bit command that is loaded into the control register of the 8255 PPI to specify the bit to be set or reset and the operation to be performed.

The control word format for BSR mode is as follows:

bit 7: 0 (must be 0)
bit 6: 0 (must be 0)
bit 5: 0 (must be 0)
bit 4: 0 (must be 0)
bit 3: 0 (must be 0)
bit 2: 0 (must be 0)
bit 1: BSR mode selection (1=BSR mode, 0=not BSR mode)
bit 0: Operation selection (1=set bit, 0=reset bit)

The interpretation of the bits is as follows:

- Bit 1 selects BSR mode when set to 1. In BSR mode, only the selected bit of Port C will be affected by the operation specified in bit 0.

- Bit 0 selects the operation to be performed on the selected bit. When set to 1, the selected bit will be set to 1, and when set to 0, the selected bit will be reset to 0.

To perform a BSR operation, the user must load the control word into the control register of the 8255 PPI and then write the bit number to the output latch of Port C. Writing a 1 to the selected bit in the output latch will set the bit if the Operation selection bit is set to 1, or reset the bit if the Operation selection bit is set to 0. Writing a 0 to the selected bit in the output latch will have no effect on the state of the bit.

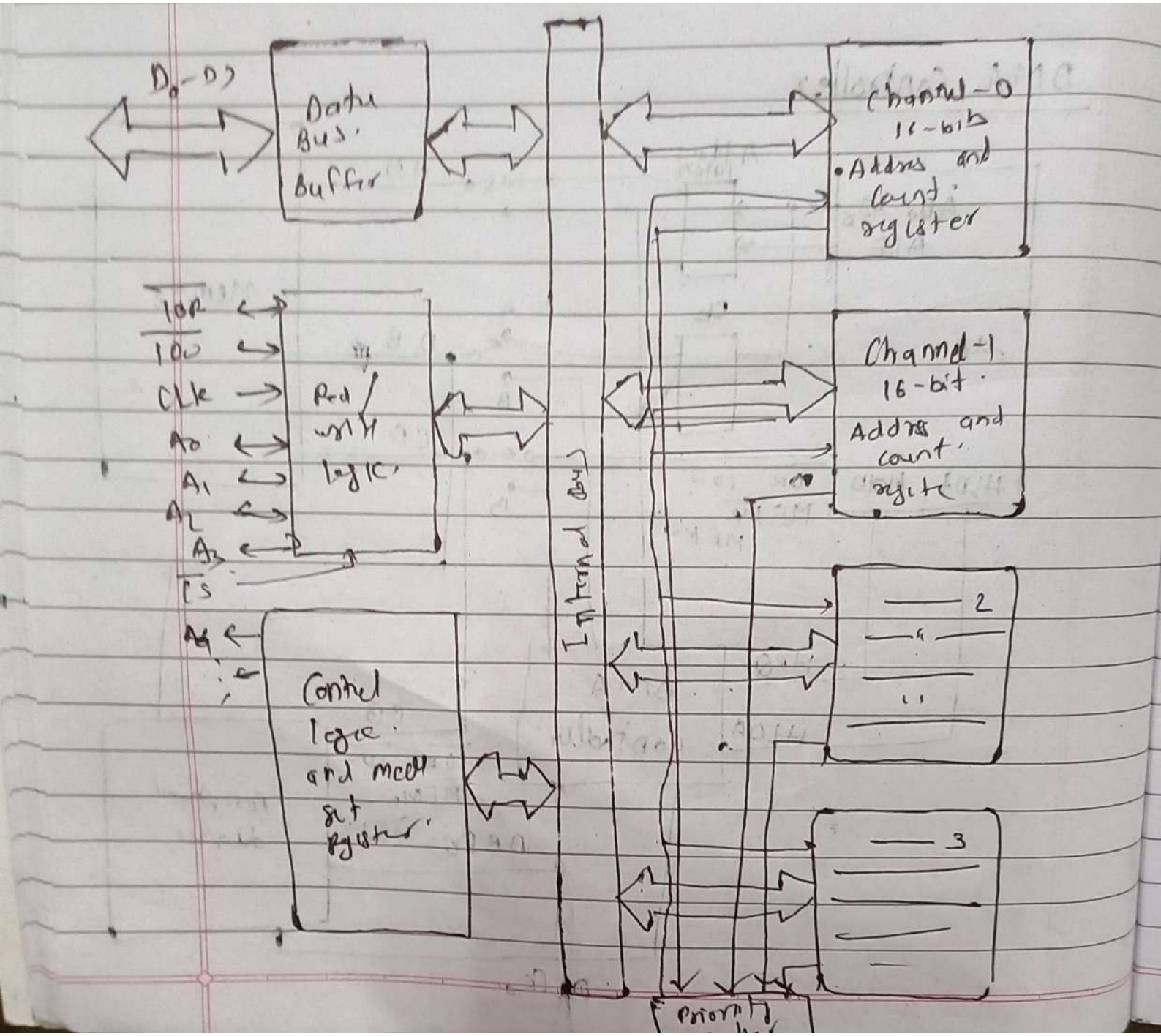
The BSR mode of 8255 PPI is useful in applications where individual bits of Port C need to be controlled independently, without affecting the other bits.

6. Draw and explain the block diagram of 8257 DMA controller

The 8257 DMA (Direct Memory Access) controller is an integrated circuit that provides high-speed data transfer between memory and I/O devices. It is designed to offload the CPU from the burden of data transfer, thus improving the overall system performance. The block diagram of the 8257 DMA controller is shown below:

The main components of the 8257 DMA controller are:

1. Control Logic: It controls the operation of the DMA controller and generates the necessary control signals for the DMA transfer. It also provides the interface between the DMA controller and the CPU.
2. Channel Controllers: The 8257 DMA controller can support up to four independent DMA channels, each with its own channel


$D_0 - D_7$
Data Bus Buffer
Internal Bus
Channel-0 16-bit • Addrs and count register
Channel-1 16-bit Addrs and count regis
2
3
IOR
IOW
CLK
A0
A1
A2
A3
CS
Rd/Wr logic
Control logic and mode set Register
Priority

controller. The channel controller manages the data transfer between the I/O device and the memory. It also generates the necessary timing signals for the DMA transfer.
3. DMA Request: The DMA request signal is used by the I/O device to request the DMA transfer. When the DMA request signal is received, the DMA controller takes control of the system bus and initiates the data transfer.
4. DMA Acknowledge: The DMA acknowledge signal is used by the DMA controller to acknowledge the DMA request signal from the I/O device. It also indicates the status of the DMA transfer to the I/O device.
5. Data Bus: The data bus is used to transfer the data between the I/O device and the memory. The DMA controller takes control of the data bus during the DMA transfer.
The 8257 DMA controller operates in three modes: single transfer, block transfer, and demand transfer. In the single transfer mode, a single block of data is transferred from the I/O device to the memory. In the block transfer mode, multiple blocks of data are transferred in a single DMA transfer. In the demand transfer mode, the DMA controller continuously transfers data between the I/O device and the memory until the transfer is stopped by the CPU. The mode of operation is determined by the control signals generated by the control logic.

7. Design 8086 based system for following specifications:
a. 8086 in minimum mode with clock frequency 5MHZ
b. 128 KB EPROM using 32KB
c. 32KB RAM using 16KB

8. Explain EFLAGS registers of 80386 DX
The EFLAGS (Extended FLAGS) register is a 32-bit register in the Intel 80386 DX microprocessor that contains the current

state of various processor flags. These flags are used to control the operation of the processor and to provide information about the results of arithmetic and logical operations. The EFLAGS register is an extension of the 16-bit FLAGS register found in earlier Intel processors.

The EFLAGS register consists of the following bit fields:

- Bit 0 (CF) Carry Flag: This flag is set if an arithmetic operation generates a carry out of the most significant bit, or if a shift or rotate operation generates a carry out of the least significant bit. It is also used for bit test and string instructions.
- Bit 1 (Reserved): This bit is reserved and always set to 1.
- Bit 2 (PF) Parity Flag: This flag is set if the low-order byte of the result of an arithmetic or logical operation contains an even number of 1 bits.
- Bit 3 (Reserved): This bit is reserved and always set to 0.
- Bit 4 (AF) Auxiliary Carry Flag: This flag is set if an arithmetic operation generates a carry out of bit 3 (the "half-carry" bit) of the result.
- Bit 5 (Reserved): This bit is reserved and always set to 0.
- Bit 6 (ZF) Zero Flag: This flag is set if the result of an arithmetic or logical operation is zero.
- Bit 7 (SF) Sign Flag: This flag is set if the result of an arithmetic or logical operation is negative.
- Bit 8 (TF) Trap Flag: This flag is used for debugging and is set to enable single-step mode. When set, the processor generates a trap after executing each instruction.
- Bit 9 (IF) Interrupt Flag: This flag is used to enable or disable external interrupts. When set, interrupts are enabled.
- Bit 10 (DF) Direction Flag: This flag is used to control the direction of string instructions. When set, string instructions decrement the address.

- Bit 11 (OF) Overflow Flag: This flag is set if an arithmetic operation generates a result that is too large to be represented in the result register.
- Bit 12-31 (Reserved): These bits are reserved and always set to 0.

In summary, the EFLAGS register provides important information about the state of the processor and the results of arithmetic and logical operations. It is used by the processor to control its operation and by programmers for debugging and error handling.

9. Explain V86 mode of 80386 DX

The V86 (Virtual 8086) mode of the 80386 DX processor allows multiple 8086-compatible virtual machines to run concurrently on a single physical machine. Each virtual machine appears to have its own set of hardware resources, including memory, I/O ports, and interrupts, but in reality, these resources are shared among all virtual machines by the underlying operating system.

In V86 mode, the processor emulates the behavior of an 8086 processor, allowing legacy 16-bit DOS applications to run on a 32-bit operating system without the need for a dedicated 8086 processor. The V86 mode supports both real-mode and protected-mode virtual machines, allowing 16-bit applications to access up to 1 MB of memory and protected-mode applications to access up to 4 GB of memory.

When a V86 virtual machine is created, the operating system sets up a separate address space for the virtual machine and maps the virtual machine's memory and I/O resources into this address space. The processor then switches to V86 mode and begins executing code in the virtual machine's memory space. When the virtual machine accesses an I/O port or generates an

interrupt, the processor traps to the operating system, which emulates the requested operation in a way that is transparent to the virtual machine.

In summary, the V86 mode of the 80386 DX processor provides a way to run legacy 16-bit applications on a 32-bit operating system without the need for dedicated hardware. It allows multiple virtual machines to run concurrently, each with its own set of hardware resources, and provides a mechanism for the operating system to emulate I/O operations and interrupts on behalf of the virtual machines.

10. Explain modes of operation of 80386 microprocessor

The 80386 microprocessor has three modes of operation: real mode, protected mode, and virtual 8086 mode.

1. Real Mode: In real mode, the processor behaves like a standard 16-bit microprocessor. The processor can address only 1 MB of memory, using 20-bit addresses. In this mode, there is no memory protection, no multitasking, and no virtual memory. All instructions and memory accesses are executed directly by the processor without any operating system intervention.

2. Protected Mode: In protected mode, the processor provides a mechanism for memory protection, multitasking, and virtual memory. The processor can access up to 4 GB of memory, using 32-bit addresses. In protected mode, the memory is divided into segments, each with its own access permissions and attributes. The processor also provides support for virtual memory, allowing the operating system to manage the mapping of physical memory to virtual memory.

3. Virtual 8086 Mode: In virtual 8086 mode, the processor provides a mechanism for running multiple 8086-compatible virtual machines on a single physical machine. Each virtual machine

appears to have its own set of hardware resources, including memory, I/O ports, and interrupts, but in reality, these resources are shared among all virtual machines by the underlying operating system. The processor emulates the behavior of an 8086 processor in this mode, allowing legacy 16-bit DOS applications to run on a 32-bit operating system without the need for a dedicated 8086 processor.

In summary, the 80386 microprocessor provides multiple modes of operation to support different types of applications and operating systems. Real mode is used for running legacy 16-bit applications, protected mode provides memory protection, multitasking, and virtual memory, and virtual 8086 mode provides a way to run multiple 8086-compatible virtual machines on a single physical machine.

11. Explain memory management of 80386 in detail

The 80386 microprocessor provides a sophisticated memory management system that allows the operating system to manage the allocation and protection of memory in a flexible and efficient manner. The memory management system consists of several key features, including virtual memory, segmentation, and paging.

1. Virtual Memory: The 80386 processor provides support for virtual memory, allowing the operating system to map the physical memory into virtual memory addresses. The processor provides a mechanism for translating virtual memory addresses into physical memory addresses using a page table, which is maintained by the operating system. Virtual memory allows the operating system to allocate memory to processes on demand, even if physical memory is not available.

2. Segmentation: The 80386 processor also provides segmentation, which allows the memory to be divided into

segments, each with its own attributes and permissions. Segmentation allows the operating system to protect memory by restricting access to specific segments of memory. The segment registers contain the segment base addresses, and the offset registers contain the offsets within the segments.
3. Paging: The 80386 processor also supports paging, which is an additional layer of memory management that maps virtual memory addresses to physical memory addresses. Paging is similar to segmentation, but instead of dividing memory into segments, it divides memory into fixed-size pages. The page table maps virtual memory pages to physical memory pages, allowing the operating system to efficiently manage large amounts of memory.
In addition to these features, the 80386 processor provides support for memory protection through the use of page-level protection bits and segment-level access rights. The processor also provides support for demand paging, which allows the operating system to load pages from disk into physical memory on demand.
Overall, the memory management system of the 80386 processor provides a powerful set of tools for managing memory efficiently and securely, allowing the operating system to allocate memory to processes as needed, protect memory from unauthorized access, and manage large amounts of memory efficiently.

12. Explain with neat diagram , address translation mechanism implemented on 80386 DX
The 80386 DX microprocessor implements a sophisticated address translation mechanism to support virtual memory. The address translation mechanism consists of several key components,

including the page directory, the page table, and the translation lookaside buffer (TLB).
Here is a block diagram of the address translation mechanism implemented on the 80386 DX:

1. Linear Address: The 80386 DX processor uses 32-bit linear addresses, which are generated by the CPU.
2. Paging Unit: The paging unit is responsible for translating the linear address into a physical address. The paging unit consists of a page directory and a page table.
3. Page Directory: The page directory is a table that contains entries for each page table in the system. Each entry in the page directory points to a page table.
4. Page Table: The page table is a table that contains entries for each page in the system. Each entry in the page table contains the physical address of the page and information about the page's protection and attributes.
5. Translation Lookaside Buffer (TLB): The TLB is a cache that stores recently used page table entries. The TLB can be accessed faster than the page table, so if a page table entry is found in the TLB, the translation can be performed more quickly.
6. Physical Address: The physical address is the final result of the address translation mechanism. Once the paging unit has translated the linear address into a physical address, the CPU can access the memory at that physical address.
In summary, the address translation mechanism implemented on the 80386 DX processor provides a powerful tool for managing virtual memory. The mechanism consists of a page directory, a page table, and a TLB, which work together to translate linear

addresses into physical addresses efficiently and securely.