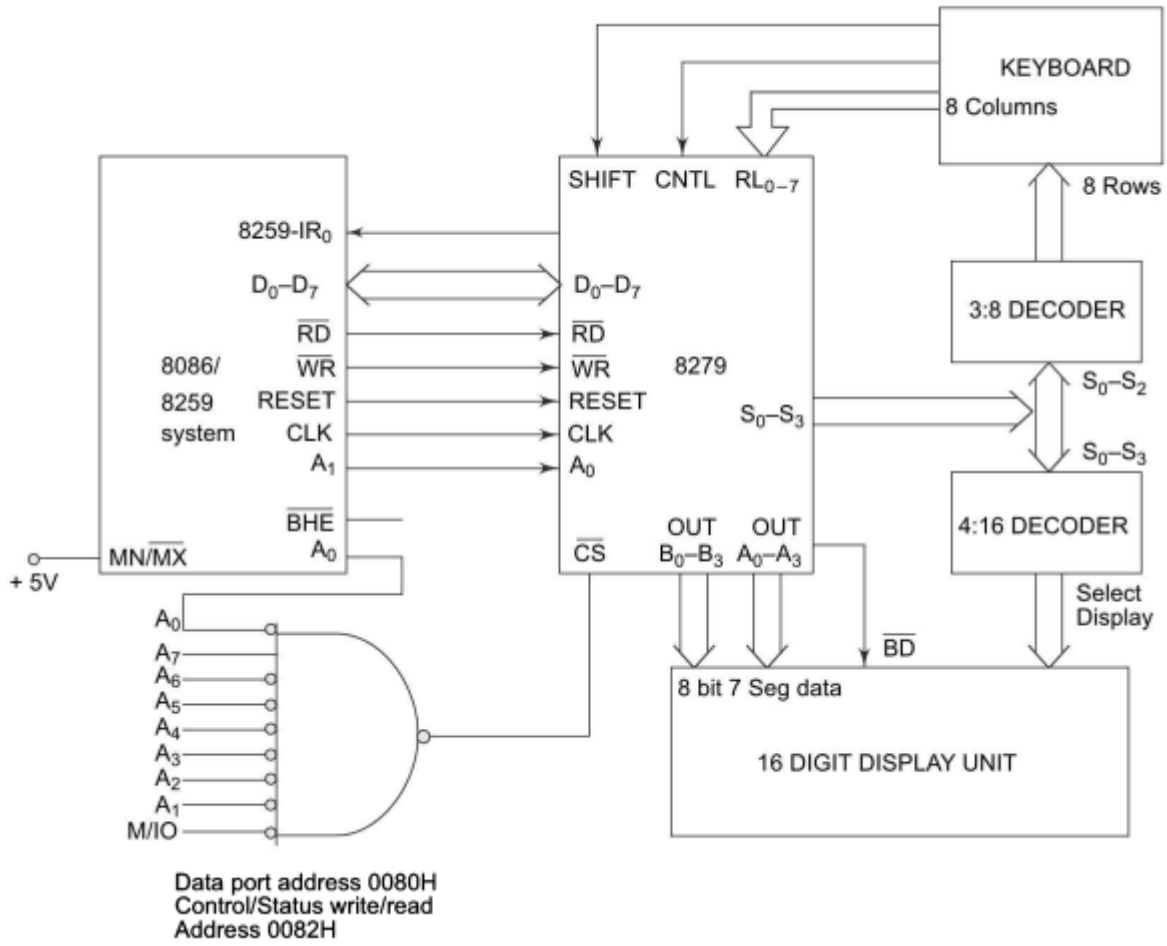
Why segmentation of memory concept (segment registers) was introduced in 8086 microprocessors?

1. It allows the memory addressing capacity to be 1 Mbyte even though the address associated with individual instruction is only 16-bit.
2. It allows instruction code, data, stack, and portion of program to be more than 64 KB long by using more than one code, data, stack segment, and extra segment.
3. It facilitates use of separate memory areas for program, data and stack.
4. It permits a program or its data to be put in different areas of memory, each time the program is executed i.e. program can be relocated which is very useful in multiprogramming.

Draw 16 digits display interface of 8279.



Describe interrupt cycle of 8086.

The interrupt cycle of the 8086 is a series of steps that the processor takes when it receives an interrupt. The interrupt cycle is as follows:

1. The interrupting device sends an interrupt signal to the 8086.
2. The 8086 checks the interrupt priority level.
3. If the interrupt priority level is higher than the current interrupt, the 8086 saves the current state of the processor on the stack.

4. The 8086 loads the address of the interrupt service routine (ISR) into the program counter.
5. The 8086 executes the ISR.
6. When the ISR completes, the 8086 returns to the interrupted program.

Write the name of modes in 8254? Also tell which mode in 8086 is used to generate square wave?

The modes in 8254 are:

- **Mode 0 (Interrupt on Terminal Count):** This mode is typically used for event counting. When the counter reaches 0, an interrupt is generated.
- **Mode 1 (Programmable One Shot):** This mode is used to generate a one-shot pulse. When the counter reaches 0, the output goes low and remains low until the counter is reloaded with a new value.
- **Mode 2 (Rate Generator):** This mode is used to generate a square wave with a period equal to the count value. The output goes high when the counter reaches 0 and goes low when the counter reaches the count value minus 1.
- **Mode 3 (Square Wave Generator):** This mode is similar to mode 2, but the output is always high when the counter is counting down and goes low when the counter reaches 0.
- **Mode 4 (Software Triggered Strobe):** This mode is used to generate a strobe pulse. When the GATE input is asserted, the output goes high and remains high until the GATE input is de-asserted.
- **Mode 5 (Hardware Triggered Strobe):** This mode is similar to mode 4, but the strobe pulse is generated when the GATE input is asserted by an external signal.

The mode in 8086 that is used to generate square wave is **mode 3**.

Explain the following directives for Intel 8086 microprocessor: ENM, EQU and PTR.

ENM directive is used to define an enumerated data type. Enumerated data types are a way of representing a set of named constants.

EQU directive is used to define a symbol. Symbols are a way of giving a name to a value.

PTR directive is used to specify the size of a pointer. Pointers are a way of storing the address of a memory location. The `PTR` directive tells the assembler the size of the pointer in bytes.

Draw the architecture of 8255.

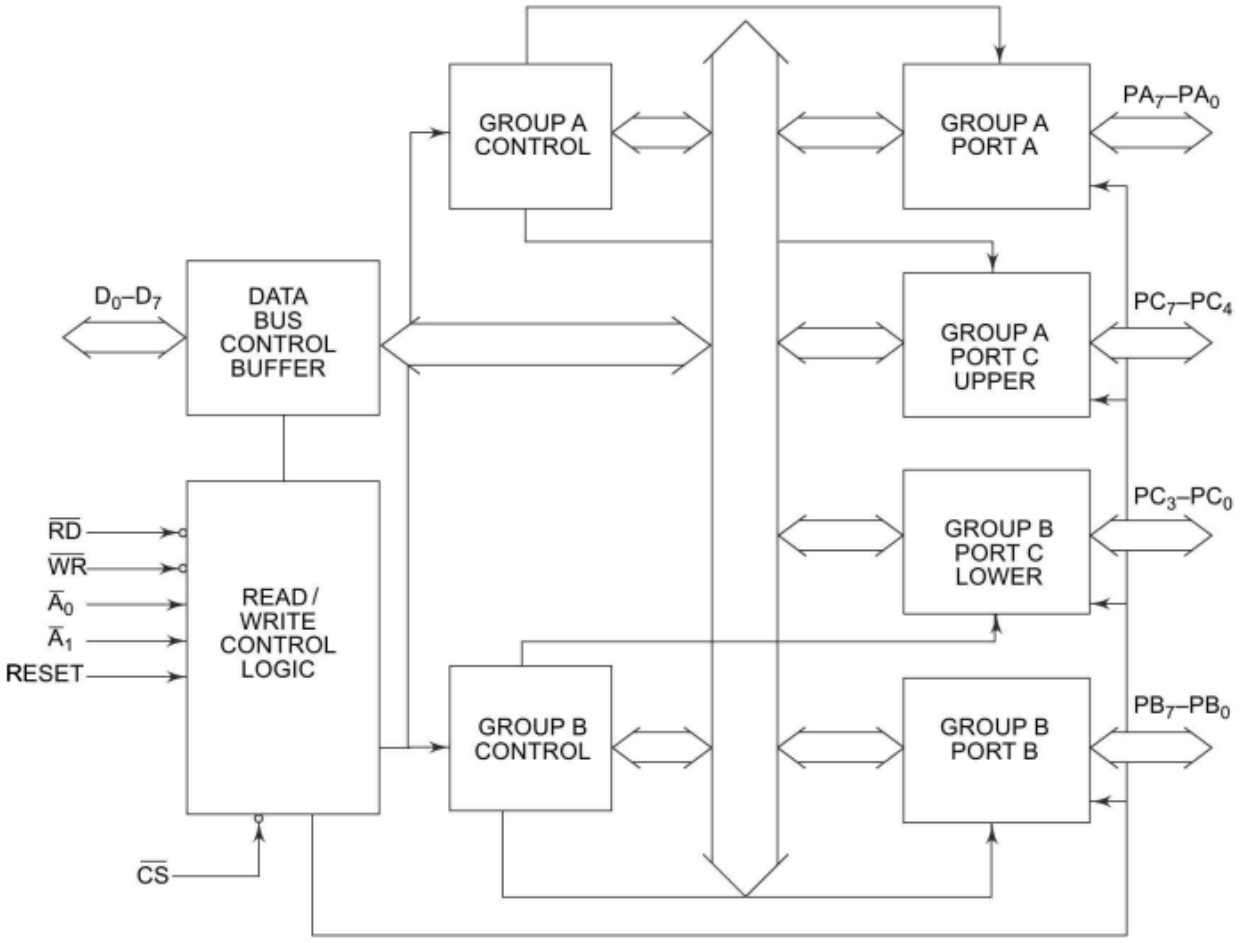


Fig. 5.17(a) *8255 Internal Architecture*

Write function of following pins of 8251 (i)T*RDY (ii) T*D.

- **T*RDY (Transmit Ready):** This pin indicates that the 8251 is ready to receive data from the CPU. The CPU can write data to the 8251 when the T*RDY pin is low.
- **T*D (Transmit Data):** This pin outputs the data that is being transmitted by the 8251. The data on the T*D pin is valid when the T*RDY pin is low.

What are the different exceptions generated by 8087.

Invalid Operation: These are the exceptions generated due to stack overflow, or, stack underflow, indeterminate form as result, or, non-number (NAN) as operand.

Overflow: A too big result to fit in the format generates this exception. The condition code bits indicate that the result is prohibitively large (infinity).

Underflow: If a small (in magnitude) result is generated, to fit in the specified format, 8087 generates this exception. If this exception is masked, the 8087 denormalizes the fraction until the exponent fits in the specified destination format.

Zero Divide: If any non-zero finite operand is divided by zero, this exception is generated. The resulting condition code bits indicate that the result is infinity, even if the exception is masked.

Denormalized Operand: This exception is generated, if at least one of the operands is denormalized. This may also be generated, if the result is denormalized. If this is masked, 8087 continues the execution normally.

Inexact Result: If it is impossible to fit the actual result in the specified format, the result is rounded according to the rounding control bits and an exception is generated. This sets the precision exception flag.

Discuss various addressing modes in brief.

Addressing Mode	Description	Example
Register mode	The operand is held in a register.	MOV AX, BX
Immediate mode	The operand is encoded in the instruction itself.	MOV AX, 5
Direct mode	The operand is specified as a memory address in the instruction.	MOV AX, [5000H]
Register indirect mode	The operand is indirectly specified through a register.	MOV AX, [BX]
Indexed mode	The operand is indirectly specified through a register and an offset.	MOV AX, [SI]
Register relative mode	The operand is indirectly specified through a register and an offset, where the offset is relative to the instruction pointer.	MOV AX, 50H[BX]
Base indexed mode	The operand is indirectly specified through a base register and an offset.	MOV AX, [BX] [SI]
Relative base indexed mode	The operand is indirectly specified through a base register and an offset, where the offset is relative to the instruction pointer.	MOV AX,50H [BX][SI]
Intrasegment direct mode	The operand is specified as a memory address within the current segment.	JMP SHORT LABEL
Intrasegment indirect mode	The operand is indirectly specified through a register that holds the address of a memory location within the current segment.	JMP [BX]

Intersegment direct mode	The operand is specified as a memory address in a different segment.	JPM 5000H: 2000H
Intersegment indirect mode	The operand is indirectly specified through a register that holds the address of a memory location in a different segment.	JMP [2000H]

How macros are handled by assembler during conversion of assembly language code in to machine language code.

The macro preprocessor scans the assembly language source code for macro definitions. When it finds a macro definition, it expands the macro into a sequence of assembly language instructions. The expanded code is then passed to the assembler, which generates the machine language code.

Example:

```
macro add_one(number)
   mov ax, number
   add ax, 1
end macro
add_one 10
```

The macro is expanded by the macro preprocessor, which generates the following assembly language instructions:

```
mov ax, 10
add ax, 1
```

What are the advantages of 8086 microprocessors over 8085 microprocessor?

- **16-bit data bus:** The 8086 has a 16-bit data bus, which allows it to access 64KB of memory. The 8085 has an 8-bit data bus, which limits it to 256KB of memory.
- **16-bit registers:** The 8086 has 16-bit registers, which allows it to store larger values. The 8085 has 8-bit registers, which limits the size of the values it can store.
- **Pipelined architecture:** The 8086 has a pipelined architecture, which allows it to execute instructions more efficiently. The 8085 does not have a pipelined architecture, which can lead to slower execution times.
- **More instructions:** The 8086 has a larger instruction set than the 8085, which gives it more flexibility. The 8085 has a smaller instruction set, which can make it less versatile.

Write short note on microprocessor bus types and buffering techniques.

The three main types of buses in a microprocessor are:

- **Data bus:** The data bus is used to transfer data between the microprocessor and other devices, such as memory and I/O devices.
- **Address bus:** The address bus is used to specify the location of the data that is being transferred.
- **Control bus:** The control bus is used to coordinate the activities of the different devices on the bus.

There are three main types of microprocessor buffering techniques:

- **Single buffering:** In single buffering, only one buffer is used to store data that is being transferred between the microprocessor and an external device. This is the simplest type of buffering, but it can be inefficient if the microprocessor needs to access the external device frequently.
- **Double buffering:** In double buffering, two buffers are used to store data that is being transferred between the microprocessor and an external device. One buffer is used to store data that is being read from the external device, and the other buffer is used to store data that is being written to the external device. This allows the microprocessor to continue processing data while the other buffer is being filled or emptied.
- **Circular buffering:** In circular buffering, a single buffer is used to store data that is being transferred between the microprocessor and an external device. The buffer is arranged in a circular fashion, so that the data is overwritten when the buffer is full. This allows the microprocessor to access data in a continuous fashion, without having to wait for the buffer to be emptied.

Explain the following instructions in detail with the help of examples:- (i) RAR (ii) CPI Data (iii) SPHL (iv) XRI data

RAR (Rotate Accumulator Right Through Carry): Each binary bit of accumulator is rotated by one position through the carry flag. Bit D0 is placed in carry flag, and the carry flag is placed in the most significant position D7. CY is modified according to bit D0. S, Z, P, AC are not affected.

Syntax: RAR none

Example: RAR

CPI Data (Compare Immediate Data with accumulator): The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of comparison is shown by setting the flags of the PSW as follows:

If A< data=> carry flag is set

If A=data=>zero flag is set

If A>data=>carry and zero flag are reset

Syntax: CPI 8-bit data

Example: CPI 89H

SPHL (Move content of H&L to stack pointer): The instruction loads the content of H & L registers to stack pointer register, the contents of H register provide the high order address and the contents of L register provide the low order address. The contents of H and L registers are not altered.

Syntax: SPHL none

Example: SPHL

XRI Data (Exclusive OR of Immediate data with accumulator): The contents of accumulator are Exclusive ORed with 8-bit data operand and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

Syntax: XRI 8-bit data

Example: XRI 86H

Write note on 8086 minimum mode and maximum mode CPU module.

Minimum Mode

- The 8086 microprocessor operates in minimum mode when MN/MX' = 1.
- In minimum mode,8086 is the only processor in the system which provides all the control signals which are needed for memory operations and I/O interfacing.
- Here the circuit is simple but it does not support multiprocessing.
- The other components which are transceivers, latches, 8284 clock generator, 74138 decoder, memory and i/o devices are also present in the system.
- The address bus of 8086 is 20 bits long. By this we can access 2^{20} byte memory i.e. 1MB . Out of 20 bits, 16 bits A_0 to A_{15}(or 16 lines) are multiplexed with a data bus. By multiplexing, it means they will act as address lines during the first T state of the machine cycle and in the rest, they act as data lines. A_{16} to A_{19} are multiplexed S_3 to S_6 and BHE' is multiplexed with S_7.

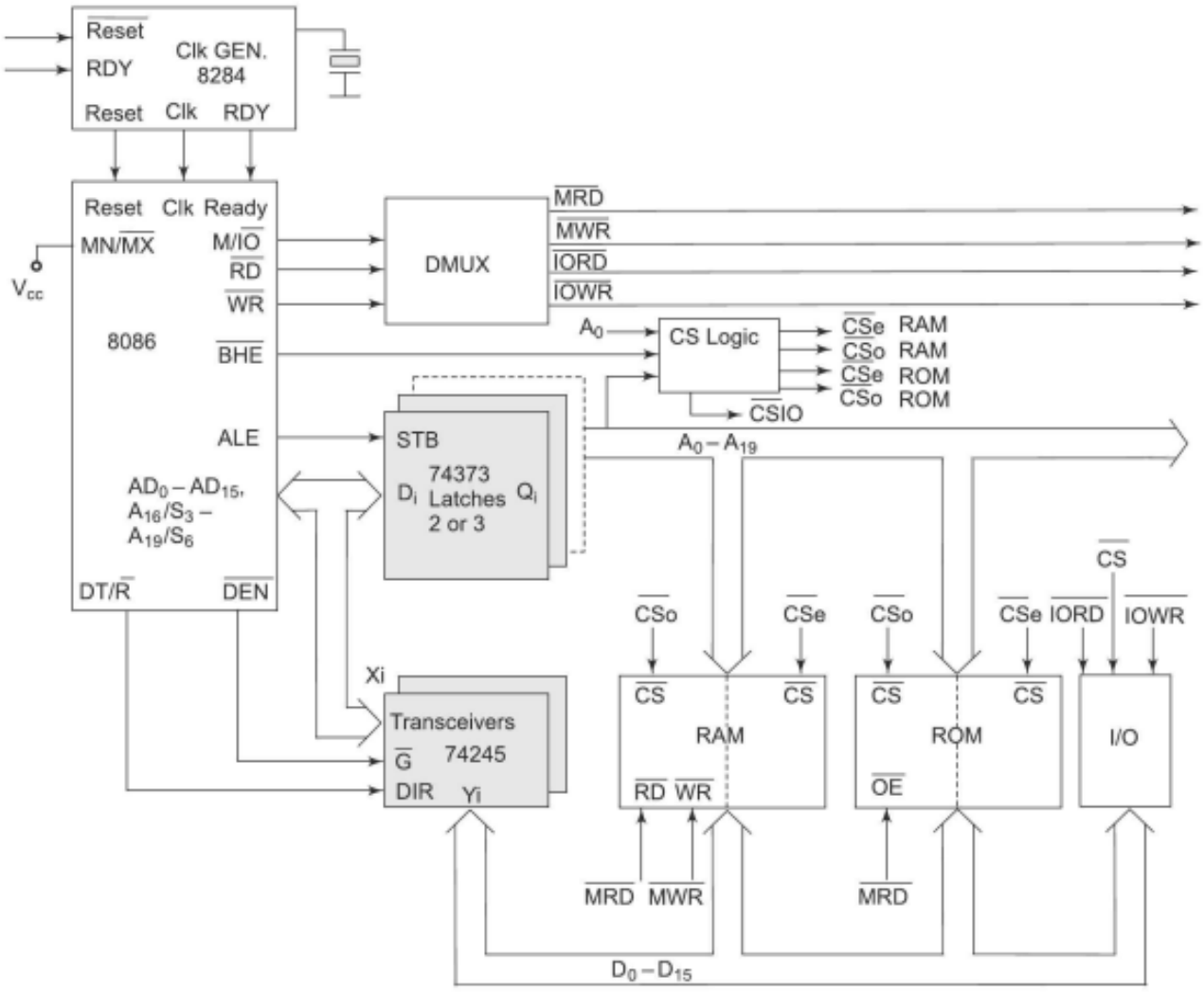


Fig. 1.13 *Minimum Mode 8086 System*

Maximum Mode

- In this we can connect more processors to 8086 (8087/8089).
- 8086 max mode is basically for implementation of allocation of global resources and passing bus control to other coprocessor(i.e. second processor in the system), because two processors can not access system bus at same instant.
- All processors execute their own program.
- The resources which are common to all processors are known as global resources.
- The resources which are allocated to a particular processor are known as local or private resources.

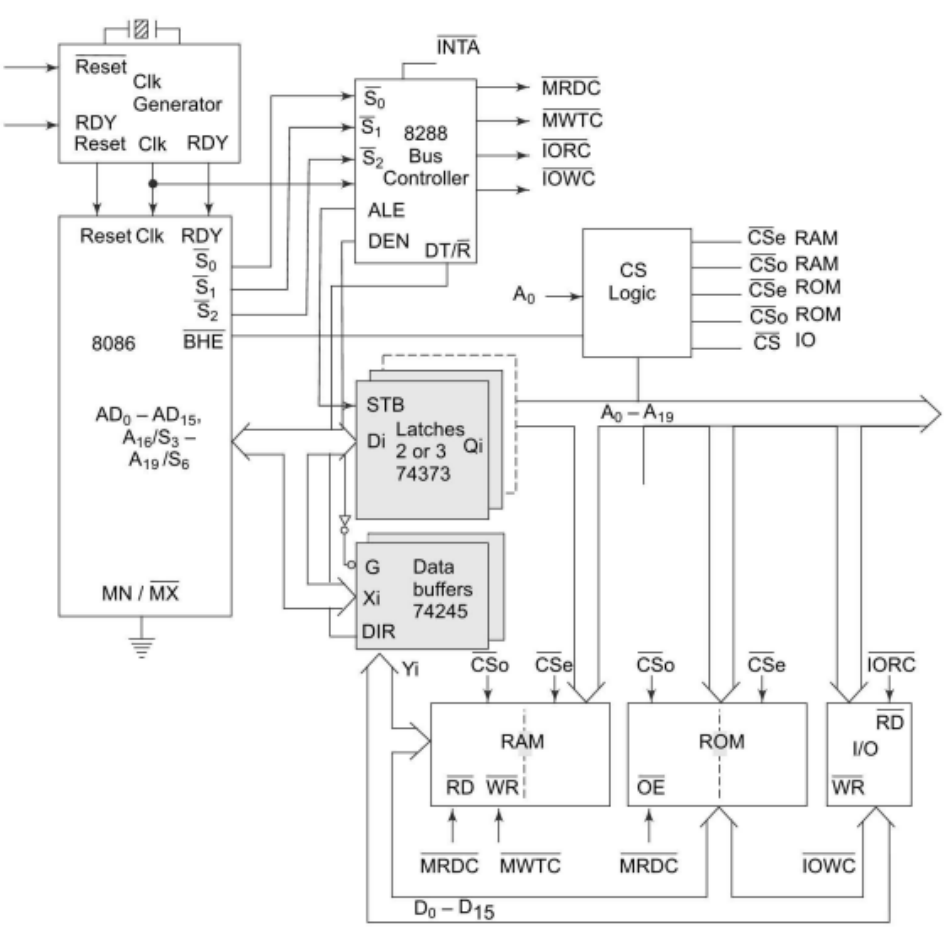
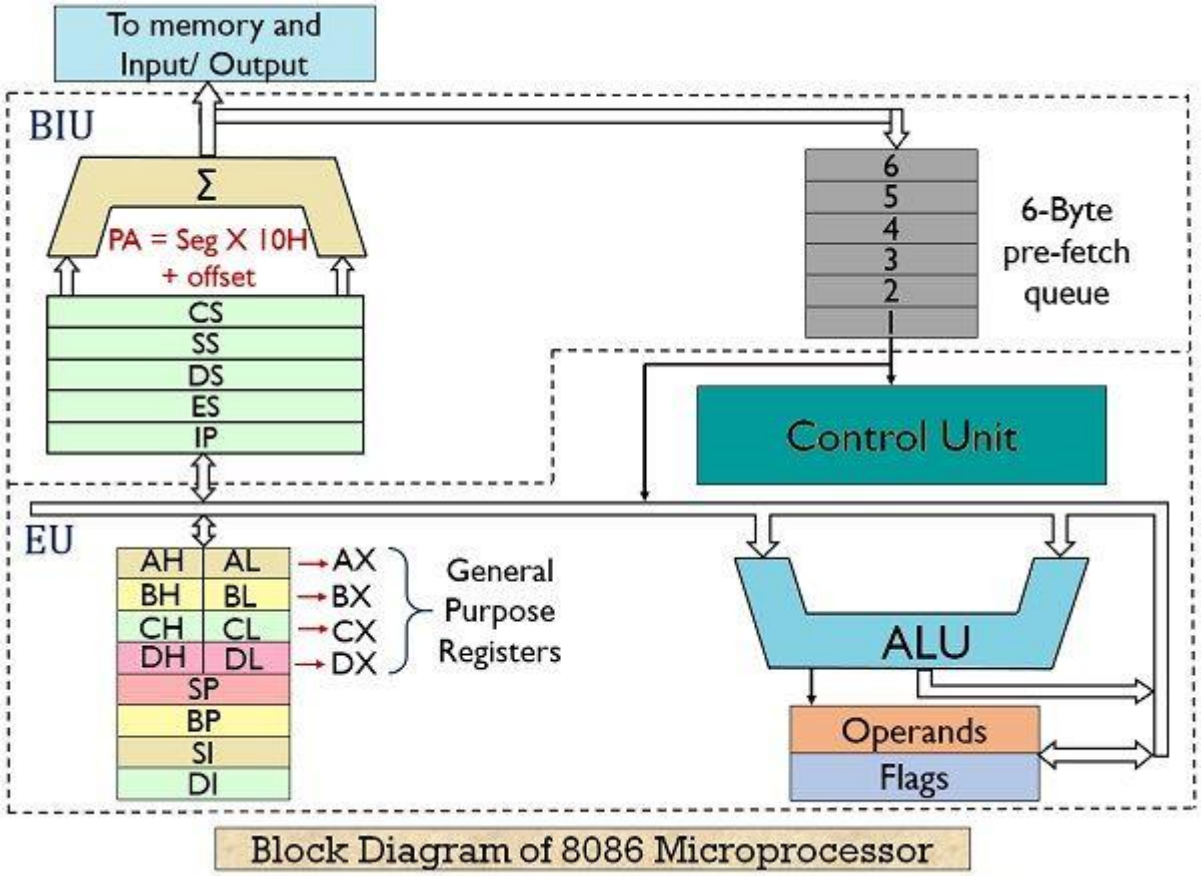


Fig. 1.15 *Maximum Mode 8086 System*

Explain block diagram of 8086 microprocessor and explain its PSW.



Block Diagram of 8086 Microprocessor

The 8086 microprocessor has the following major components:

- **Arithmetic Logic Unit (ALU):** The ALU is responsible for performing arithmetic and logical operations on data.
- **General-Purpose Registers (GPRs):** The GPRs are used to store data and intermediate results.
- **Instruction Pointer (IP):** The IP points to the next instruction to be executed.
- **Program Status Word (PSW):** The PSW contains status information about the processor, such as the carry flag, the zero flag, and the interrupt flag.
- **Memory Management Unit (MMU):** The MMU is responsible for managing the memory of the system.
- **Bus Interface Unit (BIU):** The BIU is responsible for communicating with the external bus of the system.

The **PSW** in 8086 is a 32-bit register that consists of 16-bit flag register and 16-bit accumulator register. The contents of PSW are saved in stack during the handling of interrupt.

The control flags can be set or cleared by the programmer

- **DF (Direction Flag):** This flag controls the direction of string operations. If DF = 0, string operations proceed in ascending order. If DF = 1, string operations proceed in descending order.
- **IF (Interrupt Flag):** This flag controls whether or not the processor can interrupt. If IF = 0, the processor cannot interrupt. If IF = 1, the processor can interrupt.
- **TF (Trap Flag):** This flag controls whether or not the processor enters single-step mode. If TF = 0, the processor runs normally. If TF = 1, the processor enters single-step mode, which means that it executes instructions one at a time.

The status flags are set or reset depending on the results of some arithmetic or logical operations during program execution.

- **CF (Carry Flag):** This flag is set if there is a carry out of the MSB position resulting from an addition operation or subtraction.
- **AF (Auxiliary Carry Flag):** This flag is set if there is a carry out of bit 3 resulting from an addition operation.
- **SF (Sign Flag):** This flag is set if the most significant bit of the result of an arithmetic operation is1.
- **ZF (Zero Flag):** This flag is set if the result of an arithmetic or logical operation is zero.
- **PF (Parity Flag):** This flag is set if there is an even number of one bits in the result of an arithmetic or logical operation.
- **OF (Overflow Flag):** This flag is set if there is a signed overflow.

Explain the concept of memory segmentation and its advantages.

Memory segmentation is a technique that divides the physical memory of a computer into logical segments. Each segment can be accessed independently, which allows the operating system to protect different parts of the memory from each other.

In the 8086 microprocessor, memory is segmented into four different segments:

- **Code segment:** This segment contains the program code.
- **Data segment:** This segment contains the data that is used by the program.
- **Stack segment:** This segment contains the stack, which is used to store temporary data.
- **Extra segment:** This segment is used for miscellaneous purposes.

The advantages of memory segmentation include:

- **Protection:** Memory segmentation allows the operating system to protect different parts of the memory from each other. This is important to prevent unauthorized access to sensitive data.
- **Efficiency:** Memory segmentation can improve the efficiency of memory access by allowing the operating system to access different parts of the memory independently.
- **Flexibility:** Memory segmentation allows the operating system to be more flexible in how it manages memory. This is important for supporting different types of applications.

Explain following 8086 instruction:- (i) XLAT (ii) LDS (ii) AAM (iv) SCASW

XLAT: The content of 8 bit memory is transferred to AL. The effective address of memory is given by sum of BX and AL.

Symbolic Representation: $MA=DS*16_{10}+BX+AL$

$AL \leftarrow MA$

LDS: The word from first two memory location is moved to the 16 bit register and the word from next two memory location is moved to DS register.

Symbolic Representation: reg16←mem

DS←(mem+2)

AAM: After multiplication of two 8 bit unpacked BCD data the result in AX will be in binary. This instruction can be executed after multiplication to convert the result in AX to unpacked BCD.

Symbolic Representation: Adjust AH to unpacked BCD data

$AH=AL+0A_H$

$AL=AL \text{ MOD } 0A_H$

Note: $0A_H=10_{10}$

SCASW: One word of string data in extra segment is subtracted from the content of AX and the result is used to modify flags. The content of DI and SI are automatically incremented/decremented by 2 depending on direction flag.

What is demultiplexing of buses in 8086? Explain demultiplexing of address bus in 8086 and 8088.

Demultiplexing of buses in 8086 is used to split the address bus into two signals: the segment address and the offset address.

The demultiplexing of the address bus in the 8086 is handled by the address latch and the data buffer. The address latch captures the segment address from the address bus during the first clock cycle. The data buffer holds the data from memory or I/O devices during the third clock cycle and delivers it to the processor.

The demultiplexing of the address bus in the 8088 is similar to the demultiplexing in the 8086. The main difference is that the 8088 has an 8-bit data bus, so only the lower 8 bits of the address bus are multiplexed.

Explain different data transfer instructions with proper example in 8086 microprocessor.

Data transfer instructions are used to transfer the data from the source operand to the destination operand. Following are the list of instructions under this group −

Instruction to transfer a word

- **MOV** − Used to copy the byte or word from the provided source to the provided destination. e.g. MOV BX, AX
- **PPUSH** − Used to put a word at the top of the stack. e.g. PPUSH AX
- **POP** − Used to get a word from the top of the stack to the provided location. e.g. POP BX
- **PUSHA** − Used to put all the registers into the stack. e.g. PUSHA
- **POPA** − Used to get words from the stack to all registers. e.g. POPA
- **XCHG** − Used to exchange the data from two locations. e.g. XCHG AX, BX
- **XLAT** − Used to translate a byte in AL using a table in the memory. e.g. XLAT 0x1234

Instructions for input and output port transfer

- **IN** − Used to read a byte or word from the provided port to the accumulator. e.g. IN AL, 0x60
- **OUT** − Used to send out a byte or word from the accumulator to the provided port. e.g. OUT 0x61, AL

Instructions to transfer the address

- **LEA** − Used to load the address of operand into the provided register. e.g. LEA AX, BX
- **LDS** − Used to load DS register and other provided register from the memory e.g. LDS BX, 0x1234
- **LES** − Used to load ES register and other provided register from the memory. e.g. LES BX, 0x1234

Instructions to transfer flag registers

- **LAHF** − Used to load AH with the low byte of the flag register. e.g. LAHF
- **SAHF** − Used to store AH register to low byte of the flag register. e.g. SAHF
- **PUSHF** − Used to copy the flag register at the top of the stack. e.g. PUSHF
- **POPF** − Used to copy a word at the top of the stack to the flag register. e.g. POPF

Write an assembly language program to multiply two 16 bits numbers.

Memory	Mnemonics	Operands	Comment
2000	MOV	AX, [3000]	[AX] <- [3000]
2004	MOV	BX, [3002]	[BX] <- [3002]
2008	MUL	BX	[AX] <- [AX] * [BX]
200A	MOV	[3004], AX	[3004] <- AX
200E	MOV	AX, DX	[AX] <- [DX]
2010	MOV	[3006], AX	[3006] <- AX
2014	HLT		Stop

Discuss string instructions with suitable example. Explain why REP prefix is added with string instruction. Which string instructions should be used to ensure that two strings in the memory are equal.

String instructions are a set of instructions that are used to manipulate strings of data. Strings are a sequence of bytes or words that are stored in memory.

The REP prefix is added with string instructions to repeat the instruction until the CX register is zero. This is useful for looping through a string and performing an operation on each byte.

OPCODE	OPERAND	EXPLANATION	EXAMPLE
REP	instruction	repeat the given instruction till CX != 0	REP MOVSB
REPE	instruction	repeat the given instruction while CX = 0	REPE
REPZ	instruction	repeat the given instruction while ZF = 1	REPZ
REPNE	instruction	repeat the given instruction while CX != 0	REPNE
REPNZ	instruction	repeat the given instruction while ZF = 0	REPNZ
MOVSB	none	moves contents of byte given by DS:SI into ES:DI	MOVSB
MOVSW	none	moves contents of word given by DS:SI into ES:DI	MOVSW

Page No 152 of Bhavya

CMPS (Compare String): This instruction compares two strings and sets the zero flag if the strings are equal.

Explain assembly language program development tools.

Tool	Description	File Extension
Assembler	Translates assembly language programs into machine language.	.asm
Editor	Creates and edits assembly language source code.	.asm, .txt
Debugger	Debugs assembly language programs.	.asm, .exe
Linker	Combines multiple object files into a single executable file.	.obj, .exe
Loader	Loads executable files into memory.	.exe
Disassembler	Converts machine code into assembly language.	.obj
Simulator	Emulates the hardware of a computer.	.asm, .exe
Profiler	Analyzes the performance of a program.	.exe
Locator	Finds the addresses of symbols in an object file.	.obj
Library builder	Creates libraries.	.lib
Emulator	Simulates the hardware of a computer.	.asm, .exe

Write the procedure to determine physical address for the following instructions: - (a) MOV AL, CS: [BX+0400] (b) MOV AL, [BX+SI+22] assume CS-4000h, IP-2300, SI-02300 and DS-5000

Register	Offset stored in
CS	IP
DS	BX,DI,SI
ES	DI
SS	SP,BP

Write the difference between the following instructions:- (a)MUL and IMUL (b) DIV and IDIV (c) JUMP and LOOP (d) Shift and Rotate

Instruction	Syntax	Description	Example
MUL	MUL reg/mem	Performs an unsigned multiplication of the contents of the accumulator with the contents of the operand and stores the result in the accumulator.	`MUL AX` multiplies the contents of the accumulator with the contents of register AX and stores the result in the accumulator.
IMUL	IMUL reg/mem	Performs a signed multiplication of the contents of the accumulator with the contents of the operand and stores the result in the accumulator.	`IMUL AX, BX` multiplies the contents of the accumulator with the contents of register BX and stores the result in the accumulator.
DIV	DIV reg/mem	Performs an unsigned division of the contents of the accumulator by the contents of the operand and stores the result in the accumulator.	`DIV BX` divides the contents of the accumulator by the contents of register BX and stores the result in the accumulator.
IDIV	IDIV reg/mem	Performs a signed division of the contents of the accumulator by the contents of the operand and stores the result in the accumulator.	`IDIV BX` divides the contents of the accumulator by the contents of register BX and stores the result in the accumulator.

JUMP	JUMP addr	Unconditionally transfers control to the specified address.	`JUMP label` unconditionally transfers control to the label `label`.
LOOP	LOOP addr	Transfers control to the specified address if the specified condition is met.	`LOOP label` transfers control to the label `label` if the value of the CX register is not zero.
Shift	Shift reg, count	Shifts the bits in the register to the left or right by the specified number of bits.	`SHL AX, 1` shifts the bits in register AX to the left by one bit.
Rotate	Rotate reg, count	Shifts the bits in the register to the left or right by the specified number of bits and then rotates the bits back into the register.	`ROR AX, 1` shifts the bits in register AX to the right by one bit and then rotates the bits back into the register.

Explain the interfacing between 8087 and 8086 microprocessor. Draw the block diagram of 8279.

- The 8086 and 8087 are connected in maximum mode, with MN/MX grounded.
- The 8284 provides the common CLK, RESET, and READY signals.
- The 8286 are used as data transceivers.
- The 8288 generates control signals using S2, S1, and S0 as input from the currently active processor.
- A homogeneous program is written that contains both 8086 and 8087 instructions.
- The ESC prefix is used to activate the 8087.
- The 8087 executes its instruction and the 8086 moves ahead with its next instruction. This is called multiprocessing.
- During execution, if the 8087 needs to read/write more words from the memory, then it does so by stealing bus cycles from the microprocessor.
- If the 8086 requires the result of the 8087 operation, it first executes the WAIT instruction.
- During execution, if an exception occurs, which is unmasked, the 8087 interrupts the microprocessor using the INT output pin through the PIC 8259.

QS_1	QS_0	**8087 operation**
0	0	NOP
0	1	8087 compares the 5 MSB bits with 11011 (ESC code)
1	0	8087 clears its queue
1	1	If earlier comparison succeeds, 8087 fetches the subsequent byte else NOP

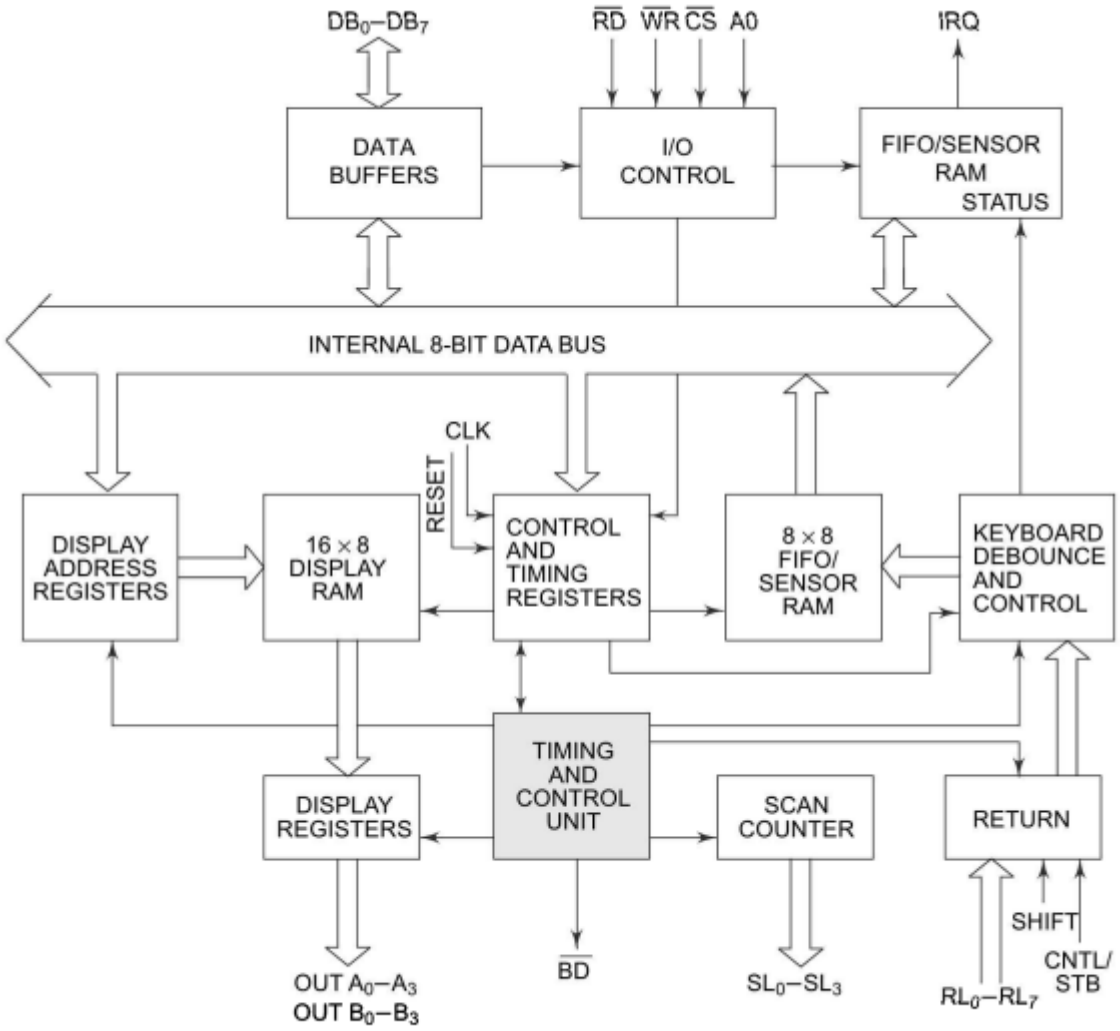


Fig. 6.23 *8279 Internal Architecture*

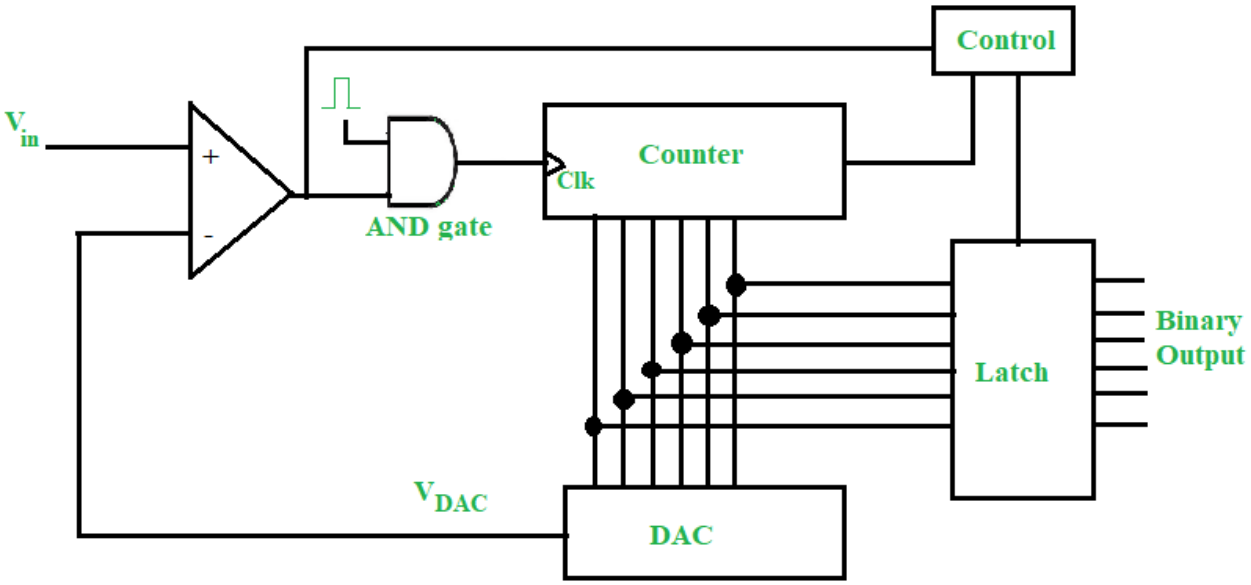
Explain counting type ADC with the suitable diagram what are the limitations of this converter? How can you improve the performance of ADC.

A counting type ADC is a type of analog-to-digital converter that converts an analog signal into a digital signal by counting the number of clock cycles it takes for the analog signal to cross a threshold. The threshold is set by a reference voltage.

The analog signal is applied to the input of the ADC. The reference voltage is applied to the comparator. The comparator compares the analog signal to the reference voltage. When the analog signal crosses the reference voltage, the comparator output goes high.

The counter is clocked by a clock signal. When the comparator output goes high, the counter is reset to zero. The counter then starts counting up. The counter continues counting up until the analog signal again crosses the reference voltage.

The digital output of the ADC is the value of the counter. The value of the counter is proportional to the input analog signal.



Limitations of counting type ADC

The main limitation of the counting type ADC is that it is not very accurate. The accuracy of the ADC depends on the reference voltage and the clock frequency. The higher the reference voltage and the clock frequency, the more accurate the ADC will be.

Another limitation of the counting type ADC is that it is slow. The conversion time of the ADC depends on the reference voltage and the clock frequency. The higher the reference voltage and the clock frequency, the longer the conversion time will be.

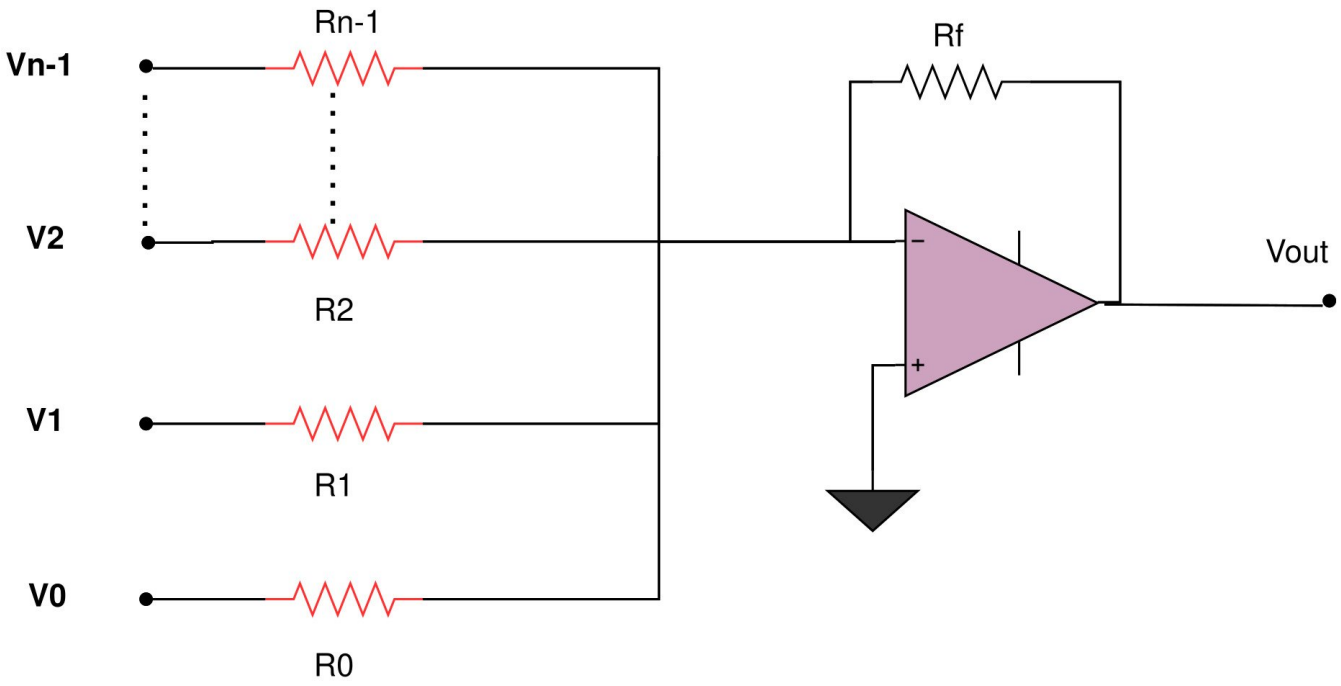
How to improve the performance of ADC

The performance of the counting type ADC can be improved by increasing the reference voltage and the clock frequency. However, increasing the reference voltage and the clock frequency will also increase the cost of the ADC.

Another way to improve the performance of the counting type ADC is to use a technique called successive approximation. In successive approximation, the ADC starts by guessing the value of the analog signal. The ADC then increments or decrements the guess based on the output of the comparator. The ADC continues incrementing or decrementing the guess until the comparator output goes high.

Successive approximation is a more accurate and faster method of converting analog signals to digital signals than the counting type ADC. However, successive approximation is also more complex and expensive.

Draw N-bit binary weight DAC and explain its operation. What are the disadvantages of binary weight DAC? What is difference between N-bit DAC and R-2R ladder DAC.



N-bit binary weight DAC

An N-bit binary weight DAC is a type of digital-to-analog converter that converts an N-bit digital number into an analog voltage. The N-bit digital number is represented by a binary number, where each bit represents a different power of 2.

The N-bit digital number is applied to the input of the DAC. The DAC then converts the digital number into an analog voltage. The analog voltage is proportional to the digital number.

The analog voltage is generated by a set of resistors. The resistors are arranged in a binary weight configuration. The value of each resistor is proportional to the weight of the corresponding bit in the digital number.

The sum of the voltages across these resistors is the analog voltage generated by the DAC.

Disadvantages of binary weight DAC

The main disadvantage of the binary weight DAC is that it is not very accurate. The accuracy of the DAC depends on the value of the resistors. The larger the value of the resistors, the more accurate the DAC will be. However, larger resistors also increase the size and cost of the DAC.

Another disadvantage of the binary weight DAC is that it is not very efficient. The DAC wastes power because the resistors are not always fully utilized.

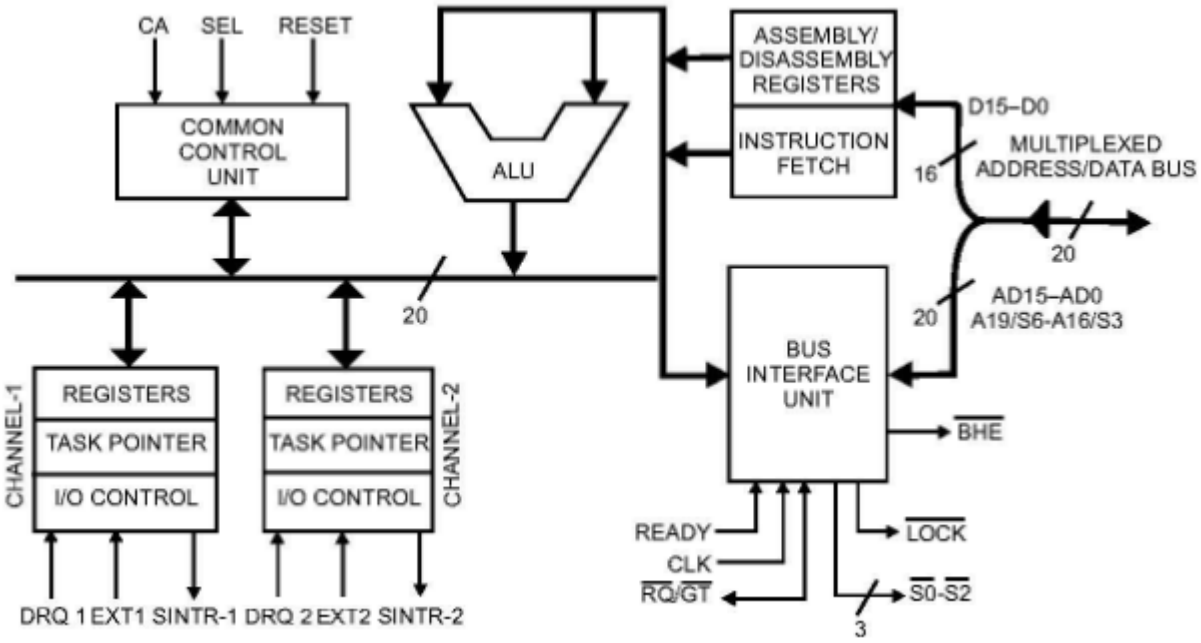
Difference between N-bit DAC and R-2R ladder DAC

The main difference between N-bit DAC and R-2R ladder DAC is the way the resistors are arranged. In N-bit DAC, the resistors are arranged in a binary weight configuration. In R-2R ladder DAC, the resistors are arranged in a R-2R configuration.

The R-2R ladder DAC is more efficient than the N-bit DAC because the resistors are always fully utilized. The R-2R ladder DAC is also more accurate than the N-bit DAC because the resistors are smaller.

However, the R-2R ladder DAC is more complex than the N-bit DAC. The R-2R ladder DAC also requires more resistors than the N-bit DAC.

What is 8089 I/O processor explain with its functional block diagram. Write different features of 8089 I/O processor.



- **Common Control Unit (CCU):** The CCU is responsible for controlling the operation of the 8089 I/O processor. It determines which channel will execute the next cycle, and it also handles interrupts from I/O devices.
- **Arithmetic & Logic Unit (ALU):** The ALU performs arithmetic and logical operations. It can add, subtract, multiply, and divide numbers, and it can also perform logical operations such as AND, OR, and XOR.
- **Assembly/Disassembly registers:** These registers allow the 8089 to deal with 8-bit and 16-bit devices. For example, if an 8-bit device is sending data to a 16-bit memory interface, the 8089 can use the assembly/disassembly registers to combine two bytes of data into a single 16-bit word.
- **Bus Interface Unit (BIU):** The BIU is responsible for communicating with the system bus. It can fetch instructions and data from memory, and it can also send data to I/O devices.
- **Instruction Fetch:** This unit fetches instructions from memory and stores them in a queue to be executed later.

Features:

1. 8089 has very high speed DMA capability.
2. It has 1 MB address capability.
3. It is compatible with iAPX 86, 88.
4. It supports local mode and remote mode I/O processing.
5. 8089 allows mixed interface of 8-and 16-bit peripherals, to 8-and 16-bit processor buses.
6. It supports two I/O channels.
7. Multibus compatible system interface.
8. Memory based communications with CPU.