MPMC
Course File

UNIT-I

Evolution of microprocessor:

The first microprocessor was introduced in the year 1971. It was introduced by Intel and was named Intel 4004.

Intel 4004 is a 4 bit microprocessor and it was not a powerful microprocessor. It can perform addition and subtraction operation on 4 bits at a time.

However it was Intel's 8080 was the first microprocessor to make it to Home computers. It was introduced during the year 1974 and it can perform 8 bit operations. Then during the year 1976, Intel introduced 8085 processors which is nothing but an update of 8080 processors.8080 processors are updated by adding two Enable/Disable Instructions, Three added interrupt pins and serial I/O pins.

Intel introduced 8086 pins during the year 1976. The major difference between 8085 and 8086 processor is that 8085 is an 8 bit processor, but 8086 processor is a 16 bit processor.

Architecture of 8086:

The Architecture of 8086 mainly consists of 2two parts. Bus Interface Unit (BIU) and Execution Unit (EU)

•EU executes instructions from the instruction system byte queue.

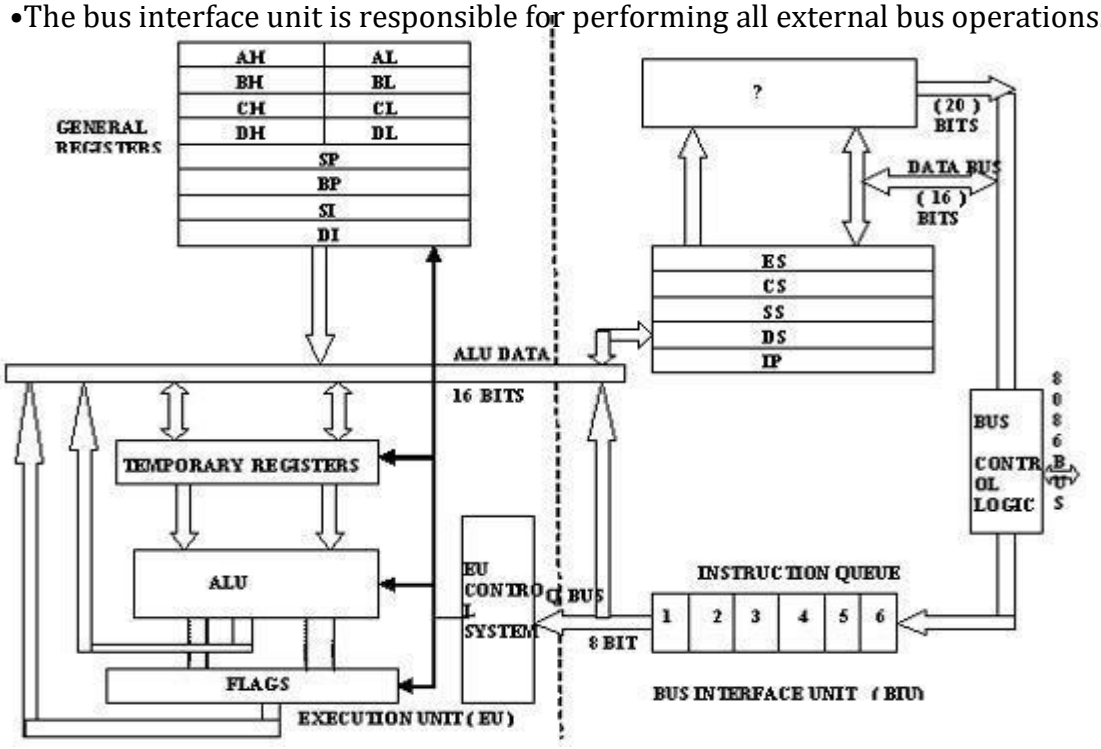
•Both units operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as Pipelining. This results in efficient use of the system bus and system performance.

•BIU contains Instruction queue, Segment registers, Instruction pointer, Address adder.

•EU contains Control circuitry, Instruction decoder, ALU, Pointer and Index register, Flag register.

BUS INTERFACR UNIT:

- It provides a full 16 bit bidirectional data bus and 20 bit address bus.
- The bus interface unit is responsible for performing all external bus operations.



•This queue permits prefetch of up to six bytes of instruction code. When ever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by prefetching the next sequential instruction.

•These prefetching instructions are held in its FIFO queue. With its 16 bit data bus, the BIU fetches two instruction bytes in a single memory cycle.

•After a byte is loaded at the input end of the queue, it automatically shifts up through the FIFO to the empty location nearest the output.

•The EU accesses the queue from the output end. It reads one instruction byte after the other from the output of the queue. If the queue is full and the EU is not requesting access to operand in memory.

•These intervals of no bus activity, which may occur between bus cycles are known as ***Idle state***.

•If the BIU is already in the process of fetching an instruction when the EU request it to read or write operands from memory or I/O, the BIU first completes the instruction fetch bus cycle before initiating the operand read / write cycle.

•The BIU also contains a dedicated adder which is used to generate the 20bit physical address that is output on the address bus. This address is formed med by combining the current contents of the code segment CS register and the current contents of the instruction pointer IP register.

•The BIU is also responsible for generating bus control signals such as those for memory read or write and I/O read or write.

EXECUTION UNIT

The Execution unit is responsible for decoding and executing all instructions.

•The EU extracts instructions from the top of the queue in the BIU, decodes them, generates operands if necessary, passes them to the BIU and requests it to perform the read or write bys cycles to memory or I/O and perform the operation specified by the instruction on the operands.

•During the execution of the instruction, the EU tests the status and control flags and updates them based on the results of executing the instruction.

•If the queue is empty, the EU waits for the next instruction byte to be fetched and shifted to top of the queue.

•When the EU executes a branch or jump instruction, it transfers control to a location

corresponding to another set of sequential instructions.

SPECIAL FUNCTIONS OF GENERAL PURPOSE REGISTERS

Accumulator register consists of 2 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX. AL in this case contains the low-order byte of the word, and AH contains the high-order byte. Accumulator can be used for I/O operations and string manipulation.

Base register consists of 2 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX. BL in this case contains the low-order byte of the word, and BH contains the high-order byte. BX register usually contains a data pointer used for based, based indexed or register indirect addressing.

Count register consists of 2 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX. When combined, CL register contains the low-order byte of the word, and CH contains the high-order byte. Count register can be used as a counter in string manipulation and shift/rotate instructions.

Data register consists of 2 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX. When combined, DL register contains the low-order byte of the word, and DH contains the high-order byte. Data register can be used as a port number in I/O operations. In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number.

SPECIAL FUNCTIONS OF SPECIAL PURPOSE REGISTERS

Stack Pointer (SP) is a 16-bit register pointing to program stack.

Base Pointer (BP) is a 16-bit register pointing to data in stack segment. BP register is usually used for based, based indexed or register indirect addressing.

Source Index (SI) is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data address in string manipulation instructions.

Destination Index (DI) is a 16-bit register. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions. The si and di registers (Source Index and Destination Index) have some special purposes as well. You may use these registers as pointers (much like the bx register) to indirectly access memory. You'll also use these registers with the 8086 string instructions when processing character strings.

The bp register (Base Pointer) is similar to the bx register. You'll generally use this register to access parameters and local variables in a procedure.

The sp register (Stack Pointer) has a very special purpose - it maintains the *program stack*. Normally, you would not use this register for arithmetic computations. The proper operation of most programs depends upon the careful use of this register.

SEGMENTATION:

Since address registers and address operands are only 16 bits they can only address 64k bytes. In order toaddress the 20-bit address range of the 8086, physicaladdresses (those that are put on the address bus)are always formed by adding the values of one of the instruction is executed? The use of segment registers reduces the size ofpointers to 16 bits.

This reduces the code size but also restricts the addressing range of a pointer to 64k bytes. Performing address arithmetic within data structures larger than 64k is awkward. This is the biggest drawback of the 8086 architecture. We will restrict ourslves to short programs where all of the code, data and stack are placed into thesame 64k segment (i.e. CS=DS=SS).

Most of the registers contain data/instruction offsets within 64 KB memory segment. There are four different 64 KB segments for instructions, stack, data and extra data. To specify where in 1 MB of processor memory these 4 segments are located the processor uses four segment registers:

Memory

•Program, data and stack memories occupy the same memory space. As the most of the processor instructions use 16-bit pointers the processor can effectively address only 64 KB of memory.

• To access memory outside of 64 KB the CPU uses special segment registers to specify where the code, stack and data 64 KB segments are positioned within 1 MB of memory (see the "Registers" section below).

•16-bit pointers and data are stored as:

address: low-order byte

address+1: high-order byte

•**Program memory** - program can be located anywhere in memory. Jump and call instructions can be used for short jumps within currently selected 64 KB code segment, as well as for far jumps anywhere within 1 MB of memory.

•All conditional jump instructions can be used to jump within approximately +127 to -127 bytes from current instruction.

•**Data memory** - the processor can access data in any one out of 4 available segments, which limits the size of accessible memory to 256 KB (if all four segments point to different 64 KB blocks).

•Accessing data from the Data, Code, Stack or Extra segments can be usually done by prefixing instructions with the DS:, CS:, SS: or ES: (some registers and instructions by default may use the ES or SS segments instead of DS segment).

•Word data can be located at odd or even byte boundaries. The processor uses two memory accesses to read 16-bit word located at odd byte boundaries. Reading word data from even byte boundaries requires only one memory access.

•**Stack memory** can be placed anywhere in memory. The stack can be located at odd memory addresses, but it is not recommended for performance reasons (see "Data Memory" above).

Reserved locations:

•0000h - 03FFh are reserved for interrupt vectors. Each interrupt vector is a 32-bit pointer in format segment: offset.

•FFFF0h - FFFFFh - after RESET the processor always starts program execution at the FFFF0h address.

segment registers to the 16-bit address to form a 20-bit address.

The segment registers themselves only contain themost-significant 16 bits of the 20-bit value that iscontributed by the segment registers. The least significantfour bits of thesegment address are alwayszero.

By default, the DS (data segment) is used fordata transfer instructions (e.g. MOV), CS(codesegment) is used with control transfer instructions(e.g. JMP or CALL), and SS is used with the stackpointer (e.g. PUSH or to save/restore addresses duringCALL/RET or INT instructions).

Exercise: If DS contains 0100H, what address will be written by the instruction MOV [2000H],AL? If CX contains 1122H, SP contains 1234H, and SS contains 2000H, what memory values will change and what will be their values when the PUSH CX

Code segment (CS) is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. CS register cannot be changed directly.

The CS register is automatically updated during far jump, far call and far return instructions.

Stack segment (SS) is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction.

Data segment (DS) is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment. DS register can be changed directly using POP and LDS instructions.

Extra segment (ES) is a 16-bit register containing address of 64KB segment, usually with program data. By default, the processor assumes that the DI register references the ES segment in string manipulation instructions. ES register can be changed directly using POP and LES instructions.

It is possible to change default segments used by general and index registers by prefixing instructions with a CS, SS, DS or ES prefix.

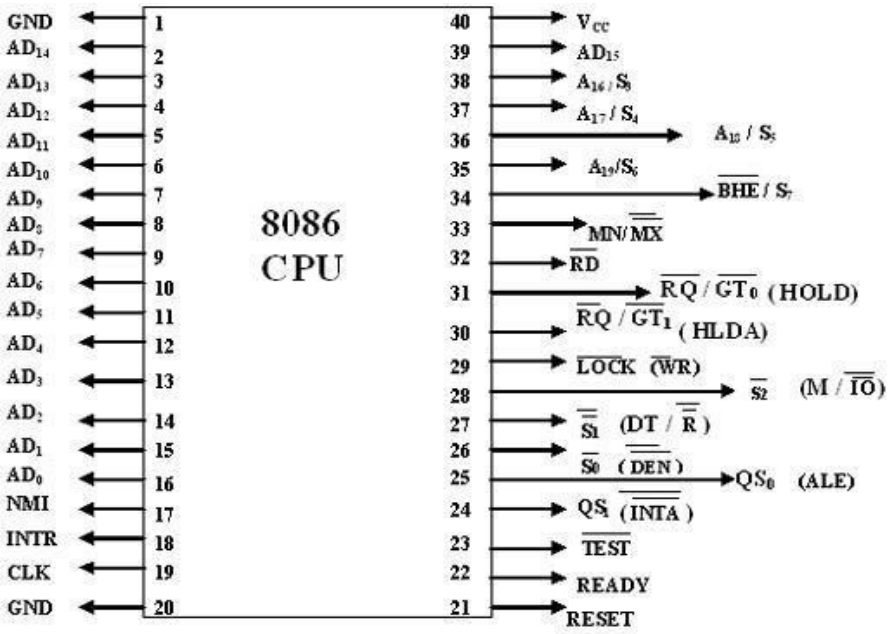
8086 FLAG REGISTER

Flags is a 16-bit register containing 9 1-bit flags:

- Overflow Flag (OF) - set if the result is too large positive number, or is too small negative number to fit into destination operand.
- Direction Flag (DF) - if set then string manipulation instructions will auto-decrement index registers. If cleared then the index registers will be auto-incremented.
- Interrupt-enable Flag (IF) - setting this bit enables maskable interrupts.
- Single-step Flag (TF) - if set then single-step interrupt will occur after the next instruction.
- Sign Flag (SF) - set if the most significant bit of the result is set.
- Zero Flag (ZF) - set if the result is zero.
- Auxiliary carry Flag (AF) - set if there was a carry from or borrow to bits 0-3 in the AL register.
- Parity Flag (PF) - set if parity (the number of "1" bits) in the low-order byte of the result is even.

Carry Flag (CF) - set if there was a carry from or borrow to the most significant bit during last result calculation

Signal description of 8086



The Microprocessor 8086 is a 16-bit CPU available in different clock rates and packaged in a 40 pin CERDIP or plastic package.

•The 8086 operates in single processor or multiprocessor configuration to achieve high performance. The pins serve a particular function in minimum mode (single processor mode) and other function in maximum mode configuration (multiprocessor mode).

•The 8086 signals can be categorized in three groups. The first are the signal having common functions in minimum as well as maximum mode

•The second are the signals which have special functions for minimum mode and third are the signals having special functions for maximum mode.

The following signal descriptions are common for both modes.

AD15-AD0 : These are the time multiplexed memory I/O address and data lines.

• Address remains on the lines during T1 state, while the data is available on the data bus during T2, T3, Tw and T4.

• These lines are active high and float to a tristate during interrupt acknowledge and local bus hold acknowledge cycles

A19/S6,A18/S5,A17/S4,A16/S3 : These are the time multiplexed address and status lines.

• During T1 these are the most significant address lines for memory operations.

•During I/O operations, these lines are low. During memory or I/O operations, status information is available on those lines for T2,T3,Tw and T4.

• The status of the interrupt enable flag bit is updated at the beginning of each clock cycle.

•The S4 and S3 combinedly indicate which segment register is presently being used for memory accesses as in below fig.

•These lines float to tri-state off during the local bus hold acknowledge. The status line S6 is always low .

•The address bit are separated from the status bit using latches controlled by the ALE signal.

BHE /S7 : The bus high enable is used to indicate the transfer of data over the higherorder (D15-D8) data bus as shown in table. It goes low for the data transfer over D15- D8 and is used to derive chip selects of odd address memory bank or peripherals. BHE is low during T1 for read, write and interrupt acknowledge cycles, whenever a byte is to be transferred on higher byte of data bus. The status informatIon is available during T2, T3 and T4. The signal is active low and tristated during hold. It is low during T1 for the first pulse of the interrupt acknowledge cycle.

BHE	A_0	Indication
0	0	Whole word
0	1	Upper byte from or to even address
1	0	Lower byte from or to even address
1	1	None

RD – Read : This signal on low indicates the peripheral that the processor is performing s memory or I/O read operation. RD is active low and shows the state for T2, T3, Tw of any read cycle. The signal remains tristated during the hold acknowledge.

•**READY** : This is the acknowledgement from the slow device or memory that they have completed the data transfer. The signal made available by the devices is synchronized by the 8284A clock generator to provide ready input to the 8086. the signal is active high.

•**INTR-Interrupt Request** : This is a triggered input. This is sampled during the last clock cycles of each instruction to determine the availability of the request. If any interrupt request is pending, the processor enters the interrupt acknowledge cycle.

•This can be internally masked by resulting the interrupt enable flag. This signal is

active high and internally synchronized.

• **TEST** : This input is examined by a 'WAIT' instruction. If the TEST pin goes low, execution will continue, else the processor remains in an idle state. The input is synchronized internally during each clock cycle on leading edge of clock.

•**CLK**- Clock Input : The clock input provides the basic timing for processor operation and bus control activity. Its an asymmetric square wave with 33% duty cycle

MN/ MX : The logic level at this pin decides whether the processor is to operate in either minimum or maximum mode.

•The following pin functions are for the minimum mode operation of 8086.

•**M/ IO – Memory/IO** : This is a status line logically equivalent to S2 in maximum mode. When it is low, it indicates the CPU is having an I/O operation, and when it is high, it indicates that the CPU is having a memory operation. This line Becomes active high in the previous T4 and remains active till final T4 of the current cycle. It is tri stated during local bus "hold acknowledge ".

INTA – Interrupt Acknowledge : This signal is used as a read strobe for interrupt acknowledge cycles. i.e. when it goes low, the processor has accepted the interrupt.

ALE – Address Latch Enable :This output signal indicates the availability of the valid address on the address/data lines, and is connected to latch enable input of latches. This signal is active high and is never tri stated.

•**DT/ R – Data Transmit/Receive**: This output is used to decide the direction of data flow through the transceivers (bidirectional buffers). When the processor sends out data, this signal is high and when the processor is receiving data, this signal is low.

•**DEN – Data Enable** :This signal indicates the availability of valid data over the address/data lines. It is used to enable the transceivers (bi directional buffers) to separate the data from the multiplexed address/data signal. It is active from the middle of T2 until the middle of T4. This is tri stated during hold acknowledge' cycle.

HOLD, HLDA- Acknowledge : When the HOLD line goes high, it indicates to the processor that another master is requesting the bus access.•The processor, after receiving the HOLD request, issues the hold acknowledge signal on HLDA pin, in the

middle of the next clock cycle after completing the current bus cycle.•At the same time, the processor floats the local bus and control lines. When the processor detects the HOLD line low, it lowers the HLD**A** signal. HOLD is an asynchronous input, and is should be externally synchronized.•If the DMA request is made while the CPU is performing a memory or I/O cycle, it will release the local bus during T4 provided :

1.The request occurs on or before T2 state of the current cycle.

2.The current cycle is not operating over the lower byte of a word.

3.The current cycle is not the first acknowledge of an interrupt acknowledge sequence.

4. A Lock instruction is not being executed

The following pin function are applicable for maximum mode operation of 8086.

•**S2, S1, S0 – Status Lines** : These are the status lines which reflect the type of operation, being carried out by the processor. These become activity during T4 of the previous cycle and active during T1 and T2 of the current bus cycles.

S_2	S_1	S_0	Indication
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Code Access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive

LOCK: This output pin indicates that other system bus master will be prevented from gaining the system bus, while the LOCK signal is low

•The LOCK signal is activated by the 'LOCK' prefix instruction and remains active until the completion of the next instruction. When the CPU is executing a critical instruction which requires the system bus, the LOCK prefix instruction ensures that other processors connected in the system will not gain the control of the bus.

• The 8086, while executing the prefixed instruction, asserts the bus lock signal output, which may be connected to an external bus controller.

•**QS1, QS0 – Queue Status:** These lines give information about the status of the code-prefetch queue. These are active during the CLK cycle after while the queue operation is

performed.

•This modification in a simple fetch and execute architecture of a conventional microprocessor offers an added advantage of pipelined processing of the instructions.

•The 8086 architecture has 6-byte instruction prefetch queue. Thus even the largest (6-bytes) instruction can be prefetched from the memory and stored in the prefetch. This results in a faster execution of the instructions.

•In 8085 an instruction is fetched, decoded and executed and only after the execution of this instruction, the next one is fetched.

•By prefetching the instruction, there is a considerable speeding up in instruction execution in 8086. This is known as ***instruction pipelining***.

•At the starting the CS:IP is loaded with the required address from which the execution is to be started. Initially, the queue will be empty an the microprocessor starts a fetch operation to bring one byte (the first byte) of instruction code, if the CS:IP address is odd or two bytes at a time, if the CS:IP address is even.

•The first byte is a complete opcode in case of some instruction (one byte opcode instruction) and is a part of opcode, in case of some instructions (two byte opcode instructions), the remaining part of code lie in second byte.

•The second byte is then decoded in continuation with the first byte to decide the instruction length and the number of subsequent bytes to be treated as instruction data.

•The queue is updated after every byte is read from the queue but the fetch cycle is initiated by BIU only if at least two bytes of the queue are empty and the EU may be concurrently executing the fetched instructions

• **RQ / GT0 , RQ / GT1 – Request/Grant :** These pins are used by the other local bus master in maximum mode, to force the processor to release the local bus at the end of the processor current bus cycle.

•Each of the pin is bidirectional with RQ/GT0 having higher priority than RQ/GT1.

•RQ/GT pins have internal pull-up resistors and may be left unconnected.

Request/Grant sequence is as follows:

1.A pulse of one clock wide from another bus master requests the bus access to 8086.

2.During T4(current) or T1(next) clock cycle, a pulse one clock wide from 8086 to the requesting master, indicates that the 8086 has allowed the local bus to float and that it will enter the 'hold acknowledge' state at next cycle. The CPU bus interface unit is likely to be disconnected from the local bus of the system.

3.A one clock wide pulse from the another master indicates to the 8086 that the hold request is about to end and the 8086 may regain control of the local bus at the next clock cycle. Thus each master to master exchange of the local bus is a sequence of 3 pulses. There must be at least one dead clock cycle after each bus exchange.

•The request and grant pulses are active low.

•For the bus request those are received while 8086 is performing memory or I/O cycle, the granting of the bus is governed by the rules as in case of HOLD and HLDA in minimum mode.

General Bus Operation:

•The 8086 has a combined address and data bus commonly referred as a time multiplexed address and data bus.

• The main reason behind multiplexing address and data over the same pins is the maximum utilization of processor pins and it facilitates the use of 40 pin standard DIP package. Maximum utilization of processor pins and it facilitates the use of 40 pin standard DIP package.

•The bus can be de multiplexed using a few latches and transceivers, when ever required.

•Basically, all the processor bus cycles consist of at least four clock cycles. These are refered to as T1, T2, T3, T4. The address is transmitted by the processor during T1. It is present on the bus only for one cycle.

•The negative edge of this ALE pulse is used to separate the address and the data or status

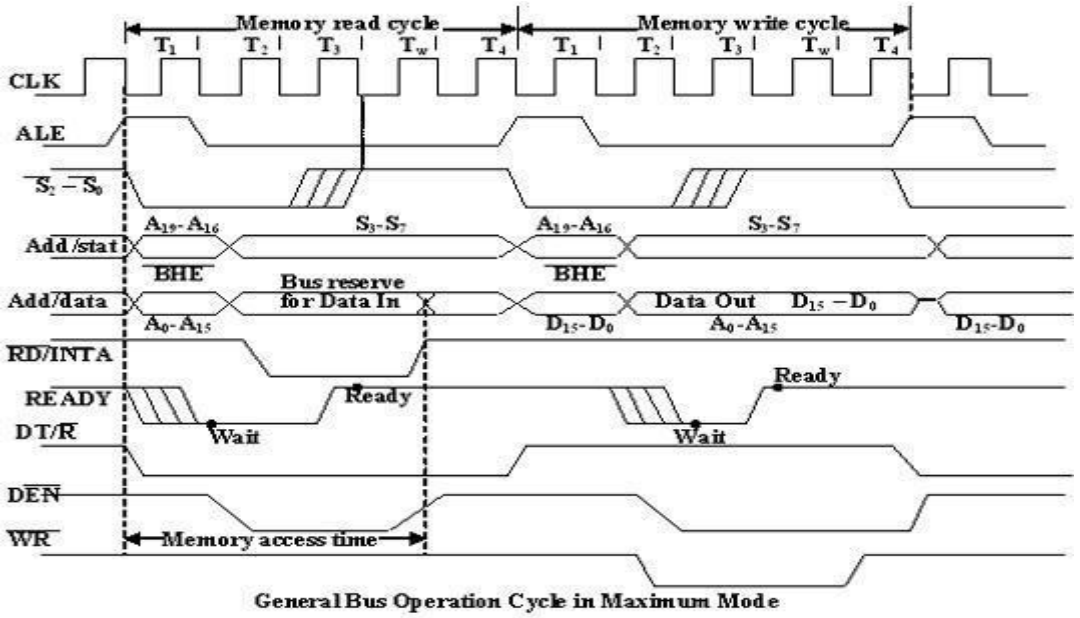
information. In maximum mode, the status lines S0, S1 and S2 are used to indicate the type of operation.

•In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/MX pin to logic 1.

•In this mode, all the control signals are given out by the microprocessor chip itself. There is a single microprocessor in the minimum mode system.

•The remaining components in the system are latches, transreceivers, clock generator, memory and I/O devices. Some type of chip selection logic may be required for selecting memory or I/O devices, depending upon the address map of the system.

•In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/MX pin to logic 1.



General Bus Operation Cycle in Maximum Mode

•In this mode, all the control signals are given out by the microprocessor chip itself. There is a single microprocessor in the minimum mode system.

•The remaining components in the system are latches, transreceivers, clock generator, memory and I/O devices. Some type of chip selection logic may be required for selecting memory or I/O devices, depending upon the address map of the system.

•Latches are generally buffered output D-type flip-flops like 74LS373 or 8282. They are used for separating the valid address from the multiplexed address/data signals and arecontrolled by the ALE signal generated by 8086.

•Transreceivers are the bidirectional buffers and some times they are called as data amplifiers. They are required to separate the valid data from the time multiplexed address/data signals.

•They are controlled by two signals namely, DEN and DT/R The DEN signal indicates the direction of data, i.e. from or to the processor. The system contains memory for the monitor and users program storage.

•Usually, EPROM are used for monitor storage, while RAM for users program storage. A system may contain I/O devices.

•The working of the minimum mode configuration system can be better described in terms of the timing diagrams rather than qualitatively describing the operations.

•The opcode fetch and read cycles are similar. Hence the timing diagram can be categorized in two parts, the first is the timing diagram for read cycle and the second is the timing diagram for write cycle.

•The read cycle begins in T1 with the assertion of address latch enable (ALE) signal and

also M / IO signal. During the negative going edge of this signal, the valid address is latched on the local bus.

•The BHE and A0 signals address low, high or both bytes. From T1 to T4 , the M/IO signal indicates a memory or I/O operation.

•At T2, the address is removed from the local bus and is sent to the output. The bus is then tristated. The read (RD) control signal is also activated in T2.

•The read (RD) signal causes the address device to enable its data bus drivers. After RD goes low, the valid data is available on the data bus.

•The addressed device will drive the READY line high. When the processor returns the read signal to high level, the addressed device will again tristate its bus drivers.

•A write cycle also begins with the assertion of ALE and the emission of the address. The M/IO signal is again asserted to indicate a memory or I/O operation. In T2, after sending the address in T1, the processor sends the data to be written to the addressed location.

•The data remains on the bus until middle of T4 state. The WR becomes active at the beginning of T2 (unlike RD is somewhat delayed in T2 to provide time for floating).

•The BHE and A0 signals are used to select the proper byte or bytes of memory or I/O word to be read or write.

•The M/IO, RD and WR signals indicate the type of data transfer as specified in table below.

Hold Response sequence: The HOLD pin is checked at leading edge of each clock pulse. If it is received active by the processor before T4 of the previous cycle or during T1 state of the current cycle, the CPU activates HLDA in the next clock cycle and for succeeding bus cycles, the bus will be given to another requesting master.

•The control of the bus is not regained by the processor until the requesting master does not drop the HOLD pin low. When the request is dropped by the requesting master, the HLDA is dropped by the processor at the trailing edge of the next clock.

Maximum Mode 8086 System:

•In the maximum mode, the 8086 is operated by strapping the MN/MX pin to ground.

•In this mode, the processor derives the status signal S2, S1, S0. Another chip called bus controller derives the control signal using this status information .

•In the maximum mode, there may be more than one microprocessor in the system configuration.

•The components in the system are same as in the minimum mode system.

•The basic function of the bus controller chip IC8288, is to derive control signals like RD and WR (for memory and I/O devices), DEN, DT/R, ALE etc. using the information by the processor on the status lines.

•The bus controller chip has input lines S2, S1, S0 and CLK. These inputs to 8288 are driven by CPU.

•It derives the outputs ALE, DEN, DT/R, MRDC, MWTC, AMWC, IORC, IOWC and AIOWC. The AEN, IOB and CEN pins are specially useful for multiprocessor systems.

•AEN and IOB are generally grounded. CEN pin is usually tied to +5V. The significance of the MCE/PDEN output depends upon the status of the IOB pin.

•If IOB is grounded, it acts as master cascade enable to control cascade 8259A, else it acts as peripheral data enable used in the multiple bus configurations.

•INTA pin used to issue two interrupt acknowledge pulses to the interrupt controller or to an interrupting device

IORC, IOWC are I/O read command and I/O write command signals respectively .These signals enable an IO interface to read or write the data from or to the address port.

•The MRDC, MWTC are memory read command and memory write command signals respectively and may be used as memory read or write signals.

•All these command signals instructs the memory to accept or send data from or to the bus.

•For both of these write command signals, the advanced signals namely AIOWC and AMWTC are available.

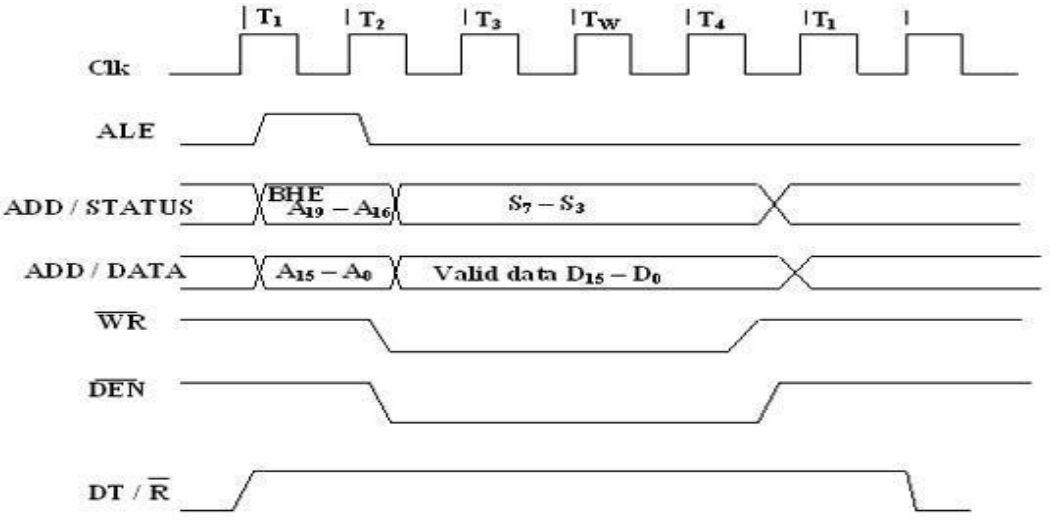
•Here the only difference between in timing diagram between minimum mode and maximum mode is the status signals used and the available control and advanced command signals.

•R0, S1, S2 are set at the beginning of bus cycle.8288 bus controller will output a pulse as on the ALE and apply a required signal to its DT / R pin during T1.

•In T2, 8288 will set DEN=1 thus enabling transceivers, and for an input it will activate MRDC or IORC. These signals are activated until T4. For an output, the AMWC or AIOWC is activated from T2 to T4 and MWTC or IOWC is activated from T3 to T4.

•The status bit S0 to S2 remains active until T3 and become passive during T3 and T4.

•If reader input is not activated before T3, wait state will be inserted between T3 and T4.



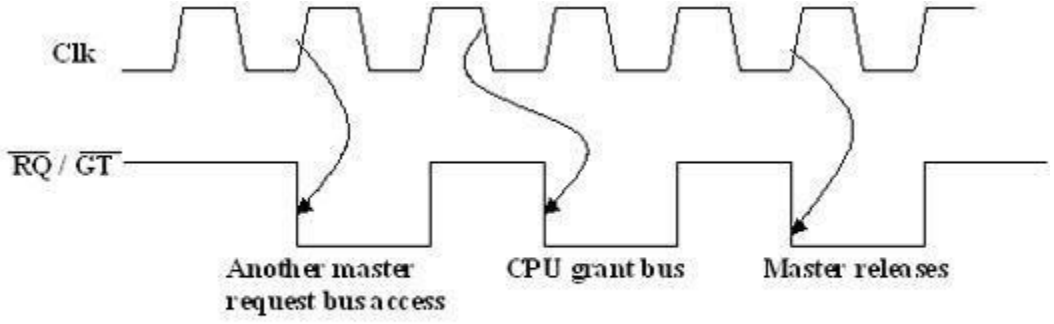
Write Cycle Timing Diagram for Minimum Mode

Timings for RQ/ GT Signals :

The request/grant response sequence contains a series of three pulses. The request/grant pins are checked at each rising pulse of clock input.

When a request is detected and if the condition for HOLD request is satisfied, the processor issues a grant pulse over the RQ/GT pin immediately during T4 (current) or T1 (next) state.

When the requesting master receives this pulse, it accepts the control of the bus; it sends a release pulse to the processor using RQ/GT pin.



RQ/GT Timings in Maximum Mode.

Minimum Mode Interface:

•When the Minimum mode operation is selected, the 8086 provides all control signals

needed to implement the memory and I/O interface.The minimum mode signal can be divided into the following basic groups : address/data bus, status, control, interrupt and DMA.

•**Address/Data Bus** : these lines serve two functions. As an address bus is 20 bits long and consists of signal lines A0 through A19. A19 represents the MSB and A0 LSB. A 20bit address gives the 8086 a 1Mbyte memory address space. More over it has an independent

I/O address space which is 64K bytes in length.

•The 16 data bus lines D0 through D15 are actually multiplexed with address lines A0 through A15 respectively. By multiplexed we mean that the bus work as an address bus during first machine cycle and as a data bus during next machine cycles. D15 is the MSB and D0 LSB.

•When acting as a data bus, they carry read/write data for memory, input/output data for

I/O devices, and interrupt type codes from an interrupt controller.

S_4	S_3	Segment Register
0	0	Extra
0	1	Stack
1	0	Code / none
1	1	Data

Memory segment status codes.

Status signal:

The four most significant address lines A19 through A16 are also multiplexed but in this case with status signals S6 through S3. These status bits are output on the bus at the same

time that data are transferred over the other bus lines.

•Bit S4 and S3 together from a 2 bit binary code that identifies which of the 8086

internal
segment registers are used to generate the physical address that was output on the address
bus during the current bus cycle.

•Code S4S3 = 00 identifies a register known as ***extra segment register*** as the source of the segment address.
•Status line S5 reflects the status of another internal characteristic of the 8086. It is the logic level of the internal enable flag. The last status bit S6 is always at the logic 0 level

•**Control Signals** :

The control signals are provided to support the 8086 memory I/O interfaces. They control functions such as when the bus is to carry a valid address in which direction data
are to be transferred over the bus, when valid write data are on the bus and when to put read data on the system bus.
ALE is a pulse to logic 1 that signals external circuitry when a valid address word is on the bus. This address must be latched in external circuitry on the 1-to-0 edge of the pulse
•Another control signal that is produced during the bus cycle is BHE bank high enable. Logic 0 on this used as a memory enable signal for the most significant byte half of the data bus D8 through D1. These lines also serves a second function, which is as the S_7 status line.
•Using the M/IO and DT/R lines, the 8086 signals which type of bus cycle is in progress and in which direction data are to be transferred over the bus.
•The logic level of M/IO tells external circuitry whether a memory or I/O transfer is taking place over the bus. Logic 1 at this output signals a memory operation and logic 0 an I/O operation.
•The direction of data transfer over the bus is signaled by the logic level output at DT/R. When this line is logic 1 during the data transfer part of a bus cycle, the bus is in the transmit mode. Therefore, data are either written into memory or output to an I/O device.

•On the other hand, logic 0 at DT/R signals that the bus is in the receive mode. This corresponds to reading data from memory or input of data from an input port.

•The signal read RD and write WR indicates that a read bus cycle or a write bus cycle is in progress. The 8086 switches WR to logic 0 to signal external device that valid write or output data are on the bus

• On the other hand, RD indicates that the 8086 is performing a read of data of the bus. During read operations, one other control signal is also supplied. This is DEN (data enable) and it signals external devices when they should put data on the bus.

•There is one other control signal that is involved with the memory and I/O interface. This is the READY signal.

•READY signal is used to insert wait states into the bus cycle such that it is extended by a number of clock periods. This signal is provided by an external clock generator device and can be supplied by the memory or I/O sub-system to signal the 8086 when they are ready to permit the data transfer to be completed

Maximum Mode Interface

•When the 8086 is set for the maximum-mode configuration, it provides signals for implementing a multiprocessor / coprocessor system environment.

•By multiprocessor environment we mean that one microprocessor exists in the system and that each processor is executing its own program.

• Usually in this type of system environment, there are some system resources that are common to all processors.

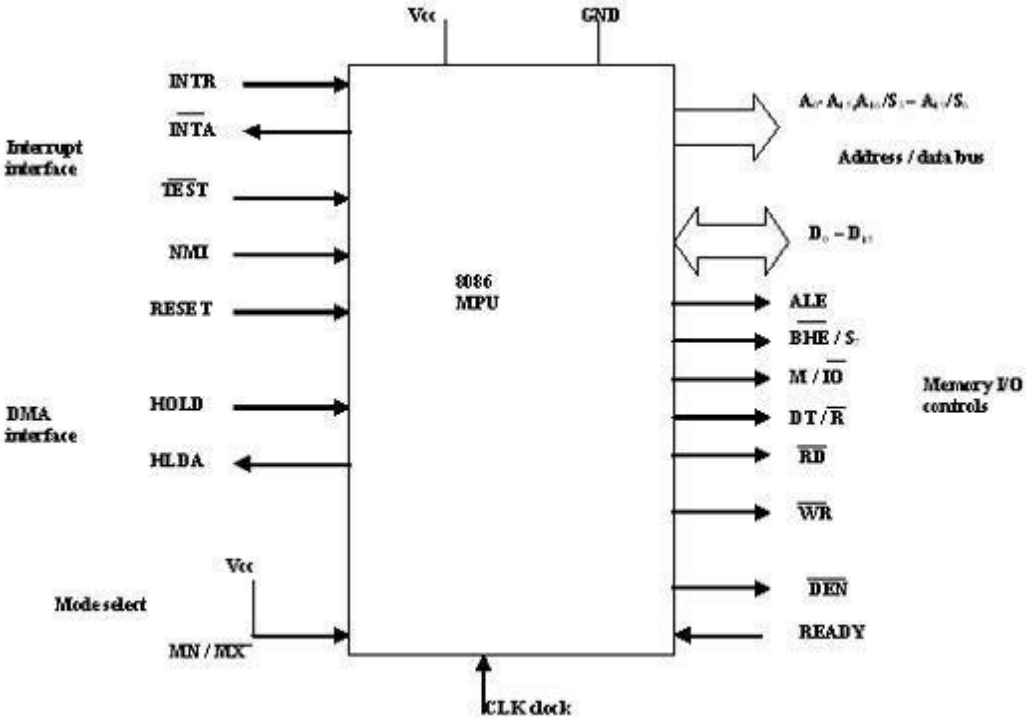
•They are called as ***global resources***. There are also other resources that are assigned to

specific processors. These are known as ***local or private resources***.

•Coprocessor also means that there is a second processor in the system. In this two processor does not access the bus at the same time.

•One passes the control of the system bus to the other and then may suspend its operation.

•In the maximum-mode 8086 system, facilities are provided for implementing allocation of global resources and passing bus control to other microprocessor or coprocessor .



Block Diagram of the Minimum Mode 8086 MPU

ADDRESSING MODES OF 8086:

Implied - the data value/data address is implicitly associated with the instruction.

Direct - the instruction operand specifies the memory address where data is located.

•**Register indirect** - instruction specifies a register containing an address, where data is located. This addressing mode works with SI, DI, BX and BP registers.

•**Register** - references the data in a register or in a register pair.

•**Immediate** - the data is provided in the instruction.

•**Based** :- 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP), the resulting value is a pointer to location where data resides.

•**Indexed** :- 8-bit or 16-bit instruction operand is added to the contents of an index register (SI or DI), the resulting value is a pointer to location where data resides

•**Based Indexed** :- the contents of a base register (BX or BP) is added to the contents of an index register (SI or DI), the resulting value is a pointer to location where data resides.

•**Based Indexed with displacement** :- 8-bit or 16-bit instruction operand is added to the contents of a base register (BX or BP) and index register (SI or DI), the resulting value is a pointer to location where data resides.

Instruction set of 8086:

Data transfer instructions

GENERAL – PURPOSE BYTE OR WORD TRANSFER INSTRUCTIONS:

MOV

PUSH

POP

XCHG

XLAT

SIMPLE INPUT AND OUTPUT PORT TRANSFER INSTRUCT

IN

OUT

SPECIAL ADDRESS TRANSFER INSTRUCTIONS

LEA

LDS

LES

FLAG TRANSFER INSTRUCTIONS:

LAHF

SAHF

PUSHF

POPF

ADITION INSTRUCTIONS:

ADD

ADC

INC

AAA

DAA

SUBTSUBTRACTION INSTRUCTIONS:

SUB

SBB

DEC

NEG

CMP

AAS

DAS

MULTIPLICATION INSTRUCTIONS:

MUL

IMUL

AAM

DIVISION INSTRUCTIONS:

DIV

IDIV

AAD

CBW

CWD

BIT MANIPULATION INSTRUCTIONS

LOGICAL INSTRUCTIONS:

NOT

AND

OR

XOR

TEST

SHIFT INSTRUCTIONS:

SHL / SAL

SHR

SAR

PROGRAM EXECUTION TRANSFER INSTRUCTIONS

UNCONDITIONAL TRANSFER INSTRUCTIONS:

CALL

RET

JMP

CONDITIONAL TRANSFER INSTRUCTIONS:

JA / JNBE

JAE / JNB

JB / JNAE

JBE / JNA

JC

JE / JZ

JG / JNLE

JGE / JNL

JL / JNGE

JLE / JNG

JNC

JNE / JNZ

JNO

JNP / JPO

JNS

JO

JP / JPE

JS

ITERATION CONTROL INSTRUCTIONS:

LOOP

LOOPE / LOOPZ

LOOPNE / LOOPNZ

JCXZ

INTERRUPT INSTRUCTIONS:

INT

INTO

IRET

PROCESS CONTROL INSTRUCTIONS

FLAG SET / CLEAR INSTRUCTIONS:

STC

CLC

CMC

STD

CLD

STI

CLI

EXTERNAL HARDWARE SYNCHRONIZATION INSTRUCTIONS:

H

L

T

W

A

I

T

E

S

C

L

O

C

K

N

O

P

Instruction Description

AAA Instruction - ASCII Adjust after Addition

AAD Instruction - ASCII adjust

before Division **AAM**

Instruction - ASCII adjust after

Multiplication **AAS** Instruction

- ASCII Adjust for Subtraction

ADC Instruction - Add with carry.

ADD Instruction - ADD destination, source

AND Instruction - AND corresponding bits of two operands

ASSEMBLER DIRECTIVES:

An Assembler is a program used to convert an assembly language program in to equivalent machine code modules which may further be converted in to executable codes. The assembler decides the address of each label and substitutes the values for each of the constants and variables. It then forms the machine code for the mnemonics and data in assembly language program. While doing all these things, the assembler may find out Syntax errors. The logical errors or other programming errors are not found by the assembler. For completing all these tasks, an assembler needs some hints from the programmer, i.e. the required storage for a particular constant or variable, logical names of the segments, types of the different routines and modules, end of file etc. These types of hints are given to assembler using some predefined alphabetical strings called "ASSEMBLER DIRECTIVES".

DB:

This is used to reserve byte or bytes of memory locations in the available memory. While preparing the EXE file, this directive directs the assembler to allocate the specified number of memory bytes to the said data type that may be constant, variable or stringent.

DW:

This directive serves the same purposes as the DB directive, but it now makes the assembler reserve the number of memory words instead of bytes.

DQ:

This directive is used to direct the assembler to reserve words of memory for the specified variable and may initialize it with the specified values.

ASSUME:

The ASSUME directive is used to inform the assemble, the names of the logical segments to be assumed for different segments used in the program.

END:

The END directive marks the end of an assembly language program. When the assembler comes across this directive, it ignores the source lines available later on.

ENDP:

In assembly language programming, the subroutines are called procedures. Thus, procedures may be
Independent program modules which return particular results or values to the calling programs. The ENDP
Directive is used to indicate an end of procedures. A procedure is usually assigned a name, i.e. label. To mark
The end of a particular procedure, the name of the procedure, i.e label appear as a prefix with the directive
ENDP.

ENDS:

This directive marks the end of a logical segment. The logical segments are assigned with names
Using the ASSUME directive. The names appear with the ENDS directive as prefixes to mark the end of
Those particular segments.

LABEL:

This is used to assign a name to the current content of the location counter. At the start of the assembly process, the assembler initializes a location counter to keep track of memory Locations assigned to the program. As the program assembly proceeds, the contents of the
Location counter are updated.
0

LENGTH:

This is used to refer the length of a data array or string.

OFFSET:

When the assembler comes across the offset operator along with a label, it first computes the 6-bit
Displacement of the particular label, and replaces the string ‘offset label’ by the computed displacement.

SEGMENT:

This marks the staring of a logical segment .The started segment is also assigned a name
i.e., label by this statement. The SEGMENT and ENDS directive must bracket each logical Segment of a program.

UNIT -2

Advantages of Assembly language program over machine level program:

1. Programs written in machine language are replaceable by mnemonics which are easier to remember.
2. Memory Efficient.
3. It is not required to keep track of memory locations.
4. Faster in speed.
5. Easy to make insertions and deletions.
6. Hardware Oriented.
7. Requires fewer instructions to accomplish the same result.

Introduction To Masm/Tasm

The assembler is a program that converts an assembly input file also called source file to an object file that can further be converted in to machine codes or an executable file using a linker. There are a number of assemblers available like MASM, TASM and DOS assembler. TASM is one of the popular assemblers used along with a link program to structure the codes generated By TASM in the form of executable files.

TASM reads the source program as its input and provides an object file. The linker accepts the object file produced by TASM as input and produces an EXE file.

Before staring the process ensure that all the files namely, TASM, .EXE, LINK .EXE, DEBUG. EXE are available in the same directory which is working. Start the procedure with the following command after boot the terminal and enter the directory containing all the files mentioned.

Examples of Assembly language programs:

MULTIBYTE ADDITION:

```
ALP:

ASSUME CS: CODE, DS: DATA

DATA SEGMENT

OPR1 DD 12345678H

OPR2 DD 12345678H

RES DD ?
```

```
DATA ENDS

CODE SEGMENT

START: MOV AX, DATA

MOV DS, AX

LEA SI, OPR1

LEA BX, OPR2

LEA DI, RES

MOV CX, 0004H

LOOP: MOV AL, [SI]

ADC AL, [BX]

MOV [DI], AL

INC SI

INC BX

INC DI

DEC CX

JNZ LOOP

HLT

CODE ENDS

END START
```

Multi byte Subtraction:

```
ALP:
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
OPR1 DD 12345678H
OPR2 DD 02345678H
RES DD ?
DATA ENDS
CODE SEGMENT
START: MOV AX, DATA
MOV DS, AX
LEA SI, OPR1
LEA BX, OPR2
LEA DI, RES
MOV CX, 0004H
```

```
LOOP: MOV AL, [SI]
SBB AL, [BX]
MOV [DI], AL
INC SI
INC BX
INC DI
DEC CX
JNZ LOOP
HLT
CODE ENDS
END START
```

Interrupt Cycle of 8086/88

An 8086 interrupt can come from any one the three sources :

- External signal
- Special Instruction in the program
- Condition produced by instruction

8.2.1 External Signal (Hardware Interrupt)

An 8086 can get interrupt from an external signal applied to the nonmaskable interrupt (NMI) input pin, or the interrupt (INTR) input pin.

8.2.2 Special Instruction

8086 supports a special instruction, INT to execute special program. At the end of the interrupt service routine, execution is usually returned to the interrupted program.

8.2.3 Condition Produced by Instruction

An 8086 is interrupted by some condition produced in the 8086 by the execution of an instruction. For example divide by zero : Program execution will automatically be interrupted if you attempt to divide an operand by zero.

At the end of each instruction cycle 8086 checks to see if there is any interrupt request. If so, 8086 responds to the interrupt by performing series of actions (Refer Fig. 8.1).

1. It decrements stack pointer by 2 and pushes the flag register on the stack .
2. It disables the INTR interrupt input by clearing the interrupt flag in the flag register.
3. It resets the trap flag in the flag register.
4. It decrements stack pointer by 2 and pushes the current code segment register contents on the stack.
5. It decrements stack pointer by 2 and pushes the current instruction pointer contents on the stack.
6. It does an indirect far jump at the start of the procedure by loading the CS and IP values for the start of the interrupt service routine (ISR).

An IRET instruction at the end of the interrupt service procedure returns execution to the main program.

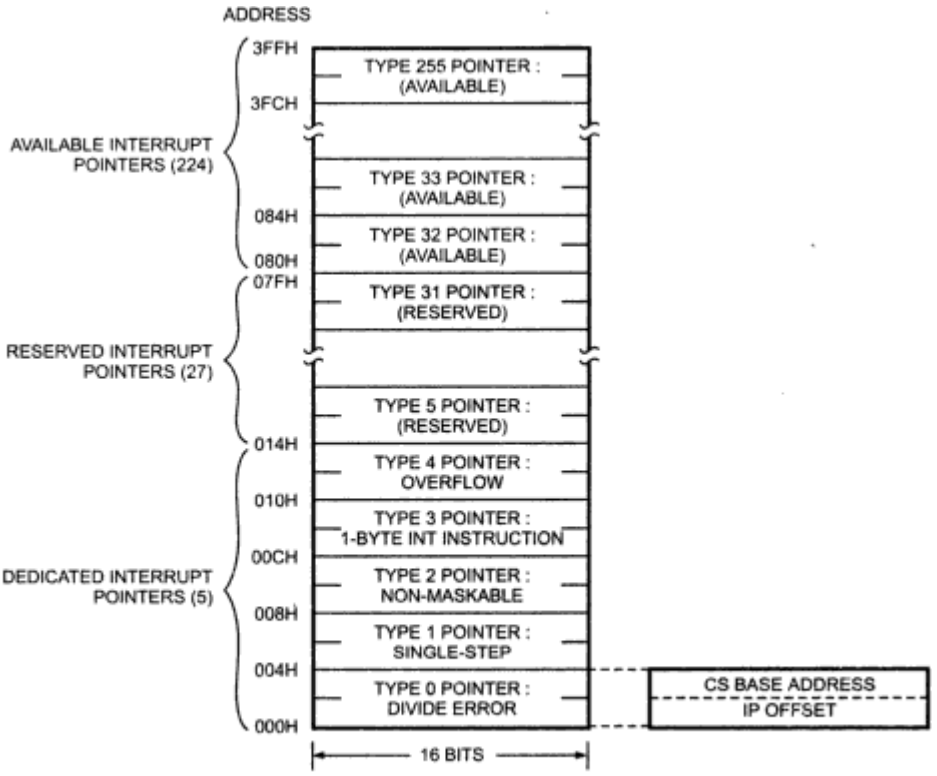


Fig. 8.2 8086 interrupt vector table

8086 Interrupt Types

Divide by Zero Interrupt (Type 0)

When the quotient from either a DIV or IDIV instruction is too large to fit in the result gister; 8086 will automatically execute type 0 interrupt.

Single Step Interrupt (Type 1)

The type 1 interrupt is the single step trap. In the single step mode, system will execute one instruction and wait for further direction from user. Then user can examine the contents of registers and memory locations and if they are correct, user can tell the system to execute the next instruction. This feature is useful for debugging assembly language programs.

Software Interrupts

Type 0 - 255 :

The 8086 INT instruction can be used to cause the 8086 to do one of the 256 possible interrupt types. The interrupt type is specified by the number as a part of the instruction. You can use an INT2 instruction to send execution to an NMI interrupt service routine. This allows you to test the NMI routine without needing to apply an external signal to the NMI input of the 8086.

With the software interrupts you can call the desired routines from many different programs in a system e.g. BIOS in IBM PC. The IBM PC has in its ROM collection of routines, each performing some specific function such as reading character from keyboard, writing character to CRT. This collection of routines referred to as **Basic Input Output System** or **BIOS**.

The BIOS routines are called with INT instructions. We will summarize interrupt response and how it is serviced by going through following steps.

1. 8086 pushes the flag register on the stack.
2. It disables the single step and the INTR input by clearing the trap flag and interrupt flag in the flag register.
3. It saves the current CS and IP register contents by pushing them on the stack.
4. It does an indirect far jump to the start of the routine by loading the new values of CS and IP register from the memory whose address calculated by multiplying 4 to the interrupt type, For example, if interrupt type is 4 then memory address is $4 \times 4 = 10_{10} = 10H$. So 8086 will read new value of IP from 00010H and CS from 00012H.
5. Once these values are loaded in the CS and IP, 8086 will fetch the instruction from the new address which is the starting address of interrupt service routine.
6. An IRET instruction at the end of the interrupt service routine gets the previous values of CS and IP by popping the CS and IP from the stack.
7. At the end the flag register contents are copied back into flag register by popping the flag register form stack.

Priority Resolver

The priority resolver determines the priorities of the bits set in the IRR. The bit corresponding to the highest priority interrupt input is set in the ISR during the $\overline{INTA}$ input.

Cascade Buffer Comparator

This section generates control signals necessary for cascade operations. It also generates Buffer-Enable signals. As stated earlier, the 8259 can be cascaded with other 8259s in order to expand the interrupt handling capacity to sixty-four levels. In such a case, the former is called a **master**, and the latter are called **slaves**. The 8259 can be set up as a master or a slave by the $\overline{SP}$ / $\overline{EN}$ pin.

CAS 0 - 2

For a master 8259, the CAS_0-CAS_2 pins are outputs, and for slave 8259s, these are inputs. When the 8259 is a master (that is, when it accepts interrupt requests from other 8259s), the CALL opcode is generated by the Master in response to the first $\overline{INTA}$. The vectoring address must be released by the slave 8259. The master sends an identification code of three-bits (to select one out of the eight possible slave 8259s) on the CAS_0-CAS_2 lines. The slave 8259s accept these three signals as inputs (on their CAS_0-CAS_2 pins) and compare the code sent by the master with the codes assigned to them during initialisation. The slave thus selected (which had originally placed an interrupt request to the master 8259) then puts out the address of the interrupt service routine during the second and third $\overline{INTA}$ pulses from the CPU.

$\overline{SP}$ / $\overline{EN}$ (Slave Program /Enable Buffer)

The $\overline{SP}$ / $\overline{EN}$ signal is tied high for the master. However, it is grounded for the slave.

In large systems where buffers are used to drive the data bus, the data sent by the 8259 in response to $\overline{INTA}$ cannot be accessed by the CPU (due to the data bus buffer being disabled).

If an 8259 is used in the buffered mode (buffered or non-buffered modes of operation can be specified at the time of initialising the 8259), the $\overline{SP}$ / $\overline{EN}$ pin is used as an output which can be used to enable the system data bus buffer whenever the 8259's data bus outputs are enabled (when it is ready to send data).

Means, in non-buffered mode, the $\overline{SP}/\overline{EN}$ pin of an 8259 is used to specify whether the 8259 is to operate as a master or as a slave, and in the buffered mode, the $\overline{SP}/\overline{EN}$ pin is used as an output to enable the data bus buffer of the system.

8.5.3 Interrupt Sequence

The events occur as follows in an 8086 system :

1. One or more of the INTERRUPT REQUEST lines (IR0-IR7) are raised high, setting the corresponding IRR bit(s).

2. The priority resolver checks three registers : The IRR for interrupt requests, the IMR for masking bits, and the ISR for the interrupt request being served. It resolves the priority and sets the INT high when appropriate.
3. The CPU acknowledges the INT and responds with an $\overline{INTA}$ pulse.
4. Upon receiving an $\overline{INTA}$ from the CPU, the highest priority ISR bit is set and the corresponding IRR bit is reset. The 8259A does not drive data bus during this cycle.
5. A selection of priority modes is available to the programmer so that the manner in which the requests are processed by the 8259A can be configured to match his system requirements. The priority modes can be changed or reconfigured dynamically at any time during the main program. This means that the complete interrupt service structure can be defined as required, based on the total system environment.
6. The 8086 will initiate a second INTA pulse. During this pulse, the 8259A releases a 8-bit pointer (interrupt type) onto the Data Bus where it is read by the CPU.
7. This completes the interrupt cycle. In the AEOI mode the ISR bit is reset at the end of the second INTA pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt subroutine.

8.5.4 Priority Modes and Other Features

The various modes of operation of the 8259 are :

(a) Fully Nested Mode,

(b) Rotating Priority Mode,

(c) Special Masked Mode, and

(d) Polled Mode.

a) Fully Nested Mode :

After initialization, the 8259A operates in fully nested mode so it is called as default mode. The 8259 continues to operate in the Fully Nested Mode until the mode is changed through Operation Command Words. In this mode, IR0 has highest priority and IR7 has lowest priority. When the interrupt is acknowledged, it sets the corresponding bit in ISR. This bit will prevent all interrupts of the same or lower level, however it will accept higher priority interrupt requests. The vector address corresponding to this interrupt is then sent. The bit in the ISR will remain set until an EOI command is issued by the microprocessor at the end of interrupt service routine.

But if AEOI (Automatic End of Interrupt) bit is set, the bit in the ISR resets at the trailing edge of the last $\overline{INTA}$.

(i) Automatic Rotation

In this mode, a device, after being serviced, receives the lowest priority. Assuming that IR_3 has just been serviced, it will receive the seventh priority.

IR_0	IR_1	IR_2	IR_3	IR_4	IR_5	IR_6	IR_7
4	5	6	7	0	1	2	3

(ii) Specific Rotation

In the Automatic Rotation mode, the interrupt request last serviced is assigned the lowest priority, whereas in the Specific Rotation mode, the lowest priority can be assigned to any interrupt input (IR_0 to IR_7) thus fixes all other priorities.

For example if the lowest priority is assigned to IR_2, other priorities are as shown below.

IR_0	IR_1	IR_2	IR_3	IR_4	IR_5	IR_6	IR_7
5	6	7	0	1	2	3	4

d) Special Mask Mode :

If any interrupt is in service then the corresponding bit is set in ISR and the lower priority interrupts are inhibited. Some applications may require an interrupt service routine to dynamically alter the system priority structure during its execution under software control, for example, the routine may wish to inhibit lower priority requests for a portion of its execution but enable some of them for another portion. In these cases we have to go for special mask mode.

In the special mask mode it inhibits further interrupts at that level and enables interrupts from all other levels (lower as well as higher) that are not masked. Thus any interrupt may be selectively enabled by loading the mask register.

e) Poll Mode :

In this mode the INT output is not used. The microprocessor checks the status of interrupt requests by issuing poll command. The microprocessor reads contents of 8259A after issuing poll command. During this read operation the 8259A provides polled word and sets ISR bit of highest priority active interrupt request FORMAT.

I	X	X	X	X	W_2	W_1	W_0

$I = 1 \rightarrow$ One or more interrupt requests activated.

$I = 0 \rightarrow$ No interrupt request activated.

$W_2\ W_1\ W_0 \rightarrow$ Binary code of highest priority active interrupt request.

Programming the 8259A

The 8259A requires two types of command words. Initialization Command Words (ICWs) and Operational Command Words (OCWs).

The 8259A can be initialized with four ICWs; the first two are compulsory, and the other two are optional based on the modes being used. These words must be issued in a given sequence. After initialization, the 8259A can be set up to operate in various modes by using three different OCWs; however, they no longer need to be issued in a specific sequence.

Flow chart :

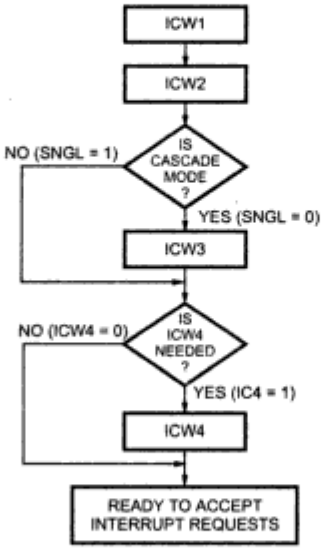


Fig. 8.7 8259 A initialization flowchart

Initialization Command Word 1 (ICW1)

Fig. 8.8 shows the Initialization Command Word 1 (ICW1).

A write command issued to the 8259 with $A_0 = 0$ and $D_4 = 1$ is interpreted as ICW1, which starts the initialization sequence.

It specifies

1. Single or multiple 8259As in the system.
2. 4 or 8 bit interval between the interrupt vector locations.
3. The address bits A_7 - A_5 of the CALL instruction.
4. Edge triggered or level triggered interrupts.
5. ICW4 is needed or not.

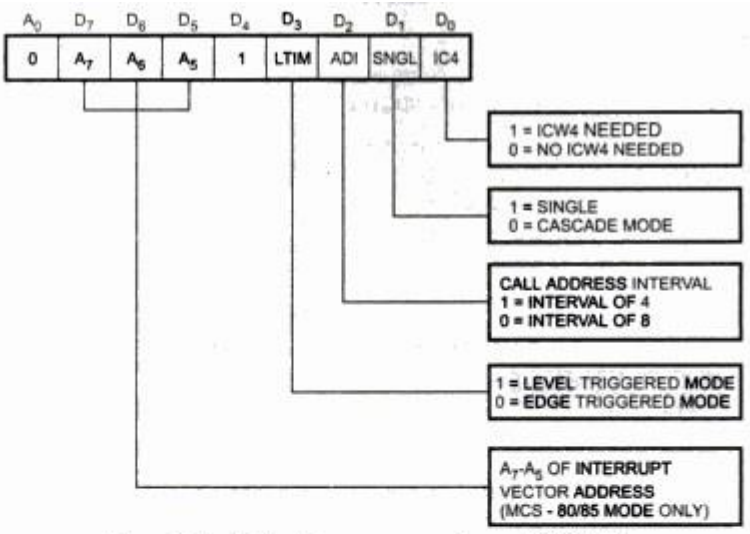


Fig. 8.8 Initialization command word 1 (ICW1)

Initialization Command Word 2 (ICW2)

Fig 8.9 shows the Initialization Command Word 2 (ICW2).

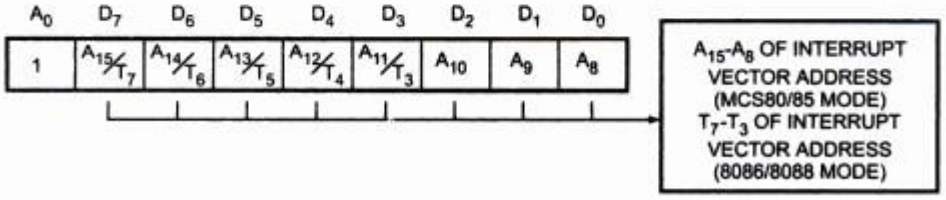


Fig. 8.9 Initialization command word 2 (ICW2)

A write command following ICW1, with A0 = 1 is interpreted as ICW2. This is used to load the high order byte of the interrupt vector address of all the interrupts.

Initialization Command Word 3 (ICW3)

ICW3 is required only if there is more than one 8259 in the system and if they are cascaded. An ICW3 operation loads a slave register in the 8259. The format of the byte to be loaded as an ICW3 for a master 8259 or a slave is shown in the Fig. 8.10. For master, each bit in ICW3 is used to specify whether it has a slave 8259 attached to it on its corresponding IR (Interrupt Request) input. For slave, bits D_0-D_2 of ICW3 are used to assign a slave identification code (slave ID) to the 8259.

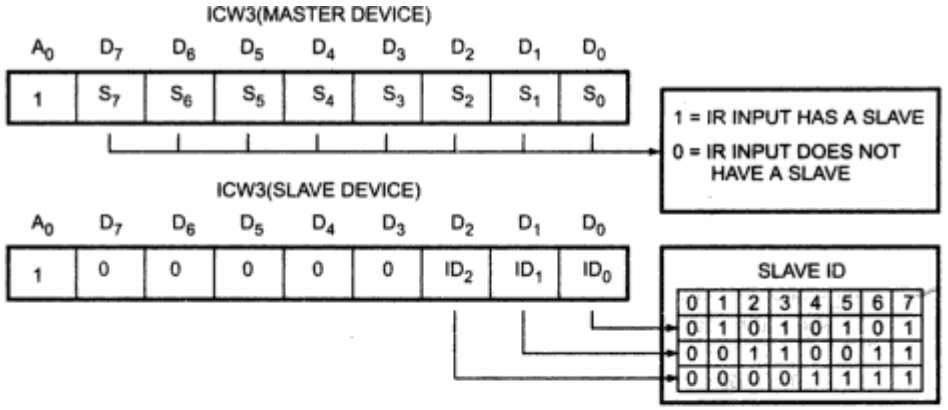


Fig. 8.10 Initialization command word 3 (ICW3)

Initialization Command Word 4 (ICW4)

It is loaded only if the D_0 bit of ICW1 (IC 4) is set. The format of ICW4 is shown in Fig. 8.11.

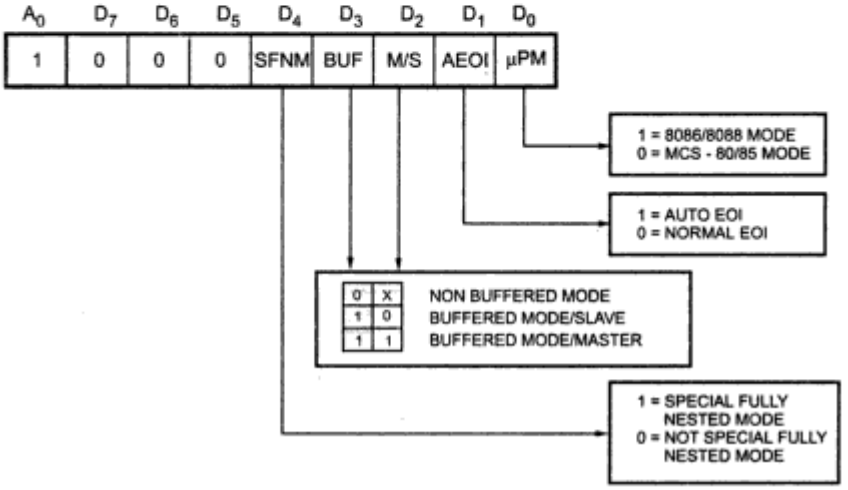


Fig. 8.11 Initialization command word 4 (ICW4)

It specifies.

1) Whether to use special fully nested mode or non special fully nested mode.
2) Whether to use buffered mode or non buffered mode.
3) Whether to use Automatic EOI or Normal EOI
4) CPU used, 8086/8088 or 80810.

SERIAL COMMUNICATION:

Classification

Serial data transmission can be classified on the basis of how transmission occurs.

1. Simplex
2. Half duplex
3. Full duplex

Simplex

In simplex, the hardware exists such that data transfer takes place only in one direction. There is no possibility of data transfer in the other direction. A typical example is transmission from a computer to the printer.

10.1.2 Half Duplex

The half duplex transmission allows the data transfer in both directions, but no simultaneously. A typical example is a walkie-talkie.

10.1.3 Full Duplex

The full duplex transmission allows the data transfer in both direction simultaneously The typical example is transmission through telephone lines.

10.2 Transmission Formats

The data in the serial communication may be sent in two formats :

a) Asynchronous b) Synchronous

10.2.1 Asynchronous

Fig. 10.1 shows the transmission format for asynchronous transmission. Asynchronou formats are character oriented. In this, the bits of a character or data word are sent at constant rate, but characters can come at any rate (asynchronously) as long as they do no overlap. When no characters are being sent, a line stays high at logic 1 called **mark**, logic is called **space**. The beginning of a character is indicated by a start bit which is alway low. This is used to synchronize the transmitter and receiver. After the start bit, the dat bits are sent with least significant bit first, followed by one or more stop bits (active high) The stop bits indicate the end of character. Different systems use 1, 1 1/2 or 2 stop bits The combination of start bit, character and stop bits is known as **frame.** The start and sto bits carry no information, but are required because of the asynchronous nature of data Fig. 10.2 illustrates how the data byte CAH would look when transmitted in th asynchronous serial format.

10.1.2 Half Duplex

The half duplex transmission allows the data transfer in both directions, but no simultaneously. A typical example is a walkie-talkie.

10.1.3 Full Duplex

The full duplex transmission allows the data transfer in both direction simultaneously The typical example is transmission through telephone lines.

10.2 Transmission Formats

The data in the serial communication may be sent in two formats :

a) Asynchronous b) Synchronous

10.2.1 Asynchronous

Fig. 10.1 shows the transmission format for asynchronous transmission. Asynchronou formats are character oriented. In this, the bits of a character or data word are sent at a constant rate, but characters can come at any rate (asynchronously) as long as they do no overlap. When no characters are being sent, a line stays high at logic 1 called **mark**, logic 0 is called **space**. The beginning of a character is indicated by a start bit which is alway low. This is used to synchronize the transmitter and receiver. After the start bit, the data bits are sent with least significant bit first, followed by one or more stop bits (active high) The stop bits indicate the end of character. Different systems use 1, 1 1/2 or 2 stop bits The combination of start bit, character and stop bits is known as **frame.** The start and stop bits carry no information, but are required because of the asynchronous nature of data Fig. 10.2 illustrates how the data byte CAH would look when transmitted in the asynchronous serial format.

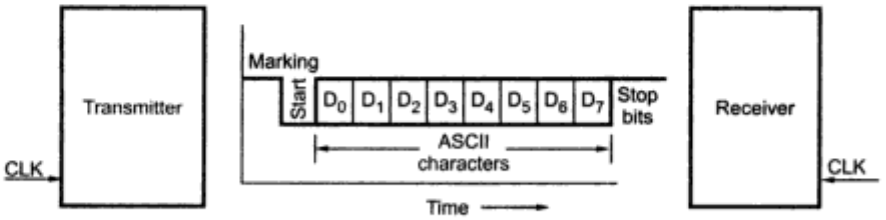


Fig. 10.1 Transmission format for asynchronous transmission

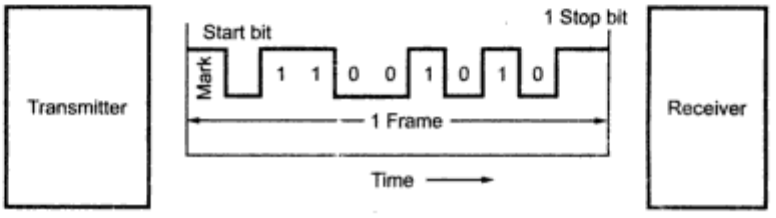


Fig. 10.2 Asynchronous format with data byte CAH

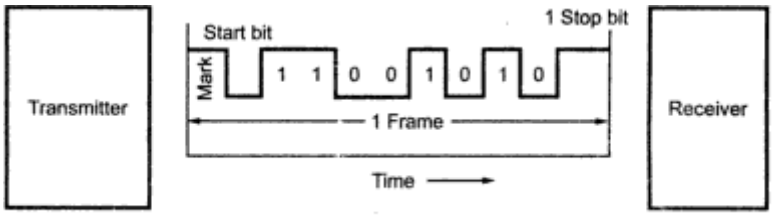


Fig. 10.2 Asynchronous format with data byte CAH

The data rate can be expressed as bits/sec. or characters/sec. The term bits/sec is also called the **baud rate**. The asynchronous format is generally used in low-speed transmission (less than 20 Kbits/sec).

10.2.2 Synchronous

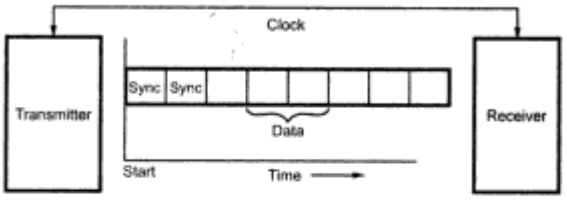


Fig. 10.3 Synchronous transmission format

The start and stop bits in each frame of asynchronous format represents wasted overhead bytes that reduce the overall character rate. These start and stop bits can be eliminated by synchronizing receiver and transmitter. They can be synchronized by having a common clock signal. Such a communication is called **synchronous serial communication**. The Fig. 10.3 shows the transmission format of synchronous serial communication. In this transmission synchronous bits are inserted instead of start and stop bits.

Sr. No.	Asynchronous Serial Communication	Synchronous Serial Communication
1.	Transmitters and receivers are not synchronized by clock.	Transmitter and receivers are synchronized by clock.
2.	Bits of data are transmitted at constant rate.	Data bits are transmitted with synchronisation of clock.
3.	Character may arrive at any rate at receiver.	Character is received at constant rate.
4.	Data transfer is character oriented.	Data transfer takes place in blocks.
5.	Start and stop bits are required to establish communication of each character.	Start and stop bits are not required to establish communication of each character; however, synchronisation bits are required to transfer the data block.
6.	Used in low-speed transmissions at about speed less than 20 kbits/sec.	Used in high-speed transmissions

Table 10.1 Comparison between asynchronous and synchronous serial data transfer

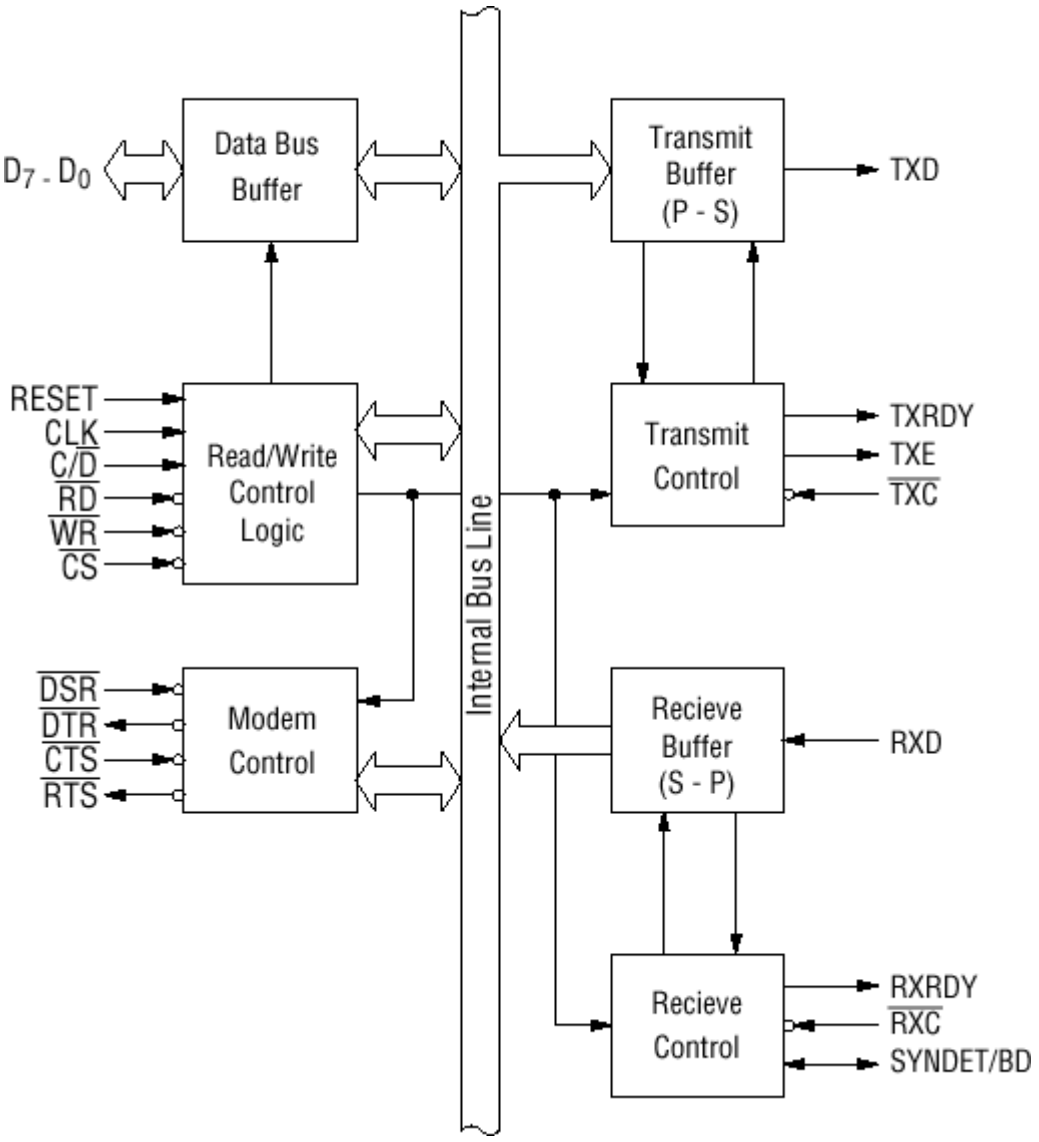
Command Instruction

After the mode instruction, command character should be issued to the USART. It controls the operation of the USART within the basic frame work established by the mode instruction. Fig. 10.7 shows command instruction format.

It does function such as : Enable Transmit/Receive, Error Reset and modem control.

Universal Synchronous Asynchronous Receiver Transmitter (USART)

The 8251 is a USART (Universal Synchronous Asynchronous Receiver Transmitter) for serial data communication. As a peripheral device of a microcomputer system, the 8251 receives parallel data from the CPU and transmits serial data after conversion. This device also receives serial data from the outside and transmits parallel data to the CPU after conversion.



Block diagram of the 8251 USART

The 8251 functional configuration is programed by software. Operation between the 8251 and a CPU is executed by program control. Table 1 shows the operation between a CPU and the device.

$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	
1	×	×	×	Data Bus 3-State
0	×	1	1	Data Bus 3-State
0	1	0	1	Status → CPU
0	1	1	0	Control Word ← CPU
0	0	0	1	Data → CPU
0	0	1	0	Data ← CPU

Table 1 Operation between a CPU and 8251

Control Words

There are two types of control word.

1. Mode instruction (setting of function)

2. Command (setting of operation)

1) Mode Instruction

Mode instruction is used for setting the function of the 8251. Mode instruction will be in "wait for write" at either internal reset or external reset. That is, the writing of a control word after resetting will be recognized as a "mode instruction."

Items set by mode instruction are as follows:

• Synchronous/asynchronous mode

• Stop bit length (asynchronous mode)

• Character length

• Parity bit

• Baud rate factor (asynchronous mode)

• Internal/external synchronization (synchronous mode)

• Number of synchronous characters (Synchronous mode)

The bit configuration of mode instruction is shown in Figures 2 and 3. In the case of synchronous mode, it is necessary to write one-or two byte sync characters. If sync characters were written, a function will be set because the writing of sync characters constitutes part of mode instruction.

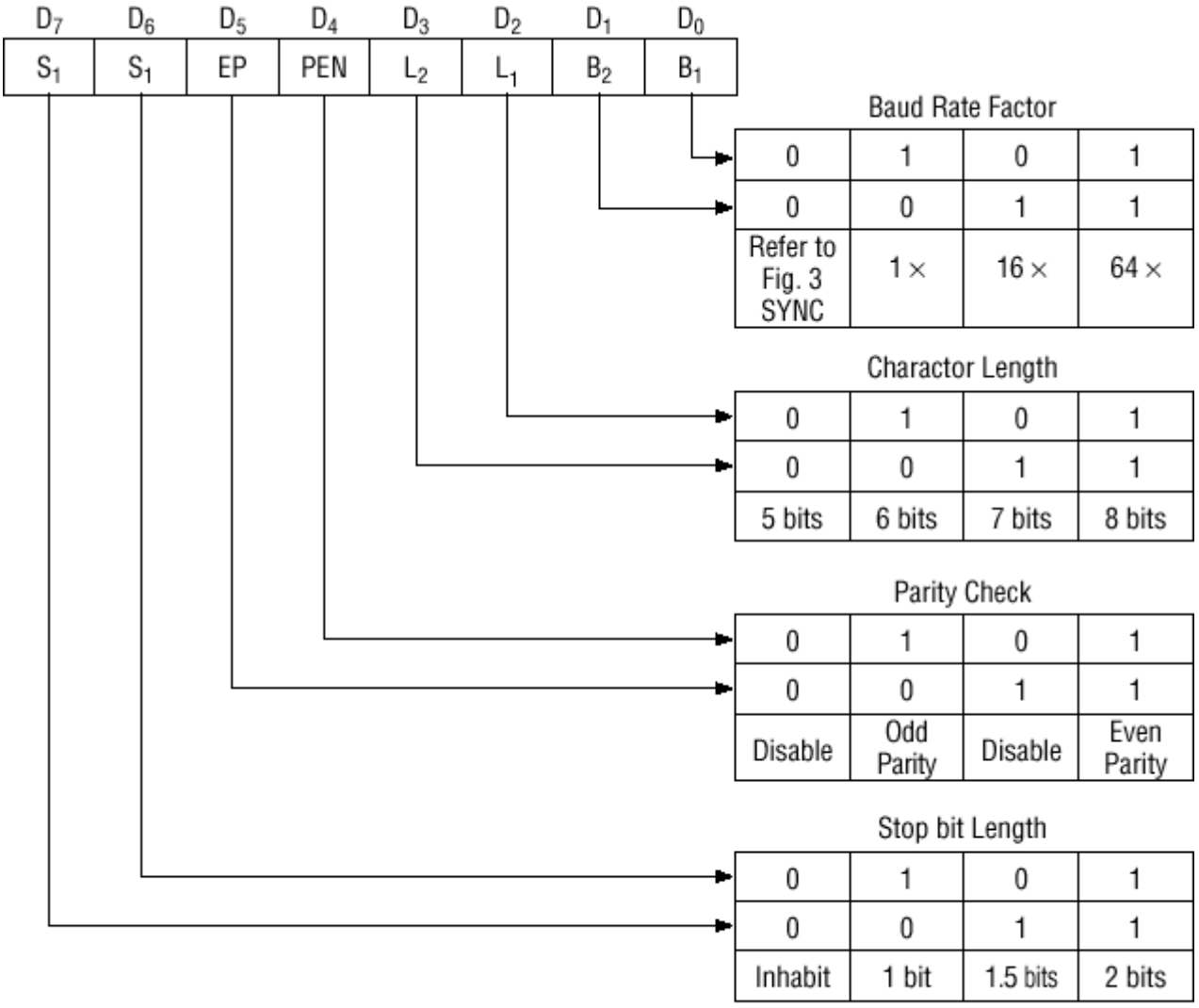


Fig. 2 Bit Configuration of Mode Instruction (Asynchronous)

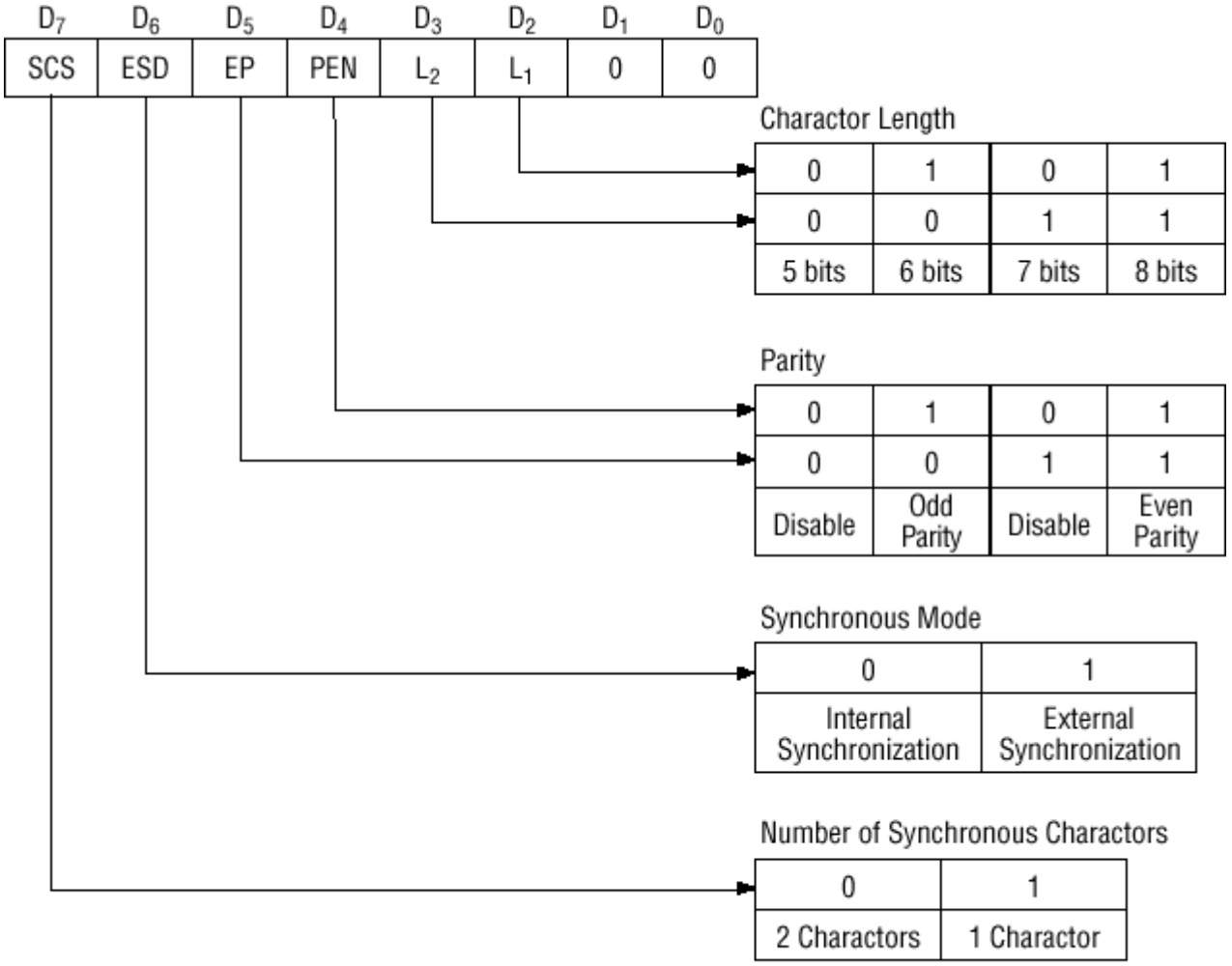


Fig. 3 Bit Configuration of Mode Instruction (Synchronous)

2) Command

Command is used for setting the operation of the 8251. It is possible to write a command whenever necessary after writing a mode instruction and sync characters.

Items to be set by command are as follows:

• Transmit Enable/Disable

• Receive Enable/Disable

• DTR, RTS Output of data.

• Resetting of error flag.

• Sending to break characters

• Internal resetting

• Hunt mode (synchronous mode)

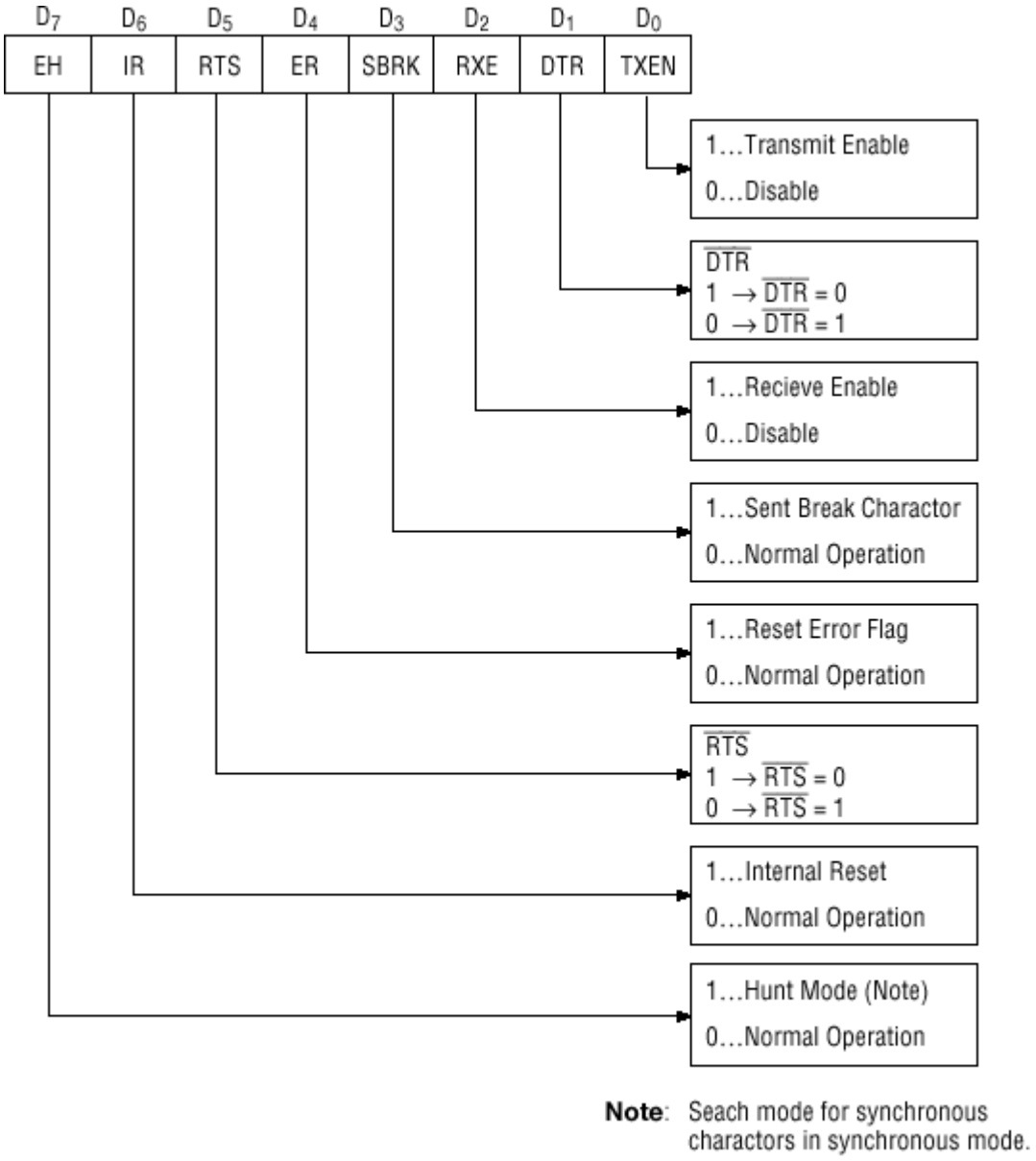


Fig. 4 Bit Configuration of Command

Status Word

It is possible to see the internal status of the 8251 by reading a status word. The bit configuration of status word is shown in Fig. 5.

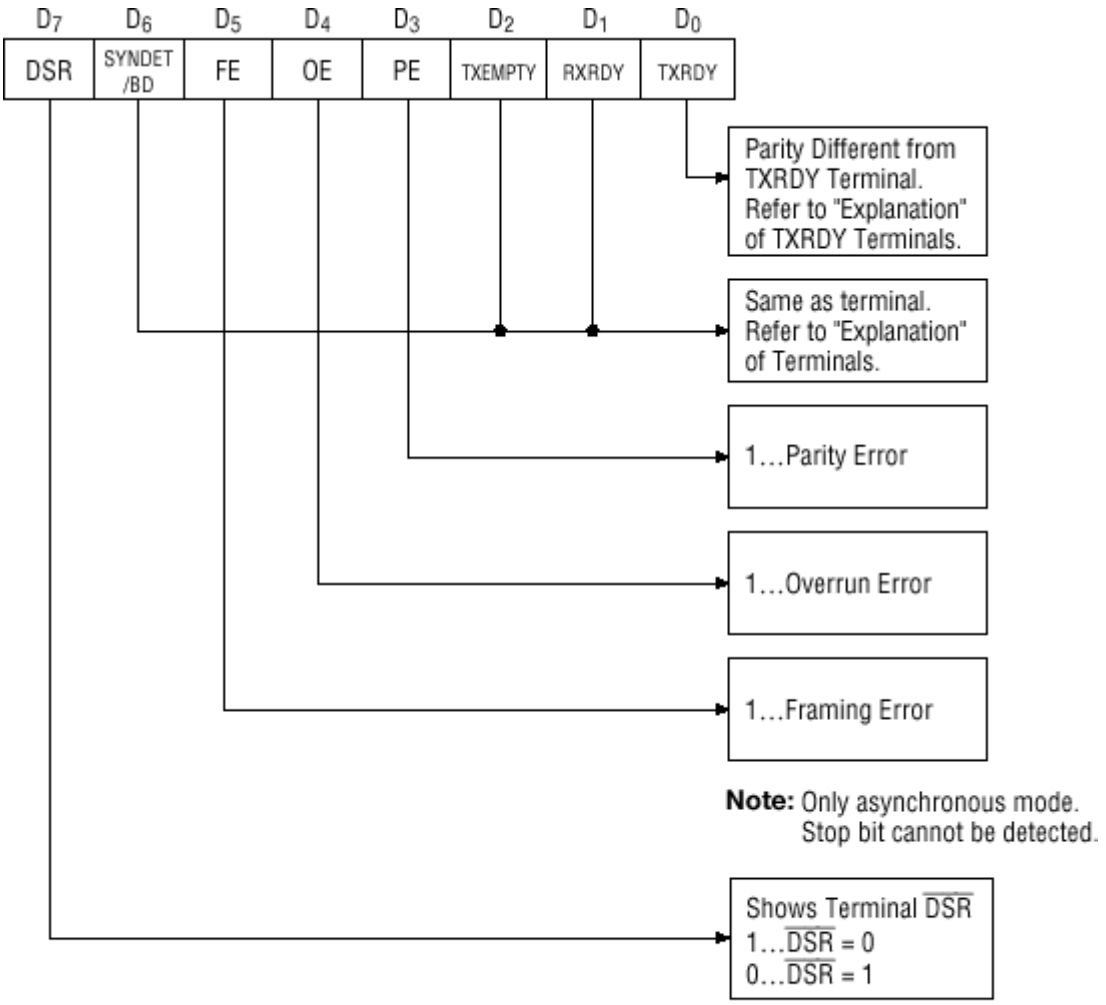


Fig. 5 Bit Configuration of Status Word

Pin Description

D 0 to D 7 (l/O terminal)

This is bidirectional data bus which receive control words and transmits data from the CPU and sends status words and received data to CPU.

RESET (Input terminal)

A "High" on this input forces the 8251 into "reset status." The device waits for the writing of "mode instruction." The min. reset width is six clock inputs during the operating status of CLK.

CLK (Input terminal)

CLK signal is used to generate internal device timing. CLK signal is independent of RXC or TXC. However, the frequency of CLK must be greater than 30 times the RXC and TXC at Synchronous mode and Asynchronous "x1" mode, and must be greater than 5 times at Asynchronous "x16" and "x64" mode.

WR (Input terminal)

This is the "active low" input terminal which receives a signal for writing transmit data and control words from the CPU into the 8251.

RD (Input terminal)

This is the "active low" input terminal which receives a signal for reading receive data and status words from the 8251.

C/D (Input terminal)

This is an input terminal which receives a signal for selecting data or command words and status words when the 8251 is accessed by the CPU. If C/D = low, data will be accessed. If C/D = high, command word or status word will be accessed.

CS (Input terminal)

This is the "active low" input terminal which selects the 8251 at low level when the CPU accesses. Note: The device won‘t be in "standby status"; only setting CS = High.

TXD (output terminal)

This is an output terminal for transmitting data from which serial-converted data is sent out. The device is in "mark status" (high level) after resetting or during a status when transmit is disabled. It is also possible to set the device in "break status" (low level) by a command.

TXRDY (output terminal)

This is an output terminal which indicates that the 8251is ready to accept a transmitted data character. But the terminal is always at low level if CTS = high or the device was set in "TX disable status" by a command. Note: TXRDY status word indicates that transmit data character is receivable, regardless of CTS or command. If the CPU writes a data character, TXRDY will be reset by the leading edge or WR signal.

TXEMPTY (Output terminal)

This is an output terminal which indicates that the 8251 has transmitted all the characters and had no data character. In "synchronous mode," the terminal is at high level, if transmit data characters are no longer remaining and sync characters are automatically transmitted. If the CPU writes a data character, TXEMPTY will be reset by the leading edge of WR signal. Note : As the transmitter is disabled by setting CTS "High" or command, data written before disable will be sent out. Then TXD and TXEMPTY will be "High". Even if a data is written after disable, that data is not sent out and TXE will be "High".After the transmitter is enabled, it sent out. (Refer to Timing Chart of Transmitter Control and Flag Timing)

TXC (Input terminal)

This is a clock input signal which determines the transfer speed of transmitted data. In "synchronous mode," the baud rate will be the same as the frequency of TXC. In "asynchronous mode", it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16 or 1/64 the TXC. The falling edge of TXC sifts the serial data out of the 8251.

RXD (input terminal)

This is a terminal which receives serial data.

RXRDY (Output terminal)

This is a terminal which indicates that the 8251 contains a character that is ready to READ. If the CPU reads a data character, RXRDY will be reset by the leading edge of RD signal. Unless the CPU reads a data character before the next one is received completely, the preceding data will be lost. In such a case, an overrun error flag status word will be set.

RXC (Input terminal)

This is a clock input signal which determines the transfer speed of received data. In "synchronous mode," the baud rate is the same as the frequency of RXC. In "asynchronous mode," it is possible to select the baud rate factor by mode instruction. It can be 1, 1/16, 1/64 the RXC.

SYNDET/BD (Input or output terminal)

This is a terminal whose function changes according to mode. In "internal synchronous mode." this terminal is at high level, if sync characters are received and synchronized. If a status word is read, the terminal will be reset. In "external synchronous mode, "this is an input terminal. A "High" on this input forces the 8251 to start receiving data characters.

In "asynchronous mode," this is an output terminal which generates "high level"output upon the detection of a "break" character if receiver data contains a "low-level" space between the stop bits of two continuous characters. The terminal will be reset, if RXD is at high level. After Reset is active, the terminal will be output at low level.

DSR (Input terminal)

This is an input port for MODEM interface. The input status of the terminal can be recognized by the CPU reading status words.

DTR (Output terminal)

This is an output port for MODEM interface. It is possible to set the status of DTR by a command.

CTS (Input terminal)

This is an input terminal for MODEM interface which is used for controlling a transmit circuit. The terminal controls data transmission if the device is set in "TX Enable" status by a command. Data is transmitable if the terminal is at low level.

RTS (Output terminal)

This is an output port for MODEM interface. It is possible to set the status RTS by a command.

Data Communication Types

We know that, 8251A is Universal Synchronous, Asynchronous, Receiver, and Transmitter. Therefore communication can take place with four different ways.

1. Asynchronous transmission
2. Asynchronous reception
3. Synchronous transmission
4. Synchronous reception

These communication modes can be enabled by writing proper mode and command instructions. The mode instruction defines the baud rate (in case of asynchronous mode), character length, number of stop bit(s) and parity type. After writing proper mode instruction it is necessary to write appropriate command instruction depending on the communication type.

Asynchronous Transmission

Transmission can be enabled by setting transmission enable bit (bit 0) in the command instruction. When transmitter is enabled and $\overline{CTS}$ = 0 the transmitter is ready to transfer data on TxD line.

Operation : When transmitter is ready to transfer data on TxD line, CPU sends data character and it is loaded in the transmit buffer register. The 8251A then automatically adds a start bit (low level) followed by the data bits (least significant bit first), and the programmed number of STOP bit(s) to each character. It also adds parity information prior to STOP bit(s), as defined by the mode instruction. The character is then transmitted as a serial data stream on the TxD output at the falling edge of $\overline{TxC}$. The rate of transmission is equal to 1, $1/16$ or $1/64$ that of the $\overline{TxC}$, as defined by the mode instruction. Fig. 10.9 shows the transmitter output in the asynchronous mode.

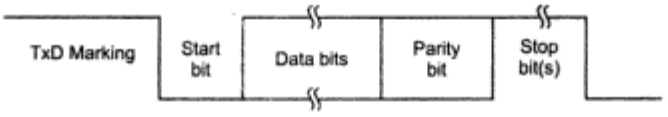


Fig. 10.9 Transmitter output in asynchonous mode

Asynchronous Reception

Reception can be enabled by setting receive enable bit (bit 2) in the command instruction.

Operation :

The RxD line is normally high. 8251A looks for a low level on the RxD line. When it receives the low level, it assumes that it is a START bit and enables an internal counter. At a count equivalent to one-half of a bit time, the RxD line is sampled again. If the line is still low, a valid START bit is detected and the 8251A proceeds to assemble the character. After succesful reception of a START bif the 8251A receives data, parity, and STOP bits and then transfers the data on the receiver input register. The data is then transferred into the receiver buffer register. Fig. 10.10 shows the receiver input in the asynchronous mode.

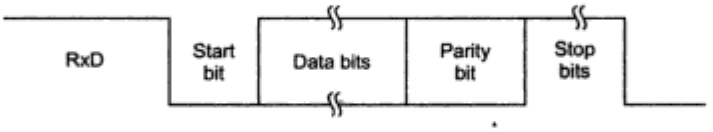


Fig. 10.10 Receiver input in asynchronous mode

Synchronous Transmission

Transmission can be enabled by setting transmission enable bit (bit 0) in the command instruction. When transmitter is enabled and $\overline{CTS}$ = 0, the transmitter is ready to transfer data on TxD line.

Operation : When transmitter is ready to transfer data on TxD line, 8251A transfers characters serially out on the TxD line at the falling edge of the $\overline{TxC}$. The first character usually is the SYNC character.

Once transmission has started, the data stream at the TxD output must continue at the $\overline{TxC}$ rate. If CPU does not provide 8251A with a data character before transmitter buffers become empty, the SYNC characters will be automatically inserted in the TxD data stream, as shown in the Fig. 10.11. In this case, the TxEMPTY pin is raised high to indicate CPU that transmitter buffers are empty. The TxEMPTY pin is internally reset when CPU writes data character in the transmitter buffer.

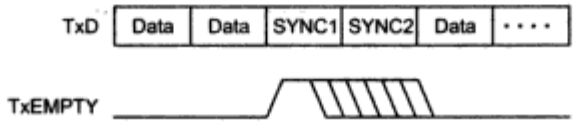


Fig. 10.11 Insertion of SYNC characters

Synchronous Reception : Reception can be enabled by setting receive enable bit (bit 2) in the command instruction.

Operation : In this mode character synchronization can be achieved internally or externally.

Internal SYNC To detect the SYNC character 8251A should be programmed in the 'Enter HUNT' mode by setting bit 7 in the command insturction. Once 8251A enters in the 'Enter HUNT' mode it starts sampling data on the RxD pin on the rising edge of the $\overline{RxC}$. The content of the receiver buffer is compared at every bit boundary with the first SYNC character until a match occurs. If the 8251A has been programmed for two SYNC characters, the subsequent SYNC characters are compared until the match occurs. Once 8251A detects SYNC character(s) it enters from 'HUNT' mode to character synchronization mode, and starts receiving the data characters on the rising edge of the next $\overline{RxC}$. To indicate that the synchronization is achieved 8251A sets the SYNDET pin high. It is reset automatically when CPU reads the status register.

External SYNC

In the external SYNC mode, synchronization is achieved by applying a high level on the SYNDET pin, thus forcing the 8251A out of the HUNT mode.

Serial Communication Protocol (RS232C)

In response to the need for signals and handshake standards between DTE and DCE, the Electronic Industries Association (EIA) introduced EIA standard RS-232 in 1962. It was revised and named as RS-232C, in 1969 by EIA. It is widely accepted for single ended data transmission over short distances with low data rates.

This standard describes the functions of 25 signal and handshake pins for serial data transfer. It also describes the voltage levels, impedance levels, rise and fall times, maximum bit rate, and maximum capacitance for these signal lines. RS-232C specifies 25 signal pins and it specifies that the DTE connector should be male, and the DCE connector should be a female. The most commonly used connector should be a female. The most commonly used connector, DB-25P is shown in the Fig. 10.16.

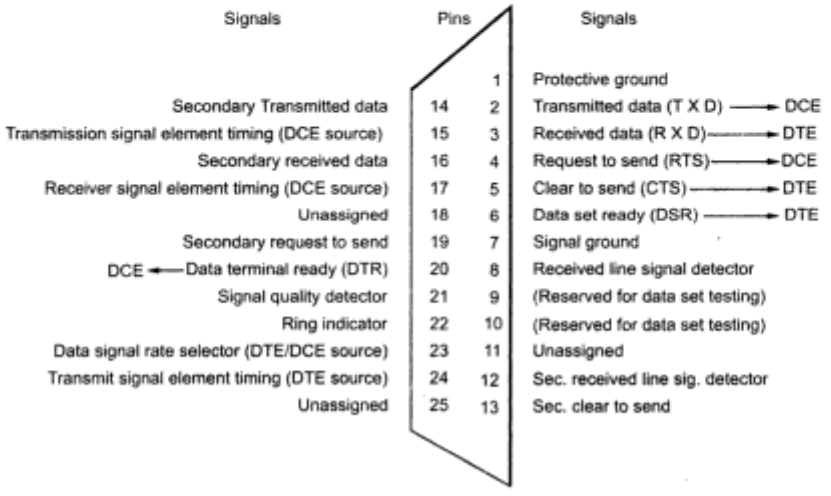


Fig. 10.16 RS 232C 25 pin connector

The Table 10.2 shows pins and signals description for RS-232C for data lines, The voltage level + 3V to + 15V is defined as logic 0; from –3 V to –15 V is defined as logic 1. The control and timing signals are compatible with the TTL level. Because of the incompatibility of the data lines with the TTL logic, voltage translators, called line drivers and line receivers, are required to interface TTL logic with the RS-232 signals. Fig. 10.17 shows the interfacing between TTL and RS-232 signals. The line driver, MC1488, converts logic 1 into approximately 9 V. These levels at the receiving end are again converted by the line receiver, MC1489, into TTL-compatible logic.

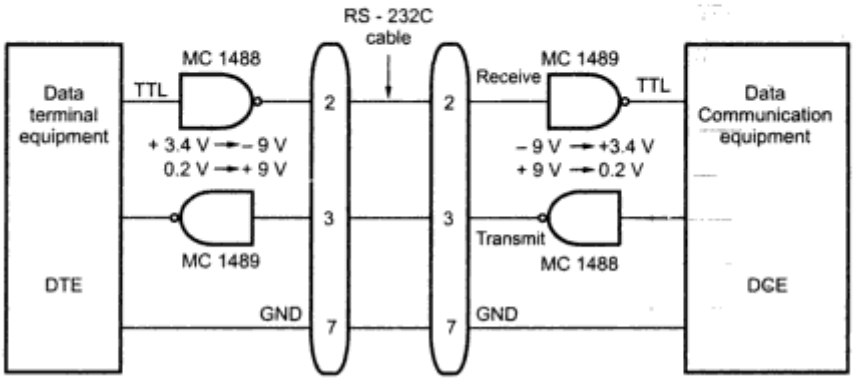


Fig. 10.17 Line drivers and receivers

Unit-III

8255-PPI:

The 8255 is a general purpose programmable I/O device used for parallel data transfer. It has 24 I/O pins which can be grouped in three 8-bit parallel ports : Port A, Port B and Port C. The eight bits of port C can be used as individual bits or be grouped in two 4-bit ports : C_{upper} (C_U) and C_{lower} (C_L).

The 8255, primarily, can be programmed in two basic modes : Bit Set/Reset (BSR) mode and I/O mode. The BSR mode is used to set or reset the bits in port C. The I/O mode is further divided into three modes :

Mode 0 : Simple Input/Output
Mode 1 : Input/Output with handshake
Mode 2 : Bi-directional I/O data transfer

The function of I/O pins (input or output) and modes of operation of I/O ports can be programmed by writing proper control word in the control word register. Each bit in the control word has a specific meaning and the status of these bits decides the function and operating mode of the I/O ports.

Features of 8255A

7. The 8255 can operate in 3 I/O modes : (i) Mode 0, (ii) Mode 1, and (iii) Mode 2.

a) In Mode 0, Port A and Port B can be configured as simple 8-bit input or output I/O ports without handshaking. The two halves of Port C can be programmed separately as 4-bit input or output ports.

b) In Mode 1, two groups each of 12 pins are formed. Group A consists of Port A and the upper half of Port C while Group B consists of Port B and the lower half of Port C. Ports A and B can be programmed as 8-bit Input or Output ports with three lines of Port C in each group used for handshaking.

c) In Mode 2, only Port A can be used as a bidirectional port. The handshaking signals are provided on five lines of Port C (PC_3 - PC_7). Port B can be used in Mode 0 or in Mode 1.

8. All I/O pins of 8255 has 2.5 mA DC driving capacity (i.e. sourcing current of 2.5 mA).

Block Diagram

Fig. 7.2 shows the internal block diagram of 8255A. It consists of data bus buffer. control logic and Group A and Group B controls.

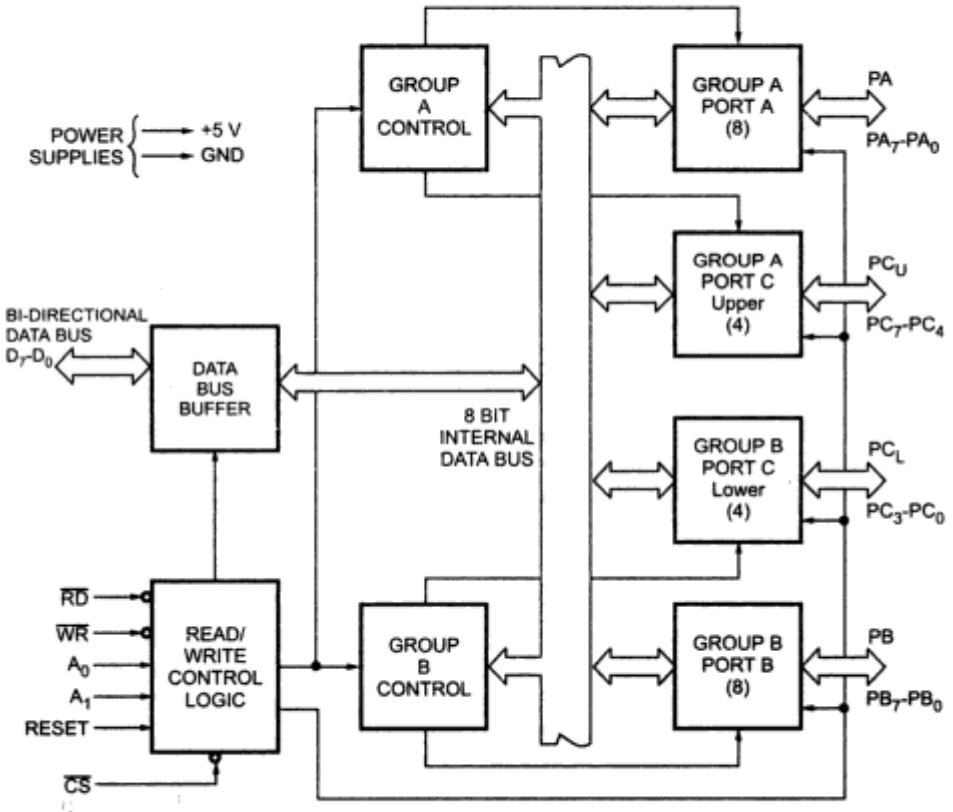


Fig. 7.2 Block diagram of 8255A

Mode 1 : Input/Output with handshake

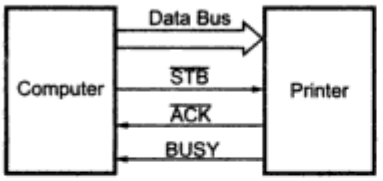


Fig. 7.3 Data transfer between computer and printer using handshaking signals

In this mode, input or output data transfer is controlled by handshaking signals. Handshaking signals are used to transfer data between devices whose data transfer speeds are not same. For example, computer can send data to the printer with large speed but printer can't accept data and print data with this rate. So computer has to send data with the speed with which printer can accept. This type of data transfer is achieved by using handshaking signals alongwith data signals. Fig. 7.3 shows data transfer between computer and printer using handshaking signals.

These handshaking signals are used to tell computer whether printer is ready to accept the data or not. If printer is ready to accept the data then after sending data on data bus, computer uses another handshaking signal ($\overline{STB}$) to tell printer that valid data is available on the data bus.

The 8255 mode 1 which supports handshaking has following features.

1. Two ports (A and B) function as 8-bit I/O ports. They can be configured either as input or output ports.
2. Each port uses three lines from Port C as handshake signals. The remaining two lines of Port C can be used for simple I/O functions.
3. Input and output data are latched.
4. Interrupt logic is supported.

Mode 2 : Bi-directional I/O data transfer

This mode allows bi-directional data transfer (transmission and reception) over a single 8-bit data bus using handshaking signals. This feature is available only in Group A with Port A as the 8-bit bi-directional data bus; and $PC_3 - PC_7$ are used for handshaking purpose. In this mode, both inputs and outputs are latched. Due to use of a single 8-bit data bus for bi-directional data transfer, the data sent out by the CPU through Port A appears on the bus connecting it to the peripheral, only when the peripheral requests it. The remaining lines of Port C i.e. PC_0-PC_2 can be used for simple I/O functions. The Port B can be programmed in mode 0 or in mode 1. When Port B is programmed in mode 1, PC_0-PC_2 lines of Port C are used as handshaking signals.

7.5 Control Word Formats

A high on the RESET pin causes all 24 lines of the three 8-bit ports to be in the input mode. All flip-flops are cleared and the interrupts are reset. This condition is maintained even after the RESET goes low. The ports of the 8255 can then be programmed for any other mode by writing a single control word into the control register, when required.

For Bit Set/Reset Mode

Fig. 7.4 shows bit set/reset control word format.

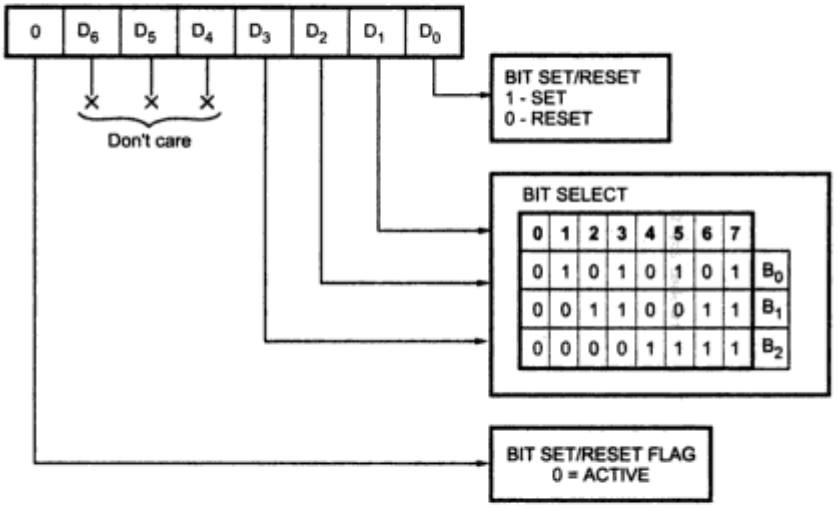


Fig. 7.4 Bit set/reset control word format

The eight possible combinations of the states of bits D_3 - D_1 (B_2 B_1 B_0) in the Bit Set-Reset format (BSR) determine particular bit in PC_0 - PC_7 being set or reset as per the status of bit D_0. A BSR word is to be written for each bit that is to be set or reset. For example, if bit PC_3 is to be set and bit PC_4 is to be reset, the appropriate BSR words that will have to be loaded into the control register will be, 0XXX0111 and 0XXX1000, respectively, where X is don't care.

The BSR word can also be used for enabling or disabling interrupt signals generated by Port C when the 8255 is programmed for Mode 1 or 2 operation. This is done by setting or resetting the associated bits of the interrupts. This is described in detail in next section.

For I/O Mode

The mode definition format for I/O mode is shown in Fig. 7.5. The control words for both, mode definition and Bit Set-Reset are loaded into the same control register, with bit D_7 used for specifying whether the word loaded into the control register is a mode

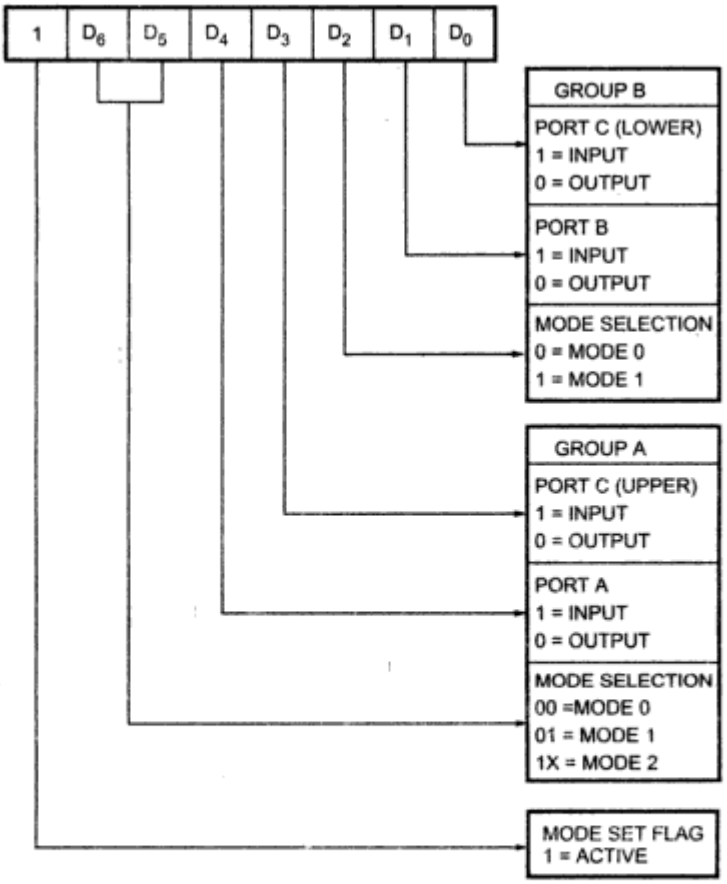


Fig. 7.5 8255 Mode definition format

definition word or Bit Set-Reset word. If D_7 is high, the word is taken as a mode definition word, and if it is low, it is taken as a Bit Set-Reset word. The appropriate bits are set or reset depending on the type of operation desired, and loaded into the control register.

Interfacing 8255 to 8086 in I/O Mapped I/O Mode

The 8086 has four special instructions IN, INS, OUT, and OUTS to transfer data through the input/output ports in I/O mapped I/O system. M/$\overline{IO}$ signal is always low when 8086 is executing these instructions. So M/$\overline{IO}$ signal is used to generate separate addresses for, memory and input/output. Only 256 (2^8) I/O addresses can be generated when direct addressing method is used. By using indirect address method this range can be extended upto 65536 (2^{16}) addresses.

Fig. 7.17 shows the interfacing of 8255 with 8086 in I/O mapped I/O technique. Here, $\overline{RD}$ and $\overline{WR}$ signals are activated when M/$\overline{IO}$ signal is low, indicating I/O bus cycle. Only lower data bus (D_0 – D_7) is used as 8255 is 8-bit device. Reset out signal from clock generator is connected to the Reset signal of the 8255. In case of interrupt driven I/O INTR signal (PC_3 or PC_0) from 8255 is connected to INTR input of 8088.

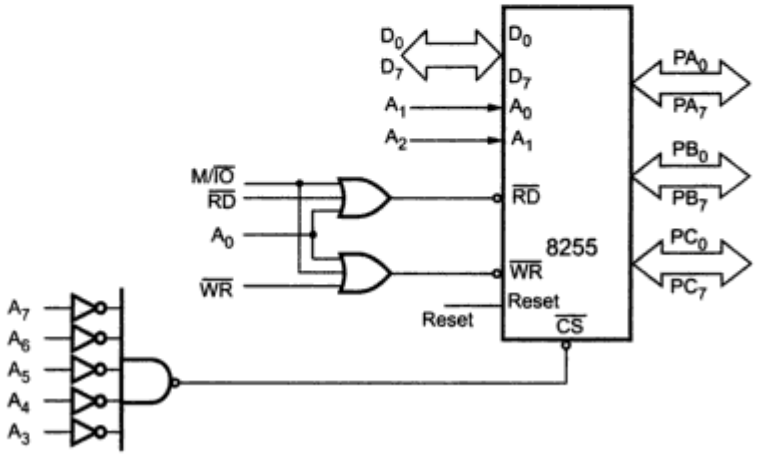


Fig. 7.17 I/O mapped I/O

I/O Map :

Port / control Register	Address lines								Address
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
Port A	0	0	0	0	0	0	0	0	00H
Port B	0	0	0	0	0	0	1	0	02H
Port C	0	0	0	0	0	1	0	0	04H
Control register	0	0	0	0	0	1	1	0	06H

Note : It is assumed that the direct addressing is used.

Interfacing 8255 to 8086 in Memory Mapped I/O

In this type of I/O interfacing, the 8086 uses 20 address lines to identify an I/O device; an I/O device is connected as if it is a memory register. The 8086 uses same control signals and instructions to access I/O as those of memory. Fig. 7.18 shows the interfacing of 8255 with 8086 in memory mapped I/O technique. Here $\overline{RD}$ and $\overline{WR}$ signals are activated when M/$\overline{IO}$ signal is high, indicating memory bus cycle. Address lines A_0 - A_1 are used by 8255 for internal decoding. To get absolute address, all remaining address lines (A_3 - A_{19}) are used to decode the address for 8255. Other signal connections are same as in I/O mapped I/O.

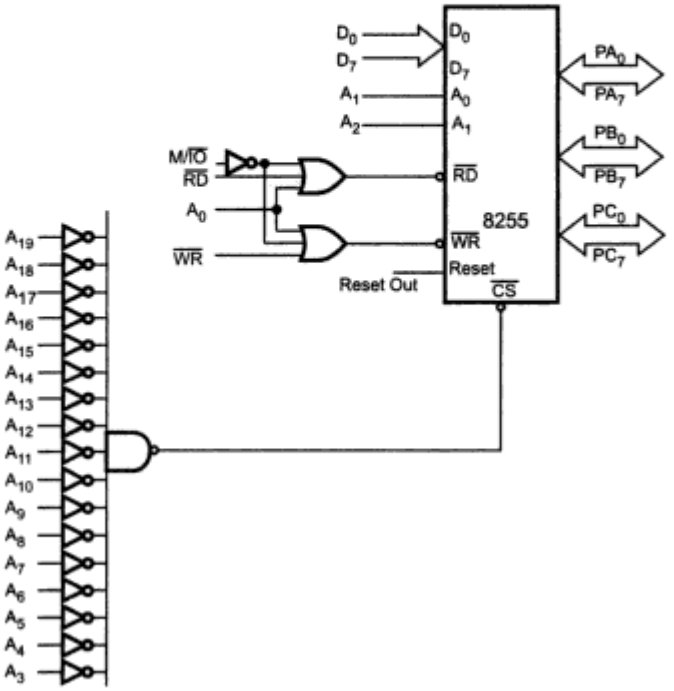


Fig. 7.18 Memory mapped I/O

I/O Map :

Register	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Address
Port A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00000H
Port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	00002H
Port C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	00004H
Control register	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	00006H

ADC 0808/0809

The ADC 0808 and ADC 0809 are monolithic CMOS devices with an 8-channel multiplexer. These devices are also designed to operate from common microprocessor control buses, with tri-state output latches driving the data bus. The main features of these devices are :

Features

- 8-bit successive approximation ADC.
- 8-channel multiplexer with address logic.
- Conversion time 100 μs.
- It eliminates the need for external zero and full-scale adjustments.
- Easy to interface to all microprocessors.
- It operates on single 5 V power supply.
- Output meet TTL voltage level specifications.

Pin Diagram

Fig. 7.37 shows pin diagram of 0808/0809 ADC.

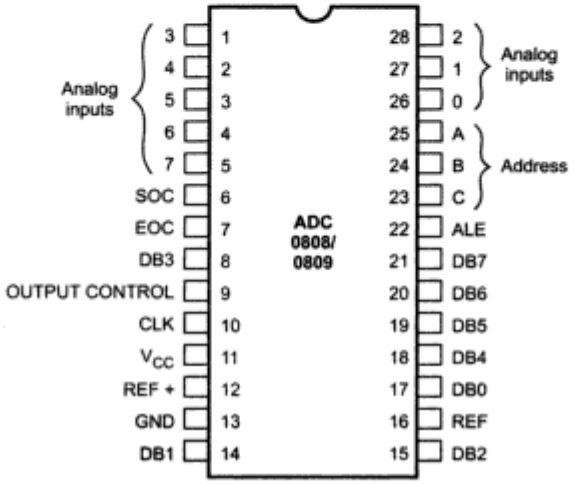


Fig. 7.37 Pin diagram of 0808/0809

Operation

ADC 0808/0809 has eight input channels, so to select desired input channel, it is necessary to send 3-bit address on A, B and C inputs. The address of the desired channel is sent to the multiplexer address inputs through port pins. After at least 50 ns, this address must be latched. This can be achieved by sending ALE signal. After another 2.5 μs, the start of conversion (SOC) signal must be sent high and then low to start the conversion process. To indicate end of conversion ADC 0808/0809 activates EOC signal. The microprocessor system can read converted digital word through data bus by enabling the output enable signal after EOC is activated. This is illustrated in Fig. 7.38.

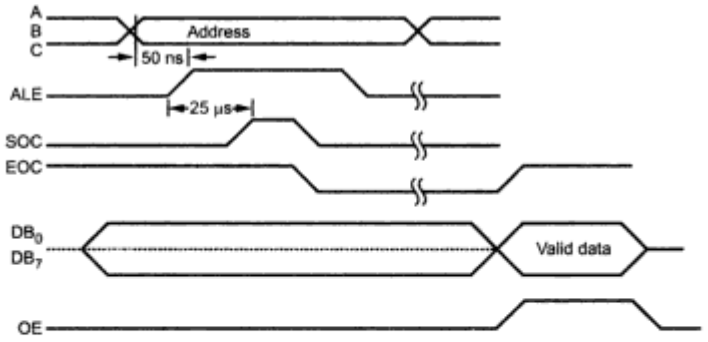


Fig. 7.38 Timing waveforms for ADC 0808

Interfacing

Fig. 7.39 shows typical interfacing circuit for ADC 0808 with microprocessor system.

The zener diode and LM 308 amplifier circuitry is used to produce a V_{CC} an $+V_{REF}$ of 5.12 V for the A/D converter. With this reference voltage the A/D converter will have 256 steps of 20 mV each.

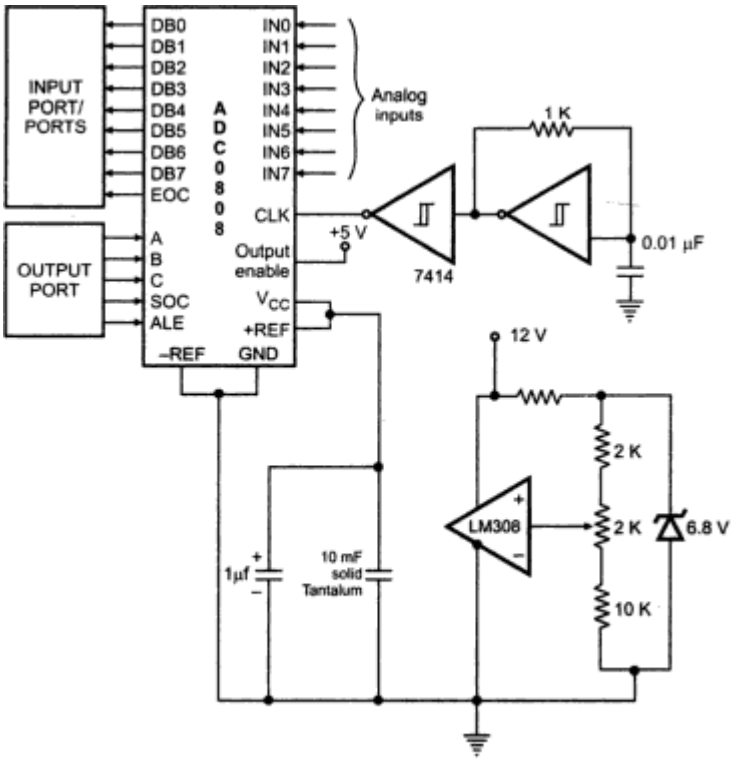


Fig. 7.39 Typical interface for 0808/0809

Stepper Motor Interfacing

A stepper motor is a digital motor. It can be driven by digital signal. Fig. 7.40 shows the typical 2 phase motor interfaced using 8255. Motor shown in the circuit has two phases, with center-tap winding. The center taps of these windings are connected to the 12 V supply. Due to this, motor can be excited by grounding four terminals of the two windings. Motor can be rotated in steps by giving proper excitation sequence to these windings. The lower nibble of port A of the 8255 is used to generate excitation signals in the proper sequence.

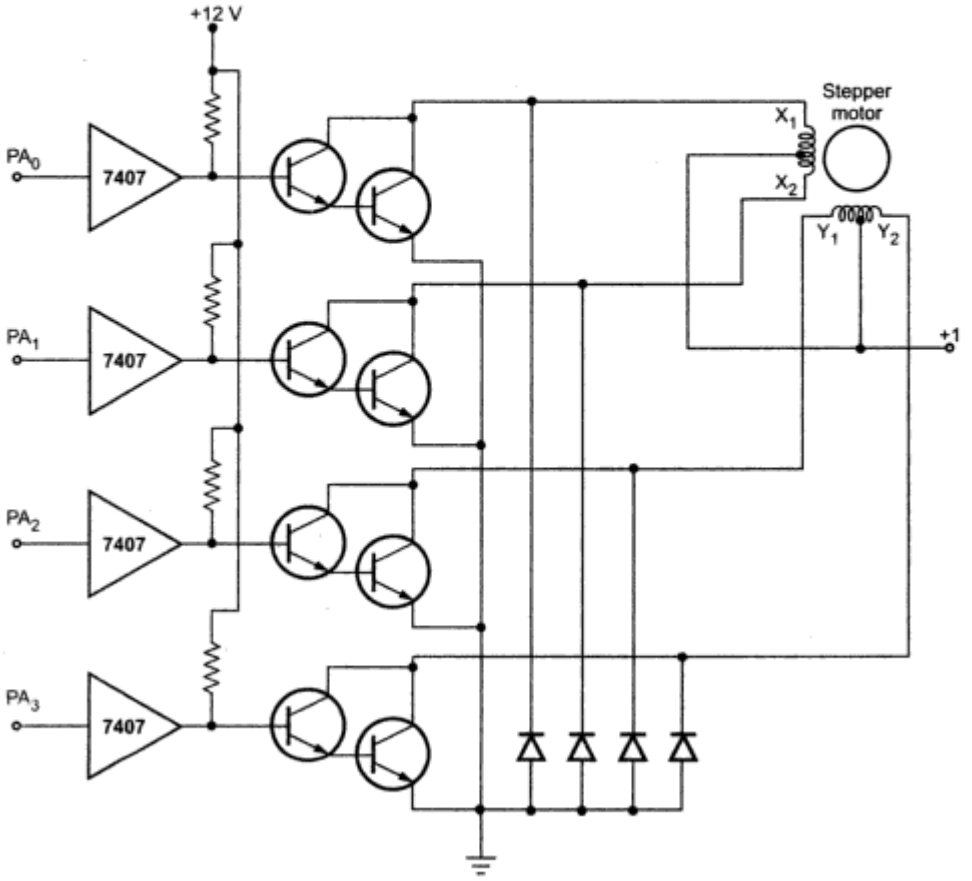


Fig. 7.40 Stepper motor interface

The Table 7.3 shows typical excitation sequence. The given excitation sequence rotates the motor in clockwise direction. To rotate motor in anticlockwise direction we have to excite motor in a reverse sequence. The excitation sequence for stepper motor may change due to change in winding connections. However, it is not desirable to excite both the ends

of the same winding simultaneously. This cancels the flux and motor winding may damage. To avoid this, digital locking system must be designed. Fig. 7.41 shows a simple digital locking system. Only one output is activated (made low) when properly excited; otherwise output is disabled (made high).

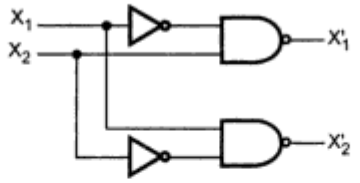


Fig. 7.41 Digital locking system

Step	X_1	X_2	Y_1	Y_2
1	0	1	0	1
2	1	0	0	1
3	1	0	1	0
4	0	1	1	0
1	0	1	0	1

Table 7.3 Full step excitation sequence

The excitation sequence given in Table 7.3 is called **full step sequence** in which excitation ends of the phase are changed in one step. The excitation sequence given in Table 7.4 takes two steps to change the excitation ends of the phase. Such a sequence is called **half step sequence** and in each step the motor is rotated by 0.9°.

Step	X_1	X_2	Y_1	Y_2
1	0	1	0	1
2	0	0	0	1
3	1	0	0	1
4	1	0	0	0
5	1	0	1	0
6	0	0	1	0
7	0	1	1	0
8	0	1	0	0
1	0	1	0	1

Table 7.4 Half step excitation sequence

We know that stepper motor is stepped from one position to the next by changing the currents through the fields in the motor. The winding inductance opposes the change in current and this puts limit on the stepping rate. For higher stepping rates and more torque, it is necessary to use a higher voltage source and current limiting resistors as shown in Fig. 7.42. By adding series resistance, we decrease L/R time constant, which allows the current to change more rapidly in the windings. There is a power loss across series resistor, but designer has to compromise between power and speed.

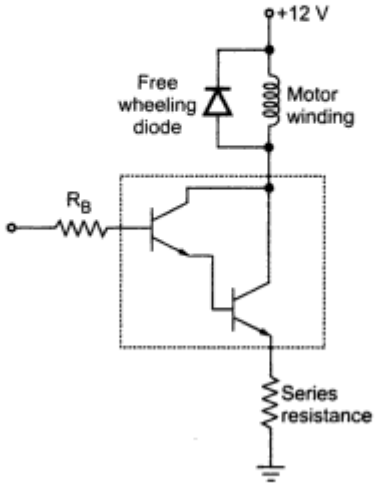


Fig. 7.42 Excitation circuit with series resistance

MEMORY INTERFACING:

•***Control connections***: A ROM usually has only one control input, while a RAM often has one or two control inputs.

•The control input most often found on the ROM is the output enable (OE) or gate (G), this allows data to flow out of the output data pins of the ROM.

•If OE and the selected input are both active, then the output is enable, if OE is inactive, the output is disabled at its high-impedance state.

•The OE connection enables and disables a set of three-state buffer located within the memory device and must be active to read data.

•A RAM memory device has either one or two control inputs. If there is one control input it is often called R/ W .

•This pin selects a read operation or a write operation only if the device is selected by the selection input (CS). If the RAM has two control inputs, they are usually labeled WE or W and OE or G .

•(WE) write enable must be active to perform a memory write operation and OE must be active to perform a memory read operation.

•When these two controls WE and OE are present, they must never be active at the same time.

•The ROM read only memory permanently stores programs and data and data was always present, even when power is disconnected.

•It is also called as nonvolatile memory.

•EPROM (erasable programmable read only memory) is also erasable if exposed to high intensity ultraviolet light for about 20 minutes or less, depending upon the type of EPROM.

•We have PROM (programmable read only memory)

•RMM (read mostly memory) is also called the flash memory.

•The flash memory is also called as an EEPROM (electrically erasable programmable ROM), EAROM (electrically alterable ROM), or a NOVROM (nonvolatile ROM).

•These memory devices are electrically erasable in the system, but require more time to erase than a normal RAM.

•EPROM contains the series of 27XXX contains the following part numbers :

2704(512 * 8), 2708(1K * 8), 2716(2K * 8), 2732(4K * 8), 2764(8K * 8), 27128(16K * 8) etc..

•Each of these parts contains address pins, eight data connections, one or more chip selection inputs (CE) and an output enable pin (OE).

•This device contains **11** address inputs and **8** data outputs.

•If both the pin connection CE and OE are at logic 0, data will appear on the output connection . If both the pins are not at logic 0, the data output connections remains at their high impedance or off state.

•To read data from the EPROM Vpp pin must be placed at a logic 1.

Static RAM Interfacing

The semiconductor RAM is broadly two types – Static RAM and Dynamic RAM.

•The semiconductor memories are organized as two dimensional arrays of memory locations.

•For example 4K * 8 or 4K byte memory contains 4096 locations, where each locations contains 8-bit data and only one of the 4096 locations can be selected at a time. Once a location is selected all the bits in it are accessible using a group of conductors called Data bus.

•For addressing the 4K bytes of memory, 12 address lines are required.

•In general to address a memory location out of N memory locations, we will require at least n bits of address, i.e. n address lines where n = Log2 N.

•Thus if the microprocessor has n address lines, then it is able to address at the most N locations of memory, where 2^n=N. If out of N locations only P memory locations are to be interfaced, then the least significant p address lines out of the available n lines can be directly connected from the microprocessor to the memory chip while the remaining (n-p) higher order address lines may be used for address decoding as inputs to the chip selection logic.

•The memory address depends upon the hardware circuit used for decoding the chip select (CS). The output of the decoding circuit is connected with the CS pin of the memory chip.

•The general procedure of static memory interfacing with 8086 is briefly described as follows:

1.Arrange the available memory chip so as to obtain 16- bit data bus width. The upper 8-bit bank is called as odd address memory bank and the lower 8-bit bank is called as

even address memory bank.

2.Connect available memory address lines of memory chip with those of the microprocessor and also connect the memory RD and WR inputs to the corresponding processor control signals. Connect the 16-bit data bus of the memory bank with that of the microprocessor 8086.

3.The remaining address lines of the microprocessor, BHE and A0 are used for decoding the required chip select signals for the odd and even memory banks. The CS of memory is derived from the o/p of the decoding circuit.

•As a good and efficient interfacing practice, the address map of the system should be continuous as far as possible, i.e. there should not be no windows in the mapand no fold back space should be allowed. A memory location should have a single address corresponding to it, i.e. absolute decoding should be preferred and minimum hardware should be used for decoding large capacity memory is required in a microcomputer system, the memory subsystem is generally designed using dynamic RAM because there are various advantages of dynamic RAM.

• E.g. higher packing density, lower cost and less power consumption. A typical static RAM cell may require six transistors while the dynamic RAM cell requires only a transistors along with a capacitor. Hence it is possible to obtain higher packaging density and hence low cost units are available.

•The basic dynamic RAM cell uses a capacitor to store the charge as a representation of data. This capacitor is manufactured as a diode that is reverse- biased so that the storage capacitance comes into the picture.

•This storage capacitance is utilized for storing the charge representation of data but the reverse-biased diode has leakage current that tends to discharge the capacitor giving rise to the possibility of data loss. To avoid this possible data loss, the data stored in a dynamic RAM cell must be refreshed after a fixed time interval regularly. The process of refreshing the data in RAM is called as

Refresh cycle.

•The refresh activity is similar to reading the data from each and every cell of memory, independent of the requirement of microprocessor. During this refresh period all other operations related to the memory subsystem are suspended. Hence the refresh activity causes loss of time, resulting in reduce system performance.

•However keeping in view the advantages of dynamic RAM, like low power consumption, high packaging density and low cost, most of the advanced computing system are designed using dynamic RAM, at the cost of operating speed.

• A dedicated hardware chip called as dynamic RAM controller is the most important part of the interfacing circuit.

•The ***Refresh cycle*** is different from the memory read cycle in the following aspects.

1.The memory address is not provided by the CPU address bus, rather it is generated by a refresh mechanism counter called as refresh counter.

2.Unlike memory read cycle, more than one memory chip may be enabled at a time so as to reduce the number of total memory refresh cycles.

3.The data enable control of the selected memory chip is deactivated, and data is not allowed to appear on the system data bus during refresh, as more than one memory units are refreshed simultaneously. This is to avoid the data from the different chips to appear on the bus simultaneously.

4.Memory read is either a processor initiated or an external bus master initiated and carried out by the refresh mechanism. Dynamic RAM is available in units of several kilobits to megabits of memory.

This memory is arranged internally in a two dimensional matrix array so that it will have n rows and m columns. The row address n and column address m are important for the refreshing operation.

• For example, a typical 4K bit dynamic RAM chip has an internally arranged bit array of dimension 64 * 64 , i.e. 64 rows and 64 columns. The row address and column address will require 6 bits each. These 6 bits for each row address and column address will be generated by the refresh counter, during the refresh cycles.

•A complete row of 64 cells is refreshed at a time to minimizes the refreshing time. Thus the refresh counter needs to generate only row addresses. The row address are multiplexed, over lower order address lines.

•The refresh signals act to control the multiplexer, i.e. when refresh cycle is in process the refresh counter puts the row address over the address bus for refreshing. Otherwise, the address bus of the processor is connected to the address bus of DRAM, during normal processor initiated activities.

• A timer, called refresh timer, derives a pulse for refreshing action after each refresh

interval.

•Refresh interval can be qualitatively defined as the time for which a dynamic RAM cell can hold data charge level practically constant, i.e. no data loss takes place.

• Suppose the typical dynamic RAM chip has 64 rows, then each row should be refreshed after each refresh interval or in other words, all the 64 rows are to refreshed in a single refresh interval.

•This refresh interval depends upon the manufacturing technology of the dynamic RAM cell. It may range anywhere from 1ms to 3ms.

• Let us consider 2ms as a typical refresh time interval. Hence, the frequency of the refresh pulses will be calculated as follows:

•Refresh Time (per row) tr = (2 * 10 -3) / 64.

•Refresh Frequency fr = 64 / (2 * 10 -3) = 32 * 103 Hz.

•The following block diagram explains the refreshing logic and 8086 interfacing with dynamic RAM.

•Each chip is of 16K * 1-bit dynamic RAM cell array. The system contains two 16K byte dynamic RAM units. All the address and data lines are assumed to be available from an 8086 microprocessor system.

•The OE pin controls output data buffer of the memory chips. The CE pins are active high chip selects of memory chips. The refresh cycle starts, if the refresh output of the refresh timer goes high, OE and CE also tend to go high.

•The high CE enables the memory chip for refreshing, while high OE prevents the data from appearing on the data bus, as discussed in memory refresh cycle. The 16K * 1-bit dynamic RAM has an internal array of 128*128 cells, requiring 7 bits for row address. The lower order seven lines A_0-A_6 are multiplexed with the refresh counter output A10-A16.

•If the RAM has two control inputs, they are usually labeled WE or W and OE or G .

•(WE) write enable must be active to perform a memory write operation and OE must be active to perform a memory read operation.

•When these two controls WE and OE are present, they must never be active at the same time

•The ROM read only memory permanently stores programs and data and data was always present, even when power is disconnected.

•It is also called as nonvolatile memory.

•EPROM (erasable programmable read only memory) is also erasable if exposed to high intensity ultraviolet light for about 20 minutes or less, depending upon the type of EPROM.

•We have PROM (programmable read only memory)

•RMM (read mostly memory) is also called the flash memory.

•The flash memory is also called as an EEPROM (electrically erasable programmable ROM), EAROM (electrically alterable ROM), or a NOVROM (nonvolatile ROM).

•These memory devices are electrically erasable in the system, but require more time to erase than a normal RAM.

•EPROM contains the series of 27XXX contains the following part numbers :

2704(512 * 8), 2708(1K * 8), 2716(2K * 8), 2732(4K * 8), 2764(8K * 8), 27128(16K

* 8) etc..

•Each of these parts contains address pins, eight data connections, one or more chip selection inputs (CE) and an output enable pin (OE).

•This device contains **11** address inputs and **8** data outputs.

•If both the pin connection CE and OE are at logic 0, data will appear on the output connection . If both the pins are not at logic 0, the data output connections remains at their high impedance or off state.

•To read data from the EPROM Vpp pin must be placed at a logic 1.

UNIT-IV

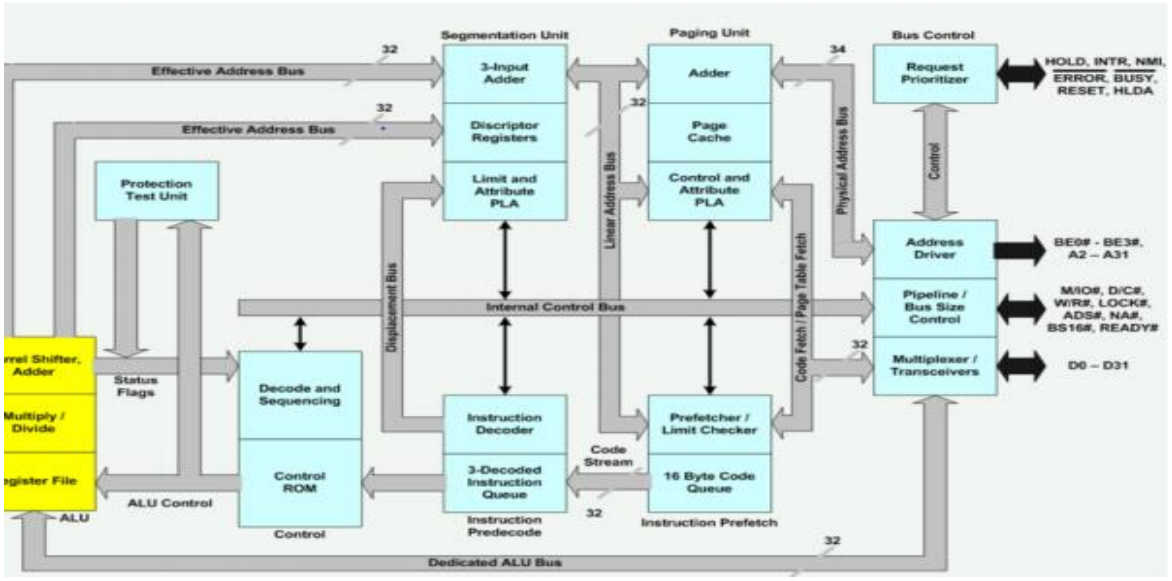
Introduction of 80386DX processor INTEL 80386

The Intel 80386 also known as i386 or just 386, is a 32-bit microprocessor introduced in 1985. The first versions had 275,000 transistors] and were the CPU of many workstations and high-end personal computers of the time.

As the original implementation of the 32-bit extension of the 80286 architecture, the 80386 instruction set, programming model, and binary encodings are still the common denominator for all 32-bit x86 processors, which is termed the i386-architecture, x86, or IA-32, depending on context. The 32-bit 80386 can correctly execute most code intended for the earlier 16-bit processors such as 8088 and 80286 that were ubiquitous in early PCs. (Following the same tradition, modern 64-bit x86 processors are able to run most programs written for older x86 CPUs, all the way back to the original 16-bit 8086 of 1978.) Over the years, successively newer implementations of the same architecture have become several hundreds of times faster than the original 80386 (and thousands of times faster than the 8086).

A 33 MHz 80386 was reportedly measured to operate at about 11.4 MIPS. The 80386 was launched in October 1985, but full-function chips were first delivered in the third quarter of 1986. Mainboards for 80386-based computer systems were cumbersome and expensive at first, but manufacturing was rationalized upon the 80386's mainstream adoption. The first personal computer to make use of the 80386 was designed and manufactured by Compaq and marked the first time a fundamental component in the IBM PC compatible de factostandard was updated by a company other than IBM.

In May 2006, Intel announced that 80386 production would stop at the end of September 2007.[9] Although it had long been obsolete as a personal computer CPU, Intel and others had continued making the chip for embedded systems. Such systems using an 80386 or one of many derivatives are common in aerospace technology and electronic musical instruments, among others. Some mobile phones also used (later fully static CMOS variants of) the 80386 processor, such as



Architecture of 80386DX processor

The Internal Architecture of 80386 is divided into 3 sections - •

Central processing unit •

Memory management unit •

Bus interface unit

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 offset addresses

80386 (32-Bit) Processor •

It Supports following Features

Multitasking.

Pipelining.

- Segmentation.
- Paging.

• 80386 operates in various modes as follows: 1. Real mode. 2. Protected mode. 3. Virtual mode. The Instruction unit decodes the opcode bytes received from the 16-byte instruction code queue and arranges them in a 3- instruction decoded instruction queue. After decoding them pass it to the control section for deriving the necessary control signals. The barrel shifter increases the speed of all shift and rotate operations. The multiply / divide logic implements the bit-shift-rotate algorithms to complete the operations in minimum time.

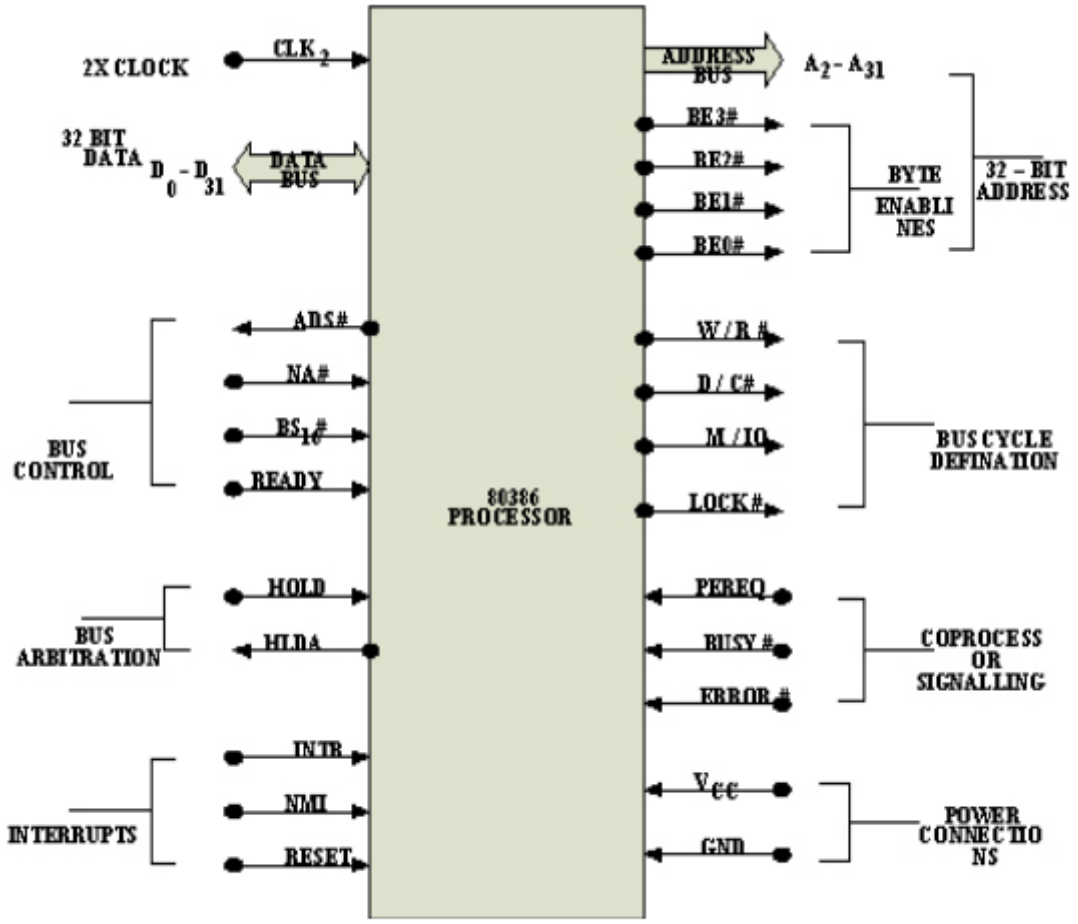
Even 32- bit multiplications can be executed within one microsecond by the multiply / divide logic. The Memory management unit consists of a Segmentation unit and a Paging unit. Segmentation unit

allows the use of two address components, viz. segment and offset for relocability and sharing of code and data. Segmentation unit allows segments of size 4Gbytes at max. The Paging unit organizes the physical memory in terms of pages of 4kbytes size each. Paging unit works under the control of the segmentation unit, i.e. each segment is further divided into pages. The virtual memory is also organizes in terms of segments and pages by the memory management unit.

The Segmentation unit provides a 4 level protection mechanism for protecting and isolating the system code and data from those of the application program. Paging unit converts linear addresses into physical addresses. The control and attribute PLA checks the privileges at the page level. Each of the pages maintains the paging information of the task.

The limit and attribute PLA checks segment limits and attributes at segment level to avoid invalid accesses to code and data in the memory segments. The Bus control unit has a prioritizer to resolve the priority of the various bus requests. This controls the access of the bus. The address driver drives the bus enable and address signal A0 – A31. The pipeline and dynamic bus sizing unit handle the related control signals. The data buffers interface the internal data bus with the system bus.

Pin diagram of 80386 processor



Data bus(D31-D0):-It consists of 32 pins. These lines are used to transfer 8,16,24 or 32 bit data at one time.

ADDRESS BUS(A31-A0):-It generates 32 bit address. The higher 30 bits of address are sent on the A31-A2. The lower 2 bits select one of four bytes of 32 bit data bus.

W/R#: The write / read output distinguishes the write and read cycles from one another.

D/C#: This data / control output pin distinguishes between a data transfer cycle from a machine control cycle like interrupt acknowledge.

M/IO#: This output pin differentiates between the memory and I/O cycles.

LOCK#: The LOCK# output pin enables the CPU to prevent the other bus masters from gaining the control of the system bus.

NA#: The next address input pin, if activated, allows address pipelining, during 80386 bus cycles.

ADS#: The address status output pin indicates that the address bus and bus cycle definition pins(W/R#, D/C#, M/IO#, BE0# to BE3#) are carrying the respective valid signals. The 80383 does not have any ALE signals and so this signals may be used for latching the address to external latches.

READY#: The ready signals indicates to the CPU that the previous bus cycle has been terminated and the bus is ready for the next cycle. The signal is used to insert WAIT states in a bus cycle and is useful for interfacing of slow devices with CPU. VCC: These are system power supply lines.

VSS: These return lines for the power supply. BS16#: The bus size – 16 input pin allows the interfacing of 16 bit devices with the 32 bit wide 80386 data bus. Successive 16 bit bus cycles may be executed to read a 32 bit data from a peripheral. •

HOLD: The bus hold input pin enables the other bus masters to gain control of the system bus if it is asserted.

• **HLDA:** The bus hold acknowledge output indicates that a valid bus hold request has been received and the bus has been relinquished by the CPU.

• **BUSY#:** The busy input signal indicates to the CPU that the coprocessor is busy with the allocated task.

ERROR#: The error input pin indicates to the CPU that the coprocessor has encountered an error while executing its instruction.

• **PEREQ:** The processor extension request output signal indicates to the CPU to fetch a data word for the coprocessor.

• **INTR:** This interrupt pin is a maskable interrupt, that can be masked using the IF of the flag register.

• **NMI:** A valid request signal at the non-maskable interrupt request input pin internally generates a non- maskable interrupt of type2.

RESET: A high at this input pin suspends the current operation and restart the execution from the starting location. • N / C : No connection pins are expected to be left open while connecting the 80386 in the circuit.

Register sets of80836:

The 80386 contains a total of sixteen registers that are of interest to the applications programmer.

1. **General registers**. These eight 32-bit general-purpose registers are used primarily to contain operands for arithmetic and logical operations.
2. **Segment registers.** These special-purpose registers permit systems software designers to choose either a flat or segmented model of memory organization. These six registers determine, at any given time, which segments of memory are currently addressable.

3. **Status and instruction registers**. These special-purpose registers are used to record and alter certain aspects of the 80386 processor state.

General Registers

The general registers of the 80386 are the 32-bit registers EAX, EBX, ECX, EDX, EBP, ESP, ESI, and EDI. These registers are used interchangeably to contain the operands of logical and arithmetic operations. They may also be used interchangeably for operands of address computations (except that ESP cannot be used as an index operand).,the low-order word of each of these eight registers has a separate name and can be treated as a unit.

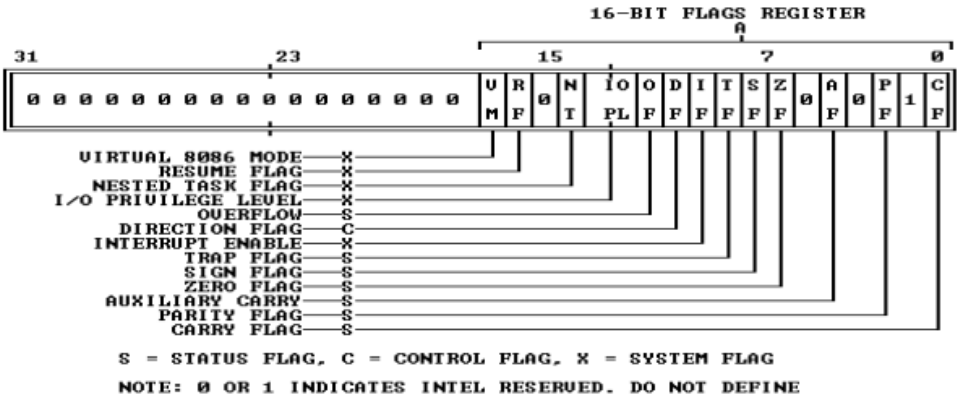
This feature is useful for handling 16-bit data items and for compatibility with the 8086 and 80286 processors. The word registers are named AX, BX, CX, DX, BP, SP, SI, and DI. fig. also illustrates that each byte of the 16-bit registers AX, BX, CX, and DX has a separate name and can be treated as a unit. This feature is useful for handling characters and other 8-bit data items. The byte registers are named AH, BH, CH, and DH (high bytes); and AL, BL, CL, and DL (low bytes).

Segment Registers

The segment registers of the 80386 give systems software designers the flexibility to choose among various models of memory organization. Implementation of memory models is the subject of Part II -- Systems Programming. Designers may choose a model in which applications programs do not need to modify segment registers, in which case applications programmers may skip this section.

Complete programs generally consist of many different modules, each consisting of instructions and data. However, at any given time during program execution, only a small subset of a program's modules are actually in use. The 80386 architecture takes advantage of this by providing mechanisms to support direct access to the instructions and data of the current module's environment, with access to additional segments on demand.

Flags Register



Status Flags

Memory Organization and Segmentation

The physical memory of an 80386 system is organized as a sequence of 8-bit bytes. Each byte is assigned a unique address that ranges from zero to a maximum of 2^(32) -1 **(4 gigabytes).**

80386 programs, however, are independent of the physical address space. This means that programs can be written without knowledge of how much physical memory is available and without knowledge of exactly where in physical memory the instructions and data are located.

The model of memory organization seen by applications programmers is determined by systemssoftware designers. The architecture of the 80386 gives designers the freedom to choose a model for each task. The model of memory organization can range between the following extremes:

A "flat" address space consisting of a single array of up to 4 gigabytes.

- A segmented address space consisting of a collection of up to 16,383 linear address• spaces of up to 4 gigabytes each.

Data Types

Bytes, words, and double words are the fundamental data types. A byte is eight contiguous bits starting at any logical address. The bits are numbered 0 through 7; bit zero is the least significant bit.

Integer:

A signed binary numeric value contained in a 32-bit doubleword, 16-bit word, or 8-bit byte. All operations assume a 2's complement representation. The sign bit is located in bit 7 in a byte, bit 15 in a word, and bit 31 in a doubleword. The sign bit has the value zero for positive integers and one for negative. Since the high-order bit is used for a sign, the range of an 8-bit integer is -128 through +127; 16-bit integers may range from -32,768 through +32,767; 32-bit integers may range from -2^(31) through +2^(31) -1. The value zero has a positive sign

Ordinal:

An unsigned binary numeric value contained in a 32-bit doubleword, 16-bit word, or 8- bit byte. All bits are considered in determining magnitude of the number. The value range of an 8-bit ordinal number is 0-255; 16 bits can represent values from 0 through 65,535; 32 bits can represent values from 0 through 2^(32) -1

Bit field:

A contiguous sequence of bits. A bit field may begin at any bit position of any byte and may contain up to 32 bits. Bit string:

BCD:

A byte (unpacked) representation of a decimal digit in the range 0 through 9. Unpacked decimal numbers are stored as unsigned byte quantities. One digit is stored in each byte. The magnitude of the number is determined from the low-order half-byte; hexadecimal values 0-9 are valid and are interpreted as decimal numbers. The high-order half-byte must be zero for multiplication and division; it may contain any value for addition and subtraction.

Packed BCD:

A byte (packed) representation of two decimal digits, each in the range 0 through 9. One digit is stored in each half-byte. The digit in the high-order half-byte is the most significant. Values 0-9 are valid in each half-byte. The range of a packed decimal byte is 0-99. Figure 2-4 graphically summarizes the data types supported by the 80386.

UNIT-V

8051 MICROCONTROLLER:

The Intel 8051 microcontroller is one of the most popular general purpose microcontrollers in use today. The success of the Intel 8051 spawned a number of clones which are collectively referred to as the MCS-51 family of microcontrollers, which includes chips from vendors such as Atmel, Philips, Infineon, and Texas Instruments

The Intel 8051 is an 8-bit microcontroller which means that most available operations are limited to 8 bits. There are 3 basic "sizes" of the 8051: Short, Standard, and Extended. The Short and Standard chips are often available in DIP (dual in-line package) form, but the Extended 8051 models often have a different form factor, and are not "drop-in compatible". All these things are called 8051 because they can all be programmed using 8051 assembly language, and they all share certain features (although the different models all have their own special features).

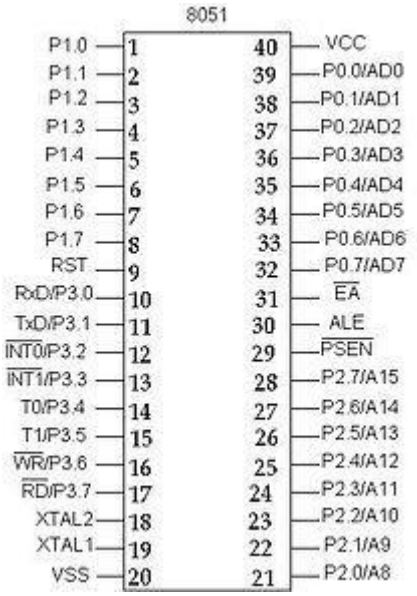
Some of the features that have made the 8051 popular are:

- 64 KB on chip program memory.
- 128 bytes on chip data memory(RAM).
- 4 reg banks.
- 128 user defined software flags.
- 8-bit data bus
- 16-bit address bus
- 32 general purpose registers each of 8 bits
- 16 bit timers (usually 2, but may have more, or less).
- 3 internal and 2 external interrupts.
- Bit as well as byte addressable RAM area of 16 bytes.
- Four 8-bit ports, (short models have two 8-bit ports).
- 16-bit program counter and data pointer.
- 1 Microsecond instruction cycle with 12 MHz Crystal.

8051 models may also have a number of special, model-specific features, such as UARTs, ADC, OpAmps, etc...

Typical applications

8051 chips are used in a wide variety of control systems, telecom applications, robotics as well as in the automotive industry. By some estimations, 8051 family chips make up over 50% of the embedded chip market.



Pin diagram of the 8051 DIP

Basic Pins

PIN 9: PIN 9 is the reset pin which is used reset the microcontroller‘s internal registers and ports upon starting up. (Pin should be held high for 2 machine cycles.)

PINS 18 & 19: The 8051 has a built-in oscillator amplifier hence we need to only connect a crystal at these pins to provide clock pulses to the circuit.

PIN 40 and 20: Pins 40 and 20 are VCC and ground respectively. The 8051 chip needs +5V 500mA to function properly, although there are lower powered versions like the Atmel 2051 which is a scaled down version of the 8051 which runs on +3V.

PINS 29, 30 & 31: As described in the features of the 8051, this chip contains a built-in flash memory. In order to program this we need to supply a voltage of +12V at pin 31. If external memory is connected then PIN 31, also called EA/VPP, should be connected to ground to indicate the presence of external memory. PIN 30 is called ALE (address latch enable), which is used when multiple memory chips are connected to the controller and only one of them needs to be selected.We will deal with this in depth in the later chapters. PIN 29 is called PSEN. This is "program store enable". In order to use the external memory it is required to provide the low voltage (0) on both PSEN and EA pins.

Ports

There are 4 8-bit ports: P0, P1, P2 and P3.

PORT P1 (Pins 1 to 8): The port P1 is a general purpose input/output port which can be used for a variety of interfacing tasks. The other ports P0, P2 and P3 have dual roles or additional functions associated with them based upon the context of their usage.

PORT P3 (Pins 10 to 17): PORT P3 acts as a normal IO port, but Port P3 has additional functions such as, serial transmit and receive pins, 2 external interrupt pins, 2 external counter inputs, read and write pins for memory access.

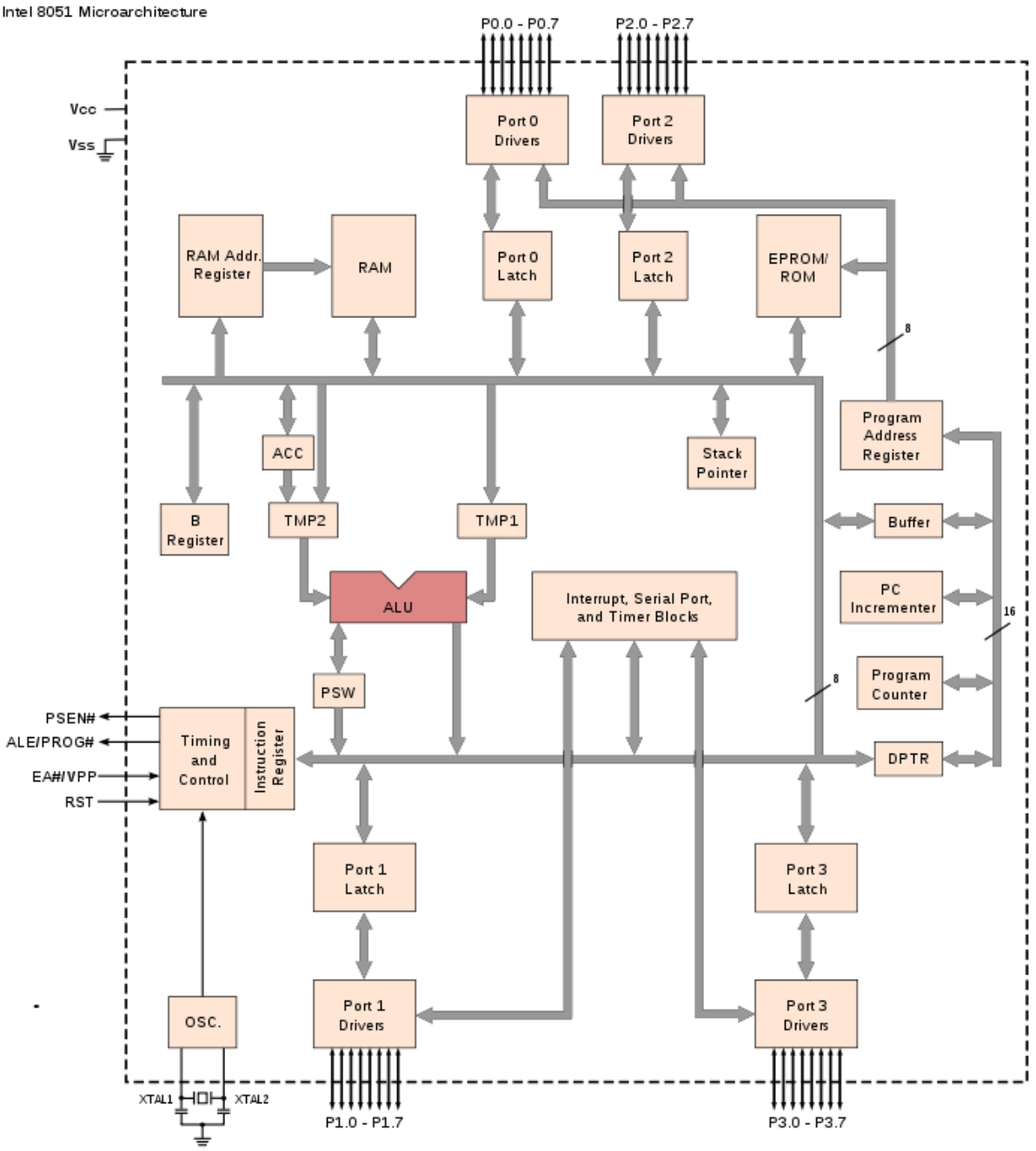
PORT P2 (pins 21 to 28): PORT P2 can also be used as a general purpose 8 bit port when no external memory is present, but if external memory access is required then PORT P2 will act as an address bus in conjunction with PORT P0 to access external memory. PORT P2 acts as A8-A15, as can be seen from fig 1.1

PORT P0 (pins 32 to 39) PORT P0 can be used as a general purpose 8 bit port when no external memory is present, but if external memory access is required then PORT P0 acts as a multiplexed address and data bus that can be used to access external memory in conjunction with PORT P2. P0 acts as AD0-AD7, as can be seen from fig 1.1

Oscillator Circuits

The 8051 requires the existence of an external oscillator circuit. The oscillator circuit usually runs around 12MHz, although the 8051 (depending on which specific model) is capable of running at a maximum of 40MHz. Each machine cycle in the 8051 is 12 clock cycles, giving an effective cycle rate at 1MHz (for a 12MHz clock) to 3.33MHz (for the maximum 40MHz clock).

Internal Architecture



Data and Program Memory

The 8051 Microprocessor can be programmed in PL/M, 8051 Assembly, C and a number of other high-level languages. Many compilers even have support for compiling C++ for an 8051.

Program memory in the 8051 is read-only, while the data memory is considered to be read/write accessible. When stored on EEPROM or Flash, the program memory can be rewritten when the microcontroller is in the special programmer circuit.

Program Start Address

The 8051 starts executing program instructions from address 0000 in the program memory.

Direct Memory

The 8051 has 256 bytes of internal addressable RAM, although only the first 128 bytes are available for general use by the programmer. The first 128 bytes of RAM (from 0x00 to 0x7F) are called the **Direct Memory**, and can be used to store data.

Special Function Register

The **Special Function Register** (SFR) is the upper area of addressable memory, from address 0x80 to 0xFF. A, B, PSW, DPTR are called SFR.This area of memory cannot be used for data or program storage, but is instead a series of memory-mapped ports and registers. All port input and output can therefore be performed by memory **mov** operations on specified addresses in the SFR. Also, different status registers are mapped into the SFR, for use in checking the status of the 8051, and changing some operational parameters of the 8051.

General Purpose Registers

The 8051 has 4 selectable banks of 8 addressable 8-bit registers, R0 to R7. This means that there are essentially 32 available general purpose registers, although only 8 (one bank) can be directly accessed at a time. To access the other banks, we need to change the current bank number in the flag status register.

A and B Registers

The A register is located in the SFR memory location 0xE0. The A register works in a similar fashion to the AX register of x86 processors. The A register is called the **accumulator**, and by default it receives the result of all arithmetic operations. The B register is used in a similar manner, except that it can receive the extended answers from the multiply and divide operations. When not being used for multiplication and Division, the B register is available as an extra general-purpose register.

Comparison between Microprocessor and Microcontroller

We have discussed what is a microprocessor and a microcontroller. Let us see the points of differences between them.

No.	Microprocessor	Microcontroller
1.	Microprocessor contains ALU, control unit (clock and timing circuit), different register and interrupt circuit.	Microcontroller contains microprocessor, memory (ROM and RAM), I/O interfacing circuit and peripheral devices such as A/D converter, serial I/O, timer etc.
2.	It has many instructions to move data between memory and CPU.	It has one or two instructions to move data between memory and CPU.
3.	It has one or two bit handling instructions.	It has many bit handling instructions.
4.	Access times for memory and I/O devices are more.	Less access times for built-in memory and I/O devices.
5.	Microprocessor based system requires more hardware.	Microcontroller based system requires less hardware reducing PCB size and increasing the reliability.
6.	Microprocessor based system is more flexible in design point of view.	Less flexible in design point of view.
7.	It has single memory map for data and code.	It has separate memory map for data and code.
8.	Less number of pins are multifunctioned.	More number pins are multifunctioned.

Features of 8051

The features of the 8051 family are as follows :

1) 4096 bytes on - chip program memory.
2) 128 bytes on - chip data memory.
3) Four register banks.
4) 128 User-defined software flags.
5) 64 Kilobytes each program and external RAM addressability.
6) One microsecond instruction cycle with 12 MHz crystal.
7) 32 bidirectional I/O lines organized as four 8-bit ports (16 lines on 8031).
8) Multiple mode, high-speed programmable serial port.
9) Two multiple mode, 16-bit Timers/Counters.
10) Two-level prioritized interrupt structure.
11) Full depth stack for subroutine return linkage and data storage.
12) Direct Byte and Bit addressability.
13) Binary or Decimal arithmetic.
14) Signed-overflow detection and parity computation.
15) Hardware Multiple and Divide in 4 μsec.
16) Integrated Boolean Processor for control applications.
17) Upwardly compatible with existing 8084 software.

8051 Microcontroller Hardware

The Fig. 11.1 shows the internal block diagram of 8051. It consists of a CPU, two kinds of memory sections (data memory - RAM and program memory - EPROM/ROM), input/output ports, special function registers and control logic needed for a variety of peripheral functions. These elements communicate through an eight bit data bus which runs throughout the chip referred as internal data bus. This bus is buffered to the outside world through an I/O port when memory or I/O expansion is desired.

Central Processing Unit (CPU)

The CPU of 8051 consists of eight-bit Arithmetic and Logic unit with associated registers like A, B, PSW, SP, the sixteen bit program counter and "Data pointer" (DPTR) registers. Alongwith these registers it has a set of special function registers. Along with these registers it has a set of special function registers.

The 8051's ALU can perform arithmetic and logic functions on eight bit variables. The arithmetic unit can perform addition, subtraction, multiplication and division. The logic unit can perform logical operations such as AND, OR, and Exclusive-OR, as well as rotate, clear, and complement. The ALU also looks after the branching decisions. An important and unique feature of the 8051 architecture is that the ALU can also manipulate one bit as well as eight-bit data types. Individual bits may be set, cleared, complemented, moved, tested, and used in logic computation.

Internal RAM

The 8051 has 128-byte internal RAM. It is accessed using RAM address register. The Fig. 11.3 shows the organisation of internal RAM. As shown in the Fig. 11.3, internal RAM of 8051 is organised into three distinct areas :

- Working registers
- Bit Addressable
- General Purpose

1. First thirty-two bytes from address 00H to 1FH of internal RAM constitute 32 working registers. They are organised into four banks of eight registers each. The four register banks are numbered 0 to 3 and are consists of eight registers named R_0 to R_7. Each register can be addressed by name or by its RAM address. Only one register bank is in use at a time. Bits RS_0 and RS_1 in the PSW determine which bank of registers is currently in use. Register banks when not selected can be used as general purpose RAM. On reset, the Bank 0 is selected.
2. The 8051 provides 16 bytes of a bit-addressable area. It occupies RAM byte addresses from 20H to 2FH, forming a total of 128 (16 × 8) addressable bits. An addressable bit may be specified by its bit address of 00H to 7FH, or 8 bits may form any byte address from 20H to 2FH. For example, bit address 4EH refers bit 6 of the byte address 29H.
3. The RAM area above bit addressable area from 30H to 7FH is called general purpose RAM. It is addressable as byte.

See Fig. 11.3 on next page.

11.3.4 Internal ROM

The 8051 has 4 Kbyte of internal ROM with address space from 0000H to 0FFFH. It is programmed by manufacturer when the chip is built. This part cannot be erased or altered after fabrication. This is used to store final version of the program.

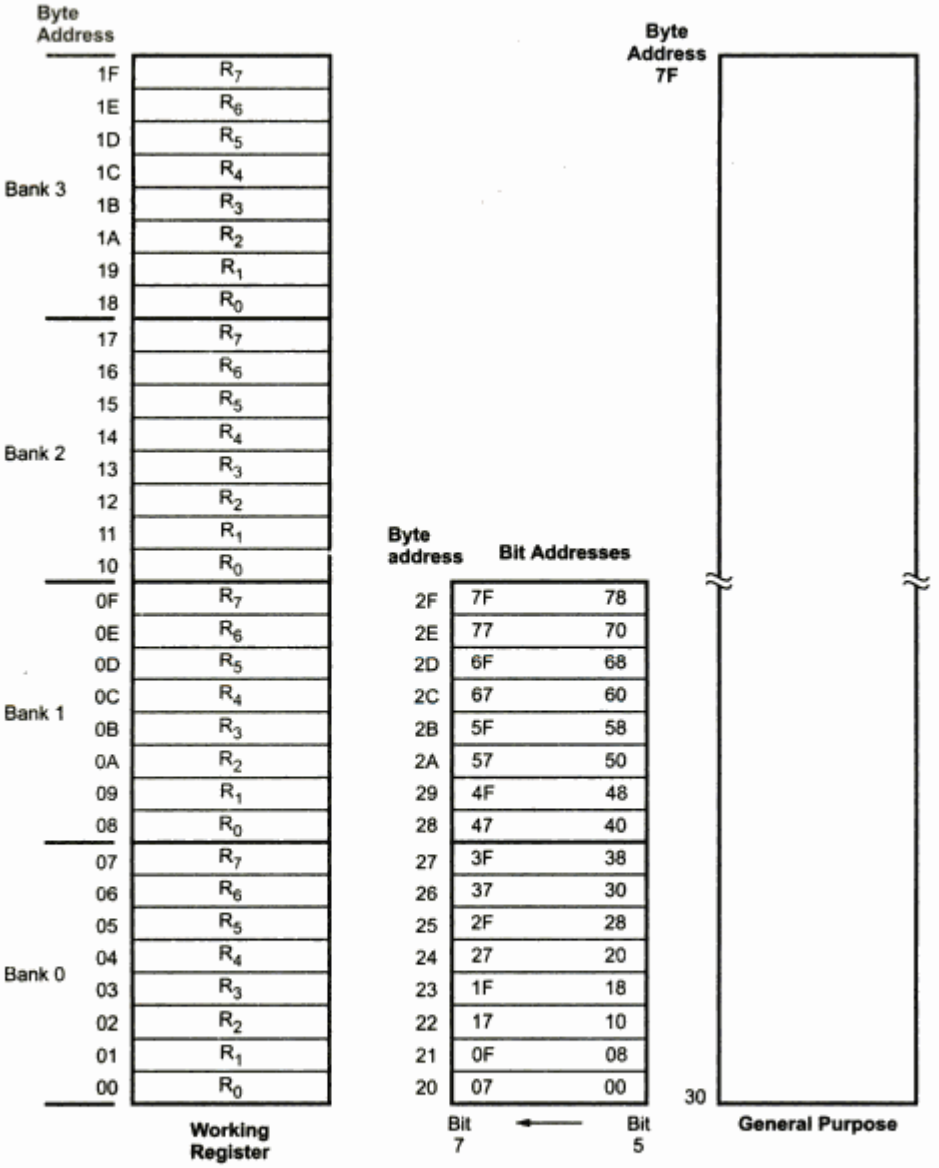


Fig. 11.3 Organisation of Internal RAM of 8051

Input/Output Ports

The 8051 has 32 I/O pins configured as four eight-bit parallel ports (P0, P1, P2, and P3). All four ports are bidirectional, i.e. each pin will be configured as input or output (or both) under software control. Each port consists of a latch, an output driver, and an input buffer.

The output drives of Ports 0 and 2 and the input buffers of Port 0, are used to access external memory. As mentioned earlier, Port 0 outputs the low order byte of the external memory address, time multiplexed with the data being written or read, and Port 2 outputs the high order byte of the external memory address when the address is 16 bits wide. Otherwise Port 2 gives the contents of special function register P2.

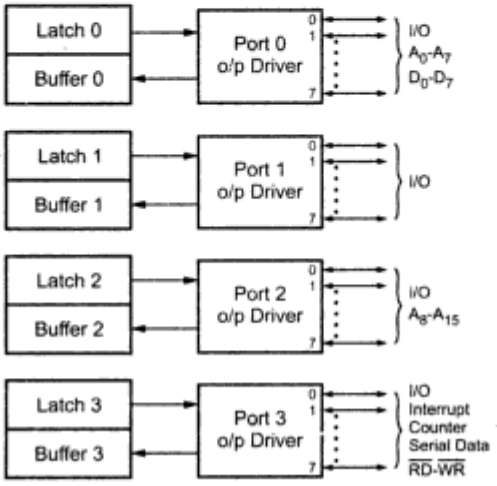


Fig. 11.4 I/O Ports

11.3.6 Register Set of 8051

11.3.6.1 Register A (Accumulator)

It is an 8-bit register. It holds a source operand and receives the result of the arithmetic instructions (addition, subtraction, multiplication, and division). The accumulator can be the source or destination for logical operations and a number of special data movement instructions, including look-up tables and external RAM expansion. Several functions apply exclusively to the accumulator : rotate, parity computation , testing for zero , and so on.

11.3.6.2 Register B

In addition to accumulator, an 8-bit B-register is available as a general purpose register when it is not being used for the hardware multiply/divide operation.

11.3.6.3 Program Status Word (Flag Register)

Many instructions implicitly or explicitly affect (or are affected by) several status flags, which are grouped together to form the Program Status Word. Fig. 11.5 shows the bit pattern of the program status word. It is an 8-bit word, containing the information as follows.

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
CY	AC	F0	RS1	RS0	OV	-	P

CY Bit 7 - Carry flag

AC Bit 6 - Auxiliary carry flag for BCD operations

F0 Bit 5 - User defined flag (Flag zero)

RS1, RS0 Bit 4-3 - Select the working register banks as follows :

RS1	RS0	Bank Selection	
0	0	00H - 07H	Bank 0
0	1	08H - 0FH	Bank 1
1	0	10H - 17H	Bank 2
1	1	18H - 1FH	Bank 3

OV Bit 2 - Overflow flag

– Bit 1 - Reserved

P Bit 0 - Parity flag (1 = Even parity)

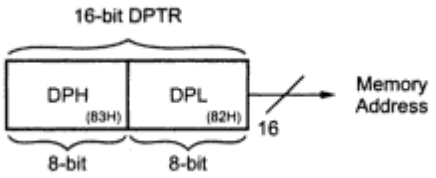
Fig. 11.5 Program status word

11.3.6.4 Stack and Stack Pointer

The stack refers to an area of internal RAM that is used in conjunction with certain opcodes data to store and retrieve data quickly. The stack pointer register is used by the 8051 to hold an internal RAM address that is called **top of stack.** The stack pointer register is 8-bit wide. It is increased **before** data is stored during PUSH and CALL instructions and decremented **after** data is restored during POP and RET instructions. Thus stack array can reside anywhere in on-chip RAM. The stack pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H. The operation of stack and stack pointer is illustrated in Fig. 11.6.

11.3.6.5 Data Pointer (DPTR)

The data pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its function is to hold a 16 bit address. It may be manipulated as a 16 bit data register or as two independent 8 bit registers. It serves as a base register in indirect jumps, lookup table instructions and external data transfer. The DPTR does not have a single internal address; DPH (83H) and DPL (82H) have separate internal addresses.



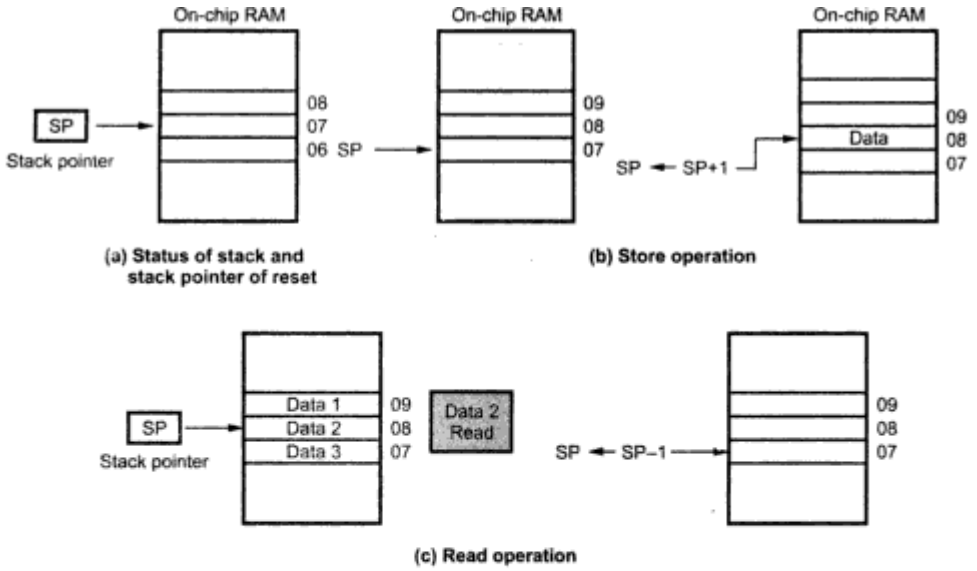


Fig. 11.6

11.3.6.6 Program Counter

The 8051 has a 16-bit program counter. It is used to hold the address of memory location from which the next instruction is to be fetched. Due to this the width of the program counter decides the maximum program length in bytes. For example, 8051 is 16-bit hence it can address upto 2^{16} bytes (64 K) of memory.

The PC is automatically incremented to point the next instruction in the program sequence after execution of the current instruction. It may also be altered by certain instructions. The PC is the only register that does not have an internal address.

11.3.6.7 Special Function Registers

Unlike other microprocessors in the Intel family, 8051 uses memory mapped I/O through a set of special function registers that are implemented in the address space immediately above the 128 bytes of RAM. Fig. 11.7 shows special function bit addresses. All access to the four I/O ports, the CPU registers, interrupt-control registers, the timer/counter, UART, and power control are performed through registers between 80H and FFH.

Direct Byte Address (MSB)	Bit Address (LSB)								Hardware Register Symbol
0FFH									
0F0H	F7	F6	F5	F4	F3	F2	F1	F0	B
0E0H	E7	E6	E5	E4	E3	E2	E1	E0	ACC
0D0H	D7	D6	D5	D4	D3	D2	D1	D0	PSW
0B8H	---	---	---	BC	BB	BA	B9	B8	IP
0B0H	B7	B6	B5	B4	B3	B2	B1	B0	P3
0A8H	AF	---	---	AC	AB	AA	A9	A8	IE
0A0H	A7	A6	A5	A4	A3	A2	A1	A0	P2
98H	9F	9E	9D	9C	9B	9A	99	98	SCON
90H	97	96	95	94	93	92	91	90	P1
88H	8F	8E	8D	8C	8B	8A	89	88	TCON
80H	87	86	85	84	83	82	81	80	P0

Fig. 11.7 SFR bit address

Symbol	Name	Address	Value in Binary
*ACC	Accumulator	0E0H	0 0 0 0 0 0 0 0
*B	B Register	0F0H	0 0 0 0 0 0 0 0
*PSW	Program Status Word	0D0H	0 0 0 0 0 0 0 0
SP	Stack Pointer	81H	0 0 0 0 0 1 1 1
DPTR DPL DPH	Data Pointer 2 Bytes Low Byte High Byte	 82H 83H	 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
*P0	Port 0	80H	1 1 1 1 1 1 1 1
*P1	Port 1	90H	1 1 1 1 1 1 1 1
*P2	Port 2	0A0H	1 1 1 1 1 1 1 1
*P3	Port 3	0B0H	1 1 1 1 1 1 1 1
*IP	Interrupt Priority Control	0B8H	8051 X X X 0 0 0 0 0 8052 X X 0 0 0 0 0 0
*IE	Interrupt Enable Control	0A8H	8051 0 X X 0 0 0 0 0 8052 0 X 0 0 0 0 0 0
TMOD	Timer/Counter Mode Control	89H	0 0 0 0 0 0 0 0
*TCON	Timer/Counter Control	88H	0 0 0 0 0 0 0 0
* + T2CON	Timer/Counter 2 Control	0C8H	0 0 0 0 0 0 0 0
TH0	Timer/Counter 0 High Byte	8CH	0 0 0 0 0 0 0 0
TL0	Timer/Counter 0 Low Byte	8AH	0 0 0 0 0 0 0 0
TH1	Timer/Counter 1 High Byte	8DH	0 0 0 0 0 0 0 0
TL1	Timer/Counter 1 LowByte	8BH	0 0 0 0 0 0 0 0
+ TH2	Timer/Counter 2 High Byte	0CDH	0 0 0 0 0 0 0 0
+ TL2	Timer/Counter 2 Low Byte	0CCH	0 0 0 0 0 0 0 0
+ RCAP2H	T/C 2 Capture Reg. High Byte	0CBH	0 0 0 0 0 0 0 0
+ RCAP2L	T/C 2 Capture Reg. Low Byte	0CAH	0 0 0 0 0 0 0 0
* SCON	Serial Control	98H	0 0 0 0 0 0 0 0
SBUF	Serial Data Buffer	99H	Interminate
PCON	Power Control	87H	HMOS 0 X X X X X X X CHMOS 0 X X X 0 0 0 0

Table 11.3 List of all SFRs (* = Bit addressable, + = 8052 only)

Memory Organization in 8051

Fig. 11.8 shows the basic memory structure for 8051. It can access upto 64 K program memory and 64 K data memory. The 8051 has 4 Kbytes of internal program memory and 256 bytes of internal data memory.

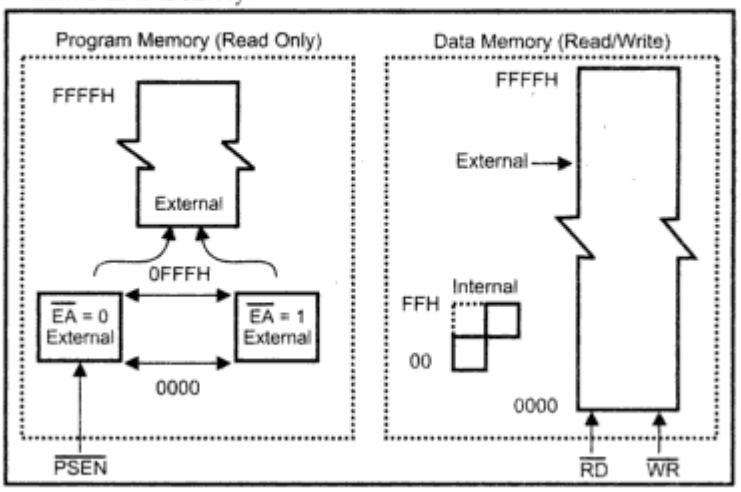


Fig. 11.8 Memory structure

External Data Memory and Program Memory

We have seen that 8051 has internal data and code memory with limited memory capacity. This memory capacity may not be sufficient for some applications. In such situations, we have to connect external ROM/EPROM and RAM to 8051 microcontroller to increase the memory capacity. We also know that ROM is used as a program memory and RAM is used as a data memory. Let us see how 8051 accesses these memories.

External Program Memory

Fig. 11.10 shows a map of the 8051 program memory.

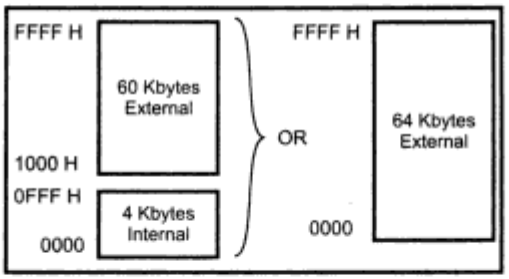


Fig. 11.10 The 8051 program memory

In 8051, when the $\overline{EA}$ pin is connected to V_{CC}, program fetches to addresses 0000H through 0FFFH are directed to the internal ROM and program fetches to addresses 1000H through FFFFH are directed to external ROM/EPROM. On the other hand when $\overline{EA}$ pin is grounded, all addresses (0000H to FFFFH) fetched by program are directed to the external ROM/EPROM. The $\overline{PSEN}$ signal is used to activate output enable signal of the external ROM/EPROM, as shown in the Fig. 11.11.

External Data Memory and Program Memory

We have seen that 8051 has internal data and code memory with limited memory capacity. This memory capacity may not be sufficient for some applications. In such situations, we have to connect external ROM/EPROM and RAM to 8051 microcontroller to increase the memory capacity. We also know that ROM is used as a program memory and RAM is used as a data memory. Let us see how 8051 accesses these memories.

External Program Memory

Fig. 11.10 shows a map of the 8051 program memory.

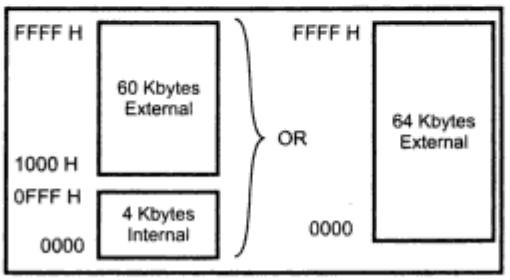


Fig. 11.10 The 8051 program memory

In 8051, when the $\overline{EA}$ pin is connected to V_{CC}, program fetches to addresses 0000H through 0FFFH are directed to the internal ROM and program fetches to addresses 1000H through FFFFH are directed to external ROM/EPROM. On the other hand when $\overline{EA}$ pin is grounded, all addresses (0000H to FFFFH) fetched by program are directed to the external ROM/EPROM. The $\overline{PSEN}$ signal is used to activate output enable signal of the external ROM/EPROM, as shown in the Fig. 11.11.

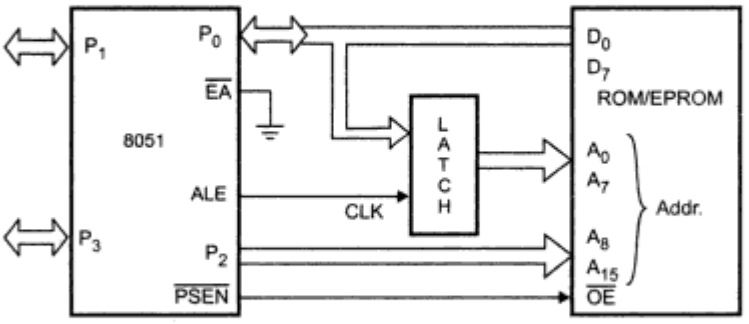


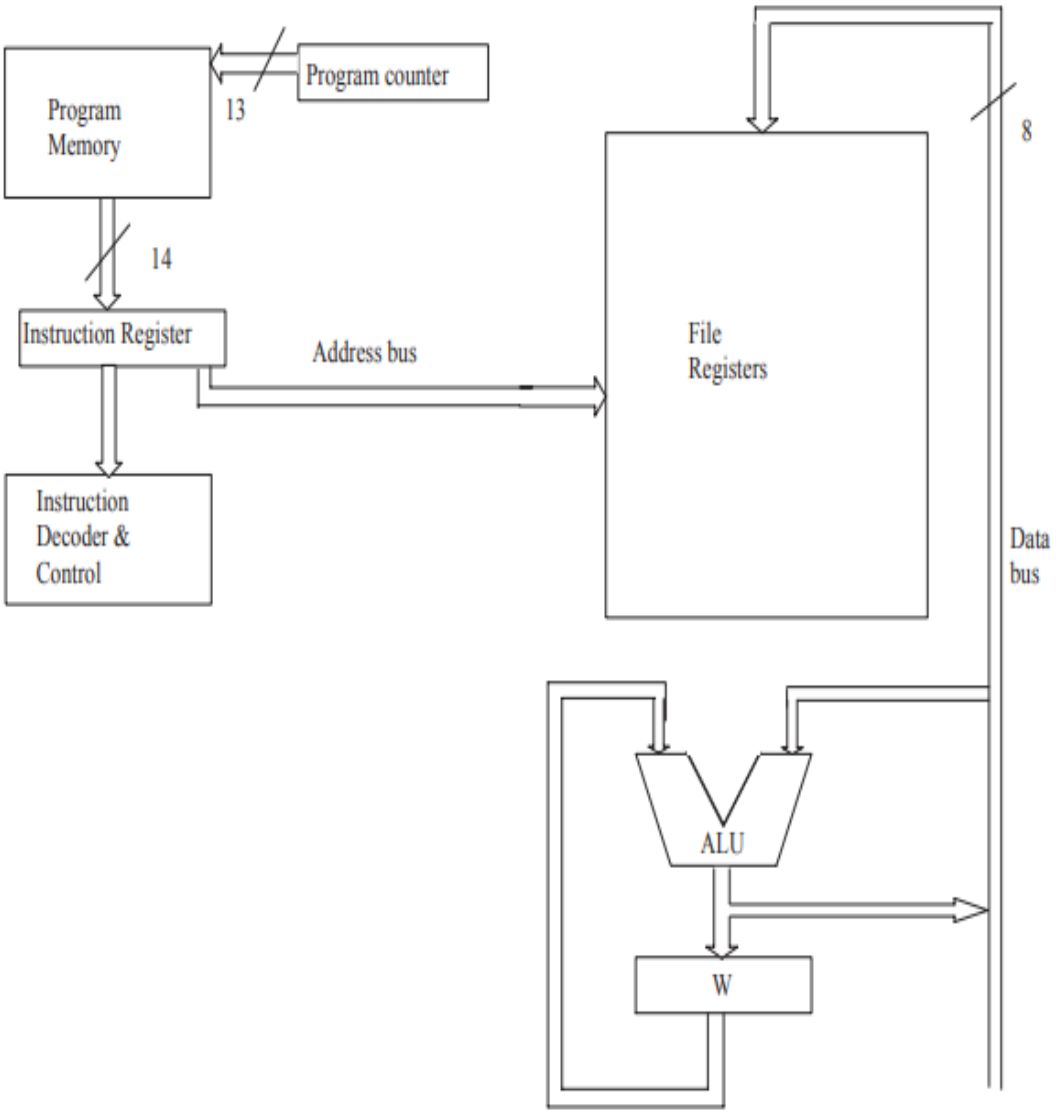
Fig. 11.11 Accessing external program memory

UNIT-VI

PIC Microcontrollers:

PIC stands for Peripheral Interface Controller coined by Microchip Technology to identify its singlechip microcontrollers. These devices have been phenomenally successful in 8-bit microcontroller market. The main reason is that Microchip Technology has constantly upgraded the device architecture and added needed peripherals to the microcontroller to 'suit customers' requirements.

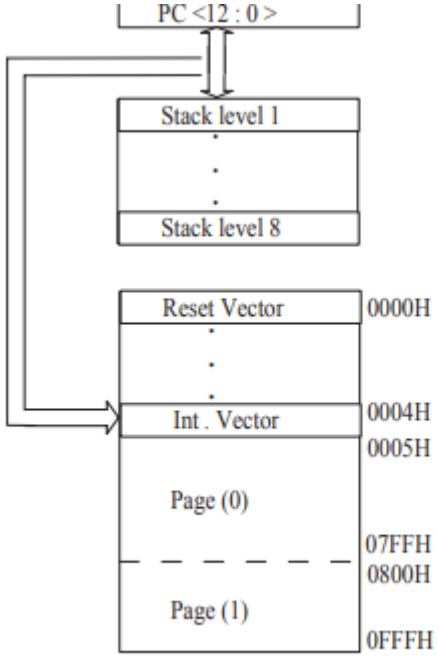
Basic Architecture of PIC Microcontroller



Memory Organization

The PIC 16C7X family has a 13-bit program counter capable of addressing 8k×14 program memory. PIC16C74A has 4k×14 program memory. For those devices with less than 8k program memory, accessing a location above the physically implemented address will cause a wraparound.

Program memory map and stack



Register Structure

When a ”write” message string is sent, the first byte selects the chip for a write and the second byte loads the pointer register. The write message string can stop there or it can continue with a 2-byte write of TOS (Over tem shutdown). Once the pointer has been set, any of their register can be read, reading two bytes for temperature, TOS, or THY ST or reading just 1 byte for the configuration register.

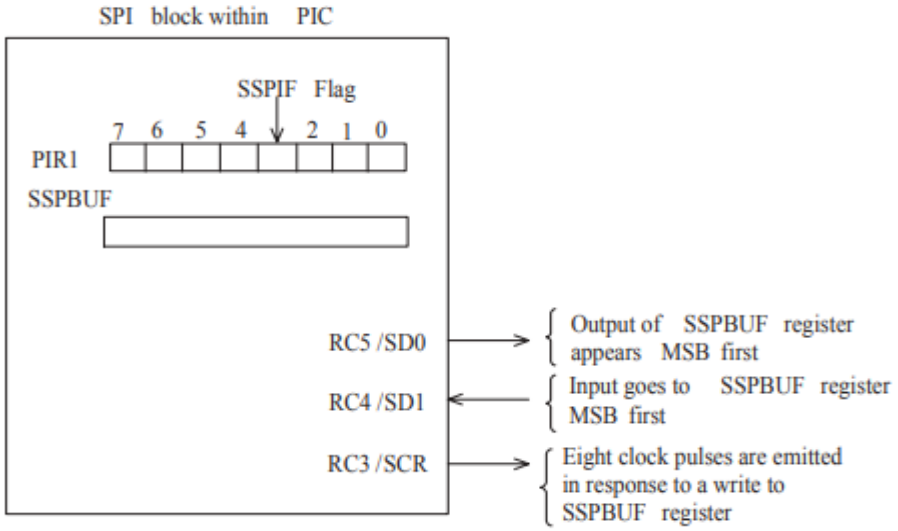
Synchronous Serial Port Module

Mid range PIC microcontroller includes a Synchronous Serial Port (SSP) module, which can be configured into either of two modes

Serial Peripheral Interface (SPI)

Inter-Integrated Circuit (I2C)

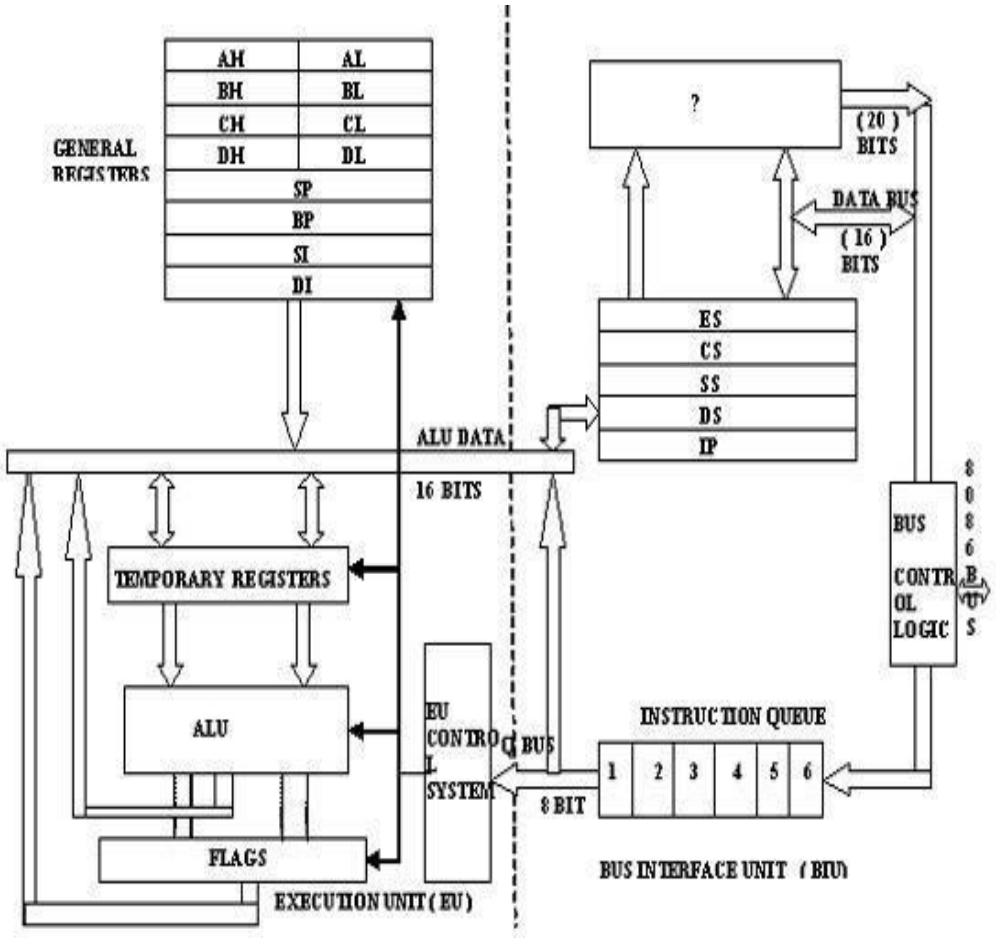
Serial Peripheral Interface



Portc three pins RC5, RC4 and RC3 are used for Synchronous Serial Interface. These pins revert to their normal general purpose I/O pins if neither of the two SSP modes is selected. The SPI port requires the RC3/SCK pin to be an output that generates the clock signal used by the external shift registers. This output line characterizes the SPI's master mode. In slave mode, RC3/SCK works as the input for the clock.

When a byte of data is written to SSPBUF register, it is shifted out the SDO pip in synchronism with the emitted pulses on the SCK pin. The MSB of SSPBUF is the first bit to appear on SDO pin. Simultaneously, the same write to SSPBUF also initiates the 8-bit data reception into SSPBUF of whatever appears on SDI pin at the time of rising edges of the clock on SCK pin.

ARM architecture



ARM is an 32 bit microcontroller which means that most available operations are limited to 8 bits. Short, Standard, and Extended. The Short and Standard chips are often available in DIP (dual in-line package) form, but the Extended models often have a different form factor, and are not "drop-in compatible". All these things are called because they can all be programmed using assembly language, and they all share certain features (although the different models all have their own special features).