DMI – ST.EUGENE UNIVERSITY

(Run by sisters of Daughters of Mary Immaculate and Collaborators)

Institute of Virtual and Distance Learning

Module Code : **777PH24**

Module Name : **MICRO PROCESSOR, MICRO CONTROLLER AND EMBEDDED SYSTEM**

Programme : **M.ED. IN PHYSICS EDUCATION**

CONTENT

UNIT 1 8086 MICRO PROCESSOR (16 BIT)

Overview of 8085 – Architecture - ALU Operations – Registers - 8085 Pin Description - Address and Data Buses - Addressing Modes for 8085 – Types -Instruction Set for 8085 - Features of 8086 Micro processor - Architecture of 8086 – Register - Addressing Modes of 8086 - Types - Instruction Set of 8086 - Pin Diagram.

UNIT II ADVANCED PROCESSOR

Architecture of 80286 - Features of 80286 - Register of 80286 - Functional blocks – Register – Interrupt - Architecture of 80386 - Features of 80386 – Architecture - Memory Management Unit - Pin Diagram - Pin Description - Register Organization - Addressing Modes – Segmentation - Paging Operation - Architecture of 80486 - Pin Definitions.

UNIT – III MICROCONTROLLER (8051)

Introduction - Features of 8051 - Difference between Micro processor and Microcontroller - Architecture of 8051 – Registers - Interrupt Control - Pin Diagram - Addressing Modes – Types - Instruction set - Interfacing with seven segment display Circuit – Program – LCD.

UNIT 4 ADVANCED MICROCONTROLLER

ARM Architecture – Registers - Block Diagram - Pin Diagram - Instruction Set - Addressing modes - THUMB Instruction - PIC Architecture – Memory – Registers - USART or UART – Oscillators - Memory Organization of PIC16F877 - Peripheral Features - Pin Diagram - Instruction Set - Register Operations - PC Architecture - Operation of Memory - Bus Characteristics - Addressing Modes - Instruction Formats.

UNIT 5 - INTRODUCTION TO EMBEDDED HARDWARE AND SOFTWARE

Logic Gates - Timing diagram - Instruction Cycle - Machine cycle - Flip Flop – Hazards – Memory - Micro processor Buses - Memory Access – Interrupts - Real-Time Operating Systems - Features of RTOS - Challenges in RT Systems Design - properties of RT Systems – Optimality.

UNIT 1

8086 MICRO PROCESSOR (16 BIT)

OVERVIEW OF 8085

The 8085 is a conventional von Neumann design based on the Intel 8080. Unlike the 8080 it does not multiplex state signals onto the data bus, but the 8-bit data bus is instead multiplexed with the lower 8-bits of the 16-bit address bus to limit the number of pins to 40. State signals are provided by dedicated bus control signal pins and two dedicated bus state ID pins named S0 and S1. Pin 40 is used for the power supply (+5 V) and pin 20 for ground. Pin 39 is used as the Hold pin. The PROCESSOR was designed using nMOS circuitry, and the later "H" versions were implemented in Intel's enhanced nMOS process called HMOS ("High-performance MOS"), originally developed for fast static RAM products. Only a single 5 volt power supply is needed, like competing PROCESSOR and unlike the 8080. The 8085 uses approximately 6,500 transistors.

The 8085 incorporates the functions of the 8224 (clock generator) and the 8228 (system controller) on chip, increasing the level of integration. A downside compared to similar contemporary designs (such as the Z80) is the fact that the buses require de-multiplexing; however, address latches in the Intel 8155, 8355, and 8755 memory chips allow a direct interface, so an 8085 along with these chips is almost a complete system.

The 8085 has extensions to support new interrupts, with three maskable vectored interrupts (RST 7.5, RST 6.5 and RST 5.5), one non-maskable interrupt (TRAP), and one externally serviced interrupt (INTR). Each of these five interrupts has a separate pin on the PROCESSOR, a feature which permits simple systems to avoid the cost of a separate interrupt controller.

The RST 7.5 interrupt is edge triggered (latched), while RST 5.5 and 6.5 are level-sensitive. All interrupts are enabled by the EI instruction and disabled by the DI instruction. In addition, the SIM (Set Interrupt Mask) and RIM (Read Interrupt Mask) instructions, the only instructions of the 8085 that are not from the 8080 design, allow each of the three maskable RST interrupts to be individually masked. All three are masked after a normal CPU reset. SIM and RIM also allow the global interrupt mask state and the three independent RST interrupt mask states to be read, the pending-interrupt states of those same three interrupts to be read, the RST 7.5 trigger-latch flip-flop to be reset (cancelling the pending interrupt without servicing it), and serial data to be sent and received via the SOD and SID pins, respectively, all under program control and independently of each other.

SIM and RIM each execute in 4 clock cycles (T states), making it possible to sample SID and/or toggle SOD considerably faster than it is possible to toggle or sample a signal via any I/O or memory-mapped port, e.g. one of the port of an 8155. (In this way, SID can be compared to the SO ["Set Overflow"] pin of the 6502 CPU contemporary to the 8085.)

Like the 8080, the 8085 can accommodate slower memories through externally generated wait states (pin 35, READY), and has provisions for Direct Memory Access (DMA) using HOLD and HLDA signals (pins 39 and 38). An improvement over the 8080 is that the 8085 can itself drive a piezoelectric crystal directly connected to it, and a built-in clock generator generates the internal high amplitude two-phase clock signals at half the crystal frequency (a 6.14 MHz crystal would yield a 3.07 MHz clock, for instance). The internal clock is available on an output pin, to drive peripheral devices or other CPUs in lock-step synchrony with the CPU from which the signal is output. The 8085 can also be clocked by an external oscillator (making it feasible to use the 8085 in synchronous multi-PROCESSOR

systems using a system-wide common clock for all CPUs, or to synchronize the CPU to an external time reference such as that from a video source or a high-precision time reference).The 8085 is a binary compatible follow up on the 8080. It supports the complete instruction set of the 8080, with exactly the same instruction behavior, including all effects on the CPU flags (except for the AND/ANI operation, which sets the AC flag differently). This means that the vast majority of object code (any program image in ROM or RAM) that runs successfully on the 8080 can run directly on the 8085 without translation or modification. (Exceptions include timing-critical code and code that is sensitive to the aforementioned difference in the AC flag setting or differences in undocumented CPU behavior.) 8085 instruction timings differ slightly from the 8080—some 8-bit operations, including INR, DCR, and the heavily used MOV r,r' instruction, are 1 clock cycle faster, but instructions that involve 16-bit operations, including stack operations (which increment or decrement the 16-bit SP register) generally 1 cycle slower. It is of course possible that the actual 8080 and/or 8085 differs from the published specifications, especially in subtle details. (The same is not true of the Z80.) As mentioned already, only the SIM and RIM instructions were new to the 8085.

FEATURES OF 8085

- It is a 8-bit general purpose Micro processor
- Capable of addressing 64 k of memory
- It is a 40 pins IC
- Requires +5 v power supply
- Can operate with 3 -5MHz clock frequency.
- 50% of duty cycle.

ARCHITECTURE:

The architecture of Intel 8085 consists of three main sections:

1. Arithmetic and logic unit.
2. Timing and control unit.
3. Registers.

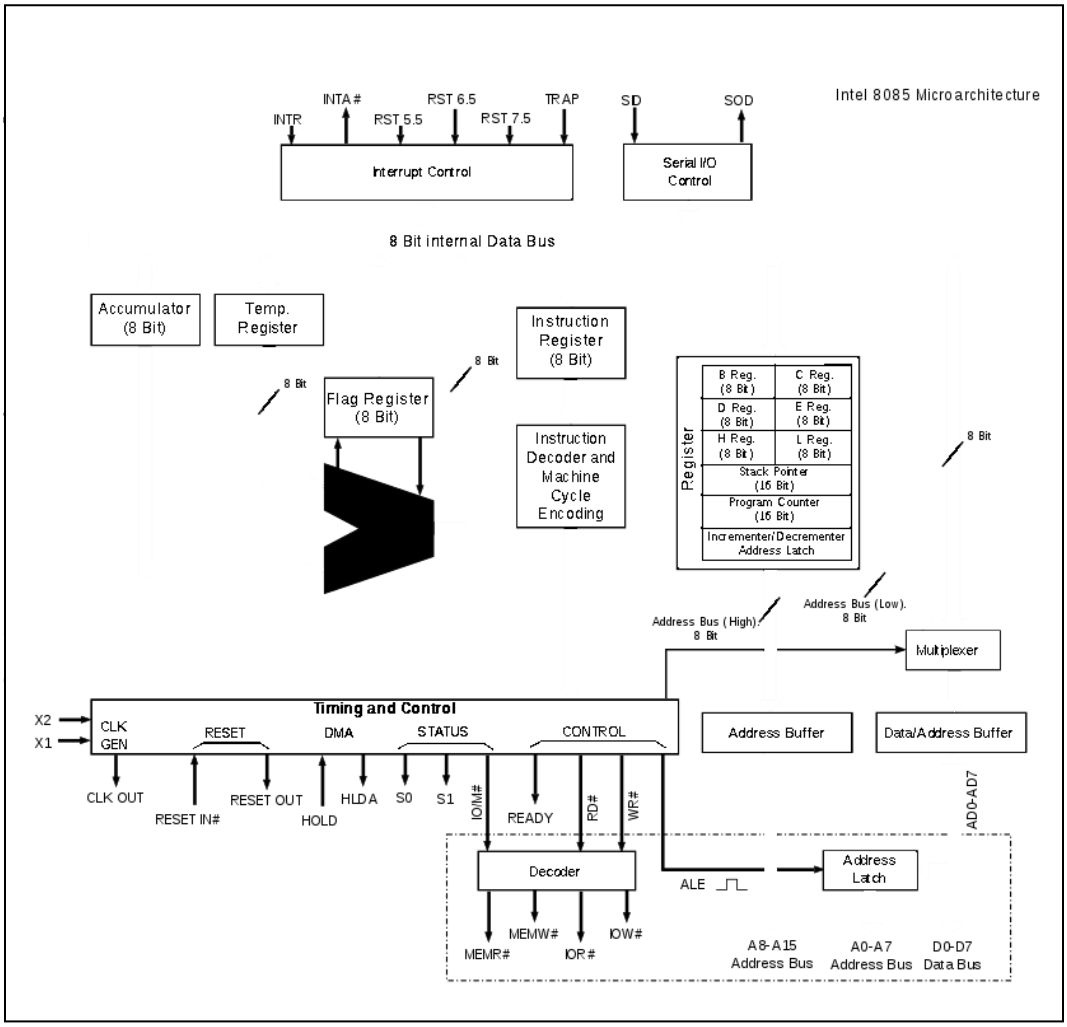


Fig : Architecture of Intel 8085

Arithmetic and Logic Unit (ALU)

The ALU performs the following arithmetic and logical operations. It

consists of Temporary register, Flag Register, and Accumulator. One operand in A and another operand in Memory or general purpose register. After perform the operation result stores result of operation stored in Accumulator. Depend upon the Result Flag register will be changed.

Timing and Control Unit (Brain of the Micro processor):

It controls the operation of different units of the CPU. It controls the data flow between **CPU** and Memory, CPU and peripheral devices. It provides control, status and reset signals to perform any memory and input output related operation.

Registers

1. Accumulator
2. Six 8-bit General Purpose Register: B, C, D, E, H, and L.
3. Temporary Register- W and Z register which is not available to programmer. Because it is used internally located above B and C register
4. Special Purpose Register
    - Instruction Register
    - Accumulator
    - Flag Register
    - Stack pointer-16 bit
    - Program counter- 16 bit
    - Memory Address Register

Instruction Register

They can be combined as register pairs BC, DE, and HL - to perform some 16 - bit operations. The programmer can use these registers to store or copy data into the registers by using data copy instructions.

Accumulator

The accumulator is an 8-bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flag Register:

The status each flip flop is known as Flag register. The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags; they are listed in the Table and their bit positions in the flag register are shown below.

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	X	AC	X	P	X	CY

The most commonly used flags are Zero, Carry, and Sign. The Micro processor uses these flags to test data conditions.

Sign Flag

After the arithmetic or logical operation D7 bit is 1.sign flag will be set. Otherwise Reset.

Zero Flag

when arithmetic operation results is zero, the Zero (Z) flag is set to one Auxiliary Carry Flag: The flag is set overflow out of bit 3

Parity Flag

Number of one's present in accumulator. If there is even number of one's Even parity will set. If there is odd number of one's odd parity will set.

Carry Flag

The flag is set if there is overflow out of bit 7.

Program Counter (PC)

This 16-bit Special Purpose register deals with sequencing the execution of instructions. This register is a memory pointer. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location.

Stack Pointer (SP)

The stack pointer is also a 16-bit register which is used to point memory location called stack. It sequence of memory location in R/W memory. The beginning of the stack is defined by loading 16-bit address in the stack pointer.

Memory Address Register (MAR)

It holds the address of next program instruction. Then MAR feeds the address bus with address of the memory location of the program instruction which will be executed.

Instruction Register/Decoder

Temporary storage for the current instruction of a program. Latest instruction sent here from memory prior to execution. Decoder then takes instruction and 'decodes' or interprets the instruction. Decoded instruction then passed to next stage.

System Bus:

It is a collection of wire which is used to transfer the data from source to destination.

Types:

1. Address Bus→It transfer 16 bit address
2. Data Bus→It carry 8 bit data
3. Control Bus→It carry control signals

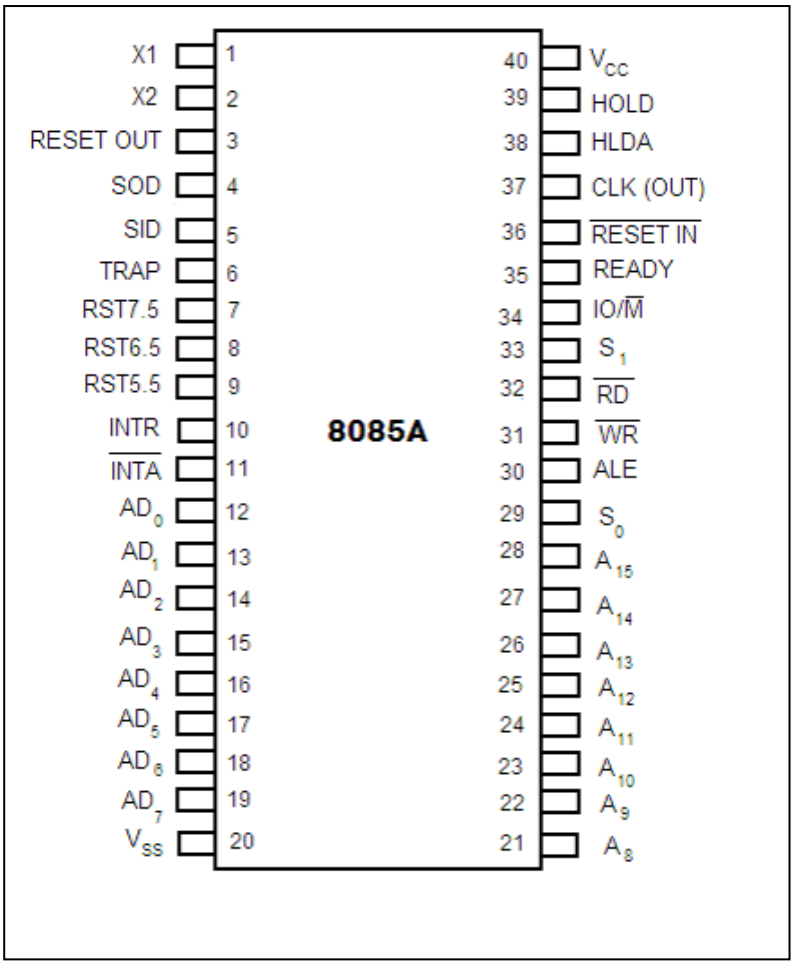
Data Bus:

Data bus carries data in binary form between Micro processor and other external units such as memory. It is used to transmit data i.e. information, results of arithmetic etc between memory and the Micro processor. Data bus is bidirectional in nature. The data bus width of 8085 Micro processor is 8-bit i.e. 28 combination of binary digits and are typically identified as D0 - D7. Thus size of the data bus determines what arithmetic can be done. If only 8-bit wide then largest number is 11111111 (255 in decimal). Therefore, larger numbers have to be broken down into chunks of 255. This slows Micro processor.

Address Bus: The address bus carries addresses and is one way bus from Micro processor to the memory or other devices. 8085 Micro processor contain 16-bit address bus and are generally identified as A0 - A15. The higher order address lines (A8 - A15) are unidirectional and the lower order lines (A0 - A7) are multiplexed (time-shared) with the eight data bits (D0 - D7) and hence, they are bidirectional.

Control Bus: Control buses are various lines which have specific functions for coordinating and controlling Micro processor operations. The control bus carries control signals partly unidirectional and partly bidirectional. The following control and status signals are used by 8085 PROCESSOR:

(i) **8085 PIN DESCRIPTION:**



ALE (output)

Address latch enable, during the first clock state of a machine cycle, it become high and enables the address to get latched either in to the memory or external latch. The falling edge of ALE is set to guarantee set-up, can hold times for the address information. The falling edge ALE can also be used to strobe the status information.

RD (active low output)

The Read signal indicates that data are being read from the selected I/O or memory device and that they are available on the data bus.

WR (active low output)

The Write signal indicates that data on the data bus are to be written into a selected memory or I/O location.

IO/M (output)

It is a signal that distinguished between a memory operation and an I/O operation. When IO/M= 0 it is a memory operation and IO/M= 1 it is an I/O operation.

S1 and SO (output)

These are status signals used to specify the type of operation being performed; they are listed in below table.

IO/M(Active Low)	S1	S2	Data Bus Status(Output)
0	0	0	Halt
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	IO WRITE
1	1	0	IO READ
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge

Fig : S1 and SO (output)

The Micro processor performs primarily four operations:

I. Memory Read: Reads data (or instruction) from memory.

II. Memory Write: Writes data (or instruction) into memory.

III. I/O Read: Accepts data from input device.

IV. I/O Write: Sends data to output device.

Address and Data Buses

- A8 - A15 (output, 3-state): Most significant eight bits of memory addresses and the eight bits of the I/O addresses. These lines enter into tri-state high impedance state during HOLD and HALT modes.

- AD0 - AD7 (input/output, 3-state): Lower significant bits of memory addresses and the eight bits of the I/O addresses during first clock cycle. Behaves as data bus
- Power Supply & Clock Frequency
    - Vcc: +5 V power supply
    - Vss: Ground reference
    - X1, X2: A crystal having frequency of 6 MHz is connected at these two pins
    - CLK: Clock output
- Externally Initiated and Interrupt Signals
    - RESET IN: When the signal on this pin is low, the PC is set to 0, the buses are tri-stated and the PROCESSOR is reset.
    - RESET OUT: This signal indicates that the PROCESSOR is being reset. The signal can be used to reset other devices.
    - READY: When this signal is low, the PROCESSOR waits for an integral number of clock cycles until it goes high.
    - HOLD: This signal indicates that a peripheral like DMA (direct memory access)
    - HLDA: This signal acknowledges the HOLD request.
    - INTR: Interrupt request is a general-purpose interrupt.
    - INTA : This is used to acknowledge an interrupt.
    - RST 7.5, RST 6.5, RST 5,5 - restart interrupt:
    - These are vectored interrupts and have highest priority than INTR interrupt.
    - TRAP: This is a non-maskable interrupt and has the highest priority.

Interrupt	Interrupt vector address	Maskable or non-maskable	Edge or level triggered	priority
TRAP	0024H	Non-makable	Level	1
RST 7.5	003CH	Maskable	Rising edge	2
RST 6.5	0034H	Maskable	Level	3
RST 5.5	002CH	Maskable	Level	4
INTR	Decided by hardware	Maskable	Level	5

- Serial I/O Signals
    - SID: Serial input signal. Bit on this line is loaded to D7 bit of register A using RIM instruction.
    - SOD: Serial output signal. Output SOD is set or reset by using SIM instruction.

ADDRESSING MODES FOR 8085

The method of specifying the data or operand to be operated.

Types:

1. Immediate Addressing Mode
2. Register Addressing Mode
3. Direct Addressing Mode
4. Indirect Addressing Mode
5. Implied Addressing Mode

Immediate Addressing Mode

The data is present in the instruction itself is called Immediate Addressing Mode.

Ex: MVI A, 03H-^The data 03 is moved to accumulator.

ADI 03H^The data 03 is added immediately to the accumulator.

Register Addressing Mode

The data is moved from one register to another register is called Register Addressing Mode .

Ex: MOV A, The data is moved from one register to another register ADD C→The data present in C register is added with data present in accumulator and saved in accumulator.

Direct Addressing Mode

The address of the operand always exist within the instruction.

Ex: LDA 4500H -→Load the content of the memory location 4500 in to the accumulator.

STA 4600H→Store the content of the accumulator in the memory location 4600.

Indirect Addressing Mode

The content of the register is used to specify the address of the operand

Ex: MOV A, M→Move the content of the memory location whose address is given in H and L register in the accumulator.

ADD M→ Addition of the content of the memory location whose address is given in H and L register and in the accumulator.

Implied Addressing Mode

Opcode specifies the address of the operand.

EX: CMA→complements the content of the accumulator RAL→Rotate the content of the accumulator left through carry.

Instruction set for 8085

The command applied to the Micro processor to perform a specific function.

Types:

1. Data Transfer Instruction
2. Arithmetic Instructions
3. Logical Instructions
4. Branching Instructions
5. Stack Input/output and Machine Control Instructions

Data Transfer Instruction

The data is copied from source to Destination without any change.

Ex:MOV A,B MOV A,B→ The data is moved from one register to another register

MVI A,03H-→The data 03 is moved to accumulator.

LDA 4500H -→Load the content of the memory location 4500 in to the accumulator STA 4600H-→ Store the content of the accumulator in the memory location 4600.

Arithmetic Instructions

To perform arithmetic operation such as addition, subtraction, Increment and decrement.

EX: ADD C→The data present in C register is added with data present in accumulator and saved in accumulator.

SUB The data present in B register is subtracted with data present in accumulator and saved in accumulator.

INR B→The content of B register incremented by one DCR C→The content of C register decremented by one..

Logical Instructions:

The group of instruction perform logical operation such as AND,OR, Exclusive-OR, Rotate, Compare, and Complement with the content of the accumulator.

Ex: ANA B→The data present in the B register AND with data present in accumulator and stored in the accumulator.

ORA M→ The data present accumulator Ors with data present in the memory pointed by HL register pair and stored in the accumulator.

CMA→Complement the content of the accumulator.

RLC→Rotate the content of the accumulator left by one position.

Branching Instructions

The instruction changes the sequence of program execution using conditional, unconditional jumps, subroutine call and return

EX: JMP 4500H→The Program execution jump to the specified address.

JC -→Jump on carry

JNZ -→Jump on non zero

RET→Return from subroutine unconditionally.

Stack input /output and Control Instructions

It is used to manipulate stack related operation and Input/output Machine control operation. EX: PUSH B→Push the content of register B and C to the stack.

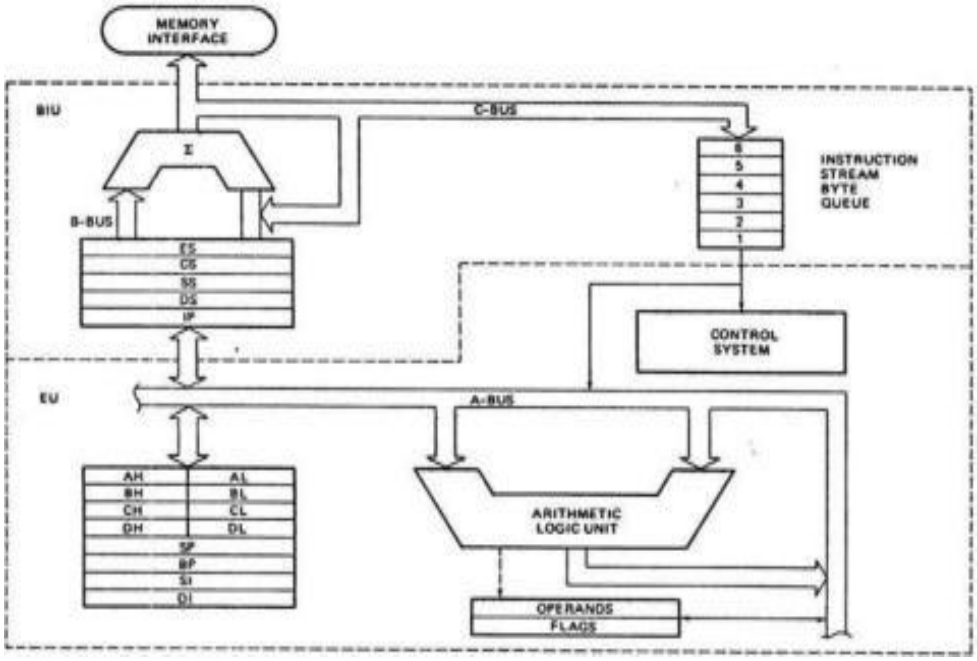
POP H→Pop off stack to register H and L pair.

NOP→No operation HALT→Halt and Enter wait state.

2. Features of 8086 Micro processor

- It is 16 bit PROCESSOR.
- It is designed using HMOS technology in , available in three clock rates:5,8,10MHZ.
- It has 29000 transistors.
- 8086 is packed 40 Pin IC.
- It doesn't have internal clock circuit.
- 33% of duty cycle.
- Its operating frequency is 5 V.
- It has fourteen 16 bit registers.
- It has 16 bit data bus and 20 bit address bus(1 MB memory)

2. ARCHITECTURE OF 8086



It is Pipelined architecture. The 8086 CPU is divided into two functional units: Bus Interface Unit (BIU) , Execution Unit (EU)

The Bus Interface Unit (BIU)

The BIU handles all data and addresses on the buses for the execution unit such as it sends out addresses, fetches instructions from memory, reads/writes data from ports and memory. It contains the bus interface logic, Segment register, stack pointer, base pointer, index pointer, memory address logic and 6 byte Instruction queue (FIFO). BIU fetches the instruction code from memory and stores in the queue.

Instruction Queue

To increase the execution speed, BIU fetches as many as six instruction bytes ahead to time from memory. The prefetched instruction bytes are held for the EU in a first in first out group of registers called a **instruction**

queue. When the EU is ready for its next instruction, it simply reads the instruction from this instruction queue. This is much faster than sending out an address to the system memory and to send back the next instruction byte. Fetching the next instruction while the current instruction executes is called **pipelining**.

Execution unit (EU):

It consists of ALU, General purpose register, Flag register, Instruction decoder, Pointer and index register and the control unit which are required to execute an instruction. EU gets the opcode of the instruction from instruction queue. Then it will be decoded and executed .BIU and EU are independent. The overlapping operation of BIU and EU functional unit of a Micro processor is called pipelining.

Register:

It has fourteen 16 bit register .All the register subdivide in to

1. Data Register
2. Segment Register
3. Pointer Register
4. Index register (program counter)
5. Flag register
6. Data register

It has four 16 bit general purpose register (AX, BX, CX, DX)

16 bit Register	8 bit High-order Register	8 bit lower order Register
AX	**AH**	**AL**
BX	**BH**	**BL**
CX	**CH**	**CL**
DX	**DH**	**DL**

AX register:

- It serves as accumulator.
- It performs I/O operation and process data through AX, AH, AL.
- Result stored in accumulator.
- For 16 bit multiplication/division AX contain one word operand.
- For 32 bit multiplication/division AX contain lower order word operand.

BX register

It acts as a index register for MOVE operation. Base register for data memory address.

CX register

- CX register can be used as a count register for string operation.
- It will have the count.
- It has large number of iteration.
- CX holds desired number of repetition and it automatically decremented by one after the execution of instruction.
- CX become zero , the execution instruction is terminated.

DX register

- DX can be used as a port address for IN and OUT instruction.
- The DX can be used in I/O instruction, Divide, Multiple Instruction.
- In 32 bit multiplication/division instruction DX is used to

hold the higher order word operand.

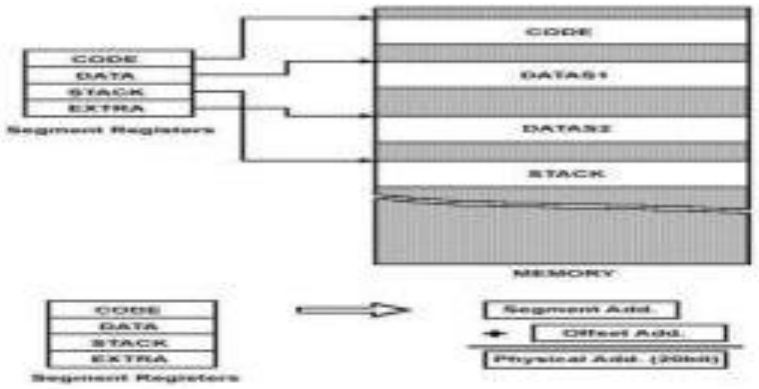
Segment Registers:

The BIU contains four 16-bit segment registers, one MB memory. Each divided in to 16 parts which are called segments. Each segment occupies the 64KB memory space.

They are:

- Code segment (CS) register
- Data segment (DS) register
- Stack segment (SS) register
- Extra segment (ES) register

These segment registers are used to hold the upper 16 bits of the starting address for each of the segments. The part of a segment starting address stored in a segment register is often called **the segment base.**



1. Code Segment (CS)

The CS register is used for addressing a memory location in the Code Segment of the memory, where the executable program is stored.

2. Data Segment (DS)

The data segment register points to the data segment of the memory, where

data is stored

3. Stack Segment (SS)

It is used to addressing stack segment of the memory in which data is stored.

4. Extra Segment (ES)

It can be used as another data segment of the memory.

Pointer and Index register:

1. Stack Pointer(SP)
2. Base Pointer(BP)
3. Source Index(SI)
4. Destination Index(DI)
5. Instruction Pointer(IP**)**

Flag register

15	**1**	**1**	**1**	**1**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
X	**X**	**X**	**X**	**OF**	**DF**	**IF**	**TF**	**SF**	**ZF**	**X**	**AF**	**X**	**PF**	**X**	**CF**

***S*- Sign Flag :** This flag is set, when the result of any computation is negative.

***Z*- Zero Flag**: This flag is set, if the result of the computation or comparison performed by the previous instruction is zero.

P- **Parity Flag:** This flag is set to 1, if the lower byte of the result contains even number of 1's.

C- **Carry Flag**: This flag is set, when there is a carry out of MSB in case of addition or a borrow in case of subtraction.

***T*- Tarp Flag**: If this flag is set, the PROCESSOR enters the single step exe. mode.

Interrupt Flag: If this flag is set, the maskable interrupt are recognized by the CPU, otherwise they are ignored.

***D*- Direction Flag:** This is used by string manipulation instructions. If this

flag bit is ‘0’, the string is processed beginning from the lowest address to the highest address, i.e., auto incrementing mode. Otherwise, the string is processed from the highest address towards the lowest address, i.e., auto incrementing mode.

C-**Auxilary Carry Flag**: This is set, if there is a carry from the lowest nibble, i.e, bit three during addition, or borrow for the lowest nibble, i.e, bit three, during subtraction.

O- **Over flow Flag:** This flag is set, if an overflow occurs, i.e, if the result of a signed operation is large enough to accommodate in a destination register. The result is of more than 7-bits in size in case of 8-bit signed operation and more than 15-bits in size in case of 16-bit sign operations, then the overflow will be set.

3. Addressing modes of 8086

The method of specifying the data to be operated**.**

Types:

1. Immediate Addressing mode:
2. Register Addressing mode:
3. Direct addressing mode
4. Register indirect addressing mode
5. Based addressing mode
6. Indexed addressing mode
7. Based Indexed addressing mode
8. Based Indexed with displacement addressing mode
9. String addressing mode
10. Branch addressing mode
11. Direct IO port addressing
12. Indirect IO port addressing

13. Relative addressing mode
14. Implied addressing mode

Immediate Addressing mode

The 8 bit or 16 bit data is provided in the instruction itself.

EX: MOV BX, 5000H^Load 5000H in to BX register.

MOV CX, 4500H^ Store 4500H in to CX register.

Register Addressing mode

The data in a register or in a register pair specified by the instruction.

EX: MOV AL, BL^The content present in BL register is copied to AL register.

MOV AX, BX^The content present in BX register is copied to AX register.

Direct addressing mode

The instruction or operand specifies the memory address where data is located.

EX: MOV AX, [5000H] ^copies the 2 byte of data starting from memory location DS*10+5000H to AX register. LSB^Data from DS*10+5000H, MSB^ Data from DS*10+5001H, MOV AX, SS: [2000H]: Similarly to access the memory from Stack segment.

Register indirect addressing mode

The instruction specifies the register containing the address where the data is located. Where data is located. This addressing mode works with SI, DI, BX and BP registers.

EX: MOV AL, [BX]^The Data present in BX register pointed to memory location (which default indicates the data segment) is moved to AL register. MOV AL, CS:[SI]^ The Data present in SI register pointed to memory location which indicates the code segment is moved to AL register.

Based addressing mode

The 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.

EX: MOV AL, [BX+8 Bit DISP]^The content of memory location pointed by physical address (PA) of the base register BX with displacement is copied to AL register.

MOV AH, [BP+8 Bit DISP]^ The content of memory location pointed by physical address (PA) of the base register BP with displacement is copied to AH register.

Indexed addressing mode

The 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.

EX: MOV AL, CS: [SI+DISP]^ The content of memory location pointed by physical address (PA) of the index register SI with displacement which is present in code segment is copied to AL register.

MOV AL, SS: [DI+DISP]^ The content of memory location pointed by physical address (PA) of the index register DI with displacement which is present in stack segment is copied to AL register.

Based Indexed addressing mode

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.

EX: MOV AL, [BX+DI]^ The content of memory location pointed by physical address(PA) of the summation base register BX and index register DI is copied to AL register.

MOV AL, [BP+SI The content of memory location pointed by physical

address (PA) of the summation base register BP and index register SI is copied to AL register.

Based Indexed with displacement

The 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) with displacement value, the resulting value is a pointer to location where data resides.

EX: MOV AL, [BX+DI+DISP]^The 16-bit operand is added to the contents of a BX and DI with displacement ,the resulting value is a pointer to location of AL. MOV AL, [BP+SI+DISP]^ The 16-bit operand is added to the contents of a BP and SI with displacement ,the resulting value is a pointer to location of AL.

String Addressing Mode

String is a byte or word s which are stored in memory.

EX: MOV SB→The memory source address is the SI register in the data segment. The memory of destination address is the DI register in the extra segment.

Branch Addressing Mode

- Intra segment mode^ Intra segment Direct, Intra segment Indirect
- Inter segment mode^ Inter segment Direct, Inter segment Indirect

EX: JNC START If CY=O, then PC is loaded with current PC contents plus 8 bit signed value of START, otherwise the next instruction is executed.

Direct IO port addressing

It is used to access data from standard IO-mapped devices or ports. In the direct port addressing mode, an 8-bit port address is directly specified in the instruction.

EX: IN AL, [09H] →The content of the port with address 09H is moved to the AL register.

Indirect IO port addressing

It is used to access data from standard IO mapped devices or ports. In the indirect port addressing mode, the instruction will specify the name of the register which holds the port address.

EX: OUT [DX],AX→The content of the AX is moved to the port whose address is specified by the DX register.

Relative Addressing mode

The effective address of a program instruction is specified relative to instruction pointer(IP) by an 8 bit signed displacement.

EX: JZ 0AH^Jump on Zero

Implied Addressing mode

The instruction itself will specify the data to be operated by the instruction.

EX: CLC→clear flag CMC→ complement carry flag

INSTRUCTION SET OF 8086:

The command applied to the Micro processor to perform a specific function.

TYPES:

1. Data transfer Instruction
2. Arithmetic and Logical Instruction
3. Branch Instruction
4. Loop Instruction
5. Process control Instruction
6. Flag Manipulation Instruction
7. Shift and Rotate Instruction.
8. String Manipulation Instruction

Data transfer Instruction

The data is copied from source to Destination without any change. i.e register to register Memory to register; register to memory, Data given

immediately to register or memory. [MOV,LDS,XCHG,PUSH,POP]

EX: MOV AX, SI→The content of SI is moved to the AX register. XCHG DH, CL→The content of CL register is exchanged with content of DH register.

Arithmetic and Logical Instruction

It performs the arithmetic, logical, increment, decrement, compare and scan instruction.[ADD,SUB,MUL,DIV,INC,CMP,DAS,AND,OR,NOT,TEST,XOR]

EX: ADC BH,CL→The content of BH register ,the AL register and the carry flag are added. The result is stored in the BH- register.

XOR BX,DX→The content of the BX and DX registers are Exclusive ORed .This result is stored in BX register.

Control Transfer Instruction

- Branch Instruction
- Loop Instruction

Branch Instruction

The instruction will transfer the control of execution to the specified address. The JUMP, CALL, interrupt and Return belongs to the Branch instruction.

EX: JNC 4500H→Jump if no carry to the memory location 4500H

CALL -→This Instruction is used to transfer execution to a subprogram or procedure. There are two basic types of CALL: Near and Far.

JCXZ Instruction → Jump if the CX register is zero

Loop Instruction

These instructions have REP prefix with CX used as count register,

and they can be used to implement unconditional and conditional loops. The LOOP, LOOPNZ, LOOPZ instruction belong to this Loop instruction. It is used to implement different delay loops.

EX: LOOP Instruction →Loop to specified label until CX = 0

LOOPE / LOOPZ → loop while CX →0 and ZF = 1

Process control Instruction

The instruction control the machine status CLC, CMC, CLI, STD, STI, NOP, HLT, WAIT and LOCK instruction.

EX:CLC→Carry flag is reset to zero.

CMC→Carry flag is complemented.

HLT→Halt the program execution.

WAIT→Wait for test line active.

Flag Manipulation Instruction

All the instruction which directly affect the flag registers come under this group of instruction. Instructions like CLD, STD, CLI, and STI belong to this category.

EX:STD: Set direction Flag

STC: Set Carry Flag

Shift and Rotate Instruction: The instruction involves in the bitwise shifting OR Rotation in either direction with or without a count in CX. The example instructions are RCL, RCR, ROL, ROR, SAL, SHL, SAR and SHR.

EX: ROR AX, CL→ Rotate word or byte operand right by CL times. So CF and OF flag gets affected.

SAL AX, CL→ Shift word or byte operand CL times.

Note: Shift and Rotate examples comes under Logical

String Instruction

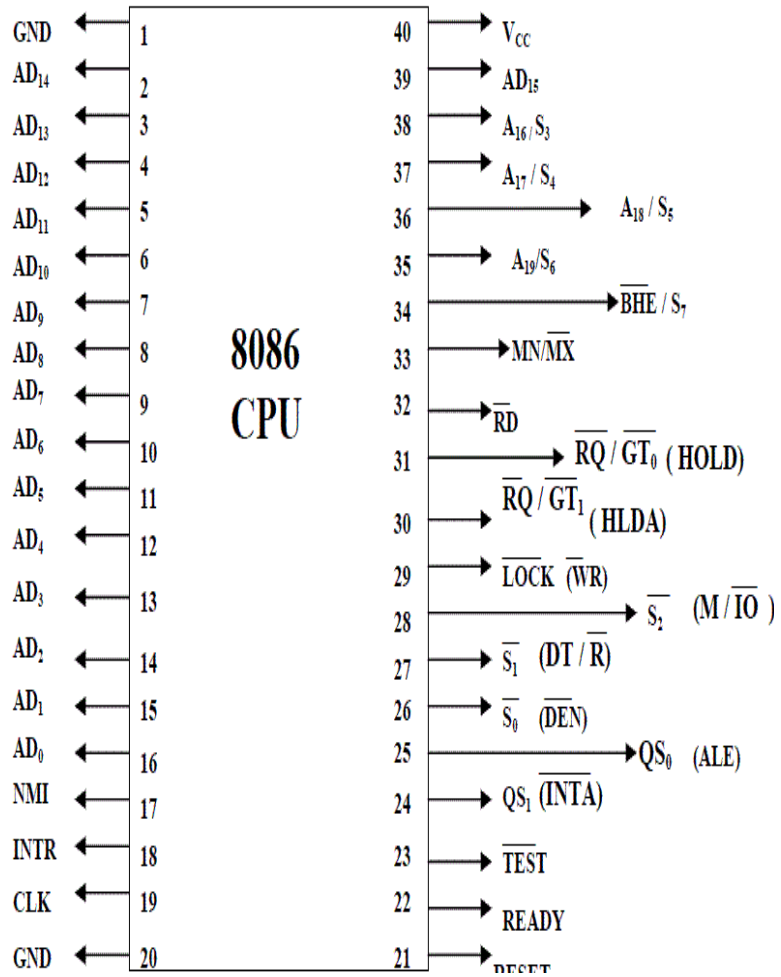
These instructions involve various string manipulation operations like load, move, scan, compare, store .

EX: MOVS, LODS and STOS.

EX: LODSW Instruction → Load string byte into AL or Load string word into AX

MOVSW Instruction → Move string byte or string word.

5. PIN DIAGRAM



- AD7-AD0 : address/data bus(multiplexed)
  - memory address or I/O port no : whenever ALE = 1
  - data : whenever ALE = 0
  - high-impedance state : during a hold acknowledge
- A15-A8 : address bus
  - high-impedance state : during a hold acknowledge
- AD15-AD8 : address/data bus(multiplexed)
  - Memory address bits A15 - A8 : whenever ALE = 1
    - Data bits D15 - D8 : whenever ALE = 0
    - high - impedance state : during a hold acknowledge
- AD15 - AD0 → Address/Data Bus
- A19 - A16 → Address/Status
- A19/S6 - A16/S3 : address/status bus(multiplexed)
    - Memory address A19 - A16, status bits S6 - S3
    - High - impedance state : during a hold acknowledge
    - S6 : always remain a logic 0
    - S5 : indicate condition of IF flag bits
    - S4, S3 : show which segment is accessed during current bus cycle
    - S4, S3 : can used to address four separate 1M byte memory banks by decoding them as A21, A20
- TEST'(BUSY') : tested by the WAIT instruction
    - WAIT instruction function as a NOP : if TEST'= 0
    - WAIT instruction wait for TEST' to become 0:if TEST'=1
- INTR: Interrupt request
  - It is a level triggered input which is sampled during the last clock cycle of each instruction to determine the PROCESSOR should enter

in to interrupt vector look up table located in the system memory.

- It is internally masked by software

- NMI : Non-maskable interrupt
  - similar to INTR except that no check IF flag bit
  - if NMI is activated : use interrupt vector 2
- RESET :

  •Reset signal active high for minimum of four clock cycles.
  - The signals terminate its present activity and the system is reset.
- VCC(power supply) : +5.0V, ±10%
- GND(Ground) : two pins labeled GND
- MN/MX' : select either minimum or maximum mode

  •Low—Maximum

  •High--Minimum
- BHE'/S7 : bus high enable
  - Enable the most significant data bus bits(D15 - D8) during read or write operation
  - Status of S7 : always a logic 1
- Minimum Mode Pins: MN = 1(directly to +5.0V) next IO/M'(8088) or M/IO'(8086) : select memory or I/O
  - Address bus : whether memory or I/O port address
- WR: write signal(high impedance state during hold ack.)

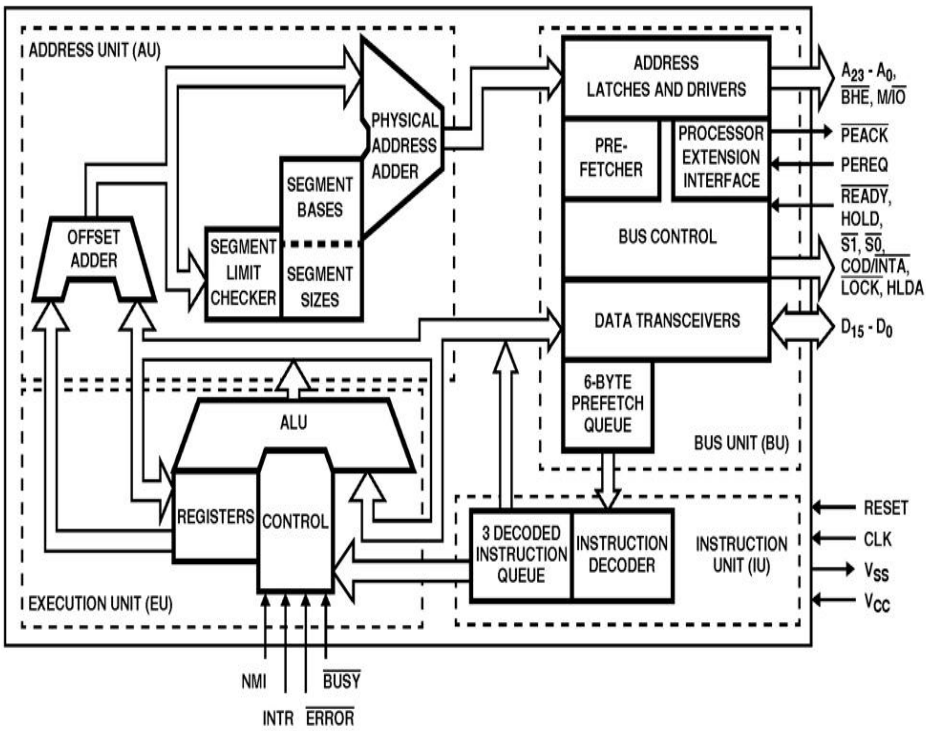
  •strobe that indicate that output data to memory or I/O during WR'=0 : data bus contains valid data for M or I/O
- RD': Control signal read operation (it is an active low signal)
  - Depends upon the status of S2 pin it will read the memory of I/O or memory.

UNIT II

ADVANCED PROCESSOR

Review of PROCESSOR and its types-80286-80386-80486-Introduction to Pentium family-MMX architecture and instruction set- core 2duo and core 2 quad PROCESSOR.

1. ARCHITECTURE OF 80286



Features of 80286

- The 80286 is the first family of advanced Micro processor with memory management and protection mode.
- It is 16 bit PROCESSOR with 134000 transistors.
- The CPU, with its 24-bit address bus is able to address up to 16 Mbytes of physical memory.
- It has 12.5 MHz, 10 MHz and 8 MHz clock frequencies.

- Average speed is 0.21 instructions per clock cycle.
- It compatible with 8086 in terms of instruction set.
- It has two operating modes namely real address mode and virtual address mode.
- In real address mode, the 80286 can address up to 1Mb of physical memory address.
- In virtual address mode, it can address up to 16 Mb of physical memory address space and 1 GB of virtual memory address space.
- The instruction set of 80286 includes the instructions of 8086 and 80186.
- It has some extra instructions to support operating system and memory management.
- In real address mode, the 80286 is object code compatible with 8086.
- In protected virtual address mode, it is source code compatible with 8086.
- It is five times faster than the standard 8086.

Register of 80286

The 80286 contains almost the same set of registers, as in 8086, namely

1. Eight 16-bit general purpose registers
2. Four 16-bit segment registers
3. Status and control registers
4. Instruction Pointer

Functional blocks

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

Address Unit:

- The address is responsible for calculating the physical address (PA or MA) of instructions and data accessed. PA$\rightarrow$29 bit, off set $\rightarrow$ 16 bit.
- The address lines derived by this unit may be used to address different peripherals.
- The PA handed over to the bus unit (BU) of the CPU.

Bus Unit:

- It has 16-bit data bus, 24 bit address bus and control bus. It performs all external operation.
- Latches and drivers of address bus transmit address from A19-A0 for read and write operation.
- It will fetch instruction bytes from the memory.
- Instructions are fetched in advance and stored in a queue to enable faster execution of the instructions. This concept is called instruction pipelining.
- It controls the prefetcher module. These prefetched instructions are arranged in a 6-byte instructions queue.

Instruction Unit:

- The 6-byte pre-fetch queue forwards the instructions arranged in it to the instruction unit (IU).
- The instruction unit accepts instructions from the prefetch queue and an instruction decoder decodes them one by one.
- The output of the decoding circuit drives a control circuit in the execution unit, which is responsible for executing the instructions received from decoded instruction queue.

Execution Unit:

- The decoded instruction queue sends the data part of the instruction over the data bus.
- The EU contains the register bank used for storing the data as scratch pad, or used as special purpose registers.
- The ALU, the heart of the EU, carries out all the arithmetic and logical operations and sends the results over the data bus or back to the register bank.

Register:

The architecture has fifteen registers. It is grouped in to four categories are below.

1. General purpose registers
2. Segment registers
3. Base and Index registers
4. Status and control registers

General purpose Registers:

It is used to store arithmetic and logical operations. They are AX, BX, CX , DX can be used either as 16 bit words or split in to pair of 8 bit registers.

Segment Registers:

It is used to select the segment of memory that is immediately addressable for code, stack and data.

Base and Index Registers:

General purpose register can also used to determine offset address of operand in memory. It holds base address or indexes to particular location within a segment.

Status and control registers:

It is 16 bit special purpose register are used of record and control of the 80286 PROCESSOR. The instruction pointer contains the offset address of the next sequential instruction to be executed.

Flag Word Description:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	NT			OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

D32	D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19		D17	D16
X	X	X	X	X	X	X	X	X	X	X	X	X	TS	EM	MP	PE

Status Flag:

1. Carry Flag (CF)^D0 set on carry/borrow produced ,when adding/subtracting bits.
2. Parity Flag (PF)^D2 set on even number of ones.
3. Auxiliary carry Flag (AF)^D4 set on carry or borrow to LSB to MSB otherwise cleared.
4. Zero Flag (ZF)^Set on D6 is zero
5. Sign Flag (SF)^Set on D7 is one
6. Overflow Flag (OF)^Too large or small numbers.

Control Flag:

1. Trap Flag (TF)^Set, a single interrupts occurs after the next instruction executes.TF is cleared by the single step interrupt.

2. Interrupt Flag (IF)^Set, Maskable interrupt will cause the CPU to transfer control to an interrupt vector specified location.
3. Direction Flag (DF)^Causes string instruction to auto decrement the appropriate index registers when set. Clearing DF causes auto increment.

Machine Status Word:

1. Task switch^ TS is set, This flag indicates the next instruction using extension will generate exception7, permitting the emulation of PROCESSOR extension is for the current task.
2. PROCESSOR Extension (EM)^The emulate PROCESSOR extension flag causes a PROCESSOR extension absent exception and permits the emulation of PROCESSOR extension by CPU.
3. Monitor PROCESSOR Extension (MP)^The monitor process extension flag allows WAIT instruction to generate a PROCESSOR extension not present in the extension.
4. Protection enable (PE) ^Protection enable flag places the 80286 in protected mode, PE is set. This can only be cleared by resetting the CPU.

Other Flag:

1. Nested Flag
2. I/O privilege level

Interrupt:

The Interrupts of 80286 may be divided into three categories,

1. External or hardware interrupts
2. INT instruction or software interrupts
3. Interrupts generated internally by exceptions

While executing an instruction, the CPU may sometimes be confronted with a special situation because of which further execution is not permitted. While trying to execute a divide by zero instruction, the CPU detects a major error and stops further execution. In this case, we say that an exception has been generated. In other words, an instruction exception is an unusual situation encountered during execution of an instruction that stops further execution. The return address from an exception, in most of the cases, points to the instruction that caused the exception. As in the case of 8086, the interrupt vector table of 80286 requires 1Kbytes of space for storing 256, four-byte pointers to point to the corresponding 256 interrupt service routines (lSR). Each pointer contains a 16-bit offset followed by a 16-bit segment selector to point to a particular ISR. The calculation of vector pointer address in the interrupt vector table from the (8-bit) INT type is exactly similar to 8086. Like 8086, the 80286 supports the software interrupts of type 0 (INT 00) to type FFH (INT FFH).

Maskable Interrupt INTR :

This is a maskable interrupt input pin of which the INT type is to be provided by an external circuit like an interrupt controller. The other functional details of this interrupt pin are exactly similar to the INTR input of 8086.

Non-Maskable Interrupt NMI :

It has higher priority than the INTR interrupt. Whenever this interrupt is received, a vector value of 02 is supplied internally to calculate the pointer to the interrupt vector table. Once the CPU responds to a NMI request, it does

not serve any other interrupt request (including NMI). Further it does not serve the PROCESSOR extension (COPROCESSOR) segment overrun interrupt, till either it executes IRET or it is reset. To start with, this clears the IF flag which is set again with the execution of IRET, i.e. return from interrupt.

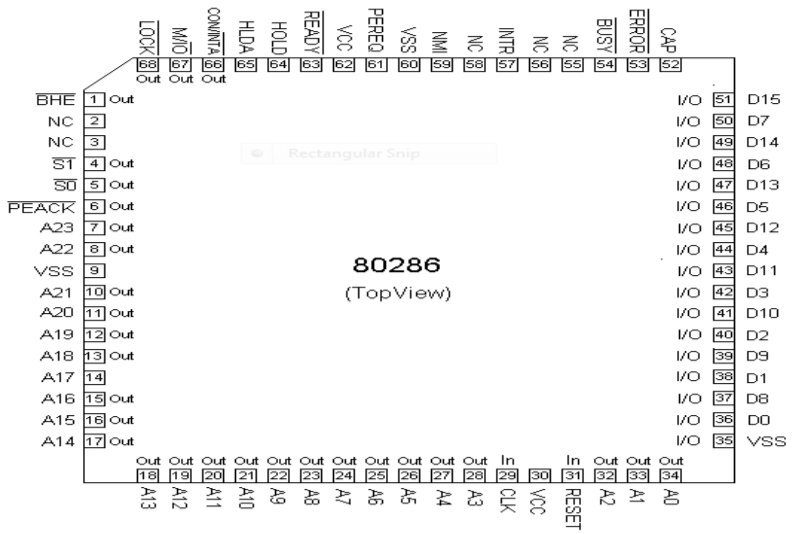
Single Step Interrupt

As in 8086, this is an internal interrupt that comes into action, if trap flag (TF) of 80286 is set. The CPU stops the execution after each instruction cycle so that the register contents (including flag register), the program status word and memory, etc. may be examined at the end of each instruction execution. This interrupt is useful for troubleshooting the software. An interrupt vector type 01 is reserved for this interrupt.

CLK:

This is the system clock input pin. The clock frequency applied at this pin is divided by two internally and is used for deriving fundamental timings for basic operations of the circuit. The clock is generated using 8284 clock generator.

Signal Description of 80286



D15-D0 :

These are sixteen bidirectional data bus lines.

A23-A0 :

These are the physical address output lines used to address memory or I/O devices. The address lines A23 - A16 are zero during I/O transfers

BHE :

This output signal, as in 8086, indicates that there is a transfer on the higher byte of the data bus (D15 - D8) .

SISO :

These are the active-low status output signals which indicate initiation of a bus cycle and with M/IO and COD/INTA, they define the type of the bus cycle.

M/$\overline{IO}$:

This output line differentiates memory operations from I/O operations. If this signal is it “0” indicates that an I/O cycle or INTA cycle is in process and if it is “1” it indicates that a memory or a HALT cycle is in progress.

COD/lNTA :

This output signal, in combination with M/ IO signal and S1 , S0 distinguishes different memory, I/O and INTA cycles.

$\overline{LOCK}$

This active-low output pin is used to prevent the other masters from gaining the control of the bus for the current and the following bus cycles. This pin is activated by a "LOCK" instruction prefix, or automatically by

hardware during XCHG, interrupt acknowledge or descriptor table access

COD/INTA	*mo*	SI	so	Bus Cycle
l(High)	0	0	0	Will not occur
1	0	0	]	I/O read
1	0	1	0	I/O write
1	0	1	]	None; not a status cycle
1	1	0	0	Will not occur
1	1	0	]	Will not occur Memory instruction read
1	1	1	0	Will not occur
1	1	1	]	None; not a status cycle

READY

This active-low input pin is used to insert wait states in a bus cycle, for interfacing low speed peripherals. This signal is neglected during HLDA cycle.

HOLD and HLDA

This pair of pins is used by external bus masters to request for the control of the system bus (HOLD) and to check whether the main PROCESSOR has

granted the control (HLDA) or not, in the same way as it was in 8086.

INTR :

Through this active high input, an external device requests 80286 to suspend the current instruction execution and serve the interrupt request. Its function is exactly similar to that of INTR pin of 8086.

NMI :

The Non-Maskable Interrupt request is an active-high, edge-triggered input that is equivalent to an INTR signal of type 2. No acknowledge cycles are needed to be carried out.

PEREG and PEACK (PROCESSOR Extension Request and Acknowledgement)

PROCESSOR extension refers to coPROCESSOR (80287 in case of 80286 CPU). This pair of pins extends the memory management and protection capabilities of 80286 to the PROCESSOR extension 80287. The PEREQ input requests the 80286 to perform a data operand transfer for a PROCESSOR extension. The PEACK active-low output indicates to the PROCESSOR extension that the requested operand is being transferred.

BUSY and **ERROR:**

PROCESSOR extension both are active-low input signals indicate the operating conditions of a PROCESSOR extension to 80286. The BUSY goes low, indicating 80286 to suspend the execution and wait until the BUSY become inactive. In this duration, the PROCESSOR extension is busy with its allotted job. Once the job is completed the PROCESSOR extension drives the BUSY input high indicating 80286 to continue with the program

execution. An active ERROR signal causes the 80286 to perform the PROCESSOR extension interrupt while executing the WAIT and ESC instructions. The active ERRORsignal indicates to 80286 that the PROCESSOR extension has committed a mistake and hence it is reactivating the PROCESSOR extension.

CAP :

A 0.047 pf, 12V capacitor must be connected between this input pin and ground to filter the output of the internal substrate bias generator. For correct operation of 80286 the capacitor must be charged to its operating voltage. Till this capacitor charges to its full capacity, the 80286 may be kept stuck to reset to avoid any spurious activity.

Vss :

This pin is a system ground pin of 80286.

Vcc :

This pin is used to apply +5V power supply voltage to the internal circuit of 80286. RESET the active-high RESET input clears the internal logic of 80286, and reinitializes it

Reset

The active-high reset input pulse width should be at least 16 clock cycles. The 80286 requires at least 38 clock cycles after the trailing edge of the RESET input signal, before it makes the first opcode fetch cycle.

Real Address Mode

It act as a fast 8086.Instruction set is upwardly compatible. It address only 1 M byte of physical memory using A0-A19.In real addressing mode of operation of 80286, it just acts as a fast 8086. The instruction set is upward compatible with that of 8086.The 80286 addresses only IMbytes of physical memory using A0- A19. The lines A20-A23 are not used by the internal circuit of 80286 in this mode. In real address mode, while addressing the physical memory, the 80286 uses BHE along with A0- A19. The 20-bit physical address is again formed in the same way as that in 8086. The contents of segment registers are used as segment base addresses. The other registers, depending upon the addressing mode, contain the offset addresses. Because of extra pipelining and other circuit level improvements, in real address mode also, the 80286 operates at a much faster rate than 8086, although functionally they work in an identical fashion. As in 8086, the physical memory is organized in terms of segments of 64Kbyte.An exception is generated, if the segment size limit is exceeded by the instruction or the data. The overlapping of physical memory segments is allowed to minimize the memory requirements for a task. The 80286 reserves two fixed areas of physical memory for system initialization and interrupt vector table. In the real mode the first 1Kbyte of memory starting from address 0000H to 003FFH is reserved for interrupt vector table. Also the addresses from FFFF0H to FFFFFH are reserved for system initialization. The program execution starts from FFFFH after reset and initialization. The interrupt vector table of 80286 is organized in the same way as that of 8086. Some of the interrupt types are reserved for exceptions, single-stepping and PROCESSOR extension segment. When the 80286 is reset, it always starts the execution in real address mode. In real address mode, it performs the

following functions: it initializes the IP and other registers of 80286, it prepares for entering the protected virtual address mode.

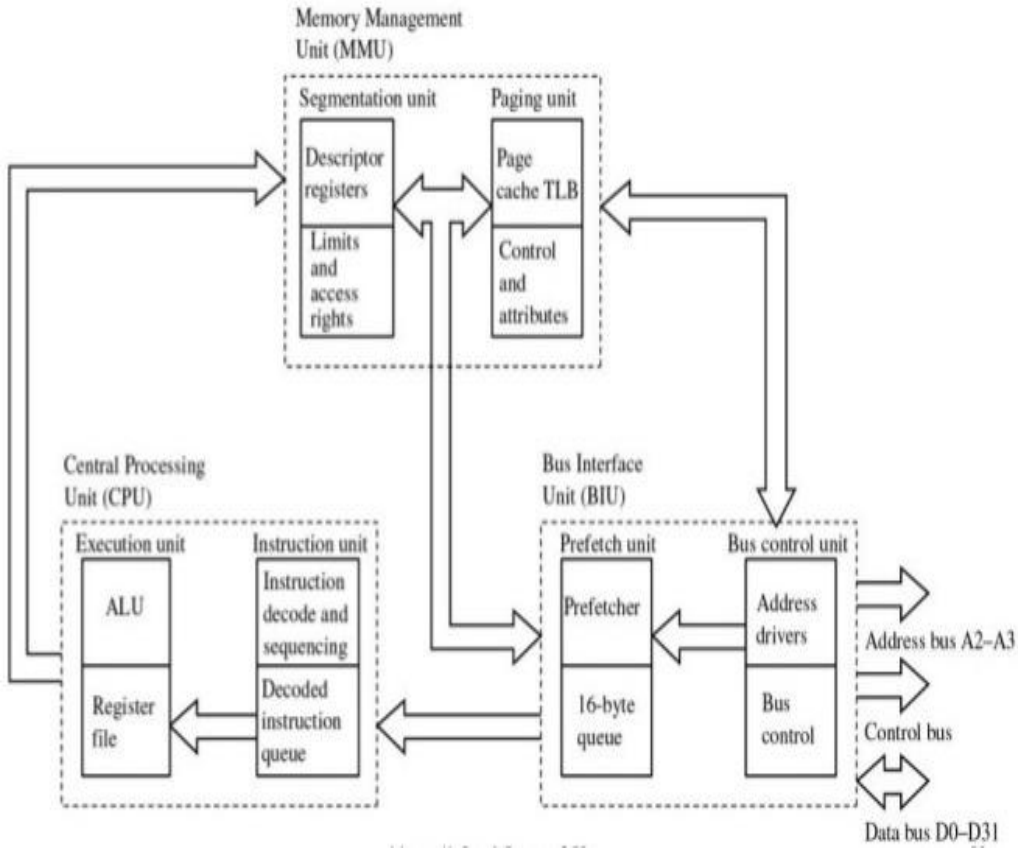
Protected virtual address mode (PVAM):

80286 is the first PROCESSOR to support the concepts of virtual memory and memory management. The virtual memory does not exist physically it still appears to be available within the system. The concept of VM is implemented using Physical memory that the CPU can directly access and secondary memory that is used as storage for data and program, which are stored in secondary memory initially. The Segment of the program or data required for actual execution at that instant, is fetched from the secondary memory into physical memory. After the execution of this fetched segment, the next segment required for further execution is again fetched from the secondary memory, while the results of the executed segment are stored back into the secondary memory for further references. This continues till the complete program is executed. During the execution the partial results of the previously executed portions are again fetched into the physical memory, if required for further execution. The procedure of fetching the chosen program segments or data from the secondary storage into physical memory is called swapping. The procedure of storing back the partial results or data back on the secondary storage is called un swapping. The virtual memory is allotted per task. The 80286 is able to address 1 G byte of virtual memory per task. The complete virtual memory is mapped on to the 16Mbyte physical memory.

If a program is larger than 16Mbyte is stored on the hard disk and is to be executed, if it is fetched in terms of data or program segments of less than 16Mbyte in size into the program memory by swapping sequentially as per sequence of execution. Whenever the portion of a program is required for

execution by the CPU, it is fetched from the secondary memory and placed in the physical memory is called swapping in of the program. A portion of the program or important partial results required for further execution , may be saved back on secondary storage to make the PM free for further execution of another required portion of the program is called 80286 uses the 16-bit content of a segment register as a selector to address a descriptor stored in the physical memory. The descriptor is a block of contiguous memory locations containing information of a segment, like segment base address, segment limit, segment type, privilege level, segment availability in physical memory, descriptor type and segment use another task.

2. ARCHITECTURE OF 80386



Features of 80386:

- This 80386 is a 32bit PROCESSOR that supports 8bit/32bit data operands.
- The 80386 instruction set is upward compatible with all its predecessors.
- The 80386 can run 8086 applications under protected mode in its virtual 8086 mode of operation.
- With the 32 bit address bus, the 80386 can address up to 4Gbytes of physical memory. The physical memory is organized in terms of segments of 4Gbytes at maximum.
- The 80386 CPU supports 16K number of segments and thus the total virtual space of 4Gbytes * 16K = 64 Terabytes.
- The memory management section of 80386 supports the virtual memory, paging and four levels of protection, maintaining full compatibility with 80286.
- The 80386 offers a set of 8 debug registers DR 0-DR 7 for hardware debugging and control. The 80386 has on-chip address translation cache.
- The concept of paging is introduced in 80386 that enables it to organize the available physical memory in terms of pages of size 4Kbytes each, under the segmented memory.
- The 80386 can be supported by 80387 for mathematical data processing.

Architecture:

The Internal Architecture of 80386 is divided into 3 sections.

- Central processing unit -Two types
  => Execution unit and Instruction unit
- Memory management unit
- Bus interface unit

Execution unit :

It has 8 General purpose and 8 Special purpose registers which are either used for handling data or calculating offset addresses.

Instruction unit:

It 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.

Memory management unit :

The Memory management unit consists of

- Segmentation unit
- Paging unit.

Segmentation unit:

It allows the use of two address components, viz. segment and offset for relocability and sharing of code and data. The segments of size 4Gbytes. The Segmentation unit provides a 4 level protection mechanism for protecting and isolating the system code and data from those of the application program

The Paging unit :

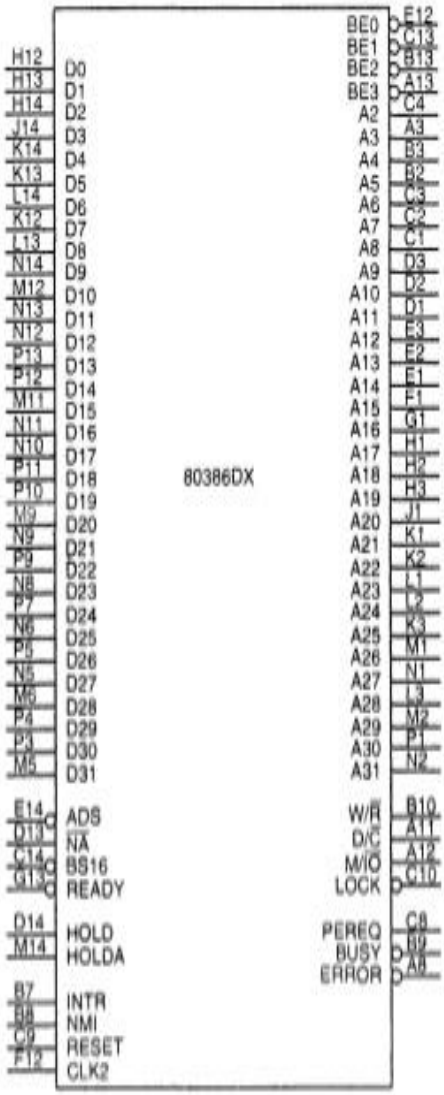
It organizes the physical memory in terms of pages of 4kbytes size each. It works under

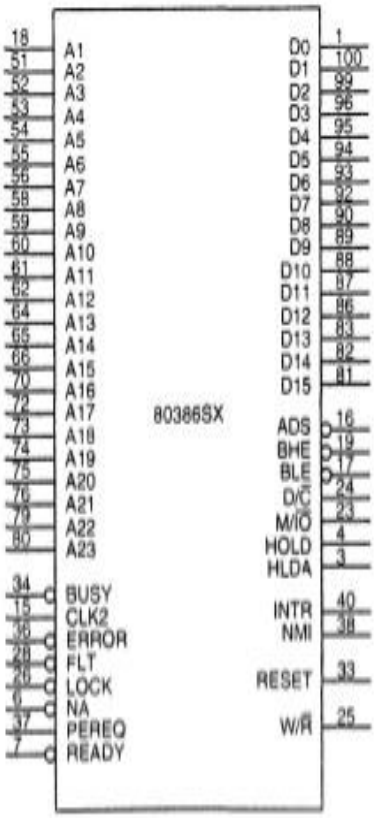
the control of the segmentation unit, i.e. each segment is further divided into pages. The virtual memory is also organizes in terms of segments and pages by the memory management unit. It converts linear addresses into physical addresses. The control and attribute PLA checks the privileges at the page 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.

Bus control unit

It has a prioritize to resolve the priority of the various bus requests. which 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:


80386DX
H12 D0
H13 D1
H14 D2
J14 D3
K14 D4
K13 D5
L14 D6
K12 D7
L13 D8
N14 D9
M12 D10
N13 D11
N12 D12
P13 D13
P12 D14
M11 D15
N11 D16
N10 D17
P11 D18
P10 D19
M9 D20
N9 D21
P9 D22
N8 D23
P7 D24
N6 D25
P5 D26
N5 D27
M6 D28
P4 D29
P3 D30
M5 D31
E14 ADS
D13 NA
C14 BS16
G13 READY
D14 HOLD
M14 HOLDA
B7 INTR
B8 NMI
C9 RESET
F13 CLK2
BE0 E12
BE1 C13
BE2 B13
BE3 A13
A2 C4
A3 A3
A4 B3
A5 B2
A6 C3
A7 C2
A8 C1
A9 D3
A10 D2
A11 D1
A12 E3
A13 E2
A14 E1
A15 F1
A16 G1
A17 H1
A18 H2
A19 H3
A20 J1
A21 K1
A22 K2
A23 L1
A24 L2
A25 K3
A26 M1
A27 N1
A28 L3
A29 M2
A30 P1
A31 N2
W/R B10
D/C A11
M/IO A12
LOCK C10
PEREQ C8
BUSY B9
ERROR A9


80386SX
18 A1
51 A2
52 A3
53 A4
54 A5
55 A6
56 A7
58 A8
59 A9
60 A10
61 A11
62 A12
64 A13
65 A14
66 A15
70 A16
72 A17
73 A18
74 A19
75 A20
76 A21
79 A22
80 A23
34 BUSY
15 CLK2
36 ERROR
28 FLT
35 LOCK
6 NA
37 PEREQ
7 READY
D0 1
D1 100
D2 99
D3 96
D4 95
D5 94
D6 93
D7 92
D8 90
D9 89
D10 88
D11 87
D12 86
D13 83
D14 82
D15 81
ADS 16
BHE 19
BLE 17
D/C 24
M/IO 23
HOLD 4
HLDA 3
INTR 40
NMI 38
RESET 33
W/R 25

Pin Description:

CLK2: The input pin provides the basic system clock timing for the operation of 80386.

D 0 - D31: These 32 lines act as bidirectional data bus during different access cycles ***A31 - A 2:*** These are upper 30 bit of the 32- bit address bus.

BE0 to BE 3: The 32- bit data bus supported by 80386 and the memory system of 80386 can be viewed as a 4- byte wide memory access mechanism. The 4 byte enable lines BE 0 to BE 3, may be used for enabling these 4 blanks. Using these 4 enable signal lines, the CPU may transfer 1 byte / 2 / 3 / 4 byte of data simultaneously.

W/R#: The write / read output distinguish 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#, BE 0# to BE 3#) 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 indicate 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 g 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 Organization:

The 80386 has eight 32 - bit general purpose registers which may be used as either 8 bit or 16 bit registers. 32 - bit register known as an extended register, is represented by the register name with prefix E. Example : A 32 bit register corresponding to AX is EAX, similarly BX is EBX etc. The 16 bit registers BP, SP, SI and DI in 8086 are now available with their extended size of 32 bit and are names as EBP, ESP, ESI and EDI. AX represents the lower 16 bit of the 32 bit register EAX. BP, SP, SI, DI represents the lower 16 bit of their 32 bit counterparts, and can be used as independent 16 bit registers. The six segment registers available in 80386 are CS, SS, DS, ES, FS and GS. The CS and SS are the code and the stack segment registers respectively, while DS, ES, FS, GS are 4 data segment registers. The 16 bit instruction pointer IP is available along with 32 bit counterpart EIP.

Flag Register of 80386: It is a 32 bit register. Out of the 32 bits, Intel has reserved bits D18 to D31, D 5 and D 3, while D1 is always set at 1.Two extra new flags are added to the 80286 flag to derive the flag register of 80386. They are VM and RF flags.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	NT	IOPL	IOPL	OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

D31-D17 reserved for Intel

D3	D3	D2	D2	D2	D2	D2	D2	D2	D2	D21	D20	D1	D1	D1	D16	D15
															VM	RF

VM - Virtual Mode Flag:

If this flag is set, the 80386 enters the virtual 8086 mode within the protection mode. This is to be set only when the 80386 is in protected mode. In this mode, if any privileged instruction is executed an exception 13 is generated. This bit can be set using IRET instruction or any task switch operation only in the protected mode.

RF- Resume Flag:

This flag is used with the debug register breakpoints. It is checked at the starting of every instruction cycle and if it is set, any debug fault is ignored during the instruction cycle. The RF is automatically reset after successful execution of every instruction, except for IRET and POPF instructions. Also, it is not automatically cleared after the successful execution of JMP, CALL and INT instruction causing a task switch. These instructions are used to set the RF to the value specified by the memory data available at the stack.

Segment Descriptor Registers:

This registers are not available for programmers, rather they are internally used to store the descriptor information, like attributes, limit and base addresses of segments. The six segment registers have corresponding six 73 bit descriptor registers. Each of them contains 32 bit base address, 32 bit base limit and 9 bit attributes. These are automatically loaded when the corresponding segments are loaded with selectors.

Control Registers:

The 80386 has three 32 bit control registers CR), CR 2 and CR 3 to hold global machine status independent of the executed task. Load and store instructions are available to access these registers. System Address Registers: Four special registers are defined to refer to the descriptor tables supported by 80386. The 80386 supports four types of descriptor table, viz. global descriptor table (GDT), interrupt descriptor table (IDT), local descriptor table (LDT) and task state segment descriptor (TSS).

Debug and Test Registers: Intel has provide a set of 8 debug registers for hardware debugging. Out of these eight registers DR 0 to DR 7, two registers DR 4 and DR 5 are Intel reserved. The initial four registers DR 0 to DR 3 store four program controllable breakpoint addresses, while DR 6 and DR 7 respectively hold breakpoint status and breakpoint control information. Two more test register are provided by 80386 for page caching namely test control and test status register.

ADDRESSING MODES:

The 80386 supports overall eleven addressing modes to facilitate efficient execution of higher level language programs. In case of all those modes, the 80386 can now have 32-bit immediate or 32- bit register operands or displacements. The 80386 has a family of scaled modes. In case of scaled modes, any of the index register values can be multiplied by a valid scale factor to obtain the displacement. The valid scale factor are 1, 2, 4 and 8.

The different scaled modes are as follows. Scaled Indexed Mode: Contents of the index register are multiplied by a scale factor that may be added further to get the operand offset.

Based Scaled Indexed Mode:

Contents of the index register are multiplied by a scale factor and then added to base register to obtain the offset. Based Scaled Indexed Mode with Displacement: The Contents of the index register are multiplied by a scaling factor and the result is added to a base register and a displacement to get the offset of an operand.After reset, the 80386 starts from memory location FFFFFFF0H under the real address mode. In the real mode, 80386 works as a fast 8086 with 32- bit registers and data types. In real mode, the default operand size is 16 bit but 32- bit operands and addressing modes may be used with the help of override prefixes. The segment size in real mode is 64k, hence the 32-bit effective addressing must be less than 0000FFFFFH. The real mode initializes the

80386 and prepares it for protected mode.

Memory Addressing in Real Mode:

In the real mode, the 80386 can address at the most 1Mbytes of physical memory using address lines A 0-A19. Paging unit is disabled in real addressing mode, and hence the real addresses are the same as the physical addresses. To form a physical memory address, appropriate segment registers contents (16-bits) are shifted left by four positions and then added to the 16-bit offset address formed using one of the addressing modes, in the same way as in the 80386 real address modes. The segment in 80386 real mode can be read, write or executed, i.e. no protection is available.Any fetch or access past the end of the segment limit generates exception 13 in real address mode. The segments in 80386 real mode may be overlapped or no overlapped. The interrupt vector table of 80386 has been allocated 1Kbyte space starting from 00000H to 003FFH.

Protected Mode of 80386

All the capabilities of 80386 are available for utilization in its protected mode of operation. The 80386 in protected mode support all the software written for 80286 and 8086 to be executed under the control of memory management and protection abilities of 80386. The protected mode allows the use of additional instruction, addressing modes and capabilities of 80386.

Addressing in protected mode:

In this mode, the contents of segment registers are used as selectors to address descriptors which contain the segment limit, base address and access rights byte of the segment. The effective address (offset) is added with segment base address to calculate linear address. This linear address is further used as physical address, if the paging unit is disabled. Otherwise the paging unit converts the linear address into physical address. The paging unit is a memory management unit enabled only in protected mode. The paging mechanism allows handling of large segments of memory in terms of pages of

4Kbyte size. The paging unit operates under the control of segmentation unit. The paging unit if enabled converts linear addresses into physical address, in protected mode.

Segmentation

Descriptor tables: These descriptor tables and registers are manipulated by the operating system to ensure the correct operation of the PROCESSOR, and hence the correct execution of the program. Three types of the 80386 descriptor tables are listed as follows:

- Global Descriptor Table (GDT)
- Local Descriptor Table (LDT)
- Interrupt Descriptor Table (IDT)

Descriptors:

The 80386 descriptors have a 20-bit segment limit and 32-bit segment address. The descriptor of 80386 are 8-byte quantities access right or attribute bits along with the base and limit of the segments. Descriptor Attribute Bits: The A (accessed) attributed bit indicates whether the segment has been accessed by the CPU or not. The TYPE field decides the descriptor type and hence the segment type. The S bit decides whether it is a system descriptor (S=0) or code/data segment descriptor (S=1).The DPL field specifies the descriptor privilege level. The D bit specifies the code segment operation size. If D=1, the segment is a 32-bit operand segment, else, it is a 16-bit operand segment. The P bit (present) signifies whether the segment is present in the physical memory or not. If P=1, the segment is present in the physical memory. The G (granularity) bit indicates whether the segment is page addressable. The zero bit must remain zero for compatibility with future process.

The AVL (available) field specifies whether the descriptor is for user or for operating system. • The 80386 has five types of descriptors listed as follows:

1. Code or Data Segment Descriptors.
2. System Descriptors.
3. Local descriptors.
4. TSS (Task State Segment) Descriptors.
5. GATE Descriptors.

The 80386 provides a four level protection mechanism exactly in the same way as the 80286 does.

Paging operation:

Paging is one of the memory management techniques used for virtual memory multitasking operating system. The segmentation scheme may divide the physical memory into a variable size segments but the paging divides the memory into a fixed size pages. The segments are supposed to be the logical segments of the program, but the pages do not have any logical relation with the program. The pages are just fixed size portions of the program module or data. The advantage of paging scheme is that the complete segment of a task need not be in the physical memory at any time. Only a few pages of the segments, which are required currently for the execution, need to be available in the physical memory. Thus the memory requirement of the task is substantially reduced, relinquishing the available memory for other tasks. Whenever the other pages of task are required for execution, they may be fetched from the secondary storage. The previous pages which are executed need not be available in the memory, and hence the space occupied by them may be relinquished for other tasks.

Paging Descriptor Base Register:

The control register CR2 is used to store the 32-bit linear address at which the previous page fault was detected. The CR 3 is used as page directory physical base address register, to store the physical starting address of the page directory. The lower 12 bit of the CR3 are always zero to ensure the page size aligned directory. A move operation to CR 3 automatically loads the page table entry caches and a task switch operation, to load

CR 0 suitably.

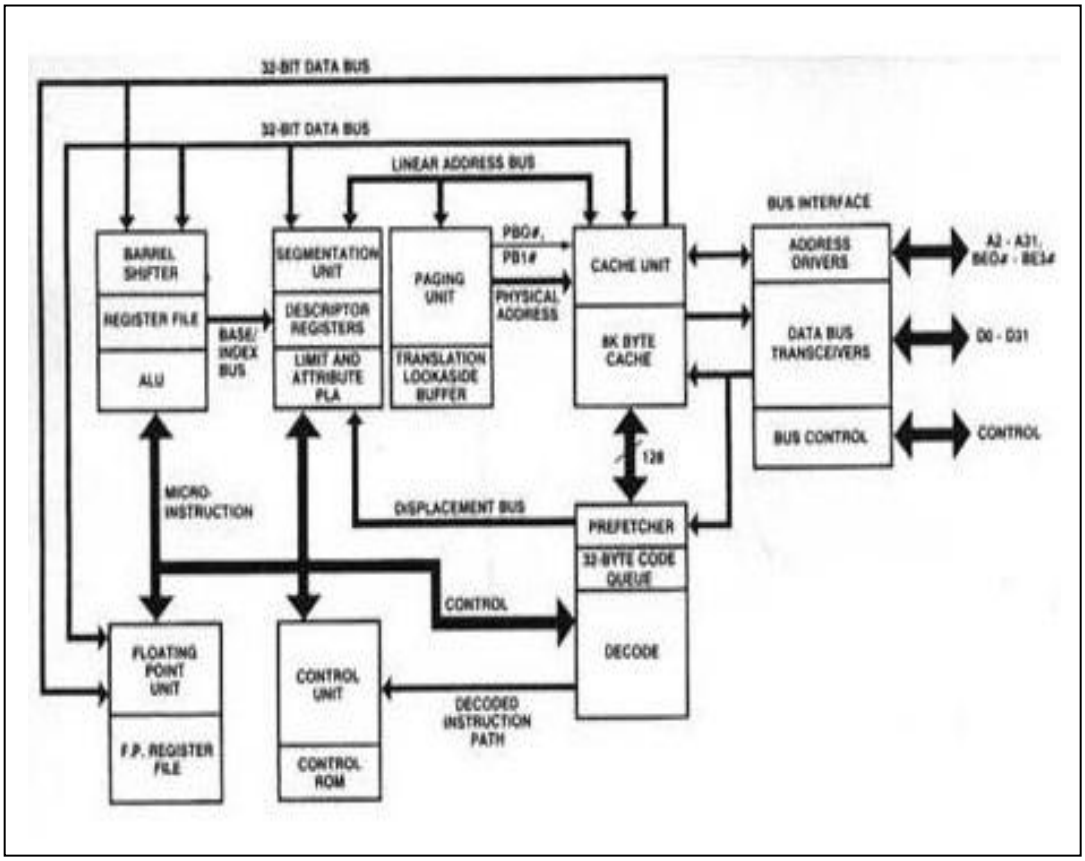
Page Directory:

This is at the most 4Kbytes in size. Each directory entry is of 4 bytes, thus a total of 1024 entries are allowed in a directory. The upper 10 bits of the linear address are used as an index to the corresponding page directory entry. The page directory entries point to page tables.

Page Tables:

Each page table is of 4Kbytes in size and many contain a maximum of 1024 entries. The page table entries contain the starting address of the page and the statistical information about the page. The upper 20 bit page frame address is combined with the lower 12 bit of the linear address. The address bits A12- A21 are used to select the 1024 page table entries. The page table can be shared between the tasks. The P bit of the above entries indicates, if the entry can be used in address translation. If P=1, the entry can be used in address translation, otherwise it cannot be used. The P bit of the currently executed page is always high. The accessed bit A is set by 80386 before any access to the page. If A=1, the page is accessed, else un accessed. The D bit is set before a write operation to the page is carried out. The D-bit is undefined for page director entries. The OS reserved bits are defined by the operating system software. The User / Supervisor (U/S) bit and read/write bit are used to provide protection. These bits are decoded to provide protection under the 4 level protection models. The level 0 is supposed to have the highest privilege, while the level 3 is supposed to have the least privilege. This protection provide by the paging unit is transparent to the segmentation unit.

ARCHITECTURE OF 80486



It is 32 bit PROCESSOR. One of the most obvious feature included in a 80486 is a built in math coPROCESSOR. This coPROCESSOR is essentially the same as the 80387 PROCESSOR used with a 80386, but being integrated on the chip allows it to execute math instructions about three times as fast as a 80386/387 combination. 80486 is an 8Kbyte code and data cache. To make room for the additional signals, the 80486 is packaged in a 168 pin, pin grid array package instead of the 132 pin PGA used for the 80386.

Flag Register of 80486:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	NT	IOPL	IOPL	OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

31 18 17 16 Reserved for INTEL E F L A G Flags CF: Carry Flag

AF: Auxiliary carry

ZF: Zero Flag

SF : Sign Flag

TF : Trap Flag

IE : Interrupt Enable

DF : Direct Flag

OF : Over Flow

IOPL : I/O Privilege Level

NT : Nested Task Flag

RF : Resume Flag

VM : Virtual Mode AC :

Alignment Check

The memory system for the 486 is identical to 386 Micro processor. The 486 contains 4G bytes of memory beginning at location 00000000H and ending at FFFFFFFFH. The major change to the memory system is internal to 486 in the form of 8K byte cache memory, which speeds the execution of instructions and the acquisition of data. Another addition is the parity checker/ generator built into the 80486 Micro processor. Parity Checker / Generator : Parity is often used to determine if data are correctly read from a memory location. INTEL has incorporated an internal parity generator / decoder.

Parity is generated by the 80486 during each write cycle. Parity is generated as even parity and a parity bit is provided for each byte of memory. The parity check bits appear on pins DP0-DP3, which are also parity inputs as well as parity outputs. These are typically stored in memory during each write cycle and read from memory during each read cycle. On a read, the Micro processor checks parity and generates a parity check error, if it occurs on the PCHK# pin. A parity error causes no change in processing unless the user applies the PCHK signal to an interrupt input.

Interrupts are often used to signal a parity error in DS-based computer systems. This is same as 80386, except the parity bit storage. If parity is not used, Intel recommends that the DP0 - DP3 pins be pulled up to +5v.

Cache memory: The cache memory system stores data used by a program and also the instructions of the program. The cache is organized as a 4 way set associative cache with each location containing 16 bytes or 4 double words of data.

Control register CR0: It is used to control the cache with two new control bits not present in the 80386 Micro processor. The CD (cache disable) , NW (non-cache write through) bits are new to the 80486 and are used to control the 8K byte cache. If the CD

bit is a logic 1, all cache operations are inhibited. This setting is only used for debugging software and normally remains cleared. The NW bit is used to inhibit cache write through operation. As with CD, cache write through is inhibited only for testing. For normal operations CD = 0 and NW = 0. Because the cache is new to 80486 Micro processor and the cache is filled using burst cycle not present on the 386.

The 80486 contains the same memory-management system as the 80386. This includes a paging unit to allow any 4K byte block of physical memory to be assigned to any 4K byte block of linear memory. The only difference between 80386 and 80486 memory-management system is paging. The 80486 paging system can disabled caching for section of translation memory pages, while the 80386 could not. If these are compared with 80386 entries, the addition of two new control bits is observed (PWT and PCD). The page write through and page cache disable bits control caching.

The PWT controls how the cache functions for a write operation of the external cache memory. It does not control writing to the internal cache. The logic level of this bit is found on the PWT pin of the 80486 Micro processor. Externally, it can be used to dictate the write through policy of the external caching. The PCD bit controls the on-chip cache. If the PCD = 0, the on-chip cache is enabled for the current page of memory. Note that 80386 page table entries place a logic 0 in the PCD bit position, enabling caching. If PCD = 1, the on-chip cache is disable. Caching is disable regard less of condition of KEN#, CD, and NW.

Test registers : TR3,TR5 **Cache data register (TR3):**

It is used to access either the cache fill buffer for a write test operation or the cache read buffer for a cache read test operation. In order to fill or read a cache line (128 bits wide), TR3 must be written or read four times. The contents of the set select field in TR5 determine which internal cache line is written or read through TR3. The 7 bit test field selects one of the 128 different 16 byte wide cache lines. The entry select bits of TR5

select an entry in the set or the 32 bit location in the read buffer.

Control bits in TR5:

It is used to enable the

- Fill buffer or read buffer operation (00).
- Perform a cache write (01),
- Perform a cache read (10)
- Flush the cache (11).

The cache status register (TR4) hold the cache tag, LRU bits and a valid bit. This register is loaded with the tag and valid bit before a cache a cache write operation and contains the tag, valid bit, LRU bits, and 4 valid bits on a cache test read. Cache is tested each time that the Micro processor is reset if the AHOLD pin is high for 2 clocks prior to the RESET pin going low. This causes the 486 to completely test itself with a built in self test or BIST. The BIST uses TR3, TR4, TR5 to completely test the internal cache. Its outcome is reported in register EAX. If EAX is a zero, the Micro processor, the COPROCESSOR and cache have passed the self test. The value of EAX can be tested after reset to determine if an error is detected. In most of the cases we do not directly access the test register unless we wish to perform our own tests on the cache or TLB.

Pin Definitions :

A 31-A2 : Address outputs A31-A2 provide the memory and I/O with the address during normal operation. During a cache line invalidation A31-A4 are used to drive the Micro processor.

A20 : The address bit 20 mask causes the 80486 to wrap its address around from location 000FFFFFH to 00000000H as in 8086. This provides a memory system that functions like the 1M byte real memory system in the 8086 PROCESSOR.

ADS: The address data strobe becomes logic zero to indicate that the address bus contains a valid memory address.

AHOLD: The address hold input causes the Micro processor to place its address bus connections at their high-impedance state, with the remainder of the buses staying active. It is often used by another bus master to gain access for a cache invalidation cycle.

BREQ: This bus request output indicates that the 486 has generated an internal bus request. BE 3-BE 0 : Byte enable outputs select a bank of the memory system when information is transferred between the Micro processor and its memory and I/O. The BE 3 signal enables D31 - D24 , BE2 enables D23-D16, BE1 enables D15 - D8 and BE 0 enables D 7-D 0

BLAST: The burst last output shows that the burst bus cycle is complete on the next activation of BRDY# signal

BOFF : The Back-off input causes the Micro processor to place its buses at their high impedance state during the next cycle. The Micro processor remains in the bus hold state until the BOFF# pin is placed at a logic 1 level.

NMI : The non-maskable interrupt input requests a type 2 interrupt.

BRDY : The burst ready input is used to signal the Micro processor that a burst cycle is complete.

KEN : The cache enable input causes the current bus to be stored in the internal.

LOCK : The lock output becomes a logic 0 for any instruction that is prefixed with the lock prefix.

W / R : current bus cycle is either a read or a write.

IGNNE : The ignore numeric error input causes the coPROCESSOR to ignore floating point error and to continue processing data. The signal does not affect the state of the FERR pin.

FLUSH : The cache flush input forces the Micro processor to erase the contents of its 8K byte internal cache

EADS: The external address strobe input is used with AHOLD to signal that an external address is used to perform a cache invalidation cycle.

FERR : The floating point error output indicates that the floating point coPROCESSOR has detected an error condition. It is used to maintain compatibility with DOS software.

BS 8 : The bus size 8, input causes the 80486 to structure itself with an 8-bit data bus to access byte-wide memory and I/O components.

BS16: The bus size 16, input causes the 80486 to structure itself with an 16-bit data bus to access word-wide memory and I/O components.

PCHK : The parity check output indicates that a parity error was detected during a read operation on the DP 3 - DP 0 pin.

PLOCK : The pseudo-lock output indicates that current operation requires more than one bus cycle to perform. This signal becomes a logic 0 for arithmetic coPROCESSOR operations that access 64 or 80 bit memory data.

PWT: The page write through output indicates the state of the PWT attribute bit in the page table entry or the page directory entry.

RDY : The ready input indicates that a non-burst bus cycle is complete. The RDY signal must be returned or the Micro processor places wait states into its timing until RDY is asserted.

M / IO : Memory / IO defines whether the address bus contains a memory address or an I/O port number. It is also combined with the W/ R signal to generate memory and I/O read and write control signals.

The 80486 data bus, address bus, byte enable, ADS#, RDY#, INTR, RESET, NMI, M/IO#, D/C#, W/R#, LOCK#, HOLD, HLDA and BS16# signals function as we described for 80386. • The 80486 requires 1 clock instead of 2 clock required by 80386. A new signal group on the 486 is the PARITY group DP 0-DP 3 and PCHK#. These signals allow the 80486 to implement parity detection / generation for memory reads and memory writes. During a memory write operation, the 80486 generates an even parity bit for each byte and outputs these bits on the DP 0-DP3 lines.

A normal 80486 memory read operation to read a line into the cache requires 2 clock cycles. However, if a series of reads is being done from successive memory locations, the reads can be done in burst mode with only 1 clock cycle per read. To start the process the 80486 sends out the first address and asserts the BLAST# signal high. When the external DRAM controller has the first data bus, it asserts the BRDY# signal. The 80486 reads the data word and outputs the next address. Since the data words are at successive addresses, only the lower address bits need to be changed. If the DRAM controller is operating in the page or the static column modes then it will only have to output a new column address to the DRAM.

In this mode the DRAM will be able to output the new data word within 1 clock cycle. When the PROCESSOR has read the required number of data words, it asserts the BLAST# signal low to terminate the burst mode. The final signal we want to discuss here are the bus request output signal BREQ, the back-off input signal BOFF#, the HOLD signal and the hold-acknowledge signal HLDA. These signals are used to control sharing the local 486 bus by multiple PROCESSOR (bus master). When a master on the bus need to use the bus, it asserts its BERQ signal .An external parity circuit will evaluate requests to use the bus and grant bus use to the highest - priority master. To ask the 486 to release the bus, the bus controller asserts the 486 HOLD input or BOFF# input. If the HOLD input is asserted, the 486 will finish the current bus cycle, float its buses and assert the HLDA signal. To prevent another master from taking over the bus

during a critical operation, the 486 can assert its LOCK# or PLOCK# signal.

The extended flag register EFLAG is illustrated in the figure. The only new flag bit is the AC alignment check, used to indicate that the Micro processor has accessed a word at an odd address or a double word boundary. Efficient software and execution require that data be stored at word or double word boundaries.

UNIT – III

MICROCONTROLLER (8051)

Introduction to microcontroller-Architecture-Addressing mode-instruction set-instruction execution- simple programs, Interfacing-seven segment LED and LCD, ADC/DAC-water level control.

1. Introduction to microcontroller

The microcontroller is a single chip Micro processor system which consists of CPU, data and program memory, serial and parallel I/O ports, timers and external as well as internal interrupts. Microcontrollers are used in Printers, Fax machine, Telephone, CD players, Process control, automation etc. There are few more advantages of built in peripherals have smaller access times hence speed is more. Hardware reduces due to single chip microcomputer system. Less hardware reduces PCB size and increases reliability of the system.

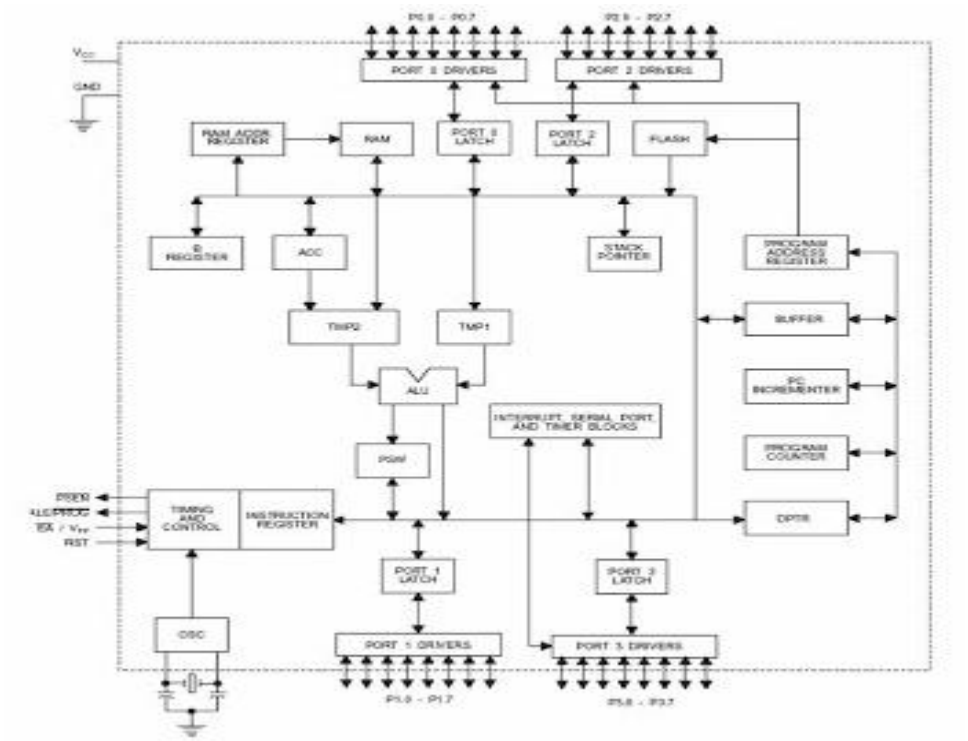
Features of 8051:

1. On chip data memory-128 bytes
2. On chip program memory-4096 bytes.
3. Four register banks.
4. 64 kilobytes each program and external RAM addressability.
5. 32 bidirectional I/O lines organized as four 8-bit ports.
6. 16 bit Timer and counter.
7. One microsecond instruction cycle with 12 MHZ crystal
8. Signed -overflow detection and parity calculation
9. Multiple and divide in four micro second in Hardware
10. Full depth stack for subroutine return linkage and data storage.

2. Difference between Micro processor and Microcontroller

Micro processor	Microcontroller
Micro processor is heart of Computer system.	Micro Controller is a heart of embedded system.
It is just a PROCESSOR. Memory and I/O components have to be connected externally	Micro controller has external PROCESSOR along with internal memory and i/O components
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present internally, the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique
Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the Micro processor do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instruction, hence speed is fast.
Micro processor have less number of registers, hence more operations are memory based.	Micro controller have more number of registers, hence the programs are easier to write.

Architecture of 8051:



ALU

All arithmetic and logical functions are carried out by the ALU. Addition, subtraction with carry, and multiplication come under arithmetic operations. Logical operations are AND, OR and exclusive OR (XOR) come under logical operations.

Program Counter (PC)

A program counter is a 16-bit register and it has no internal address. The basic function of program counter is to fetch from memory the address of the next instruction to be executed. The PC holds the address of the next instruction residing in memory and when a command is encountered, it produces that instruction. This way the PC increments automatically, holding the address of the next instruction.

Registers

Registers are usually known as data storage devices. 8051 microcontroller has 2 registers, namely Register A and Register B. Register A serves as an accumulator while Register B functions as a general purpose register. These registers are used to store the output of mathematical and logical instructions. The operations of addition, subtraction, multiplication and division are carried out by Register A. Register B is usually unused and comes into picture only when multiplication and division functions are carried out by Register A. Register A also involved in data transfers between the microcontroller and external memory.

Special Function Registers (SFRs)

1. Serial Port Data Buffer (SBUF)
2. Timer/Counter Control (TCON)
3. Timer/Counter Mode Control (TMOD)
4. Serial Port Control (SCON)
5. Power Control (PCON)
6. Interrupt Priority (IP)
7. Interrupt Enable Control (IE)

Timers and Counters

Synchronization among internal operations can be achieved with the help of clock circuits which are responsible for generating clock pulses. During each clock pulse a particular operation will be carried out, thereby, assuring synchronization among operations. For the formation of an oscillator, we are provided with two pins XTAL1 and XTAL2 which are used for connecting a resonant network in 8051 microcontroller device. In addition to this, circuit also consists of four more pins. They are,Internal operations can be synchronized using clock circuits which produce clock pulses. With each clock pulse, a particular function will be

accomplished and hence synchronization is achieved. There are two pins XTAL1 and XTAL2 which form an oscillator circuit which connect to a resonant network in the microcontroller.

The circuit also has 4 additional pins :

1. EA: External enable
2. ALE: Address latch enable
3. PSEN: Program store enable and
4. RST: Reset.

- Quartz crystal is used to generate periodic clock pulses.

Internal RAM and ROM

ROM

A code of 4K memory is incorporated as on-chip ROM in 8051. The 8051 ROM is a nonvolatile memory meaning that its contents cannot be altered and hence has a similar range of data and program memory, i.e, they can address program memory as well as a 64K separate block of data memory.

RAM

The 8051 microcontroller is composed of 128 bytes of internal RAM. This is a volatile memory since its contents will be lost if power is switched off. These 128 bytes of internal RAM are divided into 32 working registers which in turn constitute 4 register banks (Bank 0-Bank 3) with each bank consisting of 8 registers (R0 - R7). There are 128 addressable bits in the internal RAM.

The 8051 microcontroller has four 8-bit input/output ports. These are:

PORT P0: When there is no external memory present, this port acts as a general purpose input/output port. In the presence of external memory, it functions as a multiplexed address and data bus. It performs a dual role.

PORT P1: This port is used for various interfacing activities. This 8-bit port is a normal I/O port

i. e. it does not perform dual functions.

PORT P2: Similar to PORT P0, this port can be used as a general purpose port when there is no external memory but when external memory is present it works in conjunction with PORT PO as an address bus. This is an 8-bit port and performs dual functions.

PORT P3: PORT P3 behaves as a dedicated I/O port

Interrupt Control

An event which is used to suspend or halt the normal program execution for a temporary period of time in order to serve the request of another program or hardware device is called an interrupt. An interrupt can either be an internal or external event which suspends the microcontroller for a while and thereby obstructs the sequential flow of a program. There are two ways of giving interrupts to a microcontroller - one is by sending software instructions and the other is by sending hardware signals. The interrupt mechanism keeps the normal program execution in a "put on hold" mode and executes a subroutine program and after the subroutine is executed, it gets back to its normal program execution. This subroutine program is also called an interrupt handler. A subroutine is executed when a certain event occurs.

In 8051, 5 sources of interrupts are provided. They are:

- 2 external interrupt sources connected through INTO and INT1
- 3 external interrupt sources- serial port interrupt, Timer Flag 0 and Timer Flag 1.

The pins connected are as follows

1. ALE (Address Latch Enable) - Latches the address signals on Port P0
2. EA (External Address) - Holds the 4K bytes of program memory
3. PSEN (Program Store Enable) - Reads external program memory
4. RST (Reset) - Reset the ports and internal registers upon start up

Serial Data Communication

A method of establishing communication among computers is by transmitting and receiving data bits is a serial connection network. In 8051, the SBUF (Serial Port Data Buffer) register holds the data; the SCON (Serial Control) register manages the data communication and the PCON (Power Control) register manages the data transfer rates. Further, two pins - RXD and TXD, establish the serial network.The SBUF register has 2 parts - one for storing the data to be transmitted and another for receiving data from outer sources. The first function is done using TXD pin and the second function is done using RXD pin.

There are 4 programmable modes in serial data communication. They are:

1. Serial Data mode 0 (shift register mode)
2. Serial Data mode 1 (standard UART)
3. Serial Data mode 2 (multiPROCESSOR mode)
4. Serial Data mode 3

PSW (Program Status Word)

Program Status Word or PSW is a hardware register which is a memory location which holds a program's information and also monitors the status of the program this is currently being executed. PSW also has a pointer which points towards the address of the next instruction to be executed. PSW register has 3 fields namely are instruction address field, condition code field and error status field. We can say that PSW is an internal register that keeps track of the computer at every instant. Generally, the instruction of the result of a program is stored in a single bit register called a 'flag'. The are7 flags in the PSW of 8051. Among these 7 flags, 4 are math flags and 3 are general purpose or user flags.

CY	**AC**	**FO**	**RS1**	**RSO**	**OV**	**-**	**P**

The 4 Math flags are:

- Carry (c)
- Auxiliary carry (AC)
- Overflow (OV)
- Parity (P)

The 3 General purpose flags or User flags are:

- FO
- GFO
- GF 1

Data Pointer (DPTR)

The data pointer or DPTR is a 16-bit register. It is made up of two 8-bit registers called DPH and DPL. Separate addresses are assigned to each of DPH and DPL. These 8-bit registers are used for the storing the memory addresses that can be used to access internal and external data/code.

Stack Pointer (SP)

The stack pointer (SP) in 8051 is an 8-bit register. The main purpose of SP is to access the stack. As it has 8-bits it can take values in the range 00 H to FF H. Stack is a special area of data in memory. The SP acts as a pointer for an address that point to the top of the stack.

Data and Address Bus

A bus is group of wires using which data transfer takes place from one location to another within a system. Buses reduce the number of paths or cables needed to set up connection between components.

There are mainly two kinds of buses: -

- Data Bus
- Address

Bus Data Bus:

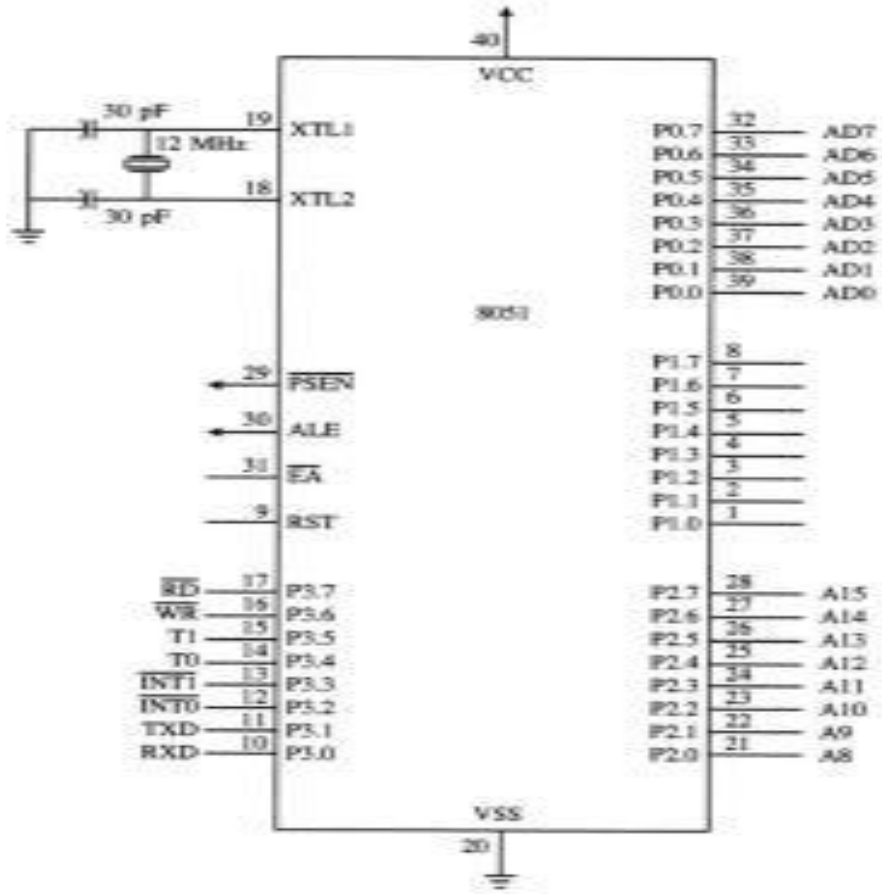
The purpose of data bus is to transfer data. It acts as an electronic channel using

which data travels. Wider the width of the bus, greater will be the transmission of data.

Address Bus:

The purpose of address bus is to transfer information but not data. The information tells from where within the components, the data should be sent to or received from. The capacity or memory of the address bus depends on the number of wires that transmit a single address bit.

Pin Diagram:



One of the most useful features of the 8051 is that it contains four I/O ports (P0 - P3)

Port 0 (***pins 32-39***): P0 (P0.0~P0.7)

8-bit R/W - General Purpose I/O

Or acts as a multiplexed low byte address and data bus for external memory design

Port 1 (***pins 1-8***): P1 (P1.0~P1.7) Only 8-bit R/W - General Purpose I

Port 2 (***pins 21-28***): P2 (P2.0~P2.7)

8-bit R/W - General Purpose I/O

Or high byte of the address bus for external memory design ***Port 3***

(***pins 10-17***): P3 (P3.0~P3.7)

General Purpose I/O

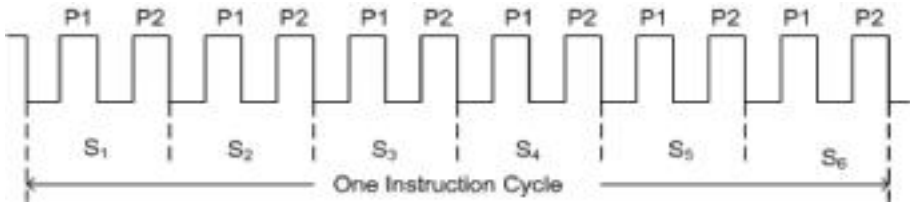
if not using any of the internal peripherals (timers) or external interrupts. Each port can be used as input or output (bi-direction)

Port Pin	Alternate Function
R3,0	RXD (serial input port)
P3h1	TXD (serial output port)
P3.2	INTO (external interrupt 0)
P3 3	INT1 (external interrupt 1)
P3.4	TO (Tinner 0 external input)
P3.5	T1 (Tinner 1 external input)
P3,e	WR (external data memory write strobe)
P3.7	RD (external data memory read strobe)

PSEN (out): Program Store Enable, the read signal for external program memory (active low).

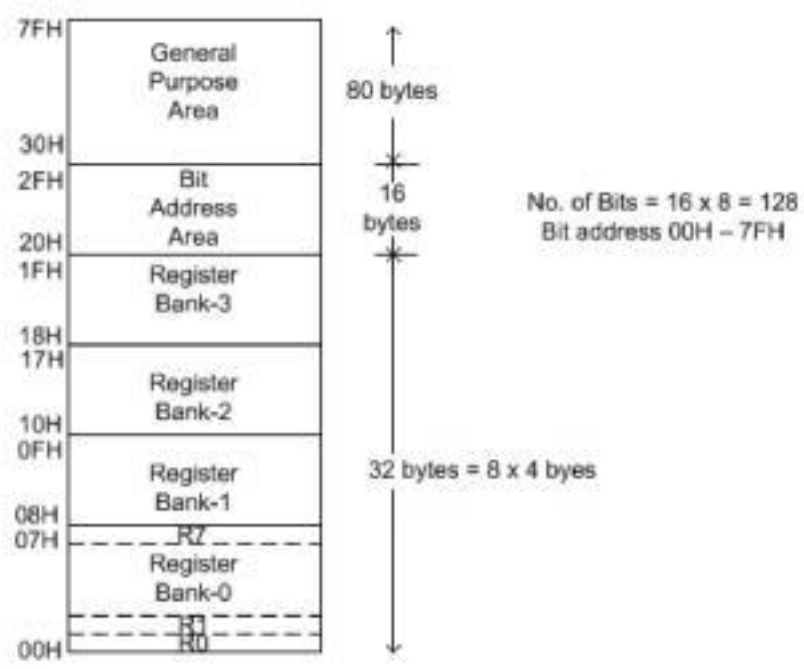
ALE (out): Address Latch Enable, to latch address outputs at Port0 and Port2 ***EA (in):*** External Access Enable, active low to access external program memory locations 0 to 4K ***RXD,TXD:*** UART pins for serial I/O on Port 3 ***XTAL1 & XTAL2:*** Crystal inputs for internal oscillator.

8051 Clock and Instruction Cycle:



In 8051, one instruction cycle consists of twelve (12) clock cycles. Instruction cycle is sometimes called as Machine cycle by some authors.

128 bytes of Internal RAM Structure (lower address space)



The lower 32 bytes are divided into 4 separate banks. Each register bank has 8 registers of one byte each. A register bank is selected depending upon two bank select bits in the PSW register. Next 16bytes are bit addressable. In total, 128bits (16X8) are available in bit addressable area. Each bit can be accessed and modified by suitable instructions. The bit addresses are from 00H (LSB of the first byte in 20H) to 7FH (MSB of the last byte in 2FH). Remaining 80bytes of RAM are available for general purpose.

4. Addressing Modes

The method of specifying the data to be operated by the instruction is called addressing mode.

Types:

1. Immediate addressing mode
2. Direct addressing mode
3. Register addressing mode
4. Register indirect addressing mode
5. Implied addressing mode
6. Relative addressing mode

Immediate addressing mode

Where data is available in the instruction itself.

Example: ADD A, #77H →The immediate data 77H is added to the data present in the accumulator. MOV A, #33H^The immediate data 33H is given in the instruction A register.

Register addressing mode

The instruction will specify the name of the register in which the data is available.

Example:

MOV R2, A - The content of the A register is moved to the register R2 of the currently selected register bank.

Direct addressing mode

The address of the data is specified directly to the instruction. The direct address can be the address of an internal RAM location (00H to &7FH) or the address of a special function register.

Example:

MOV A, 07H^The address of the register R7 of bank-0 is 07.The instruction will move the content the R7 register to the A-register (Accumulator).

Register indirect addressing mode

The instruction specifies the name of the register in which the address of the data is available. The internal RAM locations (00H-7FH) can be addressed indirectly through registers R1 and R0.The external RAM can be addressed indirectly

through DPTR.

Example:

MOV A,@R0^The internal RAM location R0 holds the address of the data. The content of the RAM location addressed by R0 is moved to the A register.

Implied Addressing mode

The instruction itself specifies the data to be operated by the instruction

Example: CPL^ Complement the carry flag.

Relative addressing mode

The instruction specifies the address relative to the program counter. The instruction will carry an offset whose range is-128 to +127.The offset is added to the PC to generate the 16 bit physical address.

Example:

JC offset^ If carry is one, the program control jumps to the address obtained by adding the content of the program counter and offset value in the instruction.

Instruction Set :

Mnemonic	*Description*	*Byte*	*Cycle*
Arithmetic Operations			
ADD A,Rn	Add register to accumulator	1	1
ADD A,direct	Add direct byte to accumulator	2	1
ADD A, @Ri	Add indirect RAM to accumulator	1	1
ADD A,#data	Add immediate data to accumulator	2	1
ADDC A,Rn	Add register to accumulator with carry flag	1	1
ADDC A,direct	Add direct byte to A with carry flag	2	1
ADDC A, @Ri	Add indirect RAM to A with carry flag	1	1
ADDC A, #data	Add immediate data to A with carry flag	2	1
SUBB A,Rn	Subtract register from A with borrow	1	1
SUBB A,direct	Subtract direct byte from A with borrow	2	1
SUBB A,@Ri	Subtract indirect RAM from A with borrow	1	1
SUBB A,#data	Subtract immediate data from A with borrow	2	1
INC A	Increment accumulator	1	1
INC Rn	Increment register	1	1
INC direct	Increment direct byte	2	1
INC @Ri	Increment indirect RAM	1	1
DEC A	Decrement accumulator	1	1
DEC Rn	Decrement register	1	1
DEC direct	Decrement direct byte	2	1
DEC @Ri	Decrement indirect RAM	1	1
INC DPTR	Increment data pointer	1	2
MUL AB	Multiply A and B	1	4
DIV AB	Divide A by B	1	4
DAA	Decimal adjust accumulator	1	1

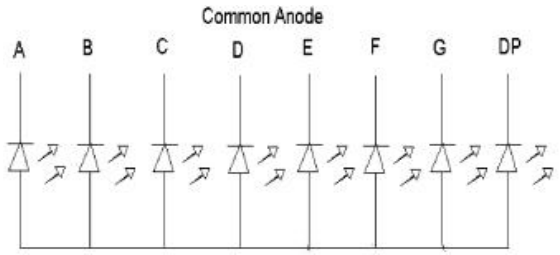
Mnemonics		*Description*	*Byte*	*Cycle*
ANL	A,Rn	AND register to accumulator	1	1
ANL	A,direct	AND direct byte to accumulator	2	1
ANL	A,@Ri	AND indirect RAM to accumulator	1	1
ANL	A,#data	AND immediate data to accumulator	2	1
ANL	direct,A	AND accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,direct	OR direct byte to accumulator	2	1
ORL	A,@Ri	OR indirect RAM to accumulator	1	1
ORL	A,#data	OR immediate data to accumulator	2	1
ORL	direct,A	OR accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive OR register to accumulator	1	1
XRL	A direct	Exclusive OR direct byte to accumulator	2	1
XRL	A,@Ri	Exclusive OR indirect RAM to accumulator	1	1
XRL	A,#data	Exclusive OR immediate data to accumulator	2	1
XRL	direct,A	Exclusive OR accumulator to direct byte	2	1
CLR	A	Clear accumulator	1	1
CPL	A	Complement accumulator	1	1
RL	A	Rotate accumulator left	1	1
RLC	A	Rotate accumulator left through carry	1	1
RR	A	Rotate accumulator right	1	1

Boolean Variable Manipulation

Mnemonics		***Description***	***Byte***	***Cycle***
CLR	C	Clear carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set carry flag	1	1
SETB	bit	Set direct bit	2	1
CPL	C	Complement carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to carry flag	2	2
ANL	C,/bit	AND complement of direct bit to carry	2	2
ORL	C,bit	OR direct bit to carry flag	2	2
ORL	C,/bit	OR complement of direct bit to carry	2	2
MOV	C,bit	Move direct bit to carry flag	2	1
MOV	bit,C	Move carry flag to direct bit	2	2

7.Interfacing with seven segment display Circuit Principle:

Seven segment displays internally consist of 8 LEDs. In these LEDs, 7 LEDs are used to indicate the digits 0 to 9 and single LED is used for indicating decimal point. Generally seven segments are two types, one is common cathode and the other is common anode. In common cathode, all the cathodes of LEDs are tied together and labeled as com. and the anode are left alone. In common anode, seven segment display all the anodes are tied together and cathodes are left freely. Below figure shows the internal connections of seven segment Display.

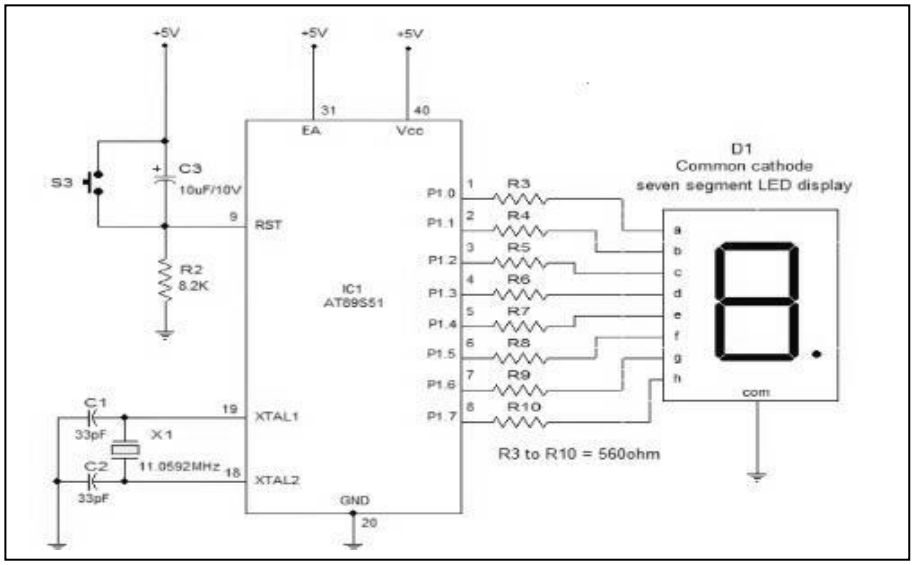


Circuit Design:

Here, common cathode seven segment is used to display the digits. In this circuit, pins a to h of the 7 segment are connected to the PORT 2 of the microcontroller and *com* pin is connected to the ground through the 330 ohm resistor. This resistor is used to drop the voltage. Since we are using common cathode seven segment we need to send LOGIC 1 to the segments to glow. Here dot is used for indicating the decimal point. Here all the cathodes of LED's are connected to the Gnd pin. The operating voltage of this LED's is 2 to 3 V but from controller we will get 5 V so to drop the remaining voltage we have to connect a to g pins to the controller through the resistor.

Digit Drive Pattern:

To display the digits on 7 segment, we need to glow different logic combinations of segments. For example if you want to display the digit 3 on seven segment then you need to glow the segments a, b, c, d and g. The below table show you the Hex decimal values what we need to send from PORT2 to Display the digits from 0 to 9.



Program:

OPCODE	*MNEMONICS*	*COMMENTS*
21,2C,41	LXI H,412C	LOAD IMMEDIATE DATA IN TO HL REGISTER PAIR
16,0F	MVI D,0FH	LOAD IMMEDIATE DATA 0FH TO D REGISTER
3E,10	MVI A,10H	LOAD IMMEDIATE DATA 10H TO THE ACCUMULATOR
D3,C2	OUT CNT(C2H)	OUTPUT DATA FROM ACCUMULATOR IS MOVED TO PORT
3E,CC	MVI A,CCH	LOAD IMMEDIATE DATA CCH TO THE ACCUMULATOR
D3,C2	OUT CNT	OUTPUT DATA FROM ACCUMULATOR IS MOVED TO PORT
3E,90	MVI A.90H	MOVE IMMEDIATE DATA 90H TO THE ACCUMULATOR
D3,C2	OUT CNT	OUTPUT DATA FROM ACCUMULATOR IS MOVED TO PORT
7E	MOV A,M	MOVE THE CONTENT OF MEMORY LOCATION TO THE ACCUMULATOR
D3,C0	OUT DAT(C0H)	OUTPUT DATA FROM ACCUMULATOR IS MOVED TO PORT
CD,1F,41	CALL DELAY	CALL THE DELAY PROGRAM
23	INX H	INCREMENT THE HL REGISTER PAIR
15	DCR D	DECREMENT D REGISTER
C2,11,41	JNZ LOOP	JUMP ON NO ZERO
C3,00,41	JMP START	JUMP UNCONDITIONALLY(ST ART)
06,A0	MVI B,0AH	MOVE IMMEDIATE DATA 0AH TO THE B REGISTER
0E,FF	MVI C,FFH	MOVE IMMEDIATE DATA FFH TO THE C REGISTER
C2,23,41	JNZ LOOP2	JUMP ON NON ZERO

LCD:

16^2 LCD module is a very common type of LCD module that is used in 8051 based embedded projects. It consists of 16 rows and 2 columns of 5x7 or 5x8 LCD dot matrices. The module were are talking about here is type number JHD162A which is a very popular one . It is available in a 16 pin package with back light ,contrast adjustment function and each dot matrix has 5x8 dot resolution.VEE pin is meant for adjusting the contrast of the LCD display and the contrast can be adjusted by varying the voltage at this pin. This is done by connecting one end of a POT to the Vcc (5V), other end to the Ground and connecting the center terminal (wiper) of of the POT to the VEE pin. See the circuit diagram for better understanding. The JHD162A has two built in registers namely data register and command register. Data register is for placing the data to be displayed , and the command register is to place the commands. The 16*2 LCD module has a set of commands each meant for doing a particular job with the display. We will discuss in detail about the commands later. High logic at the RS pin will select the data register and Low logic at the RS pin will select the command register. If we make the RS pin high and the put a data in the 8 bit data line (DB0 to DB7) , the LCD module will recognize it as a data to be displayed . If we make RS pin low and put a data on the data line, the module will recognize it as a command.

R/W pin is meant for selecting between read and write modes. High level at this pin enables read mode and low level at this pin enables write mode. E pin is for enabling the module. A high to low transition at this pin will enable the module.DB0 to DB7 are the data pins. The data to be displayed and the command instructions are placed on these pins.LED+ is the anode of the back

light LED and this pin must be connected to Vcc through a suitable series current limiting resistor. LED- is the cathode of the back light LED and this pin must be connected to ground.

16x2 LCD module commands.

16x2 LCD module has a set of preset command instructions. Each command will make the module to do a particular task. The commonly used commands and their function are given in the table below.

Command	Function
06	Increment cursor
80	Force cursor to the beginning of 1st line
C0	Force cursor to the beginning of 2nd line
38	Use 2 lines and 5*7 matrix
83	Cursor line 1 position 3
3C	Activate second line
08	Display OFF, Cursor OFF
C1	Jump to second line, positionl
OC	Display ON, Cursor OFF
C1	Jump to second line, positionl
C2	Jump to second line, position2

LCD initialization

The steps that has to be done for initializing the LCD display is given below and these steps are common for almost all applications.

- Send 38H to the 8 bit data line for initialization
- Send 0FH for making LCD ON, cursor ON and cursor blinking ON.

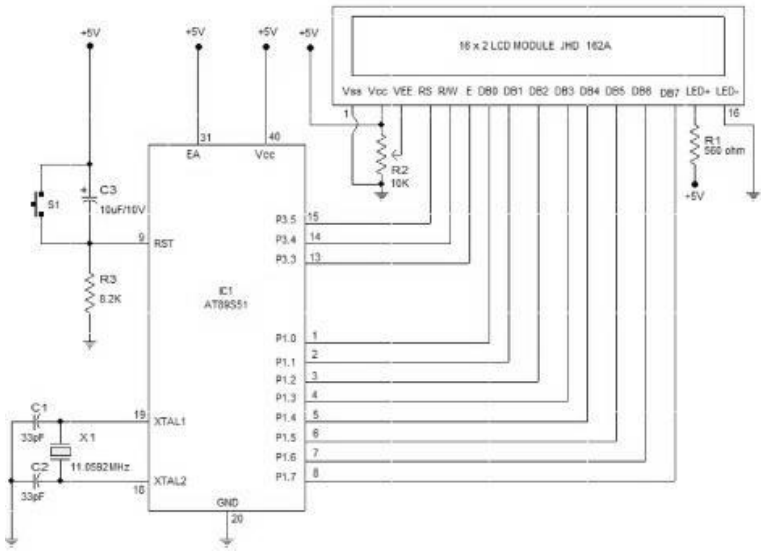
- Send 06H for incrementing cursor position.

  Send 01H for clearing the display and return the cursor.

Sending data to the LCD.

The steps for sending data to the LCD module is given below. I have already said that the LCD module has pins namely RS, R/W and E. It is the logic state of these pins that make the module to determine whether a given data input is a command or data to be displayed.

- Make R/W low.
- Make RS=0 if data byte is a command and make RS=1 if the data byte is a data to be displayed.
- Place data byte on the data register.
- Pulse E from high to low.
- Repeat above steps for sending another data.



he circuit diagram given above shows how to interface a 16*2 LCD module with AT89S1 microcontroller. Capacitor C3, resistor R3 and push button switch S1 forms the reset circuitry. Ceramic capacitors C1,C2 and crystal X1 is related to the clock circuitry which produces the system clock frequency. P1.0 to P1.7 pins of the microcontroller is connected to the DB0 to DB7 pins of the module respectively and through this route the data goes to the LCD module. P3.3, P3.4 and P3.5 are connected to the E, R/W, RS pins of the microcontroller and through this route the control signals are transffered to the LCD module. Resistor R1 limits the current through the back light LED and so do the back light intensity. POT R2 is used for adjusting the contrast of the display.

UNIT 4

ADVANCED MICROCONTROLLER

PIC/ ARM-RISC architecture, PC, Architecture, memory-Addressing modes-instruction set.

1. ARM ARCHITECTURE

The ARM is a 32-bit reduced instruction set computer (RISC) instruction set architecture (ISA) developed by ARM Holdings. It was known as the Advanced RISC Machine. Alcatel, Atmel, Broadcom are the companies using ARM architecture. It has 16 bit thumb instruction set.

REGISTERS

ARM has 37 registers all of which are 32-bits long. It has one dedicated program status register, five saved program status registers and thirty general purpose registers. The current PROCESSOR mode governs which of several banks is accessible. Each mode can access a particular set of r0-r12 registers particular r13 (the stack pointer, sp) and r14 (the link register, lr), program counter r15 (pc), cpsr Privileged modes (except System) can also access a particular spsr (saved program status register).

User32	FIQ32	Supervisor32	Abort32	IRQ32	Undefined32
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8_fiq	R8	R8	R8	R8
R9	R9_fiq	R9	R9	R9	R9
R10	R10_fiq	R10	R10	R10	R10
R11	R11_fiq	R11	R11	R11	R11
R12	R12_fiq	R12	R12	R12	R12
R13	R13_fiq	R13_svc	R13_abt	R13_irq	R13_und
R14	R14_fiq	R14_svc	R14_abt	R14_irq	R14_und
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)

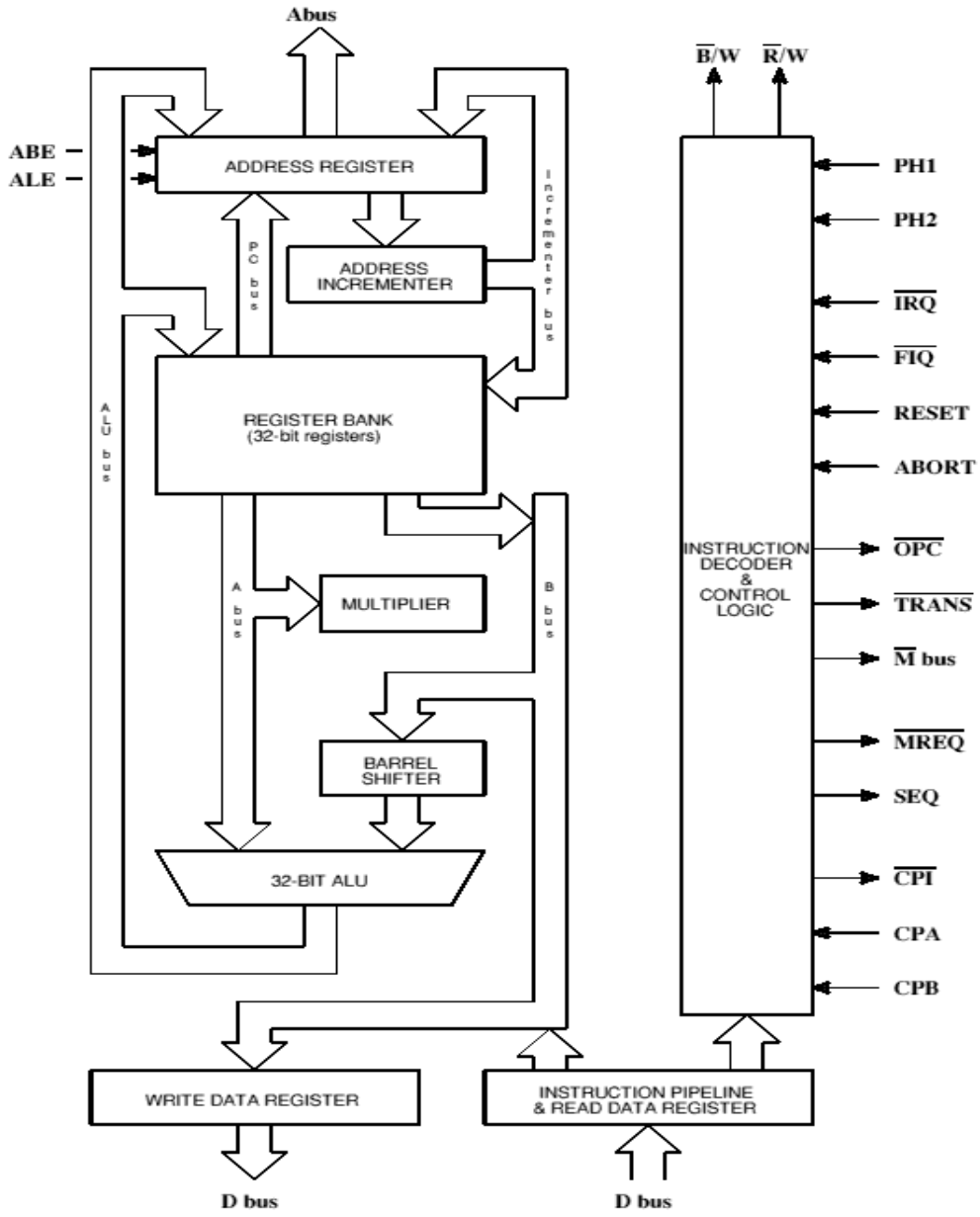
Program Status Registers

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

Program Status Register:

- Condition code flags
  - N = Negative result from ALU
  - Z = Zero result from ALU
  - C = ALU operation Carried out
  - V = ALU operation overflowed
- Sticky Overflow flag - Q flag
  - Indicates if saturation has occurred
- J bit
  - J = 1 PROCESSOR in Jazelle state (Jazelle - Direct byte code execution)
- Interrupt Disable bits.
- I = 1 Disables the IRQ.
  - F = 1 Disables the FIQ.
- T Bit
  - T = 0: PROCESSOR in ARM state > T = 1: PROCESSOR in Thumb state

BLOCK DIAGRAM



Modes:

M [4:01	Mode	Accessible register set	

Note: System: **Privileged mode using the same registers as user mode**

Program Counter (r15)

During the PROCESSOR is executing in ARM state at the time all instructions are 32 bits wide word aligned. Therefore the pc value is stored in bits [31:2] with bits [1:0] undefined (as instruction cannot be half word or byte aligned). Instead the PROCESSOR is executing in Thumb state: Here all instructions are 16 bits wide. All instructions must be half word aligned. Therefore the pc value is stored in bits [31:1] with bit [0] undefined (as instruction cannot be byte aligned). Executing in Thumb state: When the PROCESSOR is executing in Jazelle state: All instructions are 8 bits wide PROCESSOR performs a word access to read 4 instructions at once.

Exception Handling

- When an exception occurs, the ARM: Copies CPSR into SPSR_<mode>
- Sets appropriate CPSR bits
    - It will Change to ARM state or exception mode or disable interrupts Stores the return address in LR_<mode>
- Sets PC to vector address

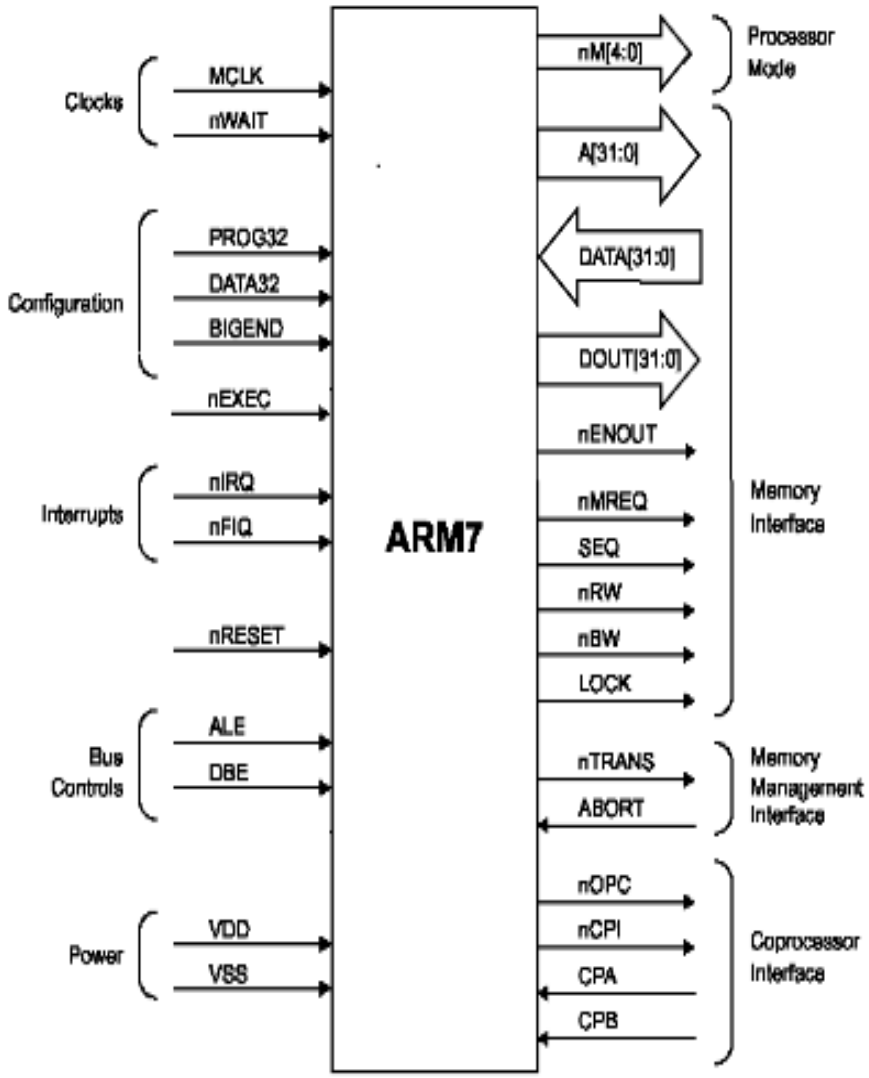
Vector address:

0x1C	FIQ
0X18	IRQ
0X14	Reserved
0X10	Data Abort
0X0C	Prefetch Abort
0X08	Software Interrupt
0X04	Undefined Instruction
0X00	Reset

• To return, exception handler needs to:

- Restore CPSR from SPSR_<mode>
- Restore PC from LR <mode>

Pin Diagram:



Conditional Execution and Flags

ARM instructions can be made to execute conditionally by post fixing them with the appropriate condition code field. This improves code density and performance by reducing the number of forward branch instructions. By default, data processing instructions do not affect the condition code flags but the flags can be optionally set by using “S”. CMP does not need “S”.

Instruction Set

Branch instructions:

Branch : B{<cond>} label

Branch with Link : BL{<cond>} subroutine_label

The PROCESSOR core shifts the offset field left by 2 positions, sign-extends it and adds it to the PC(± 32 Mbyte range)

31 28	27 25	24	23 0
Cond	1 0 1	L	Offset

L -Link Bit 0-Branch 1- Branch with Link

Cond-Condition Field.

Data processing Instructions:

It consists of:

Arithmetic:	ADD	ADC	SUB	SBC
Logical:	AND	ORR	EOR	BIC
Comparisons: CMP	CMN	TST	TEQ	
Data movement:	MOV	MVN		

Note: These instructions only work on registers, NOT memory.

Syntax: <Operation>{< cond >}{S} Rd, Rn, Operand2

- Comparisons set the flags only - they do not specify the Rd
- Data movement does not specify Rn
- Second operand is sent to the ALU via barrel shifter.

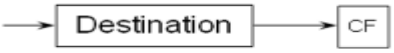
Logical Left Shift:

Multiplication by a power of 2

Arithmetic Right Shift:

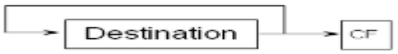
Division by a power of 2, preserving the sign bit

Logical Shift Right:



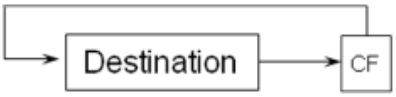
Division by a power of 2

Rotate Right:



Bit rotate with wrap around from LSB to MSB

Rotate Right Extended:



Single bit rotate with wrap around from CF to MSB

The Barrel Shifter:

1. LSL : Logical Left Shift
2. ASR: Arithmetic Right Shift
3. LSR : Logical Shift Right
4. ROR: Rotate Right
5. RRX: Rotate Right Extended

Condition Codes

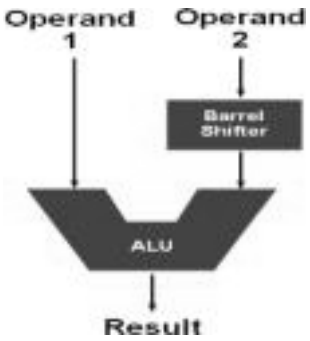
The possible condition codes are listed below:

Suffix	Description	Flags tested
EQ	**Equal**	Z=1
NE	**Not equal**	Z=0
CS/HS	**Unsigned higher or same**	C=1
CC/LO	**Unsigned lower**	c=0
MI	**Minus**	N=1
PL	**Positive or Zero**	N=0
VS	**Overflow**	V=1
VC	**No overflow**	v=0
HI	**Unsigned higher**	c=uz=0
LS	**Unsigned lower or same**	C=0or Z=1
GE	**Greater or equal**	N=V
LT	**Less than**	N!=V
GT	**Greaterthan**	Z=0 & N=V
LE	**Less than or equal**	Z=1 or N=!V
AL	**Always**	

Note: AL is the default and does not need to be specified

Using the Barrel Shifter:

The Second Operand



Register, optionally with shift operation, Shift value can be 5 bit unsigned integer. Specified in bottom byte of another register. Used for multiplication by constant immediate value representation should be a 8 bit number, with a range of 0-255.Rotated right through even number of positions. Allows increased range of 32-bit constants to be loaded directly into registers

Immediate constants:

- No ARM instruction can contain a 32 bit immediate constant
- All ARM instructions are fixed as 32 bits long
- The data processing instruction format has 12 bits available for operand
- The 4 bit rotate value (0-15) is multiplied by two to give range 0-30 in steps of 2
- 8-bits shifted by an even number of bit positions

Multiply:

Syntax:

MUL{<cond>}{S} Rd, Rm, Rs Rd = Rm * Rs

Eg: MUL r10,r2, r5

MLA r10, r2, r1, r5

No of Cycle:

2-5 cycles for ARM7, 1-3 cycles for Strong ARM, 2 cycles for ARM9 and one cycle for accumulate and long.

Single register data transfer:

Address accessed by LDR/STR is specified by a base register plus an offset

LDR STR Word

LDRB STRB Byte

Syntax:

LDR{<cond>}{<size>} Rd, <address>

For word and unsigned byte accesses, offset can be

Unsigned 12-bit immediate value LDR r0, [r1,#8]

A register, optionally shifted by an immediate value LDR r0,[r1,r2]

LDR r0,[r1,r2,LSL#2]

This can be either added or subtracted from the base register LDR r0,[r1,#-8]

LDR r0,[r1,-r2]

LDR r0,[r1,-r2,LSL#2]

For half word and signed half word / byte, offset can be An unsigned 8 bit immediate value (ie 0-255 bytes).

A register (unshifted).

Pre-indexed Addressing

STR r0, [r1,#12]

Post-indexed Addressing **STR r0, [r1],#12**
LDM / STM operation Syntax:

<LDM|STM>{<cond>}<addressing_mode> Rb{!}, <register list>

Addressing modes:

Increment after- LDMIA / STMIA Increment before- LDMIB / STMIB Decrement after- LDMDA / STMDA Decrement before- LDMDB / STMDB **Implementing stacks with LDM and STM:**

- **Descending or ascending**

The stack grows downwards, starting with a high address and progressing to a lower one (a descending stack), or upwards, starting from a low address and progressing to a higher address (an ascending stack).

- Full or empty

The stack pointer can either point to the last item in the stack (a full stack), or the next free space on the stack (an empty stack).

Software Interrupt (SWI):

- Causes an exception trap to the SWI hardware vector

- The SWI handler can examine the SWI number to decide what operation has been requested.
- By using the SWI mechanism, an operating system can implement a set of privileged operations which applications running in user mode can request.

Syntax:

SWI{<cond>} <SWI number>

PSR Transfer Instructions:

MRS and MSR allow contents of CPSR / SPSR to be transferred to / from a general purpose register.

Syntax:

MRS{<cond>} Rd,<psr> ; Rd = <psr>

ARM Branches and Subroutines:

- B <label>
  - PC relative. ±32 Mbyte range.
- BL <subroutine>
  - Stores return address in LR
  - Returning implemented by restoring the PC from LR > For non-leaf functions, LR will have to be stacked

THUMB Instruction:

Thumb is a 16-bit instruction set. Optimized for code density from code improved performance from narrow memory. Subset of the functionality of

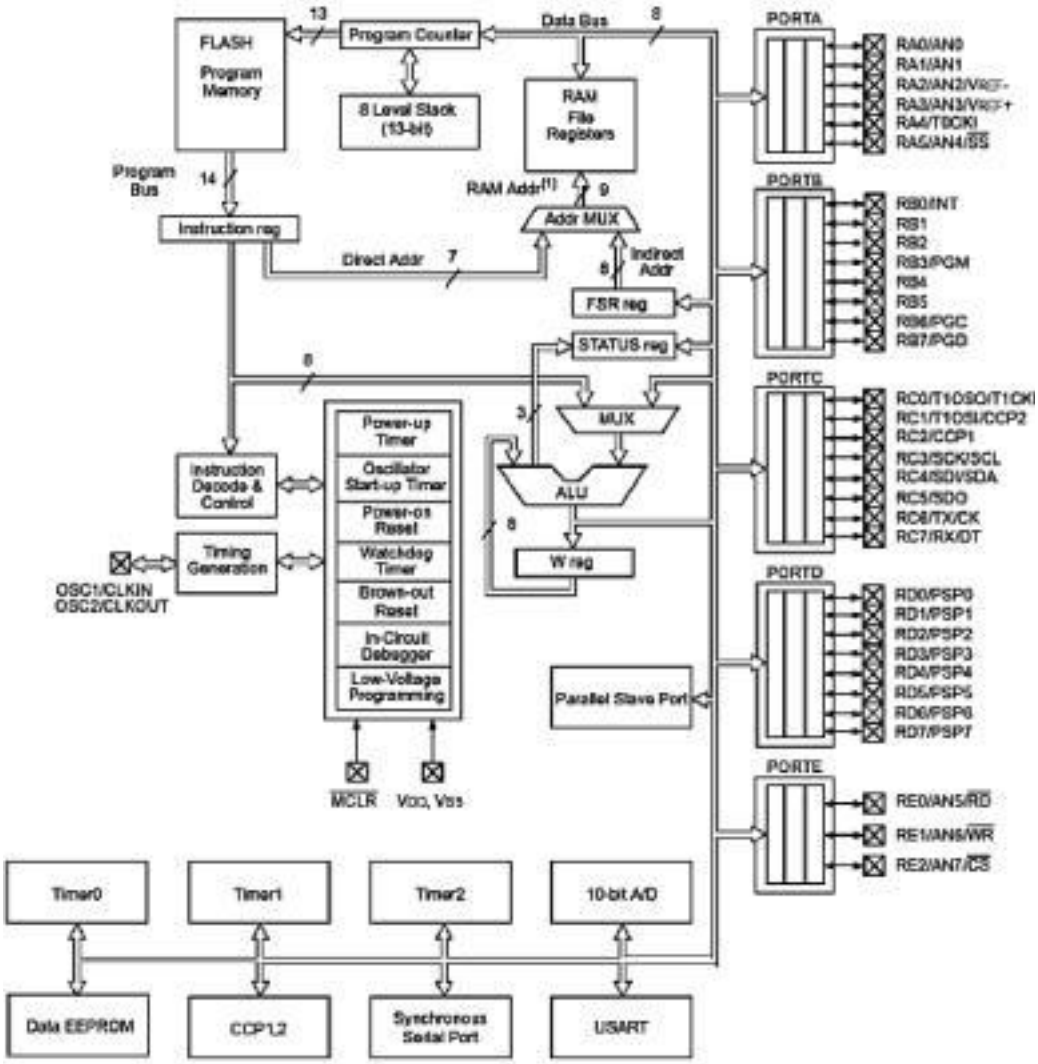
the ARM instruction set. Core has additional execution state. Switch between ARM and Thumb using BX instruction

For most instructions generated by compiler:

- Conditional execution is not used
- Source and destination registers identical
- Only Low registers used
- Constants are of limited size
- Inline barrel shifter not used

2. PIC Architecture:

General Features



The function of CPU in PIC is same as a normal microcontroller CPU. A PIC, CPU consists of several sub units such as instruction decoder, ALU,

accumulator, control unit, etc. The CPU in PIC normally supports Reduced Instruction Set Computer (RISC) architecture. RISC design is based on the premise that most of the instructions. The computer decodes and executes are simple. As a result, RISC architecture limits the number of instructions. Execution Time is less.

Memory

The memory in a PIC chip used to store the data and programs (temporary or permanently). PIC also has certain amount of memory space for RAM, ROM, and EEPROM and other flash memory, etc. ROM memory is used for permanent storage memory. The contents in the EEPROM changes during run time and at that time it acts like a RAM memory. But the difference is after the power goes off, the data remains in this ROM chip. This is the one of the special advantages of EEPROM. In the PIC chip the function of EPROM is to store the values created during the runtime.RAM memory is the one of the complex memory module in a PIC chip. This memory associated with various type of registers (special function registers and general purpose registers) and memory BANK modules (BANK 0, BANK 1, etc.). Once the power goes off, the contents in the RAM will be cleared. As like normal microcontrollers, the RAM memory is used to store temporary data and provide immediate results. The flash memory is a special type of memory where READ, WRITE, and ERASE operations can be done many times.

Registers

Information is stored in a CPU memory location called a register. Registers can be thought of as the CPUs tiny scratchpad, temporarily storing instructions or

data. Registers basically classified into the following.

General Purpose Register (GPR)

A general purpose register (or PROCESSOR register) is a small storage area available on a CPU whose contents can be accessed more quickly than other storage that available on PIC. A general purpose register can store both data addresses simultaneously.

Special Function registers (SFR)

These are also a part of RAM memory locations. As compared to GPR, their purpose is predetermined during the manufacturing time and cannot be changed by the user. It is only for special dedicated functions.

Interrupts

Interrupt is the temporary delay in a running program. These delays stop the current execution for a particular interval. This interval/delay is usually called as interrupt. When an interrupt request arrives into a current execution program, then it stops its regular execution. Interrupt can be performed by externally (hardware interrupt) or internally (by using software).

BUS

BUS is the communication or data transmission/reception path in a microcontroller unit. In a normal microcontroller chip, two types of buses are normally available.

Data bus

Data bus is used for memory addressing. The function of data bus is interfacing all the circuitry components inside the PIC chip.

Address bus

Address bus mostly used for memory addressing. The function of address bus is to transmit the address from the CPU to memory locations.

USART or UART

These ports are used for the transmission (TX) and reception (RX) of data. These transmissions possible with help of various digital data transceiver modules like RF, IR, Bluetooth, etc. This is the one of the simplest way to communicate the PIC chip with other devices.

Oscillators

Oscillator unit basically an oscillation/clock generating circuit which is used for providing proper clock pulses to the PIC chip. This clock pulses also helps the timing and counting applications. A PIC chip normally use various types of clock generators. According to the application and the type of PIC used, the oscillators and its frequencies may vary. RC (Resistor- Capacitor), LC (Inductor-Capacitor), RLC (Resistor-Inductor-capacitor), crystal oscillators, etc are the normal oscillators used with A PIC chip.

Stack

The entire PIC chip has an area for storing the return addresses. This area or unit called Stack is used in some Peripheral interface controllers. The hardware stack is not accessible by software. But for most of the controllers, it can be easily accessible.

Input/ Output Ports

These ports are used for the interfacing various input/output devices and memories. According to the type of PIC, the number of ports may change.

Advanced functioning blocks

These sections include various advanced features of a PIC chip. According to the type of PIC, these features may change. Various advanced features in a peripheral interface controller are power up timer, oscillator start up timer, power on reset, watch dog timer, brown out reset, in circuit debugger, low voltage programming, voltage comparator, CCP modules etc.

Memory Organization of PIC16F877

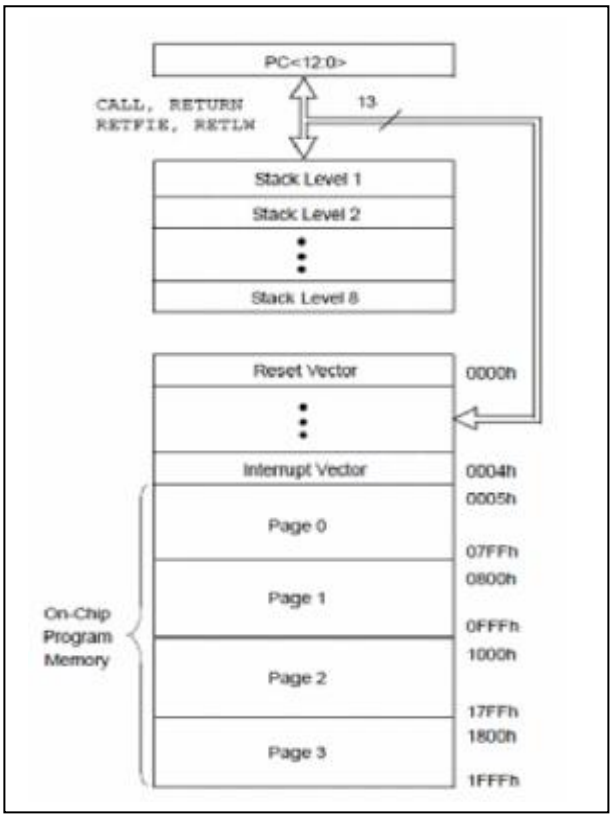
The memory of a PIC 1 6F877 chip is divided into 3 sections. They are

1. Program memory
2. Data memory and
3. Data EEPROM

Program memory

Program memory contains the programs that are written by the user. The program counter (PC) executes these stored commands one by one. Usually PIC1 6F877 devices have a 13 bit wide program counter that is capable of addressing 8K*14 bit program memory space. This memory is primarily used for storing the programs that are written (burned) to be used by the PIC. These devices also have 8K*14 bits of flash memory that can be electrically erasable /reprogrammed. Each time we write a new program to the controller, we must delete the old one at that time. The figure below shows the program memory map and stack.

PROGRAM MEMORY MAP AND STACK



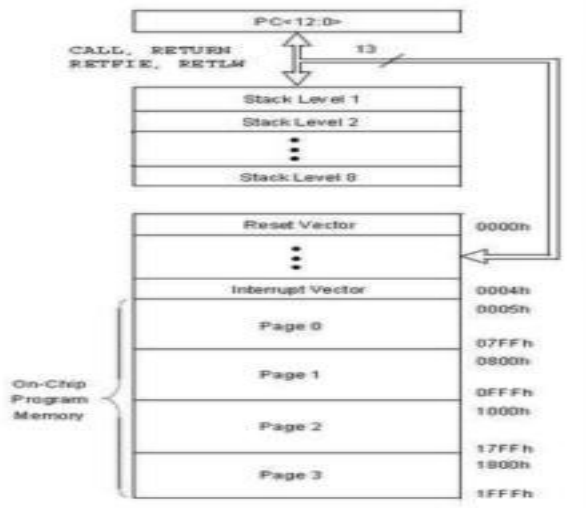
PIC16f877 Program Memory

Program counters (PC) is used to keep the track of the program execution by holding the address of the current instruction. The counter is automatically incremented to the next instruction during the current instruction execution.

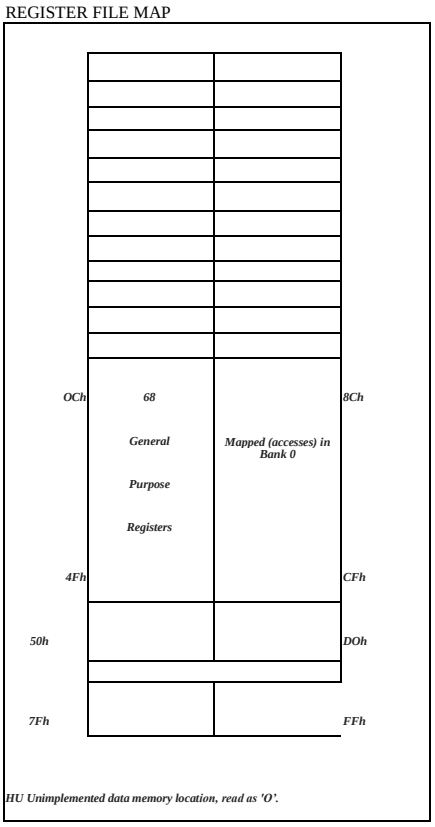
The PIC16F87XA family has an 8-level deep 13-bit wide hardware stack. The stack space is not a part of either program or data space and the stack pointers are not readable or writable. In the PIC microcontrollers, this is a special block of RAM memory used only for this purpose. Each time the main program execution starts at address 0000 - Reset Vector. The address 0004 is “reserved” for the “interrupt service routine” (ISR).

PIC16F87XA Data Memory Organization

The data memory of PIC1 6F877 is separated into multiple banks which contain the general purpose registers (GPR) and special function registers (SPR). According to the type of the microcontroller, these banks may vary. The PIC1 6F877 chip only has four banks (BANK 0, BANK 1, BANK 2, and BANK4). Each bank holds 128 bytes of addressable memory.



The banked arrangement is necessary because there are only 7 bits are available in the instruction word for the addressing of a register, which gives only 128 addresses. The selection of the banks are determined by control bits RP1, RP0 in the STATUS registers Together the RP1, RP0 and the specified 7 bits effectively form a 9 bit address. The first 32 locations of Banks 1 and 2, and the first 16 locations of Banks2 and 3 are reserved for the mapping of the Special Function Registers (SFR).



Data EEPROM and FLASH

The data EEPROM and Flash program memory is readable and writable during

normal operation (over the full VDD range). This memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers. There are six SFRs used to read and write this memory:

- EECON1
- EECON2
- EEDATA
- EEDATH
- EEADR
- EEADRH

The EEPROM data memory allows single-byte read and writes. The Flash program memory allows single-word reads and four-word block writes. Program memory write operations automatically perform an erase-before write on blocks of four words. A byte write in data EEPROM memory automatically erases the location and writes the new data (erase-beforewrite). The write time is controlled by an on-chip timer. The write/erase voltages are generated by an on-chip charge pump, rated to operate over the voltage range of the device for byte or word operations.

General Features

- High performance RISC CPU.
- ONLY 35 simple word instructions.
- All single cycle instructions except for program branches which are two cycles.
- Operating speed: clock input (200MHz), instruction cycle (200nS).

- Up to 368*8bit of RAM (data memory), 256*8 of EEPROM (data memory), 8k*14 of flash memory.
- Pin out compatible to PIC 1 6C74B, PIC 1 6C76, PIC 1 6C77.
- Eight level deep hardware stack.
- Interrupt capability (up to 14 sources).
- Different types of addressing modes (direct, Indirect, relative addressing modes).
- Power on Reset (POR).
- Power-Up Timer (PWRT) and oscillator start-up timer.
- Low power- high speed CMOS flash/EEPROM.
- Fully static design.
- Wide operating voltage range (2.0 - 5.56)volts.
- High sink/source current (25mA).
- Commercial, industrial and extended temperature ranges.
- Low power consumption (<0.6mA typical @3v-4MHz, 20pA typical @3v-32MHz and <1 A typical standby).

Peripheral Features

- Timer 0: 8 bit timer/counter with pre-scalar.
- Timer 1:16 bit timer/counter with pre-scalar.
- Timer 2: 8 bit timer/counter with 8 bit period registers with pre-scalar and post-scalar.
- Two Capture (16bit/12.5nS), Compare (16 bit/200nS), Pulse Width Modules (10bit).
- 1 Obit multi-channel A/D converter

- Synchronous Serial Port (SSP) with SPI (master code) and I2C (master/slave).
- Universal Synchronous Asynchronous Receiver Transmitter (USART) with 9 bit address detection.
- Parallel Slave Port (PSP) 8 bit wide with external RD, WR and CS controls (40/46pin).
- Brown Out circuitry for Brown-Out Reset (BOR).

Key Features

- Maximum operating frequency is 20MHz.
- Flash program memory (14 bit words), 8KB.
- Data memory (bytes) is 368.
- EEPROM data memory (bytes) is 256.
- 5 input/output ports.
- 3 timers.
- 2 CCP modules.
- 2 serial communication ports (MSSP, USART).
- PSP parallel communication port

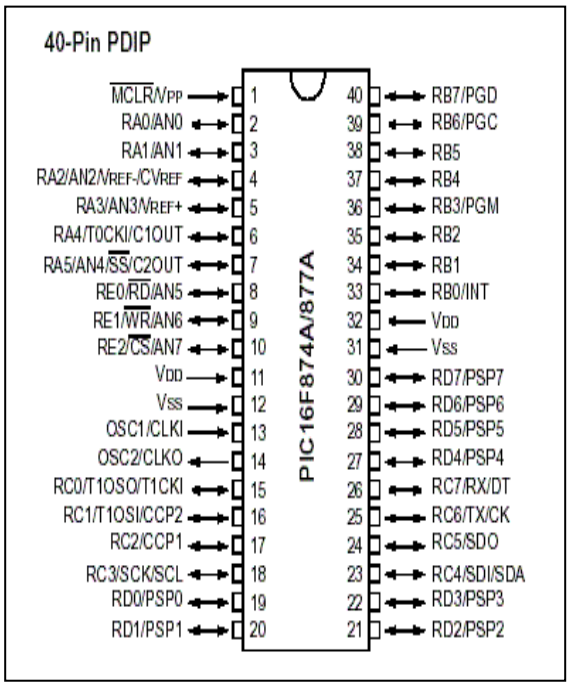
Analog Features

- 1 Obit, up to 8 channel A/D converter.
- Brown Out Reset function.
- Analog comparator module.

Special Features

- 100000 times erase/write cycle enhanced memory.
- 1000000 times erase/write cycle data EEPROM memory.
- Self programmable under software control.
- In-circuit serial programming and in-circuit debugging capability.
- Single 5V,DC supply for circuit serial programming
- WDT with its own RC oscillator for reliable operation.
- Programmable code protection.
- Power saving sleep modes.
- Selectable oscillator options.

Pin Diagram



PIC16F877 chip is available in different types of packages. According to the type of applications and usage, these packages are differentiated. The pin diagrams of a PIC16F877 chip in different packages. PIC1 6F877 has 5 basic input/output ports. They are usually denoted by PORT A (R A), PORT B (RB), PORT C (RC), PORT D (RD), and PORT E (RE). These ports are used for input/ output interfacing. In this controller, PORT A is only 6 bits wide (RA-0 to RA-7), PORT B , PORT C,PORT D are only 8 bits wide (RB-0 to RB-7,RC-0 to RC-7,RD-0 to RD-7), PORT E has only 3 bit wide (RE-0 to RE-7).

All these ports are bi-directional. The direction of the port is controlled by using TRIS(X) registers (TRIS A used to set the direction of PORT-A, TRIS B used to set the direction for PORT- B, etc.). Setting a TRIS(X) bit1 will set the corresponding PORT(X) bit as input. Clearing a TRIS(X) bit 0 will set the corresponding PORT(X) bit as output.(If we want to set PORT A as an input, just set TRIS(A) bit to logical „1 and want to set PORT B as an output, just set t he PORT B bits to logical 0.)

- Analog input port (AN0 TO AN7): these ports are used for interfacing analog inputs.
- TX and RX: These are the USART transmission and reception ports.
- SCK: These pins are used for giving synchronous serial clock input.
- SCL: These pins act as an output for both SPI and I2C modes.
- DT: These are synchronous data terminals.
- CK: Synchronous clock input.
- SD0: SPI data output (SPI Mode).
- SD1: SPI Data input (SPI mode).
- SDA: Data input/output in I2C Mode.

- CCP1 and CCP2: These are capture/compare/PWM modules.
- OSC1: Oscillator input/external clock.
- OSC2: Oscillator output/clock out.
- MCLR: Master clear pin (Active low reset).
- Vpp: programming voltage input.
- THV: High voltage test mode controlling.
- Vref (+/-): reference voltage.
- SS: Slave select for the synchronous serial port.
- T0CK1: clock input to TIMER 0.
- T1OSO: Timer 1 oscillator output.
- T1OS1: Timer 1 oscillator input.
- T1CK1: clock input to Timer 1.
- PGD: Serial programming data.
- PGC: serial programming clock.
- PGM: Low Voltage Programming input.
- INT: external interrupt.
- RD: Read control for parallel slave port.
- CS: Select control for parallel slave.
- PSP0 to PSP7: Parallel slave port.
- VDD: positive supply for logic and input pins.
- VSS: Ground reference for logic and input/output pins

Input/ Output Ports

In order to synchronize the operation of I/O ports with the internal 8-bit organization of the microcontroller, they are, similar to registers, grouped into five ports denoted by A, B, C, D and E. All of them have several features in

common: If a pin performs any of these functions, it may not be used as a general-purpose input/output pin. TRIS register: TRISA, TRISB, TRISC etc which determines the performance of port bits, but not their contents. By clearing any bit of the TRIS register (bit=0), the corresponding port pin is configured as an output. Similarly, by setting any bit of the TRIS register (bit=1), the corresponding port pin is configured as an input. This rule is easy to remember 0 = Output, 1 = Input.

PORTC and TRISC register

Port C is an 8-bit wide, bidirectional port. Bits of the TRISC register determine the function of its pins. Similar to other ports, a logic one (1) in the TRISC register configures the appropriate portC pin as an input. Port D is an 8 bit wide, bidirectional port Bits .

PORTE and TRISE register

Port E is a 4-bit wide, bidirectional port. The TRISE registers bits determine the function of its pins. Similar to other ports, a logic one in the TRISE register configures the appropriate portE pin as an input. The exception is the RE3 pin which is always configured as an input. Similar to ports A and B, three pins can be configured as analog inputs in this case. The ANSELH register bits determine whether a pin will act as an analog input (AN) or digital input/output:

RE0 = AN5
RE1 = AN6
RE2 = AN7

Instruction Set of PIC 16F877

The instruction set for the 16F8XX includes 35 instructions. The reason for such a small number of instructions lies in the RISC architecture. It means that instructions are well optimized from the aspects of operating speed, simplicity in architecture and code compactness.

Instruction Execution Time

All instructions are single-cycle instructions. The only exception may be conditional branch instructions (if condition is met) or instructions performed upon the program counter. In both cases, two cycles are required for instruction execution, while the second cycle is executed as an NOP *(No Operation).* Single-cycle instructions consist of four clock cycles. If 4MHz oscillator is used, the nominal time for instruction execution is 1gs. As for jump instructions, the instruction execution time is 2gs.

- Data transfer Instruction
- Arithmetic and Logic Instruction
- Bit oriented Instruction
- Program Control Instruction

Data transfer Instruction:

The data is copied from source to Destination without any change.

EX: MOVLW k-^Move constant to W.

MOV WF f^Move W to F CLR W^ Clear W

Arithmetic and Logic Instruction:

To perform arithmetic operation such as addition, subtraction, Increment and decrement. The group of instruction perform logical operation such as AND, OR, Exclusive-OR, Rotate, Compare, and Complement the content.

EX: ADDLW k^ Add W and Constant

SUB LW k -^Subtract W from constant

IORLW k^Logical OR with W with constant

Bit oriented Instruction:

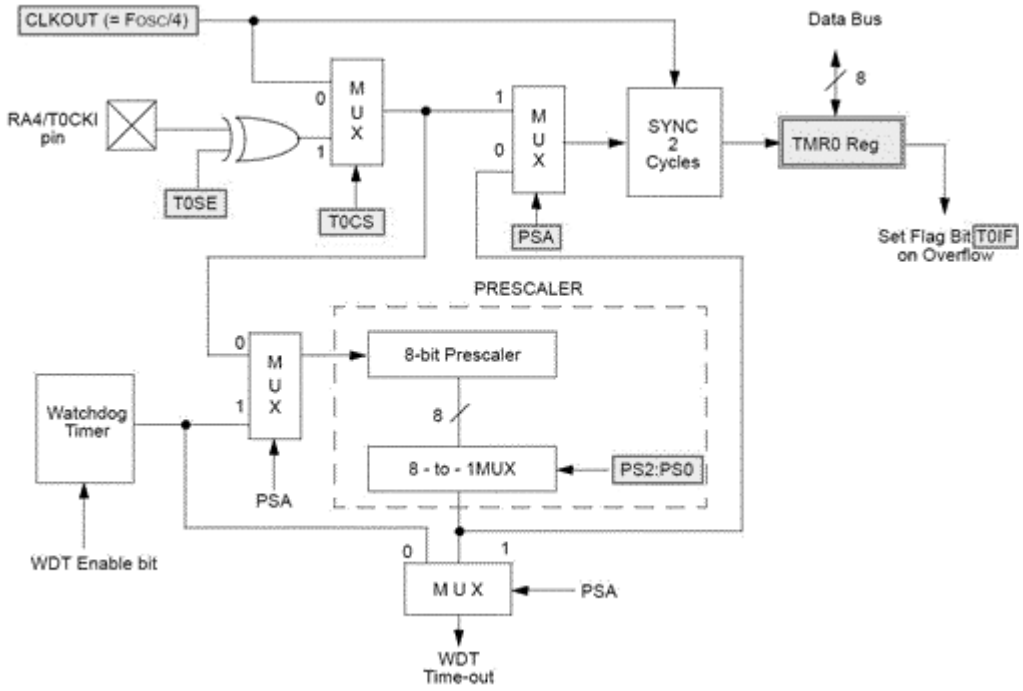
BC F f, b^Clear bit b in f

Program Control Instruction:

CALL k^Call subroutine

RETURN^Return from subroutine

Block Diagram of WDT pre scalar

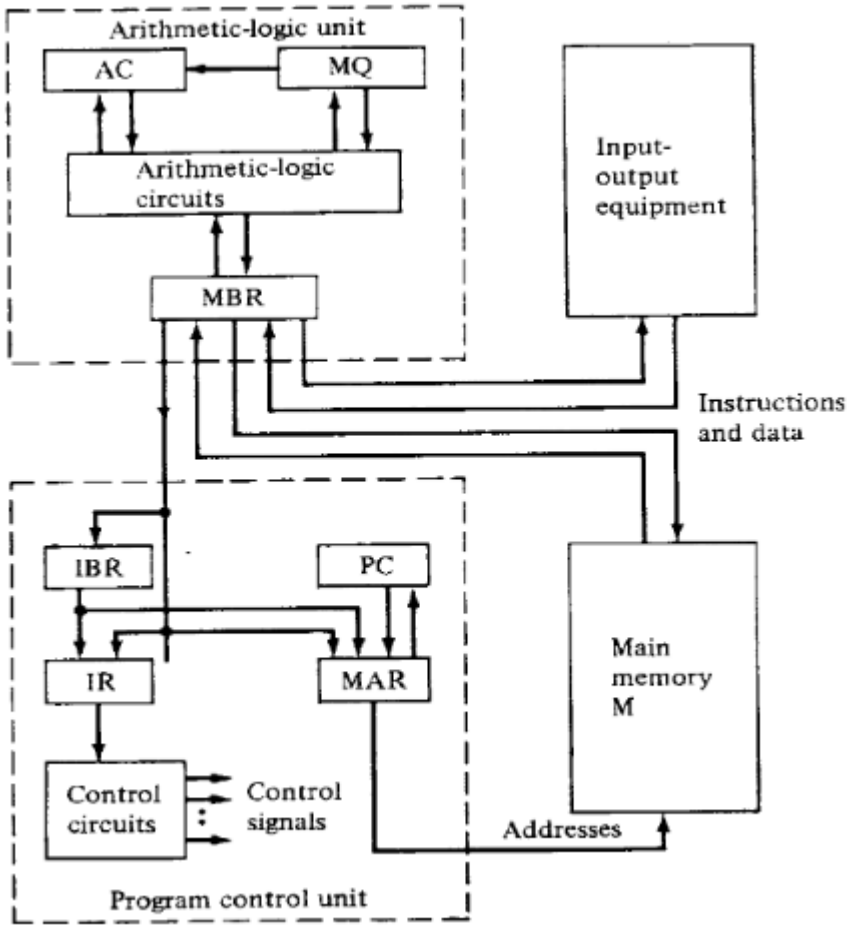


Block Diagram of Diagram pre scalar

Register Operations:

Stores values from other locations (registers and memory). It plays a role in Addition and subtraction Shift or rotate data and to test contents for conditions such as zero or positive.

3. PC ARCHITECTURE:

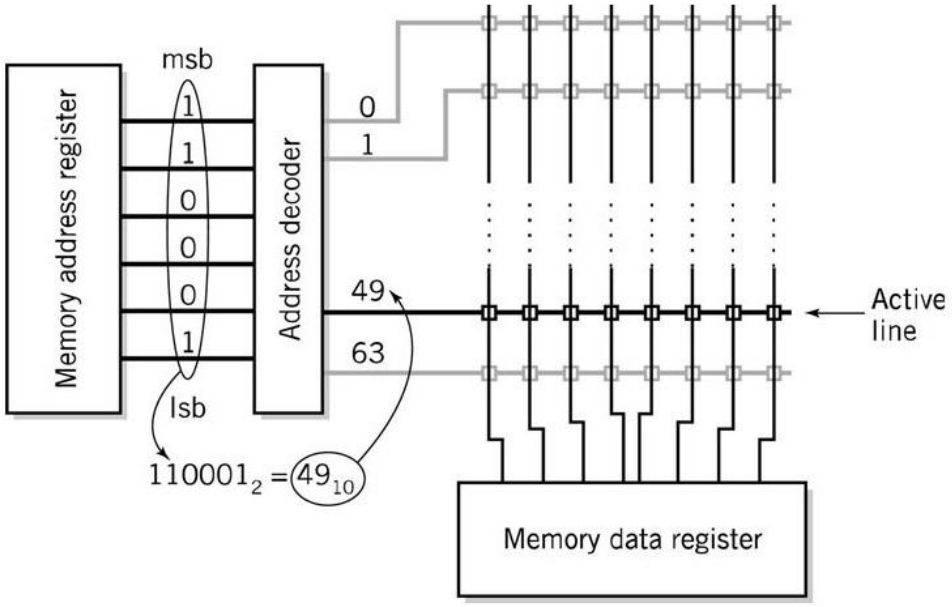


Operation of Memory:

- Each memory location has a unique address
- Address from an instruction is copied to the MAR which finds the location in memory
- CPU determines if it is a store or retrieval

- Transfer takes place between the MDR and memory
- MDR is a two way register

Relationship between MAR, MDR and Memory:



Memory Capacity:

Determined by two factors Number of bits in the MAR

- LMC = 100 (00 to 99)
- 2^K where K = width of the register in bits

Size of the address portion of the instruction

- 4 bits allows 16 locations
- 8 bits allows 256 locations
- 32 bits allows

4,294,967,296 or 4 GB Important for performance

- Insufficient memory can cause a PROCESSOR to work at 50%

below performance **Random Access Memory**

DRAM (Dynamic RAM)

- Most common, cheap
- Volatile: must be refreshed (recharged with power) 1000's of times each second SRAM (static RAM)
- Faster than DRAM and more expensive than DRAM
- Volatile
- Frequently small amount used in cache memory for high-speed access used **ROM - Read Only Memory**
- Non-volatile memory to hold software
- Magnetic core memory
- EEPROM
  - Electrically Erasable Programmable ROM
  - Slower and less flexible than Flash ROM
  - Flash *ROM*

Fetch-Execute Cycle

- Two-cycle process because both instructions and data are in memory
- Fetch
  - Decode or find instruction, load from memory into register and signal ALU
- Execute
  - Performs operation that instruction requires
  - Move/transform data

Transfer the address from the PC to the MAR Transfer the instruction to the IR Address portion of the instruction loaded in MAR Actual data copied into the accumulator Program Counter incremented

ADD Fetch/Execute Cycle

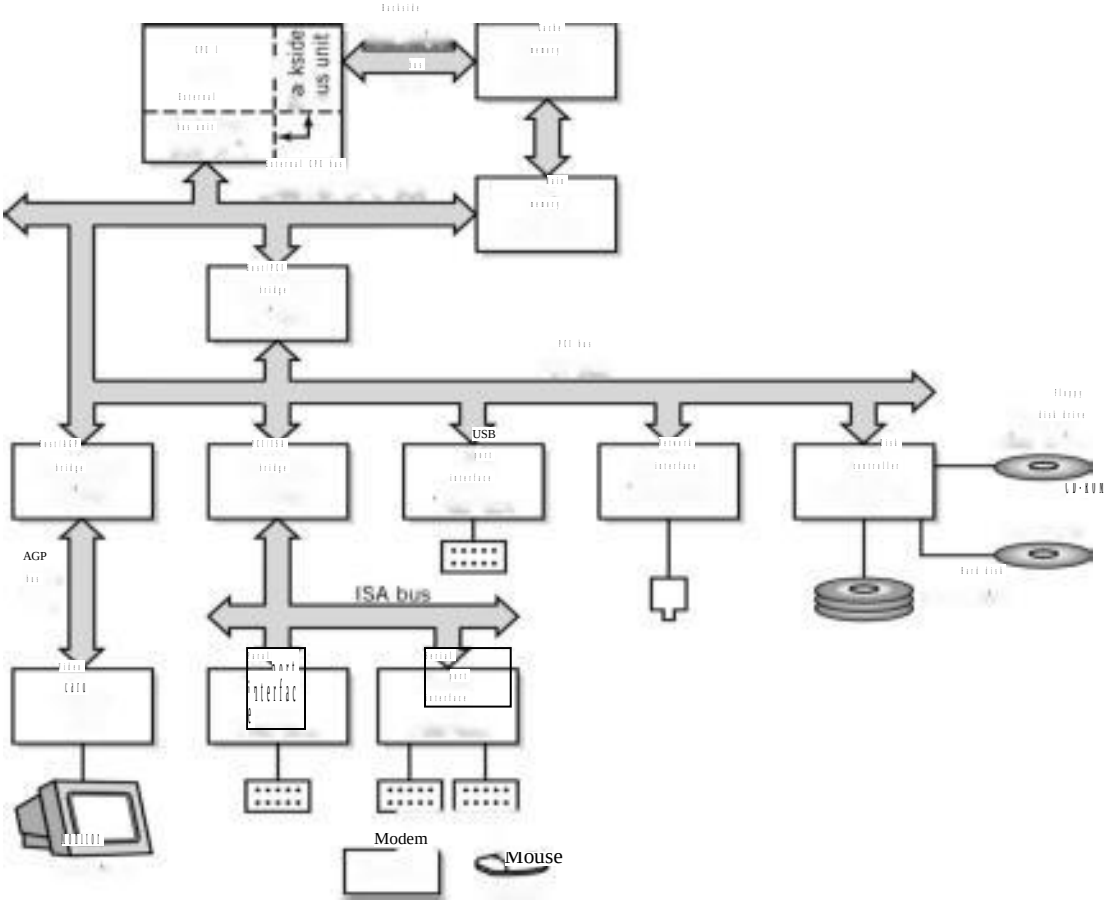
1. PC -> MAR	Transfer the address from the PC to the MAR
2. MDR -> IR	Transfer the instruction to the IR.
3. IR(address) -> MAR	Address portion of the instruction loaded in MAR
4. A-> MDR*	Accumulator copies data into MDR
5. PC + 1 -> PC	Program Counter incremented

- The physical connection that makes it possible to transfer data from one location in the computer system to another
- Group of electrical conductors for carrying signals from one location to another
- *Line:* each conductor in the bus
- 4 kinds of signals
    - Data (alphanumeric, numerical, instructions)
    - Addresses
    - Control signals
    - Power (sometimes)

Bus Characteristics

- Documented agreement for communication
- Specification that spells out the meaning of each line and each signal on each line
- Throughput, i.e., data transfer rate in bits per second > Data width in bits carried simultaneously

PC Interconnections



4. Addressing Modes:

1. Immediate Addressing Mode
2. Direct Addressing Mode
3. Indirect Addressing Mode
4. Register Addressing Mode
5. Register Indirect Addressing Mode
6. Displacement (Indexed) Addressing Mode
7. Stack addressing mode

Immediate Addressing Mode:

Operand (address field) is part of instruction.

E.X. ADD 5^ Add 5 to contents of accumulator

Direct Addressing Mode:

Address field contains address of operand. Effective address (EA) = address field (A). Single memory reference to access data. No additional calculations to work out effective address. Limited address space

E.X: ADD A^Add contents of cell A to accumulator, Look in memory at address A for operand

Indirect Addressing Mode:

Memory cell pointed to by address field contains the address of (pointer to) the operand. EA = (A).In A, it find address (A) and look there for operand. Large address space. EA = (((A))).so it is slower.

E.X: ADD (A) -^Add contents of cell pointed to by contents of A to accumulator

Register Addressing Mode:

Operand is held in register named in address filed. EA = R. Limited number of registers. Very small address field needed. Here the instruction fetching is fast. EA = (R).Operand is in memory cell pointed to by contents of register R. Large address space (2^n).One fewer memory access than indirect addressing.

Displacement Addressing Mode:

EA = A + (R)

Address field hold two values

• A = base value

R = register that holds displacement

Base-Register Addressing Mode:

A holds displacement. R holds pointer to base address. R may be explicit or implicit EA = A + R

A = base, R = displacement

Indexed Addressing Mode:

Types

1. Pre index addressing mode
2. Post index addressing mode

Post index Addressing Mode:

EA = (A) + (R)

Pre index Addressing Mode:

EA = (A+(R))

Stack Addressing Mode:

Operand is (implicitly) on top of stack

EX: ADD^Pop top two items from stack and add

Effective address:

Virtual or effective address is offset into segment. Starting address plus offset gives linear address. This goes through page translation if paging enabled.

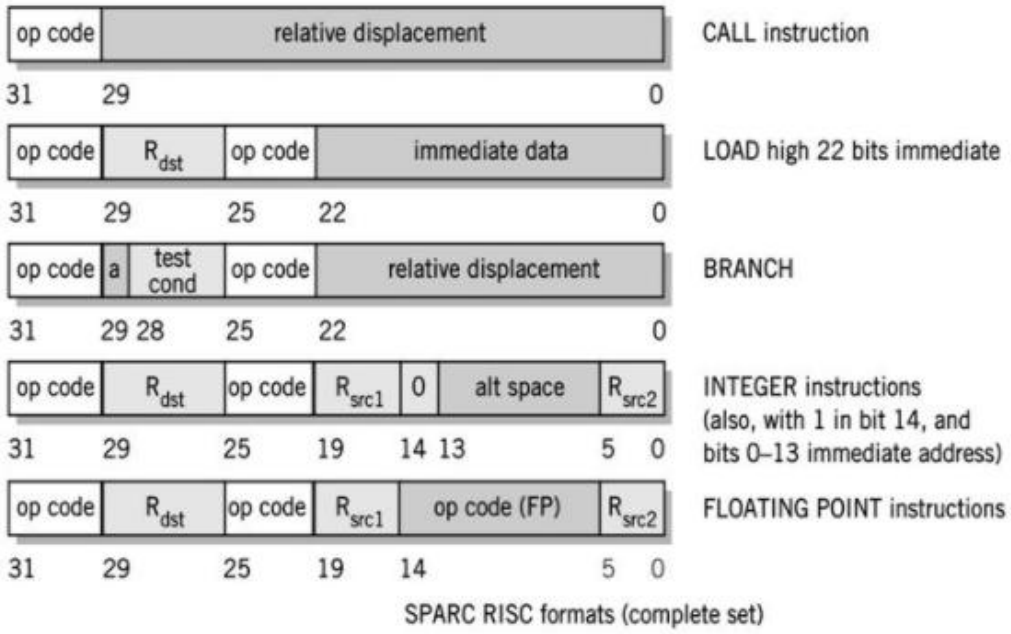
PCI Bus Connections Instructions Set:

It is method of specifying the data is called Instruction.

- Number of instructions
- Complexity of operations performed by individual instructions
- Data types supported
- Format (layout, fixed vs. variable length)
- Use of registers

  Addressing (size, modes)

Instruction Formats: RISC



SPARC RISC formats (complete set)

5.Instruction Types:

Data Transfer (load, store) Instruction:

- Most common, greatest flexibility
- Involve memory and registers

Syntax: Ld/St Indirect DR,BR Displacement (Note:Similarly all the instruction)

Arithmetic Instruction Instruction:

- Integers and floating point
- Operators + - / * [A]

Logical or Boolean Instruction:

- Relational operators: > < =
- Boolean operators AND, OR, XOR, NOR, and NOT • Single operand manipulation instructions
- Negating, decrementing, incrementing

Conditional Register Logical Instruction:

CR Dest Bit, Source Bit

Branch Instruction:

BR conditional options CR bit

BR: Branch

CR : Condition

DR: Destination Reg

SR: Source Reg

Additional Instruction set

- Bit manipulation instructions
- Flags to test for conditions
- Shift and rotate
- Program control
- Stack instructions
- Multiple data instructions
- I/O and machine control

UNIT 5

INTRODUCTION TO EMBEDDED HARDWARE AND SOFTWARE

Terminology-Gates-Timing diagram-Memory-Micro processor buses-Direct memory access- Interrupts-Built interrupts-Interrupts basis-shared data problems-Interrupt latency-Embedded system evolution trends-Interrupt routines in an RTOS environment.

1. Gates:

AND Gate:

Logic diagram:

Truth table:

A	B	Y = AB
0	0	0
0	1	0
1	0	0
1	1	1

NOT GATE: Logic Diagram:

Truth table:

A	Y = A'
0	1
1	0

OR GATE:

Logic Diagram:

Truth table:

A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

NAND GATE:

Logic Diagram

A	B	Y=(AB)'
0	0	1
0	1	1
1	0	1
1	1	0

NOR GATE:

Logic Diagram:

Truth table:

A	B	Y=(A+B)'
0	0	1
0	1	0
1	0	0
1	1	0

XOR GATE:

Logic diagram:

Truth Table:

A	B	Y=A0B
0	0	0
0	1	1
1	0	1
1	1	0

XNOR GATE:

Logic Diagram

Truth table:

A	B	Y=AG>B
0	0	1
0	1	0
1	0	0
1	1	1

Timing diagram

- Graphical representation of circuit behavior over time.
- It is used represent device specification and device performance.
- It plays a vital role in module or system specification.
- May be used as a tool in system analysis.
- It represents the clock cycle and duration, delay, content of address bus and data bus, type of operation i.e. Read/write/status signals.
- It is one of the best ways to understand to process of micro-PROCESSOR/controller/Embedded system.
- Easy to understand the working of any system, step by step working of each instruction and its execution.

Instruction Cycle

The time required to complete execution of an instruction is called instruction cycle.

Clock cycle

The speed of a computer PROCESSOR, or CPU, is determined by the clock cycle, which is the amount of time between two pulses of an oscillator, The higher number of pulses per second, will increase the speed of PROCESSOR. The clock speed is measured in Hz, typically either megahertz (MHz) or gigahertz (GHz). For example, a 4GHz PROCESSOR performs 4,000,000,000 clock cycles per second.

Machine cycle

The machine cycle is a 4 process cycle that includes reading and interpreting the machine language, executing the code and then storing that code. The four steps involved in Machine cycle are

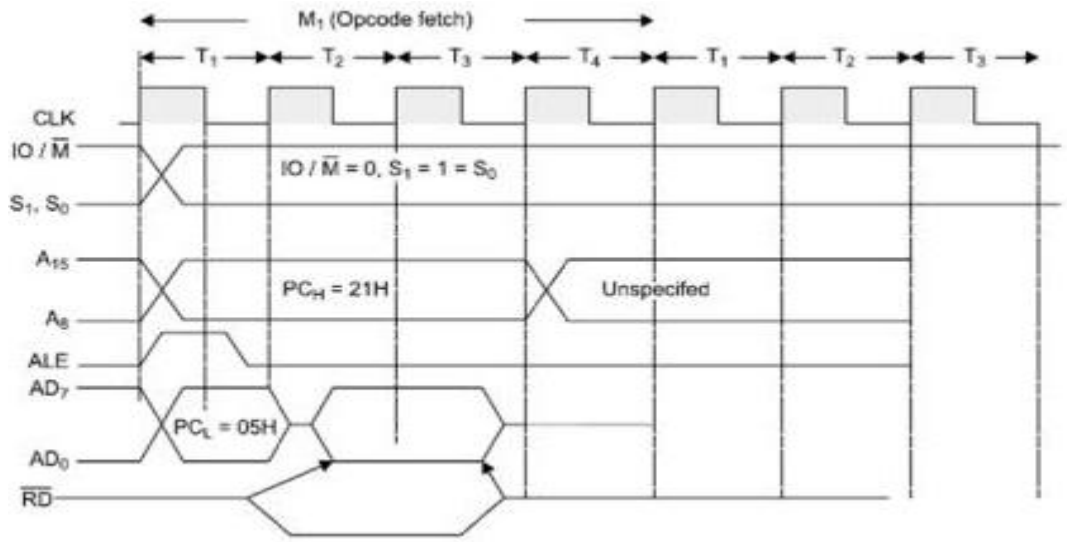
1. Fetch - Retrieve an instruction from the memory.
2. Decode - Translate the retrieved instruction into a series of computer commands.
3. Execute - Execute the computer commands.
4. Store - Read and write the results back in memory.
5. T-state: Each clock cycle is called as T-states.

Rules of machine cycles

1. If an addressing mode is direct, immediate or implicit then No. of machine cycles = No. of bytes.
2. If the addressing mode is indirect then No. of machine cycles = No. of bytes + 1. Address +1 to the No. of machine cycles if it is memory read/write operation.
3. If the operand is 8-bit or 16-bit address then, No. of machine cycles = No. of bytes +1.
4. These rules are applicable to 80% of the instructions of 8085.

Opcode fetch:

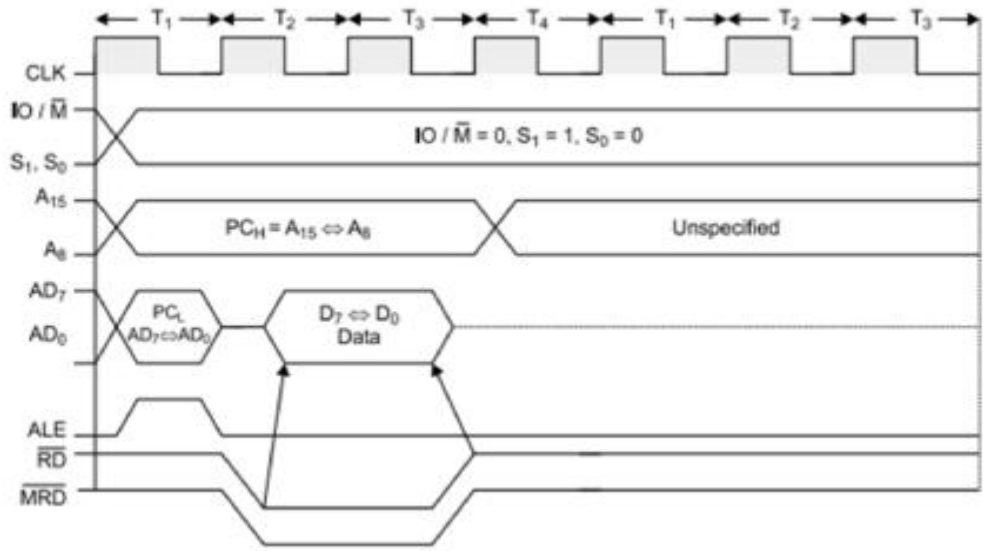
The Micro processor requires instructions to perform any particular action. In order to perform these actions Micro processor utilizes opcode which is a part of an instruction



Operation:

- During T1 state, Micro processor uses IO/M (bar), S0 and S1 signals are used to instruct Micro processor to fetch opcode.
- Thus when IO/M (bar) =0, S0=S1= 1, it indicates opcode fetch operation.
- At T2 state Micro processor uses read signal and make data ready from that memory location to read opcode from memory and at the same time program counter increments by 1 and points next instruction to be fetched.
- In this state Micro processor also checks READY input signal, if this pin is at low logic level i.e. '0' then Micro processor wait state immediately between T2 and T3.
- At T3, Micro processor reads opcode and store it into instruction register to decode it further.
- During T4 Micro processor performs internal operation like decoding opcode and providing necessary actions.
- The opcode is decoded to know whether T5 or T6 states are required, if they are not required then performs next operation.

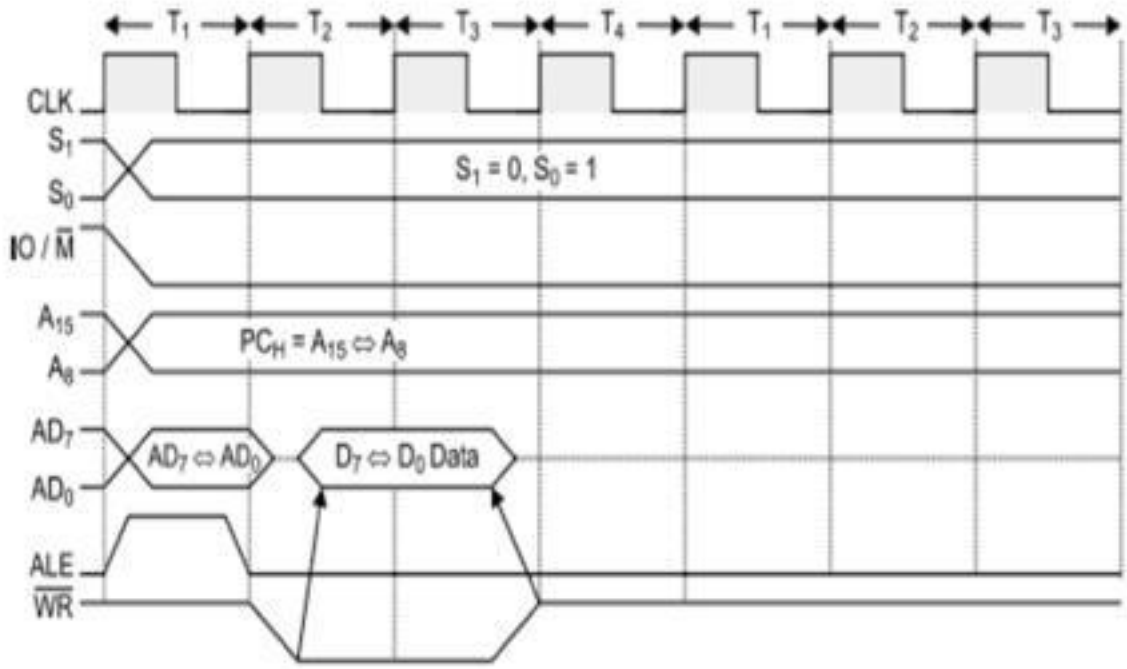
Read and write timing diagram for memory and I/O Operation Memory Read:



Operation:

- It is used to fetch one byte from the memory.
- It requires 3 T-States.
- It can be used to fetch operand or data from the memory.
- During T1, A8-A15 contains higher byte of address. At the same time ALE is high. Therefore Lower byte of address A0-A7 is selected from AD0-AD7.
- Since it is memory ready operation, IO/M (bar) goes low.

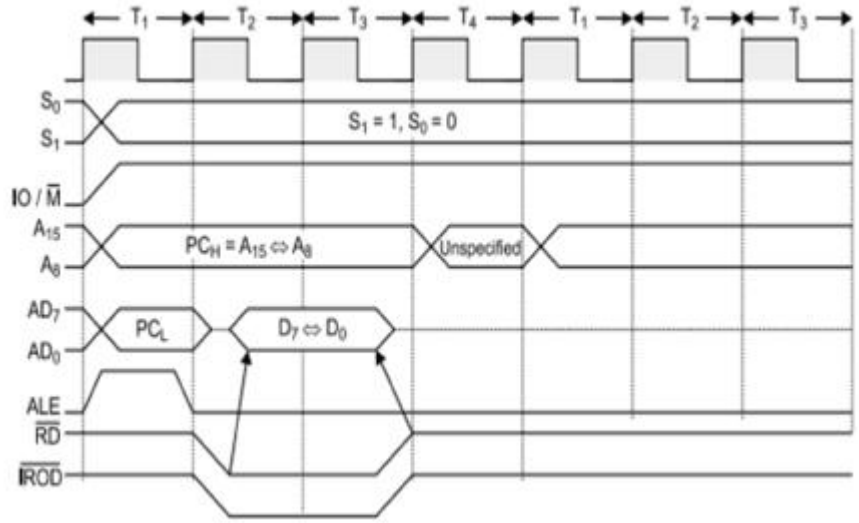
Memory Write:



Operation:

- During T2 ALE goes low, RD (bar) goes low. Address is removed from AD0-AD7 and data D0-D7 appears on AD0-AD7.
- During T3, Data remains on AD0-AD7 till RD (bar) is at low signal.
- It is used to send one byte into memory.
- It requires 3 T-States.
- During T1, ALE is high and contains lower address A0-A7 from AD0-AD7.
- A8-A15 contains higher byte of address.
- As it is memory operation, IO/M (bar) goes low.
- During T2, ALE goes low, WR (bar) goes low and Address is removed from AD0- AD7 and then data appears on AD0-AD7.
- Data remains on AD0-AD7 till WR (bar) is low.

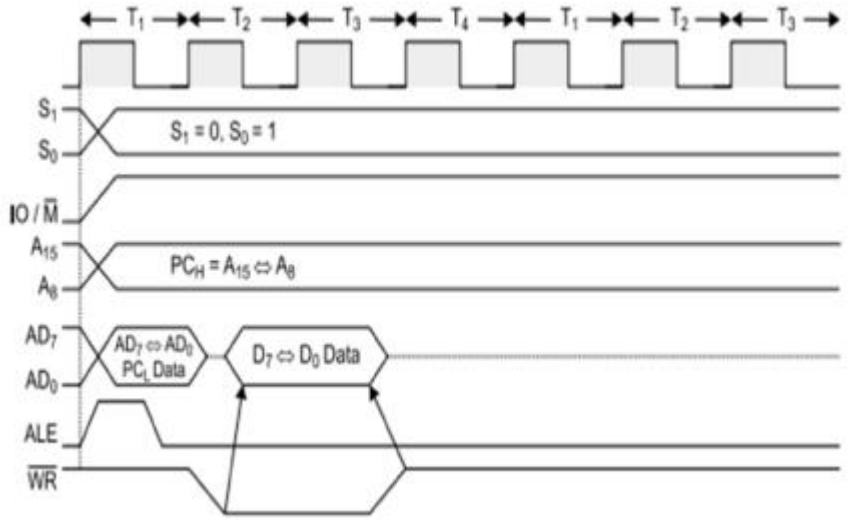
IO Read:



Operation:

- It is used to fetch one byte from an IO port.
- It requires 3 T-States.
- During T1, The Lower Byte of IO address is duplicated into higher order address bus A8- A15.
- ALE is high and AD0-AD7 contains address of IO device.
- IO/M (bar) goes high as it is an IO operation.
- During T2, ALE goes low, RD (bar) goes low and data appears on AD0-AD7 as input from IO device.
- During T3 Data remains on AD0-AD7 till RD (bar) is low.

IO Write:

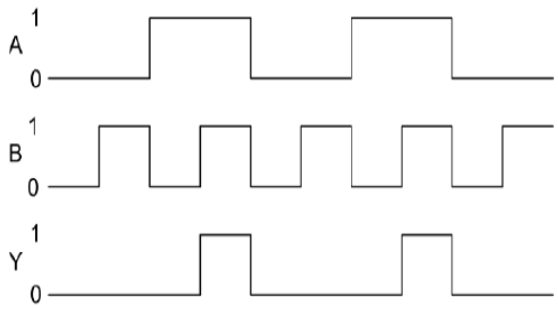


Operation:

- It is used to writ one byte into IO device.
- It requires 3 T-States.
- During T1, the lower byte of address is duplicated into higher order address bus A8- A15.
- ALE is high and A0-A7 address is selected from AD0-AD7.
- As it is an IO operation IO/M (bar) goes low.
- During T2, ALE goes low, WR (bar) goes low and data appears on AD0-AD7 to write data into IO device.
- During T3, Data remains on AD0-AD7 till WR (bar) is low

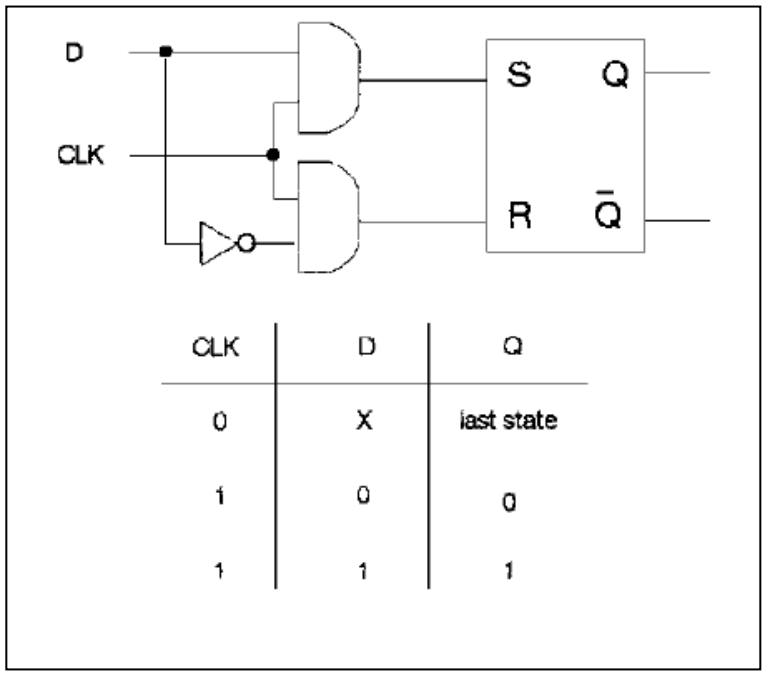
Example

AND Gate:



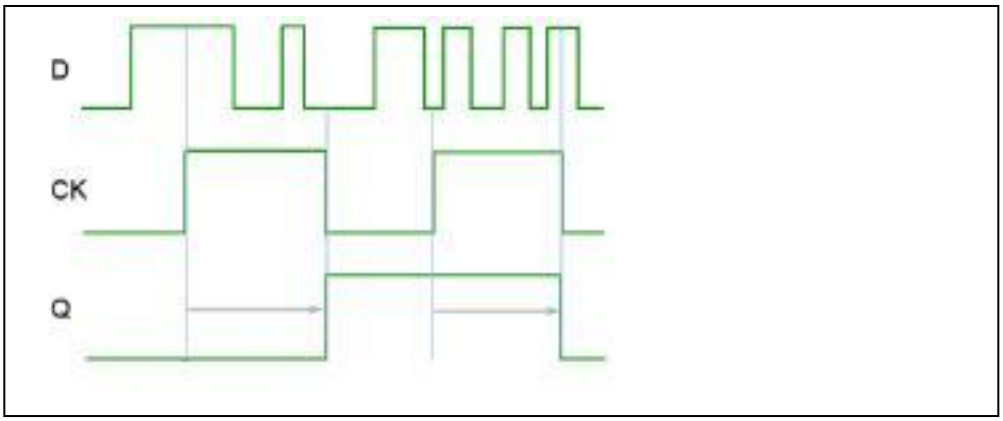
D Flip Flop:

Logic Diagram



CLK	D	Q
0	X	last state
1	0	0
1	1	1

Timing Diagram for D flip flop:



Setup Time:

Setup time is the minimum amount of time the data signal should be held steady before the clock event so that the data are reliably sampled by the clock. This applies to synchronous circuits such as the flip-flop.

Hold Time:

The hold time is the amount of time that the system can continue to run without resetting or rebooting during a power interruption.

Propagation Delay:

- The delay between an input transition and the resultant output transition is called propagation delay
- Propagation delay is measured between the centers of the transition time period
- Propagation delay for an output transition from high to low is t_{pHL}

- Propagation delay for an output transition from low to high is t_{pLH}
- Notice that these two times are not necessarily the same for the same device
- Notice the difference from Transition Times Timing Hazard
- Transient output behavior may not agree with predicted output due to delay differences
- A glitch is the presence of extra signal transitions which are not predicted from the logic equations

Static Hazards

- A static hazard is the possibility of a glitch when the output should not change
- A static-1 hazard is present if changing a single input variable may produce output transitions but the output logic function is high (1) independent of this change.

Detection of Static-1 Hazards

- A properly designed OR-AND circuit will never have static-1 hazards.
- An AND-OR circuit may have static-1 hazards.
- These can be detected from a K-map.
- Check for adjacent min terms NOT covered by the same product term (AND gate) in the actual realization.

Static-0 hazard

- A static-0 hazard is present if changing a single input variable may produce output transitions but the output logic function is low (0) independent of this change.

  While hazards produce unexpected transitions and may be intermittent, proper design of synchronous sequential circuits should tolerate hazards.

Detection of Static-0 Hazards

- A properly designed AND-OR circuit will never have static-0 hazards
- An OR-AND circuit may have static-0 hazards
- Check K-map for adjacent max terms NOT covered by the same sum term (OR gate) in the actual realization

Dynamic Hazards

- the output is supposed to change, the presence of a glitch means the output makes more than one transition

  - low ^ high ^ low ^ high
  - high ^ low ^ high ^ low
- Do not occur in properly designed two-level circuits

3. Memory:

It is one or more sets of chips that store data/program instructions, either temporarily or permanently. It is critical processing component in any computer.

(OR)

Any sequential circuit has memory of a sort, since each flip-flop or latch stores one bit of information. The term memory is usually reserved for that part of a system where bits of information are stored in a structured way. This is usually a 2-dimensional array in which one row of bits is accessed

RAM

- ROM
- PROM
- EPROM

RAM:

- Volatile

  RAM is packaged as a chip. Basic storage unit is a cell (one bit per cell).

- DRAM and SRAM are volatile memories

- Used for temporary storage
- Typical ranges 256 MB - 4 GB
- Random Access means direct access to any part of memory

The specifications of the RAM

Bus speed: Measured in MHz, is that the volume of data in RAM can be transmitted to the CPU once handled.

Retrieve data speed: Measured in billionths of a second (nanosecond), the time interval between two data acquisition of RAM

Capacity: Measured in MB (megabytes), showing a maximum level of stocks of RAM when the RAM data completely empty.

RAM ECC (Error Correction Code): This is a technique to test and debug in case of a bit of memory was incorrect value data during transport.

Bandwidth of RAM Single Channel

Band Width = Bus Speed *8 Ex: DDR-SDRAM 400 MHZ Band Width = 400 *8 = 3200MB/s.

Dual Channel

Band Width = Bus Speed * 2*8 = Bus Speed * 16 Ex: DDR 400 MHz => Bandwidth = 400*16 = 6400

SRAM (Static RAM)

- Memory behaves like latches or flip-flops
- Address must be stable before writing cell.
- Data must be stable before ending a write.

  SRAMs typically use six transistors per bit of storage.
- Allow faster access than DRAM.
- The memory chips are made of the transistors (switches) and capacitors.
- Transistor SRAM can hold state power.
- SRAM is more expensive than DRAM DRAM (Dynamic RAM)
- Memory lasts only for a few milliseconds
- Must refresh locations by reading or writing

- DRAMs use only one transistor per bit
- 1/0 = capacitor charged/discharged
- DRAM is a type of RAM should regularly update the data (high refresh rate).
- Dynamic RAM is mounted on the modules: DIMM, SIMM or RIMM.

  SIMM - Single In-Line Memory Module DIMM - Dual Inline Memory Module RIMM - RAM bus DRAM
- Plugged directly into the motherboard

ROM:

- Nonvolatile Memories (ROM)
- Lose information if powered off.
- Nonvolatile memories retain value even if powered off.
- ROM is memory but it is also a combination element.
- As a combination element it can perform logic functions.
- It is used to produce chips with integrated CMOS BIOS program

PROM (Programmable read only memory):

It is otherwise called as field programmable read-only memory (FPROM) or onetime programmable non-volatile memory (OTP NVM) is a form of digital memory where the setting of each bit is locked by a fuse or antifuse.

- Type of ROM that information is only installed once.
- The CD can be called PROM.

EPROM (Erasable Programmable ROM):

It is programmable read-only memory (programmable ROM) that can be erased and re-used. Erasure is caused by shining an intense ultraviolet light through a window that is designed into the memory chip.

- Type of ROM that can erase and rewrite it
- CD-Erasable can be called EPROM

EEPROM (Electronic Erasable Programmable ROM):

It form enhance of EPROM, a difference compared to the EPROM is able to write and remove the information again and again by software rather than hardware. Application specific EEPROM is "flash BIOS". ROM is the type of information can install or upgrade software.

Example: CD-Rewritable.

4. Micro processor Buses:

- A group of digital signals which carry multi-bit data
- With a single (or double) line representation in block diagram
- Shown on a logic diagram (schematic) with a single (sometimes heavier) line
- Usually named using an indexed notation:

Data Bus:

Data bus carries data in binary form between Micro processor and other external units such as memory. It is used to transmit data i.e. information, results of arithmetic etc between memory and the Micro processor. Data bus is bidirectional in nature

Address Bus:

The address bus carries addresses and is one way bus from Micro processor to the memory or other devices.

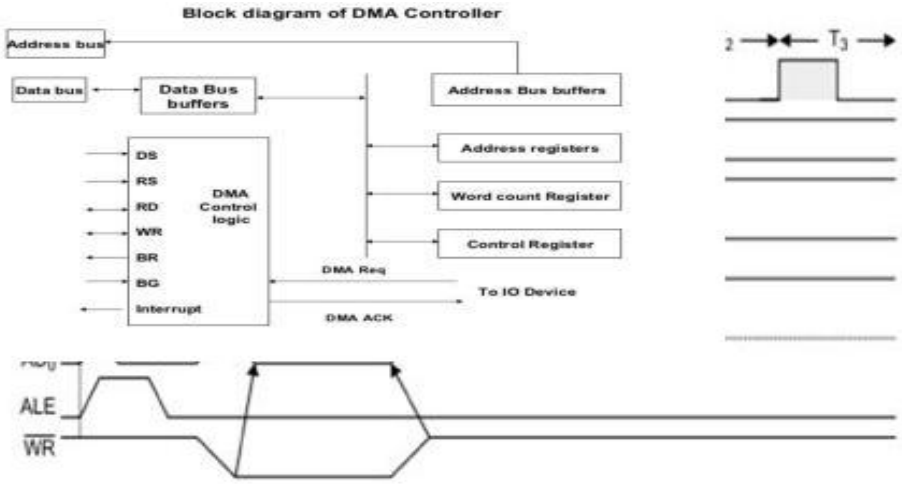
Control Bus:

Control buses are various lines which have specific functions for coordinating and controlling Micro processor operations. The control bus carries control signals partly unidirectional and partly bidirectional.

DIRECT MEMORY ACCESS

A DMA is required when a multi-byte data set or a burst of data or a block of data is to be transferred between the external device and system or two systems. A device facilitates DMA transfer with a processing element (single purpose PROCESSOR) and that device is called DMAC (DMA Controller). DMA based method useful, when a block of bytes are transferred, for example, from disk to the RAM or RAM to the disk.

Repeatedly interrupting the PROCESSOR for transfer of every byte during bulk transfer of data will waste too much of PROCESSOR time in context switching. System performance improves by separate processing of the transfers from and to the peripherals (for example, between camera memory and USB port).



Consider a system consisting of a CPU, memory and one or more input/output devices. Assume one of the I/O devices is a disk drive and that the computer must load a program from this drive into memory. The CPU would read the first byte of the program and then write that byte to memory. Then it would do the same for the second byte, until it had loaded the entire program into memory.

- A DMA controller implements direct memory access in a computer system.
- It connects directly to the I/O device at one end and to the system buses at the other end. It also interacts with the CPU, both via the system buses and two new direct connections.
- It is sometimes referred to as a channel. In an alternate configuration, the DMA controller may be incorporated directly into the I/O device.

- To transfer data from an I/O device to memory, the DMA controller first sends a Bus Request to the CPU by setting BR to 1. When it is ready to grant this request, the CPU sets its Bus grant signal, BG to 1.
- The CPU also tri-states it's address, data, and control lines thus truly granting control of the system buses to the DMA controller.

    - The CPU will continue to tri-state its outputs as long as BR is asserted.
- The DMA controller includes several registers :-
    - The DMA Address Register contains the memory address to be used in the data transfer. The CPU treats this signal as one or more output ports.
    - The DMA Count Register, also called Word Count Register, contains the no. of bytes of data to be transferred. Like the DMA address register, it too is treated as an O/P port (with a diff. Address) by the CPU.
    - The DMA Control Register accepts commands from the CPU. It is also treated as an O/P port by the CPU.

Process of DMA Transfer

- To initiate a DMA transfer, the CPU loads the address of the first memory location of the memory block (to be read or written from) into the DMA address register. It does his via an I/O output instruction, such as the OTPT instruction for the relatively simple CPU.
- Then it writes the no. of bytes to be transferred into the DMA count register in the same manner. Finally, it writes one or more

commands to the DMA control register.

- These commands may specify transfer options such as the DMA transfer mode, but should always specify the direction of the transfer, either from I/O to memory or from memory to I/O.
- The last command causes the DMA controller to initiate the transfer. The controller then sets BR to 1 and, once BG becomes 1, seizes control of the system buses.

DMA Transfer Modes

Modes vary by how the DMA controller determines when to transfer data, but the actual data transfer process is the same for all the modes.

BURST mode

- Sometimes called Block Transfer Mode.
- An entire block of data is transferred in one contiguous sequence. Once the DMA controller is granted access to the system buses by the CPU, it transfers all bytes of data in the data block before releasing control of the system buses back to the CPU. This mode is useful for loading programs or data files into memory, but it does render the CPU inactive for relatively long periods of time.

CYCLE STEALING Mode

- The alternative for the systems in which the CPU should not be disabled for the length of time needed for Burst transfer modes.
- DMA controller obtains access to the system buses as in burst mode, using BR & BG signals. However, it transfers one byte of data and then desserts BR, returning control of the system buses to the CPU. It continually issues requests via BR, transferring one byte of data per request, until it has transferred it's entire block of data.

- By continually obtaining and releasing control of the system buses, the DMA controller essentially interleaves instruction & data transfers. The CPU processes an instruction, and then the DMA controller transfers a data value, and so on.
- The data block is not transferred as quickly as in burst mode, but the CPU is not idled for as long as in that mode.
- Useful for controllers monitoring data in real time.

Advantages of DMA

- Computer system performance is improved by direct transfer of data between memory and I/O devices, bypassing the CPU.
- CPU is free to perform operations that do not use system buses.

Disadvantages of DMA

- In case of Burst Mode data transfer, the CPU is rendered inactive for relatively long periods of time.

6. Interrupts

Interrupt means event, which invites attention of the PROCESSOR on occurrence of some action at hardware or software interrupt instruction event. In response to the interrupt, the routine or program, which is running presently interrupts and an interrupt service routine (ISR) executes.

Interrupt Service Routines

ISR is also called device driver in case of the devices and called exception or signal or trap handler in case of software interrupts

- Copy peripheral data into a buffer
- Indicate to other code that data has arrived

Acknowledge the interrupt (tell hardware)

Longer reaction to interrupt performed outside interrupt routine

E.g: causes a process to start or resume running

Priority-based Scheduling

- Typical RTOS based on fixed-priority preemptive scheduler
- Assign each process a priority
- At any time, scheduler runs highest priority process ready to run
- Process runs to completion unless preempted Priority-Based Preempting Scheduling

- Multiple processes at the same priority level
- Simply prohibit: Each process has unique priority
- Time-slice processes at the same priority
- Extra context-switch overhead
- No starvation dangers at that level
- Processes at the same priority never preempt the other
- More efficient
- Priority Inheritance
- Temporarily increase process's priority when it acquires a lock
- Level to increase: highest priority of any process that might want to acquire same lock i.e., high enough to prevent it from being preempted

- Danger: Low-priority process acquires lock, gets high priority and hogs the PROCESSOR.
- Low-priority processes should acquire high-priority locks.
- No equivalent result when locks and priority inheritance is used.

Interrupt approach for the port or device Interrupt approach for the port or device functions

PROCESSOR executes the program, called interrupt service routine or signal handler or trap handler or exception handler or device driver, related to input or output from the port or device or related to a device function on an interrupt and does not wait and look for the input ready or output completion or device-status ready or set

Hardware interrupt

Examples — When a device or port is ready, a device or port generates an interrupt, or when it completes the assigned action or when a timer overflows or when a time at the timer equals a preset time in a compare register or on setting a status flag (for example, on timer overflow or compare or capture of time) or on click of mice in a computer , Hardware interrupt generates call to an ISR

Software Interrupt:

When software run-time exception condition (for examples, division by 0 or overflow or illegal opcode detected) the PROCESSOR-hardware generates an interrupt, called trap, which calls an ISR. When software run-time exception condition defined in a program occurs, and then a software

instruction (SWI) is executed, then it is called software interrupt or exception or signal, which calls an ISR .When a device function is to be invoked, for example, open (initialize/configure) or read or write or close, then a software instruction (SWI) is executed, then it is called software interrupt to execute the required device driver function for open or read or write or close operations.

Software can execute the software instruction (SWI) or Interrupt (INT n) to signal execution of ISR (interrupt service routine). The n is as per the handler address. Signal interrupt [The signal differs from the function in the sense that execution of signal handler (ISR) can be masked and till mask is reset, the handler will not execute on interrupt. Function on the other hand always executes on the call after a call-instruction].

7. Shared Data problem:

Inconsistency in data used by a task and updated by an ISR; arises because ISR runs at just the wrong time. Data is often shared because it is undesirable to have ISRs do all the work - they would take too long to run. ISRs typically -hand off some of the processing to task code. This implies shared variables or communication between the ISR and the related task. This problem arises when task code accesses shared data non-atomically. Some data is common to different processes or tasks. Examples are as follows: Time, which is updated continuously by a process, is also used by display process in a system Port input data, which is received by one process and further processed and analyzed by another process. Memory Buffer data which is inserted by one process and further read (deleted),

processed and analyzed by another process. Assume that at an instant when the value of variable operates and during the operations on it, only a part of the operation is completed and another part remains incomplete. At that moment, assume that there is an interrupt. Assume that there is another function. It also shares the same variable. The value of the variable may differ from the one expected if the earlier operation had been completed. Whenever another process sharing the same partly operated data, and then shared data problem arises.

Solution:

Put a shared variable in a circular queue. A function that requires the value of this variable always deletes (takes) it from the queue front, and another function, which inserts (writes) the value of this variable, always does so at the queue back. Disable the interrupts (DI) before a critical section starts executing and enable the interrupts (EI) on its completion. DI— powerful but a drastic option. An interrupt, even if of higher priority than the critical function, gets disabled.

Use lock ( ) a critical section starts executing and use unlock ( ) on its completion. A software designer usually not use the drastic option of disabling interrupts in all the critical sections, except in automobile system like software. Use IPC (Inter-Process Communication).

9. Interrupt Latency

In computing, interrupt latency is the time that elapses from when an interrupt is generated to when the source of the interrupt is serviced. For many operating systems, devices are serviced as soon as the device's interrupt handler is executed.

10. Embedded system

A combination of hardware and software which together form a component of a larger machine. An example of an embedded system is a Micro processor that controls an automobile engine. An embedded system is designed to run on its own without human intervention, and may be required to respond to events in real time.

Application

TV, Stereo, Remote control, Phone / mobile phone, Refrigerator, Microwave, Washing machine, Electric tooth brush, Oven / rice or Bread cooker, Watch, Alarm clock, Electronic musical instrument, Electronic toys ,Medical home equipment like blood pressure, thermometer.

11. Real-Time Operating Systems

A variant of OS that operates in constrained environment in which computer memory and processing power is limited. Moreover they often need to provide their services in definite amount of time. Hard, Soft & Firm RTOS.

A variant of OS that operates in constrained environment in which computer memory and processing power is limited. Moreover they often need to provide their services in definite amount of time. Hard, Soft & Firm RTOS.

Example: Flight Control System

Definition:

Real-time systems have been defined as those systems in which the correctness of the system depends not only on the logical result of the computation, but also on the time at which the results are produced.

Role of an OS in real time systems

- Standalone Applications
- Often no OS involved
- Micro controller based Embedded Systems
- Some Real Time Applications are huge & complex
- Multiple threads
- Complicated Synchronization Requirements
- File system / Network / Windowing support
- OS primitives reduce the software design time

Features of RTOS

- Scheduling
- Resource Allocation
- Interrupt Handling
- Other issues like kernel size

Hard Versus Soft RTOS

Hard real time means strict about adherence to each task deadline. When an event occurs, it should be serviced with in the predictable time at all times in a given hard real time system. The preemption period for the hard real time task in worst case should be less than a few seconds. A hard RTOS is one, which has predictable performance with no deadline, even in case of sporadic tasks (sudden bursts of occurrence of events requiring attention). Automobile engine control system and anti lock brake are the examples of hard real time systems. Provision of asynchronous IOs. Provision of locks or spin locks. Predictions of interrupt latencies and Context switching latencies of the tasks. Predictability is achieved by

writing all functions which execute always take the same predefined time intervals in case of varying rates of occurrences of the events.

Response in all the time slots for the given events in the system and thus providing the guaranteed task deadlines even in case of sporadic and periodic tasks. Sporadic tasks means tasks executed on the sudden-bursts of the corresponding events at high rates, and periodic tasks mean task having no definite period of event occurrence. Video transmission, each picture frame and audio must be transferred at fixed rate. One in which deadlines are mostly met. Soft real time means that only the precedence and sequence for the task operations are defined, interrupt latencies and context switching latencies are small but there can be few deviations between expected latencies of the tasks and observed time constraints and a few deadline misses are accepted.

The preemption period for the soft real time task in worst case may be about a few ms. Mobile phones, digital cameras and orchestra playing robots are examples of soft real time systems.

- A real time OS (function as standard OS, with predictable behavior and well-defined functionality)
- A collection of RT tasks/processes (share resources, communicate/synchronize with each other and the environment)
- Large and complex: Vary from a few hundred lines to 20 million lines estimated for the Space Station Freedom

- Concurrent control of separate system components devices operate in parallel in the real- world, better to model this parallelism by concurrent entities in the program
- Facilities to interact with special purpose hardware need to be able to program devices in a reliable and abstract way
    - Mixture of Hardware/Software: some modules implemented in hardware, even whole systems, SoC.
    - Extreme reliability and safety Embedded systems typically control the environment in which they operate: failure to control can result in loss of life, damage to environment or economic loss
    - Guaranteed response times: we need to be able to predict with confidence the worst case response times for systems; efficiency is important but predictability is essential

Challenges in RT Systems Design

- Predictability: able to predict the future consequences of current actions
- Testability: easy to test if the system can meet all the deadlines
- Cost optimality: e.g. Energy consumption, memory blocks etc

Main properties of RT Systems

Maintainability: Modular structure to ease system modification

Robustness: Must not collapse when subject to peak load, exception, manage all possible scenarios

Fault tolerance: Hardware and software failures should not cause the system to

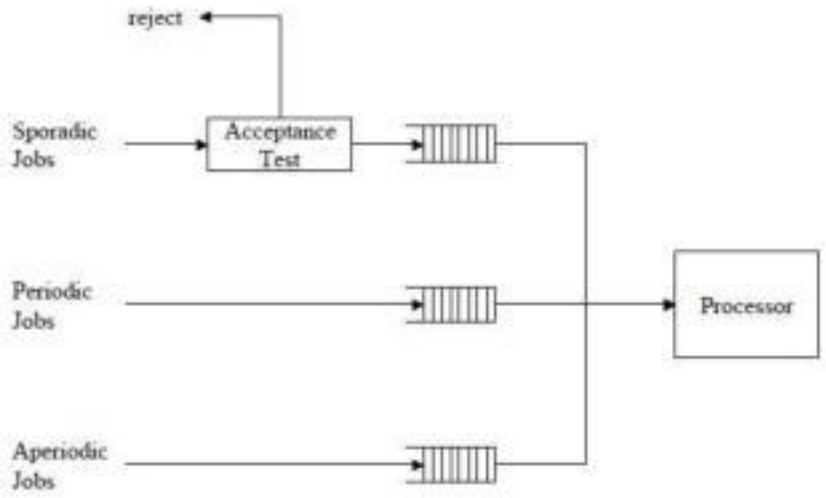
crash - function down-grading

Performance Metrics

- Miss rate: percentage of jobs executed but completed late
- Loss rate: percentage of jobs discarded
- Invalid rate: sum of *miss* and *loss* rate.
- Makespan: If all jobs have same release time and deadline, then make span = response time of the last job to execute.
- Max or average response times
  Max or average tardiness/lateness

Scheduling Aperiodic and Sporadic Jobs

- Polling Server
- Deferrable Server
- Sporadic Server
- Generalized PROCESSOR Sharing
- Constant Utilization Server
- Preemptive Weighted Fair Queuing
- sporadic job: acceptance test
- scheduling of accepted job



Priority Queues for Periodic/Sporadic/Aperiodic Jobs

- Aperiodic job: schedule job to complete ASAP

Correctness

- A correct schedule is one where all periodic tasks, and any sporadic tasks that have been *accepted,* meet their deadlines
- A scheduling algorithm supporting aperiodic and/or sporadic jobs is a correct algorithm if it only produces correct schedules for the system

Optimality

- An aperiodic job scheduling algorithm is optimal if it minimizes either:
- The response time of the job at the head of the aperiodic job queue.
- The average response time of all aperiodic jobs for a given queuing discipline

- A sporadic job scheduling algorithm is optimal if it accepts a new sporadic job, and schedules that job to complete by its deadline, if and only if the new job can be correctly scheduled to complete in time
- An optimal algorithm always produces a feasible schedule job accepted
- Note: this is different from the definition of optimal on-line algorithms discussed previously, as that definition required that *all* offered jobs had to be accepted and completed in time

Scheduling Aperiodic Jobs

- Consider the simple case: scheduling periodic jobs along with a system of periodic jobs
- Ignore sporadic jobs for now
- Two simple approaches:
- Execute the a periodic jobs in the background
- Execute the a periodic jobs by interrupting the periodic jobs Advantages
- Clearly produces correct schedules
- Extremely simple to implement.

Disadvantages

Not optimal since it is almost guaranteed to delay execution of periodic jobs in favour of periodic and sporadic jobs, giving unduly long response times for the aperiodic jobs.