Objectives

1. Describe the Intel family of microprocessors.
2. Explain the function of the Execution Unit and BIU.
3. Describe pipelining and how it enables the CPU to work faster.
4. Explain the list of registers of 8086 and this operation.

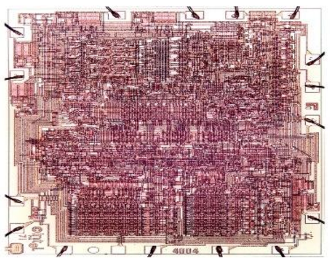
Introduction

Webster's New Twentieth Century Dictionary defines microprocessor as "the controlling unit of a microprocessor, laid out on a tiny chip and containing the logical elements for handling data, performing calculations, carrying out stored instructions, etc."

Intel has brought out a series of microprocessors. The characteristic of each starting from 4-bit microprocessor to 64 bits microprocessor is listed below. For each processor note the number of transistors used, the clock speed, number of bits as well as the pin package has been indicated. There are many other features which will be discussed in detail as we go through the course.

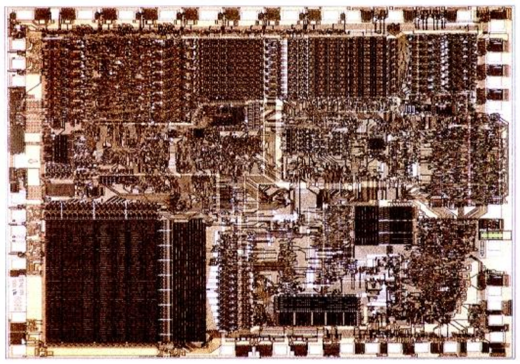
Intel brought out the first 4 bit processor (1971)

- 2300 transistors
- 400 – 800 kHz
- 4-bit word size
- 16-pin DIP package


8-bit layout

8-bit processor (1974-1976) 8085 Microprocessor

- 4.5k – 6.5k transistors
- NMOS Technology
- 5-10 MHz
- 8-bit word size
- 40-pin DIP package
- No. of instructions 111/113
- 64K memory
- Address lines 16
- Data lines 8
- Microcode ROM


16-bit layout

16-bit processor (1978-1979) 8086/8088 Microprocessor

- 3 µm process
- NMOS Technology
- 29k transistors
- 5-10 MHz
- 133 Instructions
- 16-bit word size
- 40-pin DIP package
- 1 M Bytes Physical Memory
- Microcode ROM

Difference between 8085 and 8086

No	Description	8085	8086
1	Address lines	16	20
2	Memory	64K	1MB
3	Data Bus	8	16
4	Co-processor	-	8087
5	Operating system	CP/M	DOS

CP/M, originally standing for Control Program/Monitor and later Control Program for Microcomputers, is a mass-market operating system created for Intel 8080/85-based microcomputers by Gary Kildall of Digital Research, Inc.

Difference between 8086 and 8088 Processor

No	Description	8088	8086
1	Address lines	20	20
2	Memory	1MB	1MB
3	Data Bus	8	16
4	Co-processor	8087	8087
5	Operating system	DOS	OS/2

80286 Microprocessor
Virtual memory (1982)

- 1.5 µm process
- NMOS Technology
- 134k transistors
- 10-16 MHz
- 16-bit word size
- 16M Physical Memory
- 68-pin PGA(pin grid array)
- Bit-slices clearly visible

Difference between 8086 and 80286

No	Description	8086	80286
1	Address lines	20	24
2	No. of Pins	40	68
3	Virtual Memory	Nil	1G

4	Physical memory	1M	16M
5	Protective mode	Nil	16MB

80386 Microprocessor
32-bit processor (1985): CMOS Tech

- 1.5-1 µm process
- 275k transistors
- 16-33 MHz
- 32-bit word size
- 132-pin PGA
- 32 Address lines
- 32-bit data-path,
- 64 T Bytes Virtual Memory
- Synthesized control

80486 Microprocessor

- Pipelining (1989)
    - CMOS Tech.
    - Floating point unit
    - 8 KB cache
    - 168 pins
    - 1-0.6 µm process
    - 1.2M transistors
    - 25-33 MHz
    - 32-bit word size
    - Cache, Integer data path,
    - FPU, microcode,
    - synthesized control
    - 168-pin PGA
    - 64 T Bytes Virtual memory

Pentium Processor

- Superscalar (1993)
    - 2 instructions per cycle
    - Separate 8KB Cache I$ & D$
    - 0.8-0.35 mm process
    - 3.2M transistors
    - 60-300 MHz
    - 32-bit word size
    - 296-pin PGA
    - 4G Bytes Physical memory.
    - 64T Bytes Virtual memory
    - Caches, data-path,
    - FPU, control

Pentium Pro / II / III

- Dynamic execution (1995-99)
    - 3 micro-ops / cycle
    - Out of order execution
    - 16-32 KB Cache I$ & D$
    - Multimedia instructions
    - PIII adds 256+ KB L2$ Cache

- Dynamic execution (1995-99)
- 3 micro-ops / cycle
- Out of order execution
- 16-32 KB Cache I$ & D$
- Multimedia instructions
- PIII adds 256+ KB L2$ Cache

Pentium 4

- ❑ Deep pipeline (2001)
    - ❑ Very fast clock
    - ❑ 256-1024 KB L2$
- ❑ Characteristics
    - ❑ 180 – 90 nm process
    - ❑ 42-125M transistors
    - ❑ 1.4-3.4 GHz
    - ❑ 32-bit word size
    - ❑ 478-pin PGA
- ❑ Units start to become invisible on this scale

8086 Microprocessor

The Intel 8086 μp is a 16-bit microprocessor intended to be used as a CPU in a microcomputer. The term "16-bits" means that its arithmetic logic unit, internal registers, and most of its instructions are designed to work with 16-bit binary words. The 8086 have a 16-bit data bus, so that it can read data from or write data to memory and ports (16-bits or 8-bits at a time).

The 8086 contain approximately 29000 transistors and is fabricated using the HMOS technology. The 8086 can be operated at 3 different clock speeds. The standard 8086 runs at 5Mhz and the other versions of the 8086, the 8086-2 and 8086 -1 permit a clock frequencies of up to 8Mhz and 10Mhz respectively.

8086/8088 - CPU Architecture

The 8086/8088 architecture can be broadly divided into two groups:

(i) Execution Unit (EU)
(ii) Bus Interface Unit (BIU)

The execution unit contains the Data and Address registers, the Arithmetic and Logic Unit and the Control Unit. The Bus Interface Unit contains Bus Interface Logic, Segment registers, Memory addressing logic and a Six byte instruction object code queue (4-byte instruction object-code queue in case of 8088 microprocessor).

The execution unit and the Bus Interface unit operate asynchronously. The EU waits for the instruction object code to be fetched from the memory by the BIU.

The BIU fetches or pre-fetches the object code (16-bits at a time) and loads it into the six bytes queue. Whenever the EU is ready to execute a new instruction, it fetches the instruction object code from the front of the instruction queue and executes the instruction in specified number of clock periods.

If memory or Input/output devices must be accessed in the course of executing an instruction, then the EU informs the BIU of its needs. The BIU completes its operation code (opcode) fetch cycle, if in progress, and executes an appropriate external access machine cycle in response to the EU demand.

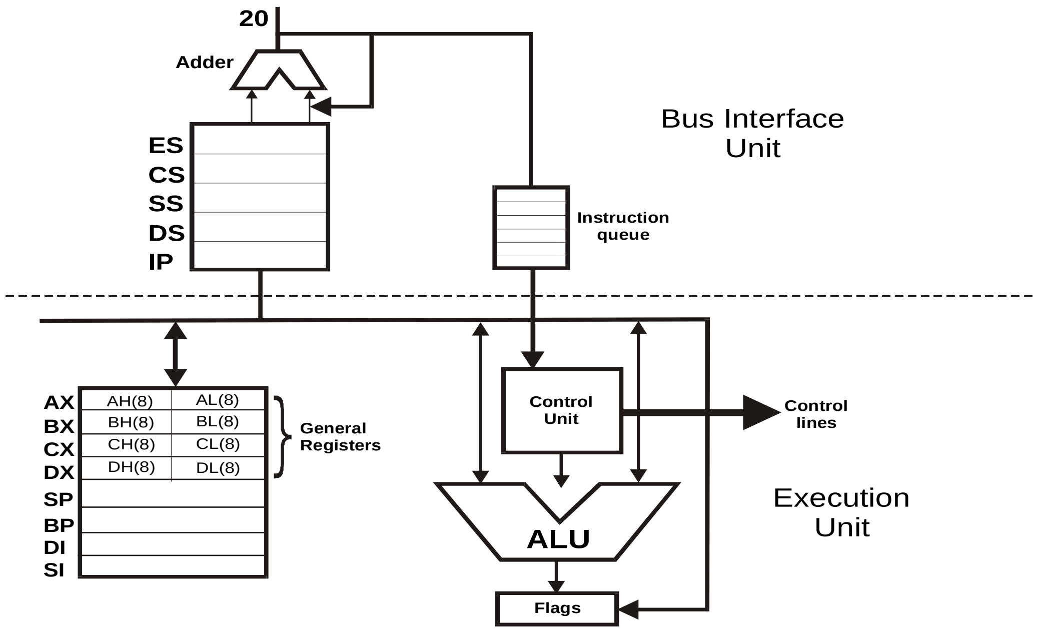


Fig 1.3 8086 Architecture

The BIU is independent of the EU and attempts to keep the six-byte queue filled with instruction object codes. If two or more of these six bytes are empty, then the BIU executes instruction fetch machine cycles as long as the EU does not have an active request for the bus access pending. If the EU issues a request for the bus access while the BIU is in the middle of an instruction fetch machine cycle, then the BIU will complete the instruction fetch machine cycle before honoring the EU bus access request.
The EU does not use machine cycles; it executes instructions in some number of clock periods that are not subjected to any type of machine cycles. The only time clock periods are grouped is clock when the bus control logic wishes to access memory or I/O devices.

Execution Unit (EU)

The execution unit consists of

(i) General Registers
(ii) Arithmetic Logic Unit
(iii) Control unit
(iv) Flag Registers

General Registers

The CPU has eight 16-bit general registers. They are divided into two files of four registers each. They are:

(a) The data register file and
(b) The pointer and index register file

AH	**AL**	**AX**
BH	**BL**	**BX**
CH	**CL**	**CX**
DH	**DL**	**DX**

Fig. 1.4 Data Register File

AX, BX, CX and DX registers are the data registers. The upper and lower halves of the data registers are individually addressable. AX register can be addressed as AL and AH registers, BX register can be addressed as BL and BH register, CX register can be addressed as CL and CH register, DX register can be addressed as DL and DH.

The data registers can be used in most arithmetic and logic operations. Some instructions however require these registers for specific use. This implicit register usage allows a more compact instruction encoding. Fig.1.4 shows the data registers specific one. The index register file consists of the Stack Pointer (SP), the Base Pointer (BP), Source Index (SI) and Destination Index (DI) registers all are of 16-bits. They can also be used in most arithmetic and logic operations. These registers are usually used to hold offset addresses for addressing within a segment. Offset addressing reduces program size by eliminating the need for each instruction to specify frequently used addresses.

The pointer and index register files are further divided into the pointer sub-file (containing the Stack Pointer and the Base Pointer registers) and the index sub-file (containing the Source index and Destination index registers). The Pointer registers are used to access the current stack segment. The index registers are used to access the current data. (Stack segment and data segment are specific areas of memory. Their application will be explained in later chapters). Unless otherwise specified in the instruction, stack pointer registers refer to the current stack segment while index register refers to the current data segment.

The BP and SP registers are both used to point to the stack, a linear array in the memory used for subroutine parameters, subroutine return addresses, and the data temporarily saved during execution of a program

The implicit register usage is as follows:

AX Register	Word Multiplication Word Division and Word I/O Operation.
AL Register	Byte Multiplication Byte Division Byte I/O Translate, and Decimal Arithmetic
AH Register	Byte Multiplication Byte Division.
BX Register	Base Register Translate
CX Register	String Operations
CL Register	Variable Shift and Rotate
DX Register	Word Multiplication, Word Division, Indirect I/O.

Fig. 1.5

Most microprocessors have a single stack pointer register called the SP. 8086 / 8088 has an additional pointer register called BP register to access the content of stack. While the SP is used similar to the stack pointer in other machine (for pointing to subroutine and interrupt return addresses), the BP register is used to hold an old stack pointer value, or it can mark a place in the subroutine stack independent of the SP register. Using the BP register to mark the stack saves the juggling of a single stack pointer to reference subroutine parameters and addresses.

SI and DI are both 16-bits wide and are used by string manipulation instructions and in building some of the more powerful 8086/8088 data structures and addressing modes. Both the SI and the DI registers have auto incrementing and auto-decrementing capabilities.

1.5.1.2 Arithmetic Logic Unit (ALU)

ALU is 16-bits wide. It can do the following 16-bits arithmetic operations

(i) Addition
(ii) Subtraction
(iii) Multiplication
(iv) Division

Arithmetic operations may be performed on four types of numbers

- ➢ Unsigned binary numbers
- ➢ Signed binary numbers (Integers)
- ➢ Unsigned packed decimal numbers
- ➢ Unsigned unpacked decimal numbers

The ALU can also perform logical operations such as

(i) NOT
(ii) AND
(iii) OR
(iv) EXCLUSIVE OR
(v) TEST

Flag Register

The Execution Unit has a 16-bit flag register which indicates some conditions affected by the execution of an instruction. Some bits of the flag register control certain operations of the EU. The flag register in the EU contains nine active flags shown in fig.1.6 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0

				O	D	I	T	S	Z		AF		PF		CF

Fig, 1.6 Flag Register

Six of the nine flags are used to indicate some condition produced by an instruction. These condition flags are also called status flags of 8086/8088 microprocessor. These are the Carry flag, Parity flag, Auxiliary carry flag, Zero flag, and Sign flag. The other three Control flags are Trap Flag, Direction Flag and Interrupt flag.

Condition Flags

Carry Flag (CF)

This flag will be set to one if the addition of two 16-bit binary numbers produces a carry out of the most significant bit position or if there is a borrow to the MSB after subtraction. This flag is also affected when other arithmetic and logical instruction are executed.

Parity Flag (PF)

This flag is set, if the result of the operation has an even number of 1's (in the lower 8 bits of the result). This flag can be used to check for data transmission error.

Auxiliary Carry Flag (AF)

\This flag is set, when there is a carry out of the lower nibble to the higher nibble or a borrow from the higher nibble to the lower. The auxiliary carry flag is used for decimal adjust operation. The AF flag is of significance only for byte operations duringwhich the lower order byte of the 16-bit word is used.

Zero Flag (Z)
This flag is set when the result of an operation is zero. The flag is reset when theresult is not zero.
Overflow Flag (O)
This flag is set, when an arithmetic overflow occurres. Overflow means that the size of the result exceeded the storage capacity of the destination, and a significant digithas been lost.
Sign flag (S)
This flag is set, when an MSB bit of the result is high after an arithmetic operation. When this flag is set the data in assumed to be negative and when this flag iszero it is assumed to be positive.

Control Flags
Control flags are used to control certain operations of the processor. The application of these flags are different from that of six conditional flags. The conditional flags are set or reset by the EU on the basis of the result of some arithmetic or logic operations. The control flags are deliberately set or reset with specific instructions included in the program.

Trap flag (T)
This is used for single stepping through a program. It is used for debugging theprograms. (Discusses with interrupts).

Interrupt Flag (I)
It is used to allow / prohibit the interruption of a program. When the flag set, it enables the interrupt from INTR. When the flag is reset (0), it disables the interrupt.

Direction Flag (D)
It is used for string instructiion (Discussed with the specific instructions later inthe book). If the direction flag is set, the pointers are decremented else the pointers areincremented.

Bus Interface Unit (BIU)
The BIU sends out addresses, fetches instructions from memory, reads data from memory and ports, and writes data to ports and memory. In other words the BIU handlesall transfers of data and addresses on the buses for the execution unit. The BIU has

1. An instruction queue
2. An Instruction pointer
3. Segment registers
4.

Instruction Queue
To speed up program execution, the BIU fetches as many as 6 insturction bytes ahead of time from memory. The prefetched instruction bytes are held for the EU in a first-in-first-out group of register called a queue. The EU decodes an instruction or executes an instruction which does not require the buses. When the EU is ready for its next instruction, it simply reads the instruction from the queue in the BIU. Fetching thenext instruction while the current instruction executes, is called pipelining.

Note: The 8088 microprocessor has only a 4-byte queue.

Instruction Pointer (IP)

The Instruction Pointer is a 16-bit register. This register is always used as the effective memory address, and is added to the Code segment with a displacement of four bits to obtain the physical address of the opcode. The code segment cannot be changed by the move instruction. The instruction pointer is incremented after each opcode fetchto point to the next instruction.

Segment Registers

The 8086 / 8088 microprocessor has 20-bit address lines. All the registers in 8086 / 8088 are 16-bits in length. Hence to obtain 20-bit addresses from the available 16-bit registers, all 8086 / 8088 memory addresses are computed by summing the contents of a segment register and a effective memory address. The effective memory address is computed via a variety of addressing modes. The process of adding, to obtain 20-bit address is as follows:

The selected segment register contents are shifted-left four bits (i.e., the contents are multiplied by 16 decimal), and then added to the effective memory address to generate the actual physical address output.

Segment Register value z(CS, DS, ES or SS)	x x x x x x x x x x x x x x x xH
Effective Memory Address	y y y y y y y y y y y y y y y yH
Physical Address	wwwwwwwwwwwwwwwwwwwwH

Table 1.1

The table 1.1 shows 16-bits of the segment registers CS, DS, ES or SS displaced by 4-bits to the left. The effective address is calculated depending on the type of addressing mode. The effective address is shown as “yyyyyyyyyyyyyyyy”. The 20-bit physical address “wwwwwwwwwwwwwwwwwwww” is obtained after adding the segment register value and effective address. The physical address is 20-bits wide.

To understand how the segmentation is used, it is required to know the memory structure of the 8086 / 8088 microprocessor.

The memory in an 8086/8088 system is a sequence of up to 2^{20} = one million bytes. A word is any two consecutive bytes in memory (word alignment is not required). Words are stored in memory with the most significant byte at the higher memory address. These bytes are stored sequentially from byte 00000 to byte FFFFF hex.

Programs view memory space as a group of segments defined by the application. A segment is a logical unit of memory that may be up to 64K bytes long. Each segment is made up of contiguous memory locations and is an independent, separately addressable unit. Each segment is assigned a base address, which is its starting location in the memory space. All segments start on 16-bit memory boundaries. Segments may be adjacent, disjoint, partially overlapped, or fully overlapped. It is as shown in fig. 1.7

The segment registers point to the four immediately addressable segments. The four segment registers are

- Code Segment register [points to the instruction opcode]
- Data Segment register [points to the data memory]
- Stack Segment register [points to the Stack memory]
- Extra Segment register [points to the data memory]

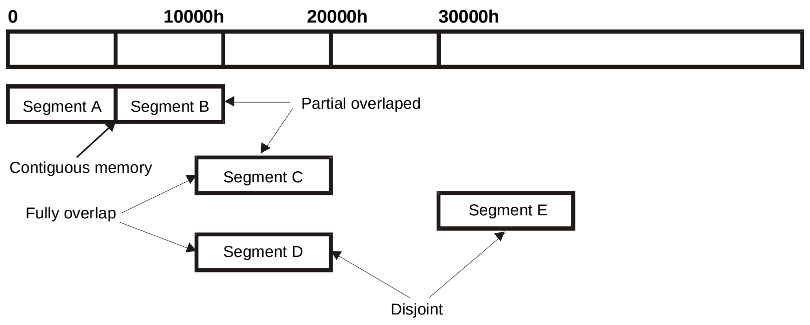


Fig 1.7

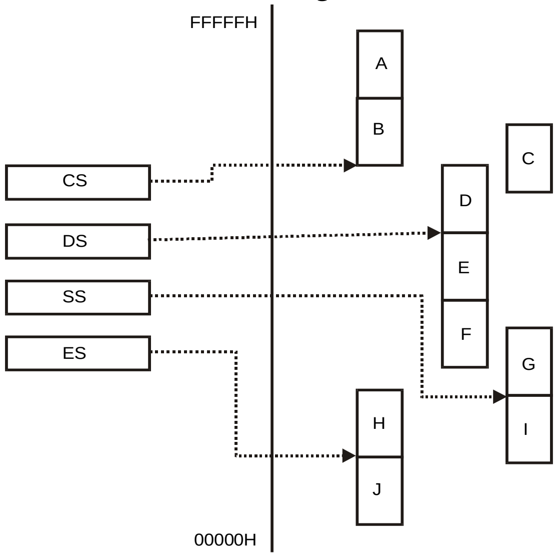


Fig-1.8

Fig 1.8 shows the segment registers pointing to the various memory segments. Since logical addresses are 16-bits wide, up to 64K (65536) bytes in a given segment can be addressed.Each time the CPU need to generate a memory address, one of the segment registers is automatically chosen and its contents added to a logical address. For an instruction fetch, the code segment register is automatically added to the logical address (in this case, the contents of the instruction pointer) to compute the value of the instruction address.
For stack referencing the stack segment register is automatically added to the logical address (the SP or BP register contents) to compute the value of the stack address.

For data reference operations, where either the data or extra segment register is chosen as the base, the logical address can be made up of many different types of values: it can be simply the immediate data value contained in the instruction, or it can be the sum of an immediate data value and a base register, plus an index register. Generally, the selection of the DS or ES register is made automatically, though provisions do exist to override this selection. Thus any memory location may be addressed without changing the value of the segment base register. In systems that use 64K or fewer bytes of memory for each memory area (code, stack, data and extra), the segment registers can be initialized to zero at the beginning of the program and then ignored, since zero plus a 16-bit offset yields a 16-bit address. In a system where the total amount of memory is 64K bytes or less, it is possible to set all segments equal and have fully overlapping segments.

Segment registers are also very useful for large programming tasks, which require isolation of program code from the data area, or isolation of module data from the stack information etc.

Segmentation makes it easy to build re-locatable and reentrant programs. In many cases, the task of relocating a program (relocation means having the ability to run the same program in several different areas of memory without changing addresses in the program itself) simply requires moving the program code and then adjusting the code segment register to point to the base of the new code area. Since programs can be written for the 8086 / 8088 in which all branches and jumps are relative to the instruction pointer, it does not matter what value is kept in the code segment register. Every application will define and use segments differently. The currently addressable segment override provide, a generous workspace: 64K bytes for code, 64K bytes stack and 128K bytes of data storage.

The Intel-8086, high performance 16-bit CPU is available in three clock speeds: 5, 8 and 10 MHz. The CPU is implemented in N-channel, depletion load, silicon gate technology (HMOS), and packaged in a 40 pin DIP package: The 8086 operates in both single processor and multiple processor configurations to achieve high performance levels.

Pin Configuration

Fig. 1.9 shows the pin configuration.

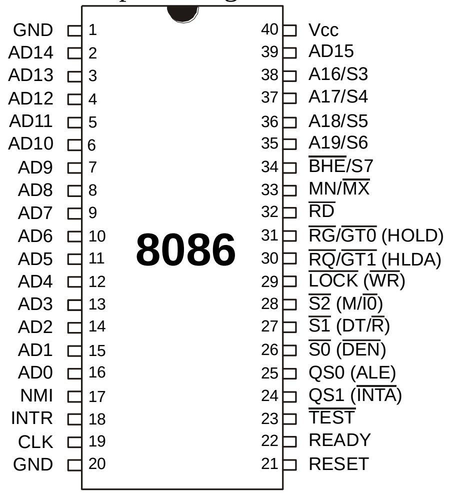


Fig. 1.9 Pin Configuration

The following pin function descriptions are for the microprocessor 8086 in either minimum or maximum mode. The 8086 pins signals are TTL compatible.

AD0 - AD15 (I/O): Address Data Bus

These lines constitute the time multiplexed memory/IO address during the first clock cycle (T1) and data during T2, T3 and T4 clock cycles. A0 is analogous to $\overline{BHE}$ for the lower byte of the data bus, pins D0-D7. A0 bit is Low during T1 state when a byte is to be transferred on the lower portion of the bus in memory or I/O operations. 8-bit oriented devices tied to the lower half would normally use A0 to condition chip select functions. These lines are active high and float to tri-state during interrupt acknowledge and local bus "Hold acknowledge". Fig. 1.10 shows the timing of AD_0 – AD_{15} lines to access data and address.

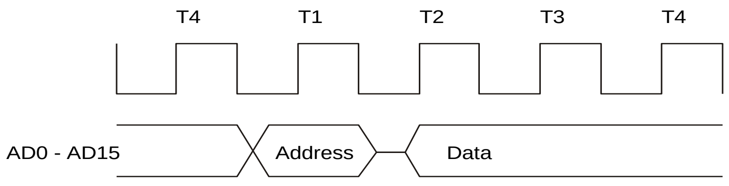


Fig. 1.10

A19/S6, A18/S5, A17/S4, A16/S3 (0): Address/Status
During T1 state these lines are the four most significant address lines for memory operations. During I/O operations these lines are low. During memory and I/O operations, status information is available on these lines during T2, T3, and T4 states.

S5: The status of the interrupt enable flag bit is updated at the beginning of each cycle. The status of the flag is indicated through this bus.

S6: When Low, it indicates that 8086 is in control of the bus. During a "Hold acknowledge" clock period, the 8086 tri-states the S6 pin and thus allows another bus master to take control of the status bus.

S3 & S4: Lines are decoded as follows:

A17/S4	A16/S3	Function
0	0	Extra segment access
0	1	Stack segment access
1	0	Code segment access
1	1	Data segment access

Table 1.2
After the first clock cycle of an instruction execution, the A17/S4 and A16/S3 pins specify which segment register generates the segment portion of the 8086 address. Thus by decoding these lines and using the decoder outputs as chip selects for memory chips, up to 4 Megabytes (one Mega per segment) of memory can be accesses. This feature also provides a degree of protection by preventing write operations to one segment from erroneously overlapping into another segment and destroying information in that segment.

$\overline{\text{BHE}}$ **/S7 (O): Bus High Enable/Status**
During T1 state $\overline{\text{BHE}}$ should be used to enable data onto the most significant half of the data bus, pins D15 - D8. Eight-bit oriented devices tied to the upper half of the bus would normally use $\overline{\text{BHE}}$ to control chip select functions. $\overline{\text{BHE}}$ is Low during T1 state of read, write and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus.The S7 status information is available during T2, T3 and T4 states. The signal is active Low and floats to 3-state during "hold" state. This pin is Low during T1 state for the first interrupt acknowledge cycle.

$\overline{\text{RD}}$ **(O): READ**
The Read strobe indicates that the processor is performing a memory or I/O read cycle. This signal is active low during T2 and T3 states and the Tw states of any read cycle. This signal floats to tri-state in "hold acknowledge cycle".

$\overline{\text{TEST}}$ **(I)**
$\overline{\text{TEST}}$ Pinis examined by the "WAIT" instruction. If the $\overline{\text{TEST}}$ pin is Low, execution continues. Otherwise the processor waits in an “idle” state. This input issynchronized internally during each clock cycle on the leading edge of CLK.

INTR (I): Interrupt Request
It is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored

to via an interrupt vector look up table located in system memory. It can be internally masked by software resetting the interrupt enable bitINTR is internally synchronized. This signal is active HIGH.

NMI (I): Non-Maskable Interrupt

An edge triggered input, causes a type-2 interrupt. A subroutine is vectored to via the interrupt vector look up table located in system memory. NMI is not maskable internallyby software. A transition from a LOW to HIGH on this pin initiates the interrupt at the end of the current instruction. This input is internally synchronized.

Reset (I)
Reset causes the processor to immediately terminate its present activity. To be recognised, the signal must be active high for at least four clock cycles, except after power-on which requires a 50 Micro Sec. pulse. It causes the 8086 to initialize registers DS, SS, ES, IP and flags to all zeros. It also initializes CS to FFFF H. Upon removal of the RESET signal from the RESET pin, the 8086 will fetch its next instruction from the 20 bit physical address FFFF0H. The reset signal to 8086 can be generated by the 8284. (Clock generation chip). To guarantee reset from power-up, the reset input must remain below 1.5 volts for 50 Micro sec. after Vcc has reached the minimum supply voltage of 4.5V. The RES input of the 8284 can be driven by a simple RC circuit as shown in fig.1.10.

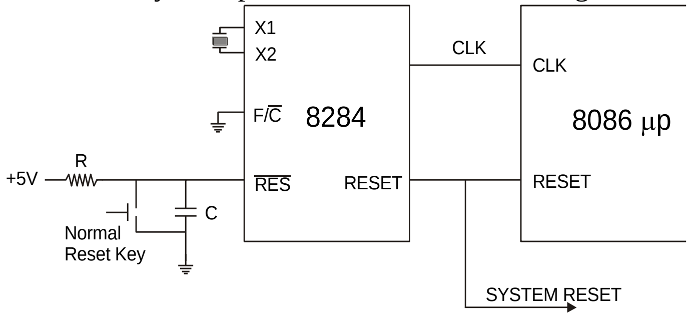


Fig. 1.10

The value of R and C can be selected as follows:
$V_c(t) = V(1 - e^{-t/RC})$ t = 50 Micro sec.
V = 4.5 volts, Vc = 1.05V and RC = 188 Micro sec.
C = 0.1 Micro F; R = 1.88 K ohms.

CPU component	Contents
Flags	Cleared
Instruction Pointer	0000H
CS register	FFFFH
DS register	0000H
SS register	0000H
ES register	0000H
Queue	Empty

Table – 1.3 System Registers after Reset

8086/88 RESET line provide an orderly way to start an executing system. When the processor detects the positive-going edge of a pulse on RESET, it terminates all activities until the signal goes low, at which time it initializes the system as shown in table 1.3

Ready (I)
Ready is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The READY signal from memory or I/O is synchronized by the 8284 clock generator to form READY. This signal is active HIGH. The 8086 READY input is not synchronized. Correct operation is not guaranteed if the setup and hold times are not met.

CLK (I): Clock

Clock provides the basic timing for the processor and bus controller. It is asymmetric with 33% duty cycle to provide optimized internal timing. Minimum frequency of 2 MHz is required, since the design of

8086 processors incorporates dynamic cells. The maximum clock frequencies of the 8086-4, 8086 and 8086-2 are

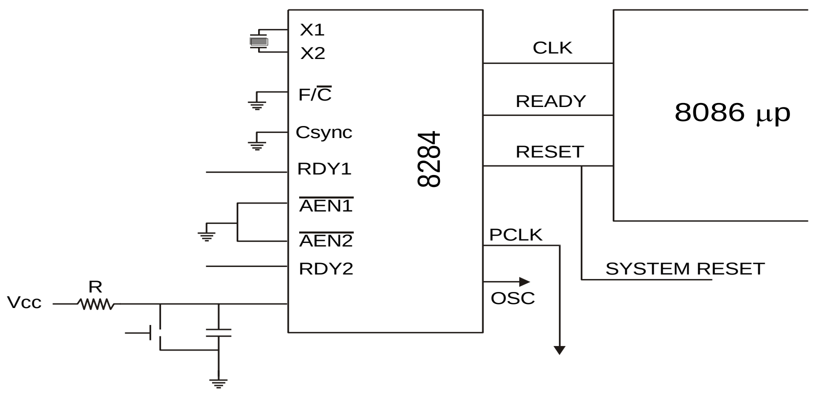


Fig. 1.11

4MHz, 5MHz and 8MHz respectively. Since the 8086 does not have on-chip clock generation circuitry, and 8284 clock generator chip must be connected to the 8086 clock pin. The crystal connected to 8284 must have a frequency 3 times the 8086 internal frequency. The 8284 clock generation chip is used to generate READY, RESET andCLK. It is as shown in fig. 1.11

MN/$\overline{MX}$ (I): Maximum / Minimum

This pin indicates what mode the processor is to operate in. In minimum mode, the 8086 itself generates all bus control signals. In maximum mode the three status signals are to be decoded to generate all the bus control signals.

Minimum Mode Pins

The following 8 pins function descriptions are for the 8086 in minimum mode; MN/$\overline{MX}$ = 1. The corresponding 8 pins function descriptions for maximum mode is explained later.

M/$\overline{IO}$ (O): Status line

This pin is used to distinguish a memory access or an I/O accesses. When this pin is Low, it accesses I/O and when high it access memory. M / $\overline{IO}$ becomes valid in the T4 state preceding a bus cycle and remains valid until the final T4 of the cycle. M/$\overline{IO}$ floats to 3 - state OFF during local bus "hold acknowledge".

$\overline{WR}$ (O): Write

Indicates that the processor is performing a write memory or write IO cycle, depending on the state of the M /$\overline{IO}$ signal. $\overline{WR}$ is active for T2, T3 and Tw of any write cycle. It is active LOW, and floats to 3-state OFF during local bus "hold acknowledge ".

$\overline{INTA}$ (O): Interrupt Acknowledge

It is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T2, T3, and T4 of each interrupt acknowledge cycle.

ALE (O): Address Latch Enable
ALE is provided by the processor to latch the address into the 8282/8283 address latch. It is an active high pulse during T1 of any bus cycle. ALE signal is never floated.

DT/$\overline{R}$ (O): DATA Transmit/Receive
In minimum mode, 8286/8287 transceiver is used for the data bus. DT/$\overline{R}$ is used to control the direction of data flow through the transceiver. This signal floats to tri-state off during local bus "hold acknowledge".

$\overline{\text{DEN}}$ (O): Data Enable
It is provided as an output enable for the 8286/8287 in a minimum system which uses the transceiver. $\overline{\text{DEN}}$ is active LOW during each memory and IO access. It will be low beginning with T2 until the middle of T4, while for a write cycle, it is active from the beginning of T2 until the middle of T4. It floats to tri-state off during local bus "hold acknowledge".

HOLD & HLDA (I/O): Hold and Hold Acknowledge

Hold indicates that another master is requesting a local bus "HOLD". To be acknowledged, HOLD must be active HIGH. The processor receiving the "HOLD" request will issue HLDA (HIGH) as an acknowledgement in the middle of the T1-clock cycle. Simultaneous with the issue of HLDA, the processor will float the local bus and control lines. After "HOLD" is detected as being Low, the processor will lower the HLDA and when the processor needs to run another cycle, it will again drive the local bus and control lines.

Maximum Mode

The following pins function descriptions are for the 8086/8088 systems in maximum mode (*i.e.*. MN/$\overline{\text{MX}}$= 0). Only the pins which are unique to maximum mode are described below.
.
S2, S1, S0 (O): Status Pins

These pins are active during T4, T1 and T2 states and is returned to passive state (1,1,1 during T3 or Tw (when ready is inactive). These are used by the 8288 bus controller to generate all memory and I/O operation) access control signals. Any change by S2, S1, S0 during T4 is used to indicate the beginning of a bus cycle. These status lines are encoded as shown in table 1.4.

S2	S1	S0	Characteristics
0	0	0	Interrupt acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive State

Table 1.4

QS0, QS1 (O): Queue – Status

Queue Status is valid during the clock cycle after which the queue operation is performed. QS0, QS1 provide status to allow external tracking of the internal 8086instruction queue. The condition of queue status is shown in table 1.5

Queue status allows external devices like In-circuit Emulators or special instruction set extension co-processors to track the CPU instruction execution. Since instructions are executed from the 8086 internal queue, the queue status is presented each CPU clock cycle and is not related to the bus cycle activity. This mechanism allows

(1) A processor to detect execution of a ESCAPE instruction which directs the co-processor to perform a specific task and
(2) An in-circuit Emulator to trap execution of a specific memory location.

QS1	QS1	Characteristics
0	0	No operation
0	1	First byte of opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

Table 1.5

$\overline{\text{LOCK}}$ **(O)**

It indicates to another system bus master, not to gain control of the system bus while $\overline{\text{LOCK}}$ is active Low. The $\overline{\text{LOCK}}$ signal is activated by the "LOCK" prefix instruction and remains active until the completion of the instruction. This signal is activeLow and floats to tri-state OFF during 'hold acknowledge".

Example:

LOCK XCHG reg., Memory ; Register is any register and memory GT_0
; is the address of the semaphore.

$\overline{\text{RQ}}$ / $\overline{\text{GT}_0}$ **and** $\overline{\text{RQ}}$ / $\overline{\text{GT}_1}$ **(I/O): Request/Grant**

These pins are used by other processors in a multiprocessor organization. Local bus masters of other processors force the processor to release the local bus at the end of the processors current bus cycle. Each pin is bi-directional and has an internal pull upresistors. Hence they may be left un-connected.

Comparison of 8086 with the 8088 Microprocessor

The 8088 CPU is an 8-bit processor designed around the 8086 internal structure. Most internal functions of the 8088 are identical to the equivalent 8086 functions. The 8088 handles the external bus the same way the 8086 does, one difference being that the 8088 handles only 8-bits at a time. 16-bit operands are fetched or written in two consecutive bus cycles. To an assembly language programmer both processors will appear identical with the exception of execution times. The internal register structure is identical and all instructions produce the same end result. The pin configuration of 8088 is illustrated in fig. 1.14.

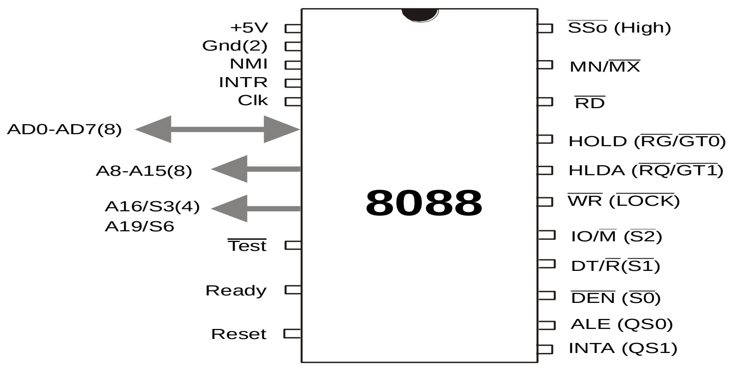


Fig . 1.14 Pin Configuration of 8088 Microprocessor

The major differences between 8088 and 8086 are outlined below:

- The queue length is 4 bytes in the 8088, whereas the 8086 queue comprises of 6 bytes.
- The 8088 BIU will fetch a new instruction to load into the queue as soon as it finds a byte hole (space available) in the queue. The 8086 waits until a 2 byte space is available.
- The internal execution time of the instruction set is affected by the 8-bit interface. All 16-bit fetches and writes from / to memory take an additional four clock cycles. The CPU is also limited by the speed of instruction fetch. When the more sophisticated instructions of the 8088 are being used, the queue has time to fill and the execution proceeds as fast as the execution unit will allow.

The hardware interface of the 8088 has some major differences as compared to the 8086. The pin assignments are nearly identical, however, with the following functional changes.

- A8-A15: These pins are only address outputs on the 8088. These address lines are latched internally and remain valid throughout a bus cycle in a manner similar to the 8085 upper address lines.
- $\overline{SS0}$ provides the $\overline{S0}$ status information in the minimum mode. This output occurs on pin 34 in minimum mode only. DT/$\overline{R}$, IO/$\overline{M}$ and $\overline{SS0}$ provide the complete bus status in minimum mode. This is shown in table 11.5

IO/M	DT/R	SSO	CHARACTERISTICS
0	0	0	Code Access
0	0	1	Read Memory
0	1	0	Write Memory
0	1	1	Passive
1	0	0	Interrupt Acknowledge
1	0	1	Read I/O port
1	1	0	Write I/O port
1	1	1	Halt

Table 1.6

- ❖ $\overline{\text{BHE}}$ has no meaning on the 8088 and has been eliminated.
- ❖ IO/$\overline{\text{M}}$ has been inverted. *i.e.,* (In 8086, this pin as $\overline{\text{IO}}$/M)
- ❖ ALE is delayed by one clock cycle in the minimum mode when entering HALT to allow the status to be latched with ALE.

Questions

1. Compare 8086 and 8088 microprocessors. In what ways are they similar? In what ways do they differ?
2. What is the purpose of the ALE signal in an 8086 system?
3. What is the major difference between an 8086 operating in minimum mode and an 8086 operating in maximum mode?
4. Describe the response of an 8086 when its RESET input is asserted high.
5. Why are buffers often needed on the address, data and control buses in a microprocessor system?
6. What are the function of the 8086 DT/$\overline{\text{R}}$ and $\overline{\text{DEN}}$ signals?
7. Explain the difference between a memory read cycle and an I/O read cycle.
8. What does the ASYNC input to the 8284 accomplish?
9. What logic levels must be applied to AEN1 and RDY1 to obtain a logic 1 at the ready pin?
10. Explain the operation of the LOCK pin.
11. What conditions do the QS1 and QS0 pins indicate about the 8086/8088?
12. Explain the operation of the TEST pin and the WAIT instruction.
13. What is the function of QS0 and QS1 signals?

Objective Questions

1. State true or false
   - (a) Both the address and data bus are unidirectional.
   - (b) Both the 8086 and 8088 microprocessors address 64 K bytes of memory.
   - (c) No difference exits between the memory maps of 8086 and 8088.
   - (d) The string source (SI) is located in the Data Segment (DS) and the string destination is located in the Extra segment (ES) for string instructions.

2. The INTR input is -- sensitive.

3. The 8286 microprocessor has a 1-bit data bus. Its memory is organized into -- banks, each--bit wide.

4. A microprocessor with a 24-bit address bus could access --- MB of memory.

5. A bus cycle is equal to clocking --- periods.

6. If the ready pin is grounded, it will introduce ---state into the bus cycle.

7. The PCLK output of the 8284A is --- MHz if the crystal frequency is 14 MHz.

8. The RES input to the 8284A is placed at a logic --- level in order to reset the

   8086/8088.

9. The 8086 and 8088 processor can be operated in either the minimum or maximummode. The maximummode is so named because the maximum mode

(a) Let you execute the maximum number of instructions.
(b) Let you address the maximum number of memory locations (IMB)
(c) Requires more support hardware than the minimum mode.
(d) All of the above.

10. The 8086/8088 use a multiplexed address and data bus because

(a) 40 pins is a good size for the IC.
(b) Multiplexing is supported by 8085 Microprocessor.
(c) Multiplexing reduces the number of lines between the microprocessor and the auxiliary Ics.
(d) All of the above.

11. The DEN signal is active ------- output from 8288 bus controller.

12. An 8086/8088 microprocessor requires -------- and ------- chips is maximum mode systems configuration.

13.The 8288 bus controller must be used in the ------ mode to provide ------ signals to the memory and I/O.

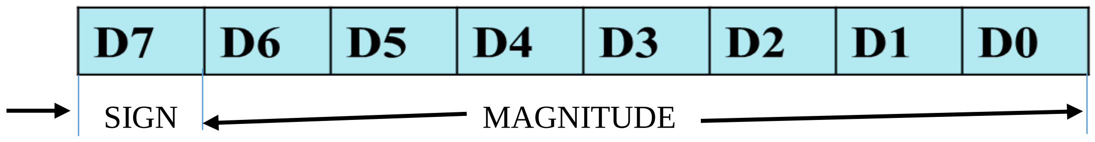
14. 8088 microprocessor does not required latch for a A8 – A15 lines because ---------.

15. SSO of 8088 microprocessor indicates -------.

Signed number Arithmetic

Signed and Unsigned Numbers. An 8 bit number system can be used to create 256 combinations (from 0 to 255), and the first 128 combinations (0 to 127) represent positive numbers and next 128 combinations (128 to 255) represent negative numbers.

Signed byte operands (8 bits): In signed byte operands, D7 (MSB) is the sign and D0 and D6 are set aside for the magnitude of the number.



If D7 bit is 0 = It is a positive number
D7 bit is 1 = It is a negative number

Positive Number are represented

0	**0 0 0 0 0 0 0 0**
+1	**0 0 0 0 0 0 0 1**
+5	**0 0 0 0 1 0 0 1**
.	
.	
.	
+127	**0 1 1 1 1 1 1 1**

Negative number is represented as

-128	1 0 0 0 0 0 0 0
-127	1 0 0 0 0 0 0 1
….	
-2	1 1 1 1 1 1 1 0
-1	1 1 1 1 1 1 1 1
0	0 0 0 0 0 0 0 0

IMUL Source

IMUL (Integer Multiply) performs a signed multiplication of the source operand and the accumulator. If the source is a byte, then AL is multiplied and the double-byte result is returned in register AH and AL. If the source is a word, then it is multiplied by register AX, and the double-word result is returned to registers DX and AX. If the upper half of the result (AH for byte source, DX for word source) is not the sign extended from the lower half of the result, CF and OF are set; otherwise they are cleared. When CF and OF are set, they indicate that AH or DX contain significant digits of the result. The contents of AF, PF, SF and ZF are undefined following execution of IMUL.

Example: 1. AL = 69 H, BL = 14 H
IMUL BL; AX = +0834 H, Flags: SF = 0, CF = 1, OF = 1

2. AX = -28H, BL = 59H
   IMUL BL; AX = F218H   Flags: SF = 1, CF = 1, OF = 1.
   ; In place of IMUL if MUL instruction is used then AX = 4B18H.

IDIV Source

IDIV (Integer Divide) performs a signed division of the accumulator (and its extension) by the source operand. The source operand may be a register or the contents of memory (any memory addressing mode). If the source operand is a byte, it is divided into the double-length dividend assumed to be in register AH and AL. The single-length quotient is stored in AL, and the single-length remainder is stored in AH. For byte integer division, the maximum positive quotient is +127 (7F) and the minimum negative quotient is -128 (81H). If the source operand is a word, it is divided into the double-length dividend in register DX and AX (the high-order 16-bits are in DX and the low-order 16-bits in AX). The single-length remainder is stored in DX. For word integer division, the maximum positive quotient is +32767 (7FFFH) and the minimum negative quotient is -32,768 (8001H). If the quotient is positive and exceeds the maximum, or is negative and is less than the minimum, the quotient and the remainder are undefined, and a type-0 interrupt is generated. A division by zero interrupt results in the following action.

1. Push the flag register onto the stack.
2. Clear IF and TF.(Interrupt Flag and Trap Flag)
3. Push the CS register onto the stack.
4. Load the word at the location 00002H into the CS.
5. Push the IP onto the stack
6. Load the word at memory location 00000H into the IP.

The value of OF, SF, ZF, AF, PF and CF are undetermined for this operation.

Example: IDIV BL;
Before: AX = 03ABH; BL = D3H = -2DH
After: Quotient in AL = ECH; Remainder in AH = 27 H.

CBW

CBW (Convert Byte to Word) extends the sign of the byte in register AL throughout register AH. It does not affect any flags. CBW can be used to produce a double length (word) dividend from a byte prior to performing byte division.

CBW

Before: AX = -155 decimal = 0000000100011011 binary
After: AX = -155 decimal = 1111111100011011 binary

Encoding

10011000

CWD

CWD (Convert Word to Double Word) extends the sign of the word in register AX throughout register DX. CWD does not affect any flags. CWD can be used to produce a double-length (double word) dividend from a word prior to performing word division.

CWD

```
Before: DX = 00000000  00000000 = 0000 H;
        AX = 11110000  11000111 = F0C7 H
After:  DX = 11111111  11111111 = FFFF H;
        AX = 11110000  11000111 = F0C7 H.
```

Shifts

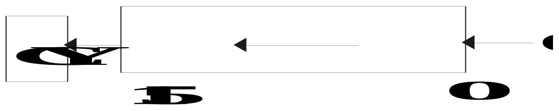
The bits in bytes and words may be shifted arithmetically or logically. Up to 255 shifts may be performed according to the value of the count operand coded in the instruction. The count may be specified as the constant (only one shift if a constant is specified), or as register CL, allowing the shift count to be a variable supplied at execution time.

Arithmetic shifts may be used to multiply and divide binary numbers by powers of two. Logical shifts can be used to isolate bits in bytes or words. Shift instructions affect the flags as follows:

AF	: is undefined
PF, SF, ZF	: are updated normally, as in logical instructions.
CF	: always contains the value of the last bit shifted out of the destination operand.
OF	: Undefined for multi-bit shift.

The sign-bit is set if the value of the high-order (sign) bit was changed by the operation; if the sign bit retains its original value, OF is cleared.

SAL/SHL Destination, Count

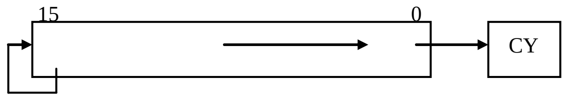


SAL/SHL (Shift Arithmetic Left and Shift Logical Left) perform the same operation and are physically the same instruction. The destination byte or word is shifted left by the number specified in the count operand. Zeroes are shifted in from the right. If the sign bit retains its original value, then OF is cleared.

Example: 1. SAL AH, 01;
Before: AH = 32H, CF = 0, AF = 0, PF = 0, SF = 0, ZF = 0
After: AH = 64H, CF = 0, AF = 0, PF = 0, SF = 0, ZF = 0
2. SAL DI, CL
Before: DI = 0032H, CL = 04, CF = 0, AF = 0, PF = 0, SF = 0, ZF = 0
After: DI = 0320H, CL = 00, CF = 0, AF = 0, PF = 0, SF = 0, ZF = 0

SAR Destination, Count

SAR (Shift Arithmetic Right) shifts the bits in the destination operand (byte or word) to the right by the number of bits specified in the count operand. Bits equal to the original high-order (sign) bit are shifted in on the left preserving the sign of the original value. Note that SAR does not produce the same result as the divide of an equivalent IDIV instruction if the destination operand is negative and 1-bit is shifted out.



Example: Shifting –5 right by 1 yields –3, while integer division of –5 by 2 yields –2. The difference in the instructions is that IDIV truncates all numbers towards zero, while
SAR truncates positive numbers towards zero and negative numbers towards infinity.

1. SAR DX, 01
   Before: DX = 2345H After: 11A2H
2. SAR DI, CL
   Before: DI = 3000H, CL = 04H After: DI = 0300H, CL= 00

CMP Destination, Source

CMP (Compare) subtracts the source from the destination, which may be bytes or words, but does not return the result. The operands are unchanged, but the flags are updated and can be tested by a subsequent conditional jump instruction. CMP updates AF, CF, OF, PF, SF and ZF.

The comparison reflect on the three important condition flags. They are CF, SF and ZF.

	CF	ZF	SF
Destination = Source	0	1	0
Destination > Source	0	0	0
Destination < Source	1	0	1

1. CMP CX, BX

```
 Before:  CX = 1000,   BX = 1000
          CF = 0   AF = 0   OF = 0   PF = 0   ZF = 0 SF = 0
 After:   CX = 1000,   BX = 1000
          CF = 0   AF = 0   OF = 0   PF = 1   ZF = 1 SF = 0
```

2. CMP DH, ALPHA

```
 Before: DH = 3FH, ALPHA = 1000, M [DS*16 + 1000] = 42H
         CF = 0   AF = 0   OF = 0   PF = 0   ZF = 0 SF = 0
 After:  DH = 3FH, ALPHA = 1000, M [DS*16 + 1000] = 42H
         CF = 1   AF = 0   OF = 0   PF = 0   ZF = 0 SF = 1
```

3. CMP BL, 02H

```
 Before:  BL = 30H, CF = 0   AF = 0   OF = 0   PF = 0   ZF = 0 SF = 0
 After:   BL = 30H, CF = 0   AF = 1   OF = 0   PF = 1   ZF = 1 SF = 0
```

Signed number condition jump instructions

JG/JNLE: Jump on Greater/Jump on Not Less or Equal transfers control to the target operand (IP + displacement) if the conditions ((SF XOR OF) or ZF = 0) are greater than / not less than or equal to the tested value.

JGE/JNL: Jump on Greater than Equal / Jump on Not Less than transfers control to the target operand (IP + displacement) if the condition (SF XOR OR = 0) is greater than or equal / not less than the tested value.

JL/JNGE: Jump on Less than / Jump on Not Greater than or Equal transfers control to the target operand (IP+ displacement) if the condition (SF XOR OF =1) is less than / not greater than or equal to the tested value.
JLE/JNG: Jump on Less than or Equal to/ Jump on Not Greater than transfers control to the target operand (IP + displacement) if the conditions tested ((SF XOR OF) or ZF = 1) are less than or equal to/not greater than the tested value.

Exercises

1. Explain the difference between the IDIV and DIV instruction.

2. Write program sequences that will carry out the following binary operations.
   (a) Z = W + (Z-X)   (b) Z = W-(X + 6)-(Y + 9)   (c) Z = (W*X)/(Y + 6)
   (d) R = Remainder   (e) Z = ((W-X)/10*Y)**2

3. Write a program sequence for performing an unsigned binary division on an n-word number

   by a one _word number.

4. Write program sequences that will perform the following operations on two digit packed BCD numbers.
   (a) U = V + (S-6) (b) U = (X + W)-(Z-U)

Objective questions

1.Select the correct instruction to perform each of the following tasks:
   (a) Shift DI right three places with zeros moved into the leftmost bit.
   (b) move all bits in AL left one place, making sure that the sign of the result is the same as the sign of the original number.
   (c) Rotate all the bits of AL left three places
   (d) Move the DH register right one place, making sure that the sign of the result is the same as the sign of the original number.

2. The 8086 arithmetic instructions can process:
   (a) Signed and unsigned numbers (b) ASCII data
   (c) Packed BCD data (d) All of the above.

3. The purpose of the convert byte to word (CBW) instruction is to ----- .
   (a) Allow a proper calculation with data from a shorter word
   (b) Allow calculations with data such as externally provided 8-bit numbers
   (c) Fill the unused bits with a replication of the number's sign bit
   d) All of the above.

5. The 8086 TEST instruction performs the same function as the ______ instruction, but without storing any result. Only condition flags are affected.
   (a) XOR (b) AND (c) SUB (d) OR
6. The instruction sequence CMP BL, AL JA 100H will transfer program control to memory location 100h if BL ------ AL.
7. State true or false:
   i. BX register is used as an index register in a data segment.
   ii. CX register is assumed to work like a counter.
   iii. The source index (SI) and destination index (DI) registers in 8086 can also be used as general registers.
   iv. The Trap Flag (TF) is a condition flag.
   v. The CALL instruction should be followed by a RET instruction.
   vi. Conditional jump instructions require one of the flags to be tested.