Unit 4
Intel 8085/8086/8088

Intel 8085 microprocessor

Functional Block Diagram

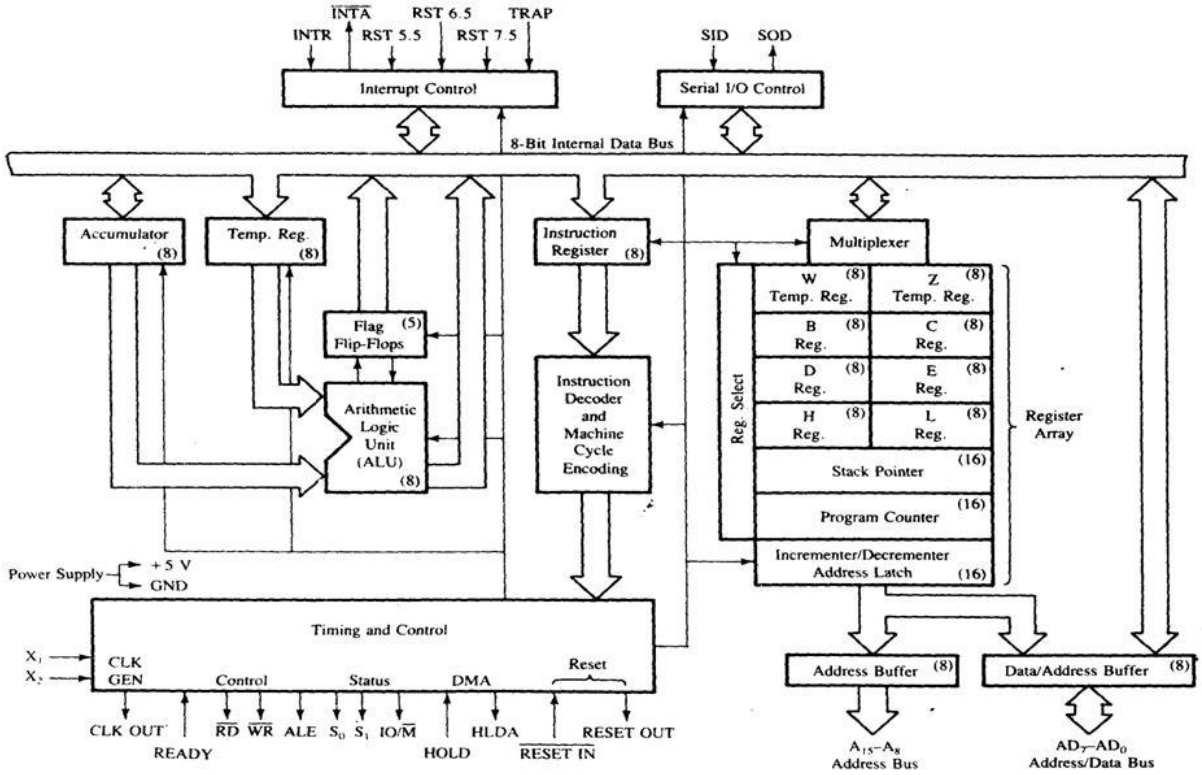


Fig: Functional Block Diagram of 8085 Microprocessor

1. ALU

→ The ALU performs the actual numerical and logic operation such as 'add', 'subtract', 'AND', 'OR' etc.

→ Uses data from memory and from Accumulator to perform arithmetic operation and always stores result of operation in Accumulator.

→ The ALU consists of accumulator, flag register and temporary register.

a. 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.

b. Flag register

→ 8085 has 8-bit flag register. There are only 5 active flags.

S	Z		AC		P		CY

Fig: 8085 flag register

→ Flags are flip-flops which are used to indicate the status of the accumulator and other register after the completion of operation.

→ These flip-flops are set or reset according to the data condition of the result in the accumulator and other registers.

i. Sign flag(S):

→ Sign flag indicates whether the result of a mathematical or logical operation is negative or positive.

→ If the result is negative, this flag will be set (i.e. S=1) and if the result is positive, the flag will be reset (i.e. S=0).

ii. Zero flag (Z):

→ Zero flag indicates whether the result of a mathematical or logical operation is zero or not.

→ If the result of current operation is zero, the flag will be set (i.e. Z=1) otherwise the flag will be reset (Z=0).

→ This flag will be modified by the result in the accumulator as well as in the other register.

iii. Auxiliary carry flag (AC):

→ In operation when a carry is generated by bit D_3 and passes on to bit D_4, the AC flag will be set otherwise AC flag will be reset.

→ This flag is used only internally for BCD operation and is not available for the programmer to change the sequence of program with the jump instruction.

iv. Parity flag (P):

→ This flag indicates whether the current result is of even parity (no. of 1's is even) or odd parity (no. of 1's is odd).

→ If even parity, P flag will be set otherwise reset.

v. Carry flag (CY):

→ This flag indicates whether during an addition or subtraction operation carry or borrow is generated or not.

→ If carry or borrow is generated, the flag will be set otherwise reset.

2. Timing and control unit

→ This unit produces all the timing and control signal for all the operation.

→ This unit synchronizes all the MP operations with the clock and generates the control signals necessary for communication between the MP and peripherals.

3. Instruction register and decoder

→ The instruction register and decoder are part of ALU. When an instruction is fetched from memory, it is loaded in the instruction register.

→ The decoder decodes the instruction and establishes the sequence of events to follow.

→ The IR is not programmable and cannot be accessed through any instruction.

4. Register array

→ The register unit of 8085 consists of

- Six general-purpose data registers B,C,D,E,H,L
- Two internal registers W and Z
- Two 16-bit address registers PC (program counter) and SP (stack pointer)
- One increment/decrement counter register
- And, one multiplexer (MUX)

→ The six general-purpose registers are used to store 8-bit data. They can be combined as register pairs BC, DE, and HL to perform some 16-bit operations.

→ The two internal registers W and Z are used to hold 8-bit data during the execution of some instructions, **CALL** and **XCHG** instructions.

→ SP is 16-bit registers used to point the address of data stored in the stack memory. It always indicates the top of the stack.

→ PC is 16-bit register used to point the address of the next instruction to be fetched and executed stored in the memory.

5. System bus

a. Data bus

→ It carries 'data', in binary form, between MP and other external units, such as memory.

→ Typical size is 8 or 16 bits.

b. Address bus

→ It carries 'address' of operand in binary form.

→ Typical size is 16-bit.

c. Control Bus

→ Control Bus are various lines which have specific functions for coordinating and controlling MP operations.

→ E.g.: Read/Write control line.

6. Interrupt Control

→ Interrupt is a signal, which suspends the routine what the MP is doing, brings the control to perform the subroutine, completes it and returns to main routine.

→ May be hardware or software interrupts. Some interrupts may be ignored (maskable), some cannot (non-maskable).

→ E.g. INTR, TRAP, RST 7.5, RST 6.5, RST 5.5

7. Serial I/O Control

→ The MP performs serial data input or output (one bit at a time). In serial transmission, data bits are sent over a single line, one bit at a time.

→ The 8085 has two signals to implement the serial transmission: SID (serial input data) and SOD (serial output data).

Pin Configuration

Properties

Single + 5V Supply
4 Vectored Interrupts (One is Non Maskable)
Serial In/Serial Out Port
Decimal, Binary, and Double Precision Arithmetic
Direct Addressing Capability to 64K bytes of memory

The Intel 8085 is a new generation, complete 8-bit parallel central processing unit (CPU). The 8085 uses a multiplexed data bus. The address is split between the 8-bit address bus and the 8-bit data bus.

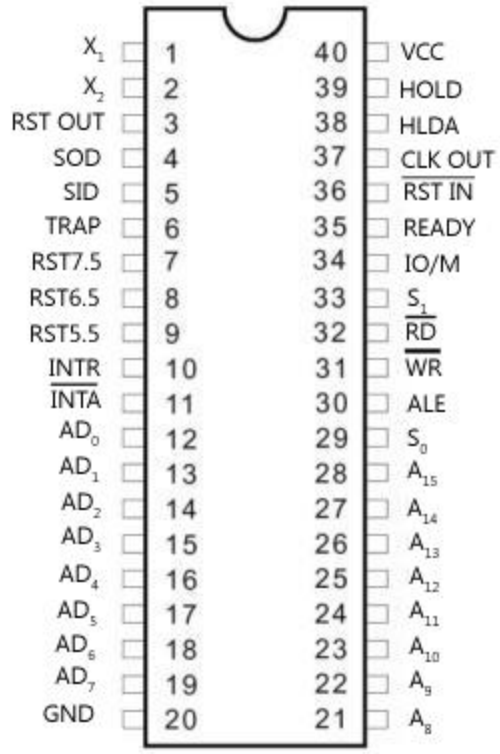


Fig: 8085 Pin Diagram

X1, X2 (Input)
A crystal (or RC, LC network) is connected at these two points. The frequency is divided into two; therefore to operate a system at 3 MHz, the crystal should have a frequency of 6 MHz.

RESET OUT (Output)
This signal indicates that the microprocessor is being reset. It is also used to reset other devices.

SOD (Output)/ SID (Input)
This signal is used for the transmission of data per bit in and out of the processor.

TRAP (Input)
It is a non-maskable interrupt and has the highest priority of any interrupt.

RST 5.5 / RST 6.5 / RST 7.5 (Inputs)
RESTART interrupts. These are the vector interrupts that transfer the program control to the specific memory locations.
RST 7.5 Highest Priority
RST 6.5
RST 5.5 Lowest Priority
The priority of these interrupts is ordered as shown above. These interrupts have a higher priority than the INTR.

INTR (Input)
INTERRUPT REQUEST. This is used as a general purpose interrupt.

INTA (Output)
INTERRUPT ACKNOWLEDGE; this is used to acknowledge an interrupt.

AD0 -AD 7 (Input / Output- 3state)
Multiplexed Address/Data Bus; Lower 8 bits of the memory address (or I/0 address) appear on the bus during the first clock cycle of a machine state. It then becomes the data bus during the second and third clock cycles.

Vss
Ground Reference.

A8 - A15 (Output -3 State)
Address Bus; The most significant 8 bits of the memory address or the 8 bits of the I/0 address,

S0, S1 (Output)
These status signal, similar to IO/M, can identify various operations, but they are rarely used in small systems.

S_1	S_0	
0	0	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

S1 can be used as an advanced R/W status.

ALE (Output)
Address Latch Enable: it indicates that the bits on AD7-AD0 act as lower 8-bit address bus (A7-A0) when logic high. If ALE=0, it act as data bus (D7-D0).

WR (Output 3-state)
WRITE; indicates the data on the Data Bus is to be written into the selected memory or I/O location.

RD (Output 3state)
READ; indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer.

IO/M (Output)
This is a status signal used to differentiate between I/O and memory operations. When it is high, it indicates an I/O operation; when it is low, it indicates memory operation. This signal is combined with RD and WR to generate I/O and memory control signal.

READY (Input)
If Ready is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If Ready is low, the CPU will wait for Ready to go high before completing the read or write cycle.

RESET IN (Input)
When this pin goes low, it sets the Program Counter to zero and resets the MP, Interrupt Enable and HLDA flip-flops. None of the other flags or registers (except the instruction register) are affected. The CPU is held in the reset condition as long as Reset is applied.

CLK OUT (Output)
This signal can be used as the system clock for other devices.

HLDA (Output)
HOLD ACKNOWLEDGE; indicates that the CPU has received the Hold request and acknowledge that request.

HOLD (Input)
This signal indicates that a peripheral such as DMA controller is requesting the use of the address and data buses.

Vcc
+5 volt supply

Instruction

→ Instruction is a command or information given to the MP to perform a given specific task on specified data.
→ Each instruction has two parts: one the task to be performed and the other is data to be operated.
→ The code which specifies what operation to be performed is called operation code or opcode.
→ The data on which operation is performed is called operand.
E.g. ADD B
ADD is opcode and B is operand.
→ The complete instruction is combination of opcode and operand.

Addressing Modes

→ Each instruction performs an operation on the specified data. An operand must be specified for an instruction to be executed. The operand may be in the general purpose registers, accumulator or a memory location.
→ The method in which the operand is specified in an instruction is called addressing mode.
→ The various modes used in 8085 microprocessor are:

1. **Implied or Implicit or Inherent Addressing**

The instructions of this mode do not have operands. For example:
EI (Enable Interrupt),
STC (set the carry flag),
NOP (No operation)

2. **Immediate Addressing**

This is the simplest way of addressing. When it executes the instruction will operate on immediate hexadecimal number. The operand is present in instruction in this mode. This mode is used to define and use constants of set initial values of variables. The operand may be 8 bit data or 16 bit data. For example:
MVI B, 05H
LXI B, 7A21H (B←7A and C←21)
ADI 72H

3. **Register Addressing**

Register direct addressing mode means that a register is the source of an operand for an instruction. It is similar to direct addressing. For example:
MOV A, B
ADD B

4. **Direct Addressing**

This addressing mode is called direct because the effective address of the operand is specified directly in the instruction. Instructions using this mode may contain 2 or 3 bytes, with first byte as the Op-code followed by 1 or 2 bytes of address of data. Loading and storing in memory use 2 bytes of address while IN and OUT have one byte address. For example:

LDA 2035H (A←M [2035H])
STA 2500H (M [2500H]←A)
IN 07H (A←port address 07H)

5. Register Indirect Addressing

The address of the operand is specified by register pair.

LDAX B (if B=23 and C=50 then A←M [2350H])
STAX D (if D=30 and E=10 then M [3010H]←A)
MOV A, M (M=HL; if H=68 and L=32, then A←M [6832H)

Instruction and Data Flow

An instruction is a binary pattern designed to perform a specific function. The list of entire instructions is called the instruction set. The instruction set determines what function the microprocessor can perform.

The following notations are used in the description of the instructions:

R = 8085 8-bit registers (B, C, D, E, H, L)
M = memory register (location) pointed by value HL
Rs = register source
Rd = register destination (B, C, D, E, H, L)
Rp = register pair (BC, DE, HL)
( ) = contents of

The 8085 instruction set can be classified into the following five categories:

1. Data Transfer (copy) Instructions

These instructions perform the following six operations:

→ Load 8-bit number in a register.
→ Load 16-bit number in a register pair.
→ Copy from register to register.
→ Copy between register and memory.
→ Copy between I/O and accumulator.
→ Copy between registers and stack memory.

MVI R, 8-bit
MOV Rd, Rs
LXI Rp, 16-bit
OUT 8-bit
IN 8-bit
LDA 16-bit
STA 16-bit
LDAX Rp
STAX Rp
MOV R, M
MOV M, R

2. Arithmetic Instructions

The frequently used arithmetic operations are: Add, Subtract, Increment (add 1), Decrement (subtract 1)

ADD R
ADI 8-bit

ADD M
SUB R
SUI 8-bit
SUM M
INR R
INR M
DCR R
DCR M
INX Rp
DCX Rp

3. Logical and Bit Manipulation Instructions

These instructions include the following operations: AND, OR, X-OR, Compare, Rotate bits
ANA R
ANI 8-bit
ANA M
ORA R
ORI 8-bit
ORA M
XRA R
XRI 8-bit
XRA M
CMP R
CPI 8-bit

4. Branching Instructions

The following instructions change the program sequence.
JMP 16-bit
JZ 16-bit
JNZ 16-bit
JC 16-bit
JNC 16-bit
CALL 16-bit
RET

5. Miscellaneous Instructions

There are a number of instructions related with data transfer among the register, the stack operation instructions and interrupt operations of 8085 MP which are kept in this group. They are:
PUSH, POP
EI, DI

6. Machine Control Instructions

These instructions affect the operation of the processor.
HLT, NOP

Intel 8086/8088 microprocessor

8086 Microprocessor: Functional Block Diagram

The 8086 is a 16-bit microprocessor. The term 16 bit implies that its arithmetic logic unit, its internal registers, and most of its instructions are intended to work with 16 bit binary data. The 8086 has a 16 bit data bus, so it can read data from or write data to memory and ports either 16 bits or 8 bits at a time. The 8086 has a 20 bit address bus, so it can address any one of 2^{20}, or 1,048,576 memory locations.

8086 CPU is divided into 2 independent functional parts to speed up the processing namely **BIU** (Bus interface unit) & **EU** (execution unit).

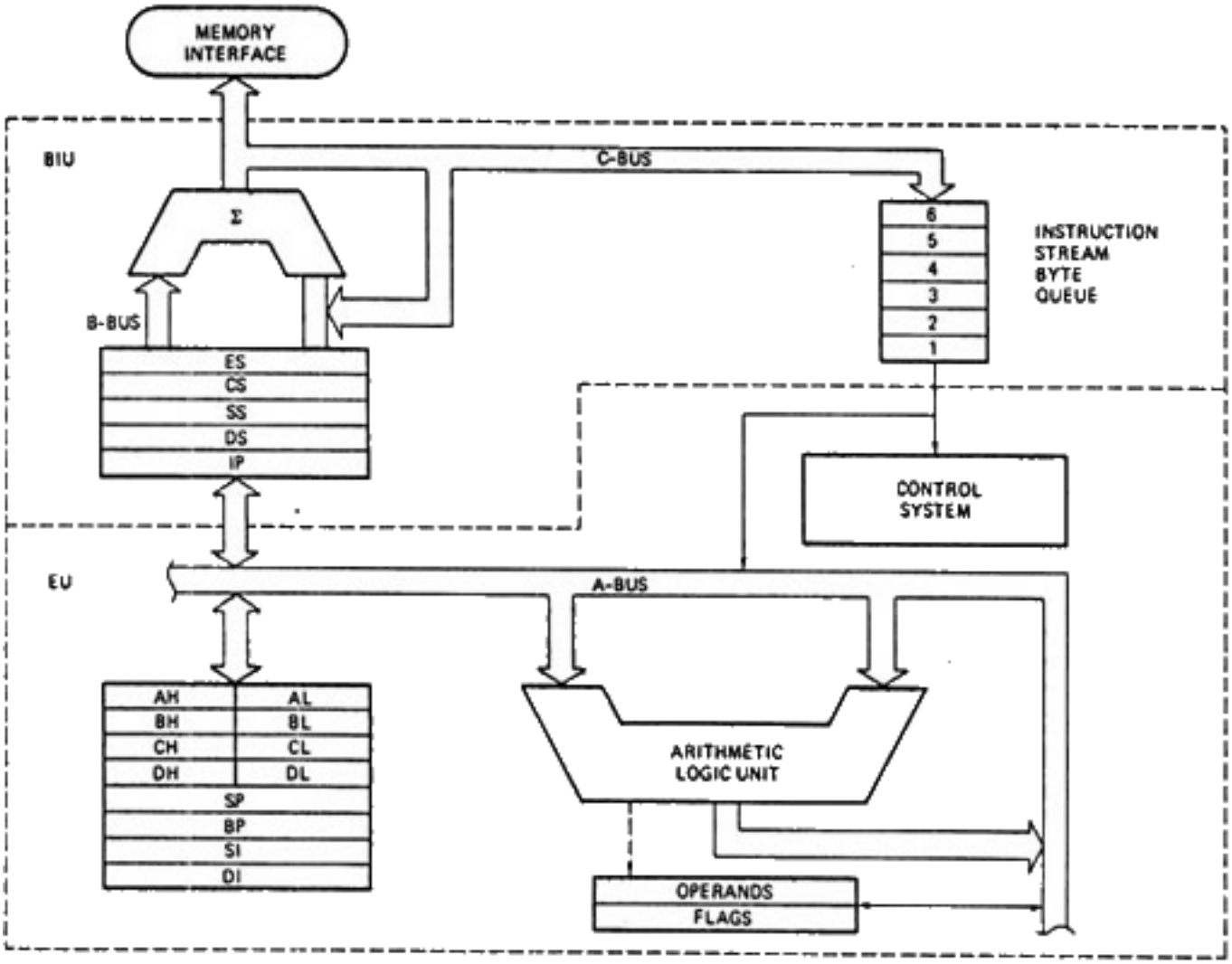


Fig: Functional Block Diagram of Intel 8086 microprocessor

BIU: It handles all transfers of data and addresses on the buses for the execution unit.

- Sends out addresses
- Fetches instructions from memory
- Read / write data from/to ports and memory i.e. handles all transfers of data and addresses on the busses

EU

- Tells BIU where to fetch instructions or data from
- Decodes instructions
- Executes instructions

Execution Unit

Instruction Decoder & ALU:
Decoder in the EU translates instructions fetched from the memory into a series of actions which the EU carries out.16-bit ALU in the EU performs actions such as AND, OR, XOR, increment, decrement etc.

FLAG Register:

It is a 16-bit register. 9-bit are used as different flags, remaining bits unused

				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Fig: 16-bit flag register

Out of 9-flags, 6 are conditional (status) flags and three are control flags

Conditional flags:
These are set or reset by the EU on the basis of the results of some arithmetic or logic operation. 8086 instructions check these flags to determine which of two alternative actions should be done in executing the instructions.

- OF (Overflow flag): is set if there is an arithmetic overflow, i.e. the size of the result exceeds the capacity of the destination location.
- SF (Sign flag): is set if the MSB of the result is 1
- ZF (Zero flag): is set if the result is zero
- AF (Auxiliary carry flag): is set if there is carry from lower nibble to upper nibble or from lower byte to upper byte
- PF (Parity flag): is set if the result has even parity
- CF (Carry flag): is set if there is carry from addition or borrow from subtraction

Control flags:
They are set using certain instructions. They are used to control certain operations of the processor.

- TF (Trap flag): for single stepping through the program
- IF (Interrupt flag): to allow or prohibit the interruption of a program
- DF (Direction flag): Used with string instructions

General purpose Registers (GPRs):

- 8 GPRs AH, AL (Accumulator), BH, BL, CH, CL, DH, DL are used to store 8 bit data.
- AL register is also called the accumulator
- Used individually for the temporary storage of data
- GPRs can be used together (as register pair) to store 16-bit data words. Acceptable register pairs are:

AH-AL pair AX register
BH-BL pair BX register (to store the 16-bit data as well as the base address of the memory location)
CH-CL pair CX register (to store 16-bit data and can be used as counter register for some instructions like loop)
DH-DL pair DX register (to store 16-bit data and also used to hold the result of 16-bit data multiplication and division operation)

Pointer and Index registers: SP (Stack Pointer), BP (Base pointer), SI (Source Index), DI (Destination index)

Pointer Registers:
The two pointer registers, SP and BP are used to access data in the stack segment. The SP is used as offset from current Stack Segment during execution of instruction that involve stack. SP is automatically updated. BP contains offset address and is utilized in based addressing mode. Overall, these are used to hold the offset address of the stack address.

Index Registers:
EU also contains a 16-bit source index (SI) register and 16-bit destination index (DI) register. These registers can be used for temporary storage of data similarly as the general purpose registers. However they are specially

to hold the 16-bit offset of the data *word*. SI and DI are used to hold the offset address of the data segment and extra segment memory respectively.

Bus Interface Unit

The QUEUE:
When EU is decoding or executing an instruction, bus will be free at that time. BIU pre-fetches up to 6-instructions bytes to be executed and places them in QUEUE. This improves the overall speed because in each time of execution of new instruction, instead of sending address of next instruction to be executed to the system memory and waiting from the memory to send back the instruction byte, EU just picks up the fetched instruction byte from the QUEUE.
The BIU stores these pre-fetched bytes in a first-in-first-out (FIFO) register set called a queue. Fetching the next instruction while the current instruction executes is called pipelining.
Segment Registers:
The BIU contains a dedicated address, which is used to produce the 20 bit address. The bus control logic of the BIU generates all the bus control signals, such as the READ and WRITE signals, for memory and I/O. The BIU also has four 16 bit segments registers namely:

- Code segment: holds the upper 16-bits of the starting addresses of the segment from which BIU is currently fetching instruction code bytes.
- Stack segment: store addresses and data while subprogram executes
- Extra segment: store upper 16-bits of starting addresses of two memory segments that are used for data.
- Data segment: store upper 16-bits of starting addresses of two memory segments that are used for data.

Code Segment Register (CS) and Instruction Pointer (IP)
All program instructions located in memory are pointed using 16 bits of segment register CS and 16 bits offset contained in the 16 bit instruction pointer (IP). The BIU computes the 20 bit physical address internally using the logical address that is the contents of CS and IP. 16 bit contents of CS will be shifted 4 bits to the left and then adding the 16 bit contents of IP. Thus, all instructions of the program are relative contents of IP. Simply stated, CS contains the base or start of the current code segment, and IP contains the distance or offset from this address to the next instruction byte to be fetched.
Graphically,

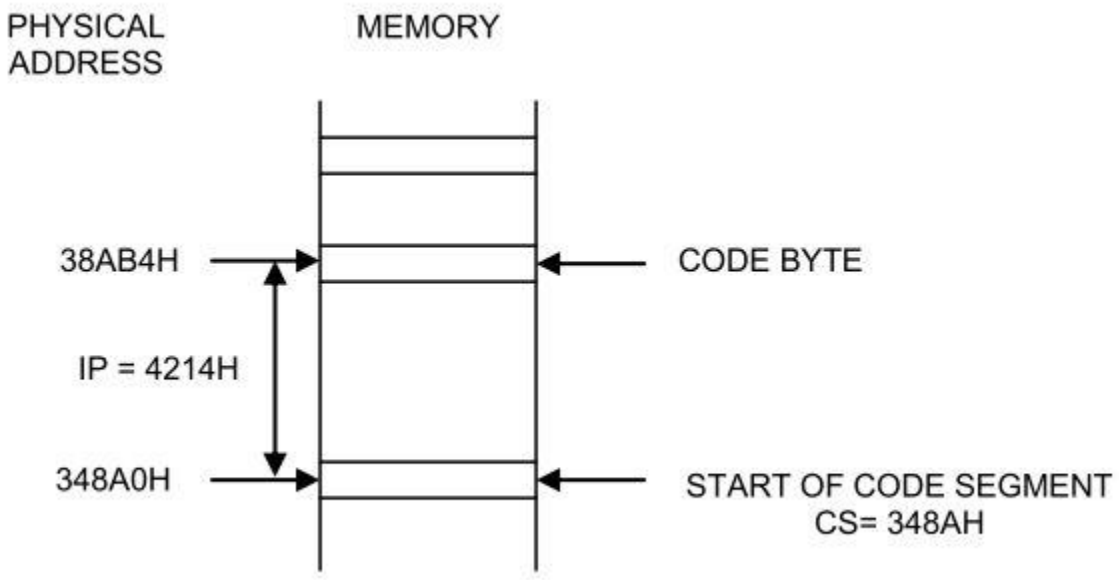


Fig: Diagram showing addition of IP to CS to produce the physical address of code byte Stack Segment Register (SS) and Stack Pointer (SP)

A stack is a section of memory to store addresses and data while a subprogram is in progress. The stack segment registers points to the current stack. The 20 bit physical stack address is calculated from the SS and SP. The programmer can also use Base Pointer (BP) instead of SP for addressing. In this case, the 20 bit physical address is calculated using SS and BP.

Introduction to 8088 microprocessor and its block diagram

(8088 is as same as 8086, however, there are very few differences between them. This note will make you clearer about 8086. Differences are listed at the end.)

Below is a block diagram of the organizational layout of the Intel 8088 processor. It includes two main sections: the Execution Unit (EU) and the Bus Interface Unit (BIU). The EU takes care of the processing including arithmetic and logic. The BIU controls the passing of information between the processor and the devices outside of the processor such as memory, I/O ports, storage devices, etc.

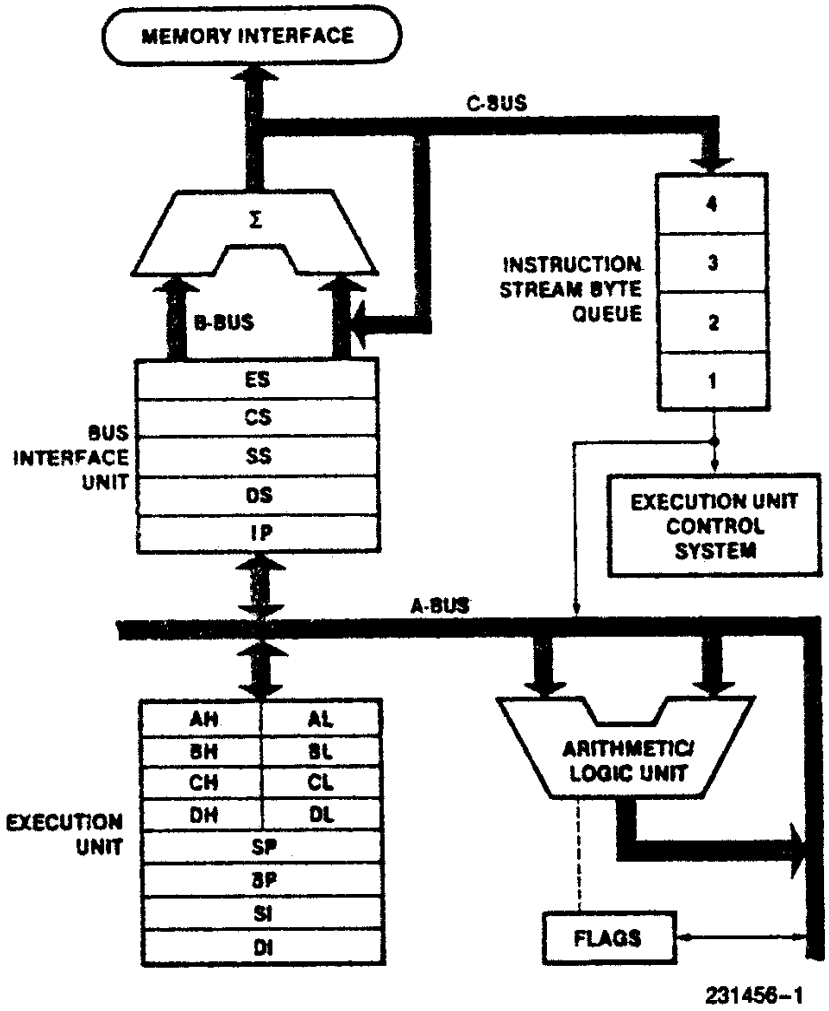


Figure 1. 8088 CPU Functional Block Diagram

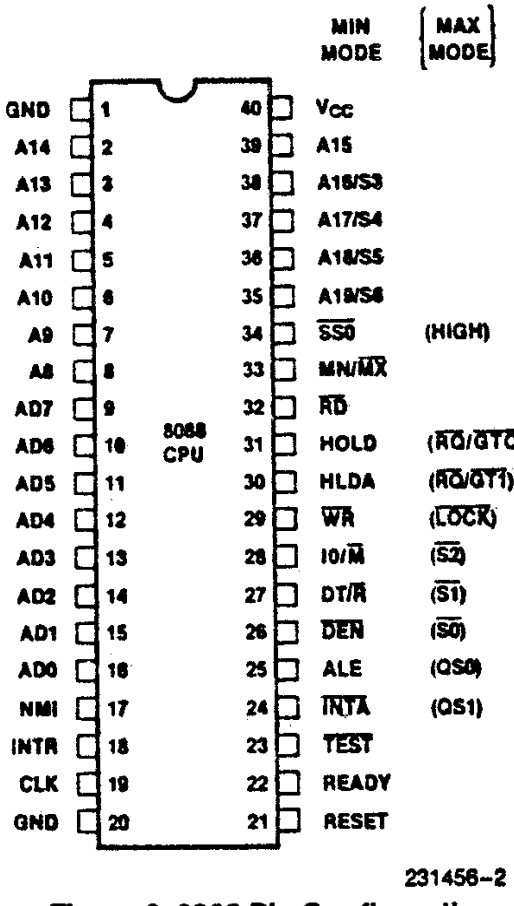


Figure 2. 8088 Pin Configuration

Fig: Functional block diagram of 8088 microprocessor

General Registers

The general registers are categorized into two sets: data and address. The data registers are for calculations; the address registers contain memory addresses and are used to point to the locations in memory where data will be retrieved or stored.

Examining the diagram shows that there are four pairs of registers at the top labeled AH, AL, BH, BL, CH, CL, DH, and DL. These are the data registers. Each of these registers is 8 bits long. Each pair, however, can also operate as a single 16 bit register. AH and AL can operate as a pair referred to as AX. This combining of registers is simply a concatenation, the 8 bits of AL simply tacked to the end of the 8 bits of AH. For example, if AH contains 10110000_2 ($B0_{16}$) and AL contains 01011111_2 ($5F_{16}$), then the virtual register AX contains 1011000001011111_2 ($B05F_{16}$).

Example: If CX contains the binary value 0110 1101 0110 1011_2, what value does CH have?
Answer: CH contains 0110 1101_2.

Intel has given each of these computational registers a name. These names are listed below:

- AX - Accumulator register
- BX - Base register
- CX - Counter register
- DX - Data register

Below the data registers in the block diagram are the address registers: SP, BP, DI, and SI. These are officially referred to as the pointer (SP and BP) and index registers (DI and SI). These registers are used with the segment registers to point to specific addresses in the memory space of the processor. We will address their operation in the section on the segment registers. It is sufficient at this point to say that they act like pointers in the programming language C or C++. Their basic function is as follows:

- SP is the stack pointer and it points to the "top plate" or last piece of data placed on the stack.
- BP (base pointer), SI (source index), and DI (destination index) are all pointers that the programmer has for their own use.

The Flags

Imagine the instrumentation on the dash board of a car. Blinking on and off occasionally behind the speedometer, tachometer, fuel gauge, and such, are a number of lights informally called "idiot lights". Each of these lights has a unique purpose. One comes on when the fuel is low. Another light up when the high beams are on. Another warns the driver of low coolant. There are many more lights, and depending on the type of car you drive, some may even replace a gauge such as oil pressure.

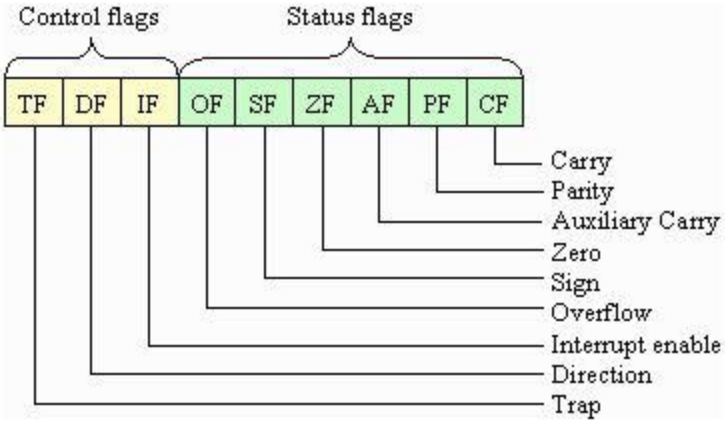
Now let's go back to the processor. There are a number of "idiot lights" that the processor can use, each one based on the result of the previous operation. For example, the addition of two numbers might produce a negative sign, an erroneous overflow, a carry, or a value of zero. Well, that would be four idiot lights: sign, overflow, carry, and zero.

Each of these idiot lights, otherwise known as flags, can be represented with a single bit. If the resulting number had a negative sign, the sign flag would equal 1. If the result was not a negative number, (zero or greater than zero) the sign flag would equal 0. (Side note: Motorola processors more correctly refer to this flag as the negative flag.)

For the sake of organization, these flags are grouped together to form a single number. That number is the ***flags register*** shown at the bottom of the EU section of the processor diagram. The individual bits of the flags are arranged as shown in the figure below:

				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Fig: Flag Register



Example: Assume the flag register is set as shown below after an addition. Using these flags, what can you tell us about the result?

TF	DF	IF	OF	SF	ZF	AF	PF	CF
0	0	0	0	1	0	0	0	1

Answer: As a result of the addition, there was no overflow (OF=0), the result is negative (SF=1), it isn't zero (ZF=0, but you could've also told us that because it is negative), and there was a carry.

Example: If you were to add the binary number 10110101_2 and 10010110_2, how would the flags be set?
Answer: First, let's add the two numbers to see what the result is.

```
  1 0 1 1 0 1 0 1
+ 1 0 0 1 0 1 1 0
1 0 1 0 0 1 0 1 1
```

Now just go from left to right through the status flags.

- OF=1 -- There was an overflow, i.e., adding two negative numbers resulted in a positive number.
- SF=0 -- The result is positive.
- ZF=0 -- The result does not equal zero.
- AF=0 -- For now we won't worry about the auxiliary flag.
- PF=0 -- For now we won't worry about the parity flag.
- CF=1 -- There was a carry.

Arithmetic Logic Unit

As implied by the name, the Arithmetic Logic Unit (ALU) is the computation portion of the EU. Any time arithmetic or logic needs to be performed on numbers, the numbers are sent from the general registers to the ALU, the ALU performs the function, and the result is sent back to the general registers.

EU Control System

The EU Control System is a set of gates that control the timing, passing of data, and other items within the execution unit. It's analogous to a manager in business who doesn't necessarily know the details of the operation, but they plan what happens, where it happens, and when it happens.

Instruction Pointer

All program instructions located in memory are pointed using 16 bits of segment register CS and 16 bits offset contained in the 16 bit instruction pointer (IP). The BIU computes the 20 bit physical address internally using the logical address that is the contents of CS and IP. 16 bit contents of CS will be shifted 4 bits to the left and then adding the 16 bit contents of IP. Thus, all instructions of the program are relative contents of IP. Simply stated, CS contains the base or start of the current code segment, and IP contains the distance or offset from this address to the next instruction byte to be fetched.

Comparison with 8085 microprocessor

Differences between 8085 and 8086

8085	8086
8085 is 8-bit microprocessor	8086 is 16-bit microprocessor
8085 has 16-bit address bus	8086 has 20-bit address bus
Clock speed: 3 MHz	Clock speed can vary between 5, 8 and 10 MHz for three different MP.
8085 can access up to 2^16 = 64 KB of memory	8086 can access up to 2^20 = 1 MB of memory
8085 doesn't have an instruction queue	8086 has instruction queue
8085 does not support pipelined architecture	8086 supports pipelined architecture.
8085 does not support multiprocessing support	8086 supports
8085 can address 2^8 = 256 I/O's	8086 can access 2^16 = 65,536 I/O's
8085 only supports integer and decimal	8086 supports integer, decimal and ASCII arithmetic.
8085 does not support Multiplication and Division	8086 supports.
8085 supports only single operating mode	8086 operates in two modes.
8085 requires less external hardware	8086 requires more external hardware.
The cost of 8085 is low.	The cost of 8086 is high.
In 8085, memory space is not segmented.	In 8086, memory space is segmented.

Differences between 8086 and 8088

8086	8088
The instruction Queue is 6 byte long.	The instruction Queue is 4 byte long.
In 8086, memory divides into two banks -even or lower bank -odd or higher bank	The memory in 8088 does not divide into two banks.
The data bus of 8086 is 16-bit wide	The data bus of 8088 is 8-bit wide.
It has BHE (bar) signal on pin no. 34 & there is no SSO (bar) signal.	It does not have BHE (bar) signal on pin no. 34 & has only SSO (bar) signal. It has no S7 pin.
Control pin in 8086 is M/IO (bar).	Control pin in 8088 is IO/M (bar).
It needs one machine cycle to R/W signal if it is at even location otherwise it needs two.	It needs one machine cycle to R/W signal if it is at even location otherwise it needs two.
In 8086, all address & data Buses are multiplexed.	In 8088, address bus; AD_7- AD_0 buses are multiplexed.
It needs two IC 74343 for de-multiplexing AD_0-AD_{19}.	It needs one IC 74343 for de-multiplexing AD_0-AD_7.
Maximum supply current 360mA.	Maximum supply current 340mA.
Three clock speed: 5, 8, 10 MHz	Two clock speed: 5, 8 MHz

Addressing modes of 8086

Implied (or implicit or Inherent) Addressing mode: The data value/data address is implicitly associated with the instruction. Example: STC ; set carry flag
CLC ; clear (reset) carry flag
AAA ; ASCII adjust after addition

Immediate Addressing mode: Data required for executing the instruction is given directly along with the instruction. Example: MOV AL, 78H
MOV CX, 234AH
ADD AL, 23H

Register Addressing mode: Data required for executing the instruction is present in the register. Example
MOV AL, BL ; transfers the content of BL to AL register pair
MOV DX, BX ; transfers the content of BX to DX register pair

Direct Addressing mode: Data required for executing the instruction is present in the memory location and the effective address of the memory location is given directly along with instruction. Example
MOV AL, [4371H] ; copies 16-bit word into a memory pointed by the displacement address 4371H to the AL register.

Register Indirect Addressing mode: Data required for executing the instruction is present in the memory location and effective address of this memory location is present in the **base** or **index** register along with instruction. Example
MOV AX, [BX] ; copies the word-sized data from data segment offset address indexed by BX into AX

Register Relative Addressing mode: Data required for executing the instruction is present in the memory location and effective address of this memory location is obtained by adding the displacement value with register value. Example
MOV AX, [BX+4]

Base-Plus-Index Addressing mode: Data required for executing the instruction is present in the memory location and effective address of this memory location is obtained by adding base register and index register value. Example
MOV AX, [BX+SI]
MOV [BX+SI], CX

Based-Plus-Index with Displacement Addressing mode: Data required for executing the instruction is present in the memory location and effective address of this memory location is obtained by adding base register, index register and displacement. Example
MOV AX, [BX+DI+4]
MOV [BX+DI+6], CX

Bus structure

A microprocessor unit performs basically four operations: memory read, memory write, I/O read, I/O write. These operations are part of communication between MPU & peripheral devices.
A communication includes identifying peripheral or memory location, transfer of data & control functions. These are carried out using address bus, data bus and control bus respectively. All the buses together are called the system bus.
In case of 8085 MPU we have

- 8 unidirectional address pins
- 8 bi directional multiplexed address/ data pins
- 11 control output pins
- 11 control input pins

Data bus:
The data bus provides a path for data flow between the system modules. It consists of a number of separate lines, generally 8, 16, 32 or 64. The number of lines is referred to as width of the data bus. A

single line can only carry one bit at a time, the number of lines determine how many bits can be transmitted at a time. The width also determines the overall system performance.
An 8 bit data bus would require twice the time required by 16 bit data bus to transmit 16 bit data

8085 – 8 bit data bus
8086 – 16 bit data bus

Address bus:
The address bus is used to designate the source and destination of data in data bus. In a computer system, each peripheral or memory location is identified by a binary number called an address.
The width of the address bus determines the maximum possible memory capacity of the system. The address bus is also used to address I/O ports. Usually higher order bits are used to select particular modules and lower order bit select a memory location or I/O port within a module.

8085 – 16 bit address bus

Thus, maximum amount of memory locations it can address 2^{16} = 65, 536 or 64 Kb

Control bus:
These are group of lines used to control the data and address bus. Since this bus is shared by all the component of the microcomputer system; there must be some control mechanism to distinguish between data and address. The timing signals indicate the validity of data and address information; while command signal specify operations to be performed. Some of the control signals are:

- Memory write
- Memory Read
- I/O write
- I/O read
- ALE
- Interrupt Request
- Interrupt Acknowledge