Segment 2

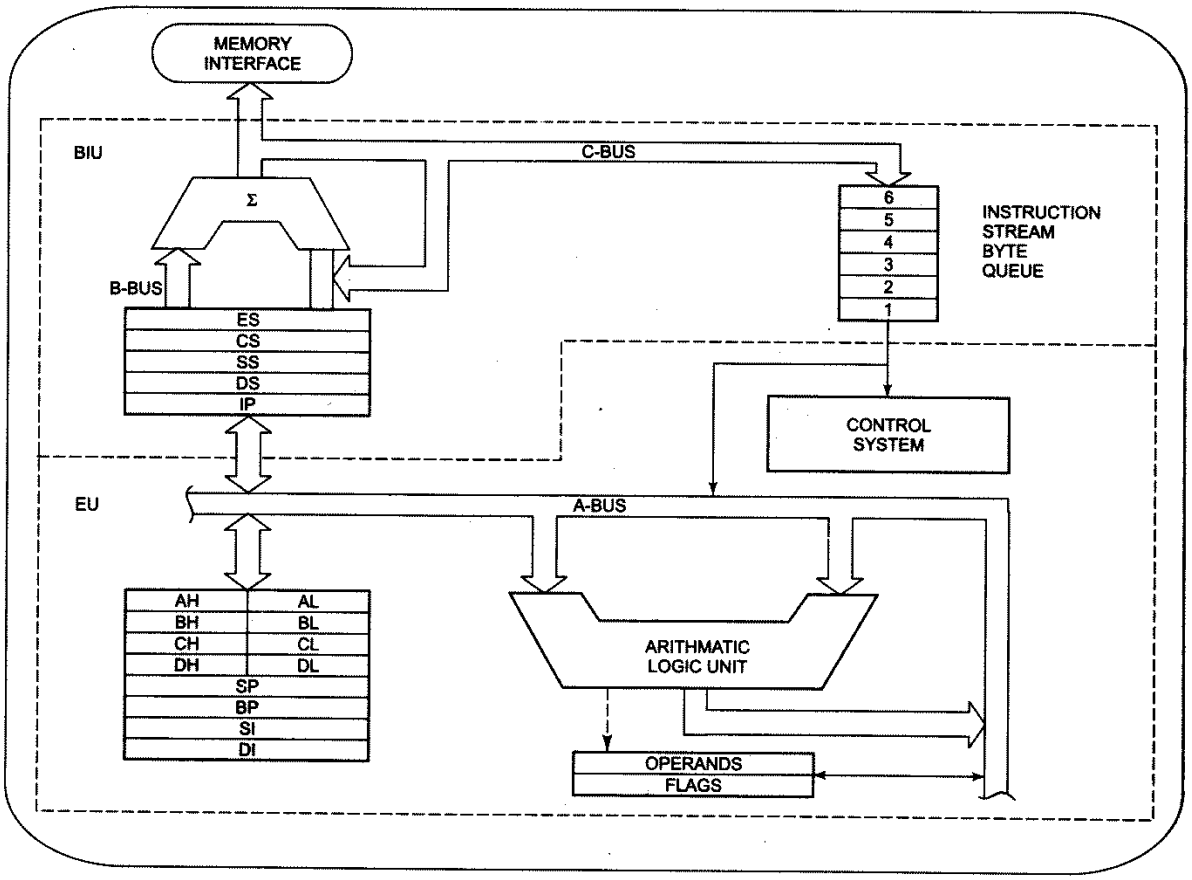
Intel 8086 Microprocessor

8086 Internal Architecture

The 8086 CPU is divided into two independent functional parts:

a) The Bus interface unit (BIU)
b) Execution Unit (EU)

Dividing the work between these two units speeds up processing.



(a) The execution Unit

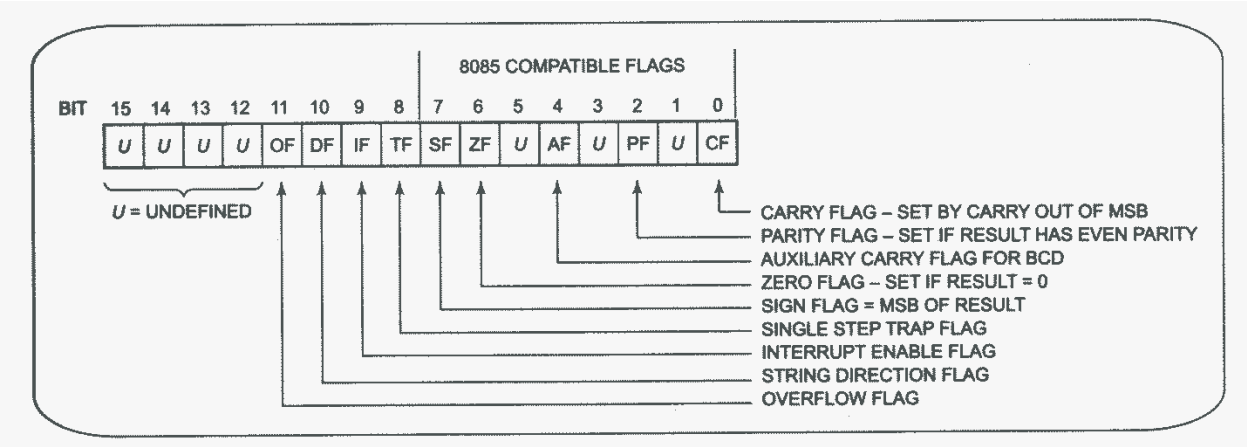
The execution unit of the 8086 tells the BIU where to fetch instructions or data form, decodes instructions and executes instructions. Let's take a look at some of the parts of the execution unit.

Control Circuitry, Instruction decoder and ALU

EU contains control circuitry which directs internal operations. A decoder in the EU translates instructions fetched from memory into a series of actions which the EU carries out. The EU has 16 bit arithmetic and logic unit which can add, subtract, AND, OR, XOR, increment, decrement, complement or shift binary numbers.

Flag Register

A flag register is flip flop that indicates some condition produced by the execution of an instruction or controls certain operations of the EU. A 16-bit flag register in the EU contains nine active flags. Six of the nine flags are used to indicate some condition produced by an instruction. The six conditional flags are the carry flag (CF), the parity flag (PF), the auxiliary carry flag (AF), the zero flag (ZF), the sign flag (SF) and the overflow flag (OF). The six conditional flags are set or reset by the EU on the basis of the results of some arithmetic and logic operation.



The three remaining flags in the flag register are used to control certain operations of the processor. These flags are different from the six conditional flags described above in the way they are set or reset. The control flags are deliberately set or reset with specific instructions you put in your program. The three control flags are the trap flag (TF), interrupt flag (IF) and direction flag (DF).

C (Carry): Carry holds the carry after addition or the borrow after subtraction. The carry flag also indicates error condition, as dictated by some programs and procedures.

P (parity): Parity is a logic 0 for odd parity and a logic 1 for even parity. Parity is count of ones in a number expressed as even or odd. For example, if a number contains three binary one bits, it has odd parity.

A (auxiliary carry): The auxiliary carry holds the carry (half carry) after addition or the borrow after subtraction between bit position 3 and 4 of the result. This highly specified flag bit is tested by the DAA and DAS instructions to adjust the value of AL after a BCD addition or subtraction. Otherwise, the A flag bit is not used by the microprocessor or any other instructions.

Z (Zero): The zero flag shows that the result of an arithmetic or logic operation is zero. If Z=1, the result is zero; if Z=0, the result is not zero. This may be confusing, but that is how Intel decided to name this flag.

S (Sign): The sign flag holds the arithmetic sign of the result after an arithmetic or logic instruction executes. If S=1, the sign bit (the leftmost bit of a number) is set or negative; if S=0, the sign bit is cleared or positive.

T (trap): The trap flag enables trapping through an on chip debugging feature. A program is debugged to find an error or bug. If the T flag is enabled (1), the microprocessor interrupts the flow of the program on conditions as indicated by the debug resisters and control resisters. If the T flag is logic 0, the trapping (debugging) feature is disabled.

I (interrupt): The interrupt flag controls the operation of the INTR (interrupt request) input pin. If I=1, the interrupt pin is enabled. If I=0, the interrupt pin is disabled. The state of the I flag bit is controlled by the STI (set I flag) and CLI (clear I flag) instruction.

D (direction): The direction flag selects either the increment and decrement mode for the DI (direction index) and SI (source index) registers during string instructions. If D=1, the registers are automatically decremented; if D=0; the register are automatically incremented. The D flag is set with the STD (set direction) and clear with CLD (clear direction) instructions.

O (Overflow): Overflows occur when signed numbers are added or subtracted. An overflow indicates that the result has exceeded the capacity of the machine.

General Purpose Registers or Multipurpose Registers or Data Registers

AX, BX, CX and DX are called data registers. These four registers are available to the programmer for general data manipulation.

Even though the processor can operate on data stored in memory, the same instruction is faster (required fewer clock cycles) if the data are stored in registers. The advantage of using internal registers for the temporary storage of data is that, since the data is already in the EU, it can be accessed much more quickly than it could be accessed in external memory. This is why modern processors tend to have a lot of registers.

The high and low bytes of the data registers can be accessed separately. The high byte of AX is called AH, and low byte is AL. Similarly, the high and low bytes of BX, CX and DX are BH and BL, CH and CL, DH and DL respectively. This arrangement gives us more registers to use when dealing with byte-size data.

These four registers, in addition being general-purpose registers, also perform special functions such as the following:

AX (Accumulator Register): It is called 16 bit accumulator while AL is the 8-bit accumulator. AX is the preferred register to use in arithmetic, logic and data transfer instructions because its use generates the shortest machine code. The I/O instruction always uses AX or AL for inputting/ outputting 16 or 8 bit data to or from an I/O port. Multiplication and division instruction also use the AX or AL.

BX (Base Register): It is called base Register because it sometimes holds the offset address of a location in the memory system in all version of microprocessor.

CX Register (Counter Register): It is known as “Counter Register” because some instructions such as SHIFT, ROTATE and LOOP use the content of CX or CL as a counter.

DX Register (Data Register): This general purpose register holds a part of the result from multiplication or part of the dividend before a division and reminder after the division.

(B)THE BIU

The Queue

While the EU is decoding an instruction or executing an instruction which does not require use of the buses, the BIU fetches up to six instruction bytes for the following instructions. The BIU stores these prefetched bytes in a first-in-first-out register set called a queue. When the EU is ready for its next instruction, it simply reads the instruction bytes for the instruction from the queue in the BIU. This is much faster than sending out an address to the system memory and waiting for memory to send back the next instruction byte or bytes. Fetching the next instruction while the current instruction executes is called pipelining.

If an instruction such as "Jump" or "Call" is encountered, the BIU will reset the queue and begin refilling after passing the new instruction to EU.

Segment Registers

The 8086 BIU sends out 20-bit addresses, so it can address any of 2^{20} or, 1, 048, 576 bytes in memory. However, at any given time the 8086 works with only four 65,536 byte (64 Kbyte) segments within this 1,048,576 byte (1-Mbyte) range. Four segment registers in the BIU are used to hold the upper 16 bits of the starting addresses of four memory segments that the 8086 is working with at a particular time.

The four segment registers are:

(a) Code segment (CS) register
(b) Stack segment (SS) register
(c) Extra segment (ES) register, and
(d) Data segment (DS) register.

The figure shows how these four segments might be positioned in memory at a given time.

Code Segment (CS) register

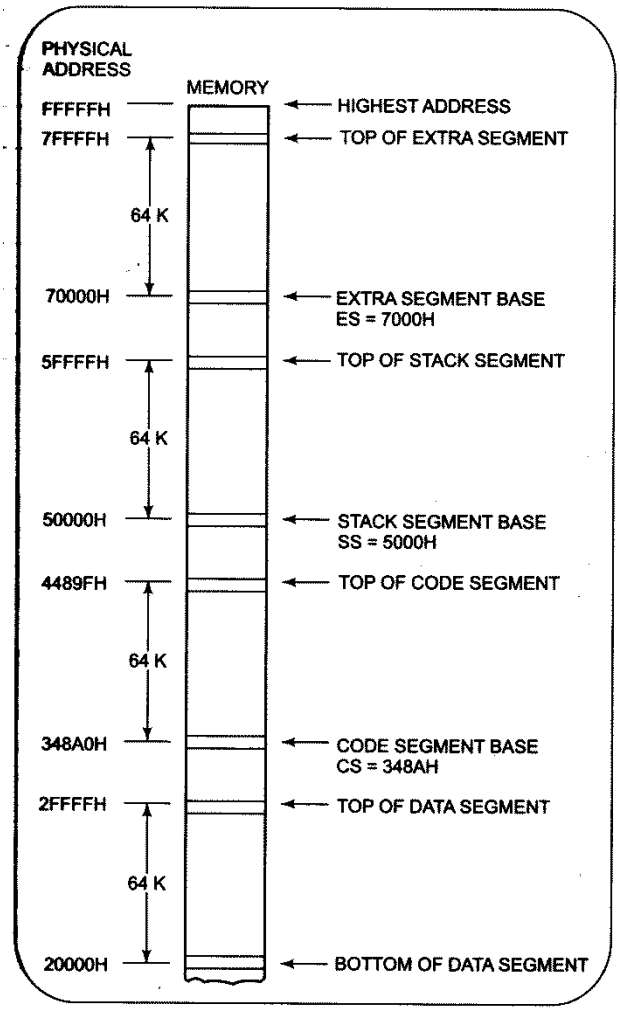
The code segment register holds the upper sixteen bits of the starting address for the segment from which the BIU is currently fetching instruction code bytes. IP (Instruction Pointer) contains the distance or offset from this address to the next instruction byte to be fetched. Note that immediate data are considered as a part of code segment.

Stack Segment (SS) register

A stack is a section of memory set aside to store addresses and data while a subprogram executes. The stack segment register is used to hold the upper 16 bits of the starting address for the program stack.

Extra segment (ES) and data segment register (DS)

The extra segment register and the data segment register are used to hold the upper 16 bits of the starting addresses of two memory segments that are used for data.



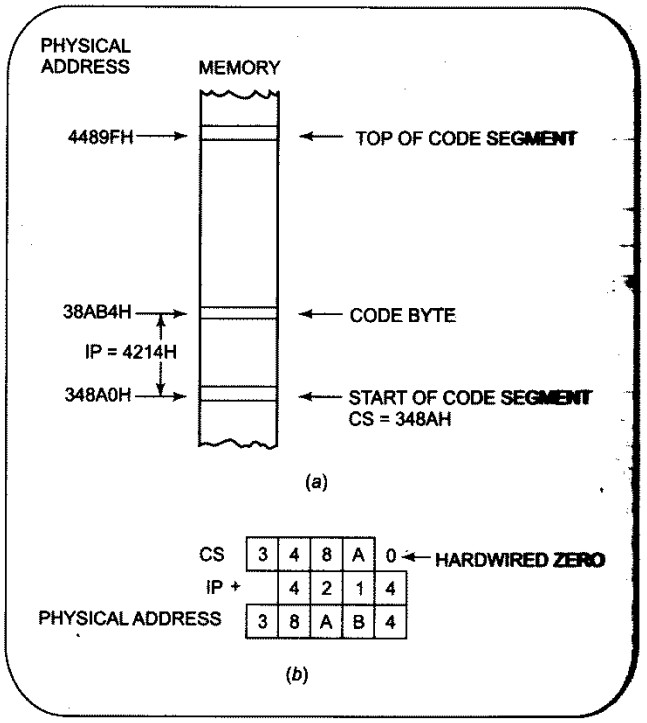
The BIU always inserts zero for the lowest four bits (nibble) of the 20 bit starting address for a segment. If the code segment register contains 348AH, for example, then the code segment will start at address 348A0H. In other words, a 64-Kbytes segment can be located anywhere within the 1-Mbyte address space, but the segment will always start at an address with zeros in the lowest 4 bits.

Instruction Pointer

The instruction pointer register holds the 16 bit address or offset of the next code byte within the code segment. The value contained in the IP is referred to as an offset because this value must be offset from the segment base address in CS to produce the required 20-bit physical address send out by the BIU. The CS register points to the base or start of the current code segment. The IP contains the distance or offset from this base address to the next instruction byte to be fetched.

If the CS register, for example, contains 348AH, the starting address for the code segment is 348A0H. When the BIU adds the offset of 4214H in the IP to this segment base address, the result is a 20-bit physical address of 38AB4H.

To summarize, then, the CS register contains the upper 16 bits of the starting address of the code segment in the one Mbyte address range of the 8086. The instruction pointer register contains a 16 bit offset which tells, where in that 64 Kbyte code segment the next instruction byte is to be fetched from. The actual physical address send to memory is produced by adding the offset contains in the IP register to the segment base represented by the upper 16 bits in the CS register.



Stack segment register and Stack pointer register

A stack, remember, is section of memory set aside to store addresses and data while a subprogram is executing. The 8086 allows you to set aside an entire 64 Kbyte segment as a stack. The upper 16 bits of the starting address for this segment are kept in the stack segment register. The stack pointer (SP) register in the execution unit holds the 16 bit offset from the start of the segment to the memory location where a word was most recently stored on the stack. The memory location where a word was most recently stored is called the top of the stack.

The physical address for a stack read or a stack write is produced by adding the contents of the stack pointer register to the segment base address represented by the uppers 16 bits of the base address in SS.

- Stack is a last-in-first out memory.
- PUSH operation is defined as writing to the top or bottom of the stack. POP operation read from top or bottom of stack.

Pointer and Index registers in the execution unit

In addition to the stack pointer register, the EU contains a 16 bit base pointer register. It also contains a 16 bit source index (SI) register and a 16 bit destination index (DI) register. These three registers can be used for temporary storage of data just as the general purpose registers describe above. However, there main use is to hold the 16 bit offset of a data word in one of the segments. SI, for example, can be used to hold the offset of a data word in the data segment. The

physical address of the data in memory will be generated in this case by adding the contents of SI to the segment base address represented by the 16 bit number in the DS register.

Summary

Functions of EU

- It contains control circuitry which directs internal operations.
- A decoder in EU decodes instructions
- The EU has a 16 bit ALU which can add, subtract, AND, OR, XOR, increment, decrement, complement or shift binary numbers.

Functions of BIU

- To fetch instructions
- To read data from memory and ports
- To write data to memory and I/O ports.
- To interface the 8086 to the outside world
- To provide all external bus operations

8086 Register Set

EU Registers:

The EU has nine 16 bit registers AX, BX, CX, DX, SP, BP, SI, DI and Flag Register.

The 16 bit general registers AX, BX, CX, DX can be considered as 8 bit registers (AH, AL; BH, BL; CH, CL; DH, DL)

BIU Registers:

Segment Registers: ES, CS, SS, DS and Instruction Pointer: IP

Segment Registers and respective offset registers

Segment	*Offset*	*Special Purpose*
CS	IP	Instruction address
SS	SP or BP	Stack address
DS	BX, DI, SI, an 8-bit number or a 16-bit number	Data address
ES	DI for string instructions	String destination address

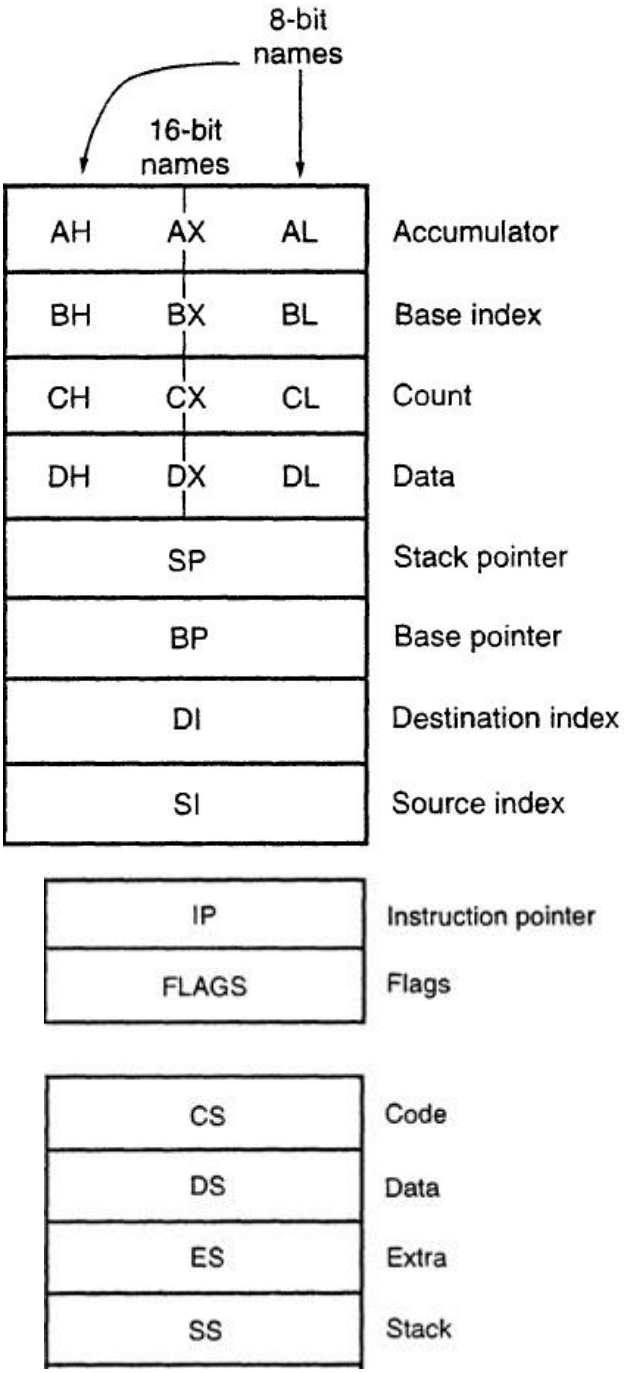


Figure: All registers in 8086