Dr. SUNDUS DHAMAD

Microprocessor Engineering

1. Introduction to the microprocessor and computer

a. A Historical Background.

b. The Microprocessor-Based Personal Computer System.

c. High Level and Low Level Languages.

2. Addressing Modes.

Register, immediate, direct, register indirect, based-index, and base relative-index addressing.

3. Instruction Set and Programming.

a- An Instruction Set.

- Data Movement Instructions.
- Arithmetic and Logical Instructions.
- Program Control Instruction.

b- Programming the Microprocessor.

- Using Debugger
- Using Assembler.

4. The 8086 Hardware Specifications.

a. Internal Architecture.

b. Pin-outs and the pin functions.

c. Clock Generator (8284A).

d. Bus timing.

e. Ready and the wait state.

f. Minimum and Maximum mode, 8288 Bus controller.

5. Memory Interface.

a. Memory Devices: ROM, EEPROM, SRAM, DRAM.

b. Address Decoding.

c. Memory System Design.

d. Memory Interfacing.

6. Input/Output.

Bus buffering and latching, Demultiplexing the busses, The buffered System I/O Instructions, Isolated and Memory-Mapped I/O, I/O Map, Handshaking, I/O Port Address Decoding, 8 and 16-Bit I/O Port, The PPI(8255)1-2 Key Matrix Interface. The 8279 Programmable Keyboard/Display Interface 8254 Programmable Interval Timer. ADC and DAC.

7. Interrupts.

Basic Interrupt Processing, Hardware Interrupts, Expanding the Interrupt Structure, 8259 PIC, Interrupt examples.

8. Direct Memory Access.

References:

- Barry B. Brey, " The Intel Microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, and Pentium Pro processor Architecture, Programming, and Interfacing", 6th Edition, Prentic-Hall Inc., 2003.
- Walter A. Triebe, " The 8086 Microprocessor: Architecture, Software, and Interfacing Techniques", Prentic-Hall Inc., 1998.

1-Introduction to the Microprocessor and Computer

A microprocessor is the chip containing some control and logic circuits that is capable of making arithmetic and logical decisions based on input data, and produce the corresponding arithmetic or logical output.

The microprocessor is the **heart** of any computer, whether it is a **desktop** machine, a **server or a laptop.** They all do approximately the same thing in approximately the same way. No logically enabled device can do anything without it. The microprocessor not only forms the very basis of computers, but also many other devices such as cell phones, satellites, and many other hand held devices. They are also present in modern day cars in the form of microcontrollers. **A microprocessor** is also known as a **CPU** or central processing unit, which is a complete computational engine that is fabricated on a single chip.

❖ A Historical Background:

- As changes in technology were incorporated into the design of computers, their cost and size were reduced dramatically. The earliest computers were as large as an average home and were available only to a select group of scientists.
- The invention of transistors and subsequent advances in their design have made the computer commonly available.
- In the 1940s, CPUs were desgined using vacuum tubes. The vacuum tube was bulky and consumed a lot of electricity or power. The invetation of transistors changed all of that. In 1950s, transistors replaced vacuum tubes in the desgin of computers.
- Then in 1959, the first IC (Integrated Circuit) was invented. In the 1960s the use of IC chips in the design of CPU boards become common. It was not until the 1970s that the entire CPU was put on a single IC chip.
- Intel developed and delivered the first commercially viable microprocessor way back in the early 1970's: the 4004 and 4040 devices. The 4004 was not very powerful and all it could do was **add** and **subtract** with 4-bit data only at a time.

But it was amazing those days that everything was on one chip. The 4004 changed the scene with all its circuitry on a single chip. The 4004 powered one of the first portable electronic calculators named **'Busicom'.**

- The first microprocessor, the 4004, had a 4-bit data bus and was made of 2300 transistors. it was desgined primarily for the hand-held caculator but soon came to be used in applications such as traffic-light controllers.
- The trends in processor design had impact of historical development of microprocessors from different manufacturers. Intel started facing competition from Motorola, MOS Technology, and an upstart company formed by disgruntled Intel employees, Zilog.
- The advances in IC fabrication made during the 1970s made it possible to design microprocessor with 8-bit data bus and a 16-bit address bus. By the late 1970s, the Intel 8080/8085 was one of the most widely used microprocessor, appearing in everything from microwave ovens to homemade computers.
- Sometime between 1976 and 1978 Intel decided that they needed to *leap-frog* the competition and produced a 16-bit microprocessor that offered substantially more power than their competitor's eight-bit offerings. This initiative led to the design of the 8086 microprocessor. The 8086 microprocessor was not the world's first 16-bit microprocessor (there were some oddball 16-bit microprocessors prior to this point) but it was certainly the highest performance single-chip 16-bit microprocessor when it was first introduced. Intel achieved their design goal and programs written for the 8086 were comparable in size to code running on 8-bit microprocessors. However, those design decisions still haunt us today.
- The first microprocessor to make a real splash in the market was the Intel 8088, introduced in 1979 and incorporated into the IBM PC (which appeared around 1982 for the first time). If we are familiar with the PC market and its history, we know that the PC market moved from the 8088 to the 80286 to the 80386 to the 80486 to the Pentium to the Pentium II to the Pentium III to the Pentium 4. Intel

makes all of these microprocessors and all of them are improvements of design base of the 8088. The Pentium 4 can execute any piece of code that ran on the original 8088, but it does it about 5,000 times faster!

Table-1shows the evolution of IC technology. Table-2 helps to understand the differences between the different processors that Intel has introduced over the years.

Table-1: Evolution of IC Technology.

Year	Technology	No. of Devices	Typical products
1947	Invention of transistor	1	-
1950-1960	Discrete components	1	Junction diodes and Transistors
1961-1965	SSI	10-100	Digital Gates , FF.
1966-1970	MSI	100-1000	Counters, MUX., decoders, Adders
1971-1979	LSI	1000-20,000	8-bit MP, RAM, ROM
1980-1984	VLSI	20,000-50,000	DSP, 16-bit MP, 32-bit MP
1985---	ULSI	>50,000	64-bit MP

Table-2: Comparison of different microprocessors

Name	Date	Max clock speed	Address lines	Data width	Max addressable memory
8080	1974	2 MHz	16-bit	8-bit	64KB, no cache
8086	1978	8 MHz	20-bit	16-bit	1MB, no cache
8088	1979	5 MHz	20-bit	8-bit bus/16-bit	1MB, no cache
80286	1982	12.5 MHz	24-bit	16-bit	16MB, no cache
80386	1985	20 MHz	32-bit	32 bit	4GB, no cache
80486	1989	25 MHz	32-bit	32 bit	4GB, 8K level 1
Pentium	1993	100 MHz	32-bit	64-bit/32-bit	4GB, 16K level 1
Pentium Pro	1995	440 MHz	32-bit	64-bit/32-bit	64GB, 16K level 1
Pentium II	1997	266 MHz	32-bit	64-bit/32-bit	64GB, 32K level 1 256K/512 level 2
Pentium III	1999	500 MHz	32-bit	64-bit/32-bit	64GB, 32K level 1 256K/512 level 2
Pentium 4	2000	1.5 GHz	32-bit	64-bit/32-bit	64GB, 32K level 1 256K/512 level 2
Pentium 4 Prescott	2004	3.6 GHz	32-bit	64-bit/32-bit	64GB, 32K level 1 256K/512 level 2

❖ Some terms and concepts for digital computer:

- Bit: is a **Bi**nary digi**t** with a value of one or zero.
- Nibble: data width of four bits.
- Byte: data width of eight bits.
- Word: data width of 16 bits.

- Double word: data width of 32 bits.
- ROM: **R**ead **O**nly **M**emory.
- Bus: a group of lines that carry the same type of information.
- RAM: **R**andom **A**ccess Memory. Or R/WM: **R**ead/**W**rite **M**emory.
- Mnemonic: a combination of letters to suggest the operation of an instruction.
- Program: a set of instructions written in a specific sequence for the computer to accomplish a given task.
- Machine language: is constructed of ones and zeros using binary codes that are stored in computer memory system as groups of instructions called a program.
- Assembly language: is used to simplify the chore of entering binary code into a computer as its instructions.
- Assembler: it allows the programmer to use mnemonic codes such as ADD for addition in place of binary number such as 01000111.
- A hexadecimal number: is a number represented radix 16 or base 16, with each digit representing a value from 0 to 9 and A to F.

❖ **The Microprocessor-Based Personal Computer System:**

Figure-1 shows the block diagram of the personal computer.

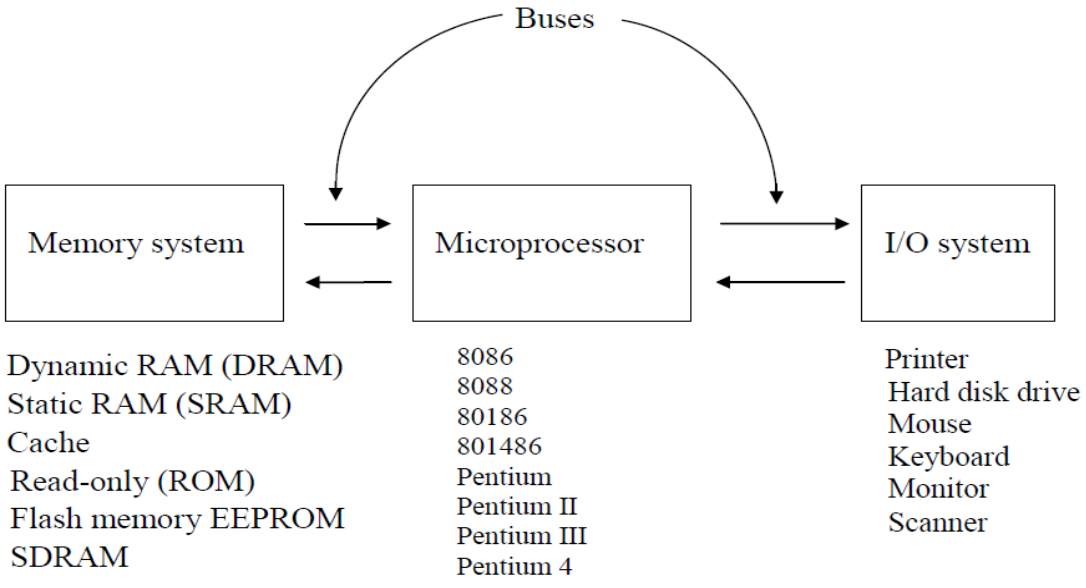


Figure-1: The block diagram of a microprocessor-based computer system.

The block diagram is composed of four parts:

I. **The memory:** The memory structures of all Intel 80X86-Pentium-4 personal computer systems are similar. This includes the first personal computers based upon the 8088 introduced in 1981 by IBM to the most powerful high-speed versions of today based on the Pentium 4.

Figure-2 illustrates the memory map of a personal computer system. The memory system is divided into three main parts: TPA **(T**ransient **P**rogram **A**rea**),** system area, and EMS **(Extended Memory System).** The type of microprocessor in your computer determines whether an extended memory system exists. If the computer is based upon an older 8086 or 8088 (a PC or XT), the TPA and system areas exist, but there is no extended memory area. The PC and XT contain 640K bytes of TPA and 384K bytes of system memory, for a total memory size of 1M bytes. We often call the first 1M byte of memory the real or conventional memory system because each Intel microprocessor is designed to function in this area by using its real mode of operation.

Computer systems based on the 80286 through the Pentium-4 not only contain the TPA (640K bytes) and system area (384K bytes), they also contain extended memory. These machines are often called AT class machines.

- **The TPA:** The transient program area (TPA) holds the DOS operating system and other programs that control the computer system. The TPA also stores DOS application programs. The length of the TPA is 640K bytes. Figure-3 shows the memory map of the TPA.
- **The System Area:** The system area contains programs on either a ROM or flash memory, and areas of RAM for data storage. Figure-4 shows the memory map of the system area.

Note: The video card in some computer uses memory locations E1800000H-E2FFFFFF in Windows for its video memory aperture.

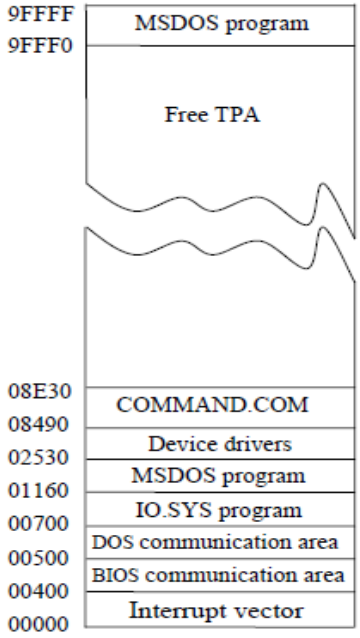


Figure-3: The memory map of the TPA area of a PC.

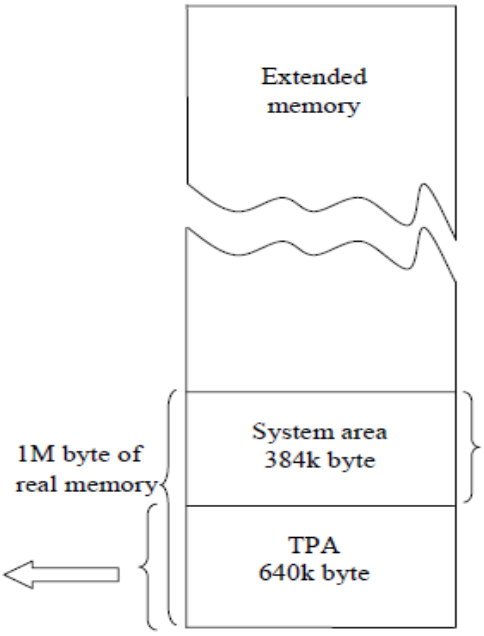


Figure-2: The memory map of the PC.

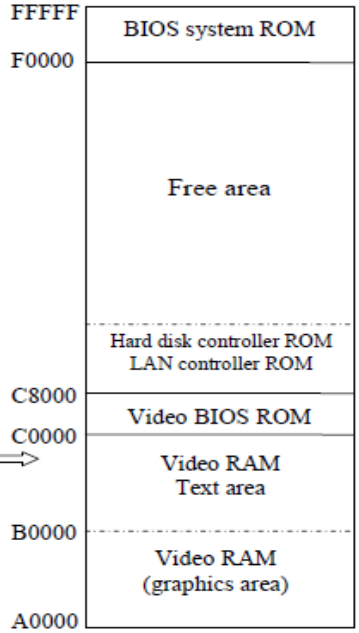


Figure-4: The system area of a typical PC.

II. I/O System:

The I/O devices allow the microprocessor to communicate between itself and the outside world. The I/O space in a computer system extends from port 0000H to port FFFFH. The I/O space allows the computer to access up to 64K different 8-bit I/O devices. Figure-5 shows the I/O map found in many personal computer systems. The I/O area contains two major sections. The area below I/O location 0400H is considered reserved for system devices. The remaining area is available I/O space for expansion or newer devices.

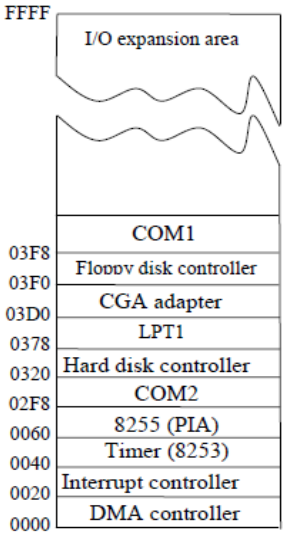


Figure-5: The I/O map of a personal computer.

III. The Microprocessor:

The microprocessor, sometimes referred to as the **CPU** (**C**entral **P**rocessing **U**nit), is the controlling element in a computer. The microprocessor performs three main tasks:

1. data transfer between itself and the memory or I/O system,
2. simple arithmetic and logic operations, and
3. program flow via simple decisions.

IV. Buses:

A bus is a number of wires organized to provide a means of communication among different elements in a microcomputer system. Figure-6 shows the buses of 8086 microprocessor, these buses are:

• **Address bus:** the address bus is a group of 20-bit (A0-A19). The address bus is **unidirectional.**

• **Data bus:** the data bus is a group of 16 lines. These lines are bidirectional.

• **Control bus:** It contains lines that select the memory or I/O and cause them to perform a read or write operation.

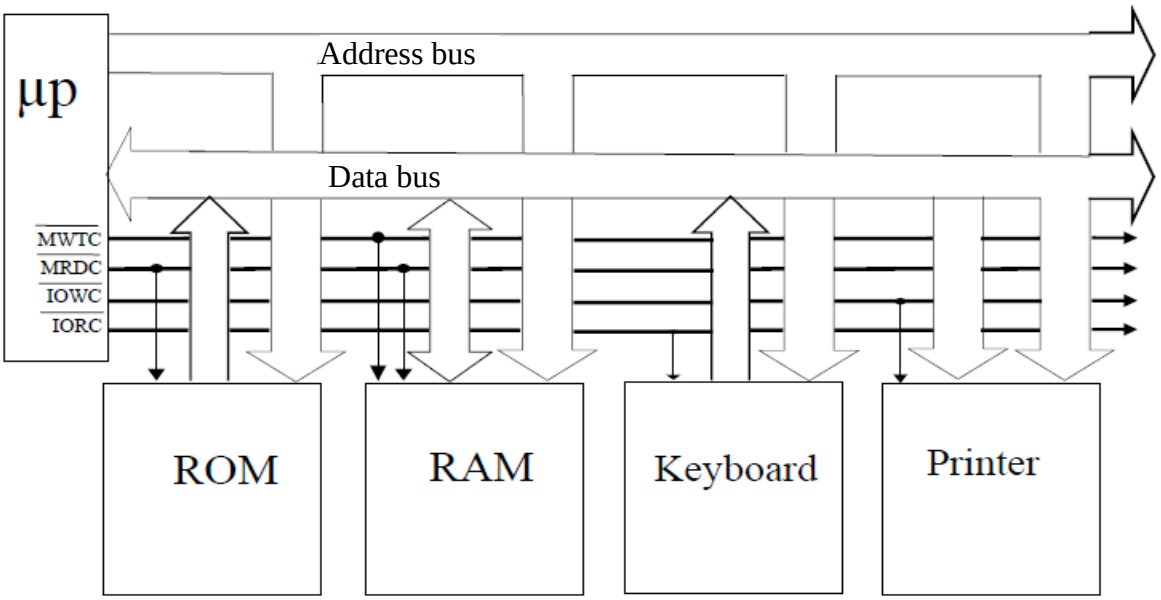


Figure-6: The block diagram of computer system showing the buses structure.

2-Micro-architecture of the 8086 Microprocessor:

The micro-architecture of a processor is its internal architecture, or it is the circuit building blocks that implement the software and hardware architectures of the 8086 microprocessors. The micro-architecture of the 8086 microprocessors employs *parallel processing;* they are implemented with several simultaneously operating processing units.

Features of 8086 Microprocessor:

1. Intel 8086 was launched in 1978.
2. It was the first 16-bit microprocessor. It's ALU, internal registers works with 16-bit binary word.
3. This microprocessor had major improvement over the execution speed of 8085.
4. It is available as 40-pin Dual-Inline-Package (DIP).
5. It is available in three versions:
   a. 8086 (5 MHz)
   b. 8086 (8 MHz)
   c. 8086 (10 MHz)
6. It consists of 29,000 transistors.
7. 8086 has a 16bit data bus. It can read or write data to a memory/port either 16bits or 8 bit at a time.
8. 8086 has a 20-bit address bus which means, it can address up to 2^{20} = 1MB memory location.

Figure-7 shows the block diagram of Intel 8086. The 8086 CPU is divided into two independent functional units:

1. Bus Interface Unit (BIU)

2. Execution Unit (EU)

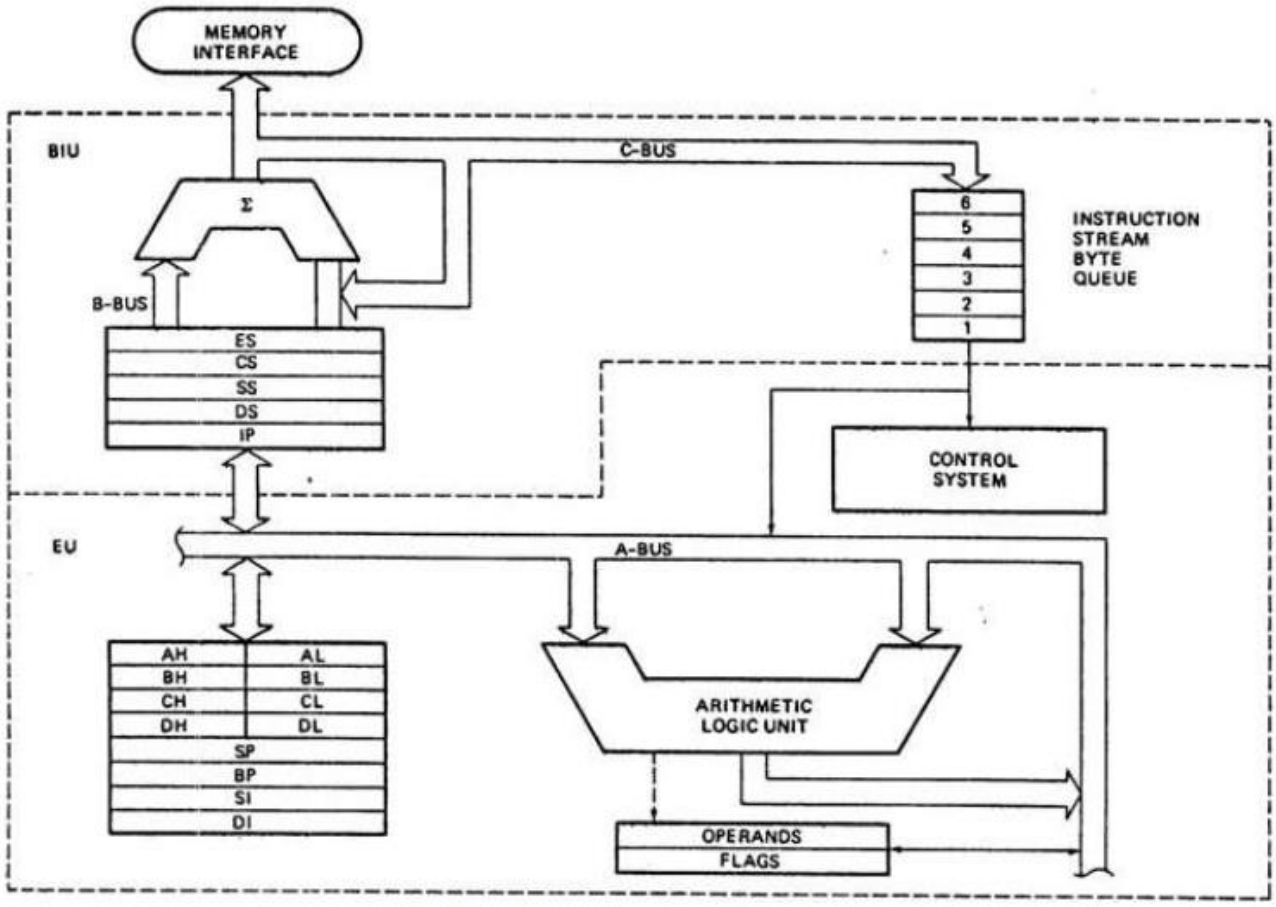


Figure-7 : Block Diagram of Intel 8086

Each unit has dedicated functions and both operate at the same time. In essence, this parallel processing effectively makes the fetch and execution of instructions independent operations. This results in efficient use of the system bus and higher performance for 8086 microcomputer systems.

The BIU: It is the 8086's connection to the outside world. The BID is responsible for performing all external bus operations, such as:

- ✓ Fetch the instruction or data from memory.
- ✓ Write the data to memory.
- ✓ Write the data to the port.
- ✓ Read data from the port.

For instance, it is responsible for instruction queuing and address generation. The BIU uses a mechanism known as an ***instruction queue*** to implement a pipelined

architecture. This queue permits the 8086 to prefetch up to 6 bytes of instruction code. The BIU is free to read the next instruction code when:

• The queue is not full-that is, it has room for at least 2 more bytes.

• The execution unit is not asking it to read or write data from memory.

Instruction Queue

1. To increase the execution speed, BIU fetches as many as six instruction bytes ahead to time from memory.

2. All six bytes are then held in First-In-First-Out (FIFO) 6 byte register called instruction queue.

3. Then all bytes have to be given to EU one by one.

4. This pre fetching operation of BIU may be in parallel with execution operation of EU, which improves the speed execution of the instruction.

Figure-8 shows the BIU contains the segment registers, the instruction pointer, the address generation adder, bus control logic, and an instruction queue.

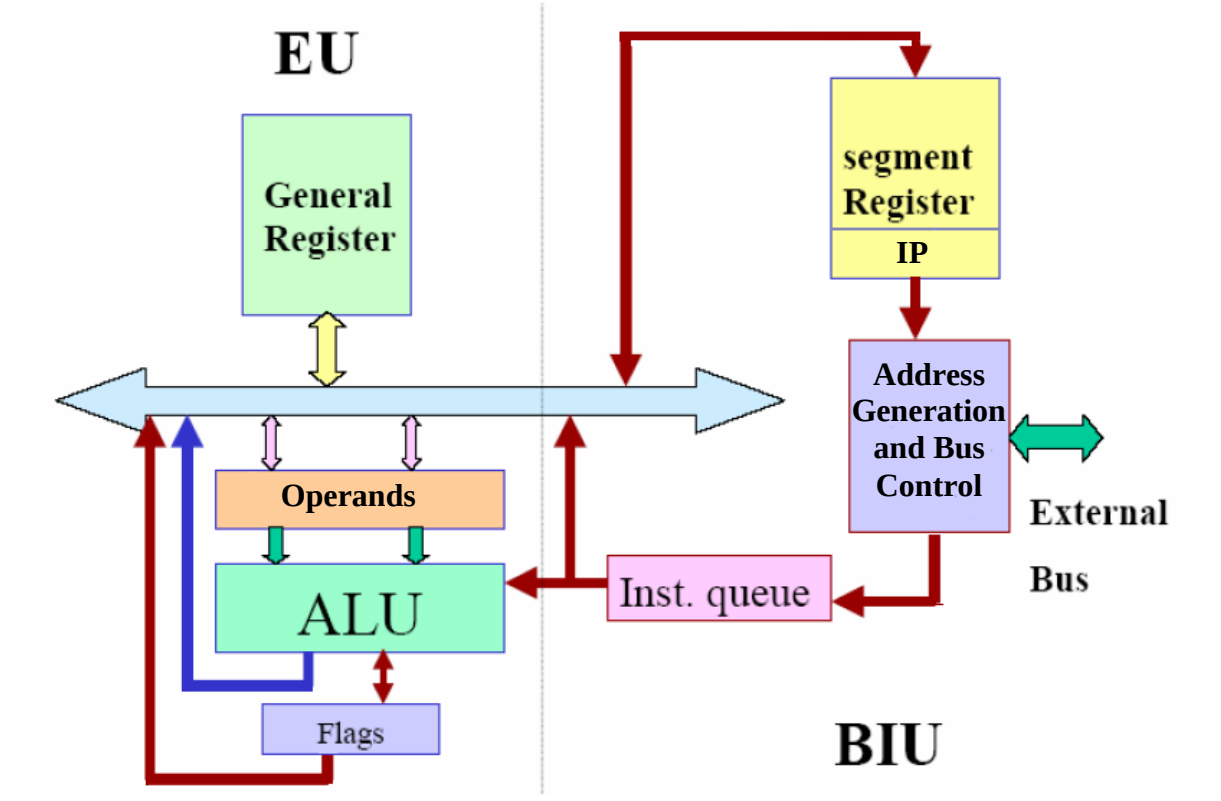


Figure-8: Execution and bus interface units.

The EU: The execution unit is responsible for decoding and executing instructions. EU, as shown in Figure-8, consists of the *arithmetic logic unit* (ALU), status and control flags, general-purpose registers, and temporary-operand registers.

The EU accesses instructions from the output end of the instruction queue and data from the general-purpose registers or memory. It reads one instruction byte after the other from the output of the queue, decodes them, generates data addresses if necessary, passes them to the BIU and requests it to perform the read or write operations to memory or I/O, and performs the operation specified by the instruction.
In other wards, the functions of execution unit are:

- To tell BIU where to fetch the instructions or data from.
- To decode the instructions.
- To execute the instructions.

The ALU performs the arithmetic, logic, and shift operations required by an instruction. During execution of the instruction, the EU may test the status and control flags, and updates these flags based on the results of executing the instruction. If the queue is empty, the EU waits for the next instruction byte to be fetched and shifted to the top of the queue.

3- Software Model of the 8086 Microprocessor:

To be able to program a microprocessor, one does not need to know all of its hardware architectural features. What is important to the programmer is to know the various registers within the device and to understand their purpose, functions, operating capabilities, and limitations.

Figure-9 illustrates the software architecture of the 8086 microprocessor. From this diagram, we see that it includes fourteen l6-bit internal registers: the *instruction pointer* (IP), four *data registers* (AX, BX, CX, and DX), two *pointer registers* (BP and SP), two *index registers* (SI and DI), four *segment registers* (CS, DS, SS, and ES) and *status register* (SR), with nine of its bits implemented as status and control flags.

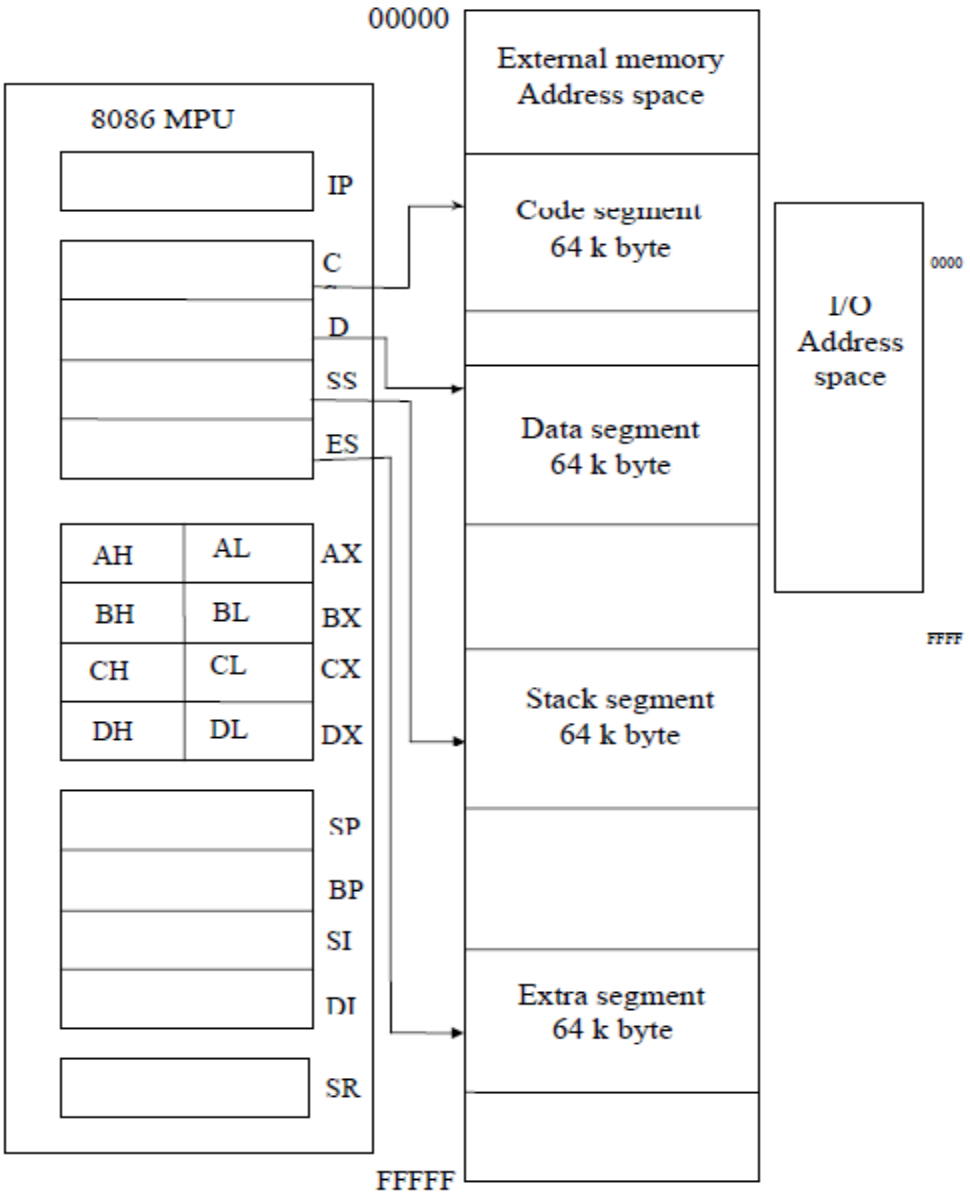
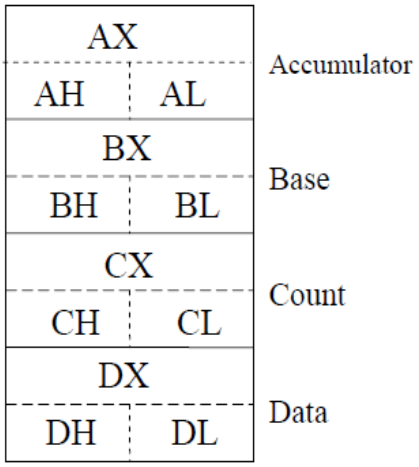


Figure-9: Software model of 8086 microprocessor.

1) **Instruction Pointer (IP):** IP is 16 bits in length and identifies the next word of instruction code to be fetched from the current code segment however; it contains the offset of the instruction code instead of its actual address. This is because IP and CS are both 16 in length, but a 20-bit address is needed to access memory. The value of the next address for the next code access is often denoted as CS: IP.

Actual address = CS + IP.

2) **Data Register:** Figure-10 shows the 8086 has four **general-purpose data registers.** Notice that they are referred to as the accumulator register (A), the base register (B), the count register (C), and the data register (D). Any of the general-purpose data registers can be used as the source or destination of an operand during an arithmetic operation such as ADD or a logic operation such as AND.



(a)

Regester	Operation
AX	Word multiply, word divide, word I/O
AL	Byte multiply, byte divide, byte I/O, translate, decimal arithmetic.
AH	Byte multiply, byte divide.
BX	Translate
CX	String operation, loops.
CL	Variable shift and rotate.
DX	Word multiply, word divide, indirect I/O.

(b)

Figure-10: (a) General-purpose data registers. (b) Dedicated register functions.

3) **Pointer and Index Registers**: Figure-11 shows these registers. These registers are four general-purpose registers: two pointer registers and two index registers. They store what are called **offset addresses**. An **offset address** selects any location within 64 k byte memory segment.

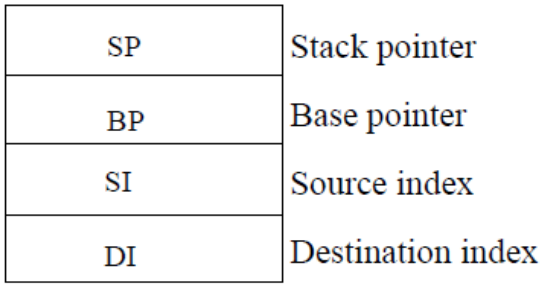


Figure-11: pointer and index registers.

4) **Segment Registers and Memory Segmentation:** The 8086 microprocessor operate in the real mode memory addressing. **Real mode operation** allows the microprocessor to address only the first 1M byte of memory space-even if it is the Pentium 4 microprocessor. Note that the first 1M byte of memory is called either the **real memory** or **conventional memory** system**.** Even though the 8086 has a 1M byte address space, not all this memory is active at one time. Actually, the 1M bytes of memory are partitioned into 64K byte (65,536) **segments.** The 8086-80286 microprocessors allow four memory segments a. Figure-12 shows these memory segments. Note that a memory segment can touch or even overlap if 64K bytes of memory are not required for a segment. Think of segments as windows that can be moved over any area of memory to access data or code. Also note that a program can have more than four segments, but can only access four segments at a time.

In the real mode a combinational of a segment address and offset address access a memory location. All real mode memory address must consist of a segment address plus an offset address. The microprocessor has a set of rules that apply to segments whenever memory is addressed. These rules define the segment register and offset register combination (see Table-3). For example, the code segment register is always used with the instruction pointer to address the next instruction in a program. This combination is CS: IP. The code segment register defines the start of the code segment and the instruction pointer locates the next instruction within the code segment.

This combination (CS: IP) locates the next instruction executed by the mi-coprocessor. Figure-13 shows an example that if CS = 1400H and IP = 1200H, the microprocessor fetches its next instruction from memory location:

Physical Address = Segment Base Address*10+Offset (Effective) Address

PA = SBA * 10 + EA

= 1400H*10 + 1200H

=15200H.

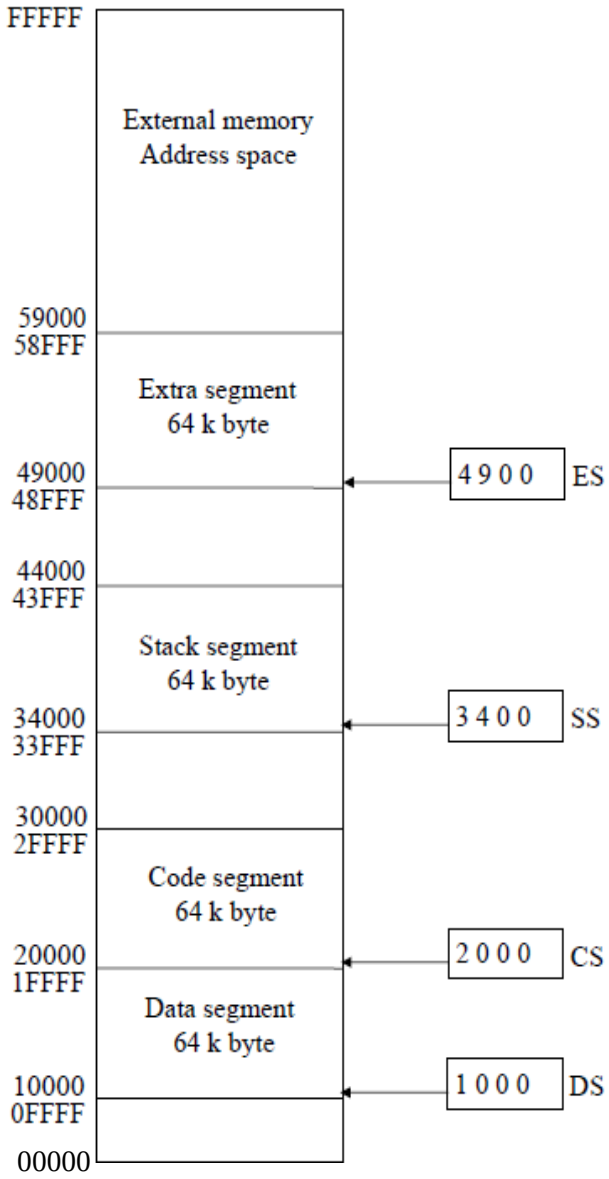


Figure-12: A memory system showing the placement of four memory segments.

Table-3: 8086 default 16 bit segment and offset address

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

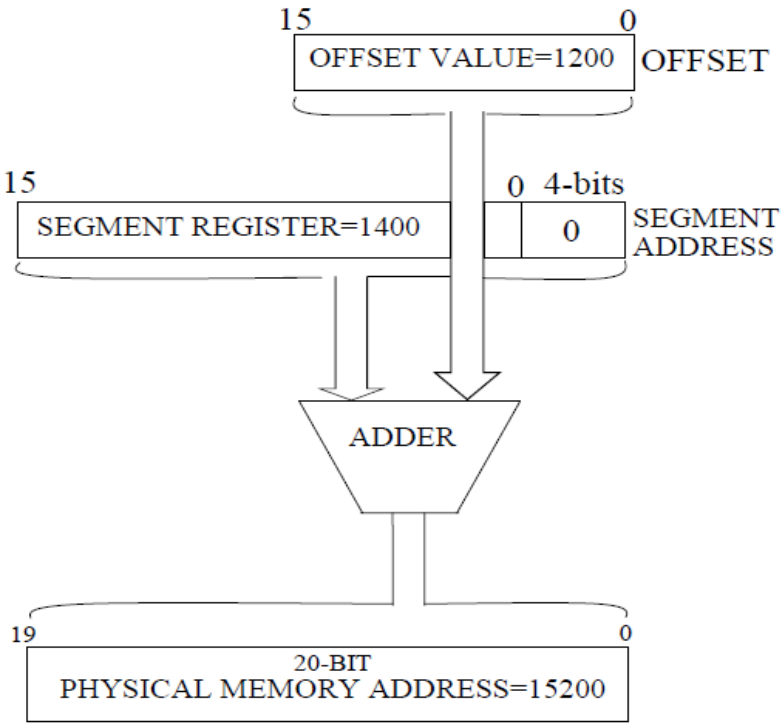


Figure-13: generating a physical address.

Table-4 shows more examples of memory addresses.

Table-4: examples of memory addresses.

Segment register	Offset address	Physical address
002A	0023	002C3
4900	0A23	49A23
1820	FE00	28000

Note:

The Stack: The stack is implemented in the memory of 8086, and it is used for temporary storage. Starting address of stack memory (top of the stack) obtained from the contents of the stack pointer (SP) and the stack segment (SS) (SS: SP). Figure-14 shows the stack region for SS = 0400H and SP = A000H. Data transferred to and from the stack are **word-wide**, not byte-wide. Whenever a word of data is pushed onto the top of the stack, the high-order 8 bits are placed in the location addressed by SP-1.

The low-order 8 bits are placed in the location addressed by SP-2. The SP is then decremented by 2. Whenever data are popped from the stack, the low-order 8 bits are removed from the location addressed by SP. The high-order 8 bits are removed from the location addressed by SP + 2. The SP is then incremented by 2.

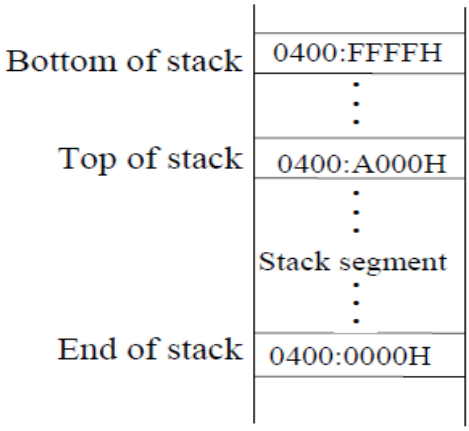


Figure-14: The stack region.

5) **Flag (Status) Register:** This register is another 16-bit register within the 8086. Figure-15 shows the organization of this register. Notice that just nine of its bits are implemented. Six of these bits represent status flags:

- ❖ The carry flag (CF): CF is set if there is a carry-out or a borrow-in for the most significant bit of the result during the execution of an instruction. Otherwise, CF is reset.
- ❖ The parity flag (PF): PF is set if the result produced by the instruction has even parity-that is, if it contains an even number of bits at the 1 logic level. If parity is odd, PF is reset.
- ❖ The auxiliary carry flag (AF): AF is set if there is a carry-out from the low nibble into the high nibble or a borrow-in from the high nibble into the low nibble of the lower byte in a l6-bit word. Otherwise, AF is reset.
- ❖ The zero flag (ZF): ZF is set if the result produced by an instruction is zero. Otherwise, ZF is reset.

- The sign flag (SF): The MSB of the result is copied into SF. Thus, SF is set if the result is a negative number or reset if it is positive.
- The overflow flag (OF): When OF is set, it indicates that the signed result is out of range. If the result is not out of range, OF remains reset.

The other three flags are control flags. These three flags provide control functions of the 8086 as follows:

- The trap flag (TF): If TF is set, the 8086 goes into the single-step mode of operation.
- The interrupt flag (IF): The IF controls the operation of the INTR (interrupt request) input pin If IF=1, the INTR pin is enabled; If IF=0, the INTR pin is disabled. The state of IF bit is controlled by the STI **(set IF)** and CLI **(clear IF)** instructions.
- The direction flag (DF): The direction flag select either the increment or decrement mode for the DI and/or SI registers during string instructions. If DF=1, the registers are automatically decremented; if DF=0, these registers are automatically incremented. The DF is set with STD **(set direction))** and cleared with CLD **(clear direction)** instructions.

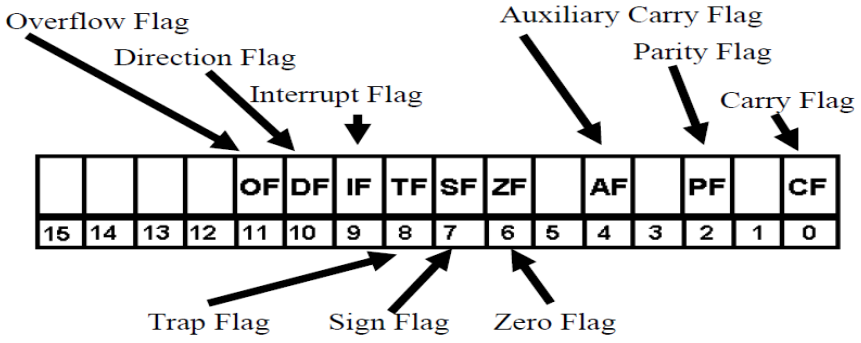


Figure-15: Flag Register.