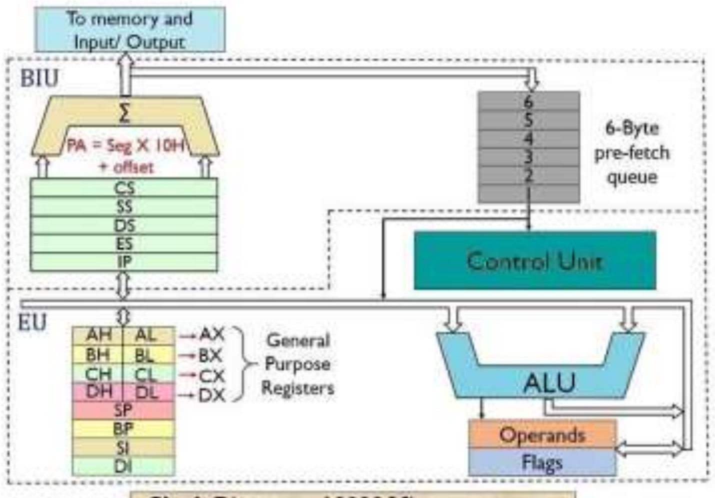
Software Systems-1

BASIC COMPONENTS OF COMPUTER SYSTEM- PART 2

Dr. Azam E. Al-Rawachy
COLLEGE OF COMPUTER SCIENCE AND MATHEMATICS

1 Introduction

The number of bits processed at one time is the processor's register size (word size). Intel's 8086 processor's register size is 16 bits or 2 bytes. Today's CPUs have register sizes of 64 or 128 bits.



Block Diagram of 8086 Microprocessor

Buses Processors operate on 1s and 0s. The 1s and 0s must travel from one place to another inside the processor, as well as outside to other chips. To move the 1s and 0s around, electronic lines called a **bus** are used. The electronic lines inside the CPU are known as the **internal data bus** or system bus. In the 8086, the internal data bus comprises 16 separate lines, with each line carrying one 1 or zero 0. The word size and the number of lines for the internal data bus are equal. The 8086, for example, has a 16-bit word size, and 16 lines carry 16 bits on the internal data bus. In today's processors, 64 or 128 internal data bus lines operate concurrently.

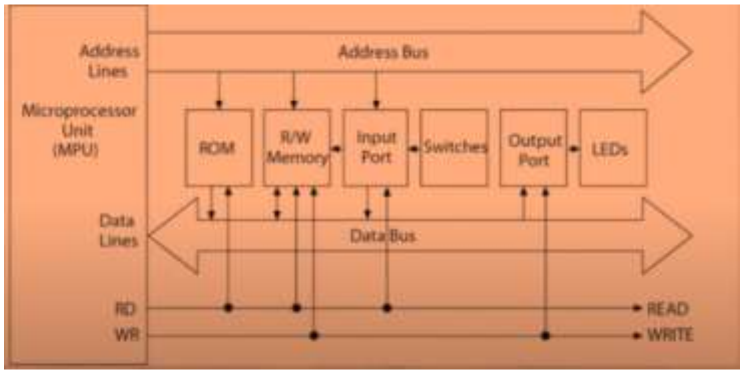
For a CPU to communicate with devices in the outside world, such as a printer, the 1s and 0s travel on the **external** data **bus**. The external data bus connects the processor to adapters, the keyboard, the mouse, the hard drive, and other devices.

Three types of the bus are used:

- **Address bus -** carries memory addresses from the processor to other components such as primary storage and input/output devices. The address bus is **unidirectional**. The ***number of address lines determines the number of memory locations that the CPU can address***. If the CPU has ***N*** address lines, then it can directly address $\boldsymbol{2^N}$ memory locations.

- **Data bus -** carries the data between the processor and other components. The data bus is bidirectional. This means that the CPU can read data in on these lines from memory or a port as well as send data out on these lines to a memory location or a port. Many devices in a system will have their outputs connected to the data bus, but the outputs of only one device at a time will be enabled.

- **Control bus -** carries control signals from the processor to other components. The control bus also carries the clock's pulses. The control bus is unidirectional. Typical control bus signals are memory read, memory write, I/O read, and I/O writer. To read a byte of data from a memory location, for example, the CPU sends out the address of the desired byte on the address bus and then sends out a memory read signal on the control bus.

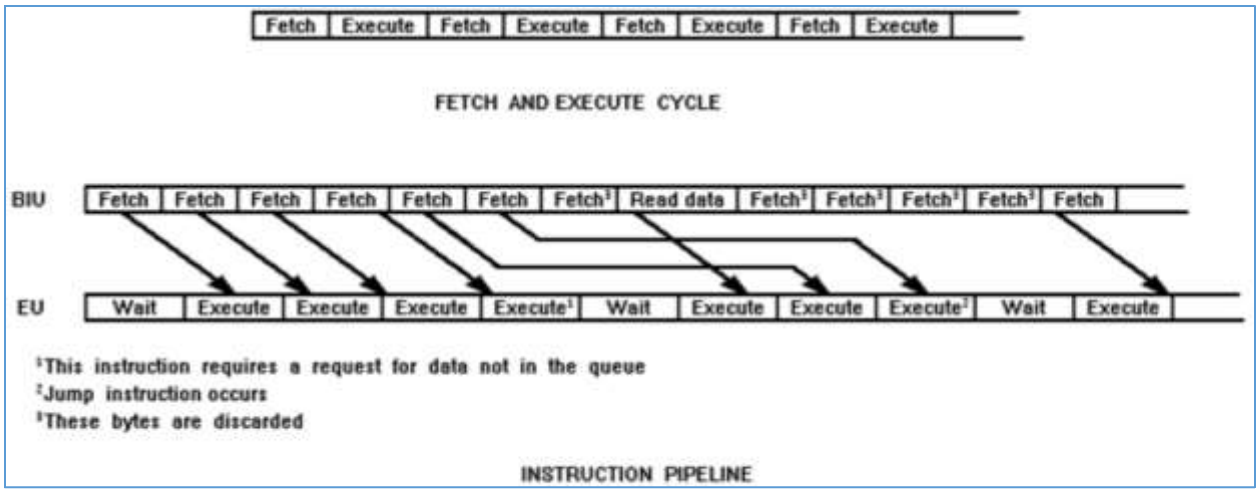


2 Working Principles of Intel 8086 Microprocessor

Execution of instructions can be used to explain the working principles of the microprocessor. This is given below:

- The BIU outputs the contents of the instruction pointer register(IP) onto the address bus, causing the selected byte or word to be read into the BIU.
- Register IP is incremented by 1 to prepare for the next instruction fetch.
- Once inside the BIU, the instruction is passed to the queue.
- Assuming that the queue is initially empty, the EU immediately draws this instruction from the queue and begins execution.

- While the EU is executing this instruction, the BIU proceeds to fetch a new instruction. Depending on the execution time of the first instruction, the BIU may fill the queue with several new instructions before the EU is ready to draw its next instruction.



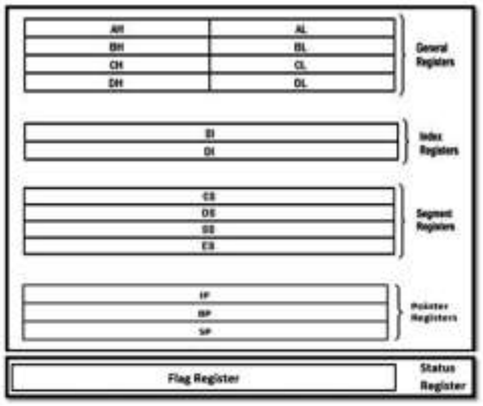
The BIU is programmed to fetch a new instruction whenever the queue has room for two additional bytes. There are three conditions that will cause the EU to enter a "wait" mode or BIU to suspend fetching.

1. When an instruction requires access to a memory location not in the queue. The BIU must suspend fetching instructions and output the address of this memory location. After waiting for the memory access, the EU can resume executing instruction codes from the queue (and the BIU can resume filling the queue).
2. When the instruction to be executed is a "jump" instruction. In this case control is to be transferred to a new (**non-sequential**) address. The queue, however, assumes that instructions will always be executed in sequence and thus will be holding the "wrong" instruction codes. The EU **must wait** while the instruction at the jump address is fetched. **Note that any bytes presently in the queue must be discarded (they are overwritten**).
3. During execution of instructions that are slow to execute. For example, the instruction AAM (ASCII Adjust for Multiplication) requires 83 clock cycles to complete. At four cycles per instruction fetch, the queue will be completely filled during the execution of this single instruction.

2.1 Arithmetic and Logic Unit

The "arithmetic and logic unit" (ALU) performs math computations, such as subtraction, addition, division, and Boolean functions. The ALU also executes comparisons and logic testing.

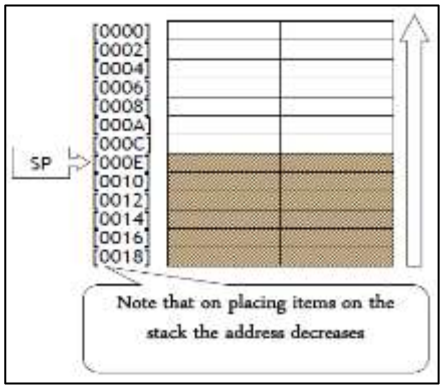
2.2 CPU Registers



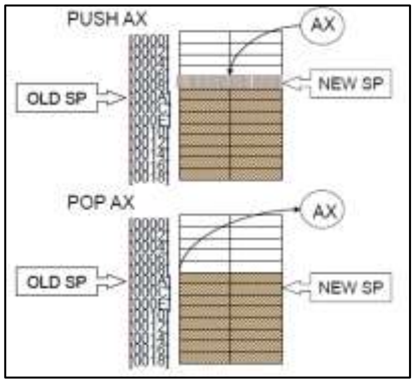
Microprocessors have temporary data holding places called registers. These memory areas maintain data, such as computer instructions, storage addresses, characters, and other data. Some computer instructions may require the use of certain registers as part of a command. Each register has a specific function, such as an instruction register, program counter, and accumulator. We can classify the processors based on the structure of these registers and how the processor uses them. Typically, we can divide the registers into:

- **General-purpose registers.**
- **Special-purpose registers.**

The four **general-purpose registers** are the **AX, BX, CX, and DX** registers. All of the general-purpose registers can be treated as a 16-bit quantity or as two 8 bit quantities.



SP: The Stack pointer keeps track of the position of the last item placed on the stack (i.e. the Top of Stack). It is organized in words, (i.e. ***two bytes at a time***). Thus the stack pointer is **incremented or decremented by 2**. *The Stack Pointer points at **the last occupied locations on the stack.***



The two set of instructions which explicitly modify the stack are the ***PUSH*** (which places items on the stack) and the ***POP*** (which retrieves items from the stack). In both cases, the stack pointer is adjusted accordingly to point always to the top of stack. Thus, PUSH AX means SP=SP-2 and AX -> [SP].

POP AX means [SP] ➔ SP=SP+2.

- **BP:** is a 16-bit register pointing to data in stack segment. BP register is usually used for based indexed or register indirect addressing. Also, it acts as an offset for Stack Segment (SS).
- **SI:** This is the source index register. It is 16 bits. It is used in the pointer addressing of data and as a source in some string related operations. **Its offset is relative to the data segment (DS)**.
- **DI:** This is the destination index register. It is 16 bits. It is used in the pointer addressing of data and as a destination in some string related operations. **It's offset is relative to the extra segment (ES)**.

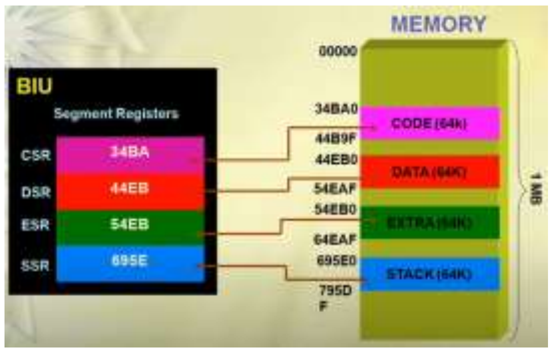
2.3 SEGMENT REGISTERS

CS - points at the segment containing the current program.

DS - generally points at the segment where variables are defined.

ES - extra segment register, it's up to a coder to define its usage.

SS - points at the segment containing the stack.

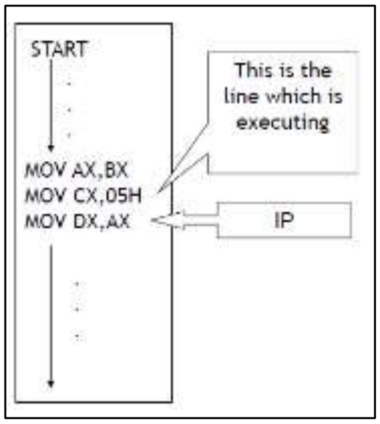


Although it is possible to store any data in the segment registers, this is never a good idea. The segment registers have a very special purpose - pointing at accessible blocks of memory. Segment registers work together with a general-purpose register to access any memory value. For example, if we would like to access memory at the physical address 12345h (hexadecimal), we should set the DS = 1230h and SI = 0045h. This is good, since this way we can access much more memory than with a single register that is limited to 16-bit values. CPU calculates physical address by multiplying the segment register by 10h and adding a general-purpose register to it (1230h * 10h + 45h = 12345h).

2.4 Special Purpose Registers

2.4.1 The Instruction Pointer (IP)

- The computer keeps track of the next line to be executed by keeping its address in a special register called the Instruction Pointer (IP).
- This register is relative to CS as segment register and points to the next instruction to be executed.
- The contents of this register is updated with every instruction executed.
- Thus a program is executed sequentially line by line.

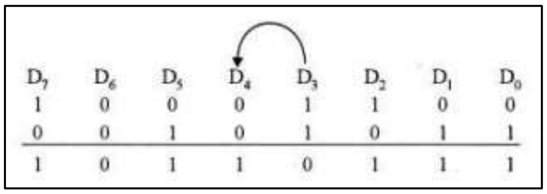


2.4.2 FLAGS Register

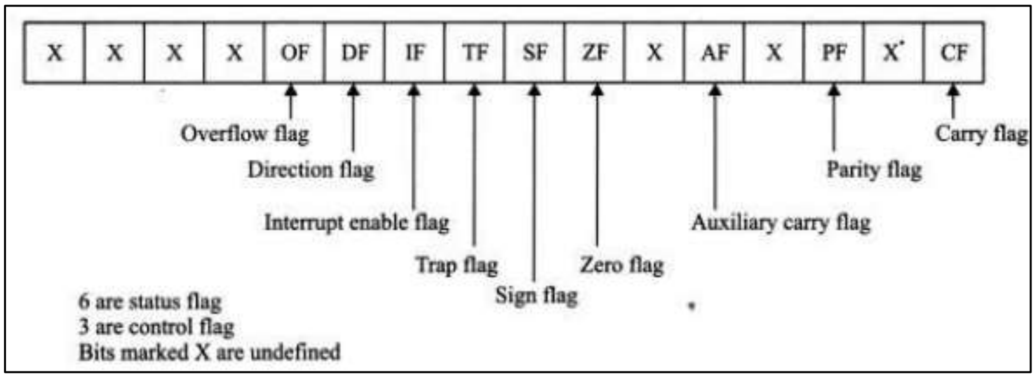
The Flags register consists of individual binary bits that control the operation of the CPU or reflect the outcome of some CPU operation. Some instructions test and manipulate individual processor flags.

1. **Control Flags:** Control flags control the CPU's operation. For example, they can cause the CPU to break after every instruction executes, interrupt when arithmetic overflow is detected. Programs can set individual bits in the FLAGS register to control the CPU's operation. Examples are the *Direction* and *Interrupt* flags.
2. **Status Flags***:* The status flags reflect the outcomes of arithmetic and logical operations performed by the CPU. They are:
    - **Carry** flag (CF) is set when the result of an ***unsigned*** **arithmetic** operation is too large to fit into the destination.
    - **Overflow** Flag (OF) is set when the result of a ***signed*** **arithmetic** operation is too large or too small to fit into the destination.

- **Sign** flag (SF) is set when the result of an arithmetic or logical operation generates a negative result.
- **Zero** flags (ZF) is set when the result of an arithmetic or logical operation generates a result of zero.
- **Auxiliary Carry** flag (AC) is set when an arithmetic operation causes a carry from bit 3 to bit 4 in an 8-bit operand.



- **Parity** flag (PF) is set if the **least-significant byte** in the result contains an even number of 1 bit. Otherwise, PF is clear. In general, it is used for error checking when there is a possibility that data might be altered or corrupted.


X
X
X
X
OF
DF
IF
TF
SF
ZF
X
AF
X
PF
X
CF
Overflow flag
Direction flag
Interrupt enable flag
Trap flag
Sign flag
Zero flag
Auxiliary carry flag
Parity flag
Carry flag
6 are status flag
3 are control flag
Bits marked X are undefined