8086 CPU ARCHITECTURE

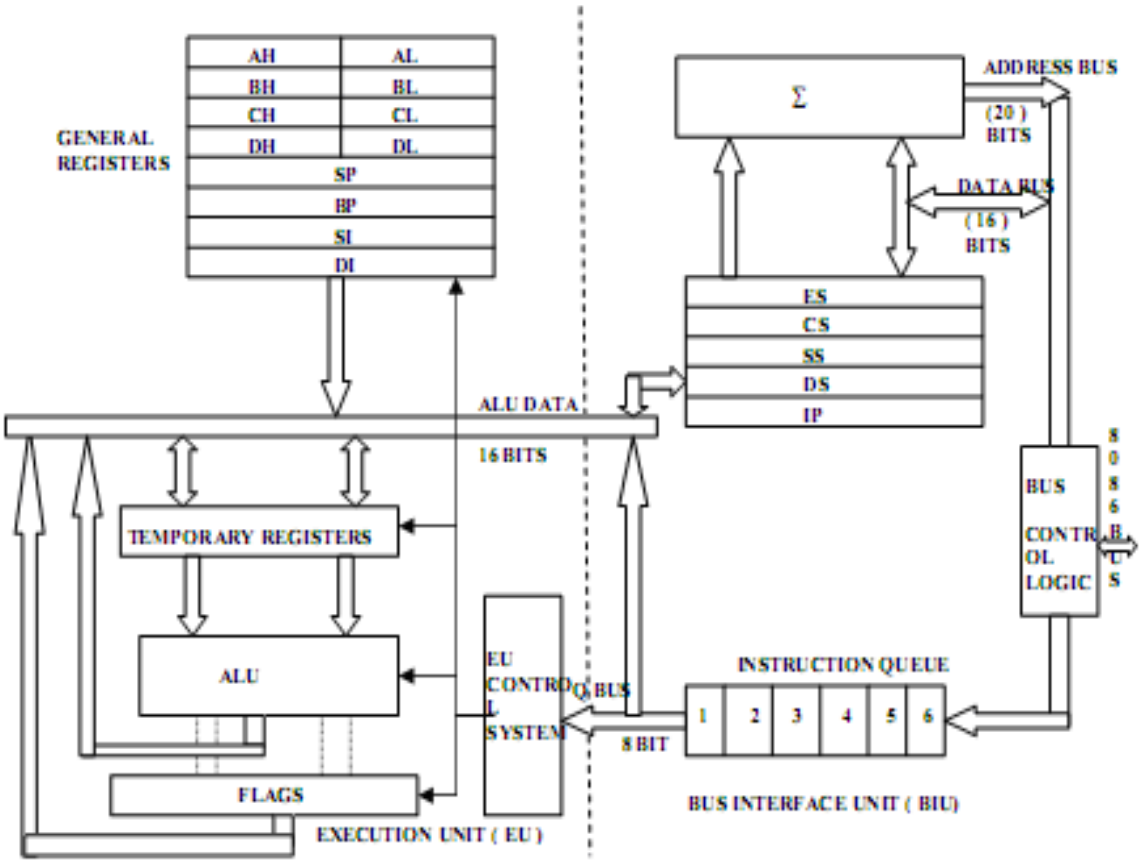
The microprocessors functions as the CPU in the stored program model of the digital computer. Its job is to generate all system timing signals and synchronize the transfer of data between memory, I/O, and itself. It accomplishes this task via the three-bus system architecture previously discussed.

The microprocessor also has a S/W function. It must recognize, decode, and execute program instructions fetched from the memory unit. This requires an Arithmetic-Logic Unit (ALU) within the CPU to perform arithmetic and logical (AND, OR, NOT, compare, etc) functions.

The 8086 CPU is organized as two separate processors (or units), called the Bus Interface Unit (BIU) and the Execution Unit (EU). The BIU provides H/W functions, including generation of the memory and I/O addresses for the transfer of data between the outside world -outside the CPU, that is- and the EU.

The EU receives program instruction codes and data from the BIU, executes these instructions, and store the results in the general registers. By passing the data back to the BIU, data can also be stored in a memory location or written to an output device. Note that the EU has no connection to the system buses. It receives and outputs all its data thru the BIU. The basic architecture of 8086 is shown below.

The only difference between an 8088 microprocessor and an 8086 microprocessor is the BIU. In the 8088, the BIU data bus path is 8 bits wide versus the 8086's 16-bit data bus. Another difference is that the 8088 instruction queue is four bytes long instead of six bytes in 8086.



Block Diagram of 8086

FETCH AND EXECUTE

Although the 8086/88 still functions as a stored program computer, organization of the CPU into a separate BIU and EU allows the fetch and execute cycles to overlap. To see this, consider what happens when the 8086 or 8088 is first started.

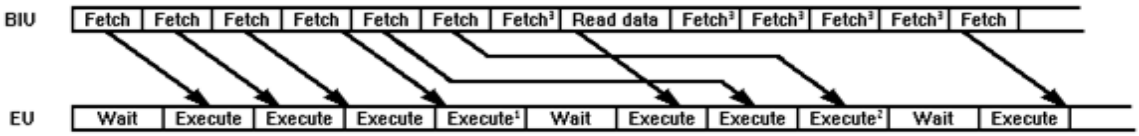
1. The BIU outputs the contents of the instruction pointer register (IP) onto the address bus, causing the selected byte or word to be read into the BIU.
2. Register IP is incremented by 1 to prepare for the next instruction fetch.
3. Once inside the BIU, the instruction is passed to the queue. This is a first-in, first-out (FIFO) storage register sometimes likened to a "pipeline
4. Assuming that the queue is initially empty, the EU immediately draws this instruction from the queue and begins execution.
5. 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.

Instruction Queue

To increase the execution speed, BIU fetches as many as six instruction bytes ahead to time from memory. The pre fetched instruction bytes are held for the EU in a first in first out group of registers called an instruction queue. When the EU is ready for its next instruction, it simply reads the instruction from this instruction queue. This is much faster than sending out an address to the system memory and to send back the next instruction byte. Fetching the next instruction while the current instruction executes is called pipelining.

The BIU is programmed to fetch a new instruction whenever the queue has room for one (with the 8088) or two (with the 8086) additional bytes. The advantage of this pipelined architecture is that the EU can execute instructions almost continually instead of having to wait for the BIU to fetch a new instruction

FETCH AND EXECUTE CYCLE



[1]This instruction requires a request for data not in the queue
[2]Jump instruction occurs
[3]These bytes are discarded

INSTRUCTION PIPELINE

There are three conditions that will cause the EU to enter a "wait" mode;

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 (nonsequential) 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.

SYSTEM BUS

A bus is collection of signal wires which connect between thee components of the computer systems – below the figure shows that the CPU Is connected to the memory as well as I/O through the system bus, but only one at a time – if the memory and I/O wants to use the bus at the same time, there is a conflict, as there is only one system bus. The system bus comprises of the address bus, data bus and the control bus:

1. **Data bus:** the set of lines used to transfer data is called the data bus. It is a bidirectional bus, as data has to be sent from the CPU to memory and I/O, and has to be received as well by the CPU. The width of the data bus determines the data transfer rate, size of the internal registers of the CPU and the processing capability of the CPU. In short, it is a reflection of the complexity of the processor. The 8086 has a data bus width of 16 bits.

2. **Address bus:** The address bus width determines the maximum size of the physical memory that the CPU can access. With an address bus width of 20 bits, the 8086 can address 2^{20} different locations. It can use a memory size of 2^{20} bytes or 1MB. When a particular memory location is to be accessed, the corresponding address is placed on the address bus by the CPU. I/O devices also have addresses. In both cases, it is the CPU which supplies the address, and as such, the address bus is unidirectional.

3. **Control bus:** the control bus is a set of control signals which needs to be activated for activities like writing/reading to/from memory/I/O, or special activities of the CPU like interrupts and DMA. Hence in the control bus, we have signals traveling in either direction. Some control lines may be bidirectional too.

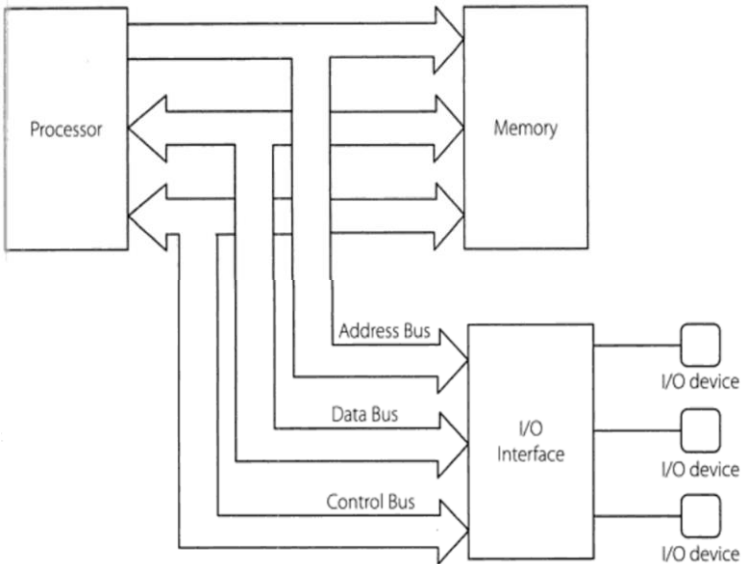


Fig. System bus

PROGRAMING MODEL

The programming model for a microprocessor shows the various internal registers that are accessible to the programmer. The Figure is a model for the 8086. In general, each register has a special function.

Registers of 8086

- General Purpose Registers
- Pointer Registers
- Index Registers
- Segment Registers
- Flags Registers

General Purpose Registers

There are four 16-bit general purpose registers:

1. AX Register: AX register is also known as accumulator register. It is 16-bit registers, andt it is divided into two 8-bit registers (AH and AL) to also perform 8-bit instructions. AX generally used for arithmetic and logical instructions.

2. BX Register: This register is mainly used as a base register. It is 16-bit registers, and it is divided into two 8-bit registers (BH and BL) to also perform 8-bit instructions. It holds the starting base location (offset address) of a memory region within a data segment.

3. CX Register: It is defined as a counter register. It is 16-bit registers, and it is divided into two 8-bit registers (CH and CL) to also perform 8-bit instructions. .It is primarily used in loop instruction to store loop counter .

4. DX Register: This is the data register. It is of 16 bits, and it is divided into two 8-bit registers DH and DL to also perform 8-bit instructions. It is used in multiplication an input/output port addressing.