short note on 4 bit,8 bit,16 bit and 32 bit ,64 bit microprocessor

ChatGPT

Microprocessors are essential components of modern computing devices, varying in their processing power and capabilities. Here's a brief overview of 4-bit, 8-bit, 16-bit, 32-bit, and 64-bit microprocessors:

**4-bit Microprocessor:** Early microprocessors with limited processing power, primarily used in simple electronic devices like calculators and early digital watches. They can process data in 4-bit chunks at a time, making them suitable for basic arithmetic operations.

**8-bit Microprocessor:** More capable than 4-bit counterparts, 8-bit microprocessors can handle larger data sizes and perform more complex tasks. They were popular in early personal computers and home gaming consoles, providing improved computational abilities.

**16-bit Microprocessor:** Offering enhanced performance and efficiency compared to 8-bit processors, 16-bit microprocessors became prevalent in the 1980s and early 1990s. They enabled more advanced computing tasks, including multitasking and graphical user interfaces, leading to the development of early PCs and gaming systems.

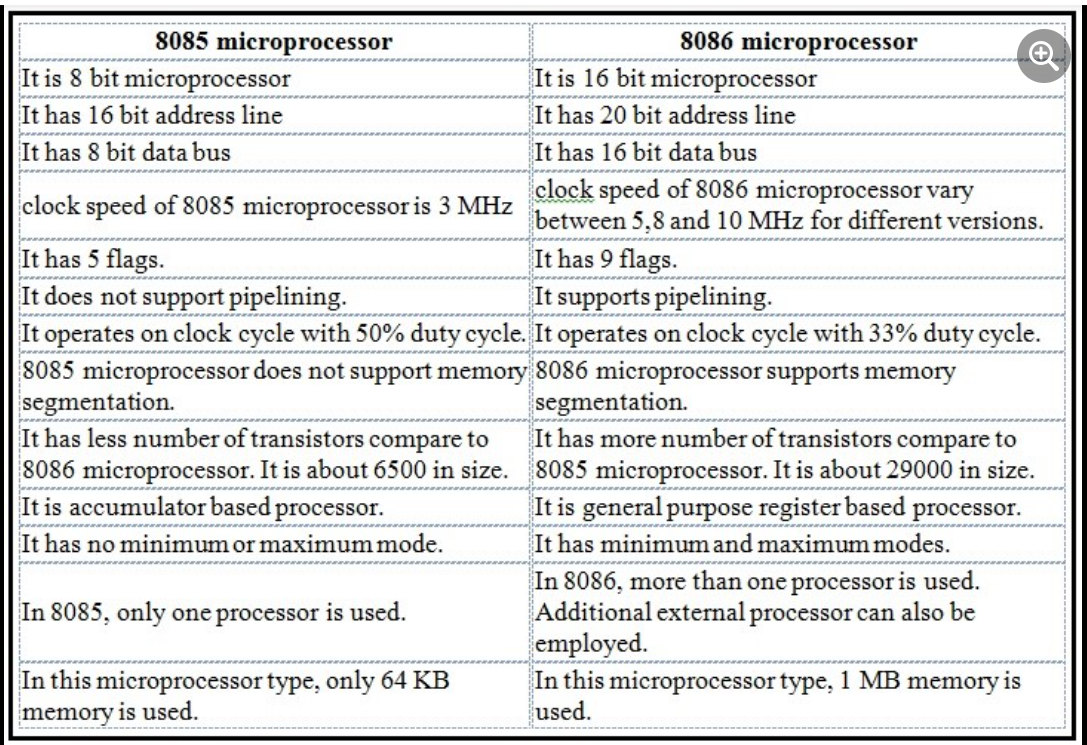
**32-bit Microprocessor:** With even greater processing power and expanded memory addressing capabilities, 32-bit microprocessors revolutionized computing in the 1990s and early 2000s. They facilitated faster data processing, improved system performance, and supported more sophisticated software applications, including operating systems and multimedia programs.

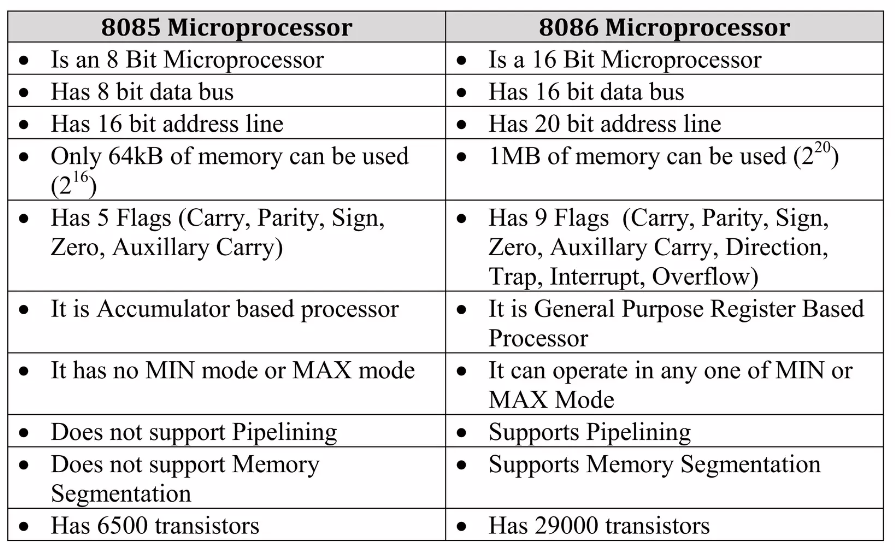
**64-bit Microprocessor:** The latest standard in microprocessor architecture, 64-bit processors offer significantly increased computational capabilities and memory addressing space. They enable more efficient handling of vast amounts of data, support for larger memory capacities, and enhanced performance for complex computing tasks such as scientific simulations, high-definition multimedia processing, and advanced gaming.

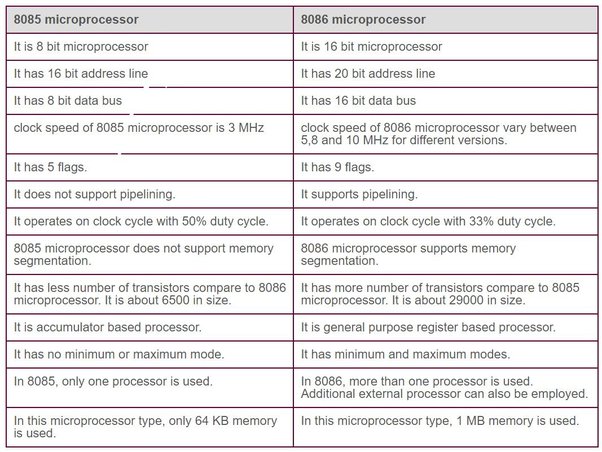
Each advancement in microprocessor architecture represents a significant milestone in the evolution of computing technology, enabling progressively more powerful and versatile devices to meet the demands of modern computing applications.

**difference between 8085 and 8086**

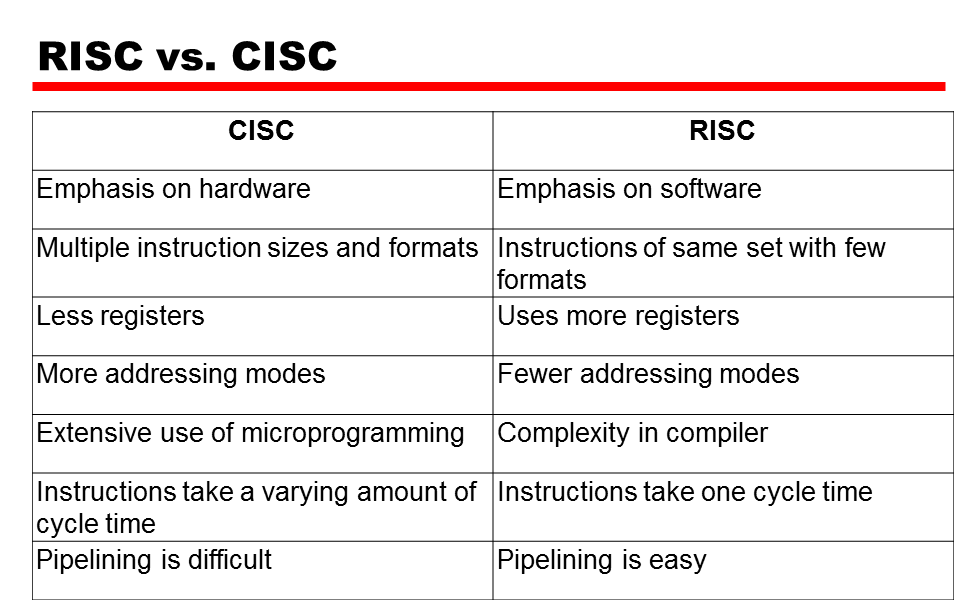
The 8085 can perform operations like addition, subtraction, increment, decrement, comparison, AND, shift, OR, and X-OR, but it cannot perform multiplication or division. The 8086 can perform all microprocessor operations, including multiplication and division.

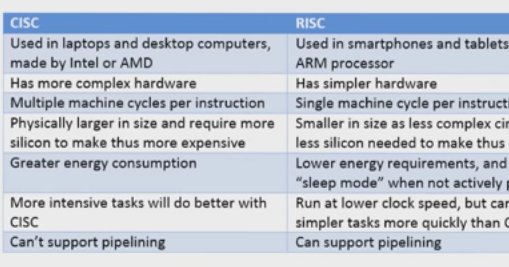


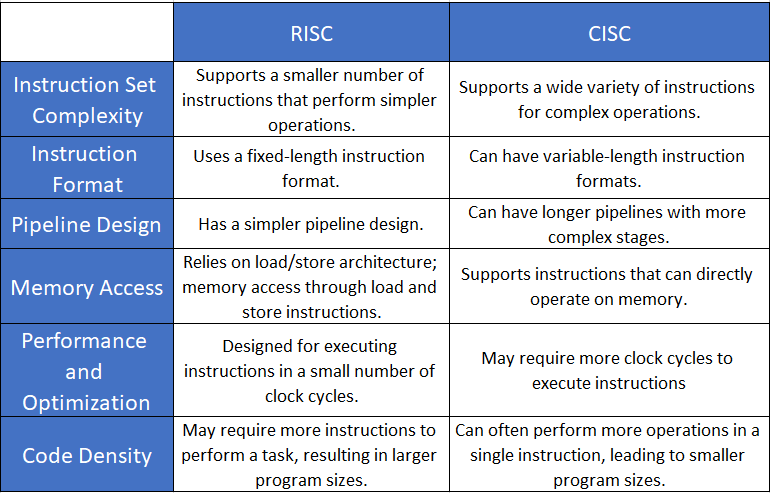




**difference between risk and cisk**







**Memory banking in 8086**

In any processor , there is memory. This memory is stored in locations. Each location can hold one byte( 8-bits ) of data. Memory banking in 8086 is , having divided the memory into two banks ( two parts ).

Why can one location store only one byte?  
Minimum byte operation needs 1 byte. So it is preferred to assign minimum memory to one location, so that memory is not wasted.  
If each location can hold 2 bytes( 16-bits ) of memory,

* When processor performs 16-bit operations it is good
* But when the processor performs 8-bit operation, then the remaining 8 bits are wasted.

Even if it is a 64-bit intel processor, each location can hold 8 bits of memory only.

Whenever 16-bit data is stored in locations, it is stored in two consecutive memory locations, lower byte is stored in lower address and higher byte is stored in higher address. This concept of lower byte lower location, higher byte in higher location is known as **little Endian rule.**

If each location can store 8-bits how can the processor perform 16-bit operation?

By memory banking.

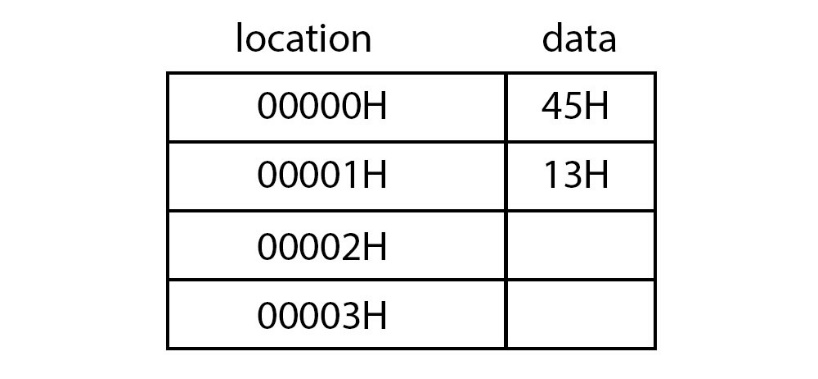
**Memory Banking:**

Here we shall consider an 8086 processor which can hold 1MB of memory due to its 20 address lines ( 220 = 1MB ) [ [Refer introduction tutorials](https://roboticelectronics.in/introduction-to-processors/) ]. So this 1MB memory is divided into two parts ( two banks ), where one bank consists of all even addresses and the other bank consists of odd addresses.

Memory banking in 8086 ppt


Purpose of memory banking:

As the 16-bit data is stored in 2 locations, lower byte in lower address and upper byte in next consecutive address location.



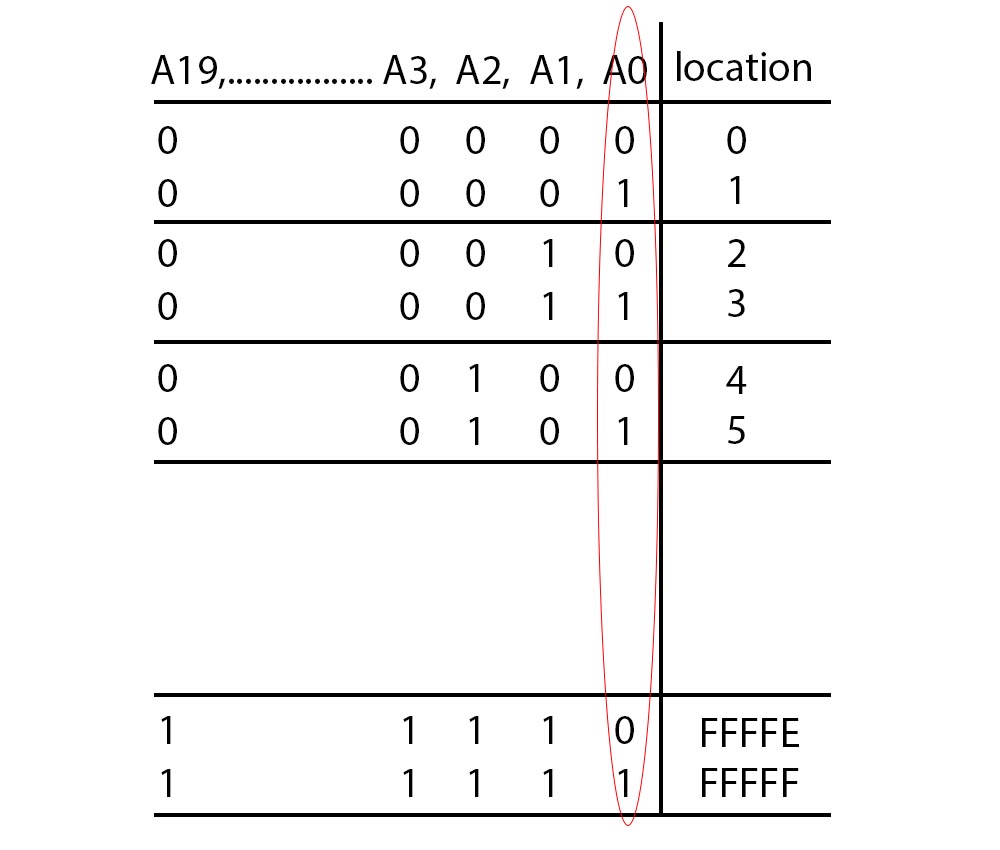
The data stored here is 1345H. This is representation of data in hexadecimal form. So whenever the processor needs to fetch this , it first gives the address to the address bus and fetches 45H and then it gives the next address, then it fetches 13H. So overall a 16-bit processor needs 2 cycles to perform a 16-bit operation, which is completely a wrong concept. This thing led to memory banking.

If the memory can be divided into two parts, with some algorithm, two 8-bit datas can be fetched in one cycle.

As Even bank starts from 00000H and goes up to FFFFEH , it stores the lower byte so also known as **lower bank.** Odd bank locations start from 00001H and go upto FFFFFH , this stores higher byte , thus also known as **Higher bank**in 8086.

How does banking work?

There are totally 20 address lines in 8086, that can access the locations. Each location is decided by these 20 address lines ( A0 , A1……. A19 ).



If observed, for every consecutive location starting from even location, there is only change in address ( A0 is only different ).

For Eg:

Locations 0H and 1H are consecutive and have the same address except A0.

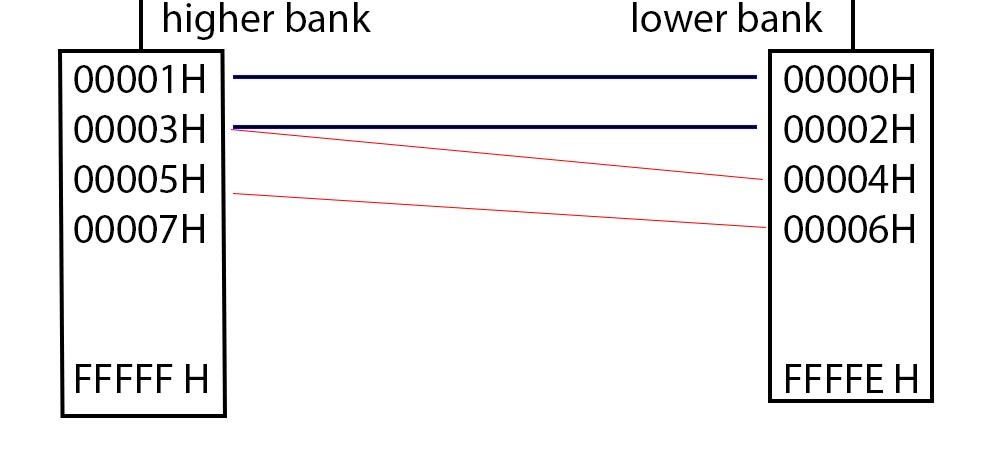
Locations 3H and 4H are consecutive but their addresses don’t have that similarity.

**Aligned data:**

Locations 0H and 1H is the example of aligned data. 8086 can fetch this type of data in one cycle.

**Misaligned data:**

Locations 3H and 4H are the example of misaligned data, 8086 needs 2 cycles to fetch this type of data.



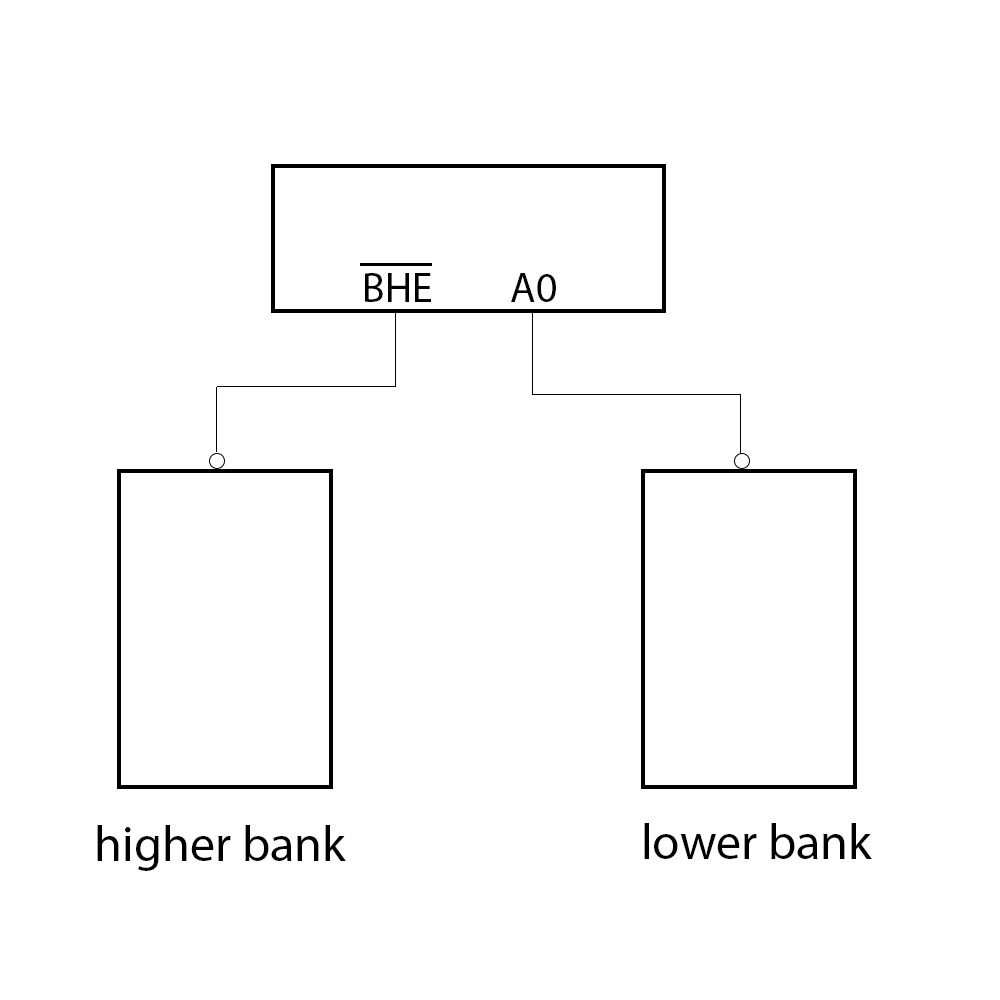
Now we would learn how we can access 16-bit aligned data in one cycle.

If A0 address line is discarded, 8086 sends a 19-bit address, there are two locations, one location in higher bank and other location in lower with the same address, in this way 8086 can fetch 16 bit data in one cycle. But if A0 is always discarded, 8086 can only perform 16-bit operations, and cannot perform 8-bit operations.

So we need that

* 8086 can be able to select lower bank only for 8-bit operations.
* 8086 can be able to select higher bank only for 8-bit operations.
* 8086 can be able to select both banks at single time for 16-bit operations.

Then BHE, comes into picture. It stands for bus high enable or bank high enable, and it is pronounced as BHE bar.



BHE bar is connected to the active low of the higher bank and A0 is connected to the active low of lower bank.

Whenever A0 = 0, lower bank is selected

And if A0 = 1, lower bank is not selected

A0 has nothing to do with higher bank

Whenever BHE  = 0, higher bank is selected

And if BHE = 1, higher bank is not selected

BHE has nothing to do with lower bank

|  |  |  |
| --- | --- | --- |
| BHE | A0 | Mode |
| 0 | 0 | Both the banks are selected and 8086 performs 16-bit operations. |
| 0 | 1 | Higher bank is selected and 8086 performs 8-bit operations with higher bank memory only. |
| 1 | 0 | Lower bank is selected and 8086 performs 8-bit operations with Lower bank memory only. |
| 1 | 1 | Both the banks are discarded and 8086 is in idle mode. |

BHE bar and A0 are decided by the processor according to instructions given.

* If it is an 8-bit instruction given an odd address then BHE=0 and A0 = 1.
* If it is a 16-bit instruction then BHE = 0 , A0 = 0

If it is a pentium processor, which is a 64-bit processor. Then there will be 8 memory banks.

