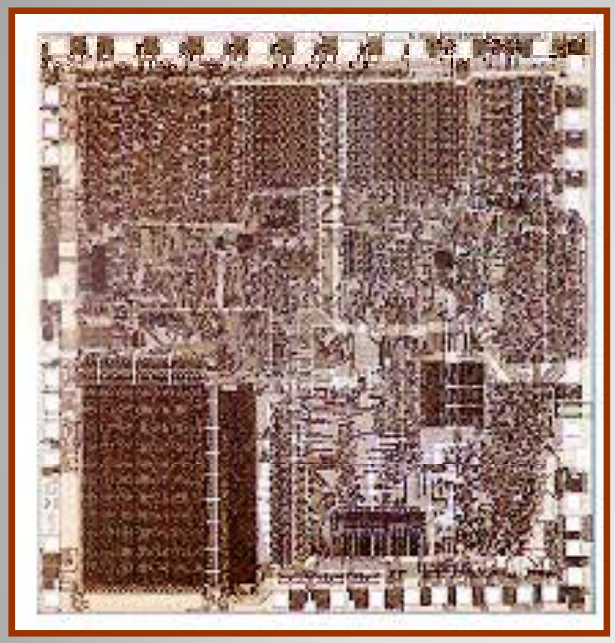


8086/8088 Microprocessor and its pin configuration

REFERENCES:

1. MICROPROCESSORS AND INTERFACING PROGRAMMING AND HARDWARE, SECOND EDITION, D.V. HALL – CHAPTER 7, CHAPTER 11

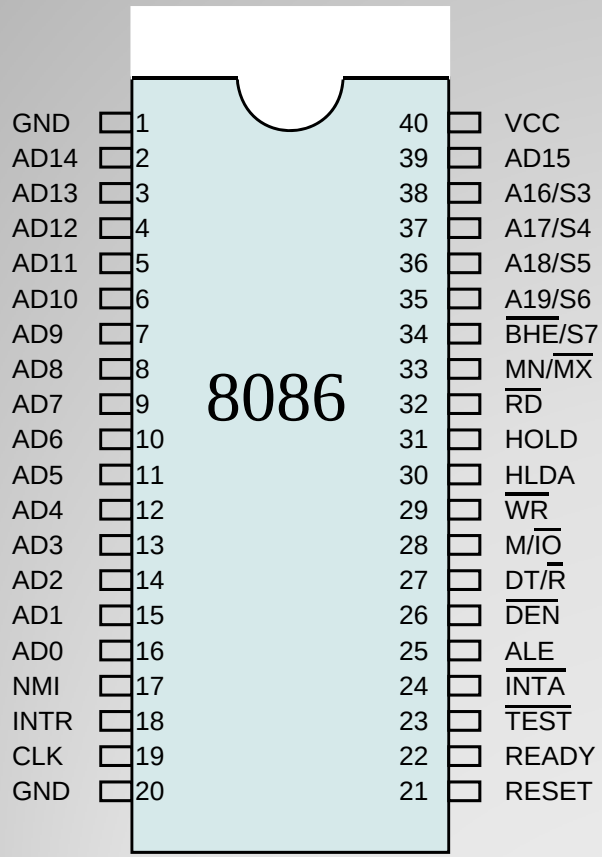
Topics

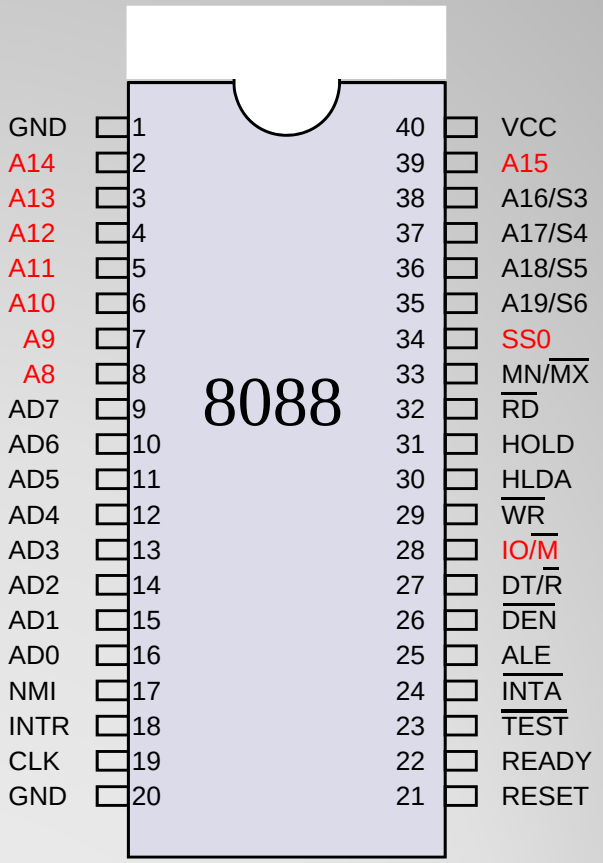
- Basic Features
- Pin Diagram
- Minimum and Maximum modes
- Description of the pins

Basic Features

- 8086 announced in 1978; 8086 is a 16 bit microprocessor with a 16 bit data bus
- 8088 announced in 1979; 8088 is a 16 bit microprocessor with an 8 bit data bus
- Both manufactured using High-performance Metal Oxide Semiconductor (HMOS) technology
- Both contain about 29000 transistors
- Both are packaged in 40 pin dual-in-line package (DIP)

8086/8088 Pin Diagrams

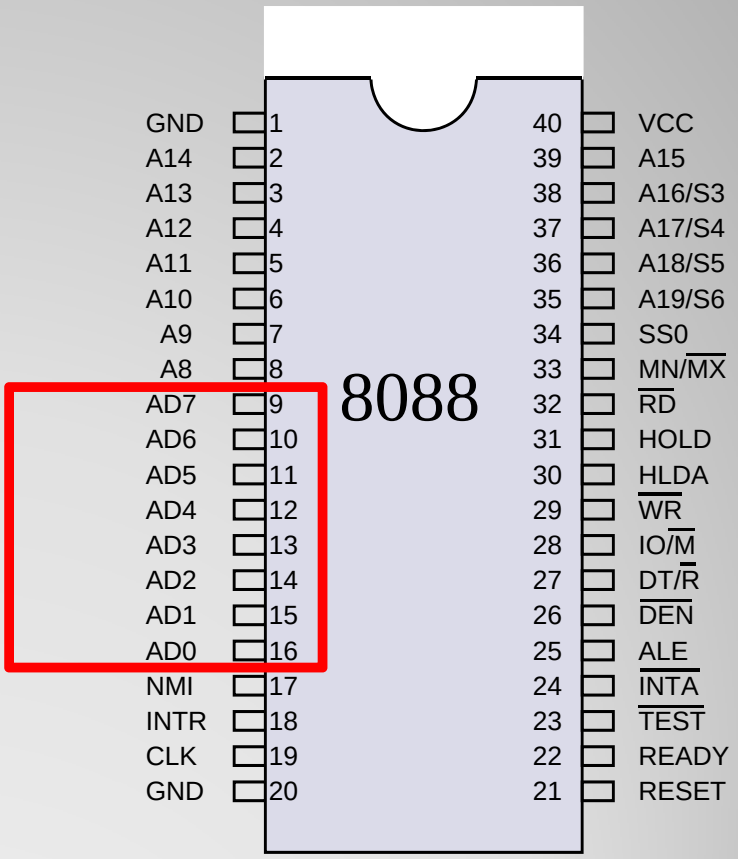




BHE has no meaning on the 8088 and has been eliminated

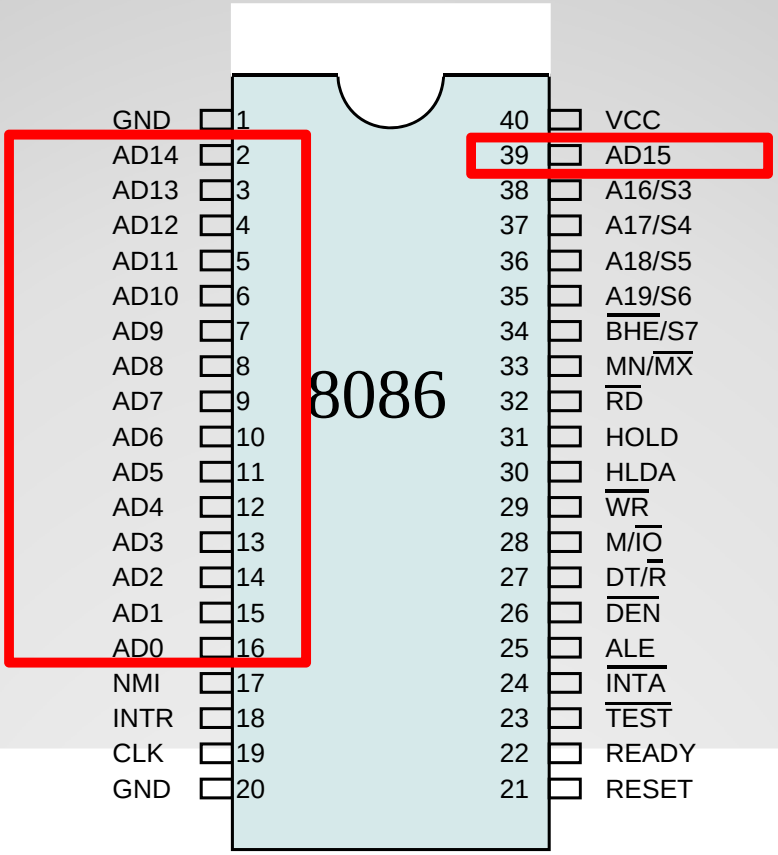
Multiplex of Data and Address Lines in 8088

- Address lines A0-A7 and Data lines D0-D7 are multiplexed in 8088. These lines are labelled as AD0-AD7.
  - By multiplexed we mean that the same physical pin carries an address bit at one time and the data bit another time



Multiplex of Data and Address Lines in 8086

- Address lines A0-A15 and Data lines D0-D15 are multiplexed in 8086. These lines are labeled as AD0-AD15.

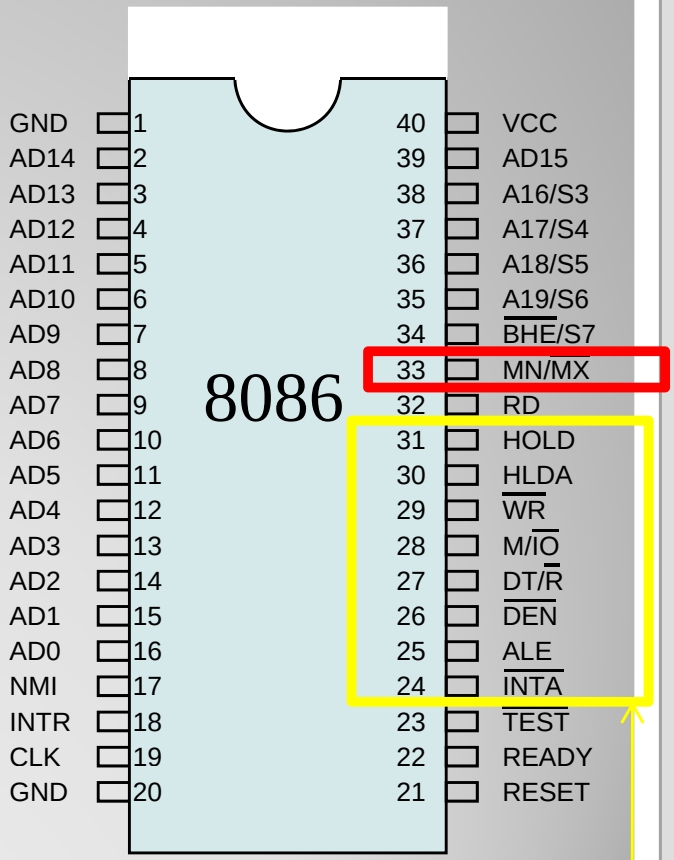


Minimum-mode and Maximum-mode Systems

- 8088 and 8086 microprocessors can be configured to work in either of the two modes: the minimum mode and the maximum mode

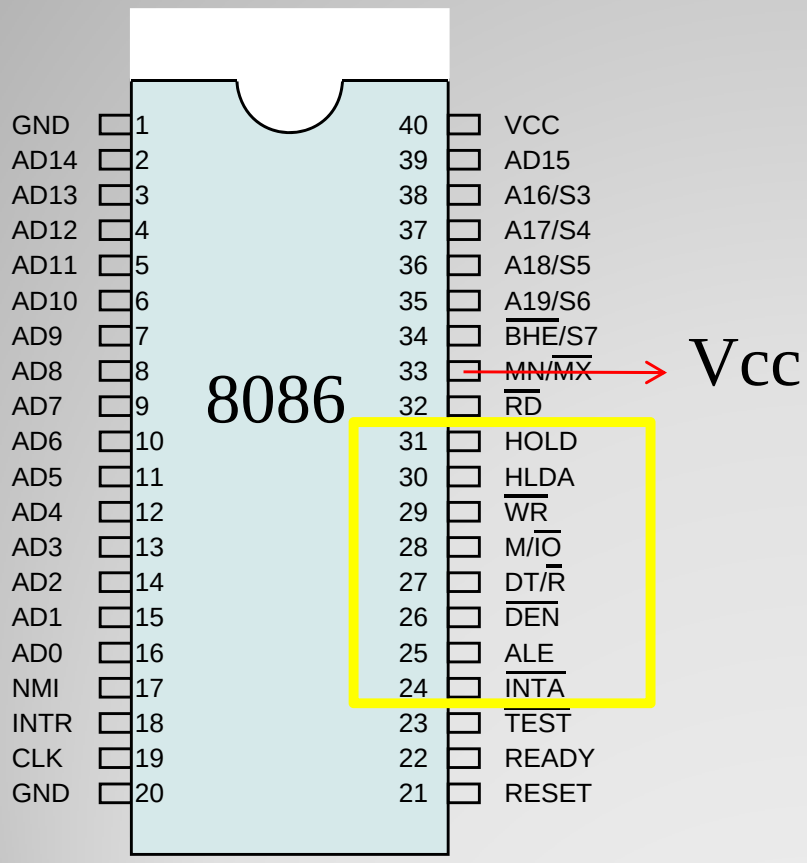
✓ Minimum mode:
- Pull MN/MX to logic 1
- Typically smaller systems and contains a single microprocessor
- Cheaper since all control signals for memory and I/O are generated by the microprocessor.

✓ Maximum mode
- Pull MN/**MX** logic 0
- Larger systems with more than one processor (*designed to be used when a coprocessor (8087) exists in the system)*

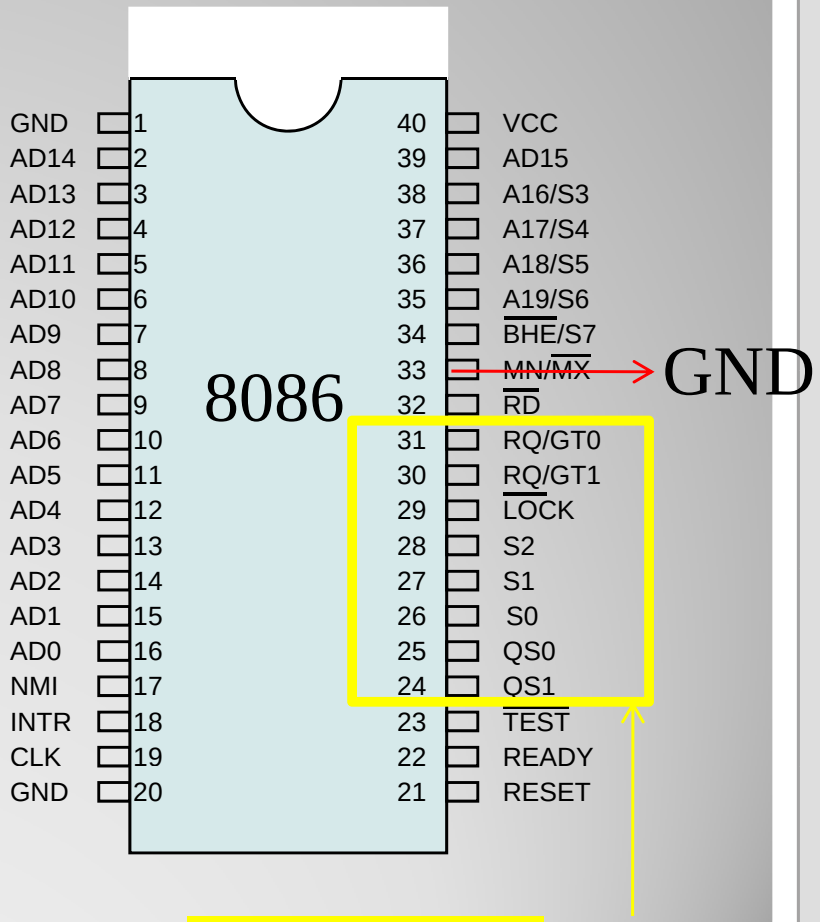


Lost Signals in Max Mode

Minimum-mode and Maximum-mode Signals



Min Mode



Max Mode

RESET Operation results

CPU component	Contents
Flags	Cleared
Instruction Pointer	0000H
CS	FFFFH
DS, SS and ES	0000H
Queue	Empty

S0, S1 and S2 Signals

S2	S1	S0	Characteristics
0	0	0	Interrupt acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Inactive

A17/S4, A16/S3 Address/Status

A17/S4	A16/S3	Function
0	0	Extra segment access
0	1	Stack segment access
1	0	Code segment access
1	1	Data segment access

A19/S6, A18/S5 Address/Status

A18/S5: The status of the interrupt enable flag bit is updated at the beginning of each cycle. The status of the flag is indicated through this pin.

A19/S6: When Low, it indicates that 8086 is in control of the bus. During a "Hold acknowledge" clock period, the 8086 tri-states the S6 pin and thus allows another bus master to take control of the status bus.

QS0 and QS1 Signals

QS1	QS0	Characteristics
0	0	No operation
0	1	First byte of opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

- $\overline{\text{INTA}}$
  - Interrupt Acknowledge generated by the microprocessor in response to INTR. Causes the interrupt vector to be put onto the data bus.
- $\overline{\text{BHE}}$
  - Bus High Enable. Enables the most significant data bus bits (D15-D8) during a read or write operation.
- VCC/GND
  - Power supply (5V) and GND (0V)
- MN/$\overline{\text{MX}}$
  - Select minimum (5V) or maximum mode (0V) of operation.

- $\overline{RQ}/\overline{GT1}$ and $\overline{RQ}/\overline{GT0}$
  - Request/grant pins request/grant direct memory accesses (DMA) during maximum mode operation.
- $\overline{LOCK}$
  - Lock output is used to lock peripherals off the system. Activated by using the LOCK: prefix on any instruction.

INTR (input)
Hardware Interrupt Request Pin

- INTR is used to request a hardware interrupt.
- It is recognized by the processor only when IF = 1, otherwise it is ignored (STI instruction sets this flag bit).
- The request on this line can be disabled (or masked) by making IF = 0 (use instruction CLI)
- If INTR becomes **high** and IF = 1, the 8086 enters an interrupt acknowledge cycle (INTA becomes active) after the current instruction has completed execution.

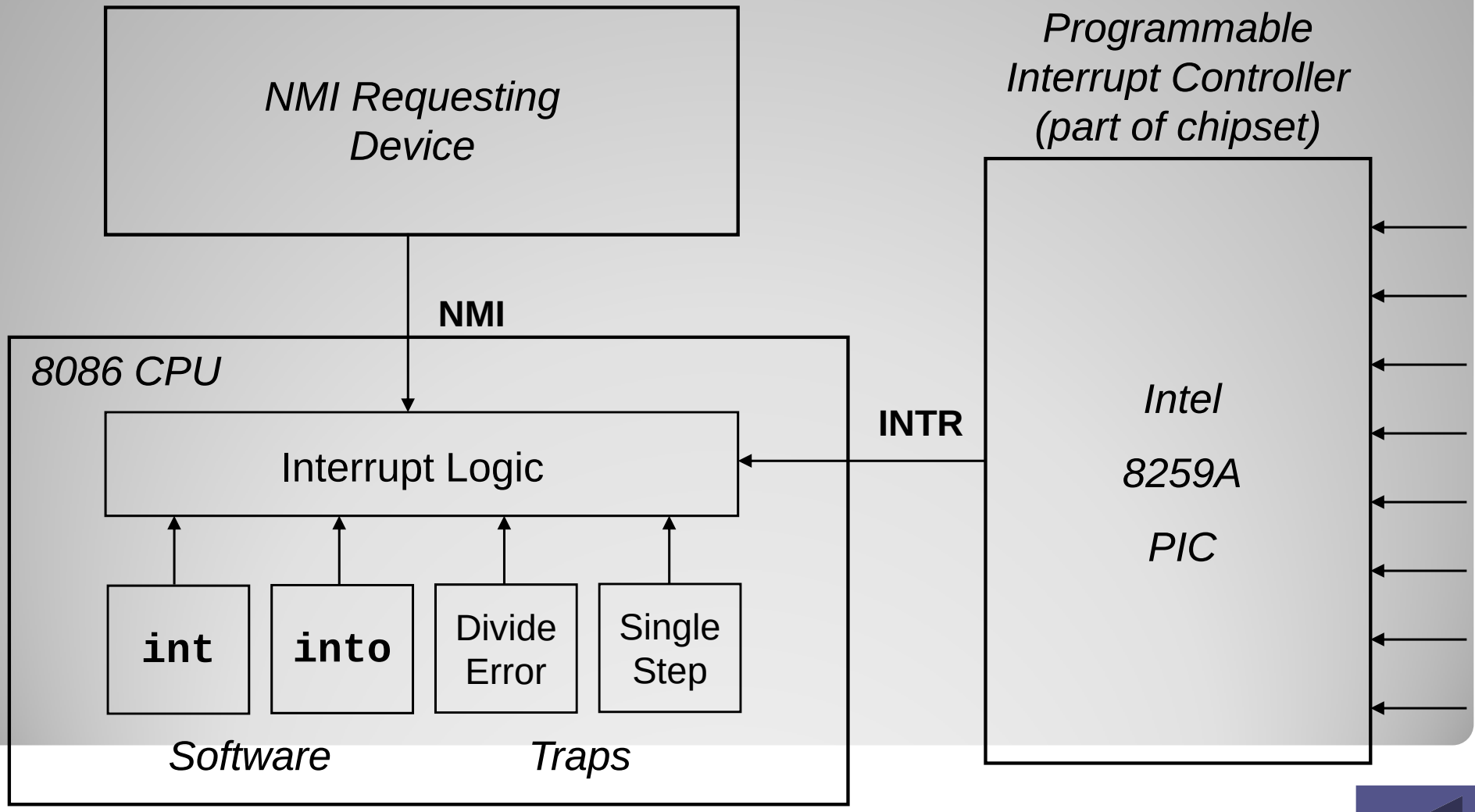
NMI (input) Non-Maskable Interrupt line

- The Non Maskable Interrupt input is similar to INTR except that the NMI interrupt does not check to see if the IF flag bit is at logic 1.
- This interrupt cannot be masked (or disabled) and no acknowledgment is required.
- It should be reserved for “catastrophic” events such as power failure or memory errors.

8086 External Interrupt Connections

NMI - *Non-Maskable Interrupt*

INTR - *Interrupt Request*



TEST (input)

- The $\overline{\text{TEST}}$ pin is an input that is tested by the WAIT instruction.
- If $\overline{\text{TEST}}$ is at logic 0, the WAIT instruction functions as a NOP.
- If $\overline{\text{TEST}}$ is at logic 1, then the WAIT instruction causes the 8086 to idle, until $\overline{\text{TEST}}$ input becomes a logic 0.
- This pin is normally driven by the 8087 co-processor (numeric coprocessor).

Ready (input)

- This input is used to insert wait states into processor Bus Cycle.
- If the READY pin is placed at a logic 0 level, the microprocessor enters into wait states and remains idle.
- If the READY pin is placed at a logic 1 level, it has no effect on the operation of the processor.
- It is sampled at the end of the T2 clock pulse
- Usually driven by a slow memory device

HOLD (input)

- The HOLD input is used by DMA controller to request a Direct Memory Access (DMA) operation.
- If the HOLD signal is at logic 1, the microprocessor places its address, data and control bus at the high impedance state.
- If the HOLD pin is at logic 0, the microprocessor works normally.

HLDA (output)

Hold Acknowledge Output

- Hold acknowledge is made high to indicate to the DMA controller that the processor has entered hold state and it can take control over the system bus for DMA operation.

8086 Memory Addressing

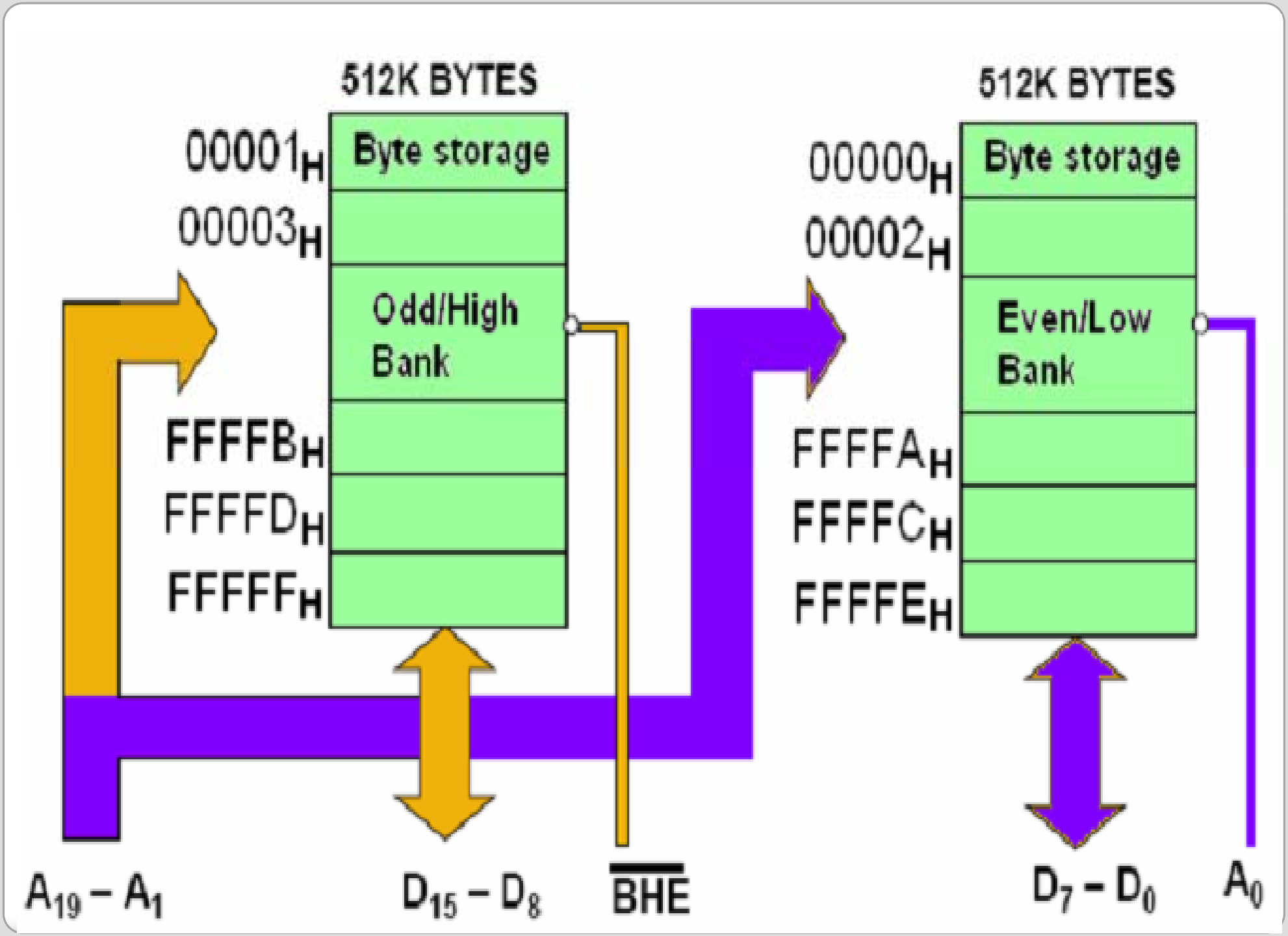
Data can be accessed from the memory in four different ways:

- 8 - bit data from Lower (Even) address Bank.
- 8 - bit data from Higher (Odd) address Bank.
- 16 - bit data starting from Even Address.
- 16 - bit data starting from Odd Address.

1. Accessing 8-bit data from Lower (Even) address bank :

- The two bank memory module of 8086 based storage system requires one bus-cycle to read/write a data-byte.

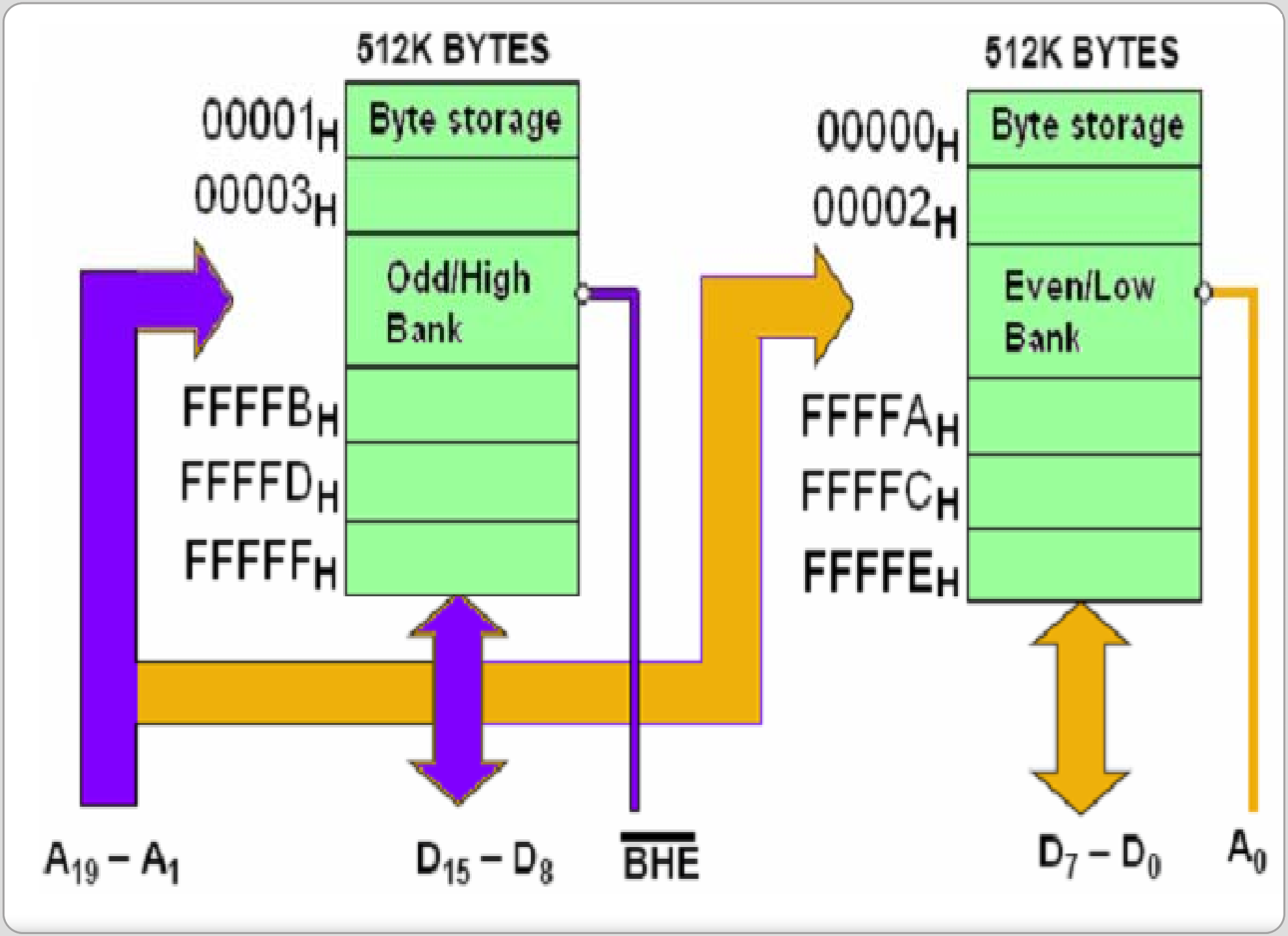
- To access a Byte of data in Low-bank, valid address is provided via address pins A1 to A19 together with A0=’0’ and $\overline{BHE}$=’1’.


512K BYTES
512K BYTES
00001H
00003H
Byte storage
Odd/High Bank
FFFFBH
FFFFDH
FFFFFH
00000H
00002H
Byte storage
Even/Low Bank
FFFFAH
FFFFCH
FFFFEH
A19 – A1
D15 – D8
BHE
D7 – D0
A0

2. Accessing 8-bit data from Higher (Odd) address bank

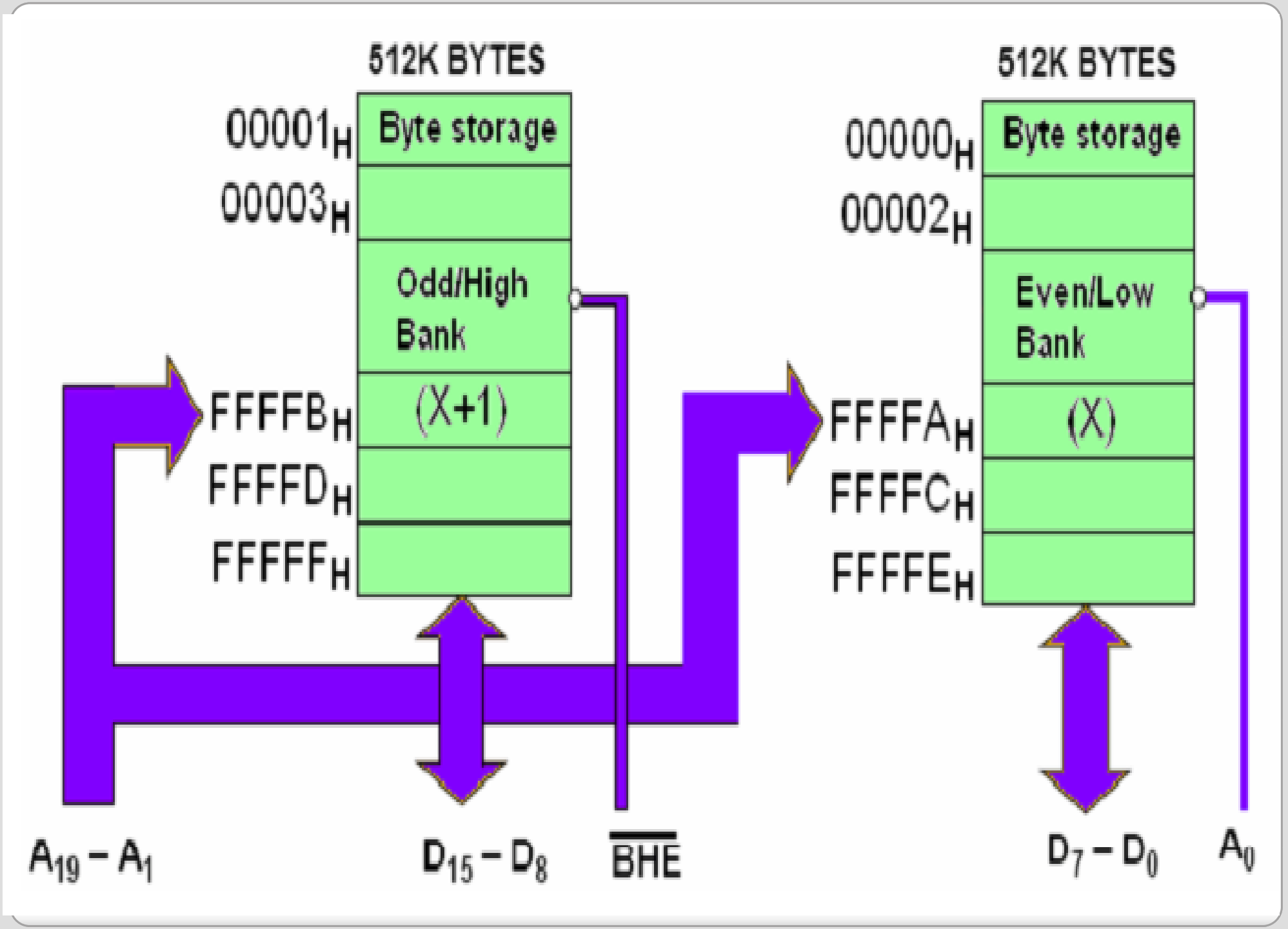
- Similarly to access a Byte of data in High-bank, valid address in pins A1 to A19, A0=’1’ and $\overline{\text{BHE}}$=’0’ are required to access the data through D8 to D15 of the data-bus.

- These signals disable the Low bank and enable the High bank to transfer (in/out) data through D8 to D15 of the data-bus.


512K BYTES
512K BYTES
00001H
Byte storage
00003H
Odd/High Bank
FFFFBH
FFFFDH
FFFFFH
00000H
Byte storage
00002H
Even/Low Bank
FFFFAH
FFFFCH
FFFFEH
A19 – A1
D15 – D8
BHE
D7 – D0
A0

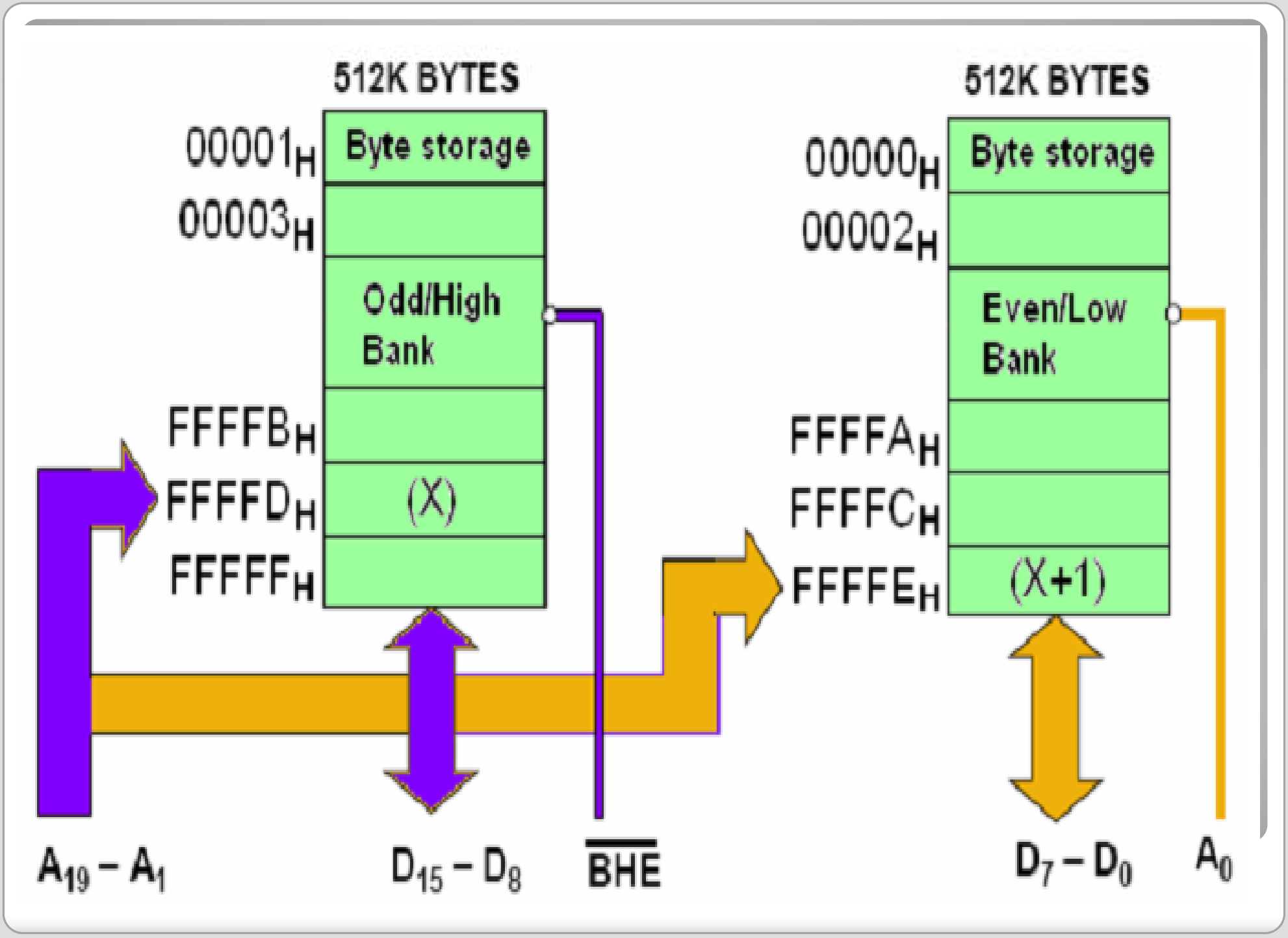
3. Accessing 16 - bit data starting from Even Address.

- For even-addressed (aligned) words, only one bus-cycle is needed to access the word, as both low and high banks are activated at the same time using A0=’0’ and $\overline{\text{BHE}}$=’0’
- Note that during this bus-cycle, all 16-bit data is transferred via D0 to D15 of the data bus.

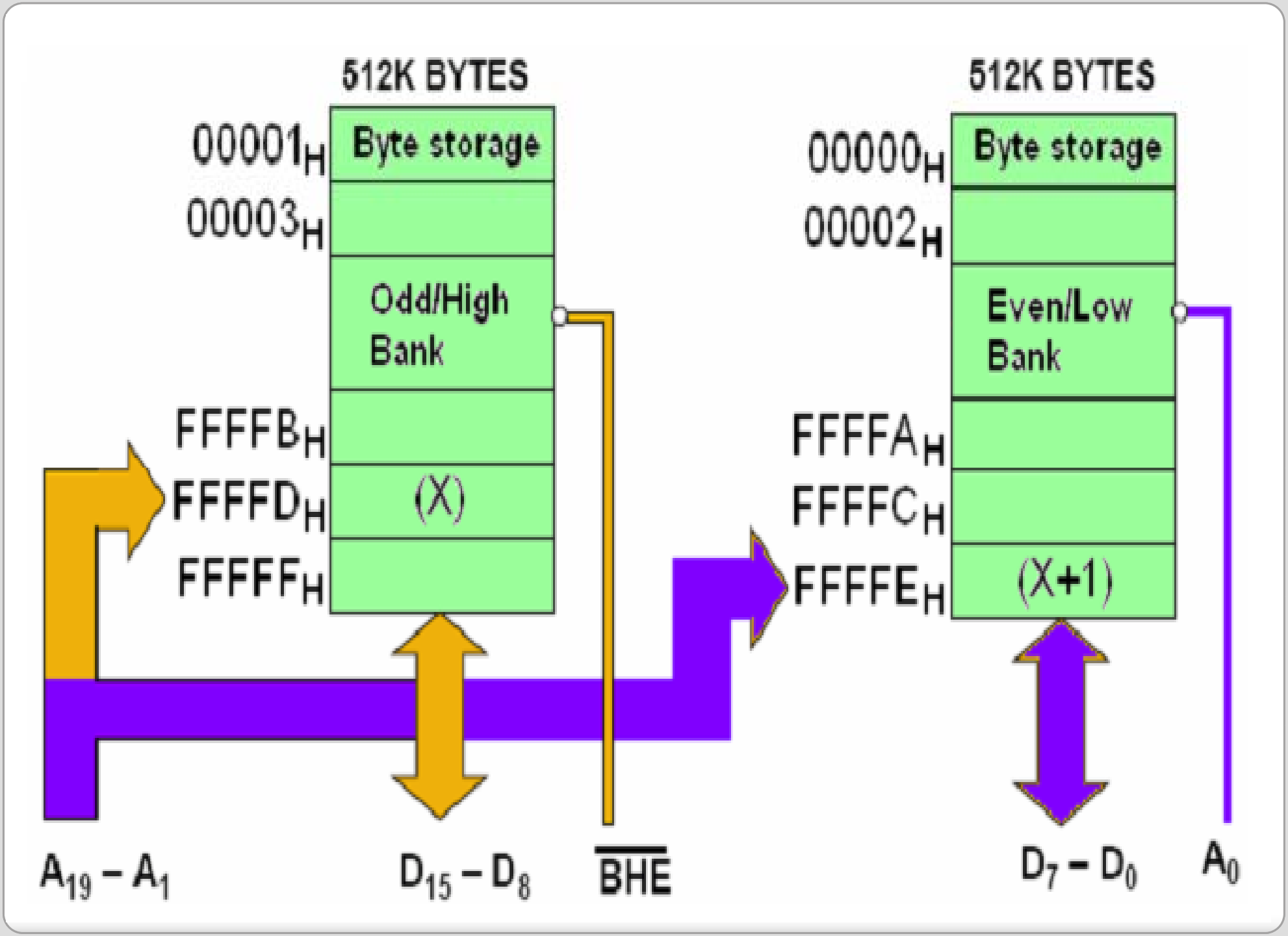

512K BYTES
00001H
Byte storage
00003H
Odd/High Bank
FFFFBH
(X+1)
FFFFDH
FFFFFH
512K BYTES
00000H
Byte storage
00002H
Even/Low Bank
FFFFAH
(X)
FFFFCH
FFFFEH
A19 – A1
D15 – D8
BHE
D7 – D0
A0

4. Accessing 16 - bit data starting from Odd Address.

- For odd-addressed (unaligned) words (with odd P.A of the LSB), two bus-cycles are required to access the Word-data.
- During the 1st bus-cycle, odd addressed LSB of the word is accessed from the High-memory-bank via D8 to D15 of data bus


512K BYTES
512K BYTES
00001H
Byte storage
00003H
Odd/High Bank
FFFFBH
FFFFDH
(X)
FFFFFH
00000H
Byte storage
00002H
Even/Low Bank
FFFFAH
FFFFCH
FFFFEH
(X+1)
A19 – A1
D15 – D8
BHE
D7 – D0
A0

- During 2nd bus-cycle, P.A. is auto-incremented to access the even address MSB of the word from the Low bank via D0 to D7.
- Note that A0 and $\overline{BHE}$ signals are reset (violet) accordingly to enable the required memory bank.


512K BYTES
00001H
Byte storage
00003H
Odd/High Bank
FFFFBH
FFFFDH
(X)
FFFFFH
512K BYTES
00000H
Byte storage
00002H
Even/Low Bank
FFFFAH
FFFFCH
FFFFEH
(X+1)
A19 – A1
D15 – D8
BHE
D7 – D0
A0

Direct Memory Access (DMA)

Direct memory access (DMA) is a process in which an external device takes over the control of system bus from the CPU.

DMA is for **high-speed data transfer** from/to mass storage peripherals, e.g. hard-disk drive,CD-ROM, and sometimes video controllers.

The basic idea of **DMA** is to transfer *blocks of data* directly between memory and peripherals. The data don't go through the microprocessor but the system data bus is occupied.

"Normal" transfer of one data byte takes up to 29 clock cycles. The DMA transfer requires only 5 clock cycles.

Nowadays, DMA can transfer data as fast as 60 MB per second or more. The transfer rate is limited by the speed of memory and peripheral devices.

Basic process of DMA

For 8088/8086 in minimum mode:

The HOLD and HLDA pins are used to receive and acknowledge the hold request respectively.

Normally the CPU has full control of the system bus. In a DMA operation, the peripheral takes over bus control temporarily.

Sequence of events of a typical DMA process

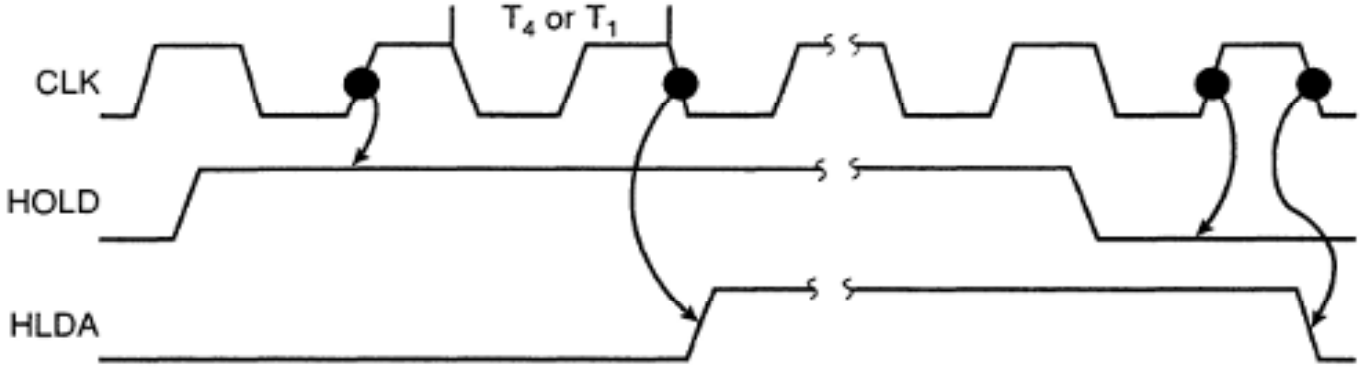
1) DMA controller asserts the request on the HOLD pin

2) 8086 completes its current bus cycle and enters into a HOLD state

3) 8086 grants the right of bus control by asserting a grant signal via the HOLDA pin. 8086 pins (Address, Data, C-trol ⇒ 3-rd state)

4) DMA operation starts

5) Upon completion of the DMA operation, the DMA controller asserts low the request/grant pin again to relinquish bus control.



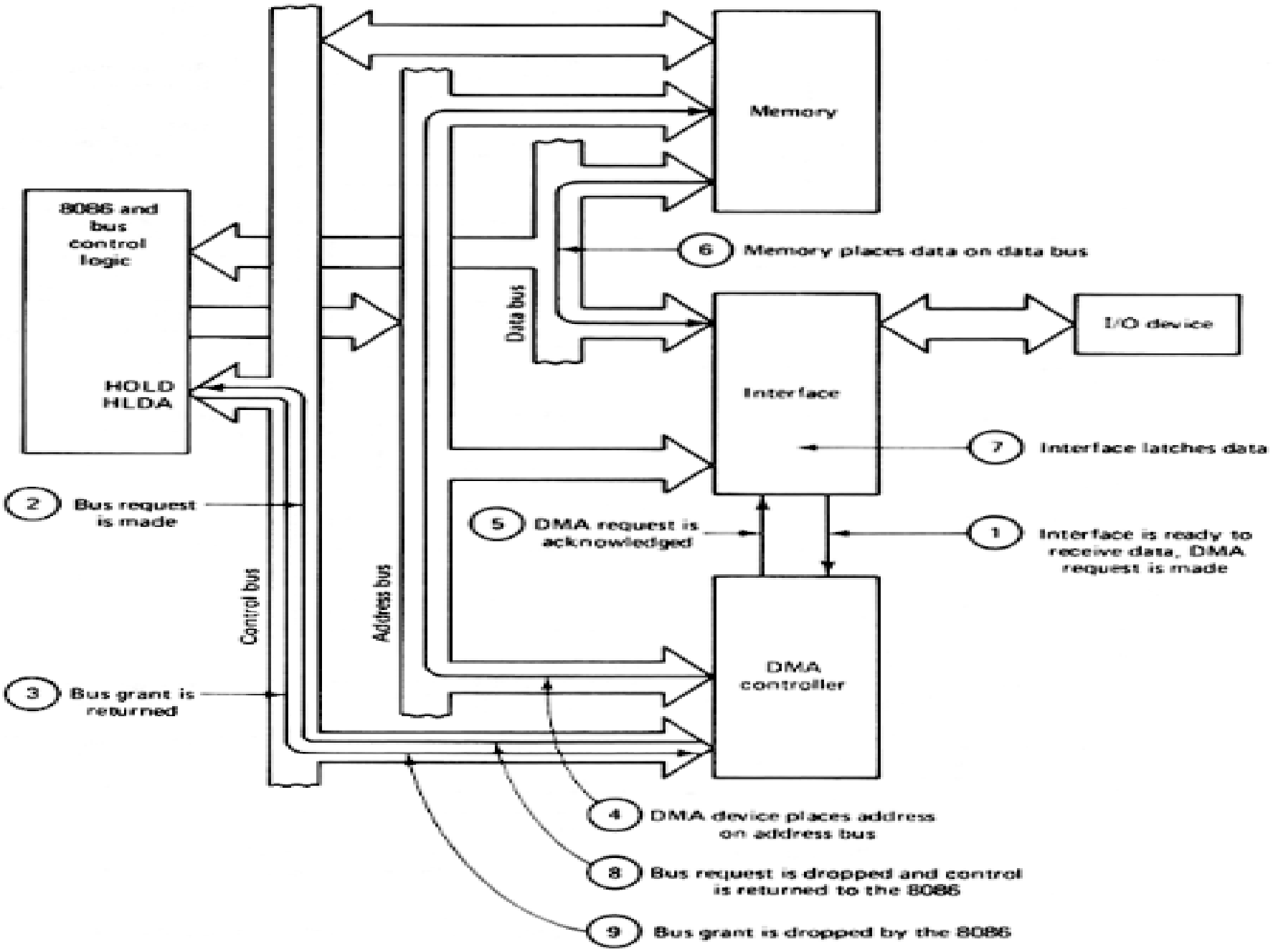
Basic process of DMA

For 8088/8086 in maximum mode:

The RQ/GT1 and RQ/GT0 pins are used to issue DMA request and receive acknowledge signals.

Sequence of events of a typical DMA process

1) DMA controller asserts one of the request pins, e.g. RQ/GT1 or RQ/GT0 (RQ/GT0 has higher priority)
2) 8086 completes its current bus cycle and enters into a HOLD state
3) 8086 grants the right of bus control by asserting a grant signal via the same pin as the request signal.
4) DMA operation starts
5) Upon completion of the DMA operation, the DMA controller asserts the request/grant pin again to relinquish bus control.


Memory
8086 and bus control logic
HOLD
HLDA
6 Memory places data on data bus
Data bus
I/O device
Interface
7 Interface latches data
2 Bus request is made
5 DMA request is acknowledged
1 Interface is ready to receive data, DMA request is made
Control bus
Address bus
DMA controller
3 Bus grant is returned
4 DMA device places address on address bus
8 Bus request is dropped and control is returned to the 8086
9 Bus grant is dropped by the 8086

General organization of the DMA controller