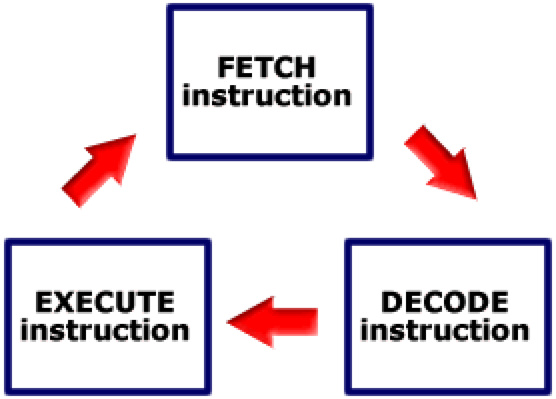
ADVANCED MICROPROCESSOR 8086

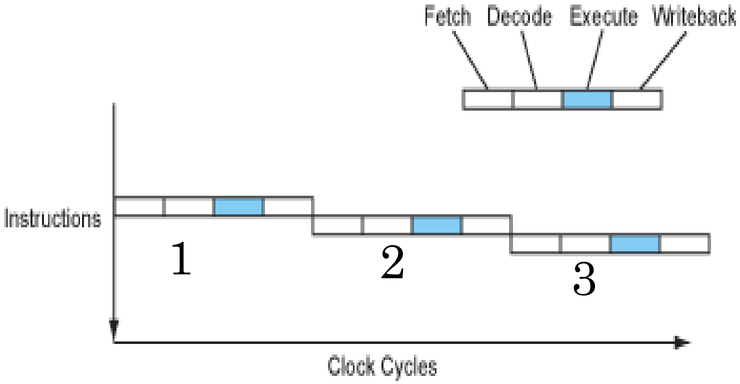
Architecture of 8086

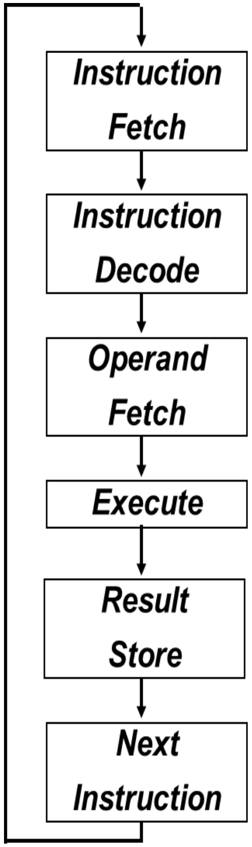
By
Dr. Uday Maji

PIPELINE PROCESSING

- General execution sequence of processor:

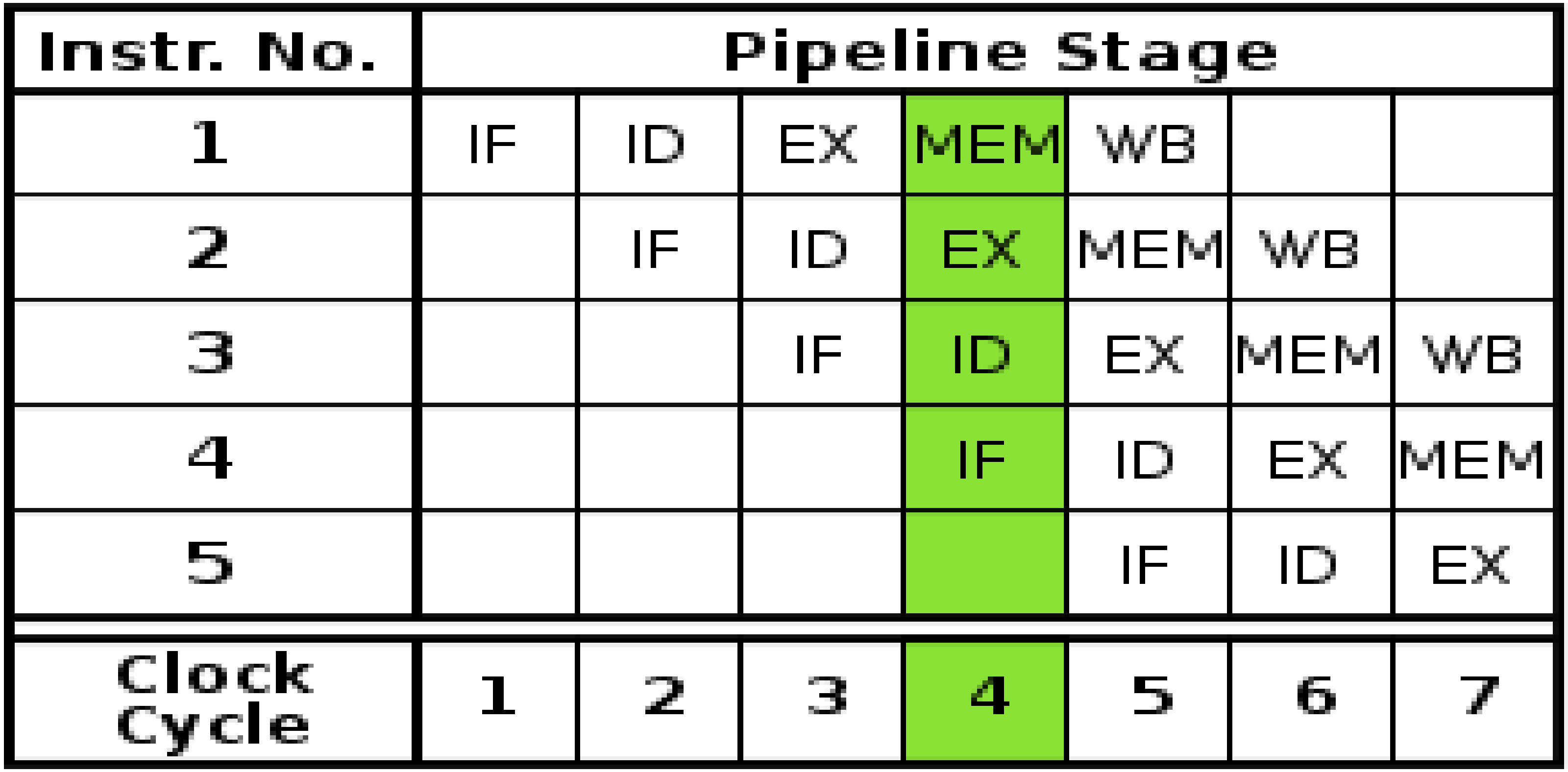






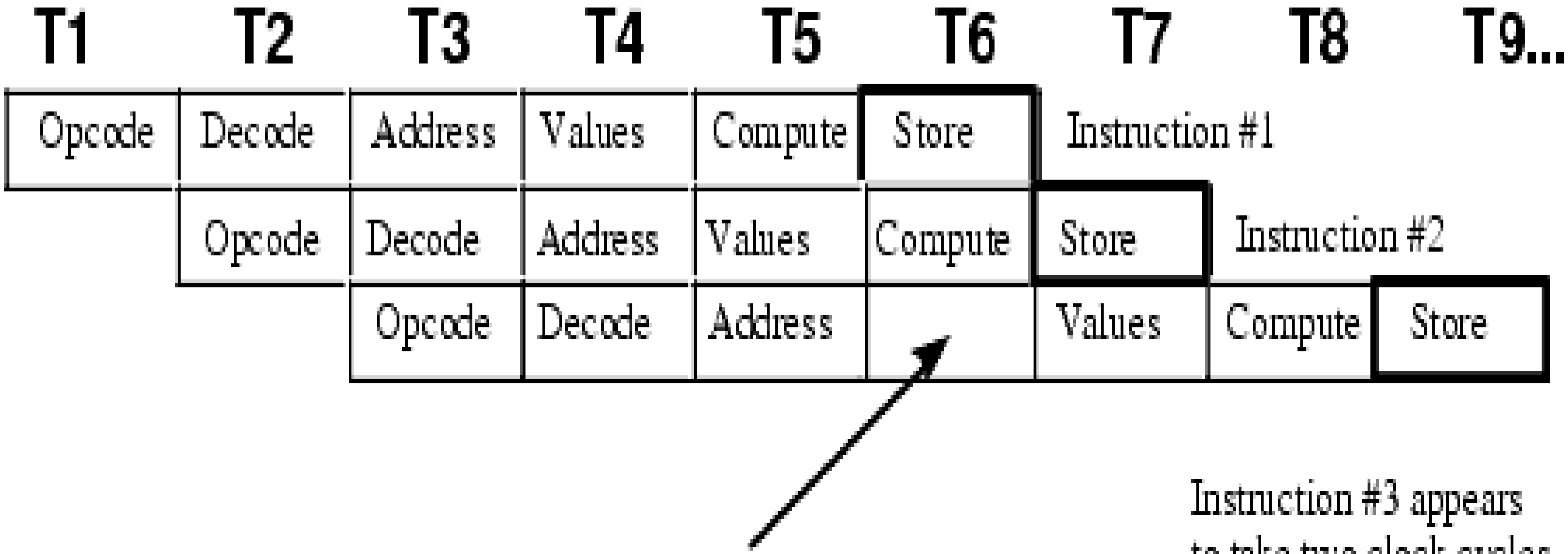
Instruction Fetch	Obtain instruction from program storage
Instruction Decode	Determine required actions and instruction size
Operand Fetch	Locate and obtain operand data
Execute	Compute result value or status
Result Store	Deposit results in storage for later use
Next Instruction	Determine successor instruction; can generally be combined w/ Decode

Pipeline processing



Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

Pipeline processing



T1	T2	T3	T4	T5	T6	T7	T8	T9...
Opcode	Decode	Address	Values	Compute	Store	Instruction #1		
	Opcode	Decode	Address	Values	Compute	Store	Instruction #2	
		Opcode	Decode	Address		Values	Compute	Store

Pipeline stall occurs here because instruction #1 is attempting to store a value to memory at the same time instruction #2 is attempting to read a value from memory.

Instruction #3 appears to take two clock cycles to execute because of the pipeline stall.

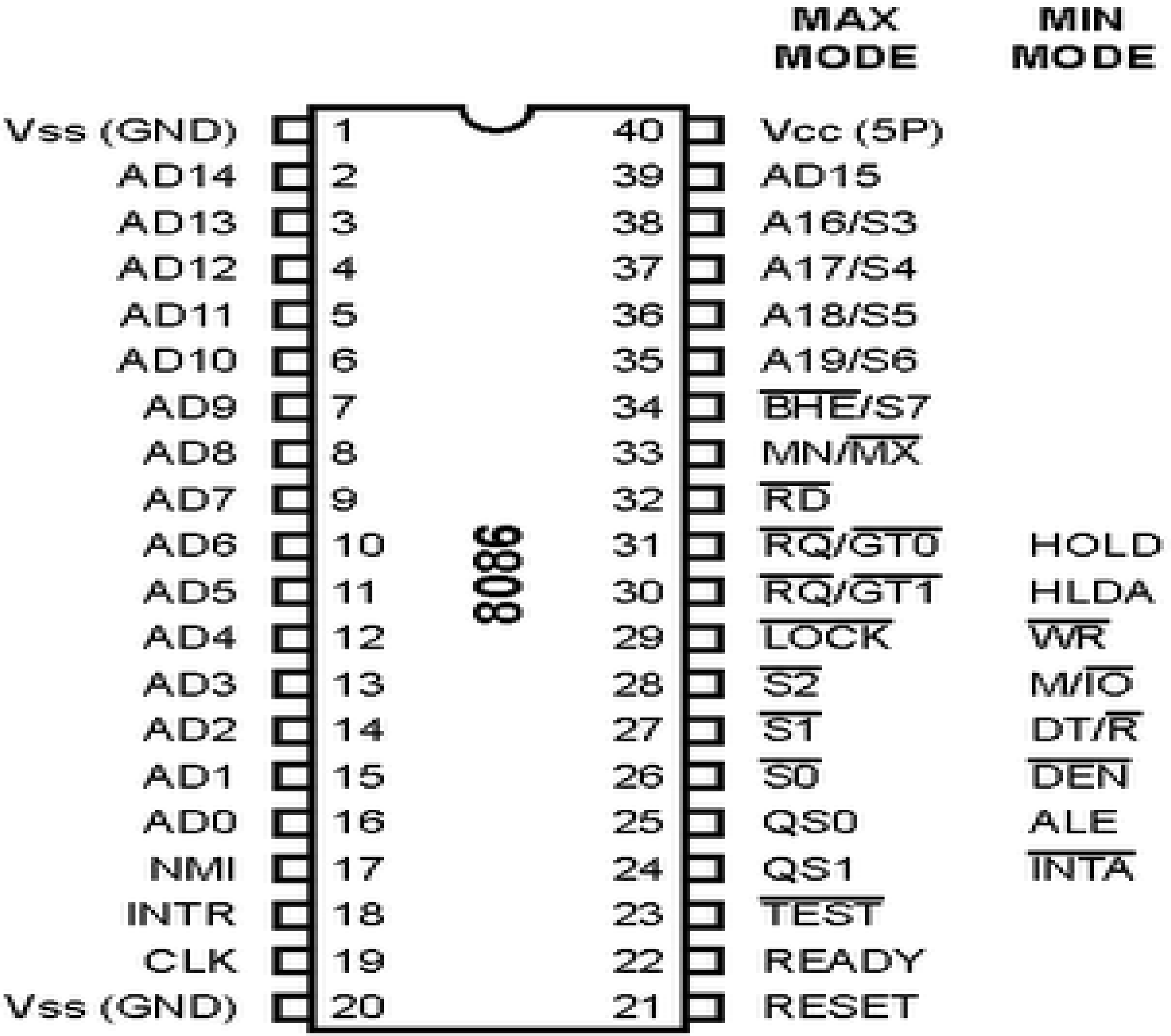
Pipeline processing in 8086

- In case of 8085, once the opcode is fetched and decoded, the external bus remain free for some time, while processor internally execute the instruction.
- This time slot is utilised in 8086 to achieve the overlapped fetch and execution cycles.
- EU execute the prefetched instruction from queue while BIU fetch the next machine code of the next instruction and arrange it in a queue.
- 8086 has FIFO 6-byte queue.
- Once a byte is decoded, queue is rearranged by pushing it out and queue status is checked for the possibility of the next opcode fetch cycle.

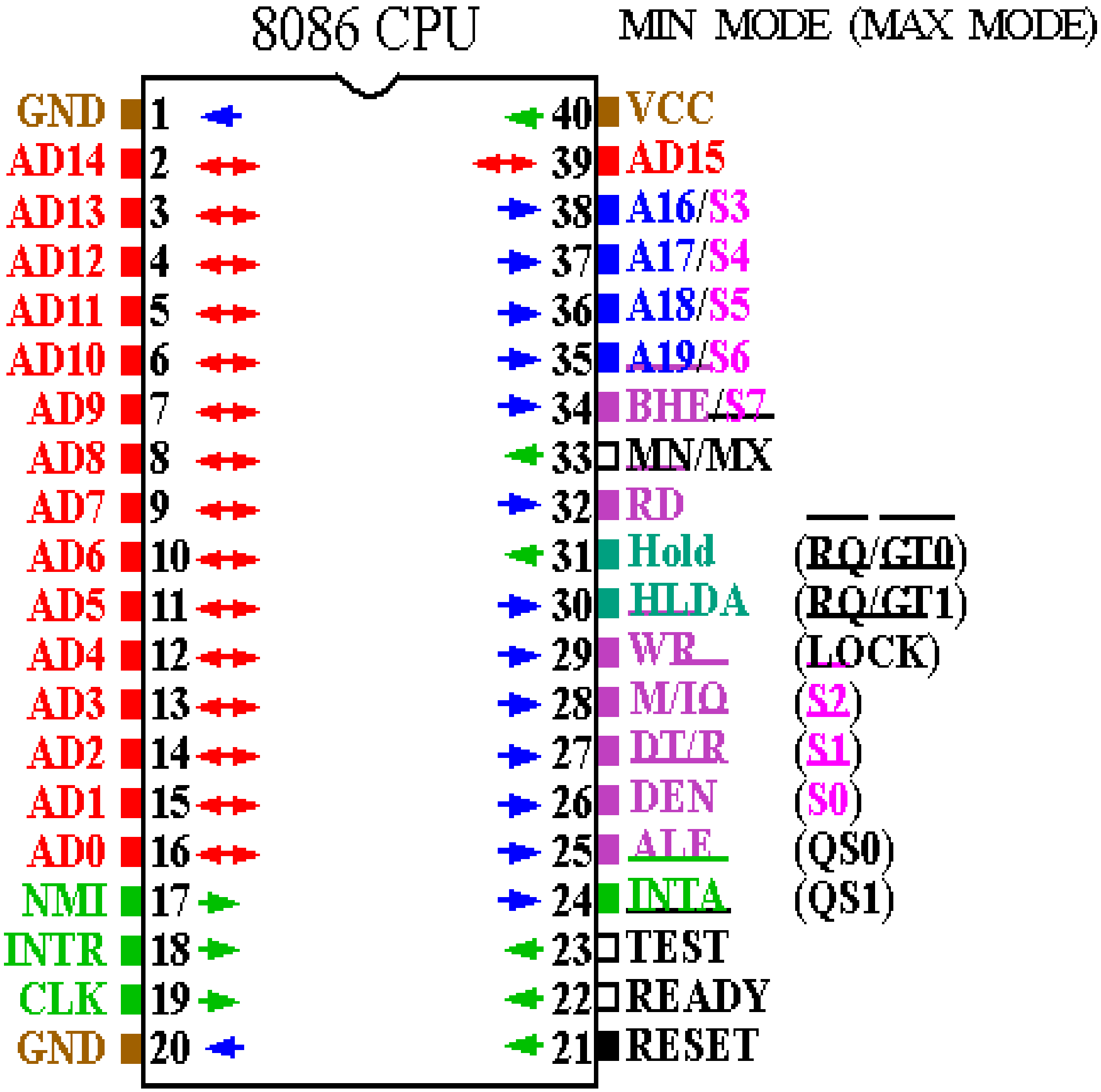
Pipeline processing

- Thus, while BIU is fetching the opcode, the EU executes the previously decoded instruction concurrently.
- The BIU along with EU thus forms a **pipeline.**

Signal Description of 8086


MAX MODE
MIN MODE
8086
Vss (GND) 1
AD14 2
AD13 3
AD12 4
AD11 5
AD10 6
AD9 7
AD8 8
AD7 9
AD6 10
AD5 11
AD4 12
AD3 13
AD2 14
AD1 15
AD0 16
NMI 17
INTR 18
CLK 19
Vss (GND) 20
40 Vcc (5P)
39 AD15
38 A16/S3
37 A17/S4
36 A18/S5
35 A19/S6
34 $\overline{\text{BHE}}$/S7
33 MN/$\overline{\text{MX}}$
32 $\overline{\text{RD}}$
31 $\overline{\text{RQ}}$/$\overline{\text{GT0}}$ HOLD
30 $\overline{\text{RQ}}$/$\overline{\text{GT1}}$ HLDA
29 $\overline{\text{LOCK}}$ $\overline{\text{WR}}$
28 $\overline{\text{S2}}$ M/$\overline{\text{IO}}$
27 $\overline{\text{S1}}$ DT/$\overline{\text{R}}$
26 $\overline{\text{S0}}$ $\overline{\text{DEN}}$
25 QS0 ALE
24 QS1 $\overline{\text{INTA}}$
23 $\overline{\text{TEST}}$
22 READY
21 RESET

Signal Description of 8086



Signal Description of 8086

- The Microprocessor 8086 is a 16-bit CPU available in different clock rates and packaged in a 40 pin DIP or plastic package.
- The 8086 operates in single processor or multiprocessor configuration to achieve high performance.
- The pins serve a particular function in minimum mode (single processor mode) and other function in maximum mode configuration (multiprocessor mode).

SIGNAL DESCRIPTION OF 8086

- The 8086 signals can be categorized in three groups.
- The first are the signal having common functions in minimum as well as maximum mode.
- The second are the signals which have special functions for minimum mode
- The third are the signals having special functions for maximum mode.

AD15-AD0

- The following signal descriptions are common for both modes.
- **AD15-AD0 :**
  - These are the time *multiplexed* memory I/O address and data lines.

– Address remains on the lines during T1 state, while the data is available on the data bus during T2, T3, Tw and T4. These lines are active high and float to a tristate during interrupt acknowledge and local bus hold acknowledge cycles.

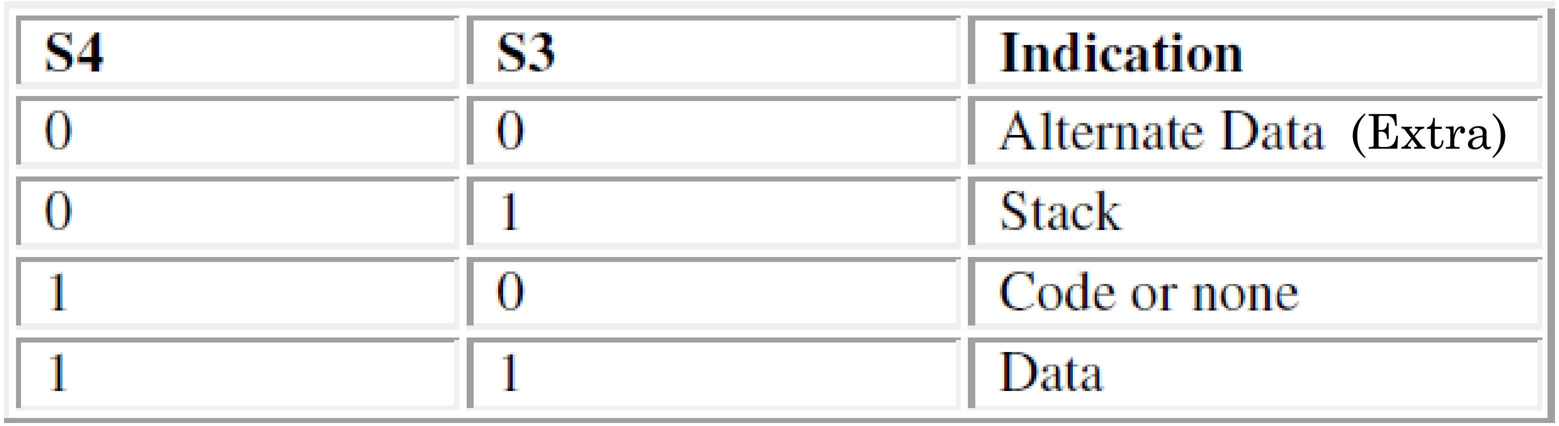
A19-A16

- **A19/S6, A18/S5, A17/S4, A16/S3 :**
  - These are the time multiplexed address and status lines. During T1 these are the most significant address lines for memory operations.
- During I/O operations, these lines are low.
- During memory or I/O operations, status information is available on those lines for T2,T3,Tw and T4.
- The status of the interrupt enable flag bit is updated at the beginning of each clock cycle.

Signal Description of 8086

- The S4 and S3 combinely indicate which segment register is presently being used for memory accesses.
- These lines float to tri-state off during the local bus hold acknowledge. The status line S6 is always low.
- The address bit are separated from the status bit using latches controlled by the ALE signal

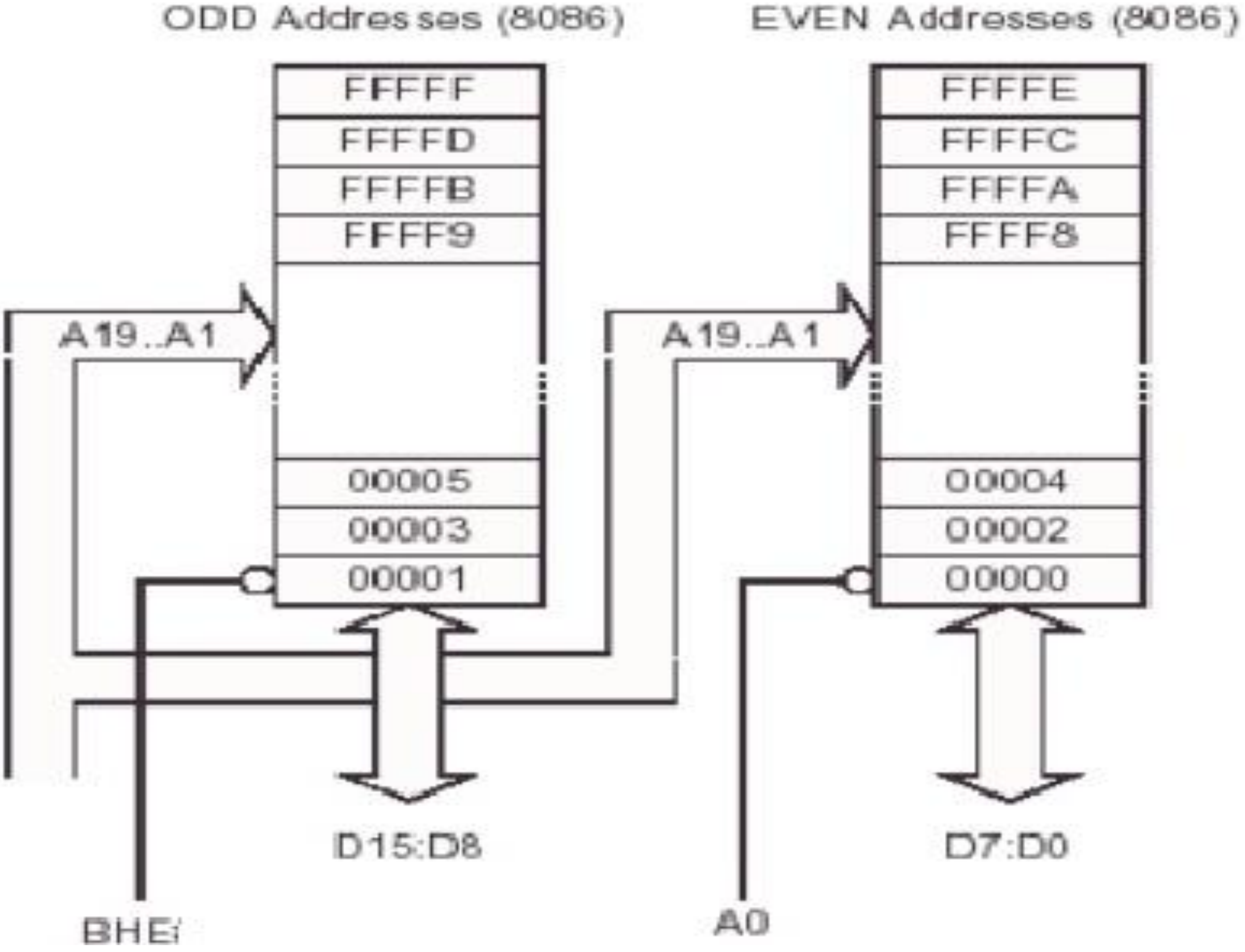
Signal Description of 8086

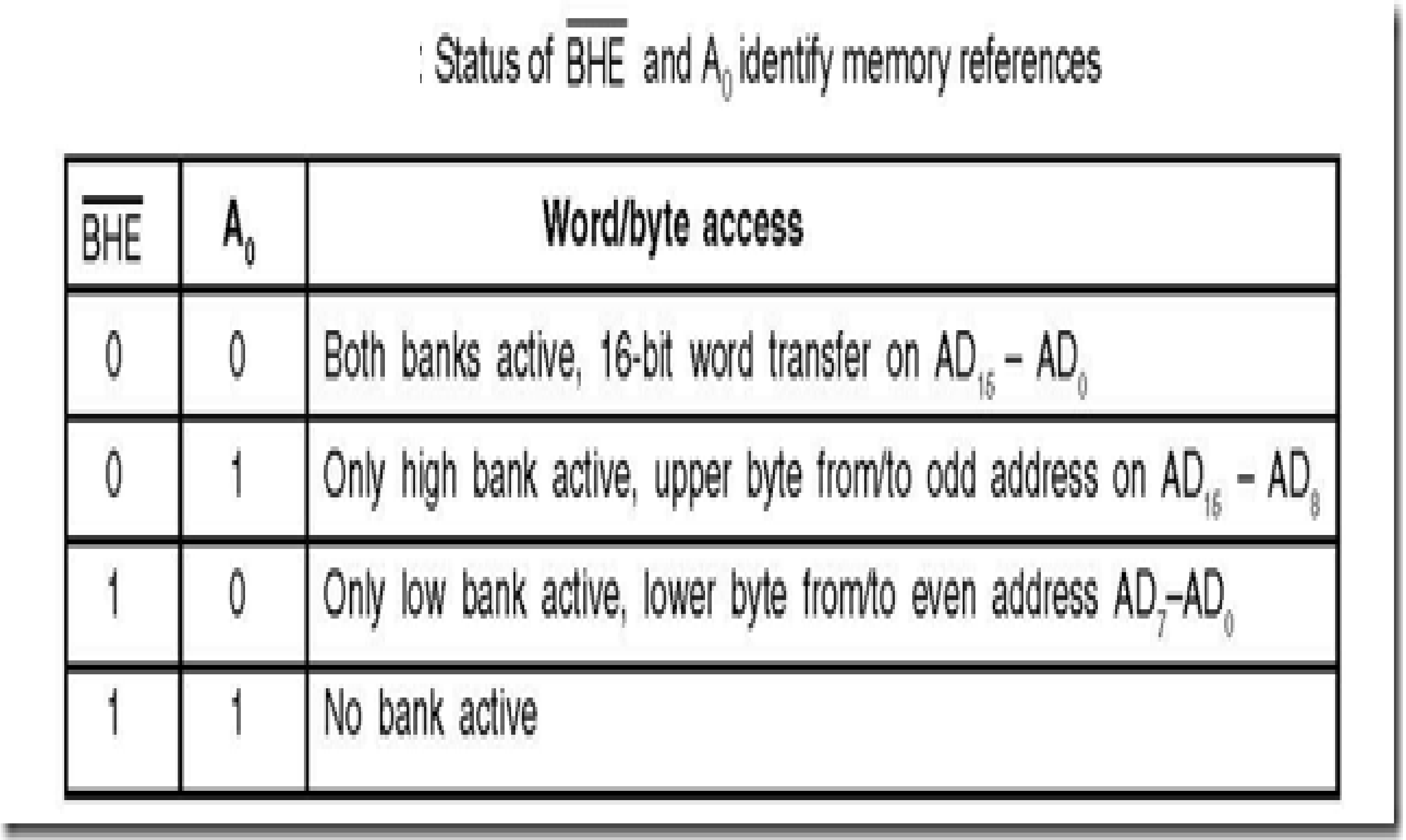


S4	S3	Indication
0	0	Alternate Data (Extra)
0	1	Stack
1	0	Code or none
1	1	Data

$\overline{\text{BHE}}$/S7

- **BHE/S7 :** The bus high enable is used to indicate the transfer of data over the higher order (D15-D8) data bus .
- It goes low for the data transfer over D15-D8 and is used to derive chip selects of odd address memory bank or peripherals.
- BHE is low during T1 for read, write and interrupt acknowledge cycles, whenever a byte is to be transferred on higher byte of data bus.
- The status information is available during T2, T3 and T4.
- The signal is active low and tristated during hold.
- It is low during T1 for the first pulse of the interrupt acknowledge cycle.


ODD Addresses (8086)
EVEN Addresses (8086)
FFFFF
FFFFD
FFFFB
FFFF9
00005
00003
00001
FFFFE
FFFFC
FFFFA
FFFF8
00004
00002
00000
A19..A1
A19..A1
D15:D8
D7:D0
BHE
A0



: Status of $\overline{BHE}$ and A_0 identify memory references

$\overline{BHE}$	A_0	Word/byte access
0	0	Both banks active, 16-bit word transfer on AD_{15} – AD_0
0	1	Only high bank active, upper byte from/to odd address on AD_{15} – AD_8
1	0	Only low bank active, lower byte from/to even address AD_7–AD_0
1	1	No bank active

RD, READY

- **$\overline{\text{RD}}$ – Read :**
  - This signal on low indicates the peripheral that the processor is performing memory or I/O read operation. RD is active low and shows the state for T2, T3, Tw of any read cycle. The signal remains tristated during the hold acknowledge.
- **READY :**
  - This is the acknowledgement from the slow device or memory that they have completed the data transfer. The signal made available by the devices is synchronized by the 8284A clock generator to provide ready input to the 8086. the signal is active high.

INTR

- **INTR-Interrupt Request :** This is a triggered input.
- This is sampled during the last clock cycles of each instruction to determine the availability of the request.
- If any interrupt request is pending, the processor enters the interrupt acknowledge cycle.
- This can be internally masked by resulting the interrupt enable flag.
- This signal is active high and internally synchronized.

TEST,CLK

- **TEST :**
  - This input is examined by a ‘WAIT’ instruction. If the TEST pin goes low, execution will continue, else the processor remains in an idle state.
- The input is synchronized internally during each clock cycle on leading edge of clock.
- **CLK- Clock Input :**
  - The clock input provides the basic timing for processor operation and bus control activity. Its an asymmetric square wave with 33% duty cycle. The range of frequency for different version is form 5 MHz to 10MHz

RESET

- It is a system reset.
- It is an active high signal.
- When high, microprocessor enters into reset state and terminates the current activity and start execution from FFFF0H.
- It must be active for at least four clock cycles to reset the microprocessor.
- It is also internally synchronised

NMI

- It is a non-maskable interrupt signal.
- It is an active high signal.
- It is an edge triggered interrupt which causes Type-2 interrupt.
- A transition from low to high initiate the interrupt response at the end of the current instruction.
- The input is synchronised internally.

Vcc AND Vss

- Vcc is power supply signal.
- +5V DC is supplied through this pin.
- Vss is ground signal.

MN / $\overline{\text{MX}}$

- 8086 works in two modes:
  - Minimum Mode
  - Maximum Mode
- If MN/$\overline{\text{MX}}$ is high, it works in minimum mode.
- If MN/$\overline{\text{MX}}$ is low, it works in maximum mode.
- Pins 24 to 31 issue two different sets of signals.
- One set of signals is issued when CPU operates in minimum mode.
- Other set of signals is issued when CPU operates in maximum mode.

Pin Description for Minimum Mode

$\overline{\text{INTA}}$

- This is an interrupt acknowledge signal.
- When microprocessor receives INTR signal, it acknowledges the interrupt by generating this signal.
- It means that the processor has accepted the interrupt.
- It is an active low during T_2, T_3 and Tw of each interrupt acknowledgement cycle.

ALE

- This is an Address Latch Enable signal.
- It indicates that valid address is available on bus AD0 – AD15.
- It is an active high signal and remains high during T1 state.
- It is connected to enable pin of latch 8282.

$\overline{\text{DEN}}$

- This is a Data Enable signal.
- This signal is used to enable the transceiver
- Transceiver is used to separate the data from the address/data bus.
- It is active from the middle of T2 until the middle of T4.
- DEN in tristated during hold acknowledgement cycle.
- It is an active low signal.

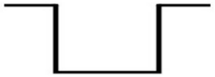
DT / $\overline{R}$

- This is a Data Transmit/Receive signal.
- It decides the direction of data flow through the transceiver.
- When it is high, data is transmitted out.
- When it is low, data is received in.
- DEN in tristated during hold acknowledgement cycle.

M / $\overline{\text{IO}}$

- This signal is issued by the microprocessor to distinguish memory access from I/O access.
- When it is high, memory is accessed.
- When it is low, I/O devices are accessed.
- This line becomes active in the previous T4 and remain active till final T4 of the current cycle.
- It is tristated during hold acknowledgement cycle.

$\overline{WR}$

- It is a Write signal.
- It is used to write data in memory or output device depending on the status of M/IO signal.
- It is an active low signal. 

HOLD

- It is an active high signal.
- When the HOLD line goes high, it indicates another master is requesting the bus access.
- When microprocessor receives HOLD signal, it issues HLDA signal in the middle of the next clock cycle after completing the current bus cycle.

HOLD

- If the DMA request is made while the CPU is performing the memory or I/O cycle it will release the local bus during T4 provided:
  - The request occurs on or before T2 state of the current cycle.
  - The current cycle is not operating over the lower byte of a word (or operating on odd address)
  - The current cycle is not the first acknowledgement of an acknowledge sequence.
  - A LOCK instruction is not being executed.

HLDA

- It is a Hold Acknowledge signal.
- It is issued after receiving the HOLD signal.
- It is an active high signal.

Pin Description for Maximum Mode

QS1 AND QS0

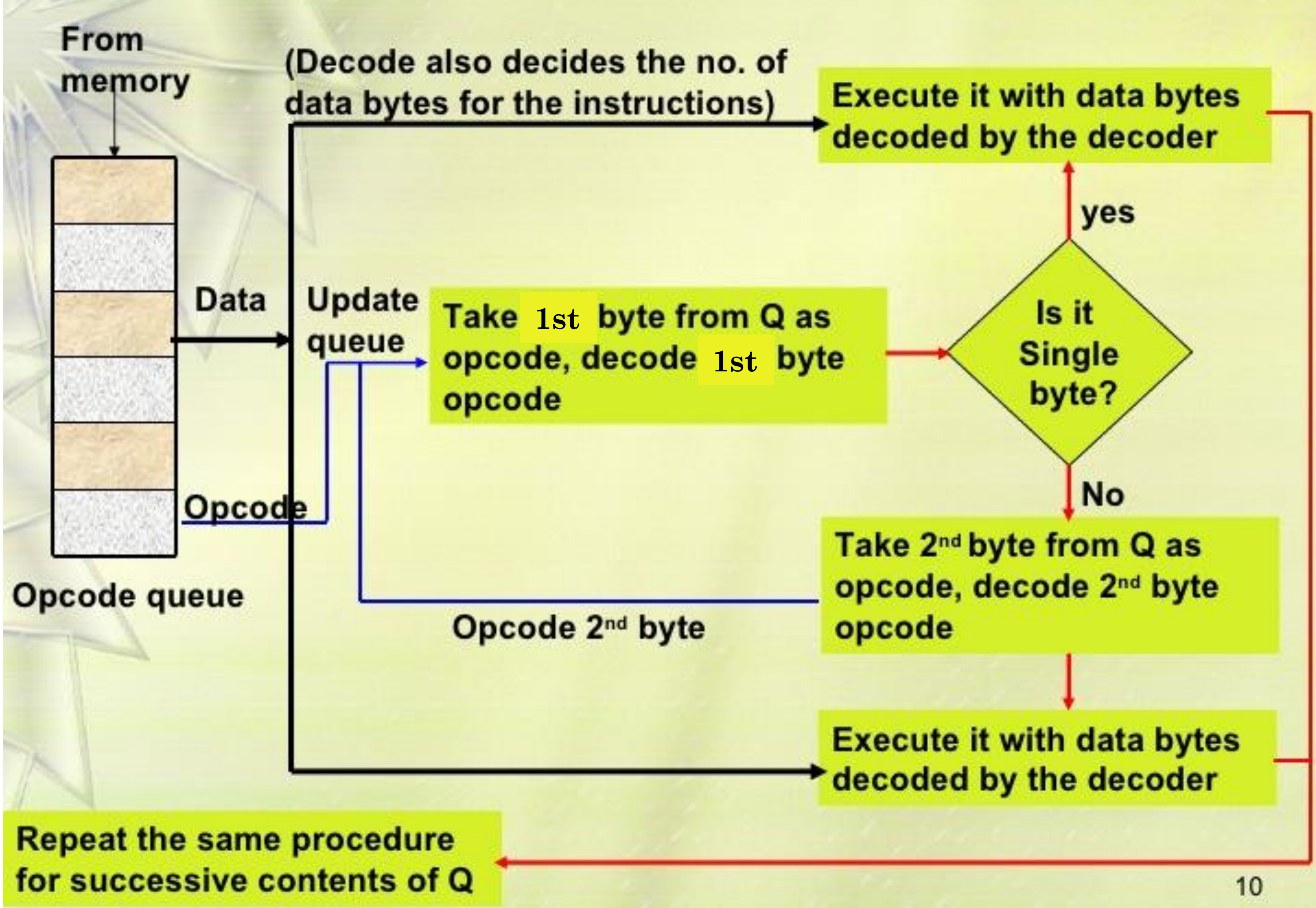
- These pins provide the status of instruction queue.

QS1	QS0	Status
0	0	No operation
0	1	1st byte of opcode from queue
1	0	Empty queue
1	1	Subsequent byte from queue

THE QUEUE (Q)

- The BIU uses a mechanism known as an **instruction stream queue** to implement a ***pipeline architecture.***
- This queue permits pre-fetch of up to **6 bytes** of instruction code. Whenever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by pre-fetching the next sequential instruction.

The Queue Operation



From memory

Data

Opcode

Opcode queue

Update queue

(Decode also decides the no. of data bytes for the instructions)

Execute it with data bytes decoded by the decoder

Take 1st byte from Q as opcode, decode 1st byte opcode

Is it Single byte?

yes

No

Take 2nd byte from Q as opcode, decode 2nd byte opcode

Opcode 2nd byte

Execute it with data bytes decoded by the decoder

Repeat the same procedure for successive contents of Q

$\overline{S0}$, $\overline{S1}$, $\overline{S2}$

- These status signals indicate the operation being done by the microprocessor.
- This information is required by the Bus Controller 8288.
- Bus controller 8288 generates all memory and I/O control signals.

S2	S1	S0	Status
0	0	0	Interrupt Acknowledge
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	Halt
1	0	0	Opcode Fetch
1	0	1	Memory Read
1	1	0	Memory Write
1	1	1	Passive

$\overline{\text{LOCK}}$

- This signal indicates that other processors should not ask CPU to **relinquish** the system bus.
- When it goes low, all interrupts are masked and HOLD request is not granted.
- This pin is activated by using LOCK *prefix* on any instruction and remains active until the completion of the instruction.
- When critical instruction executes, which requires the system buses, the LOCK *prefix* is used.

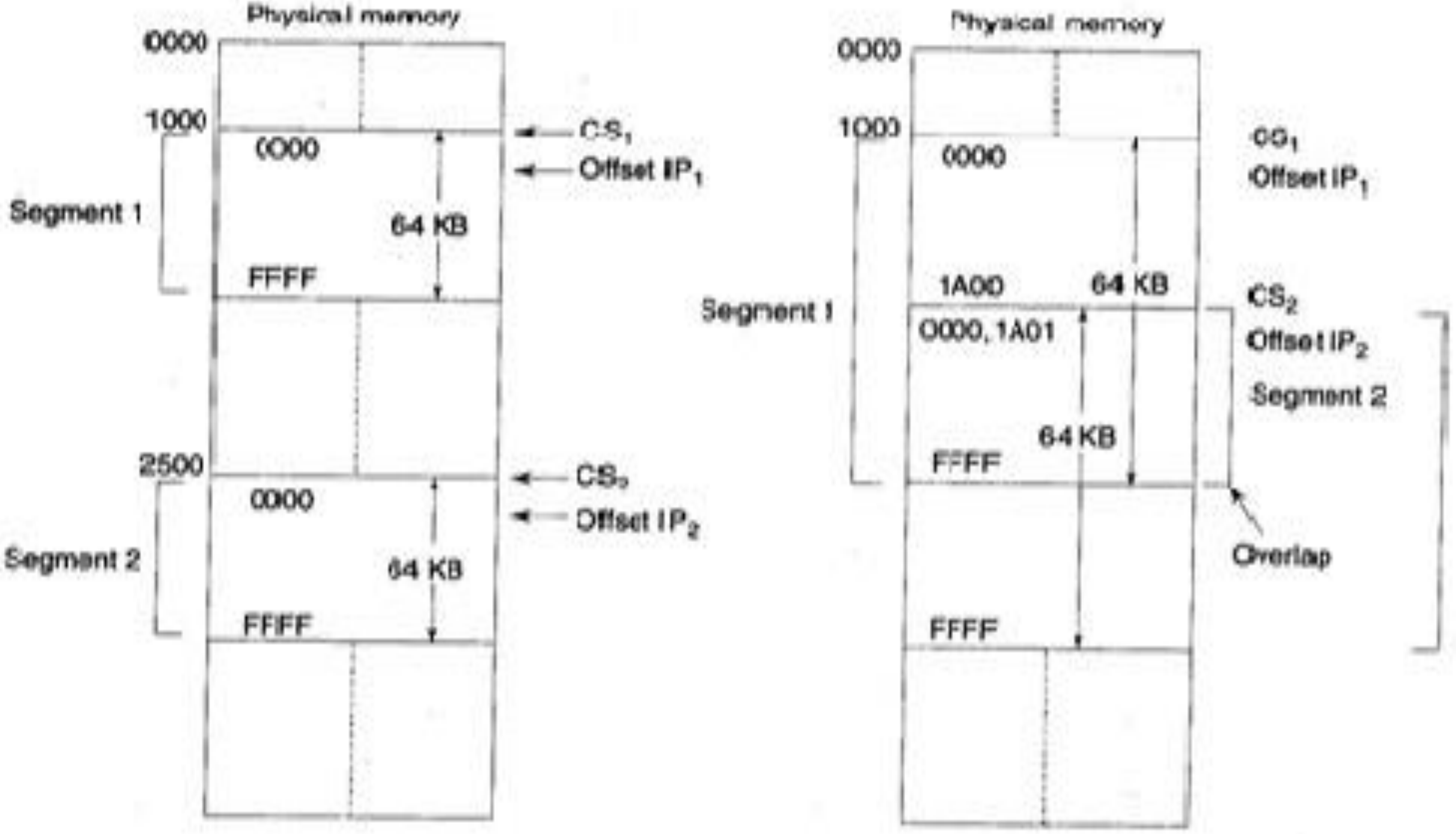
$\overline{RQ}/\overline{GT}_1$ AND $\overline{RQ}/\overline{GT}_0$

- These are Request/Grant pins.
- Other processors request the CPU through these lines to release the system bus.
- After receiving the request, CPU sends acknowledge signal on the same lines.
- $\overline{RQ}/\overline{GT}_0$ has higher priority than $\overline{RQ}/\overline{GT}_1$.

$\overline{RQ}/\overline{GT}_1$ AND $\overline{RQ}/\overline{GT}_0$

- The request/ grant sequence is as follows:
  - A pulse of one clock wide from another bus master requests the bus access to 8086.
  - During T_4 (current) or T_1 (next) clock cycle, a pulse of one clock wide from 8086 to the requesting master, indicates that the 8086 has allowed the local bus to float and that it will enter the 'hold acknowledgement' state in the next clock cycle.
  - A one clock wide pulse from the another master indicates to 8086 that the 'hold' request is about to end and the 8086 may regain control of the bus at the next clock cycle.

Segmentation of Memory



ADVANTAGE OF MEMORY SEGMENTATION

- Allows the memory capacity to be 1 MB although the actual address to be handled are of 16 bit size.
- Allows the placing the code, data and stack portion of the same program in the different parts (segment) of memory, for data code protection.
- Permits a program and/or its data to be put into different areas of memory each time the program is executed, i.e. provision for relocation is done.