INTEL 8086
Software & Hardware Architecture

Microprocessor Notes By Prof. Nafisa M. Mapari

MODULE 1 & 2

Complete idea about INTEL 8086 Microprocessor

Topics to be covered

1. **Software Architecture of the INTEL 8086.**

1. **Hardware Architecture of INTEL 8086.**

1. **8086 Programming and program development.**

Features

- **8086 is a 16- bit microprocessor.**
- **8086 has a 16-bit data bus.**
- **8086 has 20 –bit address bus.**
- **8086 can generate 16-bit I/O address, hence it can access 2^{16} = 65536 I/O ports**
- **8086 provides fourteen 16-bit registers.**
- **8086 has multiplexed address & data bus which reduces the number of pins needed.**
- **external bus controller (8288).**

Features

- **8086 supports 8086 is possible to perform bit, byte, word & block operations.**
- **8086 is designed to operate in two modes, namely the *minimum mode & the maximum mode.***
- **In minimum mode- only one 8086 CPU is to be used in a microcomputer system. The CPU issues the control signals required by the memory & I/O devices.**
-

Features

- **In multiprocessor system, 8086 operates in maximum mode. The control signals are generated with the help of**
- **multi-programming. In multi-programming, the code for two or more processes is in memory at the same time & executed in a time-multiplexed fashion.**

Features

- **8086 provides powerful instruction set with following addressing modes: Register, Immediate, Direct, Indirect through an index or base, Indirect through the sum of a base & an index register, relative & implied.**

Software architecture of the INTEL 8086

- *Memory segmentation and addressing*
- *Block diagram of 8086*
- *Address space & Data organization*
- *Data Types*
- *Registers*
- *Stack*
- *I/O space*

Hardware Architecture of INTEL 8086

- *Pin Diagram and Pin Details*
- *min/max mode*
- *Coprocessor and Multiprocessor configuration*
- *Hardware organization of address space*
- *Control signals*
- *I/O interfaces*

8086 programming and program development.

- *Assembly Language Programming.*
- *Instruction Set.*
- *Assembler Directives.*
- *Programming Exercises.*

Software Architecture of INTEL 8086

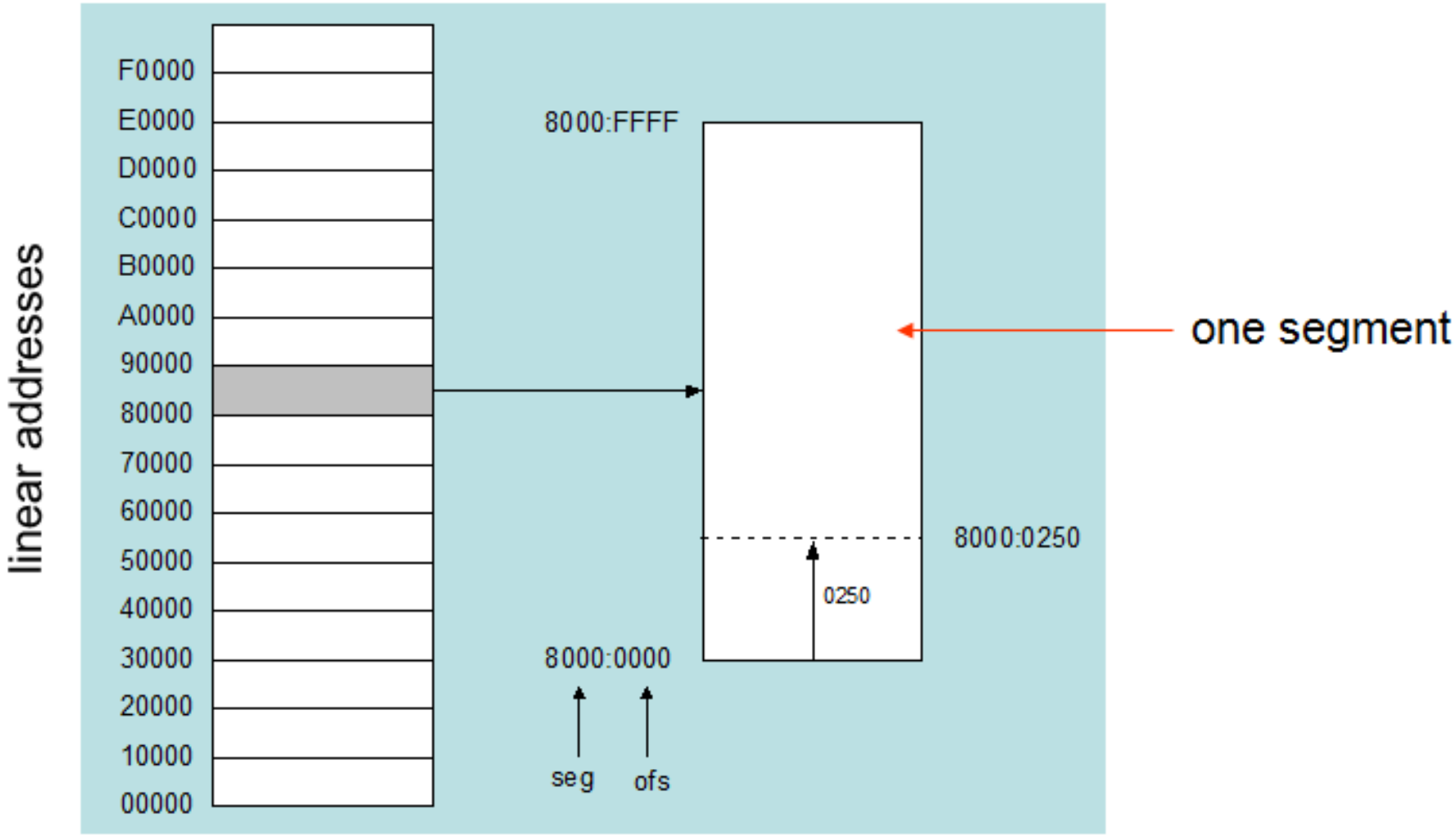
Software architecture of the INTEL 8086

- *Memory segmentation and addressing*
- *Block diagram of 8086*
- *Address space & Data organization*
- *Data Types*
- *Registers*
- *Stack*
- *I/O space*

Memory segmentation and addressing

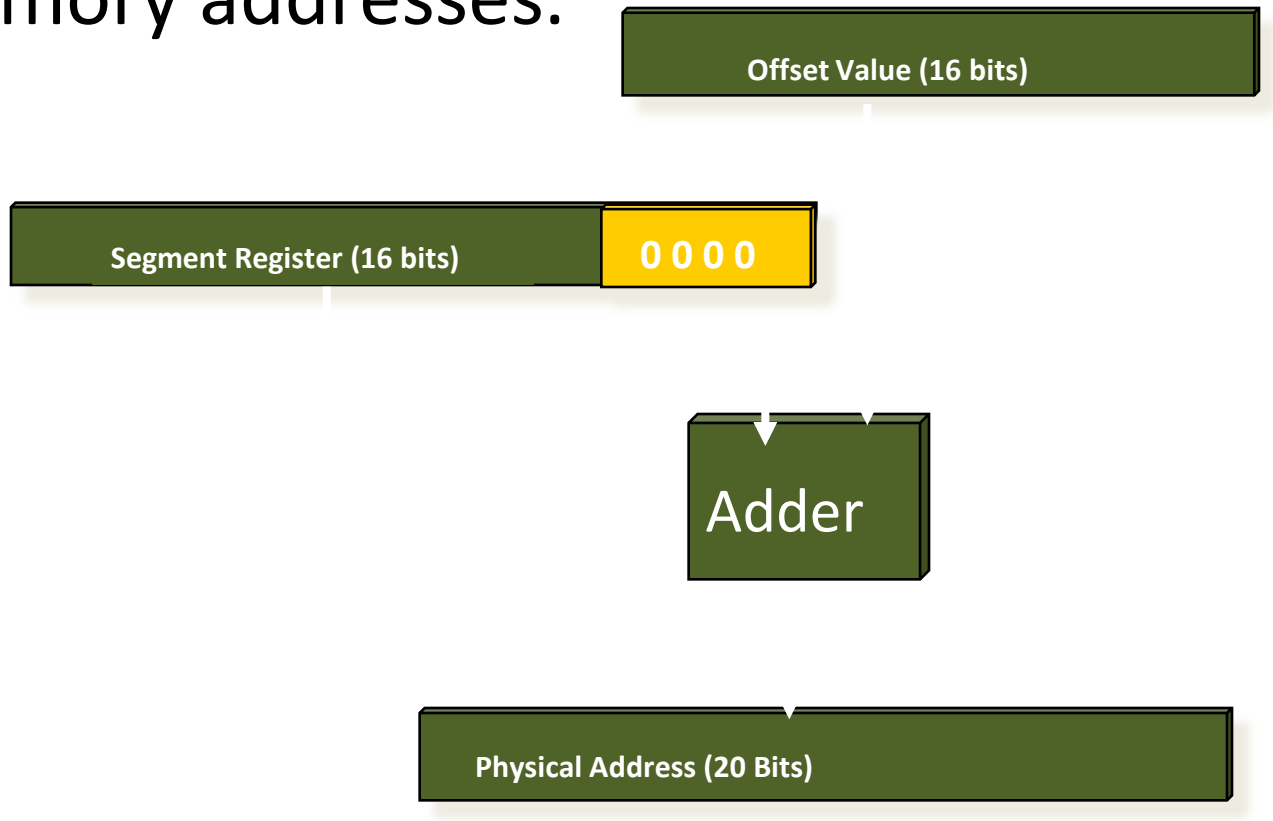
- Von – Newman architecture & Harvard architecture
- Program Memory & Data Memory
- Need for Segmentation
  - To implement Harvard architecture
  - Easy to debug
  - Same Interfacing ICs can be used
  - To avoid overlap of stack with normal memory
  - Compatible with 8085

Segmented Memory



Memory Address Generation

- The BIU has a dedicated adder for determining physical memory addresses.



Segment : Offset Address

- Logical Address is specified as **segment:offset**
- Physical address is obtained by shifting the segment address 4 bits to the left and adding the offset address.
- Thus the physical address of the logical address **A4FB:4872** is:

```
  A4FB0
+ 4872
-------
  A9822
```

Segments, Segment Registers & Offset Registers

- Segment Size = 64KB
- Maximum number of segments possible = 14
- Logical Address – 16 bits
- Physical Address – 20 bits
- 2 Logical Addresses for each Segments.
  - Base Address (16 bits)
  - Offset Address (16 bits)
- Segment registers are used to store the Base address of the segment.

Segments, Segment Registers & Offset Registers

- 4 Segments in 8086
  - Code Segment (CS)
  - Data Segment (DS)
  - Stack Segment (SS)
  - Extra Segment (ES)

SEGMENT	SEGMENT REGISTER	OFFSET REGISTER
Code Segment	CSR	**Instruction Pointer (IP)**
Data Segment	DSR	**Source Index (SI)**
Extra Segment	ESR	**Destination Index (DI)**
Stack Segment	SSR	**Stack Pointer (SP) / Base Pointer (BP)**

Block diagram of 8086

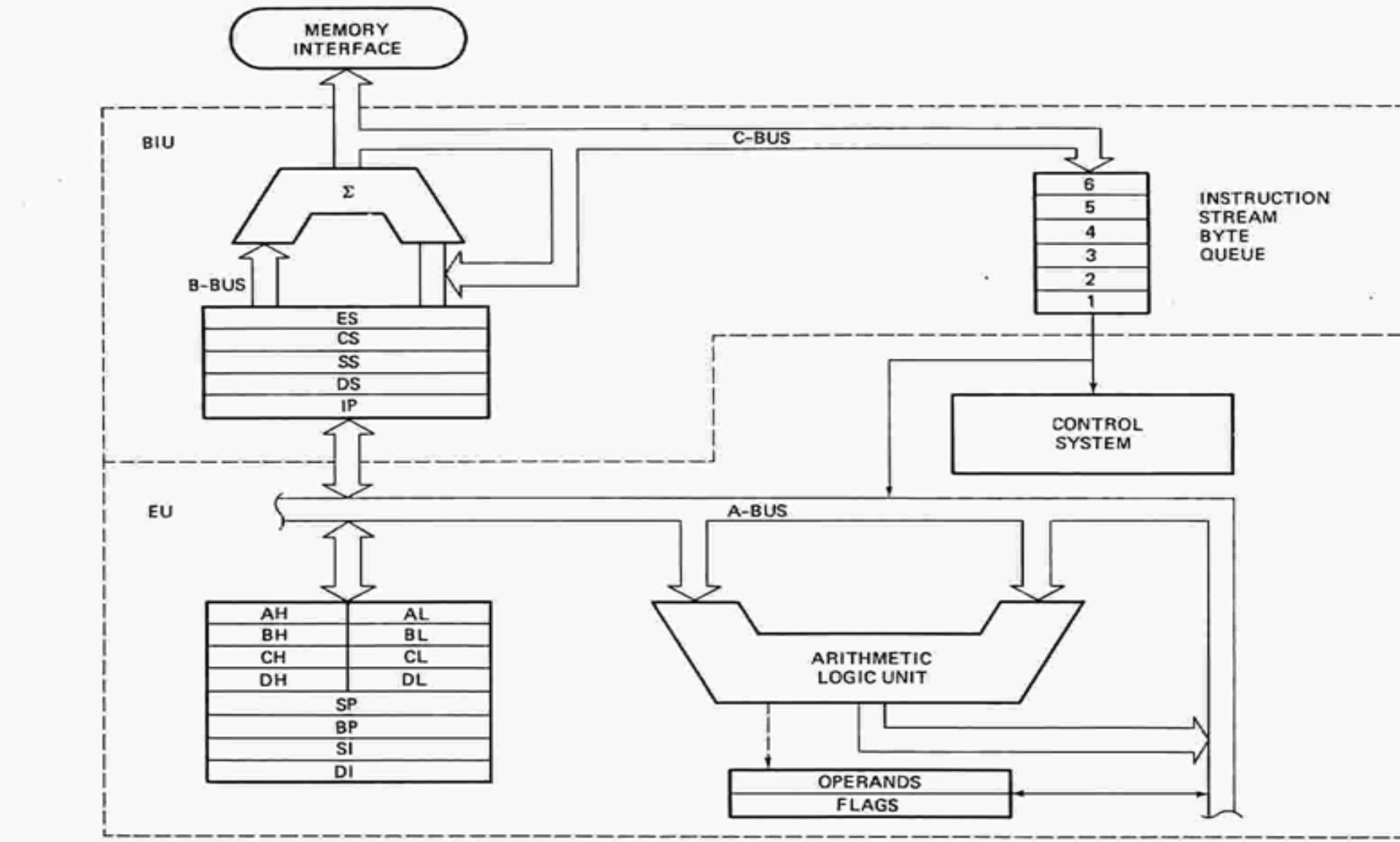
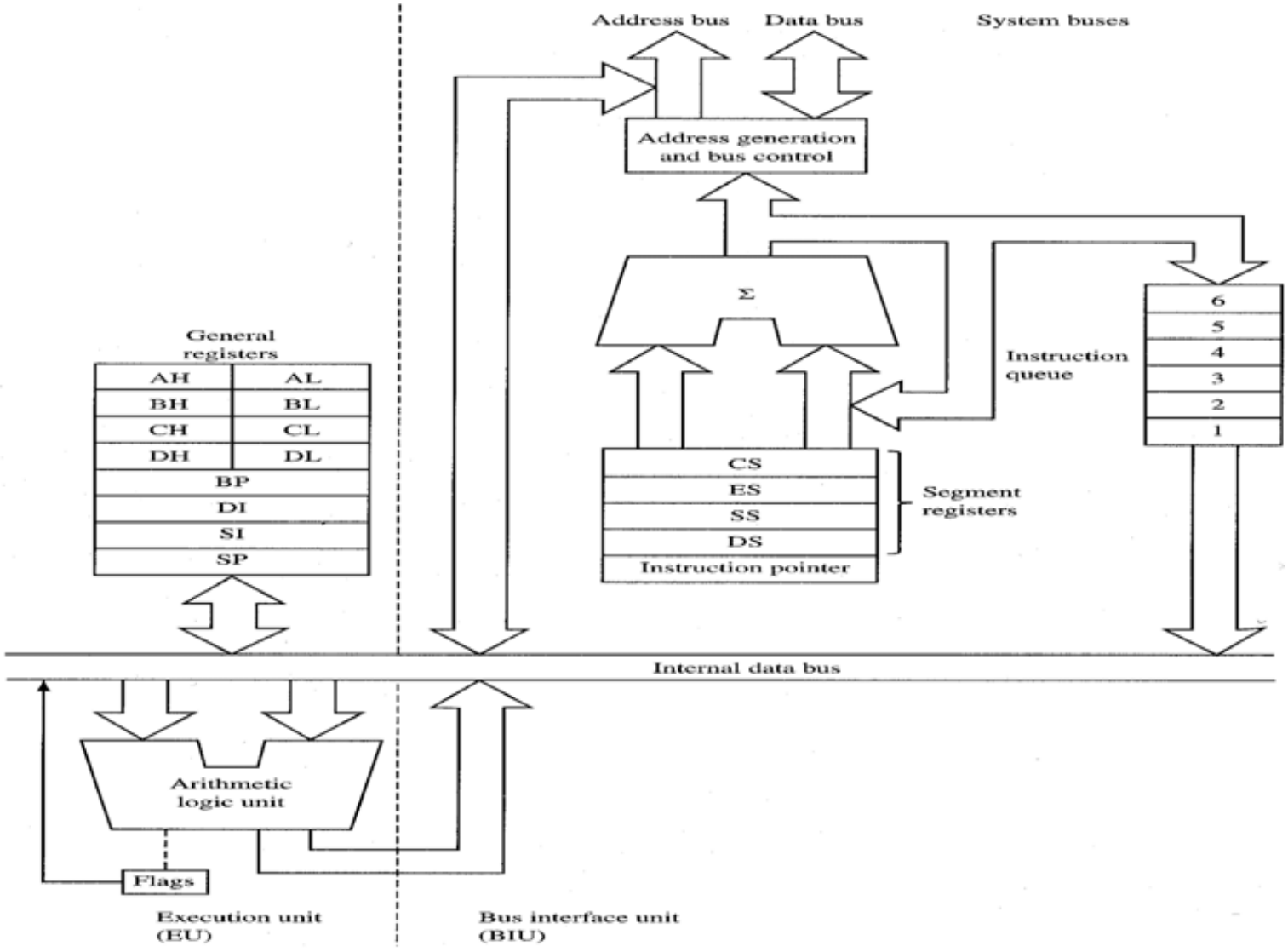


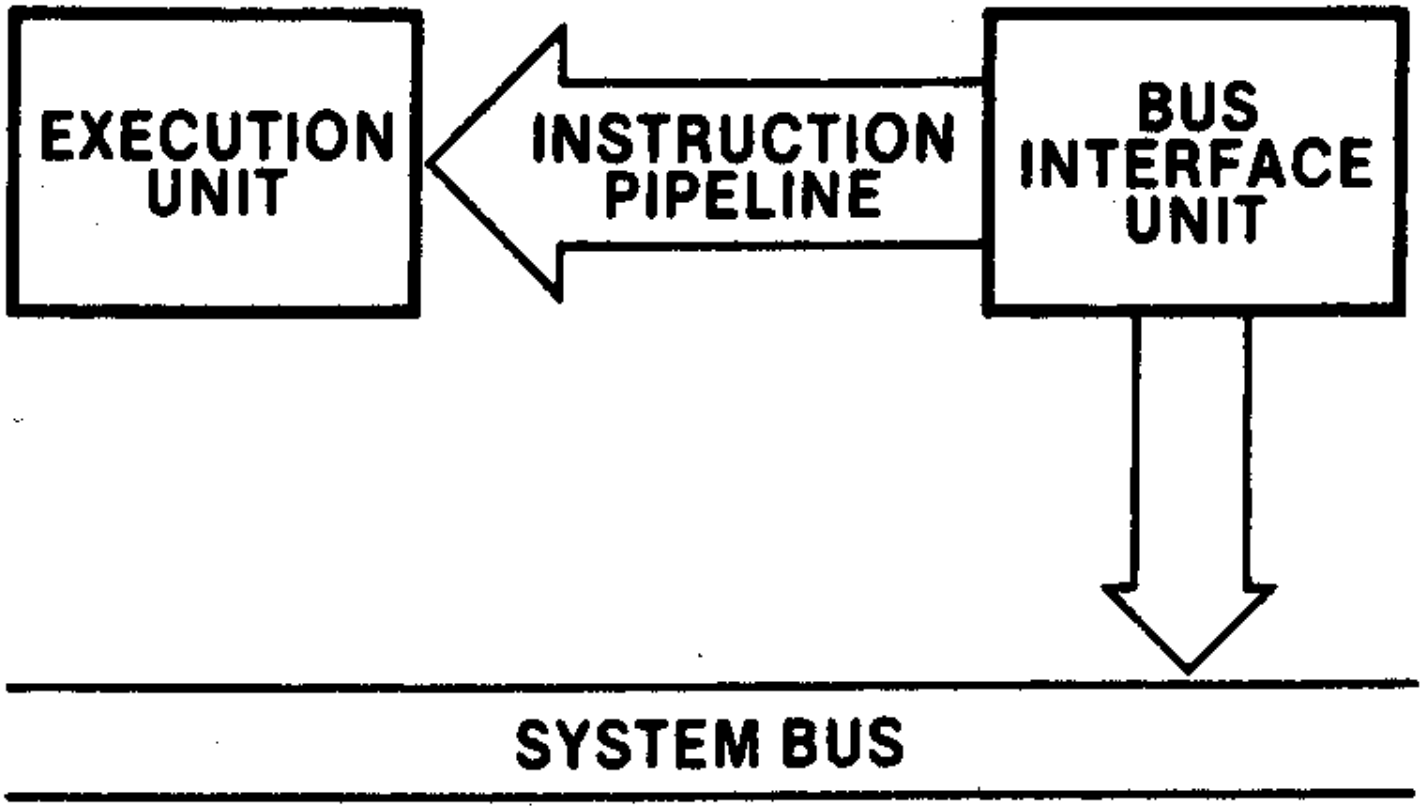
FIGURE 2-7 8086 internal block diagram. (*Intel Corp.*)

Block diagram of 8086

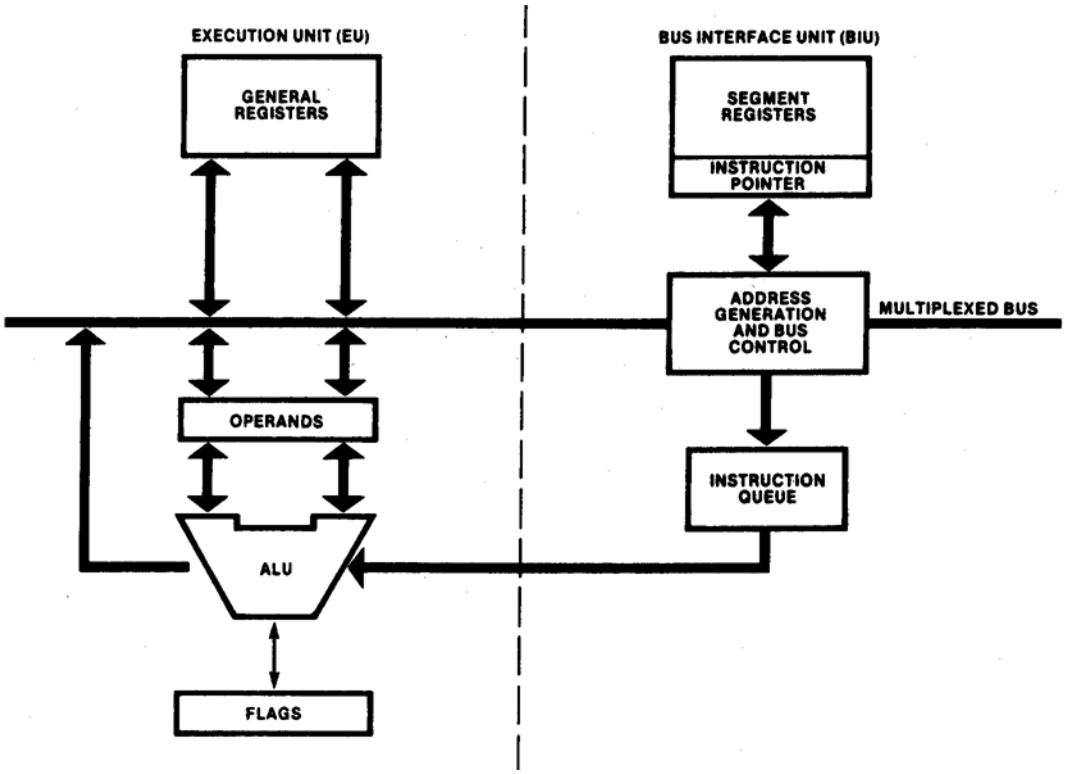
Figure 3.1 Processor model for the 8086 microprocessor. A separate execution unit (EU) and bus interface unit (BIU) are provided.



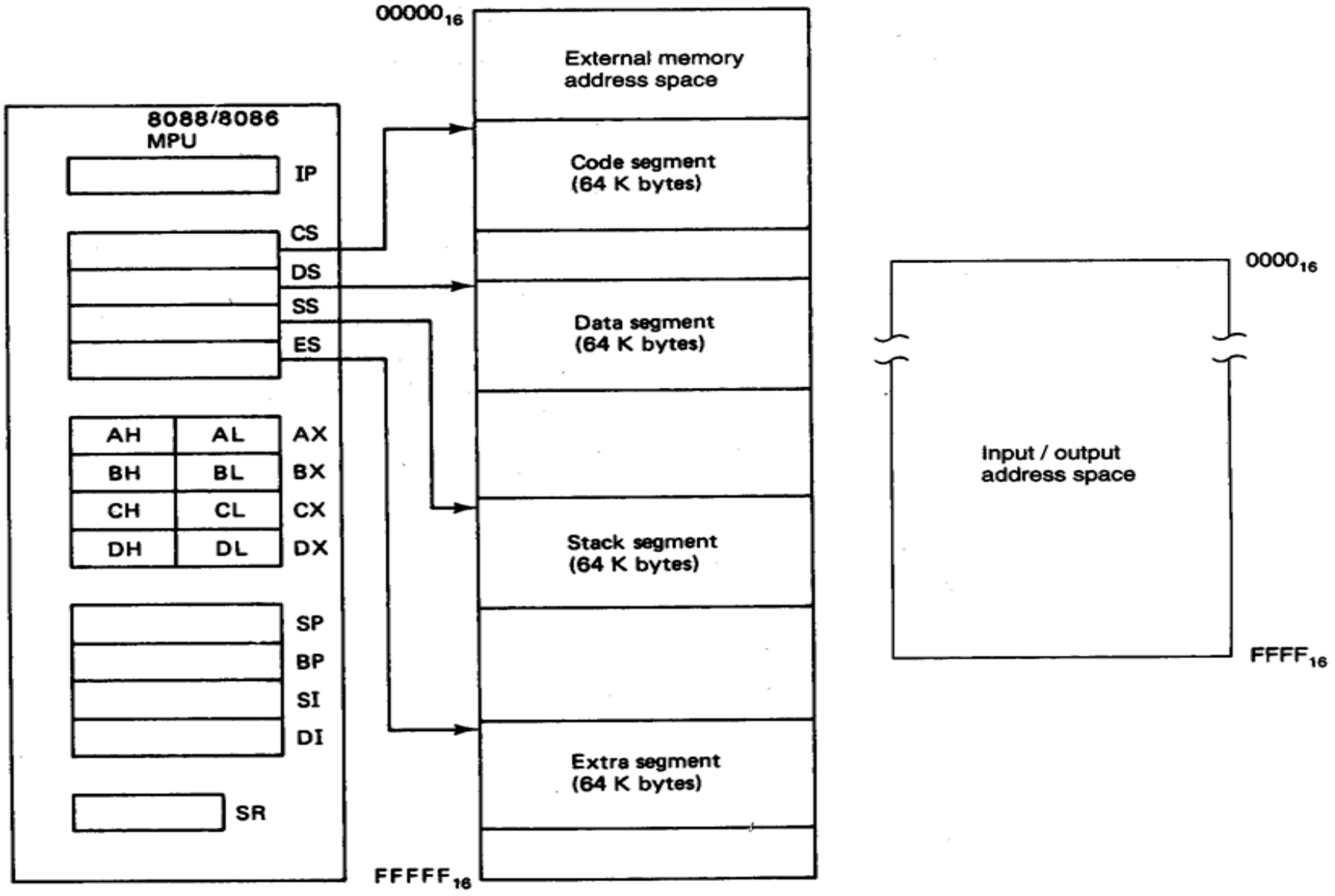
Pipelined architecture of the 8086 microprocessors



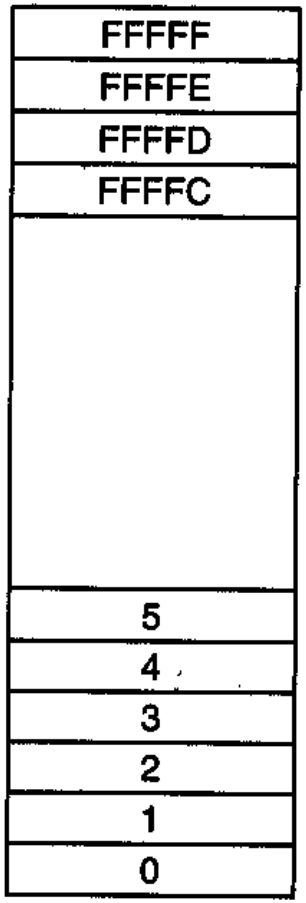
Execution and bus interface units



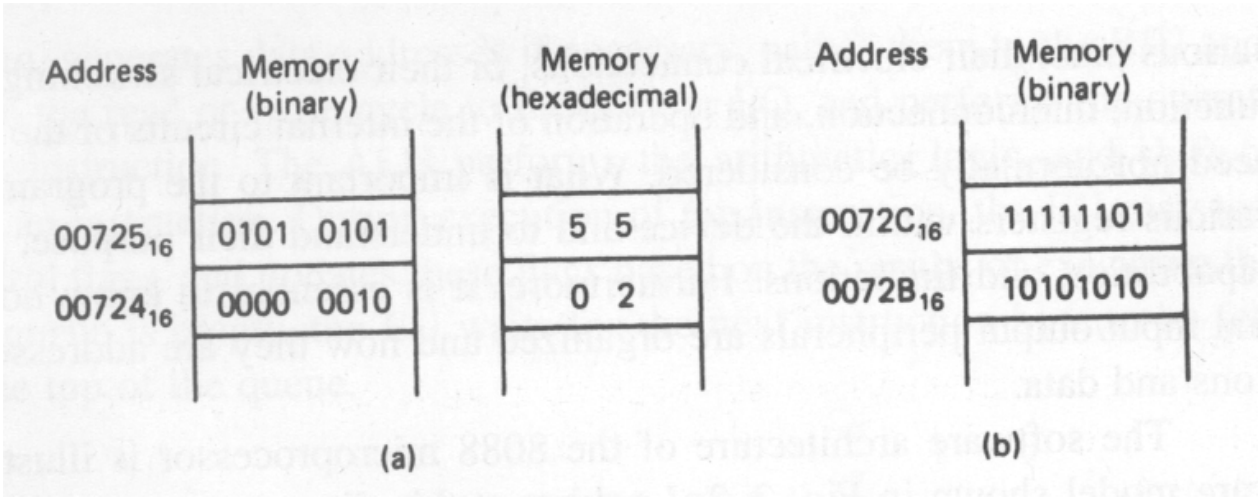
Software Model of the 8086 Microprocessors



Address space & Data organization



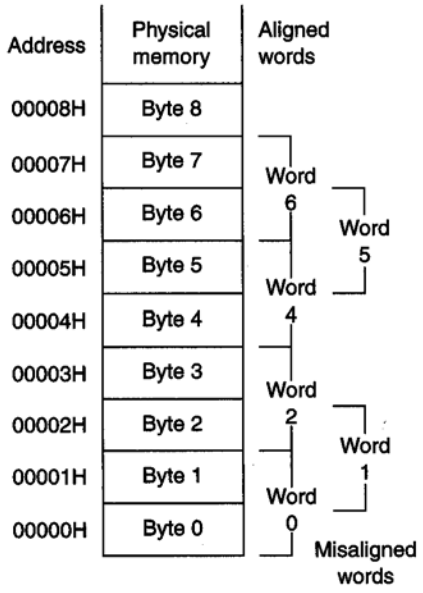
Memory address space



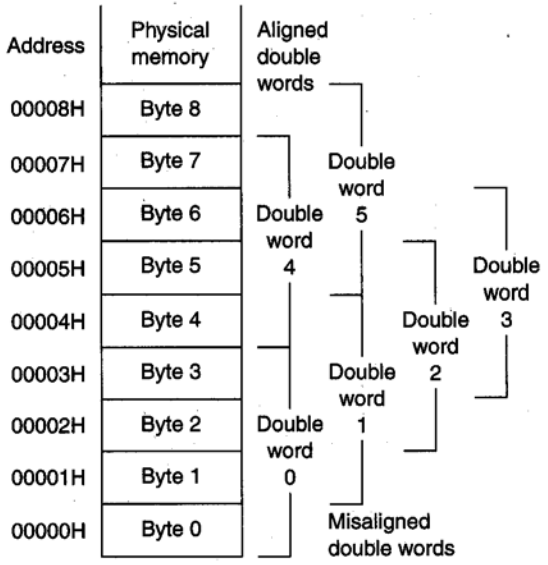
Storing a word in memory

What is the word in (b) in Hex?

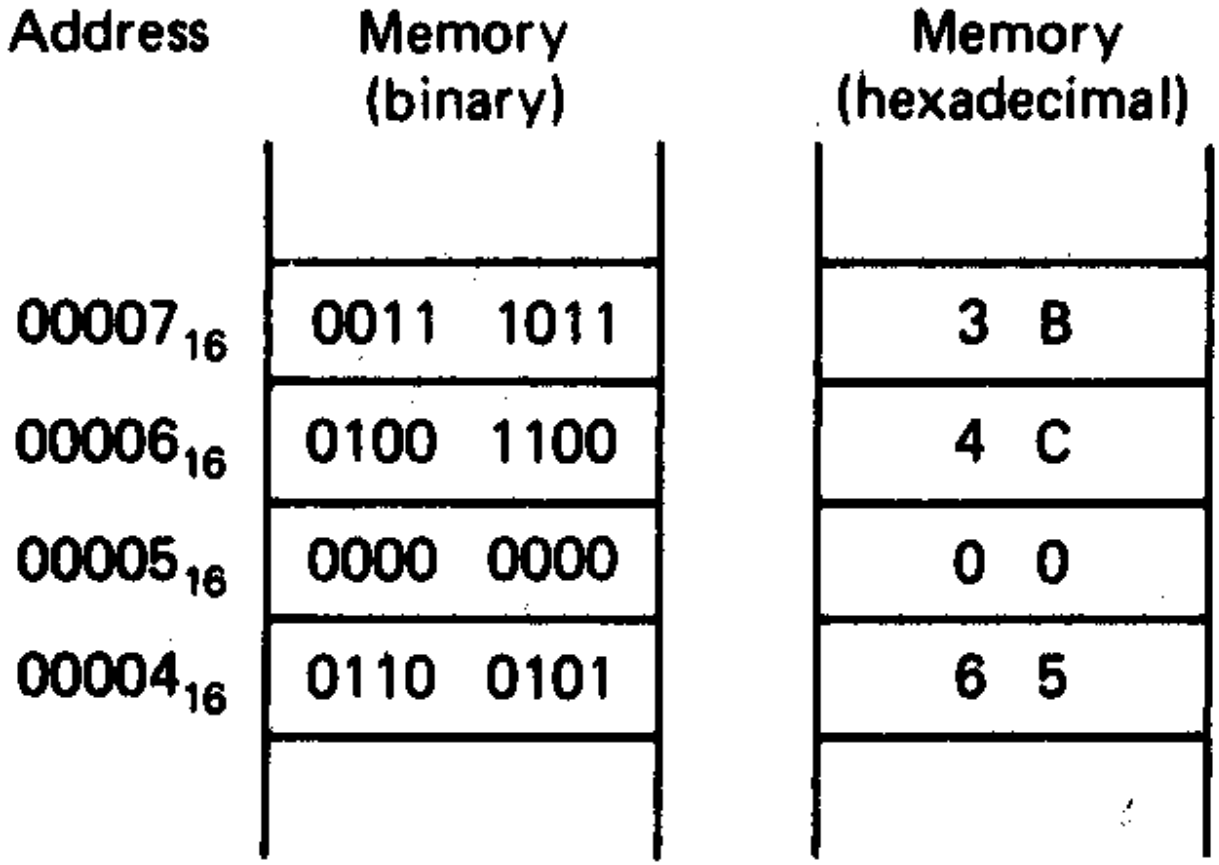
Aligned and misaligned data word



Aligned and misaligned double words of data

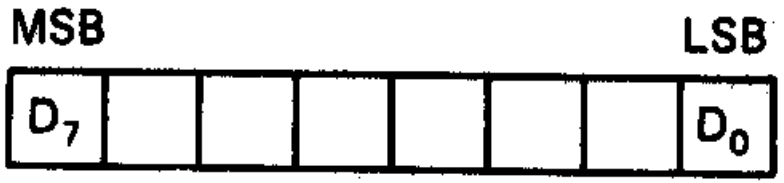


Storing double word in memory



Data Types

Unsigned byte integer
0 - 255

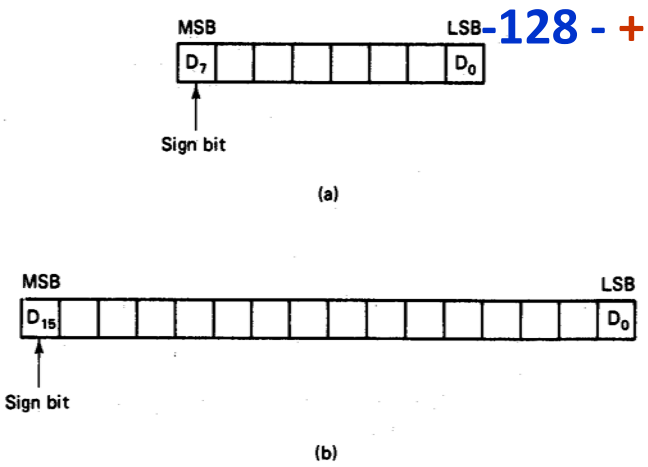


(a)

Unsigned word integer
0 – 65,535

Data Types

Signed integers



-128 - +127

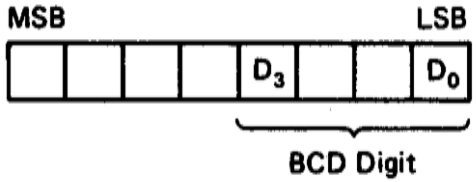
-32,768 - +32,767

Data Types

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

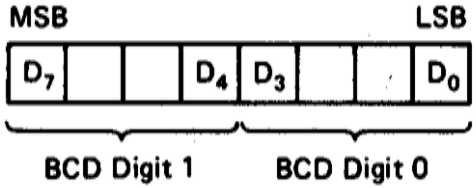
(a)

Binary Coded Decimal (BCD)



(b)

Unpacked BCD



(c)

Packed BCD

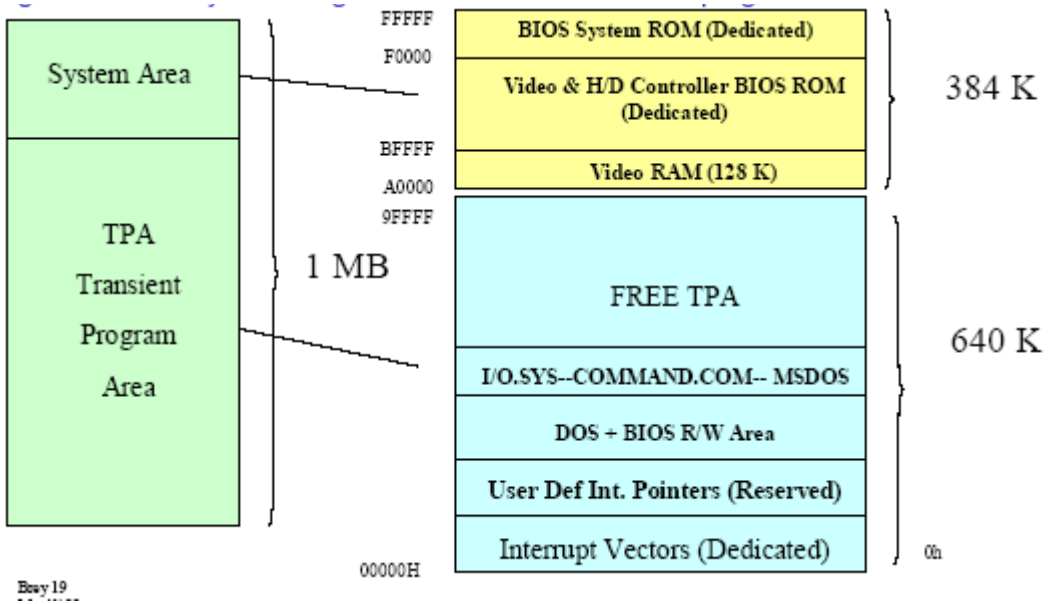
American Standard Code for Information Interchange (ASCII)

	b_7	0	0	0	0	1	1	1	1
	b_6	0	0	1	1	0	0	1	1
	b_5	0	1	0	1	0	1	0	1
b_4 b_3 b_2 b_1	H_1 / H_0	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	'	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	A	LF	SUB	*	:	J	Z	j	z
1 0 1 1	B	V	ESC	+	;	K	[	k	}
1 1 0 0	C	FF	FS	,	<	L	\	l	|
1 1 0 1	D	CR	GS	−	=	M	]	m	{
1 1 1 0	E	SO	RS	.	>	N	^	n	~
1 1 1 1	F	SI	US	/	?	O	-	o	DEL

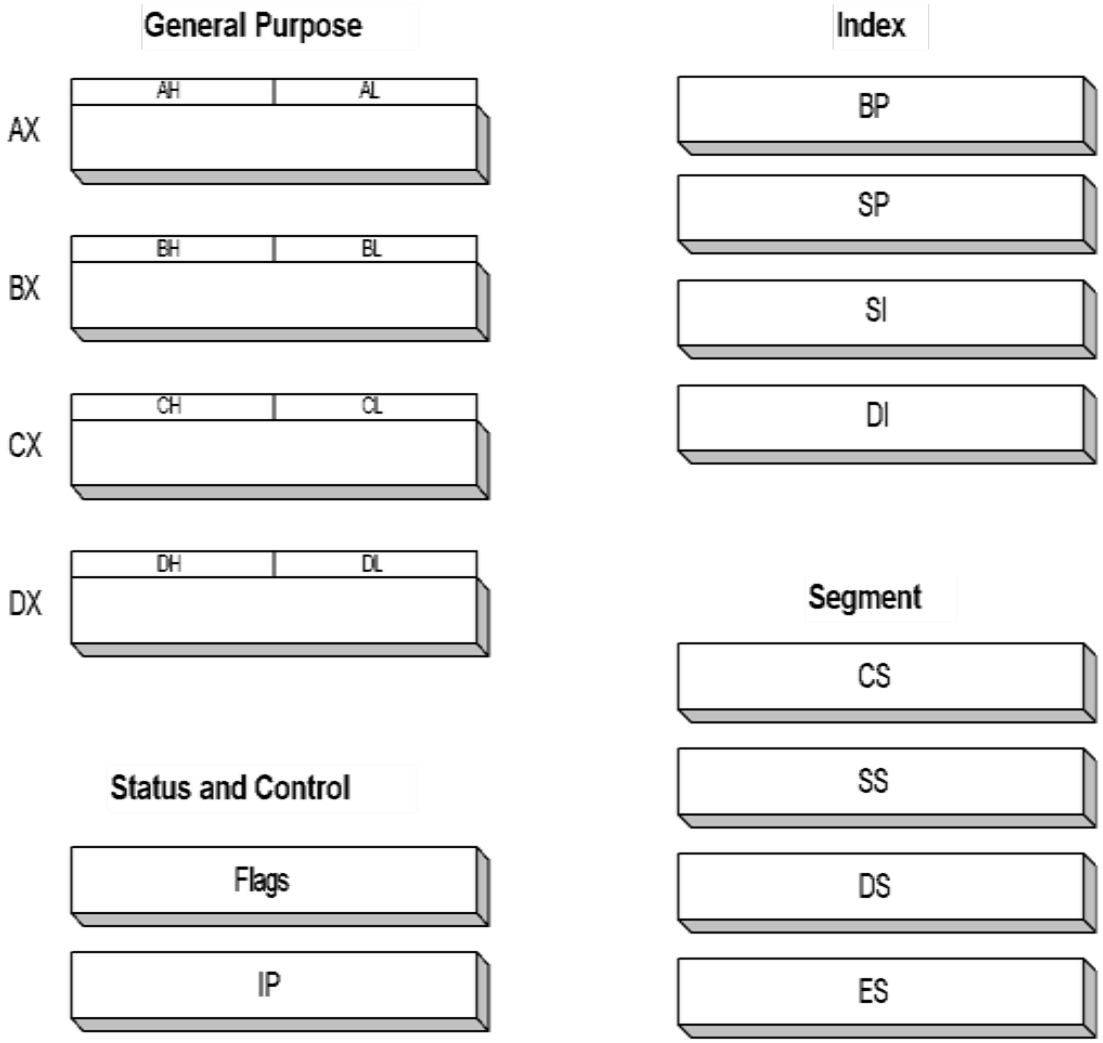
(a)

(b)

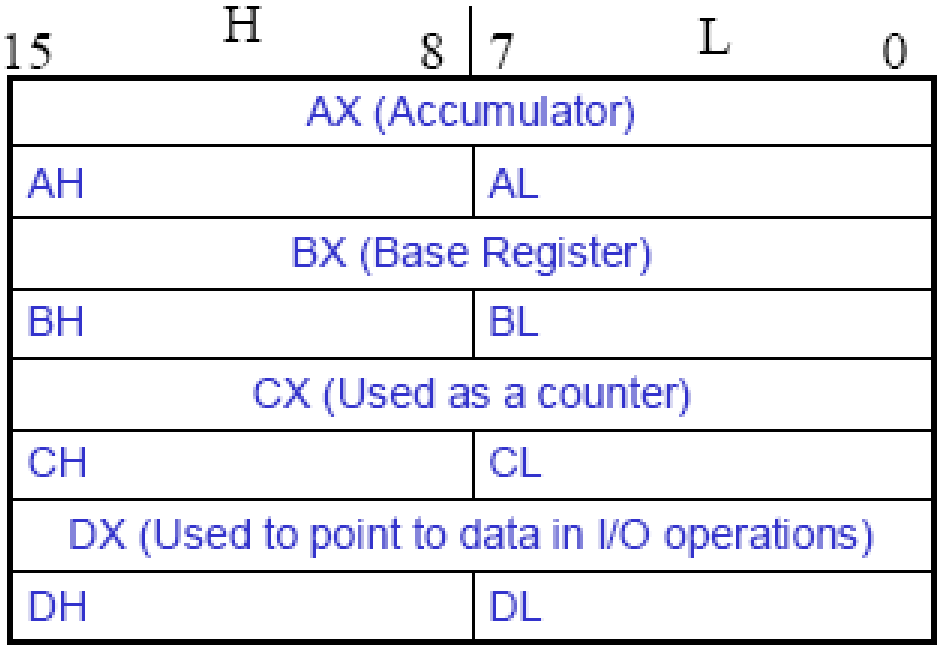
Dedicated, Reserved, and General use Memory



8086 Registers



General Purpose Registers



AX - the Accumulator
BX - the Base Register
CX - the Count Register
DX - the Data Register

- Normally used for storing temporary results
- Each of the registers is 16 bits wide **(AX, BX, CX, DX)**
- Can be accessed as either 16 or 8 bits AX, AH, AL

General Purpose Registers

- **AX**
  - Accumulator Register
  - Preferred register to use in arithmetic, logic and data transfer instructions because it generates the shortest Machine Language Code
  - Must be used in multiplication and division operations
  - Must also be used in I/O operations
- **BX**
  - Base Register
  - Also serves as an address register

General Purpose Registers

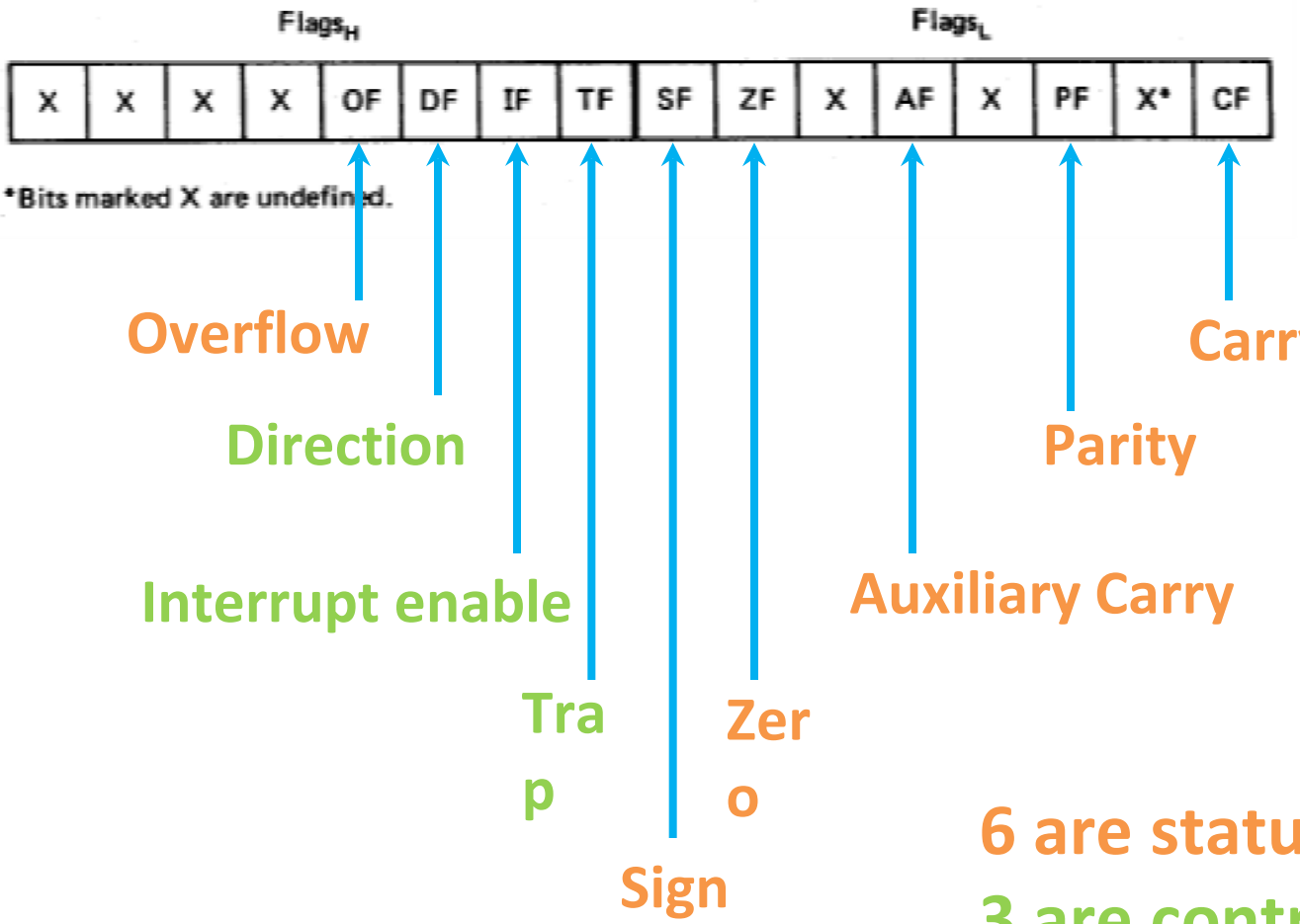
- **CX**
  - Count register
  - Used as a loop counter
  - Used in shift and rotate operations
- **DX**
  - Data register
  - Used in multiplication and division
  - Also used in I/O operations

Pointer and Index Registers

SP	Stack Pointer
BP	Base Pointer
SI	Source Index
DI	Destination Index
IP	Instruction Pointer

- All 16 bits wide, L/H bytes are not accessible
- Used as memory pointers
  - Example: MOV AH, [SI]
    - *Move the byte stored in memory location whose address is contained in register SI to register AH*
- IP is not under direct control of the programmer

Flag Register



8086 Programmer's Model

BIU registers

(20 bit adder)

ES	Extra Segment
CS	Code Segment
SS	Stack Segment
DS	Data Segment
IP	Instruction Pointer

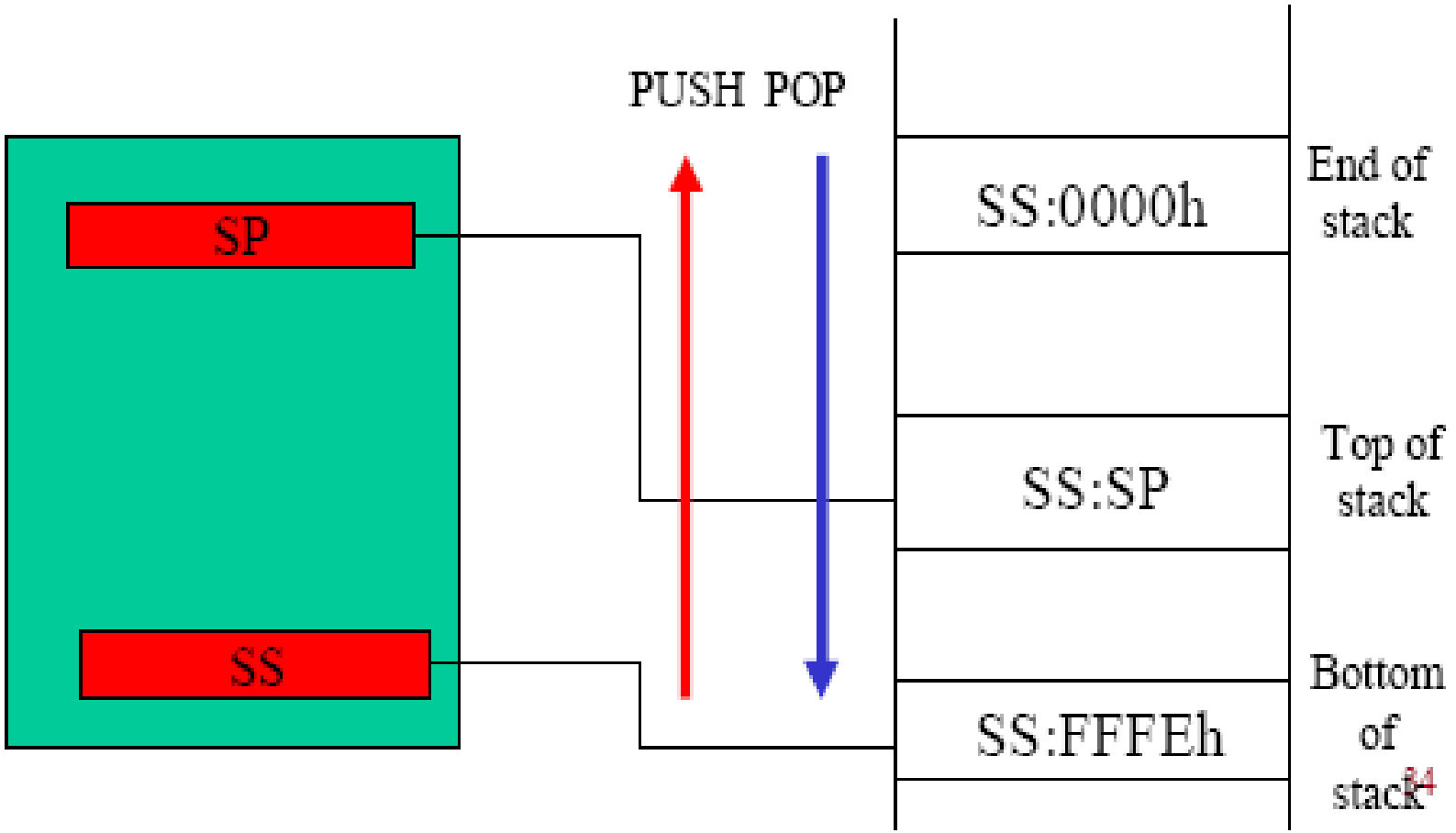
EU registers

AX	AH	AL	Accumulator
BX	BH	BL	Base Register
CX	CH	CL	Count Register
DX	DH	DL	Data Register
	SP		Stack Pointer
	BP		Base Pointer
	SI		Source Index Register
	DI		Destination Index Register
	FLAGS		

The Stack

- The stack is used for temporary storage of information such as data or addresses.
- When a **CALL** is executed, the 8086 automatically **PUSH**es the current value of CS and IP onto the stack.
- Other registers can also be pushed
- Before return from the **subroutine**, **POP** instructions can be used to pop values back from the stack into the corresponding registers.

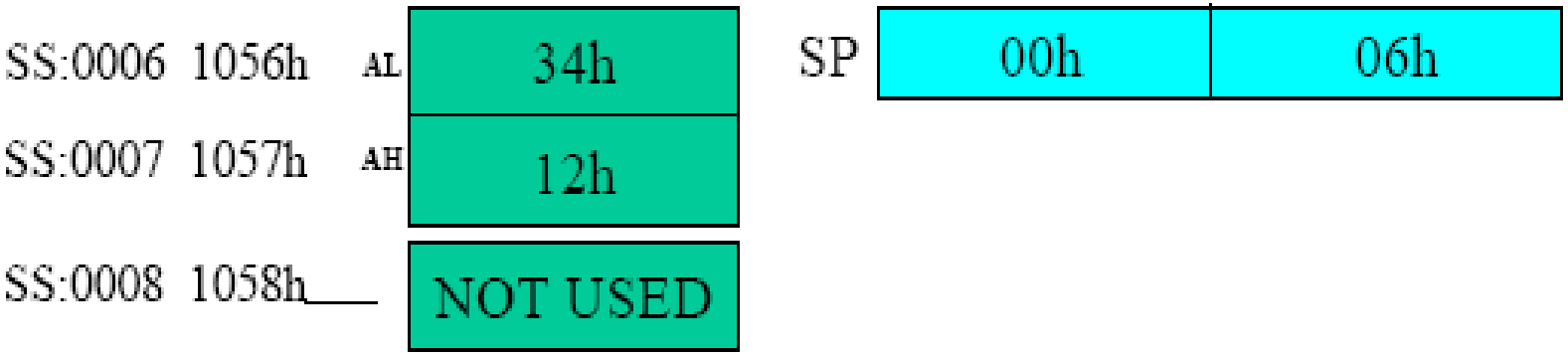
The Stack


PUSH POP
SP
SS
SS:0000h
SS:SP
SS:FFFEh
End of stack
Top of stack
Bottom of stack

Example for PUSH

- Given
  - SS = 0105h
  - SP = 0008h
  - AX = 1234h
  - What is the outcome of the PUSH AX instruction?
- A_{BOS} = 01050 + FFFEh = 1104h
- A_{TOS} = 01050 + 0008h = 1058h

- Decrement the SP by 2 and write AX into the word location 1056h.



Example for POP

- What is the outcome of the following
  - POP AX
  - POP BX
  - if originally 1058h contained AABBh?

1056	34
1057	12
1058	BB
1059	AA

- Read into the specified register from the stack and increment the stack pointer for each POP operation
- At the first POP
  - AX = 1234h SP = 0008h
- At the second POP
  - BX = AABBh

Question? What is SP and Physical address now? →

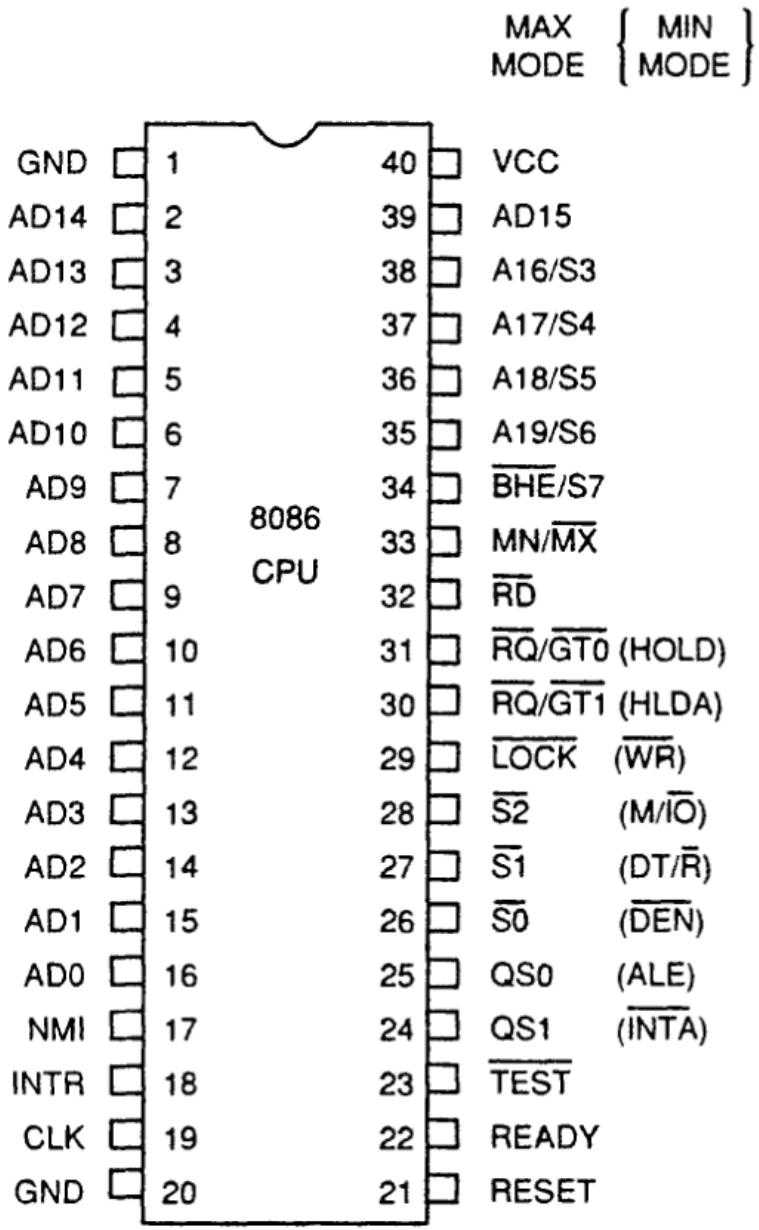
SP = 000Ah and
Physical Address: 105A

Hardware Architecture of INTEL 8086

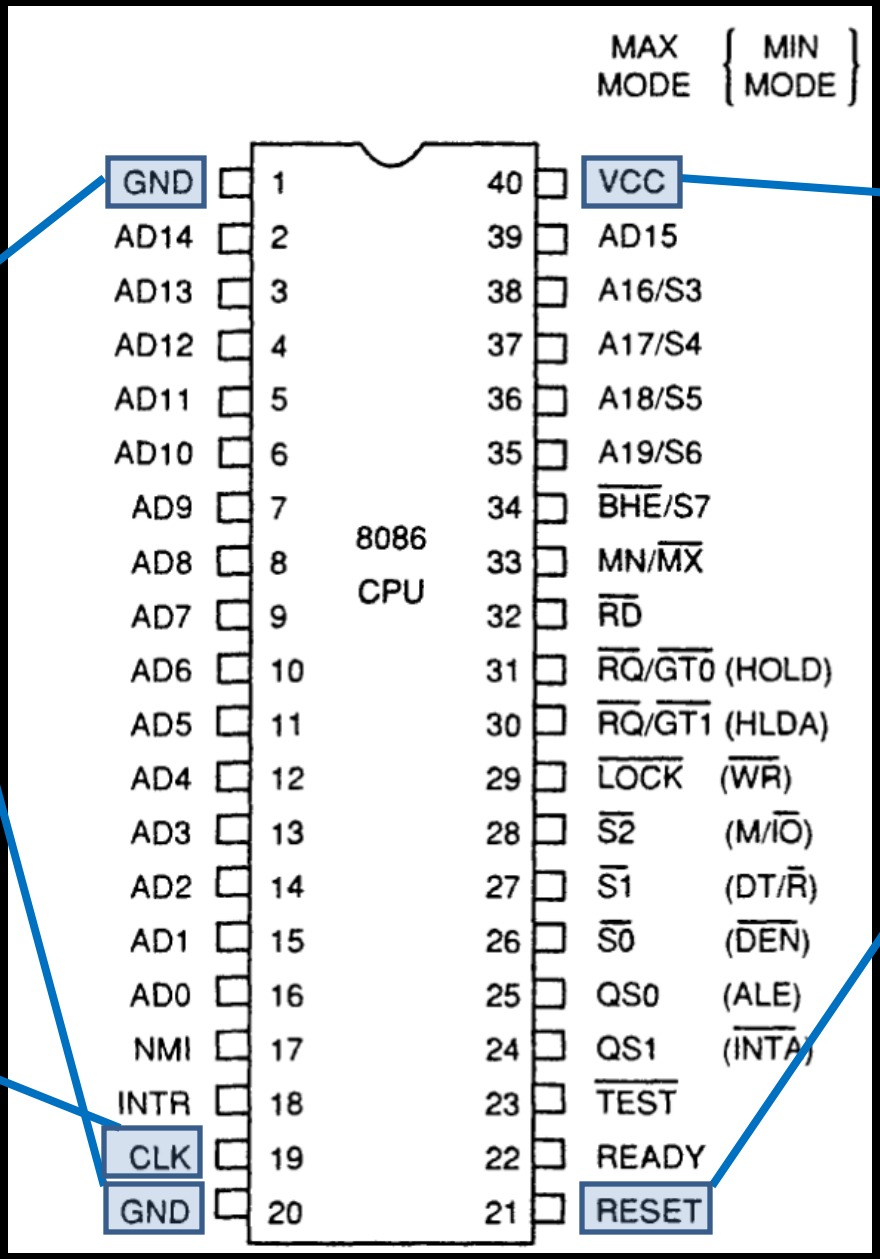
Hardware Architecture of INTEL 8086

- *Pin Diagram and Pin Details*
- *min/max mode*
- *Hardware organization of address space*
- *Control signals*
- *Coprocessor and Multiprocessor configuration*
- *I/O interfaces*

INTEL 8086 - Pin Diagram



INTEL 8086 - Pin Details



Ground

Power Supply

5V ± 10%

Clock

Duty cycle: 33%

Reset

Registers, seg regs, flags

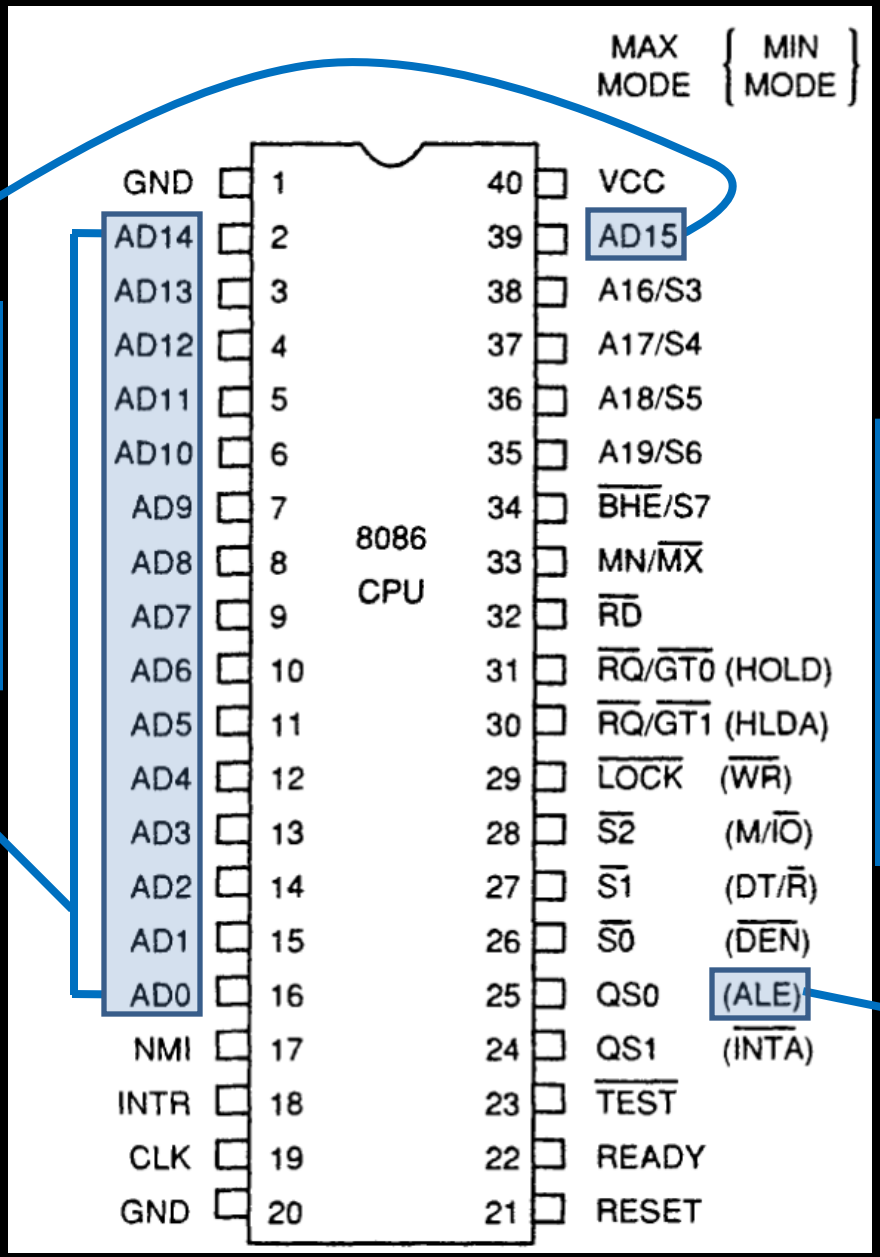
CS: FFFFH, IP: 0000H

If high for minimum 4 clks

INTEL 8086 - Pin Details

Address/Data Bus:

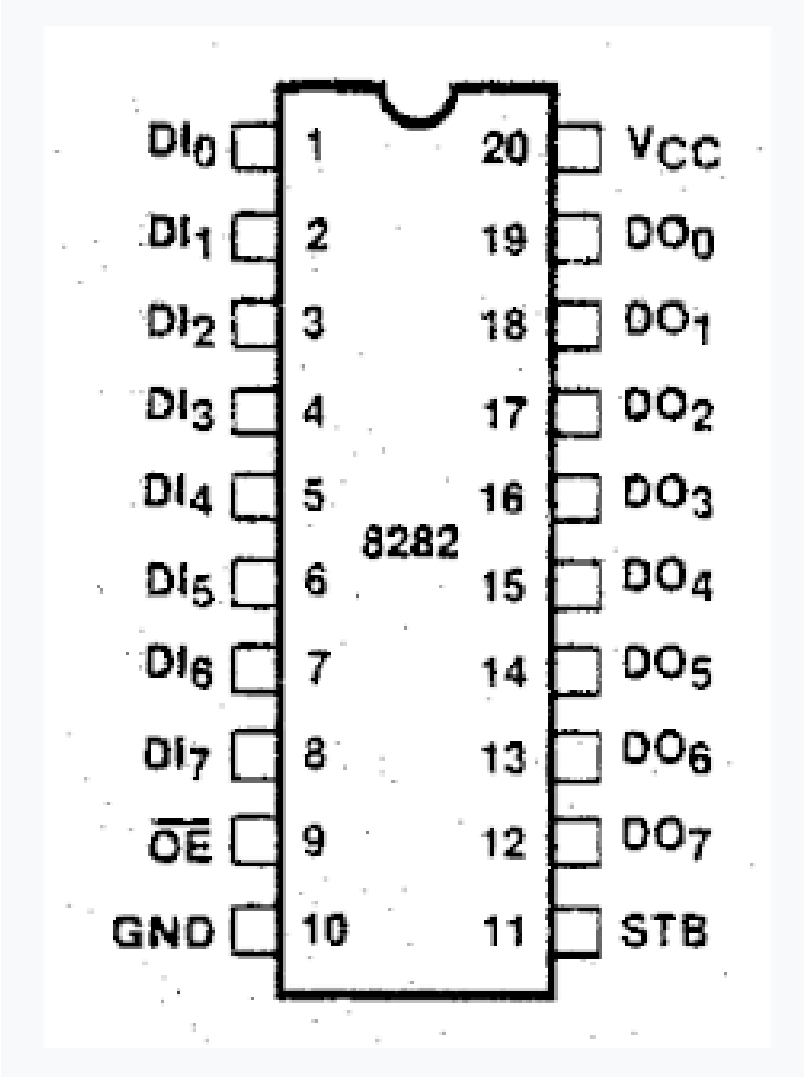
Contains address bits A_{15}-A_0 when ALE is 1 & data bits D_{15} – D_0 when ALE is 0.



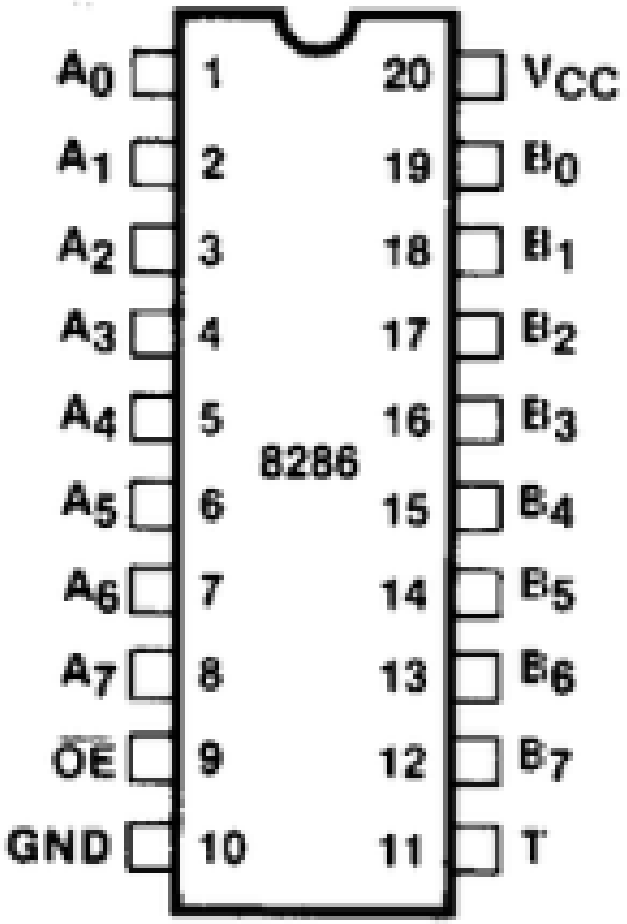
Address Latch Enable:

When high, multiplexed address/data bus contains address information.

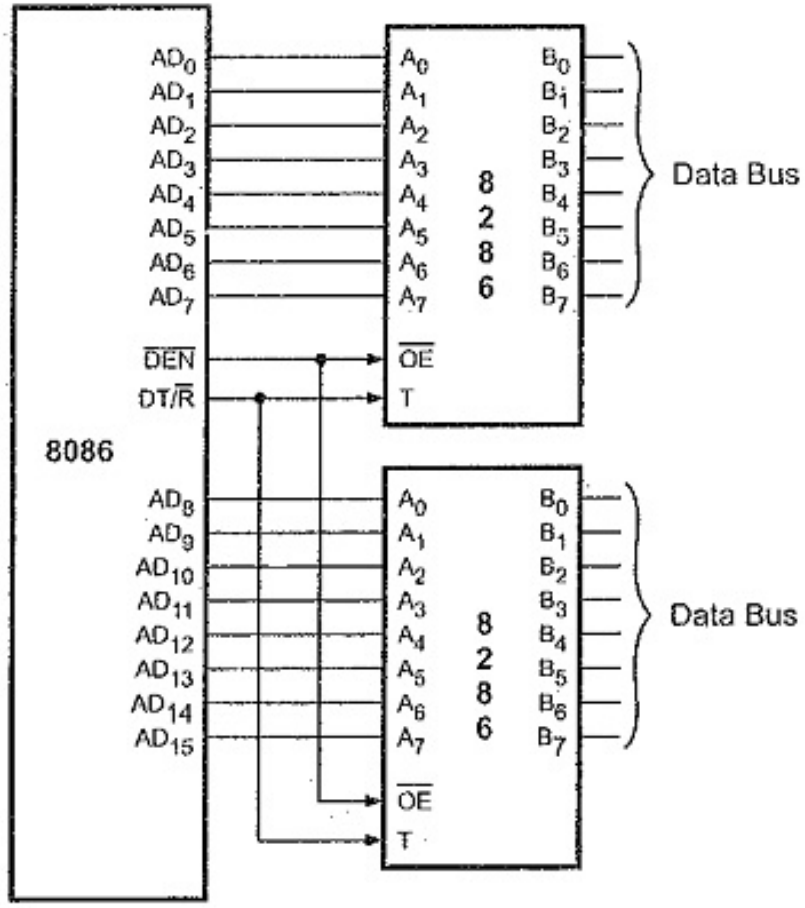
8282 - Octal Latch



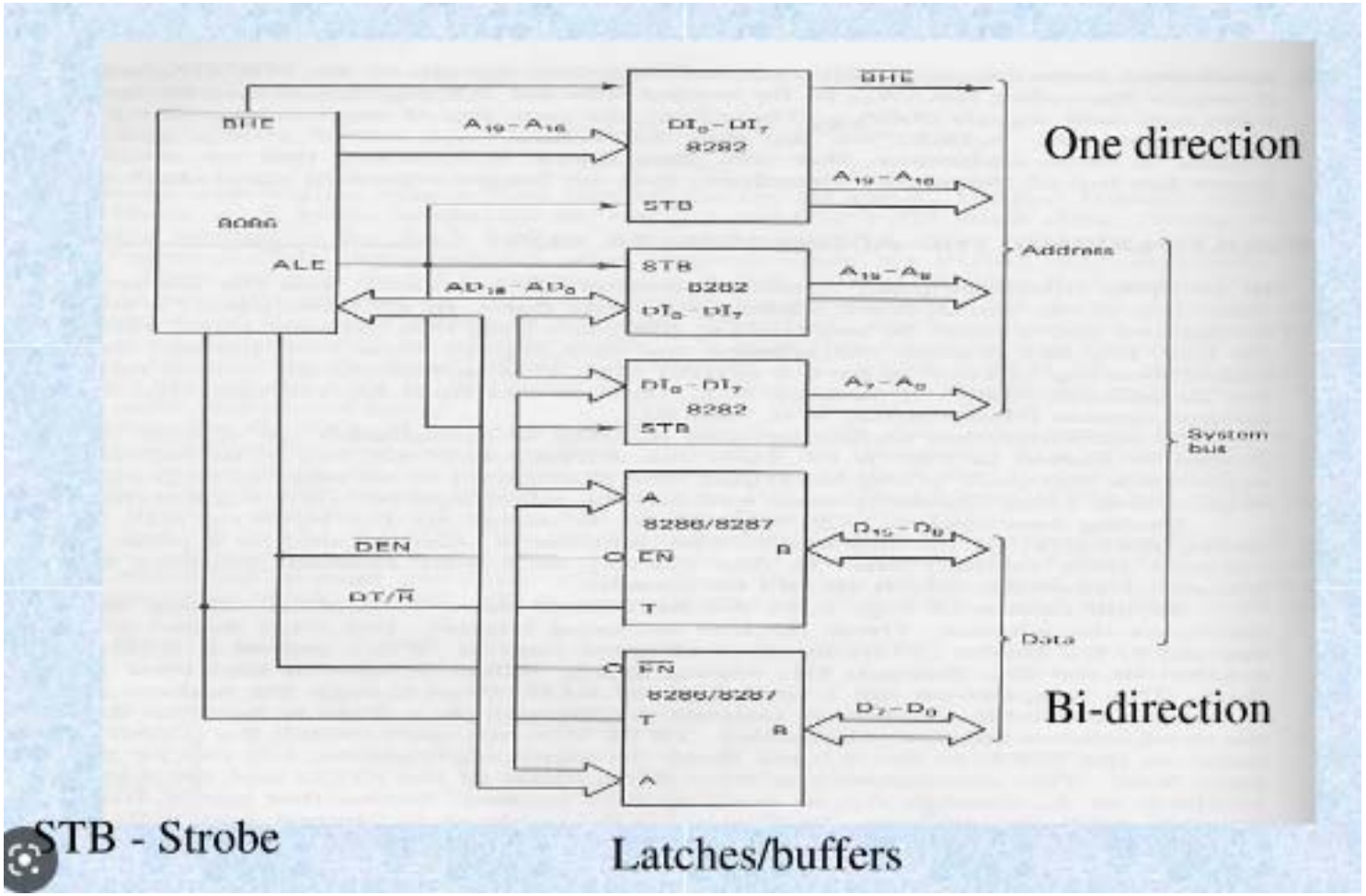
8286 - Octal Transceiver



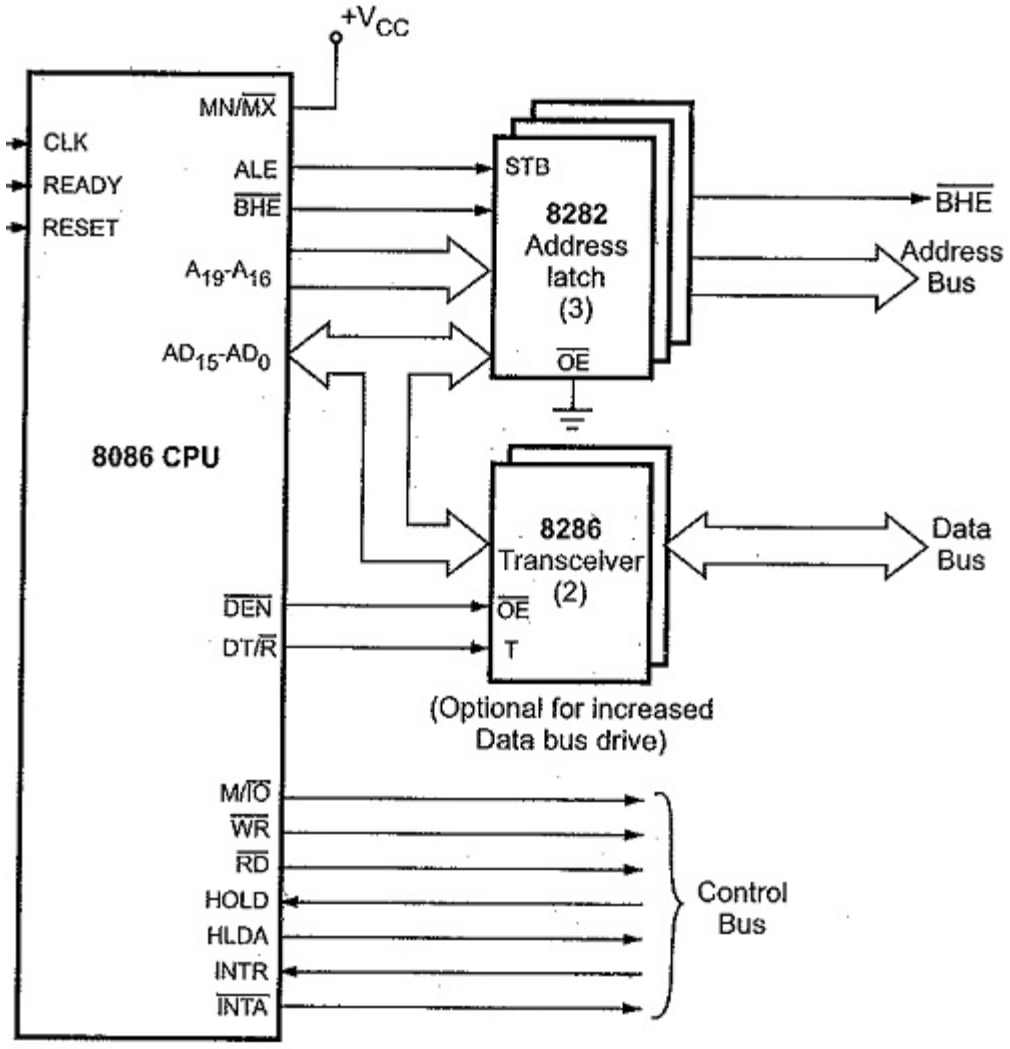
8286 - Octal Transceiver



Demultiplexing of Address and Data Bus in 8086

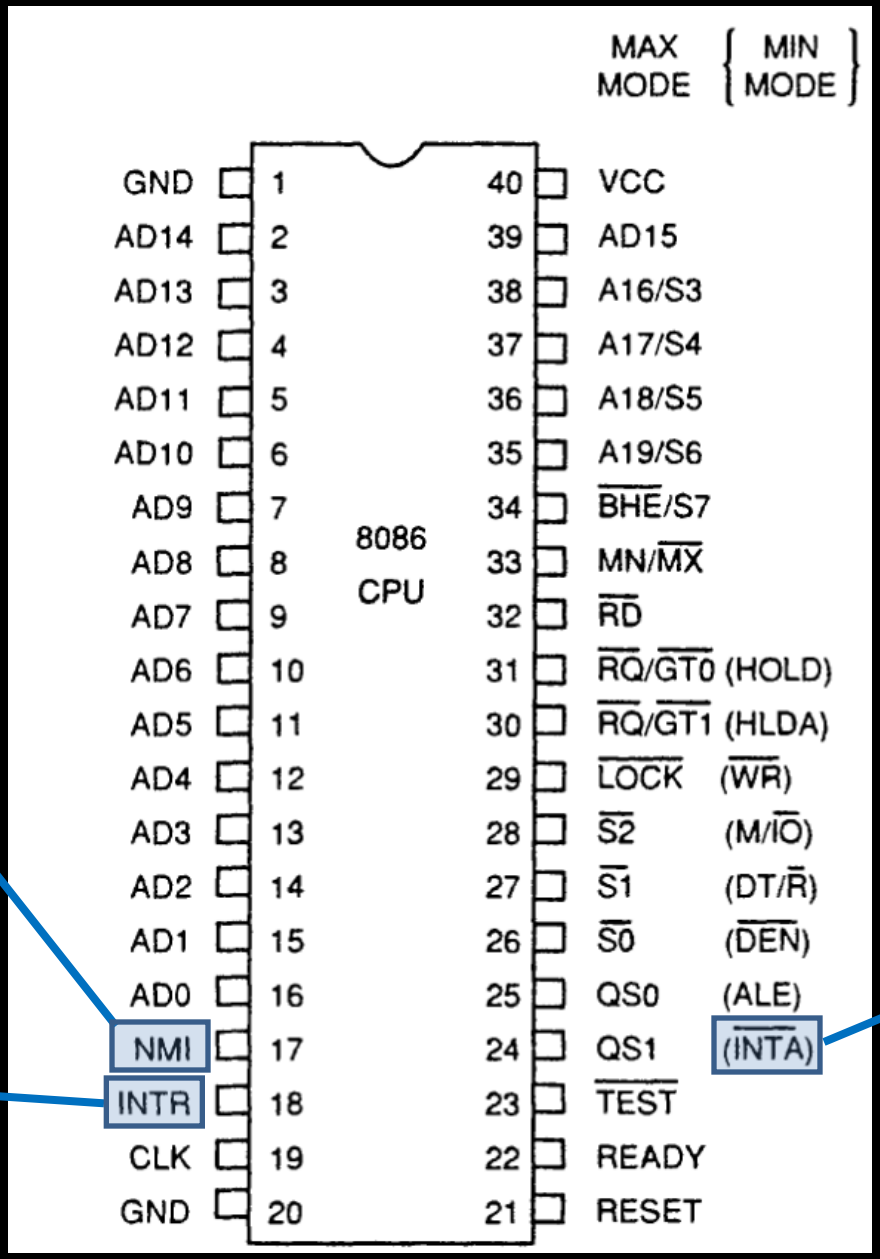


Demultiplexing of Address and Data Bus in 8086



INTEL 8086 - Pin Details

INTERRUPT

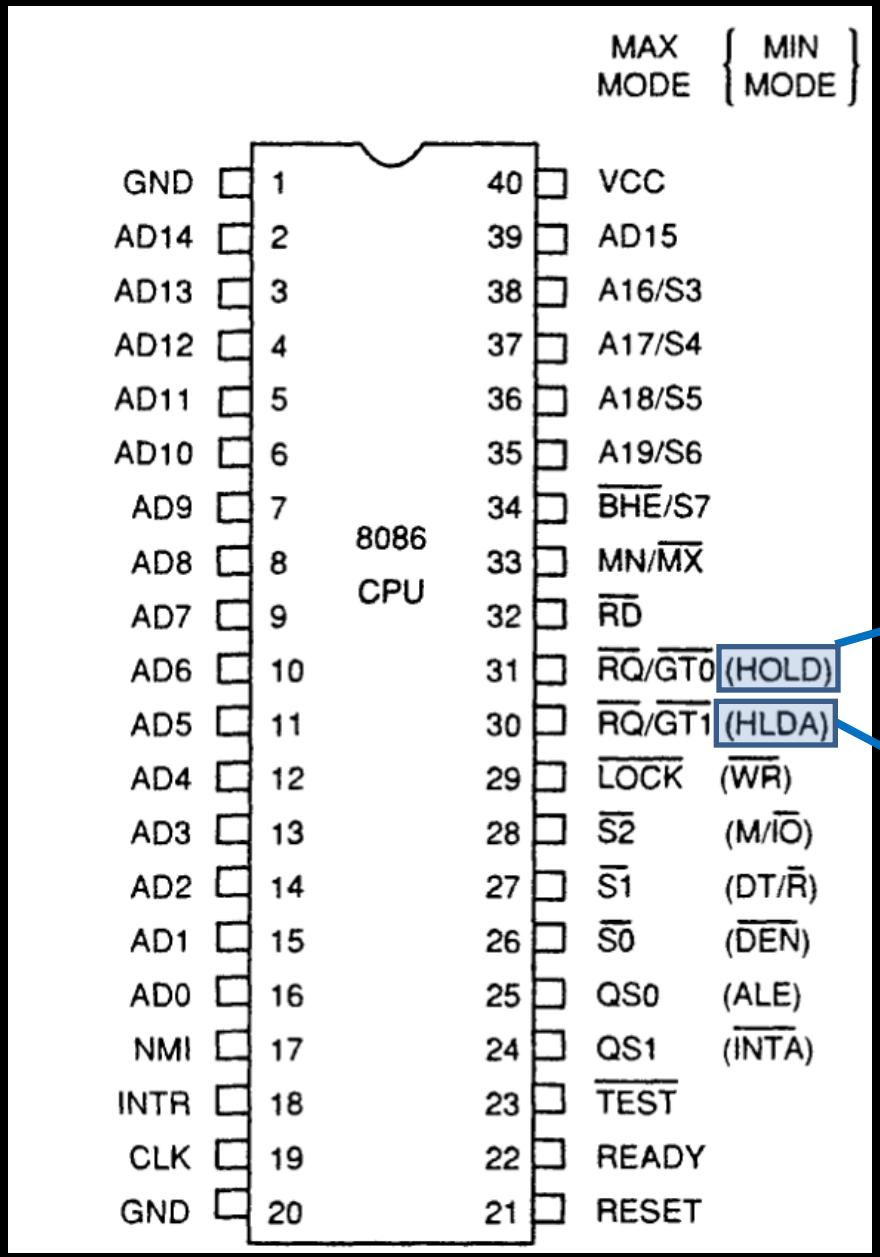


Non - maskable interrupt

Interrupt request

Interrupt acknowledge

INTEL 8086 - Pin Details



Direct Memory Access

Hold

Hold acknowledge

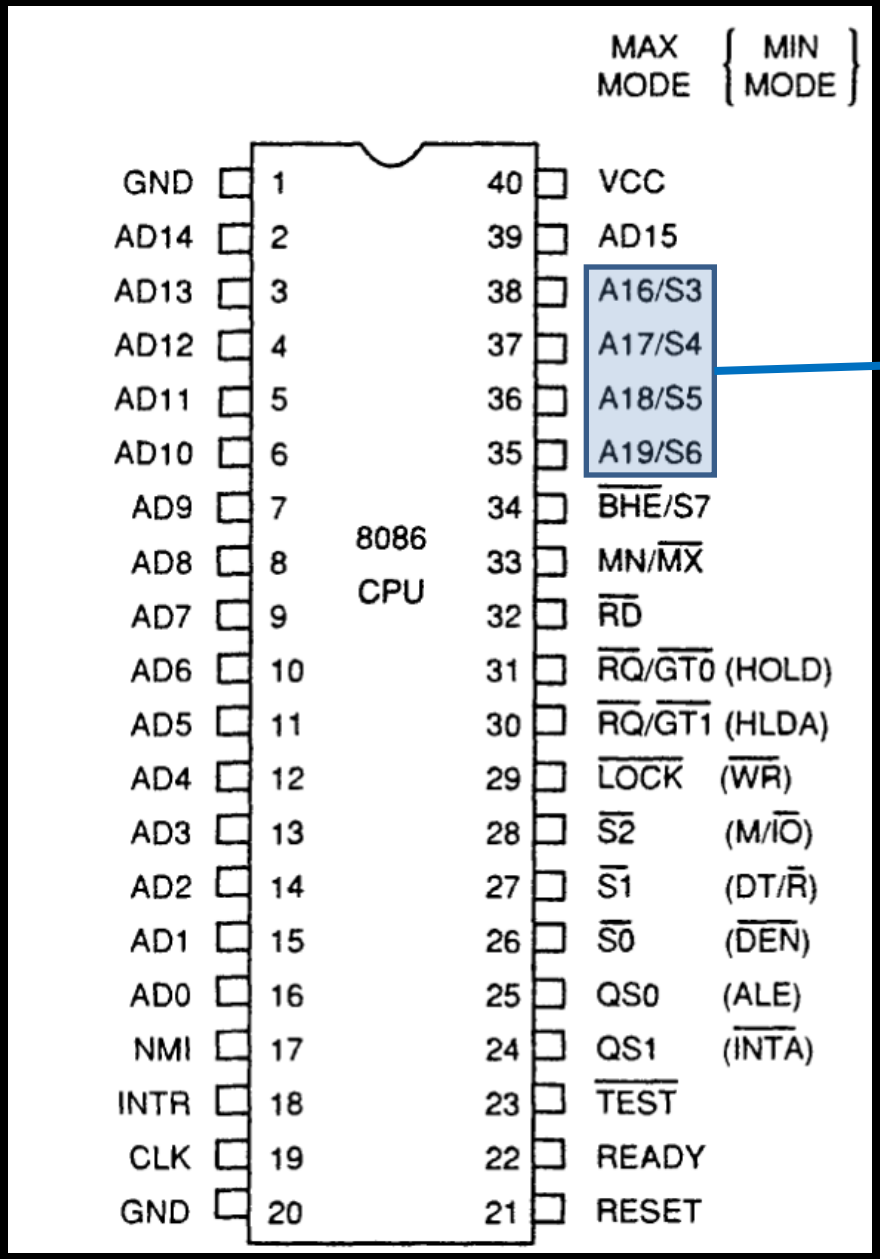
INTEL 8086 - Pin Details

S6: Logic 0.

S5: Indicates condition of IF flag bits.

S4-S3: Indicate which segment is accessed during current bus cycle:

S4	S3	*Function*
0	0	Extra segment
0	1	Stack segment
1	0	Code or no segment
1	1	Data segment



Address/Status Bus

Address bits A_{19} – A_{16} & Status bits S_6 – S_3

INTEL 8086 - Pin Details

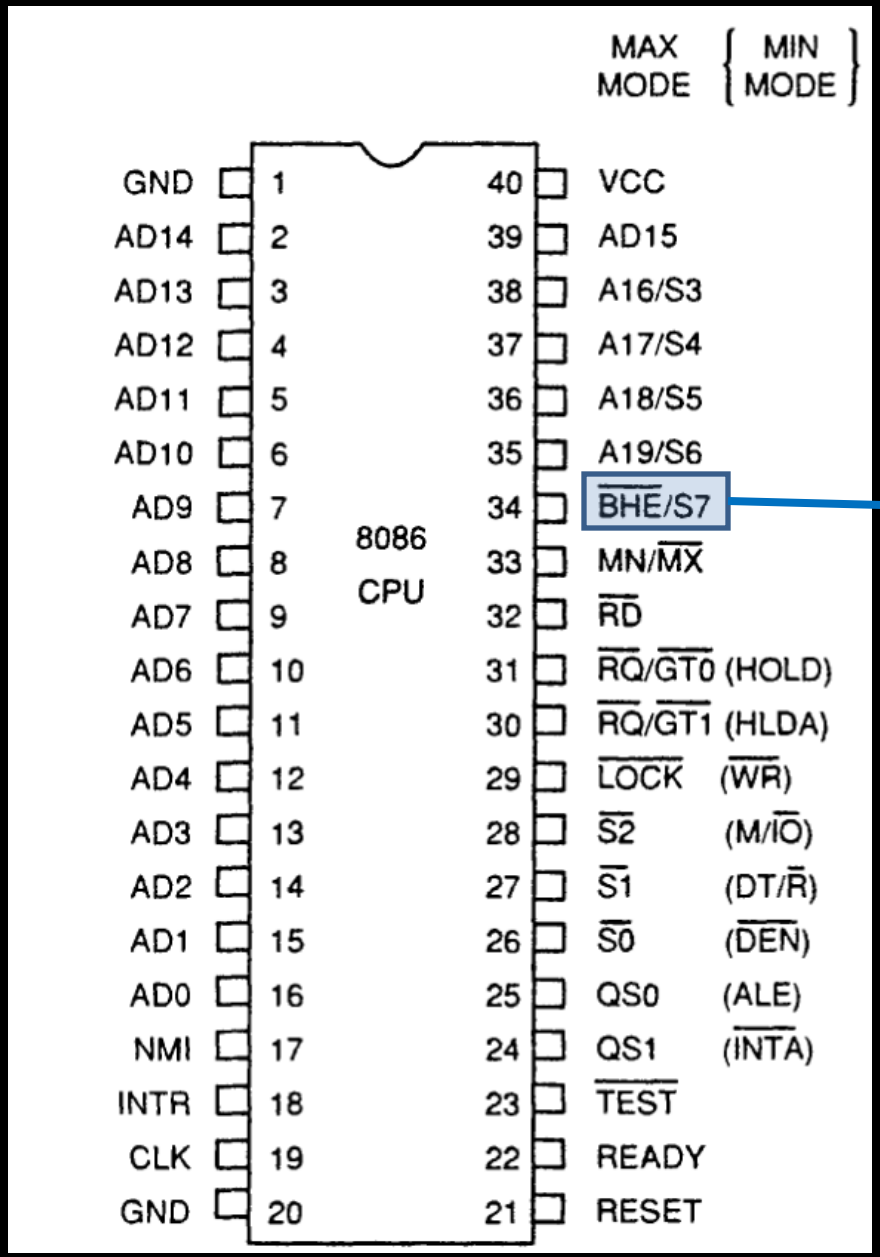
BHE#, A_0:

0,0: Whole word (16-bits)

0,1: High byte to/from odd address

1,0: Low byte to/from even address

1,1: No selection

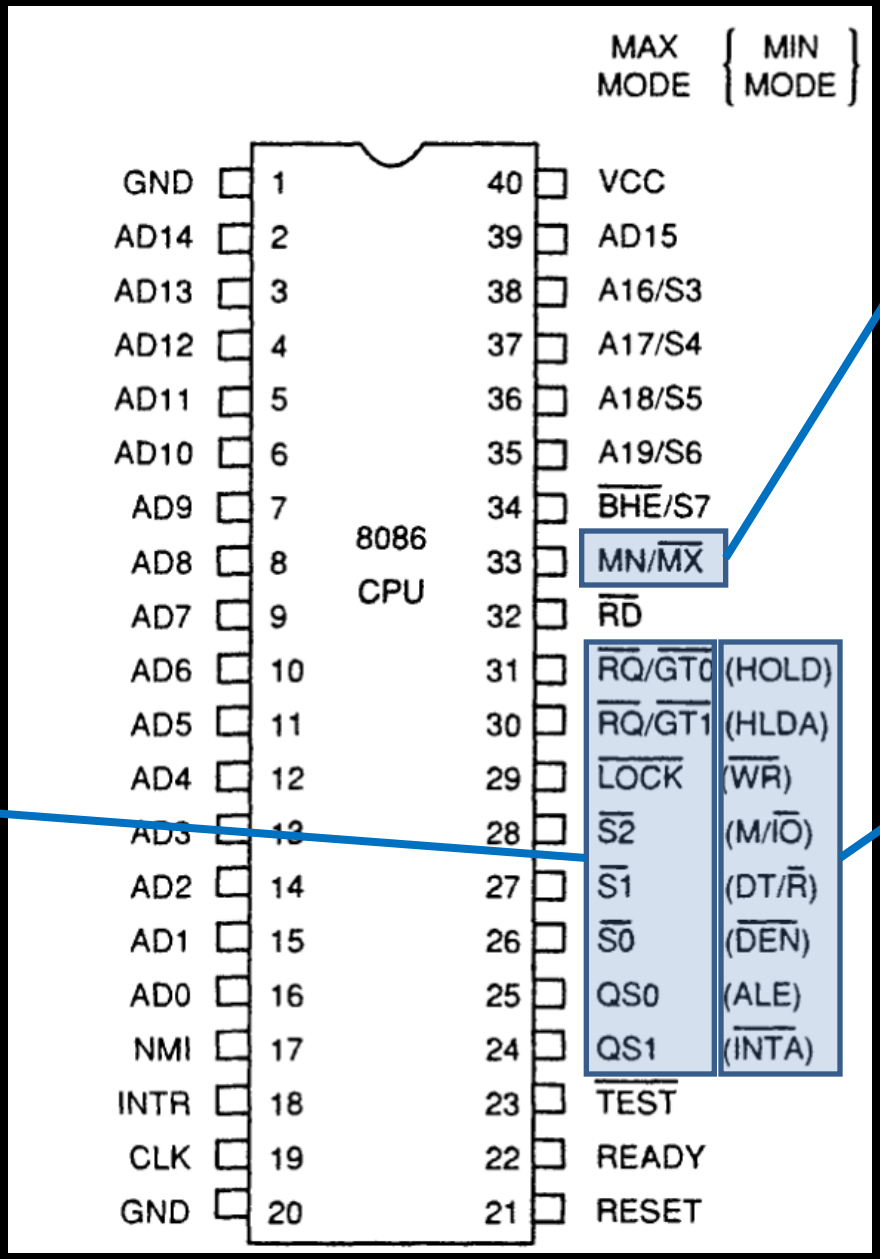


Bus High Enable/S7

Enables most significant data bits $D_{15} – D_8$ during read or write operation.

S_7: Always 1.

INTEL 8086 - Pin Details



Min/Max mode

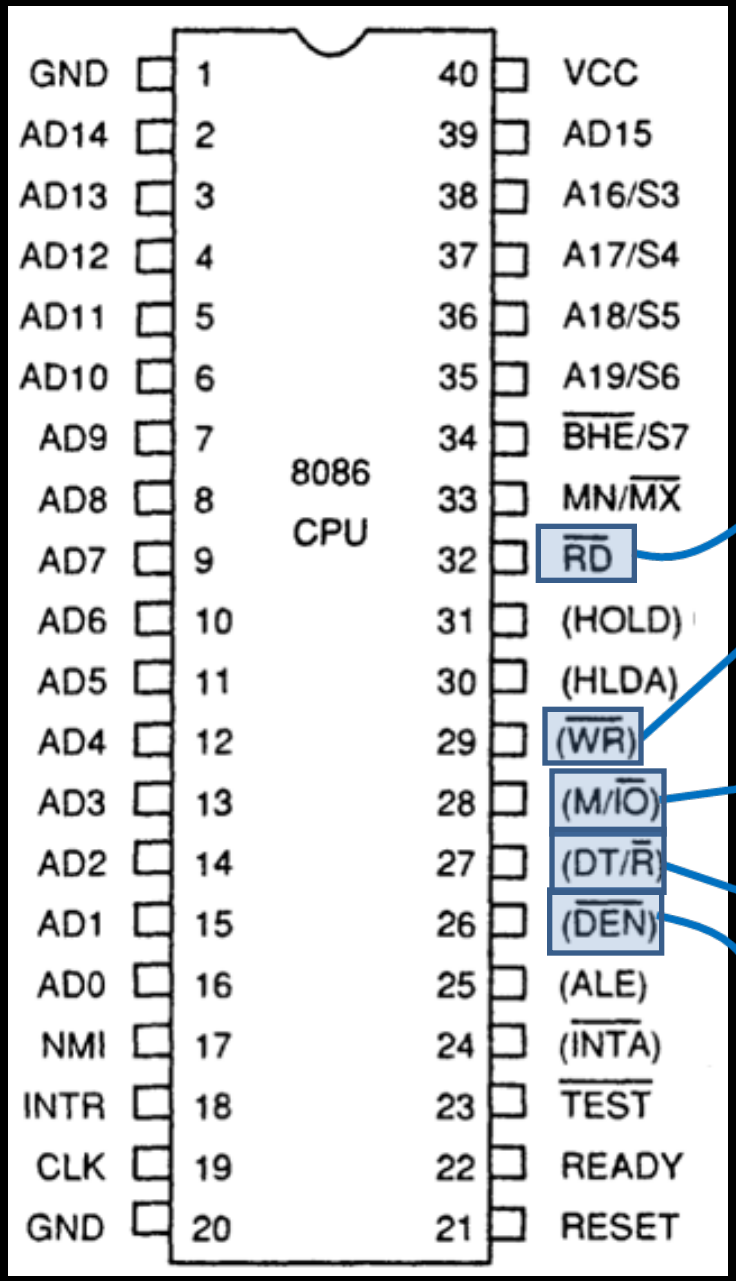
Minimum Mode: +5V

Maximum Mode: 0V

Minimum Mode Pins

Maximum Mode Pins

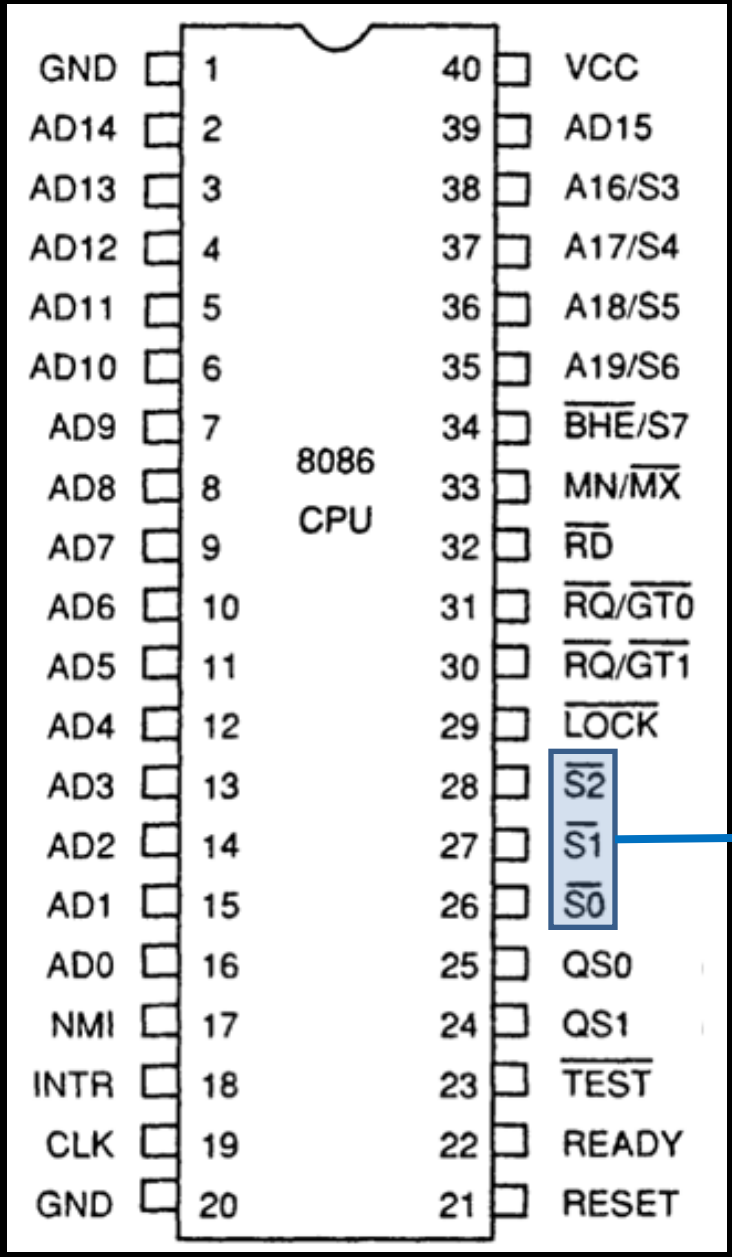
Minimum Mode- Pin Details



Maximum Mode - Pin Details

S2 S1 S0

000: INTA
001: read I/O port
010: write I/O port
011: halt
100: code access
101: read memory
110: write memory
111: none -passive



Status Signal

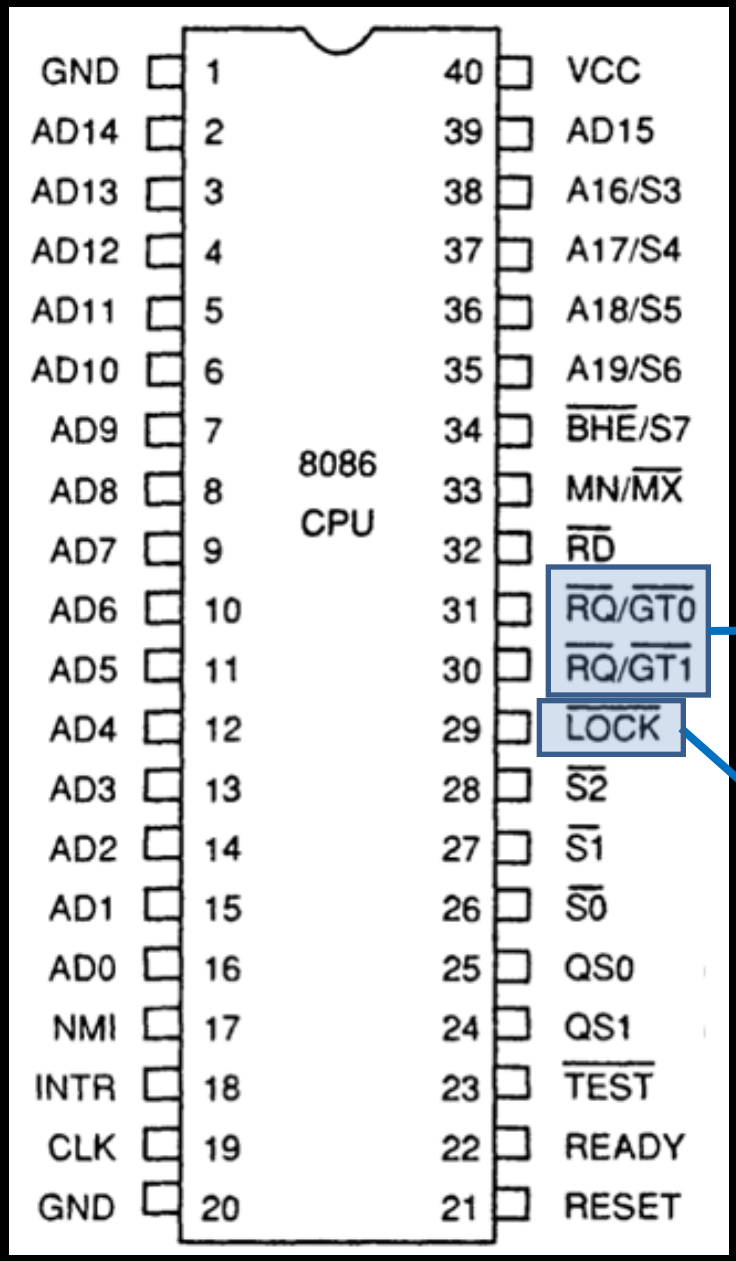
Inputs to 8288 to generate eliminated signals due to max mode.

Maximum Mode - Pin Details

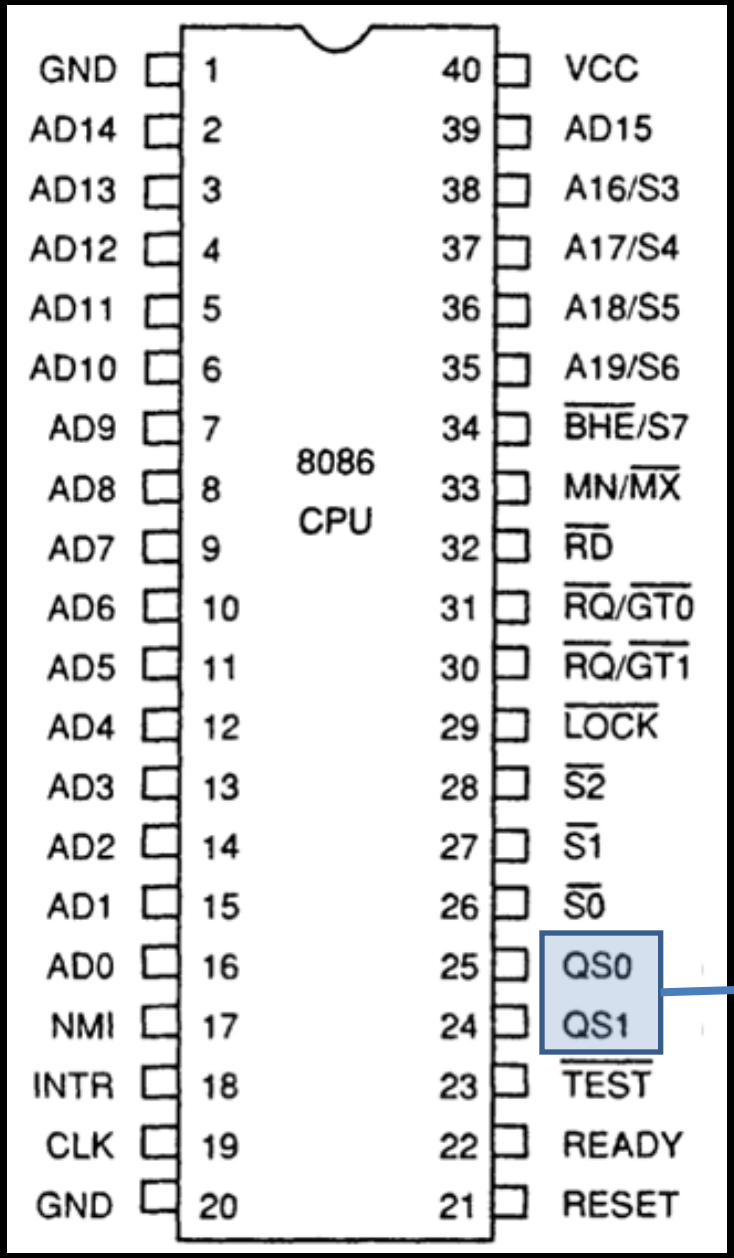
Lock Output

Used to lock peripherals off the system

Activated by using the LOCK: prefix on any instruction



Maximum Mode - Pin Details



QS1 QS0

00: Queue is idle

01: First byte of opcode

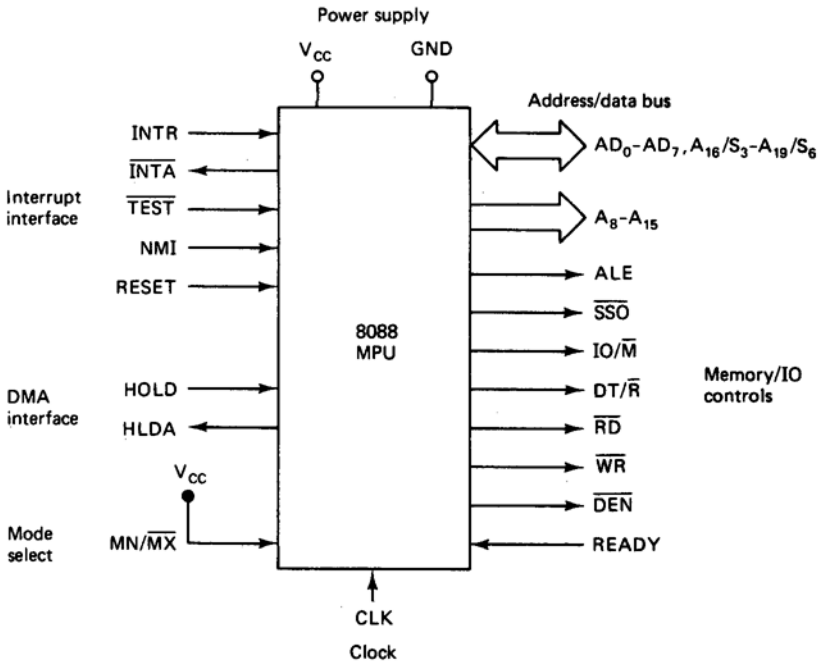
10: Queue is empty

11: Subsequent byte of opcode

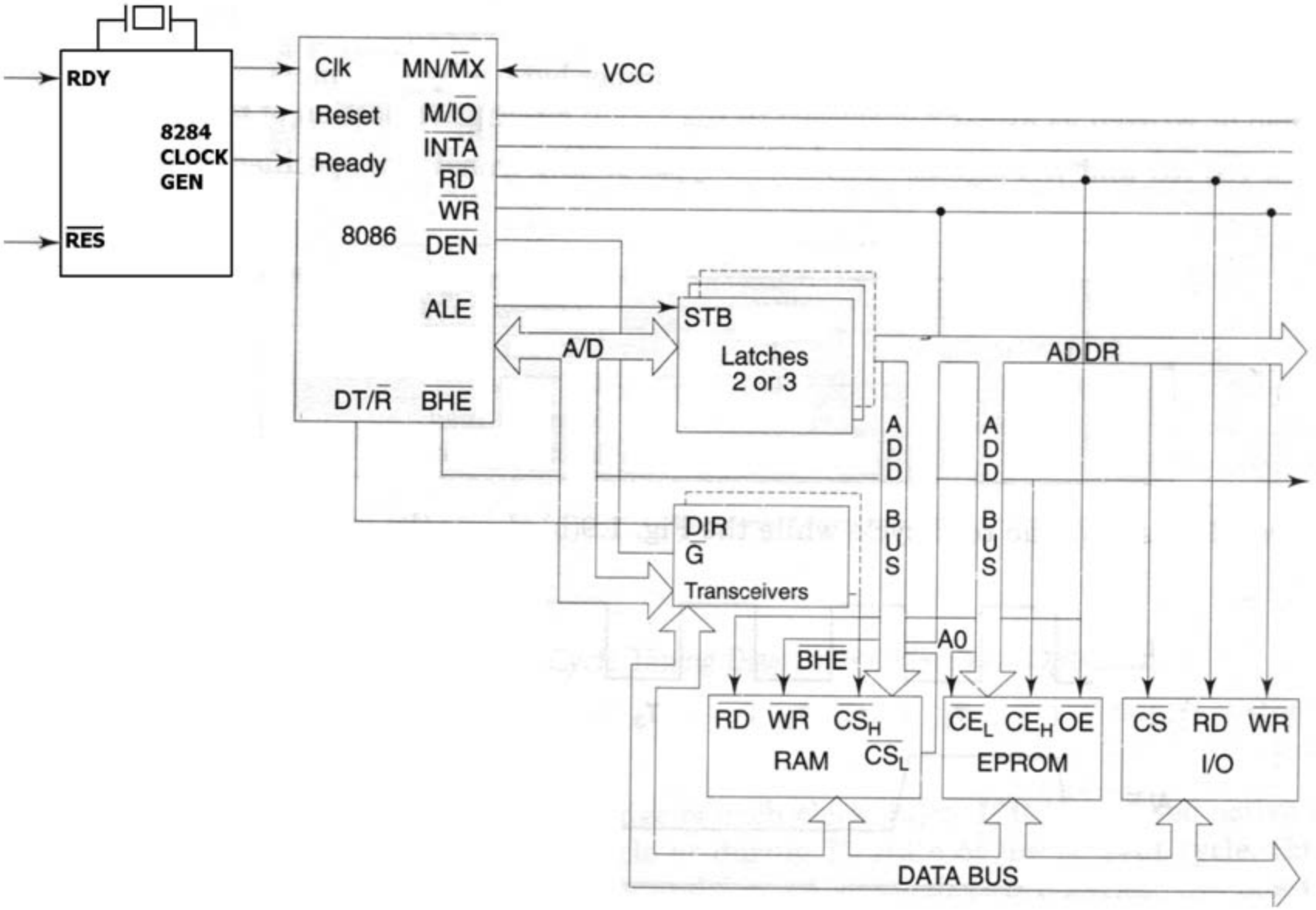
Queue Status

Used by numeric coprocessor (8087)

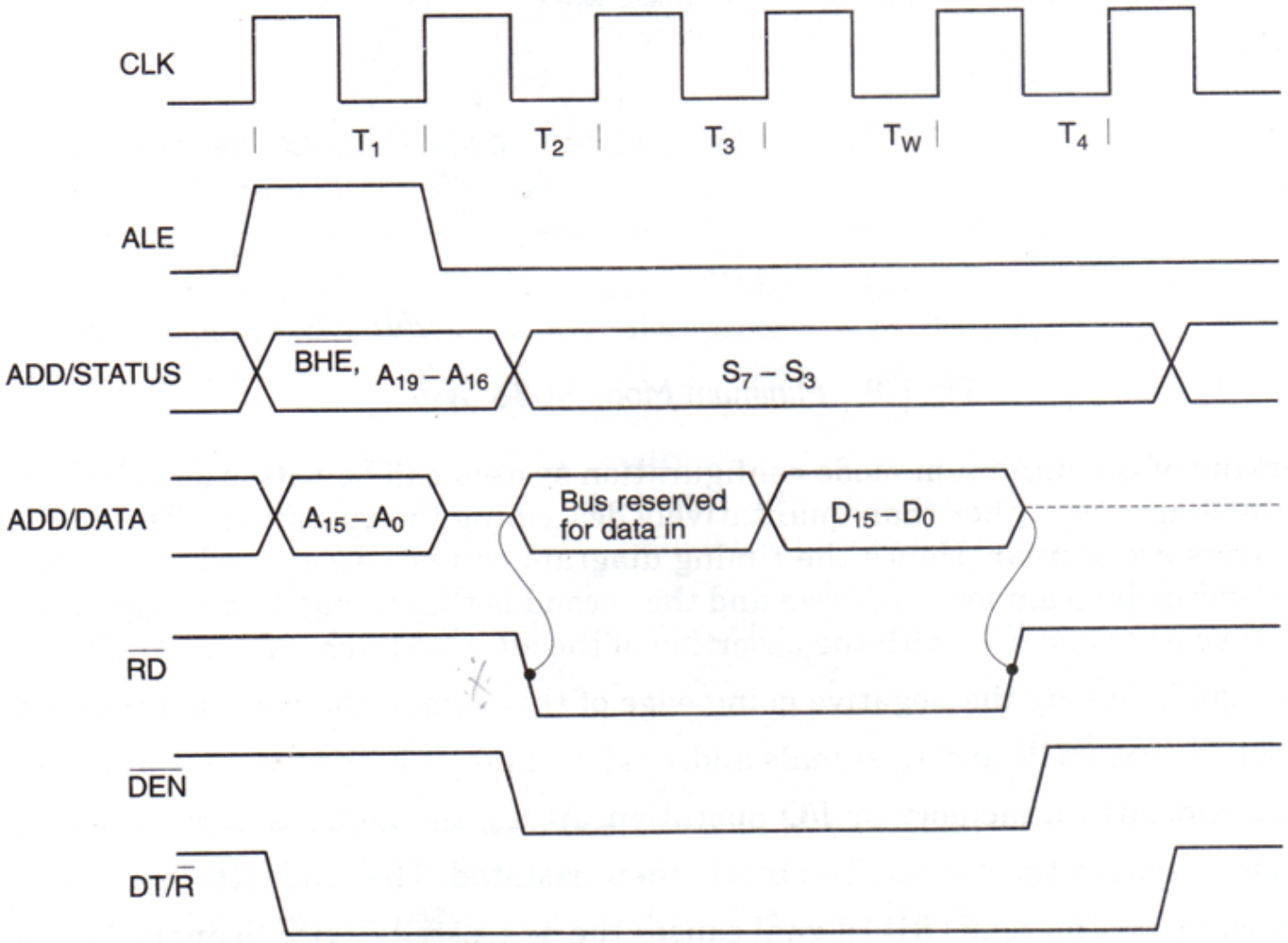
Minimum Mode 8086 System



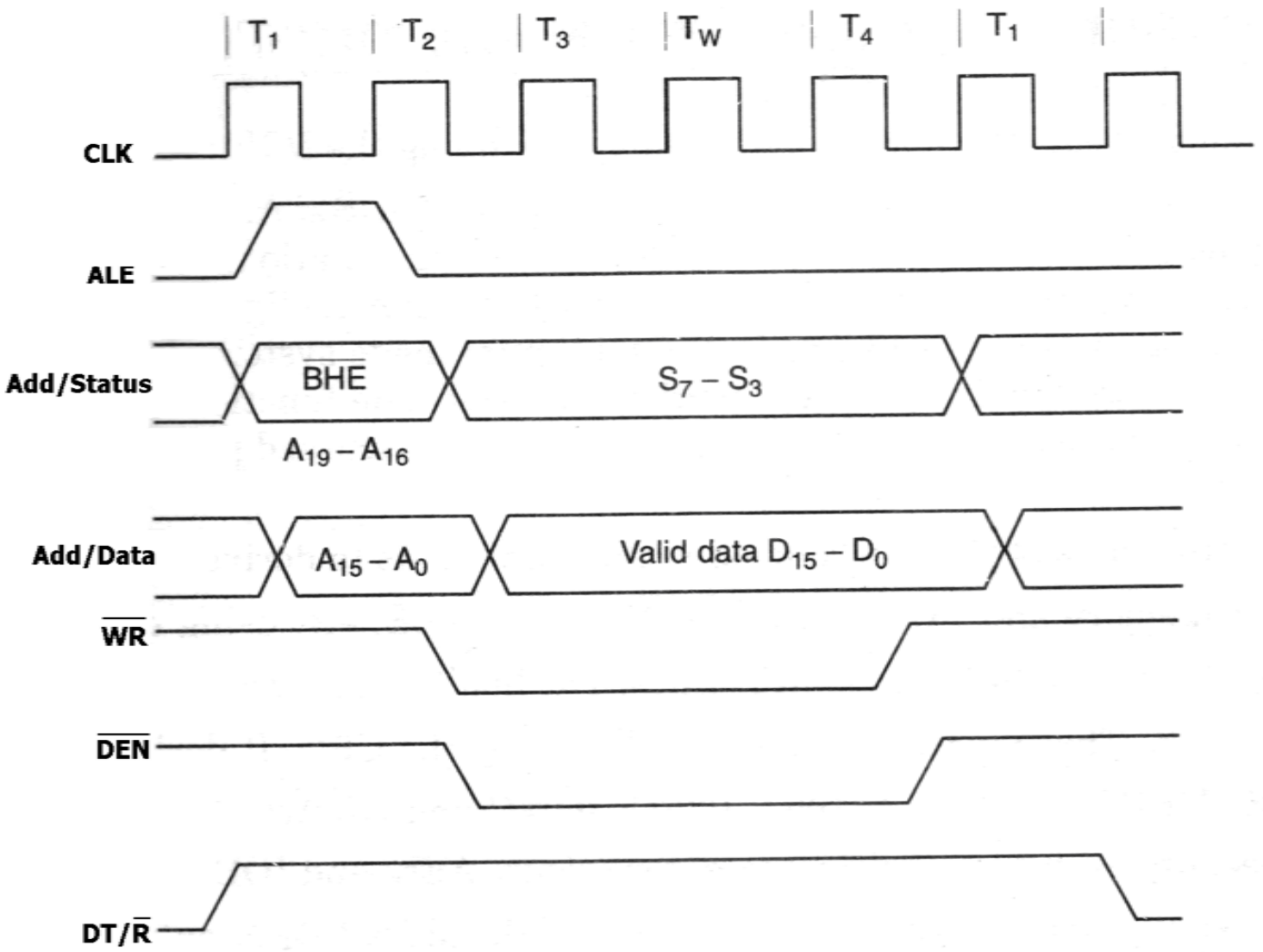
Minimum Mode 8086 System



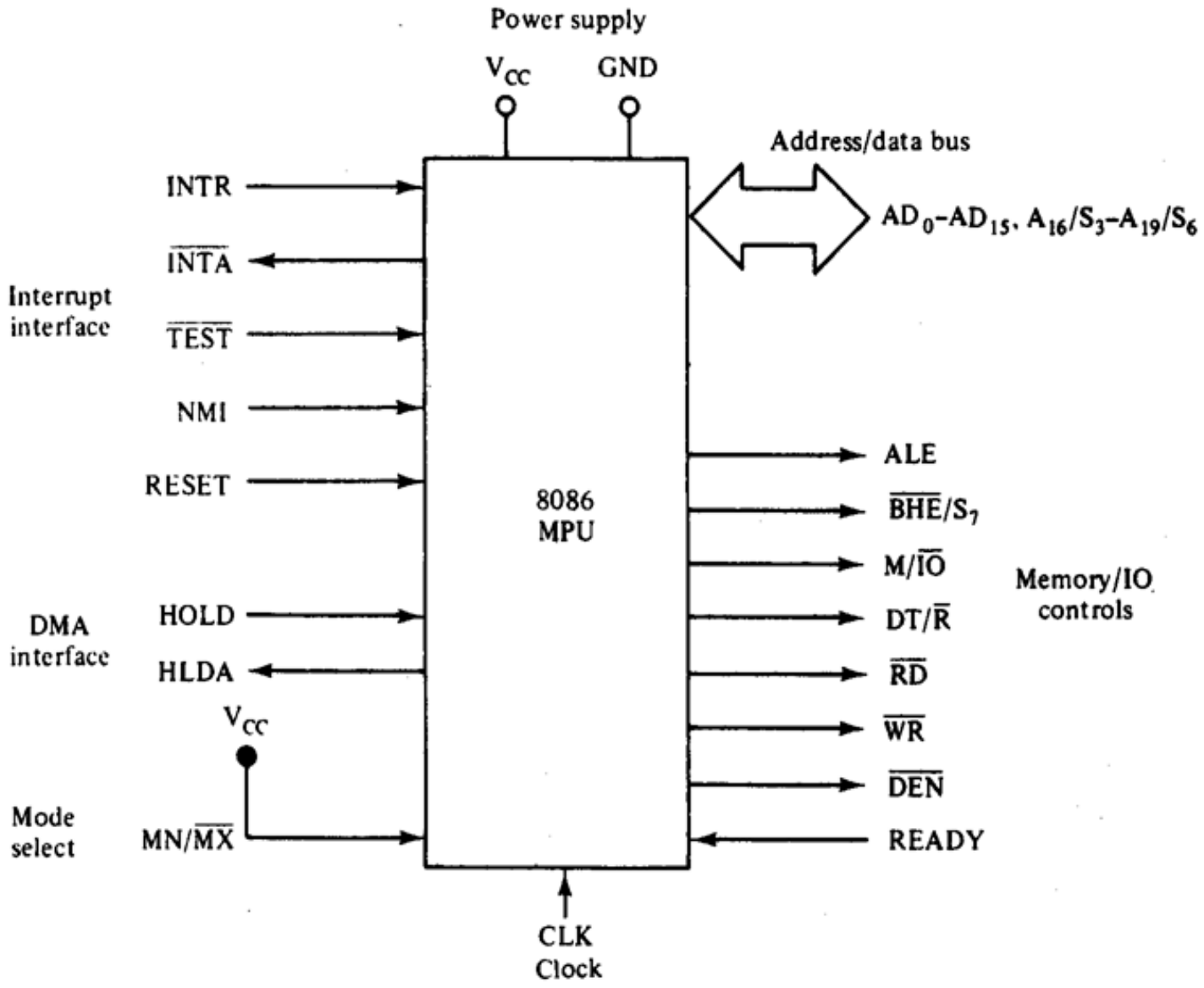
‘Read’ Cycle timing Diagram for Minimum Mode



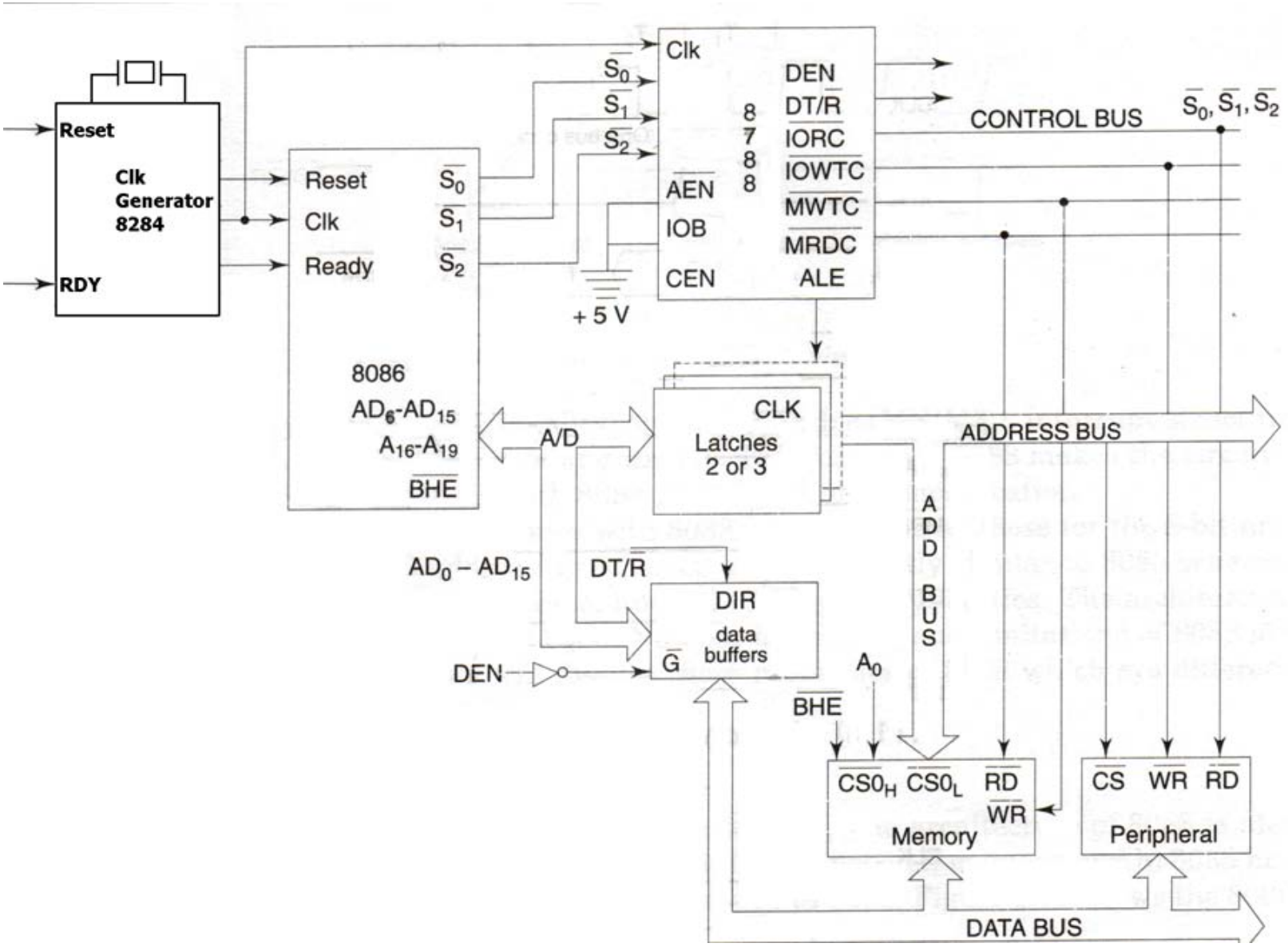
‘Write’ Cycle timing Diagram for Minimum Mode



Maximum Mode 8086 System



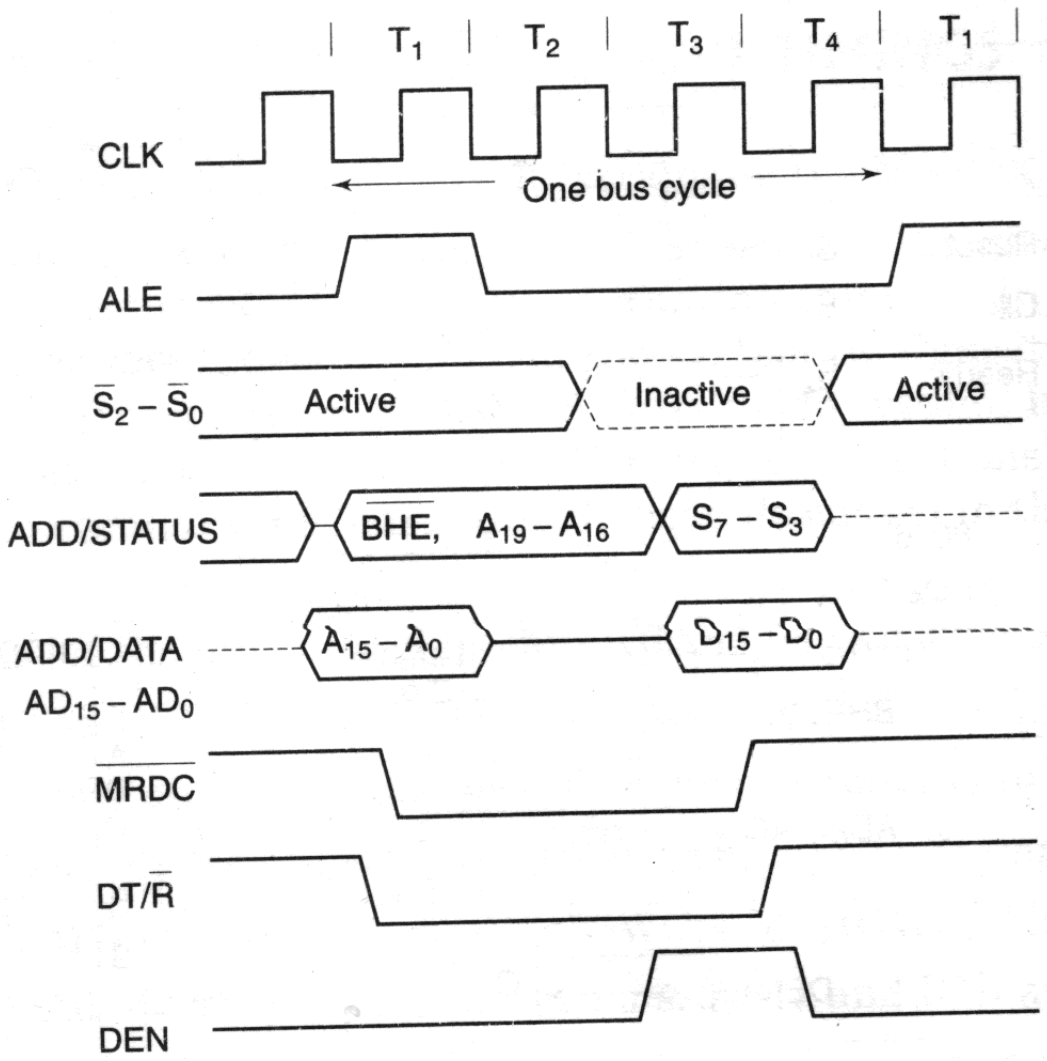
Maximum Mode 8086 System



Maximum Mode 8086 System

- Here, either a numeric coprocessor of the type 8087 or another processor is interfaced with 8086.
- The Memory, Address Bus, Data Buses are shared resources between the two processors.
- The control signals for Maximum mode of operation are generated by the Bus Controller chip 8788.
- The three status outputs S0*, S1*, S2* from the processor are input to 8788.
- The outputs of the bus controller are the Control Signals, namely DEN, DT/R*, IORC*, IOWTC*, MWTC*, MRDC*, ALE etc.

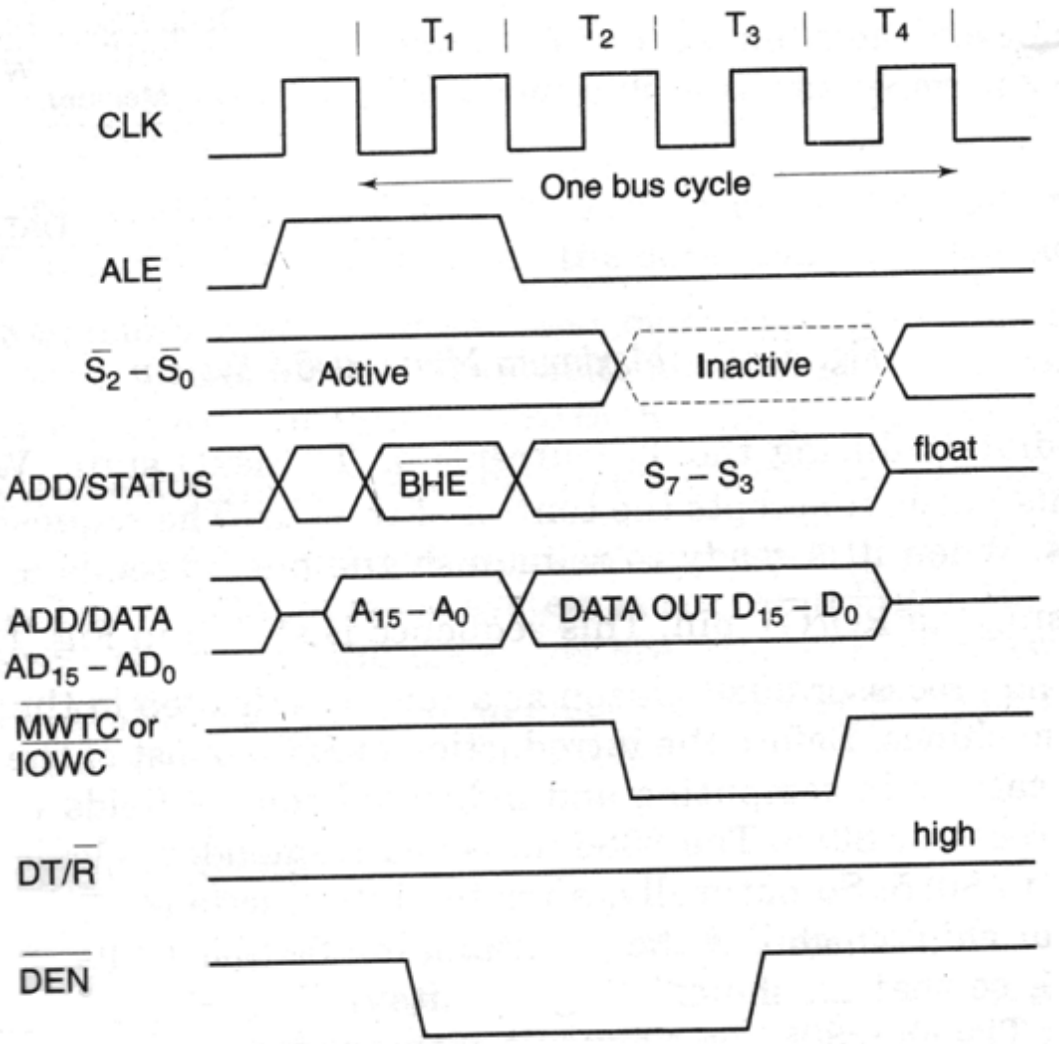
Memory Read timing in Maximum Mode



S2	S1	SO	*Function*
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

TABLE 8–6 Bus control functions generated by the bus controller (8288) using $\overline{S2}$, $\overline{S1}$, and $\overline{SO}$

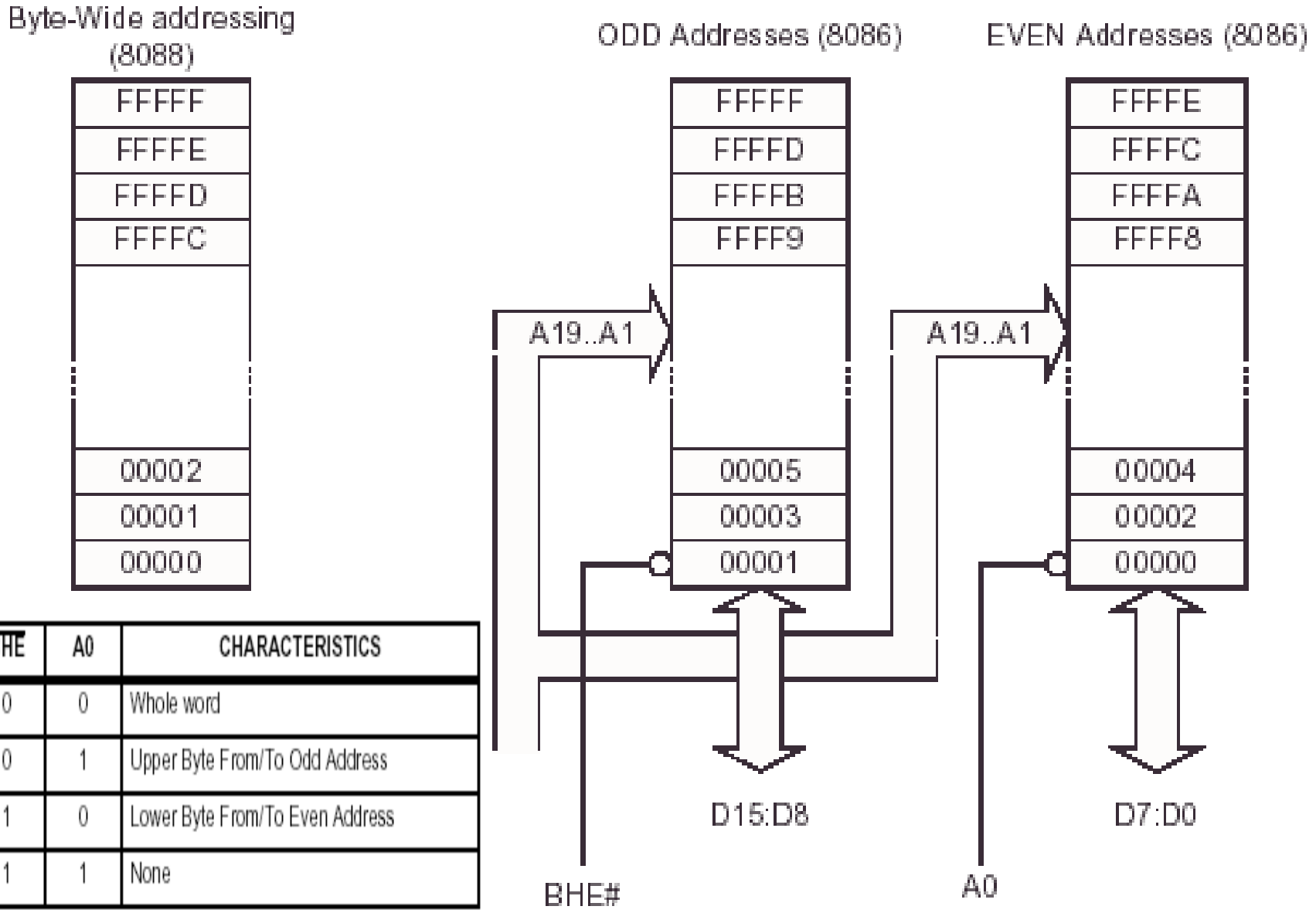
Memory Write timing in Maximum Mode



$\overline{S2}$	$\overline{S1}$	$\overline{SO}$	*Function*
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

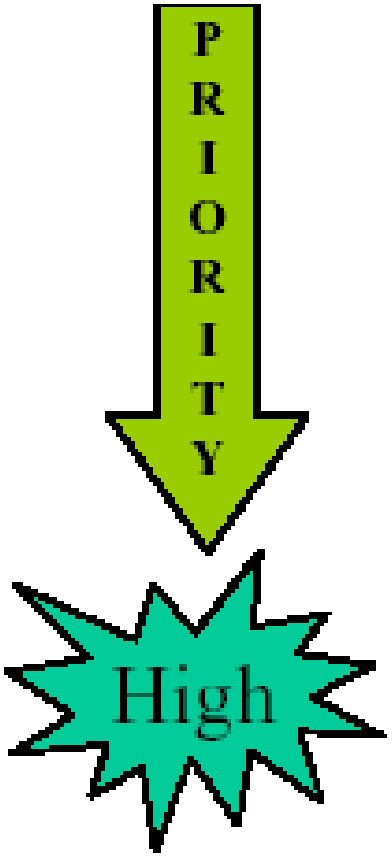
TABLE 8–6 Bus control functions generated by the bus controller (8288) using $\overline{S2}$, $\overline{S1}$, and $\overline{SO}$

Memory Banking



$\overline{BHE}$	A0	CHARACTERISTICS
0	0	Whole word
0	1	Upper Byte From/To Odd Address
1	0	Lower Byte From/To Even Address
1	1	None

8086 Interrupts



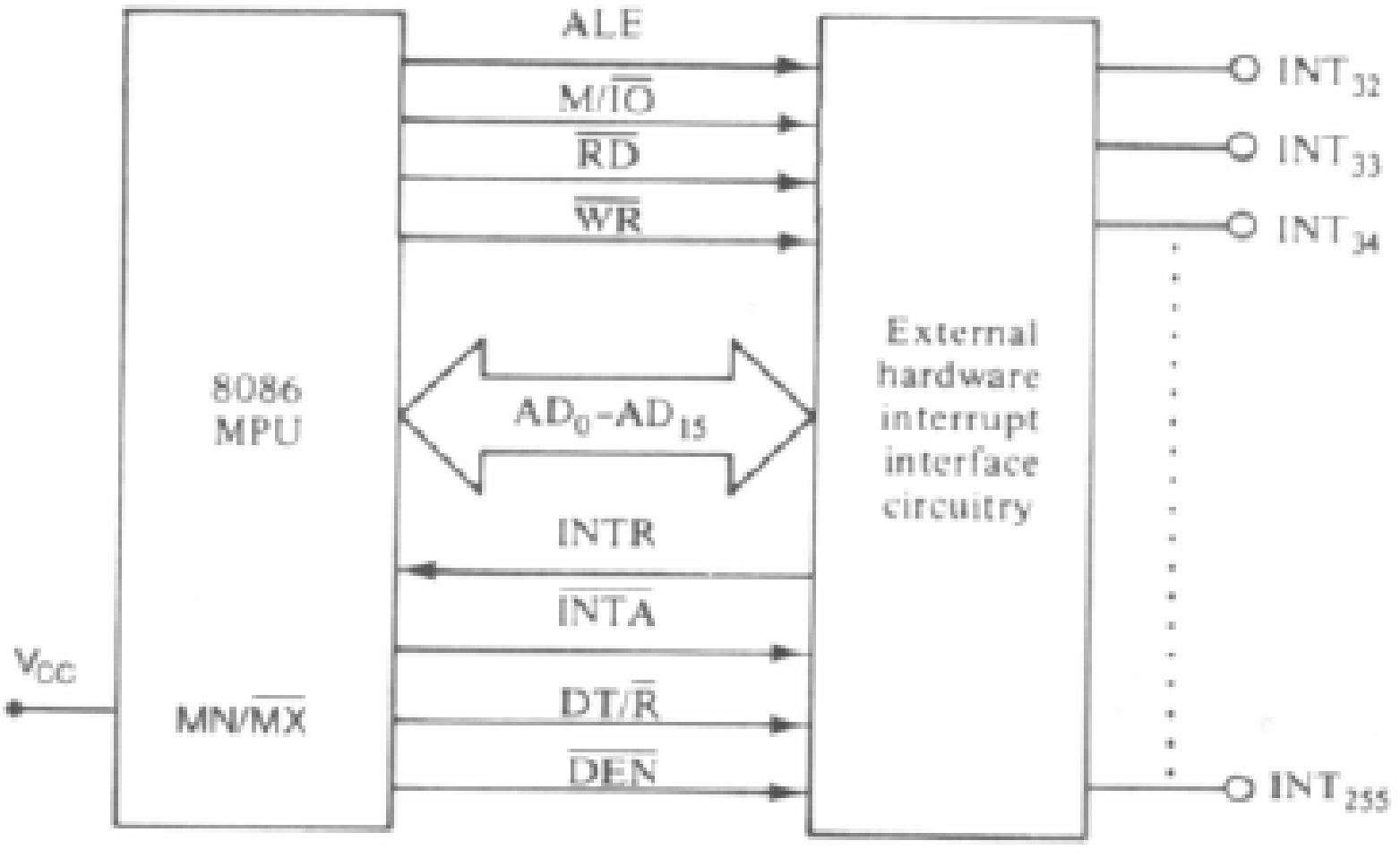
- ✓ External Hardware Interrupts
- ✓ Nonmaskable Interrupt
- ✓ Software Interrupts
- ✓ Internal Interrupts
- ✓ Reset

8086 Interrupts Procedure

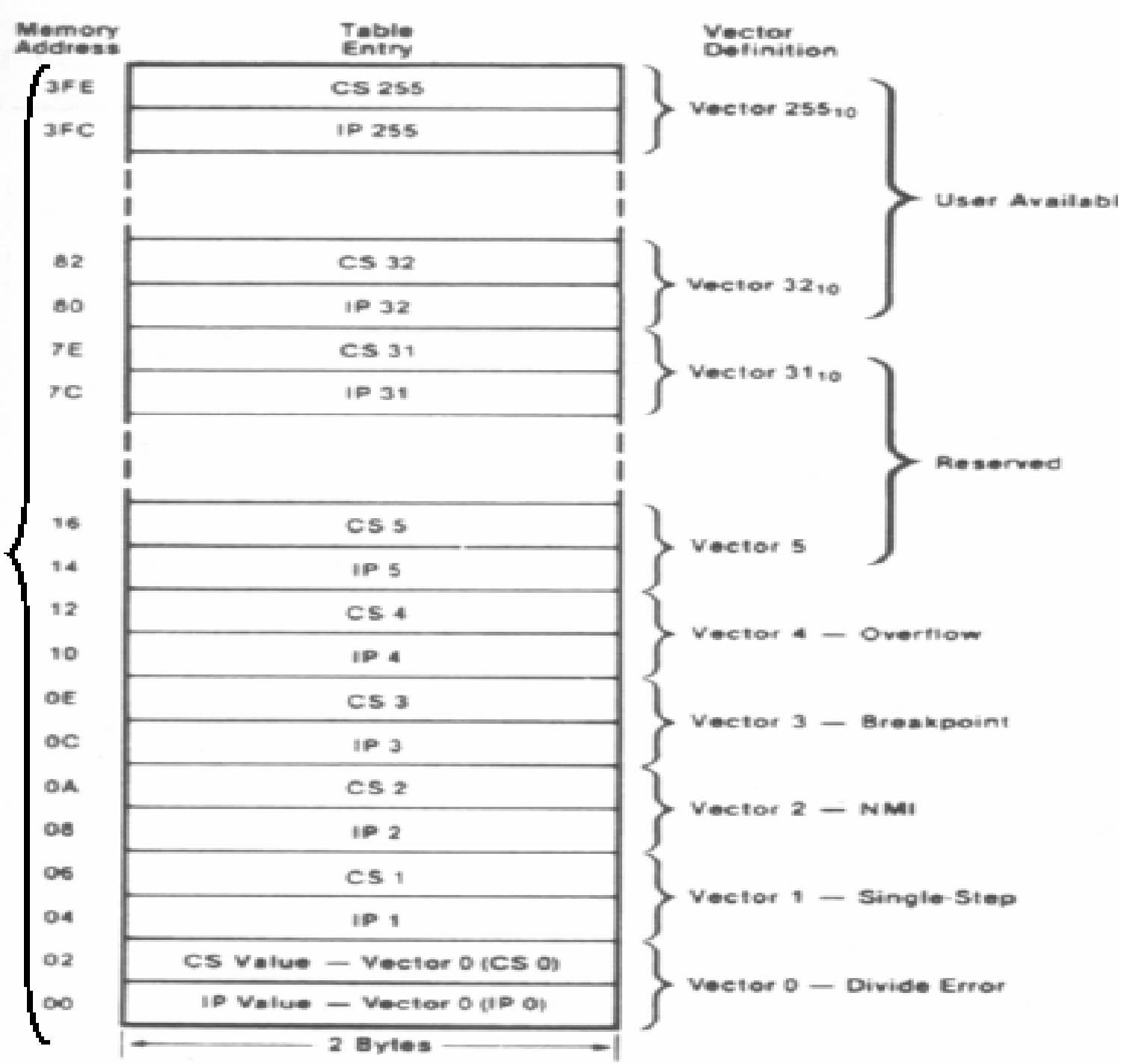
The Operation of Real Mode Interrupt

1. The contents of the FLAG REGİSTERS are pushed onto the stack
2. Both the interrupt (IF) and (TF) flags are cleared. This disables the INTR pin and the trap or single-step feature. (Depending on the nature of the interrupt, a programmer can unmask the INTR pin by the STI instruction)
3. The contents of the code segment register (CS) is pushed onto the stack.
4. The contents of the instruction pointer (IP) is pushed onto the stack.
5. The interrupt vector contents are fetched, and then placed into both IP and CS so that the next instruction executes at the interrupt service procedure addressed by the interrupt vector.
6. While returning from the interrupt-service routine by the instruction IRET, flags return to their state prior to the interrupt and and operation restarts at the prior IP address.

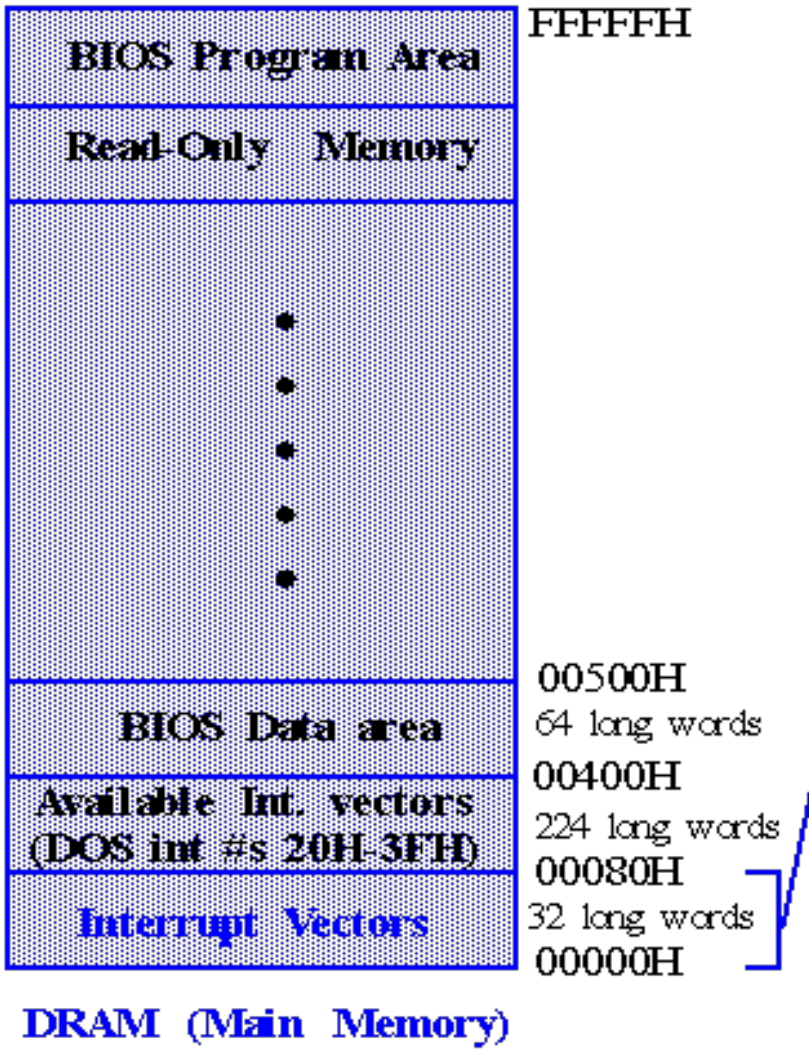
8086 External Interrupts

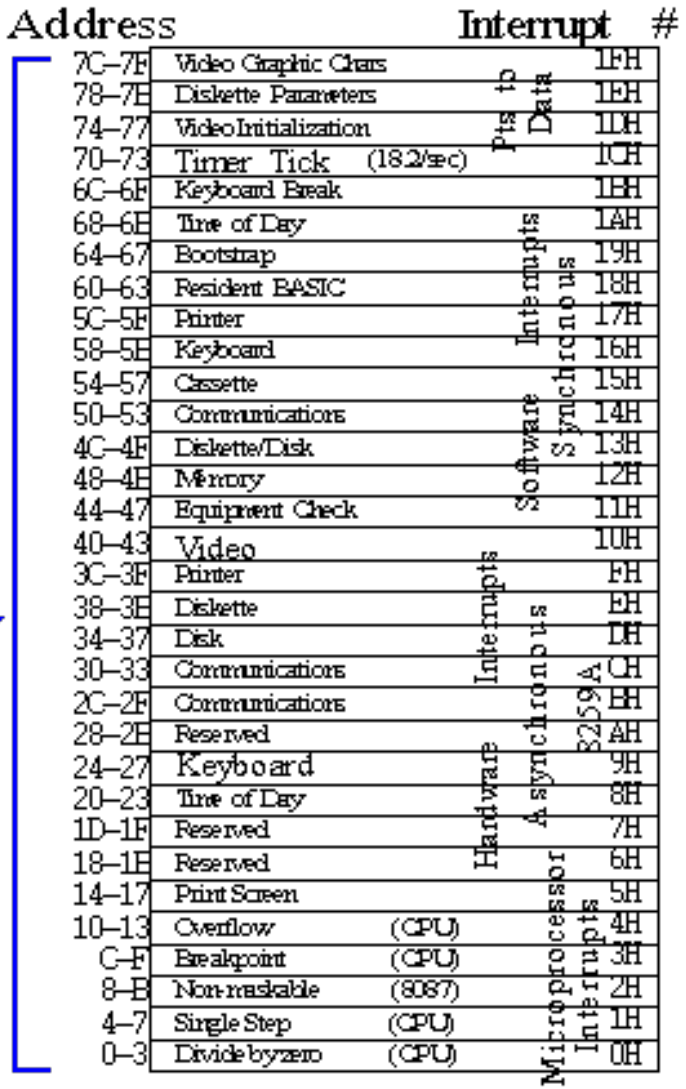


First 1 K memory



Total Memory and IVT





8086 Control Signals

1. ALE
2. BHE
3. M/IO
4. DT/R
5. RD
6. WR
7. DEN

Coprocessor and Multiprocessor configuration

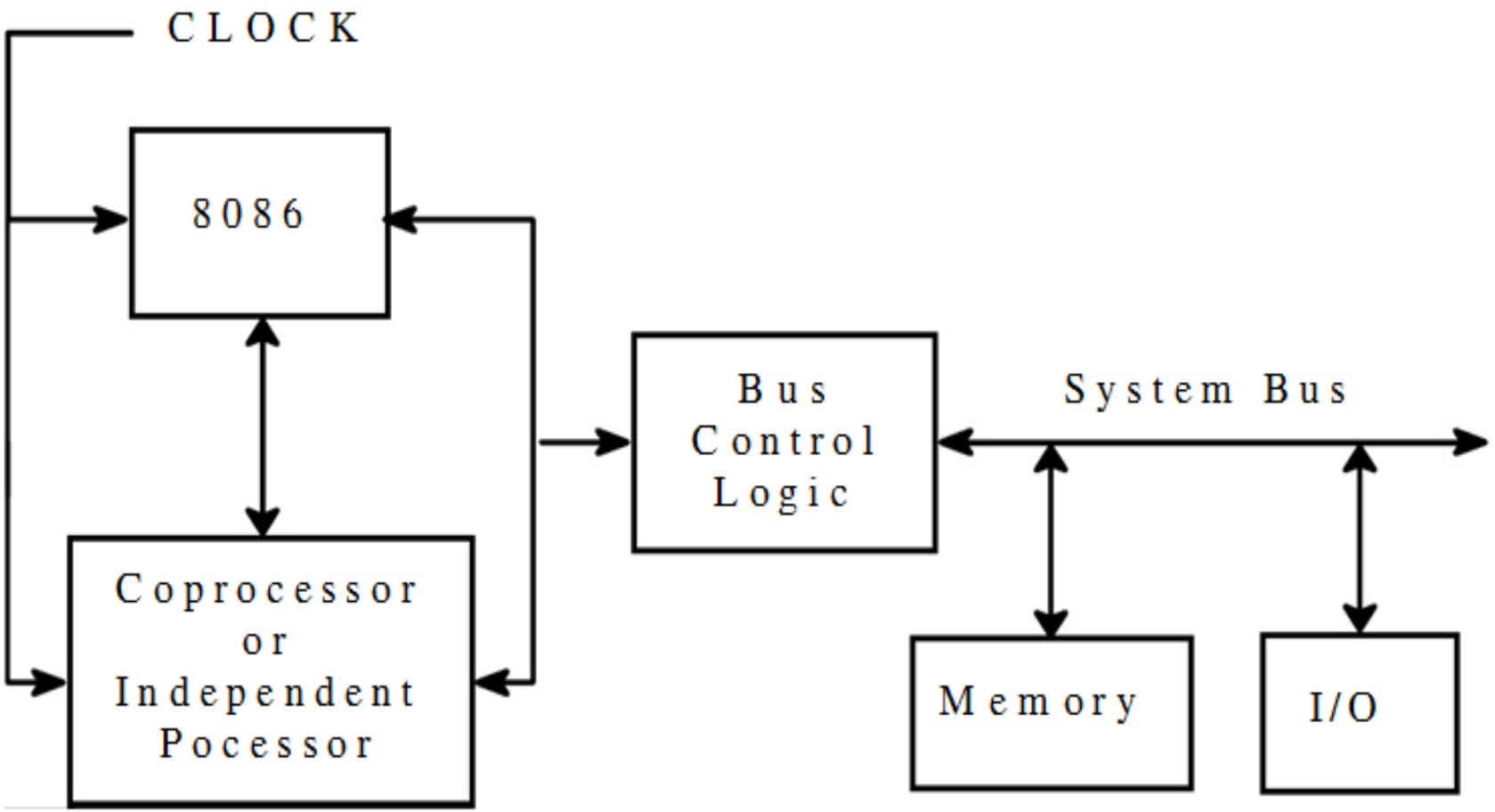
- Multiprocessor Systems refer to the use of multiple processors that **executes instructions simultaneously** and **communicate with each other** using mail boxes and Semaphores.

- Maximum mode of 8086 is designed to implement 3 basic multiprocessor configurations:
    1. **Coprocessor (8087)**
    2. **Closely coupled (8089)**
    3. **Loosely coupled (Multibus)**

Coprocessor and Multiprocessor configuration

- **Coprocessors** and **Closely coupled** configurations are similar in that both the 8086 and the external processor shares the:
  - **Memory**
  - **I/O system**
  - **Bus & bus control logic**
  - **Clock generator**

Coprocessor / Closely Coupled Configuration

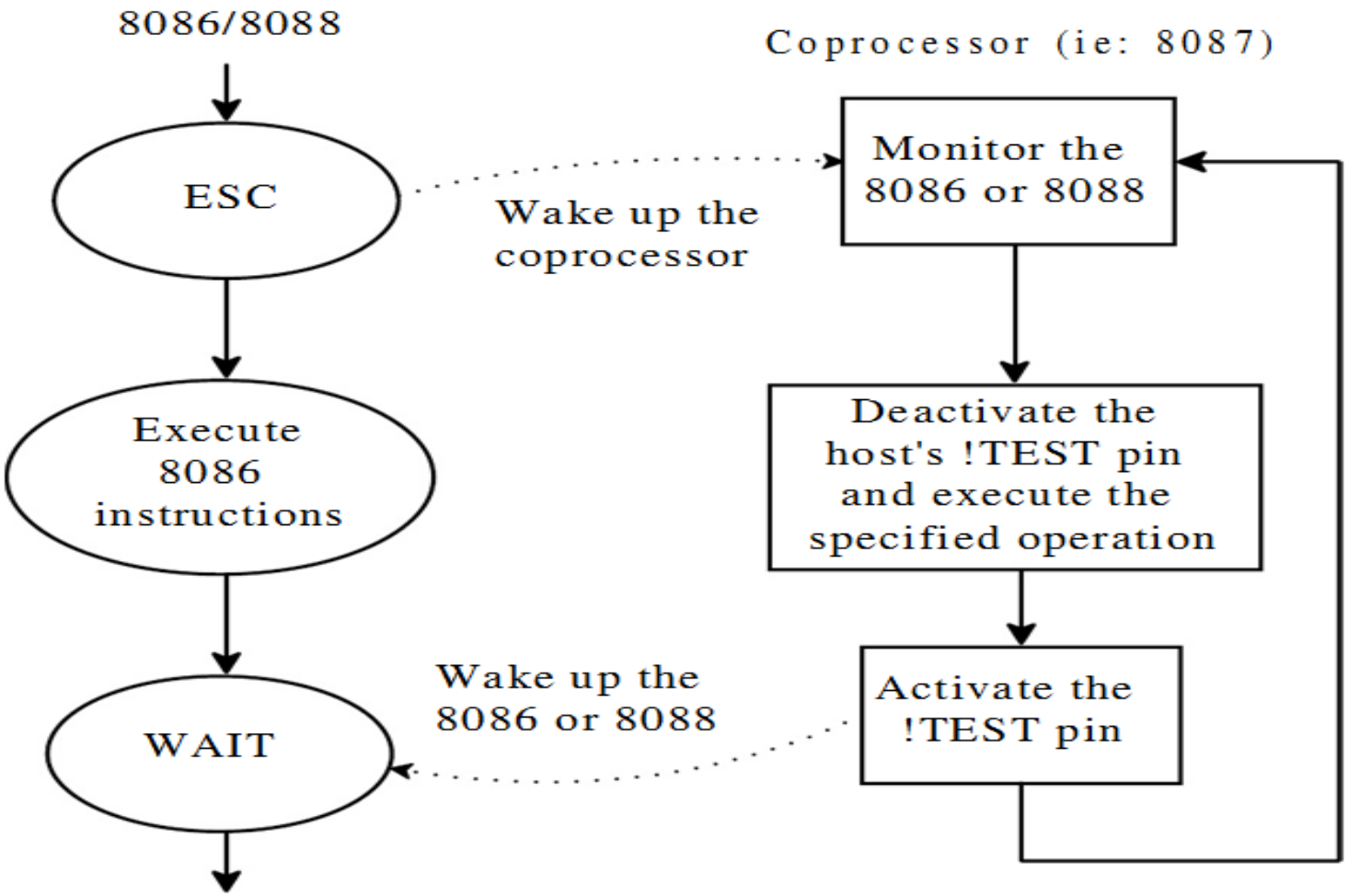


TEST pin of 8086

- Used in conjunction with the WAIT instruction in multiprocessing environments.
- This is input from the 8087 coprocessor.
- During execution of a wait instruction, the CPU checks this signal.
- If it is low, execution of the signal will continue; if not, it will stop executing.

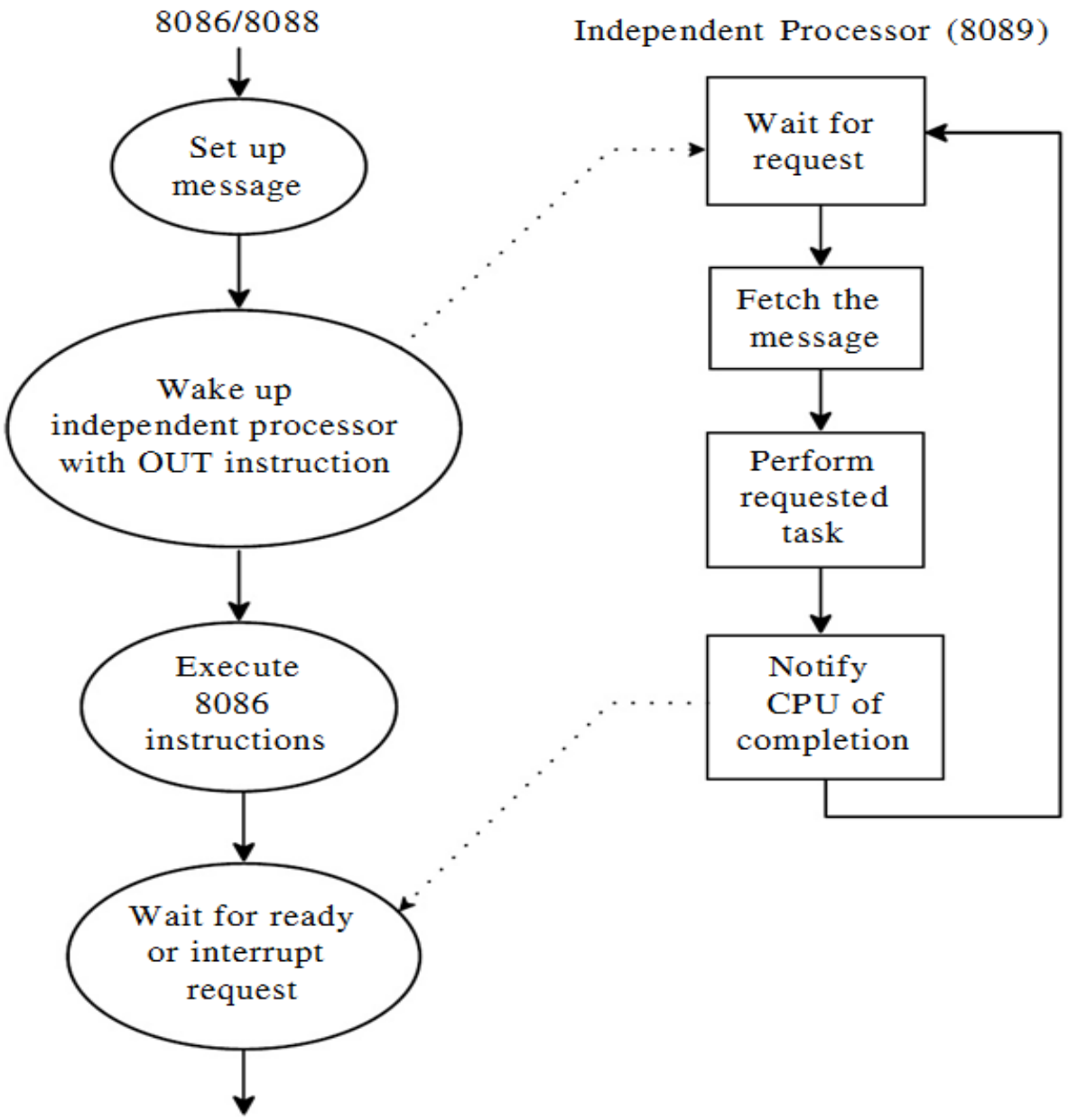
Coprocessor Execution Example

Coprocessor cannot take control of the bus, it does everything through the CPU



Closely Coupled Execution Example

- **Closely Coupled processor may take control of the bus independently.**
- **Two 8086's cannot be closely coupled.**



Loosely Coupled Configuration

- has **shared system bus, system memory, and system I/O**.
- each processor has **its own clock** as well as **its own memory** (in addition to access to the system resources).
- Used for medium to **large multiprocessor systems**.
- Each module is capable of being the **bus master**.
- Any module could be a processor capable of being a bus master, a coprocessor configuration or a closely coupled configuration.

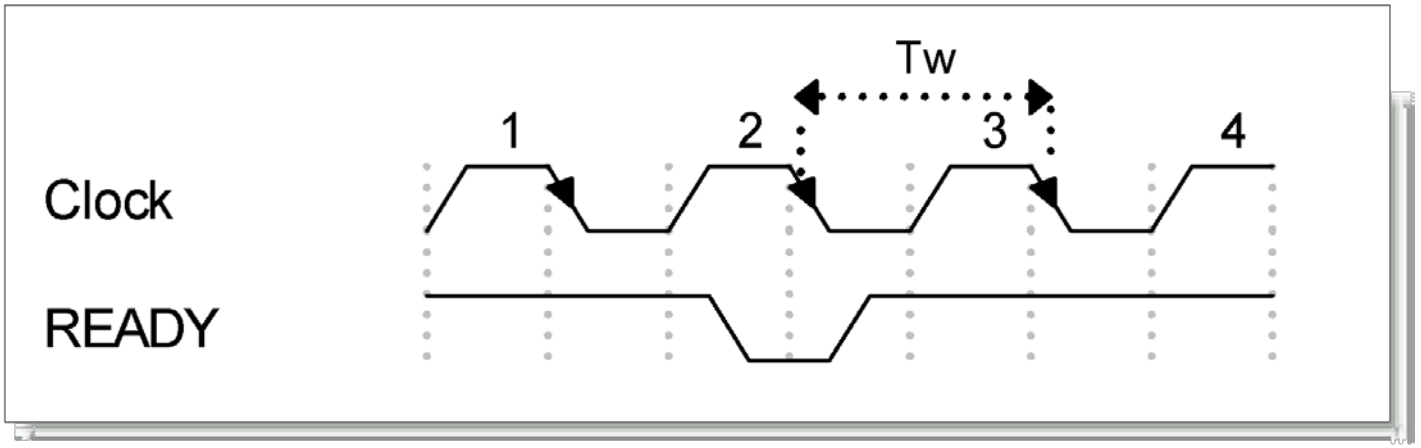
Loosely Coupled Configuration

- **No direct connections** between the modules.
- Each share the system bus and **communicate through shared resources.**
- Processor in their separate modules can simultaneously access their private subsystems through their local busses, and perform their local data references and instruction fetches independently. This results in **improved degree of concurrent processing.**
- **Excellent** for real time applications, as separate modules can be assigned specialized tasks

Advantages of Multiprocessor Configuration

1. High **system throughput** can be achieved by having more than one CPU.
2. The system can be **expanded** in modular form.
   Each bus master module is an independent unit and normally resides on a separate PC board. One can be added or removed without affecting the others in the system.
3. A **failure** in one module normally does not affect the breakdown of the entire system and the faulty module can be easily detected and replaced
4. Each bus master has its own local bus to access dedicated memory or IO devices. So a greater **degree of parallel processing** can be achieved.

WAIT State



- A **wait state (T_w)** is an extra clocking period, inserted between **T2** and **T3**, to lengthen the bus cycle, allowing slower memory and I/O components to respond.

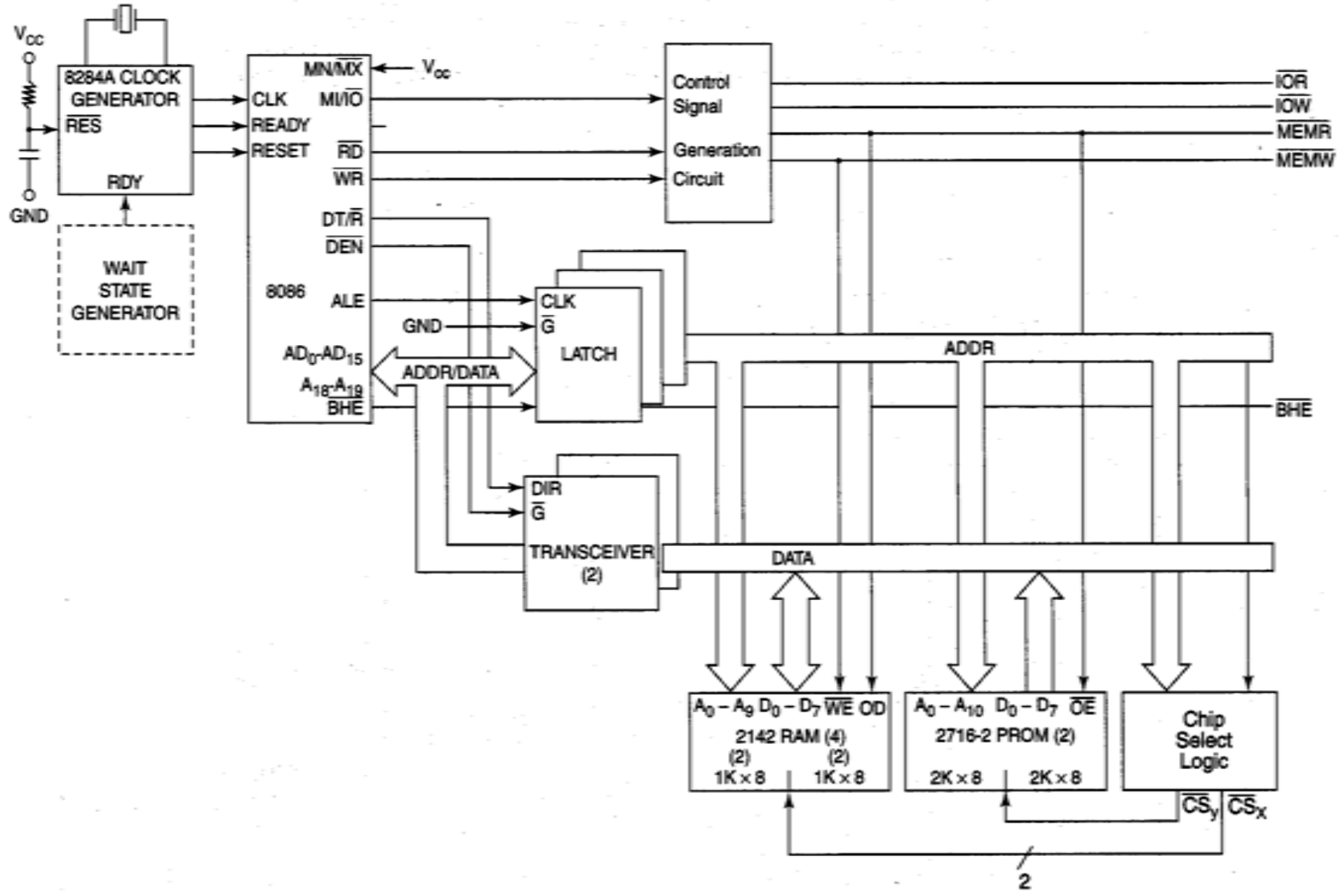
- The **READY** input is sampled at the end of **T2**, and again, if necessary in the middle of Tw. **If READY is '0' then a Tw is inserted.**

8086 System Memory Circuitry

1. **Minimum Mode System Memory Circuitry**

1. **Maximum Mode System Memory Circuitry**

Minimum Mode System Memory Circuitry



Maximum Mode System Memory Circuitry