CSE-3103: Microprocessor and Microcontroller

Dept. of Computer Science and Engineering
University of Dhaka

Prof. Sazzad M.S. Imran, PhD
Dept. of Electrical and Electronic Engineering
sazzadmsi.webnode.com

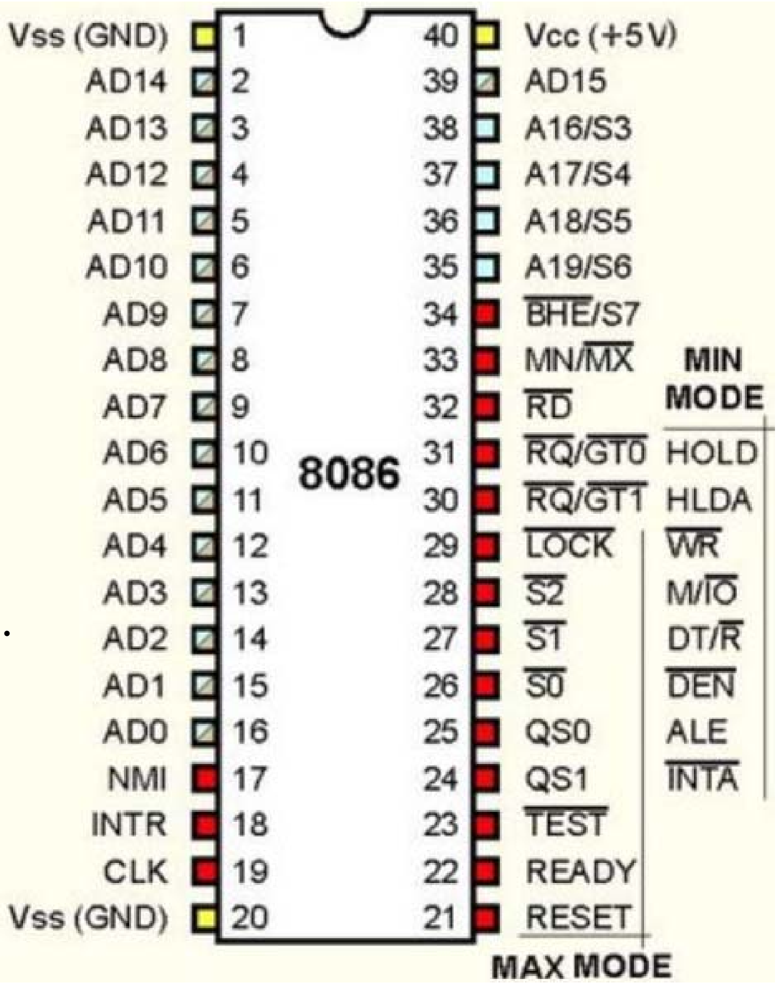
Pin Diagram of 8086 Microprocessor

pin 29 →

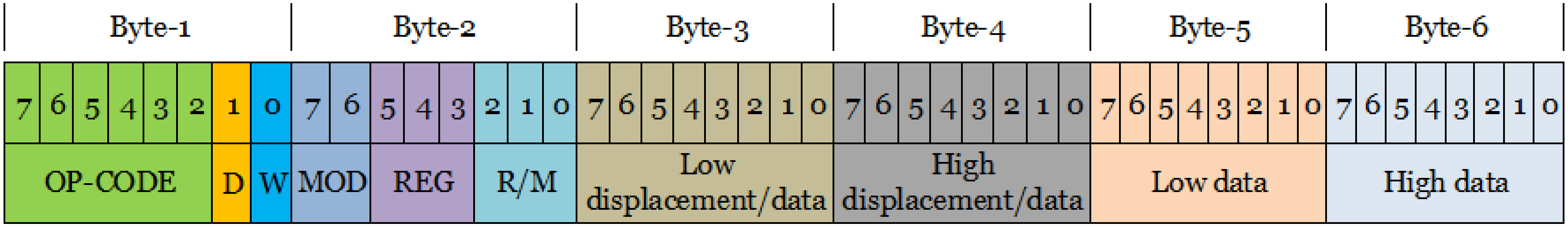
$\overline{\text{LOCK}}$ signal (MAX mode),
activated by LOCK prefix instruction,
remains active until completion of next instruction.
$\overline{\text{LOCK}}$ = low → all interrupts get masked,
HOLD request is not granted.

pin 30 and 31 →

$\overline{\text{RQ}}/\overline{\text{GT}_1}$ and $\overline{\text{RQ}}/\overline{\text{GT}_0}$ =
request/grant signals (MAX mode),
other processors request CPU to release system bus.
when signal is received, CPU sends acknowledgment.



Instruction Format of 8086



<table>
<tr><td colspan="8">Byte-1</td><td colspan="8">Byte-2</td><td colspan="8">Byte-3</td><td colspan="8">Byte-4</td><td colspan="8">Byte-5</td><td colspan="8">Byte-6</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">OP-CODE</td><td>D</td><td>W</td><td colspan="2">MOD</td><td colspan="3">REG</td><td colspan="3">R/M</td><td colspan="8">Low
displacement/data</td><td colspan="8">High
displacement/data</td><td colspan="8">Low data</td><td colspan="8">High data</td></tr>
</table>

Instruction varies from 1 to 6 bytes.

1st byte →

Op-code field (6-bit) →

specifies operation to be performed.

Register direction bit (D bit) →

register operand specified in byte 2 = source or destination?

D = 1 → destination operand.

D = 0 → source operand.

Data size bit (W bit) →

W = 0 → 8-bit operation.

W = 1 → 16-bit operation.

Instruction Format of 8086

Byte-1			Byte-2			Byte-3	Byte-4	Byte-5	Byte-6
7 6 5 4 3 2	1	0	7 6	5 4 3	2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
OP-CODE	D	W	MOD	REG	R/M	Low displacement/data	High displacement/data	Low data	High data

2nd byte →

3 fields: MOD (2-bit), R/M (3-bit), REG (3-bit).

3-bit REG field →

identify register for 1st operand,
source or destination by D bit in byte-1.

D = 1 → destination operand.

D = 0 → source operand.

8086 operation →

one operand is in memory or
both operands are in registers.

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Instruction Format of 8086

<table>
<tr><th colspan="8">Byte-1</th><th colspan="8">Byte-2</th><th colspan="8">Byte-3</th><th colspan="8">Byte-4</th><th colspan="8">Byte-5</th><th colspan="8">Byte-6</th></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="6">OP-CODE</td><td>D</td><td>W</td><td colspan="2">MOD</td><td colspan="3">REG</td><td colspan="3">R/M</td><td colspan="8">Low displacement/data</td><td colspan="8">High displacement/data</td><td colspan="8">Low data</td><td colspan="8">High data</td></tr>
</table>

2nd byte →

3 fields: MOD (2-bit), R/M (3-bit), REG (3-bit).

2-bit MOD field →

specify 2nd operand is in register or memory?

addressing mode is defined by R/M field.

8086 operation →

one operand is in memory or both operands are in registers.

<table>
<tr><th>MOD</th><th colspan="2">2nd operand</th></tr>
<tr><td>0 0</td><td colspan="2">Memory addressing without displacement</td></tr>
<tr><td>0 1</td><td colspan="2">Memory addressing with 8-bit displacement</td></tr>
<tr><td>1 0</td><td colspan="2">Memory addressing with 16-bit displacement</td></tr>
<tr><td rowspan="2">1 1</td><td rowspan="2">Register addressing</td><td>W = 0 → 8-bit data</td></tr>
<tr><td>W = 1 → 16-bit data</td></tr>
</table>

Instruction Format of 8086

Byte-1			Byte-2			Byte-3	Byte-4	Byte-5	Byte-6
7 6 5 4 3 2	1	0	7 6	5 4 3	2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
OP-CODE	D	W	MOD	REG	R/M	Low displacement/data	High displacement/data	Low data	High data

3-bit R/M field →

R/M	MOD 00	MOD 01	MOD 10	MOD 11		Segment Register
				W = 0	W = 1	
000	BX+SI	BX+SI+D8	BX+SI+D16	AL	AX	DS
001	BX+DI	BX+DI+D8	BX+DI+D16	CL	CX	DS
010	BP+SI	BP+SI+D8	BP+SI+D16	DL	DX	SS
011	BP+DI	BP+DI+D8	BP+DI+D16	BL	BX	SS
100	SI	SI+D8	SI+D16	AH	SP	DS
101	DI	DI+D8	DI+D16	CH	BP	DS
110	*D16 [DS]	BP+D8 [SS]	BP+D16 [SS]	DH	SI	DS or SS
111	BX	BX+D8	BX+D16	BH	DI	DS

*direct addressing

Instruction Format of 8086

Two exceptions →

1) Direct memory to memory data transfer is not allowed.

AX is used as intermediate stage of data.

example →

MOV [DI], [SI] is not allowed.

this must be done as →

```
MOV AH, [SI]
MOV [DI], AH
```

2) DS register cannot be loaded directly by data segment address.

Operation is done as →

```
MOV AX, DS ADDR       ; AX is loaded with initial address of DS
MOV DS, AX            ; DS register is loaded with AX
```

Address Capability and Memory Map of 8086

8086 microprocessor →

address bus = 20 bit,

address 2^{20} = 1 MB of different memory locations.

memory address = 00000H to FFFFFH.

Total memory space →

blocks = 16,

each block →

2^{16} = 64 kB [16-bit register].

most significant hex digit increases by 1.

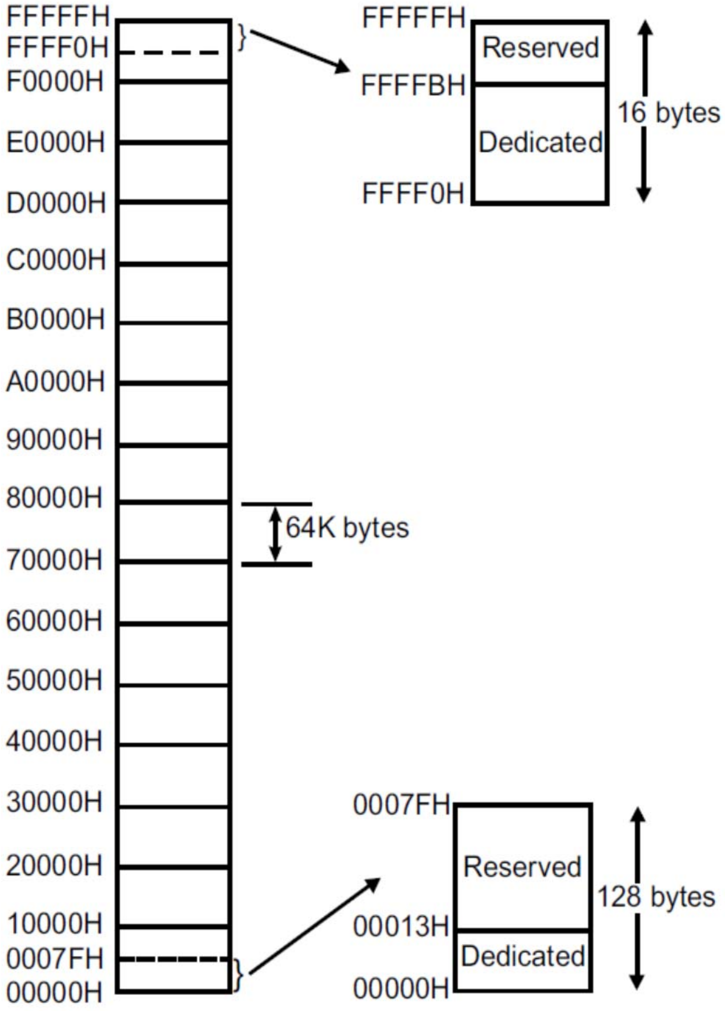
Reserved locations →

future hardware and software needs.

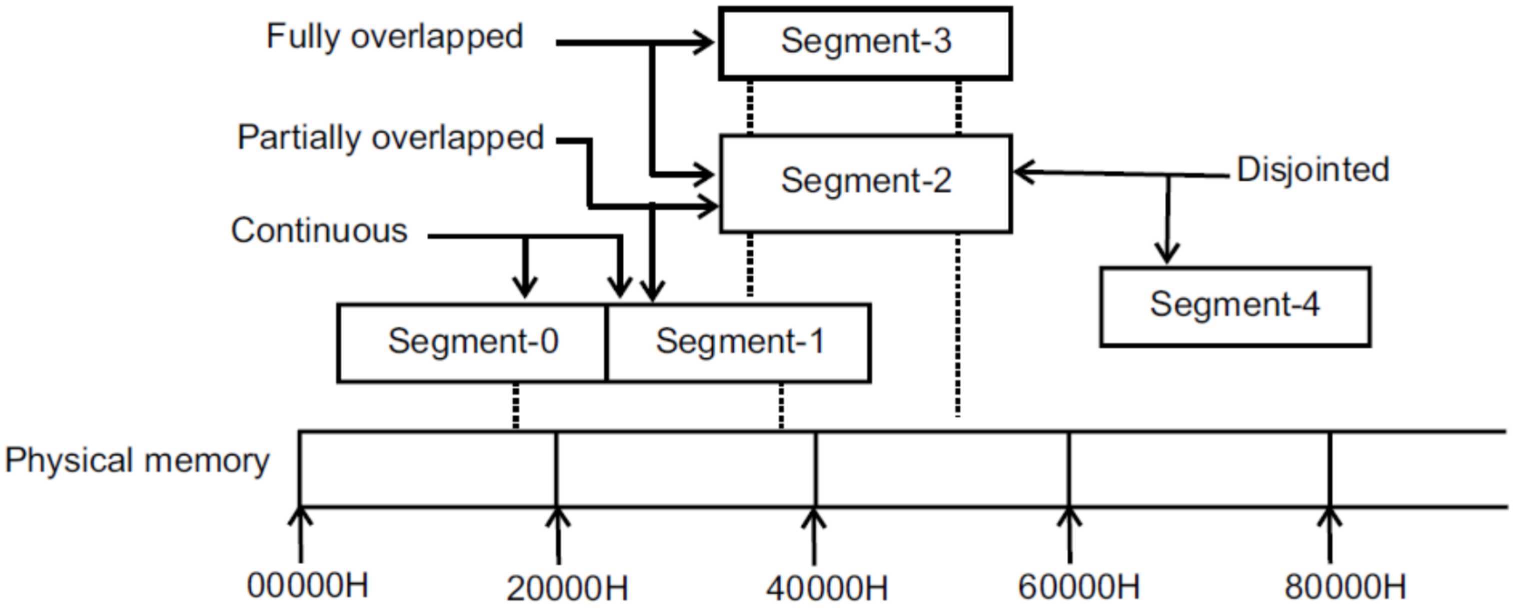
Dedicated locations →

specific system interrupts,

reset functions.



Memory Segmentation of 8086



Different memory segmentations of 8086 →

1) continuous,
2) partially overlapped,
3) fully overlapped,
4) disjointed.

Memory Segmentation of 8086

1 MB memory = 16 segments.
each segment = 2^{16} = 64 kB [16-bit register].

4 segments can be active at any given instant of time →

Memory segment = 2^{16} = 64 kB	*Corresponding segment register = 16 bits	Content/uses of memory segment
Code segment	Code Segment Register (CS)	instruction codes of program
Data segment	Data Segment Register (DS)	data, variables, constants
Stack segment	Stack Segment Register (SS)	interrupt and subroutine return addresses
Extra segment	Extra Segment Register (ES)	destination of data for string instructions

*Segment register ← starting address of particular memory segment

Maximum size of active memory →
64×4 = 256 kB.
program storage → 64 kB in CS
stack → 64 kB in SS.
data storage → 128 kB in DS and ES.