Computer Engineering

Electrical Engineering Department

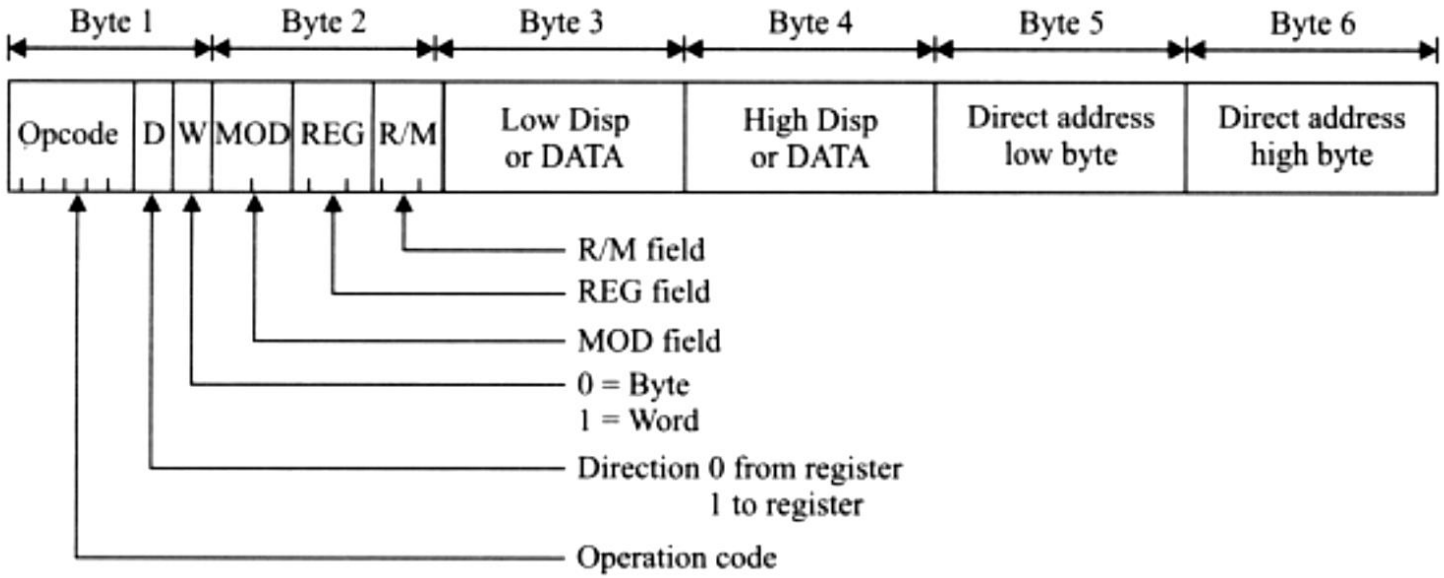
Third Stage

Instructor: Asst. Lecturer

Adnan Ali Abdullah

Converting Assembly Language Instructions to Machine Co

The general instruction format for machine code is shown below :



BYTE 1 Specification

- **OPCODE field (6-bits) :** Specifies the operation to be performed such as MOV , ADD , SUB …..etc.
- **Register direction bit (D-bit):**

D = 1 : the register operand specified by **REG** in byte 2 is a destination operand.

D = 0 : the register operand specified by **REG** in byte 2 is a source operand

- **Data size bit (W-bit):** Specifies whether the operation will be performed on 8-bit or 16-bit data.

W = 1 : 16-bit data size

W = 0 : 8-bit data size

BYTE 2 Specification: byte 2 has three fields:

- **Mode (MOD) field (2-bits) :** Indicates whether the operand is in register or memory.

- **Register (REG) field (3-bit) :** Identifies the register for the first operand
- **Register/Memory (R/M) field (3-bit):** Together with MOD field to specify the second operand.

MOD	Explanation
00	Memory Mode no displacement
01	Memory Mode 8-bit displacement
10	Memory Mode 16-bit displacement
11	Register Mode (no displacement)

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

MOD = 11			Effective Address Calculation			
Reg.	W=0	W=1	R/M	MOD=00	MOD=01	MOD=10
000	AL	AX	000	(BX)+(SI)	(BX)+(SI)+D8	(BX)+(SI)+D16
001	CL	CX	001	(BX)+(DI)	(BX)+(DI)+D8	(BX)+(DI)+D16
010	DL	DX	010	(BP)+(SI)	(BP)+(SI)+D8	(BP)+(SI)+D16
011	BL	BX	011	(BP)+(DI)	(BP)+(DI)+D8	(BP)+(DI)+D16
100	AH	SP	100	(SI)	(SI)+D8	(SI)+D16
101	CH	BP	101	(DI)	(DI)+D8	(DI)+D16
110	DH	SI	110	Direct Address	(BP)+D8	(BP)+D16
111	BH	DI	111	(BX)	(BX)+D8	(BX)+D16

Ex : Encode the following instruction in machine code. Assume that the OPCODE for MOV instruction is 100010_2.

MOV BL , AL

Ans.

OPCODE = 100010 (for MOV), D = 0 (source), W = 0 (8-bit) this leads to :

BYTE 1 = $10001000_2 = 88_{16}$

In byte 2 the source operand, specified by REG is AL , then:

REG = 000, MOD = 11, R/M = 011

Therefore:

BYTE 2 = $11000011_2 = C3_{16}$

The machine code for the instruction is:

MOV BL , AL = 88C3H

Ex : Encode the following instruction in machine code. Assume that the OPCODE for ADD instruction is 000000_2.

ADD AX , [SI]

Ans.

OPCODE = 000000 (for ADD), D = 1 (destination), W = 1 (16-bit) this leads to

BYTE 1 = $00000011_2 = 03_{16}$

In byte 2 the destination operand, specified by REG is AX , then:

REG = 000, MOD = 00, R/M = 100

Therefore:

BYTE 2 = $00000100_2 = 04_{16}$

The machine code for the instruction is:

ADD AX, [SI] = 0304_{16}

Ex: Encode the following instruction in machine code. Assume that the OPCODE for XOR instruction is 001100_2.

XOR CL , [1234H]

Ans.

OPCODE = 001100 (for XOR), D = 1 (destination), W = 0 (8-bit) this leads to:

BYTE 1 = $00110010_2 = 32_{16}$

In byte 2 the destination operand, specified by REG is CL , then:

REG = 001 , MOD = 00, R/M = 110

Therefore:

BYTE 2 = 000011102 = $0E_{16}$

BYTE 3 = 34_{16} and BYTE 4 = 12_{16}

The machine code for the instruction is:

XOR CL , [1234H] = $320E3412_{16}$

Ex: Encode the following instruction in machine code. Assume that the OPCODE for ADD instruction is 000000_2.

ADD [BX+DI+1234H] , AX

Ans.

OPCODE = 000000 (for ADD) , D = 0 (source), W = 1 (16-bit) This leads to :

BYTE 1 = $00000001_2 = 01_{16}$

In byte 2 the destination operand, specified by REG is AX, then :

REG = 000 , MOD = 10 , R/M = 001

Therefore :

BYTE 2 = $10000001_2 = 81_{16}$

BYTE 3 = 34_{16} and BYTE 4 = 12_{16}

The machine code for the instruction is:

ADD [BX+DI+1234H] , AX = 01813412_{16}

Ex: Encode the following instruction in machine code

MOV [BP+DI+1234H] , 0ABCDH

Ans.

This example does not follow the general format. The OPCODE of MOV (immediate to memory) is 1100011W, and W = 1 for word-size data, then:

BYTE 1 = $11000111_2 = C7_{16}$

BYTE 2 = (MOD)000(R/M) = $100000112 = 83_{16}$

BYTE 3 = 34_{16} and BYTE 4 = 12_{16}

BYTE 5 = CD_{16} and BYTE 6 = AB_{16}

The machine code for the instruction is:

MOV [BP+DI+1234H] , 0ABCDH = $C7833412CDAB_{16}$

Instructions Involve a Segment Register (SR-field)

Instructions that involve a segment register need a 2-bit field to encode which register is to be affected, this field is called the *SR field*. The four segment registers are encoded as shown in the following table.

Segment Register	SR
ES	00
CS	01
SS	10
DS	11

Ex: Encode the above instruction in machine code:

MOV [BP+DI+1234H] , DS

Ans.

This example does not follow the general format. The OPCODE of MOV (Seg. Reg. to memory) is 10001100_2, and the instruction is:

10001100(MOD)0(SR)(R/M)(DISP)

Then we find that for DS , the SR = 11 therefore, the instruction is coded as:

MOV [BP+DI+1234H] , DS $=1000110010011011001101000001001 0_2$

$=8C9B3412_{16}$

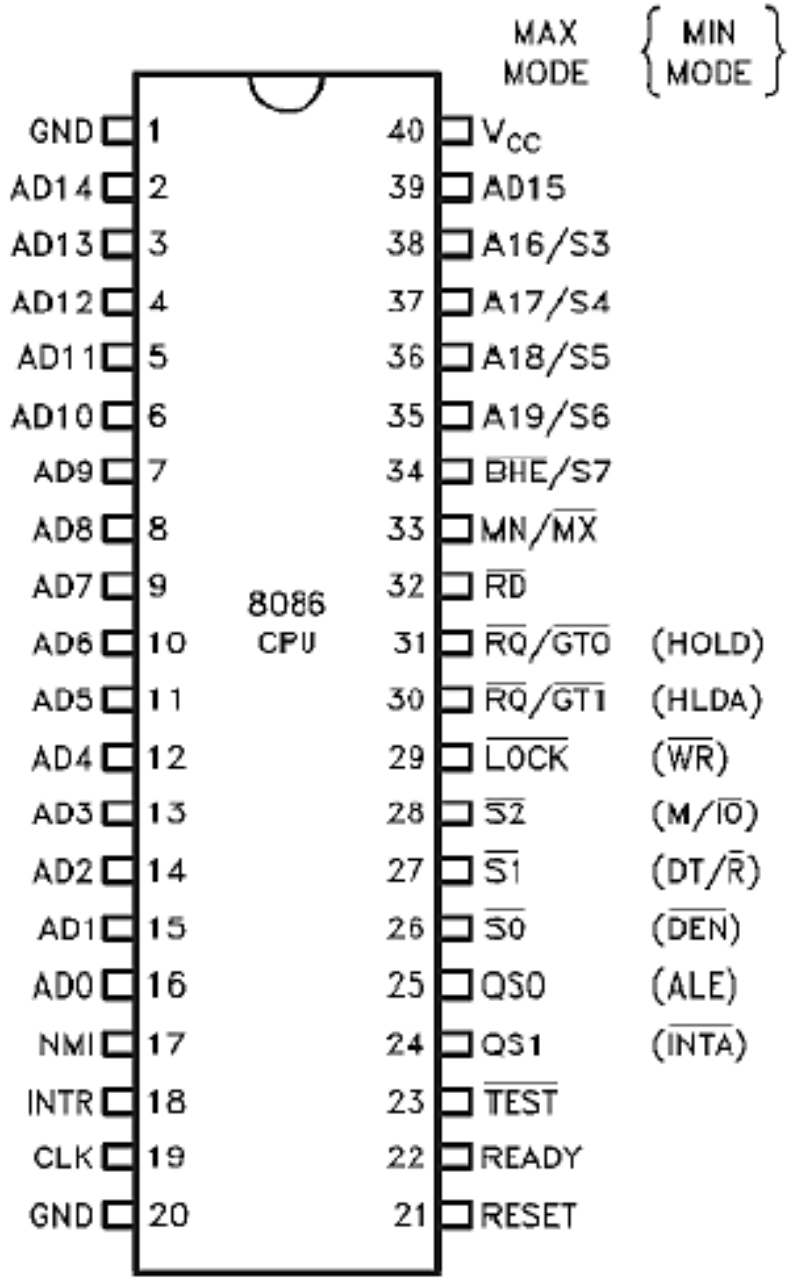
Encoding a Complete Program in Machine Code

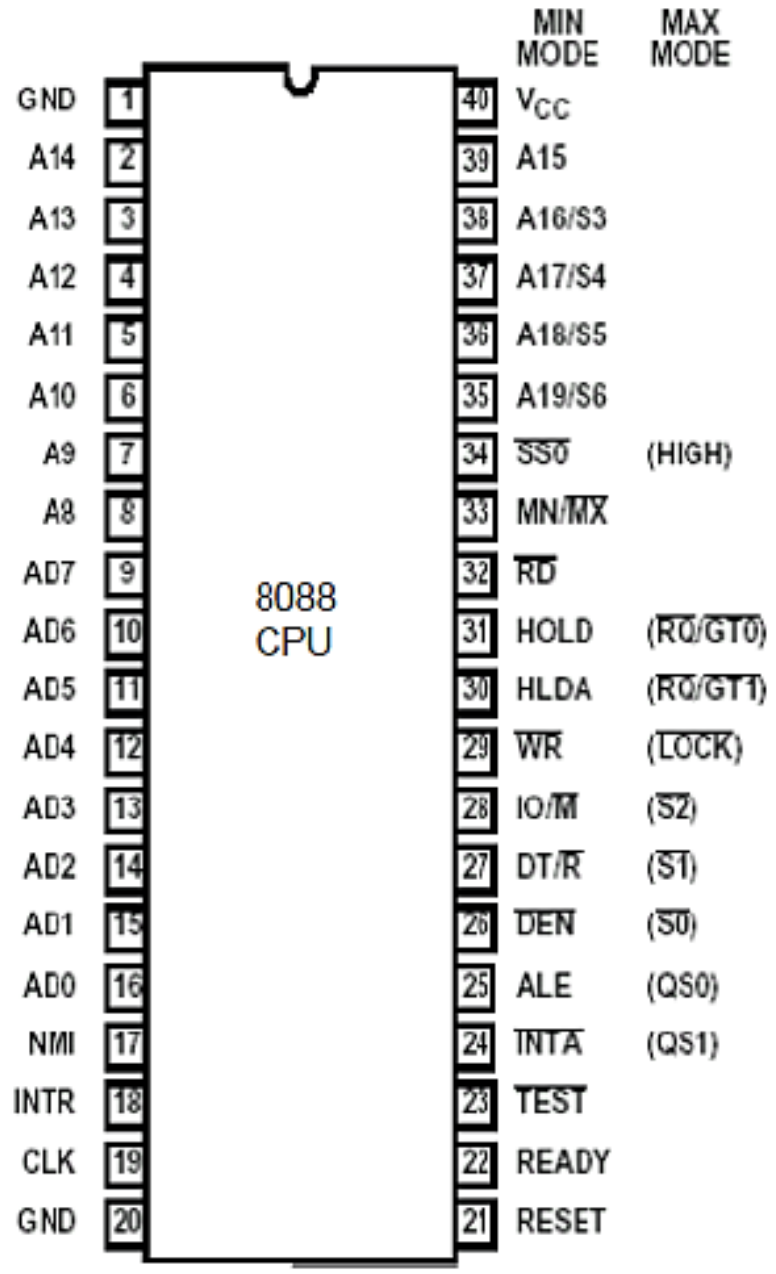
The following steps have to be followed in encoding a complete assembly program:

- Identify the general machine code format.
- Evaluate the bit fields.
- Express the binary-code instruction in hexadecimal form
- To execute the program, the machine code of the program must be stored in the code segment of memory.
- The first byte of the program is stored at the lowest address.

The 8086/8088Microprocessor Architecture and Pin-configuration

- The 8086μP, announced in 1978, was the first 16-bit microprocessor introduced by Intel Corporation.
- The 8086 μP is internally a 16-bit μP and externally it has a 16-bit data bus.
- The 8086 μP is internally a 16-bit μP and externally it has a 8-bit data bus.
- It has the ability to address up to 1 Mbyte of memory via its 20-bit address bus.
- In addition, it can address up to 64K of byte-wide input/output ports.
- It is manufactured using high-performance metal-oxide semiconductor (HMOS) technology, and the circuitry on its chip is equivalent to approximately 29000 transistors.
- The 8086μP is housed in a 40-pin dual in-line package. The signals pinned out to each lead are shown in Figure below.
- The address bus lines A_0 through A_{15} and data bus lines D_0 through D_{15} are multiplexed. For this reason, they are labeled AD_0 through AD_{15}. By *multiplexed* we mean that the same physical pin carries an address bit at one time and the data bit at another time.


MAX MODE
MIN MODE
8086 CPU
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
GND 20
40 VCC
39 AD15
38 A16/S3
37 A17/S4
36 A18/S5
35 A19/S6
34 BHE/S7
33 MN/MX
32 RD
31 RQ/GT0 (HOLD)
30 RQ/GT1 (HLDA)
29 LOCK (WR)
28 S2 (M/IO)
27 S1 (DT/R)
26 S0 (DEN)
25 QS0 (ALE)
24 QS1 (INTA)
23 TEST
22 READY
21 RESET


MIN MODE
MAX MODE
8088 CPU
GND 1
A14 2
A13 3
A12 4
A11 5
A10 6
A9 7
A8 8
AD7 9
AD6 10
AD5 11
AD4 12
AD3 13
AD2 14
AD1 15
AD0 16
NMI 17
INTR 18
CLK 19
GND 20
40 VCC
39 A15
38 A16/S3
37 A17/S4
36 A18/S5
35 A19/S6
34 SS0 (HIGH)
33 MN/MX
32 RD
31 HOLD (RQ/GT0)
30 HLDA (RQ/GT1)
29 WR (LOCK)
28 IO/M (S2)
27 DT/R (S1)
26 DEN (S0)
25 ALE (QS0)
24 INTA (QS1)
23 TEST
22 READY
21 RESET

Minimum-Mode and Maximum-Mode System

- The 8086µP and 8088µP microprocessors can be configured to work in either of two modes:

1. Minimum Mode:

- The 8086µP operates in minimum mode by connecting its $\mathbf{MN/\overline{MX}}$ pin to Vcc. ($\mathbf{MN/\overline{MX} = 1}$).
- In this mode, all the control signals are given out by the microprocessor chip.
- It is typically used for **single microprocessor** systems.

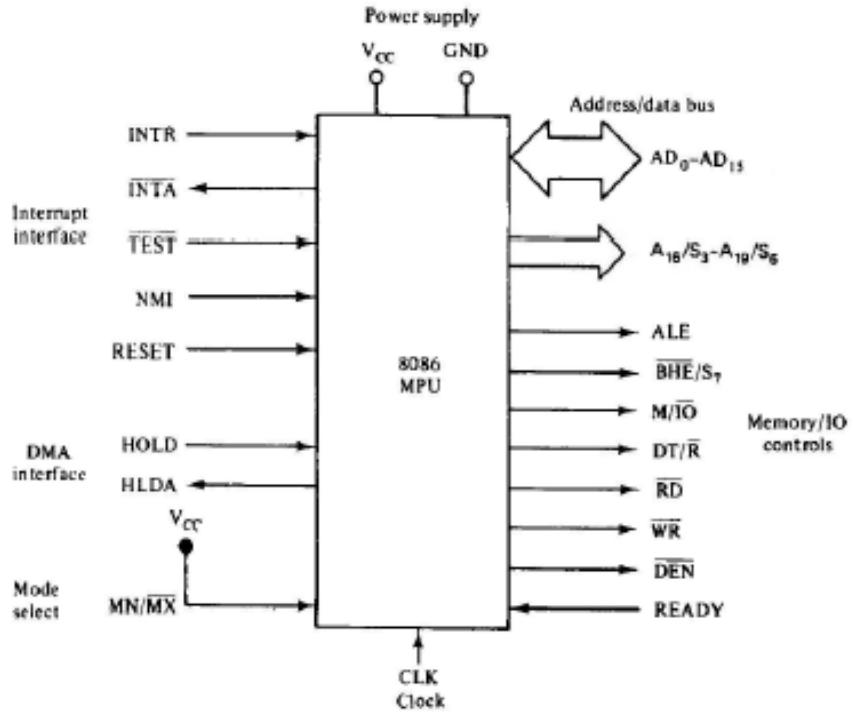
2. Maximum Mode:

- The 8086µP operates in minimum mode by connecting its $\mathbf{MN/\overline{MX}}$ pin to ground. ($\mathbf{MN/\overline{MX} = 0}$).
- In this mode, the processor derives the status signal S2, S1, S0 by another chip called bus controller.
- In the maximum mode, there may be more than one microprocessor in the system configuration.

Minimum-Mode Interface Signals

The minimum-mode signals can be divided into the following basic groups:

- Address/Data bus
- Status signals
- Control signals
- Interrupt signals
- Interface Signals



Address/Data Bus:

- The address bus is 20 bits long and consists of signal lines A_0 (LSB) through A_{19} (MSB). However, only address lines A_0 through A_{15} are used when accessing I/O.
- The data bus lines are multiplexed with address lines. For this reason, they are denoted as AD_0 through AD_{15}. Data line D_0 is the LSB.

Status Signals:

- The four most significant address lines A_{16} through A_{19} of the 8086μP are multiplexed with status signals S_3 through S_6. These status bits are output on the bus at the same time that data are transferred over the other bus lines.
- Bit S4 and S3 together from a 2 bit binary code that identifies which of the 8086 internal segment registers are used to generate the physical address as shown in the table below.
- Status line S5 reflects the status of the IF. The last status bit S6 is always at the logic 0 level.

S_4	S_3	Segment Register
0	0	Extra
0	1	Stack
1	0	Code / none
1	1	Data

Control Signals:

- When Address latch enable (**ALE**) is logic 1 it signals that a valid address is on the bus. This address can be latched in external circuitry on the 1-to-0 edge of the pulse at **ALE**.
- **M/$\overline{\text{IO}}$** (memory/IO) tells external circuitry whether a memory or I/O transfer is taking place over the bus. Logic 1 signals a memory operation and logic 0 signals an I/O operation.
- **DT/$\overline{\text{R}}$** (data transmit/receive) signals the direction of data transfer over the bus. Logic 1 indicates that the bus is in the transmit mode (i.e., data are either written into memory or to an I/O device). Logic 0 signals that the bus is in the receive mode (i.e., reading data from memory or from an input port).
- The bank high enable (**$\overline{\text{BHE}}$**) signal is used as a memory enable signal for the most significant byte half of the data bus, D_8 through D_{15}.
- **$\overline{\text{WR}}$** (write) is switched to logic 0 to signal external devices that valid output data are on the bus.
- **$\overline{\text{RD}}$** (read) indicates that the MPU is performing a read of data off the bus. During read operations, one other control signal, **$\overline{\text{DEN}}$** (data enable), is also supplied. It enables external devices to supply data to the microprocessor.
- The **READY** signal can be used to insert wait states into the bus cycle so that it is extended by a number of clock periods. This signal is supplied by a slow memory or I/O subsystem to signal the MPU when it is ready to permit the data transfer to be completed.

Interrupt Signals:

- Interrupt request (INTR) is an input to the 8086μP that can be used by an external device to signal that it needs to be serviced. Logic 1 at INTR represents an active interrupt request.
- When the MPU recognizes an interrupt request, it indicates this fact to external circuits with logic 0 at the interrupt acknowledge (INTA) output.
- On the 0-to-1 transition of non-maskable interrupt (NMI), control is passed to a non-maskable interrupt service routine at completion of execution of the current instruction. NMI is the interrupt request with highest priority and cannot be masked by software.
- The RESET input is used to provide a hardware reset for the MPU. Switching RESET to logic 0 initializes the internal registers of the MPU and initiates a reset service routine.

DMA Interface Signals:

The Direct Memory Access (DMA) interface of the 8086 minimum mode consist of the HOLD and HLDA signals.

- When an external device wants to take control of the system bus, it signals the MPU by switching HOLD to the logic level 1.
- When in the hold state, lines AD0 through AD15, A16/S3 through A19/S6, **$\overline{\text{BHE}}$, M/$\overline{\text{IO}}$, DT/$\overline{\text{R}}$, $\overline{\text{WR}}$, $\overline{\text{RD}}$, $\overline{\text{DEN}}$** and **INTR** are all put in the high-Z state. The MPU signals external devices that it is in this state by switching **HLDA** to 1.