Lecture # 38

- x86 Instruction Encoding Part-3
- Direct Memory Access (DMA) – Definition
- Instruction Set Architecture & CISC vs RISC (Discussion)

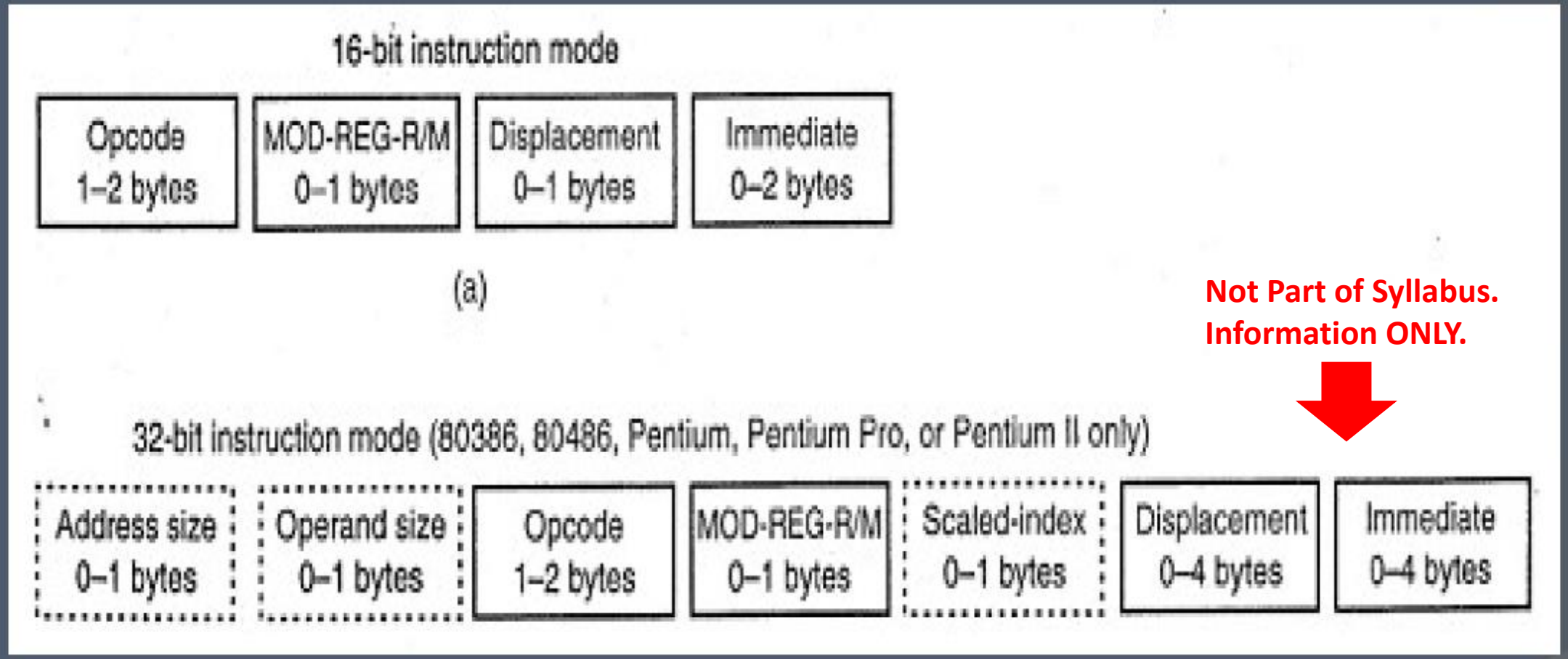
Instruction Encoding

Intel µP Instruction Encoding and Decoding

- **Encoding** of instruction must includes:
  - Opcode
  - Operands
  - Addressing information
- **Encoding** is process representing entire instruction as a **binary value**
  - Number of bytes needed depends on how much information must be encoded .
- Instructions are encoded by **assembler:**
  - **.OBJ** file (link, then loaded by loader)
- Instructions are **decoded** by processor during execution cycle

8086 – 80486 Instruction Format

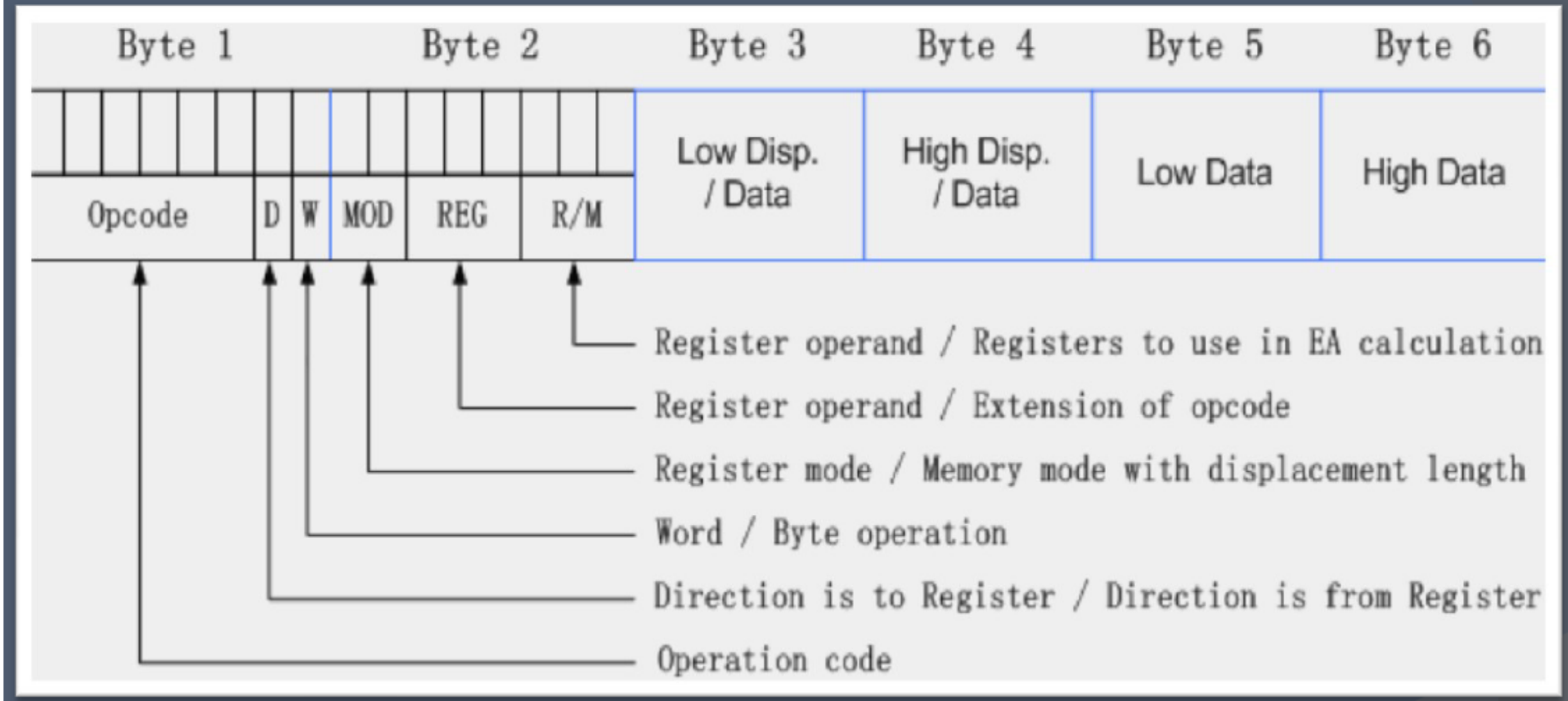
Intel µP Instruction Encoding and Decoding



- **Figure 1** – The format of the 8086 – 80486 Instruction a) 16-bit instruction, b) 32-bit instruction

8088/8086 Instruction Format

Intel µP Instruction Encoding and Decoding



Examples (Encoding)

Intel µP Instruction Encoding and Decoding

- Determine the equivalent machine code of the following assembly code:

1. MOV DX, AX
2. MOV DX, [BX + DI + 1234h]
3. ADD AX, 1023
4. SUB DX, 1234h
5. AND [BX + 12h], AX
6. MOV EAX, [EBX + 4*ECX]
7. MOV [5267h], DH
8. MOV CS, AX
9. MOV DS, AX
10. MOV AX, [BX]

Examples (Decoding)

Intel µP Instruction Encoding and Decoding

- Given the following machine code (hex code), determine the equivalent assembly instruction for each.

1. 8B923412h
2. 81C17856h
3. 2C26h
4. 8B163412h
5. 20D8h

Using Mod and R/M Fields

mod	*r/m*	*observation*
11		no memory operand
00	110	memory operand: offset in direct addressing mode
00	code of [Reg]/[Reg1+Reg2]	memory operand: offset = [Reg] or [Reg1] + [Reg2]
01	code of [Reg]/[Reg1+Reg2]	memory operand: offset = [Reg] or [Reg1] + [Reg2] + 1 byte-displacement
10	code of [Reg]/[Reg1+Reg2]	memory operand: offset= [Reg] or [Reg1] + [Reg2] + 1 word-displacement

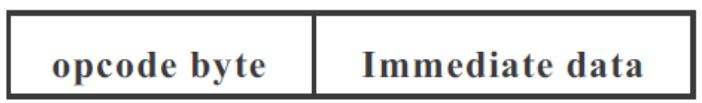
Exercise

'Mod r/m' Byte	Mod	Register	r/m	Remarks
D1	**11**	**010**	**001**	**No operand in memory.**
8E	**10**	**001**	**110**	**One operand in memory; offset = one word displacement + contents of register BP.**
0C				
1E				
18				
C2				
17				
91				

Single Immediate Data

When the only operand of an instruction is an immediate data, the machine language instruction is the opcode followed by that immediate data. For example, the code for **RETN 8** is **C2 08 00**, where **C2** is the opcode and **0008** is the immediate data. Note that a 16-bit immediate data is specified with its low and high byte swapped.

Register, Immediate Operands (with no 'mod r/m' byte)



Using the information in Appendix 2, we have the following machine language instruction:

Assembly Language	Opcode	Machine Language
ADD AX, 7	05	05 07 00

Encoding One-Operand Instructions

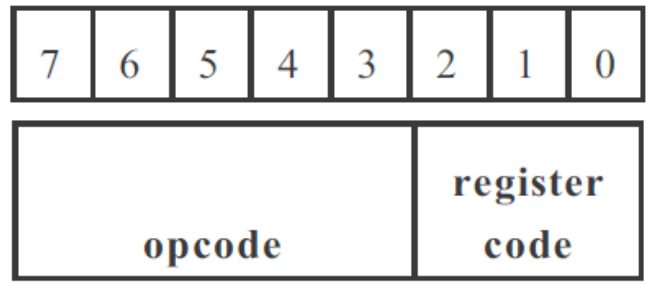
The operand of a one-operand instruction is either in a register, in memory, or as an immediate data. Their formats are discussed as follows:

Single Operand in a Register

One-operand instructions that use register addressing mode have two encoding patterns: a **short form** which consists of just one byte, and a more **general form** which consists of the opcode byte followed by the 'mod r/m' byte.

The short form is used when the register operand is a 16-bit register, whereas the general form is used in any other situation.

a) Short Form for 16-bit Registers that are not Segment Registers



Note that the machine language instruction that corresponds to a register may be obtained by adding to the opcode byte (in which bits 0, 1, and 2 are set to zero) the code of that register. Using the information in Appendix 2, we have the following machine language instructions:

Assembly Language	Machine Language		
PUSH AX	50h + 00	or	50h
PUSH BX	50h + 03	or	53h
INC AX	40h + 00	or	40h
INC BX	40h + 03	or	43h
INC SI	40h + 06	or	46h

Transfer of Control Instructions and Relative Addressing

Immediately after an instruction is fetched from the main memory and before it is executed, the instruction pointer register IP is incremented by the length of that instruction so that its contents is the offset of the next instruction to be fetched from the memory. However, the Intel 8086 processor also has instructions generally referred to as *transfer of control instructions*, that alter the order in which instructions are fetched from the memory by writing a new offset into register IP.

There are two types of transfer of control instructions: those that only affect the contents of register IP, and those that also affect the contents of the code segment register CS in addition to affecting the contents of register IP.

opcode byte	Relative Address

The **relative address** is computed as follows:

<Offset> - (Offset of current instruction + length of current instruction)

or **<Offset> - (Offset of next instruction)**

Using the information in Appendix 2, we have the following machine language instructions:

Offset	Assembly Language	Opcode	R. Address	Machine Language
015A	JL 016Eh	7C	016E - 015C	7C 12
015C			= 12	
0170	JMP 02FFh	E9	02FF - 0173	E9 8C 01
0173			= 01 8C	