Lec-8.

XLAT → Performs table lookup.

```
Data segment
    HA_table  DB '0123456789ABCDEF'
```
means lookup table for hex digits.

```
    Ha_digit DB 7
```
means input hex digit (example 7).

```
    asc_digit  DB ?
```
means output ascii chracter.

```
main proc far
    mov  BX, offset HA_table
```
means BX points to the start of the table.

```
    mov  AL, h_digit
```
means. AL contains the index (4byte position in the table.)

```
    XLAT  ;  AL ← [BX + AL]
    MOV  asc_digit, AL.
main endp.
```

LEA (Load effective Address).

LEA ⇒ destination, source.

msg. LEA, DX, msg.

msg = msg is a memory label (a variable or string stored in memory)

LEA loads the offset address of msg into register DX.

MOV AX, [BX]	The data stored at the memory location	AX ← value stored at address in BX
LEA. AX [BX + SI + 5]	The offset address of the memory operand	AX ← [BX + SI + 5]

MOV fetches the value from memory.
Lea fetches the address of that memory location.

```
data segment
 var DB 10h
data ends
code segment
 mov BX, offset var ; BX = address of var
 mov AL,[BX] ; AL = value of var, AL = 10h.
 LEA DX, var ; DX = address of var
code ends
```

LDS/LES.

LDS : load DS and a GPR

LES : load ES and a GPR.

Each instruction copies 4 bytes (2 words) from memory.

The first word or 2 bytes goes into the destination register.

The next word or 2 bytes goes into the DS or ES

LDS BX, [4326H].
offset address

DS = 1000H ; Data segment register

Physical address = DS x 10h + offset address.

1st 2 byte into register BX
- BL ⟵ [DS x 10h + 4326h]
- BH ⟵ [DS x 10h + 4327h]

DS ⟵ [DS x 10h + 4328h] : [DS x 10h + 4329h]
↳ second two bytes goes into DS.

inc/dec:

```
MOV  CX, 5      ; CX = 5.
inc  CX         ; CX = 6.
dec  CX         ; CX = 5.

MOV DL, 0Ah     ; DL = 10
Inc DL          ; DL = 11
dec DL          ; DL = 10
```