COAL LECTURE 3

Virtual machine → for 16 bit environment

The command MOUNT C /home/name/ee213 in DOSBox would attempt to mount the directory /home/name/ee3 on your local system as the C drive within the DOS environment of DOSBox. This means that the files and folders within /home/name/ee3 would be accessible as if they were located on the C drive within DOSBox.

setup.sh

```
#!/bin/bash
# setting up a 16-bit environment for learning

sudo apt install dosbox
doxbox
mount c /home/nam/ee213

nasm c.asm -o c.com
afd c.com
```

```
[org 0x0100]

; start of code

mov  ax, 5              ; move the constant 5 into register ax
mov  bx, 10

add  ax, bx             ; add value of bx into the value of ax

mov  bx, 15             ; add constant 15 into the value of bx
add  ax, bx

mov  ax, 0x4c00         ; exit ..
int  0x21               ; .. is what the OS should do for me
```

. ax → register

. A bit is a single binary digit, a 0 or 1. A nibble is defined as half of a byte. A byte is two nibbles. A word is two bytes. We also have the double-word/DWORD, which is two words, and quad-word/QWORD. In Intel/AMD architecture, that works out to the following values:

Bit: 1 bit

Nibble: 4 bits

Byte: 8 bits

Word: 16 bits

DWORD: 32 bits

QWORD: 64 bits

1. **`mov ax, 5`:** This line moves the constant value 5 into the 16-bit register `ax`. Here, `ax` is being used as a general-purpose register to hold a value.
2. **`mov bx, 10`:** Similarly, this line moves the constant value 10 into the 16-bit register `bx`.
3. **`add ax, bx`:** This line adds the value stored in register `bx` to the value stored in register `ax` and stores the result back in register `ax`. This instruction effectively performs `ax = ax + bx`.
4. **`mov bx, 15`:** This line moves the constant value 15 into register `bx`, overwriting the previous value.
5. **`add ax, bx`:** Again, this line adds the value stored in register `bx` (which is now 15) to the value stored in register `ax` and stores the result back in register `ax`.(30)

MOV AX →OP CODE → B8 , 0500 →5 (0500 OPERANDS B8 OPERATION CODE OF AX)

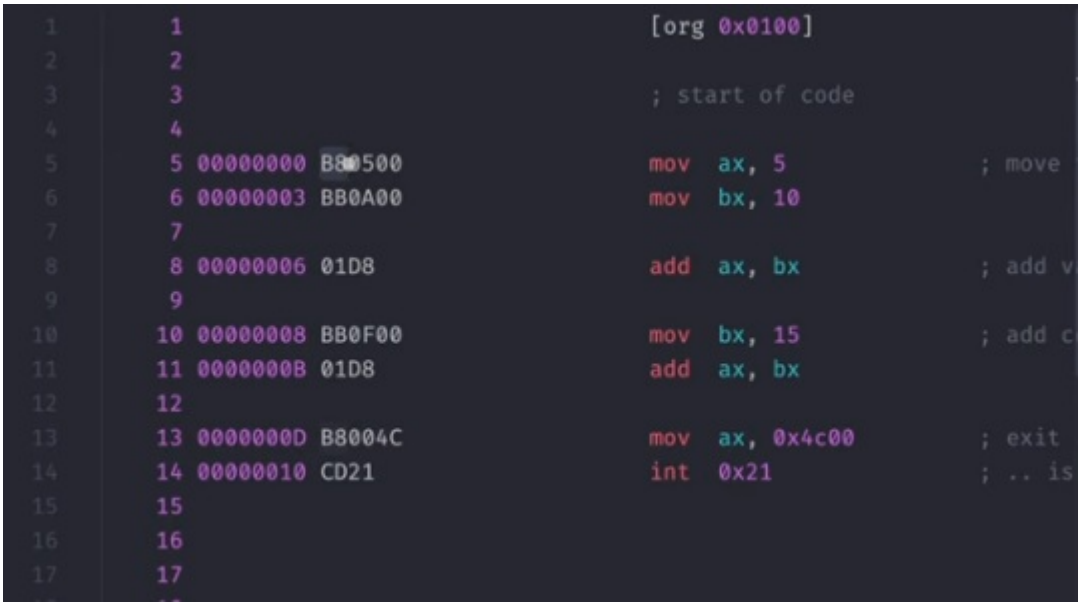
MOV BX → BB 0A00 → 10

WHERE MY PROGRAM LOAD IN RAM → BASE ADDRESS

SO 000000000 IS NOT RAM 0 ADDRESS BUT MY PROGRAM

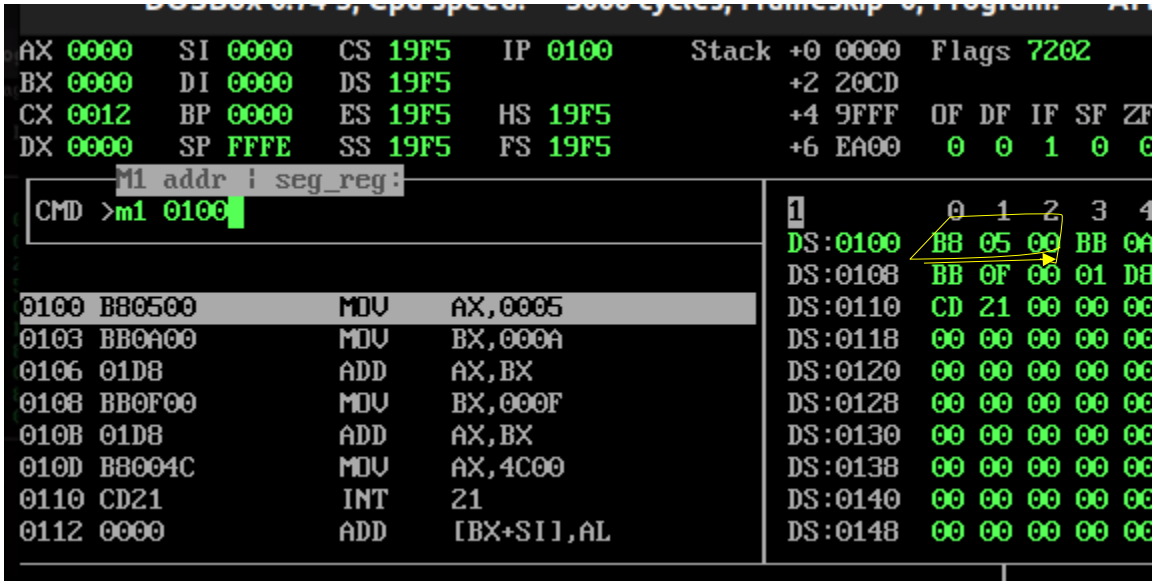
In this case, `MOV` instructions typically take up 3 bytes - 1 for the opcode and 2 for the operand), the next instruction would be stored at the next memory address after the current instruction.

if `MOV AX, 5` is stored at memory address `0x00`, then `MOV BX, 10` would be stored at memory address `0x03` (assuming `MOV AX, 5` takes up bytes `0x00`, `0x01`, and `0x02`).

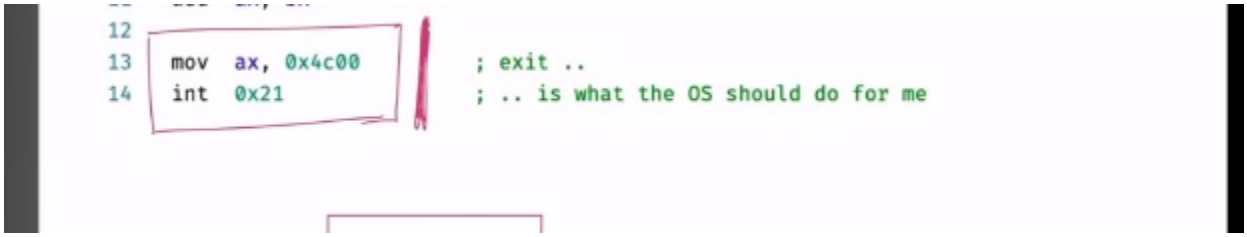


Dosbox start addresses from 100 thats why we write it org 0x0100 my program if load at 1000 then it starts from 1100.

When using DOSBox, memory addresses start from 100h. Therefore, if a program is loaded at address 1000h, it starts executing from 1100h, as indicated by the `ORG` directive in assembly language.



.m1 0100 showing the same address here B80500



-Application execution via OS:

- *Our application relies on the operating system for all tasks.*

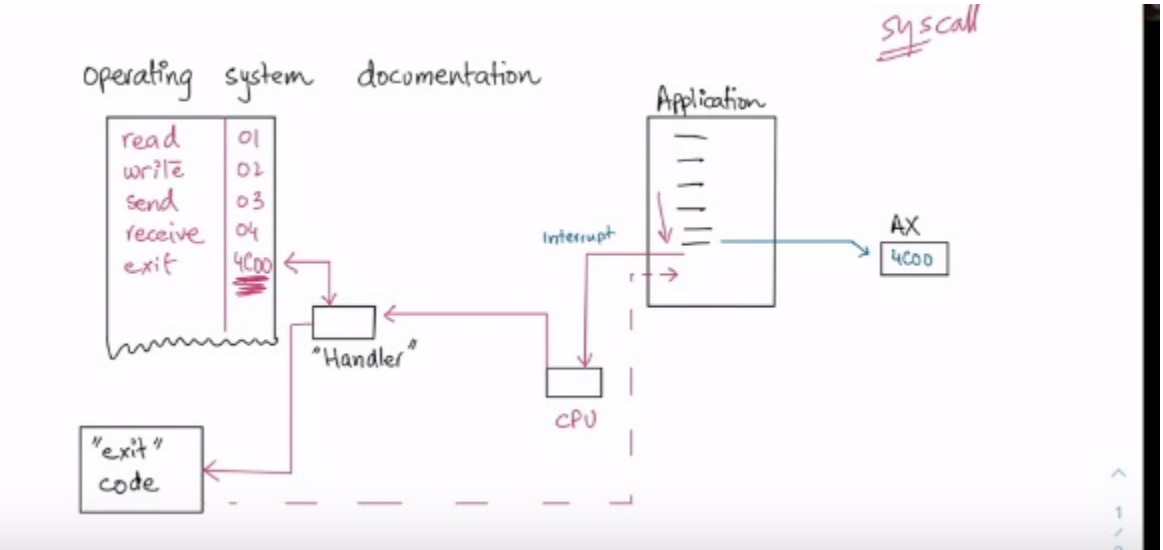
- *System Call Handling:*

  - *Occasionally, the OS interrupts our execution and assigns CPU tasks.*

- *Interrupt 0x21:*

  - *This interruption prompts the CPU to perform specific work.*

- *Exit Operation:- Tasked with the work code 4C00, which signifies program termination.*



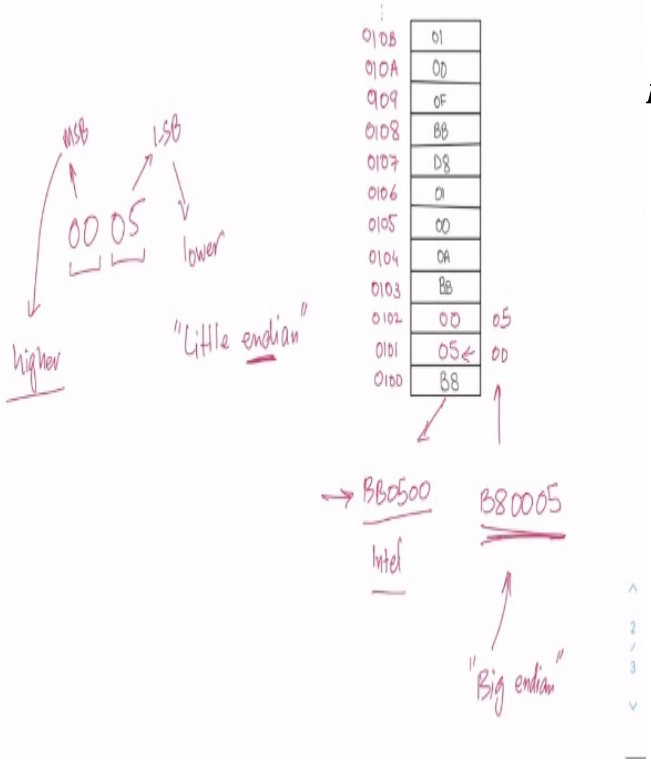
- ***Data Storage in Memory:***

  - According to Intel's convention, when storing data like `AX` with a value of `0005h`:

    - The LSB (Least Significant Byte) `05` occupies the lower memory address.

    - The MSB (Most Significant Byte) `00` occupies the higher memory address.

- Hence, in memory, `AX` with the value `0005h` would be stored as `0500h`.



Little Endian and Big Endian:

- *Little Endian:*
  - *In little-endian systems, the least significant byte (LSB) is stored at the lowest memory address.*
  - *The most significant byte (MSB) is stored at the highest memory address.*
- *Big Endian:*
  - *In big-endian systems, the most significant byte (MSB) is stored at the lowest memory address.*
  - *The least significant byte (LSB) is stored at the highest memory address.*