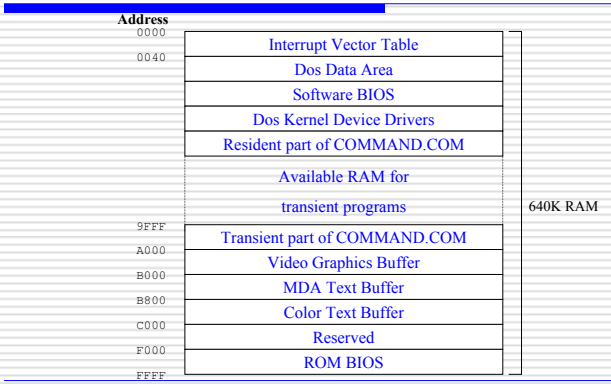
Executing Computer Instructions

Objective

- DEBUG Program
  - DEBUG Commands
  - Rules of DEBUG Commands
  - DEBUG Display
  - Viewing Memory Locations
- Machine and Assembly Language
  - Keying in program instructions and data
  - Execute program instructions

Map of the First MB of Memory



Registers

- High-speed storage directly inside the CPU (CPU has internal data bus that is twice as wide as external data bus)
  - Segment registers
  - Pointer registers
  - General-purpose registers
  - Index registers
  - Flags register

Segment Registers

- Used as base locations (addresses) for program instructions, data, and stack
  - CS (code segment) -- holds the base location for all executable instruction (code) in a program
  - DS (data segment) -- default base location for variables
  - SS (stack segment) -- holds the stack base location
  - ES (extra segment) -- additional base location

Pointer Registers

Contain the offsets of data instructions

- IP (instruction pointer)
  - contains the offset address of the next instruction to be executed
  - associated with CS register
- BP (base pointer)
  - contains an assumed offset from SS register
  - used by subroutine to locate variables
- SP (stack pointer)
  - contains the offset of the top of the stack
  - SS and SP combine to form a complete address

Instruction Pointer Register

Segment address in CS	**39B40h**
Offset address in IP	**+514h**
Address of next instruction	**3A054h**

General Purpose Registers

- Data registers -- used for arithmetic and data movement
- can be addressed as either an 8-bit or a 16-bit
- AX (accumulator) -- arithmetic operations
- BX (base) -- hold address of a procedure or variable
- CX (counter) -- a counter for repeating or looping instructions
- DX (data) -- a special role in multiply and divide operations

Index Registers

Contain the offsets of data instructions

- SI (source index) -- source string is pointed to by the SI register
- DI (destination index) -- acts as the destination for string movement instruction

Register Names

8-Bit	16-Bit	32-Bit
AL	AX	EAX
AH		
BL	BX	EBX
BH		
CL	CX	ECX
CH		
DL	DX	EDX
DH		
	SI	ESI
	DI	EDI
	BP	EBP
	SP	ESP

Flags Register

O = Overflow -- indicate overflow of the left most bit following arithmetic

D = Direction -- determine left or right direction for moving or comparing data

I = Interrupt -- indicate that all interrupts to be processed or ignored

T = Trap -- permit operation of the processor in single-step-mode

S = Sign -- indicate the resulting sign of an arithmetic operation, 0 (negative), 1 (positive)
Z = Zero -- indicate the resulting sign of an arithmetic or comparison operation, 0 (nonzero), 1 (zero) result
A = Auxiliary carry -- contain a carry out of bit 3 on 8-bit data

P = Parity -- indicate even or odd parity of a low-order 8-bit data operation

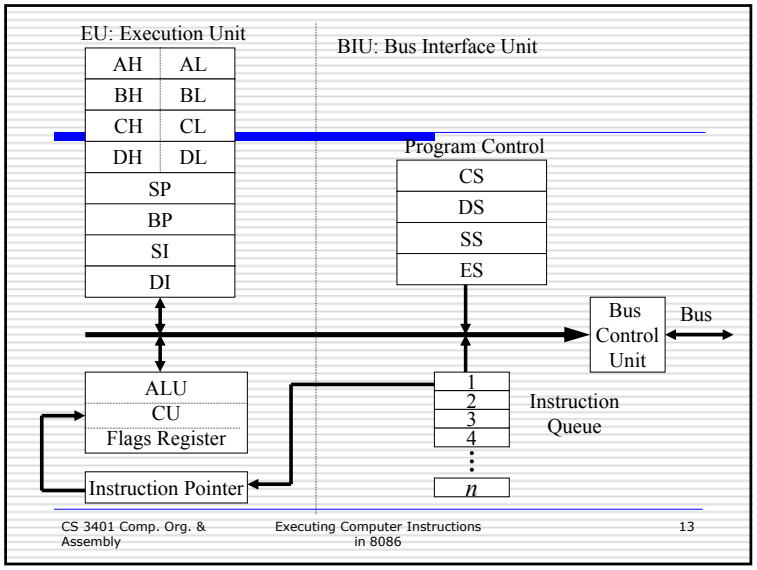
C = Carry -- contain the leftmost bit

x = undefined

Flags Registers

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

x x x x O D I T S Z x A x P x C



DEBUG Program

- DEBUG Commands
- Rules of DEBUG Commands
- DEBUG Display
- Viewing Memory Locations

DEBUG Program

DEBUG is a DOS program that is used to

- view memory
- enter machine code or assembly code in memory
- enter data
- trace code execution
- single step tracing

DEBUG Commands

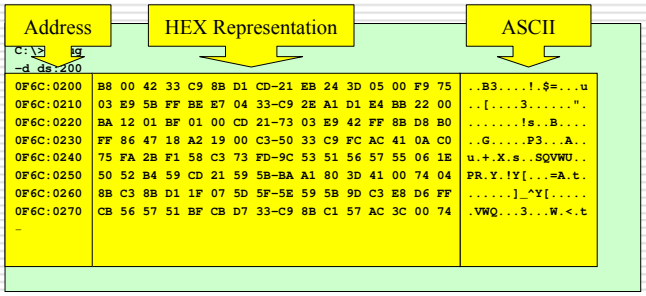
A: Assemble symbolic instructions into machine code
D: Display contents of memory at a specific address
E: Enter data/instructions into memory
U: Unassemble machine code into symbolic code
G: Run the program
T: Trace the execution of one instruction
P: Proceed or execute a set of instructions
R: display contents of Registers
W: Save a program onto disk
Q: Quit

Rules of DEBUG Commands

- Not case insensitive
- use colon to specify segment and offset
- use hexadecimal numbers
- use a space to separate parameters in a command

The DEBUG Display

Display contents of memory at offset 200H in DS using D command



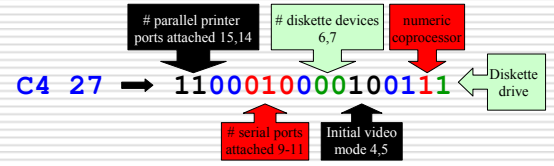
Viewing Memory Locations

- System equipment
- memory size
- serial number and copyright notice
- ROM BIOS Date
- Model ID

System Equipment

Location of equipment status word in the BIOS data area is at 410h-411h

```
-d 40:10
0040:0010  27 C4 00 80 02 20 00 00-00 00 32 00 32 00 33 04   '.... ....2.2.3.
0040:0020  0D 1C 64 20 20 39 34 05-30 0B 3A 27 31 02 30 0B   ..d  94.0.:'1.0.
```



Based Memory Size

Size of based memory is at location 413h and 414h

```
-d 40:13
0040:0010           80 02 20 00 00-00 00 22 00 22 00 33 04      .. ....".".3.
0040:0020  0D 1C 08 0E 08 0E 30 0B-30 0B 3A 27 30 0B 0D 1C   ......0.0.:'0...
```

02 80 ⟷ 640

ROM BIOS Date

ROM BIOS manufacture date begins at location FFFF5h

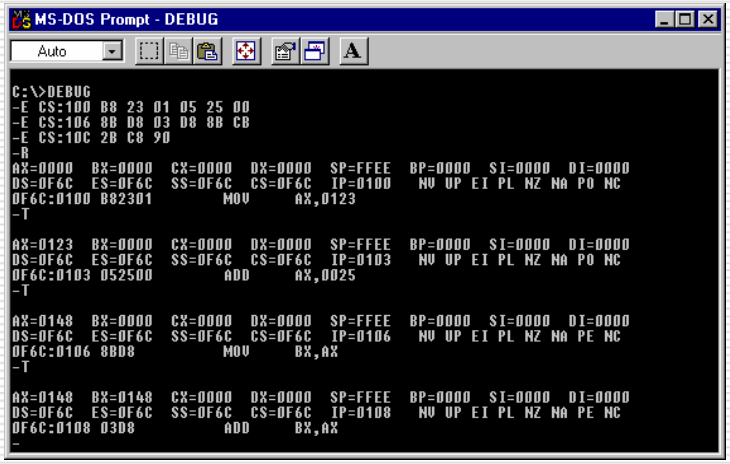
```
-d ffff:5
FFFF:0000                  31 30 2F-31 36 2F 39 36 00 FC 56        10/16/96..V
FFFF:0010  00 00 00 56 44 49 53 4B-33 2E 33 80 00 01 01 00   ...VDISK3.3.....
```

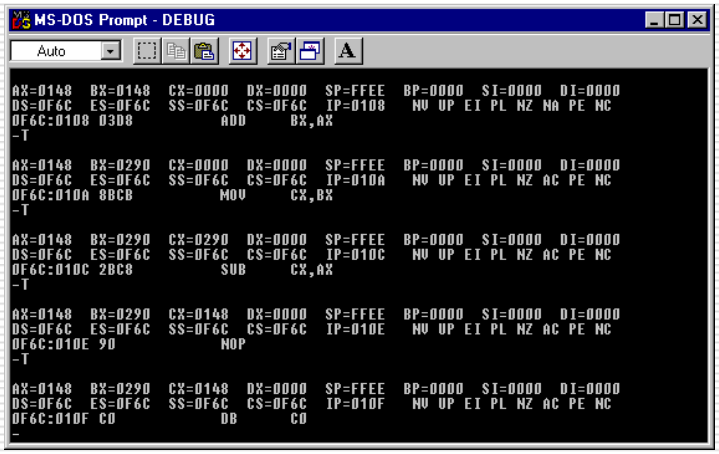

Machine and Assembly Language

- Key in program instructions
- Execute program instructions
- Save a program

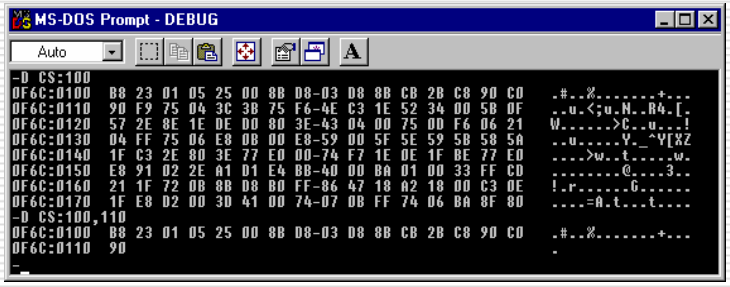
Machine Language Example

Machine Code	Assembly Code
B82301	MOV AX,0123
052500	ADD AX,0025
8BD8	MOVE BX,AX
03D8	ADD BX,AX
8BCB	MOV CX,BX
2BC8	SUB CX,AX
90	NOP





Code Segment Display



Debug Operations

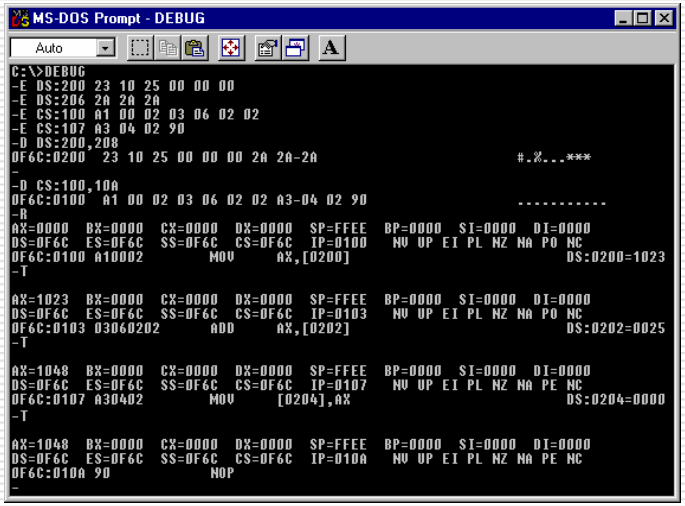
- Keying in program instructions
- Executing program instructions
- Displaying memory contents
- Correcting an entry

Machine Language Example with Defined Data

DS Offset	Hex Contents
0200h	2301h
0202h	2500h
0204h	0000h
0206h	2A2A2Ah

Program Instructions with Defined Data

Machine Code	Assembly Code
A10002	MOV AX,[0200]
03060202	ADD AX,[0202]
A30402	MOV [0204],AX
90	NOP



Assembly Language Example

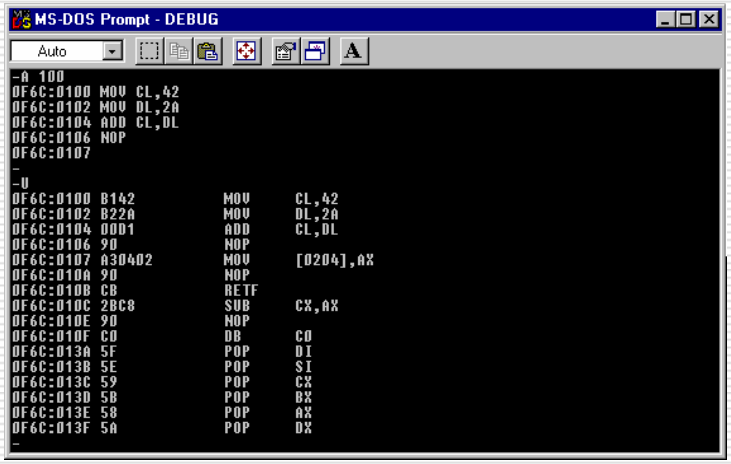
Assemble Command (A)

Use assemble command to key in assembly instructions

```
MOV CL,42
MOV DL,2A
ADD CL,DL
NOP
```

Unassemble Command (U)

Unassemble command displays the machine code for assembly language instructions

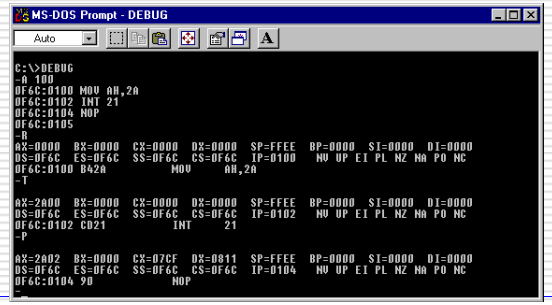


Another Example

```
MOV AX,5
ADD AX,10
ADD AX,20
MOV [0102],AX
```

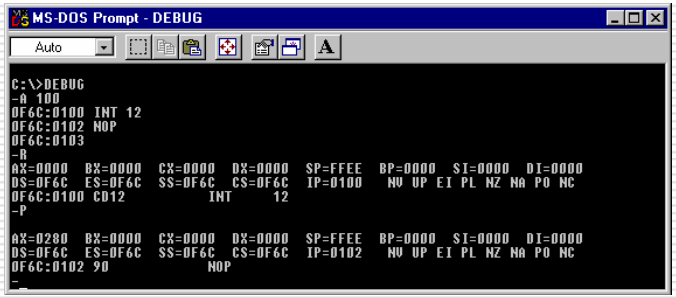

INT (Interrupt) Instruction

Current date - **INT 21h**

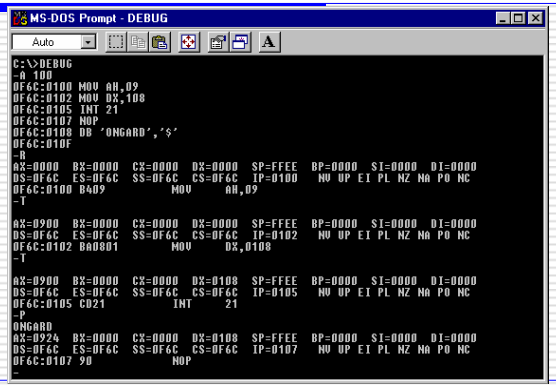


Size of Memory

INT 12h



Display with INT



INT for Keyboard Input