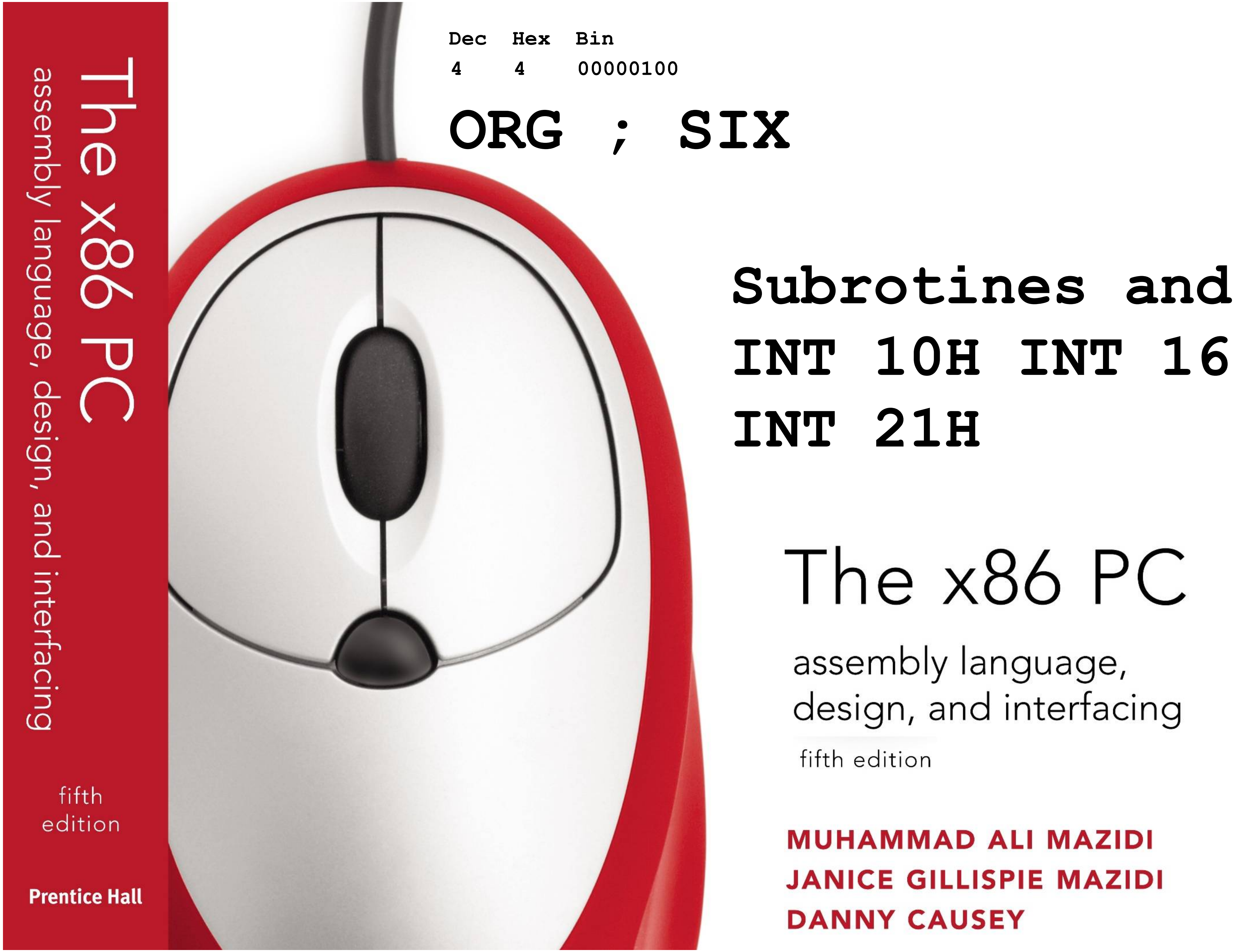
Dec	Hex	Bin
4	4	00000100

ORG ; SIX

Subrotines and INT 10H INT 16 INT 21H



OBJECTIVES
this chapter enables the student to:

- 8086 Subroutines
- Use INT 10H function calls to:
  - Clear the screen.
  - Set the cursor position.
  - Write characters to the screen in text mode.
  - Draw lines on the screen in graphics mode.
  - Change the video mode.
- Use INT 16H function calls
- Use INT 21H function calls to:
  - Input characters from the keyboard.
  - Output characters to the screen.
  - Input or output strings.

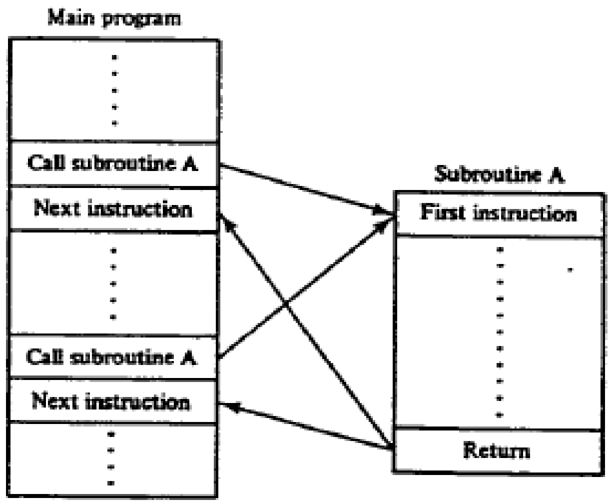
Subroutines and Subroutine Handling Functions

✓ A subroutine is a special segment of a program that can be called for execution from any point in the program

✓ A RET instruction must be included at the end of the subroutine to initiate the return sequence to the main program environment

Examples. **Call 1234h**
Call BX
Call [BX]

Two calls
- intrasegment
- intersegment



(a)

Mnemonic	Meaning	Format	Operation	Flags Affected
CALL	Subroutine call	CALL operand	Execution continues from the address of the subroutine specified by the operand. Information required to return back to the main program such as IP and CS are saved on the stack.	None

(b)

Operand
Near-proc
Far-proc
Memptr16
Regptr16
Memptr32

(c)

Figure 6–20 (a) Subroutine concept. (b) Subroutine call instruction. (c) Allowed operands.

Calling a NEAR proc

- ✓ The CALL instruction and the subroutine it calls are in the same segment.
- ✓ Save the current value of the IP on the stack.
- ✓ load the subroutine's offset into IP (nextinst + offset)

Calling Program | Subroutine | Stack

Calling Program	Subroutine
Main proc	sub1 proc
001A: call sub1	0080: mov ax,1
001D: inc ax	…
.	ret
Main endp	sub1 endp

1ffd	1D
1ffe	00
1fff	(not used)

Calling a FAR proc

- ✓ The CALL instruction and the subroutine it calls are in the "Different" segments.
- ✓ Save the current value of the CS and IP on the stack.
- ✓ Then load the subroutine's CS and offset into IP.

Calling Program | Subroutine | Stack

```
            Main proc                    sub1 proc far
1FCB:001A: call far ptr sub1             4EFA:0080: mov ax,1
1FCB:001F: inc ax                        ….
            …                            ….
            …                            ret (retf opcode generated)
            Main endp                    sub1 endp
```

1ffb	1F	IP
1ffc	00	IP
1ffd	CB	SEG
1ffe	1F	SEG
1fff	N/A	

PEARSON
The x86 PC
Assembly Language, Design, and Interfacing
By Muhammad Ali Mazidi, Janice Gillespie Mazidi and Danny Causey

Example on Far/Near Procedure Calls

```
0350:1C00 Call FarProc
0350:1C05 Call NearProc
0350:1C08 nop
```

1ff0	08
1ffa	1C
1ffb	05
1ffc	1C
1ffd	50
1ffe	03
1fff	X

Nested Procedure Calls

A subroutine may itself call other subroutines.

Example:

```
        main proc
000A    call subr1
000C    mov ax,…
…
        main endp
```

```
        subr1  proc
0030    nop
        …
        call subr2
0040    ret …
        subr1 endp
```

```
        subr2  proc
0050    nop
        …
        call subr3
0060    ret …
        subr2 endp
```

```
        subr3  proc
0070    nop
        …
0079    nop
007A    ret
        subr3 endp
```

Q: show the stack contents at 0079?

1ff0	60
1ffa	00
1ffb	40
1ffc	00
1ffd	0c
1ffe	00
1fff	X

Do NOT overlap Procedure Declarations

Push and Pop Instructions

To save registers and parameters on the stack
```
PUSH XX
PUSH YY
PUSH ZZ
```

Push S (16/32 bit or Mem)
(SP) ← (SP) - 2
((SP)) ← (S)

Main body of the subroutine
```
.
.
.
.
.
.
```

To restore registers and parameters from the stack
```
POP   ZZ
POP   YY
POP   XX
```

Return to main program
```
RET
```

Pop D (16/32 bit or Mem)
(D) ← ((SP))
(SP) ← (SP) + 2

80x86 Interrupts

- An interrupt is an event that causes the processor to suspend its present task and transfer control to a new program called the interrupt service routine (ISR)
- There are three sources of interrupts
  - Processor interrupts
  - Hardware interrupts generated by a special chip, for ex: 8259 Interrupt Controller.
  - Software interrupts
- Software Interrupt is just similar to the way the hardware interrupt actually works!. The INT Instruction requests services from the OS, usually for I/O. These services are located in the OS.
- INT has a range 0→ FFh. Before INT is executed AH usually contains a function number that identifies the subroutine.

80x86 Interrupts

- Each interrupt must supply a type number which is used by the processor as a pointer to an interrupt vector table (IVT) to determine the address of that interrupt's service routine
- Interrupt Vector Table: CPU processes an interrupt instruction using the interrupt vector table (This table resides in the lowest 1K memory)
- Each entry in the IVT=segment+offset address in OS, points to the location of the corresponding ISR.
- Before transferring control to the ISR, the processor performs one very important task
  - It saves the current program address and flags on the stack
  - Control then transfers to the ISR
  - When the ISR finishes, it uses the instruction IRET to recover the flags and old program address from the stack
- Many of the vectors in the IVT are reserved for the processor itself and others have been reserved by MS-DOS for the BIOS and kernel.
  - 10 -- 1A are used by the BIOS -> So today's lecture INT10h and INT16h are BIOS Int
  - 20 -- 3F are used by the MS-DOS kernel -> INT21h is DOS Int

INT

INT operates similar to Call

- Processor first pushes the flags
- Trace Flag and Interrupt-enable flags are cleared
- Next the processor pushes the current CS register onto the stack
- Next the IP register is pushed

Example: What is the sequence of events for INT 08? If it generates a CS:IP of 0100:0200. The flag is 0081H.

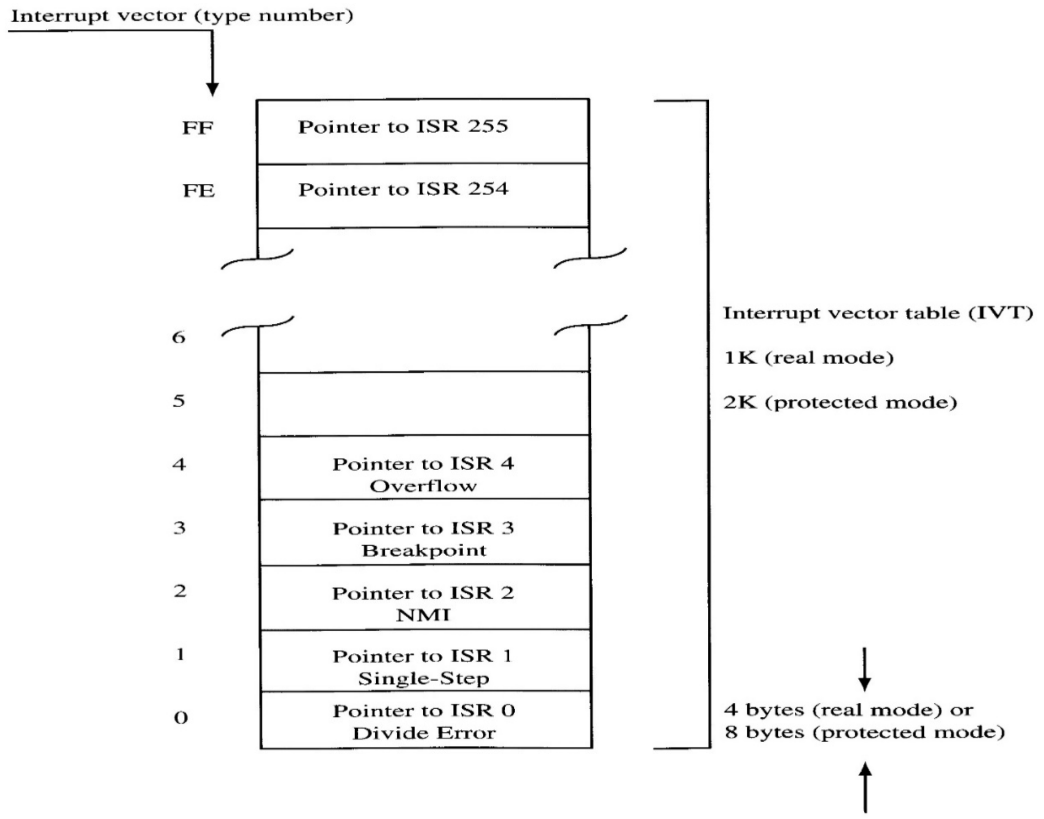
SP-6	00
SP-5	02
SP-4	00
SP-3	01
SP-2	81
SP-1	00

SP initial

MEMORY / ISR table		
00020	10	IP
00021	00	IP
00022	80	SEG
00023	05	SEG

0580:0010

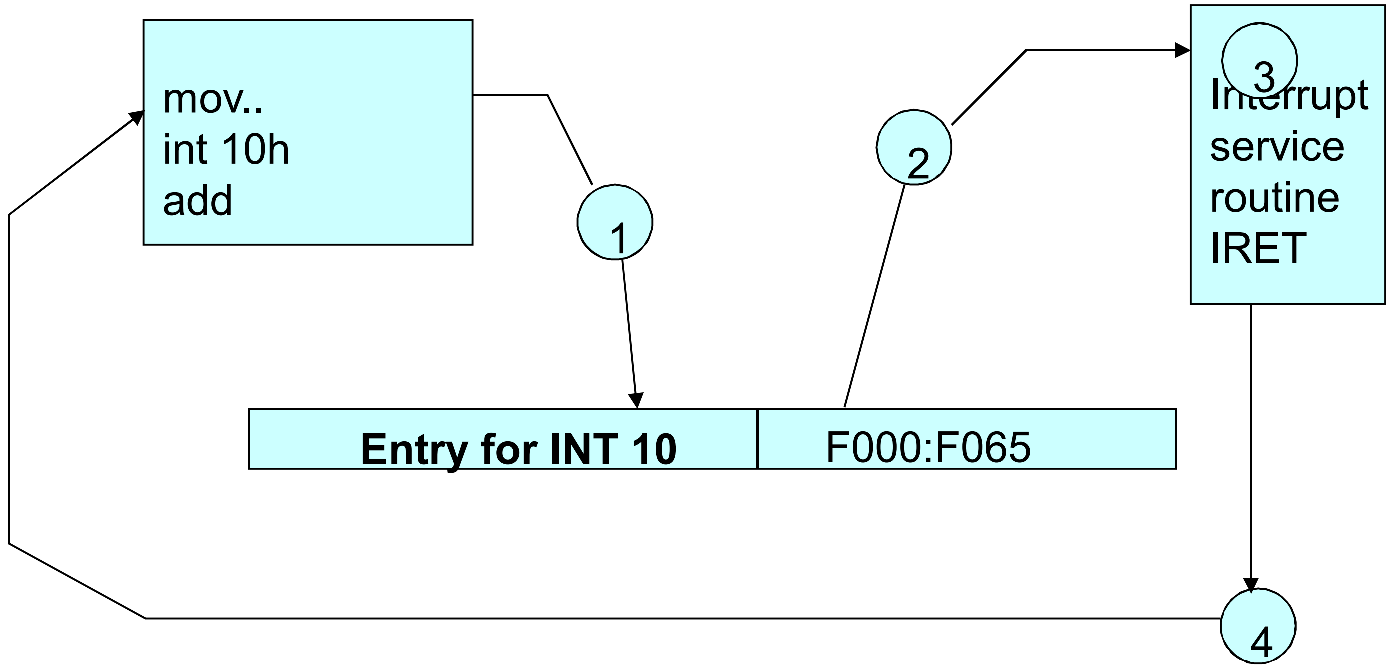
Interrupt Vector Table



Processor	Pointer Size	IVT Location
Real Mode	4 bytes	Address 00000000–000003FF
Protected Mode	8 bytes	Anywhere in Physical Memory

80x86 Interrupts

- The number after the mnemonic tells which entry to locate in the table. For example INT 10h requests a video service.



Interrupts

- There are some extremely useful subroutines within BIOS or DOS that are available to the user through the INT (Interrupt) instruction.
- Format:
  - INT xx ; the interrupt number xx can be 00-FFH
  - This gives a total of 256 interrupts
  - Common Interrupts
    - INT 10h Video Services
    - INT 16h Keyboard Services
    - INT 17h Printer Services
    - INT 21h MS-DOS services
  - Before the services, certain registers must have specific values in them, depending on the function being requested.

4.0: INT 10H and 21H

- The INT instruction is somewhat like a FAR call.
  - Saves CS:IP and the flags on the stack and goes to the subroutine associated with that interrupt.

```
INT  xx;the interrupt number xx can be 00 - FFH
```

- In x86 processors, 256 interrupts, numbered 00 to FF.
  - INT 10H and INT 21H are the most widely used with various functions selected by the value in the AH register.

4.1: BIOS INT 10H PROGRAMMING

- INT 10H subroutines are burned into the ROM BIOS.
  - Used to communicate with the computer's screen video.
    - Manipulation of screen text/graphics can be done via INT 10H.
- Among the functions associated with INT 10H are changing character or background color, clearing the screen, and changing the location of the cursor.
  - Chosen by putting a specific value in register AH.

4.1: BIOS INT 10H PROGRAMMING
changing the video mode (AH=00)

- To change the video mode, use INT 10H with AH = 00 and AL = video mode.

01h: 40x25 Text, 16 colors
03h: 80x25 Text, 16 colors

4.1: BIOS INT 10H PROGRAMMING
graphics: modes

- Text mode of 80 × 25 characters.
  - A total of 2K (80 × 25 = 2000) for characters, plus 2K for attributes, as each character has one attribute byte.
    - Each screen (frame) takes 4K, which results in CGA supporting a total of four pages of data, where each page represents one full screen.
- In this mode, 16 colors are supported.
  - To select this mode, use AL = 03 for mode selection in INT 10H option AH = 00.

attribute byte in CGA text mode

- CGA mode is the common denominator for all color monitors, as S all color monitors & video circuitry are upwardly compatible,
  - CGA attribute byte bit definition is as shown:

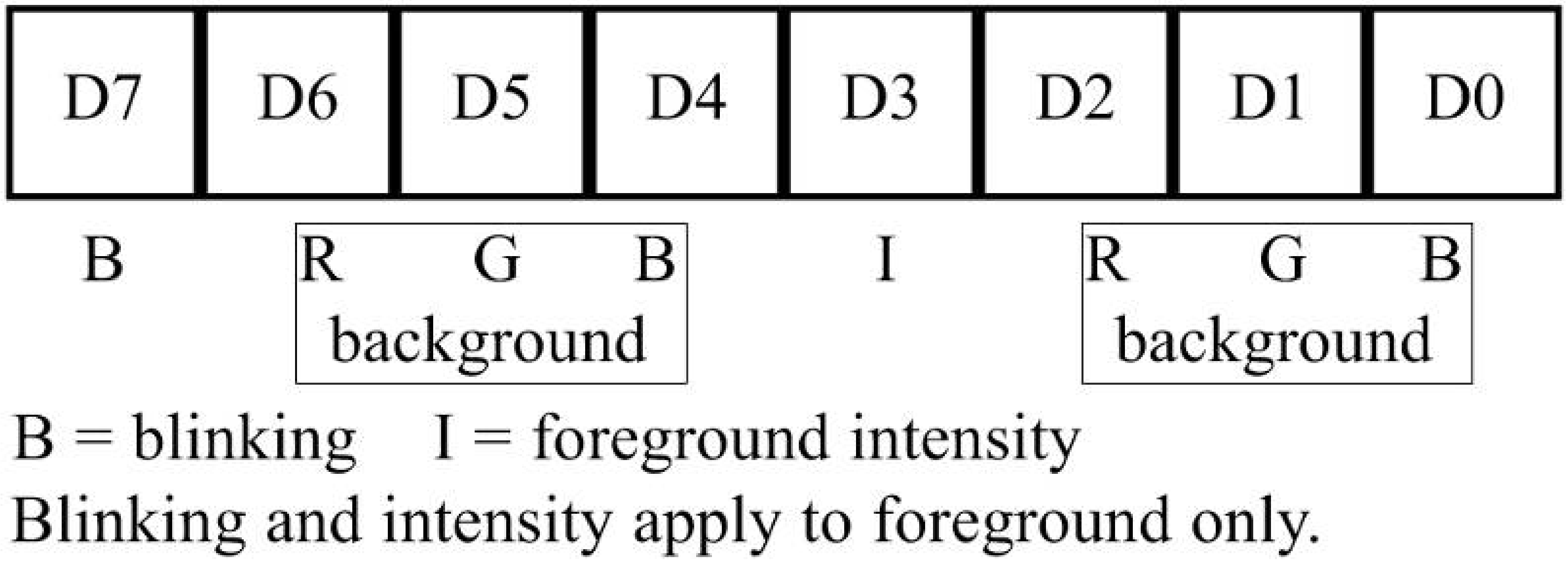


Figure 4-3 CGA Attribute Byte

4.1: BIOS INT 10H PROGRAMMING attribute byte in CGA text mode (AH=09h)

- The background can take eight different colors by combining the prime colors **red**, **blue**, and **green**.
- The foreground can be any of 16 different colors by combining **red**, **blue**, **green**, and **intensity**

Example 4-4

Write a program that puts 20H (ASCII space) on the entire screen. Use high-intensity white on a blue background attribute for characters.

Solution:

```
MOV   AH,00    ;SET MODE OPTION
MOV   AL,03    ;CGA COLOR TEXT MODE OF 80×25
INT   10H
MOV   AH,09    ;DISPLAY OPTION
MOV   BH,00    ;PAGE 0
MOV   AL,20H   ;ASCII FOR SPACE
MOV   CX,800H  ;REPEAT IT 800H TIMES
MOV   BL,1FH   ;HIGH-INTENSITY WHITE ON BLUE
INT   10H
```

Example 4-4 shows the use of the attribute byte in CGA mode.

4.1: BIOS INT 10H PROGRAMMING
attribute byte in CGA text mode (AH=09h)

Table 4-1: The 16 Possible Colors

I	R	G	B	Color
0	0	0	0	black
0	0	0	1	blue
0	0	1	0	green
0	0	1	1	cyan
0	1	0	0	red
0	1	0	1	magenta
0	1	1	0	brown
0	1	1	1	white
1	0	0	0	gray
1	0	0	1	light blue
1	0	1	0	light green
1	0	1	1	light cyan
1	1	0	0	light red
1	1	0	1	light magenta
1	1	1	0	yellow
1	1	1	1	high intensity white

Some possible CGA colors and variations.

Binary	Hex	Color effect
0000 0000	00	Black on black
0000 0001	01	Blue on black
0001 0010	12	Green on blue
0001 0100	14	Red on blue
0001 1111	1F	High-intensity white on blue

4.1: BIOS INT 10H PROGRAMMING
monitor screen in text mode

- The monitor screen in the x86 PC is divided into 80 columns and 25 rows in normal text mode.
  - Columns are numbered from 0 to 79.
  - Rows are numbered 0 to 24.

The top left corner has been assigned 00,00, the top right 00,79.

Bottom left is 24,00, bottom right 24,79.

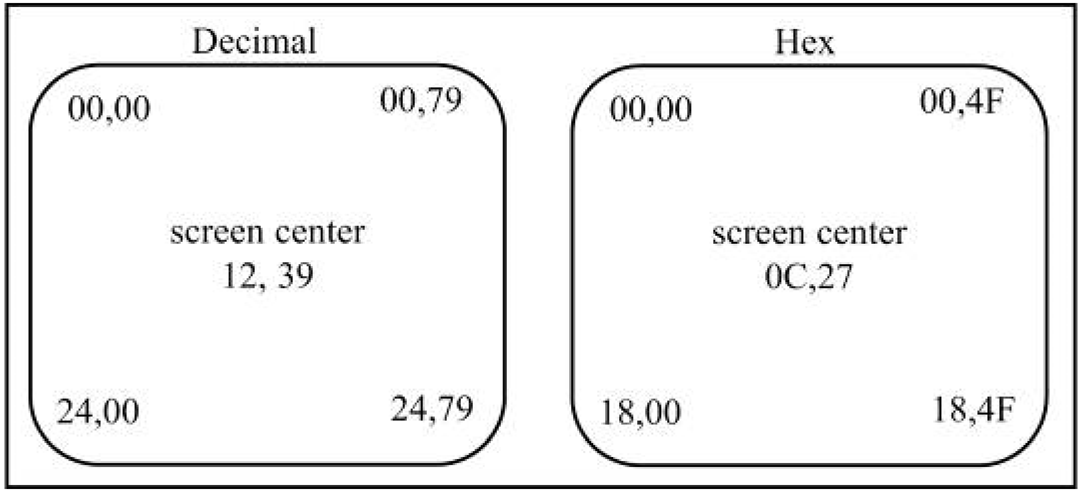


Figure 4-1 Cursor Locations (row, column)

INT 10h AH = 06h

- AL = Number of lines to be scrolled up (AL = 00h will clear the window).
- BH = Color attribute for blank lines. In text mode, this corresponds to the attribute byte. In VGA graphics modes, this is the color number to which all the pixels in the blank lines will be set.
- CH = Top row of window to be scrolled up.
- CL = Leftmost column of window.
- DH = Bottom row of window.
- DL = Rightmost column of window.

4.1: BIOS INT 10H PROGRAMMING

screen clearing with INT 10H function 06H

- To clear the screen using INT 10H, these registers must contain certain values before INT 10H is called:
  - AH = 06, AL = 00, BH = 07, CX = 0000, DH = 24, DL = 79.

```
MOV   AH,06     ;AH=06 to select scroll function
MOV   AL,00     ;AL=00 the entire page
MOV   BH,07     ;BH=07 for normal attribute
MOV   CH,00     ;CH=00 row value of start point
MOV   CL,00     ;CL=00 column value of start point
MOV   DH,24     ;DH=24 row value of ending point
MOV   DL,79     ;DL=79 column value of ending point
INT   10H       ;invoke the interrupt
```

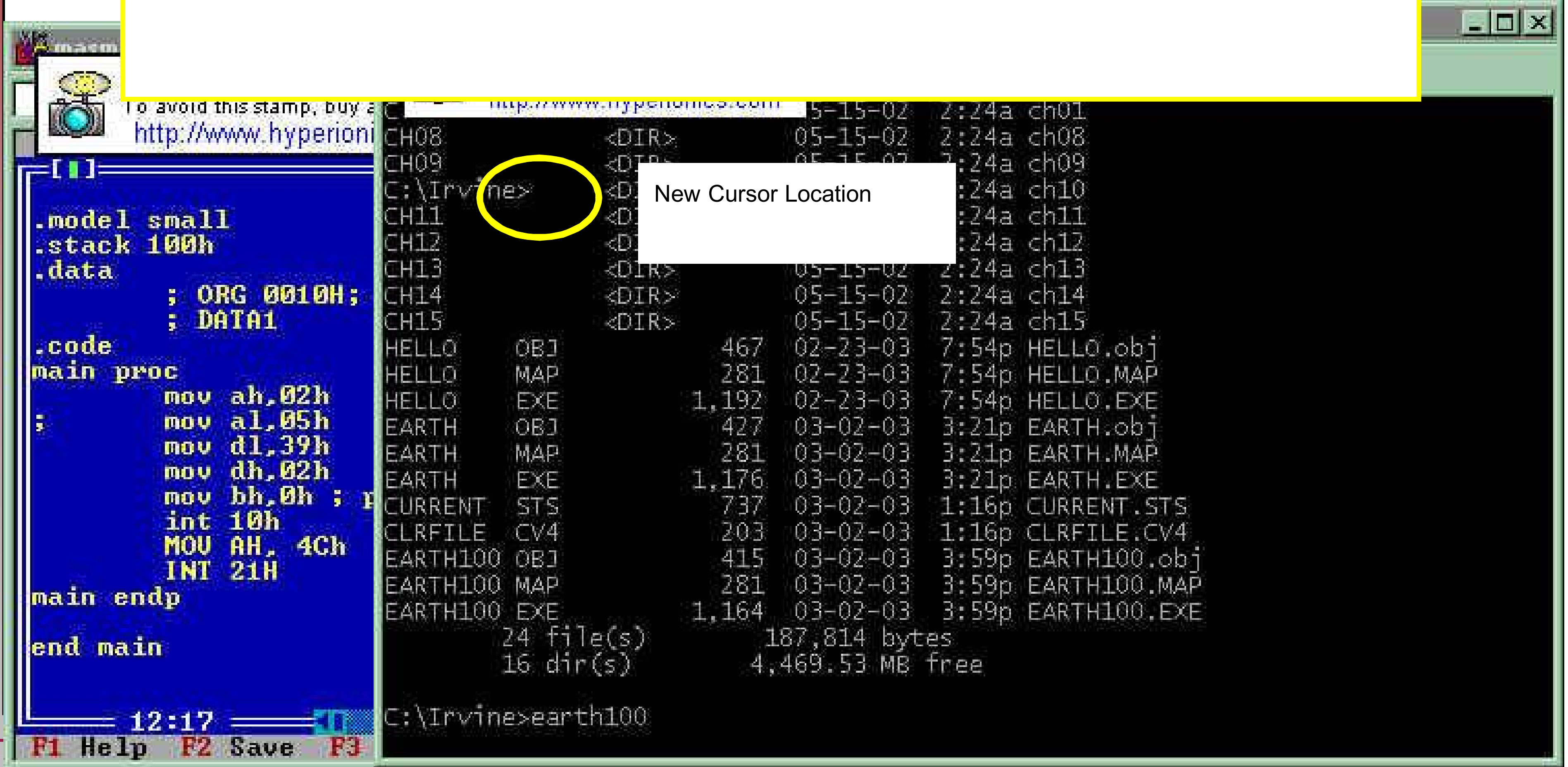
  - Option **AH = 06** calls the scroll function, to scroll upward.
  - **CH** & **CL** registers hold starting row & column.
  - **DH** & **DL** registers hold ending row & column.

4.1: BIOS INT 10H PROGRAMMING AH=02 setting the cursor to a specific location

- INT 10H function AH = 02 will change the position of the cursor to any location.
  - Desired position is identified by row/column values in DX.
    - Where DH = row and DL = column.
- Video RAM can have multiple pages of text.
  - When AH = 02, page zero is chosen by making BH = 00.
- After INT 10H (or INT 21H) has executed, registers not used by the interrupt remain unchanged.

Int 10 AH=02H SET CURSOR POSITION

- **INT 10H function 02**; setting the cursor to a specific location
  - Function AH = 02 will change the position of the cursor to any location.
  - The desired cursor location is in DH = row, DL = column



4.1: BIOS INT 10H PROGRAMMING AH=02 setting the cursor to a specific location

- Example 4-1 demonstrates setting the cursor to a specific location.

Example 4-1

Write the code to set the cursor position to row = 15 = 0FH and column = 25 = 19H.

Solution:

```
MOV   AH,02        ;set cursor option
MOV   BH,00        ;page 0
MOV   DL,25        ;column position
MOV   DH,15        ;row position
INT   10H          ;invoke interrupt 10H
```

4.1: BIOS INT 10H PROGRAMMING AH=03 get current cursor position

- In text mode, determine where the cursor is located at any time by executing the following:

```
MOV    AH,03          ;option 03 of BIOS INT 10H
MOV    BH,00          ;page 00
INT    10H            ;interrupt 10H routine
```

  – After execution of the program, registers DH & DL will have current row and column positions.
    - CX provides information about the shape of the cursor.
  – In text mode, **page 00** is chosen for the currently viewed page.

4.1: BIOS INT 10H PROGRAMMING graphics: pixel resolution & color

- In text mode, the screen is viewed as a matrix of rows and columns of characters.
- In graphics mode, a matrix of horizontal & vertical pixels.
  - Number of pixels depends on monitor resolution & video board.
- Two facts associated with every pixel on the screen must be stored in the video RAM:
  - Location of the pixel and attributes. (color and intensity)
    - The higher the number of pixels and colors, the larger the amount of memory that is needed to store them
    - Memory requirements go up with resolution & number of colors.
  - CGA mode can have a maximum of 16K bytes of video memory due to its inherent design structure.

4.1: BIOS INT 10H PROGRAMMING AH=0Ch
INT 10H and pixel programming

- To address a single pixel on the screen, use INT 10H with AH = 0CH.
  - The X (column) and Y (row) coordinates of the pixel must be known, and vary, depending on monitor resolution.
    - Registers are CX = the column point (the X coordinate) and DX = the row point. (Y coordinate)
  - **To turn the pixel on/off, AL=1 or AL=0 for black and white.**
    - **The value of AL can be modified for various colors.**
- If the display mode supports more than one page, BH = page number.

4.1: BIOS INT 10H PROGRAMMING drawing lines in graphics mode

- To draw a horizontal line, choose row/column values to point to the beginning of the line and increment the column until it reaches the end of the line.
  - To draw a vertical line, increment the vertical value held by the DX register, and keep CX constant.
  - Linear equation $y = mx + b$ can be used for any line.

4.1: BIOS INT 10H PROGRAMMING
drawing lines in graphics mode

Drawing a horizontal line

Example 4-5

Write a program to: (a) clear the screen, (b) set the mode to CGA of 640 × 200 resolution, and (c) draw a horizontal line starting at column = 100, row = 50, and ending at column 200, row 50.

Solution:

```
       MOV   AX,0600H     ;SCROLL THE SCREEN
       MOV   BH,07        ;NORMAL ATTRIBUTE
       MOV   CX,0000      ;FROM ROW=00,COLUMN=00
       MOV   DX,184FH     ;TO ROW=18H,COLUMN=4FH
       INT   10H          ;INVOKE INTERRUPT TO CLEAR SCREEN
       MOV   AH,00        ;SET MODE
       MOV   AL,06        ;MODE = 06 (CGA HIGH RESOLUTION)
       INT   10H          ;INVOKE INTERRUPT TO CHANGE MODE
       MOV   CX,100       ;START LINE AT COLUMN =100 AND
       MOV   DX,50        ;ROW = 50
BACK:  MOV   AH,0CH       ;AH=0CH TO DRAW A LINE
       MOV   AL,01        ;PIXELS = WHITE
       INT   10H          ;INVOKE INTERRUPT TO DRAW LINE
       INC   CX           ;INCREMENT HORIZONTAL POSITION
       CMP   CX,200       ;DRAW LINE UNTIL COLUMN = 200
       JNZ   BACK
```

Int 10 03 GET CURSOR POSITION

- **INT 10H function 03**; get current cursor position

MOV AH, 03
MOV BH, 00
INT 10H

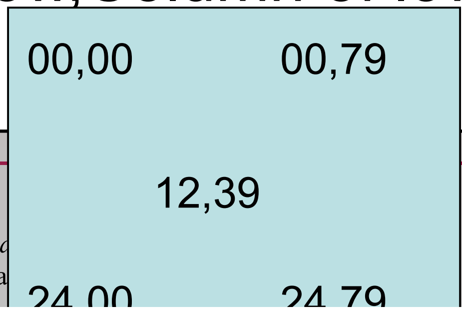
- Registers DH and DL will have the current row and column positions and CX provides info about the shape of the cursor.
- Useful in applications where the user is moving the cursor around the screen for menu selection

INT 10 – AH=06 SCROLL

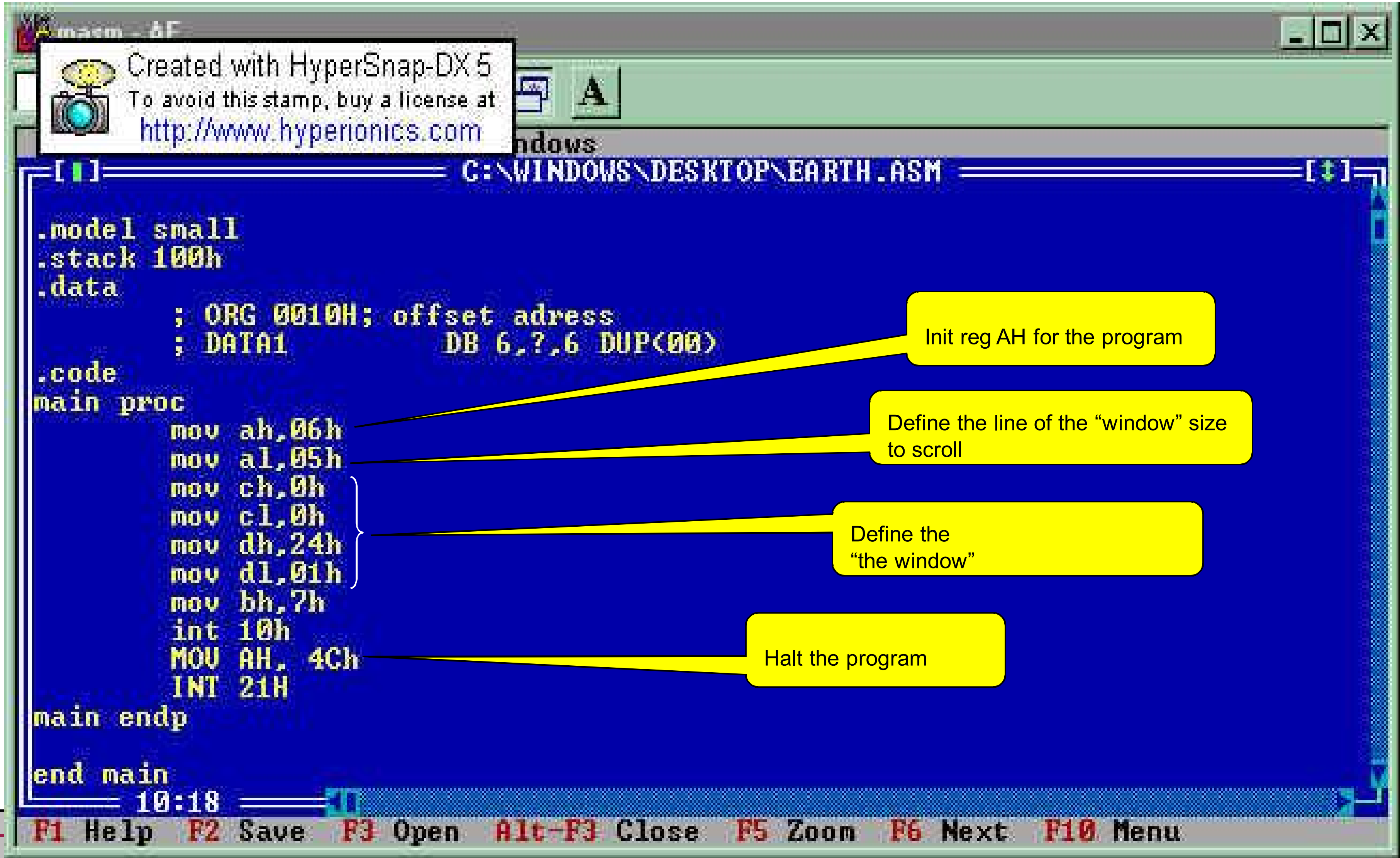
- INT 10H Function 06 (AH = 06) Scroll a screen windows.
  - **Moves the data on the video display up or down**. As screen is rolled the bottom is replaced by a blank line. Rows:0-24 from top, bottom: 0-79 from the left. (0,0) to (24,79). Lines scrolled can not be recovered!
  - AL = number of lines to scroll (with AL=00, window will be cleared)
  - BH = Video attribute of blank rows
  - CH, CL = Row,Column of upper left corner
  - DH, DL = Row,Column of lower right corner

<u>Example:</u> Clear the screen by scrolling it upward with a normal attribute

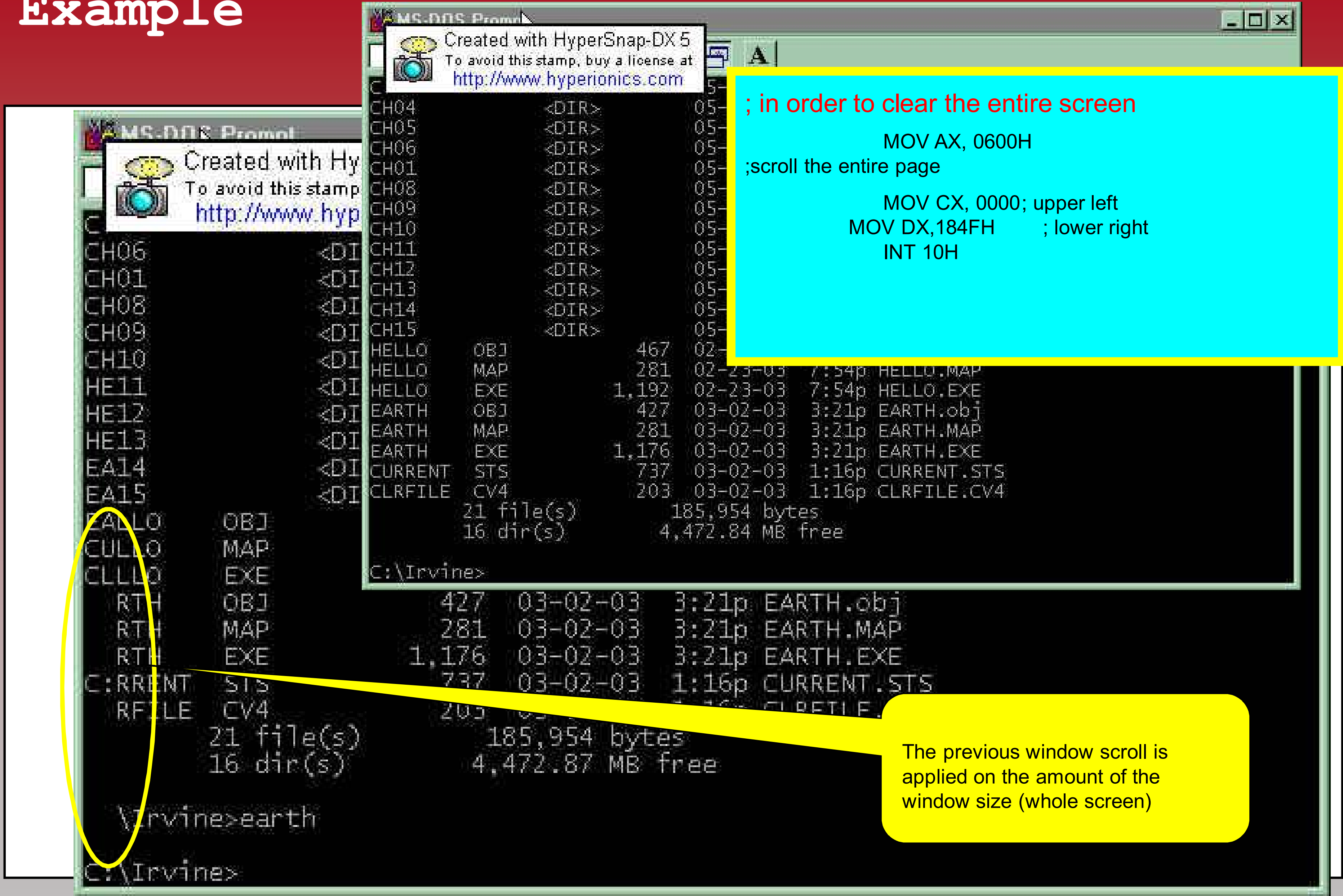
```
mov ah,6h
mov al,0h
mov ch,0h
mov cl,0h
mov dh,24h
mov dl,01h
mov bh,7h
int 10h
```



Example Int10 06



Example



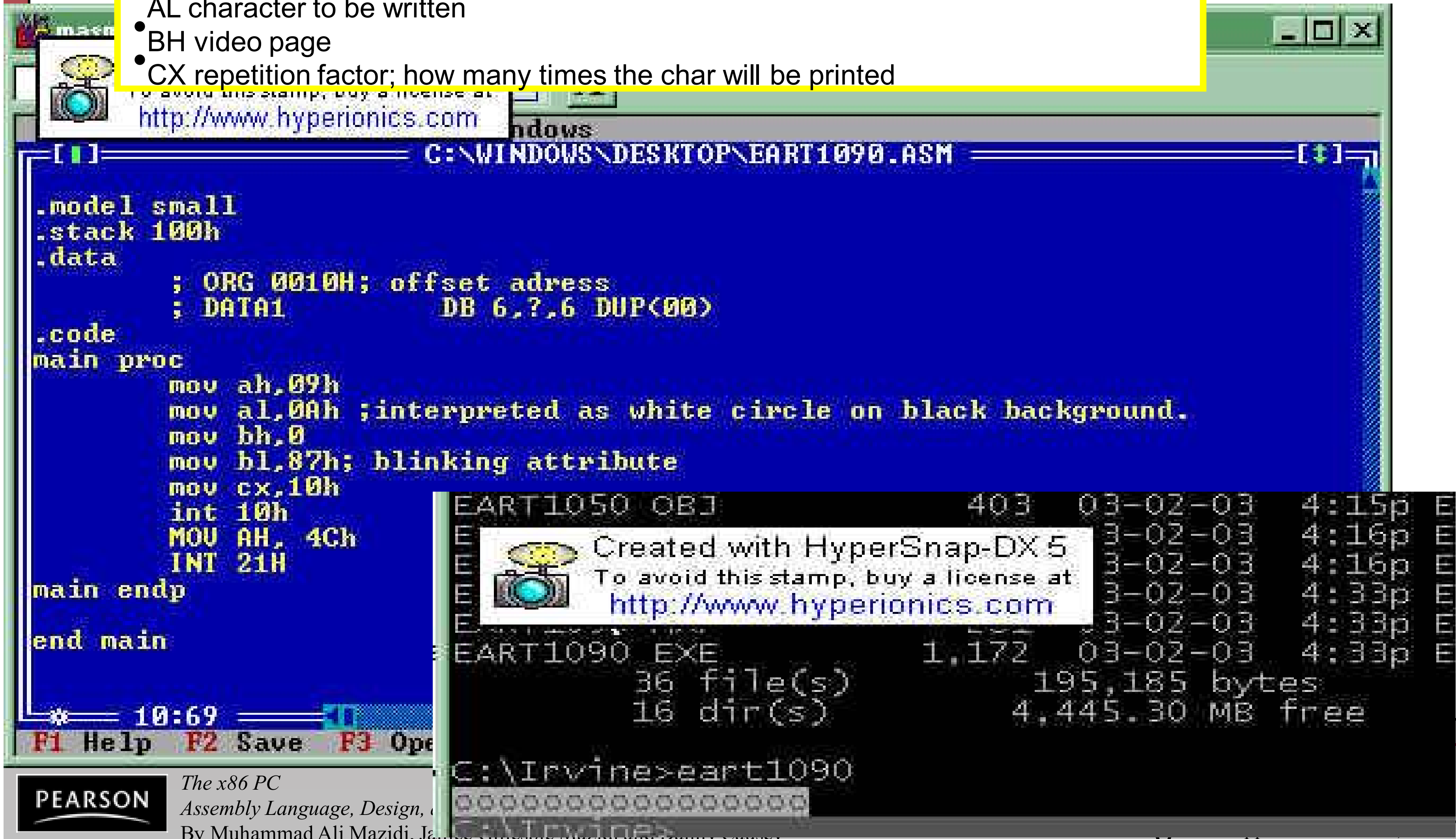
; in order to clear the entire screen

```
        MOV AX, 0600H
;scroll the entire page
        MOV CX, 0000; upper left
    MOV DX,184FH        ; lower right
        INT 10H
```

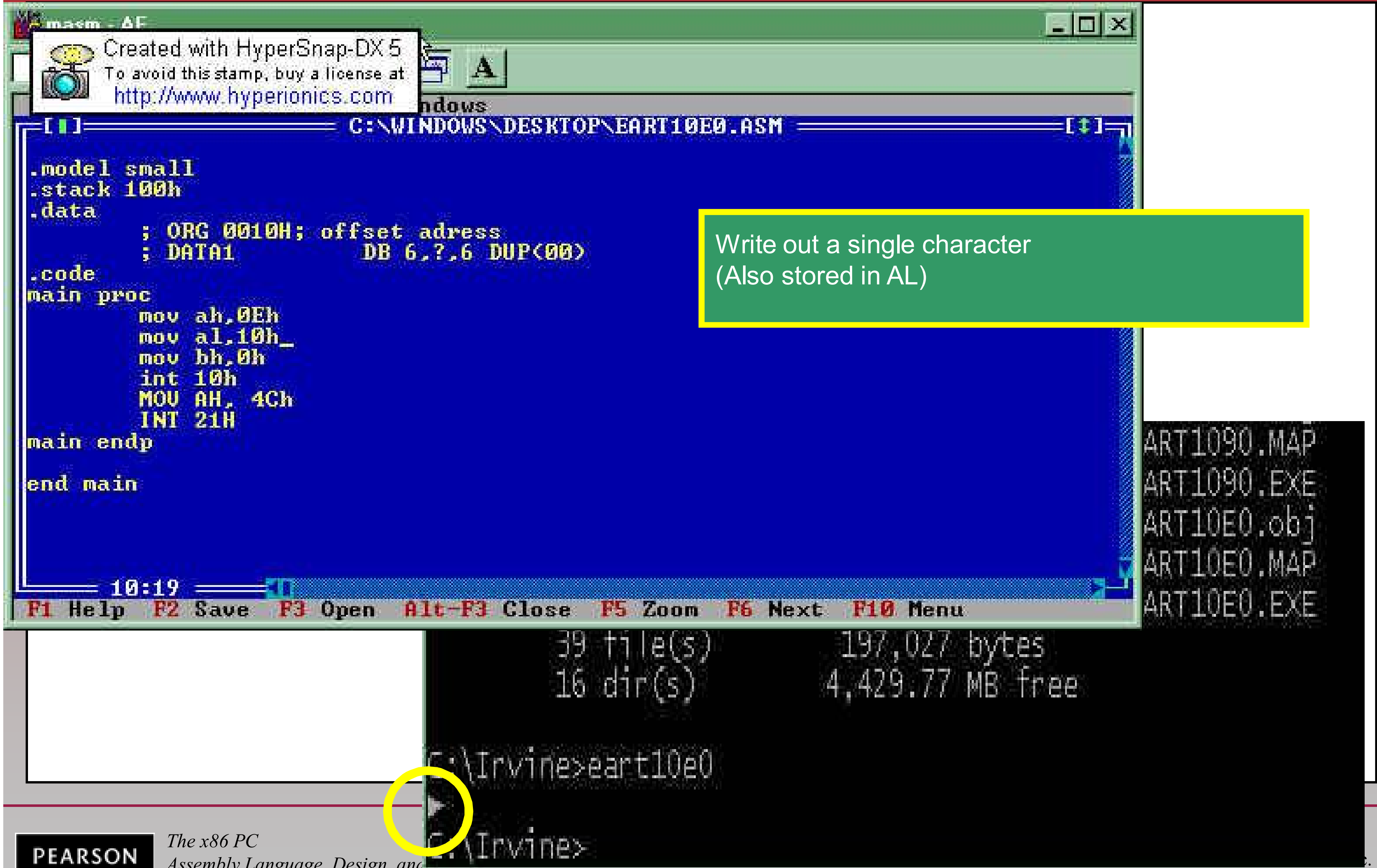
The previous window scroll is applied on the amount of the window size (whole screen)

INT 10 – 0A PRINT CHARACTERS

- Write *one or more* characters at the current cursor position
- This function can display any ASCII character.
- AH function code
- AL character to be written
- BH video page
- CX repetition factor; how many times the char will be printed



Int 10 – 0E PRINT SINGLE CHARACTER



Write out a single character
(Also stored in AL)

INT 16h Keyboard Services

- Checking a key press, we use INT 16h function AH = 01

```
MOV AH, 01
INT 16h
```

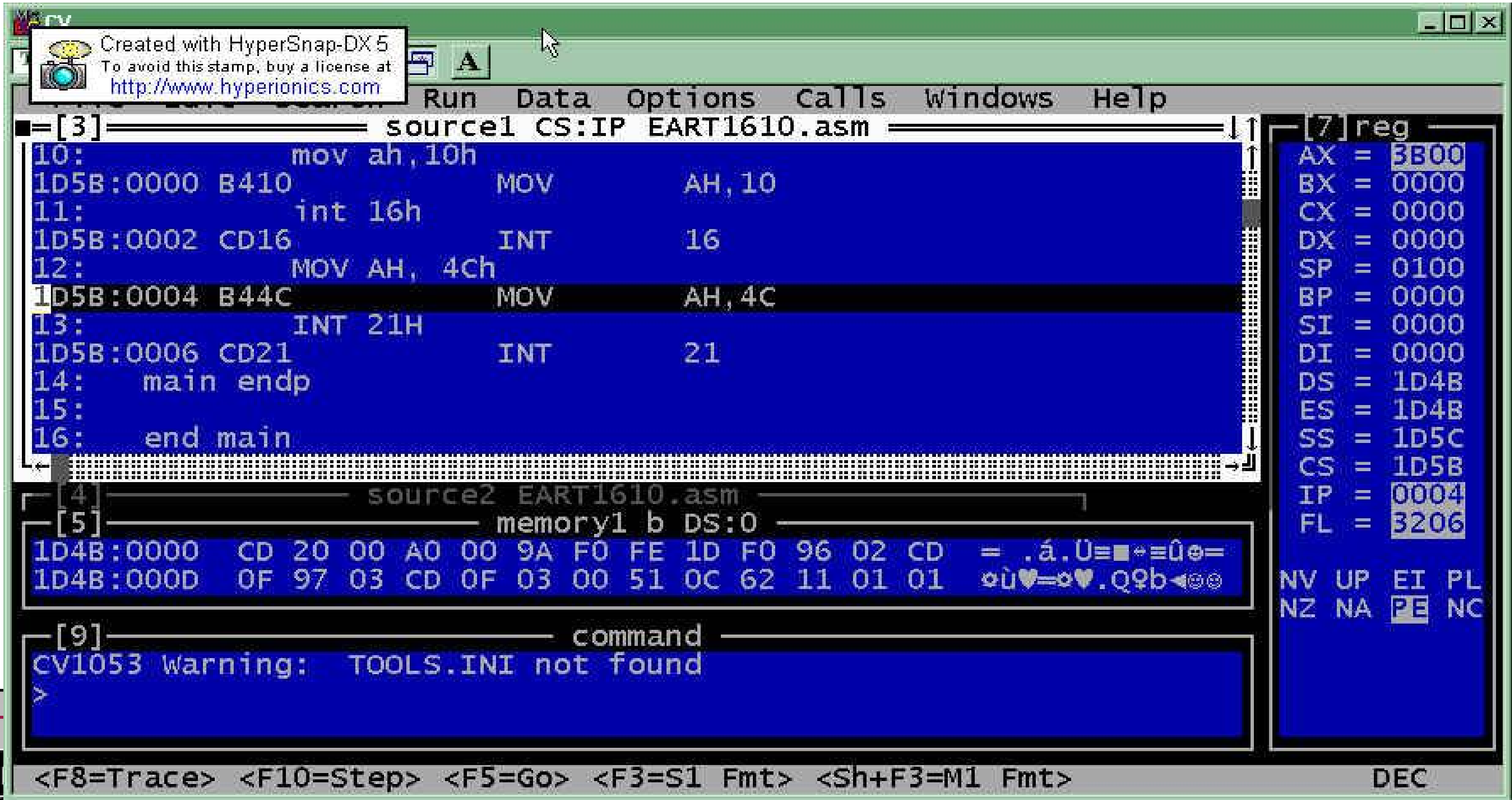
- Upon return, ZF = 0 if there is a key press; ZF = 1 if there is no key press
- Which key is pressed?
- To do that, INT 16h function can be used immediately after the call to INT 16h function AH=01

```
MOV AH,0
INT 16h
```

- Upon return, AL contains the ASCII character of the pressed key

Example INT 16 – 00

- BIOS Level Keyboard Input (more direct)
- Suppose F1 pressed (Scan Code 3BH). AH contains the scan code and AL contains the ASCII code (0).



Example. The PC Typewriter

- Write an 80x86 program to input keystrokes from the PC's keyboard and display the characters on the system monitor. Pressing any of the function keys F1-F10 should cause the program to end.
- Algorithm:
  1. Get the code for the key pressed
  2. If this code is ASCII, display the key pressed on the monitor and continue
  3. Quit when a non-ASCII key is pressed
- INT 16, BIOS service 0 – Read next keyboard character
  – Returns 0 in AL for non-ASCII characters or the character is simply stored in AL
- To display the character, we use INT 10, BIOS service 0E- write character in teletype mode. AL should hold the character to be displayed.
- INT 20 for program termination

Example

```
        MOV DX, OFFSET MES
        MOV AH,09h
        INT 21h ; to output the characters starting from the offset
AGAIN:  MOV AH,0h
        INT 16h; to check the keyboard
        CMP AL,00h
        JZ QUIT ;check the value of the input data
        MOV AH, 0Eh
        INT 10h; echo the character to output
        JMP AGAIN
QUIT:   INT 20h
MES     DB ‘type any letter, number or punctuation key’
        DB ‘any F1 to F10 to end the program”
        DB 0d,0a,0a,’$’
```

Application

4.2: DOS INTERRUPT 21H

- In previous chapters, a fixed set of data was defined in the data segment & results viewed in a memory dump.
  - This section uses information inputted from the keyboard, and displayed on the screen.
    - A much more dynamic way of processing information.
- When the OS is loaded, INT 21H can be invoked to perform some extremely useful functions.
  - Commonly referred to as DOS INT 21H function calls.
    - In contrast to BIOS-ROM based INT 10H.

4.2: DOS INTERRUPT 21H Option 09 outputting a data string the monitor

- INT 21H can send a set of ASCII data to the monitor.
  - Set AH = 09 and DX = offset address of the ASCII data.
    - Displays ASCII data string pointed at by DX until it encounters the dollar sign "$".
- The data segment and code segment, to display the message *"The earth is but one country"* :

```
DATA_ASC     DB     'The earth is but one country','$'

MOV    AH,09              ;option 09 to display string of data
MOV    DX,OFFSET DATA_ASC        ;DX= offset address of data
INT    21H                       ;invoke the interrupt
```

4.2: DOS INTERRUPT 21H Option 02 outputting a single character

- To output only a single character, **02** is put in **AH**, and **DL** is loaded with the character to be displayed.
- The following displays the letter "*J*" :

```
MOV    AH,02        ;option 02 displays one character
MOV    DL,'J'       ;DL holds the character to be displayed
INT    21H          ;invoke the interrupt
```

  – This option can also be used to display '$' on the monitor as the string display option (option 09) will not display '$‘.

4.2: DOS INTERRUPT 21H Option 01 inputting a single character, with echo

- This functions waits until a character is input from the keyboard, then echoes it to the monitor.
  - After the interrupt, the input character will be in AL.

```
MOV    AH,01 ;option 01 inputs one character
INT    21H   ;after the interrupt, AL = input character (ASCII)
```

4.2: DOS INTERRUPT 21H Option 01 inputting a single character, with echo

- Program 4-1 combines INT 10H and INT 21H.

```
TITLE        PROG4-1 SIMPLE DISPLAY PROGRAM
PAGE         60,132
             .MODEL SMALL
             .STACK      64
;----------------------------
             .DATA
MESSAGE      DB     'This is a test of
                     the display routine','$'
;----------------------------
      .CODE
MAIN  PROC   FAR
      MOV    AX,@DATA
      MOV    DS,AX
      CALL   CLEAR          ;CLEAR THE SCREEN
      CALL   CURSOR         ;SET CURSOR POSITION
      CALL   DISPLAY        ;DISPLAY MESSAGE
      MOV    AH,4CH
      INT    21H            ;GO BACK TO DOS
MAIN  ENDP
```

The program does the following:

(1) Clears the screen.

(2) Sets the cursor to the center of the screen.

(3) Displays the message *"This is a test of the display routine".*

See the entire program listing on page 139 of your textbook.

The x86 PC
Assembly Language, Design, and Interfacing
By Muhammad Ali Mazidi, Janice Gillespie Mazidi and Danny Causey

4.2: DOS INTERRUPT 21H Option 0AH inputting a data string from the keyboard

- A means by which one can get keyboard data from & store it in a predefined data segment memory area.
  - Register AH = 0AH.
  - DX = offset address at which the string of data is stored.
    - Commonly referred to as a buffer area.
- DOS requires a buffer area be defined in the data segment.
  - The first byte specifies the size of the buffer.
  - The number of characters from the keyboard is in the second byte.
  - Keyed-in data placed in the buffer starts at the third byte.

4.2: DOS INTERRUPT 21H Option 0AH inputting a data string from the keyboard

- This program accepts up to six characters from the keyboard, including the return (carriage return) key.
  - Six buffer locations were reserved, and filled with FFH.

```
ORG    0010H
DATA1  DB     6,?,6 DUP (FF);0010H=06, 0012H to 0017H = FF

       MOV    AH,0AH              ;string input option of INT 21H
       MOV    DX,OFFSET DATA1     ;load offset address of buffer
       INT    21H                 ;invoke interrupt 21H
```

  - Memory contents of offset 0010H:

0010	0011	0012	0013	0014	0015	0016	0017
06	00	FF	FF	FF	FF	FF	FF

  - The PC won't exit INT 21H until it encounters a RETURN.

4.2: DOS INTERRUPT 21H Option 0AH inputting a data string from the keyboard

- Assuming the data entered through the keyboard was "USA" <RETURN>, the contents of memory locations starting at offset 0010H would look like:

```
0010   0011   0012   0013   0014   0015   0016   0017
06     03     55     53     41     0D     FF     FF
USACR
```

  – **`0010H = 06`** DOS requires the size of the buffer here.
  – **`0011H = 03`** The keyboard was activated three times (excluding the RETURN key) to key in letters **U**, **S**, and **A**.
  – **`0012H = 55H`** ASCII hex value for letter **U**.
  – **`0013H = 53H`** ASCII hex value for letter **S**.
  – **`0014H = 41H`** ASCII hex value for letter **A**.
  – **`0015H = 0DH`** ASCII hex value for **CR**. (carriage return)

4.2: DOS INTERRUPT 21H
inputting more than buffer size

- Entering more than six characters (five + the CR = 6) will cause the computer to sound the speaker.
  - The contents of the buffer will look like this:

0010	0011	0012	0013	0014	0015	0016	0017
06	05	55	53	41	20	61	0D
		U	S	A	SP	a	CR

  - Location **0015** has **ASCII 20H** for <SPACE>
  - Location **0016** has **ASCII 61H** for "a“.
  - Location **0017** has **0D** for <RETURN> key.
  - The actual length is **05** at memory offset **0011H**.

4.2: DOS INTERRUPT 21H
inputting more than buffer size

- If only the CR key is activated & no other character:

```
      ORG   20H
DATA4 DB    10,?,10 DUP (FF)
```

  - **0AH** is placed in memory 0020H.
  - **0021H** is for the count.
  - 0022H IS the first location to have data that was entered.

0020	0021	0022	0023	0024	0025	0026
0A	00	0D	FF	FF	FF	FF

0027	0028	0029	002A	002B	002C
FF	FF	FF	FF	FF	FF

CR is *not* included in the count.

  - If only the <RETURN> key is activated, **0022H** has **0DH**, the hex code for CR.
    - The actual number of characters entered is **0** at location **0021**.

4.2: DOS INTERRUPT 21H
use of carriage return and line feed

- In Program 4-2, the EQU statement is used to equate CR (carriage return) with its ASCII value of 0DH, and LF (line feed) with its ASCII value of 0AH.
  - See pages 141 & 142
- Program 4-3 prompts the user to type in a name with a maximum of eight letters.
  - The program gets the length and prints it to the screen.
  - See page 143.
- Program 4-4 demonstrates many functions described in this chapter.
  - See pages 144 & 145.

4.2: DOS INTERRUPT 21H Option 07 keyboard input without echo

- Option 07 requires the user to enter a single character, which is not displayed (or echoed) on the screen.
  - The PC waits until a single character is entered and provides the character in AL.

```
MOV    AH,07 ;keyboard input without echo
INT    21H
```

4.2: DOS INTERRUPT 21H
using LABEL to define a string buffer

- The LABEL directive can be used in the data segment to assign multiple names to data.

```
name   LABEL attribute
```

  - Used to assign the same offset address to two names.
  - The attribute can be:
    - BYTE; WORD; DWORD; FWORD; QWORD; TBYTE.
- In the following:

```
JOE    LABEL  BYTE
TOM    DB     20 DUP(0)
```

  - The offset address assigned to ***JOE*** is the same offset address for ***TOM*** since the LABEL directive does not occupy any memory space.

4.2: DOS INTERRUPT 21H
using LABEL to define a string buffer

- Use this directive to define a buffer area for the string keyboard input:

```
DATA_BUF     LABEL BYTE
MAX_SIZE     DB    10
BUF_COUNT    DB    ?
BUF_AREA     DB    10 DUP(20H)
```

- In the code segment the data can be accessed by name as follows:

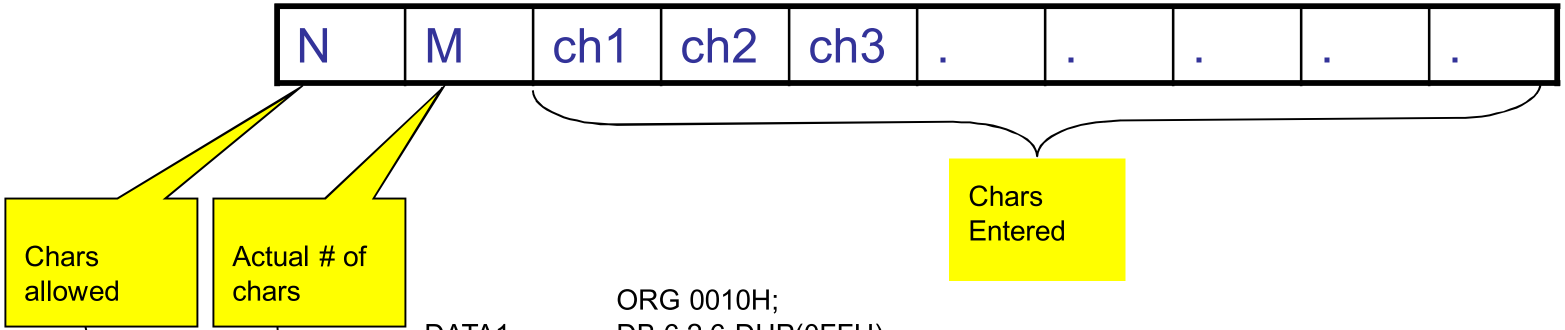
```
MOV   AH,0AH        ;load string into buffer
MOV   DX,OFFSET DATA_BUF
INT   21H
MOV   CL,BUF_COUNT;load the actual length of string
MOV   SI,OFFSET BUF_AREA;SI=address of first byte of string
```

INT 21h

- **INT 21H Option 01**: Inputs a single character with echo
  - This function waits until a character is input from the keyboard, then echoes it to the monitor. After the interrupt, the input character will be in AL.

INT 21h

- **INT 21H Option 0AH/09H**: Inputs/outputs a string of data stored at DS:DX
  - AH = 0AH, DX = offset address at which the data is located
  - AH = 09, DX = offset address at which the data located




```
        ORG 0010H;
DATA1   DB 6,?,6 DUP(0FFH)

        MOV AH, 0AH
        MOV DX, OFFSET DATA1
        INT 21H
```

Ex. What happens if one enters USA and then <RETURN>

0010	0011	0012	0013	0014	0015	0016	0017
06	03	55	53	41	0D	FF	FF

IRET

- IRET must be used for special handling of the stack.
- Must be used at the end of an ISR

SP-6	00
SP-5	02
SP-4	00
SP-3	01
SP-2	81
SP-1	00

SP initial

Return address + flags are loaded