Lecture 7

BIOS and DOS
Interrupt Instructions

Topics

- Disk Operating System
- Software Interrupts
  - 10h BIOS Video
  - 16h BIOS Keyboard
  - 21h MS-DOS

Disk Operating System (DOS)

- An Operating System is a collection of software running on a computer, providing operations such as writing strings to files, reading strings from the keyboard, writing strings to screen, and allocating blocks of memory.

- Examples of Operating Systems: DOS, Windows, Linux, Unix, MacOS.

- The Disk Operating System (DOS) was designed to run on the original IBM PC (International Bussiness Machines - Personal Computer).
- Intel 8086 CPUs were used in the first IBM PCs.

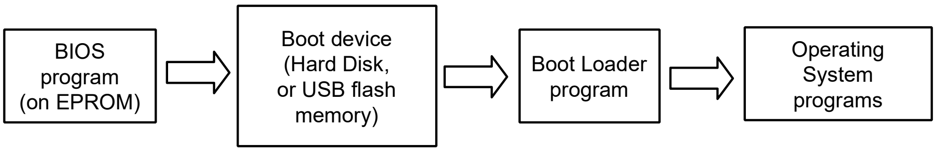
- There are different DOS implementations such as followings:
  MS-DOS (Microsoft), PC-DOS, FreeDOS.

- DOS is an 16-bit operating system, with a text-based command-line user interface, without a GUI (Graphical User Interface).

- DOS operates in real-address-mode.
- In real-address-mode, only one program at a time can be run by the CPU.
- In Windows and Linux operating systems the protected-address-mode is used. More than one program can be run simultaneously.

Booting Process of a Computer

- Booting is the process of starting a computer and bringing up the operating system.
- When a computer turns on, the CPU looks for instructions in the BIOS ROM chip and executes them.
- BIOS (Basic Input/Output System) is a firmware program on an EPROM chip.
- From a boot device (master boot record), the bootstrap-loader program is loaded to RAM memory, and executed by CPU.
- The boot loader then loads the operating system (DOS, Linux, Windows, etc.) from the boot device to the RAM memory.



BIOS program (on EPROM)

Boot device (Hard Disk, or USB flash memory)

Boot Loader program

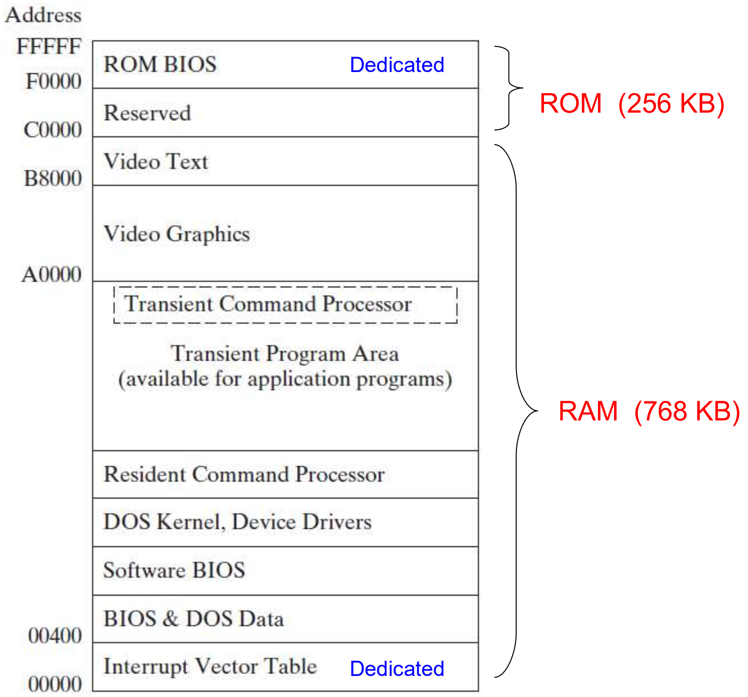
Operating System programs

Basic Input/Output System (BIOS)

- BIOS is a collection of programs stored in an EPROM memory that operates many of the I/O devices connected to computer system.
- BIOS was originally developed by IBM company for the IBM PCs.
- BIOS programs are low-level subroutines that communicate directly with basic hardware devices such as keyboard and video display.
- BIOS program is installed to a BIOS ROM chip by the computer's manufacturer.
- Operating System programs communicate with the BIOS.

Memory Map of a PC with DOS

- In IBM PC, total 1 MB memory (1024 KB) space is allocated to various sections for DOS.
- The memory allocation is called a memory map.



Address

FFFFF

F0000

C0000

B8000

A0000

00400

00000

ROM BIOS Dedicated

Reserved

Video Text

Video Graphics

Transient Command Processor

Transient Program Area (available for application programs)

Resident Command Processor

DOS Kernel, Device Drivers

Software BIOS

BIOS & DOS Data

Interrupt Vector Table Dedicated

ROM (256 KB)

RAM (768 KB)

System Programs of DOS

Program	Description
Interrupt Service Routines	• They are a collection of interrupt service procedures (interrupt handlers). • The procedures access various features of the BIOS and DOS. • Their memory addresses are taken from the interrupt vector table.
Device Drivers	• Device drivers are programs that permit the operating system to communicate directly with hardware devices and BIOS. • They are programs that control installable I/O devices such as mouse, disk, printer, scanner, etc. • DOS device drivers are files that have an extension of .SYS, such as MOUSE.SYS. • Boot loader transfers the system files IO.SYS (Input/Output driver program) and MSDOS.SYS program from disk drive to main memory.
Command Interpreter	• COMMAND.COM is a text-based command interpreter program (user interface program). • It shows the DOS prompt on the screen that gives the user access to DOS's built-in commands like DIR, COPY, DATE, TIME, etc.

Topics

- Disk Operating System
- Software Interrupts
  - 10h BIOS Video
  - 16h BIOS Keyboard
  - 21h MS-DOS

Software Interrupts

- An interrupt is an event that causes the microprocessor to suspend its present task, and transfer control to a new program called the Interrupt Service Routine (ISR).
- There are two sources of interrupts:
  - Hardware interrupts generated by a special chip, such as Intel 8259 Interrupt Controller.
  - Software interrupts generated by the INT instruction in an Assembly program.
- Software Interrupt is similar to the way the hardware interrupt works.
- The INT instruction requests services from the BIOS or DOS, mostly for Input/Output to/from external devices such as keyboard, screen, etc.
- INT instruction has a range of 0 - 255 (hexadecimal 00h - FFh) interrupt numbers, so there can be at most 256 software interrupts.
- Before the INT instruction is executed, the AH register should be assigned in the Assembly program, so that it contains a function number which identifies the interrupt service subroutine.

Interrupt Handlers

- A software interrupt is a call to a BIOS or Operating System procedure.
- The interrupt procedures (Interrupt Service Rotines) are Interrupt Handlers.
- Mostly they provide input/output operations for application programs.

- Interrupt Handlers are used for many tasks such as the followings.
  - Displaying characters and strings on console screen
  - Reading characters and strings from the keyboard
  - Displaying text in color
  - Drawing pixels on graphics screen (in video mode)
  - Opening and closing files on hard disk
  - Reading data from files
  - Writing data to files
  - Setting and retrieving the system time and date
  - Tracking the mouse inputs

INT Instruction

General Syntax `INT   Interrupt_Number`

- **INT : Interrupt**

- The INT instruction calls a system subroutine also known as an interrupt handler procedure.
- INT instruction works like a FAR procedure call.
  When it is invoked, it saves CS:IP and the flags on the stack and goes to the subroutine associated with the interrupt.
- Before the INT instruction executes, one or more parameters must be assigned in registers by the programmer.
- An interrupt number identifying the particular procedure must be moved to the AH register.
- Depending on the interrupt function, other values may have to be passed to the interrupt routines in registers.
- The Interrupt_Number in the INT instruction format refers to a number between 0 and 255, which identify the interrupt type.

Interrupt Numbers

```
Number   Description

00   Divide Error. CPU-generated: activated when attempting to divide by zero.
01   Single Step. CPU-generated: actived when the CPU Trap flag is set.
02   Nonmaskable Interrupt. External hardware: activated when a memory error occurs.
03   Breakpoint. CPU-generated: activated when the 0CCh (INT 3) instruction is executed.
04   INTO Detected Overflow. CPU-generated: Activated when Overflow flag is set.
05   Print Screen. Activated by the INT 5 instruction or pressing the Shift-PrtSc keys.
08   IRQ0: System Timer Interrupt.
09   IRQ1: Keyboard Hardware Interrupt. Activated when a key is pressed.
0A   IRQ2: Programmable Interrupt Controller
0B   IRQ3: Serial Communications (COM2)
0C   IRQ4: Serial Communications (COM1)
0D   IRQ5: Fixed Disk
0F   IRQ7: Parallel Printer
10   Video Services. Routines for manipulating the video display.
11   Equipment Check. Return a word showing all the peripherals attached to the system.
12   Memory Size. Return the amount of memory (in 1024-byte blocks) in AX.
14   Asynchronous (Serial) Port Services. Read/write asynchronous communications port.
16   Keyboard Services. Read and inspect keyboard input.
17   Printer Services. Initialize, print, and return the status of the printer.
19   Bootstrap Loader. Reboot MS-DOS.
1B   Keyboard Break. This interrupt handler is executed by INT 9h when CTRL-BREAK is pressed.
1D   Video Parameters. Point to a table containing initialization for video controller chip.
1F   Graphics Table of all extended graphics characters.
21   MS-DOS Services
22   MS-DOS Terminate Address. When current program ends, this will be the return address.
23   MS-DOS Break Address. MS-DOS jumps here when CTRL-BREAK is pressed.
33   Microsoft Mouse. Functions that track and control the mouse.
40–41   Fixed Disk Services. Fixed disk controller.
60–6B   Available for application programs to use.
F1–FF   Available for application programs.
```

Interrupt Vector Table

- 8086 CPU processes the INT instructions by using the Interrupt Vector Table in BIOS ROM.
- Interrupt Vector Table is located in the first 1024 bytes of BIOS ROM (locations 0000h through 03FFh).
- Each entry in the interrupt vector table is a 32-bit segment:offset address that points to one of the existing service routines (interrupt handlers).
- Different computers may have different vector addresses, because of different versions of the BIOS and MS-DOS.
- Example: In the vector table, the address of the INT 0 handler (Divide by zero error) is 02C1:5186h.
- The offset of any interrupt vector may be computed by multiplying its interrupt number by 4.
- Example: The offset of the vector for INT 09h is 9 * 4.
  (0024h = 36).

Entries in Interrupt Vector Table

The following is an example of vector table entries (rows) in ROM.

Interrupt Number	Offset Address	Interrupt Vector Table Entries (4-byte Addresses)
0	0000h	02C1:5186
1	0004h	0070:0C67
2	0008h	0DAD:2C1B
3	000Ch	0070:0C67
4	0010h	F000:FF54
5	0014h	F000:837B
6	0018h	0D70:022C
7	001Ch	0DAD:2BAD
8	0020h	0070:0325
9	0024h	0070:039F
10	0028h	0070:0419
11	002Ch	0070:0493
..	....	....
255	03FFh	....

Invoking an Interrupt Handler

Example: The followings are the steps taken by the CPU,
when the **INT 10h** instruction is invoked by an Assembly program.

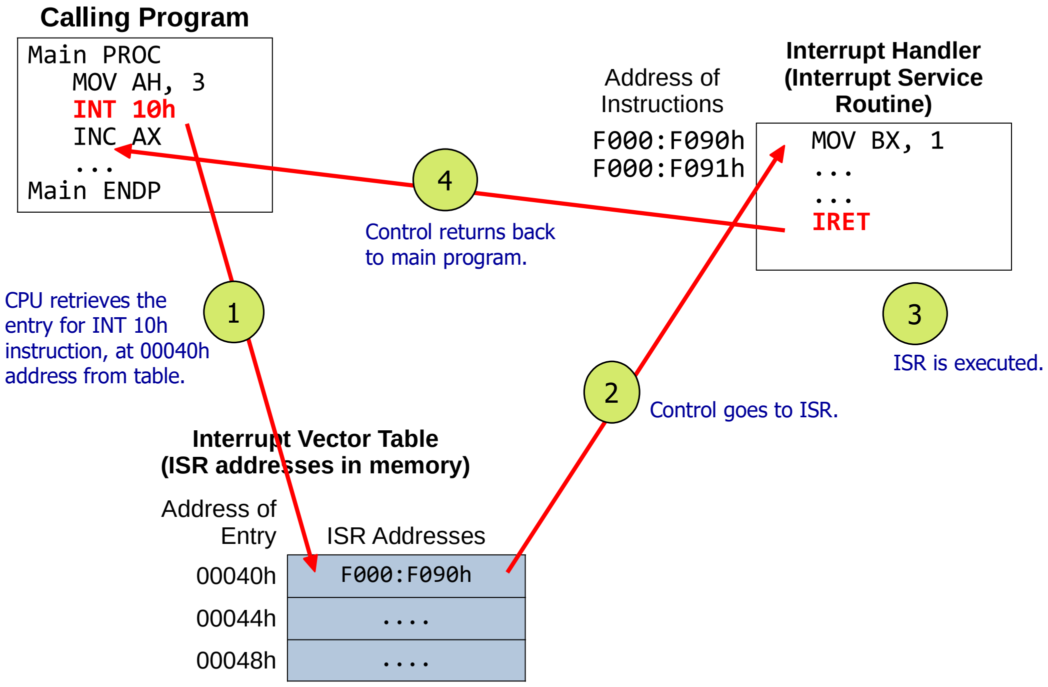
Step 1) The operand of the **INT 10h** instruction is multiplied by 4 to locate the matching interrupt vector table entry.

Step 2) CPU pushes the Flags and a 32-bit segment:offset return address on the stack, and executes a far call to the address stored at location (10h * 4) in the interrupt vector table (F000:F090).

Step 3) The interrupt handler (as part of BIOS program) at the F000:F090 memory address executes until it reaches an IRET (interrupt return) instruction.

Step 4) The IRET instruction pops the flags and the return address off the stack, causing the processor to resume execution immediately following the INT 10h instruction in the calling program.

Invoking an Interrupt Handler



Calling Program

```
Main PROC
   MOV AH, 3
   INT 10h
   INC AX
   ...
Main ENDP
```

Interrupt Handler (Interrupt Service Routine)

Address of Instructions	
F000:F090h	MOV BX, 1
F000:F091h	...
	...
	IRET

Interrupt Vector Table (ISR addresses in memory)

Address of Entry	ISR Addresses
00040h	F000:F090h
00044h	....
00048h	....

1 CPU retrieves the entry for INT 10h instruction, at 00040h address from table.

2 Control goes to ISR.

3 ISR is executed.

4 Control returns back to main program.

Common Software Interrupts

- Software interrupts call the interrupt service routines (ISR) in BIOS or DOS.
- Some commonly used interrupts are given below.

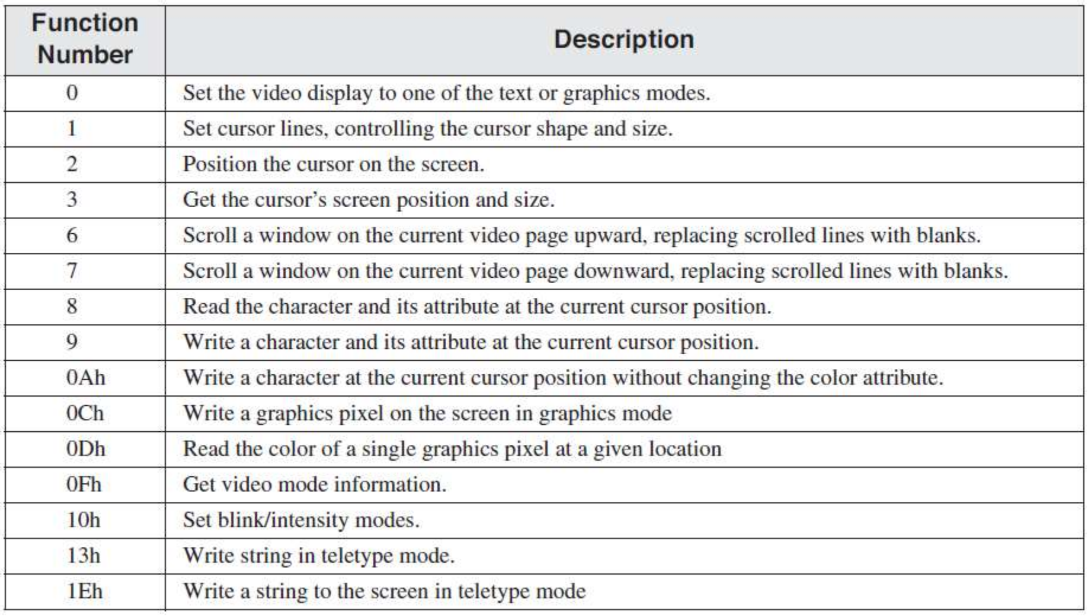
Interrupt Instruction	Name of Service	Description
INT 10h	BIOS Video Functions	Procedures that display text in color, scroll the screen, and display video graphics. INT 10h subroutines are in the ROM BIOS. The AH register is used as a parameter containing the interrupt function number.
INT 16h	BIOS Keyboard Functions	Procedures that read the keyboard and check its status.
INT 17h	BIOS Printer Functions	Procedures that initialize, print, and return the printer status.
INT 21h	MS-DOS Functions	Procedures that provide input/output, file handling, and memory management, high-level keyboard and screen services. The INT 21h service is valid only in 16-bit applications. Not valid in 32-bit or 64-bit applications.
INT 33h	Mouse Functions	Procedures that provide mouse input.

Topics

- Disk Operating System
- Software Interrupts
  - 10h BIOS Video
  - 16h BIOS Keyboard
  - 21h MS-DOS

INT 10h BIOS Video Functions

- The INT 10h BIOS interrupt service provides low-level video functions.
- AH register should be loaded with a function number.
- The followings are the most commonly used INT 10h functions.



Function Number	Description
0	Set the video display to one of the text or graphics modes.
1	Set cursor lines, controlling the cursor shape and size.
2	Position the cursor on the screen.
3	Get the cursor's screen position and size.
6	Scroll a window on the current video page upward, replacing scrolled lines with blanks.
7	Scroll a window on the current video page downward, replacing scrolled lines with blanks.
8	Read the character and its attribute at the current cursor position.
9	Write a character and its attribute at the current cursor position.
0Ah	Write a character at the current cursor position without changing the color attribute.
0Ch	Write a graphics pixel on the screen in graphics mode
0Dh	Read the color of a single graphics pixel at a given location
0Fh	Get video mode information.
10h	Set blink/intensity modes.
13h	Write string in teletype mode.
1Eh	Write a string to the screen in teletype mode

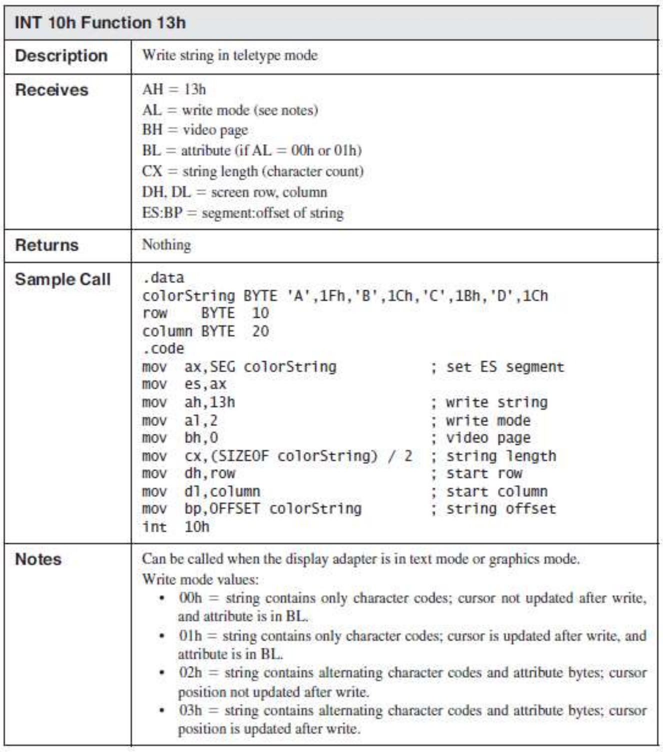
INT 10h , Function 13h (Text mode)

- INT 10h Function 13h writes a string to the screen at a given row and column location.
- The function code number 13h must be loaded to AH register, along with other necessary parameters shown below.
- The string can optionally contain both characters and attribute values.
- The 13h function can be used in text mode or graphics mode.
- Following is an example of text screen dimensions (not as pixels).

Screen
in text
mode

Text Mode
80 x 25
Columns x Rows

INT 10h , Function 13h (Parameters)



INT 10h Function 13h	
Description	Write string in teletype mode

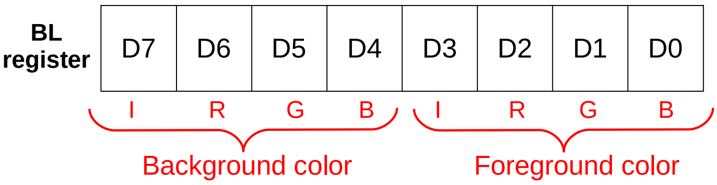
Receives	AH = 13h
	AL = write mode (see notes)
	BH = video page
	BL = attribute (if AL = 00h or 01h)
	CX = string length (character count)
	DH, DL = screen row, column
	ES:BP = segment:offset of string

Returns	Nothing
Sample Call	see code below
Notes	Can be called when the display adapter is in text mode or graphics mode. Write mode values: • 00h = string contains only character codes; cursor not updated after write, and attribute is in BL. • 01h = string contains only character codes; cursor is updated after write, and attribute is in BL. • 02h = string contains alternating character codes and attribute bytes; cursor position not updated after write. • 03h = string contains alternating character codes and attribute bytes; cursor position is updated after write.

```
.data
colorString BYTE 'A',1Fh,'B',1Ch,'C',1Bh,'D',1Ch
row    BYTE  10
column BYTE  20
.code
mov  ax,SEG colorString          ; set ES segment
mov  es,ax
mov  ah,13h                      ; write string
mov  al,2                        ; write mode
mov  bh,0                        ; video page
mov  cx,(SIZEOF colorString) / 2  ; string length
mov  dh,row                      ; start row
mov  dl,column                   ; start column
mov  bp,OFFSET colorString       ; string offset
int  10h
```

Setting the Text Attributes

- **BL register** is used to set the attributes of text, before calling the INT 10.
- The followings are bit definitions for text attributes.
- D7 bit controls the Blink/Intensity mode. (1=Blink, 0=Intensity)
- I=Intensity, R=Red, G=Green, B=Blue



BL register	D7	D6	D5	D4	D3	D2	D1	D0
	I	R	G	B	I	R	G	B

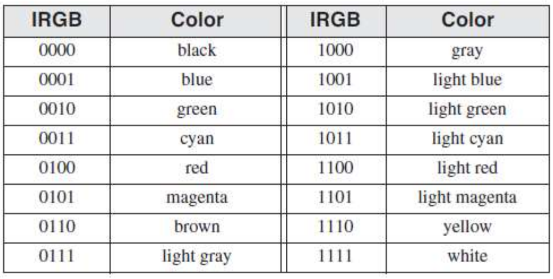
Example instruction

MOV BL, 10101111b

Bits (left to right) :

1 = Blinking
010 = Green background
1 = Intensity
111 = White foreground

4-Bit Color Text Encoding (16 different colors)



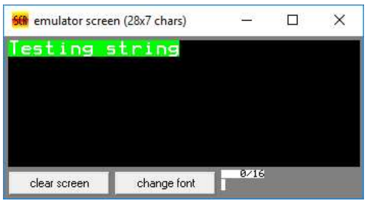
IRGB	Color	IRGB	Color
0000	black	1000	gray
0001	blue	1001	light blue
0010	green	1010	light green
0011	cyan	1011	light cyan
0100	red	1100	light red
0101	magenta	1101	light magenta
0110	brown	1110	yellow
0111	light gray	1111	white

Example1: Displaying color text (INT 10h)

- Write a program to print a string on screen in text mode.
- Text should have a background color and foreground color.
- Also the text should blink.

```
.MODEL small
.data
Cumle DB "Testing string"
.code
 MOV DX, @data
 MOV ES, DX             ; Extra segment required
 MOV BP, offset Cumle  ; BP required
 MOV AH, 03h            ; Select function for getting cursor coordinate
 INT 10h                ; Call BIOS interrupt for Video Service
                        ; DH, DL now contain screen row, column
 ;Set the text attributes in BL register.
 MOV BH,00h             ; Video page
 MOV BL,10101111b  ; Text attribute (color)
 MOV CX, 14             ; Number of characters in message text
 MOV AL,01h             ; Write mode
 MOV AH, 13h            ; Select function for text displaying on screen
 INT 10h                ; Call BIOS interrupt for Video Service
 .EXIT
END
```

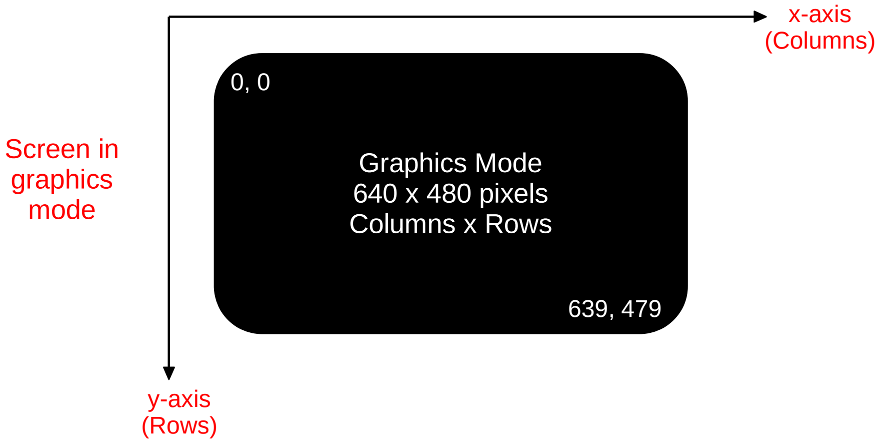
Screen Output
(Text blinking)



```
Testing string
```

INT 10h , Function 0Ch (Graphics mode)

- INT 10h Function 0Ch draws one pixel on the screen, when the video controller is in graphics mode.
- The following is an example of graphics screen resolution (as pixels).
  - Left corner coordinate is (0, 0).
  - Right corner coordinate is (639, 479).
  - Horizontal axis is for columns, vertical axis is for rows.



x-axis
(Columns)

0, 0

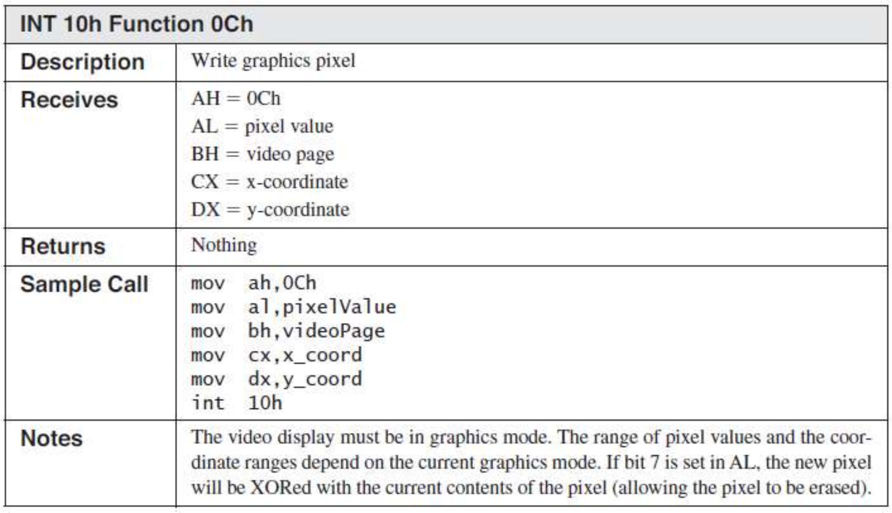
Screen in
graphics
mode

Graphics Mode
640 x 480 pixels
Columns x Rows

639, 479

y-axis
(Rows)

INT 10h , Function 0Ch (Parameters)



INT 10h Function 0Ch	
Description	Write graphics pixel

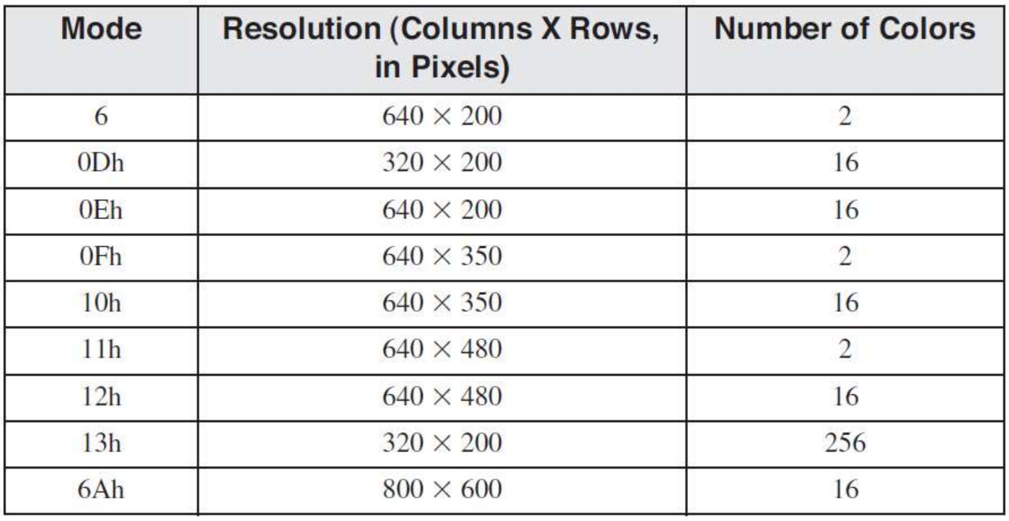
Receives	AH = 0Ch
	AL = pixel value
	BH = video page
	CX = x-coordinate
	DX = y-coordinate

Returns	Nothing

Sample Call	`mov  ah,0Ch`
	`mov  al,pixelValue`
	`mov  bh,videoPage`
	`mov  cx,x_coord`
	`mov  dx,y_coord`
	`int  10h`

Notes	The video display must be in graphics mode. The range of pixel values and the coordinate ranges depend on the current graphics mode. If bit 7 is set in AL, the new pixel will be XORed with the current contents of the pixel (allowing the pixel to be erased).

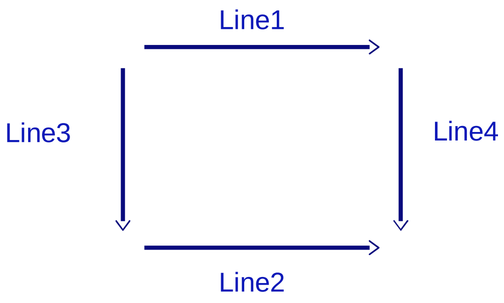
Video Graphics Modes for INT 10h



Mode	Resolution (Columns X Rows, in Pixels)	Number of Colors
6	640 × 200	2
0Dh	320 × 200	16
0Eh	640 × 200	16
0Fh	640 × 350	2
10h	640 × 350	16
11h	640 × 480	2
12h	640 × 480	16
13h	320 × 200	256
6Ah	800 × 600	16

Example2: Drawing a rectangle (INT 10h)

- Write a program to draw a rectangle on screen in graphics mode.
- Column offset from the (0,0) point is = 100 pixel
- Row offset from the (0,0) point is = 20 pixel
- Width of rectangle is = 70 pixels
- Height of rectangle is = 40 pixels
- To draw the rectangle, draw two parallel vertical lines, and also draw two parallel horizontal lines.

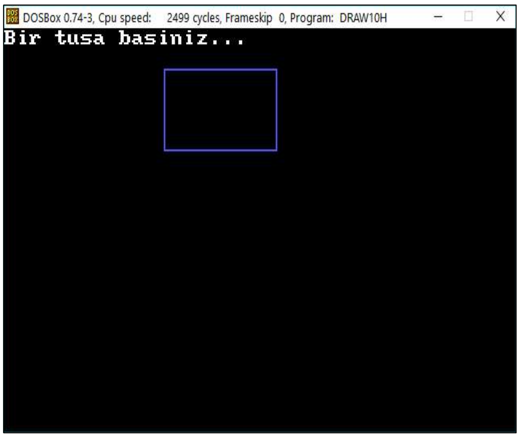


Program

Part1

```
.model small
.data
Cumle DB  'Bir tusa basiniz...','$'
;-------------------------------------------
; Dimensions of rectangle (in pixels)
rwidth    equ  70     ; Width
rheight   equ  40     ; Height
renk  equ  9          ; Light blue color (IRGB = 1001)
;-------------------------------------------

.code
; Set video mode as VGA 320x200 pixels (13h)
   mov ah, 0            ; Select function
   mov al, 13h
   int 10h
;-------------------------------------------
; Left corner coordinates of rectangle:  x=100, y=20
; Draw the upper horizontal line
    mov cx, 100+rwidth  ; column
    mov dx, 20           ; row
    mov al, renk         ; select color
dongu1:
    mov ah, 0ch          ; put one pixel at a time
    int 10h
    dec cx
    cmp cx, 100
    jae dongu1
```

Screen Output
(Rectangle is drawn)


DOSBox 0.74-3, Cpu speed: 2499 cycles, Frameskip 0, Program: DRAW10H

```
Bir tusa basiniz...
```

Part2

```
; Draw the lower horizontal line
    mov cx, 100+rwidth     ; column
    mov dx, 20+rheight     ; row
    mov al, renk           ; select color
dongu2:
    mov ah, 0ch            ; put one pixel
    int 10h
    dec cx
    cmp cx, 100
    ja dongu2
;-------------------------------------------

; Draw the left vertical line
    mov cx, 100            ; column
    mov dx, 20+rheight     ; row
    mov al, renk           ; select color
dongu3:
    mov ah, 0ch            ; put one pixel
    int 10h
    dec dx
    cmp dx, 20
    ja dongu3
```

Part3

```
; Draw the right  vertical line
    mov cx, 100+rwidth     ; column
    mov dx, 20+rheight     ; row
    mov al, renk           ; select color
dongu4:
    mov ah, 0ch            ; put one pixel
    int 10h
    dec dx
    cmp dx, 20
    ja dongu4
;-------------------------------------------
     ;Metin adresini yukle
     MOV  AX, @data
     MOV  DS, AX
     LEA  DX, Cumle
     ;Metni ekrana yaz
     MOV  AH, 09H
     INT  21H
;-------------------------------------------
; Pause the screen, wait for keypress.
  mov ah,00  ;Select function
  int 16h    ; BIOS keyboard service
;-------------------------------------------
; Return to text mode
  mov ah,00   ;Select function
  mov al,03   ;Text mode 3
  int 10h     ;BIOS video service

.EXIT
END
```

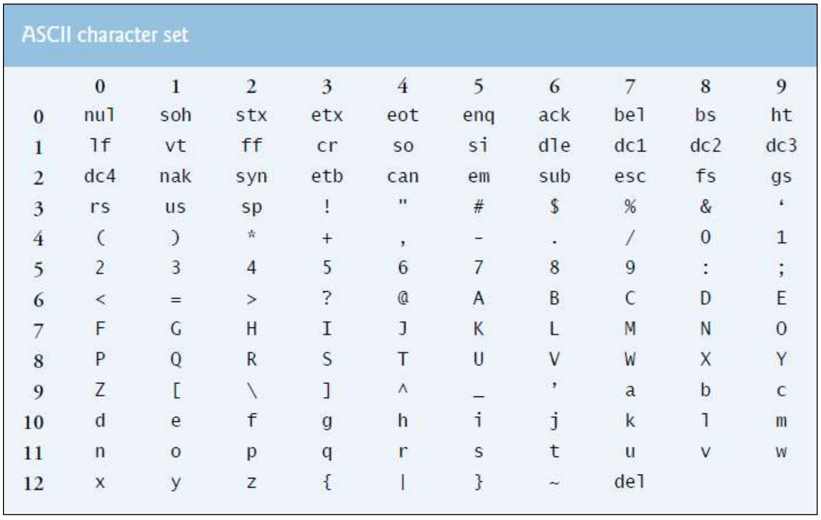
Topics

- Disk Operating System
- Software Interrupts
    - 10h BIOS Video
    - 16h BIOS Keyboard
    - 21h MS-DOS

INT 16h BIOS Keyboard Functions

- The BIOS handles keyboard input using calls to the interrupt 16h.
- INT 16h instruction allow programming directly at the BIOS level, by calling keyboard functions that were installed by the computer manufacturer.
- Each keypress generates an 8-bit ASCII scan code.
- ASCII codes are standardized on all computers.

Example: ASCII decimal code for character lowercase ‘a’ is 97.



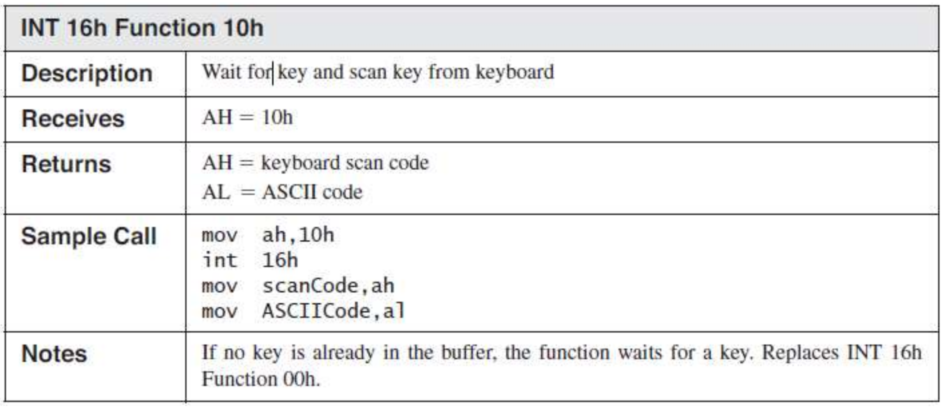
ASCII character set

	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	lf	vt	ff	cr	so	si	dle	dc1	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	$	%	&	‘
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	’	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	del		

BEL	Bell
BS	Backspace
CAN	Cancel
CR	Carriage return
ESC	Escape
FF	Form feed
HT	Horizontal tab
LF	Line feed
NUL	Null
VT	Vertical tab

INT 16h , Function 10h (Parameters)

- INT 16h Function 10h waits for keypress.
- It removes the next available key from the keyboard buffer.
- If the keyboard buffer is empty, then the keyboard handler waits for the user to press a key.



INT 16h Function 10h	
Description	Wait for key and scan key from keyboard
Receives	AH = 10h

Returns	AH = keyboard scan code
	AL = ASCII code

Sample Call	`mov ah,10h`
	`int 16h`
	`mov scanCode,ah`
	`mov ASCIICode,al`

Notes	If no key is already in the buffer, the function waits for a key. Replaces INT 16h Function 00h.

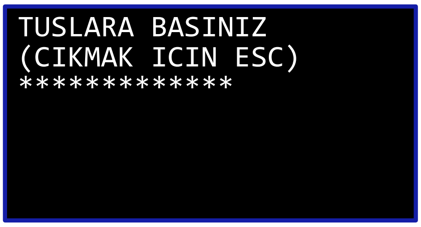
Example: Waiting for key-presses (INT 16h)

- Write a program to do followings.
- Program runs in endless loop and waits for user to press keys.
- Displays the * symbol on screen for each pressed key.
- Exits when user presses the Escape key.

Part1

```
.model small
.data
Cumle1  DB   'TUSLARA BASINIZ', 13, 10, '$'
Cumle2  DB   '(CIKMAK ICIN ESC)', 13, 10, '$'
;ASCII codes: 13 is Carriage Return,
;10 is Line Feed, $ is string end symbol

.code
.startup
mov ah, 09h     ;Function code for displaying string
mov dx, OFFSET  Cumle1     ;Address of Cumle1
int 21h
mov dx, OFFSET  Cumle2     ;Address of Cumle2
int 21h
;-----------------------------------------------
DEVAM:   ;Endless loop
  ; Wait for a keypress (pause).
  ; Remove char from keyboard buffer if any.
  mov ah, 10h
  int 16h
```

Screen Output



```
TUSLARA BASINIZ
(CIKMAK ICIN ESC)
*************
```

Part2

```
;-----------------------------------------------
  CMP AL, 27         ; Compare with ASCII code of Escape Key
  JE  SON            ; Quit the program, if ESC was pressed.
;-----------------------------------------------
  ; Display the pressed key (in AL) on the screen
  MOV    AH, 0Eh       ; Select displaying service subfunction
  MOV    BL, AL        ; Copy AL to BL (can be used later)
  MOV    AL, '*'       ; Copy * symbol to AL
  INT    10h           ; Display the char in AL

  JMP DEVAM            ; Goto endless loop
;-----------------------------------------------
SON:
.EXIT                  ; Stop program

END
```

Topics

- Disk Operating System
- Software Interrupts
  - 10h BIOS Video
  - 16h BIOS Keyboard
  - 21h MS-DOS

INT 21h MS-DOS Functions

- The INT 21h interrupt service provides MS-DOS Function calls for many high-level functions (200 functions) for displaying text on the console.
- The AH register should be loaded with a function number, before calling INT 21h.
- Each function uses certain input parameters and return values.
- The followings are the most used INT 21h functions, grouped as categories.

INT 21h Output Functions

Function	Description
2	Write one character to standard output (screen). Receives: DL=character.
4	Write one character to standard auxiliary output (serial port).
5	Write one character to printer. Receives: DL=character.
6	Direct the console input/output. If DL is FFh, read a waiting character from standard input. If DL is any other value, write the character in DL to standard output.
9	Write a string terminated by a $ character, to standard output (screen). Receives: DS:DX address of the string.

INT 21h Input Functions

Function	Description
1	Read one character from standard input. If no character is ready, wait for input. Returns: AL=character.
3	Read one character from standard auxiliary input (serial port).
7	Direct the character input without echo. Wait for a character from standard input. Returns: AL=character.
8	Character input without echo. Wait for a character from the standard input device. Returns: AL=character. Character not echoed.
0Ah	Buffered keyboard input. Read a string of characters from the standard input device. Receives: DS:DX .
0Bh	Check standard input status. Check to see if an input character is waiting. Returns: AL=0FFh if the character is ready; otherwise, AL=0

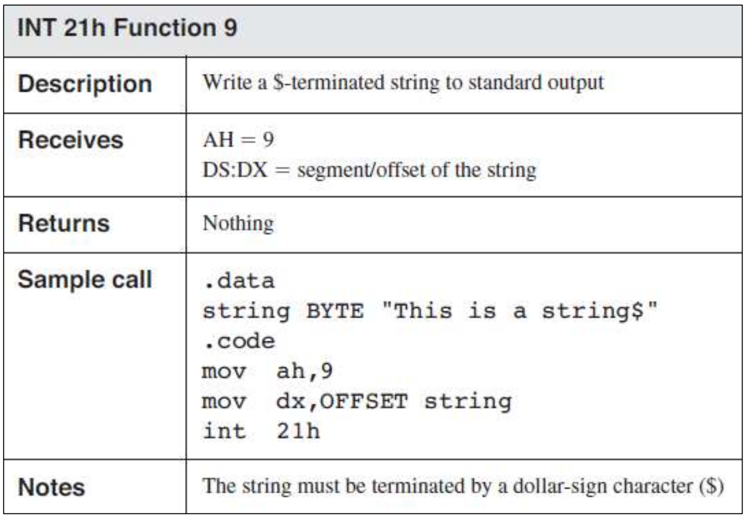
INT 21h Date and Time Functions

Function	Description
2A	Get system date. Returns: AL Day of the week (0–6, where Sunday is 0), CX year, DH month, and DL day.
2B	Set system date. Receives: CX year, DH month, and DL day. Returns: AL 0 if the date is valid.
2C	Get system time. Returns: CH hour, CL minutes, DH seconds, and DL hundredths of seconds.
2D	Set system time. Receives: CH hour, CL minutes, DH seconds, and DL hundredths of seconds. Returns: AL 0 if the time is valid.

INT 21h Terminate Function

Function	Description
4Ch	Terminate the process. INT 21h Function 4Ch terminates the current program (process).

INT 21h , Function 09h
(Write whole string to screen)

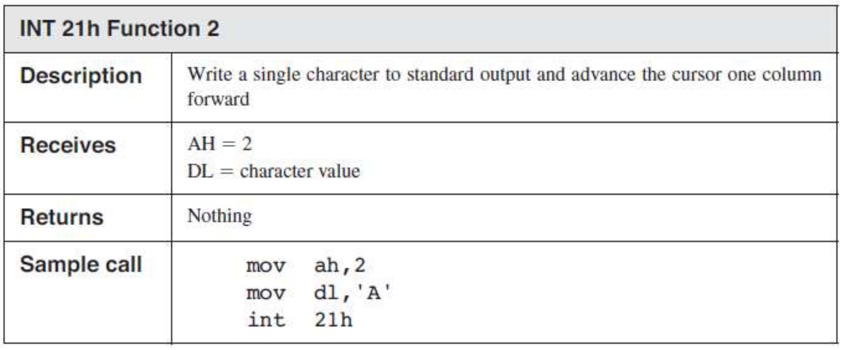


INT 21h Function 9	
Description	Write a $-terminated string to standard output
Receives	AH = 9 DS:DX = segment/offset of the string
Returns	Nothing

Sample call	`.data`
	`string BYTE "This is a string$"`
	`.code`
	`mov  ah,9`
	`mov  dx,OFFSET string`
	`int  21h`

Notes	The string must be terminated by a dollar-sign character ($)

INT 21h , Function 02h
(Write one character to screen)



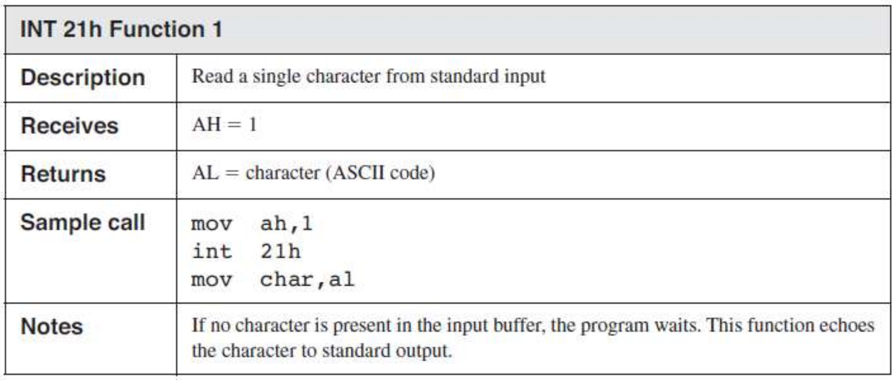
INT 21h Function 2	
Description	Write a single character to standard output and advance the cursor one column forward

Receives	AH = 2
	DL = character value

Returns	Nothing

Sample call	`mov  ah,2`
	`mov  dl,'A'`
	`int  21h`

INT 21h , Function 01h (Read one character from keyboard)

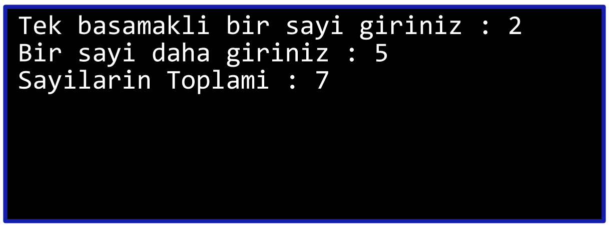


INT 21h Function 1	
Description	Read a single character from standard input
Receives	AH = 1
Returns	AL = character (ASCII code)
Sample call	mov ah,1 int 21h mov char,al
Notes	If no character is present in the input buffer, the program waits. This function echoes the character to standard output.

Example1: Console Input/Output (INT 21h)

- Write a program that reads two numbers from keyboard.
- Program calculates the total and displays the result on screen.
- Suppose the user enters numbers that contain only one single decimal digit between 0-9.
- Also suppose the result (total) will be only one single decimal digit (0-9).
- Use the following MS-DOS interrupts :
  INT 21h / Function 09h, Function 01h, and Function 02h.

Example of screen output



```
Tek basamakli bir sayi giriniz : 2
Bir sayi daha giriniz : 5
Sayilarin Toplami : 7
```

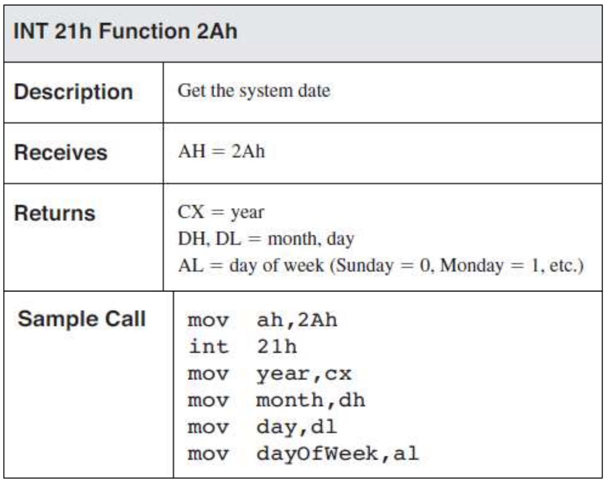
Part1

```
; Makro fonksiyon
; Makro ismi: Yazdir
; Parametre ismi : msj

Yazdir  MACRO  msj
     LEA  DX, msj      ;Load Effective Address
     MOV  AH, 09H    ;Select function (string output)
     INT  21H
     ENDM
;===============================================
.model small
.data
Cumle1  DB  'Tek basamakli bir sayi giriniz : ','$'
Cumle2  DB  13,10,'Bir sayi daha giriniz : ','$'
Cumle3  DB  13,10,'Sayilarin Toplami : ','$'
```

Part2

```
.code
Basla   PROC
.STARTUP
     Yazdir  Cumle1      ; Macro called
     ;Birinci sayiyi oku (AL contains ascii input)
     MOV    AH, 1        ;Read char with echo
     INT    21h
     MOV BL, AL         ;AL yi BL ye kopyala
;---------------------------------------------------
     Yazdir  Cumle2      ; Macro called
     ;Ikinci sayiyi oku (AL contains ascii input)
     MOV    AH, 1         ;Read char with echo
     INT    21h
;---------------------------------------------------
     ;Toplama islemini yap
     SUB  BL, 30h    ;BL yi asciiden sayiya cevir
     ADD  BL, AL      ;AL yi BL ye ekle
;---------------------------------------------------
     Yazdir  Cumle3       ; Macro called
     ;Sonucu (ascii) ekrana yaz
     MOV    DL, BL        ;BL den DL ye kopyala
     MOV    AH, 2         ;Write one character
     INT    21h
;---------------------------------------------------
.EXIT
Basla   ENDP
END Basla
```

INT 21h , Function 2Ah (Get system date)



INT 21h Function 2Ah	
Description	Get the system date
Receives	AH = 2Ah

Returns	CX = year
	DH, DL = month, day
	AL = day of week (Sunday = 0, Monday = 1, etc.)

Sample Call	`mov   ah,2Ah`
	`int   21h`
	`mov   year,cx`
	`mov   month,dh`
	`mov   day,dl`
	`mov   dayOfWeek,al`

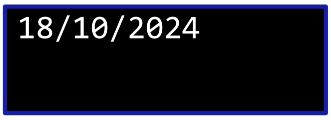
Example2: Displaying System Date (INT 21h)

- Write a program that displays the computer’s system date on screen.
- Display format is : `dd/mm/yyyy`
- Using MS-DOS : INT 21h / Function 2Ah, Function 02h, and Function 21h.

Part1

```
.model small
.data
sonuc db 5 dup(0)
;String buffer for date component displaying.
;16-bit Year data can be maximum 65535.
;65535 can be stored as 5 ASCII characters.
;--------------------------------
.code

Basla   PROC
.startup

; Read DOS system date
MOV AH, 2AH
INT 21H
;Returned values:
;CX year
;DH month
;DL day
;AL Day of week (0 is Sunday)
```

Screen
Output



```
18/10/2024
```

Part2

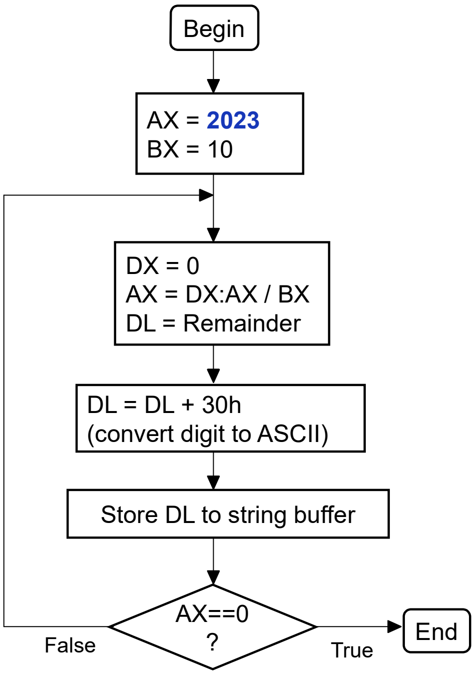
```
PUSH DX                     ;Save backup of DX on stack
MOV AX,0
MOV AL, DL                  ;DAY
CALL  Ekrana_Yaz            ;Pass parameter in AX
;--------------------------------
;DL should contain one ASCII character to print
MOV DL, '/'                 ;Separator
MOV AH, 2                   ;Write char
INT    21h
;--------------------------------
POP DX                      ;Restore backup of DX from stack
MOV AX,0
MOV AL, DH                  ;MONTH
CALL  Ekrana_Yaz            ;Pass parameter in AX
;--------------------------------
;DL should contain one ASCII character to print
MOV DL, '/'                 ;Separator
MOV AH, 2                   ;Write char
INT    21h
;--------------------------------
MOV AX, CX                  ;YEAR
CALL  Ekrana_Yaz            ;Pass parameter in AX
;--------------------------------
.EXIT
Basla   ENDP
```

Part3

```
Ekrana_Yaz  PROC
;Input parameter is AX register (number for displaying)
;Obtain digits one-by-one, store them as ascii to string buffer.

  mov si, offset  sonuc     ; SI is used as index on string
  add  si, 5                ; String buffer will be filled from right-to-left
  mov  byte ptr [si], '$     ; String terminator
  mov  bx, 10                ; Used for division
Dongu:
  mov  dx, 0         ; DX used for 32-bit division. (Clear remainder in DL)
  div  bx            ; Divide DX:AX by BX, Remainder in DL
  add  dl, 30h       ; Add 30h to DL to Convert Digit to Ascii
  dec  si            ; Decrement by 1
  mov  [si], dl       ;Store one Ascii digit to string buffer
  cmp ax, 0           ; Compare with zero
  jnz  Dongu

  mov  dx, si        ; Beginning address of string
  mov  ah, 09h       ; Write whole string to screen
  int  21h
  RET
Ekrana_Yaz ENDP

END  Basla  ; End of file
```

Flowchart for Ekrana_Yaz Subroutine (Obtaining Digits of a Number in DX:AX registers)

- DX:AX registers contain the year (Example for testing purpose : DX=0, AX=2023).
- By dividing DX:AX with 10 in a loop, digits of the year are obtained from righ-to-left.
- At each loop iteration, remainder (DL register) is converted to ASCII and stored to string buffer.



Example of Loop Iterations

Loop Iterations	Operations
Initial	DX:AX = **2023**

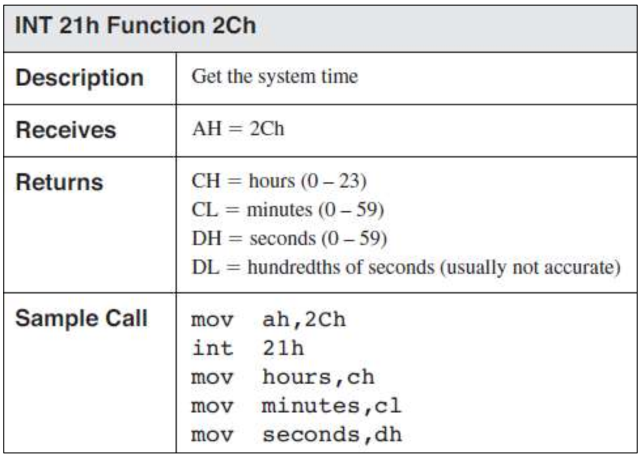
1	DX:AX = 2023
	AX = 2023 / 10 = 202
	DL = 3 (remainder)

2	DX:AX = 202
	AX = 202 / 10 = 20
	DL = 2 (remainder)

3	DX:AX = 20
	AX = 20 / 10 = 2
	DL = 0 (remainder)

4	DX:AX = 2
	AX = 2 / 10 = 0
	DL = 2 (remainder)

INT 21h , Function 2Ch
(Get system time)



INT 21h Function 2Ch	
Description	Get the system time
Receives	AH = 2Ch

Returns	CH = hours (0 – 23)
	CL = minutes (0 – 59)
	DH = seconds (0 – 59)
	DL = hundredths of seconds (usually not accurate)

Sample Call	`mov ah,2Ch`
	`int 21h`
	`mov hours,ch`
	`mov minutes,cl`
	`mov seconds,dh`

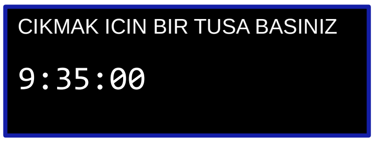
Example3: Displaying System Time
(INT 21h)

- Write a program that displays the system time on screen.
- Display format is : hh:mm:ss
- Program runs in endless loop, exits when user presses a key on the keyboard.
- Using MS-DOS : INT 21h / Function 2ch, Function 02h, and Function 21h.
- Using keypress : INT 16h / Function 01h.

Part1

```
.model small
.data
sonuc db 3 dup(0)
;String buffer for time component displaying.
;8-bit Hour data can be maximum 255.
;255 can be stored as 3 ASCII characters.
;--------------------------------
Cumle  DB   'CIKMAK ICIN BIR TUSA BASINIZ', 13, 10,10, '$'
;ASCII codes: 13 is Carriage Return,
;10 is Line Feed, $ is string end symbol
;--------------------------------
.code
Basla   PROC
.startup
mov dx, OFFSET  Cumle     ;Address of Cumle
mov ah, 09h           ;Function code for displaying string
int 21h
```

Screen
Output



```
CIKMAK ICIN BIR TUSA BASINIZ

9:35:00
```

Part2

```
DEVAM: ;Endless loop
; Read DOS system time
MOV AH, 2Ch
INT 21h
;Returned values:
;CH hour
;CL minute
;DH second
;--------------------------------
PUSH DX                          ;Save backup of DX on stack
MOV AX,0
MOV AL, CH                       ;HOUR
CALL  Ekrana_Yaz  ;Pass parameter in AX
;--------------------------------
;DL should contain one ASCII character to print
MOV DL, ':'                     ;Separator
MOV AH, 2                       ;Write char
INT    21h
;--------------------------------
MOV AX,0
MOV AL, CL   ;MINUTE
CALL  Ekrana_Yaz  ;Pass parameter in AX
;--------------------------------
MOV DL, ':'    ;Separator
MOV AH, 2      ;Write char
INT    21h
```

Part3

```
POP DX    ;Restore backup of DX from stack
MOV AX,0
MOV AL, DH   ;SECOND
CALL  Ekrana_Yaz  ;Pass parameter in AX
;--------------------------------
;Set the cursor to first location on screen row
;for overwriting the time.
MOV DL, 13   ;Carriage return
MOV AH, 2    ;Write char
INT    21h
;-----------------------------------------------
; Detect a keypress.
MOV AH, 01      ;Check for key press
INT  16h        ;Using INT 16H
JNE  SON        ;Jump if ZF not equal 1
;ZF=0,if there is a key press
;ZF=1 if there is no key press
JMP DEVAM       ;Go to endless loop
;-----------------------------------------------
SON:
.EXIT
Basla   ENDP

Ekrana_Yaz  PROC
; .....  (SAME AS IN DATE DISPLAY PROGRAM)
Ekrana_Yaz ENDP

END Basla    ; End of file
```