CONTENTS

Study of Mouse Interrupt - 33H

Interrupt is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor. The microprocessor responds to that interrupt with an ISR (Interrupt Service Routine), which is a short program to instruct the microprocessor on how to handle the interrupt

Hardware and Software Interrupts – When microprocessors receive interrupt signals through pins (hardware) of microprocessor, they are known as Hardware Interrupts.

Software Interrupts – These are instructions inserted within the program to generate interrupts. There are 256 software interrupts in the 8086 microprocessor. The instructions are of the format INT type, where the type ranges from 00 to FF. The starting address ranges from 00000 H to 003FF H.

DOS Mouse Interrupt 33h

MS DOS is non-graphical command line based operating system. However it supports mouse interfacing system calls. Mouse support is provided by an application called MOUSE.COM. Mouse is widely used in graphical applications and in games. DOS implements mouse interfacing subsystem through software interrupt INT 0x33 call. Below are the list of subroutines under this interrupt

AH = 0 : Initialize mouse driver. Returns: AX = number of buttons

AH = 1 : Turn mouse on

AH = 2 : Turn mouse off

AH = 3 : Report status. BX = Mouse button pressed (1 - left, 2 - right, 3 - centre), CX = X-coordinate, DX = Y-coordinate

Call Interrupt using DOS.H

The geninterrupt() function invokes a software interrupt by specifying the interrupt number. Software interrupts are used to signal the CPU to stop what it's currently doing and execute a specific interrupt handler routine. In DOS, many system-level operations like accessing hardware devices (disks, keyboard, display) were managed through software interrupts.

Code:

```
#include<stdio.h>
#include<dos.h>
void initmouse()
{
 _AX=0;
geninterrupt(0x33);
}
void showmouse()
{
 _AX=1;
geninterrupt(0x33);
}
void hidemouse()
{
 _AX=2;
geninterrupt(0x33);
}
void main()
{
 int c,r,bt;
initmouse();
showmouse();
clrscr();
printf("Mouse on Screen");
 do
 {
  _AX=3;
geninterrupt(0x33);
bt=_BX;
   c=_CX/8;
   r=_DX/8;
gotoxy(24,12);
printf("Mouse at %d col,%d row and Button %d",c,r,bt);
 }while(!kbhit());
hidemouse();
printf("Mouse is hidden");
getch();
}
```

Access Screen Memory

Video Memory

RAM in PC stores the information and programs at the time of PC working. The microprocessor writes the information to be displayed on the screen into the Video memory, whereas the display adapter circuitry transfers this information from Video memory on to the screen. The more pixels are displaying on-screen at once (higher resolution), the more Video RAM it takes on the video card to track the colors for all of the pixels. And the more colors we are displaying, the more RAM it takes to track the color for each pixel. A 64K segment of memory is assigned for use with text video modes. This segment starts at address B8000000 and ends at B800:FFFF. In Mode 3, one byte represents one character and next byte represents its color on the screen.

Screen Memory Address Mapping

Address	Value
B8000000	First Character on screen
B8000001	Color of the First Character
B8000002	Second Character on screen
B8000003	Color of the Second Character
B8000004	Third Character on screen
B8000005	Color of the Third Character
*	
*	
*	
B8000FA0	Color of the Last Character

Code:

```
#include<stdlib.h>
#include<conio.h>
#include<dos.h>
char far *s = (char far *)0xB8000000;

void showchar(int r, int c, char ch)
{
   *(s+r*160+c*2)=ch;
}

void main() {
        clrscr();
        showchar(15,8,'I');
        showchar(15,10,'M');
        showchar(15,12,'A');
        showchar(15,14,'N');
        getch();
}
```

Study of Video Interrupt 10H

BIOS INT 10H PROGRAMMING

- INT 10H subroutines are in the ROM BIOS of the 8086-based PC.
- Depending on the value put in AH many function associated with the manipulation of screen text or graphics is performed.
- Among these functions, clearing the screen, changing the cursor position, change the screen color and drawing lines on the screen.

The following table shows subroutine present in the Interrupt 10H

Function	Function code	Parameters	Return
Set video mode	AH=00h	AL = video mode	AL = video mode
Set cursor position	AH=02h	BH = Page Number, DH = Row, DL = Column	
Get cursor position and shape	AH=03h	BH = Page Number	AX = 0, DH = Row, DL = Column
Write character and attribute at cursor position	AH=09h	AL = Character, BH = Page Number, BL = Color, CX = Number of times to print character	

Code:

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
void setmode(int n)
{
  _AH=0;
  _AL=n;
geninterrupt(0x10);
}
void setcursor (char r,char c)
{
  _AH=2;
  _DH=r;
  _DL=c;
geninterrupt(0x10);
}
void myprint(char ch,char color)
{
  _AH=9;
  _AL=ch;
  _BH=0;
  _CX=1;
  _BL=color;
geninterrupt(0x10);
}
void main()
{
        clrscr();
        myprint('A',11);
        getch();
        setmode(4);
        printf("Number of AAGASC,DCS");
        getch();
        setcursor(10,20);
        printf("IMAN");
        getch();
}
```

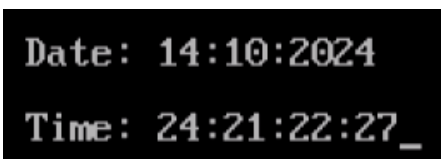
Display System Date & Time

Code:

```
#include <dos.h>
#include <stdio.h>
#include<conio.h>
struct mydate
{
 int year;
 char day;
 char month;
};
struct mytime
{
 char hour;
 char min;
 char hund;
 char sec;
};

void main()
{
        struct mydate dt;
        struct mytime tm;
        getdate(&dt);
        clrscr();
        gotoxy(5,5);
        printf("Date: %d:%d:%d",dt.day,dt.month,dt.year);
        do
        {
                gettime(&tm);
                gotoxy(5,10);
                printf("Time: %d:%d:%d:%d",tm.hour,tm.min,tm.sec,tm.hund);
        }while( !kbhit() );
        getch();
}
```

Output:



Multiplication Table using Command Line Argument

Command-line arguments are simple parameters that are given on the system's command line, and the values of these arguments are passed on to your program during program execution. When a program starts execution without user interaction, command-line arguments are used to pass values or files to it.

The most important function of C is the main() function. It is mostly defined with a return type of int and without parameters.

We can also give command-line arguments in C. Command-line arguments are the values given after the name of the program in the command-line shell of Operating Systems. Command-line arguments are handled by the main() function of a C program.

To pass command-line arguments, we typically define main() with two arguments: the first argument(argc) is the number of command-line arguments and the second(argv[]) is a list of command-line arguments.

```
void main(int argc, char *argv[]) { }
```

After coded the program we can save the file such as TABLE.C, Give build all on compile menu. Then TABLE.EXE will be created on destination directory. In file menu select os shell then control goes to command prompt. There run the TABLE.EXE and give argument then it will run.

Code:

```
void main(int na,char *lst[])
{
  int n,i;
  n=atoi(lst[1]);
  for(i=1;i<=20;i++)
  {
        printf("%dx%d=%d\n",n,i,n*i);
  }
}
```

Output

```
C:\TURBOC3\SOURCE>table 5
5x1=5
5x2=10
5x3=15
5x4=20
5x5=25
5x6=30
5x7=35
5x8=40
5x9=45
5x10=50
5x11=55
5x12=60
5x13=65
5x14=70
5x15=75
5x16=80
5x17=85
5x18=90
5x19=95
5x20=100

C:\TURBOC3\SOURCE>_
```

Evaluation of Arithmetic Expression

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
int evalvuate(char *expr) {
 int i;
 int cn = 0;
 int result = 0;
 int sign = 1;
 char c;
 for (i = 0; expr[i] != '\0'; i++) {
c = expr[i];
if (isdigit(c)) {
 cn = (cn * 10) + (c - '0');
}
if (c == '+' || c == '-' || expr[i + 1] == '\0') {
 result += sign * cn;
 cn = 0;
 sign = (c == '-') ? -1 : 1;
}
 }
 return result;
}
int main() {
 char exp[100];
 int result;
 printf("Enter an arithmetic expression: ");
 fgets(exp, 100, stdin);
 result = evalvuate(exp);
 printf("Result: %d\n", result);
 return 0;
}
```

Output :