# Lab 9

## Software Interrupts

After completing this lab

- Students will know about interrupts and their types.
- Students will know how software interrupts are invoked.
- Students will know how software interrupts are serviced.
- Students will know various DOS and BIOS interrupts.

## Interrupts

The notion of "interrupt" was originally conceived to allow devices to interrupt the current operation of the CPU. For example, whenever a key is pressed, the 8086 must be notified to read a key code from the keyboard buffer. Such interrupts are called hardware interrupts. On the other hand, the software interrupts are generated by the programs to get a service from the operating system or the BIOS. For example, to read a file from disk or a character from the keyboard

Another type of interrupt is an exception, which is generated automatically when an error occurs during the execution of an instruction. For example, divide by zero or overflow.

In this manual, we are going to discuss only software interrupts. The software interrupts are invoked by using an "INT" instruction, which is followed by an interrupt number. There are 256 interrupts in a system. Some are reserved for exceptions, and the rest can be used as hardware or software interrupts.

The processor uses the interrupt number to index the Interrupt Vector Table (IVT) to get the address of a special type of procedure called an Interrupt Service Routine (ISR). The ISR is executed whenever the interrupt is generated. There are 256 entries in IVT, and each entry stores the logical address of a respective ISR. IVT resides in memory at physical address 0x00000. Each entry stores the logical address, which is 4 bytes; the total size of the IVT is 256 x 4B = 1 KB. To index the IVT, the processor multiplies the interrupt number by 4 to get the physical address of the entry against that interrupt number.

The control transfer to an ISR is similar to that of a procedure call. Before transferring control to the ISR, the processor first saves the address of the next instruction on the stack, called the return address. The 8086 also saves the FLAG register on the stack. When the ISR returns, the FLAG register and return address are restored from the stack.

### BIOS and DOS Interrupts

There are some interrupts that are provided by the BIOS to perform basic I/O operations, and some are provided by the DOS. There are multiple services against each interrupt number. The following table shows few interrupts with few of its services. The service number is moved to the AH register before invoking an interrupt instruction. Depending on your needs, you may find more interrupts and services in your textbook.

| Interrupt# | Service# | Description | BIOS/DOS |
|---|---|---|---|
| 0x21 | 0x1 | read character from standard input, with echo, result is stored in **AL**. if there is no character in the keyboard buffer, the function waits until any key is pressed. | DOS |
| 0x21 | 0x7 | read character from standard input, without echo, result is stored in **AL**. if there is no character in the keyboard buffer, the function waits until any key is pressed. | DOS |

| | | | |
|---|---|---|---|
| 0x21 | 0x2 | write character to standard output device.<br>entry: **DL = character** to write. After execution AL = DL. | DOS |
| 0x21 | 0x9 | Print string at DS:DX on screen. String must be terminated by '$'. | DOS |
| 0x21 | 0x4C | Terminate program and return control to DOS | DOS |
| 0x10 | 0x2 | Set Cursor position.<br>DH ← ROW#<br>DL ← COL #<br>BH ← Page# (i.e., 0) | BIOS |
| 0x10 | 0x9 | Display character & attributes on current cursor position.<br>AL = character to display.<br>BH = page number( i.e., 0).<br>BL = Color attributes.<br>CX = number of times to write character. | BIOS |

Example#1: Program to write “Hello” on screen and terminate itself using interrupts.

```
.model small

.stack 100h

.data
Msg db “Hello$”
.code

Mov ax,@data
Mov ds,ax

Mov ah,0x9                ;display string on screen
Mov dx,offset msg
Int 0x21

Mov ah,0x4c               ;terminating program
Int 0x21
```

Example#2: Program to read a single-digit number as character from keyboard and convert it to a number and store in variable.

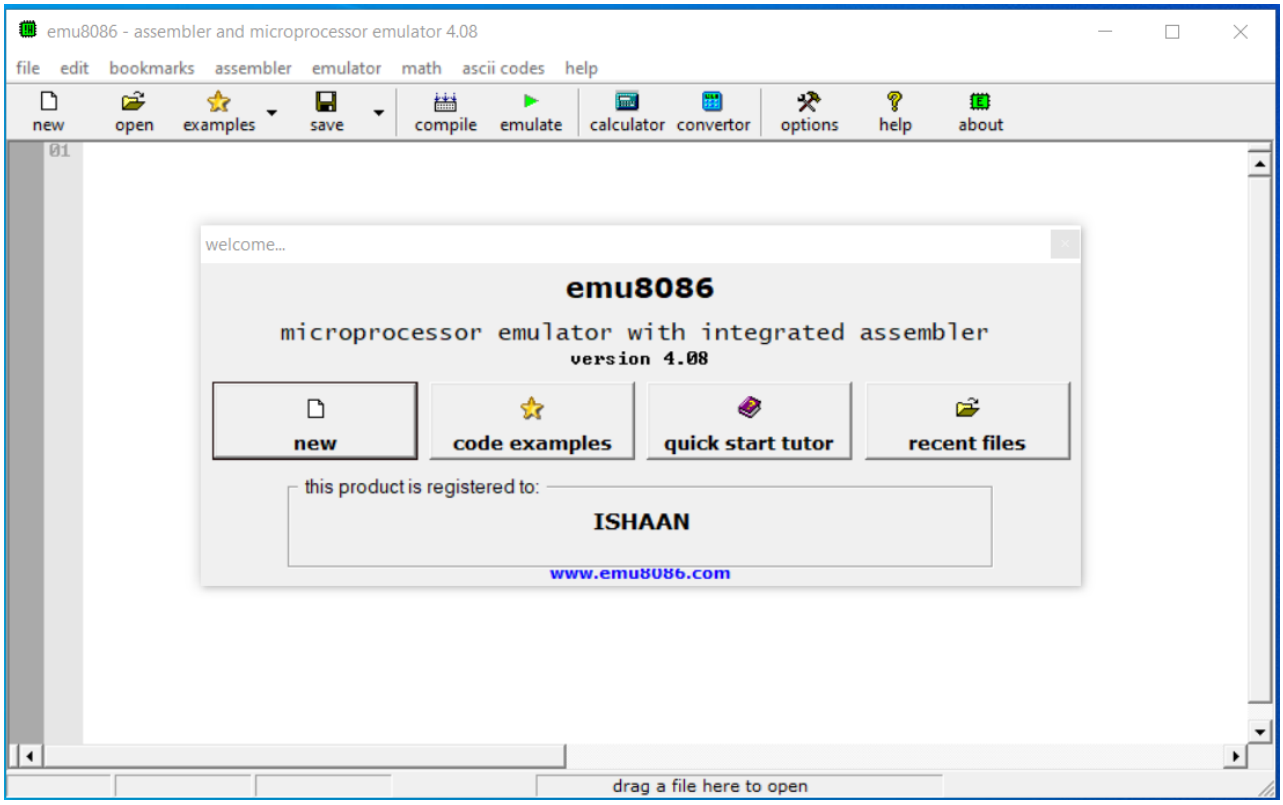
```
.model small

.stack 100h

.data

n db ?

.code

Mov ax,@data
Mov ds,ax

mov ah,0x1    ; read character from keyboard
int 0x21

sub al,0x30    ; subtract 0x30 from the character to make it a number
mov n,al       ; store the number to variable

mov ah,0x4c
int 0x21
```

## Emu8086 Tutorial Step by Step

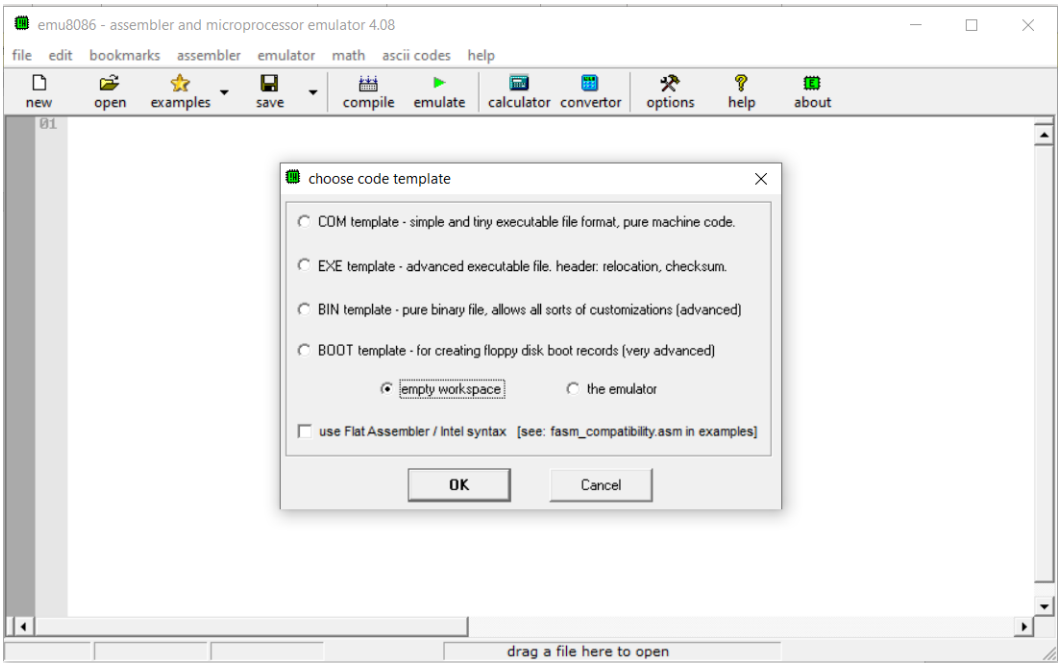
### Step-1

Double click on the icon on the desktop 

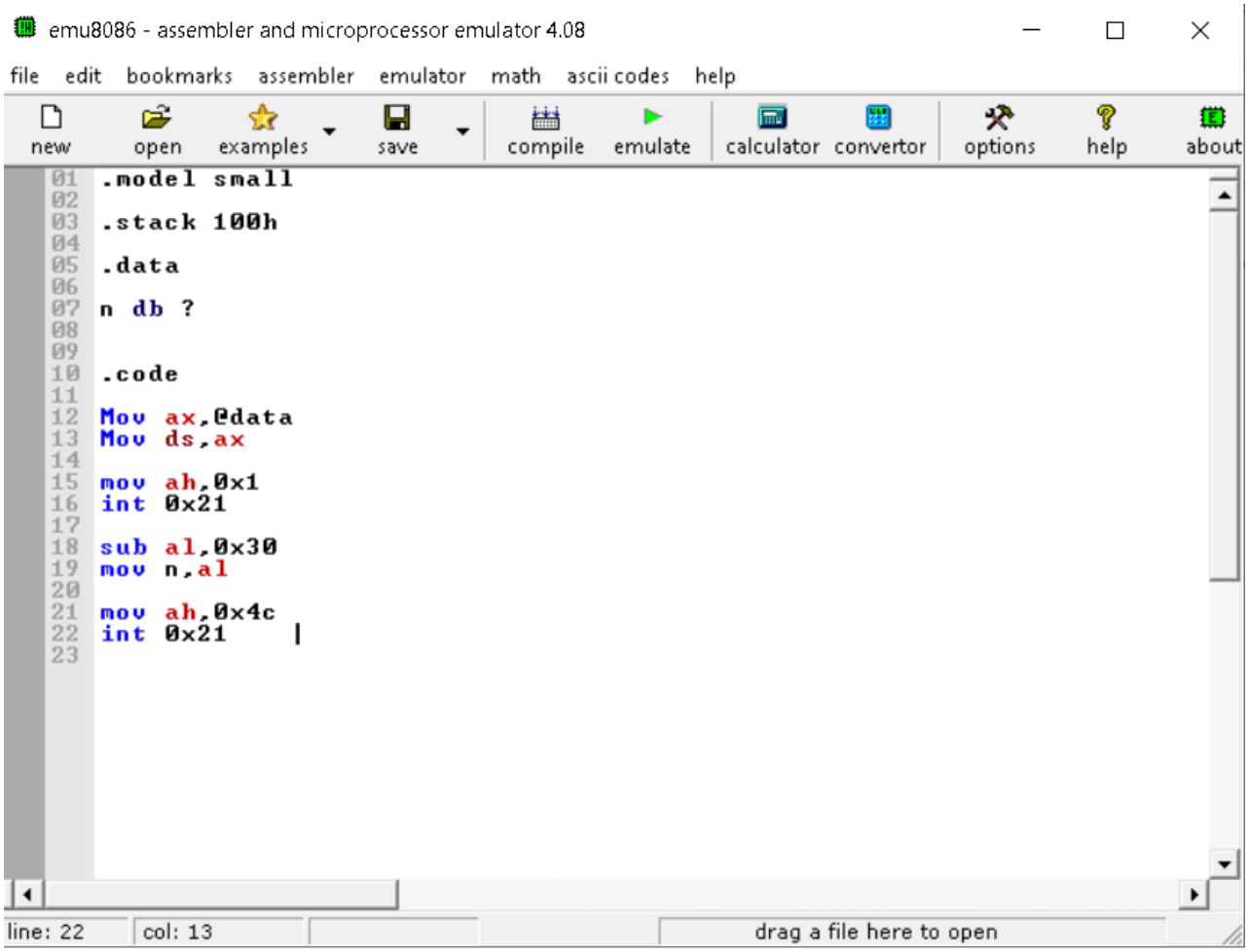

### Step-2
### The following window will appear. Click on new.



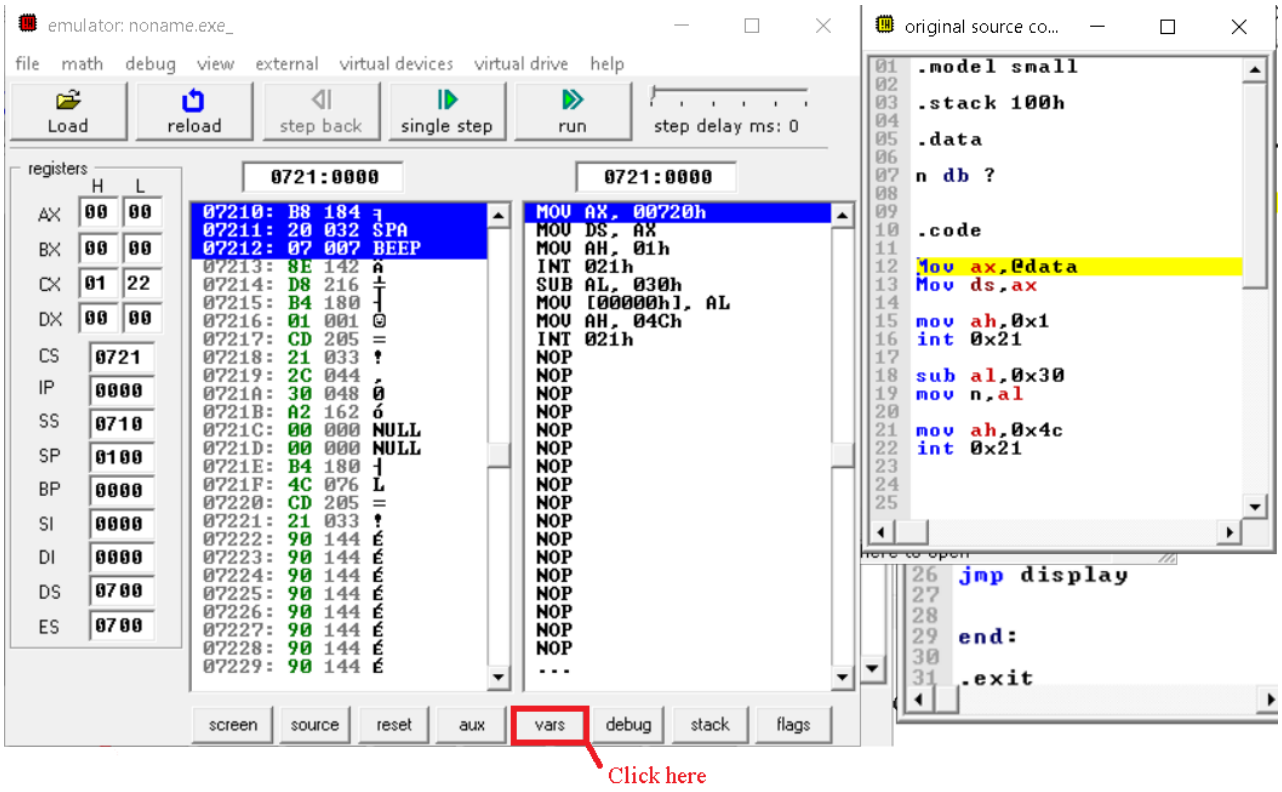
## Step-3: Click on empty workspace and press OK.



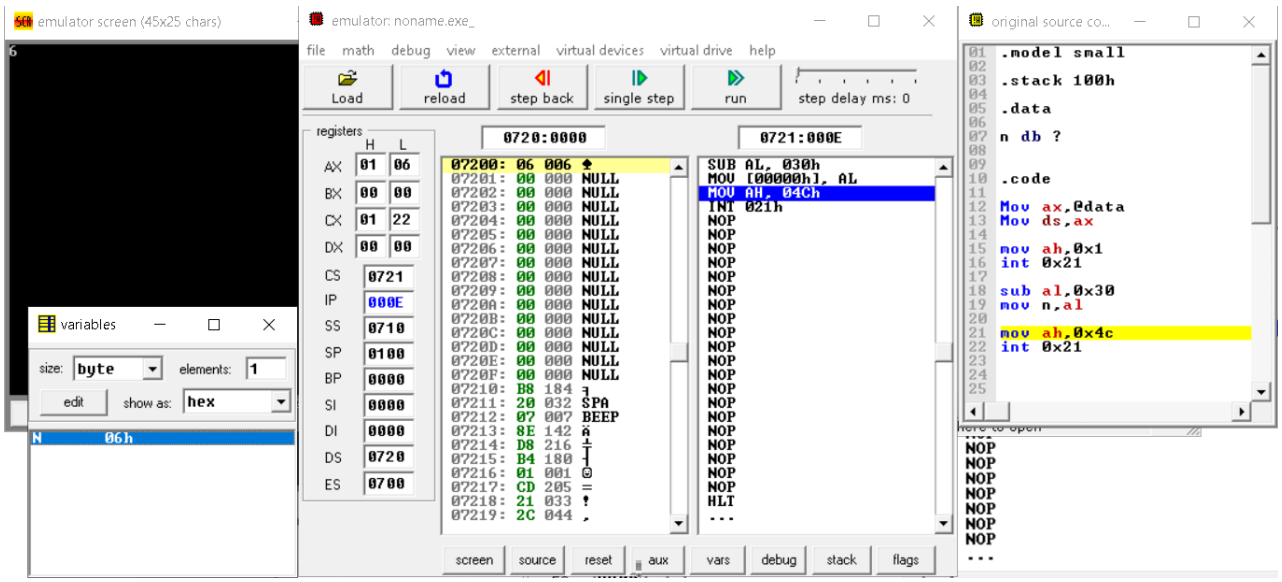

## Step-4: Type the code given in example#2 and click on emulate.



## Step-5: Click on the vars button.




## Step-6: Keep clicking on "Single step" till the last instruction and provide input by pressing a numeric key when prompted.



## Practice Exercise

### Task-1

Write a code that inputs a 16-bit integer number in hexadecimal radix from the keyboard and stores it in a variable.

### Task-2

Write a code that prints a number stored in a 16-bit variable on screen in a hexadecimal radix.

### Task-3

Write a program that inputs two 16-bit hexadecimal numbers from the user and displays their sum on the screen.