

CSE2006

Microprocessor & Interfacing

Module – 2 & 3

Introduction to ALP

Advanced ALP

Dr. E. Konguvel

Assistant Professor (Sr. Gr. 1),
Dept. of Embedded Technology,
School of Electronics Engineering (SENSE),
konguvel.e@vit.ac.in
9597812810



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Module 3: Advanced ALP

- **Interrupts**
- Interrupt Programming
 - DOS
 - BIOS
- File Management

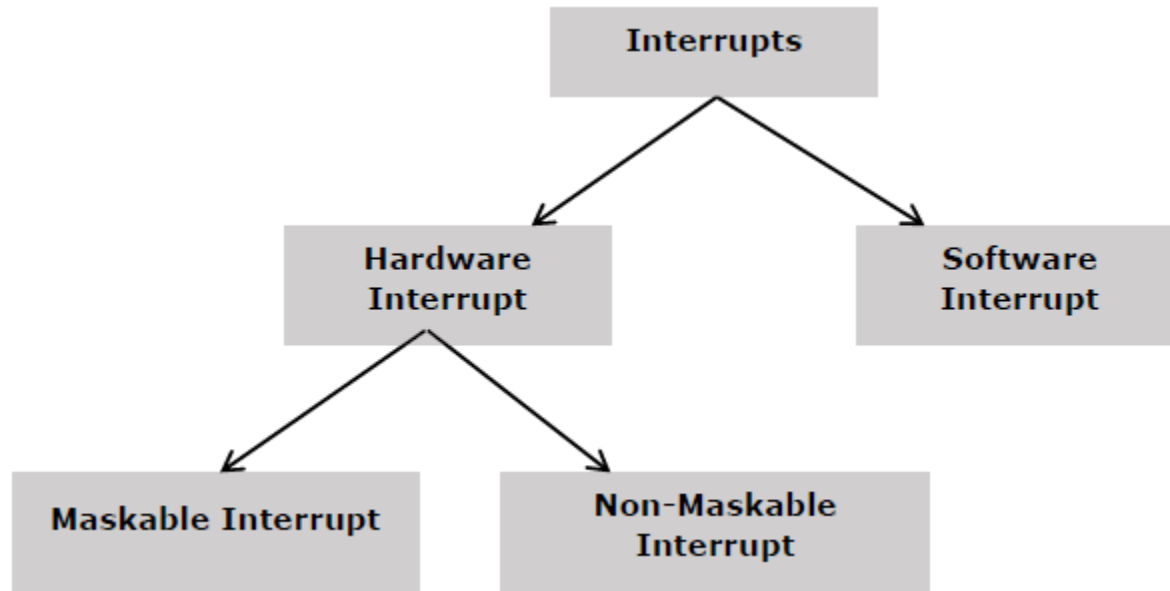
Interrupts

Need for Interrupt:

- Microprocessors allow normal program execution to be interrupted in to carry out a specific task/work in following ways:
 - By an external signal generated by a peripheral.
 - By an internal signal generated by a special instruction in the program.
 - By an internal signal generated due to an exceptional condition which occurs while executing an instruction.
- The external interrupts are used to implement the interrupt driven data transfer scheme.
- The interrupts generated by special instructions are called software interrupts and they are used to implement system services/calls (or monitor services/calls).
- The interrupts generated by exceptional conditions are used to implement error conditions in the system.

Interrupts

Classification of Interrupt:



- Hardware: INTR, NMI pins
- Software (All NMI): INT n instructions (Range 255_{10})
- When an interrupt signal is accepted by the processor, if the program control automatically branches to a specific address (vector address), the interrupt is vectored.

Interrupts

Hardware Interrupts:

- The interrupts initiated by applying appropriate signals to INTR and NMI pins of 8086 are called hardware interrupts.
- When a high signal is applied to the INTR pin and the hardware interrupt is enabled/unmasked, then the processor runs an interrupt acknowledge cycle to get the type number ($0 - 255_{10}$) of the interrupt from the device which sends the interrupt signal.
- Hardware interrupts through INTR are maskable ($IF = 0$) and unmaskable ($IF = 1$)
- Priority: Software > NMI > INTR

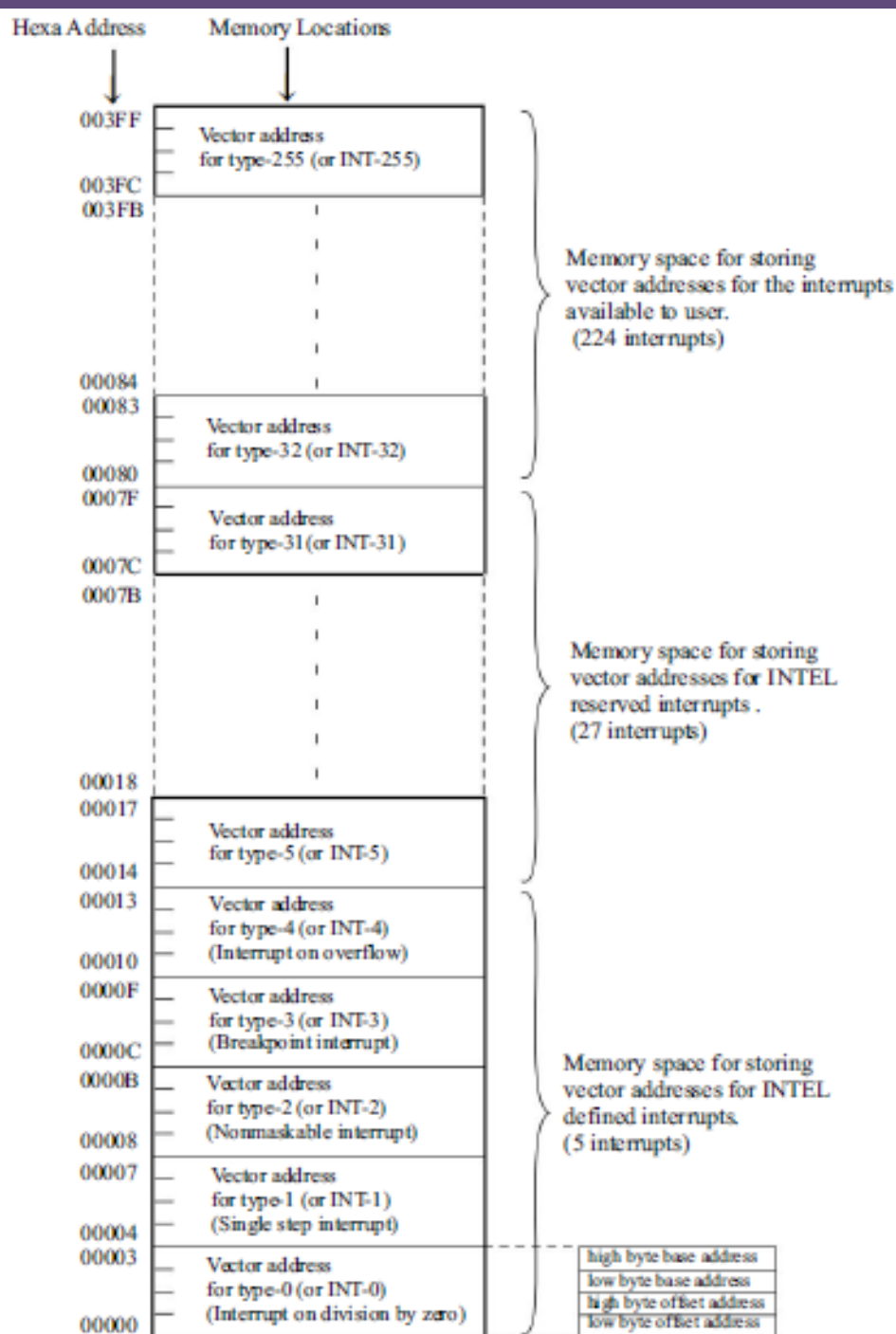
Interrupts

Interrupt Vector Table:

- For each and every interrupt to be implemented, the system designer has to write an Interrupt Service Subroutine (ISS) and store them in memory.
- Create an interrupt vector table in the first 1kb memory space (00000_H to $003FF_H$)
- 16-bit offset and 16-bit segment base address of ISS are stored in four consecutive memory locations, called vector addresses.
- For storing the vector addresses of all 256 interrupt types, the vector table requires 1kb ($256 \times 4 = 1\text{kb}$) memory space.
- Memory address is given by multiplying type number by four and sign extending it to 20-bit.

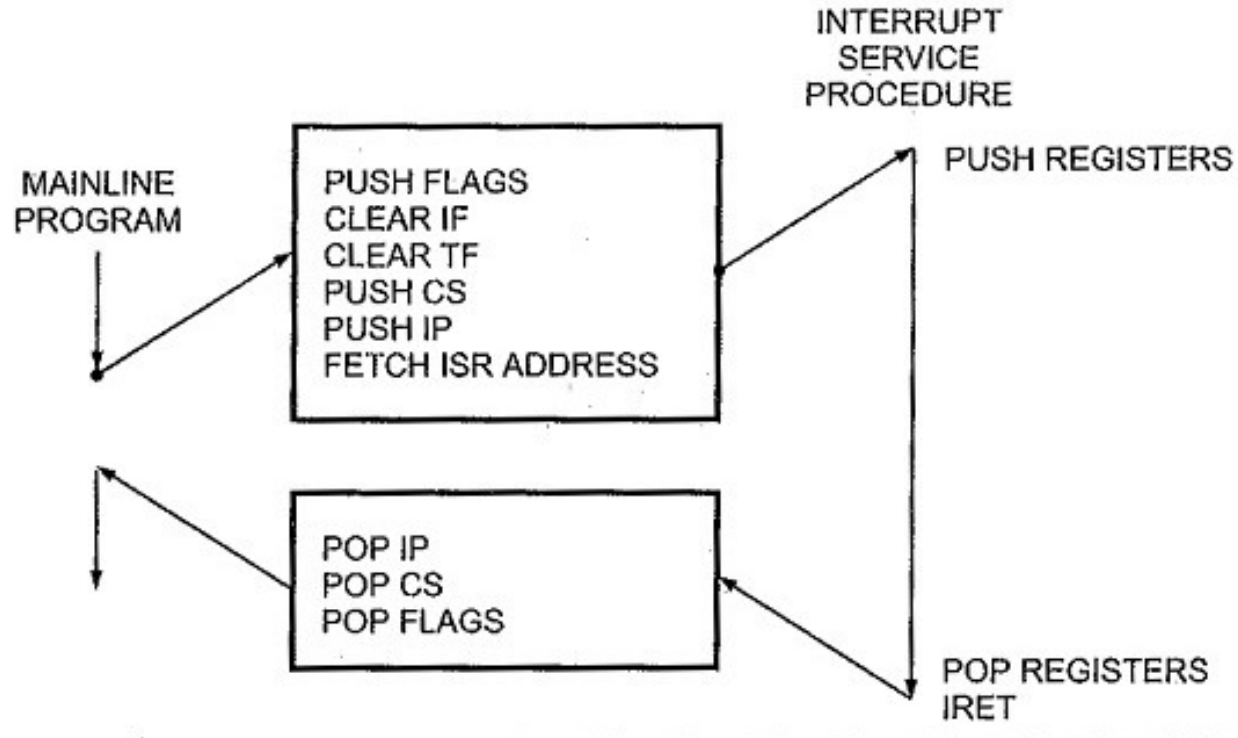
Interrupts

Interrupt Vector Table:



Interrupts

Servicing an Interrupt:



Module 3: Advanced ALP

- Interrupts
- **Interrupt Programming**
 - DOS
 - BIOS
- File Management

Advanced ALP

Introduction to Interrupt Programming:

- While executing the programs, the IO devices like keyboard, monitor, printer,.. can be used as interactive devices for input data and output data.
- Done using predefined interrupts: Type 0 – Type 4.
- Reserved for advanced services/calls (Type 5 – Type 31)
- Non-predefined interrupts (Type 32 – Type 255) can be defined by user.
- Classification:
 1. Interrupts generated from peripherals or exceptions.
 2. Interrupts for services (system calls) through software interrupts.
 3. Interrupts used to store pointers to the device parameters.

Interrupt Programming

Hardware or Exception Interrupts:

Interrupt number	Function assigned
INT 00H	Division by zero
INT 01H	Single-step
INT 02H	Nonmaskable
INT 03H	Breakpoint
INT 04H	Overflow
INT 05H	Print screen
INT 06H	Reserved
INT 07H	Reserved
INT 08H	Timer
INT 09H	Keyboard
INT 0AH to INT 0DH	Hardware Interrupts
INT 0EH	Diskette
INT 0FH	Hardware Interrupt

Interrupt Programming

Software Interrupts For Implementing System Calls:

Interrupt number	Function assigned
INT 10H to INT 17H	BIOS Interrupts
INT 18H	ROM - BASIC
INT 19H	Bootstrap
INT 1AH	Time IO
INT 1BH	Keyboard Break
INT 1CH	User timer Interrupt
INT 20H to INT 2FH	DOS Interrupts
INT 67H	Expanded Memory Functions

Interrupt Programming

Interrupts used to Store Pointers to Device Parameters:

Interrupt number	Function assigned
INT 1DH	Video Parameters
INT 1EH	Diskette Parameters
INT 1FH	Graphics Characters
INT 41H	Hard Disk-0 Parameters
INT 46H	Hard Disk-1 Parameters
INT 44H	EGA Graphic Characters
INT 4AH	User Alarm Address
INT 50H	CMOS Timer Interrupt

DOS Interrupts

Introduction:

- DOS (**D**isk **O**perating **S**ystem) provides a large number of procedures to access devices, files, memory and process control services.
- DOS interrupt INT 21H provides a large number of services.
- Steps:
 1. Load function code in AH-register. Sub-function in AL-register.
 2. Load the other registers as indicated in the DOS service formats.
 3. Prepare buffers, ASCIIZ (ASCII string terminated by zero) and control blocks , if necessary.
 4. Set the location of Disk Transfer Area, if necessary.
 5. Invoke DOS service INT 21H.
 6. The DOS service will return the required parameters in the specified registers.

DOS Interrupts

DOS Interrupts:

Interrupt type	Service provided by the interrupt
INT 20H	Program Terminate
INT 21H	DOS services (DOS system call)
INT 22H	Terminate Address
INT 23H	Control Break Address
INT 24H	Critical Error Handler Address
INT 25H	Absolute Disk Read
INT 26H	Absolute Disk Write
INT 27H	Terminate and Stay Resident (TSR)
INT 28H	DOS time slice
INT 2EH	Perform DOS Command
INT 2FH	Multiplex Interrupts

INT 21H

Multi-Function Interrupt:

1. To input a character
2. To input a string
3. To display a character
4. To display a string
5. To terminate a program

INT 21H

Input a Character

MOV AH, 01H

INT 21H

Character from Screen to 'AL'

ASCII Value for 'Z' is 90_D (5A_H)

```
ASSUME CS: CODE, DS: DATA
```

```
DATA SEGMENT
```

```
DATA ENDS
```

```
CODE SEGMENT
```

```
START: MOV AX, DATA
```

```
MOV DS, AX
```

```
MOV AH, 01H
```

```
INT 21H
```

```
MOV AH, 4CH
```

```
INT 21H
```

```
CODE ENDS
```

```
END START
```

- registers

	H	L
AX	4C	5A
BX	00	00
CX	00	00
DX	00	00

SCR emulator screen (80x25 chars)

Z

INT 21H

Input a String

MOV AH, 0AH

INT 21H

String from Screen to Memory

```
ASSUME CS:CODE, DS:DATA
```

```
DATA SEGMENT
```

```
    BUFFER DB 08
```

```
    DB 09 DUP(?)
```

```
DATA ENDS
```

```
CODE SEGMENT
```

```
START: MOV AX, DATA
```

```
MOV DS, AX
```

```
MOV DX, OFFSET BUFFER
```

```
MOV AH, 0AH
```

```
INT 21H
```

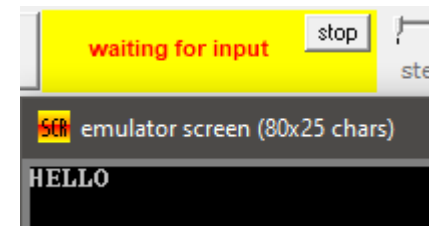
```
MOV AH, 4CH
```

```
INT 21H
```

```
CODE ENDS
```

```
END START
```

08	XX	XX	XX	XX	XX	XX	XX	XX	XX
----	----	----	----	----	----	----	----	----	----



0710:0000	update					<input checked="" type="radio"/> table	<input type="radio"/> list				
0710:0000	08	05	48	45	4C	4C	4F	0D-00	00	00	00
			H	E	L	L	O				

Number of Characters

Carriage Return

INT 21H

Display a Character

MOV AH, 02H

INT 21H

Content from 'DL' to Screen

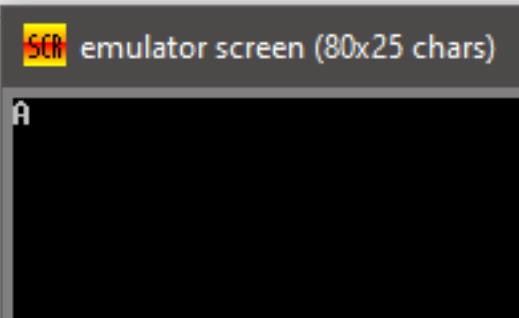
```
ASSUME CS: CODE, DS: DATA

DATA SEGMENT
    MESSAGE DB "A"
DATA ENDS

CODE SEGMENT
    START: MOV AX, DATA
           MOV DS, AX
           MOV DL, MESSAGE
           MOV AH, 02H
           INT 21H
           MOV AH, 4CH
           INT 21H
        CODE ENDS

END START
```

registers		H	L
AX		02	41
BX		00	00
CX		00	21
DX		00	41

The emulator screen is a black rectangle with the character 'A' displayed in the top-left corner. Above the screen, the text 'emulator screen (80x25 chars)' is visible.

INT 21H

Display a String

MOV AH, 09H

INT 21H

Conditions:

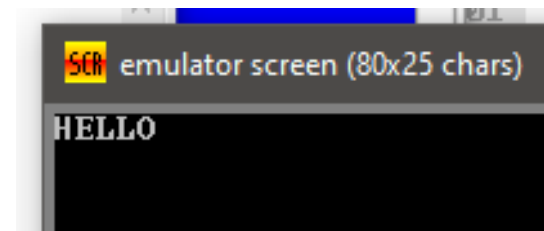
1. Offset address of string to be stored in DX
2. String must be terminated by '\$'

```
ASSUME CS: CODE, DS: DATA

DATA SEGMENT
    MESSAGE DB "HELLO$"
DATA ENDS

CODE SEGMENT
    START: MOV AX, DATA
           MOV DS, AX
           MOV AH, 09H
           MOV DX, OFFSET MESSAGE
           INT 21H
           MOV AH, 4CH
           INT 21H
CODE ENDS

END START
```



INT 21H

Terminate a Program

MOV AH, 4CH

INT 21H

```
ASSUME CS: CODE, DS: DATA

DATA SEGMENT
    MESSAGE DB "HELLO$"
DATA ENDS

CODE SEGMENT
    START: MOV AX, DATA
           MOV DS, AX
           MOV AH, 09H
           MOV DX, OFFSET MESSAGE
           INT 21H
           MOV AH, 4CH
           INT 21H
    CODE ENDS

END START
```

BIOS Interrupts

Introduction:

- The basic interface between the hardware and software is provided by a program stored in ROM called BIOS program. (**B**asic **I**nput **O**utput control **S**ystem).
- Consists of a large number of procedures to access various hardwares. (DOS uses BIOS interrupts to control the hardware)
- Steps:
 1. Load function number in the AH-register. Subfunction in AL-register.
 2. Load the other register as indicated in the BIOS service formats.
 3. Prepare buffers, ASCIIZ (ASCII string terminated by zero) and control blocks , if necessary.
 4. Invoke BIOS call.
 5. The BIOS service will return the required parameters in the specified register.

BIOS Interrupts

BIOS Interrupts:

Interrupt type	Service name
INT 10H	Video services
INT 11H	Machine configuration
INT 12H	Usable RAM Memory size
INT 13H	Disk IO
INT 14H	Serial port IO (RS 232C)
INT 15H	AT services
INT 16H	Keyboard IO
INT 17H	Printer IO

INT 10H

Set Video Mode (AH = 00H):

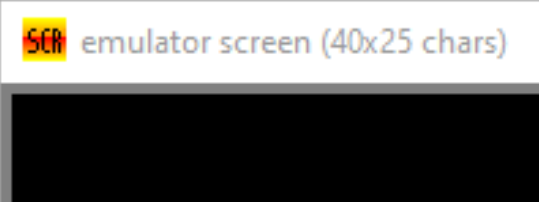
AL = desired video mode.

00h - text mode. 40x25. 16 colors. 8 pages.

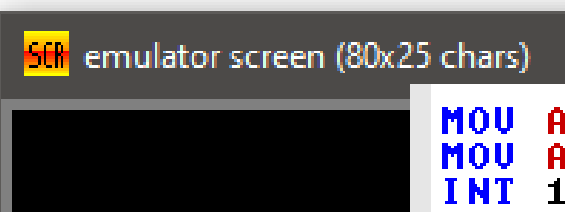
03h - text mode. 80x25. 16 colors. 8 pages.

13h - graphical mode. 40x25. 256 colors. 320x200 pixels. 1 page.

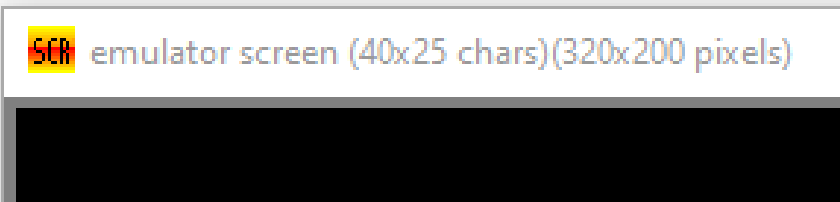
```
MOV AL, 00H
MOV AH, 0
INT 10H
```

A screenshot of an emulator window titled "emulator screen (40x25 chars)". The window contains a black rectangular area representing the video output.

```
MOV AL, 03H
MOV AH, 0
INT 10H
```

A screenshot of an emulator window titled "emulator screen (80x25 chars)". The window contains a black rectangular area representing the video output.

```
MOV AL, 13H
MOV AH, 0
INT 10H
```

A screenshot of an emulator window titled "emulator screen (40x25 chars)(320x200 pixels)". The window contains a black rectangular area representing the video output.

INT 10H

Set Cursor Position (AH = 02H):

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    MESSAGE DB "A"
DATA ENDS
CODE SEGMENT
START: MOV AX, DATA
    MOV DS, AX

    MOV AL, 03
    MOV AH, 0
    INT 10H

    MOV AH, 02
    MOV BH, 0
    MOV DH, 12
    MOV DL, 40
    INT 10H

    MOV DL, MESSAGE
    MOV AH, 02H
    INT 21H

    MOV AH, 4CH
    INT 21H
CODE ENDS
END START
```

BH – Video Page No. (0)

DH – Row

DL – Column

