Microprocessor Based Systems

Spring 2013

Department of Electrical Engineering

University of Gujrat

PROGRAM STRUCTURE

- Assembly language program occupies code, data and stack segment in memory
- Same organization reflected in assembly language programs as well
- Code data and stack are structured as program segments
- Program segments are translated to memory segments by assembler

MEMORY MODELS

Size of code and data, a program can have is determined by specifying a memory model using .MODEL directive

```
MODEL      memory_model
```

Model	Description
SMALL	code in one segment data in one segment
MEDIUM	code in more than one segment data in one segment
COMPACT	code in one segment data in more than one segment
LARGE	code in more than one segment data in more than one segment no array larger than 64k bytes
HUGE	code in more than one segment data in more than one segment arrays may be larger than 64k bytes

DATA SEGMENT

- A program's data segment contains all the variable definitions.
- Constant definitions are often made here as well, but they may be placed elsewhere in the program since no memory allocation is involved.

`.data` directive to declare a data segment

```
.DATA
WORD1          DW 2
WORD2          DW 5
MSG            DB 'THIS IS A MESSAGE'
MASK           EQU 10010111B
```

STACK SEGMENT

- The purpose of the stack segment declaration is to set aside a block of memory (the stack area) to store the stack.
- The stack area should be big enough to contain the stack at its maximum size.

```
 .STACK     100H
```

- If size is omitted, by default 1kB is set aside

CODE SEGMENT

- The code segment contains a program’s instructions.

```
 .CODE name
```

- Inside a code segment, instructions are organized as procedures.

```
name PROC
; body of the procedure
name ENDP
```

- The last line in the program should be the END directive, followed by name of the main procedure.

```
MAIN            PROC
; instructions go here
MAIN            ENDP
; other procedures go here
```

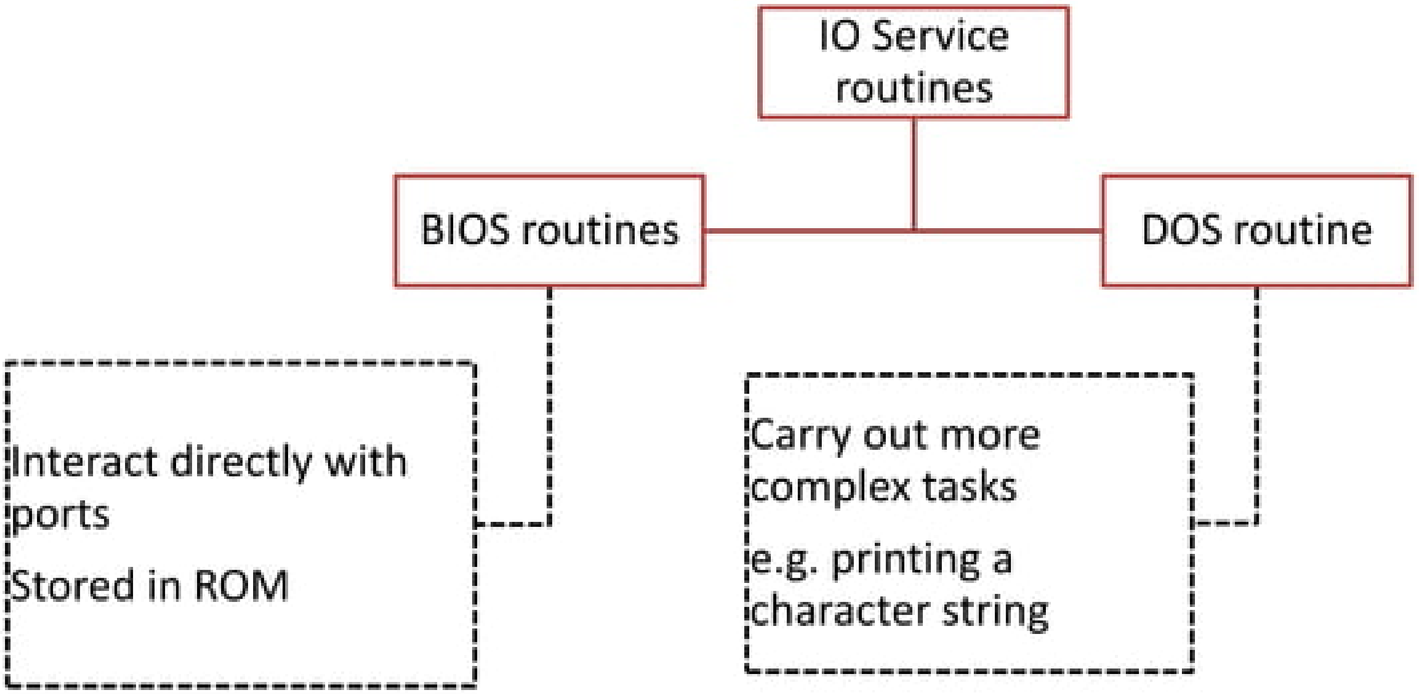
PUTTING IT TOGETHER

```
.MODEL   SMALL
.STACK   100H
.DATA
; data definitions go here
.CODE
MAIN     PROC
; instructions go here
MAIN     ENDP
; other procedures go here
END      MAIN
```

INPUT AND OUTPUT INSTRUCTIONS

- CPU communicates with the peripherals through IO ports
  - IN and OUT instructions to access the ports directly
    - Used when fast IO is essential
    - Seldom used as
      - Port address varies among compluter models
      - Easier to program IO with service routine

IO SERVICE ROUTINES



IO Service routines
BIOS routines
DOS routine
Interact directly with ports
Stored in ROM
Carry out more complex tasks
e.g. printing a character string

INT

- I/O service routines
  - The **B**asic **I**nput/**O**utput **S**ystem (BIOS) routines
  - The DOS routines
- The **INT (interrupt)** instruction is used to invoke a DOS or BIOS routine.
- `INT 16h`
  - invokes a BIOS routine that performs keyboard input.

INT 21H

- INT 21h may be used to invoke a large number of DOS functions.
- A particular function is requested by placing a function number in the AH register and invoking INT 21h.

FUNCTION 1: SINGLE-KEY INPUT

Input:

AH = 1

Output:

AL = ASCII code if character key is pressed
 = 0 if non-character key is pressed

FUNCTION 1: SINGLE-KEY INPUT

```
MOV   AH, 1              ; input key function
INT   21h          ; ASCII code in AL
```

FUNCTION 2: DISPLAY A CHARACTER OR EXECUTE A CONTROL FUNCTION

Input:

AH = 2

DL = ASCII code of the display character or control character

Output:

AL = ASCII code of the display character or control character

FUNCTION 2: DISPLAY A CHARACTER OR EXECUTE A CONTROL FUNCTION

```
• MOV    AH, 2      ; display character function
  MOV    DL, '?'    ; character is '?'
  INT    21h        ; display character
```

PRINCIPAL CONTROL CAHARCTERS

ASCII Code HEX	Symbol	Function
7	BEL	beep
8	BS	backspace
9	HT	tab
A	LF	line feed (new line)
D	CR	carriage return (start of current line)

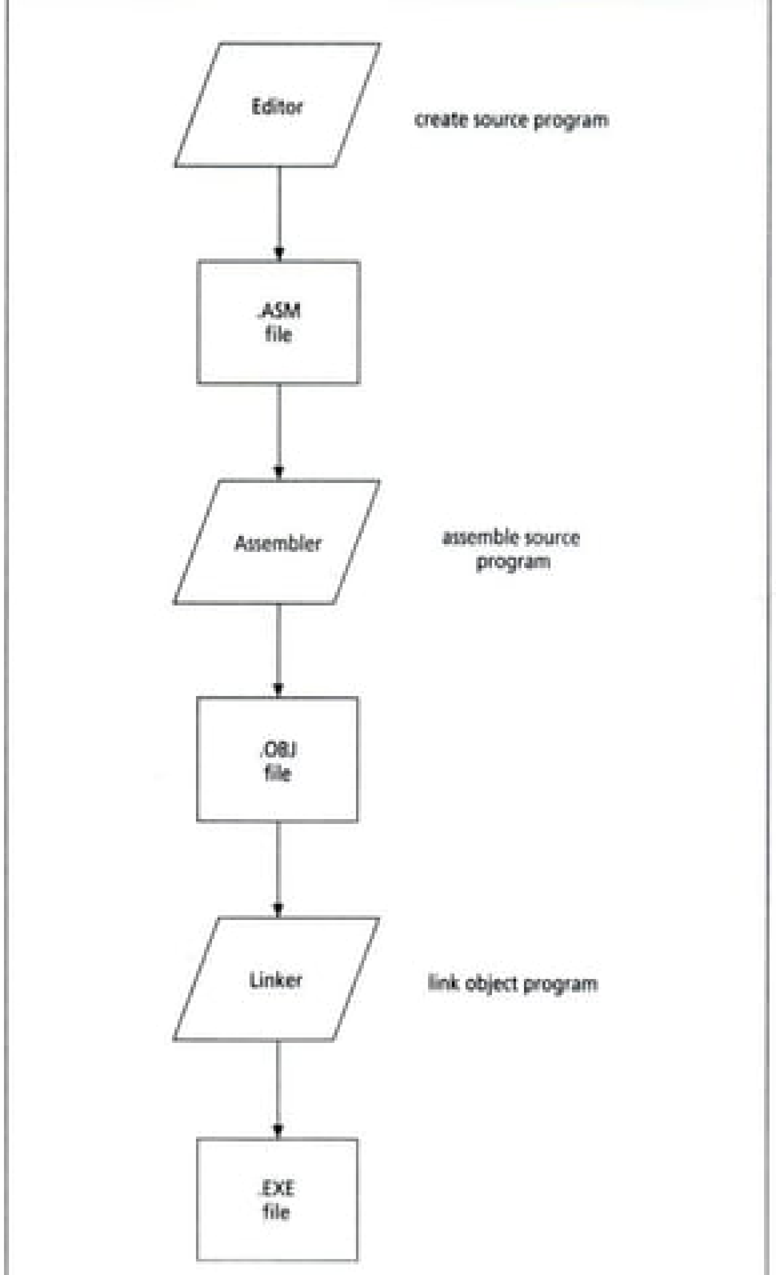
A FIRST PROGRAM

- ECH.ASM will read a character from the keyboard and display it at the beginning of the next line.
- The data segment was omitted because no variables were used.
- When a program terminates, it should return control to DOS.
- This can be accomplished by executing INT 21h, function 4Ch.

ASSEMBLY CODE

```
TITLE    ECHO PROGRAM
.MODEL   SMALL
.STACK   100H
.CODE
MAIN     PROC
; display prompt
   MOV   AH, 2              ; display character function
   MOV   DL, '?'            ; character is '?'
   INT   21H                ; display it
; input a character
   MOV   AH, 1              ; read character function
   INT   21H                ; character in AL
   MOV   BL, AL             ; save it in BL
; go to a new line
   MOV   AH, 2              ; display character function
   MOV   DL, 0DH            ; carriage return
   INT   21H                ; execute carriage return
   MOV   DL, 0AH            ; line feed
   INT   21H                ; execute line feed
; display character
   MOV   DL, BL             ; retrieve character
   INT   21H                ; and display it
; return to DOS
   MOV   AH, 4CH            ; DOS exit function
   INT   21H                ; exit to DOS
MAIN     ENDP
   END MAIN
```

PROGRAMMING STEPS



STEP 1. CREATE THE SOURCE PROGRAM FILE

- An editor is used to create the preceding program.
- The .ASM is the conventional extension used to identify an assembly language source file.

STEP 2. ASSEMBLE THE PROGRAM

- The Microsoft Macro Assembler (MASM) is used to translate the source file (.ASM file) into a machine language object file (.OBJ file).
- MASM checks the source file for syntax errors.
- If it finds any, it will display the line number of each error and a short description.
- C:\>MASM *File_Name*;

STEP 3. LINK THE PROGRAM

- The Link program takes one or more object files, fills in any missing addresses, and combines the object files into a single executable file (.EXE file)
- This file can be loaded into memory and run.
- C:\>LINK *File_Name*;

STEP 4. RUN THE PROGRAM

- To run it, just type the run file name.
- C:\>*File_Name*

INT 21H, FUNCTION 9: DISPLAY A STRING

Input:

DX = offset address of string.
The string must end with a ‘$’ character.

LEA

- LEA is used to load effective address of a character string.
- **LEA destination, source**
-
```
MSG     DB          'HELLO!$'
LEA     DX, MSG     ; get message
MOV     AH, 9       ; display string function
INT     21h         ; display string
```

PROGRAM SEGMENT PREFIX

- When a program is loaded into memory, DOS prefaces it 256 byte PSP which contains information about the program
- DOS places segment no of PSP in DS and ES before executing the program
- To correct this, a program containing a data segment must start with these instructions;

```
MOV AX, @DATA
MOV DS, AX
```

Print String Program

```
.MODEL      SMALL
.STACK      100H
.DATA
MSG   DB    'HELLO!$'
.CODE
MAIN  PROC
; initialize DS
  MOV AX, @DATA
  MOV DS, AX              ; intialize DS
; display message
  LEA DX, MSG             ; get message
  MOV AH, 9          ; display string function
  INT 21H            ; display message
; return to DOS
  MOV AH, 4CH
  INT 21H            ; DOS exit
MAIN  ENDP
  END MAIN
```

A CASE CONVERSION PROGRAM

- CASE.ASM begins by prompting the user to enter a lowercase letter, and on the next line displays another message with the letter in uppercase.
- The lowercase letters begin at 61h and the uppercase letters start at 41h, so subtraction of 20h from the contents of AL does the conversion.

CASE CONVERSION PROGRAM

```
.MODEL       SMALL
.STACK 100H
.DATA
CREQU 0DH
LF EQU 0AH
MSG1  DB     'ENTER A LOWER CASE LETTER: $'
MSG2  DB     CR, LF, 'IN UPPER CASE IT IS: '
CHAR  DB     ?, '$'
.CODE
MAIN  PROC
; intialize DS
   MOV       AX, @DATA     ; get data segment
   MOV       DS, AX        ; intialize DS
; print user prompt
   LEA  DX, MSG1      ; get first message
   MOV       AH, 9         ; display string function
   INT  21H           ; display first message
```

```
; input a character and convert to upper case
    MOV        AH, 1             ; read character function
    INT 21H               ; read a small letter into AL
    SUB        AL, 20H                  ; convert it to upper case
    MOV        CHAR, AL      ; and store it
; display on the next line
    LEA DX, MSG2      ; get second message
    MOV        AH, 9             ; display string function
    INT 21H               ; display message and upper case
    letter in front
; DOS exit
    MOV        AH, 4CH
    INT 21H               ; DOS exit
MAIN ENDP
    END MAIN
```

CASE CONVERSION PROGRAM