Assembly Language Program Segment Structure

- Data Segments
  - Storage for variables
  - Variable addresses are computed as offsets from start of this segment
- Code Segment
  - contains executable instructions
- Stack Segment
  - used to set aside storage for the stack
  - Stack addresses are computed as offsets into this segment
- Segment directives

  .data

  .code

  .stack ***size***

Memory Models

.Model *memory_model*

- tiny: code+data <= 64K (.com program)
- small: code<=64K, data<=64K, one of each
- medium: data<=64K, one data segment
- compact: code<=64K, one code segment
- large: multiple code and data segments
- huge: allows individual arrays to exceed 64K
- flat: no segments, 32-bit addresses, protected mode only (80386 and higher)

Program Structure

```
.model small
.stack 100H
.data
  ;declarations
.code
main proc
  ;code
main endp
  ;other procs
end main
```

- Select a memory model
- Define the stack size
- Declare variables

- Write code
  - organize into procedures

- Mark the end of the source file
  - optionally, define the entry point

Program Statements

name operation operand(s) comment

- Operation is a predefined or reserved word
  - › mnemonic - symbolic operation code
  - › directive - pseudo-operation code
- Space or tab separates initial fields
- Comments begin with semicolon

Most assemblers are not case sensitive

- Pseudo-ops to define data or reserve storage
  - › DB - byte(s)
  - › DW - word(s)
  - › DD - doubleword(s)
  - › DQ - quadword(s)

  - DT - tenbyte(s)
- Names can be associated with storage locations

  ANum DB -4

  DW 17

  ONE

  UNO DW 1

  X DD ?
- These names are called variables

Interrupts

- The interrupt instruction is used to cause a software interrupt
  - An interrupt interrupts the current program and executes a subroutine, eventually returning control to the original program
  - Interrupts may be caused by hardware or software
- int *interrupt_number* ;software interrupt
- Output to Monitor
  - DOS Interrupts : interrupt 21h
  - This interrupt invokes one of many support routines provided by DOS
  - The DOS function is selected via AH
  - Other registers may serve as arguments
  - AH = 2, DL = ASCII of character to output
  - Character is displayed at the current cursor position, the cursor is advanced, AL = DL
- Output a String
  - Interrupt 21h, function 09h

- › DX = offset to the string (in data segment)
- › The string is terminated with the '$' character
- › To place the address of a variable in DX, use one of the following
- › lea   DX,theString   ;load effective address
- › mov   DX, offset theString   ;immediate data

- Input a Character
  - › Interrupt 21h, function 01h
  - › Filtered input with echo
  - › This function returns the next character in the keyboard buffer (waiting if necessary)
  - › The character is echoed to the screen
  - › AL will contain the ASCII code of the non-control character
  - › AL=0 if a control character was entered

Example program

Lower to Upper case

```
.model small
.stack 100h
.code
main proc
    mov ah,1
    int 21h
    sub al,32
    mov dl,al
    mov ah,2
    int 21h
    mov ah,4ch
    int 21h
    main endp
    end main
```