Requirements For Coding in Assembly Language

Dr . Shahenda sarhan

Lecturer @ CS Department

- **Comments** : A comment begins with a semicolon (;)

```
MOV    AX,DATASEG          ;Set address of data
MOV    DS,AX               ;Segment in DS
```

- **Reserved Words** :Examples such as MOV, ADD, END, SEGMENT, FAR, SIZE, @Data, and @Model.
- **Identifiers** :an identifier is a name that you apply to items in your program.
  - name :which refer to the address of a data item

    (e.g: FLDD DW 215)
  - Label: which refers to the address of an instruction.

    (e.g: MAIN PROC FAR)
  - The first character of an identifier must be an alphabetic letter or a special character, except for the period.

- **Identifiers**
  - not case sensitive.
  - The maximum length of an identifier is 247characters
  - An identifier can use the following characters:
    - Alphabetic letters: A-Z and a-z
    - Digits: 0-9 (may not be the first character)
    - Special characters: question mark (?)
      underline (_)
      dollar ($)
      at (@)
      period(.) (May not be the first character)

- **Statements** : the two types of statements are

1. **Instructions** such as MOV and ADD, which the assembler translates to object code.
2. **Directives**, which tell the assembler to perform a specific action, such as define a data item.

- **Statements : General format for a statement**

  [identifier] operation [operand(s)] [;comment]

  Example:

  Directive: COUNT DB 1 ;name, operation, operand

  Instruction: MOV AX,0 ;operation, two operands

- **Operand :**
  - The operand (if any) provides information for the operation to act on. For a data item, the operand defines its initial value. For example:

    COUNTER DB 0
  - For an instruction, an operand indicates where to perform the action. An instruction may have no, one, two or three operands. For example:
    - **No operands: RET**
    - **One operand: MUL 10**
    - **Two operands: MOV CX,10**
    - **Three operands: SHRD ECX,EBX,CL**

Directives : Control the way a source program assembles and lists . Generate no machine code

Page directive

```
; Add two numbers and store the results into the third variable
page    60,132          page [length(10-255)],[width(60-132)]
TITLE       A04ASM1 (EXE)  Move and add operations
; -----------------------------------------------
STACK       SEGMENT PARA STACK 'Stack'
      DW       32 DUP(0)
STACK       ENDS
; -----------------------------------------------
DATASEG  SEGMENT PARA 'Data'
      FLDD              DW          215
      FLDE              DW          125
      FLDF              DW          ?
DATASEG  ENDS
; -----------------------------------------------
CODESEG SEGMENT PARA 'Code'
MAIN                    PROC    FAR
            ASSUME  SS:STACK,DS:DATASEG,CS:CODESEG
            MOV     AX,DATASEG              ;Set address of data
            MOV     DS,AX                   ;Segment in DS
            MOV         AX,FLDD             ;Move 0215 to AX
            ADD         AX,FLDE             ;Add  0125 to AX
            MOV         FLDF,AX             ;Store sum in FLDF
            MOV         AX,4C00H                        ;End processing
            INT         21H
MAIN                    ENDP                ;End of procedure
CODESEG ENDS                                ;End of segment
END         MAIN                                        ;End of program
```

Title directive

```
; Add two numbers and store the results into the third variable
page    10,70
TITLE   A04ASM1 (EXE)  Move and add operations
; -----------------------------------------
STACK       SEGMENT PARA STACK 'Stack'
      DW        32 DUP(0)
STACK       ENDS
; -----------------------------------------
DATASEG  SEGMENT PARA 'Data'
      FLDD              DW          215
      FLDE              DW          125
      FLDF              DW          ?
DATASEG  ENDS
; -----------------------------------------
CODESEG SEGMENT PARA 'Code'
MAIN                    PROC    FAR
            ASSUME  SS:STACK,DS:DATASEG,CS:CODESEG
            MOV     AX,DATASEG          ;Set address of data
            MOV     DS,AX               ;Segment in DS
            MOV         AX,FLDD         ;Move 0215 to AX
            ADD         AX,FLDE         ;Add  0125 to AX
            MOV         FLDF,AX         ;Store sum in FLDF
            MOV         AX,4C00H                ;End processing
            INT         21H
MAIN                    ENDP            ;End of procedure
CODESEG ENDS                            ;End of segment
END         MAIN                                ;End of program
```

Segment Directive

- An assem. program in .EXE format consists of one or more segments.
  - A stack segment defines stack storage.
  - A data segment defines data items.
  - A code segment provides for executable code.
- The directive for defining a segment, SEGMENT and ENDS, have the following format:

```
name        SEGMENT    [options]          ;begin segment
name        ENDS                          ;end segment
```

```
segment-name  SEGMENT    [align]    [combine]    ['class']
segment-name  ENDS
```

Segment Directive

- **Alignment type**: indicates the boundary on which the segment is to begin. PARA is typically used and is the default.
- **Combine type:**
  - indicates whether to combine the segment with other segments when they are linked after assembly. Combine types are STACK, COMMON, PUBLIC, and AT.
  - You may use PUBLIC and COMMON where you intend to combine separately assembled programs when linking. When a program is not to be linked with others this option may be omitted or code NONE.
- **Class type:**
  - The class entry is enclosed in apostrophes, is used to group related segments when linking.
  - use 'code' for the code segment, 'data' for the data segment, and 'stack' for the stack segment.

Segment directive

```
; Add two numbers and store the results into the third variable
page    60,132
TITLE       A04ASM1 (EXE)  Move and add operations
; -----------------------------------------
STACK SEGMENT PARA STACK 'Stack'
    DW      32 DUP(0)
STACK   ENDS
; -----------------------------------------------
DATASEG         SEGMENT PARA 'Data'
    FLDD                DW          215
    FLDE                DW          125
    FLDF                DW          ?
DATASEG         ENDS
; -----------------------------------------------
CODESEG         SEGMENT PARA 'Code'
MAIN                    PROC    FAR
            ASSUME  SS:STACK,DS:DATASEG,CS:CODESEG
            MOV     AX,DATASEG              ;Set address of data
            MOV         FLDF,AX             ;Store sum in FLDF
            MOV         AX,4C00H                    ;End processing
            INT         21H
MAIN                    ENDP                ;End of procedure
CODESEG ENDS                                ;End of segment
END         MAIN                                    ;End of program
```

PROC Directive

- Format:

  Procedure-name PROC Operand Comment

  Procedure-name ENDP

  Operand: relates to program execution (FAR)

PROC directive

```
; Add two numbers and store the results into the third variable
page    60,132
TITLE       A04ASM1 (EXE)  Move and add operations
; -----------------------------------------------
STACK       SEGMENT PARA STACK 'Stack'
      DW        32 DUP(0)
STACK       ENDS
; -----------------------------------------------
DATASEG  SEGMENT PARA 'Data'
      FLDD                DW            215
      FLDE                DW            125
      FLDF                DW            ?
DATASEG  ENDS
; -----------------------------------------------
CODESEG SEGMENT PARA 'Code'
MAIN              PROC    FAR
                  ASSUME  SS:STACK,DS:DATASEG,CS:CODESEG
                  MOV     AX,DATASEG              ;Set address of data
                  MOV     DS,AX                   ;Segment in DS
                  MOV           AX,FLDD           ;Move 0215 to AX
                  MOV           FLDF,AX           ;Store sum in FLDF
                  MOV           AX,4C00H                    ;End processing
                  INT           21H
MAIN              ENDP            ;End of procedure
CODESEG ENDS                      ;End of segment
END             MAIN                              ;End of program
```

ASSUME directive

- Tells the assembler the purpose of each segment in the program

Example:

```
ASSUME  SS:STACK,DS:DATASEG,CS:CODESEG
```

END Directive

- ENDS end a segment, ENDP ends a procedure. An END directive ends an entire program.

Simplified Segment Directives

- The general format is

.MODEL memory-model

MODEL	NUMBER OF CODE SEGMENTS	NUMBER OF DATA SEGMENTS
TINY	*	*
SMALL	1	1
MEDIUM	MORE THAN 1	1
COMPACT	1	MORE THAN 1
LARGE	MORE THAN 1	MORE THAN 1

The general formats for the directives that define the stack, data, and code segments are:

.STACK [size]
.DATA
.CODE

Segment Initializing

- **Conventional Mode**

```
MOV     AX, DATASEG  ;Set address of data
MOV     DS,AX                ;Segment in DS
```

- **Simplified Mode**

```
MOV     AX, @data          ;Set address of data
MOV     DS,AX                ;Segment in DS
```

Data Definition

- A data item may contain an undefined value, or a constant, or a character string, or a numeric value.

[name] Dn expression

Name: identifier

Dn: Directives can be:

DB: byte	DF:farword
DW: word	DQ:quadword
DD: doubleword	DT:tenbytes

Expression:

can be unnitialized: ?

can be assigned a constant: such as 25, 21.

Example:

```
DATAZ   DB   21.
```

Data Definition

- It also allows for repeated duplications of the same value:

  syntax: [name] Dn repeat-count DUP(expression)..

  FLDD DW 10 DUP(?)

- **String** : “ “
- **Character :** ‘ ’
- **Numeric constants** can be in decimal, hexadecimal, binary, or real.


good
Luck
U
U