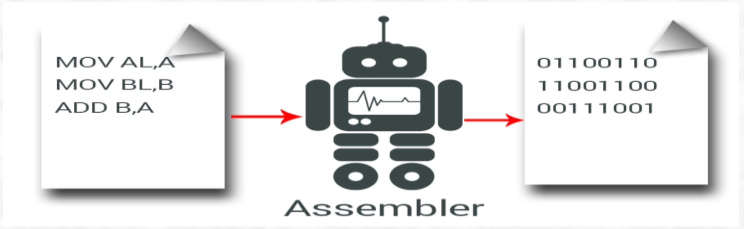


Computer Science Department
Faculty of Computers and Information
Mansoura University

Assembly Language

"Requirements for Coding in Assembly Language"

Sara El-Metwally, Ph.D.
Faculty of Computers and Information,
Mansoura University, Egypt.

Email: sarah_almetwally4@mans.edu.eg
sara.elmetwally.2007@gmail.com

Assembly Language Syntax (comments)

```
; calculate ratio
100 MOV AX, [11A]
```

; calculate ratio

No machine codes generated for comments.

Assembly Language Syntax (Reserved words)

- **Instructions:** such as `MOV`, `ADD`, etc.
- **Directives:** such as `END`, `SEGMENT`, used to provide information for an assembler.
- **Operators:** such as `FAR`, `SIZE` used in expressions.
- **Predefined symbols:** such as `@data` and `@model`, which return information to your program during the assembly.

Assembly Language Syntax (Identifiers)

- An identifier or symbol is a name that you apply to an item in your program that you expect to reference.
  - **Name:** refers to the address of a data item.

    ```
    COUNTER DB 0
    ```

  - **Label:** refers to the address of an instruction, procedure, or segment.

    ```
    MAIN PROC FAR
    B30: ADD BL,25
    ```

Assembly Language Syntax (Identifiers)

- An identifier can use the following chars:
  - Alphabetic letters (A-Z, a-z).
  - Digits (0-9, not the first char).
  - Special chars:
    - ?
    - _
    - $
    - @ (avoided)
    - . (not the first char)

Assembly Language Syntax (Statements)

- **Instruction statement**, starts with operation and the assembler translates it to the machine code.
- **Directive statement**, tells the assembler to perform a specific action, such as define a data item and it generates no machine code.

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

Assembly Language Syntax (Statements)

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

```
COUNTER DB  1 ; Define byte counter

 L30: MOV AX,0  ; Moving operation
      RET
      INC BX
      ADD CX, 25
```

Assembly Language Syntax (Directives)

- **Acts only during the assembly of a program and generates no machine-executable code.**

```
PAGE [length] [,width]
```

Number of lines per page

Number of chars per line

Default: PAGE 50,80

Assembly Language Syntax (Directives)

- **A title of a program to print on line 2 of each page of a program listing.**

```
TITLE text [;comment]
```

Assembly Language Syntax

(Directives)

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

- ❑ Boundary on which the segment is to begin; the typical requirement is PARA, which causes the segment to align on a paragraph boundary.
- ❑ Combine the segment with other segments when they are linked after assembly.
- ❑ PUBLIC, COMMON: combine separately assembled programs when linking them. Omit NONE.
- ❑ Group related segments when linking. Typical: 'code', 'stack', 'data'

Assembly Language Syntax (Directives)

```
CodeSeg SEGMENT PARA 'Code'
            ----
            ----
CodeSeg ENDS
```

Assembly Language Syntax (Directives)

```
Procedure-name PROC FAR
            ----
            ----
Procedure-name ENDP
```

Program loader Uses this procedure as the entry point for the first execution to execute.

- ❑ **Code segment could contain many procedures, each with its own** `PROC`, `ENDP`.
- ❑ `FAR` **will be changed to** `NEAR`.

Assembly Language Syntax (Directives)

```
ENDS
ENDP
END
```

End segment

End procedure

End program

```
END [procedure-name]
```

Blank: not an executed program i.e. data definitions, data linkage

Name of the procedure designated as FAR.

Assembly Language Syntax (Directives)

- **Tell assembler the purpose of each segment in the program.**

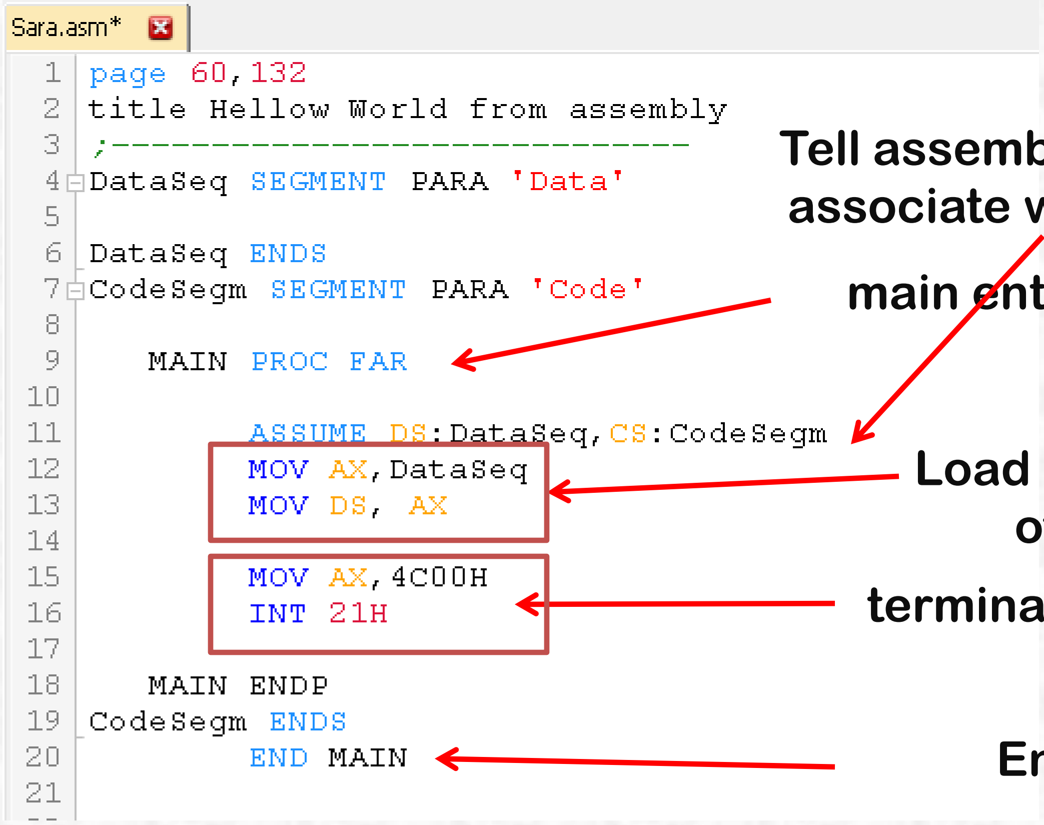
```
ASSUME SS: stackname, DS: datasegname,
CS: codesegname
```

Assembly Language Syntax (Directives)

- **Before the code segment or at the start of the program.**

.286	.386	.486
POPA	MOVSX/MOVZX	CMPXCHG
PUSHA	SHLD/SHRD	XADD

Assembly Language Syntax
(Put all together)



```
page 60,132
title Hellow World from assembly
;-------------------------------
DataSeq SEGMENT PARA 'Data'

DataSeq ENDS
CodeSegm SEGMENT PARA 'Code'

   MAIN PROC FAR

         ASSUME DS:DataSeq,CS:CodeSegm
         MOV AX,DataSeq
         MOV DS, AX

         MOV AX,4C00H
         INT 21H

   MAIN ENDP
CodeSegm ENDS
         END MAIN
```

Tell assembler which segments to associate with segment registers

main entry point

Load DS with the address of data segment.

terminate program execution

End program

Simplified Segment Directives

- Segment directives are shortcuts to define segments (SEGMENT& ENDS ✗).
- You have to initialize the memory model before defining any segment.
- It tells the assembler how to use segments, to provide enough space for the object code, and to ensure the optimum execution speed.

```
.MODEL memory-model
```

Simplified Segment Directives

```
.MODEL memory-model
```

MODEL	Number of code Segments	Number of data Segments
Small	1 <= 64K	1 <= 64K
Medium	Any number, any size	1 <= 64K
Compact	1 <= 64K	Any number, any size
Large	Any number, any size	Any number, any size
Huge	Any number, any size	Any number, any size

```
.MODEL Tiny
```

Simplified Segment Directives

- MODEL directive automatically generates the required ASSUME statement for all models.
- The new segment names are:

```
.STACK [size]
.DATA
.CODE [segment-name]
```

- Each of these directives causes the assembler to generate the required SEGMENT& ENDS statement.

Simplified Segment Directives

- The default segment names (which you do not have to define) are :

```
STACK
_DATA
_TEXT
```

- The default stack size is `1024` bytes and you can override it.
- You can override the name of code segment too.

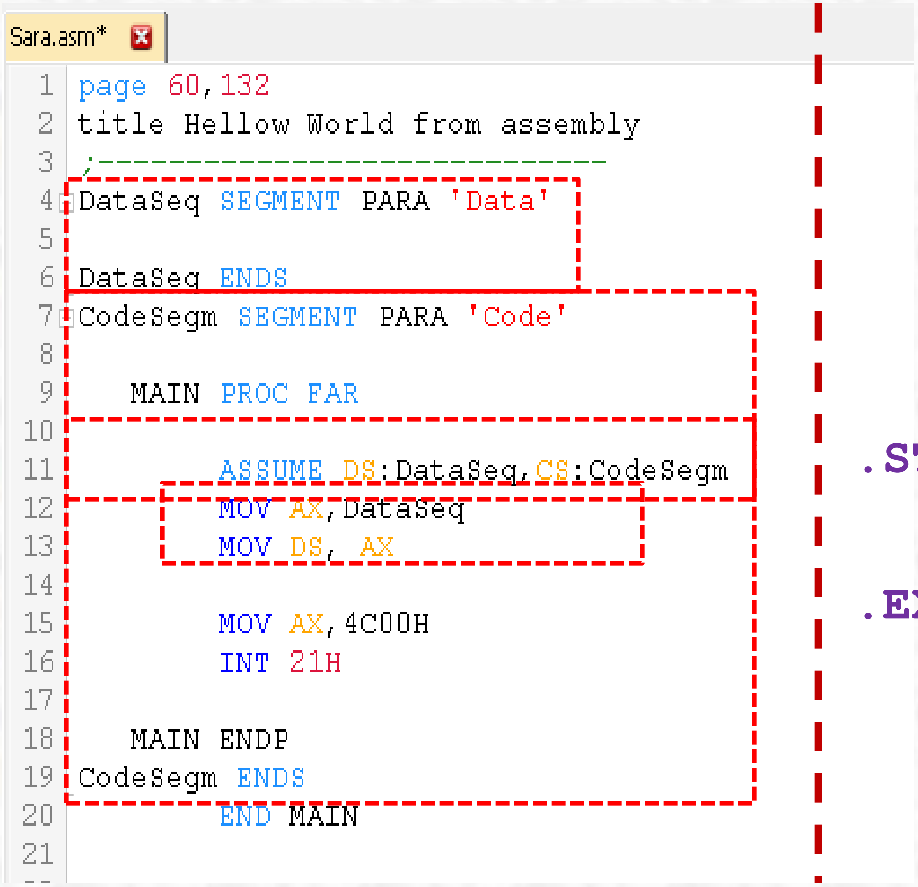
Simplified Segment Directives

- **The instructions used to initialize the address of the data segment in DS:**

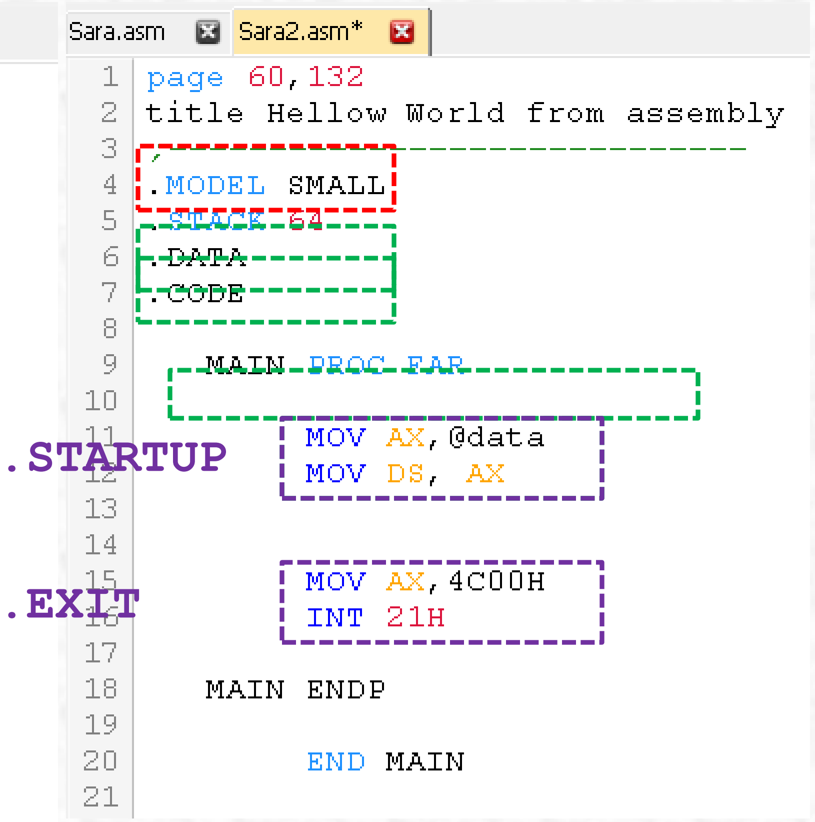
```
MOV AX, @data
MOV DS,AX
```

- **The default stack size is** 1024 **bytes and you can override it.**
- **You can override the name of code segment too.**

Simplified Segment Directives

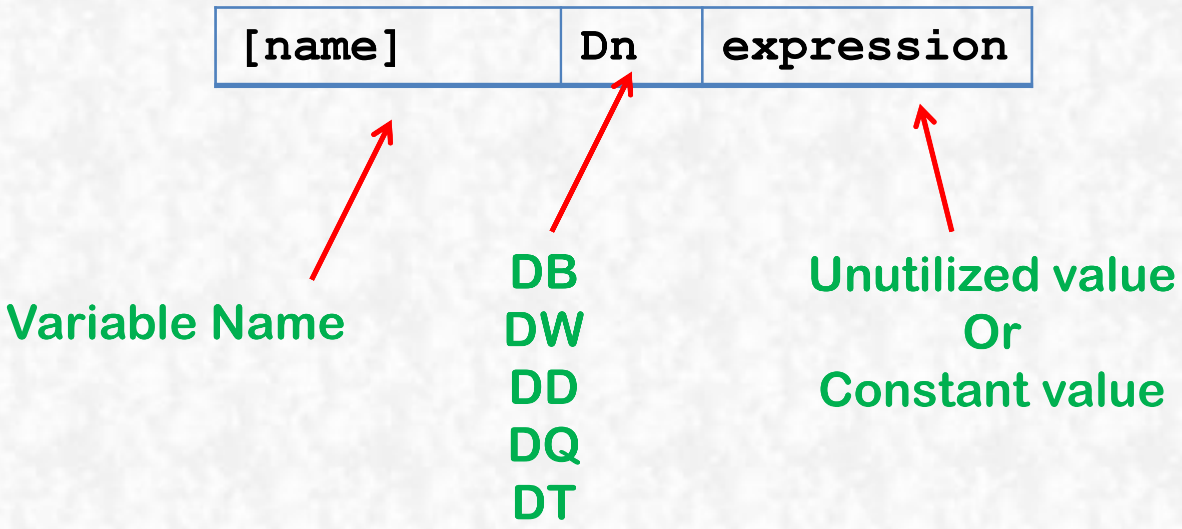


```
page 60,132
title Hellow World from assembly
;-------------------------------
DataSeg SEGMENT PARA 'Data'

DataSeg ENDS
CodeSegm SEGMENT PARA 'Code'

   MAIN PROC FAR

        ASSUME DS:DataSeg,CS:CodeSegm
        MOV AX,DataSeq
        MOV DS, AX

        MOV AX,4C00H
        INT 21H

   MAIN ENDP
CodeSegm ENDS
        END MAIN
```



```
page 60,132
title Hellow World from assembly
;---------------------------------------
.MODEL SMALL
.STACK 64
.DATA
.CODE

   MAIN PROC FAR


        MOV AX,@data
        MOV DS, AX


        MOV AX,4C00H
        INT 21H

   MAIN ENDP

        END MAIN
```

Defining Data Types



Defining Data Types

```
DATAX DB ?
DATAX DB 25
DATAX DB 21, 22, 23, 24, 25, 26
DW 10 DUP(?)
DB 5  DUP(12)        0C0C0C0C0C
```

Character Strings

```
DB ‘Computer Science‘
DB “Computer Science”
DB “ Sara’s Computer”
DB ‘ Sara”s Computer’
```

Numeric Constants

```
1B     ; Binary
12D    ; Decimal (default)
12H    ; Hexadecimal
   (first digit of a hex constant must be 0 to 9)
12.4R  ; Real value
```

EQUATE Directives

```
Val=5                  Val EQU 5
MOV AX, Val            MOV AX, Val

Val EQU 5
TABLE DB Val DUP(?)
```