

Assembler directives

- **Instructions to the Assembler regarding the program being executed.**
- **Control the generation of machine codes and organization of the program; but no machine codes are generated for assembler directives.**
- **Also called 'pseudo instructions'**
- **Used to :**
 - › **specify the start and end of a program**
 - › **attach value to variables**
 - › **allocate storage locations to input/ output data**
 - › **define start and end of segments, procedures, macros etc..**

DB

DW

SEGMENT
ENDS

ASSUME

ORG
END
EVEN
EQU

PROC
FAR
NEAR
ENDP

SHORT

MACRO
ENDM

■ Define Byte

■ Define a byte type (8-bit) variable

■ Reserves specific amount of memory locations to each variable

■ Range : $00_H - FF_H$ for unsigned value;
 $00_H - 7F_H$ for positive value and $80_H - FF_H$ for negative value

■ General form : **variable DB value/ values**

Example:

```
LIST DB 7FH, 42H, 35H
```

Three consecutive memory locations are reserved for the variable LIST and each data specified in the instruction are stored as initial value in the reserved memory location

DB

DW

SEGMENT
ENDS

ASSUME

ORG
END
EVEN
EQU

PROC
FAR
NEAR
ENDP

SHORT

MACRO
ENDM

■ Define Word

■ Define a word type (16-bit) variable

■ Reserves two consecutive memory locations to each variable

■ Range : $0000_H - FFFF_H$ for unsigned value;
 $0000_H - 7FFF_H$ for positive value and
 $8000_H - FFFF_H$ for negative value

■ General form : **variable DW value/ values**

Example:

ALIST DW 6512H, 0F251H, 0CDE2H

Six consecutive memory locations are reserved for the variable ALIST and each 16-bit data specified in the instruction is stored in two consecutive memory location.

DB

DW

SEGMENT
ENDS

ASSUME

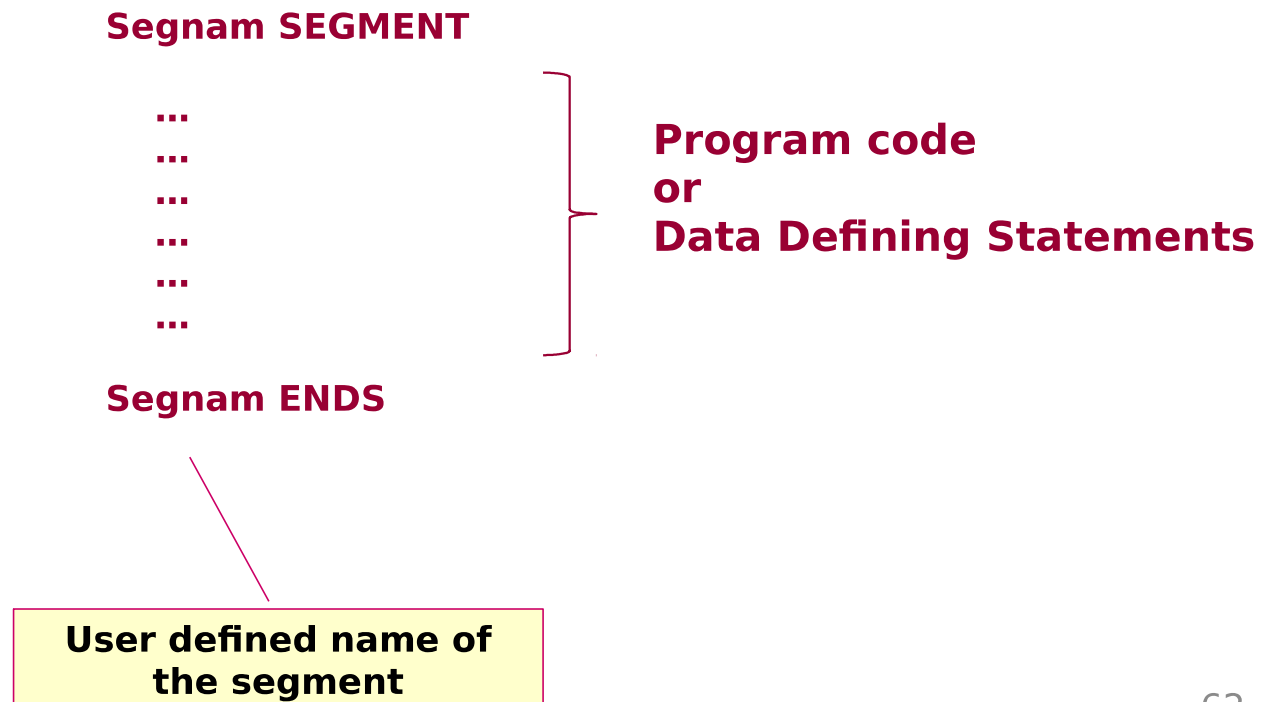
ORG
END
EVEN
EQU

PROC
FAR
NEAR
ENDP

SHORT

MACRO
ENDM

- **SEGMENT** : Used to indicate the beginning of a code/ data/ stack segment
- **ENDS** : Used to indicate the end of a code/ data/ stack segment
- **General form:**



Assemble Directives

DB

DW

SEGMENT
ENDS

ASSUME

ORG
END
EVEN
EQU

PROC
FAR
NEAR
ENDP

SHORT

MACRO
ENDM

- Informs the assembler the name of the program/ data segment that should be used for a specific segment.

- General form:

ASSUME segreg : segnam, .. , segreg : segnam

Segment Register

User defined name of
the segment

Example:

```
ASSUME CS: ACODE, DS:ADATA
```

Tells the compiler that the instructions of the program are stored in the segment ACODE and data are stored in the segment ADATA

DB

DW

**SEGMENT
ENDS**

ASSUME

**ORG
END
EVEN
EQU**

**PROC
FAR
NEAR
ENDP**

SHORT

**MACRO
ENDM**

- **ORG** (Origin) is used to assign the starting address (Effective address) for a program/ data segment
- **END** is used to terminate a program; statements after END will be ignored
- **EVEN** : Informs the assembler to store program/ data segment starting from an even address
- **EQU** (Equate) is used to attach a value to a variable

Examples:

ORG 1000H	Informs the assembler that the statements following ORG 1000H should be stored in memory starting with effective address 1000_H
LOOP EQU 10FEH	Value of variable LOOP is 10FE_H
_SDATA SEGMENT ORG 1200H A DB 4CH EVEN B DW 1052H _SDATA ENDS	In this data segment, effective address of memory location assigned to A will be 1200_H and that of B will be 1202_H and 1203_H .

DB

DW

SEGMENT
ENDS

ASSUME

ORG
END
EVEN
EQU

PROC
ENDP
FAR
NEAR

SHORT

MACRO
ENDM

■ **PROC** Indicates the beginning of a procedure

■ **ENDP** End of procedure

■ **FAR** Intersegment call

■ **NEAR** Intra-segment call

■ **General form**

procname PROC[NEAR/ FAR]

...
...
...

RET



Program statements of the procedure

Last statement of the procedure

procname ENDP

User defined name of the procedure

DB

DW

SEGMENT
ENDS

ASSUME

ORG
END
EVEN
EQU

PROC
ENDP
FAR
NEAR

SHORT

MACRO
ENDM

Examples:

```
ADD64 PROC NEAR
```

```
...  
...  
...
```

```
RET  
ADD64 ENDP
```

The subroutine/ procedure named **ADD64** is declared as **NEAR** and so the assembler will code the **CALL** and **RET** instructions involved in this procedure as near call and return

```
CONVERT PROC FAR
```

```
...  
...  
...
```

```
RET  
CONVERT ENDP
```

The subroutine/ procedure named **CONVERT** is declared as **FAR** and so the assembler will code the **CALL** and **RET** instructions involved in this procedure as far call and return

DB

- Reserves one memory location for 8-bit signed displacement in jump instructions

DW

**SEGMENT
ENDS**

Example:

ASSUME

**JMP SHORT
AHEAD**

The directive will reserve one memory location for 8-bit displacement named **AHEAD**

**ORG
END
EVEN
EQU**

**PROC
ENDP
FAR
NEAR**

SHORT

**MACRO
ENDM**

DB

DW

SEGMENT
ENDS

ASSUME

ORG
END
EVEN
EQU

PROC
ENDP
FAR
NEAR

SHORT

MACRO
ENDM

■ **MACRO** Indicate the beginning of a macro

■ **ENDM** End of a macro

■ **General form:**

macroname **MACRO**[Arg1, Arg2 ...]

...
...
...



**Program
statements in
the macro**

macroname **ENDM**

User defined name of
the macro