COURSE: UCS1502 - MICROPROCESSORS AND INTERFACING

Assembler directives

Dr. K. R. Sarath Chandran
Assistant Professor, Dept. of CSE

This presentation covers

• MASM Assembler directives

Learning Outcome of this module

• To understand various assembler directives of MASM assembler

Assembler directives

MASM (Microsoft Macro Assembler) program for adding two, 8 bit numbers

```
assume cs:code, ds:data
data segment
opr1 db 11h
opr2 db 99h
result db 00H
carry db 00H
data ends
code segment
          org 0100h
start:    mov ax,data
          mov ds,ax
          mov ah,opr1
          mov bh,opr2
          mov ch,00h
          add ah,bh
          jnc here
          inc ch
here:     mov result,ah
          mov carry,ch
          mov ah,4ch
          int 21h
          code ends
end start
```

Assembler directives

ASSUME: The ASSUME directive is used to tell the assembler that the name of the logical segment should be used for a specified segment.

Eg: ASSUME CS:CODE
This tells the assembler that the logical segment named CODE contains the instruction statements for the program and should be treated as a code segment.

DB – Define byte
It is used to declare a byte type variable or to store a byte in memory location.

Eg: string1 DB 11h, 22h, 33h
Declare an array of 3 bytes, named as string1 and initialize.

TEMP1 DB 10 DUP(?)
Set 10 bytes of storage in memory and give it the name as TEMP1, but leave the 10 bytes uninitialized. Program instructions will load values into these locations.

Assembler directives

DW - Define word

The DW directive is used to define a variable of type word or to reserve storage location of type word in memory.

Eg: value1 dw fffeh

DD - Define Double Word
DQ - Defined Quad Word
DT - Define Ten Bytes

END - END directive is placed after the last statement of a program to tell the assembler that this is the end of the program module. The assembler will ignore any statement after an END directive.

PROC
ENDP

Eg1:
SQUARE_ROOT **PROC**; It start the procedure
;Some steps to find the square root of a number
SQUARE_ROOT **ENDP**; End for the procedure

Eg2: SQRT PROC **FAR** ; means that SQRT is going to be called from a differently named code segment .

Assembler directives

Segment
Ends

Eg.
CODE **SEGMENT** ;Start the logic segment named code
;Some instructions
CODE **ENDS** ;End of segment named CODE

EQU
The EQU directive is used to give a name to some value or to a symbol. Each time the assembler finds the name in the program, it will replace the name with the value or symbol you given to that name.

Eg:
FACTOR1 EQU 05H ; this statement should be written at the starting portion of your program and later in the program you can use this as follows
ADD AL, FACTOR1 ; When it codes this instruction, the assembler will code it as ADD AL, 05H

Assembler directives

TYPE
Gives the number of bytes specified to that type of variable

```
String1 dw 1234h,6789h
---
---
mov ax, type string1;  0002 will be loaded into ax; (2 bytes in word type)
```

SHORT
Tells the assembler that only 1 byte displacement is needed to code the jmp instruction.

Eg: jmp **short** nearlabel1

LABEL
To give a name to the current location counter value

Eg:

```
Data segment
            string db 50 dup(0)
            datalast label byte
Data ends
;give name 'datalast' to the next location after allocating 50 bytes.
```

Assembler directives

PUBLIC

The PUBLIC directive is used to instruct the assembler that a specified name or label will be accessed from other modules.

Eg:
PUBLIC DIVISOR1, DIVIDEND1 ;these two variables are public so these are available to all modules.

EXTRN

If an instruction in a module refers to a variable in another assembly module, the assembler must be told that it is external with the EXTERN directive.

GROUP

- To group related segments

It is used to tell the assembler to group the logical segments named after the directive to one logical group segment

Eg:
smallsystem group code, data, stack
An appropriate ASSUME statement to follow this would be
ASSUME cs:smallsystem, ds:smallsystem, ss:smallsystem
(here all segments will be physically in same 64k segment)

References

- Doughlas V. Hall, “Microprocessors and Interfacing, Programming and Hardware”, Second Edition, TMH.

Thank you