UNIT-5

Contents

- Assembler directives
- Simple programs
- Procedures
- Macros
- **Assembly language programs involving:**
- Logical
- Branch
- CALL instructions
- Sorting
- Evaluation of Arithmetic expressions
- String Manipulation

Assembler Directives

Assume

Used to tell the assembler the names of the logical segments to be assumed in the program.

Example, ASSUME CS:CODE ,DS:DATA

DB – Defined Byte

Used to declare a byte type variable in memory.

Example, n1 DB 49h

DW – Define Word

Used to tell the assembler to define a word type variable in memory. ex: n1 DW 1234h

Assembler Directives

DD – Define Double Word

- Used to declare a variable of type double word or to reserve a memory location which can be accessed as double word.

DQ – Define Quad word

- Used to tell the assembler to declare the variable as 4 words of storage in memory.

DT – Define Ten bytes

- Used to tell the assembler to declare the variable which is 10 bytes in length or reserve 10 bytes of storage in memory.

Assembler Directives

END – End the program

- To tell the assembler to stop fetching the instruction and end the program execution.
- ENDP – it is used to end the procedure (subprogram/subroutine)
- ENDS – used to end the segment.

EQU – Equate

- Used to give name to some value or symbol.
- EX: ADDITION EQU ADD

EVEN – Align on Even memory address

- Tells the assembler to increment the location to the next even address if it is not already at an even address.

Assembler Directives

EXTRN: EXTERNAL AND PUBLIC

- Used to tell the assembler that the names ,procedures and labels after this directive have already been defined in some other assembly language module.
- EX: MODULE1 SEGMENT

```
PUBLIC FACTORIAL
MODULE1 ENDS
MODULE2 SEGMENT
EXTRN FACTORIAL
MODULE2 ENDS
```

GROUP – Group related segment

- Used to tell the assembler to group the logical segments named after the directive into one logical segment.
- This allows the content of all the segments to be accessed from the same group.
- EX: PROGRAM GROUP CODE,DATA,STACK

Assembler Directives

LABEL

- Used to give the name to the current value in the location counter.
- The LABEL directive must be followed by a term which specifies the type you want associated with that name.

LENGTH

- Used to determine the number of items in some data such as string or array.
- EX:MOV CX,LENGTH ARRAY

Assembler Directives

OFFSET

- It is an operator which tells the assembler to determine the offset or displacement of named data item or procedure from the start of the segment which contains it.
- EX: MOV SI,OFFSET LIST

ORG – Originate

- Tells the assembler to set the location value.
- Example, ORG 7000H sets the location counter value to point to 7000H location in memory.

Assembler Directives

PROC – Procedure

- Used to identify the start of the procedure.

PTR – Pointer (BYTE OR WORD)

- Used to assign a specific type to a variable or a label.
- EX:MOV AL, BYTE PTR [SI]

  MOV AX,WORD PTR [SI]

Assembler Directives

- **+ & - operators:**
- These operators represents arithmetic addition & subtraction.
- EX:MOV AX,[SI+2]

  MOV DX,[BX-3]
- **SEGMENT:** logical segment
- Ex:CODE SEGMENT

Assembler Directives

SHORT

- Used to tell the assembler that only a 1-byte displacement is needed to code a jump instruction.
- If the jump destination is after the jump instruction in the program, the assembler will automatically reserve 2 bytes for the displacement.

TYPE

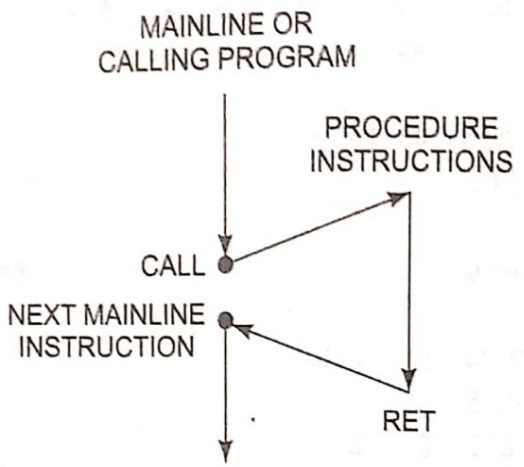
- Tells the assembler to determine the type of a specified variable.
- The TYPE operator can be used in instruction such as ADD BX, TYPE WORD_ARRAY, where we want to increment BX to point to the next word in an array of words.

Procedures

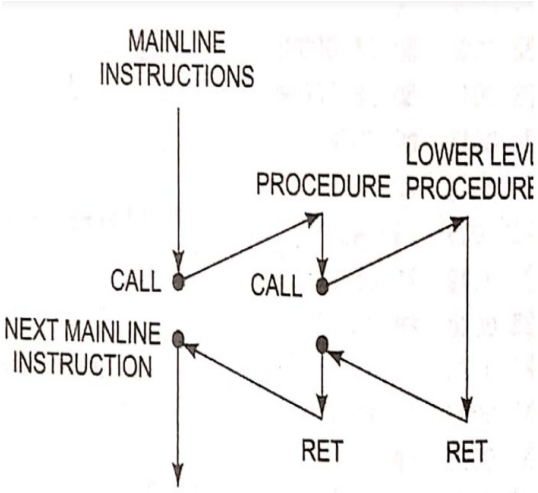
- The procedure is a group of repetitive instructions stored as a separate program in the memory and it is called from the main program whenever required.
- Type of procedure depends on where the procedure is stored in the memory.
- If it is in the same code segment where the main program is stored then It is called near procedure otherwise it is referred to as far procedure.
- This procedures are used by CALL and RET instructions.

Procedures

- The CALL instruction is used to transfer execution to procedure or subprogram. there are two types of CALLs, near & far.
- Near call is a call to a procedure which is in the same code segment as the call instruction.
- Far call is a call to a procedure which is in the different code segment from that which contains the call instruction.
- RET instruction will return the execution from a procedure to the next instruction after call instruction.



Single procedures



Nested procedures.

Macros

- Macro is a group of repetitive instructions.
- The macro assembler generates the code in the program each time where the macro is called, such that it takes more memory.
- Macros can be defined by MACRO and ENDM.

Differences b/w Procedures & Macros

Procedures	Macros
Accessed by CALL & RET instructions during program execution.	Accessed during assembly with name given to macro when defined.
Machine code for instructions is put only once in the memory.	Machine code is generated for instructions each time when macro is called.
With procedures less memory is required.	With macros more memory is required.

Programs

8 - bit Addition

```
Assume  cs : code,  ds :data
        data  segment
             num1  db  05H
             num2  db  02H
              res     db ?
        data  ends
    code segment
                start:    MOV  AX, data
                          MOV  DS ,AX
                          MOV  AX, 0000H
                          MOV   AL ,num1
                          MOV  BL,num2
                          ADD   AL,BL
                          MOV  res,AL
                          INT 3H
        code ends
                      end start
```

8 – bit Subtraction

```
Assume  cs : code,  ds :data
        data  segment
              num1  db  05H
              num2  db  02H
               res     db ?
        data  ends
      code segment
                start:    MOV  AX, data
                          MOV  DS ,AX
                          MOV  AX, 0000H
                          MOV   AL ,num1
                          MOV  BL,num2
                          SUB     AL,BL
                          MOV  res,AL
                          INT 3H
        code ends
                      end start
```

8 - bit Multiplication

```
Assume  cs : code,  ds :data

        data  segment
           num1  db  04H
           num2  db  02H
            res     dw ?
       data  ends
    code segment
            start:    MOV  AX, data
                      MOV  DS ,AX
                       MOV  AX, 0000H
                     MOV   AL ,num1
                     MOV  BL,num2
                      MUL   BL
                     MOV  res,AX
                     INT 3H
              code ends
                 end start
```

8 - bit Division

```
Assume  cs : code,  ds :data
        data  segment
             num1  db  05H
             num2  db  02H
              quo    db ?
              rem    db ?
        data  ends
    code segment
                    start:     MOV  AX, data
                               MOV  DS ,AX
                               MOV  AX, 0000H
                              MOV   AL ,num1
                              MOV  BL,num2
                              DIV    BL
                              MOV  quo ,AL
                               MOV  rem, AH
                              INT 3H
                         code ends
                           end start
```

16 - bit Addition

```
Assume  cs : code,  ds :data
        data  segment
             num1  dw  4215H
             num2  dw  3202H
              res     dw ?
        data  ends
      code segment
              start:    MOV  AX, data
                        MOV  DS ,AX
                        MOV  AX, 0000H
                        MOV   AX ,num1
                        MOV  BX,num2
                        ADD   AX,BX
                        MOV  res,AX
                        INT 3H
        code ends
                    end start
```

16– bit Subtraction

```
Assume  cs : code,  ds :data
        data  segment
              num1  dw  7505H
              num2  dw  4602H
               res      dw ?
        data  ends
      code segment
                start:    MOV  AX, data
                          MOV  DS ,AX
                          MOV  AX, 0000H
                          MOV   AX ,num1
                          MOV  BX,num2
                          SUB     AX,BX
                          MOV  res,AX
                          INT 3H
        code ends
                        end start
```

16 - bit Multiplication

```
Assume  cs : code,  ds :data
        data  segment
              num1  dw  2205H
              num2  dw  1102H
               res1   dw   ?
               res2   dw  ?
        data  ends
       code segment
              start:          MOV  AX, data
                               MOV  DS ,AX
                               MOV  AX, 0000H
                              MOV   AX ,num1
                              MOV  BX,num2
                               MUL   BX
                              MOV  res1,AX
                              MOV  res2, DX
                              INT 3H
                   code ends
                      end start
```

16 - bit Division

```
Assume  cs : code,  ds :data
        data  segment
             num1  dw  6666H
             num2  dw  3332H
              quo    dw ?
              rem    dw ?
        data  ends
    code segment
                    start:    MOV  AX, data
                              MOV  DS ,AX
                              MOV  AX, 0000H
                             MOV   AX ,num1
                             MOV  BX,num2
                             DIV    BX
                             MOV  quo ,AX
                              MOV  rem, DX
                             INT 3H
                         code ends
                           end start
```

Square of a number

```
Assume  cs : code,  ds :data
        data  segment
              num1  db  04H
              res     db ?
        data  ends
      code segment
              start:    MOV  AX, data
                        MOV  DS ,AX
                        MOV  AX, 0000H
                        MOV   AL ,num1
                        MUL  AL
                        MOV  res,AX
                        INT 3H
    code ends
                    end start
```

Cube of Number

```
Assume  cs : code,  ds :data
        data  segment
            num1  db  04H
            res     db ?
        data  ends
    code segment
                start:    MOV  AX, data
                          MOV  DS ,AX
                          MOV  AX, 0000H
                         MOV   AL ,num1
                          MOV BL,num1
                         MUL  AL
                         MUL  BL
                         MOV  res,AX
                         INT 3H
        code ends
                      end start
```

Exchange of a Number

```
Assume  cs : code,  ds :data
        data  segment
              num1  db  05H
              num2  db  02H
        data  ends
      code segment
              start:    MOV  AX, data
                        MOV  DS ,AX
                        MOV  AX, 0000H
                        MOV   AL ,num1
                        MOV  BL,num2
                       XCHG   AL,BL
                        INT 3H
        code ends
                    end start
```

32 - bit Addition

```
Assume  cs : code,  ds :data
        data  segment
             num1  dw  7777H
             num2  dw  6666H
              num3  dw 2222H
              num4  dw  1111H
               res1     dw ?
               res2     dw ?
        data  ends
    code segment    start:    MOV  AX, data
                              MOV  DS ,AX
                              MOV  AX, 0000H
                             MOV   AX ,num1
                             MOV  BX,num2
                            MOV  CX ,num3
                             MOV  DX ,num4
                             ADD  BX,DX
                             ADC   AX,CX
                             MOV  res1,BX
                            MOV  res2,AX
                             INT 3H
  code ends                          end start
```

32 - bit Subtraction

```
Assume  cs : code,  ds :data
        data  segment
             num1  dw  7777H
             num2  dw  6666H
              num3  dw 2222H
              num4  dw  1111H
               res1     dw ?
               res2     dw ?
        data  ends
    code segment    start:    MOV  AX, data
                              MOV  DS ,AX
                              MOV  AX, 0000H
                             MOV   AX ,num1
                             MOV  BX,num2
                            MOV  CX ,num3
                             MOV  DX ,num4
                             SUB  BX,DX
                             SBB   AX,CX
                             MOV  res1,BX
                            MOV  res2,AX
                             INT 3H
  code ends                          end start
```

Addition of Series of numbers

```
Assume cs : code, ds: data
data  segment
List    db  12h,34h,56h,78h,89h
RES    dw ?
data  ends
code segment
start:
        MOV AX, data
        MOV DS,AX
       MOV AX,0000h
       MOV BX, 0000h
       MOVCL,5
       MOV SI, offset  List
```

```
again:  MOV BL,[SI]
        ADD  AL,BL
        INC  SI
        DEC  CL
        JNZ      again
        MOV RES ,AX
        INT 3h
        code  ends
        end start
```

Average of Series of numbers

```
Assume cs : code, ds: data
data  segment
List    db  12h,34h,56h,78h,89h
RES    dw ?
data  ends
code segment
start:
        MOV AX, data
        MOV DS,AX
       MOV AX,0000h
       MOV BX, 0000h
       MOVCL,5
       MOV SI, offset  List
```

```
again:  MOV BL,[SI]
        ADD  AL,BL
        INC  SI
        DEC  CL
        JNZ      again
        MOV BL,05 H
        DIV  BL
       MOV RES ,AX
        INT 3h
        code  ends
        end start
```

Sum of squares of given series of numbers

```
assume cs: code, ds: data
data segment
List    db  01h,02h,03h,04h,05h
SUM  dw  ?
data ends
code segment
start:
MOV AX,data
MOV DS,AX
```

```
        MOV SI, offset List
        MOV Cl ,  05H
l1:     MOV AX,0000H
        MOV AL,[SI]
        MUL AL
        ADD BX,AX
        INC SI
        loop l1
        MOV  SUM ,BX
        INT 3H
       code ends
       end start
```

Sum of CUBES of given series of numbers

```
assume cs: code, ds: data
data segment
List    db  01h,02h,03h,04h,05h
SUM  dw  ?
data ends
code segment
start:
MOV AX,data
MOV DS,AX
```

```
        MOV SI, offset List
        MOV Cl ,  05H
l1:     MOV AX,0000H
        MOV AL,[SI]
        MOV DL,[SI]
        MUL AL
        MUL DL
        ADD BX,AX
        INC SI
        loop l1
        MOV SUM,BX
        INT 3H
code ends
end start
```

Factorial of a given number

```
        assume  cs: code, ds: data
  data segment
    n1    db  04H
   Fact   dw   ?
data ends
  code segment
 start:
  MOV AX,data
  MOVDS,AX
  MOV AL,0000H
```

```
    MOV AL,n1
    MOV CL,03H
L1:
      MUL CL
      DEC CL
      JNZ  L1
      MOV fact,AX
      INT 3H
     Code ends
     end start
```

Greatest number in list

```
assume cs: code ,ds: data
data segment
list    db 10h,20h,30h,40h,50h
res     db ?
data ends
code segment
start:
        MOV AX, data
        MOV DS,AX
```

```
        XOR AX,AX
        MOV Cl,04H
        MOV SI, offset  list
        MOV AL,[SI]
again:  CMP AL,[SI+1]
        JNC     next
        MOV AL, [SI+1]
next:   INC SI
        DEC CL
        JNZ again
        MOV res,AL
        INT 3h
code ends
end start
```

Smallest number in list

```
assume cs: code ,ds: data
data segment
list    db 10h,20h,30h,40h,50h
res     db ?
data ends
code segment
start:
        MOV AX, data
        MOV DS,AX
```

```
        XOR AX,AX
        MOV Cl,04H
        MOV SI, offset  list
        MOV AL,[SI]
again:  CMP AL,[SI+1]
        JC      next
        MOV AL, [SI+1]
next:   INC SI
        DEC CL
        JNZ again
        MOV res,AL
        INT 3h
code ends
end start
```

Unit-5 (Question bank)

- Explain about Procedures and Macro
- Explain about Assembler directives
- Write an ALP (assembly language programme)
- programs