BITS Pilani

Microprocessors & Interfacing

Programming Model

Dr. Gargi Prabhu
Department of CS & IS

MASM

Creating and running an executable file involves four steps:

1. Assembling the source code into an object file

masm filename.asm

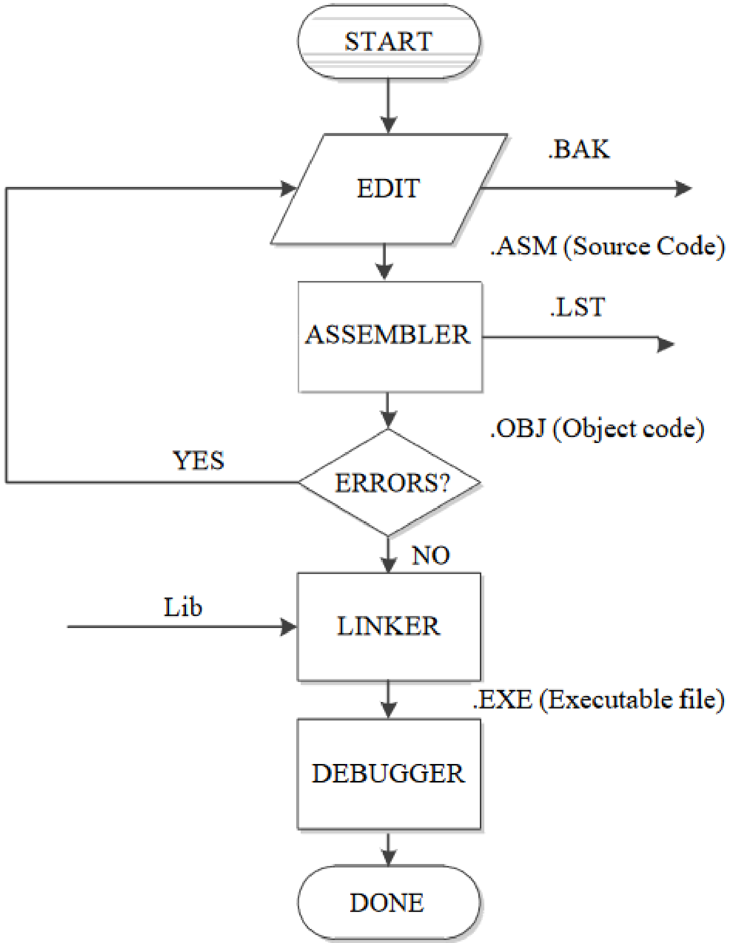
2. Linking the object file with other modules or libraries into an executable program

link filename.obj

3. Loading the program into memory and running the program

filename.exe

Creating source code



Writing assembly programs.

1. Set up and declare the data structure for the algorithm you are working with.

2. Write down the instructions required for initialization at the start of the code section.

3. Determine the instructions required to implement the major actions taken in the algorithm, and decide how dada must be positioned for these instructions.

4. Insert the instructions required to get the data in correct position.

Program Format

Line 1 MODEL SMALL ; Select small model

Line 3 .data ; Indicates data segment.

……
……
……

Data declaration

Line k .code ; indicates start of code segment

…….
……..
……..

Program body

Line n End ; End of file

Model

Memory Model	Size of Code	Size of Data	Details
TINY	Code + Data < 64KB	Code + data < 64KB	All data and code in one segment
SMALL	Less than 64KB	Less than 64KB	one data segment and one code segment
MEDIUM	Can be more than 64KB	Less than 64 KB	One data segment and two or more code segments
COMPACT	Less than 64KB	Can be more than 64KB	One code segment and two or more data segments
LARGE	Can be more than 64K	Can be more than 64KB	Any number of data and code segments
HUGE	Can be more than 64K	Can be more than 64KB	An array may have a size greater than 64 KB , allows for far pointers that can address more than 1MB of memory.

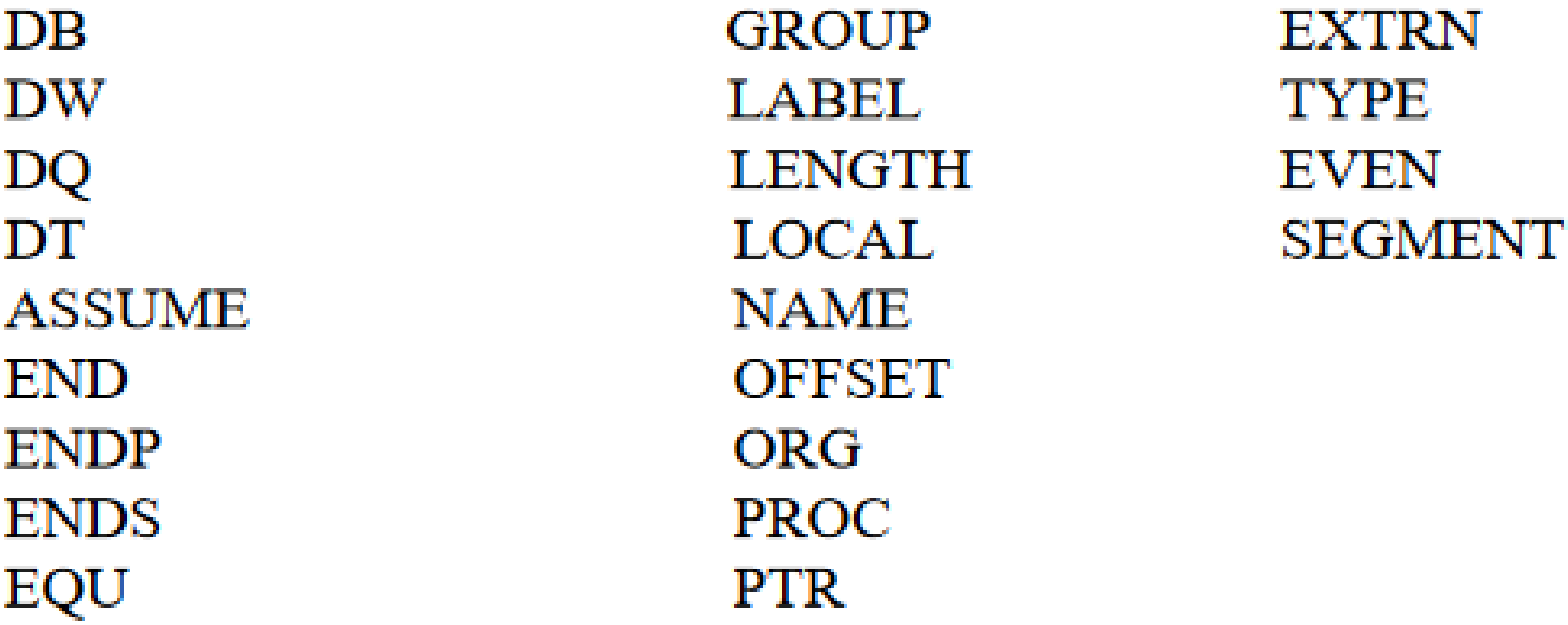
Simple MASM Code

```
.model small
.stack 100h

.data
    ; Data section (empty in this example)

.code
main proc
    ; Code section (empty in this example)

    ; Terminate the program
    mov ah, 4Ch   ; AH = 4Ch indicates exit function
    int 21h       ; Call DOS interrupt to terminate program
main endp

end main
```

Hello World Code

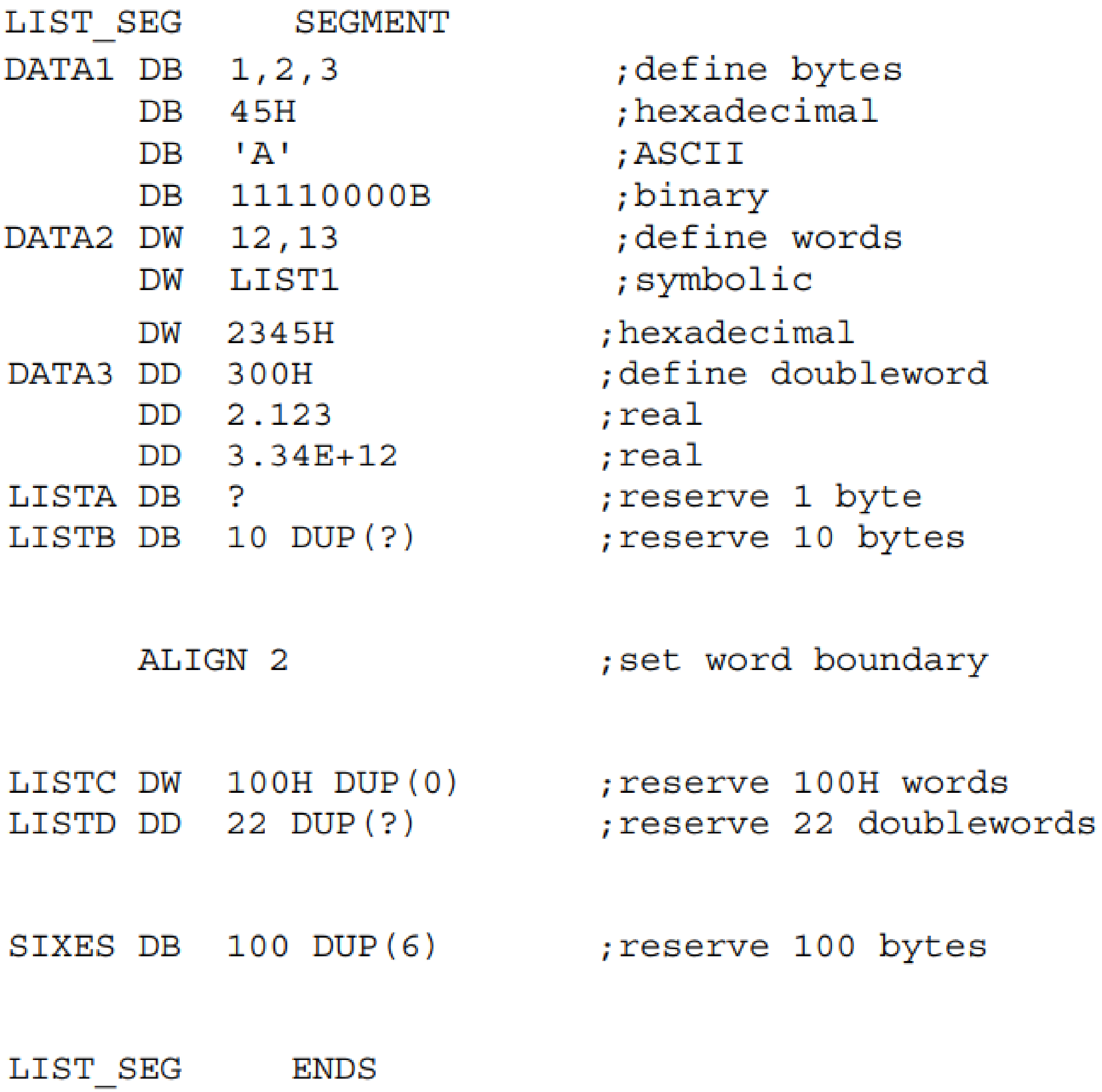
```
.model small
.stack 100h

.data
   hello_msg   db 'Hello, World!', 0

.code
main proc
   mov ax, @data   ; Load data segment address into AX
   mov ds, ax      ; Set DS to point to data segment
   ; Print "Hello, World!" message
   mov ah, 09h     ; AH = 09h indicates DOS function: print string
   lea dx, hello_msg  ; Load address of hello_msg into DX
   int 21h         ; Call DOS interrupt to print string
   ; Terminate the program
   mov ah, 4Ch     ; AH = 4Ch indicates exit function
   int 21h         ; Call DOS interrupt to terminate program
main endp

end main
```

Assembler Directives



DB
DW
DQ
DT
ASSUME
END
ENDP
ENDS
EQU
GROUP
LABEL
LENGTH
LOCAL
NAME
OFFSET
ORG
PROC
PTR
EXTRN
TYPE
EVEN
SEGMENT

Storing Data in a Memory Segment



```
LIST_SEG      SEGMENT
DATA1  DB   1,2,3                 ;define bytes
       DB   45H                   ;hexadecimal
       DB   'A'                   ;ASCII
       DB   11110000B             ;binary
DATA2  DW   12,13                 ;define words
       DW   LIST1                 ;symbolic
       DW   2345H                ;hexadecimal
DATA3  DD   300H                 ;define doubleword
       DD   2.123                ;real
       DD   3.34E+12             ;real
LISTA  DB   ?                    ;reserve 1 byte
LISTB  DB   10 DUP(?)            ;reserve 10 bytes


       ALIGN 2                   ;set word boundary


LISTC  DW   100H DUP(0)          ;reserve 100H words
LISTD  DD   22 DUP(?)            ;reserve 22 doublewords


SIXES  DB   100 DUP(6)           ;reserve 100 bytes


LIST_SEG      ENDS
```

Example 1

Implement assembly language program for addition of two 16-bit numbers.

Example 2

To implement ALP to find sum of numbers in the array.

ALGORITHM:

1. Start.
2. Initialize counter = 10.
3. Initialize array pointer.
4. Sum = 0.
5. Get the array element pointed by array pointer.
6. Add array element in the Sum.
7. Increment array pointer decrement counter.
8. Repeat steps 4, 5 & 6 until counter = 0.
9. Display Sum.
10. Stop.

Example 3

To implement ALP to find number of ONE's in a given 8-bit number.

1. Start.
2. Initialize the data segment.
3. Clear the base register.
4. Initialize the counter.
5. Rotate the number, check for '1'.
6. Result is displayed.
7. Stop

Exercise

Write a program to take two numbers as input from the user and print its addition and subtraction results.

References

- The Art of Assembly Language - https://www.ic.unicamp.br/~pannain/mc404/aulas/pdfs/Art%20Of%20Intel%20x86%20Assembly.pdf
- Programmer's Guide -

https://www.mikrocontroller.net/attachment/450367/MASM61PROGUIDE.pdf

innovate achieve lead

BITS Pilani

Pilani Campus

Thank You