CSC405 Microprocessor

RITHESH KINI

COMPUTER ENGINEERING DEPT.

TSEC

8086 Microprocessor

Assembly Language Programming

rithesh.kini@thadomal.org

8086 - Assembler Directives

Assembly language consists of 2 type of statements:

i. **Executable Statements**

ii. **Assembler Directives**

8086 - Assembler Directives

i. **Executable Statements**

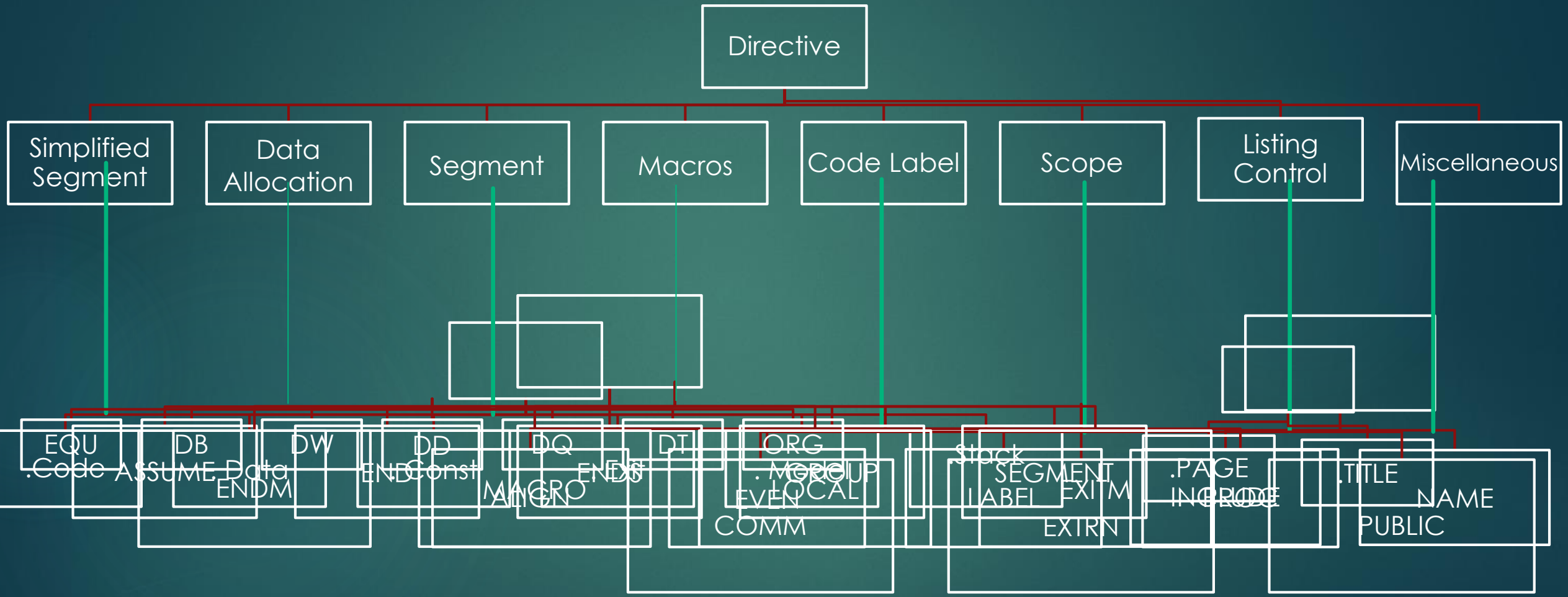
- These are statements to be executed by the processor.
- It consist of the entire instruction set of 8086
- Instructions that are translated into machine code by assembler

8086 - Assembler Directives

ii. Assembler Directives

- The statements that direct the assembler to do some special task
- They are effective only during the assembly of program.
- They do not generate code or No M/C language code is produced for these statements.
- Their main task is to inform the assembler about the start/endof a segment, procedure or program, to reserve appropriate space for data storage etc.

8086


Directive
Simplified Segment
Data Allocation
Segment
Macros
Code Label
Scope
Listing Control
Miscellaneous
EQU
.Code
ASSUME
DB
Data
ENDM
DW
DD
ENDS
Const
DQ
MACRO
ENDP
DT
ALIGN
ORG
.MODEL
GROUP
LOCAL
EVEN
COMM
Stack
SEGMENT
LABEL
EXITM
EXTRN
.PAGE
INCLUDE
.TITLE
NAME
PUBLIC

8086 - Assembler Directives

1	.CODE	Indicates the beginning of the Code seg .CODE [NAME]
2	.DATA	Indicates the beginning of the Data seg
3	.MODEL	Used for selecting a standard memory model: small, medium, compact, large, huge .MODEL [memory model]
4	.STACK	Used for defining the stack .STACK [size]
5	EQU	Used to give name to some value or symbol in the program

8086 - Assembler Directives

6	DB	Defines a byte type variable: SUM DB 11
7	DW	Defines a word type variable (2 byte)
8	DD	Defines a double word type variable (4 byte)
9	DQ	Defines a quad word type variable (8 byte)
10	DT	Defines a 10 bytes to a variable (10 byte)

8086 - Assembler Directives

11	ORG	Allows us to set the location counter to any desired value at any point in the program
12	DUP	Copies the contents of the bracket followed by this keyword into the memory location specified before it LIST DB 10 DUP(0): stores LIST as a series of 10 bytes initialized to 0

8086 - Assembler Directives

13	ASSUME	Used for telling the assembler the name of the logical segment which should be used ASSUME CS: Code, DS: Data, SS: Stack
14	END	Placed at the end of a source. Acts as a last statement. Terminates the entire program
15	SEGMENT	Used to indicate the start of a logical segment
16	ENDS	Indicates the end of a segment

8086 - Assembler Directives

17	PROC	Used to indicate the start of a procedure
18	ENDP	Indicates the end of a procedure
19	LABEL	Assigns name to the current value of the location counter

8086

Data_Here	SEGMENT	Data_Description
	LIST DB 10 DUP(0)	Stores LIST as a seriesof 10 Bytes initialized to zero
	--------	
Data_Here	ENDS	
Code_Here	SEGMENT	Body of theprogram
	ASSUME CS: Code_Here, DS: Data_Here	Makes Code_Here as code segment & Data_Here as data segment
	----	

Code_Here	ENDS	
	END	

8086 - DOS CALLS

Function no.

DOS INTERRUPT

mov ah, **Function No**

int 21h

8086 - DOS CALLS

1	mov ah,01h int 21h	➢ Input a character from the screen. ➢ Takes the user input character from the screen and returns the ascii value of character in AL register
2	mov ah,02h int 21h	To display a character on the screen, DL should contain
3	mov dx, offset msg mov ah,09h int 21h	To display a string on the screen, it displays the string whose offset address is in DX

8086

4	mov ah,4ch int 21h	Terminate the program

8086

```
.model small
.data
        num1  db 23h
        num2  db 12h
.code
        mov ax,data
        mov ds,ax

        mov al, num1
        mov bl, num2
        add al,bl

        mov ah,4ch
        int 21h
end
```

```
data segment
num1  db 23h
num2  db 12h
data ends

code segment
assume cs: code, ds: data
start:      mov ax,data
            mov ds,ax

            mov al, num1
        mov bl, num2
        add al,bl

mov ah,4ch
int 21h

code ends
end start
```

8086

```
data segment
    num1  db 23h
    num2  db 12h
    msg1 db 0dh,0ah,"the result of the addition is $"
data ends

code segment
assume cs: code, ds: data
start:   mov ax,data
         mov ds,ax

     mov dx, offset msg;
     mov ah,09h; to display a string on the screen, it displays
      int 21h;     the string whose offset address is in DX
```

8086

```
mov bl,num1
add bl,num2

mov cl,bl

mov bl,cl
and bl,F0h
ror bl,04h

call convert ; convert decimal into ASCI

mov dl,bl
mov ah,02h; to display a character on the screen, DL should
int 21h;   contain the offset address of the out put screen
```

8086

```
mov bl,cl
and bl,0Fh
call convert

mov dl,bl
mov ah,02h
int 21h

mov ah,4ch; Terminate the program
int 21h
```

8086

```
convert proc
    cmp bl,0Ah
    jc l1
    add bl,37h
    jmp l2
        l1: add bl, 30h
        l2: ret
endp
code ends
end start
```

Thank You

rithesh.kini@thadomal.org