Delay Routine (Delay Loop)

Rajeshwari. B.S
Assistant Professor
CSE Dept, BMSCE

1) What is the delay produced by the following delay program, if the frequency of MP is 5MHz? 1

```
    MOV  BL, OFFH
BACK: DEC  BL
      JNZ  BACK
      RET
```

FF ⇒ 255 in decimal

Instructions	T states / instruction	Total T states
MOV BL, OFF H	4	4
Dec BL	2	(16+2) = 18 * 255 = 4590
JNZ BACK	16	
RET	8	8

Total = 4602 T states

$$\text{Delay} = \frac{\text{Total T states}}{\text{Frequency}} = \frac{4602}{5\ \text{MHz}} = \frac{4602}{5 * 10^6} = 0.0009204$$

$$= 0.9204 * 10^{-3}$$

$$= 0.9204\ \text{ms}$$

In the above delay program, 8 bit register BL used, so maximum value that can be stored is FF. ∴ maximum delay that can be obtained using 8 bit register is 0.9204 ms.

2) write a ALP to generate a delay of 0.5ms for 8086 µP that runs at 5MHz frequency

Soln

Assume x is the value to be loaded to an 8 bit register, then the delay routine is

	T States/instn	Total T States
DELAY PROC		
MOV BL, x	4	4
BACK: DEC BL	2	(16+2)*x
JNZ BACK	16	
RET	8	8
DELAY ENDP		12 + 18x T States

$$\text{delay} = \frac{\text{Total T States}}{\text{Frequency}}$$

$$0.5\text{ms} = \frac{12+18x}{5\text{MHz}}$$

$$0.5\text{ms} = \frac{12+18x}{5*10^6}$$

$$0.5*10^{-3} = \frac{12+18x}{5*10^6}$$

$$12+18x = (0.5*10^{-3})(5*10^6)$$

$$x = \frac{(0.5*10^{-3})(5*10^6) - 12}{18}$$

$$= \frac{(0.0005 * 5000000) - 12}{18}$$

$$= \frac{2488}{18} = 138 = 8AH$$

16 | 138
8—A

$x = 8AH$

3) Write an ALP to generate a delay of 100ms for 8086 µP that runs at 5MHz frequency

0.5 ms = 138
1 ms = 138*2 = 276
10 ms = 2760
100 ms = 27600

Soln

As we have seen in Problem 1 that, using 8 bit register we can Generate a delay of maximum 0.9204 ms. So to generate a delay higher than this requires 16 bit register.

∴ Assume x is the 16 bit value to be loaded to 16 bit register, the delay routine is as follows

Instructions	T states/instn	Total T states
MOV BX, x	4	4
BACK: DEC BX	2	(16+2)*x
JNZ BACK	16	
RET	8	8
		12 + 18x T states

$$\text{Delay} = \frac{\text{Total T states}}{\text{Frequency}}$$

$$100\text{ ms} = \frac{12+18x}{5\times10^6}$$

$$100\times10^{-3} = \frac{12+18x}{5\times10^6}$$

$$\therefore (100\times10^{-3})(5\times10^6) = 12+18x$$

$$\therefore x = \frac{(100\times10^{-3})(5\times10^6)-12}{18} =$$

$$x = \frac{(0.1) * (5000000) - 12}{18} = 27,777$$

$x = 27,777$ or $6C81H$

16 | 27,777
16 | 1736
16 | 108
6 -

∴ delay routine to generate 100ms in 8086 that runs on 5MHz frequency is

```
Delay PROC
      MOV BX, 6C81H
BACK: DEC BX
      JNZ BACK
      RET
Delay ENDP
```

4) Write a program to generate a delay of 0.2 sec in 8086 µP that runs at 5MHz frequency

MOV	BX, x	4	4
BACK: Dec	BX	2	(16+2)*x
JNZ	BACK	16	
RET		8	8

Total = 12 + 18x T States

delay = Total T states / Frequency = ~~12+18x~~

$$0.2 \text{ sec} = \frac{12 + 18x}{5 \text{ MHz}}$$

$$0.2 \text{ sec} = \frac{12 + 18x}{5 \times 10^6}$$

$$x = \frac{(0.2) * (5 * 10^6) - 12}{18}$$

$x = 55,555$ or D903H

16 | 55555	
16 | 3472	– 3
16 | 217	– 0
13	– 9

∴ delay routine

```
Delay PROC
        MOV BX, D903H
BACK:   DEC BX
        JNZ BACK
        RET
Delay ENDP
```

5) What is the maximum delay produced if BX=FFFFH running on 8086 MP operating at 5MHz frequency

```
MOV BX, FFFFH    4
BACK: DEC BX     2   }
      JNZ BACK   16  }   18 * FFFF = 18 * 65535
      RET        8        8          8
```

1179642 T States

$$\text{Delay} = \frac{\text{Total T States}}{\text{Frequency}} = \frac{1179642}{5 \times 10^6} = \frac{1179642}{5000000} = 0.2359 \text{ sec}$$

Delay = 0.2359 sec

Or

Delay = 235.9×10^{-3}

Delay = 235.9 ms

∴ The maximum delay that can be obtained with 16 bit register is 0.2 sec

6) Write a program to generate a delay of 1 s

Soln:- Maximum delay obtained by 16 bit register is 0.2 sec, ∴ to generate delay of 1 sec, we require nested LOOPs.

```
        MOV   AX, X        4          4
AGAIN:  MOV   BX, FFFF     4
BACK:   DEC   BX           2
        JNZ   BACK         16   } (2+16)*FFFF X
        DEC   AX           2
        JNZ   AGAIN        16
        RET                8          8
```

$(4 + (18 * FFFF) + 12 +$

delay = $\frac{4 + (22 + 18 \times 65,535) X + 8}{5 \times 10^6}$

$$= 1 \text{ Sec} = \frac{4 + 22x + 1179630x + 8}{5000000}$$

$$5000000 = 12 + 1179652x + 8$$

$$5000000 = 20 + 1179652x$$

$$x = \frac{5000000 - 20}{1179652} =$$

$$x = 4.238 \cong 5$$

∴ Delay routine to Generate 1 Sec is

```
Delay  PROC
MOV    AX,5
Again: MOV    BX,FFFF
BACK:  DEC    BX
       JNZ    BACK
       DEC    AX
       JNZ    AGAIN
       RET
Delay  ENDP
```

7) Write a delay routine to Generate a delay of 10 minutes.

Soln 10 minutes = 10 X 60 Sec = 600 Sec

∴ maximum delay with 16 bit loaded to 16 bit register is 0.2 Sec.

∴ requires nested loop

	MOV	AX, x	4	4
AGAIN:	MOV	BX, FFFF	4	4
BACK:	DEC	BX	2	18 * 65535
	JNZ	BACK	16	2
	DEC	AX	2	16
	JNZ	AGAIN	16	8
	RET		8	

(* FFFF, * x)

$\therefore$ Total T states $= 4 + (22 + 18 * 65535)x + 8$

$= 4 + 1179652x + 8$

$=$

$$\text{delay} = \frac{\text{Total T states}}{\text{Frequency}}$$

$$600\,\text{sec} = \frac{12 + 1179652x}{5000000}$$

$$x = \frac{(600)(5000000) - 12}{1179652} = 2,543$$

$x = 2543$ or 09EFH

16 | 2543
16 | 158 - F
9 - E

$\therefore$ Delay routine to generate 600 sec or 10 minutes is

```
Delay proc
        MOV  AX, 9EFH
AGAIN:  MOV  BX, FFFFH
```

```
BACK: DEC  BX
      JNZ  BACK
      DEC  AX
      JNZ  AGAIN
      RET
Delay  Endp
```

8) Write a delay routine to calculate a delay of 4 sec for 8086 MP that operates at 10MHz frequency

Soln ∴ 4 sec means requires nested loop with FFFF, maximum delay obtained is 0.2 sec

```
      MOV  AX, x        4
AGAIN: MOV BX, FFFF     4
BACK: DEC  BX           2 ]
      JNZ  BACK         16 ] * FFFF
      DEC  AX           2
      JNZ  AGAIN        16
      RET               8
```

4 + 18 * FFFF + 2 + 16 + 8 = 18 * FFFF + 30

4 + (18 * FFFF + 30) x + 8

∴ Total T states = 4 + (22 + (18 * 65535)) x + 8

delay = Total T states / frequency

4 sec = (22 + 18 * 65535) x / 10MHz

x = (4 * 10 x 10^6 − 12) / 1179652 = (4 x 10000000 − 12) / 1179652 = 33.9 ≈ 34 = 22H

16 | 34
2 − 2

∴ delay routine to Generate 4 sec delay

```
Delay  PROC
       MOV  AX, 22H
AGAIN: MOV  BX, FFFF
BACK:  DEC  BX
       JNZ  BACK
       DEC  AX
       JNZ  AGAIN
       RET
Delay  ENDP
```

---

q) Write a delay procedure to generate a delay of 1 sec for 8086 MP working at 10MHz

Soln

1 Sec ⟹ Nested loop

```
       MOV  AX, X      4
AGAIN: MOV  BX, FFFF   4
BACK:  DEC  BX         2
       JNZ  BACK       16   ] * FFFF
       DEC  AX         2
       JNZ  AGAIN      16
       RET             8
```

⟹

```
4
4
4
18 * FFFF
2
16
8
```

X

∴ Total T States = $4 + (22 + 18 * 65535)X + 8$

$= 12 + 1179652X$

$$\text{delay} = \frac{\text{Total T States}}{\text{Frequency}}$$

$$1 \text{ Sec} = \frac{12 + 1179652x}{10 * 10^6}$$

$$x = \frac{10000000 - 12}{1179652} = 8.4 \simeq 8.5$$

∴ Delay routine to generate 1 Sec

```
Delay  PROC
       MOV   AX, 9
AGAIN: MOV   BX, FFFF
BACK:  DEC   BX
       JNZ   BACK
       DEC   AX
       JNZ   AGAIN
       RET
DELAY  ENDP
```

Instruction Template

Rajeshwari. B.S
Assistant Professor
CSE Dept, BMSCE
Bangalore

A 8086 assembler converts an 8086 Assembly language program into machine codes. Consider the instruction MOV CX, Source. The Source can be any of the 8 16-bit registers (AX, BX, CX, DX, SP, BP, SI & DI) or any of the memory location specified by any of 24 addressing modes ([BX], [SI], [DI], [BX]+[SI], [BX]+[DI], [BP]+[SI], [BP]+[DI], [BX]+[SI]+d_8, [BX]+[DI]+d_8, etc). Therefore there are 32 ways to specify source operand in an instruction with CX as destination. Similarly there are 32 ways to specify destination operand in an instruction with CX as source. Therefore there are 64 different codes for mov instruction with CX register as a source or destination.

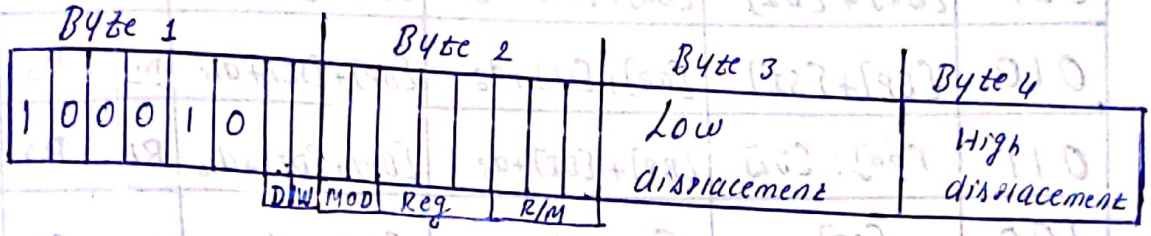
64 different codes with CL register as a source or destination

64 different codes with CH register as a source or destination.

Because of large number of possible codes for an 8086 instructions, It is impossible to take it from the table, instead assembler uses a template for each basic instruction type & fill in the bits to generate

machine codes. The instruction template for an 8086 processor is as shown below

1. Case I : Instruction template for an instruction, which moves data from register to register, register to memory & memory to register.



* The upper 6 bits are opcode.

* D bit in the first byte indicates direction, whether the data is being moved to the register or data is being moved from the register. If the data is being moved to the register, then D = 1. If the data is being moved from the register, then D = 0

* W bit in the first byte indicates whether word data or byte data

for byte movement, W = 0
for word movement, W = 1

* 3, 4, & 5th bits of the second byte specifies the register. & Mod bit & 3 R/M bits of the second byte specifies the addressing modes of the other operand.

The following table shows MOD & R/M for each of the possible 32 addressing modes.

R/M \ MOD	00	01	10	11	
000	[BX]+[SI]	[BX]+[SI]+d_8	[BX]+[SI]+d_{16}	AL	AX
001	[BX]+[DI]	[BX]+[DI]+d_8	[BX]+[DI]+d_{16}	CL	CX
010	[BP]+[SI]	[BP]+[SI]+d_8	[BP]+[SI]+d_{16}	DL	DX
011	[BP]+[DI]	[BP]+[DI]+d_8	[BP]+[DI]+d_{16}	BL	BX
100	[SI]	[SI]+d_8	[SI]+d_{16}	AH	SP
101	[DI]	[DI]+d_8	[DI]+d_{16}	CH	BP
110	d_{16} (Direct address)	[BP]+d_8	[BP]+d_{16}	DH	SI
111	[BX]	[BX]+d_8	[BX]+d_{16}	BH	DI

d_8 :- 8 bit displacement

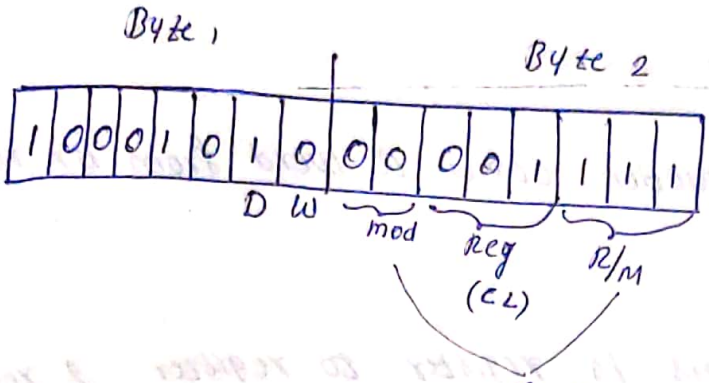
d_{16} :- 16 bit displacement.

Segment Reg	CODE
CS	01
DS	11
ES	00
SS	10

Eg: Generate Opcode for the following instruction

MOV CL, [BX]

This instruction copies a byte from the location whose effective address is in BX reg into CL reg



[BX] ⇒ mod = 00
R/M = 111

∴ 10001010 00001111
8A0F

Code for MOV CL, [BX] is 8A0F H

(2) MOV 43H [SI], DH ; This instn copies a byte from DH register in to memory location pointed by SI with displacement 43H

8 bit displacement

10001000 01 110 100 0100 0011
D mod Reg R/M 4 3 displacement

DH

SI

88 74 43 H

(3) MOV CX, [437AH]

This instruction copies content of two memory locations into CX register

10001011 00 001 110 0111 1010 0100 0011
mod Reg R/M 7A 43

⇒ 8B0E7A43H

4) MOV SP, BX

This instruction copies a word from BX register to SP register.

Since this is register to register, 2 registers are involved in the instruction. The move operation can be either to SP or from BX. When it is to SP, D=1. When it is from BX, D=0.

Case (i) To SP.

When it is to SP, D=1

1000101	1	11	100	011
D	W	mod	Reg	R/M

= 8BE3H

Case (ii)

from BX

When it from BX, D=0

1000100	1	11	011	100
D	W	mod	Reg	R/M

= 89DCH

5) MOV CS:[BX], DL

This instruction copies a byte from DL register into memory location in Code segment.

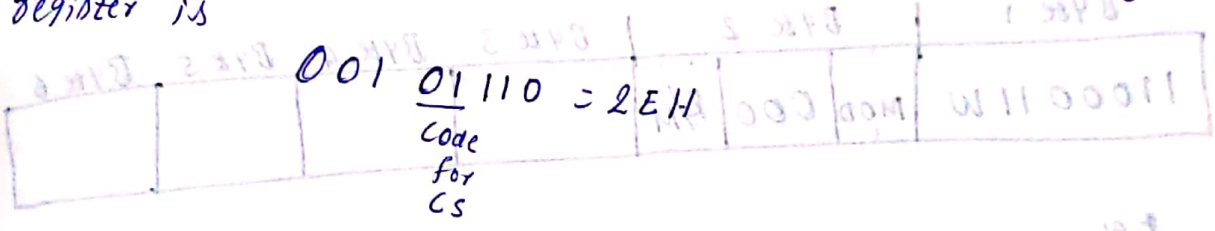
The segment register CS in an instruction

is called as Segment override prefix. The Segment override prefix has the format

001 XX 110

In place two bit code for segment registers in place of XX.

∴ The Segment override prefix byte for CS register is

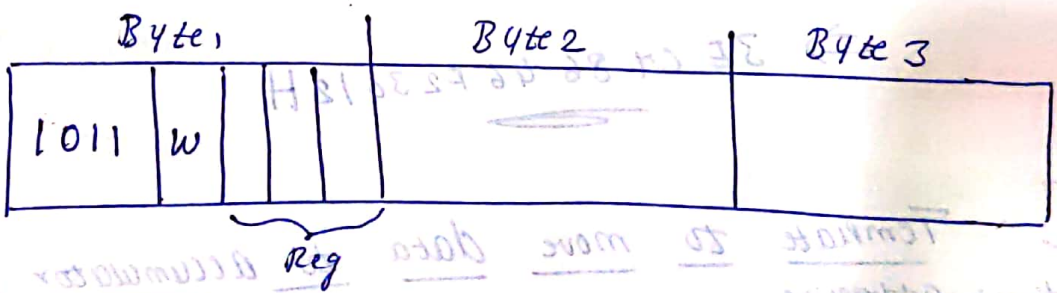


Segment override prefix is put in memory before the m/c code of an instruction.

10001000 00 010 111
D W mod Reg R/M

⇒ 2E 8817H

(ii) Template to move immediate data to register



Eg: (1) Mov AX, 1234H

This instruction moves immediate data 1234 to AX register.

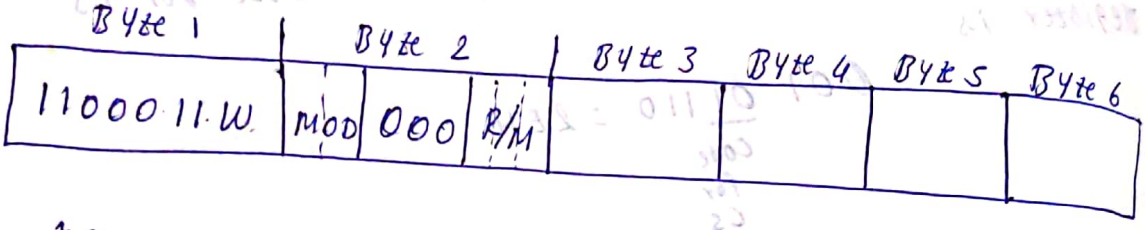
upper byte first lower byte next

1011 1 000 0011 0100 0001 0010 ⇒ B83412H
W AX 3 4 1 2

Eg: (2) MOV CL, 0F2H

1011 0 001 1111 0010 ⇒ B1F2H
W CL F 2

III. Template to move immediate data to memory location



Eg: MOV DS: F246H[BP], 1234H

Segment Override Prefix for DS register is

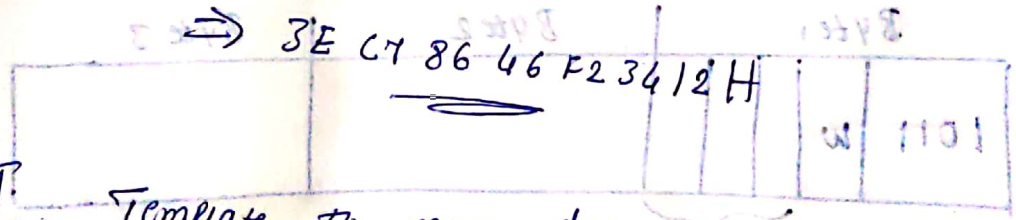
001 11110 = 3EH
DS code

for inst"

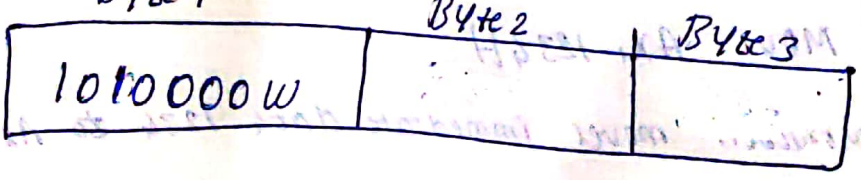
11000111 10 000 110 0100 0110 1111 0010
W MOD R/M 4 6 F 2

0011 0100 0001 0010
3 4 1 2

⇒ 3E C7 86 46 F2 34 12H



IV. Template to move data to accumulator using direct addressing



Eg: MOV AX, DS: 34

This instruction moves the word from the location 0034 in data segment to AX register.

Segment Override Prefix for DS register is

0011 1110 = 3EH.

for instn

1010 0001 0011 0100

w 3 4

3E A1 34

Segment Override Prefix

V Template to move data from accumulator to using direct addressing

Byte 1	Byte 2	Byte 3
1010 0001 w		

Eg: MOV SS:34H, AL

This instruction moves the byte from AL register to location 34H in the stack segment.

Segment Override Prefix for stack segment is

001 10 110 ⇒ 36

SS Code

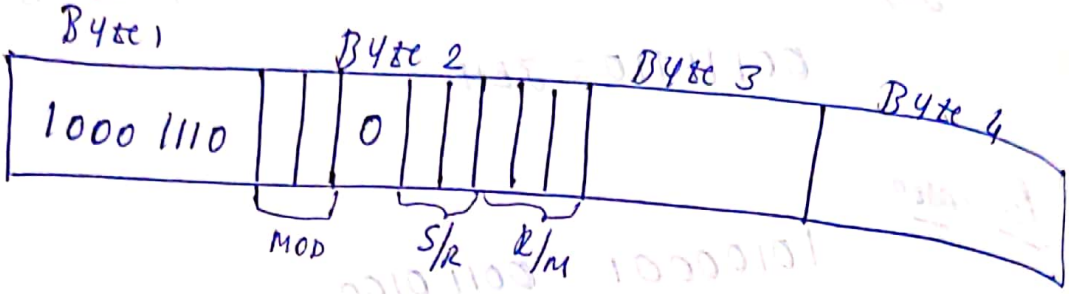
for instn

1010 0010 0011 0110

w

36 A2 34

VI Template to move data to segment register from register or memory location



Eg:-

① MOV DS, BX

This instruction moves data from BX register to DS register.

for DS register, Segment register code is 11

∴ 1000 1110 11 0 11 011 ⇒ 8EDBH

Mod S/R R/M

② MOV SS, 1234H[BX+SI]

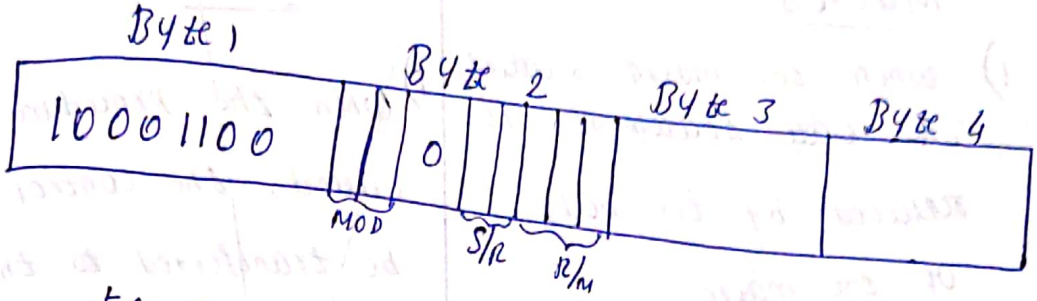
This instruction moves to SS from location, whose effective address is the sum of 16 bit displacement 1234 & content of BX & SI register.

1000 1110 10 0 10 000 0011 0100 0001 0010

MOD S/R R/M

⇒ 8E 90 34 12H

VII Template to move data from segment register to register or memory location



Eg: (1) MOV BX, DS

10001100 11 011 011 ⇒ 8C DBH

MOD S/R R/M

for BX reg

(2) MOV 1234H [BX][SI], SS

10001100 10010000 00110100 00010010

⇒ 8C 90 34 12H

NOTE

Templates

(1) Register to Register
Register to memory
memory to register

(2) Immediate data to register

(3) Immediate data to memory

(4) To move data to accumulator

(5) To move data from accumulator

(6) To move data to segment reg from reg/memory

(7) To move data from segment reg to reg/memory

Difference b/w Macros & Procedures

Macros	Procedures
1) When the macro is called, the call statement is replaced by the body of the macro	1) When the procedure is called, the control will be transferred to the beginning of the procedure
2) More memory is required, because code will be expanded at the place of call.	2) Less memory is required, because code will not be expanded at the place of call, instead control will be transferred to procedure block.
3) Stack is not required	3) Stack is required to store the return address of the program
4) Speed of execution of the program increases	4) Speed of execution of the program is less, because control has to be transferred to/from procedure
5) Macros are called as Open subroutine, because macro is replaced at the place of call	5) Procedures are called as closed subroutine
6) Macro is replaced during assembly time	6) Control is transferred to Procedure during run time.

7) Macro is defined using directives MACRO & ENDM

8) Eg:
```
Disp macro msg
    MOV AH, 09H
    LEA DX, msg
    INT 21H
Endm
```

7) Procedures are defined using the directives PROC & ENDP

8) Eg:
```
DISP PROC
    LEA DX, MSG
    MOV AH, 09
    INT 21H
DISP ENDP
```