Azmari Sultana

Id: 22201949

CSE341

Section: 05

Assignment 01

TOPIC NAME : Assignment 1 DAY : TIME : DATE : / /

(1) Purpose of the segment register:

i) To divide memory into logical sections such as code, data, stack and extra.

ii) To extend addressing beyond 64 KB using 16-bit segment and offset values.

iii) To allow program relocation, meaning programs can run from different memory locations.

iv) To provide better memory organization, keeping instruction data and stack separate.

v) To enable modular programming and efficient memory access.

Operation of the segment register:

i) Each segment register holds the starting address (base) of a memory segment that is 64KB long.

ii) The 8086 forms the 20-bit physical address by shifting the segment value 4 bits left and adding a 16-bit offset.

physical address = (segment register × 16) + offset.
= (segment reg × 10h) + offset

The four segment registers and their functions:

CS: Works with IP to fetch instructions

DS: Used for accessing data variables

SS: Used with SP and BP for stack operations

ES: Used as an additional data segment, mainly in string operations.

Differences from general purpose registers

i) Segment registers are used for memory addressing, not arithmetic.

ii) They hold base addresses, not data or results.

iii) They are special-purpose, while general purpose registers are flexible and used in calculations.

2) segment address = A4FBh

offset = 4872h

physical address = (segment address × 10h) + offset

= (A4FBh × 10h) + 4872h

= A4FB0h + 4872h.

= A9822h

(3)

$$\text{Number of segments} = \frac{\text{Total memory}}{\text{segment size}}$$

$$= \frac{1024 \times 1024}{64 \times 1024}$$

$$= 16$$

∴ 16 segments of 64KB can be allocated in the 1MB memory space.

(4)

Starting physical address = 30000h

5th smallest = 0000 + 4 = 00004h

5th largest = FFFF − 4 = FFFBh

<u>smallest:</u>

3000 × 10h + 0004 = 30004

∴ Logical address = 3000:0004

<u>Largest</u>

3000 × 10h + FFFB = 3FFFB

Logical address = 3000 : FFFB

Segment = 3000

(5) i)

Maximum size of memory $= 2^{\text{number of address lines}}$ bytes

$= 2^{32}$ bytes

$= 4294967296$ bytes

$= 4$ GB

ii) Maximum addressable memory = 4 GB

Inserted memory = 3 GB

As the maximum capacity is 4GB, so the ALU will able to generate unique address for each address. So, this is accessable if we insert a 3 GB memory.

iii) Maximum addressable memory = 4 GB

Inserted memory = 6 GB

As the ~~sy~~ maximum capacity is 4GB, address bus can only generate unique address for 4GB. If we insert 6GB memory, 2GB memory will be unaccessable. So, it's partially accessable, not fully accessable.

⑥ i) PF = 0, AF = 1

we can consider last 8 bit

CD = 1 1 0 0 1101
FE = 1 1 1 1 1110

1 1 0 0 1011

XY = FEh

PF = 0, AF = 0

XY = F2h

1 1 00 1 1 0 1
1 1 1 1 0 0 1 0

1 0 1 1 1 1 1 1

PF = 1, AF = 0

XY = F1h

1 1 00 1 1 0 1
1 1 1 1 0 0 0 1

1 0 1 1 1 1 1 0

PF = 1, AF = 1

1 1 0 0 1 1 0 1
1 1 1 1 1 1 1 1

1 1 0 0 1 1 0 0

XY = FFh

ii) PF = 0 , AF = 1

$$\begin{array}{r} 1100\,1101 \\ 0000\,0011 \\ \hline 1101\,0000 \end{array}$$

XY = 03h

PF = 1 , AF = 0

$$\begin{array}{r} 1100\,1101 \\ 0001\,0000 \\ \hline 1101\,1101 \end{array}$$

XY = 10h

PF = 0 , AF = 0

$$\begin{array}{r} 1100\,1101 \\ 0000\,0000 \\ \hline 1100\,1101 \end{array}$$

XY = 00h

PF = 1 , AF = 1

$$\begin{array}{r} 1100\,1101 \\ 0000\,0100 \\ \hline 1101\,0001 \end{array}$$

XY = 04h

~~6~~ 7 (i)

7E40 = 0111 1110 0100 0000

3BC0 = 0011 1011 1100 0000

1011 1010 0000 0000

PF = 1 AF = ~~0~~ ~~1~~ 0

CF = 0 ZF = 0

SF = 1 OF = 1

(ii)

FF = 1111 1111

01 = 0000 0001

10000 0000

PF = 1 ZF = 1

AF = 1 SF = 0

CF = 1 OF = 0

(8) a) $PA = CS * 10h + IP$

$= 2000 * 10 + 3456h$

$= 23456h$

b) $PA = SS * 10h + [BP + SI + 2000h]$

$= 2000 * 10h + [2000 + 3000 + 2000]$

$= 27000h$

Data $= 7856h$

(9)

i) direct addressing

ii) Register indirect

iii) register relative

iv) Base relative plus indexing

v) invalid

vi) addressing stack

vii) addressing stack

viii) invalid

ix) invalid

x) addressing stack

xi) I/O addressing IN

xii) invalid

xiii) I/O addressing IN

xv) I/O addressing OUT

xvi) implied

xvii) implied

xviii) implied

xix) implied

xx) implied

(10) i) MOV AX, [BX + 5634h]

Byte 1								Byte 2							
1	0	0	0	1	0	1	1	1	0	0	0	0	1	1	1
Opcode						D	W	MOD		REG			R/M		

Byte 3								Byte 4							
0	0	1	1	0	1	0	0	0	1	0	1	0	1	1	0

Hex = 8B873456h

ii) 89879140h

Byte 1								Byte 2							
1	0	0	0	1	0	0	1	1	0	0	0	0	1	1	1
Opcode						D	W	MOD		REG			R/M		

MOD [BX + 4091h], AX