Section:	All (Your Section)	Reg. No	
Exam:	Sessional II		

Instruction/Notes: This is an open note/book exam. All the answers should be written in provided space on this paper. Rough sheets can be used but will not be collected and checked. In case of any ambiguity, take reasonable assumption. Questions during exam are not allowed.

Question 1: Find the physical address of the following given the values of registers and variables. You answers should be in hex (Show calculations on given space, no marks without calculations) [5 marks]

1 point each.
If you have not shown complete answer, you will loose mark, as it is needed to show concepts of wrap arround

CS= FFF3h DS= 00BFh SS= 0A77h ES=F810h
BX= FF20h IP= 0020h SP= EE00h SI=0010h num= 8954h

1. [num] DS : Num = 00BF0 + 8954 = 09544

2. [BX+0200h] 00BF0 + (FF20 + 200) = 00BF0 + 0120
= 00D10

3. [CS:BX+SI] FFF30 + (FF20 + 10h) = FFF30 + FF30
= 0FE60.

4. [IP] FFF30 + 0020h = FFF50h

5. [SP] 0A770 + EE00 = 19570

Question 2: The following code is to print a blinking "7" in red color with blue background on the 5th row and 10th column of screen. Fill in the blanks with correct screen offset and word is to be written on it. [5 marks]

```
Mov ax, 0xb800
Mov es, ax
Mov di, ____820____
Mov [es:di], ____~~80~~ 9C37 or 9437
```

1 di
1 '7'
1 blink
1 BG color
1 FG color

Show calculations of Q2 here

Roll Number: _____ Section: _____

Question 3: Following function **print_str** takes a string address in bx, and string's length in cx and prints it on the top left corner of screen. Change the function such that the whole string is printed in upper case.
Assume that string will only have characters from a-z or A-Z or space
Note: You cannot change the Start code or the data, you can only change the body of the function. You can also not change the way input is being passed to the function. **[5 marks]**

```
; function before change
print_str:
   ;body
   ;es:di point to top left corner of screen
   push 0xb800
   pop es
   mov di,0
   mov ah,07h ; attribute of printing
   l1:
       mov al , [bx]; read a char of string
       mov [es:di],ax
       inc bx
       add di,2
       loop l1
   ;end of body
   ret
```

Write required changes in function here

cmp al, 60h or 97d
jl print
sub al, 20h ; or 32d.
print ;

you will loose marks if you have subtracted 32d from all characters.

Sample run of function print_str:

```
string: db "This is COAL exam"
length: dw 17
start:
      mov bx, string
      mov cx, [length]
      call print_str
      mov ax, 0x4c00
      int 21h
```

Result of the given function is
This is COAL exam

Result of the function after changes should be
THIS IS COAL EXAM

Ascii Table

000	(nul)	016 ►	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001 ☺	(soh)	017 ◄	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002 ☻	(stx)	018 ↕	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003 ♥	(etx)	019 ‼	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004 ♦	(eot)	020 ¶	(dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005 ♣	(enq)	021 §	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006 ♠	(ack)	022 ▬	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007 •	(bel)	023 ↨	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008 ◘	(bs)	024 ↑	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009	(tab)	025 ↓	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011 ♂	(vt)	027 ←	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012 ♀	(np)	028 └	(fs)	044	,	060	<	076	L	092	\	108	l	124	|
013	(cr)	029 ↔	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014 ♫	(so)	030 ▲	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015 ☼	(si)	031 ▼	(us)	047	/	063	?	079	O	095					

FAST School of

Roll Number: _____ Section: _____

Question 4: The following code is of custom ISR to hook interrupt 0. Identify the errors, if any, and make corrections in given space. Only give correction in given space don't copy correct part again. **[5 marks]**

```
;Current code
Jmp start
my_custom_ISR:
; body if ISR
Ret    ; return after ISR

Start:
       xor ax, ax
       mov es, ax ; load zero in es
       mov word [es:0*4+2], my_custom_ISR; store offset at n*4
       mov [es:0*4], cs ; store segment at n*4+2
       mov ax, 0x4c00 ; terminate program
       int 0x21
```

Your corrections go in this column

iret

mov word [es:0x4+2], cs

mov word [es:0x4], my_cust_ISR

Question 5: [5 marks]

```
[org 0x0100]
jmp start
num1: db 10, 20, 30, 40, 50
num2: db 0, 0, 0 ,0, 0;
len: dw 5

start:
       mov si, num1
       mov di, num2
       add di, [len]
       dec di
       mov ax, ds
       mov es, ax
       mov cx, [len]

       cld; clear direction flag

       loopC:
              movsb
              sub di, 2
              loop loopC
```

Dry run the code given in Left column and find the values of array num2, and registers si and di at the end of code?
Assume that address of num1 is 0103h

Num2: 50, 40, 30, 20, 10

si: 108h

di: 107h

Question 6: [5 marks]

```
[org 0x0100]
jmp start
str1: db 'ABCDEF'
str2: db 'ABCDTF'
length: dw 6 ; length of the strings

start:
    mov si, str1
    mov di, str2
    mov cx, [length]
    cld
    repe cmpsb
```

Dry-run the code given in Left column and find the following values after the execution of complete code

CX= ~~108h~~ 1

DI= 10Eh

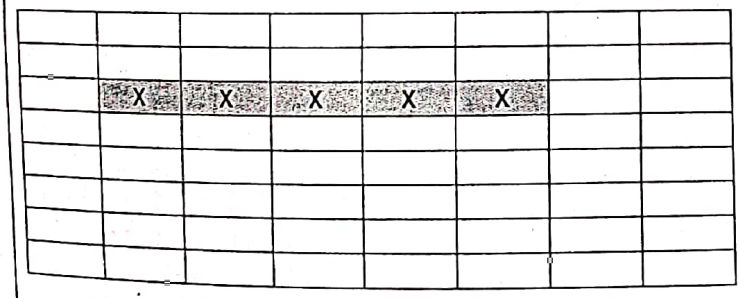
SI= 108h

Given that
Str1 address is 0103
Str2 address is 0109

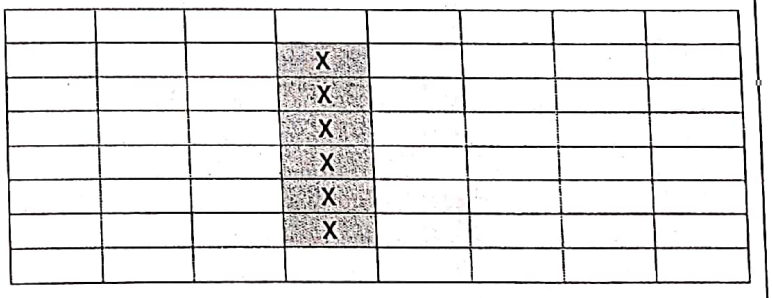
Most students have written 2, 100, 107 they only got 2.5 marks

Question 7: Write a subroutine **DrawLine** that takes four parameters r1, c1, r2 and c2; where r1, c1 are co-ordinates of Point1 and r2, c2 are co-ordinates of Point2. **The function will draw a line between Point1 and Point2 on Display Memory.** Assume that two points will always be connected through one of following lines: horizontal, vertical, right diagonal or left diagonal, depending upon the position of the points. Sample run for four different lines is shown below. **[15 marks]**

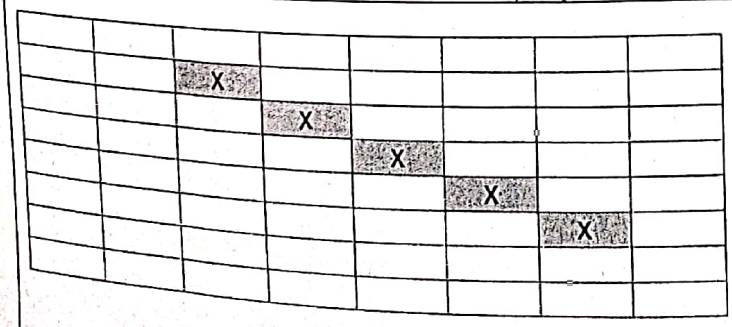
Note: The line in the sample run is drawn using character 'X' on a grid of 8x8 cells.



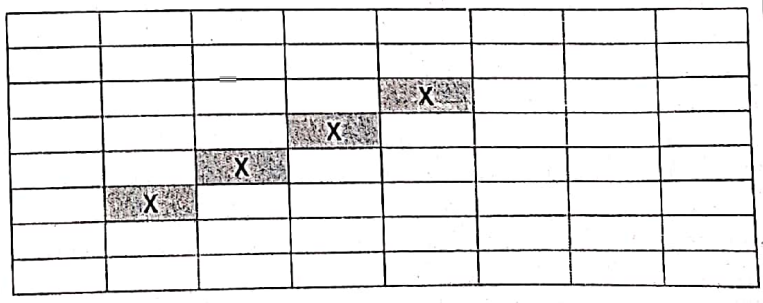
Input: Point1(r1=2, c1=1) and Point2(r2=2,c2=5)
Output: Horizontal Line displayed



Input: Point1(r1=1, c1=3) and Point2(r2=6,c2=3)
Output: Vertical Line displayed



Input: Point



; Start your code here:
; You are required to write the complete program including the function call.

```
jmp start:

start:
    push r1
    push c1
    push r2
    push c2
    call drawline
    mov ax, 4c00h
    int 21h
```


```
; function to draw on char in line
draw:
    push bp
    mov bp,sp
    ; push registers
    mov bx, [bp+4] → col
    mov ax, [bp+6] → row
    ; cal [80*ax+bx]*2
    mul 80
    add bx, ax
    shl bx, 1
    mov ax, b800h
    mov es, ax
    mov [es:bx], 0784h ; 'x'
    pop reg
    pop bp
    ret
```

```
drawline:
    push bp
    mov bp, sp
    ; push registers
    call clr-screen
    ; read input
    mov ax, [bp+10] ; r1
    mov bx, [bp+8] ; c1
    mov cx, [bp+6] ; r2
    mov dx [bp+4]; c2

    cmp ax, cx
    je horizontal-line
    cmp bx, dx
    je vertical-line

    cmp ax, cx
    jl check_dia
    mov si, ax
    mov ax, cx
    mov cx, si
    mov si, bx
    mov bx, dx
    mov dx, si
```

make sure that for diagonal line r1 < r2 if not swap r1 with r2 c2

```
check_dia:
    cmp bx, dx
    jl right-diagnol
    cmp bx, dx
    jg left_diagnol

end: ; pop registers
    ret ; pop bp
```

Roll Number: _____ Section: _____

```
horizontal_lines:
; make sure c1<c2
    cmp bx, dx
    jle loop-hl
    mov si, bx
    mov bx, dx
    mov dx, si

; draw a line in loop in row ax
; from bx to dx

loop-hl:
    push ax
    push bx
    call draw
    add bx, 1
    cmp bx, dx
    jne loop-hl
jmp end
```

```
vertical_lines
; make sure r1<r2
cmp ax, cx
jle loop vl
mov si, ax
mov ax, cx
mov cx, si
; draw a line in col bx
; from row ax to cx

loop-vl:
    push ax
    push bx
    call draw
    add ax, 1
    cmp ax, cx
    jne loop-vl
jmp end
```

```
right-diagnol:
; draw a line from ax to cx row
to bx to dx column

loop_rd
    push ax
    push bx
    call draw
    add ax, 1
    ax bx, 1
    cmp ax, cx
    jne loop-rd
jmp end
```

```
left_diagnols
; draw a line from ax to cx
row & bx to dx column

loop-ld:
    push ax
    push bx
    call draw
    add ax, 1
    Sub bx, 1
    jne cmp ax, c
        loop-ld
jmp end
```