Microprocessor Practical File

Name – Gourab Das

Course - B.Sc (Hons) Computer Science

Roll no - 88006

University Roll no - 20003570005

**1. Addition of two 16 bit no. using 16 bit registers**

.MODEL SMALL

.DATA

AA DW 1234H

BB DW 0BCDEH

.CODE

.STARTUP

MOV BX,WORD PTR AA

MOV CX,WORD PTR BB

ADD BX,CX

MOV AX,BX

.EXIT

END

**2. Addition of two 32 bit no. using 16 bit registers**

.MODEL SMALL

.DATA

AA DD 12345678H

BB DD 89ABCDEFH

.CODE

.STARTUP

MOV BX,WORD PTR AA ;5678

MOV AX,WORD PTR AA+2 ;1234

MOV DX,WORD PTR BB ;CDEF

MOV CX,WORD PTR BB+2 ;89AB

ADD BX,DX ;2467

ADC AX,CX ;9BE0

.EXIT

END

**3. Subtraction of two 16 bit no. using 16 bit registers**

.MODEL SMALL

.DATA

AA DW 0BCDEH

BB DW 1234H

.CODE

.STARTUP

MOV BX,WORD PTR AA

MOV CX,WORD PTR BB

SUB BX,CX

MOV AX,BX

.EXIT

END

**4. Subtraction of two 32 bit no. using 16 bit registers**

.MODEL SMALL

.DATA

AA DD 12345678H

BB DD 021399BDH

;AA-BB

.CODE

.STARTUP

MOV BX,WORD PTR AA ;5678

MOV AX,WORD PTR AA+2 ;1234

MOV DX,WORD PTR BB ;99BD

MOV CX,WORD PTR BB+2 ;0213

SUB BX,DX ;BCBB

SBB AX,CX ;1020

END

**5. Multiplication of two 32 bit No (User input)**

.MODEL SMALL

.DATA

ad1 DW ?

ad2 DW ?

Msg1 DD 'Enter First Number :$'

Msg2 DD 10,13,'Enter Second Number :$'

Msg3 DD 10,13,'Product :$'

.CODE

.STARTUP

LEA EDX,Msg1

MOV AH,9

INT 21H

MOV EBX,0H

MOV ECX,2

loop1: SHL EBX,8

;FIRST NUMBER FIRST DIGIT INPUT

MOV AH,1

INT 21H

CMP AL,39H

JBE jump1

SUB AL,37H

jump1: AND AL,0FH

SHL AL,4

MOV BL,AL

;FIRST NUMBER SECOND DIGIT INPUT

MOV AH,1

INT 21H

CMP AL,39H

JBE jump2

SUB AL,37H

jump2: AND AL,0FH

ADD BL,AL

LOOP loop1

MOV ad1,EBX

LEA EDX,Msg2

MOV AH,9

INT 21H

MOV EBX,0H

MOV ECX,2

loop2: SHL EBX,8

;SECOND NUMBER FIRST DIGIT INPUT

MOV AH,1

INT 21H

CMP AL,39H

JBE jump3

SUB AL,37H

jump3: AND AL,0FH

SHL AL,4

MOV BL,AL

;SECOND NUMBER SECOND DIGIT INPUT

MOV AH,1

INT 21H

CMP AL,39H

JBE jump4

SUB AL,37H

jump4: AND AL,0FH

ADD BL,AL

LOOP loop2

MOV EAX,ad1

MUL EBX

MOV EBX,EDX

MOV ad1,EAX

LEA EDX,Msg3

MOV AH,9

INT 21H

MOV ECX,4

rakshit: ROL EBX,4

MOV AL,BL

AND AL,0FH

CMP AL,09H

JBE rak

ADD AL,37H

JMP shi

rak: ADD AL,30H

shi: MOV DL,AL

MOV AH,2

INT 21H

LOOP rakshit

MOV EBX,ad1

MOV ECX,4

rakshit1: ROL EBX,4

MOV AL,BL

AND AL,0FH

CMP AL,09H

JBE rak1

ADD AL,37H

JMP shi1

rak1: ADD AL,30H

shi1: MOV DL,AL

MOV AH,2

INT 21H

LOOP rakshit1

.EXIT

END

;F298 X ABCD = A2CDCBB8

**6. Division of two 32 bit No. (User input)**

.MODEL SMALL

.DATA

ad1 DW ?

ad2 DW ?

Msg1 DD 'Enter First Number :$'

Msg2 DD 10,13,'Enter Second Number :$'

Msg3 DD 10,13,'Product :$'

.CODE

.STARTUP

LEA EDX,Msg1

MOV AH,9

INT 21H

MOV EBX,0H

MOV ECX,2

loop1: SHL EBX,8

;FIRST NUMBER FIRST DIGIT INPUT

MOV AH,1

INT 21H

CMP AL,39H

JBE jump1

SUB AL,37H

jump1: AND AL,0FH

SHL AL,4

MOV BL,AL

;FIRST NUMBER SECOND DIGIT INPUT

MOV AH,1

INT 21H

CMP AL,39H

JBE jump2

SUB AL,37H

jump2: AND AL,0FH

ADD BL,AL

LOOP loop1

MOV ad1,EBX

LEA EDX,Msg2

MOV AH,9

INT 21H

MOV EBX,0H

MOV ECX,2

loop2: SHL EBX,8

;SECOND NUMBER FIRST DIGIT INPUT

MOV AH,1

INT 21H

CMP AL,39H

JBE jump3

SUB AL,37H

jump3: AND AL,0FH

SHL AL,4

MOV BL,AL

;SECOND NUMBER SECOND DIGIT INPUT

MOV AH,1

INT 21H

CMP AL,39H

JBE jump4

SUB AL,37H

jump4: AND AL,0FH

ADD BL,AL

LOOP loop2

MOV AX,ad1

MOV DX,0000H

DIV BX

MOV ad1,DX

MOV BX,AX

LEA EDX,Msg3

MOV AH,9

INT 21H

MOV ECX,4

rakshit: ROL EBX,4

MOV AL,BL

AND AL,0FH

CMP AL,09H

JBE rak

ADD AL,37H

JMP shi

rak: ADD AL,30H

shi: MOV DL,AL

MOV AH,2

INT 21H

LOOP rakshit

MOV EBX,ad1

MOV ECX,4

rakshit1: ROL EBX,4

MOV AL,BL

AND AL,0FH

CMP AL,09H

JBE rak1

ADD AL,37H

JMP shi1

rak1: ADD AL,30H

shi1: MOV DL,AL

MOV AH,2

INT 21H

LOOP rakshit1

.EXIT

END

;F298 X ABCD = A2CDCBB8

**7. Addition of Two arrays**

.MODEL SMALL

.DATA

A DB 1,2,3,4,5,6,7,8

B DB 8,7,6,5,4,3,2,1

C DB 8 (?)

.CODE

.STARTUP

MOV SI,OFFSET A

MOV BX,OFFSET B

MOV DI,OFFSET C

MOV CX,08H

rashu: MOV AL,[SI]

MOV DL,[BX]

ADD AL,DL

MOV [DI],AL

INC SI

INC DI

INC BX

LOOP rashu

.EXIT

END

**8. Subtraction of two arrays**

.MODEL SMALL

.DATA

A DB 1,2,3,4,5,6,7,8

B DB 8,7,6,5,4,3,2,1

C DB 8 (?)

.CODE

.STARTUP

MOV SI,OFFSET A

MOV BX,OFFSET B

MOV DI,OFFSET C

MOV CX,08H

rashu: MOV AL,[SI]

MOV DL,[BX]

SUB AL,DL

MOV [DI],AL

INC SI

INC DI

INC BX

LOOP rashu

.EXIT

END

**9. Binary Search within a set of numbers.**

.MODEL SMALL

.DATA

prompt0 DB 'Enter Length of Array (<=10): $'

prompt1 DB 13, 10, 'Enter Elements: $'

prompt2 DB 13, 10, 'Enter Key: $'

msg1 DB 13, 10, 'Element Not Found in array!$'

msg2 DB 13, 10, 'Element Found in array!$'

array DB 10 DUP(?)

len DW ?

key DB ?

.CODE

.STARTUP

LEA DX, prompt0

MOV AH, 09H

INT 21H

MOV AH, 01H

INT 21H

MOV AH, 00H

SUB AL, 30H

MOV len, AX

LEA DX, prompt1

MOV AH, 09H

INT 21H

LEA SI, array

MOV AL, key

MOV CX, len

input:

MOV AH, 01H

INT 21H

MOV [SI], AL

INC SI

MOV DL, 20H

MOV AH, 02H

INT 21H

LOOP input

LEA DX, prompt2

MOV AH, 09H

INT 21H

MOV AH, 01H

INT 21H

MOV key, AL

LEA SI, array

MOV CX, len

search:

MOV BL, [SI]

CMP AL, BL

JE found

INC SI

LOOP search

LEA DX, msg1

MOV AH, 09H

INT 21H

JMP final

found:

LEA DX, msg2

MOV AH, 09H

INT 21H

final:

.EXIT

END

**10. Linear Search**

.model small ; contain two segment data and code

.STACK ; tells the assembler to reserve storage

.DATA ; start of data segment

ARRAY DB 9 DUP(?) ; Declaring an array with garbage

MESS01 DB 13,10,"MAX. NO. OF ELEMENTS IN ARRAY IS 9 $"

MESS02 DB 13,10," $" ; 10 is the ASCII control code for line feed while 13 is the code for carriage return

MESS1 DB 13,10,"ENTER THE NUMBER OF ELEMENTS: $"

MESS0 DB 13,10,"ENTER THE NUMBER: $"

MESS2 DB 13,10,"ENTER THE ELEMENT TO BE SEARCHED: $"

MESS3 DB 13,10,"VALUE FOUND AT LOCATION - $"

MESS4 DB 13,10,"VALUE NOT FOUND!!!$"

ErrMess DB 13,10,"ERROR IN INPUT DIGIT$"

DAT DB ? ; set byte size variable

number dw ? ; set double word variable

POS DW ? ; set double word variable

.code ; start of code segment

.startup ; Generates program start-up code

MOV DX, OFFSET MESS01

MOV AH, 09

INT 21H ; Output a string terminated by '$’ stored in DX, as AH =9

MOV DX, OFFSET MESS02

MOV AH, 09

INT 21H ; Output a string terminated by '$’ stored in DX, as AH =9

MOV DX,OFFSET MESS1

MOV AH, 09

INT 21H ; Output a string terminated by '$’ stored in DX, as AH =9

MOV AH, 01

INT 21H ; input from user

CMP al,39h

JBE abc ; jump if below or equal to

MOV DX, OFFSET ErrMess

MOV AH, 09

INT 21H ; Output a string terminated by '$’ stored in DX, as AH =9

JMP myexit

abc: AND AL, 0FH

MOV AH, 0

MOV number, AX

MOV CX, AX ; SET COUNTER AL TIMES

MOV DI, 0

; INPUT ELEMENTS IN ARRAY

MYLOOP:

MOV DX, OFFSET MESS0

MOV AH, 09

INT 21H ; Output a string terminated by '$’ stored in DX, as AH =9

; Tens digit

MOV AH, 01

INT 21H ; input from user

CMP AL, 39H

JBE abc2 ;; jump if below or equal to

MOV DX, OFFSET ErrMess

MOV AH,09

INT 21H ; Output a string terminated by '$’ stored in DX, as AH =9

JMP myexit

abc2: AND al,0fh

SHL AL, 4 ; multiply by 16

MOV BL, AL

; Units digit

MOV AH,01

INT 21H

cmp al,39h

jbe abcx ; jump if below or equal to

MOV DX,OFFSET ErrMess

MOV AH,09

INT 21H ; Output a string terminated by '$’ stored in DX, as AH =9

jmp myexit

abcx:

AND al,0fh

ADD al, bl

MOV ARRAY[DI], AL

INC DI

LOOP MYLOOP

;INPUT ELEMENT TO BE SEARCHED

MOV DX,OFFSET MESS2

MOV AH,09

INT 21H ; Output a string terminated by '$’ stored in DX, as AH =9

; Tens digit

MOV AH,01

INT 21H

cmp al,39h

jbe abcl ; jump if below or equal to

MOV DX,OFFSET ErrMess

MOV AH,09

INT 21H

jmp myexit

abcl:

and al,0fh

shl al,4 ; multiply by 16

mov bl,al

; Units digit

MOV AH,01

INT 21H ; input from user

cmp al,39h

jbe abcm ; jump if below or equal to

MOV DX,OFFSET ErrMess

MOV AH,09

INT 21H ; Output a string terminated by '$’ stored in DX, as AH =9

jmp myexit

abcm:

and al,0fh

add al,bl

mov DAT,AL

; SEARCH PROCESS

MOV AX, DS

MOV ES, AX

MOV AL, DAT

CLD ; Auto-Increment Mode

MOV CX, number

MOV DI, OFFSET ARRAY

REPNE SCASB

CMP CX, 0

JE NOTFOUND

MOV DX, OFFSET MESS02

MOV AH, 09

INT 21H ; Output a string terminated by '$’ stored in DX, as AH =9

MOV DX, OFFSET MESS3

MOV AH,09

INT 21H ; Output a string terminated by '$’ stored in DX, as AH =9

SUB NUMBER, CX

ADD NUMBER,30H ; convert the hexadecimal digits into its equivalent ASCII

MOV DX, NUMBER

MOV AH, 02

INT 21H ; Output a character present in DL , as AH value is 2

JMP myexit

NOTFOUND:

MOV DX,OFFSET MESS4

MOV AH,09

INT 21H ; Output a string terminated by '$’ stored in DX, as AH =9

myexit:

MOV DX, OFFSET MESS02

MOV AH, 09

INT 21H ; Output a string terminated by '$’ stored in DX, as AH =9

.EXIT

END

**11. Addition of two 16 bit BCD No.**

; doing bcd addition with 1234H and 0022H

.MODEL SMALL

.486

.DATA

data1 DW 1234H

data2 DW 0022H

result DD ?

.CODE

.STARTUP

MOV BX, data1

MOV DX, data2

ADD DL, BL

MOV AL, DL

DAA

MOV CL, AL

ADC DH, BH

MOV AL, DH

DAA

MOV CH, AL

MOV result, CX

.EXIT

END

**12. Addition of two 32 bit BCD nos.**

; doing bcd addition with 12345678H and 00000022H

.MODEL SMALL

.486

.DATA

data1 DD 12345678H

data2 DD 00000022H

result DD ?

.CODE

.STARTUP

MOV EBX, data1

MOV EDX, data2

ADD DL, BL

MOV AL, DL

DAA

MOV CL, AL

ADC DH, BH

MOV AL, DH

DAA

MOV CH, AL

BSWAP ECX

SHR EBX, 16

SHR EDX, 16

ADC DL, BL

MOV AL, DL

DAA

MOV CH, AL

ADC DH, BH

MOV AL, DH

DAA

MOV CL, AL

BSWAP ECX

MOV result, ECX

.EXIT

END

**13. Subtraction of two 16 bit BCD No.**

; doing bcd subtraction with 9876H and 8654H

.MODEL SMALL

.DATA

data1 DW 9876H

data2 DW 8654H

result DW ?

.CODE

.STARTUP

MOV DX, data1

MOV BX, data2

SUB DL, BL

MOV AL, DL

DAS

MOV CL, AL

SBB DH, BH

MOV AL, DH

DAS

MOV CH, AL

MOV result, CX

.EXIT

END

**14. Subtraction of two 32 bit BCD No.**

; doing bcd subtraction with 12345678H and 00000022H

.MODEL SMALL

.486

.DATA

data1 DD 12345678H

data2 DD 00000022H

result DD ?

.CODE

.STARTUP

MOV EDX, data1

MOV EBX, data2

SUB DL, BL

MOV AL, DL

DAS

MOV CL, AL

SBB DH, BH

MOV AL, DH

DAS

MOV CH, AL

BSWAP ECX

SHR EBX, 16

SHR EDX, 16

SBB DL, BL

MOV AL, DL

DAS

MOV CH, AL

SBB DH, BH

MOV AL, DH

DAS

MOV CL, AL

BSWAP ECX

MOV result, ECX

.EXIT

END