

***MICROPROCESSOR***

PRACTICALS

***Name : Saksham***

***Examination Roll No. : 20020570028***

***College Roll No. : 20201438***

***Course : B.SC.(H) COMPUTER SCIENCE***

***Semester : 5***

**1) Write a program for 32-bit binary division and multiplication**

**A) BINARY DIVISIONSS**

.MODEL SMALL

.DATA

prompt1 DB 'Enter the 64-bit dividend: $'

prompt2 DB 13, 10, 'Enter the 32-bit divisor: $'

message1 DB 13, 10, 'Quotient: $'

message2 DB 13, 10, 'Remainder: $'

operandms DD ?

operandls DD ?

resultq DD ?

resultr DD ?

.CODE

.STARTUP

; prompt user for the 64-bit dividend

MOV DX, OFFSET prompt1

MOV AH, 09H

INT 21H

; clear EBX for holding input

XOR BX, BX

; initialise counter for loop

MOV CX, 04H

input1:

; shift content of EBX for next byte

SHL BX, 8

; accept first digit

MOV AH, 01H

INT 21H

; check if it is a valid digit

CMP AL, 39H

JBE letter1

; adjust letters to hex

SUB AL, 37H

letter1:

; adjust hex characters to hex from ascii

AND AL, 0FH ; mask contents in AL

SHL AL, 4 ; shift contents in AL

MOV BL, AL

; accept second digit

MOV AH, 01H

INT 21H

; check if it is a valid digit

CMP AL, 39H

JBE letter2

; adjust letters to hex

SUB AL, 37H

letter2:

; adjust hex characters to hex from ascii

AND AL, 0FH ; mask contents in AL

ADD BL, AL ; adjust second digit in chunk

LOOP input1

; store the upper nibble to memory

MOV operandms, BX

; clear EBX for holding input

XOR BX, BX

; initialise counter for loop

MOV CX, 04H

input1a:

; shift content of EBX for next byte

SHL BX, 8

; accept first digit

MOV AH, 01H

INT 21H

; check if it is a valid digit

CMP AL, 39H

JBE letter1a

; adjust letters to hex

SUB AL, 37H

letter1a:

; adjust hex characters to hex from ascii

AND AL, 0FH ; mask contents in AL

SHL AL, 4 ; shift contents in AL

MOV BL, AL

; accept second digit

MOV AH, 01H

INT 21H

; check if it is a valid digit

CMP AL, 39H

JBE letter2a

; adjust letters to hex

SUB AL, 37H

letter2a:

; adjust hex characters to hex from ascii

AND AL, 0FH ; mask contents in AL

ADD BL, AL ; adjust second digit in chunk

LOOP input1a

; store the lower nibble to memory

MOV operandls, BX

; prompt user for the 32-bit divisor

MOV DX, OFFSET prompt2

MOV AH, 09H

INT 21H

; clear EBX for holding input

XOR BX, BX

; initialise counter for loop

MOV CX, 04H

input2:

; shift content of EBX for next byte

SHL BX, 8

; accept first digit of chunk

MOV AH, 01H

INT 21H

; check if it is a valid digit

CMP AL, 39H

JBE letter3

; adjust letters to hex

SUB AL, 37H

letter3:

; adjust hex characters to hex from ascii

AND AL, 0FH ; mask contents in AL

SHL AL, 4 ; shift contents in AL

MOV BL, AL

; accept second digit of chunk

MOV AH, 01H

INT 21H

; check if it is a valid digit

CMP AL, 39H

JBE letter4

; adjust letters to hex

SUB AL, 37H

letter4:

; adjust hex characters to hex from ascii

AND AL, 0FH ; mask contents in AL

ADD BL, AL ; adjust second digit in chunk

LOOP input2

; copy first number to EAX and clear EDX

MOV DX, operandms

MOV AX, operandls

; generate quotient and remainder

DIV BX

; copy quotient to ECX

MOV CX, AX

; move quotient and remainder

MOV resultq, CX

MOV resultr, DX

; print the quotient

MOV DX, OFFSET message1

MOV AH, 09H

INT 21H

; copy the quotient

MOV BX, resultq

; initialise the counter for loop

MOV CX, 04H

print1:

; rotate the contents of EBX

ROL BX, 8

; convert hex to ascii

MOV AL, BL

AND AL, 0F0H ; mask contents in AL

SHR AL, 4 ; shift msb in AL

ADD AL, 30H ; adjust hex to ascii

CMP AL, 39H ; check if it is a digit

JBE print1sub1

ADD AL, 07H ; adjust letters to ascii

print1sub1:

; print the first character of chunk

MOV DL, AL

MOV AH, 02H

INT 21H

; convert hex to ascii

MOV AL, BL

AND AL, 0FH ; mask contents in AL

ADD AL, 30H ; adjust hex to ascii

CMP AL, 39H ; check if it is a digit

JBE print1sub2

ADD AL, 07H ; adjust letters to ascii

print1sub2:

; print the second character of chunk

MOV DL, AL

MOV AH, 02H

INT 21H

LOOP print1 ; loop until all digits are printed

; print the quotient

MOV DX, OFFSET message2

MOV AH, 09H

INT 21H

; copy the remainder

MOV BX, resultr

; initalise the counter for loop

MOV CX, 04H

print2:

; rotate the contents of EBX

ROL BX, 8

MOV AL, BL

AND AL, 0F0H ; mask contents in AL

SHR AL, 4 ; shift msb in AL

ADD AL, 30H ; adjust hex to ascii

CMP AL, 39H

JBE print2sub1

ADD AL, 07H ; adjust letters to ascii

print2sub1:

; print the first character of chunk

MOV DL, AL

MOV AH, 02H

INT 21H

MOV AL, BL

AND AL, 0FH ; mask contents in AL

ADD AL, 30H ; adjust hex to ascii

CMP AL, 39H ; check if it is a digit

JBE print2sub2

ADD AL, 07H ; adjust letter to ascii

print2sub2:

; print the second character of chunk

MOV DL, AL

MOV AH, 02H

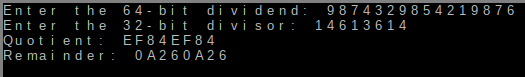
INT 21H

LOOP print2 ; loop until all digits are printed

.EXIT

END

**OUTPUT**

****

**B)BINARY MULTIPLICATION**

.MODEL SMALL

.DATA

prompt1 DB 'Enter the first 32-bit number: $'

prompt2 DB 13, 10, 'Enter the second 32-bit number: $'

message DB 13, 10, '64-bit Product: $'

operand DD ?

resultms DD ?

resultls DD ?

.CODE

.STARTUP

; prompt user for first number

MOV DX, OFFSET prompt1

MOV AH, 09H

INT 21H

; clear EBX for holding input

XOR BX, BX

; initialise counter for loop

MOV CX, 04H

input1:

; shift content of EBX for next byte

SHL BX, 8

; accept first digit

MOV AH, 01H

INT 21H

; check if it is a valid digit

CMP AL, 39H

JBE letter1

; adjust letters to hex

SUB AL, 37H

letter1:

; adjust hex characters to hex from ascii

AND AL, 0FH ; mask contents in AL

SHL AL, 4 ; shift contents in AL

MOV BL, AL

; accept second digit

MOV AH, 01H

INT 21H

; check if it is a valid digit

CMP AL, 39H

JBE letter2

; adjust letters to hex

SUB AL, 37H

letter2:

; adjust hex characters to hex from ascii

AND AL, 0FH ; mask contents in AL

ADD BL, AL ; adjust second digit in chunk

LOOP input1

MOV operand, BX

; prompt user for second number

MOV DX, OFFSET prompt2

MOV AH, 09H

INT 21H

; clear EBX for holding input

XOR BX, BX

; initialise counter for loop

MOV CX, 04H

input2:

; shift content of EBX for next byte

SHL BX, 8

; accept first digit of chunk

MOV AH, 01H

INT 21H

; check if it is a valid digit

CMP AL, 39H

JBE letter3

; adjust letters to hex

SUB AL, 37H

letter3:

; adjust hex characters to hex from ascii

AND AL, 0FH ; mask contents in AL

SHL AL, 4 ; shift contents in AL

MOV BL, AL

; accept second digit of chunk

MOV AH, 01H

INT 21H

; check if it is a valid digit

CMP AL, 39H

JBE letter4

; adjust letters to hex

SUB AL, 37H

letter4:

; adjust hex characters to hex from ascii

AND AL, 0FH ; mask contents in AL

ADD BL, AL ; adjust second digit in chunk

LOOP input2

; copy first number to EAX

MOV AX, operand

; multiply with the second number in EBX

MUL BX

; copy results to memory

MOV resultms, DX

MOV resultls, AX

; print the result

MOV DX, OFFSET message

MOV AH, 09H

INT 21H

; copy the most significant part of product

MOV BX, resultms

; initalise the counter for loop

MOV CX, 04H

print1:

; rotate the contents of EBX

ROL BX, 8

; convert hex to ascii

MOV AL, BL

AND AL, 0F0H ; mask contents in AL

SHR AL, 4 ; shift msb in AL

ADD AL, 30H ; adjust hex to ascii

CMP AL, 39H ; check if it is a digit

JBE print1sub1

ADD AL, 07H ; adjust letters to ascii

print1sub1:

; print the first character of chunk

MOV DL, AL

MOV AH, 02H

INT 21H

; convert hex to ascii

MOV AL, BL

AND AL, 0FH ; mask contents in AL

ADD AL, 30H ; adjust hex to ascii

CMP AL, 39H ; check if it is a digit

JBE print1sub2

ADD AL, 07H ; adjust letters to ascii

print1sub2:

; print the second character of chunk

MOV DL, AL

MOV AH, 02H

INT 21H

LOOP print1 ; loop until all digits are printed

; copy the least significant part of product

MOV BX, resultls

; initialise the counter for loop

MOV CX, 04H

print2:

; rotate the contents of EBX

ROL BX, 8

MOV AL, BL

AND AL, 0F0H ; mask contents in AL

SHR AL, 4 ; shift msb in AL

ADD AL, 30H ; adjust hex to ascii

CMP AL, 39H

JBE print2sub1

ADD AL, 07H ; adjust letters to ascii

print2sub1:

; print the first character of chunk

MOV DL, AL

MOV AH, 02H

INT 21H

MOV AL, BL

AND AL, 0FH ; mask contents in AL

ADD AL, 30H ; adjust hex to ascii

CMP AL, 39H ; check if it is a digit

JBE print2sub2

ADD AL, 07H ; adjust letter to ascii

print2sub2:

; print the second character of chunk

MOV DL, AL

MOV AH, 02H

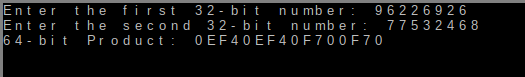
INT 21H

LOOP print2 ; loop until all digits are printed

.EXIT

END

**OUTPUT**



**2) Write a program for 32-bit BCD addition and subtraction**

**A) BCD ADDITION**

.MODEL SMALL

.DATA

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

MESS1 DB 10,13,"ENTER THE SECOND NUMBER:$"

MESS2 DB 10,13,"THE SUM IS:$"

A DD ?

B DD ?

C DD ?

COUNT DB 04h

.CODE

.STARTUP

LEA DX,MESS0

MOV AH,09

INT 21H

MOV BX,0

MOV CX,8

AGAIN:

MOV AH,01

INT 21H

CMP AL,'A'

JGE L5

SUB AL,30H

JMP L6

L5: SUB AL,37H

L6: SHL BX,4

ADD BL,AL

LOOP AGAIN

MOV A,BX

LEA DX,MESS1

MOV AH,09

INT 21H

MOV BX,0

MOV CX,8

AGAINS:

MOV AH,01

INT 21H

CMP AL,'A'

JGE L51

SUB AL,30H

JMP L61

L51: SUB AL,37H

L61: SHL BX,4

ADD BL,AL

LOOP AGAINS

MOV B,BX

MOV AX,WORD PTR A

MOV BX,WORD PTR B

ADD AL,BL

DAA

MOV BL,AL

ADC AH,BH

MOV AL,AH

DAA

MOV BH,AL

MOV WORD PTR C,BX

MOV AX,WORD PTR A+2

MOV BX,WORD PTR B+2

ADC AL,BL

DAA

MOV BL,AL

ADC AH,BH

MOV AL,AH

DAA

MOV BH,AL

MOV WORD PTR C+2,BX

LEA DX,MESS2

MOV AH,09

INT 21H

MOV BX,WORD PTR C+2

MOV DH,2

L1: MOV CH,04H

MOV CL,04H

L2: ROL BX,CL

MOV DL,BL

AND DL,0FH

CMP DL,09

JBE L4

ADD DL,07

L4: ADD DL,30H

MOV AH,02

INT 21H

DEC CH

JNZ L2

DEC DH

CMP DH,0

MOV BX,WORD PTR C

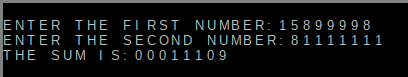
JNZ L1

MOV AH,4CH

INT 21H

END

**OUTPUT**

****

**B) BCD SUBTRACTION**

.MODEL SMALL

.DATA

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

MESS1 DB 10,13,"ENTER THE SECOND NUMBER:$"

MESS2 DB 10,13,"THE DIFFERENCE IS:$"

A DD ?

B DD ?

C DD ?

COUNT DB 04h

.CODE

.STARTUP

LEA DX,MESS0

MOV AH,09

INT 21H

MOV BX,0

MOV CX,8

AGAIN:

MOV AH,01

INT 21H

CMP AL,'A'

JGE L5

SUB AL,30H

JMP L6

L5: SUB AL,37H

L6: SHL BX,4

ADD BL,AL

LOOP AGAIN

MOV A,BX

LEA DX,MESS1

MOV AH,09

INT 21H

MOV BX,0

MOV CX,8

AGAINS:

MOV AH,01

INT 21H

CMP AL,'A'

JGE L51

SUB AL,30H

JMP L61

L51: SUB AL,37H

L61: SHL BX,4

ADD BL,AL

LOOP AGAINS

MOV B,BX

MOV AX,WORD PTR A

MOV BX,WORD PTR B

SUB AL,BL

DAS

MOV BL,AL

SBB AH,BH

MOV AL,AH

DAS

MOV BH,AL

MOV WORD PTR C,BX

MOV AX,WORD PTR A+2

MOV BX,WORD PTR B+2

SBB AL,BL

DAS

MOV BL,AL

SBB AH,BH

MOV AL,AH

DAS

MOV BH,AL

MOV WORD PTR C+2,BX

LEA DX,MESS2

MOV AH,09

INT 21H

MOV BX,WORD PTR C+2

MOV DH,2

L1: MOV CH,04H

MOV CL,04H

L2: ROL BX,CL

MOV DL,BL

AND DL,0FH

CMP DL,09

JBE L4

ADD DL,07

L4: ADD DL,30H

MOV AH,02

INT 21H

DEC CH

JNZ L2

DEC DH

CMP DH,0

MOV BX,WORD PTR C

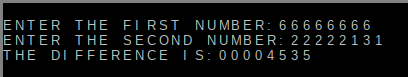
JNZ L1

MOV AH,4CH

INT 21H

END

**OUTPUT**

****

**3.Write a program for Linear search and binary search.**

**A)LINEAR SEARCH**

.model small

.stack

.data

ARRAY DB 10 DUP(?)

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

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

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 ?

number dw ?

.code

.startup

MOV DX,OFFSET MESS1

MOV AH,09

INT 21H

MOV AH,01

INT 21H

cmp al,39h

jbe abc

MOV DX,OFFSET ErrMess

MOV AH,09

INT 21H

jmp myexit

abc:

and al,0fh

mov ah,0

mov number,ax

MOV CX,AX

MOV DI,0

MYLOOP:

MOV DX,OFFSET MESS0

MOV AH,09

INT 21H

MOV AH,01

INT 21H

cmp al,39h

jbe abc2

MOV DX,OFFSET ErrMess

MOV AH,09

INT 21H

jmp myexit

abc2:

and al,0fh

MOV ARRAY[DI],AL

INC DI

LOOP MYLOOP

MOV DX,OFFSET MESS2

MOV AH,09

INT 21H

MOV AH,01

INT 21H

cmp al,39h

jbe abc3

MOV DX,OFFSET ErrMess

MOV AH,09

INT 21H

jmp myexit

abc3:

and al,0fh

MOV DAT,AL

mov ax,ds

mov es,ax

mov al,dat

CLD

mov cx,number

INC CX

mov DI, offset ARRAY

repne SCASB

CMP CX,0

JE NTFOUND

MOV DX,OFFSET MESS3

MOV AH,09

INT 21H

SUB NUMBER,CX ;FIND ELEMENT LOCATION

ADD NUMBER,30H

MOV DX,NUMBER

INC DX

MOV AH,02

INT 21H

JMP myexit

NTFOUND:

MOV DX,OFFSET MESS4

MOV AH,09

INT 21H

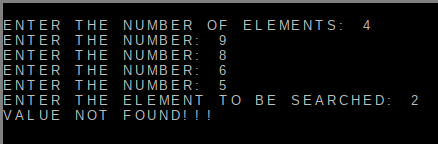
myexit:

MOV AH,4CH

INT 21H

END

**OUTPUT**

****

B) **BINARY SEARCH**

.MODEL SMALL

.DATA

ARR DW 0000H,1111H,2222H,3333H,4444H,5555H,6666H,7777H,8888H,9999H

LEN DW ($-ARR)/2

KEY EQU 1111H

MSG1 DB "KEY IS FOUND AT "

RES DB " POSITION",13,10," $"

MSG2 DB 'KEY NOT FOUND!!!.$'

.CODE

.STARTUP

MOV BX,00

MOV DX,LEN

MOV CX,KEY

AGAIN: CMP BX,DX

JA FAIL

MOV AX,BX

ADD AX,DX

SHR AX,1

MOV SI,AX

ADD SI,SI

CMP CX,ARR[SI]

JAE BIG

DEC AX

MOV DX,AX

JMP AGAIN

BIG: JE SUCCESS

INC AX

MOV BX,AX

JMP AGAIN

SUCCESS: ADD AL,01

ADD AL,'0'

MOV RES,AL

LEA DX,MSG1

JMP DISP

FAIL: LEA DX,MSG2

DISP: MOV AH,09H

INT 21H

MOV AH,4CH

INT 21H

END

**OUTPUT**

****

**4) Write a program to add and subtract two arrays**

**A) ADDITION OF ARRAYS**

.model small

.data

mat1 db 12h, 11h, 12h, 10h, 11h, 12h, 10h, 11h, 12h

mat2 db 13h, 02h, 02h, 02h, 02h, 02h, 02h, 02h, 02h

res3 dw 9 dup(?)

.code

mov ax, @data

mov ds, ax

mov cx, 09h

mov di, offset mat1

mov bx, offset mat2

mov si, offset res3

back : mov ah, 0

mov al, [di]

add al, [bx]

adc ah, 00

mov [si], ax

inc di

inc bx

inc si

inc si

loop back

mov si, offset res3

mov dh, 9

l10: mov ch, 04h

mov cl, 04h

mov bx, [si]

l2: rol bx, cl

mov dl, bl

and dl, 0fh

cmp dl, 09

jbe l4

add dl, 07

l4: add dl, 30h

mov ah, 02

int 21h

dec ch

jnz l2

mov dl, ' ' ;ye space h

int 21h

inc si

inc si

dec dh

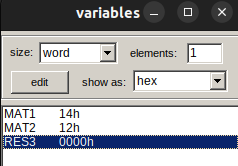
jnz l10

mov ah, 4ch

int 21h

end

**OUTPUT**

****

**B)SUBTRACTION OF ARRAYS**

.model small

.data

mat1 db 12h, 11h, 12h, 10h, 11h, 12h, 10h, 11h, 12h

mat2 db 13h, 02h, 02h, 02h, 02h, 02h, 02h, 02h, 02h

res3 dw 9 dup(?)

.code

mov ax, @data

mov ds, ax

mov cx, 09h

mov di, offset mat1

mov bx, offset mat2

mov si, offset res3

back : mov ah, 0

mov al, [di]

sub al, [bx]

adc ah, 00

mov [si], ax

inc di

inc bx

inc si

inc si

loop back

mov si, offset res3

mov dh, 9

l10: mov ch, 04h

mov cl, 04h

mov bx, [si]

l2: rol bx, cl

mov dl, bl

and dl, 0fh

cmp dl, 09

jbe l4

add dl, 07

l4: add dl, 30h

mov ah, 02

int 21h

dec ch

jnz l2

mov dl, ' ' ;ye space h

int 21h

inc si

inc si

dec dh

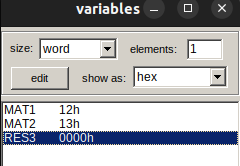
jnz l10

mov ah, 4ch

int 21h

end

**OUTPUT**

****

**5 ) Write a program for binary to ascii conversion**

.MODEL SMALL

.DATA

INPUT DB 10,13 , 'ENTER BINARY NO: $'

OUTPUT DB 10,13, 'THE ASCII CHARACTER IS:$'

ARR DB ?

.CODE

.STARTUP

MOV AH,09H

MOV DX,OFFSET INPUT

INT 21H

MOV BL, 00H

MOV CL,08H

INPUT1: MOV AH,01H

INT 21H

SUB AL,30H

SHL BL,1

ADD BL,AL

LOOP INPUT1

MOV AH,09H

LEA DX,OUTPUT

INT 21H

MOV AH,02H

MOV DL,BL

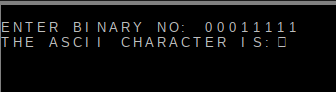
INT 21H

MOV AH,4CH

INT 21H

END

**OUTPUT**



**6) Write a program for ascii to binary conversion**

.model small

.data

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

ORG 1000H

MOV SI,1100H

MOV DI,1400H

CLD

MOV BL,20H

NEXT:LODSB

CMP AL,BL

JE EXIT

SUB AL,30H

CMP AL,0AH

JC STORE

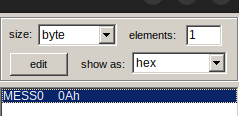
SUB AL,07H

STORE:STOSB

JMP NEXT

EXIT:HLT

**OUTPUT**

****