Visit: for notes [COAL Fall 2019](https://sites.google.com/view/coal-fall-2019)

Contents:

1. Square of a number

2. Factorial of a number

3. 8 bit multiplication, 16 bit result

4. 32 bit multiplication, 64 bit result in memory (and in ax, bx, cx, dx at the end)

5. Pairwise bit swap in ax (swap bit 1&2, 3&4 and so on)

6. Swapping nibbles in each byte of ax

7. Count 1’s in bx and complement ‘count’ lower bits in ax

8. Ch#4 Q8 (question statement shown with code)

9. Print a rectangle - complete code

10.Search for a byte in 1MB Memory

;SQUARE OF NUMBER

jmp start

num: dw 6

start:

mov ax, 0

mov bx, num

mov cx, [bx]

loop1:

add ax, [bx]

loop loop1

; FACTORIAL

jmp start

num: dw 8

start:

mov bx, [num]

mov dx, [num] ; result in dx

l1:

sub bx, 1

jz exit

mov cx, bx

mov ax, 0

l2:

add ax, dx

loop l2

mov dx, ax

jmp l1

exit:

; 8 BIT MULTIPLICATION ALGORITHM

[org 0x100]

jmp start

multiplicand: dw 13 ; 8 bit multiplicand (16 bit space)

multiplier: db 5 ; 8 bit multiplier

result: dw 0 ; 16 bit result

start:

mov cl, 8 ; initialize bit count to 8

mov bx, [multiplicand] ; load multiplicand in bx

mov dl, [multiplier] ; load multiplier in dl

checkbit: ; check if bit is 0, if yes then no need to add anything; skip

shr dl, 1 ; move right most bit in carry

jnc skip ; skip addition if bit is zero

add [result], bx ; accumulate result

skip:

shl bx, 1 ; shift multiplicand left

dec cl ; decrement bit count

jnz checkbit ; repeat if bits left

mov ax, [result]

mov ax, 0x4c00 ; terminate program

int 0x21

; 32 BIT MULTIPLICATION ALGORITHM

[org 0x100]

jmp start

multiplicand: dw 0, 0, 0xABCD, 0xD4E1 ; 32 bit multiplicand (64 bit space) upper,lower

multiplier: dd 0xAB5C32 ; 32 bit multiplier

result: dw 0, 0, 0, 0 ; 64 bit result upper, lower

;result should be 0x 73 005C B8FF 6FF2 verify on calculator

start:

mov cl, 64 ; initialize bit count to 64

mov dx, [multiplier] ; load multiplier in dx - lower half

mov bx, [multiplier+2]; - upper half

checkbit:

shr bx, 1 ; move right most bit in carry

rcr dx,1 ; these 2 lines are extended shift right

jnc skip ; skip addition if bit is zero

mov ax, [multiplicand+6] ;lowest quarter; Least significant byte

add [result+6], ax ; add lowest quarter

mov ax, [multiplicand+4]

adc [result+4],ax

mov ax, [multiplicand+2]

adc [result+2],ax

mov ax, [multiplicand]

adc [result],ax ; highest byte

skip:

shl word[multiplicand+6], 1 ; shift multiplicand left

rcl word[multiplicand+4],1 ; shifting multiplicand left

rcl word[multiplicand+2],1 ; shifting multiplicand left

rcl word[multiplicand],1 ; shifting multiplicand left

mov di,2

l2: ;completing extended left shift

add di,2

cmp di,8

jne l2

dec cl ; decrement bit count

jnz checkbit ; repeat if bits left

; for easy testing, stored result in 4 regesters

mov ax, [result] ; MSB

mov bx, [result+2]

mov cx, [result+4]

mov dx, [result+6] ; LSB

mov ax, 0x4c00 ; terminate program

int 0x21

; SWAPPING BITS IN AX (SWAP (1,2), SWAP (3,4) ...)

[org 0x100]

jmp start

start:

mov ax, 0101010101010101b; 0x5555 should become 1010 1010 1010 1010 = 0xAAAA

mov dx, ax

and dx, 0x5555 ;odd bit mask (bits on odd positions are 1, with LSB at position 1 not 0)

shl dx, 1 ; shift odd bit mask to left

mov bx, ax

and bx, 0xAAAA ; even bit mask (bits on even positions starting from 1 are set=1)

shr bx,1 ; shift even bit mask to right

or dx,bx

mov ax, dx

mov ax, 0x4c00 ; terminate program

int 0x21

; SWAPPING NIBBLES IN EACH BYTE OF AX

[org 0x100]

jmp start

start:

mov ax, 0xABCD ; result shld be 0xBADC

mov bx, ax

shr bl,4

shl al, 4

or al,bl

shr bh,4

shl ah, 4

or ah,bh

mov ax, 0x4c00 ; terminate program

int 0x21

; COUNT 1'S IN BX AND COMPLEMENTING COUNT LSB BITS IN AX

[org 0x100]

jmp start

start:

mov ax, 0xABCD ; AX= 1010 1011 1100 1101

mov bx, 0xABCD ; BX = 1010 1011 1100 1101 (no. of 1's: 10)

; so result shld be ax= 1010 10--00 0011 0010 = 0xA832

mov cx,0

l1:

cmp bx, 0

je nxt

shr bx,1

jnc l1 ; if cf = 1 increment cx, else loop again for next bit until bx=0

inc cx

jmp l1

nxt:

mov dx,1

l2:

xor ax, dx ; complement bit in ax with dx

shl dx,1 ; shift dx left to complement next bit in ax

loop l2 ; repeat cx times

mov ax, 0x4c00 ; terminate program

int 0x21

; CH#4 Q8

; AX contains a non-zero number. Count the number of ones in it and

; store the result back in AX. Repeat the process on the result (AX) until

; AX contains one. Calculate in BX the number of iterations it took to

; make AX one. For example BX should contain 2 in the following case:

; AX = 1100 0101 1010 0011 (input – 8 ones) = 0xC5A3

; AX = 0000 0000 0000 1000 (after first iteration – 1 one)= 0x0008

; AX = 0000 0000 0000 0001 (after second iteration – 1 one) STOP =0x0001

[org 0x100]

jmp start

start:

mov ax, 0xC5A3

mov bx,0

mov cx, 0

l1:

mov cx, 0

l2:

cmp ax, 0

je l2end

shr ax, 1

jnc l2

inc cx

jmp l2

l2end:

mov ax, cx

inc bx

cmp ax, 1

jne l1

mov ax, 0x4c00 ; terminate program

int 0x21

;PRINT A RECTANGLE

[org 0x100]

jmp start

TopLeft: dw 1, 0

BtmRight: dw 22, 80

; subroutine to clear the screen

clrscr: push es

push ax

push cx

push di

mov ax, 0xb800

mov es, ax ; point es to video base

xor di, di ; point di to top left column

mov ax, 0x0720 ; space char in normal attribute

mov cx, 2000 ; number of screen locations

cld ; auto increment mode

rep stosw ; clear the whole screen

pop di

pop cx

pop ax

pop es

ret

printRect:

push bp

mov bp, sp

push ax

push bx

push cx

push dx

push si

push di

push es

mov ax, 0xb800

mov es, ax

mov al, 80

mul byte[bp+10] ; top left y

add ax, [bp+8] ;top left x

shl ax, 1

mov di, ax

mov si,di;

mov cx, [bp+10] ;top left column

mov dx, [bp+8] ;top left row

mov cx, [bp+4] ; bottom right y(column) used for stos to repeat cx times

sub cx,dx ; change in y= y2-y1

mov bx,cx

sub bx,1

shl bx,1

mov ax,[bp+12] ;attribute

rep stosw ; print first row

mov cx, [bp+6] ;bottom right x (row)

dec cx ; dont print the last row in the loop

push si

nextChar:

add si,160

call sleep

mov word[es:si],ax

mov word[es:si+bx],ax

loop nextChar

pop di

mov cx,[bp+6]

l1:

add di,160

loop l1

mov cx, [bp+4] ; bottom right y used for stos to repeat cx times

sub cx,dx ; change in y= y2-y1

rep stosw ; print last row

pop es

pop di

pop si

pop dx

pop cx

pop bx

pop ax

pop bp

ret 10

sleep: push cx

mov cx, 0xFFFF

delay: loop delay

pop cx

ret

start:

call clrscr

mov ax, 0x4420 ;attribute

push ax

mov ax,[TopLeft] ;x

push ax

mov ax, [TopLeft+2] ;y

push ax

mov ax, [BtmRight] ;x

push ax

mov ax, [BtmRight+2] ;y

push ax

call printRect

mov ax, 0x4c00 ; terminate program

int 0x21

;SEARCH FOR A BYTE IN 1MB MEMORY

[org 0x0100]

jmp start

flag: db 0

start: mov al, 0x0F ;Byte to find

mov bx, 0x0000 ;Starting from segment 0x0000

l1: mov es, bx

mov cx, 0xFFFF

mov di, 0

repne scasb

je found

add bx, 1000

cmp bx, 0000

jz notFound

jnz l1

found: mov byte [flag], 1

jmp exit

notFound: jmp exit

exit:

mov dl, [flag]

mov ax, 0x4c00

int 21h