**Lab 11 (Portal 12)**

**Aamir Sohail**

**18F-0215**

**TASK 10.1**

include Irvine32.inc

.data

op1 BYTE 34h,12h,98h,74h,06h,0A4h,0B2h,0A2h

op2 BYTE 02h,45h,23h,00h,00h,87h,10h,80h

sum BYTE 9 dup(?)

.code

main proc

clc

call Extended\_Add

call Display\_Sum

exit

main endp

;

Extended\_Add proc

mov esi,0

mov eax,0

mov ebx ,0

mov ecx ,8

mov edx,0

l1:

mov al,[op1+esi]

mov bl,[op2+esi]

adc al,bl

mov [sum+esi],al

pushfd

add esi,type op1

popfd

loop l1

adc dl,0

mov [sum+esi],dl

ret

Extended\_Add endp

;

Display\_Sum proc

mov eax,0

mov ecx, LENGTHOF sum

mov esi ,offset sum

l1:

mov eax,[esi]

push eax

add esi,type sum

loop l1

mov ecx, LENGTHOF sum

mov esi ,offset sum

l2:

pop eax

mov [esi],eax

add esi,type sum

loop l2

mov esi,OFFSET sum

mov ecx,LENGTHOF sum

mov ebx,TYPE sum

call DumpMem

ret

Display\_Sum endp

end

;

**TASK 10.3**

include Irvine32.inc

.data

first BYTE "ENTER 1st NUMBER: ",0

second BYTE "ENTER 2nd NUMBER: ",0

prime BYTE "NUMBERS ARE RELATIVE PRIME! ",0

not\_prime BYTE "NUMBERS ARE NOT RELATIVE PRIME! ",0

gcd\_found BYTE "GCD IS: ",0

gcd word 0

.code

main proc

call DEC\_IN

call GCD\_AB

call DEC\_OUT

call crlf

call crlf

cmp gcd,1

je prime\_

mov edx, offset not\_prime

call writestring

jmp exit\_here

prime\_:

mov edx, offset prime

call writestring

exit\_here:

exit

main endp

;

GCD\_AB proc

mov eax,0

cmp bx,dx

je found

mov ax,bx ;greater value is now in ax

jmp find

find:

div dl ;divide ax by dl

cmp ah,0 ;compare remainder with 0

je found1

mov dh,ah

jmp found2

found2:

mov eax,0

mov al,dl

div dh

cmp ah,0 ;compare remainder with 0

je found\_final

mov dl,dh

mov dh,ah

jmp found2

found\_final:

mov eax,0

mov al,dh

mov gcd ,ax

jmp exithere

jmp exithere

found:

mov gcd,bx

jmp exithere

found1:

mov gcd ,dx

jmp exithere

exithere:

ret

GCD\_AB endp

;

DEC\_IN proc

mov eax,0

mov edx,offset first

call writestring

call readdec

call crlf

mov bx,ax

mov edx,offset second

call writestring

mov eax,0

call readdec

call crlf

mov edx,0

mov dx,ax

cmp bx,dx

jg ok

xchg bx,dx ;Greater number is stored in BX and smaller in DX

ok:

call crlf

ret

DEC\_IN endp

;

DEC\_OUT proc

mov edx,offset gcd\_found

call writestring

mov eax,0

mov ax,gcd

call writedec

ret

DEC\_OUT endp

end

;