**마이크로프로세서**

**프로젝트**



**15조 계산기 프로젝트**

**60142013 이창민**

**60171923 이현두**

기본적으로 실습시간에 배우고 해석하여 과제로 제출한 Calculator.asm을 기반으로 하여 계산기를 이용하여 PUTC 매크로나 다른 부분들을 응용하여 설계하였습니다. Calculator의 덧셈, 뺄셈, 곱셈 부분은 기본적인 형태 그대로 사용하였고, Calculator의 기본적인 나눗셈 형태는 소수점이하의 수들이 출력이 되지 않고 나머지가 생기면 "and something.... " 메시지가 출력이 되도록 되어있는데 이러한 소수점 부분 출력은 Emu8086의 examples중 하나인 float.asm을 분석하여 소수점을 출력하는 방식을 이해한 후 이를 응용하여 프로젝트에 적용하였습니다.

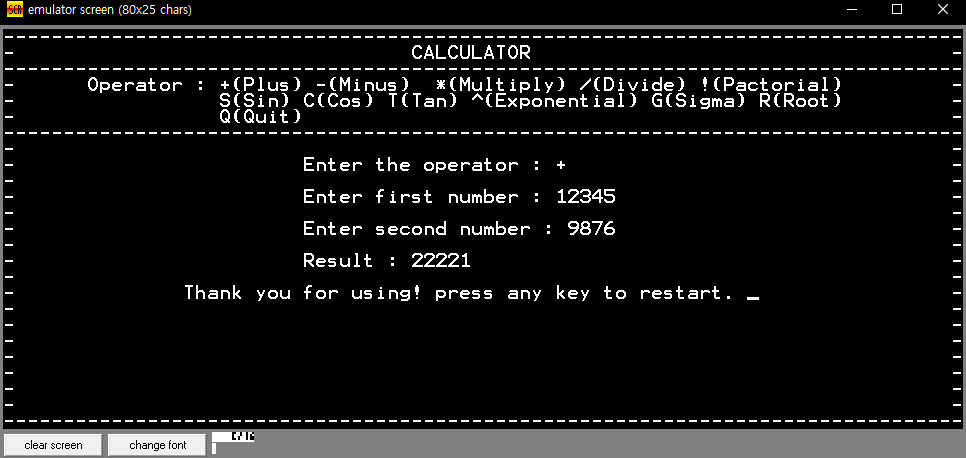
나머지 팩토리얼, 시그마, 지수함수 부분은 기본 사칙연산에 사용되는 NUM1, NUM2를 이용하고 계산에 이용되는 수식을 명령어로 표현하여 구현한 후 LOOP 명령어를 이용하여 반복해서 계산하도록 설계하였습니다.

루트의 경우에는 제곱근 값을 구하는 방법으로 수를 입력하면 해당 수부터 1씩 감소해가며 그 값이 원래의 값과 일치하는지 찾아내는 방식으로 설계를 하였는데, 해당 수부터 1씩 감소하는 방식과 0부터 1씩 증가하는 방식, 해당 수를 2로 나눈 다음에 1씩 감소시키는 방식을 각각 설계해보고 1씩 감소하는 방식은 숫자가 커질 경우 시간이 너무 오래 걸리고 2로 나눈 다음 1씩 감소하는 방식은 1숫자가 1일 때 오류가 생기므로 0부터 1씩 증가하는 방식으로 구현을 하였습니다.

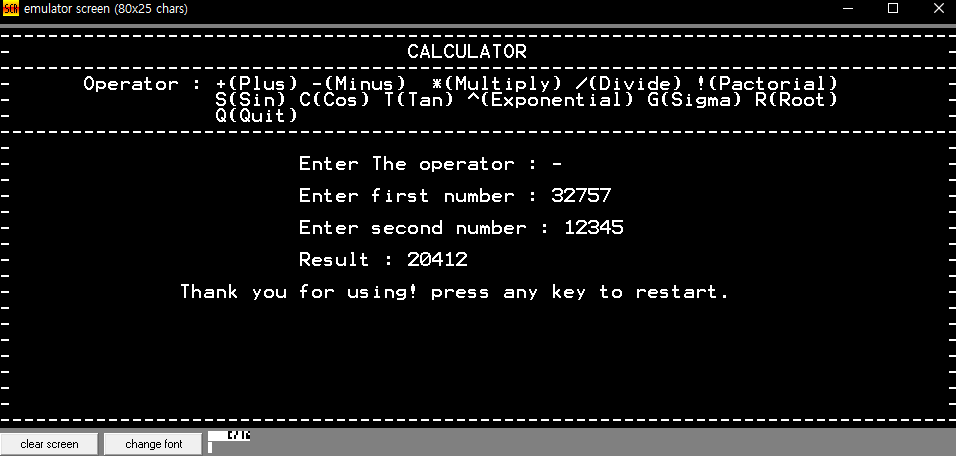
삼각함수를 계산하는 것을 구현하는 방식으로는 처음에는 float.asm을 이용하여 소수점을 구하는 방식을 구현하였으니 이것을 응용하여 삼각함수 계산기를 설계하자고 생각을 하였는데 다른 계산방법이 필요하다는 것을 인식하여 다른 방법을 찾던 중, 교수님이 12월 13일에 XLAT를 설명하다가 삼각함수부분에 이를 사용할 수 있다는 언급을 듣고 이를 응용하여 삼각함수 계산기를 구현하였습니다. 삼각함수는 0부터 90도까지만 구할 수 있다면 나머지 부분은 반복되는 부분이므로 쉽게 구할 수 있어 0부터 90도까지만 정상적인 값을 출력할 수 있도록 해두었고 숫자를 입력할 시, 0~90 사이의 값을 입력하도록 문구를 추가하였습니다.

계산기의 형태가 단순하여 모양을 만들기 위해 작업을 하던 중, INT 10H의 기능 중에 AH를 13H로 설정하면 문자열 출력이 가능하다는 것을 알게되어 계산기의 거의 모든 메시지, 문자열 출력부분에 해당 기능을 사용하여 응용하기 위해 printC라는 매크로를 만들어, 커서위치, 배경색, 문자색, 출력할 문자열을 설정할 수 있도록 매크로를 설계하였습니다. 메시지 출력 후 커서를 위쪽으로 옮겨야 하는 경우가 생겨 이것 또한 set cursor 의 약어로 setc라는 매크로를 만들어 사용하였습니다.

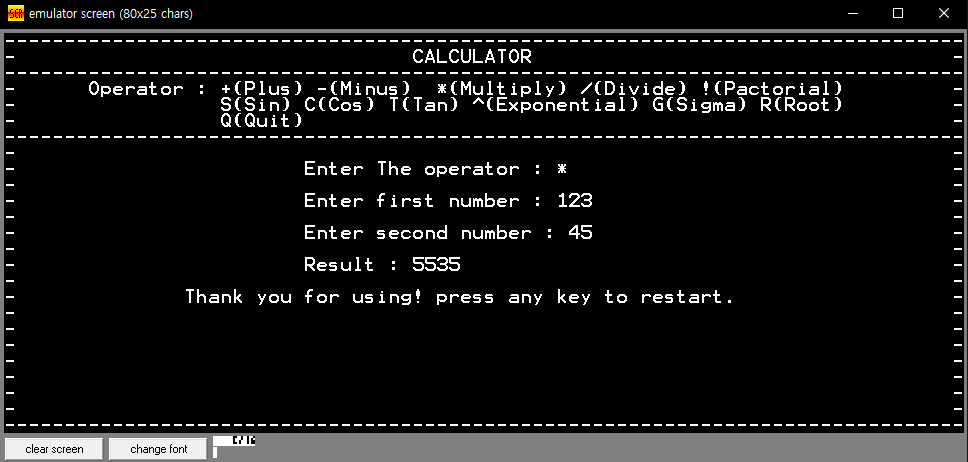
이 계산기를 이용하여 장문의 식을 이용하여 계산하는 방법은 사용할 수 없지만, 장문의 식을 부분으로 나눠서 계산이 복잡한 일부분만 계산하고 이를 다시 다음 연산자의 피연산자로 입력하여 계산을 이어 나가는 방식으로 장문의 식을 계산할 수 있을 것이라 생각합니다.

덧셈

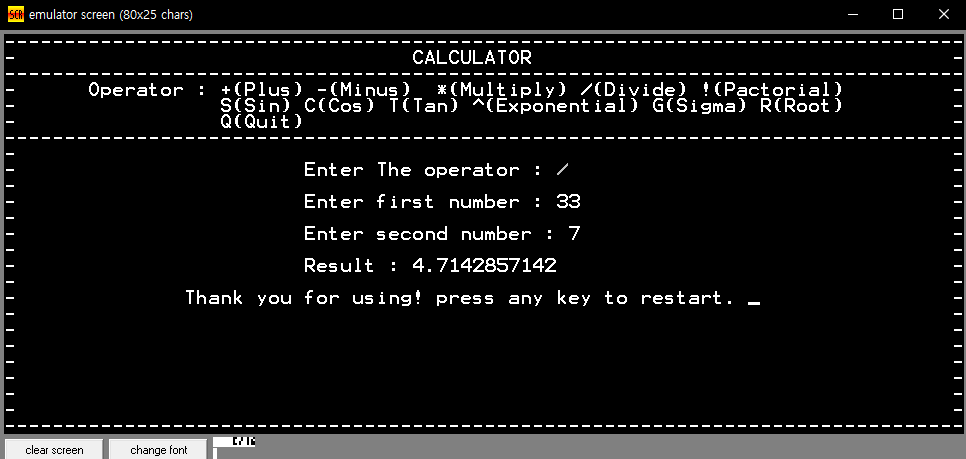
뺄셈



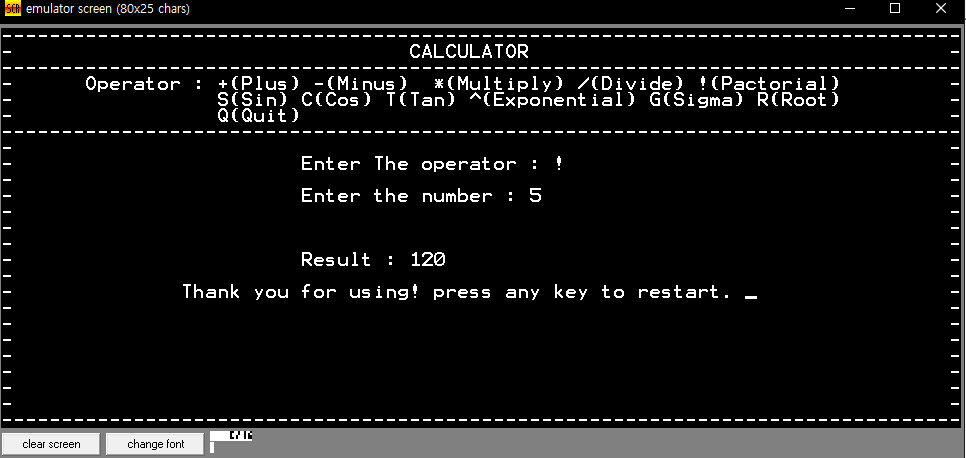
곱셈



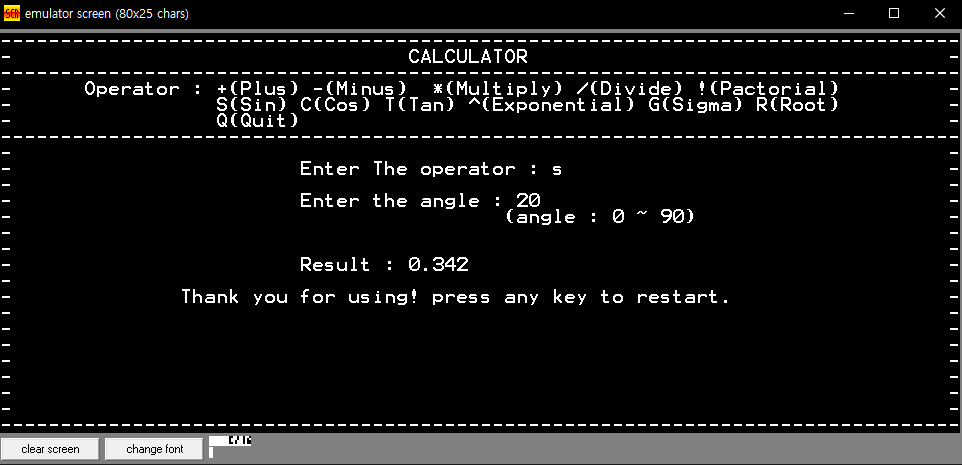
나눗셈



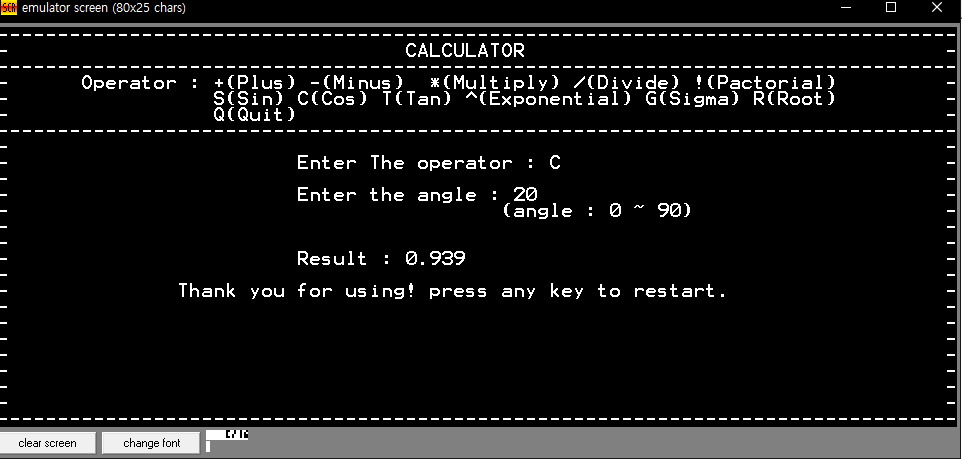
팩토리얼



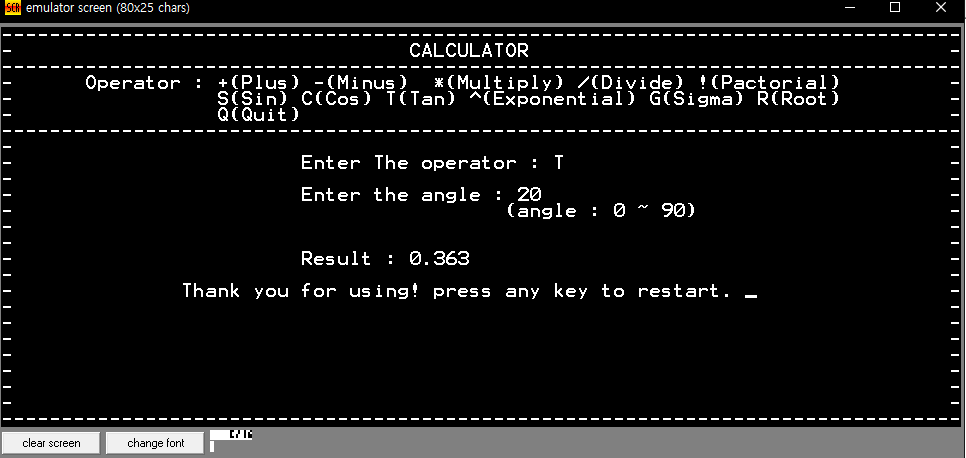
사인



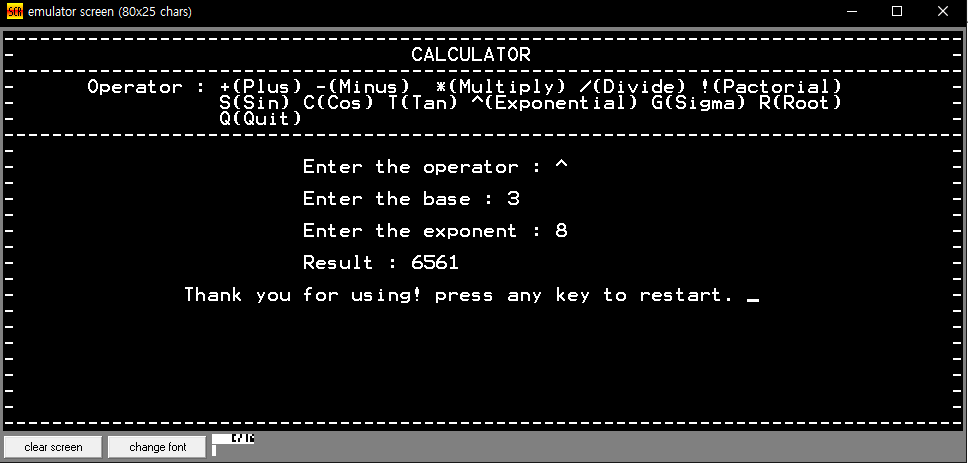
코사인



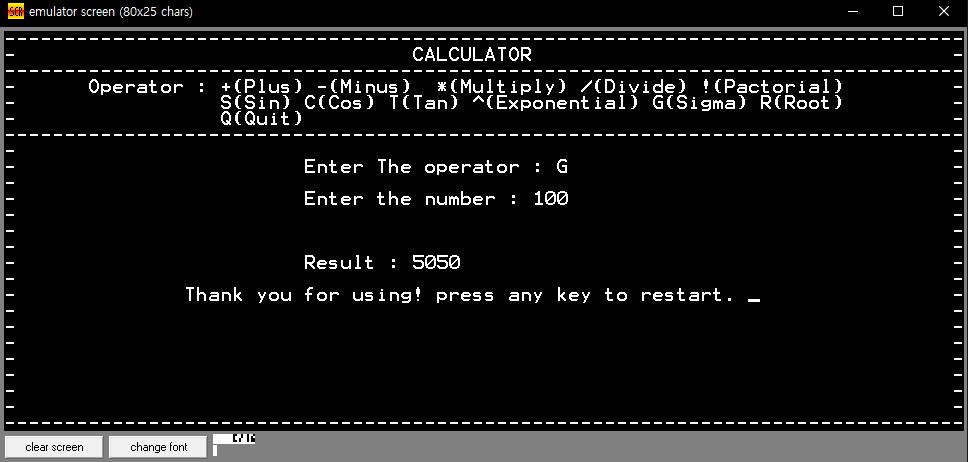
탄젠트



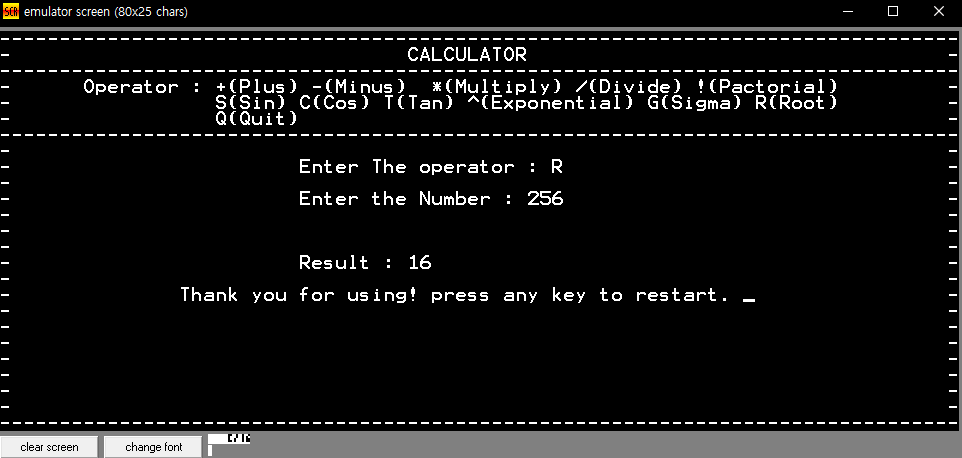
지수함수



시그마



루트



Emu8086 전체 코드

코드 설명은 주석으로 표시

;현재 커서 위치에 AL에 입력된 아스키코드의 문자를 출력하는 매크로

PUTC MACRO char

PUSH AX ;입력된 값을 스택에 저장

MOV AL, char;AL에 char의 값을 저장

MOV AH, 0Eh ;텔레타이프 출력

INT 10h

POP AX

ENDM

printC macro a,b,c,d

LOCAL e,e\_end,f

pusha

mov dx, cs

mov es, dx

mov ah, 13h

mov al, 1

mov bh, 0

mov bl, c

mov cx, offset e\_end - offset e

mov dl, b

mov dh, a

mov bp, offset e

int 10h

popa

jmp f

e DB d

e\_end DB 0

f:

endm

setc macro a,b

push bx

mov ah,2

mov bh,0

mov dl,b

mov dh,a

int 10h

pop bx

endm

;키보드 입력 매크로

INPUT MACRO

MOV AH,1

INT 21H

ENDM

;키보드로 받은 입력을 스크린에 출력하는 매크로

OUTPUT MACRO

MOV AH,2

INT 21H

ENDM

ORG 100H

JMP START

MSG5 DB "Thank you for using the calculator! press any key... ", 0Dh,0Ah, '$'

msg7 db 0Dh,0Ah,'\* Enter the angle: $'

SMTH DB "and something.... $"

sosu1 db ".0$"

sosu db ".$"

sin dw 0, 17, 34, 52, 69, 87, 104, 121, 139, 154, 173, 190, 207, 224, 241, 258, 275, 292, 309, 325, 342, 358, 374, 390, 406, 422, 438, 453, 469, 484, 5, 515, 529, 544, 559, 573, 587, 601, 615, 629, 642, 656, 669, 681, 694, 707, 719, 731, 743, 754, 766, 777, 788, 798, 809, 819, 829, 838, 848, 857, 866, 874, 882, 891, 898, 906, 913, 920, 927, 933, 939, 945, 951, 956, 961, 965, 970, 974, 978, 981, 984, 987, 990, 992, 994, 996, 997, 998, 999, 999, 1000

cos dw 1, 999, 999, 998, 997, 996, 994, 992, 990, 987, 984, 981, 978, 974, 970, 965, 961, 956, 951, 945, 939, 933, 927, 920, 913, 906, 898, 891, 882, 874, 866, 857, 848, 838, 829, 819, 809, 798, 788, 777, 766, 754, 743, 731, 719, 707, 694, 681, 669, 656, 642, 629, 615, 601, 587, 573, 559, 544, 529, 515, 5, 484, 469, 453, 438, 422, 406, 390, 374, 358, 342, 325, 309, 292, 275, 258, 241, 224, 207, 190, 173, 154, 139, 121, 104, 87, 69, 52, 34, 17, 0

tan dw 0,17,34,52,69,87,105,122,140,158,176,194,212,230,249,267,286,305,324,344,363,383,404,424,445,466,487,509,531,554,577,600,624,649,674,700,726,753,781,809,839,869,900,932,965,1000,1035,1072,1110,1150,1191,1234,1279,1327,1376,1428,1482,1539,1600,1664,1732,1804,1880,1962,2050,2144,2246,2355,2475,2605,2747,2904,3077,3270,3487,3732,4010,4331,4704,5144,5671,6313,7115,8144,9514,11430,14300,19081,28636,57289

OPR DB 0

NUM1 DW 0

NUM2 DW 0

result dw 0

result1 dw 0

sqrt dw 0

two dw 2

ten DW 10

hun dw 100

thousand dw 1000

precision = 10

;계산기 시작

START:

PRINTC 0,0,0000\_1111b, "--------------------------------------------------------------------------------"

PRINTC 1,0,0000\_1111b, "- CALCULATOR -"

PRINTC 2,0,0000\_1111b, "--------------------------------------------------------------------------------"

PRINTC 3,0,0000\_1111b, "- Operator : +(Plus) -(Minus) \*(Multiply) /(Divide) !(Pactorial) -"

PRINTC 4,0,0000\_1111b, "- S(Sin) C(Cos) T(Tan) ^(Exponential) G(Sigma) R(Root) -"

PRINTC 5,0,0000\_1111b, "- Q(Quit) -"

PRINTC 6,0,0000\_1111b, "--------------------------------------------------------------------------------"

PRINTC 7,0,0000\_1111b, "- -"

PRINTC 8,0,0000\_1111b, "- -"

PRINTC 9,0,0000\_1111b, "- -"

PRINTC 10,0,0000\_1111b,"- -"

PRINTC 11,0,0000\_1111b,"- -"

PRINTC 12,0,0000\_1111b,"- -"

PRINTC 13,0,0000\_1111b,"- -"

PRINTC 14,0,0000\_1111b,"- -"

PRINTC 15,0,0000\_1111b,"- -"

PRINTC 16,0,0000\_1111b,"- -"

PRINTC 17,0,0000\_1111b,"- -"

PRINTC 18,0,0000\_1111b,"- -"

PRINTC 19,0,0000\_1111b,"- -"

PRINTC 20,0,0000\_1111b,"- -"

PRINTC 21,0,0000\_1111b,"- -"

PRINTC 22,0,0000\_1111b,"- -"

PRINTC 23,0,0000\_1111b,"- -"

PRINTC 24,0,0000\_1111b,"--------------------------------------------------------------------------------"

RESTART:

PRINTC 8,25,0000\_1111b,"Enter The operator : "

INPUT\_OPR:

INPUT

MOV OPR,AL

cmp opr, 'q'

je quit

cmp opr, 'r'

je do\_root

cmp opr, 's'

je do\_sin

cmp opr, 'c'

je do\_cos

cmp opr, 't'

je do\_tan

cmp opr, 'Q'

je quit

cmp opr, 'R'

je do\_root

cmp opr, 'S'

je do\_sin

cmp opr, 'C'

je do\_cos

cmp opr, 'T'

je do\_tan

cmp opr, '!'

je jmp\_to\_calc

cmp opr, '^'

je jmp\_to\_calc

cmp opr, 'g'

je jmp\_to\_calc

cmp opr, 'G'

je jmp\_to\_calc

cmp opr, '\*'

jb wrong\_opr

cmp opr, '/'

ja wrong\_opr

PRINTC 10,25,0000\_1111B,"Enter first number : "

;첫번째 숫자 입력

CALL SCAN\_NUM

MOV NUM1,CX

PRINTC 12,25,0000\_1111B, "Enter second number : "

call scan\_num

mov num2,cx

printc 14,25,0000\_1111b, "Result : "

cmp opr, '+'

je do\_plus

cmp opr, '-'

je do\_minus

cmp opr, '\*'

je do\_mult

cmp opr, '/'

je do\_div

jmp\_to\_calc:

cmp opr, '!'

je do\_factorial

cmp opr, '^'

je do\_exponential

cmp opr, 'g'

je do\_sigma

cmp opr, 'G'

je do\_sigma

wrong\_opr:

printc 12,25,0000\_1111b,"Wrong operator!"

INPUT

PRINTC 12,0,0000\_1111b, "- -"

PRINTC 8,25,0000\_1111b,"Enter the operator : "

PRINTC 8,25,0000\_1111b,"Enter the operator : "

JMP INPUT\_OPR

exit:

prinTC 16,15,0000\_1111b,"Thank you for using! press any key to restart. "

input

PRINTC 8,0,0000\_1111b, "- -"

PRINTC 10,0,0000\_1111b, "- -"

PRINTC 11,0,0000\_1111b, "- -"

PRINTC 12,0,0000\_1111b,"- -"

PRINTC 14,0,0000\_1111b,"- -"

PRINTC 15,0,0000\_1111b,"- -"

PRINTC 16,0,0000\_1111b,"- -"

jmp REstart

quit:

prinTC 16,15,0000\_1111b,"Thank you for using! press any key to quit. "

input

mov ah,4ch

int 21h

;;;;;;;;;;;;;;;;;;;;;;

do\_sigma:

PRINTC 10,25,0000\_1111B,"Enter the number : "

CAll scan\_num

MOV AX,0

cal\_sigma:

ADD AX,CX

LOOP CAL\_SIGma

printc 14,25,0000\_1111b, "Result : "

call print\_num

jmp exit

;;;;;;;;;;;;;;;;;;;;;;

do\_exponential:

PRINTC 10,25,0000\_1111B,"Enter the base : "

CALL SCAN\_NUM

MOV NUM1,CX

PRINTC 12,25,0000\_1111B,"Enter the exponent : "

CALL SCAN\_NUM

mov ax,1

MOV BX,NUM1

cal\_exponential:

mul Bx

loop cal\_exponential

printc 14,25,0000\_1111b, "Result : "

call print\_num

jmp exit

;;;;;;;;;;;;;;;;;;;;;;

do\_factorial:

PRINTC 10,25,0000\_1111B,"Enter the number : "

CALL SCAN\_NUM

mov ax,1

cal\_factorial:

mul cx

loop cal\_factorial

printc 14,25,0000\_1111b, "Result : "

call print\_num

jmp exit

;;;;;;;;;;;;;;;;;;;;;;

do\_plus:

mov ax, num1

add ax, num2

call print\_num

jmp exit

;;;;;;;;;;;;;;;;;;;;;;;

do\_minus:

mov ax, num1

sub ax, num2

call print\_num

jmp exit

;;;;;;;;;;;;;;;;;;;;;;;

do\_mult:

mov ax, num1

imul num2

call print\_num

jmp exit

;;;;;;;;;;;;;;;;;;;

do\_div:

mov dx, 0

mov ax, num1

mov bx, num2

idiv bx

call print\_float

print\_float proc near

push cx

push dx

cmp bx, 0 ;a가 음수인지 확인

jns div\_not\_signed ;양수이면 점프

neg dx

div\_not\_signed:

cmp ax, 0 ; x의 정수부분이 0인지 확인

jne checked ; ax!=0 이면 점프

cmp dx, 0

jns checked ; ax=0 and dx>=0 이면 점프

push dx

mov dl, '-'

call write\_char ; '-' 출력

pop dx

checked:

; print whole part:

call print\_num-d

; 나머지가 0이면 출력할 필요가 없으므로 확인

cmp dx, 0

je done

push dx

;소수이후 숫자 출력 이전 '.' 출력

mov dl, '.'

call write\_char

pop dx

;소수점이후 숫자 출력

mov cx, precision ;소수점 아래 출력 자리수 설정

call print\_fraction

done:

pop dx

pop cx

ret

print\_float endp

print\_fraction proc near

push ax

push dx

next\_fraction:

;모든 소수점이 출력되었는지 확인

cmp cx, 0

jz end\_rem

dec cx ; 소수점자리수 카운터 1감소

;나머지가 0인지 확인, 0이면 출력종료

cmp dx, 0

je end\_rem

mov ax, dx ;나머지를 AX에 저장

xor dx, dx ;DX 초기화

cmp ax, 0 ;AX가 0보다 큰지 확인

jns not\_sig1 ;0보다 크면 (SF=0)이면 점프

not dx ;AX가 0보다작으면 DX:AX의 앞부분인 DX를 1로 채우기위해 보수를 취함

not\_sig1:

imul ten ; dx:ax = ax \* 10

idiv bx ; ax = dx:ax / bx (dx - remainder)

push dx ; 나머지를 스택에 저장.

mov dx, ax

cmp dx, 0

jns not\_sig2

neg dx

not\_sig2:

add dl, 30h ; 아스키코드로 변환

call write\_char ; DL 출력

pop dx

;다음소수점 출력을위해 점프

jmp next\_fraction

end\_rem:

pop dx

pop ax

jmp exit

print\_fraction endp

write\_char proc near

push ax

mov ah, 02h

int 21h

pop ax

ret

write\_char endp

print\_num-d proc near

push dx

push ax

;AX가 0인지 비교후 0이 아니면 점프

cmp ax, 0

jnz not\_zero-d

mov dl, '0'

call write\_char

jmp printed-d

not\_zero-d:

; AX가 음수인지 확인하고 음수이면 '-'를 따로 출력하기위해 다시 보수를 취함

cmp ax, 0

jns positive-d ;양수이면 점프

neg ax ;음수이면 보수를 취함

;음수인 경우 양수로만들고 앞에 '-'를 출력함

mov dl, '-'

call write\_char

positive-d:

call print\_numx

printed-d:

pop ax

pop dx

ret

print\_num-d endp

print\_numx proc near

push bx

push cx

push dx

; 실제 데이터가 출력되기전에 0이 출력되지 않게하려고 CX를 1로 설정

mov cx, 1

mov bx, 10000 ; 2710h -최대숫자가 32767이므로 만의단위부터 출력하기위해 BX를 10000부터

;10씩 나눠서 10000,1000,100,10,1 씩 자리수를 표현하는데 사용

; AX가 0인지 확인 0이면 점프

cmp ax, 0

jz end\_show

begin\_print-d:

; 분모가 0인지 확인하고 0이되면 점프

cmp bx,0

jz end\_show

; 실질 데이터 출력전에 0이 나타나지 않게 하기위해 사용 CX= 0이면 점프

cmp cx, 0

je calc-d

;AX<BX이면 나누는 몫은 0이 될것이므로 AX<BX이면 점프

cmp ax, bx

jb skip-d

calc-d:

xor cx, cx ; set flag.

xor dx, dx

div bx ; ax = dx:ax / bx (dx=remainder).

; print last digit

; ah is always zero, so it's ignored

push dx

mov dl, al

add dl, 30h ; convert to ascii code.

call write\_char

pop dx

mov ax, dx ; get remainder from last div.

skip-d:

; 아래자리수 출력을 위해 BX를 10으로 나눔

push ax

xor dx, dx

mov ax, bx

div ten ; ax = dx:ax / 10 (dx=remainder).

mov bx, ax

pop ax

jmp begin\_print-d

end\_show:

pop dx

pop cx

pop bx

ret

print\_numx endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

do\_root:

printc 10,25,0000\_1111b,"Enter the Number : "

call scan\_num

mov num1,cx

mov bx,0

root:

mov dx,0

mov cx,bx ; 이전의 값 저장

inc bx

mov ax,bx

mul bx

cmp ax,num1

jg skip\_root

mov AX,NUM1

div bx

cmp ax,bx

jnz root

mov sqrt,bx

push ax

push dx

printc 14,25,0000\_1111b, "Result : "

pop dx

pop ax

call print\_num

cmp dx,0

jne root\_smth

jmp exit

root\_smth proc near

printc 15,25,0000\_1111b,"and something...."

jmp exit

root\_smth endp

skip\_root:

printc 14,25,0000\_1111b, "Result : "

mov ax,cx

call print\_num

printc 15,8,0000\_1111b,"and something...."

jmp exit

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

do\_sin:

printc 10,25,0000\_1111b,"Enter the angle : "

printc 11,25,0000\_1111b," (angle : 0 ~ 90) "

setc 10,43

call scan\_num

mov num1,cx

mov dx,0

mov ax,num1

mul TWO

add ax,offset sin ;num1번째 sin값을 찾기위해 ax에 더함

mov bx, ax ;bx로 이동

mov ax, [bx] ; num1번째 sin값을 ax에 저장

idiv thousand

cmp dx,10

jnc app ;dx가 0이 아니면 점프

mov result,ax

cmp dx,5

jz sosu0

printc 14,25,0000\_1111b, "Result : "

mov ax, result

call print\_num

jmp exit

app:

mov result1,dx

mov dx, result1

cmp dx,100 ;

jnc sosu0

mov result,ax

mov result1,dx

printc 14,25,0000\_1111b, "Result : "

mov ax, result

call print\_num

lea dx, sosu1

mov ah, 09h

int 21h

mov ax, result1

call print\_num

jmp exit

sosu0:

mov result1,dx

sub ax,ax

mov ax, result1

sub dx, dx

div thousand

mov result,ax

mov result1,dx

printc 14,25,0000\_1111b, "Result : "

mov ax, result

call print\_num

lea dx, sosu

mov ah, 09h

int 21h

mov ax, result1

call print\_num

jmp exit

;;;;;;;;;;;;코사인''''''''''''

do\_cos:

printc 10,25,0000\_1111b,"Enter the angle : "

printc 11,25,0000\_1111b," (angle : 0 ~ 90) "

setc 10,43

call scan\_num

mov num1,cx

mov dx,0

mov ax,num1

mul two ; 코사인 값들의 데이터 크기가 WORD이

add ax,offset cos

mov bx, ax

mov ax, [bx]

idiv thousand

cmp dx,10

jnc app1 ;DL이 16보다 크면

mov result,dx ;16보다 작으면 나머지를 result에 저장

cmp dx,5

jz sosu2 ;dl이 5면 점프

printc 14,25,0000\_1111b, "Result : "

mov ax, result

call print\_num

jmp exit

app1:

mov result1,dx ;값저장

mov dx, result1

cmp dx,100

jnc sosu2 ;DL이 94보다 크면 점프

mov result,ax

mov result1,dx

printc 14,25,0000\_1111b, "Result : "

mov ax, result

call print\_num

lea dx, sosu1

mov ah, 09h

int 21h

mov ax, result1

call print\_num

jmp exit

sosu2:

mov result1,dx

sub ax,ax

mov ax, result1

sub dx, dx

div thousand

mov result,ax

mov result1,dx

printc 14,25,0000\_1111b, "Result : "

mov ax, result

call print\_num

lea dx, sosu

mov ah, 09h

int 21h

mov ax, result1

call print\_num

jmp exit

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

do\_tan:

printc 10,25,0000\_1111b,"Enter the angle : "

printc 11,25,0000\_1111b," (angle : 0 ~ 90) "

setc 10,43

call scan\_num

mov num1,cx

mov dx,0

mov ax,num1

cmp ax,90

je tan\_infinite

mul two

add ax,offset tan

mov bx, ax

mov ax, [bx]

idiv thousand

mov result,ax

cmp dx,0 ;

jnz app3

printc 14,25,0000\_1111b, "Result : "

mov ax, result

call print\_num

jmp exit

app3:

mov result1,dx

mov dx, result1

cmp dx,100 ;

jnc sosu4

mov result1,dx

printc 14,25,0000\_1111b, "Result : "

mov ax, result

call print\_num

lea dx, sosu1

mov ah, 09h

int 21h

mov ax, result1

call print\_num

jmp exit

sosu4:

mov result1,dx

mov ax, result1

sub dx, dx

div thousand

mov result1,dx

printc 14,25,0000\_1111b, "Result : "

mov ax, result

call print\_num

lea dx, sosu

mov ah, 09h

int 21h

mov ax, result1

call print\_num

jmp exit

tan\_infinite:

printc 14,25,0000\_1111b, "Result : Infinity"

jmp exit

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

SCAN\_NUM PROC NEAR

PUSH DX

PUSH AX

PUSH SI

MOV CX, 0

MOV CS:make\_minus, 0 ;부호 초기화

next\_digit:

input

; -가 입력되었는지 확인

CMP AL, '-'

JE set\_minus ;-가 입력되면 점프

; ENTER 키가 눌렸는지 확인

CMP AL, 0Dh ; AL에 엔터가 입력되었는지 비교

JNE not\_cr ; 엔터가 아니면 점프

JMP stop\_input ;엔터가 입력되었으면 입력을 멈춤

not\_cr:

CMP AL, 8 ; 'BACKSPACE' 가 눌렸는지 확인

JNE backspace\_checked ; 'BACKSPACE' 가 아니면 점프

MOV DX, 0 ; 마지막 자리의 숫자를 지우고 자리수를 하나 내리기 위해

MOV AX, CX ; 입력된 숫자를 10으로 나눔

DIV CS:ten ; AX = DX:AX / 10 (DX는 마지막으로 입력된 숫자이므로 무의미).

MOV CX, AX

PUTC ' ' ; 마지막으로 입력된 숫자를 지움

PUTC 8 ; 커서를 한칸 앞으로 움직임

JMP next\_digit ; 다음 숫자 입력으로 점프

backspace\_checked:

; allow only digits: 입력된 값이 숫자인지 확인, 숫자 = 30~39

CMP AL, '0' ;0보다 큰지 확인

JAE ok\_AE\_0 ;0보다 크면 점프

JMP remove\_not\_digit ;0보다 작으면 remove\_not\_digit으로 점프

ok\_AE\_0:

CMP AL, '9' ;9보다 작은지 확인

JBE ok\_digit ;9보다 작으면 점프

remove\_not\_digit:

PUTC 8 ; 백스페이스를 출력, 커서를 한 칸 앞으로 옮김

PUTC ' ' ; 커서에 공백을 출력하여 입력된 기존의 마지막 문자를 지움

PUTC 8 ; 백스페이스를 출력하여 커서를 다시 한칸 앞으로 옮겨서 원래

JMP next\_digit ; 다음 숫자 입력을 위해 점프

ok\_digit:

; CX에 저장된 기존의 숫자를 새로 입력된 수를 위해 자리수를 올림

PUSH AX ;스택에 AX 저장

MOV AX, CX

MUL CS:ten ; DX:AX = AX\*10 , 새로 입력된 수를 위해 기존에 입력된 수를 10배해서 자릿수를 높임

MOV CX, AX ;기본의 값에 자릿수를 올리고 다시 CX에 저장

POP AX ;AX를 다시 불러옴

; 숫자가 16비트 내에서 표현되는지 확인

; AX를 10배했을 때 16비트를 초과하는 수는 DX에 저장

CMP DX, 0 ; 16비트에서 Carry가 발생하는지 확인

JNE too\_big ; 숫자가 너무 커 Carry가 발생하면 점프

; ; ASCII 코드를 숫자로 변환

SUB AL, 30h

; add AL to CX: 입력된 값을 기존의 값에 더함

MOV AH, 0

MOV DX, CX ; 결과가 16bit를 초과할 경우를 대비해 CX를 DX에 저장함

ADD CX, AX

JC too\_big2 ; 숫자가 16bit 초과시 점프

JMP next\_digit ; 다음 숫자 입력을 위해 점프

;음수 설정

set\_minus:

MOV CS:make\_minus, 1 ; 해당 주소에 1을 저장

JMP next\_digit ;음수 설정 후 다음 숫자입력으로 점프

too\_big2:

MOV CX, DX ; 덧셈을 실행하기전의 수로 CX를 되돌림

MOV DX, 0 ; 덧셈하기전 DX또한 0이었으므로 0으로 되돌림

too\_big:

MOV AX, CX

DIV CS:ten ;DX:AX를 10으로 나누고 AX에 저장 reverse last DX:AX = AX\*10, make AX = DX:AX / 10

MOV CX, AX

PUTC 8 ; 백스페이스를 출력, 커서를 한 칸 앞으로 옮김

PUTC ' ' ; 커서에 공백을 출력하여 입력된 기존의 마지막 숫자를 지움

PUTC 8 ; 백스페이스를 출력하여 커서를 다시 한칸 앞으로 옮겨서 원래의 자리로 되돌림

JMP next\_digit ; 다음 입력(엔터/백스페이스)를 위해 점프

stop\_input:

; 부호를 확인

CMP CS:make\_minus, 0 ;음수가 아닌지 확인

JE not\_minus ;음수가 아니면 점프

NEG CX ;음수이면 입력된 첫번째 수(CX)를 보수를 취함

not\_minus:

;스택에 저장해놓은 값들을 불러옴

POP SI

POP AX

POP DX

RET

make\_minus DB 0 ; 숫자에 -가 입력되면 이 변수를 부호처럼 사용

SCAN\_NUM ENDP

; AX가 음수인 경우 -출력을 위한 프로시저

; -를 출력한 이후 PRINT\_NUM\_UNS로 점프함

PRINT\_NUM PROC NEAR

PUSH DX

PUSH AX

CMP AX, 0 ; 결과가 0 인지 비교

JNZ not\_zero ;0이 아니면 점프

PUTC '0'

JMP printed

not\_zero:

; AX의 부호를 확인

CMP AX, 0 ;AX이 0보다큰지 비교

JNS positive ;SF가 0이면 점프, 0보다 크면 점프

NEG AX ; 결과에 -가 따로 출력 되므로 결과값을 양수로 되돌림

PUTC '-' ; - 출력

positive:

CALL PRINT\_NUM\_UNS ;양수이면 PRINT\_NUM\_UNS으로 점프하고 다음 수행될 명령어를 스택에 저장

printed:

POP AX

POP DX

RET

PRINT\_NUM ENDP

; 무부호에서 실행되는 프로시저

; number in AX (not just a single digit)

; 0부터 65535 (FFFF)까지의 값만 허용됨

PRINT\_NUM\_UNS PROC NEAR

PUSH AX ;레지스터들을 스택에 저장

PUSH BX

PUSH CX

PUSH DX

; 결과가 123인 경우 00123을 출력하지않고 숫자앞의 0을 제외한 123만 출력하기위해 사용

MOV CX, 1

; 10000으로 나눈후의 몫은 항상 9보다 작거나 같음

MOV BX, 10000 ; 2710h, 더한결과의 최대값이 65535이기 때문에 최대 자리수가 만의 단위이므로

;해당 자리수 표현을위해 BX를 10000부터 시작해서 AX를 나누고 BX를 10으로 나눠 1000의 자리수를 표현하고 반복

; AX가 0이면 Print\_zero로 점프

CMP AX, 0

JZ print\_zero

begin\_print:

; 분모(BX)가 0인지 확인 분모(BX)가 0이면 점프

;1의 자리수까지 출력한 경우 출력을 종료

CMP BX,0

JZ end\_print

; 결과가 123인 경우 00123을 출력하지않고 숫자앞의 0을 제외한 123만 출력하기위해 사용

CMP CX, 0

JE calc

; AX<BX 이면 나눗셈 결과는 0이 되므로 AX<BX이면 점프

CMP AX, BX

JB skip

calc:

MOV CX, 0 ; set flag.

MOV DX, 0

DIV BX ; AX = DX:AX / BX (DX=아래 자릿수).

; 숫자를 출력

ADD AL, 30h ; ASCII코드로 변환

PUTC AL ;AL의 값을 출력

MOV AX, DX ; 마지막으로 나눈값의 나머지를 다음자리수 출력을 위해 AX에 저장

skip:

; BX에 BX/10을 넣음

PUSH AX ;BX를 나누기 전에 기존의 값을 저장

MOV DX, 0 ;DX 값 초기화

MOV AX, BX ;10으로 나누기 위해 BX의 값을 AX에 저장

DIV CS:ten ; AX = DX:AX / 10 (DX=remainder).

MOV BX, AX ;10으로 나눈 값을 BX에 저장

POP AX ;AX의 값을 다시 꺼내옴

JMP begin\_print

print\_zero:

PUTC '0'

end\_print:

POP DX

POP CX

POP BX

POP AX

RET

PRINT\_NUM\_UNS ENDP