```
TITLE TinyCalc.asm
; Program Description: A simple calculator with: Addition,
Subtraction, Multiplication,
;                       Division, and Modulo. Additionally, maintains
running total, and
;                       at end of session, report is displayed with
number of operations
;                       performed per operation type, total and average
of all operations.
; Author: Sybil Raphael
; Creation Date: November 29, 2024

INCLUDE Irvine32.inc
INCLUDELIB Irvine32.lib



.data
titleMsg BYTE " ----- Tiny Calculator ----- ", 0                    ;
title message
menuMsg BYTE "Select an operation (A/S/M/D/%): ", 0                 ;
menu message
op1Msg BYTE "Enter the first operand (or 'M' for memory): ", 0      ;
message for first operand
op2Msg BYTE "Enter the second operand (or 'M' for memory): ", 0     ;
message for second operand
answerMsg BYTE "Answer: ", 0                                        ;
message for answer to operation
continueMsg BYTE "Continue? (Yy/Nn): ", 0                           ;
message to continue
invalidMsg BYTE "Invalid input. Try again.", 0                      ;
error, invlaild input message
inputBuffer BYTE 20 DUP(0)                                          ;
user input buffer for up to 20 characters
DBZMsg BYTE "Error: Division by zero.", 0                           ;
error, division by zero message

operation BYTE ?                                                    ;
variable to hold operation type (+,-,*,/,%)
op1 DWORD ?                                                         ;
variable to store operand 1
op2 DWORD ?                                                         ;
variable to store operand 2
op1Buffer BYTE 20 DUP(0)                                            ;
buffer for the first operand (max 20 chars)
op2Buffer BYTE 20 DUP(0)                                            ;
buffer for the second operand (max 20 chars)
memoryResult DWORD ?                                                ;
variable to store result in memory
addCount DWORD 0                                                    ;
```

```
variable to store count of add operation
subCount DWORD 0                                                    ;
variable to store count of sub operation
mulCount DWORD 0                                                    ;
variable to store count of mul operation
divCount DWORD 0                                                    ;
variable to store count of div operation
modCount DWORD 0                                                    ;
variable to store count of mod operation
runningTotal DWORD 0                                                ;
variable to store runningTotal



.code
main PROC
;-----------------------------------------------------------------
;Description: Calls procedures that allows user to perform operations.
;Receives: user input for menu
;Returns: displays calculation results
;-----------------------------------------------------------------
mov memoryResult, 0                                                 ;
initialize memoryResult
mov runningTotal, 0                                                 ;
initialize runningTotal

mov edx, OFFSET titleMsg                                            ;
offset, points to titleMsg
call WriteString                                                    ;
displays titleMsg
call Crlf                                                           ;
displays new line

StartLoop:
    call DisplayMenu                                                ;
calls DisplayMenu
    call GetOperationType                                           ;
calls GetOperationType
    call GetOperands                                                ;
calls GetOperands
    call ProcessOperation                                           ;
calls ProcessOperation
    call AskContinue                                                ;
calls AskContinue
    cmp al, 'N'                                                     ;
compares AL to 'N'
    ;je EndProgram                                                   ;
if 'N', jumps to EndProgram
    jmp StartLoop                                                   ;
loops to start
```

```
EndProgram:
    call DisplayReport                                              ;
calls DisplayReport
    exit                                                            ;
exits program
main ENDP                                                           ;
end of main procedure



DisplayMenu PROC
;-----------------------------------------------------------------
;Description: displays menu
;Receives: menu message
;Returns: nothing
;-----------------------------------------------------------------
mov edx, OFFSET menuMsg                                             ;
offset, points to menuMsg
call WriteString                                                    ;
displays menuMsg
\
ret
DisplayMenu ENDP                                                    ;
end of DisplayMenu procedure



GetOperationType PROC
;-----------------------------------------------------------------
;Description: gets operation type from user
;Receives: A/S/M/D/%
;Returns: nothing
;-----------------------------------------------------------------
call ReadChar                                                       ;
reads char entered by user
call WriteChar                                                      ;
displays char
mov operation, al                                                   ;
AL = operation
call Crlf                                                           ;
displays new line
ret
GetOperationType ENDP                                               ;
end of GetOperationType procedure



GetOperands PROC
;-----------------------------------------------------------------
```

```
;Description: gets operands from user
;Receives: integers
;Returns: nothing
;------------------------------------------------------------------
OP1:
    mov edx, OFFSET op1Msg                                          ;
offset, points to op1Msg
    call WriteString                                                ;
displays op1Msg
    mov edx, OFFSET op1Buffer
    mov ecx, 10                                                     ;
    call ReadString                                                 ;
reads op1
    lea esi, op1Buffer

    mov al, [esi]                                                  ;
AL = address of inputBuffer
    cmp al, 'M'                                                    ;
compares AL to 'M'
    je load1                                                       ;
if equal, jumps to load1
    cmp al, 'm'                                                    ;
compares AL to 'm'
    je load1                                                       ;
if equal, jumps to load1

    call ParseOperand                                              ;
calls ParseOperand
    cmp ax, 0
    jne valid1
    cmp BYTE PTR [esi], '0'
    je valid1

    mov edx, OFFSET invalidMsg                                     ;
offset, points to invalidMsg
    call WriteString                                               ;
displays invalidMsg
    mov eax, 0                                                     ;
clears result (invalidInput)
    jmp OP1                                                        ;
jumps to Done

valid1:
    mov op1, eax                                                   ;
EAX = op1
    jmp OP2

load1:
    mov eax, memoryResult
    mov op1, eax                                                   ;
```

```
EAX = op1
    jmp OP2

OP2:
    mov edx, OFFSET
op2Msg                                              ; offset, points
to op2Msg
    call
WriteString                                                    ;
displays op2Msg
    mov edx, OFFSET op2Buffer
    mov ecx, 10
    call
ReadString                                                     ; reads
op2
    lea esi, op2Buffer

call ParseOperand                                                   ;
calls ParseOperand
mov op2, eax                                                        ;
EAX = op2
call WriteDec
ret
GetOperands ENDP                                                    ;
end of GetOperands procedure



ParseOperand PROC
;-----------------------------------------------------------------
;Description: checks if input 'M' is number, if so, loads memory
;             result into EAX, otherwise, converts string to integer
;             (uses IMUL for string-to-integer conversion)
;Receives: integers
;Returns: nothing
;-----------------------------------------------------------------
push ebx                                                            ;
saves EBX
push ecx                                                            ;
saves ECX
push edx                                                            ;
saves EDX

lea esi, inputBuffer                                                ;
loads efficient address, points to inputBuffer
mov eax, 0                                                          ;
clears EAX register (result)
mov ebx, 1                                                          ;
default sign = +1 (positive)
```

```
mov al, [esi]                                                   ;
AL = address of inputBuffer
cmp al, 'M'                                                     ;
compares AL to 'M'
je LoadMemoryResult                                             ;
if equal, jumps to LoadMemoryResult
cmp al, 'm'                                                     ;
compares AL to 'm'
je LoadMemoryResult                                             ;
if equal, jumps to LoadMemoryResult

cmp al, '-'                                                     ;
compares AL to '-'
jne CheckFirstDigit                                             ;
if NOT equal, jumps to CheckFirstDigit
mov ebx, -1                                                     ;
moves -1 to EBX for negative
inc esi                                                         ;
increments ESI
jmp CheckFirstDigit                                             ;
jumps to CheckFirstDigit

CheckFirstDigit:
    cmp al, '+'                                                 ;
compares AL to '+'
    jne ParseDigits                                             ;
if NOT equal, jumps to ParseDigits
    inc esi                                                     ;
increments ESI

    ParseDigits:
    mov ecx, 0                                                  ;
clears ECX register (digit counter)
    mov edx, 0                                                  ;
clears EDX register (multiplication intermediate)

ParseLoop:
    mov al, [esi]                                               ;
AL = address
    cmp al, 0                                                   ;
compares AL to 0
    je ConvertDone                                              ;
if equal, jumps to ConvertDone
    cmp al, '0'                                                 ;
compares AL to '0'
    jb InvalidInput                                             ;
if below, jumps to InvalidInput
    cmp al, '9'                                                 ;
compares AL to '9'
    ja InvalidInput                                             ;
```

```
if above, jumps to InvalidInput

    sub al, '0'                                                         ;
converts ASCII to number
    imul eax, 10                                                        ;
multiplies result by 10
    add eax, edx                                                        ;
adds current digit to result
    inc esi                                                             ;
increments ESI
    jmp ParseLoop                                                       ;
jumps to ParseLoop

ConvertDone:
    imul eax, ebx                                                       ;
incorporates sign (positive/negative)
    jmp Done                                                            ;
jumps to Done

LoadMemoryResult:
    mov eax, memoryResult                                               ;
EAX = memoryResult
    jmp Done                                                            ;
jumps to Done

InvalidInput:
    mov edx, OFFSET invalidMsg                                          ;
offset, points to invalidMsg
    call WriteString                                                    ;
displays invalidMsg
    mov eax, 0                                                          ;
clears result (invalidInput)
    jmp Done                                                            ;
jumps to Done

Done:
    pop edx                                                             ;
restores EDX
    pop ecx                                                             ;
restores ECX
    pop ebx                                                             ;
restores EBX
    ret
ParseOperand ENDP                                                       ;
end of ParseOperand procedure
```

```
ProcessOperation PROC
;------------------------------------------------------------------
;Description: processes operation based on user selection
;Receives: user input
;Returns: nothing
;------------------------------------------------------------------
cmp operation, 'A'                                                    ;
compares operation to 'A'
je DoAddition                                                         ;
if equal, jumps to DoAddition
cmp operation, 'S'                                                    ;
compares operation to 'S'
je DoSubtraction                                                      ;
if equal, jumps to DoSubtraction
cmp operation, 'M'                                                    ;
compares operation to 'M'
je DoMultiplication                                                   ;
if equal, jumps to DoMultiplication
cmp operation, 'D'                                                    ;
compares operation to 'D'
je DoDivision                                                         ;
if equal, jumps to DoDivision
cmp operation, '%'                                                    ;
compares operation to '%''
je DoModulo                                                           ;
if equal, jumps to DoModulo
ret

DoAddition:
    call Addition                                                     ;
calls Addition procedure
    ret

DoSubtraction:
    call Subtraction                                                  ;
calls Subtraction procedure
    ret

DoMultiplication:
    call Multiplication                                               ;
calls Multiplication procedure
    ret

DoDivision:
    call Division                                                     ;
calls Division procedure
    ret

DoModulo:
    call Modulo                                                       ;
```

```
calls Modulo procedure
    ret
ProcessOperation ENDP                                              ;
end of ProcessOperation procedure



Addition PROC uses eax
;-----------------------------------------------------------------
;Description: adds first and second operand
;Receives: nothing
;Returns: nothing
;-----------------------------------------------------------------
mov eax, op1                                                      ;
EAX = first operand
add eax, op2                                                      ;
adds first with second operand
mov memoryResult, eax                                             ;
moves result into memoryResult
inc addCount                                                      ;
increments addCount
mov edx, OFFSET answerMsg                                         ;
offset, points to answerMsg
call WriteString                                                  ;
displays answerMsg
mov eax, memoryResult                                             ;
moves result into memoryResult
call WriteDec                                                     ;
displays answer as decimal number
call Crlf                                                         ;
displays new line
ret
Addition ENDP                                                     ;
end of Addition procedure




Subtraction PROC uses eax
;-----------------------------------------------------------------
;Description: subtracts first and second operand
;Receives: nothing
;Returns: nothing
;-----------------------------------------------------------------
mov eax, op1                                                     ;
EAX = first operand
sub eax, op2                                                     ;
subtracts first with second operand
mov memoryResult, eax                                            ;
moves result into memoryResult
```

```
inc subCount                                                      ;
increments subCount
mov edx, OFFSET answerMsg                                         ;
offset, points to answerMsg
call WriteString                                                  ;
displays answerMsg
mov eax, memoryResult                                             ;
moves result into memoryResult
call WriteDec                                                     ;
displays answer as decimal number
call Crlf                                                         ;
displays new line
ret
Subtraction ENDP                                                  ;
end of Subtraction procedure



Multiplication PROC
;------------------------------------------------------------------
;Description: multiplies first and second operand
;Receives: nothing
;Returns: nothing
;------------------------------------------------------------------
; Implement multiplication using repeated addition
; Use stack for parameter passing
ret
Multiplication ENDP                                               ;
end of Multiplication procedure




Division PROC
;------------------------------------------------------------------
;Description: divides first and second operand
;Receives: nothing
;Returns: nothing
;------------------------------------------------------------------
; Implement division using repeated subtraction
; Handle divide by zero exception
ret
Division ENDP                                                     ;
end of Division procedure




Modulo PROC
```

```
;-----------------------------------------------------------------
;Description: divison that returns remainder of first and
;             second operand
;Receives: nothing
;Returns: nothing
;-----------------------------------------------------------------
; Use IDIV instruction
ret
Modulo ENDP                                                     ;
end of Modulo procedure




AskContinue PROC
;-----------------------------------------------------------------
;Description: asks user if they want to continue
;Receives: nothing
;Returns: nothing
;-----------------------------------------------------------------
mov edx, OFFSET continueMsg                                     ;
offset, points to continueMsg
call WriteString                                                ;
displays continueMsg
call ReadChar                                                   ;
reads char entered by user
ret
AskContinue ENDP                                                ;
end of AskContinue procedure




DisplayReport PROC
;-----------------------------------------------------------------
;Description: displays final report
;Receives: nothing
;Returns: nothing
;-----------------------------------------------------------------
; Display totals and averages
ret
DisplayReport ENDP                                             ; end
of DisplayReport procedure



end MAIN                                                       ; end
of source code
```