```
;Course:        CSCI 2160-201
;Assignment:    Project 1
;Date:          October 11, 2011
;Purpose:
;       Given two numbers entered by a user, do arithemetic on
;       these two numbers, including addition, subtraction,
;       multiplication, division, and modulus, and then display
;       the results of the arithmetic to the user

    .486
    .model flat
    .stack 100h
    ExitProcess PROTO NEAR32 stdcall, dwExitCode:DWORD
    include Macros201109.asm
    extern ascint:Near32, intasc:Near32

    .data

    ;Prompts
    strIntroPrompt      BYTE    "My name is Brandon Buchanan",0 ;Introduction prompt
    strFirstNumPrompt   BYTE    "Enter your first number: ",0   ;First number prompt
    strSecondNumPrompt  BYTE    "Enter your second number: ",0  ;Second number prompt
    strSumPrompt        BYTE    "The sum is ",0                 ;Sum prompt
    strDifPrompt        BYTE    "The difference is ",0          ;Different prompt
    strMulPrompt        BYTE    "The product is ",0             ;Product prompt
    strDivPrompt        BYTE    "The quotient is ",0            ;Quotient prompt
    strModPrompt        BYTE    "The remainder is ",0           ;Remainder prompt
    strInvalidNum       BYTE    "Invalid numeric string.  Re-enter",0   ;Invalid numeric
    string prompt
    strOverflow         BYTE    "Overflow occured.  Re-enter",0         ;Overflow prompt
    CRLF                BYTE    10,13,0                         ;Carriage line return feed

    ;Variables
    strInput            BYTE    20 dup(?)   ;Read input from user
    strAnswer           BYTE    20 dup(?)   ;Stores the calculated answer
    intFirstNum         DWORD   ?           ;First number from user
    intSecondNum        DWORD   ?           ;Second number from user
    intSum              DWORD   ?           ;Sum of the two numbers
    intDifference       DWORD   ?           ;Difference of the two numbers
    intProduct          DWORD   ?           ;Product of the two numbers
    intQuotient         DWORD   ?           ;Quotient of the two numbers
    intRemainder        DWORD   ?           ;Remainder of the two numbers after dividing


    .code

_start:

;Main program
main PROC
    puts strIntroPrompt         ;display introduction prompt
```

```
getFirstNum:
    puts CRLF                   ;display a new line
    puts strFirstNumPrompt      ;display first number prompt
    gets strInput,20            ;get first number from keyboard

    cmp strInput,0              ;if the first character is null
    je endProg                  ; then exit the program

    push offset strInput        ;pass inputed string's address to ascint
    call ascint                 ;convert a string to an integer
    jc invalidFirstNumber       ;if the first number is invalid, try again
    jo overflowFirstNumber      ;if the first number is larger than 4 bytes, try again
    mov intFirstNum,eax         ;retrieve returned integer from ascint
    add esp,4                   ;clean up the stack

getSecondNum:
    puts CRLF                   ;display a new line
    puts strSecondNumPrompt     ;display second number prompt
    gets strInput,20            ;get second number from keyboard

    cmp strInput,0              ;if the first character is null
    je endProg                  ; then exit the program

    push offset strInput        ;pass inputed string's address to ascint
    call ascint                 ;convert a string to an integer
    jc invalidSecondNumber      ;if the second number is invalid, tell the user and try again
    jo overflowSecondNumber     ;if the first number is larger than 4 bytes, try again
    mov intSecondNum,eax        ;retrieve returned integer from ascint
    add esp,4                   ;clean up the stack

calculateAnswers:

    ;Calculate the sum
    push intSecondNum           ;pass the second number to sum
    push intFirstNum            ;pass the first number to sum
    call sum                    ;call the sum method
    mov intSum,eax              ;retrieve the returned value from sum
    add esp,8                   ;clean up the stack

    ;Calculate the difference
    push intSecondNum           ;pass the second number to difference
    push intFirstNum            ;pass the first number to difference
    call difference             ;call the difference method
    mov intDifference,eax       ;retrieve the returned value from difference
    add esp,8                   ;clean up the stack

    ;Calculate the product
    push intSecondNum           ;pass the second number to product
    push intFirstNum            ;pass the first number to product
    call product                ;call the product method
    mov intProduct,eax          ;retrieve the returned value from product
    add esp,8                   ;clean up the stack
```

- did not check for overflow on addition or sub — or mult — 4

```
    ;Calculate the quotient
    push intSecondNum           ;pass the second number to quotient
    push intFirstNum            ;pass the first number to quotient
    call quotient               ;call the quotient method
    mov intQuotient,eax         ;retrieve the returned value from quotient
    add esp,8                   ;clean up the stack

    ;Calculate the remainder
    push intSecondNum           ;pass the second number to remainder
    push intFirstNum            ;pass the first number to remainder
    call remainder              ;call the remainder method
    mov intRemainder,eax        ;retrieve the returned value from remainder
    add esp,8                   ;clean up the stack

outputAnswers:

    ;Display the sum
    puts CRLF                   ;display a new line
    puts strSumPrompt           ;display sum prompt
    push intSum                 ;convert the sum
    push offset strInput        ;  into a string
    call intasc                 ;    and store at strInput
    add esp,8                   ;clean up the stack
    puts strInput               ;display the sum to the screen

    ;Display the difference
    puts CRLF                   ;display a new line
    puts strDifPrompt           ;display difference prompt
    push intDifference          ;conver the difference
    push offset strInput        ;  into a string
    call intasc                 ;    and store at strInput
    add esp,8                   ;clean up the stack
    puts strInput               ;display the difference to the screen

    ;Display the product
    puts CRLF                   ;display a new line
    puts strMulPrompt           ;display the product prompt
    push intProduct             ;convert the product
    push offset strInput        ;  into a string
    call intasc                 ;    and store at strInput
    add esp,8                   ;clean up the stack
    puts strInput               ;display the product to the screen

    ;Display the quotient
    puts CRLF                   ;display a new line
    puts strDivPrompt           ;display the quotient prompt
    push intQuotient            ;convert the quotient
    push offset strInput        ;  into a string
    call intasc                 ;    and store at strInput
    add esp,8                   ;clean up the stack
    puts strInput               ;display the quotient to the screen
```

```
    ;Display the remainder
    puts CRLF                   ;display a new line
    puts strModPrompt           ;display the remainder prompt
    push intRemainder           ;convert the remainder
    push offset strInput        ;  into a string
    call intasc                 ;    and store at strInput
    add esp,8                   ;clean up the stack
    puts strInput               ;display the remainder to the screen

endProg:
    INVOKE ExitProcess,0
    PUBLIC _start


invalidFirstNumber:
    puts CRLF                   ;display a new line
    puts strInvalidNum          ;display invalid number prompt
    jmp getFirstNum             ;ask for the first number again

invalidSecondNumber:
    puts CRLF                   ;display a new line
    puts strInvalidNum          ;display invalid number prompt
    jmp getSecondNum            ;ask for the second number again

overflowFirstNumber:
    puts CRLF                   ;display a new line
    puts strOverflow            ;display overflow prompt
    jmp getFirstNum             ;ask for the first number again

overflowSecondNumber:
    puts CRLF                   ;display a new line
    puts strOverflow            ;display overflow prompt
    jmp getSecondNum            ;ask for the second number again

;*********************************************
;              FUNCTIONS
;*********************************************

;Procedure that calculates the sum of two integers
sum PROC
    push ebp                    ;enter
    mov ebp,esp                 ;  the
    push ebx                    ;    method

    mov ebx,[ebp+8]             ;add the first
    add ebx,[ebp+12]            ;  number to the
    mov eax,ebx                 ;    second number

    pop ebx                     ;exit
    pop ebp                     ;  the
    ret                         ;    method
```

```
sum ENDP

;Procedure that calculates the difference of two integers
difference PROC
    push ebp                    ;enter
    mov ebp,esp                 ;  the
    push ebx                    ;    method

    mov ebx,[ebp+8]             ;subtract the first
    sub ebx,[ebp+12]            ;   number from the
    mov eax,ebx                 ;     second number

    pop ebx                     ;exit
    pop ebp                     ;  the
    ret                         ;    method
difference ENDP

;Procedure that calculates the product of two integers
product PROC
    push ebp                    ;enter the
    mov ebp,esp                 ;  method

    mov eax,[ebp+8]             ;multiply the
    imul dword ptr [ebp+12]     ;  two numbers

    pop ebp                     ;exit the
    ret                         ;  method
product ENDP

;Procedure that calculates the quotient of two integers
quotient PROC
    push ebp                    ;enter the
    mov ebp,esp                 ;  method

    mov eax,[ebp+8]             ;divide
    cdq                         ;  the two
    idiv dword ptr [ebp+12]     ;    numbers

    pop ebp                     ;exit the
    ret                         ;  method
quotient ENDP

;Procedure that calculates the remainder of two integers after dividing
remainder PROC
    push ebp                    ;enter
    mov ebp,esp                 ;  the
    push edx                    ;    method

    mov eax,[ebp+8]             ;divide the two
    cdq                         ;  numbers and
    idiv dword ptr [ebp+12]     ;    keep the
    mov eax,edx                 ;      remainder
```

```
    pop edx                     ;exit
    pop ebp                     ;  the
    ret                         ;    method
remainder ENDP

main ENDP

END                             ;end of program
```