```
;Programemr:     Brandon Buchanan
;Program:        proj3BWB.exe
;Course:         CSCI 2160-201
;Assignment:     Project 3
;Date:           November 1, 2011
;Purpose:
;       Driver program that accepts a string from
;       the console, adds 5 to the string, and displays
;       the result to the user.  If there is invalid input
;       or if an overflow occurs, a resepective message is
;       displayed.

    .486
    .model flat
    .stack 100h
    ExitProcess PROTO NEAR32 stdcall, dwExitCode:DWORD

    extern  intasc32:Near32,
            ascint32:Near32,
            getchar:Near32,
            getcharecho:Near32,
            putchar:Near32

    .data

    ;Constants
    RETURN equ 13               ;ASCII decimal value for enter key
    BACKSPACE equ 8             ;ASCII decimal value for backspace
    SPACE equ 32                ;ASCII decimal value for space
    ZERO equ 30h                ;ASCII hex value for zero
    ONE equ 31h                 ;ASCII hex value for one
    TWO equ 32h                 ;ASCII hex value for two
    THREE equ 33h               ;ASCII hex value for three
    FOUR equ 34h                ;ASCII hex value for four
    FIVE equ 35h                ;ASCII hex value for five
    SIX equ 36h                 ;ASCII hex value for six
    SEVEN equ 37h               ;ASCII hex value for seven
    EIGHT equ 38h               ;ASCII hex value for eight
    NINE equ 39h                ;ASCII hex value for nine
    PLUS equ 2Bh                ;ASCII hex value for plus
    MINUS equ 2Dh               ;ASCII hex value for minus

    strInvalid  BYTE    "INVALID INPUT",0
    strOverflow BYTE    "OVERFLOW OCCURRED",0

    ;Variables
    strInput    BYTE    21 dup(?)           ;Input data from the keyboard

    .code

_start2:
```

-20
doesn't work
-40
40

```
;Main program
main PROC

    ;allow user to enter an integer value

    push word ptr 20                ;only allow 20 characters max
    push offset strInput            ;string's address to store in
    call getstring                  ;get input from the keyboard
    add esp,6                       ;clean up the stack

    push offset strInput            ;convert the
    call ascint32                   ;  string to
    add esp,4                       ;    an integer

    add eax,5                       ;add 5 to converted integers

    push eax                        ;convert
    push offset strInput            ;  the
    call intasc32                   ;    integer to
    add esp,8                       ;      a string

    ; push esp                      ;display
    ; call putstring                ;  the new
    ; add esp,4                     ;    string

    ;end the program
    INVOKE ExitProcess,0
    PUBLIC _start2


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

;Procedure that accepts a null terminated string and outputs the
;string to the console screen
putstring PROC

    push ebp                    ;enter
    mov ebp,esp                 ;  the
    push ebx                    ;    method
    push ecx                    ;----------

    mov ebx,[ebp+8]             ;store the string's address

    xor ecx,ecx                 ;clear ecx, create a counter to reference
                                ;  within the string
    displayString:
        mov al,[ebx+ecx]        ;store the current character in the al register
        cmp al,0                ;if the contents of al is 0, then the string
        je endOfString          ;  is done and the loop is exited
        push eax                ;pass the character to display to the method
```

```
        call putchar            ;display a character to the string
        add esp,4               ;clean up the stack
        add ecx,1               ;increment the counter by 1
        jmp displayString       ;restart the loop

    endOfString:                ;label so that the displayString loop can be
                                ;  escaped

    pop ecx                     ;----------
    pop ebx                     ;exit
    pop ebp                     ;  the
    ret                         ;    method

putstring ENDP

;Procedure that accepts a string's address and the number of max
;characters.  The procedure keeps getting characters from the
;keyboard until the user presses enter.  If a backspace is displayed,
;the previous character is erased.
getstring PROC

    push ebp                    ;enter
    mov ebp,esp                 ;  the
    push ebx                    ;    method
    push ecx                    ;
    push edx                    ;----------

    xor edx,edx                 ;clear edx register
    mov ebx,[ebp+8]             ;store the string's address
    xor ecx,ecx                 ;clear ecx, store number of characters entered
    mov dx,[ebp+12]             ;store the max number of characters that
                                ; can be entered by the user

    xor eax,eax                 ;clear the eax register

    enterString:
        call getchar                ;get character from keyboard

        cmp al,RETURN               ;if the enter key was pressed,
        je enterKeyPressed          ;  then exit the loop
        cmp al,BACKSPACE            ;if the backspace was pressed,
        je backspacePressedLabel    ;  then erase the previous character
        cmp ecx,edx                 ;if user has entered the max number of
        je enterString              ;  characters, reset loop

        push eax                    ;display the
        call putchar                ;  character to
        add esp,4                   ;    the screen

        mov [ebx+ecx],eax           ;store the character in the string

        inc ecx                     ;increment counter by 1
```

```
    jmp enterString                 ;continue the loop

    backspacePressedLabel:

        call backspacePressed       ;display the backspace
        dec ecx                     ;decrease the counter by 1

        jmp enterString             ;restart the loop

    enterKeyPressed:                ;label so that the loop can be escaped
        mov byte ptr [ebx+ecx],0    ;append a null onto the end of the string

    pop edx                     ;----------
    pop ecx                     ;
    pop ebx                     ;exit
    pop ebp                     ;  the
    ret                         ;    method

getstring ENDP

;Procedure defines what to do when the backspace is pressed.
;This procedure defines that process to be what a user would
;expect to happen, that is, the previous character is deleted.
backspacePressed PROC
    push ebp                    ;enter the
    mov ebp,esp                 ;  method
    push eax

    xor eax,eax                 ;clear the eax

    mov al,BACKSPACE            ;prep the backspace char
    push eax                    ;pass the character
    call putchar                ;display a character to the screen
    add esp,4                   ;clean up the stack

    mov al,SPACE                ;prep the space char
    push eax                    ;pass the character
    call putchar                ;display a character to the screen
    add esp,4                   ;clean up the stack

    mov al,BACKSPACE            ;prep the backspace char
    push eax                    ;pass the character
    call putchar                ;display a character to the screen
    add esp,4                   ;clean up the stack

    pop eax
    pop ebp                     ;  exit the
    ret                         ;    method
backspacePressed ENDP
```

```
main ENDP

END                                 ;end of program
```

```
;Programemr:     Brandon Buchanan
;Program:        conversion.exe
;Course:         CSCI 2160-201
;Assignment:     Project 3
;Date:           November 1, 2011
;Purpose:
;       Contains two methods - one for converting a
;       string to an integer and one for converting an
;       integer to a string.  Also contains helper methods
;       for the two main methods.

    .486
    .model flat
    .stack 100h
    ExitProcess PROTO NEAR32 stdcall, dwExitCode:DWORD

    .data

    ;Constants
    RETURN equ 13               ;ASCII decimal value for enter key
    BACKSPACE equ 8             ;ASCII decimal value for backspace
    SPACE equ 32                ;ASCII decimal value for space
    ZERO equ 30h                ;ASCII decimal value for zero
    ONE equ 31h                 ;ASCII decimal value for one
    TWO equ 32h                 ;ASCII decimal value for zero
    THREE equ 33h               ;ASCII decimal value for one
    FOUR equ 34h                ;ASCII decimal value for zero
    FIVE equ 35h                ;ASCII decimal value for one
    SIX equ 36h                 ;ASCII decimal value for zero
    SEVEN equ 37h               ;ASCII decimal value for one
    EIGHT equ 38h               ;ASCII decimal value for zero
    NINE equ 39h                ;ASCII decimal value for one
    PLUS equ 2Bh                ;ASCII hex value for plus
    MINUS equ 2Dh               ;ASCII hex value for minus

    .code

_start:

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

; Procedure to convert an integer value to a string of characters
;
;  void intasc32(str @address, int numToConvert)
intasc32 PROC

    push ebp
    mov ebp,esp
    push eax
    push ebx
```

```
    push ecx
    push edx
    push esi

    mov eax,[ebp+12]            ;integer to convert to a string
    mov esi,[ebp+8]             ;starting address of the string
    xor edx,edx                 ;clear register to hold remainder
    xor ecx,ecx                 ;clear register to hold counter
    xor ebx,ebx                 ;clear ebx register

    ;program is crashing
    ;  within this loop
    .WHILE eax >= 10            ;while the integer is 10 or greater

        cdq                     ;divide
        mov ebx,10              ;
        idiv ebx                ;  by 10

        push edx                ;save the remainder onto the stack
        inc ecx                 ;increase counter by 1

    .ENDW

    .WHILE ecx >= 0

        pop eax                 ;pop the stack and
        add al,30h              ;  convert the number to a string
        mov [esi],al            ;save the value to the string
        inc esi                 ;  then move the string's pointer
        dec ecx

    .ENDW

    pop esi
    pop edx
    pop ecx
    pop ebx
    pop eax
    pop ebp
    ret

intasc32 ENDP


;Procedure to convert a string to an integer
;
;  int ascint32(str @strToConvert)
ascint32 PROC

    push ebp
    mov ebp,esp
    push eax
```

```
    push ebx
    push ecx
    push edx
    push esi

    mov esi,[ebp+8]                 ;get the string's starting address
    xor ecx,ecx                     ;clear the counter

    ;doesn't account for + or - yet
    .WHILE byte ptr [esi] != 0      ;
        mov ebx,30h                 ;
        cmp [esi],ebx               ; if the character is a valid number,
        jnge NotNumber              ;  then increment the counter and
        mov ebx,40h                 ;    move the string's pointer
        cmp [esi],ebx               ;
        jnle NotNumber              ;
        inc ecx                     ;
        inc esi                     ;
    .ENDW                           ;

    mov esi,[ebp+8]                 ;get the string's starting address again
    mov ebx,ecx                     ;save the counter for the next loop

    .WHILE ecx >= 0                 ;while there are characters
        push [esi]                  ;  left, loop through the string
        inc esi                     ;    and push each character onto
        dec ecx                     ;      the stack
    .ENDW

    mov esi,[ebp+8]                 ;get the string's starting address again
    mov ecx,ebx                     ;recover the string's length
    xor ebx,ebx                     ;clear the register to store the counter going up
    xor edx,edx                     ;clear the register to store the total

    push ebx                        ;push the string's length
    push [esi]                      ;push the string's address
    call ConvertInteger
    add esp,8

    jmp doneConverting

NotNumber:
    stc                             ;set the carry flag
    mov eax,0                       ;get ready to return 0

doneConverting:
    pop esi
    pop edx
    pop ecx
    pop ebx
    pop eax
    pop ebp
```

```
    ret

ascint32 ENDP

;Loop through the string and returns its integer representation
;  Returns 0 when there is an overflow
;
;  int ConvertInteger(str @address, int strSize)
ConvertInteger PROC

    push ebp
    mov ebp,esp
    push ecx
    push edx
    push esi

    mov esi, [ebp+12]              ;store the string's address
    mov ecx,[ebp+8]                ;store the string's size

    .WHILE ebx < ecx               ;pop each character from the stack, multiply and add
    integers as needed

        push ebx                   ;calculate
        call PowerOfTen            ;  10 to the
        add esp,4                  ;    Nth power

        mov edx,[esi]              ;store the character in the string
        sub edx,30h                ;convert ASCII numbers to decimal numbers
        imul edx                   ;multiply retrieved number by eax (10^n)
        jo Overflow

        add edx,eax                ;add to the total
        jo OverFlow                ;stop if there is an overflow

        mov eax,edx
        inc ebx
    .ENDW

    jmp done

Overflow:
    mov eax,0

done:
    pop esi
    pop edx
    pop ecx
    pop ebp
    ret

ConvertInteger ENDP
```

```
;Procedure that returns 10 to the Nth power
;  int PowerOfTen(int powerToCalculate)
PowerOfTen PROC

    push ebp
    mov ebp,esp
    push eax
    push ebx
    push ecx

    mov eax,1
    mov ebx,[ebp+8]

    .WHILE ebx > 0
        mov ecx,10
        imul ecx
        dec ebx
    .ENDW

    pop ecx
    pop ebx
    pop eax
    pop ebp
    ret

PowerOfTen ENDP

PUBLIC _start

END                             ;end of program
```