```
Microsoft (R) Macro Assembler Version 14.15.26729.0         10/10/18 10:24:39
#numberAnalysis (numberAnalysis.asm                          Page 1 - 1


                                TITLE numberAnalysis (numberAnalysis.asm)

                                ; Name: Brandon Hough
                                ; CPEN 3710
                                ; Date: October 9, 2018

                                ; This program will repeatedly prompt the user to enter in a signed integer.
                                ; The program will analyze the number and display if the number is positive/negative,
                                ; whether the absolute values of the signed integer are >,<, or = 10000, and if
                                ; the number is evenly divisible by 8 for each number, then will terminate
                                ; when the user enters the value '0'

                                include Irvine32.inc
                              C ; Include file for Irvine32.lib             (Irvine32.inc)
                              C
                              C ;OPTION CASEMAP:NONE            ; optional: make identifiers case-sensitive
                              C
                              C INCLUDE SmallWin.inc            ; MS-Windows prototypes, structures, and constants
                              C .NOLIST
                              C .LIST
                              C
                              C INCLUDE VirtualKeys.inc
                              C ; VirtualKeys.inc
                              C .NOLIST
                              C .LIST
                              C
                              C
                              C .NOLIST
                              C .LIST
                              C

 00000000                       .data

 00000000 45 6E 74 65 72                prompt BYTE 'Enter a Signed Number: ',0                     ; prompt of bytes that will displayed on   ➩
➩the command prompt to ask user for input
           20 61 20 53 69
           67 6E 65 64 20
           4E 75 6D 62 65
           72 3A 20 00
 00000018 00000000                      userValue SDWORD ?                                          ; stores the users input

 0000001C 20 69 73 20 61                negResult BYTE ' is a negative number.',0                   ; prompt of bytes that will displayed on   ➩
➩the command prompt when a negative number is displayed
           20 6E 65 67 61
           74 69 76 65 20
           6E 75 6D 62 65
           72 2E 00
 00000033 20 69 73 20 61                posResult BYTE ' is a positive number.',0                 ; prompt of bytes that will displayed on    ➾
➾the command prompt when a positive number is displayed
           20 70 6F 73 69
           74 69 76 65 20
           6E 75 6D 62 65
           72 2E 00

 0000004A 54 68 65 20 61            absVal BYTE 'The absolute value of ',0                    ; prompt of bytes that will displayed on the    ➾
➾command prompt when a positivenumber is displayed
           62 73 6F 6C 75
           74 65 20 76 61
           6C 75 65 20 6F
           66 20 00
 00000061 20 69 73 20 67            greaterThan BYTE ' is greater than 10000.',0              ; prompt of bytes that will displayed on the    ➾
➾command prompt for absolute value if > 10000
           72 65 61 74 65
           72 20 74 68 61
           6E 20 31 30 30
           30 30 2E 00
 00000079 20 69 73 20 6C            lessThan BYTE ' is less than 10000.',0                    ; prompt of bytes that will displayed on the    ➾
➾command prompt for absolute value if < 10000
           65 73 73 20 74
           68 61 6E 20 31
           30 30 30 30 2E
           00
 0000008E 20 69 73 20 65            equalTo BYTE ' is equal to 10000.',0                      ; prompt of bytes that will displayed on the    ➾
➾command prompt for absolute value if = 10000
           71 75 61 6C 20
           74 6F 20 31 30
           30 30 30 2E 00

 000000A2 20 69 73 20 6E            isNotDiv BYTE ' is not evenly divisible by 8.',0          ; prompt of bytes that will displayed on the    ➾
➾command prompt for if number not evenly divisible by 8
           6F 74 20 65 76
           65 6E 6C 79 20
           64 69 76 69 73
           69 62 6C 65 20
           62 79 20 38 2E
           00
 000000C1 20 69 73 20 65            isDiv BYTE ' is evenly divisible by 8.',0                 ; prompt of bytes that will displayed on the    ➾
➾command prompt for if number eveny divisible by 8
           76 65 6E 6C 79
           20 64 69 76 69
           73 69 62 6C 65
           20 62 79 20 38
           2E 00

 00000000                         .code
 00000000                         main proc

                                        .repeat                                                ; will repeat this section of code until    ➾
```

```
→user enters '0'.
 00000000                 *@C0001:

 00000000  BA 00000000 R               mov edx, OFFSET prompt                                       ; store a pointer to the first byte of the  →
→prompt
 00000005  E8 00000000 E               call WriteString                                             ; prints out the prompt string to the       →
→commmand prompt window
 0000000A  E8 00000000 E               call ReadInt                                                 ; read the 32-bit signed integer into eax   →
→register
 0000000F  A3 00000018 R               mov userValue, eax                                           ; store the user input from the eax         →
→register into the SDWORD userValue

 00000014  E8 0000001F                 call analyzeNumberSign                                       ; call procedure to analyze the numbers sign
 00000019  E8 00000054             call analyzeNumberAbsVal                                     ; call procedure to analyze the numbers value   →
→compared to (<,>, or =) 10000
 0000001E  E8 000000DC             call analyzeNumberDiv8                                       ; call procedure to analyze if the number is    →
→divisble by 8
 00000023  E8 00000000 E           call crlf                                                    ; does a character return to the next line

                                       .until userValue == 0                                        ; needed to ensure these previous lines     →
→will run till the user enters '0'
 00000028  83 3D 00000018 R
           00             *        cmp    userValue, 000h
 0000002F  75 CF          *        jne    @C0001

                                exit
 00000031  6A 00          *        push   +000000000h
 00000033  E8 00000000 E  *        call   ExitProcess
 00000038                       main endp

                                ; ------------------------------------------------
                                ; This sub-program will analyze user input integers
                                ; to determine whether the numbers are +/-

 00000038                       analyzeNumberSign proc

 00000038  83 F8 00                    CMP eax, 0                      ; compares the value in eax to 0
 0000003B  7F 04                       JG printPositive                ; will jump to printPositive if eax has a value greater than 0
 0000003D  7C 17                       JL printNegative                ; will jump to printNegative if eax has a value less than 0
 0000003F  74 2A                       JE zeroCase                     ; will jump to zeroCase if eax has a value equal to 0

 00000041                              printPositive:
 00000041  E8 00000000 E               call WriteInt                   ; write the user input to the screen
 00000046  BA 00000033 R               mov edx, OFFSET posResult       ; store a pointer to the first byte of the posResult (' is a positive   →
→number.')
 0000004B  E8 00000000 E               call WriteString                ; write that previous string of bytes to the command prompt
 00000050  E8 00000000 E               call crlf                       ; does a character return to the next line
 00000055  C3                          ret                             ; return to main

 00000056                              printNegative:
 00000056  E8 00000000 E               call WriteInt                   ; write the user input to the screen
 0000005B  BA 0000001C R                mov edx, OFFSET negResult      ; store a pointer to the first byte of the negResult (' is a negative   ⇾
⇾number.')
 00000060  E8 00000000 E                call WriteString               ; write that previous string of bytes to the command prompt
 00000065  E8 00000000 E                call crlf                      ; does a character return to the next line
 0000006A  C3                           ret                            ; return to main

 0000006B                               zeroCase:
                                        exit                           ; exit program if eax has a value of 0
 0000006B  6A 00            *       push   +000000000h
 0000006D  E8 00000000 E    *       call   ExitProcess

 00000072                       analyzeNumberSign endp

                                ; -------------------------------------------------------
                                ; This sub-program will analyze user input integers
                                ; to determine whether absolute values >,<, or = 10000

 00000072                       analyzeNumberAbsVal proc

 00000072  99                       cdq                             ; copies the sign of the register eax to register edx
 00000073  83 FA 00                 CMP edx, 0                      ; compare the value in edx to '0'
 00000076  74 02                    JE postiveNumAbs                ; will jump to postiveNumAbs if edx has a value equal to 0
 00000078  75 0B                    JNE negativeNumAbs              ; will jump to negativeNumAbs if edx has a value not equal to 0

 0000007A                           postiveNumAbs:
 0000007A  3D 00002710              CMP eax, 10000                  ; compare the value in eax to '10000'
 0000007F  7F 12                    JG printGreaterThan             ; will jump to printGreaterThan if eax has a value greater than '10000'
 00000081  7C 34                    JL printLessThan                ; will jump to printLessThan if eax has a value less than '10000'
 00000083  74 56                    JE equalToNum                   ; will jump to equalToNum  if eax has a value equal to '10000'

 00000085                           negativeNumAbs:
 00000085  F7 D8                    NEG eax                        ; negate the negative values to make them postive
 00000087  3D 00002710              CMP eax, 10000                 ; compare the value in eax to '10000'
 0000008C  74 4D                    JE equalToNum                  ; will jump to equalToNum if eax has a value equal to '10000'
 0000008E  7F 03                    JG printGreaterThan            ; will jump to printGreaterThan if eax has a value greater than '10000'
 00000090  7C 25                    JL printLessThan               ; will jump to printLessThan if eax has a value less than '10000'
 00000092  C3                       ret                            ; return to main

 00000093                           printGreaterThan:
 00000093  A1 00000018 R            mov eax, userValue             ; restore the eax register to the users value they entered
 00000098  BA 0000004A R            mov edx, OFFSET absVal         ; store a pointer to the first byte of the absVal ('The absolute value of ')
 0000009D  E8 00000000 E            call WriteString               ; write that previous string of bytes to the command prompt
 000000A2  E8 00000000 E            call WriteInt                  ; write the user input to the screen
 000000A7  BA 00000061 R                mov edx, OFFSET greaterThan    ; store a pointer to the first byte of the greaterThan  (' is greater   ⇾
⇾than 10000.')
 000000AC  E8 00000000 E                call WriteString               ; write that previous string of bytes to the command prompt
 000000B1  E8 00000000 E                call crlf                      ; does a character return to the next line
 000000B6  C3                           ret                            ; return to main

 000000B7                           printLessThan:
 000000B7  A1 00000018 R            mov eax, userValue             ; restore the eax register to the users value they entered
 000000BC  BA 0000004A R           mov edx, OFFSET absVal         ; store a pointer to the first byte of the absVal ('The absolute value of ')
 000000C1  E8 00000000 E           call WriteString               ; write that previous string of bytes to the command prompt
 000000C6  E8 00000000 E           call WriteInt                  ; write the user input to the screen
 000000CB  BA 00000079 R               mov edx, OFFSET lessThan       ; store a pointer to the first byte of the lessThan (' is less than     ⇾
⇾10000.')
 000000D0  E8 00000000 E               call WriteString               ; write that previous string of bytes to the command prompt
 000000D5  E8 00000000 E               call crlf                      ; does a character return to the next line
 000000DA  C3                          ret                            ; return to main

 000000DB                          equalToNum:
 000000DB  A1 00000018 R           mov eax, userValue             ; restore the eax register to the users value they entered
 000000E0  BA 0000004A R           mov edx, OFFSET absVal         ; store a pointer to the first byte of the absVal ('The absolute value of ')
 000000E5  E8 00000000 E           call WriteString               ; write that previous string of bytes to the command prompt
 000000EA  E8 00000000 E           call WriteInt                  ; write the user input to the screen
 000000EF  BA 0000008E R           mov edx, OFFSET equalTo        ; store a pointer to the first byte of the equalTo (' is equal to 10000.')
 000000F4  E8 00000000 E               call WriteString               ; write that previous string of bytes to the command prompt
 000000F9  E8 00000000 E               call crlf                      ; does a character return to the next line
 000000FE  C3                          ret                            ; return to main


 000000FF                      analyzeNumberAbsVal endp

                               ; ---------------------------------------------------------
                               ; This sub-program will analyze user input integers
                               ; to determine whether the numbers are evenly divisible by 8

 000000FF                      analyzeNumberDiv8 proc

 000000FF  BA 00000000             mov edx, 0                     ; reset edx register to '0'
 00000104  A1 00000018 R           mov eax, userValue             ; move the users value into the eax register
 00000109  BB 00000008             mov ebx, 8                     ; move 8 into the ebx register
 0000010E  F7 FB                   idiv ebx                       ; does the computation of the registers ebx/eax and store remainder in edx

 00000110  83 FA 00                CMP edx, 0                     ; compare the remainder in edx to '0'
 00000113  75 02                       JNE printNotDiv                ; will jump to printNotDiv if edx has a value other than '0'
 00000115  74 1A                       JE printDiv                    ; will jump to printDiv  if edx has a value of '0'

 00000117                          printNotDiv:
 00000117  A1 00000018 R           mov eax, userValue             ; store the user value into eax
 0000011C  E8 00000000 E           call WriteInt                  ; write the user input to the screen
 00000121  BA 000000A2 R               mov edx, OFFSET isNotDiv       ; store a pointer to the first byte of the isNotDiv (' is not evenly    ⇾
⇾divisible by 8.')
 00000126  E8 00000000 E               call WriteString               ; write that previous string of bytes to the command prompt
 0000012B  E8 00000000 E           call crlf                      ; dooes a character return to the next line
 00000130  C3                          ret                            ; return to main

 00000131                          printDiv:
 00000131  A1 00000018 R           mov eax, userValue             ; store the user value into eax
 00000136  E8 00000000 E           call WriteInt                  ; write the user input to the screen
 0000013B  BA 000000C1 R               mov edx, OFFSET isDiv          ; store a pointer to the first byte of the isDiv (' is evenly divisible ⇾
⇾by 8.')
```

```
 00000140  E8 00000000 E                call WriteString               ; write that previous string of bytes to the command prompt
 00000145  E8 00000000 E            call crlf                      ; does a character return to the next line
 0000014A  C3                       ret                            ; return to main

 0000014B                       analyzeNumberDiv8 endp
                                end main
Microsoft (R) Macro Assembler Version 14.15.26729.0        10/10/18 10:24:39
#numberAnalysis (numberAnalysis.asm                         Symbols 2 - 1




Structures and Unions:

                N a m e                 Size
                                        Offset      Type

CONSOLE_CURSOR_INFO  . . . . . .        00000008
  dwSize . . . . . . . . . . . .        00000000        DWord
  bVisible . . . . . . . . . . .        00000004        DWord
CONSOLE_SCREEN_BUFFER_INFO . . .        00000016
  dwSize . . . . . . . . . . . .        00000000        DWord
  dwCursorPosition . . . . . . .        00000004        DWord
  wAttributes  . . . . . . . . .        00000008        Word
  srWindow . . . . . . . . . . .        0000000A        QWord
  dwMaximumWindowSize  . . . . .        00000012        DWord
COORD  . . . . . . . . . . . . .        00000004
  X  . . . . . . . . . . . . . .        00000000        Word
  Y  . . . . . . . . . . . . . .        00000002        Word
FILETIME . . . . . . . . . . . .        00000008
  loDateTime . . . . . . . . . .        00000000        DWord
  hiDateTime . . . . . . . . . .        00000004        DWord
FOCUS_EVENT_RECORD . . . . . . .        00000004
  bSetFocus  . . . . . . . . . .        00000000        DWord
FPU_ENVIRON  . . . . . . . . . .        0000001C
  controlWord  . . . . . . . . .        00000000        Word
  statusWord . . . . . . . . . .        00000004        Word
  tagWord  . . . . . . . . . . .        00000008        Word
  instrPointerOffset . . . . . .        0000000C        DWord
  instrPointerSelector . . . . .        00000010        DWord
  operandPointerOffset . . . . .        00000014        DWord
  operandPointerSelector . . . .        00000018        Word
INPUT_RECORD . . . . . . . . . .        00000014
  EventType  . . . . . . . . . .        00000000        Word
  Event  . . . . . . . . . . . .        00000004        XmmWord
  bKeyDown . . . . . . . . . . .        00000000        DWord
  wRepeatCount . . . . . . . . .        00000004        Word
  wVirtualKeyCode  . . . . . . .        00000006        Word
  wVirtualScanCode . . . . . . .        00000008        Word
  uChar  . . . . . . . . . . . .        0000000A        Word
  UnicodeChar  . . . . . . . . .        00000000        Word
```