CSCI2160Proj4F11HexConversion Name: Brandon Buchanan

Due: 11/22/2011 at 11:05 (the end of lab on Tuesday, Nov. 22) (**10 point BONUS** if completed by Thursday, Nov. 17, 2011).Staple this sheet as a cover sheet to your submission. Include all source code (for both your driver and your external file). You will need an object file that contains the **getstring** and **putstring** methods. Do NOT copy code

Store in a file named **hexstring.asm** that will contain **two** methods.

-2

1) **void hextochar (&strToStoreCharacters, byte chr)**

The method will accept two parameters: the first is the address of a 2-byte string that will hold the 2 hex characters derived from the byte passed to it. For example if the byte has a value **F4** as its hex value, then the string would contain two new bytes **4634** (the character 'F' and the character '4').

2) **void hextostring (&strToStoreCharacters, &strHexBytes, dword dnum)**

The method accepts the address of a string where the characters will be stored (&strToStoreCharacters) as well as the address of string of bytes to be converted (&strHexBytes). It also accepts a dword value which represents the number of bytes of hex digits to convert UNLESS the value dNum is 0. In this case the 2nd parameter is actually a 4-byte value to BE converted.

The method **hextostring** MUST make repeated calls to first method in order to handle the multiple byte conversion. After all the bytes have been converted, the method hexttostring will append a final NULL character added at the end of the constructed string.

Examples of how the hextostring method could be used:

```
0040400C    iX      DWORD     2C4E5A42h
00404010    strHex  BYTE      2C4AFC2DE7DF456AEEh
00404019    strOut  BYTE      20 dup (?)

;convert the string of hex digit bytes to a string of hex characters
        push    dword ptr 9             ;9 bytes to be converted
        push    offset strHex           ;need the address of the beginning of the string of bytes
        push    offset strOut           ;need the address of the string to store the hex characters
        call    hextostring             ; go and convert the entire string of bytes
        add     esp,12                  ;restore the stack pointer
```

your program crashed when I tried to run it

50

(If you can download what you dropped onto d2l & it executes, let me know)

```
;convert the value of iX to a string of characters
        push    dword ptr 0         ; 0 will indicate that the second parameter is actually a value
        push    iX                  ;the value that needs to be converted
        push    offset  strOut      ;need the address where the characters will be stored
        call    hextostring         ;convert the value to its equivalent characters
        add     esp,12              ;restore the stack pointer
```

2) Write a driver named **proj4XXX.asm** that will contain a new macro called **converthex** that can be used in BOTH of the above situations where the three parameters will be in the same order as they appear in the prototype definition, e.g. the first parameter is the address where the characters will be stored, the third parameter will be the number of bytes.

3) Make a folder **proj4XXX** that contains EVERY file that your program accesses. You do not have to include **kernel32.lib** or the file that contains your getstring and putstring methods unless you just want to. My batch file will automatically link to my own kernel32.lib and my own file that contains getstring and putstring.

```
;Programemr:     Brandon Buchanan
;Program:        proj4BWB.exe
;Course:         CSCI 2160-201
;Assignment:     Project 4
;Date:           November 22, 2011
;Purpose:
;
;     Convert hex values in memory
;        to a string of hex digits that
;           can be displayed to a screen
;

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

    extern  getchar:Near32,
            getcharecho:Near32,
            putchar:Near32,
            hextostring: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

    ;Variables
    strStorage      BYTE    21 dup(?)
    strToConvert    BYTE    'F434675A',0
    dNumToConvert   DWORD   4

    .code

_start:

; Macro for converting hex
converthex macro strToStoreChars, strHexBytes, dNum

    push ebx        ;stores dNum
    push ecx        ;stores strHexBytes
    push edx        ;stores strToStoreChars

    mov ebx,dNum                    ;ready the
    push ebx                        ;  number of bytes
    mov ecx,offset strHexBytes      ;ready the
    push ecx                        ;  bytes
    mov edx,offset strtoStoreChars  ;ready where to
    push edx                        ;  store characters
    call hextostring                ;convert the value to its equivalent characters
```

```
    add esp,12                      ;restore the stack pointer

    pop edx                 ;restore
    pop ecx                 ;the
    pop ebx                 ;registers

endm

;Main program
main PROC

    ;Convert hex digits to a string
    converthex strStorage,strToConvert,dNumToConvert

    ;end the program
    INVOKE ExitProcess,0
    PUBLIC _start


;*****************************************
;             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
```

You did not preserve eax

```
    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
```

```
;Program:    hexstring.asm
;Name:       Brandon Buchanan
;Date Due:   Nov 15, 2011
;Purpose:    --Converting a byte in hex to its equivalent String of hex characters

    .486
    .model Flat
    .code


;Procedure to convert a hex digit to an ASCII character
;
;  void hextochar(int &strToStoreCharacters, byte chr)
;
hextochar PROC

    push ebp
    mov ebp,esp
    push ebx

    mov ebx,[ebp+8]             ;address of string that store characters
    mov al, byte ptr [ebp+12]   ;character to convert

    shr al,4                    ;get the high order four bits
    cmp al,0Ah                  ;if it is a hex number
    jl  add30hToFirst           ;  go add 30h
    jmp add37hToFirst           ;if it hex letter, go add 37h

add30hToFirst:
    add al,30h
    jmp addFirstChar
add37hToFirst:
    add al,37h
addFirstChar:
    mov [ebx],al

    mov al, byte ptr [ebp+12]   ;restore original byte
    and al,00001111b            ;zero out 4 high order bits
    cmp al,0Ah                  ;if it is a hex number
    jl  add30hToSecond          ;  go add 30h
    jmp add37hToSecond          ;if it hex letter, go add 37h


add30hToSecond:
    add al,30h
    jmp addSecondChar
add37hToSecond:
    add al,37h
addSecondChar:
    mov [ebx+1],al

    pop ebx
```

```
    pop ebp
    ret

hextochar ENDP

;Procedure to convert hex digits to an
;  ASCII string of characters
;
;  void  hextostring (&strToStoreCharacters, &strHexBytes, dword dnum)
;
hextostring PROC

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

    mov edx,[ebp+8]      ;address of string to store converted characters
    mov ebx,[ebp+12]     ;string of hex digits to convert
    mov ecx,[ebp+16]     ;store the number of characters to convert
    xor esi,esi          ;clear the register, stores the string index

    cmp ecx,0            ;if the number of characters is 0,
    je convert4Bytes     ;  then just convert the 4 byte value

strLoop:
    cmp esi,ecx          ;if the loop is done
    jge done             ;  then stop the loop

    push edx             ;save edx before next instructions

    mov eax,[ebx+esi]    ;get hex digit to convert
    push ax              ;pass the hex digit to convert
    add edx,esi          ;pass the address to store the converted character
    push edx
    call hextochar
    add esp,8

    pop edx              ;restore edx
    inc edx              ;increment the index
    jmp strLoop          ;restart the loop

convert4Bytes:
    push ebx             ;convert the
    shr ebx,24           ;  first hex
    push bx              ;    character to
    push edx             ;       an ASCII
    call hextochar       ;         character
    add esp,6            ;
```

```
    pop ebx             ;

    inc edx             ;increment string's index pointer

    push ebx            ;convert the
    shr ebx,16          ;  second hex
                        ;    character to
                        ;      an ASCII
                        ;        character
    push bx             ;
    push edx            ;
    call hextochar      ;
    add esp,6           ;
    pop ebx             ;

    inc edx             ;increment string's index pointer

    push ebx            ;convert the
    shr ebx,8           ;  third hex
                        ;    character to
                        ;      an ASCII
                        ;        character
    push bx             ;
    push edx            ;
    call hextochar      ;
    add esp,6           ;
    pop ebx             ;

    inc edx             ;increment string's index pointer

                    ;convert the
                ;  fourth hex
                    ;    character to
    push bx         ;       an ASCII
    push edx        ;         character
    call hextochar  ;
    add esp,6       ;


done:

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

hextostring ENDP

END
```

; restore registers