CSCI2160Proj5F11AddingColumnsOfMatrices Name: Brandon Buchanan

DUE: 12/6/2011 by 5:00 PM dropped onto D2L - I MUST have the hardcode printed/stapled with this as the cover sheet submitted to me by 5:00 PM as well. I will NOT print any hardcode.

You can actually go ahead and begin the "normal" coding for your method. You can easily modify your routine to accommodate the higher-level syntax available with the PROC directive later.

Write an external method named addcolumns contained in a file **matrix.asm** which will form a vector consisting of the sums of the columns of a matrix. Name your driver **proj5XXX.asm** where XXX represents your initials. Use the notation from Chapter 8, and make appropriate use of the PROC high-level syntax where the registers that are used within the method appear after the USES statement, and the passed parameters are defined on the PROC header. Assume the STDCALL convention. The below data is in your driver.

```
M1          dword 20,30,40
            dword 10,5,7

M2          dword 25,-16, 80,27
            dword 37, 42, -20, 37
            dword 20, 10, 4,  5

M3    dword  20 dup(?)

INVOKE addcols,ADDR M1,2,3, ADDR M3
; display the values M1 as
; M1
;      20    30    40
;      10    5     7
;M3    30    35    47

INVOKE addcols, ADDR M2,3,4,ADDR M3

; display the values M2 as
; M2
;      25    -16   80    27
;      37    42    -20   37
;      20    10    4     5
;M3    82    36    64    69
```

Deliverables

Create a folder named Proj5XXX that you will use to contain ALL files for the project, including the kernel32.lib, and any other object files (you'll need ASCINT32 and INTASC32 and a way to print your strings). Zip the folder containing these files and your .exe file and drop onto D2L

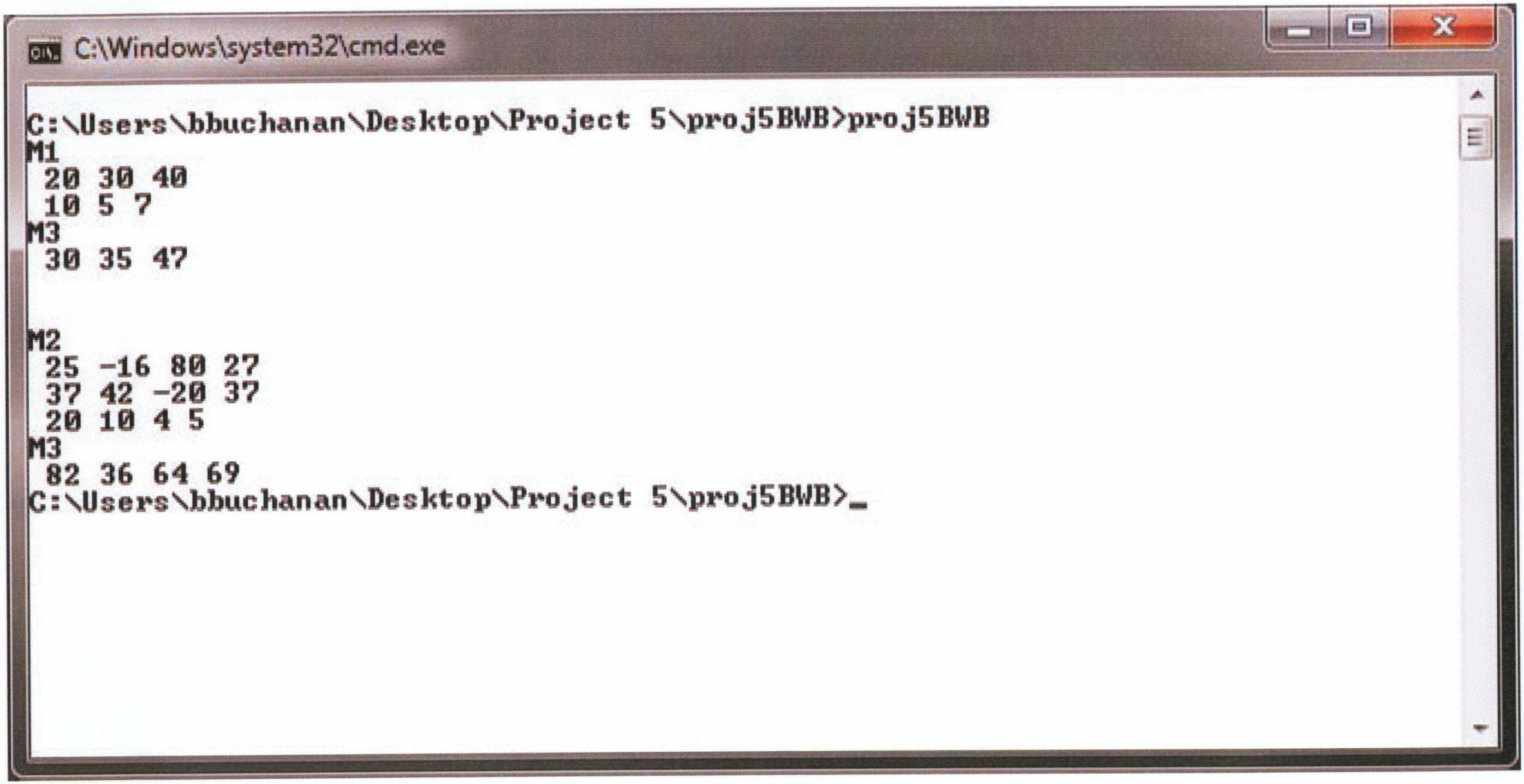
```
;Programemr:     Brandon Buchanan
;Program:        proj5BWB.asm
;Course:         CSCI 2160-201
;Assignment:     Project 5
;Date:           December 6, 2011
;Purpose:
;       Displaying a matrix and adding up each
;       column, then displaying the results in
;       a vector

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

    addcolumns PROTO stdcall, dMatrix:dword, dRows:dword, dColumns:dword, dVector:dword
    displayMatrix PROTO stdcall, dMatrix:dword, dRows:dword, dColumns:dword, strMatrix:dword

    .data

    M1      dword   20,30,40        ;M1 matrix
            dword   10,5,7          ;

    M2      dword 25,-16, 80,27     ;M2 matrix
            dword   37, 42, -20, 37 ;
            dword   20, 10, 4,  5   ;

    M3      dword  20 dup(?)        ;M3 vector

    CRLF byte 10,13,0               ;carriage return/line feed
    SPACE byte 20h,0                ;null terminiated string for a space
    strM1Prompt byte "M1",0         ;M1 prompt
    strM2Prompt byte "M2",0         ;M2 prompt
    strM3Prompt byte "M3",0         ;M3 prompt

    bString byte ?
    strM3 byte 20 dup(?)            ;stores the vector of converted ints to strings
    strM1Sum byte 20 dup(?)         ;stores the string of integers for M1
    strM2Sum byte 20 dup(?)         ;stores the string of integers for M2

    .code

_start:

;Main program
main PROC

puts strM1Prompt                                ;display M1 prompt
INVOKE displayMatrix,   ADDR M1,2,3,ADDR strM3  ;display M1 matrix
INVOKE addcolumns,      ADDR M1,2,3,ADDR M3     ;sum up columns in M1
puts CRLF
puts strM3Prompt
```

```
INVOKE displayMatrix,   ADDR M3,1,3,ADDR strM1Sum   ;display sum of M1

puts CRLF           ;separate M1 output
puts CRLF           ;  from M2 output clearly
puts CRLF           ;



puts strM2Prompt                                    ;display M2 prompt
INVOKE displayMatrix, ADDR M2,3,4,ADDR strM3        ;display M2 matrix
INVOKE addcolumns, ADDR M2,3,4,ADDR M3              ;sum up columns in M2
puts CRLF                                           ;display a new line
puts strM3Prompt                                    ;display M3 prompt
INVOKE displayMatrix,   ADDR M3,1,4,ADDR strM2Sum   ;display sum of M2




endProg:
    call endProgram


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

;Procedure that ends the program
endProgram PROC

    push ebp                        ;enter
    mov ebp,esp                     ;  the method

    INVOKE ExitProcess,0            ;end
    PUBLIC _start                   ;  program

    pop ebp                         ;  exit the
    ret                             ;    method

endProgram ENDP

main ENDP

; Procedure to display the contents of a matrix
displayMatrix PROC stdcall,
    dMatrix:dword,          ;the matrix to display
    dRows:dword,            ;number of rows in matrix
    dColumns:dword,         ;number of columns in matrix
    strMatrix:dword         ;where to store converted matrix

    mov esi,dMatrix         ;holds the pointer to the current pointed value
    xor ebx,ebx             ;holds the counter for the current column position
    xor ecx,ecx             ;holds the counter for the current row position
    xor edx,edx             ;holds current element to be converted to a string
```

```
    loopThroughRows:
        puts CRLF               ;display a new line
        puts SPACE              ;display a space for the new line

        loopThroughColumns:
            mov edx,[esi]       ;save the current element to be converted

            push edx            ;convert
            push offset bString ;  the value
            call intasc         ;    to a
            add esp,8           ;      string

            puts bString        ;display the converted value
            puts SPACE          ;display a space

            inc ebx
            add esi,type dword
            cmp ebx,dColumns            ;if done with the row,
            jnz loopThroughColumns      ;  skip ahead

        inc ecx                         ;increase the row counter
        xor ebx,ebx                     ;reset the column counter

        cmp ecx,dRows                   ;if not looped through all
        jne loopThroughRows             ;  rows, then continue loop


    ret

displayMatrix ENDP

END                             ;end of program
```

```
;Programemr:     Brandon Buchanan
;Program:        matrix.asm
;Course:         CSCI 2160-201
;Assignment:     Project 5
;Date:           December 6, 2011
;Purpose:
;
;   Calculate the sum of the columns of a matrix
;

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

    .data

    .code

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


; Find out the sum of the columns of a matrix
addcolumns PROC stdcall uses eax ebx ecx edx esi edi,
    dMatrix:dword,          ;matrix to sum columns for
    dRows:dword,            ;number of rows in the matrix
    dColumns:dword,         ;number of columns in the matrix
    dVector:dword           ;vector that holds the sums of columns

    mov esi,dMatrix         ;holds the pointer to the current pointed value
    mov edi,dVector         ;holds the pointer to the vector element being referenced
    xor ebx,ebx             ;holds how many bytes to move
                            ;  to reach the next row, same column
    xor ecx,ecx             ;ecx holds counter for the current row position
    xor edx,edx             ;edx holds which column is being tallied

    push dColumns           ;get the
    push dword ptr 4        ;  number of bytes
    call multiply           ;    that need to be moved
    add esp,8               ;      to reach the correct
    mov ebx,eax             ;        column in the next row

    ;start out at the first byte, move to the next row by moving the counter
    ;  an additional (number of columns * size of each member)
    ;  until the counter is equal to zero

    goToNextColumn:
        xor eax,eax             ;holds the sum of the column
        xor ecx,ecx             ;reset the current row position back to 0
        mov esi,dMatrix         ;restore the pointer
        mov edi,dVector
```

```
        push edx                ;preserve edx because of multiply

        push edx                ;get the
        push dword ptr 4        ;  number of bytes
        call multiply           ;    that need to be moved
        add esp,8               ;      to reach the correct
                                ;        column in the first row

        pop edx                 ;restore edx

        add esi,eax             ;move the pointer to the correct column
        add edi,eax             ;move the vector pointer to the correct column
        xor eax,eax             ;clear eax again

        sumColumn:              ;sums the column
            add eax,[esi]       ;add the current element to the sum so far
            add esi,ebx         ;increase the row pointer to the next row
            inc ecx             ;increase the row counter
            cmp ecx,dRows       ;if all rows haven't been visited,
            jnz sumColumn       ;  then continue the loop

        mov [edi],eax           ;save the sum in the new vector

        inc edx                 ;if all the columns have not
        cmp edx,dColumns        ;  been tallied, then move
        jnge goToNextColumn     ;    to the next column

    ret                         ;exit the method

addcolumns ENDP

; Procedure to multiply two integers together
multiply 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

multiply ENDP

END                             ;end of program
```


C:\Windows\system32\cmd.exe
C:\Users\bbuchanan\Desktop\Project 5\proj5BWB>proj5BWB
M1
20 30 40
10 5 7
M3
30 35 47
M2
25 -16 80 27
37 42 -20 37
20 10 4 5
M3
82 36 64 69
C:\Users\bbuchanan\Desktop\Project 5\proj5BWB>