SINE AND COSINE SUBROUTINES

OVERVIEW:
You developed a procedure for computing a fixed point sine function in Programming Assignment #4. In part 1 of this lab, you will convert that procedure into a subroutine which can be called from anywhere in your program. In part 2 you will implement a cosine function.

Part 1:

Your subroutine should:

1. include use of the MOVEM instruction to save ALL registers at the beginning of your subroutine and restore them at the end of the subroutine.
2. pass the input word to the subroutine ON THE STACK. Although other methods of parameter passing to a subroutine are allowed, only this one will be accepted for purposes of grading. Other methods will receive severe penalities.
3. return a result in the lab #4 format (one word) ON THE STACK

Your main program should:
1. set the stack pointer to the correct return address when the stack is used for parameter passing.
2. set the stack pointer to an appropriate initial value. THE DEBUGGER DOES NOT DO THIS AUTOMATICALLY FOR YOU!

Part 2:
Write a subroutine which will compute the cosine of an input angle.
HINT: Recall that cos(x)=-sine(x-90 degrees).
Write a cosine subroutine which will compute x-90 degrees, call your sine subroutine to compute sine(x-90 degrees), and set the appropriate sign. The parameter passing MUST be on the stack, i.e. the form of the cosine routine should be similar to that of the sine routine

WriteUp for Lab #5

1. how you initialized the stack and where it is in memory
2. pseudocode description of cosine routine (DO NOT include the sine function pseudo code)
3. mathematical basis for cosine function. Specifically list the trig identity used.
4. operation of sine routine. This should include a short paragraph which describes the process of parameter passing on the stack and the necessary operations your subroutine must do to correctly implement parameter passing on the stack. Your documentation MUST include a picture or drawing of the stack (i) before the subroutine is called, and (ii) after you push your parameters onto the stack, call the subroutine and save your parameters on the stack with a MOVEM. Be sure to explicitly indicate the location of the SP. A picture of the debugger screen showing the stack contents should be included. IF YOU CANNOT DO THIS PLEASE INDICATE WHY. This could be done through the debugger via the stack window (upper right corner), or by directly examining the memory using a Memory Display command. HINT: Remember the direction in which the stack grows!
5. operation of the cosine routine. Your documentation MUST include a picture or drawing of the stack (i) before the subroutine is called, and (ii) after you push your parameters onto the stack, call the cosine subroutine and save your parameters on the stack with a MOVEM, pass your parameters to the sine routine, call the sine function, and then save its registers with a MOVEM. Be sure to explicitly indicate the location of the SP. A picture of the debugger screen showing the stack contents should be included. IF YOU CANNOT DO THIS PLEASE INDICATE WHY.
6. A table of hex sine and cosine values for 10 randomly chosen digital angles plus the angles $6000 and $C000.
7. listing (assembler output) of your program

EXTRA CREDIT DOCUMENTATION (HANDED IN SEPARATELY)
This means that you have used a LINK and UNLK pair INSIDE each subroutine. Points will be not be given for programs which improperly use the LINK instruction. Your writeup for the extra credit will be IN ADDITION to the above writeup. I will grade the extra credit whereas Meng will grade the regular assignment.

1. Repeat 4 and 5 of the above write-up instructions being sure to explicitly show the location of both the frame pointer and the stack pointer.
2. Your subroutine includes a LINK and MOVEM instruction. Justify the order in which these instructions should occur in your subroutine.
3. Explain why a frame pointer is needed in addition to a stack pointer.
4. assembler listing of your program

>I do not understand the confusion but some people are unclear that
>Lab #5 requires you to implement both a sine function AND a cosine function
>subroutine. You should first implement the sine function subroutine using
>the program you wrote for lab #4. The cosine function should be
>implemented using an appropriate trig identify to re-write the cosine as an
>arguement of a sine function. Both the sine and cosine subroutines MUST
>pass parameters (both input and output) on the stack. Your algorithm for
>the sine was graded in lab #4. This lab will primarily grade you on your
>subroutine construction and parameter passing. Grading for the extra credit
>will be described below.
>
>WriteUp for Lab #5
>
>1. how you initialized the stack and where it is in memory
10 points
Look at the student's code. Some people improperly set up the stack. For example:

```
        ORG     $1000
START   DS.W    200
```

followed by

```
        LEA     START,SP
```

is not correct. The stack will use the memory below START in memory, NOT the memory above START as the student thinks. Take something like 4 points off for improper stack set-up

>2. pseudocode description of cosine routine (DO NOT include the sine
>function pseudo code)
10 points
See #3 below. Look at pseudocode as to how they set the minus sign for
Cosine(angle) = - Sine(angle-90 degrees)
Take off 4 points if they simply submitted the comments on their assembly code as pseudocode.

>3. mathematical basis for cosine function. Specifically list the trig
>identity used.
10 points
Be very careful of how the student implements the cosine function. Specifically, look for the trig formula which relates the cosine to the sine. The formula
Cosine(angle) = - Sine(angle-90 degrees)
is potentially a source of program error. Specifically, the issue is how the student implements the - Sine. The result of the Sine function is signed magniture convention which is not recognized by the 68000 math operations.
I would suggest checking the third and fourth quadrant results of students who used this formula.

>4. operation of sine routine. This should include a short paragraph which
>describes the process of parameter passing on the stack and the necessary
>operations your subroutine must do to correctly implement parameter passing
>on the stack. Your documentation MUST include a picture or drawing of the
>stack (i) before the subroutine is called, and (ii) after you push your
>parameters onto the stack, call the subroutine and save your parameters on
>the stack with a MOVEM. Be sure to explicitly indicate the location of the
>SP. A picture of the debugger screen showing the stack contents should be
>included. IF YOU CANNOT DO THIS PLEASE INDICATE WHY. This could be done
>through the debugger via the stack window (upper right corner), or by
>directly examining the memory using a Memory Display command. HINT:
>Remember the direction in which the stack grows!
>5. operation of the cosine routine. Your documentation MUST include a
>picture or drawing of the stack (i) before the subroutine is called, and
>(ii) after you push your parameters onto the stack, call the cosine
>subroutine and save your parameters on the stack with a MOVEM, pass your
>parameters to the sine routine, call the sine function, and then save its
>registers with a MOVEM. Be sure to explicitly indicate the location of the
>SP. A picture of the debugger screen showing the stack contents should be
>included. IF YOU CANNOT DO THIS PLEASE INDICATE WHY.

40 points total - 20 points for each routine
Take off 8 points per part if they have no drawing or diagram of the stack contents.

The student can use the debugger screen in place of a drawing or diagram. Take off 2 points for each part if they do not label their drawing, i.e. indicate what the numbers on the stack represent. It is very difficult to follow an unlabeled drawing. I would expect something which would indicate the size of the data pushed on the stack or the addresses of the various quantities.

Take off 4 points per part if they do not have a debugger screen. If they say they cannot do it then that is OK and we will take no points off. The explanation should count for 8 points per part.

>6. A table of hex sine and cosine values for 10 randomly chosen digital
>angles plus the angles $C000 and $C000.

1 points each

12x2=24 points, but I will only take off 20 points here.

Grade only the following unless there is another problem:

angle	sine	cosine
$6000	$2D41	$AD41
$C000	$C000	$0000

Both of these are correct with respect to sign and value.

>7. listing (assembler output) of your program
10 points, take off all 10 points if it is NOT an assembler listing