Assignment 0415
Assembly language programming is like flying a plane…to get better at it, you must log your hours…

Not for Submission
Review the notes from Dr. Toal’s web site as needed. Direct links can be found on the course web site. ABI and developer’s documents should also be quite handy.

For Submission
Write the following assembly language programs. Some of you may have fond memories of the first couple of programs :)
- A program that “makes change” for a total number of cents provided as a lone argument: the program displays the number of quarters, dimes, nickels, and pennies needed (make-change.asm).
- A program that states whether or not a given year is a leap year, according to the current algorithm: the program accepts the year as a lone argument, then states whether or not that year is a leap year (leap-year.asm).
- A program that works like the wc command: it accepts data from standard input and prints out the number of words, lines, and characters received. Use spaces, newlines, and tabs as your whitespace/word delimiters (my-wc.asm).
  Compare the size of your finished my-wc executable to the installed wc executable (invoke which wc to find out where wc lives). Make some educated guesses that explain why the file sizes differ (Hint: Invoke man wc), and write those up briefly in wc-comparison.txt.
  Yet another iteration of the calculator program; let’s call it calculator-plus-plus.c now. Make the following changes to calculator-plus.c from the last assignment (yes, that means if it had bugs, you should fix them):
  - Your BinaryOperation functions must all be implemented in assembly language. Put them in a file called calc-ops.asm.
    - Add the following new BinaryOperation functions: \text{mod} (\%), which returns the remainder after dividing the first argument by the second one, \text{gcd} (\text{gcd}), which calculates the greatest common divisor of the two arguments, and \text{power} (^), which raises the first argument to the second argument-th power (positive powers only — display a warning when ignoring the sign of a negative power).
    - Use Euclid’s algorithm for \text{gcd}:
      \[ \text{gcd}(x, y) = (y == 0) ? x : \text{gcd}(y, x \% y) \]
    - Observe how, if you wrote the original calculator-plus.c correctly, you’ll hardly need to change that code. The operation functions go away, of course, and your OpDefinition array gets two new elements. But that should be it; if you find yourself doing more to the C code, you may be doing something wrong.
  - This one’s a doozy, but if you get it, you will feel quite satisfied :) Write an “ASCII art” program that “plots,” in text form, either the \text{sin} or \text{cos} function from 0 to 2\pi (trig-art.asm). Start at \( x = 0 \) for the first line, then print a * in the column within which the value of the function resides. Assume a maximum width of 80 columns: i.e., if the value of the function is –1, you would put a * in the first column of the line, and if the value is 1, you would put a * in the 80th column. Increment \( x \) by \( \pi/45 \) for each line.
    You’ll need the \text{sin} or \text{cos} function from the standard C math library. Link to it by adding \(-lm\) to your \texttt{gcc} command:
    \[
    \texttt{gcc trig-art.o -lm}
    \]
    Use scalar doubles — a.k.a., 64-bit IEEE 754 for this program: refer to Volume 1, Chapter 11 of the Intel manuals. It is highly recommended that you write little practice programs first, to get the hang of things.
To get you started, here's a sample program that tabulates the values of \( \sin \) and \( \cos \) in increments of \( \pi/45\):

```assembly
section .text
header: db "theta sin(theta) cos(theta)", 10,
        db "------------------------------------", 10, 0
output: db "%10f %10f %10f", 10, 0
pi:     dq 3.141592653589793
delta:  dq 45.0 ; One line moves pi/delta radians.
two:    dq 2.0 ; For two pi.
main:   push rbp
        mov rdi, header ; Print a header.
xor rax, rax
        call printf
L0:     movsd xmm0, [radian]
call sin ; Result in xmm0.
movsd [sine], xmm0 ; Save to memory.
movsd xmm0, [radian]
call cos ; Ditto with cosine.
movsd [cosine], xmm0
; Display the results.
mov rdi, output
movsd xmm0, [radian]
movsd xmm1, [sine]
movsd xmm2, [cosine]
mov rax, 3 ; 3 vector registers!
call printf
movsd xmm0, [radian]
movsd xmm1, [pi]
divsd xmm1, [delta]
addsd xmm0, xmm1 ; radian += pi/delta
movsd [radian], xmm0
movsd xmm2, [p1]
mulsd xmm2, [two]
addsd xmm1, xmm2 ; 2*pi + pi/delta
comisd xmm0, xmm1
jc L0 ; CF is set if less than.
pop rbp
ret

section .data
radian: dq 0.0
sine:   dq 0.0
 cosine: dq 0.0
```

Commit your work (except for calculator-plus-plus) to CVS under the directory /homework/cmsi284/art-of-asm, using the given filenames. Since calculator-plus-plus consists of multiple sources and a Makefile, commit that separately, under /homework/cmsi284/calculator-plus-plus.