CMSI 284-01, -02
COMPUTER SYSTEMS ORGANIZATION/ SYSTEMS PROGRAMMING

Spring 2016

Assignment 0503b

This is it, time to put everything together. Congratulations on becoming a systems programmer **:)**

Outcomes

This assignment will affect your proficiency measures for outcomes *2a–2d*, *3a–3b*, and *4a–4f*.

For Submission

Write the following combination assembly language + C programs:

- An assembly language program that uses *reverse-range-in-place.c* to echo its command-line arguments, including the program name, but in reverse (*rev-echo.asm*).
- An assembly language program that uses the *madlib-by-numbers.c* code that you wrote in Assignment 0322 to perform the madlibs operation as a command (*madlibs.asm*). Let the first argument be the template (use double-quotes to accommodate a template with spaces) with the remaining arguments serving as the words to insert. Do *not* expect the word count as an argument—the program can (and should) calculate that on its own.
- A suite of three (3) integer calculation programs in C that use functions defined in a single assembly language file to perform the requested computations (*calc-ops.asm*). The *calc-ops.asm* "library" should implement a *gcd* function and two other integer operations of your choice (examples: *add*, *subtract*, *mod*, *power* [for positive exponents only, of course]). The C programs then accept arguments, process them, and finally call their corresponding assembly language function, displaying the returned result.
  - Include a text file with your submission called *build.sh* that provides the commands for building these three programs. You can use this file yourself while working.
  - Use Euclid's algorithm for your implementation of *gcd* (yes, in assembly language):

    ```
    gcd(x, y) = (y == 0) ? x : gcd(y, x % y)
    ```
- This one's a doozy, but if you get it, you will feel quite satisfied **:)** Write an "ASCII art" assembly language program that "plots," in text form, either the *sin* or *cos* function from 0 to 2π (*trig-art.asm*). Start at $x = 0$ for the first line, then print a character (your choice) 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 your chosen plot character in the first column of the line, and if the value is 1, you would put that character in the 80th column. Increment x by π/18 for each line of the output.

  You'll need the *sin* or *cos* function from the standard C math library. Link to it by adding `-lm` to your *gcc* command:

  ```
  gcc trig-art.o -lm
  ```

  Use *scalar doubles* — a.k.a., 64-bit IEEE 754 for this program: this means you will use one or more `xmm` registers to perform your calculations. You will also need to `cvtsd2si` instruction (look it up, you'll realize why you need it right away) to help position things.

To get you started, the next page contains a sample program that tabulates the values of *sin* and *cos* in increments of π/45. Note how this program has the same overall structure as the assigned *trig-art.asm* program; the differences are that your program only needs to compute one of these functions and displays a plotted character on each line rather than a tabular row of numbers. More or less, you can start with this program then eliminate extraneous code and replace the tabular-display instruction sequence with one that plots either the sine or cosine value.

```
        global  main
        extern  sin
        extern  cos
        extern  printf

        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, [pi]
        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 to folders called *rev-echo*, *madlibs*, *calculator*, and *trig-art*, respectively.