**Files to submit**: 64bitAdd.s
**Time it took Matthew to complete**: 15 mins (but I was super rusty when I did it)

- All programs must compile without warnings when using the -Wall and -Werror options
- Submit only the files requested
  - Do **NOT** submit folders or compressed files such as .zip, .rar, .tar, .targz, etc
- If submitting in a group on Grade Scope please make sure to mark your partner.
  - Only one of you has to submit there
- Your program must match the output exactly to receive credit.
  - Make sure that all prompts and output match mine exactly.
  - Easiest way to do this is to copy and paste them
- All input will be valid unless stated otherwise
- Print all real numbers to two decimal places unless otherwise stated
- The examples provided in the prompts do not represent all possible input you can receive.
- All inputs in the examples in the prompt are underlined
  - You don't have to make anything underlined it is just there to help you differentiate between what you are supposed to print and what is being given to your program
- If you have questions please post them on Piazza

(Time 15 min)Write an assembly program called **64bitAdd.s** that adds two 64 bit numbers together.

1. The first number will be referenced by the label **num1** and the second number will be referenced by the label **num2**.
2. The upper 32 bits of the sum should be placed in **EDX** and the lower 32 bits in **EAX**.
3. **AFTER** the last line of code that you wish to be executed in your program please place the label **done**.
   1. Make sure that there is an instruction after the done line and a new line after that instruction. If you don't your output won't match mine.
4. I have included a Makefile for you that will compile your program.
5. **IT IS OF VITAL IMPORTANCE THAT YOU NAME YOUR LABELS AS SPECIFIED AND MAKE THE APPROPRIATE AMOUNT OF SPACE FOR EACH VARIABLE!** I will be using gdb to test your code and if your labels do not match then the tests will fail. You must also make sure to include the done label AFTER the last line of code you want executed in your program so that I know where to set break points.
6. The following table shows how the numbers will be laid out in memory.

| | | |
|---|---|---|
| num1: | Upper 32 bits of num1 | Lower 32 bits of num1 |
| num2: | Upper 32 bits of num2 | Lower 32 bits of num2 |

7. This problem is much, much easier than the rest and you should not base the difficulty of the other problems off of this one