Upcoming schedule

So things we have to get done are
1. Put timers into the pthread-mult program
2. Make a program similar to pthread-mult, but just the serial multiplication so we can compare the values
3. Attempt to try a different parallelizing technique for pthread-mult using something similar to karatsuba's algorithm
4. Write our paper
5. Maybe do the atomic multiplication program (this can likely be cut)

As such, I think we need a general plan on when we ought to meet so we can get things done without crunch. Unfortunately I'm also busy due to other projects, but here are somethings that we can do

1. Monday 11/29 we should meet after class. I likely wont be there for the first hour but if we can do stuff from 3-6 then that would be great
2. Wednesday 12/1. I have a meeting at 3 that I need to prepare for, but given that it'll last only a half an hour, if we were to meet from 3:30-6 we could bust out a lot of stuff
3. Friday after class for whatever we need
4. Saturday from about 5-7/8 like we did last Saturday seemed to be fine. We can use this time to write the rough draft for our paper
5. Sunday, we can meet for an emergency amount of time if the paper is lacking
From there we can figure things out.

Outline for paper

1. Desire for parallelizing multiplication
    a. Further optimize fast multiplication algorithms
        i. Downsides of long multiplication
            1. Big O is slow O(n^2)
        ii. Karatsuba's algorithm
            1. First major improvement to multiplication
            2. Splitting up factor into smaller parts
            3. Significant big O decrease
        iii. Brief mention of Toom-Cook and Fourier transform (FFS algorithm)
    b. Large Numbers are typically used for
        i. Cryptography (generating really large numbers)
        ii. Cosmology (measuring really large astronomical values)
        iii. Important for solving some math problems where large numbers need to be checked, such as primes and 3n+1 conjecture
2. Difficulty of parallelizing multiplication
    a. Splitting up factors
        i. Must divide string first (use of hex)

    b. Bit shifting
        i. Takes up some of the cost that would be otherwise be saved from calculating individual partial sums
3. How we did our naive pthread program
    a. Use of the GNU library
        i. Mpz_t struct
            1. Can store an extremely large integers
    b. Give each task a part of one factor
        i. Calculate the substring
    c. Global of other factor
        i. mpz_t _init_set_str
    d. Compute partial sum
        i. Mpz_mult
    e. Bit shift over correct number of bits
        i. Mpz_mult_2exp function
    f. Add all values together(in mutex)
        i. Mpz_add
    g. Timing results (how it compares at various values)

Bibliography

Granlund, Torbj¨orn. “GNU MP 6.2.1.” The GNU Multiple Precision Arithmetic Library. Free Software Foundation. Accessed November 23, 2021. https://gmplib.org/manual.