Lyndon Engel CSC 314: Program 1 3/28/2016

For the start of this program I started with trying to reproduce the required unsigned multiply function. After rewriting the function line for line in Assembly I didn’t notice much of an improvement maybe like .1 to .01 times faster than the original unsigned multiply. After some more research and spit balling ideas with fellow class mates we came up with the following idea.

The idea for the algorithm was to only use sum to hold the information I needed. In order to do this I allocated a new array that was the size of ‘l’ plus one. This would allow me to store all needed data inside of it. I then created a pointer to the sum array which I used to fill sum. After this was done I used the unsigned multiply operation to multiply ‘r’ and the value of ‘l’. Once I had the lower bits from the multiply I added them to the current value in the sum array and then reinserted that solution into the array. I then stored the upper bits into next position of the sum array. I did this process until my counter was greater than the size of ‘l’. To end the function I put the sum array into register 0 and then returned. Below are my results.

|  |  |  |
| --- | --- | --- |
| Difference in time (seconds) | | |
| Original Code | Code With My Multiply | Percentage Faster |
| 1.290s | 1.020s | 126% |
| 1.300s | 1.060s | 125% |
| 1.320s | 1.160s | 114% |
| 1.290s | 1.050s | 123% |
| 1.310s | 1.020s | 128% |
| 1.320s | 1.110s | 119% |
| 1.330s | 1.040s | 128% |

After looking through the rest of the code when I was done with this function I noticed that the negate was unnecessary. This was mainly because the only function that called the unsigned multiply, negated that values before passing them to the function.

With some spare time before the program was due I decided to rewrite the bigint\_free() function in Assembly. I didn’t add anything special just your basic C to Assembly conversion. Which surprisingly speeded up the program by an average of 10%.

|  |  |  |  |
| --- | --- | --- | --- |
| Difference in time (seconds) | | | |
| Original Code | Code With My Multiply | Code With My Multiply & Free | Percentage Faster Compared to Original |
| 1.290s | 1.020s | .980s | 131% |
| 1.300s | 1.060s | .880s | 147% |
| 1.320s | 1.160s | .940s | 140% |
| 1.290s | 1.050s | .950s | 136% |
| 1.310s | 1.020s | .920s | 142% |
| 1.320s | 1.110s | .910s | 145% |