

MokujIN HEXADECAD (16 threads)

THE dummy command prompt x^y and $x*y$ dumper

D:\64\MokujIN_r5++_vs_GMP>timer64.exe "MokujIN_Hexadecad_ccc_730_64bit" 524288 524288 /stats

MokujIN, Multiplication of Integers, an OpenMP (multi-threaded) string multiplier, 16 threads enforced, written by Kaze, 2012-Nov-16, revision 5fix++.

```
omp_get_num_procs() = 4
omp_get_max_threads() = 4
```

```
Multiplying performance for operands 00,000,006 digits long (footprint: ~000,000KB, checksum: 6beb,c722): 36 MokujINs i.e. digits per second.
Multiplying performance for operands 00,000,012 digits long (footprint: ~000,000KB, checksum: 9e28,8598): 144 MokujINs i.e. digits per second.
Multiplying performance for operands 00,000,023 digits long (footprint: ~000,000KB, checksum: df58,ed82): 529 MokujINs i.e. digits per second.
Multiplying performance for operands 00,000,046 digits long (footprint: ~000,001KB, checksum: fbbf,9e27): 2,116 MokujINs i.e. digits per second.
Multiplying performance for operands 00,000,092 digits long (footprint: ~000,002KB, checksum: 5280,1546): 8,464 MokujINs i.e. digits per second.
Multiplying performance for operands 00,000,184 digits long (footprint: ~000,005KB, checksum: 3061,d39b): 33,856 MokujINs i.e. digits per second.
Multiplying performance for operands 00,000,367 digits long (footprint: ~000,011KB, checksum: 99a2,9b3c): 134,688 MokujINs i.e. digits per second.
Multiplying performance for operands 00,000,733 digits long (footprint: ~000,022KB, checksum: 71bd,9970): 537,289 MokujINs i.e. digits per second.
Multiplying performance for operands 00,001,465 digits long (footprint: ~000,045KB, checksum: c536,0c57): 2,146,225 MokujINs i.e. digits per second.
Multiplying performance for operands 00,002,929 digits long (footprint: ~000,091KB, checksum: f1f7,a243): 8,579,041 MokujINs i.e. digits per second.
Multiplying performance for operands 00,005,857 digits long (footprint: ~000,183KB, checksum: 5a3a,564c): 34,304,449 MokujINs i.e. digits per second.
Multiplying performance for operands 00,011,714 digits long (footprint: ~000,366KB, checksum: 464c,a182): 137,217,796 MokujINs i.e. digits per second.
Multiplying performance for operands 00,023,428 digits long (footprint: ~000,045KB, checksum: f9a9,2f30): 274,435,592 MokujINs i.e. digits per second.
Multiplying performance for operands 00,046,855 digits long (footprint: ~001,464KB, checksum: 86ba,47be): 365,898,504 MokujINs i.e. digits per second.
Multiplying performance for operands 00,093,710 digits long (footprint: ~002,928KB, checksum: b194,1027): 337,752,465 MokujINs i.e. digits per second.
Multiplying performance for operands 00,187,419 digits long (footprint: ~005,856KB, checksum: 97dc,040c): 334,532,205 MokujINs i.e. digits per second.
Multiplying performance for operands 00,374,838 digits long (footprint: ~011,713KB, checksum: 13e5,96d7): 332,159,636 MokujINs i.e. digits per second.
Multiplying performance for operands 00,749,676 digits long (footprint: ~023,427KB, checksum: 8ccc,73c7): 334,333,197 MokujINs i.e. digits per second.
Multiplying performance for operands 01,499,351 digits long (footprint: ~046,854KB, checksum: 8db2,fd7c): 334,481,985 MokujINs i.e. digits per second.
Dumping the result to 'MokujIN.txt' ... OK
Total Time: 8,964 second(s).
```

```
Kernel Time = 0.640 = 0%
User Time = 35667.000 = 397%
Process Time = 35667.640 = 397% Virtual Memory = 473 MB
Global Time = 8964.854 = 100% Physical Memory = 56 MB
```

The main loop of one of the 16 threads:

```
for (SF=1;0*DivideAndConquer; SF<=(0+1)*DivideAndConquer; SF++) {
for (QB=1;2*DivideAndConquer; QB<=(2+1)*DivideAndConquer; QB++) {
    CarryFlag = 0;
    Cycle = QB - (1+2*DivideAndConquer) + SF - (1+0*DivideAndConquer) + 1; // Here the subtractions are the initial values of SF&QB.
    TillerLeastSignificantDigit = LSDarray[ Multipland[(CandLength-SF+1)-1]-'0' ][ Multiplier[(ErLength-QB+1)-1]-'0' ];
    TillerMostSignificantDigit = MSDarray[ Multipland[(CandLength-SF+1)-1]-'0' ][ Multiplier[(ErLength-QB+1)-1]-'0' ];
    Result04[Cycle-1] = TillerLeastSignificantDigit + Result04[Cycle-1];
    if ( Result04[Cycle-1] >= 10 ) {
        Result04[Cycle-1] = Result04[Cycle-1] - 10;
        CarryFlag = 1;
    }
    NextNumPos = Cycle + 1;
    Result04[NextNumPos-1] = Result04[NextNumPos-1] + CarryFlag + TillerMostSignificantDigit;
    while (Result04[NextNumPos-1] >= 10) {
        Result04[NextNumPos-1] = Result04[NextNumPos-1] - 10;
        NextNumPos = NextNumPos + 1;
        Result04[NextNumPos-1] = Result04[NextNumPos-1] + 1;
    }
}
```

MokujINs stand for number of cycles of main loop of **MUL** function made per second.
At each iteration/cycle a digit vs digit multiplication is made.

Free download at: www.sahmayce.com/Downloads/MokujIN_r5+_vs_GMP.zip

My laptop 'Compressionette' i5-7200u gives 334 MegaMokujINs (4threads) using 64bit code.