Computer Organization and Architecture
Programming Assignment 3

Student Learning Outcomes:

1. Measure the performance of C functions
2. Apply optimization techniques to improve the performance of a given program
3. Evaluate the benefits and limitations of optimization techniques

Assignment description:

You are given the implementation of the matrix operation (multiplication) described in the C code below.

Multiplication: c[N][N] = a[N][N] x b[N][N]

```
     for(int i=0; i<N; i++)
          for(int j=0; j<N; j++){
               c[i][j] = 0;
               for(int k=0; k<N; k++)
                    c[i][j] += a[i][k] * b[k][j];
          }
```

Your job in this assignment consists in the following tasks:

1. measure the performance of the multiply function without any optimization for different sizes (N from 100 to 500) and data types (long and double).
2. optimize the performance of the function using different optimization techniques. You must use the following optimizations:
   a. Eliminating memory reference when possible
   b. Loop unrolling 2x1
   c. Loop unrolling 4x1
   d. Loop unrolling 8x1
   e. Loop unrolling 2x2
   f. Loop unrolling 4x4
   g. Loop unrolling 8x8

   The program **prog3.c** is provided to you with the implementation of the unoptimized matrix multiplication function. You need to modify it to write a function definition for each of the function prototypes at the beginning of the program. You also need to modify the main method to call the functions you defined and measure the execution time of each function.

3. Document each optimization technique you use by describing the modifications you made to the code, measuring the performance of the function after the optimization for different values of N and types of matrices, and discussing the effect of the optimization on the performance of the function.
4. Evaluate and compare the execution time of the unoptimized matrix multiplication function for the six versions shown in the textbook at page 645 (cache memory and spatial locality). Compare the six versions for different values of N (100 to 500) and different data types (long and double). The program shell **prog3b.c** is given for this part. You need to write the definition of the six functions and the main method.

To measure the execution time of a function, use the C library function **clock()** before and after the call to the function. The C code below shows an example on how to use **clock()** to measure the execution time of the function **add** in clock cycles.

```
#include <time.h>
#include <stdio.h>
int add(int x, int y);
int main(){
  int x=20, y=55, r;
  clock_t start = clock();
  r = add(x, y);
  clock_t end = clock();
  printf(“%d + %d = %d\n”, x, y, r);
  // execution time of function add
  printf(“Time = %ld\n”, (end - start));
  return 0;
}
int ad(int x, int y){
    return x+y;
}
```

Use the shell scripts **test.sh** and **testb.sh** to compile and run your two programs **prog3.c** and **prog3b.c** respectively. The two shells compile and run your two programs for N=100, 200, 300, 400, 500, and for data_t types long and double.

Your submission, for this assignment, should include the following:

1. Source files prog3.c and prog3b.c
2. Two-page maximum document to explain the optimizations you made, discuss the results obtained, and compare your results.

Important Note: Using **`repl.it`** (Web-based IDE) to write and test your code is recommended for this assignment.

Sample program results

Part 1: Optimization (**`prog3.c`**)

```
Type Long
Size    No-opt    No-mem    2x1    2x2     4x1     4x4     8x1     8x8
100     9187      6486      3328   5813    6125    6357    5794    5866
200     92190     56693     28898  56874   50198   51683   50099   49223
300     403091    268775    107621 290676  237531  228114  242626  229771
400     1188328   759705    291271 752459  701694  733246  682881  727684
500     2884970   2129087   604485 2007100 2084636 2056042 2123525 2006040

Type Double
Size    No-opt    No-mem    2x1    2x2     4x1     4x4     8x1     8x8
100     10690     7066      3345   5999    6357    6654    6844    6821
200     102296    57071     28236  60309   66505   63464   57473   60173
300     427484    284430    126563 254819  253142  265878  250808  266743
400     1184172   848805    304954 787059  738972  701847  712346  675779
500     2951751   2152340   632041 2036505 2065216 2057845 2006131 2008488
```

Part 2: Cache memory and spatial locality (**`prog3b.c`**)

```
Type Long
Size    ijk        jik       jki       kji       kij       ikj
100     2260      2627      2307      1962      3442      2078
200     28818     22508     22406     26224     16420     15783
300     110705    79956     88343     92746     51045     50243
400     296118    207521    240231    245071    119797    105541
500     613771    426652    464198    613752    264766    239939

Type Double
Size    ijk       jik       jki       kji       kij       ikj
100     3774      3958      2659      1944      1662      2037
200     35609     30950     20775     20710     12414     12210
300     131742    113687    103053    89461     56123     52259
400     386749    264096    263575    231320    117516    113767
500     656358    490795    426404    430864    222476    220977
```