Exercise 3

This exercise has total 6 points. Each point amounts to one percent of the total module mark. You are not allowed to use any external library in your code, doing so will receive a mark of 0.

All assessment deadlines in this module are strict. Late submissions will get a mark of 0. All submissions must work on the Linux lab machines as specified.

Modern computers provide hardware-level support for accelerating high-performance applications. Developing software applications that can exploit the potential of the platform requires a comprehensive understanding of the underlying platform, deep insight into the algorithms and above all creative thinking. The reward for mastering these techniques is the ability to develop optimized software applications that are orders of magnitude more efficient than their reference implementations.

Multiplication of two GF(2) square-matrices

In this exercise, you will be optimizing a program for multiplying two *square-matrices over GF(2)*. We have provided a reference implementation of the program.

GF(2) is the Galois field of the two elements: 0 and 1. Arithmetic operations in GF(2) are computed modulo 2, thus resulting in either 0 or 1. A matrix over GF(2) is a binary matrix containing only 0s and 1s.

The following example shows two GF(2) matrices A and B are multiplied to get a new matrix C, also in GF(2).

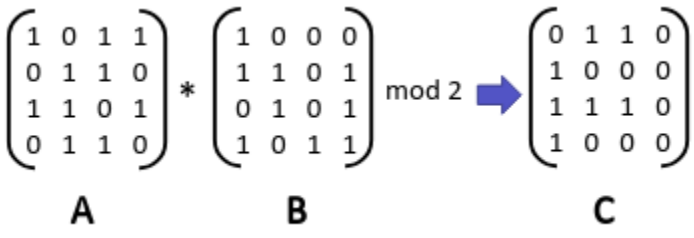

Fig. 1 Multiplication of two GF(2) matrices A and B. Result C is also a GF(2) matrix

E.g. Let us see the computation of the element C[0][0].

C[0][0] = 1•1 + 0•1 + 1•0 + 1•1 = 1 + 0 + 0 + 1 = 0 mod 2

The final result is computed by checking the parity of the sum expression: if there are even number of 1s then the result is 0, otherwise the result is 1.

Similarly, C[0][1] = 1•0 + 0•1 + 1•1 + 1•0 = 0 + 0 + 1 + 0 = 1 mod 2

<---------->

Odd number of 1s

In this exercise C=C+A*B will be computed instead of C=A*B.

Code submission

Write your optimized matrix multiplication code inside the function

void matmul_optimized(**int** n, **int*** A, **int*** B, **int*** C)

in the matmul_optimized.c file. You can have additional functions in this file.

Compile the code: gcc -O3 -Werror -Wall matmul.c -o matmul

Use Valgrind to check memory leaks and errors.

Upload the matmul_optimized.c file **ONLY**, any other form of submission will not be accepted.

Testing your code:

We will evaluate performance of your program on the UG04 computer 040. As all lab machines are not configured the same, we recommend that you run your **final solution** on this machine before submitting.

Marking scheme: We will evaluate performance of your program for 2048-by-2048 matrices.

- Your code must compile on the lab machines as above. If it fails to compile or produces any errors (including memory errors), it will be given 0 marks.
- If your code does not compute correct result then it will be given 0 marks.
- We expect a minimum 5 factor speedup with respect to the reference implementation.
- 0.5 marks are reserved for comments in the code. Put comments in the code to explain different operations/optimizations.
- If your code achieves a speedup factor greater than 140, then you get 5.5 points.
- If your code achieves a speedup factor greater than 100 but less than 140 then you get 5 points.
- If your code achieves a speedup factor greater than 70 but less than 100, then you get 4.5 points.
- If your code achieves a speedup factor greater than 40 but less than 70, then you get 3 points.
- If your code achieves a speedup factor greater than 10 but less than 40, then you get 1 point.
- If your code achieves a speedup factor greater than 5 but less than 10, then you get 0.5 points.

We remind you that plagiarism of submissions will be detected and handled in accordance with the University's policies.