UNIVERSIDADE FEDERAL RURAL DO SEMIÁRIDO
CURSO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMAÇÃO DE COMPUTADORES

TRABALHO PRÁTICO 1

Joãozinho, apesar de estar no início do curso de computação, andou pesquisando e descobriu que os processadores mais novos da Intel e AMD possuem suporte a instruções do tipo *Single Instruction Multiple Data* (SIMD). Com este tipo de instrução é possível fazer uma operação sobre múltiplos valores de uma única vez. Pode-se, por exemplo, somar quatro valores com outros quatro em uma única instrução.

Para que as operações possam ser realizadas, o primeiro passo é colocar os primeiros 4 valores em um único bloco de memória (xxma), e os outros 4 valores em um outro bloco (xxmb). Depois, uma função especial deve ser chamada para receber ambos os blocos de 4 valores e realizar a operação sobre todos eles. O resultado de uma operação SIMD pode ser visualizado pelo exemplo a seguir:

	32 bits				
	8 bits	8 bits	8 bits	8 bits	
xxma	2	10	15	30	+
xxmb	1	5	8	2	
	=				
soma	3	15	23	32	
	primeiro	segundo	terceiro	quarto	

Joãozinho ainda está começando a programar e não conhece tudo que é necessário para usar as instruções SIMD reais, mas ele já sabe o suficiente para simular o funcionamento delas usando o que aprendeu sobre os **tipos de dados inteiros** e as **operações bit a bit** que a linguagem C++ oferece.

Ajude Joãozinho a construir um programa que simule o funcionamento das operações SIMD criando uma biblioteca (simd.h e simd.cpp) que forneça as seguintes funções:

- **Armazena**: deve receber 4 valores na faixa de 0 a 255 (8 bits) e retornar um valor de 32 bits contendo os 4 valores.
- **Primeiro**: deve receber um valor de 32 bits e retornar o valor armazenado no primeiro bloco de 8 bits.
- **Segundo**: deve receber um valor de 32 bits e retornar o valor armazenado no segundo bloco de 8 bits.
- **Terceiro**: deve receber um valor de 32 bits e retornar o valor armazenado no terceiro bloco de 8 bits.
- **Quarto**: deve receber um valor de 32 bits e retornar o valor armazenado no quarto bloco de 8 bits.
- **Soma**: deve receber dois valores de 32 bits e retornar um valor de 32 bits com o resultado das somas individuais de cada valor de 8 bits.
- **Mult**: deve receber dois valores de 32 bits e retornar um valor de 32 bits com o resultado da multiplicação individual de cada valor de 8 bits.

Já que Joãozinho está manipulando bits em blocos de memória, ele achou que seria legal brincar também com criptografia de dados, construindo uma função para transformar uma cadeia de 32 bits em outra, com alguns bits modificados, de forma que um observador externo não consiga ver o número original.

Ele poderia usar este sistema de criptografia, por exemplo, para enviar o resultado da soma e multiplicação criptografados pela rede para que apenas o destinatário fosse capaz de decodificar os dados recebidos, sem que nenhum intruso ao canal de comunicação fosse capaz de conhecer os números enviados.

Para atingir esse objetivo construa as seguintes funções como parte de outra biblioteca (cripto.h e cripto.cpp):

- **Codificar**: transforma um valor de 32 bits em outro, alterando 6 posições aleatórias da cadeia original. Se o bit testado estiver ligado (igual a 1), ele deve ser desligado (para zero), e vice-versa.

Valor Original							
Valor Codificado	P1	P2	P3	P4	P5	P6	00
32 bits	5 bits	5 bits	5 bits	5 bits	5 bits	5 bits	2 bits
64 bits							

  Para poder decodificar a cadeia, será necessário guardar as posições modificadas. Isso pode ser feito guardando cada posição em 5 bits (um valor de 0 a 31), como mostrado na imagem acima. Como temos 6 valores de 5 bits para guardar, sobrarão 2 bits não utilizados no segundo bloco de 32 bits. A função codificar deve **retornar um valor de 64 bits** contendo o valor codificado e as posições alteradas.

- **Decodificar**: recebe um valor de 64 bits, gerado pela função codificar, e retorna um valor de 32 bits com o valor original, decodificado a partir do valor armazenado nos primeiros 32 bits e das 6 posições indicadas nos 32 bits seguintes.

64 bits							
32 bits	5 bits	5 bits	5 bits	5 bits	5 bits	5 bits	2 bits
Valor Codificado	P1	P2	P3	P4	P5	P6	00
Valor Original Decodificado							

- **LigarBit**: recebe um valor de 32 bits e a posição do bit a ser ligado e retorna o valor de 32 bits resultante da modificação do bit.

- **DesligarBit**: recebe um valor de 32 bits e a posição do bit a ser desligado e retorna o valor de 32 bits resultante da modificação do bit.

- **TestarBit**: recebe um valor de 32 bits e a posição do bit a ser testado, retornando um booleano (true ou false) para indicar se o bit está ou não ligado.

Utilize as funções das duas bibliotecas (SIMD e Cripto) em um programa que leia do usuário 8 valores inteiros e mostre os resultados como no exemplo abaixo. Observe que os valores foram digitados e exibidos sempre com 3 dígitos. Tanto o usuário ao digitar, quanto o programa ao exibir, completaram com zeros a esquerda os valores que possuíam menos de 3 dígitos.

```
[002,010,015,030]
[001,005,008,002]

Operandos em 32 bits = 34213662
Operandos em 32 bits = 17106946

Soma em 32 bits = 51320608
Mult em 32 bits = 36862012

[003,015,023,032] = Somas
[002,050,120,060] = Multiplicações

Soma Cripto 64 bits = 4832086673082564784
Mult Cripto 64 bits = 177111792737387716

Valor Codificado  = 1125057850 (9 3 30 4 1 12)
Soma Decodificada = 51320608

Valor Codificado  = 41237052 (22 14 18 16 9 17)
Mult Decodificada = 36862012
```

8 valores lidos do usuário

Dois valores de 32 bits cada um representando 4 valores da entrada

Resultado da soma e multiplicação obtido pelas funções SIMD

Resultado da soma e multiplicação separado em 4 valores

Soma e multiplicação criptografados através da função Codificar

Codificação da soma em 32 bits e resultado da Decodificação

Codificação da multiplicação em 32 bits e resultado da Decodificação

INSTRUÇÕES

1) Não é permitido usar variáveis globais

2) A entrada e saída de dados só podem ser realizadas no arquivo principal, que deve conter a função `main()` e incluir os arquivos de cabeçalho `cripto.h` e `simd.h`.

3) O programa não deve exibir nenhuma mensagem solicitando a entrada de dados. Os resultados deverão ser exibidos como no exemplo acima. Espaços e linhas em branco devem ser usados para separar os valores.

4) Considere que os números fornecidos pelo usuário não gerarão resultados maiores que 255, ou seja, os resultados das operações de soma e multiplicação cabem em 8 bits.

5) As funções Soma e Mult devem fazer uso das funções Primeiro, Segundo, Terceiro, Quarto e Armazena. As funções Codificar e Decodificar devem fazer uso das funções TestarBit, LigarBit e DesligarBit.

6) O trabalho final deve estar contido em 5 arquivos: principal.cpp, crypto.h, cripto.cpp, simd.h e simd.cpp.

ENTREGA DO TRABALHO

Grupos: Trabalho individual

Data da entrega: 28/03/2022 (até a meia noite)

Valor do Trabalho: 3,0 pontos (na 1a Unidade)

Forma de entrega: enviar apenas os arquivos fonte (.cpp) e os arquivos de inclusão (.h) compactados no formato **zip** através da tarefa correspondente no SIGAA.

O não cumprimento das orientações resultará em **penalidades:**

- Programa não executa no Visual Studio 2022 (3,0 pontos)
- Programa contém partes de outros trabalhos (3,0 pontos)
- Atraso na entrega (1,5 pontos por dia de atraso)
- Arquivo compactado em outro formato que não zip (0,5 ponto)
- Envio de outros arquivos que não sejam os .cpp e .h (0,5 ponto)
- Programa sem comentários e/ou desorganizado (0,5 ponto)