**Universidade Federal do Rio Grande do Norte**
**Instituto Metrópole Digital**
**IMD1116 - COMPUTAÇÃO DE ALTO DESEMPENHO - T01 (2025.1)**
**Tarefa 2: Pipeline e vetorização**

Docente: SAMUEL XAVIER DE SOUZA
Discente: Iago Gabriel Nobre de Macedo (20220037927)
Link do repositório da atividade no Github: https://github.com/IagoGMacedo/IMD1116-Computacao-de-Alto-Desempenho/blob/main/exercicios/pipelines_e_vetorizacao/Main.c

## Implementação dos casos

Para teste do cenário descrito, implementei os três algoritmos propostos e compilei o programa com diferentes níveis de otimização. Em seguida, medi os tempos de execução para cada caso e organizei os resultados em uma tabela para análise comparativa.

**Resultados obtidos**

| Caso | O0 (programa) | O2 (programa_02) | O3 (programa_03) |
|---|---|---|---|
| Caso 1 (Inicialização) | 6.476591 | 3.714004 | 3.728770 |
| Caso 2 (Soma dependente) | 11.089780 | 0.000001 | 0.000000 |
| Caso 3 (Soma independente) | 0.842914 | 0.000000 | 0.000000 |

Os resultados estão em segundos para preservar as casas decimais. A quantidade de elementos N para inicialização e cálculo da soma foi de 1000000000 (1 bilhão).

## Considerações

Quanto às otimizações, podemos corroborar a tese de que a otimização impacta significativamente o desempenho dos laços implementados. É possível perceber que o caso 3 (soma independente) consegue ser mais rápido visivelmente nos programas O0 e O2. No caso do O0, o tempo de execução foi de 0.842914, enquanto no O2 e O3 os tempos foram praticamente zero, evidenciando a eliminação da dependência entre as iterações por meio da otimização do compilador. No primeiro caso otimizado (O2), essa diferença já se torna evidente, e no segundo caso otimizado (O3), o tempo de execução permanece 0.000000, sem qualquer variação perceptível.

Outro ponto interessante é o tempo de inicialização (caso 1) em cada programa. Podemos perceber que os programas O2 e O3 obtêm um resultado quase 50% menor do que o primeiro programa sem otimização. O tempo de execução em O0 foi de 6.476591, enquanto para O2 e O3 os valores foram 3.714004 e 3.728770.

Nesse sentido, fica perceptível que as otimizações aplicadas pelos compiladores não apenas reduzem o tempo de execução ao eliminar dependências entre operações, mas também são capazes de reorganizar cálculos para maximizar o uso do paralelismo ao nível de instrução (ILP), resultando em melhorias expressivas no desempenho dos loops.

Quanto ao estilo de código e as dependências, Os resultados mostram que a forma como o código é escrito influencia diretamente a capacidade do compilador de otimizar a execução. O Caso 2, por exemplo, tinha um acoplamento forte entre as iterações, impedindo execuções em paralelo. O tempo de execução alto em O0 evidencia como uma abordagem sequencial rígida pode impactar negativamente o desempenho.

Já no Caso 3, o código foi escrito para minimizar as dependências internas, permitindo que o compilador realizasse otimizações mais agressivas. Isso reforça a importância de um estilo de codificação que favoreça a independência entre operações, sempre que possível.

Nesse sentido, entende-se por fim que a adoção de um estilo de programação impacta tanto quanto o próprio nível de otimização de um compilador.