Atividade 6

Vetorização

Integrantes :
Felipe Seiji Momma Valente 12543700
Fernando Gonçalves Campos 12542352
Thiago Shimada 12691032

Descrição da Atividade

Para esta atividade, o grupo optou por utilizar um código em C para aplicar um borramento (blur) em uma imagem. O borramento foi implementado por meio de convolução, utilizando um kernel 3x3 que combina cada pixel com seus vizinhos. Embora o kernel seja bidimensional, o filtro utilizado é separável, permitindo que a convolução 2D seja decomposta em duas convoluções 1D: uma horizontal e outra vertical. Essa abordagem reduz o número total de operações e facilita a vetorização.

O código percorre toda a matriz de pixels, aplicando o filtro pixel a pixel, o que envolve diversas operações de soma e multiplicação. A convolução horizontal é adequada à vetorização, pois os pixels de uma linha estão armazenados de forma contínua em memória. Na implementação vetorizada, utilizamos intrinsics como _mm256_loadu_ps, _mm256_add_ps e _mm256_mul_ps, possibilitando processar oito pixels por iteração.

Para comparar as duas versões (escalar e vetorizada), o filtro foi executado 100 vezes consecutivas, diminuindo variações nas medições. O tempo total de execução foi obtido utilizando a função clock(). Além da análise de desempenho, também calculamos a diferença absoluta máxima entre as imagens produzidas pelas duas versões, obtendo valores muito próximos de zero, o que confirma que a vetorização não alterou a precisão dos resultados.

A compilação foi realizada utilizando a flag -O0, garantindo que nenhuma otimização automática do compilador fosse aplicada. Essa decisão permite isolar o impacto das instruções vetoriais e evita que o compilador substitua o código escalar por uma versão vetorizada. Assim, o desempenho observado decorre exclusivamente da vetorização manual realizada pelo grupo.

O código pode ser encontrado no seguinte link: https://github.com/Pones2/SC0951-Desenvolvimento-de-Codigo-Otimizado/tree/main/Atividade%206

Dados Obtidos

Os dados foram obtidos executando cada iteração do código 100x e variando a flag entre : -O3, -O2, -O1 e O0.

flag -O3:

- Tempo ESCALAR: 0.333000 s
- Tempo AVX: 0.294000 s
- Speedup : 1.13x

flag -O2:

- Tempo ESCALAR: 0.588000 s
- Tempo AVX: 0.440000 s
- Speedup : 1.34x

flag -O1:

- Tempo ESCALAR: 0.622000 s
- Tempo AVX: 0.433000 s
- Speedup : 1.44x

flag -O0:

- Tempo ESCALAR: 3.431000 s
- Tempo AVX: 2.471000 s
- Speedup : 1.39x

Conclusão

A análise dos dados evidencia duas coisas principais, primeiramente a vetorização faz um speedup considerável, variando de 10% a 44%, evidenciando como o uso dessa instrução contribui muito para a velocidade do código. Por fim, também fica claro como nas flags mais otimizadas(O3), provavelmente já se tem o uso de alguma vetorização, pois o speedup foi muito menor quando comparamos com as flags (O0) de menores atualizações automáticas do compilador.