2º Exercício Prático
Desenvolvido no Laboratório

Objetivo

Desenrolar laços e observar o impacto no desempenho

Materiais

1. Compilador GCC
2. Arquivo Unroll.c
3. Imagem utilizada no experimento da aula passada: Lapis.ppm

Desenvolvimento

Antes de iniciar os experimentos leia o tópico 3.4.1.6 *Loop Unrolling* no *INTEL, Intel® 64 and IA-32 Architectures Optimization Reference Manual* (https://software.intel.com/content/dam/develop/external/us/en/documents-tps/64-ia-32-architectures-optimization manual.pdf).

Executando o programa original

1. Compile o programa Unroll.c
    - gcc Unroll.c -o Unroll
2. Execute o programa
    - ./Unroll
3. Visualize a imagem LapisSai.ppm, abra o arquivo Unroll.c e analise o que a função processa() faz
4. Anote o tempo medido.

   Qual foi o tempo observado na execução?

   Tempo 1: _____ s

   O que o programa faz?

Otimizando o código

1. Abra o arquivo Unroll.c em um editor de texto
2. Procure a função processa()

```
void processa(struct Pixel img[ALTU_IMG][LARG_IMG],
              struct Pixel imgSai[ALTU_IMG][LARG_IMG]) {
  int i, j;
  for (i = 0; i < ALTU_IMG; i++) {
    struct Pixel a, b, c, d, e, f, g, h, x;
    for (j = 0; j < LARG_IMG; j += 1) {
```

```
      if (j % 2 == 0) {
        a = img[i][j];
      } else {
        x = img[i][j];
        a.r = (a.r + x.r) / 2;
        a.g = (a.g + x.g) / 2;
        a.b = (a.b + x.b) / 2;
        imgSai[i][j / 2] = a;
      }
    }
  }
}
```

3. Evitar desvios (*if*'s) ajuda na execução pois instruções no *pipeline* não são perdidas nos desvios não previstos corretamente. Desenrole o laço de forma que, a cada laço do *for* mais interno, dois pixels da imagem de entrada sejam utilizados

```
void processa(struct Pixel img[ALTU_IMG][LARG_IMG],
              struct Pixel imgSai[ALTU_IMG][LARG_IMG]) {
  int i, j;
  for (i = 0; i < ALTU_IMG; i++) {
    struct Pixel a, b, c, d, e, f, g, h, x;
    for (j = 0; j < LARG_IMG; j += 2) {
      a = img[i][j];
      x = img[i][j + 1];
      a.r = (a.r + x.r) / 2;
      a.g = (a.g + x.g) / 2;
      a.b = (a.b + x.b) / 2;
      imgSai[i][j / 2] = a;
    }
  }
}
```

Certifique-se que a imagem gerada é a mesma, indicando que as alterações não afetaram o resultado.

Qual foi o tempo observado na execução?

Tempo 2: _____ s

Otimizando o código desenrolando mais um pouco o *for*

1. Desenrolar o *for* amortiza o tempo gasto no gerenciamento dele. Desenrole o laço mais uma vez de forma que, a cada laço do *for* mais interno, quatro pixels da imagem sejam processados

```
void processa(struct Pixel img[ALTU_IMG][LARG_IMG],
              struct Pixel imgSai[ALTU_IMG][LARG_IMG]) {
  int i, j;
  for (i = 0; i < ALTU_IMG; i++) {
    struct Pixel a, b, c, d, e, f, g, h, x;
    for (j = 0; j < LARG_IMG; j += 4) {
```

```
      a = img[i][j];
      x = img[i][j + 1];
      a.r = (a.r + x.r) / 2;
      a.g = (a.g + x.g) / 2;
      a.b = (a.b + x.b) / 2;
      imgSai[i][j / 2] = a;

      b = img[i][j + 2];
      x = img[i][j + 3];
      b.r = (b.r + x.r) / 2;
      b.g = (b.g + x.g) / 2;
      b.b = (b.b + x.b) / 2;
      imgSai[i][(j + 2) / 2] = b;
    }
  }
}
```

Certifique-se que a imagem gerada é a mesma indicando que as alterações não afetaram o resultado.

Qual foi o tempo observado na execução?

Tempo 3: _____ s

Otimizando o código juntando os armazenos

1. Juntar os armazenamentos abre espaço para que o processador possa, em apenas uma operação de gravação na memória, armazenar vários dados próximos. Junte os armazenamentos feitos na matriz da imagem de saída no final do *for* desenrolado.

```
void processa(struct Pixel img[ALTU_IMG][LARG_IMG],
              struct Pixel imgSai[ALTU_IMG][LARG_IMG]) {
  int i, j;
  for (i = 0; i < ALTU_IMG; i++) {
    struct Pixel a, b, c, d, e, f, g, h, x;
    for (j = 0; j < LARG_IMG; j += 4) {
      a = img[i][j];
      x = img[i][j + 1];
      a.r = (a.r + x.r) / 2;
      a.g = (a.g + x.g) / 2;
      a.b = (a.b + x.b) / 2;

      b = img[i][j + 2];
      x = img[i][j + 3];
      b.r = (b.r + x.r) / 2;
      b.g = (b.g + x.g) / 2;
      b.b = (b.b + x.b) / 2;

      imgSai[i][j / 2] = a;
      imgSai[i][(j + 2) / 2] = b;
    }
```

```
    }
}
```

Certifique-se que a imagem gerada é a mesma indicando que as alterações não afetaram o resultado.

Qual foi o tempo observado na execução?

Tempo 4: _____ s

Otimizando o código mais um pouco

Usando a última otimização, desenrole novamente para oito e dezesseis pixels, anotando o tempo de execução. Note que as variáveis c, d, e, f, g e h já foram declaradas nas rotinas processa() nas otimizações anteriores.

Quais foram os tempos observados na execução?

Tempo 5 (8 pixels por iteração do *for*): _____ s

Tempo 6 (16 pixels por iteração do *for*): _____ s

Avaliando os resultados

Faça um gráfico com os tempos observados (abscissa: número de atribuições no laço mais interno, ordenada: tempos medidos).

Envie a avaliação dos resultados como descrito no arquivo “Avaliacao Dos Resultados.pdf” na atividade desta semana, incluindo os tempos medidos e o gráfico.

Conclusão

A união de alguns motivos podem justificar o comportamento observado no gráfico. Discuta por que a sobrecarga do controle do *for* é uma delas.