Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №7

Оптимизация кода на языке С

Выполнил: Студент: гр. 053502
Герчик Артем Вадимович

Руководитель: ст. преподаватель
Шиманский В.В.

Минск 2021

СОДЕРЖАНИЕ

1. Условие задания

Это задание связано с оптимизацией кода, интенсивно использующего память. Обработка изображений предлагает множество функций, которым может быть полезна оптимизация. В этой лабораторной работе, мы рассмотрим две операции по обработке изображений: rotate, которая переворачивает изображение против часовой стрелки на 90°, а также smooth, которая “сглаживает” или “размывает” изображение.

Оптимизация поворота

В этой части, вы будете оптимизировать rotate, чтобы достичь как можно более низкого показателя CPE. Вы должны скомпилировать driver, а затем запустить его с соответствующими аргументами, чтобы протестировать ваши реализации.

Важно! Замените значения макроопределний в строках 15-19 файла config.h значениями из строки Your CPEs для функции naive_rotate. Эти значения будут использоваться для вычисления ускорения ваших реализаций.

Оптимизация сглаживания

В этой части, вы будете оптимизировать smooth, чтобы достичь как можно более низкого показателя CPE.

Важно! Замените значения макроопределний в строках 27-31 файла config.h значениями из строки Your CPEs для функции naive_rotate. Эти значения будут использоваться для вычисления ускорения ваших реализаций.

Правила написания кода:

Вы можете написать любой по вашему желанию код в рамках следующих правил:

• Он должен быть написан на ANSI Си. Вы не можете использовать встроенные операторы ассемблерных языков.

• Он не должен мешать механизму измерения. Не допускается, чтобы код выводил любую постороннюю информацию.

Вы можете модифицировать код только в файле kernels.c. Вам разрешено определять макросы, глобальные переменные и другие процедуры в этих файлах.

2. Реализация

2.1 Изменения в файле config.h

```
/************************************************************
 * config.h - файл, в котором надо заменить значения CPE
       для вашей машины для наивных реализаций
       оптимизируемых функций
 ************************************************************/
#ifndef _CONFIG_H_
#define _CONFIG_H_

/*
 * Заполните значениями CPE для базовой (наивной) версии функции
 * поворота изображения на вашей машине.
 * Запустите ./driver, чтобы получить приведенные значения и
 * впишите вместо предложенных.
 */
#define R64    1.4
#define R128   1.5
#define R256   3.7
#define R512   5.6
#define R1024  6.3

/*
 * Заполните значениями CPE для базовой (наивной) версии функции
 * сглаживания изображения на вашей машине.
 * Запустите ./driver, чтобы получить приведенные значения и
 * впишите вместо предложенных.
 */
#define S32   39.6
#define S64   42.1
#define S128  42.3
#define S256  42.4
#define S512  42.9


#endif /* _CONFIG_H_ */
```

2.2 Оптимизация rotate

P.S. Концепция одинакова с нативным, надо сделать как можно больше итераций во втором цикле, до перехода.

```
void rotate(int dim, pixel *src, pixel *dst)
{
   int i, j;
   int extension = dim*dim;

   dst += extension  - dim;
   for (i = 0; i < dim; i += 32)
   {
         for (j = 0; j < dim; j++)
         {
            *dst = *src;
            src += dim;

            *(dst + 1)=*src;
            src += dim;

            *(dst + 2)=*src;
            src += dim;

            *(dst + 3)=*src;
            src += dim;

            *(dst + 4)=*src;
            src += dim;

            *(dst + 5)=*src;
            src += dim;

            *(dst + 6)=*src;
            src += dim;

            *(dst + 7)=*src;
            src += dim;
```

```
*(dst + 8)=*src;
src += dim;

*(dst + 9)=*src;
src += dim;

*(dst + 10)=*src;
src += dim;

*(dst + 11)=*src;
src += dim;

*(dst + 12)=*src;
src += dim;

*(dst + 13)=*src;
src += dim;

*(dst + 14)=*src;
src += dim;

*(dst + 15)=*src;
src += dim;

*(dst + 16)=*src;
src += dim;

*(dst + 17)=*src;
src += dim;

*(dst + 18)=*src;
src += dim;

*(dst + 19)=*src;
src += dim;

*(dst + 20)=*src;
src += dim;
```

```
    *(dst + 21)=*src;
    src += dim;

    *(dst + 22)=*src;
    src += dim;

    *(dst + 23)=*src;
    src += dim;

    *(dst + 24)=*src;
    src += dim;

    *(dst + 25)=*src;
    src += dim;

    *(dst + 26)=*src;
    src += dim;

    *(dst + 27)=*src;
    src += dim;

    *(dst + 28)=*src;
    src += dim;

    *(dst +29)=*src;
    src += dim;

    *(dst + 30)=*src;
    src += dim;

    *(dst + 31)=*src;

     dst -= dim;
    src -= dim*31 - 1;
}
dst += 32 + extension;
src += dim*31;
```

```
        }
    return;
}
```

2.3 Оптимизация smooth

```
void smooth(int dim, pixel *src, pixel *dst)
{
   int i, j, k;

   dst[0].red = (src[0].red+src[1].red+src[dim].red+src[dim+1].red)>>2;
   dst[0].blue = (src[0].blue+src[1].blue+src[dim].blue+src[dim+1].blue)>>2;
   dst[0].green =
(src[0].green+src[1].green+src[dim].green+src[dim+1].green)>>2;

   i = dim*2-1;
   dst[dim-1].red = (src[dim-2].red+src[dim-1].red+src[i-1].red+src[i].red)>>2;
   dst[dim-1].blue =
(src[dim-2].blue+src[dim-1].blue+src[i-1].blue+src[i].blue)>>2;
   dst[dim-1].green =
(src[dim-2].green+src[dim-1].green+src[i-1].green+src[i].green)>>2;

   j = dim*(dim-1);
   i = dim*(dim-2);
   dst[j].red = (src[j].red+src[j + 1].red+src[i].red+src[i + 1].red)>>2;
   dst[j].blue = (src[j].blue+src[j + 1].blue+src[i].blue+src[i + 1].blue)>>2;
   dst[j].green = (src[j].green+src[j + 1].green+src[i].green+src[i + 1].green)>>2;

   j = dim*dim-1;
   i = dim*(dim-1)-1;
   dst[j].red = (src[j - 1].red+src[j].red+src[i - 1].red+src[i].red)>>2;
   dst[j].blue = (src[j - 1].blue+src[j].blue+src[i - 1].blue+src[i].blue)>>2;
   dst[j].green = (src[j - 1].green+src[j].green+src[i - 1].green+src[i].green)>>2;
   i = dim - 1;

   for (j = 1; j < i; j++)
   {
      dst[j].red =
(src[j].red+src[j-1].red+src[j+1].red+src[j+dim].red+src[j+1+dim].red+src[j-1+dim
].red)/6;
```

```
        dst[j].green =
(src[j].green+src[j-1].green+src[j+1].green+src[j+dim].green+src[j+1+dim].green+
src[j-1+dim].green)/6;
        dst[j].blue =
(src[j].blue+src[j-1].blue+src[j+1].blue+src[j+dim].blue+src[j+1+dim].blue+src[j-1
+dim].blue)/6;
    }

    i = dim*dim-1;
    for (j = i - dim + 2; j < i; j++)
    {
        dst[j].red = (src[j].red+src[j-1].red+src[j+1].red+src[j-dim].red+src[j+1-
dim].red+src[j-1-dim].red)/6;
        dst[j].green = (src[j].green+src[j-1].green+src[j+1].green+src[j-
dim].green+src[j+1-dim].green+src[j-1-dim].green)/6;
        dst[j].blue = (src[j].blue+src[j-1].blue+src[j+1].blue+src[j-dim].blue+src[j+1-
dim].blue+src[j-1-dim].blue)/6;
    }

    for (j = dim+dim-1; j < dim*dim-1; j+=dim)
     {
        dst[j].red = (src[j].red+src[j-1].red+src[j-dim].red+src[j+dim].red+src[j-
dim-1].red+src[j-1+dim].red)/6;
        dst[j].green = (src[j].green+src[j-1].green+src[j-
dim].green+src[j+dim].green+src[j-dim-1].green+src[j-1+dim].green)/6;
        dst[j].blue = (src[j].blue+src[j-1].blue+src[j-dim].blue+src[j+dim].blue+src[j-
dim-1].blue+src[j-1+dim].blue)/6;
    }

    i = i - (dim - 1);
    for (j = dim; j < i; j+=dim)
     {
        dst[j].red = (src[j].red+src[j-
dim].red+src[j+1].red+src[j+dim].red+src[j+1+dim].red+src[j-dim+1].red)/6;
        dst[j].green = (src[j].green+src[j-
dim].green+src[j+1].green+src[j+dim].green+src[j+1+dim].green+src[j-
dim+1].green)/6;
```

```
        dst[j].blue = (src[j].blue+src[j-
dim].blue+src[j+1].blue+src[j+dim].blue+src[j+1+dim].blue+src[j-dim+1].blue)/6;
    }

    k = dim;

    for (i = 1; i < dim-1; i++)
    {
        for (j = 1; j < dim-1; j++)
        {
            k ++;
            dst[k].red = (src[k-1].red+src[k].red+src[k+1].red+src[k-dim-1].red+src[k-
dim].red+src[k-dim+1].red+src[k+dim-1].red+src[k+dim].red+src[k+dim+1].red)/
9;
            dst[k].green = (src[k-1].green+src[k].green+src[k+1].green+src[k-
dim-1].green+src[k-dim].green+src[k-
dim+1].green+src[k+dim-1].green+src[k+dim].green+src[k+dim+1].green)/9;
            dst[k].blue = (src[k-1].blue+src[k].blue+src[k+1].blue+src[k-
dim-1].blue+src[k-dim].blue+src[k-
dim+1].blue+src[k+dim-1].blue+src[k+dim].blue+src[k+dim+1].blue)/9;
        }
        k += 2;
    }
}
```

3. Результаты

3.1 До оптимизации:

```
ubuntu@ubuntu:~/Desktop/lab7/lab7/perflab-handout$ ./driver
Rotate: Version = naive_rotate: Naive baseline implementation:
Dim             64      128     256     512     1024    Mean
Your CPEs       1.4     1.5     3.7     5.6     6.3
Baseline CPEs   1.5     2.2     4.1     7.6     7.2
Speedup         1.1     1.5     1.1     1.3     1.1     1.2

Rotate: Version = rotate: Current working version:
Dim             64      128     256     512     1024    Mean
Your CPEs       1.2     1.3     3.2     5.0     6.4
Baseline CPEs   1.5     2.2     4.1     7.6     7.2
Speedup         1.2     1.7     1.3     1.5     1.1     1.3

Smooth: Version = smooth: Current working version:
Dim             32      64      128     256     512     Mean
Your CPEs       34.4    36.6    36.7    36.8    37.1
Baseline CPEs   33.8    33.7    33.6    33.9    33.2
Speedup         1.0     0.9     0.9     0.9     0.9     0.9

Smooth: Version = naive_smooth: Naive baseline implementation:
Dim             32      64      128     256     512     Mean
Your CPEs       39.6    42.1    42.3    42.4    42.9
Baseline CPEs   33.8    33.7    33.6    33.9    33.2
Speedup         0.9     0.8     0.8     0.8     0.8     0.8

Summary of Your Best Scores:
  Rotate: 1.3 (rotate: Current working version)
  Smooth: 0.9 (smooth: Current working version)
```

3.2 После оптимизации:

```
ubuntu@ubuntu:~/Desktop/lab7/lab7/perflab-handout$ make driver
gcc -Wall -O2 -m32   -c -o kernels.o kernels.c
gcc -Wall -O2 -m32 driver.o kernels.o fcyc.o clock.o -lm -o driver
ubuntu@ubuntu:~/Desktop/lab7/lab7/perflab-handout$ ./driver
Rotate: Version = naive_rotate: Naive baseline implementation:
Dim             64      128     256     512     1024    Mean
Your CPEs       1.2     1.3     3.2     5.0     6.3
Baseline CPEs   1.5     2.2     4.1     7.6     7.2
Speedup         1.2     1.7     1.3     1.5     1.1     1.3

Rotate: Version = rotate: Current working version:
Dim             64      128     256     512     1024    Mean
Your CPEs       0.7     0.8     0.8     0.9     1.7
Baseline CPEs   1.5     2.2     4.1     7.6     7.2
Speedup         2.1     2.9     4.8     8.8     4.3     4.1

Smooth: Version = smooth: Current working version:
Dim             32      64      128     256     512     Mean
Your CPEs       9.3     10.0    10.0    10.0    10.1
Baseline CPEs   33.8    33.7    33.6    33.9    33.2
Speedup         3.6     3.4     3.4     3.4     3.3     3.4

Smooth: Version = naive_smooth: Naive baseline implementation:
Dim             32      64      128     256     512     Mean
Your CPEs       39.8    42.2    42.3    41.8    43.0
Baseline CPEs   33.8    33.7    33.6    33.9    33.2
Speedup         0.8     0.8     0.8     0.8     0.8     0.8

Summary of Your Best Scores:
  Rotate: 4.1 (rotate: Current working version)
  Smooth: 3.4 (smooth: Current working version)
ubuntu@ubuntu:~/Desktop/lab7/lab7/perflab-handout$ ./driver
Rotate: Version = naive_rotate: Naive baseline implementation:
Dim             64      128     256     512     1024    Mean
Your CPEs       1.4     1.5     3.7     5.7     6.5
Baseline CPEs   1.5     2.2     4.1     7.6     7.2
Speedup         1.1     1.5     1.1     1.3     1.1     1.2

Rotate: Version = rotate: Current working version:
Dim             64      128     256     512     1024    Mean
Your CPEs       0.7     0.8     0.8     0.9     1.5
Baseline CPEs   1.5     2.2     4.1     7.6     7.2
Speedup         2.1     2.9     4.9     8.7     4.7     4.1

Smooth: Version = smooth: Current working version:
Dim             32      64      128     256     512     Mean
Your CPEs       9.3     10.0    10.4    10.1    8.8
Baseline CPEs   33.8    33.7    33.6    33.9    33.2
Speedup         3.6     3.4     3.2     3.3     3.8     3.5

Smooth: Version = naive_smooth: Naive baseline implementation:
Dim             32      64      128     256     512     Mean
Your CPEs       39.9    42.1    42.3    42.4    38.9
Baseline CPEs   33.8    33.7    33.6    33.9    33.2
Speedup         0.8     0.8     0.8     0.8     0.9     0.8

Summary of Your Best Scores:
  Rotate: 4.1 (rotate: Current working version)
  Smooth: 3.5 (smooth: Current working version)
```