МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

**НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1

по курсу «Эффективное программирование современных микропроцессоров и мультипроцессоров»

**(Вариант №3)**

**Выполнил:** студент 3-го курса гр. 17208

Гафиятуллин А.Р

Новосибирск, 2020

1. **ЦЕЛИ РАБОТЫ:**

Научиться разрабатывать простые программы численного моделирования, применять базовые средства оптимизации программ, выполнять оценку и анализ производительности программ, пользоваться средствами профилирования.

**Вариант №3:** *решение уравнения Пуассона методом Якоби на float-ах.*

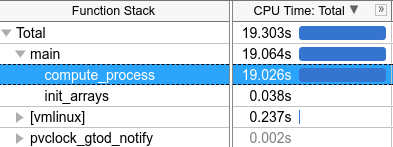
Алгоритм моделирует установление стационарного распределение тепла в пластинке с заданным распределением источников и стоков тепла. В начальный момент времени значения искомой функции на сетке инициализируются нулями. На каждом шаге моделирования значения искомой функции пересчитываются по заданной формуле.

**Nx = Ny = 9000, Nt = 110**.

1. **ХОД РАБОТЫ:**
   1. Тестирование происходило на процессоре **Intel(R) Core(TM) i7-9700F CPU @ 3.00GHz (CPU max MHz: 4700.0000 (Turbo Boost))**.
   2. Текст 1-го работающего варианта программы **(см. приложение 4.1)**.
   3. Текст самого быстрого варианта программы **(см. приложение 4.2)**.
   4. Описание использованных способов оптимизации программы с результатами:

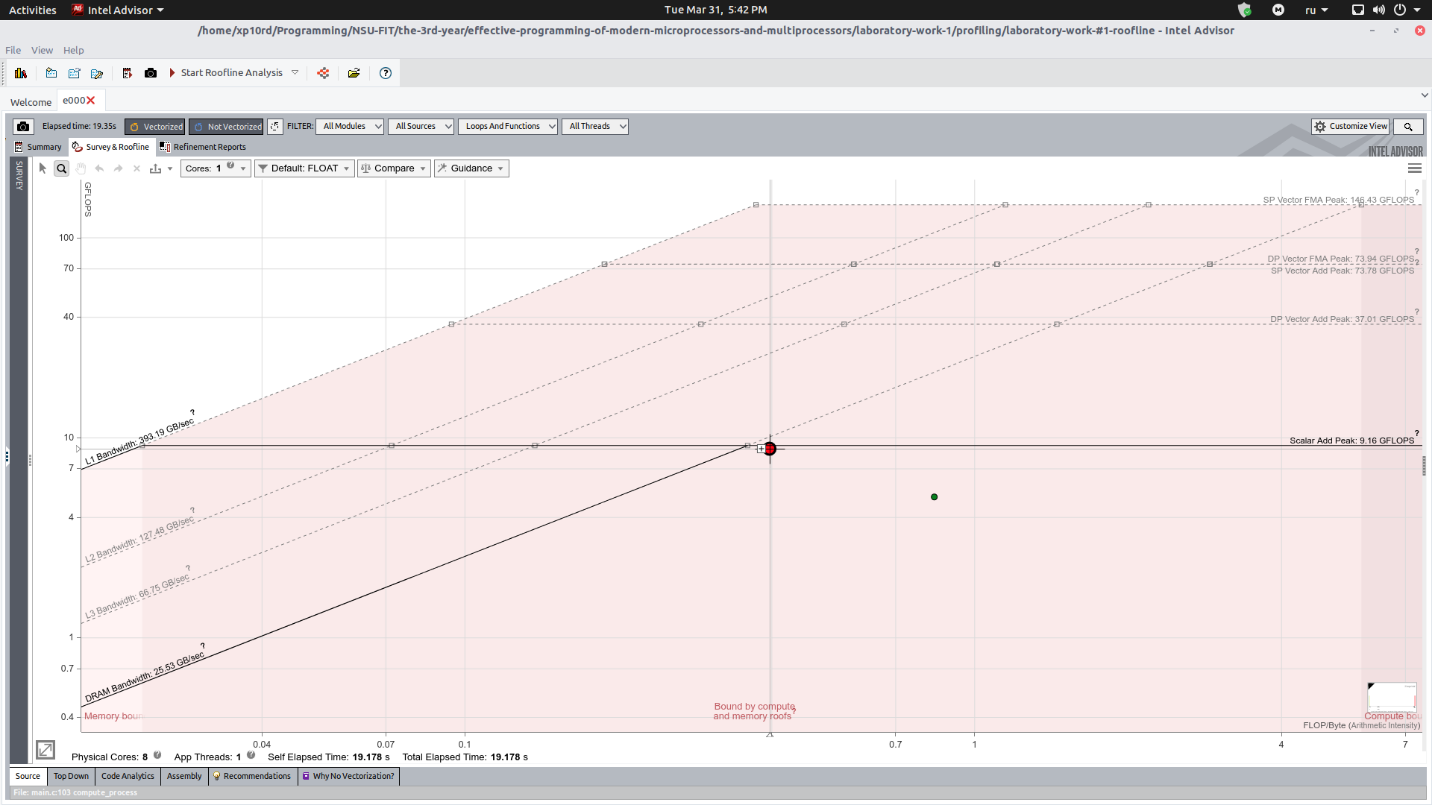
| Оптимизация, тип | Время, сек. |
| --- | --- |
| Без оптимизаций (-O0, алгоритмических оптимизаций нет) | 204 |
| Замена дефайна max на inline-функцию max (-O0) | 218 |
| Переупорядочено обращение к массивам, чтобы уменьшить количество кэш-промахов (-O0) | 200 |
| -O1 | 40 |
| -O2 | 24 |
| -O2 -ip -xcoffeelake -axcoffeelake | 22 |
| Переиспользование полученной из массивов информации на следующих итерациях, чтобы уменьшить количество обращений в кучу | 22 |
| Переупорядочивание некоторых команд после многократного получения результатов профилирования | 19 |

* 1. Граф вызовов программы:



Синим выделена горячая точка.

* 1. Аннотированный листинг «горячей точки» программы **(см. приложение 4.3).**
  2. Аннотированный ассемблерный листинг «горячей точки» программы **(см. приложение 4.4).**
  3. Характеристики исполнения программы были получены с использованием утилиты perf (**только для загрузок(load)**):
     1. **Число инструкций на такт:** 3.01;
     2. **Процент кэш-промахов для кэша 3 уровня:** 44.54%;
     3. **Процент кэш-промахов для кэша-данных 1 уровня:** 5.04%;
     4. **Процент неправильно предсказанных переходов:** 0.03%.
     5. По всей видимости, основной причиной временных затрат являются **вычислительные операции**. Так как процент промахов при обращении к кэшу 1 уровня мал, а количество обращений к кэшу 3 уровня по сравнению с кэшом 1 уровня крайне мало (для одного из замеров: к 3 уровню - **43,019,723** обращений, а к 1 уровню - **71,513,217,255**), то, скорее всего, это особо не влияет на производительность.
  4. **Roofline-модель с точкой, соответствующей основному циклу программы (красная точка посередине):**

****

**Автоматическое заключение Intel Advisor:** производительность основного цикла программы ограничена **вычислительными операциями**.

1. **ВЫВОДЫ:**
   1. Научились разрабатывать простые программы численного моделирования, применять базовые средства оптимизации программ, выполнять оценку и анализ производительности программ, пользоваться средствами профилирования.
   2. На данный момент производительность самой быстрой версии программы ограничена скоростью вычислительных операций.
2. **ПРИЛОЖЕНИЕ:**
   1. Текст 1-го работающего варианта программы.

#define comp\_type float

#define Nx 9000

#define Ny 9000

#define Nt 111

#define Xa (comp\_type)0.0

#define Xb (comp\_type)4.0

#define Ya (comp\_type)0.0

#define Yb (comp\_type)4.0

#define hx (comp\_type)((Xb - Xa) / (Nx - 1))

#define hy (comp\_type)((Yb - Ya) / (Ny - 1))

#define coeff1 ((comp\_type)0.2 / ((comp\_type)1.0 / (hx \* hx) + (comp\_type)1.0 / (hy \* hy)))

#define coeff2 ((comp\_type)0.5 \* ((comp\_type)5.0 / (hx \* hx) - (comp\_type)1.0 / (hy \* hy)))

#define coeff3 ((comp\_type)0.25 \* ((comp\_type)1.0 / (hx \* hx) + (comp\_type)1.0 / (hy \* hy)))

#define X(j) (Xa + (j) \* hx)

#define Y(i) (Ya + (i) \* hy)

#define Xs1 (Xa + (Xb - Xa) / (comp\_type)3.0)

#define Xs2 (Xa + (Xb - Xa) \* (comp\_type)2.0 / (comp\_type)3.0)

#define Ys1 (Ya + (Yb - Ya) \* (comp\_type)2.0 / (comp\_type)3.0)

#define Ys2 (Ya + (Yb - Ya) / (comp\_type)3.0)

#define R ((comp\_type)0.1 \* ((Xb - Xa) > (Yb - Ya) ? (Yb - Ya) : (Xb - Xa)))

#define GRID\_SIZE (Nx \* Ny)

#define TIME\_LAYERS 2

#define get3(F, n, i, j) F[(n) \* GRID\_SIZE + (i) \* Nx + (j)]

#define get2(p, i, j) p[(i) \* Nx + (j)]

#define max(a,b) ((a) > (b) ? (a) : (b))

#include <stdlib.h>

#include <stdio.h>

#include <errno.h>

#include <math.h>

#include <unistd.h>

#include <fcntl.h>

#include <time.h>

int main**()** **{**

/\* allocate memory \*/

comp\_type**\*** F **=** malloc**(**TIME\_LAYERS **\*** GRID\_SIZE **\*** **sizeof(**comp\_type**));**

comp\_type **\***p **=** malloc**(**GRID\_SIZE **\*** **sizeof(**comp\_type**));**

**if(!**F **||** **!**p**)** **{**

perror**(**"malloc"**);**

exit**(**errno**);**

**}**

/\* init arrays \*/

**for(**int i **=** 0**;** i **<** GRID\_SIZE**;** i**++)** **{**

F**[**i**]** **=** **(**comp\_type**)**0.0**;**

**}**

**for(**int i **=** 0**;** i **<** Ny**;** i**++)** **{**

**for(**int j **=** 0**;** j **<** Nx**;** j**++)** **{**

**if((**X**(**j**)** **-** Xs1**)** **\*** **(**X**(**j**)** **-** Xs1**)** **+** **(**Y**(**i**)** **-** Ys1**)** **\*** **(**Y**(**i**)** **-** Ys1**)** **<** R **\*** R**)** **{**

get2**(**p**,** i**,** j**)** **=** **(**comp\_type**)**0.1**;**

**}** **else** **if((**X**(**j**)** **-** Xs2**)** **\*** **(**X**(**j**)** **-** Xs2**)** **+** **(**Y**(**i**)** **-** Ys2**)** **\*** **(**Y**(**i**)** **-** Ys2**)** **<** R **\*** R**)** **{**

get2**(**p**,** i**,** j**)** **=** **(**comp\_type**)-**0.1**;**

**}** **else** **{**

get2**(**p**,** i**,** j**)** **=** **(**comp\_type**)**0.0**;**

**}**

**}**

**}**

/\* compute process \*/

time\_t start\_time **=** time**(NULL);**

comp\_type delta **=** 0**;**

**for(**int n **=** 0**;** n **<** Nt **-** 1**;** n**++)** **{**

delta **=** 0**;**

**for(**int i **=** 1**;** i **<** Ny **-** 1**;** i**++)** **{**

**for(**int j **=** 1**;** j **<** Nx **-** 1**;** j**++)** **{**

int n\_idx **=** n **%** 2**;**

int nplus1\_idx **=** **(**n **+** 1**)** **%** 2**;**

get3**(**F**,** nplus1\_idx**,** i**,** j**)** **=** coeff1 **\*** **(**

coeff2 **\*** **(**get3**(**F**,** n\_idx**,** i**,** j **-** 1**)** **+** get3**(**F**,** n\_idx**,** i**,** j **+** 1**))** **+**

coeff2 **\*** **(**get3**(**F**,** n\_idx**,** i **-** 1**,** j**)** **+** get3**(**F**,** n\_idx**,** i **+** 1**,** j**))** **+**

coeff3 **\*** **(**get3**(**F**,** n\_idx**,** i **-** 1**,** j **-** 1**)** **+**

get3**(**F**,** n\_idx**,** i **-** 1**,** j **+** 1**)** **+**

get3**(**F**,** n\_idx**,** i **+** 1**,** j **-** 1**)** **+**

get3**(**F**,** n\_idx**,** i **+** 1**,** j **+** 1**))** **+**

2.0f **\*** get2**(**p**,** i**,** j**)** **+**

0.25f **\*** **(**get2**(**p**,** i **-** 1**,** j**)** **+**

get2**(**p**,** i **+** 1**,** j**)** **+**

get2**(**p**,** i**,** j **-** 1**)** **+**

get2**(**p**,** i**,** j **+** 1**)));**

delta **=** max**(**delta**,** fabs**(**get3**(**F**,** n\_idx**,** i**,** j**)** **-** get3**(**F**,** nplus1\_idx**,** i**,** j**)));**

**}**

**}**

**}**

time\_t end\_time **=** time**(NULL);**

printf**(**"n = %d, sigma = %.8f\n"**,** Nt **-** 1**,** delta**);**

printf**(**"Total time: %lld sec.\n"**,** end\_time **-** start\_time**);**

**}**

* 1. Текст самого быстрого варианта программы.

#define Nx 9000

#define Ny 9000

#define Nt 111

#define Xa 0.0f

#define Xb 4.0f

#define Ya 0.0f

#define Yb 4.0f

#define hx ((Xb - Xa) / (Nx - 1))

#define hy ((Yb - Ya) / (Ny - 1))

#define coeff1 (0.2f / ((1.0f / (hx \* hx) + 1.0f / (hy \* hy))))

#define coeff2 (0.5f \* (5.0f / (hx \* hx) - 1.0f / (hy \* hy)))

#define coeff2b (0.5f \* (5.0f / (hy \* hy) - 1.0f / (hx \* hx)))

#define coeff3 (0.25f \* (1.0f / (hx \* hx) + 1.0f / (hy \* hy)))

#define X(j) (Xa + (j) \* hx)

#define Y(i) (Ya + (i) \* hy)

#define Xs1 (Xa + (Xb - Xa) / 3.0f)

#define Xs2 (Xa + (Xb - Xa) \* 2.0f / 3.0f)

#define Ys1 (Ya + (Yb - Ya) \* 2.0f / 3.0f)

#define Ys2 (Ya + (Yb - Ya) / 3.0f)

#define R (0.1f \* ((Xb - Xa) > (Yb - Ya) ? (Yb - Ya) : (Xb - Xa)))

#define GRID\_SIZE (Nx \* Ny)

#define TIME\_LAYERS 2

#define get3(F, n, i, j) F[(n) \* GRID\_SIZE + (i) \* Nx + (j)]

#define get2(p, i, j) p[(i) \* Nx + (j)]

#define max(a, b) ((a) > (b) ? (a) : (b))

#include <stdlib.h>

#include <stdio.h>

#include <errno.h>

#include <math.h>

#include <unistd.h>

#include <fcntl.h>

#include <time.h>

void init\_arrays**(**float **\***F**,** float **\***p**);**

float compute\_process**(**float **\***F**,** float **\***p**);**

int main**()** **{**

/\* allocate memory \*/

float**\*** F **=** malloc**(**TIME\_LAYERS **\*** GRID\_SIZE **\*** **sizeof(**float**));**

float **\***p **=** malloc**(**GRID\_SIZE **\*** **sizeof(**float**));**

**if(!**F **||** **!**p**)** **{**

perror**(**"malloc"**);**

exit**(**errno**);**

**}**

init\_arrays**(**F**,** p**);**

time\_t start\_time **=** time**(NULL);**

float delta **=** compute\_process**(**F**,** p**);**

time\_t end\_time **=** time**(NULL);**

printf**(**"n = %d, sigma = %.8f\n"**,** Nt **-** 1**,** delta**);**

printf**(**"Total time: %ld sec.\n"**,** end\_time **-** start\_time**);**

**}**

void init\_arrays**(**float **\***F**,** float **\***p**)** **{**

**for(**int i **=** 0**;** i **<** Ny**;** i**++)** **{**

**for(**int j **=** 0**;** j **<** Nx**;** j**++)** **{**

float xj **=** X**(**j**);**

float yi **=** Y**(**i**);**

**if((**xj **-** Xs1**)** **\*** **(**xj **-** Xs1**)** **+** **(**yi **-** Ys1**)** **\*** **(**yi **-** Ys1**)** **<** R **\*** R**)** **{**

get2**(**p**,** i**,** j**)** **=** 0.1f**;**

**}** **else** **if((**xj **-** Xs2**)** **\*** **(**xj **-** Xs2**)** **+** **(**yi **-** Ys2**)** **\*** **(**yi **-** Ys2**)** **<** R **\*** R**)** **{**

get2**(**p**,** i**,** j**)** **=** **-**0.1f**;**

**}** **else** **{**

get2**(**p**,** i**,** j**)** **=** 0.0f**;**

**}**

get2**(**F**,** i**,** j**)** **=** 0.0f**;**

**}**

**}**

**}**

float compute\_process**(**float **\***F**,** float **\***p**)** **{**

float delta **=** 0**;**

**for(**int n **=** 0**;** n **<** Nt **-** 1**;** n**++)** **{**

int n\_idx **=** n **%** 2**;**

int nplus1\_idx **=** **(**n **+** 1**)** **%** 2**;**

delta **=** 0**;**

**for(**int i **=** 1**;** i **<** Ny **-** 1**;** i**++)** **{**

float f\_left\_down\_cell **=** get3**(**F**,** n\_idx**,** i **-** 1**,** 0**);**

float f\_current\_down\_cell **=** get3**(**F**,** n\_idx**,** i **-** 1**,** 1**);**

float f\_right\_down\_cell **=** get3**(**F**,** n\_idx**,** i **-** 1**,** 2**);**

float f\_left\_current\_cell **=** get3**(**F**,** n\_idx**,** i**,** 0**);**

float f\_current\_current\_cell **=** get3**(**F**,** n\_idx**,** i**,** 1**);**

float f\_right\_current\_cell **=** get3**(**F**,** n\_idx**,** i**,** 2**);**

float f\_left\_up\_cell **=** get3**(**F**,** n\_idx**,** i **+** 1**,** 0**);**

float f\_current\_up\_cell **=** get3**(**F**,** n\_idx**,** i **+** 1**,** 1**);**

float f\_right\_up\_cell **=** get3**(**F**,** n\_idx**,** i **+** 1**,** 2**);**

float p\_left\_current\_cell **=** get2**(**p**,** i**,** 0**);**

float p\_current\_current\_cell **=** get2**(**p**,** i**,** 1**);**

float p\_right\_current\_cell **=** get2**(**p**,** i**,** 2**);**

**for(**register int j **=** 1**;** j **<** Nx **-** 1**;** j**++)** **{**

float rez **=** coeff1 **\*** **(**

coeff2b **\*** **(**f\_current\_down\_cell **+** f\_current\_up\_cell**)** **+**

coeff3 **\*** **(**f\_left\_up\_cell **+** f\_right\_up\_cell **+** f\_left\_down\_cell **+** f\_right\_down\_cell**)** **+**

coeff2 **\*** **(**f\_left\_current\_cell **+** f\_right\_current\_cell**)** **+**

0.25f **\*** **(**

2.0f **\*** p\_current\_current\_cell **+**

p\_left\_current\_cell **+**

p\_right\_current\_cell**)** **+**

0.25f **\*** get2**(**p**,** i **-** 1**,** j**)** **+**

get2**(**p**,** i **+** 1**,** j**));**

get3**(**F**,** nplus1\_idx**,** i**,** j**)** **=** rez**;**

delta **=** max**(**delta**,** fabs**(**f\_current\_current\_cell **-** rez**));**

f\_left\_down\_cell **=** f\_current\_down\_cell**;**

f\_current\_down\_cell **=** f\_right\_down\_cell**;**

f\_left\_up\_cell **=** f\_current\_up\_cell**;**

f\_current\_up\_cell **=** f\_right\_up\_cell**;**

f\_left\_current\_cell **=** f\_current\_current\_cell**;**

f\_current\_current\_cell **=** f\_right\_current\_cell**;**

f\_right\_down\_cell **=** get3**(**F**,** n\_idx**,** i **-** 1**,** j **+** 2**);**

f\_right\_current\_cell **=** get3**(**F**,** n\_idx**,** i**,** j **+** 2**);**

f\_right\_up\_cell **=** get3**(**F**,** n\_idx**,** i **+** 1**,** j **+** 2**);**

p\_left\_current\_cell **=** p\_current\_current\_cell**;**

p\_current\_current\_cell **=** p\_right\_current\_cell**;**

p\_right\_current\_cell **=** get2**(**p**,** i**,** j **+** 2**);**

**}**

**}**

**}**

**return** **(**delta**);**

**}**

* 1. Аннотированный ассемблерный листинг «горячей точки» программы **(ссылка на картинку кликабельна, сохранена локально в документе)**:



синим отмечена самая «горячая» точка.

* 1. Аннотированный ассемблерный листинг «горячей точки» программы:

**(ссылка на картинку кликабельна, сохранена локально в документе)**:



синим отмечены команды, соответствующие самой «горячей» точке.