Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по Практикуму №1

по курсу «Архитектура ЭВМ»

на тему: *«Разработка и отладка программ в вычислительном комплексе Тераграф»*

Студент группы ИУ7-51Б _____ (Подпись, дата) Постнов С. А. (Фамилия И.О.)

Преподаватель _____ (Подпись, дата) Ибрагимов С. В. (Фамилия И.О.)

Москва — 2023 г.

Содержание

1 Введение

Практикум посвящен освоению принципов работы вычислительного комплекса Терапраф и получению практических навыков решения задач обработки множеств на основе гетерогенной вычислительной структуры. Участникам предоставляется доступ к удаленному серверу с ускорительной картой и настроенными средствами сборки проектов, конфигурационный файл для двухъядерной версии микропроцессора Леонар Эйлер, а также библиотека leonhard x64 xrt с открытым исходным кодом.

Целью работы является разработка программы для хост–подсистемы и обработчики программного ядра, выполняющие действия по варианту.

Для *15 варианта* необходимо разработать **устройство интегрирования**. То есть сформировать в хост–подсистеме и передать в SPE 256 записей с ключами x и значениями $f(x) = x^2$ в диапазоне значений x от 0 до 1048576. Передать в sw_kernel числа $x1$ и $x2$ ($x2 > x1$). В хост–подсистему вернуть сумму значений $f(x)$ на диапазоне $(x1, x2)$. Сравнить результат с ожидаемым.

2 Выполнение работы

Для достижения поставленной цели необходимо:

1) изменить код common_struct.h под индивидуальное задание;

2) изменить код хост–подсистемы под индивидуальное задание;

3) реализовать алгоритм поиска ближайшего значения к заданному в sw_kernel;

4) реализовать общий алгоритм нахождения суммы значений из заданного промежутка в sw_kernel.

2.1 Изменение кода common_struct.h

В листинге 2.1 представлен измененный код common_struct.h.

Листинг 2.1 – Измененный код common_struct.h

```
#ifndef COMMON_STRUCT
#define COMMON_STRUCT

#ifdef __riscv64__
#include "map.h"
#endif
#include "compose_keys.hxx"

//Номера структур данных в SPE
enum Structures : uint32_t {
    null            = 0,    //Нулевая структура не используется
    users_pnum      = 1,    //Таблица 1
    resources_pnum  = 2,    //Таблица 2
    function_pnum = 3 // Рабочая таблица 3
};

#ifdef __riscv64__
//Задание даипазонов и курсоров
template<typename Range>
struct reverse {
    Range r;
```

```
22      [[gnu::always_inline]] reverse(Range r) : r(r) {}
23      [[gnu::always_inline]] auto begin() {return r.rbegin();}
24      [[gnu::always_inline]] auto end() {return r.rend();}
25  };
26
27  template<typename K, typename V>
28  struct Handle {
29      bool ret_val;
30      K k{get_result_key<K>()};
31      V v{get_result_value<V>()};
32      [[gnu::always_inline]] Handle(bool ret_val) :
            ret_val(ret_val) {
33      }
34
35      [[gnu::always_inline]] operator bool() const {
36          return ret_val;
37      }
38
39      [[gnu::always_inline]] K key() const {
40          return k;
41      }
42
43      [[gnu::always_inline]] V value() const {
44          return v;
45      }
46  };
47  #endif
48
49
50  ////////////////////////////////////
51  // Описание формата ключа и значения
52  ////////////////////////////////////
53
54
55  struct users {
56      using vertex_t = uint32_t;
57      int struct_number;
58      constexpr users(int struct_number) :
            struct_number(struct_number) {}
59      static const uint32_t idx_bits = 32;
60      static const uint32_t idx_max = (1ull << idx_bits) - 1;
```

```
61      static const uint32_t idx_min = idx_max;
62
63      //Запись для формирования ключей (* - наиболее значимые биты
            поля)
64      STRUCT(key)
65      {
66          uint32_t    idx     :idx_bits;  //Поле 0:
67          uint32_t    user    :32;        //Поле 1*
68      };
69
70      //Запись для формирования значений
71      STRUCT(val)
72      {
73          uint32_t    role    :32;        //Поле 0:
74          time_t      time    :32;        //Поле 1*
75      };
76      //Обязательная типизация
77      #ifdef __riscv64__
78      DEFINE_DEFAULT_KEYVAL(key, val)
79      #endif
80  };
81
82  struct rev_function{
83      using vertex_t = uint32_t;
84      int struct_number;
85      constexpr rev_function(int struct_number) :
           struct_number(struct_number) {}
86      static const uint32_t idx_bits = 32;
87      static const uint32_t idx_max = (1ull << idx_bits) - 1;
88      static const uint32_t idx_min = idx_max;
89
90      //Запись для формирования ключей (* - наиболее значимые биты
            поля)
91      STRUCT(key)
92      {
93          uint64_t    idx     :64;    //Поле 0:
94      };
95
96      //Запись для формирования значений
97      STRUCT(val)
98      {
```

```
99          uint64_t    value   :64;        //Поле 0:
100     };
101     //Обязательная типизация
102     #ifdef __riscv64__
103     DEFINE_DEFAULT_KEYVAL(key, val)
104     #endif
105 };
106
107 constexpr users USERS(Structures::users_pnum);
108 constexpr rev_function REV_FUNCTION(Structures::function_pnum);
109
110 #endif //COMMON_STRUCT
```

2.2 Изменение кода хост–подсистемы

В листинге 2.2 представлен измененный код хост–подсистемы.

Листинг 2.2 – Измененный код хост–подсистемы

```
1 #include <iostream>
2 #include <random>
3 #include <iterator>
4 #include <string>
5 #include <regex>
6 #include <sstream>
7 #include <fstream>
8 #include <ctime>
9 #include "host_main.h"
10
11 #define KEYS_COUNT 13
12 #define MIN_VALUE 356
13 #define MAX_VALUE 10000
14
15 using namespace std;
16
17 int main(int argc, char** argv)
18 {
19     ofstream log("task.log"); //поток вывода сообщений
20     unsigned long long offs=0ull;
21     gpc *gpc64_inst; //указатель на класс gpc
```

```
22
23      std::random_device dev;
24      std::mt19937 rng(dev());
25      std::uniform_int_distribution<std::mt19937::result_type>
            gen_int(MIN_VALUE, MAX_VALUE);
26
27      //Инициализация gpc
28      if (argc<2) {
29          log<<"Использование: host_main <путь к файлу
                rawbinary>"<<endl;
30          return -1;
31      }
32
33      //Захват ядра gpc и запись sw_kernel
34      gpc64_inst = new gpc();
35      cout<<"Открывается доступ к
            "<<gpc64_inst->gpc_dev_path<<endl;
36      if (gpc64_inst->load_swk(argv[1])==0) {
37          cout<<"Программное ядро загружено из файла
                "<<argv[1]<<endl;
38      }
39      else {
40          cout<<"Ошибка загрузки sw_kernel файла <<
                argv[1]"<<endl;
41          return -1;
42      }
43
44      //Инициализация таблицы для вложенного запроса
45      gpc64_inst->start(__event__(update)); //обработчик вставки
46
47      for (uint64_t i = 0, key=MIN_VALUE;i<KEYS_COUNT;++i, ++key)
            {
48          uint64_t val = key * key;
49          gpc64_inst->mq_send(rev_function::key{.idx=key});
                //запись о роли #idx
50          gpc64_inst->mq_send(rev_function::val{.value=val});
                //роль и время доступа
51          std::cout << "inserting by key " << key << " value " <<
                val << std::endl;
52      }
53
```

```
54      //Терминальный символ
55      gpc64_inst->mq_send(-1ull);
56      std::cout << "finished sending\n";
57      gpc64_inst->start(__event__(select)); //обработчик запроса п
            оиска
58      std::cout << "started selecting\n";
59      bool requested = true;
60      while (requested)
61      {
62          std::cout << "Введите x1 для поиска суммы значений на пр
                омежутке (x1, x2):(-1 для завершения ввода)\n";
63          uint64_t val_1;
64          std:: cin >> val_1;
65
66          if (val_1 == (uint64_t)-1)
67          {
68              requested = false;
69              break;
70          }
71
72          std::cout << "Введите x2 для поиска суммы значений на пр
                омежутке (x1, x2):(-1 для завершения ввода)\n";
73          uint64_t val_2;
74          std:: cin >> val_2;
75
76          if (val_2 == (uint64_t)-1)
77          {
78              requested = false;
79              break;
80          }
81
82          gpc64_inst->mq_send(val_1);
83          gpc64_inst->mq_send(val_2);
84
85          uint64_t result = gpc64_inst->mq_receive();
86          uint64_t result_val =
                rev_function::val::from_int(result).value;
87          std:: cout<< "\n Сумма:" << result_val << "\n";
88      }
89
90      gpc64_inst->mq_send(-1ull);
```

```
91
92
93      std::cout << "Выход!" << endl;
94      return 0;
95  }
```

2.3 Реализация алгоритма поиска ближайшего значения к заданному

В листинге 2.3 представлена реализация алгоритма поиска ближайшего значения к заданному.

Листинг 2.3 – Листинг реализации алгоритма поиска ближайшего значения к заданному

```
1  void get_nearest_value(uint64_t &key, uint64_t &val, bool
       need_bigger) {
2      if (need_bigger) {
3          auto g = REV_FUNCTION.ngr(rev_function::key{.idx=key});
4          key = g.key();
5          val = g.value();
6      } else {
7          auto l = REV_FUNCTION.nsm(rev_function::key{.idx=key});
8          key = l.key();
9          val = l.value();
10     }
11 }
```

2.4 Реализация общего алгоритма нахождения суммы значений из заданного промежутка

В листинге 2.4 представлена реализация общего алгоритма нахождения суммы значений из заданного промежутка.

Листинг 2.4 – Листинг реализации общего алгоритма нахождения суммы значений из заданного промежутка

```
void select() {
    while(1) {
        uint64_t val;
        uint64_t key_1 = mq_receive();
        if (key_1 == -1)
            break;

        uint64_t key_2 = mq_receive();
        if (key_2 == -1)
            break;

        get_nearest_value(key_2, val, false);
        get_nearest_value(key_1, val, true);

        if (key_1 && key_2) {
            uint64_t summ = val;

            for (uint64_t i = key_1; i < key_2;) {
                get_nearest_value(i, val, true);
                summ += val;
            }

            mq_send(summ);
        }
        else
            break;
    }
}
```

3 Пример работы программы

На рисунке 3.1 продемонстрирована работа программы.

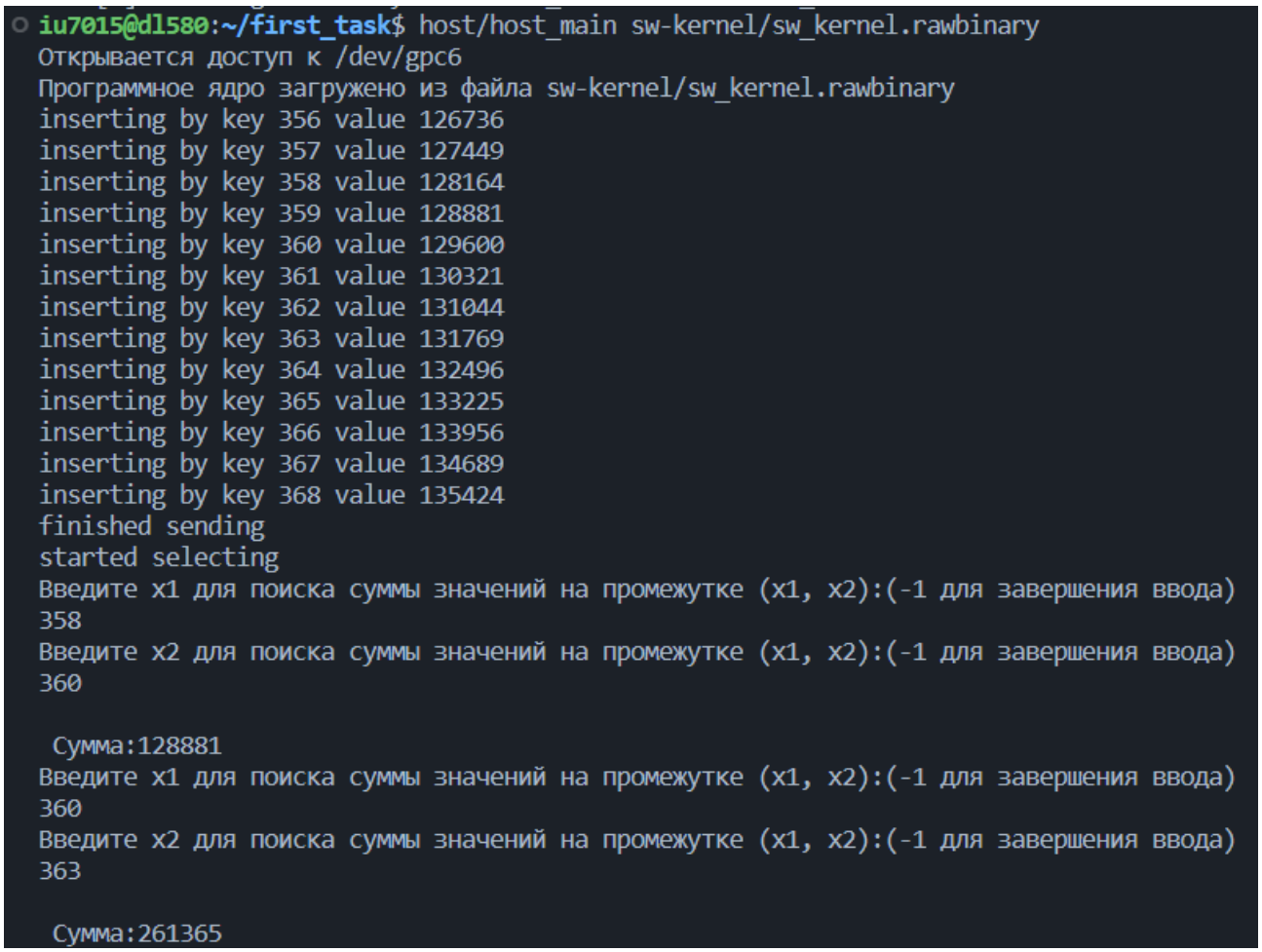

Рисунок 3.1 – Демонстрация работы программы

4 Заключение

В результате работы была разработана программа для хост–подсистемы и обработчики программного ядра, выполняющие действия по варианту.