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 и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по практикуму №1 по курсу «Архитектура ЭВМ»

Тема Разработка и отладка программ в вычислительном комплексе Терраграф

Студент Калашков П. А.

Группа ИУ7-56Б

Оценка (баллы)

Преподаватель Ибрагимов С. В. В.

Москва — 2022 г.

Задание 1

Практикум 1 выполнялся по варианту 4.

Цифровой интерполятор ЧПЗ. Сформировать в хост-подсистеме и передать в SPE 256 значений x и функции f(x)=sin(x), имеющие тип double (где x - ключ, f(x) - значение). Для представления чисел double в целочисленном диапазоне использовать функции double ull2double(uint64_t) и uint64_t double2ull(double), входящие в библиотеку sw_kernel-lib. Для случайного значения, сформированного в хост-подсистеме выполнить поиск ближайшего большего, и передать его в хост-подсистему. Выполнить тестирование работы SPE, сравнив результат с ожидаемым.

Для выполнения данного задания были изменены файлы с кодом примера

В листинге 1 представлено создание и передача данных из буфера.

Листинг 1 – Создание и передача данных из буфера

```
1     __foreach_core(group, core)
2 {
3     for (int i=0;i<BURST;i++) {
4         host2gpc_buffer[group][core][2*i] = f_rand(0, 10);
5         host2gpc_buffer[group][core][2*i+1] = rand_number;//i;
6         printf("key: %lf, value: %lf\n", host2gpc_buffer[group][core
              ][2*i], host2gpc_buffer[group][core][2*i+1]);
7     }
8 }
9
10 __foreach_core(group, core) {
11     lnh_inst.gpc[group][core]->start_async(__event__(insert_burst));
12 }
13
14 __foreach_core(group, core) {
15     lnh_inst.gpc[group][core]->buf_write(BURST*2*sizeof(double),(
          char*)host2gpc_buffer[group][core]);
16 }
17
18 __foreach_core(group, core) {
19     lnh_inst.gpc[group][core]->mq_send(IP);
20 }
```

В строках 2–8 создаётся буфер с данными, который далее передаётся в глобальную память и загружается в Тераграф с помощью функции *insert_burst* (листинг 2).

Листинг 2 – Создание и заполнение графа

```
1  void insert_burst() {
2
3      lnh_del_str_sync(TEST_STRUCTURE);
4      unsigned int count = mq_receive();
5      unsigned int size_in_bytes = 2*count*sizeof(uint64_t);
6      uint64_t *buffer = (uint64_t*)malloc(size_in_bytes);
7      buf_read(size_in_bytes, (char*)buffer);
8      for (int i=0; i<count; i++) {
9          lnh_ins_sync(TEST_STRUCTURE, buffer[2*i], buffer[2*i+1]);
10     }
11     lnh_sync();
12     free(buffer);
13 }
```

При этом используется функция *mq_send*, выполняющаяся на ядре (листинг 3).

Листинг 3 – Создание и заполнение графа

```
1          void get_interface()
2          {
3              lnh_sync();
4              unsigned int key = mq_receive();
5              lnh_search(TEST_STRUCTURE, key);
6              mq_send(lnh_core.result.value);
7      }
```

Исходя из переданных данных, производится интерполяция (листинг 4).

Листинг 4 – Интерполяция переданных величин

```
void search_burst() {
    lnh_sync();
    unsigned int count = lnh_get_num(TEST_STRUCTURE);
    unsigned int size_in_bytes = 2*count*sizeof(double);
    double delta = 0;
    double closest = DBL_MAX;
    double *buffer = (double*)malloc(2 * sizeof(double));
    double *old_buffer = (double*)malloc(size_in_bytes);
    double min_delta = DBL_MAX;
    lnh_get_first(TEST_STRUCTURE);
    for (int i=0; i<count; i++) {
        old_buffer[2*i] = ull2double(lnh_core.result.key);
        old_buffer[2*i+1] = ull2double(lnh_core.result.value);
        lnh_next(TEST_STRUCTURE,lnh_core.result.key);
    }
    double number_to_interpolate = old_buffer[1];
    for (int i=0; i<count; i++) {
        double cur = old_buffer[2*i];
        if (old_buffer[1] >= old_buffer[2*i])
        delta = number_to_interpolate - cur;
        else
        delta = cur - number_to_interpolate;
        if (delta < min_delta) {
            min_delta = delta;
            closest = cur;
        }
    }
    buffer[0] = old_buffer[1];
    buffer[1] = closest;
    buf_write(size_in_bytes, (char*)buffer);
    mq_send(count);
    free(buffer);
    free(old_buffer);
}
```