|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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 **ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04** Программная инженерия

**Отчет**

|  |  |
| --- | --- |
| **По практикуму № 1** |  |

*«Разработка и отладка программ в вычислительном комплексе Тераграф»*

**Дисциплина:** Архитектура ЭВМ

**Вариант:** 17

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | ИУ7-51Б |  | Савинова М. Г. |
|  | (Группа) |  | (Ф. И. О) |
| Преподаватель | Ибрагимов С. В. |  |  |

*2023 г.*

Оглавление

[Демо проект 1 3](#_Toc154331841)

[Демо проект 2 3](#_Toc154331842)

[Демо проект 3 4](#_Toc154331843)

[Индивидуальное задание 4](#_Toc154331844)

[Заключение 7](#_Toc154331845)

Демо проект 1

Пример демонстрирует основные механизмы инициализации гетерогенных ядер *gpc* и взаимодействие хост-подсистем с *Graph Processor Core*, используются аппаратные очереди.

1. Установка репозитория:

|  |
| --- |
| **git clone** --recursive https:**//**latex.bmstu.ru**/**gitlab**/**hackathon2023**/**lab1**/**lab1.git  **cd** lab1 |

1. Сборка проекта:

|  |
| --- |
| **make** |

1. Запуск:

|  |
| --- |
| host**/**host\_main sw-kernel**/**sw\_kernel.rawbinary |

1. Результат работы теста:

|  |
| --- |
| Open gpc on **/**dev**/**gpc0  Rawbinary loaded from sw-kernel**/**sw\_kernel.rawbinary  sw\_kernel version: 0x28102023  Leonhard clock frequency **(**LNH\_CF**)** 180.032751 MHz  Test **done** |

Демо проект 2

Программа демонстрирует принципы взаимодействия устройств в системе и примеры хранения и обработки множеств в микропроцессоре *Lnh64*.

1. Установка репозитория:

|  |
| --- |
| **git clone** --recursive https:**//**latex.bmstu.ru**/**gitlab**/**hackathon2023**/**lab2**/**lab2.git  **cd** lab2 |

1. Сборка проекта:

|  |
| --- |
| **make** |

1. Запуск:

|  |
| --- |
| host**/**host\_main sw-kernel**/**sw\_kernel.rawbinary  **select** role from **users** where user=5 and **time>**7200; |

1. Результат работы теста:

|  |
| --- |
| Роль: 999 - Время доступа: 3596400  Роль: 998 - Время доступа: 3592800  Роль: 997 - Время доступа: 3589200  Роль: 996 - Время доступа: 3585600  Роль: 995 - Время доступа: 3582000  Роль: 994 - Время доступа: 3578400 |

1. Просмотр лога:

|  |
| --- |
| **tail** -f lab2.log  Открывается доступ к **/**dev**/**gpc2  Программное ядро загружено из файла sw-kernel**/**sw\_kernel.rawbinary  Введен запрос: **select** role from **users** where user=5 and **time>**7200;  Запрос принят в обработку.  Поиск ролей пользователя 5и **time** **>** 7200 |

Демо проект 3

Пример демонстрирует основные механизмы взаимодействия микропроцессоров *CPE (resv64)* и *SPE (lnh64)*, и выполняет тестирование корректности команд *DISC.*

1. Установка репозитория:

|  |
| --- |
| **git clone** --recursive https:**//**latex.bmstu.ru**/**gitlab**/**hackathon2023**/**lab3.git  **cd** lab3 |

1. Сборка проекта:

|  |
| --- |
| **make** |

1. Запуск:

|  |
| --- |
| host**/**host\_main sw-kernel**/**sw\_kernel.rawbinary |

1. Результат работы теста:

|  |
| --- |
| Открывается доступ к /dev/gpc1  Программное ядро загружено из файла sw-kernel/sw\_kernel.rawbinary  Ядро /dev/gpc1. Эксперимент 1 - тест вставки и поиска. Ошибок: 0  Ядро /dev/gpc1. Эксперимент 2 - тест вставки в заполненную ассоциативную память. Ошибок: 0  Ядро /dev/gpc1. Эксперимент 3 - тест команд ближайшего меньшего и большего (nsm и ngr). Ошибок: 0  … |

Индивидуальное задание

* **Устройство вычисления обратной функции.** Сформировать в хост-подсистеме и передать в SPE 256 записей key-value со значениями функции f(x)=x^2 в диапазоне значений x от 0 до 1048576 (где f(x) - ключ, x - значение). Выполнить тестирование работы устройства, посылая из хост-подсистемы значение f(x) и получая от sw\_kernel значение x. Если указанного значения f(x) не сохранено в SPE, выполнить поиск ближайшего (меньшего или большего) значения к f(x) и вернуть соответствующий x. Сравнить результат с ожидаемым.
* 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*  funcs\_pnum = 1, *//Таблица 1*  resources\_pnum = 2 *//Таблица 2*  };    #ifdef \_\_riscv64\_\_  *//Задание даипазонов и курсоров*  template<typename Range>  struct reverse {  Range r;  [[gnu::always\_inline]] reverse(Range r) : r(r) {}  [[gnu::always\_inline]] auto begin() {return r.rbegin();}  [[gnu::always\_inline]] auto end() {return r.rend();}  };    template<typename K, typename V>  struct Handle {  bool ret\_val;  K k{get\_result\_key<K>()};  V v{get\_result\_value<V>()};  [[gnu::always\_inline]] Handle(bool ret\_val) : ret\_val(ret\_val) {  }    [[gnu::always\_inline]] operator bool() const {  return ret\_val;  }    [[gnu::always\_inline]] K key() const {  return k;  }    [[gnu::always\_inline]] V value() const {  return v;  }  };  #endif      *//////////////////////////////////////*  *// Описание формата ключа и значения*  *//////////////////////////////////////*      struct funcs {  using vertex\_t = uint32\_t;  int struct\_number;  constexpr funcs(int struct\_number) : struct\_number(struct\_number) {}      *//Запись для формирования ключей (\* - наиболее значимые биты поля)*  STRUCT(key)  {  uint64\_t f\_x :64;  };    *//Запись для формирования значений*  STRUCT(val)  {  uint64\_t x :64;  };  *//Обязательная типизация*  #ifdef \_\_riscv64\_\_  DEFINE\_DEFAULT\_KEYVAL(key, val)  #endif  };    constexpr funcs FUNCS(Structures::funcs\_pnum);    #endif //COMMON\_STRUCT |

* Измененный код функци *select* в *sw\_kernel\_main.cpp:*

|  |
| --- |
| void select() {  while(1){    uint64\_t cur\_f\_x = mq\_receive();    if (cur\_f\_x == -1)  **break**;    auto crole = FUNCS.search(funcs::key{.f\_x = cur\_f\_x});    uint64\_t result = crole.value();    if (!(crole))  {  uint64\_t small\_result = -1;  uint64\_t great\_result = -1;    *// меньший*  auto small = FUNCS.nsm(funcs::key{.f\_x = cur\_f\_x});  *// больший*  auto great = FUNCS.ngr(funcs::key{.f\_x = cur\_f\_x});    small\_result = small.value();  great\_result = great.value();    if (small && great)  if (abs(cur\_f\_x - small.key()) <= abs(great.key() - cur\_f\_x))  mq\_send(small\_result);  else  mq\_send(great\_result);    else if (small)  mq\_send(small\_result);  else  mq\_send(great\_result);  }  else  mq\_send(crole.value());  }  } |

* Пример работы программы:

|  |
| --- |
| Введите значение: 36  Ожидаемый результат: 6  Полученный результат: 6    Введите значение: 35  Ожидаемый результат: 5.91608  Полученный результат: 6    Введите значение: 25  Ожидаемый результат: 5  Полученный результат: 5    Введите значение: 26  Ожидаемый результат: 5.09902  Полученный результат: 5    Введите значение: 24  Ожидаемый результат: 4.89898  Полученный результат: 5    Введите значение: -1  Неверный ввод**!** |

Заключение

В результате работы изучены принципы работы вычислительного комплекса Тераграф и получены практические навыки решения задач обработки множеств на основе гетерогенной вычислительной структуры.