

Министерство образования Республики Беларусь

Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра ПОИТ

Дисциплина «Архитектура компьютерной техники и
операционных систем»

ОТЧЁТ
к лабораторной работе №4

ИЗУЧЕНИЕ СПОСОБОВ АДРЕСАЦИИ УСТРОЙСТВ ШИН PCI И
PCI EXPRESS. ОТРАБОТКА МЕТОДА ОПРОСА, ИДЕНТИФИКАЦИИ И
КОНФИГУРИРОВАНИЯ УСТРОЙСТВ

Вариант 8

Студент группы №351001
Ушаков А.Д.
Преподаватель
Леванцевич В.А.

Минск 2024

СОДЕРЖАНИЕ

Введение.....	3
1 Задания к работе.....	4
1.1 Общее задание.....	4
1.2 Индивидуальное задание.....	4
2 Реализация программы.....	5
2.1 Код программы.....	5
2.2 Запуск исполняемого файла.....	7
2.3 Пример результата работы программы	7
Заключение	8
Список использованных источников	9

ВВЕДЕНИЕ

Шина PCI являлась промежуточной шиной между ISA и PCI Express. С точки зрения программного управления шина PCI Express полностью аналогична шине PCI, однако, физическая реализация принципиально отличается.

PCI (Peripheral Component Interconnect) – параллельная шина ввода – вывода для соединения периферийных компонентов.. Она разрабатывалась в расчете на платформу Intel Pentium, но нашла применение и на более современных платформах. Первая версия PCI 1.0 появилась в 1992 г. В настоящее время развитие шины PCI приостановлено ввиду появления последовательно-параллельной шины ввода - вывода нового поколения - PCI Express. Для сохранения преемственности программный доступ к шине PCI Express аналогичен доступу к шине PCI.

Устройствами на шине PCI являются: контроллеры ввода-вывода, системные контроллеры, мосты, к которым могли подключаться дополнительные шины PCI.

Вначале шина PCI вводилась как дополнение к системам с основной шиной ISA, став позже центральной шиной: она соединяется с системной шиной процессора высокопроизводительным мостом (северным), входящим в состав чипсета системной платы. Остальные шины ввода-вывода (ISA), а также локальная ISA-подобная шина X-BUS и интерфейс LPC, к которым подключались микросхемы системной платы (ROM BIOS, контроллеры прерываний, клавиатуры, DMA, порты COM и USB, НЖМД и др.), подключаются к шине PCI через <южный> мост (рис.1).

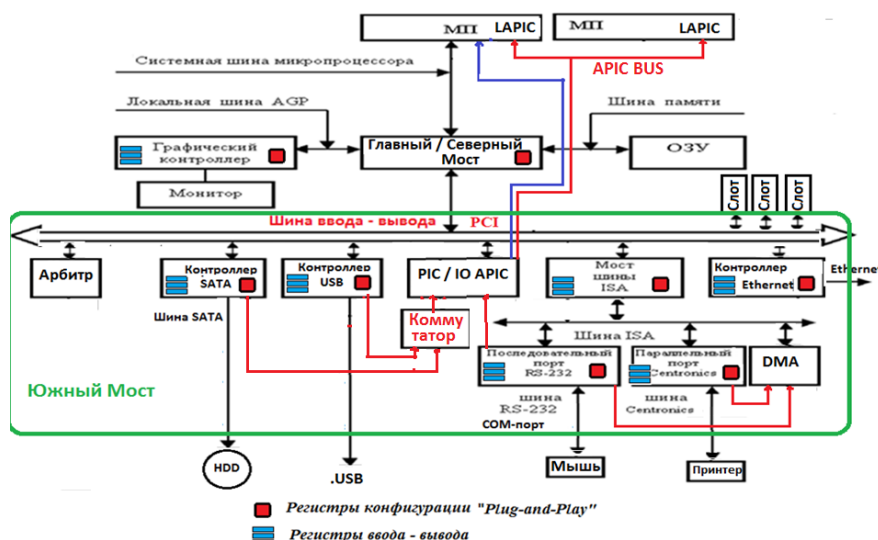


Рисунок 1 – Структура шины PCI

1 ЗАДАНИЯ К РАБОТЕ

1.1 Общее задание

В ходе лабораторной работы необходимо выполнить следующие пункты:

1 Изучить теоретические сведения о механизме конфигурирования устройств PCI и способам доступа к конфигурационному устройству.

2 Разработать программу формирования цикла опроса и идентификации устройств PCI, которая будет считывать два первых поля конфигурационного пространства - коды Vendor ID (производитель) и Device ID (устройство).

3 Используя файл PCI_DEVS.TXT или структуры, определенные в заголовочном файле pci_c_header.h, произвести расшифровку наименований производителей и устройств.

4 Результатом работы программы должна быть выводимая на экран или в текстовый файл таблица, содержащая следующую информацию:

- адрес устройства (номер шины, номер устройства и номер функции);
- 16-разрядный код производителя (в шестнадцатеричной системе);
- 16-разрядный код устройства (в шестнадцатеричной системе);
- производитель и название устройства.

1.2 Индивидуальное задание

К варианту 8 относятся:

1 Если устройство не мост (0-бит поля Header Type =0), вывести и расшифровать значение полей базовых регистров памяти.

2 Если устройство мост (0-бит поля Header Type =1), вывести и расшифровать значение полей I/O Base и I/O Limit.

3 Если устройство мост (0-бит поля Header Type =1), вывести и расшифровать значение поля Interrupt Pin и Interrupt Line.

2 РЕАЛИЗАЦИЯ ПРОГРАММЫ

2.1 Код программы

```
#include <stdio.h>
#include "pci_c_header.h"
#include <sys/io.h>
#include <stdbool.h>

const int BusIter = 65536;
const int DevIter = 2048;
const int FuncIter = 256;
const unsigned int startAddr = 2147483648;
#define CONTOROLER_NOT_EXIT 0xFFFF;

const char *PrintVendorName(int VenId){
    for (int i = 0; i < PCI_VENTABLE_LEN;i++){
        if ( VenId == PciVenTable[i].VenId){
            printf("Vendor name: %s ;\n",PciVenTable[i].VenFull);
        }
    }
}

void PrintDeviceName(int DevId, int VenId){
    for (int i = 0; i < PCI_DEVTABLE_LEN;i++){
        if ( (DevId == PciDevTable[i].DevId) && (VenId ==
PciDevTable[i].VenId)){
            //printf(PciDevTable[i].Chip);
            printf("Chip name: %s ;\n",PciDevTable[i].ChipDesc);
        }
    }
}

bool IsBridge(unsigned int addr){

    outl(addr+0x0C,0x0CF8);
    unsigned int Temp = inl(0x0CFC);
    Temp = Temp >> 16;
    return Temp & 0x01 == 0x01 ? true : false;
}

void PrintIORegisters(unsigned int addr){

    outl(addr+0x1C,0x0CF8);
    unsigned int Temp = inl(0x0CFC);

    printf("I/O Base: %02X;\n",(Temp & 0x000000FF)) ;
    printf("I/O Limit: %02X;\n",((Temp >> 8)& 0x000000FF) );
}
```

```

}

void PrintInterruptInfo(unsigned int addr) {
    outl(addr + 0x3C, 0x0CF8);
    unsigned int Temp = inl(0x0CFC);

    printf("Interrupt Pin: %02X;\n", Temp & 0x000000FF);
    printf("Interrupt Line: %02X;\n", (Temp >> 8) & 0x000000FF);
}

void PrintMemoryRegisters(unsigned int addr) {
    int regNumb = 0;
    for(unsigned int regAddr = 0x10; regAddr <= 0x24; regAddr += 0x04)
    {
        outl(addr + regAddr, 0x0CF8);
        unsigned int Temp = inl(0x0CFC);
        printf("Base Address Register %i: %08X;\n", regNumb, Temp);
        ++regNumb;
    }
}

void PrintControlerInfo(int BusNumb, int DevNumb, int FuncNumb) {
    unsigned int addr = (1 << 31) + (BusNumb << 16) + (DevNumb << 11) +
    (FuncNumb << 8);

    outl(addr, 0x0CF8);
    unsigned int DevId = inl(0x0CFC);

    if ((DevId >> 16) != 0xFFFF){

        unsigned int VendId = DevId & 0x0000FFFF;
        DevId = DevId >> 16;
        printf("Addr: %08X\n", addr);
        printf("Device Id: %04X;\nVendor Id: %04X;\n", DevId, VendId);
        PrintVendorName(VendId);
        PrintDeviceName(DevId, VendId);
        if (IsBridge(addr)){
            PrintInterruptInfo(addr);
            PrintIORegisters(addr);
        } else {
            PrintMemoryRegisters(addr);
        }
        printf("\n");
    }
}

int main(){

    if(iopl(3))

```

```

    {
        printf("I/O Privilege level change error: \nTry running
under ROOT user\n");
        return 1;
    }

    for (int i = 0; i < 256;i++){
        for (int j = 0; j < 32;j++){
            for (int k = 0; k < 8; k++){

                PrintControlerInfo(i,j,k);

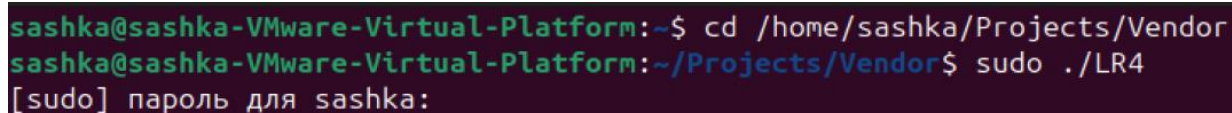
            }
        }
    }

    return 0;
}

```

2.2 Запуск исполняемого файла

Для запуска программы необходимо перейти в директорию /home/sashka/Projects/Vendor. После этого нужно запустить исполняемый файл с правами суперпользователя командой `sudo ./LR4` (см. рисунок 2.2.1).



```

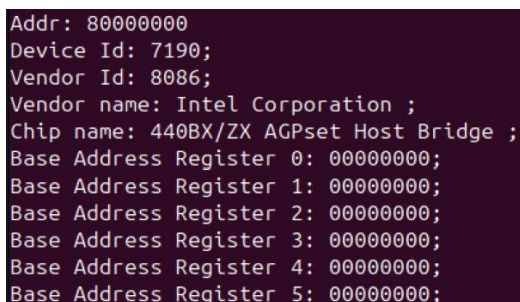
sashka@sashka-VMware-Virtual-Platform:~$ cd /home/sashka/Projects/Vendor
sashka@sashka-VMware-Virtual-Platform:~/Projects/Vendor$ sudo ./LR4
[sudo] пароль для sashka:

```

Рисунок 2.2.1 – Запуск исполняемого файла

2.3 Пример результата работы программы

На рисунке 2.3.1 представлен пример найденного в ходе работы программы устройства.



```

Addr: 80000000
Device Id: 7190;
Vendor Id: 8086;
Vendor name: Intel Corporation ;
Chip name: 440BX/ZX AGPset Host Bridge ;
Base Address Register 0: 00000000;
Base Address Register 1: 00000000;
Base Address Register 2: 00000000;
Base Address Register 3: 00000000;
Base Address Register 4: 00000000;
Base Address Register 5: 00000000;

```

Рисунок 2.3.1 – Информация о найденном устройстве

ЗАКЛЮЧЕНИЕ

В стандарт PCI заложены возможности автоматического конфигурирования системных ресурсов (пространств памяти и ввода-вывода и линий запроса прерываний).

Автоматическое конфигурирование устройств (выбор адресов и прерываний) поддерживается средствами BIOS и ориентировано на технологию PnP.

После аппаратного сброса (или при включении питания) в PCI – устройствах, подключенных к шине, доступны для операций считывания и записи, только регистры пространства конфигурации. Доступ к этим регистрам осуществляется по специальным циклам шины Configuration Read и Configuration Write. В этих операциях устройства выбирают по индивидуальным сигналам IDSEL и сообщают о потребностях в необходимых системных ресурсах, таких как линия прерывания, пространство занимаемых адресов памяти и портов ввода/вывода. После распределения ресурсов, выполняемого программой конфигурирования (POST (*Power-On Self-Test*) – самотестирование после включения), в конфигурационные регистры устройства записываются параметры конфигурирования. Только после этого становится возможным доступ к устройствам по командам обращения к памяти и портам ввода-вывода.

Конфигурационное пространство мостов PCI: регистры в диапазоне адресов 00—17h полностью совпадают с регистрами обычного устройства PCI и описывают поведение и состояние моста на первичной шине, Заметим, что бит 2 регистра команд (Bus Master Enable) управляет возможностью трансляции транзакций с вторичной шины на первичную. Если этот бит обнулен, то мост не должен на вторичной стороне отзываться как целевое устройство в транзакциях записи/чтения памяти и ввода-вывода, поскольку он не сможет транслировать эти транзакции на первичную шину. Регистры BAR описывают только область специфических (зависящих от реализации) регистров моста.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Леванцевич, В. А. Изучение способов адресации устройств шин PCI и PCI Express. Обработка метода опроса, идентификации и конфигурирования устройств / В. А. Леванцевич // Лабораторные работы. – 2024. – №4.