УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе № 4

по предмету «Архитектура компьютерной техники и операционных систем»

Вариант 6

Выполнил:

Дедов Н.Ю.

гр. 251003

Проверил:

Аврамец Д.В.

Минск 2023

Исходный код программы в VSCode:

#include <stdio.h>

#include <sys/io.h>

#include <errno.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <values.h>

#include "pci.h"

#define MAX\_BUS 256

#define MAX\_DEVICE 32

#define MAX\_FUNCTIONS 8

#define ID\_REGISTER 0

#define DEVICEID\_SHIFT 16

#define BUS\_SHIFT 16

#define DEVICE\_SHIFT 11

#define FUNCTION\_SHIFT 8

#define REGISTER\_SHIFT 2

#define CONTROL\_PORT 0x0CF8

#define DATA\_PORT 0x0CFC

void printInfo(int bus, int device, int function);

bool isBridge(int bus, int device, int function);

long readRegister(int bus, int device, int function, int reg);

void outputGeneralData(int bus, int device, int function, int regData);

char \*getDeviceName(int vendorID, int deviceID);

char \*getVendorName(int vendorID);

void outputBusData(long regData);

void outputIOMemorySpaceBARData(long regData);

FILE \*out;

void outputBARsData(int bus, int device, int function) {

puts("3. Basic input/output registers:");

int flag = 1;

for (int i = 0; i < 6; i++) {

long regData = readRegister(bus, device, function, 4 + i);

if (regData) {

if ((regData & 1)) {

printf("Input/output register %d: ", i);

flag = 0;

outputIOMemorySpaceBARData(regData);

}

}

}

if (flag)

{

printf("There are no basic input/output registers.\n");

}

}

void outputIOMemorySpaceBARData(long regData) {

unsigned long reg1Data = regData - 1;

printf("%#lx\n", reg1Data);

}

void outputLineData(long regData) {

unsigned offset = 0x0C;

printf("\n5. Register 6 data: %x\n", regData);

outl(regData + offset, 0xCF8);

unsigned char reg = inl(0xCFC);

printf("Chache: %02X \n", reg);

}

void outputMemoryBaseData(long regData) {

printf("\n10. Register 8 data: %x\n", regData);

long memBase = regData & 0xFFFF;

long memLimit = (regData >> 16) & 0xFFFF;

printf("Memory base data: %x\n", memBase);

printf ("Memory limit data: %x \n", memLimit);

}

char \*getVendorName(int vendorID) {

for (int i = 0; i < PCI\_VENTABLE\_LEN; i++) {

if (PciVenTable[i].VendorId == vendorID) {

return PciVenTable[i].VendorName;

}

}

return NULL;

}

char \*getDeviceName(int vendorID, int deviceID) {

for (int i = 0; i < PCI\_DEVTABLE\_LEN; i++) {

if (PciDevTable[i].VendorId == vendorID && PciDevTable[i].DeviceId == deviceID) {

return PciDevTable[i].DeviceName;

}

}

return NULL;

}

void outputVendorData(int vendorID) {

char \*vendorName = getVendorName(vendorID);

printf( "Vendor ID: %04x, %s\n", vendorID, vendorName ? vendorName : "Unknown vendor");

}

void outputDeviceData(int vendorID, int deviceID) {

char \*deviceName = getDeviceName(vendorID, deviceID);

printf( "Device ID: %04x, %s\n", deviceID, deviceName ? deviceName : "Unknown device");

}

void outputGeneralData(int bus, int device, int function, int regData) {

printf( "%x:%x:%x\n", bus, device, function);

int deviceID = regData >> DEVICEID\_SHIFT;

int vendorID = regData & 0xFFFF;

outputVendorData(vendorID);

outputDeviceData(vendorID, deviceID);

}

long readRegister(int bus, int device, int function, int reg) {

long configRegAddress = (1 << 31) | (bus << BUS\_SHIFT) | (device << DEVICE\_SHIFT) | (function << FUNCTION\_SHIFT) | (reg << REGISTER\_SHIFT);

outl(configRegAddress, CONTROL\_PORT);

return inl(DATA\_PORT);

}

bool isBridge(int bus, int device, int function) {

long headerTypeRegData = readRegister(bus, device, function, 3);

return ((headerTypeRegData >> 16) & 0xFF) & 1;

}

void printInfo(int bus, int device, int function) {

long idRegData = readRegister(bus, device, function, ID\_REGISTER);

if (idRegData != 0xFFFFFFFF) {

outputGeneralData(bus, device, function, idRegData);

if (isBridge(bus, device, function)) {

printf("\nA BRIDGE\n");

outputMemoryBaseData(readRegister(bus, device, function, 8));

}

else {

printf("\nNOT A BRIDGE\n\n");

outputBARsData(bus, device, function);

outputLineData(readRegister(bus, device, function, 6));

}

puts("\n---------------------------------------------------");

}

}

int main() {

if (iopl(3)) {

printf("I/O Privilege level change error: %s\nTry running under ROOT user\n", strerror(errno));

return 2;

}

puts("\n---------------------------------------------------");

for (int buses = 0; buses < MAX\_BUS; buses++) {

for (int device = 0; device < MAX\_DEVICE; device++) {

for (int function = 0; function < MAX\_FUNCTIONS; function++) {

printInfo(buses, device, function);

}

}

}

fclose(out);

return 0;

}

Результат выполнения программы:





