ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

MÔN HỌC: HỆ ĐIỀU HÀNH

ĐỀ TÀI: LẬP TRÌNH NACHOS

Giáo viên hướng dẫn:

ThS. Lê Viết Long

Thành phố Hồ Chí Minh, tháng 11 năm 2022

LỜI CẢM ƠN

Đầu tiên, nhóm em xin chân thành cảm ơn thầy ThS. Lê Viết Long, người đã hướng dẫn tận tình, giúp đỡ cho nhóm em hoàn thành đồ án này.

Bài tập đồ án lần này nhóm em đã cố gắng hết sức để hoàn thành nhưng cũng không tránh khỏi những sai sót. Vì thế em rất mong sự góp ý, chỉ bảo từ thầy cho em sau này có thể phát triển tốt hơn nữa.

Em rất mong nhận được sự quan tâm, giúp đỡ của thầy. Xin chân thành cảm ơn!

MỤC LỤC

1. Tổng quan

1.1. Thông tin thành viên trong nhóm:

- Phan Phong Lưu – 20120326
- Phạm Bảo Huy – 20120298
- Ngô Võ Quang Huy – 20120295

1.2. Bảng phân công công việc:

Thành viên / Công việc	Ngô Võ Quang Huy 20120295	Phạm Bảo Huy 20120298	Phan Phong Lưu 20120326
Viết lại file exception.cc		x	
Viết cấu trúc điều khiển chương trình	x		x
Viết hàm gia tăng Programming Counter		x	
Cài đặt System call: ReadInt, PrintInt, Ascii		x	
Cài đặt System call: ReadChar, PrintChar, Bubblesort			x
Cài đặt System call: ReadString, PrintString, Help	x		
Demo			x
Báo cáo	x		

1.3. Đánh giá đồ án

Yêu cầu	Mức độ hoàn thành (%)	Ghi chú
Viết lại file exception.cc	100	
Viết lại cấu trúc điều khiển của chương trình để nhận Nachos system call	100	
Viết mã để gia tăng program counter	100	
Cài đặt các system call: ReadInt(), PrintInt(), ReadChar, PrintChar(),ReadString(), PrintString()	100	
Viết các chương trình: help, sort, ascii	100	
Báo cáo	100	

- Đánh giá tổng thể mức độ hoàn thành: 100%

1.4. Thông tin cơ bản:

- Ngôn ngữ lập trình: C++, C
- Công cụ hỗ trợ: VMWare Workstation 16 Pro
- Hệ điều hành: Ubuntu 14.06
- Môi trường lập trình: Visual Studio Code, version 1.35

2. Mô tả các bước thực hiện:

2.1. Thiết kế

2.1.1. Chuẩn bị

- Trong file ../code/threads/system.h và ../code/threads/system.cc, thực hiện khai báo, cấp phát và xoá vùng nhớ cấp phát một biến toàn cục thuộc lớp SynchConsole để hỗ trợ việc nhập và xuất màn hình *Console*.
- Trong file ./code/userprog/exception.cc, thực hiện cài đặt hai hàm `char* User2System` và `int System2User`.
    - Hàm `char* User2System`: sao chép vùng nhớ từ user sang system.

  - Hàm *int System2User*: sao chép vùng nhớ từ system sang user.
  - Hai hàm này sẽ giúp cho chức năng đọc ghi của chương trình. Khi đọc thì ta cần dùng hàm *User2System* và khi ghi thì ta dùng hàm *System2User*.
- Truy cập ./code/userprog/exception.cc, thực hiện chỉnh sửa hàm void ExceptionHandler:
  - Chỉnh sửa câu lệnh if thành câu lệnh switch…case để bắt và xử lí các exception.
  - Ví dụ:

```
void ExceptionHandler(ExceptionType which)
{
    int type = machine->ReadRegister(2);

    switch (which)
    {
    case NoException:
        return;
    case PageFaultException:
        DEBUG('a', "Shutdown, initiated by user program.\n");
        printf("No valid translation found");
        interrupt->Halt();
        break;
    case ReadOnlyException:
        DEBUG('a', "Write attempted to page marked \"read-only\".\n");
        printf("Write attempted to page marked \"read-only\".");
        interrupt->Halt();
        break;
    case BusErrorException:
        DEBUG('a', "Translation resulted in an invalid physical address.\n");
        printf("Translation resulted in an invalid physical address.");
        interrupt->Halt();
        break;
    case AddressErrorException:
        DEBUG('a', "Unaligned reference or one that was beyond the end of the address space.\n");
        printf("Unaligned reference or one that was beyond the end of the address space.");
        interrupt->Halt();
        break;
    case OverflowException:
        DEBUG('a', "Integer overflow in add or sub.\n");
        printf("Integer overflow in add or sub.");
        interrupt->Halt();
        break;
```

2.1.2. Giá trị các thanh ghi

- R2: Lưu mã syscall và lưu kết quả trả về của các syscall.
- R4: Lưu tham số thứ nhất.
- R5: Lưu tham số thứ hai.
- R6: Lưu tham số thứ ba.
- R7: Lưu tham số thứ tư.

2.1.3. Các bước cài đặt một System Call

- Bước 1:
  - Truy cập ./code/userprog/syscall.h:

- Chèn #define <tên system call> <giá trị thanh ghi>: dùng để define các system call để dùng trong file exception.cc

Vd: *#define SC_ReadInt 11*

- Khai báo prototype của hàm: <kiểu dữ liệu> <tên hàm>(<các tham số>);

Vd: *int ReadInt();*

- Bước 2:
  - Truy cập ./code/test/start.c và ./code/test/start.s, chèn:

.globl <tên prototype>

.ent <tên prototype>

<Tên prototype>

addiu $2,$0, <tên system call>

syscall

j $31

.end <tên prototype>

Ví dụ:

```
    .globl ReadInt
    .ent    ReadInt
ReadInt:
    addiu $2,$0,SC_ReadInt
    syscall
    j   $31
    .end ReadInt
```

- Bước 3:
  - Truy cập ./code/userprog/exception.cc, thêm vào câu lệnh case SyscallException, case <tên system call> và chương trình ở mức người dùng để thực hiện system call đó.

Ví dụ:

```
case SC_ReadChar: {
    // Input: Khong co
    // Output: Tra ve 1 ki tu doc duoc tu man hinh console.
    // Chuc nang: Doc 1 ki tu tu man hinh console
    int MAX = 256;
    char* buffer = new char[MAX + 1];
    int numOfChar = gSynchConsole->Read(buffer, MAX);

    char c = buffer[0]; // Note: Neu nhap hon 1 ki tu thi chi doc 1 ki tu dau tien
    /* Ghi 1 ki tu vao thanh ghi */
    machine->WriteRegister(2, c);
    delete[] buffer;
    break;
}
```

- Bước 4:
  - Truy cập ./code/test/Makefile
  - Chèn thêm chương trình vào dòng all

    all: halt shell matmult sort <tên file chương trình>

    Ví dụ:

    ```
    all: halt shell matmult sort num_io char_io string_io ascii bubblesort help
    ```

  - Chèn thêm sau dòng all

    <tên file>.o: <tên file>.c

    $(CC) $(CFLAGS) -c <tên file>.c

    <tên file>: <tên file>.o start.o

    $(LD) $(LDFLAGS) start.o <tên file>.o -o <tên file>.coff

    ../bin/coff2noff <tên file>.coff <tên file>

    Ví dụ:

    ```
    ascii.o: ascii.c
        $(CC) $(CFLAGS) -c ascii.c
    ascii: ascii.o start.o
        $(LD) $(LDFLAGS) start.o ascii.o -o ascii.coff
        ../bin/coff2noff ascii.coff ascii
    ```

- Bước 5:
  - Biên dịch lại nachos, cd tới ./nachos/nachos-3.4/code, gõ lệnh *make*

Ví dụ:

```
nvqh@ubuntu:~/Downloads/nachos/nachos-3.4/code$ make
```

- Bước 6:
  - Chạy thử chương trình, dùng lệnh:

./userprog/nachos -rs 1023 -x ./test/<tên chương trình>

Ví dụ:

```
nvqh@ubuntu:~/Downloads/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/ascii
```

2.2. Cài đặt

2.2.1. Cài đặt các exception

- Truy cập ./code/machine/machine.h, lấy danh sách các exception được lưu trong biến ExceptionType.
- Truy cập ./code/userprog/exception.cc, viết lại hàm *ExceptionHandler* để bắt và xử lí các exception này, có thể dùng if…else if…else hoặc dùng switch…case. Mỗi exception sau khi đã xử lí thêm lệnh interrupt->Halt() để trả quyền xử lí cho hệ điều hành.

2.2.2. Cài đặt hàm *void IncreasePC()*

- Chức năng của hàm là làm tăng Programming Counter để nạp lệnh tiếp theo để thực hiện.
- Cách xây dựng: lưu giá trị của PC hiện tại cho PC trước, nạp giá trị tiếp theo cho PC hiện tại, nạp giá trị kế tiếp cho PC kế tiếp.

2.2.3. Cài đặt các system call

2.2.3.1. ReadInt

- Tên hàm: *int ReadInt();*
- Chức năng: đọc số nguyên nhập từ màn hình console.
- Input: ∅
- Output: số nguyên
- Các bước cài đặt:
  - Kiểm tra số lượng chữ số vượt quá số nguyên 32 bits.
  - Kiểm tra số dương hay âm
  - Kiểm tra số nguyên hợp lệ ở dạng số thực
  - Kiểm tra tồn tại kí tự không hợp lệ (không phải số)
  - Chuyển đổi chuỗi đọc từ console về số
  - Kiểm tra giới hạn trên và giới hạn dưới của số nguyên 32 bits.
  - Hợp lệ tất cả, ghi vào thanh ghi.

2.2.3.2. PrintInt

- Tên hàm: *void PrintInt(int number);*

- Chức năng: ghi số nguyên ra màn hình console
- Input: number - số nguyên 32 bits.
- Output: ∅
- Các bước cài đặt:
    - Xử lí các trường hợp:
        - Số 0
        - Số âm – chuyển số âm về số dương
    - Đếm số chữ số của số nguyên đọc được.
    - Chuyển số nguyên 32 bits đọc từ console về dạng chuỗi buffer.
    - Nếu là số âm, chèn thêm kí tự '-' vào đầu số.
    - Gọi hàm Write của lớp SynchConsole để in kí tự.

2.2.3.3. ReadChar

- Tên hàm: *`char ReadChar();`*
- Chức năng: Đọc 1 kí tự từ màn hình console
- Input: ∅
- Output: 1 Kí tự
- Các bước cài đặt:
    - Kiểm tra input. Nếu input nhiều kí tự thì chỉ đọc kí tự đầu tiên.
    - Ghi kí tự đọc được vào thanh ghi

2.2.3.4. PrintChar

- Tên hàm: *`void PrintChar(char character);`*
- Chức năng: in 1 kí tự lên màn hình console
- Input: charater – kí tự cần in
- Output: ∅
- Các bước cài đặt:
    - Đọc 1 kí tự từ thanh ghi
    - Gọi hàm Write của lớp SynchConsole để in kí tự

2.2.3.5. ReadString

- Tên hàm: *`void ReadString(char buffer[], int length);`*
- Chức năng: đọc một chuỗi kí tự với buffer và độ dài của chuỗi
- Input: buffer – lưu chuỗi, length – độ dài chuỗi
- Output: ∅
- Các bước cài đặt:
    - Lấy địa chỉ chuỗi đã đọc được từ thanh ghi số 4
    - Lấy độ dài tối đa của chuỗi từ thanh ghi số 5
    - Sao chép địa chỉ chuỗi từ vùng nhớ User sang System bằng hàm *`User2System`*
    - Gọi hàm Read của lớp SynchConsole để đọc chuỗi
    - Sao chép địa chỉ chuỗi từ vùng nhớ System sang lại User bằng hàm *`System2User`*

2.2.3.6. PrintString

- Tên hàm: *`void PrintString(char buffer[]);`*
- Chức năng: in chuỗi kí tự đọc được từ màn hình console
- Input: buffer – mảng các kí tự
- Output: một chuỗi kí tự
- Các bước cài đặt:
    - Lấy địa chỉ chuỗi đã đọc được từ thanh ghi 4
    - Sao chép địa chỉ chuỗi từ vùng nhớ User sang System bằng hàm *`User2System`*
    - Đếm độ dài thật sự của chuỗi
    - Gọi hàm Write của lớp SynchConsole để in chuỗi kí tự đọc được.

2.2.3.7. Help

- Đường dẫn: ./code/test/help.c
- Chức năng: in ra thông tin của nhóm, mô tả cách chạy hai system call ascii và bubblesort.
- Input: ∅
- Output: ∅
- Các bước cài đặt:
    - Dùng hàm PrintString đã được cài đặt ở trên để in ra các thông tin.

2.2.3.8. Ascii

- Đường dẫn: ./code/test/ascii.c
- Chức năng: in ra bảng mã ascii
- Input: ∅
- Output: ∅
- Các bước cài đặt:
    - Chạy vòng lặp for, với i chạy từ 32 tới 126 (do console chỉ hiển thị được các kí tự trong khoảng này)
    - Dùng PrintString đã cài đặt in ra số thứ tự mã ascii và các kí tự tương ứng.

2.2.3.9. BubbleSort

- Đường dẫn: ./code/test/bubblesort.c
- Chức năng: sắp xếp tăng dần hoặc giảm dần mảng nhập vào
- Input: ∅
- Output: ∅
- Các bước cài đặt:
    - Cho user chọn tăng dần hoặc giảm dần
    - Yêu cầu user nhập số lượng phần từ, ít nhất là 1 và nhiều nhất là 100 phần tử. Nếu nhập không hợp lệ, cho người dùng nhập lại.
    - Nhập giá trị các phần tử
    - Dùng thuật toán Bubblesort để sắp xếp
    - In ra màn hình console mảng trước khi sắp xếp

- In ra màn hình console mảng sau khi sắp xếp

3. Demo

- Chi tiết phần demo, nhấn vào liên kết:

https://www.youtube.com/watch?v=t8uef85auoU

hoặc

https://clipchamp.com/watch/esCcrgDjGoT

Tài liệu tham khảo:

1. Tài liệu về project của thầy Lê Viết Long. https://drive.google.com/drive/folders/1aWQMUf8pBRBPd6ije7lP0E8rhJV9bEFO?usp=sharing
2. (2022) *Youtube.com*. Available at: https://www.youtube.com/watch?v=t0jtY1C129s&list=PLRgTVtca98hUgCN2_2vzsAAXPiTFbvHpO&ab_channel=Th%C3%A0nhChungNguy%E1%BB%85n (Accessed: 9 November 2022).
3. profile, V. (2013) *Lập trình syscall trên nachos*, *Dangkhoahome.blogspot.com*. Available at: http://dangkhoahome.blogspot.com/2013/03/trong-file-huong-dan-cua-thay-khong.html (Accessed: 9 November 2022).
4. *Nachos Project Guide, Duke University* (2022). Available at: https://users.cs.duke.edu/~chase/nachos-guide/guide/nachos.htm (Accessed: 9 November 2022).
5. (2022) *Student.cs.uwaterloo.ca*. Available at: https://student.cs.uwaterloo.ca/~cs350/common/NachosTutorialF06.pdf (Accessed: 9 November 2022).