\*\*\*\*nachos –rs 1234 –x ../test/halt để hiểu cái này thì đọc trong threads file system, hàm initialize

Các folder trong folder code và chức năng

Folder Machine:

+translate: dịch các địa chỉ ảo sang địa chỉ vật lý, hỗ trợ tìm vùng ảo tuyến tính hay tìm kiếm các ô trống được liên kết, nếu tìm thấy thì sử dụng còn không thì sẽ tạo 1 exception

Mình đã viết thêm 2 hàm của machine.h vào đây là machine:: User2System và machine::System2User (tham khảo 2 hàm này trong file pdf hướng dẫn)

machine:: User2System truyền vào là địa chỉ ảo (kiểu int) và độ dài của cái ô địa chỉ ảo đó, cấp cho biến kernelBuf vùng nhớ bằng limit + 1 rồi với mỗi virtAddr + i (i từ 0 -> limit – 1) mình sẽ đọc vùng nhớ ảo đó và ghi xuống system.   
 machine::System2User truyền vào địa chỉ ảo, độ dài và cái buffer chứa vùng đó, kiểm tra các điều kiện về độ dài sau đó ghi vào vùng địa chỉ ảo buffer đó

+interrupt: mô phỏng các quá trình ngắt phần cứng I/O, ngắt được kích hoạt khi nó được tái kích hoạt, hướng dẫn từ người dùng được thực hiện, không có gì trong hàng đợi

Chỉ có 2 trạng thái là tắt – off và bật – on, sau đó nachOS sẽ cho biết thời gian thực hiện, ghi, đọc vào vùng nhớ như thế nào và các sự truyền dữ liệu

Trong đây cũng có hàm Halt() viết sẵn để sử dụng (Hàm này sẽ tắt hoàn toàn Nachos, in ra hiệu thống kê.

Các hàm hỗ trợ thêm là ChangeLevel (thay đổi status của interrupts là On hay Off, SetLevel (đặt status của interrupts bằng trạng thái interrupts hiện tại xảy ra), enable (bật trạng thái interrupt thành On), OneTick (kiểm tra sau mỗi thời gian nhất định xem các Interrupts đang như thế nào và cái interrupts nào đang chờ được xử lý và sẽ chuyển đổi sang chỗ khác xử lý nếu được yêu cầu sau khi rountine của kernel hiện tại hoàn tất), Idle (máy tính đang không xử lý interrupt nào và nó sẽ kiểm tra xem có cái nào đang chờ xử lý không và đem ra xử lý), Schedule (hàm sẽ đặt thời gian xử lý cho interrupt nào đó mình quy định trong thời gian đặt trước), …

+Console: Kết nối Input (bàn phím), Output (màn hình) qua console. Nó xử lý bất đồng bộ để đảm bảo một byte đã đến hoặc văn bản đã sẵn sàng xử lý (ta phải gọi các interrupts để xử lý)

Console: hàm khởi tạo mô phỏng điều khiển phần cứng, các biến truyền vào readFile – chỉ tập tin được đọc, writeFile – chỉ tập tin được ghi ra, readAvil – xử lý ngắt khi nhập từ bàn phím, writeDone – xử lý ngắt để đảm bảo một kí tự đã được hoàn tất đưa ra thì các kí tự tiếp theo mới được xử lý

CheckCharAvail: kiểm tra định kỳ để xem có ký tự nào nhập từ bán phím chưa và chỉ xử lý khi ký tự trước đã xử lý xong và ra khỏi buffer. Sau đó sẽ gọi interrupt để đưa nó vào buffer

WriteDone: kiểm tra kí tự đã ra khỏi buffer chưa

GetChar: đọc kí tự từ input buffer nếu có kí tự tại đó

PutChar: in kí tự đó ra thiết bị hiển thị (và việc này sẽ được đặt thời gian xử lý trong tương lai gần do người dùng xử lý)

+stats: Thống kê về hiệu suất NachOS gồm hàm khởi tạo và in nó ra màn hình sau tắt hệ thống

+machine: xử lý user programs 2 hàm quan trọng ReadRegister, WriteRegister (đọc từ thanh ghi và ghi xuống thanh ghi

DumpState: cho biết nội dung các thanh ghi

Debugger: gỡ lỗi user programs, nhưng mà người sang tác ghi là nó khó sử dụng và để sử dụng thì cần phải sửa đổi, thực hiện rất nhiều cái khác để sử dụng hàm này

RaiseException: chuyển user mode sang nachOS kernel (exception, system call)

//TLB: translation lookaside buffer

Folder filesys:

+filesys: Quản lý hoạt động chung của hệ thống tập tin, có thể từ tên tập tin truyền vào mà tìm được nơi lưu trữ file đó và xử lý. Tập tin gồm tiêu đề lưu trữ trong một khu vực đĩa, các khối dữ liệu của nó, mục để nhập nó vào hệ thống tập tin. Hệ thống tập tin gồm một số cấu trúc dữ liệu: một bitmap của các sector đĩa, một thư mục tên tập và tiêu đề. Các tên file nằm trong các sector cụ thể giúp hệ thống tìm được nó ngay khi khởi động. Bitmap và thư mục được giữ luôn mở khi nachOS hoạt động, vì vậy nếu tạo hay hủy thì bitmap được sửa ngay lập tức, còn nếu tạo, xóa thất bại thì phiên bản thay đổi bị xóa mà không ghi lại vào đĩa. Một số hạn chế của tạo file trong đây: không truy cập đồng thời, tập không lớn hơn 3KB và được quy định khi tập tin được tạo ra, không có cấu trúc thư mục phân cấp, và chỉ có số lượng file nhất định thêm vào hệ thống và nếu NachOS thoát trong quá trình thao tác sửa đổi thì đĩa có thể bị hỏng

Filesystem (khởi tạo hệ thống tập tin, nếu format bằng TRUE thì hầu hết các sectors sẽ free)

Create (tạo file bằng cách truyền tên file và kích thước theo điều kiện file chưa tồn tại, đủ vùng nhớ trong file header, đủ vùng nhớ trong danh mục, đủ vùng nhớ cho các block dữ liệu của file

Open (mở file để đọc và ghi, truyền vào tên file), mình viết thêm 1 hàm Open để quy định kiểu xem file là đọc hay ghi

Remove (xóa file, thứ tự thực hiện: tìm file đó theo tên, xóa blocks dữ liệu, xóa header của block rồi ghi ngược lại vào đĩa)

List (cho biết toàn bộ file được lưu trong hệ thống file)

Print (cho biết thông tin về file: nội dung bitmap, nội dung danh mục, mội file trong đó có nội dung header như thế nào và dữ liệu trong file)

+filehdr: quản lý tiêu để tập tin (xác định vị trí dữ liệu file trên disk), tổ chức theo kiểu bảng các con trỏ, chỉ ra khu vực nào trên disk có chứa phần dữ liệu của tập tin đó

Allocate: Tạo một file header cho tập tin được tạo ra, phụ thuộc vào free disk blocks, tham số truyền vào freeMap chỉ bitmap của free disk sectors và fileSize chỉ kích thước

Deallocate: phân bố không gian cho blocks dữ liệu cho file

FetchFrom: Tìm nạp nội dung của tiêu đề tệp tin từ đĩa, nó còn nói là trong các “sector” bao gồm cả file header

WriteBack: sửa đổi nội dung file header trở lại đĩa

ByteToSector: Trả về khu vực đang lưu trữ byte đó, chuyển đổi từ địa chỉ ảo sang địa chỉ vật lý

ByteLength: trả về số lượng byte của file

Print: In ra nội dung của file header và nội dung toàn bộ blocks dữ liệu quản lý bởi file header

Folder threads:

+system: nachOS xử lý và dọn dẹp các rountines

TimerInterruptHandler: xử lý ngắt định kỳ theo TimerTicks, chỉ xử lý khi thread bị gián đoạn vào thời điểm đó

Initialize: Xử lý câu lệnh nhập vào, hàm khó quá đọc không hiểu

CleanUp: halting lại nachOS, hủy bỏ cấu trúc thiết lập global (xóa NetWork, các thư viện chạy chương trình machine, gSynchConsole, xóa filesystem, synchDisk và xóa timer, scheduler, interrupt sau đó exit

+list: giống linked list

Folder userprog:

+bitmap: quản lý bitmap

BitMap: khởi tạo bitmap với số lượng bit cho trước, và thêm bit vào đâu đó trong danh sách

Mark: đặt một số lượng bit vào bitmap

Clear: xóa một số lượng bit nào đó khỏi bitmap

Test: kiểm tra một số bit xem được cài đặt chưa

Find: Trả về số lượng bit được clear (đang trống trải) và sau đó mark vào lại

NumClear: tìm số lượng bit đang trống trong bitmap

Print: in ra thông tin của bitmap

FetchFrom: Khởi tạo bitmap từ 1 file của nachOS

WriteBack: lưu trữ lại nội dung của 1 bitmap từ 1 file của nachOS

+exception: Xử lý các interrupts từ syscall.h (interrupts từ user programs) mà syscall.h đã quy định entry point cho nó. Các xử lý này do mình code vào và khi gmake all sẽ ra các exception khác mà có chữ warning mình cần thêm vào, sau mỗi exception thì nên tăng thanh ghi PC lên để đọc lệnh tiếp theo và xử lý tránh halting lại nachOS

+syscall: quy định các entry point mà được xử lý khi user programs gọi vào, có sẵn 1 số cái như tạo file - Create, mở file - Open, viết vào file đang mở một dung lượng bytes nhất định – Write, đọc một dung lượng bytes từ file đang mở - read, đóng file – Close, xử lý đa luồng – fork và yield