* progtest.cc:
  + StartProcess(char \*filename): chạy chương trình, mở tập tin thực thi, lưu nó vào bộ nhớ và nhảy tới nó
  + Console \*console, Semaphore \*readAvail, Semaphore \*writeDone: các cấu trúc dữ liệu cần cho việc chạy thử, làm cho yêu cầu nhập/ xuất dữ liệu phải chờ ở ‘Semaphore’ cho tới khi việc nhập/xuất hoàn thành
  + ReadAvail(int arg), WriteDone(int arg): Đánh thức thread yêu cầu nhập/xuất
  + ConsoleTest (char \*in, char \*out): truyền từ đã nhập vào và xuất ra từ đó, dừng khi người dùng nhập ‘q’(quit)
* syscall.h:
  + Halt(): dừng nachos và cho biết tình hình thực hiện
  + Exit(int status):Thoát bình thường nếu status=0
  + Exec(char \*name): khởi chạy tập tin, lưu file Nachos tên “name” và trả về không gian địa chỉ
  + Join(SpaceId id): Thoát một khi chương trình người dùng với id ‘id’ đã xong
  + Create(char \*name): Tạo file nachos tên ‘name’
  + Open(char \*name): mở file nachos tên ‘name’ và trả về 1 id để dùng vào việc đọc và viết trên file đó
  + Write(char \*buffer, int size, OpenFileId id): viết các bytes với số lượng ‘size’ từ ‘buffer’ ở file với id ‘id’
  + Read(char \*buffer, int size, OpenFileId id): đọc các bytes với số lượng “size” và lưu vào ‘buffer’ và trả về số lượng bytes đã đọc
  + Close(OpenFileId id): đóng file với id ‘id’
  + Fork(void (\*func)()) rẽ nhánh một thread để chạy qui trình ‘\*func’ ở cùng một không gian địa chỉ với thread hiện tại
  + Yield(): đưa CPU sang một thread khác dù có là không gian địa chỉ hay không
* exception.cc:
  + ExceptionHandler(ExceptionType which): ở đầu vào nhân Nachos, được gọi khi chương trình được thực thi hoặc là syscall hoặc khi tạo ra ngoại lệ ‘which’, các ngoại lệ được lưu ở machine.h
* bitmap.\*:
  + BitMap(int nitems): khởi tạo bitmap với ‘nitems’ bits và mỗi bits đã được clear
  + ~BitMap():xóa và đóng bitmap
  + Mark(int which): set bit thứ ‘which’
  + Clear(int which): clear bit thứ ‘which’
  + Test(int which): trả về TRUE nếu bit đó đã được set và FALSE nếu chưa
  + Find(): trả về số thứ tự bit đầu tiên trống và set bit đó, nếu không có bit nào trống, trả về -1
  + NumClear(): trả về số lượng bit trống
  + Print(): in ra bitmap
  + FetchFrom(OpenFile \*file): khởi tạo giá trị của bitmap từ file nachos tên ‘file’
  + WriteBack(OpenFile \*file): lưu giá trị của bit map vào file nachos tên ‘file’
* filesys.\*:
  + FileSystem(bool format): khởi tạo hệ thống file, nếu ‘format’ = TRUE, ổ đĩa trống thì khởi tạo ổ đỉa với 1 thư mục trống và một bitmap gồm các sectors trống(không hoàn toàn trống), nếu ‘format’ = FALSE thì sẽ mở file và trình ra bitmap và thư mục
  + Create(char \*name, int initialSize): Tạo một file trong hệ thống file nachos tên ‘name’ và size ‘initialSize’
  + Open(char \*name): mở file tên ‘name’
  + Remove(char \*name): xóa file với tên ‘name’ khỏi hệ thống file
  + List(): liệt kê các file trong thư mục hệ thống file
  + Print(): in mọi thứ về hệ thống file(nội dung của bitmap, nội dung của thư mục[nội dung của file header,dữ liệu trong file])
* openfile.h:
  + OpenFile(int sector): mở file nachos sử dụng ‘sector’(vị trí của file header của file này trên ổ đĩa)
  + ~OpenFile(): đóng file nachos xóa các cấu trúc dữ liệu còn tồn động
  + Seek(int position): di chuyển con trỏ đến vị trí ‘position’ để thực hiện việc đọc/viết tiếp theo
  + Read(char \*into, int numBytes), Write(char \*into, int numBytes): viết đọc file bắt đầu từ seekPosition,thay đổi vị trí con trỏ sau khi đọc/viết và trả về số bytes đã đọc/viết
  + ReadAt(char \*into, int numBytes, int position),WriteAt(char \*from, int numBytes, int position):giống hàm Read/Write nhưng không thay đổi vị trí con trỏ sau khi đọc/viết và bắt đầu từ vị trí position người dùng chọn
  + Length(): trả về số lượng bytes trong file
* translate.\*:
  + WordToHost(unsigned int word), ShortToHost(unsigned short shortword), WordToMachine(unsigned int word), ShortToMachine(unsigned short shortword): dịch từ word/short sang ngôn ngữ máy
  + ReadMem(int addr, int size, int \*value): đọc số byte của bộ nhớ ảo tại địa chỉ ‘addr’ vào vị trí được chỉ tới bởi ‘value’
  + WriteMem(int addr, int size, int value): viết ‘size’ bytes thuộc nội dung của ‘value’vào bộ nhớ ảo tại địa chỉ ‘addr’
  + Translate(int virtAddr, int\* physAddr, int size, bool writing): dịch địa chỉ ảo ‘virtAddr’ sang địa chỉ thực sử dụng page table hoặc TLB và lưu vào ‘physAddr’,nếu có lỗi thì trả về loại exception
* machine.\*:
  + CheckEndian(): kiểm tra rằng có phải host đang sử dụng ngôn ngữ đã nêu từ trước không, nếu không phải thì dừng chương trình và báo lỗi
  + Machine(bool debug):triển khai giả lập của việc thực thi chương trình của người dùng
  + ~Machine(): giải phóng bộ nhớ khỏi các cấu trúc dữ liệu đã sử dụng
  + RaiseException(ExceptionType which, int badVAddr): chuyển quyền kiểm soát về nhân của Nachos vì chương trình của người dùng có liên quan đến system call hoặc điều ngoại lệ nào đó đã xảy ra trong lúc chạy chương trình
  + Debugger(): debug chương trình của người dùng
  + DumpState(): in ra tình trạng CPU của hệ thống
  + ReadRegister(int num), WriteRegister(int num, int value): đọc/viết nội dung của thanh ghi hệ thống của người dùng
* mipssim.cc:
  + Run(): mô phỏng việc thực thi một chương trình có cấp độ người dùng trên Nachos,được gọi bởi nhân khi mà chương trình bắt đầu, không trả về
  + TypeToReg(RegType reg, Instruction \*instr): nhân về thanh ghi # như trong lệnh
  + OneInstruction(Instruction \*instr): thực hiện một lệnh từ chương trình có cấp độ user
  + DelayedLoad(int nextReg, int nextValue): mô phỏng hiệu ứng của delayed load
  + Decode(): giải mã lệnh MIPS
  + Mult(int a, int b, bool signedArith, int\* hiPtr, int\* loPtr): mô phỏng R2000 multiplication
* console.\*:
  + Console(char \*readFile, char \*writeFile, VoidFunctionPtr readAvail,VoidFunctionPtr writeDone, int callArg): khởi tạo mô phỏng thiết bị điều khiển phần cứng
  + ~Console(): giải phóng bộ nhớ
  + CheckCharAvail(): kiểm tra định kỳ nếu ký tự sẵn sàng để được nhập từ bàn phím ảo
  + WriteDone(): Báo về nhân Nachos khi ký tự xuất đã hoàn thành
  + GetChar():đọc ký tự từ bộ đệm nhập nếu có và trả về ký tự đó hoặc EOF khi không ký tự nào được buffered
  + PutChar(char ch): viết một ký tự vào mô phỏng hiển thị,xắp xếp một đoạn ngắt quảng về sau và trả về
* synchcons.\*:
  + Semaphore \*synchReadAvail, Semaphore \*synchWriteAvail, Semaphore \*RlineBlock, Semaphore \*WLineBlock: các cấu trúc dữ liệu cần cho việc chạy thử, làm cho yêu cầu nhập/ xuất dữ liệu phải chờ ở ‘Semaphore’ cho tới khi việc nhập/xuất hoàn thành
  + SynchReadFunct(int arg), SynchWriteFunct(int arg): Đánh thức thread yêu cầu nhập/xuất
  + SynchConsole():Tạo thiết bị điều kiển đồng bộ với input/output căn bản
  + SynchConsole(char \*in, char \*out): Tạo thiết bị điều kiển đồng bộ với tên file
  + ~SynchConsole(): xóa thiết bị điểu khiển đồng bộ
  + Write(char \*from, int numBytes), Read(char \*into, int numBytes): viết/đọc số bytes vào/từ buffer của thiết bị input/output và trả về số bytes đã viết/đọc
* test:
  + matmult.c: triển khai ma trận và nhân với nhau
  + halt.c: sử dụng hàm halt để dừng chương trình lại
  + shell.c: Tạo shell
  + sort.c:Khởi tạo mảng 1 chiều lớn gồm dãy số sắp xếp ngược và xắp xếp dãy số đó
  + start.c:bắt đầu chương trình sử dụng hợp ngữ(Assembly languague)