Introduction
The goal of this project is for you to familiarize yourself with Nachos — both the operating system and the underlying emulated machine. This project consists mostly of reading and studying the Nachos code, though you will be asked to make some modifications.

Getting Started
By now, you should already have read Dr. Robert Walker's lecture 09, and the Birrell paper on Nachos. You should also have looked at "Appendix B" on the website, and compiled Nachos to produce an executable program "nachos" in the threads directory, and made sure it runs and produces the expected output.

Reading the Nachos Source Code
Now start reading the Nachos source code. I suggest that you read the files in the order described below, and as you do so, read the corresponding sections in Thomas Narten’s “A Road Map Through Nachos” and Archna Kalra's “Salsa — An Operating Systems Tutorial”.

To begin, read through the following files. When you compiled Nachos into the threads directory, the Makefile turned on the THREAD switch. Notice what code in these files is included when they are compiled using the THREAD switch, and what code is omitted when they are not compiled with the USER_PROGRAM, FILESYS, and NETWORK switch. Notice what command line arguments you can give to Nachos, and what global data structures are created.

- **threads/main.cc, threads/threadtest.cc** — a simple test of the thread routines.

- **threads/system.h, threads/system.cc** — Nachos startup/shutdown routines. Then read through the following files, and see how Nachos implements and schedules threads. Study the thread class, its private data, and its public member functions. Study the Scheduler class, and how it dispatches threads. Glance at the code for context switching, but don’t read it in detail

- **threads/thread.h, threads/thread.cc** — thread data structures and thread operations such as thread fork, thread sleep and thread finish.

- **threads/scheduler.h, threads/scheduler.cc** — manages the list of threads that are ready to run.

- **threads/switch.h, threads/switch.s** — assembly language magic for starting up threads and context switching between them. *Don’t worry if you don’t understand these two files — you are not responsible for understanding them.*

Next, skim through the following files, so you will recognize the functions when you encounter them elsewhere. After reading about DEBUG statements, go back through the files above, and see which debugging options may be useful when working with threads.

- **threads/list.h, threads/list.cc** — generic list management.
• threads/utility.h, threads/utility.cc — some useful definitions and debugging routines.

Now that you’ve gotten an overview of the Nachos operating system, it’s time to look at the emulated machine underneath. You don’t have to read this code in great detail, as you aren’t going to be modifying it, but you should familiarize yourself with it, because you can’t really understand the operating system unless you understand the hardware that it runs on. All of these files are in the machine directory. For now, you should read the files listed below, but you can ignore the files that describe the emulated console, disk, and network.

• machine/machine.h, machine/machine.cc — emulates the part of the machine that executes user programs: main memory, processor registers, etc.

• machine/mipssim.cc — emulates the integer instruction set of a MIPS R2/3000 CPU.

• machine/interrupt.h, machine/interrupt.cc — manage enabling and disabling interrupts as part of the machine emulation.

• machine/timer.h, machine/timer.cc — emulate a clock that periodically causes an interrupt to occur.

• machine/stats.h — collect interesting statistics.

Tracing Through and Debugging Nachos Source Code
One of the goals of this project is to read and understand the thread system in Nachos. Besides just reading the source code, you should also trace the execution path (as described below) for the simple test case provided.

To trace through code in Nachos, there are three main approaches: (1) using the gdb debugger, (2) using printf, and (3) using the DEBUG function provided by Nachos. The debugger gdb usually works, and is often the best alternative, although tracking across a call to switch can be confusing. If you are on a console or using vnc, then you can use the visual version xxgdb instead. The commands in both debuggers are very similar. Adding cout statements usually works, though you need to use endl or flush to make sure that the buffer is flushed, or you’ll get confusing results.

The final debugging option, which is particularly useful when working with threads, is to use the Nachos DEBUG function, which is declared in threads/utility.h. The command line options to Nachos are specified in threads/main.cc and threads/system.cc; if you look at those files you will see that the command line option for debugging is “–d”, which should be followed by a flag to tell Nachos which type of debugging messages to print (these flags are defined in threads/utility.h). To look at the various debugging statements that are included in the thread system in Nachos, execute the command “grep DEBUG *h *cc” in the threads directory — as you can see, all of the those debugging statements have the “t” flag. In the machine directory, the debugging statements have “i” and “m” flags. Putting all this together, you might want to run Nachos as “nachos -d i”, “nachos -d l”, or “nachos -d li” to see what your code is doing while working with threads. If you need more information, add more debugging statements (add your own debugging flag), or use the Nachos ASSERT function.

Overview of the Problems
The problems given below are intended to test your knowledge of the Nachos source code as you read through it in the order described above. They do not ask about everything in
the code, but if you read a piece of code and then can answer the corresponding questions easily, you should be well prepared with a basic overview of Nachos for future projects. However, if you go through the code as quickly as possible, merely searching for the answer to these questions instead of trying to understand the code, you may encounter difficulties later.

Write your answers to these questions in a text file named `p1.answers`. When you finish, you will email the file `p1.answers` to the TA as explained later.

The Problems

1. **These questions are concerned with the Nachos operating system.**
   
   a. When you compile as instructed, and run Nachos, why does the “–z” argument do something, but the “–x” argument doesn’t work?
   
   b. Where and how is the CPU scheduler started?
   
   c. What does ThreadTest do?
   
   d. What is in a thread’s “thread control block”?

2. **These questions are concerned with the emulated machine that runs underneath the Nachos operating system.**
   
   a. What CPU state is saved during a context switch? What function does this, and how many registers are saved?
   
   b. What does a hardware timer do? How is the Nachos hardware timer different from a “real” timer?
   
   c. What function allocates space for the main memory of the machine?
   
   d. What does the MIPS ADDU instruction do?

3. Compile and run Nachos and observe the output. Then modify “ThreadTest” to fork a second thread “u” named “second forked thread” immediately after it forks thread “t” named “forked thread”. What are the results when Nachos is compiled and run? Why does thread 0 run first, instead of thread 1, give that the fork of thread 1 occurs in ThreadTest before it calls SimpleThread(0)?

PROGRAM SUBMISSION

Programs are to be submitted electronically by email to walkerel@hiram.edu. To submit a file from Linux, simply type "mail -s "OS Project 1" walkerel@hiram.edu < myfile.cc". Submissions should include a text or word-processor file containing answers to the questions, as well as source code for any files from Nachos that you modified for question 3. Be sure to include comments identifying and explaining your modifications. Any submissions received after 23:59:59 (11:59:59pm) on Wednesday, September 27, 2000 will be subject to late penalties.