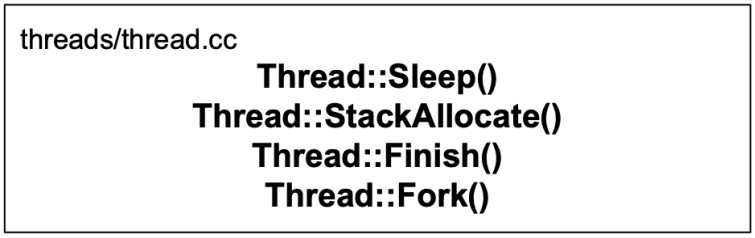
CS342301: Operating System
MP2: Multi-Programming
Deadline: 2021/11/14 23:59

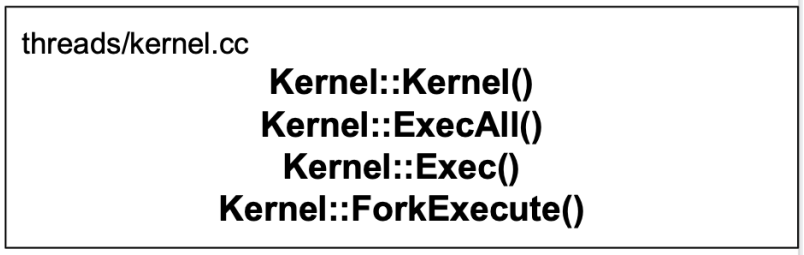
I. Goal

1. Understand how memory management works in NachOS
2. Understand how to implement page table mechanism

II. Assignment

1. Trace code
   - Starting from "threads/kernel.cc **Kernel::ExecAll()**", "threads/thread.cc **`thread::Sleep`**" until "machine/mipssim.cc **Machine::Run()**" is called for executing the first instruction from the user program.
   - You need to explain **at least** the function mentioned below in the report.





2. Implement page table in NachOS
   - Working item: Modify its memory management code to make NachOS support multi-programming.
   - Verification:
     - Wrong results without multi-programming

```
[ta@lsalab ~/2020/riya/MP2_test/code/test]$ ../build.linux/nachos -e consoleIO_test1 -e consoleIO_test2
consoleIO_test1
consoleIO_test2
9
16
15
18
19
1return value:0
7
return value:0
```

     - Correct results with multi-programming

```
[ta@lsalab ~/2020/riya/MP2_sol/code/test]$ ../build.linux/nachos -e consoleIO_test1 -e consoleIO_test2
consoleIO_test1
consoleIO_test2
9
8
7
6
1return value:0
5
16
17
18
19
return value:0
```

     - Correctly handle the exception about insufficient memory (i.e. **MemoryLimitException**)

```
[ta@localhost test]$ ../build.linux/nachos -e consoleIO_test1 -e consoleIO_test4
consoleIO_test1
consoleIO_test4
9Unexpected user mode exception 8
Assertion failed: line 201 file ../userprog/exception.cc
Aborted
```

   - **Requirement:**
     - Be careful that program size might exceed a pagesize
     - You must put the data structure recording used physical memory in kernel.h / kernel.cc
     - You must set up “valid, readOnly, use, and dirty” field for your page table, which is defined under “translate.h TranslationEntry class”
     - You must call ExceptionHandler to handle the exception when there is insufficient memory for a thread. (i.e. **MemoryLimitException**). Since there is no **MemoryLimitException** in **ExceptionType** class, you need to add it in **machine.h** and place it right before **NumExceptionTypes**. We will use hidden testcases to test this part.
     - When the thread is finished, make sure to release the address space and restore physical page status.

- **Hint**: The following files "may" be modified…
  - userprog/addrspace.*
  - threads/kernel.*

3. Report
- Cover page, including team members, Team member contribution.
- Explain your implementation as requested in Part II-2.
- Explain how NachOS creates a thread(process), load it into memory and place it into scheduling queue as requested in Part II-1. Your explanation on the functions along the code path should **at least** cover answer for the questions below:
  - How Nachos allocates the memory space for new thread(process)?
  - How Nachos initializes the memory content of a thread(process), including loading the user binary code in the memory?
  - How Nachos creates and manages the page table?
  - How Nachos translates address?
  - How Nachos initializes the machine status (registers, etc) before running a thread(process)
  - Which **object** in Nachos acts the role of **process control block**
  - When and how does a thread get added into the ReadyToRun queue of Nachos CPU scheduler?

III. Instruction

1. Copy your code for MP1 to a new folder
   - cp -r NachOS-4.0_MP1 NachOS-4.0_MP2
2. Copy test file
   - cp /home/os2021/share/consoleIO_test* NachOS-4.0_MP2/code/test/
3. Compile/Rebuild NachOS
   - cd NachOS-4.0_MP2/code/build.linux
   - make clean
   - make
4. Test your program
   - cd NachOS-4.0_MP2/code/test
   - ../build.linux/nachos -e consoleIO_test1 -e consoleIO_test2

IV. Grading

1. Implementation correctness – 60%
   - Execute "../build.linux/nachos -e consoleIO_test1 -e consoleIO_test2" correctly
   - There will be hidden testcases to test the requirements. Make sure your code meets all the requirements.
2. Report – 20%
   - Upload it to iLMS with the Filename: **MP2_report_[GroupNumber].pdf**.
3. Demo– 20%
   - Answer questions during demo.
   - Demo will take place on our server, so you are responsible to make sure your code works on our server.

***Refer to syllabus for late submission penalty.**