Chris Colella

Operating Systems

Prof Yikang

11/8/2022

Q1.

1. The first thing that I noticed was the two variables at the top of the program: flag and turn. Just skimming through I can see both of these variables being used frequently. This algorithm appears to be controlling which flag has access to the critical section. It repeatedly tests the flag looking for a certain value. And when that value is met it can fall out of spin into the critical section.
2. I ran the following commands:

Text

Description automatically generatedText

Description automatically generated./x86.py -p peterson.s -t 1 -a bx=0 -i 2 ./x86.py -p peterson.s -t 1 -a bx=1 -i 5

Write answer here

1. D
2. Test-and-test-and-set tests is testing mutex to check if the lock is free so it can move into the critical section. This program does two tests, one initially to test mutex for a lock, and then one after the atomic swap happens to also check for a lock. Compared to test-and-set.s, mutex is only checked once. To me, this seems to be just a program checking for a mutex lock and that’s it.

I think the possible savings this program offers is skipping an entire section of code if the mutex is unlocked initially. So, if it actually is unlocked, it doesn’t need to be checked, therefore saving time/resources, and moving into the critical section.

Text

Description automatically generatedText

Description automatically generated

Q2.

#include <stdio.h>

#include <stdlib.h>

// Written by Chris Colella

// 11/8/2022

// Lab 10

int main(int argc, char\* argv[]) {

// Page size 4 kb

//int pageSize = 4000;

int pageSize = 4096;

int pageNumber = 0;

int offset = 0;

int virtualAddress = 0;

// Need to use conversion, can’t do argv[1] = address

// incompatible conversion assigning to 'char \*' from 'int'

// Need to use atoi to convert int to a string

virtualAddress = atoi(argv[1]);

pageNumber = virtualAddress / pageSize;

offset = virtualAddress % pageSize;

printf("The address %s contains: \n\n", argv[1]);

printf("Page number: %d\n\n", pageNumber);

printf("Offset: %d\n\n", offset);

}

Initially when I wrote this, my pageSize variable was 4000. After running a few tests to try and get the same answer in the homework, it was always off just by a little bit. So I did some research and found I was supposed to be using 4096 to represent a 4kb page size.