WE17.1

Video explanation of solution is provided below the problem.

Operating Systems

4 points possible (ungraded)

BetaSoft, Inc, the leading provider of Beta OS software, sells an operating system for the Beta similar to that described in lecture. It uses a simple round-robin scheduler, and has no virtual memory -- all processes share a single address space with the kernel. The OS timeshares the Beta CPU among N processes using a simple, familiar scheduler shown below.

```
struct MState { int Regs[31]; } UserMState;

struct PCB {               // Process Descriptor Block
struct MState State;       // Saved process state
...;                       // Possible other stuff
} ProcTbl[N];              // One PCB per process

int Cur = 0;

Scheduler() {
  ProcTbl[Cur].State = UserMState;
  Cur = (Cur+1) % N;
  UserMState = ProcTbl[Cur].State;
}
```

Several of Betasoft's customers use the Beta for long, compute-bound applications, and have asked for a tool to help them find where their programs are spending most of their time. To accommodate these requests, BetaSoft has implemented a supervisor call, SamplePC, which allows a diagnostic program running in one process to sample the values in the PC of another. Betasoft proposes to write such a program, called a profiler, that takes many samples of PC values from a running program and produces a revealing histogram.

The SamplePC SVC takes a process number p in R0, and returns in R1 the value currently in the program counter of process p. The C portion of the SVC handler is given below:

```
SamplePC_h() {
 int p = UserMState.Regs[0];
 int pc = ProcTbl[p].State.Regs[XP];
 ??? = pc;          // incomplete code!
}
```

1. Give the missing code fragment shown above as ???. Do not include any white space in your response.

   BetaSoft writes a simple profiler using the above SVC and uses it to measure a compute-bound process consisting of a single 10000-instruction loop. Noticing a surprisingly large number of repeated values in the sampled PC data, they cleverly deduce that their profiler is making many SamplePC calls during each time quanta for which the profiling process is scheduled, returning redundant samples from the process being measured.

2. Does adding a call to Scheduler() to the **SamplePC_h** code eliminate the observed problem?

   - ◯ Yes
   - ◯ No

   BetaSoft ignores your solution (keeping the original **SamplePC_h** code), arguing that they'll just collect enough samples that the redundant values won't affect the histogram significantly. They produce a working profiler program that takes many samples of another process's PC and produces a histogram showing code "hot spots". Although the profiler proves useful on compute-intensive application code, BetaSoft tries running it on a simple echo loop running in a process:

```
                    | echo loop, as a test for profiler tool:
```

Calculator

```
      .=0x100         | Test program starts at hex 100
loop: GetKey()        | SVC: read char into R0
      WrCh()          | SVC: type char from R0
      BR(loop)        | ... and keep doing it!
```

3. When run on the above process, what does the profiler report as the most common value of the PC? Answer "NONE" if you can't tell.

   Often-reported PC value, or "NONE": 0x

   The final BetaSoft profiler program itself is mostly a big loop, consisting of a single SamplePC SVC instruction located at **0×1000**, plus lots of compute-intensive additional code to appropriately format the collected data and write it into a file. Out of curiosity, BetaSoft engineers run the profiler in process 0, and ask it to generate a histogram of sampled PC values for process 0 itself.

4. Which of the following best summarizes their findings?

   - ◯ All of the sampled PC values point to kernel OS code.
   - ◯ The sampled PC is always **0×1004**.
   - ◯ The SamplePC call never returns.
   - ◯ None of the above.

Submit

Operating Systems

(Caption will be displayed when you start playing the video.)

0:00 / 0:00 1.0x

Video
Download video file

Transcripts
Download SubRip (.srt) file
Download Text (.txt) file

Discussion

Hide Discus

Topic: 17. Virtualizing the Processor / WE17.1

Calculator

Add a Post

Show all posts by recent activity

? Can't understand how does SamplePC_h routine work... 4
According to $Samp \leq PC_h$ source code it basally copies the XP register value to $R1$ in $UserMState$ within kernel stack, howe...

[STAFF] Minor type @1:34 in video 2

‹ Previous

Next Up: Lab 10: Instruction Emulation ›
1 activity

edX

edX

About
Affiliates
edX for Business
Open edX
Careers
News

Legal

Terms of Service & Honor Code
Privacy Policy
Accessibility Policy
Trademark Policy
Sitemap

Connect

Blog
Contact Us
Help Center
Media Kit
Donate

GET IT ON Google play
Download on the App Store

Calculator

Calculator