What happens when a page fault occurs in stack?

I need to implement virtual memory for one school project and I'm confused about something. If page fault occurs while working with stack (simple push for example), how and where am I going to save return address if I cannot push?

** Unless the question is really simple, there are not enough details to answer the question. The literal interpretation of the question is that that the page fault exception handler maps the page and restarts the instruction. Simple. If it's not that simple, the question needs more detail.*

** That's actually a very good question. I edited to make it more clear.*

2 Answers

Here's what our development team did when faced with this same problem ...

We designed our in-house OS such that the page fault handler was not allowed to generate a page fault. Thus, the stack to which the page fault exception stack frame was pushed absolutely had to be guaranteed to be present in memory. Furthermore, any routine used by the page fault handler also by necessity had to be present in memory, as otherwise that could cause a page fault thereby violating the design.

** So what you want to say is that I need to load next page for stack into memory (rise page fault) before the stack becomes completely filled? Say two bytes for example as my address is two bytes long.*

** First you need to be sure which stack is being completely filled; is it a user space (US) stack or kernel space (KS) stack? (Assuming x86) If a page fault occurs in US, as ssg points out, the processor should switch to the KS stack specified in the TSS esp0 field. That KS stack had better be present in memory, or you will have problems. The KS stack is IIRC where the exception stack frame is pushed. If you are talking about running out of US stack space, the page fault handler could conceivably grow the stack so that there is more (refer to guard pages).*

** Cool, thanks. One more question to make it clear. Lets say I don't have to worry about all those OS details, and all I have to do is build emulator for some hardware (or real hardware, nvm). Is it ok if I just raised page faults whenever referenced page is not in memory, and leave all those details for OS?*

That's a very good question. Page faults cause an interrupt and how does that interrupt know where to return if it cannot preserve the return address?

On x86 architecture, TSS (task state segment) contains different stack pointers for different privilege levels. So if your user mode process runs out of stack, CPU privilege level is lowered and an exception is raised. That means as soon as you switch to the new privilege level, OS can start to use the new stack which resides in kernel memory and isn't subject to stack limits of the user process.