TP2: Scheduling y cambio de contexto II

context\_switch

. si empiezan a hacer pop de la nada lo unico q van a sacar es el puntero al trap frame

. los parámetros se pasan por el stack -> el struct a trapframe está debajo de la dirección de retorno

Queremos un stack que apunte a una región que tenga apilada: eip, cs, eflags, esp.

1. muevo el esp a la dirección del struct trapframe
2. hago popal para restaurar todos los registros de pushregs a la vez
3. popeamos es y ds
4. muevo el esp a eip
5. llamo a iret

-> info registers -> para ver el contenido de todos los registros

-> print [nombre\_de\_registro] -> esto en consola me da el valor del registro:

- en una consola hago make qemu-nox-gdb

- en otra consola hago:

- make gdb

- b i

- b i386\_init

- c

- info registers -> valores de registros en este momento

- print $eax -> valor de eax

- p $eax -> valor de eax

-> p/x $eax -> valor de eax en hexa

-> pop %eax -> popea del stack y carga en eax

Para debuggear context\_switch:

. en una consola:

> make qemu-nox-gdb

. en otra consola:

> make gdb

> b env\_run

> c

> tui enable

> focus cmd

> n

> n

> disas (siguientes instrucciones en código máquina)

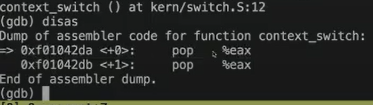
> lis (siguientes instrucciones en C)

> si (avanza 1 instrucción a la vez)

> disas (ejecutó el call y quedé adentro de la función unlock\_kernel)

> finish (salir de la función)

… sigo haciendo si y finish hasta llegar a switch.s

> 

> p/x $esp (dónde está el stack)

> x/x $esp (muestre lo que hay en la dirección de memoria donde apunta esp)

> x/10x $esp (dump de los contenidos de esa dirección)

… hacemos si para ir instrucción por instrucción y printeamos los registros para ver cómo va cambiando el stack.

→ después de problemas técnicos:

. en una consola:

> make qemu-nox-gdb

. en otra consola:

> make gdb

> b context\_switch

> c

> tui enable

> focus cmd

> p/x $esp

> x/10x $esp (en el segundo valor estan los valores del tf)

> x/10x 0x…

> p/x \*(struct Trapframe\*)(0x…)

