Multitasking

- 4 users on 1 PC.

* A really simple serial controller.
We have one of these, one for each terminal.

**Terminals**

FB20

FB21

FB22

FB23

**Terminals**

TERM 1

TERM 2

TERM 3

TERM 4

**Memory Management**

Suppose we're running the same program in all 4 terminals.

→ Compile once, run @ different addresses

**Solution 1:** use a "fix-up list"

**Solution 2:** use a HW approach w/ "segment registers"
Four Processes Calling the Same Function (Asynchronous Function Returns)

Cooperative Multitasking

- To do this, we take advantage of the multitasking stack format.

Context Switching

End P1

- Stack pointer is moving along.
- Copy registers and move SP down.
- Save stack pointer to "saved sp2".

Start P2

- Move to "saved sp2".
- Pop the registers to correct location.
- Move SP to pre-register location.

Note: Using this, you need to sort of take pre registers when starting a process for the first time.

Cooperative Multitasking - Old school method to multitask. Forced programmer to call into the OS to see if other tasks need to do something. It was not compartmentalized, if one program broke the whole OS came down.

Process Control Block - Saves all process information for the PC. In the case of context switching it's "saved sp".

Pre-emptive Multitasking

- What you do when a process may be long running.
- A physical line going into the PC.
- Keyboard controller has an interrupt on it, allows keyboard to work even w/ a long running process. Program has no knowledge it ran.
Pre-emptive Multitasking

Keyboard Buffer: a b c

Keyboard input over time

*True pre-emptive multitasking is done via a timer interrupt.