3) program devices

2) concurrency

asynchronous

Hul wants attention now

$15

6.5081. Interrupts

To save its work

process interrupt

resume its work

traps

PressZ unpl

The hardware mechanism
Where do interrupts come from?
Figure 3: FU540-C000 Interrupt Architecture Block Diagram.
Programming device
memory mapped I/O
read/write control register of the device
Case study: $1s$

$\$: device puts $1$ into UART

UART generates interrupts when the character has been sent

$1s$: keyboard cannot receive line generate $1$ interrupt
RISC-V support for interrupts

SIE: one bit for E, S, T

SSRATUs: bit enable/disable

SIP: $interrupt pending

SCAUE:

STVEC:
Interrupt (hw)

If SIE bit set:
- Clear SIE bit
- Set SEPC ← PC
- Save current mode
  Mode ← Supervisor
- Set PC ← STvec
  (→ User+trapC)
Interrupts and concurrency

1) device + CPU run in parallel
   ⇒ producer/consumer parallelism

2) interrupt stops the current program
   interrupt enable/disable

3) top of driver + bottom driver may run in parallel using locks & semaphores
Producer/consumer

\[ p_u \cdot c \]

\[
\begin{array}{cccccc}
| & | & | & | & | & |
\end{array}
\]

\[ \text{start} \]

\[ 0 \uparrow \uparrow 6 \]

\[ \text{vartunk(} \]
Interrupt evolution

Interrupt used to be fast

Simpl

Now slow

Device is more complicated

Gbit ethernet

1.5 Mpps => 1 interrupt per microsec
Polling

CPU spins until device has data
Waste CPU cycles if device is slow
but if device is fast, saves entry/exit cost
Dynamically switch between polling/interrupts