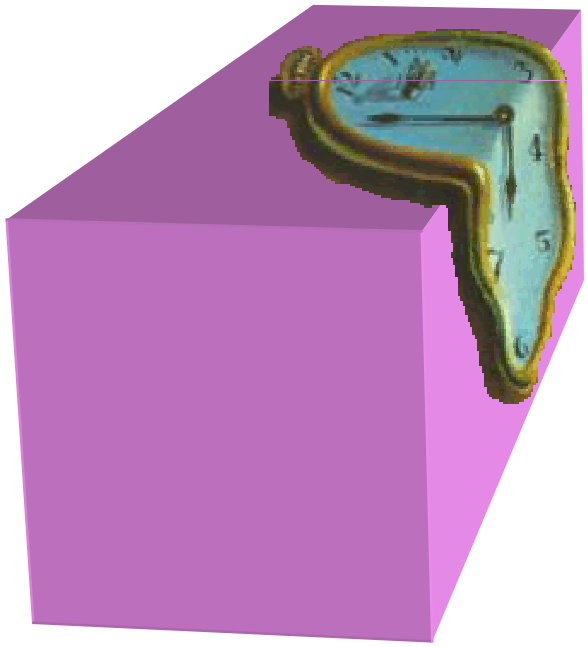
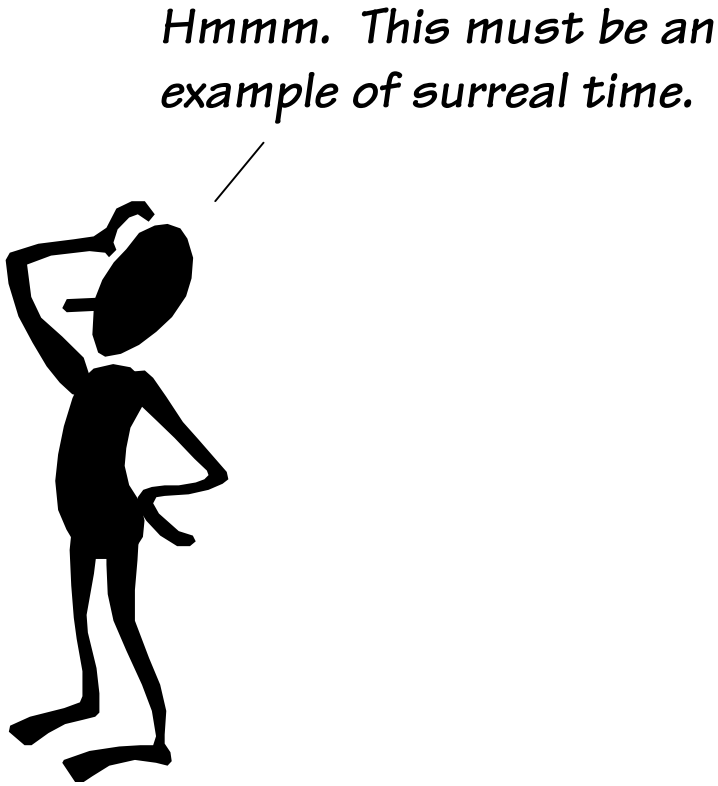
Interrupts & Real Time



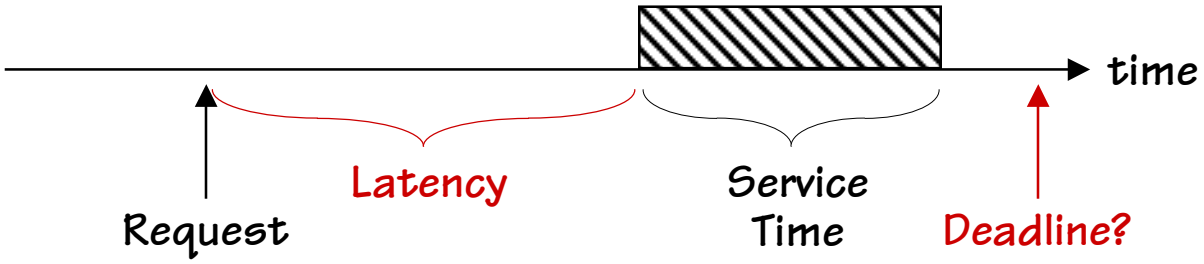


Handouts: Lecture Slides

Interrupt Latency

One way to measure the real-time performance of a system is
INTERRUPT LATENCY:

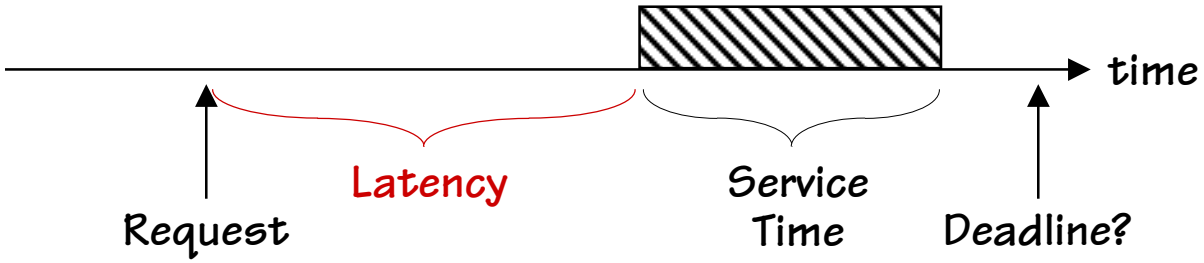
HOW MUCH TIME can elapse between an interrupt request and the START of its handler?



OFTEN bad things happen when service is delayed beyond some deadline - "real time" considerations:

- Missed characters
- System crashes
- Nuclear meltdowns

Sources of Interrupt Latency



What causes interrupt latency:

- State save, context switch.
- Periods of uninterruptability:
  - Long, uninterruptible instructions -- eg block moves, multi-level indirection.
  - Explicitly disabled periods (eg for atomicity, during service of other interrupts).

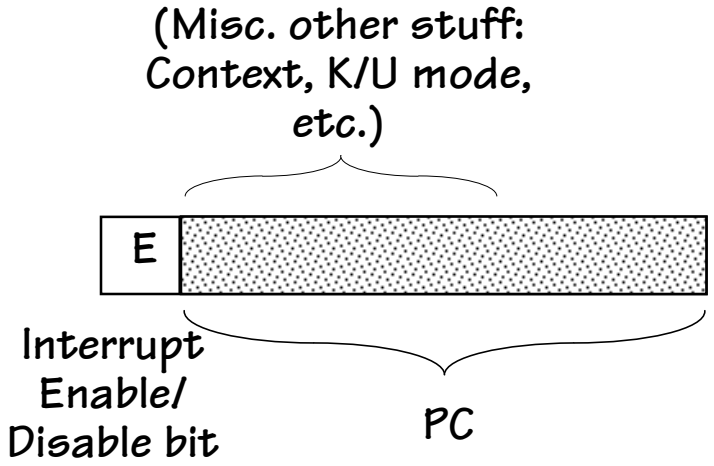
GOAL: BOUND (and minimize) interrupt latency!

- Optimize interrupt sequence context switch
- Make unbounded-time instructions INTERRUPTABLE (state in registers, etc).
- Avoid/minimize disable time
- Allow handlers to be interrupted, in certain cases (while still avoiding reentrant handlers!).

Interrupt Disable/Enable

INTERRUPT DISABLE BIT (part of processor status)... in PC:

Often in separate Processor Status Word ...



E=1: DISABLED ("Kernel" mode)
E=0: ENABLED ("User" mode)

TYPICAL OPERATION:

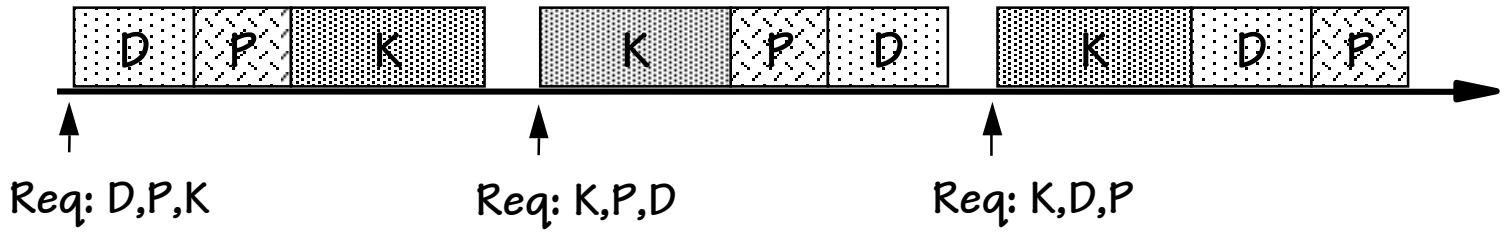
- ONLY take interrupts if E=0; else defer.
- SAVE OLD E on interrupt, install new E from interrupt vector (along with PC, etc). New E=1 (to disable interrupts during handler).
- Run handler, with interrupts disabled.
- On JMP (at return from handler), saved state restored to processor, resuming interrupted program (with E=0 again).

Scheduling of Multiple Devices

"TOY" System scenario:

Actual Latency	DEVICE	Service Time
500 + 400 = 900	Keyboard	800
800 + 400 = 1200	Disk	500
800+ 500 = 1300	Printer	400

What is the WORST CASE latency seen by each device?



Service of EACH device might be delayed by service of others ... unless we *prioritize* them somehow.

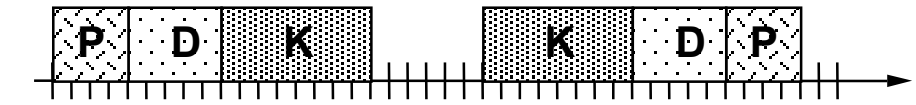
Weak (non-preemptive) Priorities

ISSUE: Processor becomes interruptible (at fetch of next instruction), several interrupt requests are pending. Which is served first?

WEAK PRIORITY ORDERING: Check in prescribed sequence, eg:
DISK > PRINTER > KEYBOARD.

LATENCIES with WEAK PRIORITIES:
Service of each device might be delayed by:

- Service of 1 other (arbitrary) device, whose interrupt request was just honored; PLUS
- Service of ALL higher-priority devices.



Req: P,D,K

Req: K,P,D

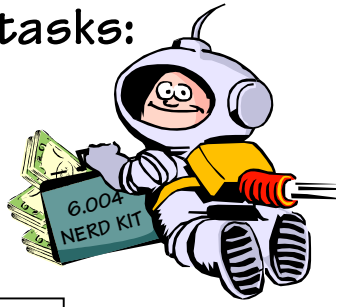
Couldn't this be a lot worse?
Suppose, D,K,P,D,P,D?
It seems we also need to know the frequency of interrupts

Actual Latency	DEVICE	Service Time
500 + 400 = 900 (INT during Keyboard SVC)	Keyboard	800
800	Disk	500
(INT during 1300 Keyboard SVC W/pending Disk)	Printer	400

Example: Ben visits ISS

International Space Station's on-board computer performs 3 tasks:

- guiding incoming supply ships to a safe docking
- monitoring gyros to keep solar panels properly oriented
- controlling air pressure in the crew cabin



	Task	Period	Service time	Deadline	
16.6 %	Supply ship guidance	30ms	5ms	25ms	[CP],G = 10 + 10 + (5) = 25
25%	Gyroscopes	40	10	20	[CP] = 10 + (10) = 20
10%	Cabin pressure	100	? 10	100	SSG,G = 5 + 10 + (10) = 25

Assuming a weak priority system:

1. What is the maximum service time for "cabin pressure" that still allows all constraints to be met? < 10 mS
2. Give a weak priority ordering that meets the constraints G > SSG > CP
3. What fraction of the time will the processor spend idle? 48.33%
4. What is the worst-case delay for each type of interrupt until completion of the corresponding service routine? 25 mS

Strong (preemptive) Priorities

Often, tight real-time demands require more control over scheduling of interrupt handlers.

EXAMPLE: 300 uSec maximum delay on disk service, to avoid missing next sector...

Priority	Latency w/preemption	Actual Latency	DEVICE	Serv. Time	Max. Delay
3	D,P 900	900	Keybrd	800	
1	~0	800	Disk	500	300
2	[D] 500	1300	Printer	400	

CAN'T SATISFY the disk requirement in this system using weak priorities!

need PREEMPTION: Disable LOWER PRIORITY interrupts without interfering with HIGHER ones!

Strong Priority Implementation

SCHEME:

- Expand E bit in PC to be a PRIORITY integer PRI
  (eg, 3 bits for 8 levels)
- ASSIGN a priority to each device.
- Prior to each instruction execution:
  - Find priority P_i of highest requesting device, say D_i
  - Take interrupt if and only if P_i > PRI, set PRI = P_i.

PC:

PRI	Program Counter

Strong priorities:

- KEY: Priority in Processor state
- Allows interruption of (certain) handlers
- Prevents preemption, but not reentrance
- BENEFIT: Latency seen at high priorities UNAFFECTED by service times at low priorities.

Example: Ben visits ISS (cont'd)

Our Russian collaborators don't like the sound of a "weak" priority interrupt system and lobby heavily to use a "strong" priority interrupt system instead.

	Task	*Period*	*Service time*	*Deadline*	
16.6%	Supply ship guidance	30ms	5ms	25ms	[G] 10 + 5
25%	Gyroscopes	40	10	20	10
50%	Cabin pressure	100	? 50	100	100

Assuming a strong priority system:

1. What is the maximum service time for "cabin pressure" that still allows all constraints to be met? 100 – (3*10) – (4*5) = 50
2. Give a strong priority ordering that meets the constraints G > SSG > C
3. What fraction of the time will the processor spend idle? 8.33%
4. What is the worst-case delay for each type of interrupt until completion of the corresponding service routine?

Example: Ben visits ISS (cont'd)

One of the astronauts (they won't say which one) has tripped over the control cables for the supply ship guidance system which increases the service time for that task from 5ms to 15ms since all commands now have to be transmitted 3 times.

	Task	*Period*	*Service time*	*Deadline*	
50%	Supply ship guidance	30ms	15ms	25ms	10 + 15 = 25
25%	Gyroscopes	40	10	20	10
15%	Cabin pressure	100	? 15	100	100

Assuming a strong priority system:

1. Explain why a weak priority system can no longer be used. SSG blocks G
2. What is the maximum service time for "cabin pressure" that still allows all constraints to be met? 100 – (3*15+10) – (3*10) = 15
3. Give a strong priority ordering that meets the constraints
4. What fraction of the time will the processor spend idle? 10%
5. What is the worst-case delay for each type of interrupt until completion of the corresponding service routine?

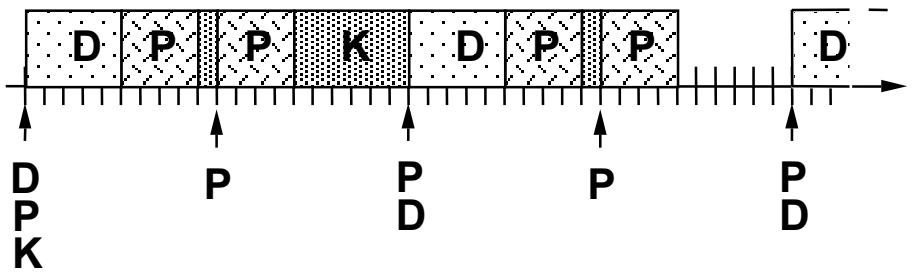
Recurring Interrupts

Consider interrupts which recur at bounded rates:

Actual Latency	DEVICE	P	Serv. Time	Max. Delay	Max. Freq
900	Keybrd	3	800		100/s
0	Disk	5	500	300	500/s
500	Printer	4	400		1000/s

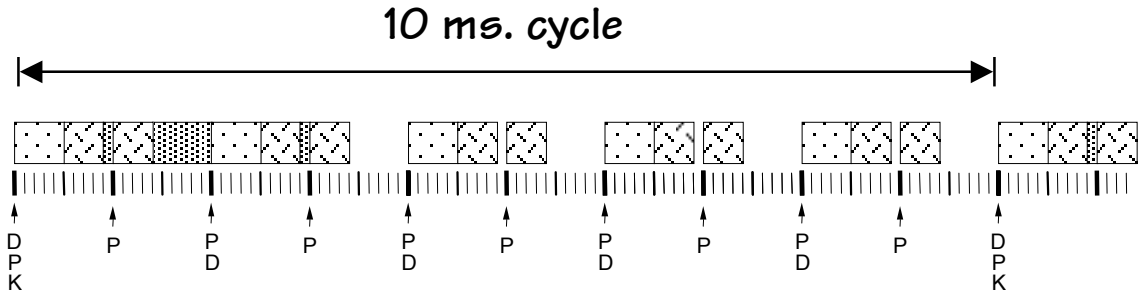
Note that interrupt LATENCIES don't tell the whole story—consider COMPLETION TIMES, eg for Keyboard in example to the right.

Keyboard service not complete until 3ms after request. Often *deadlines* used rather than max. delays.



Interrupt Load

How much CPU time is consumed by interrupt service?



Actual Latency	DEVICE	P	Serv. Time	Max. Delay	Max. Freq	% Load
900	Keybrd	3	800		100/s	800/10000 = 8%
0	Disk	5	500	300	500/s	500/2000 = 25%
500	Printer	4	400		1000/	400/1000 = 40%

Priority Assignment

BLACK ART. In our example: unique solution (verify by exhaustion!). Great body of literature; many NEGATIVE results.

TWO positive results:

RATE MONOTONIC (Liu & Layland):

CASE: Periodic, self-deadlines, static priority

RULE: Prioritize by FREQUENCY.

- NOT usually applicable (Auto Impact sensor gets LOWEST priority!)
- Extensions to DYNAMIC priorities - violate stack discipline.

EARLIEST DEADLINE (Knuth, Mok, others):

CASE: Deadline specified at each request.

RULE: Work on pending task with EARLIEST deadline.

... OPTIMAL for 1 processor; can't be extended to optimal N-processor scheme.

Obeys stack discipline!

BOTTOM LINE: Important problem, poorly addressed by modern architectures/software engineering.

"Real World" Interrupts

Typical contemporary priority interrupt system:

- Fixed number (eg, 8) of strong priority levels
- At each priority level: weak ordering among several devices.

Worst case interrupt latency seen by a device: Sum of

- Service time of all devices at higher (preemptive) priority levels; PLUS
- Service times of earlier devices in (weak) priority chain at same strong priority level; PLUS
- Greatest service time among remaining devices at same strong priority level.

Next Time:

some parallel processing issues...