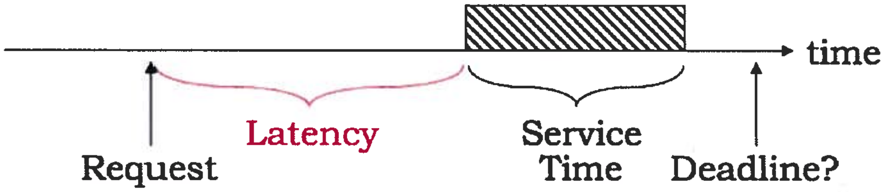
Computation Structures

Interrupts and Real Time Worksheet

Latency (L) = elapsed time before handler code starts to execute
Service time (S) = time required to execute handler code
Deadline (D) = maximum time after request by when handler execution must be *complete*
Maximum allowable latency (Lmax) = largest L such that Lmax + S = D.



Handler priority: when choosing which handler to run, choose the handler with the highest priority. Priorities are chosen to ensure that the deadlines for all handlers will be met.

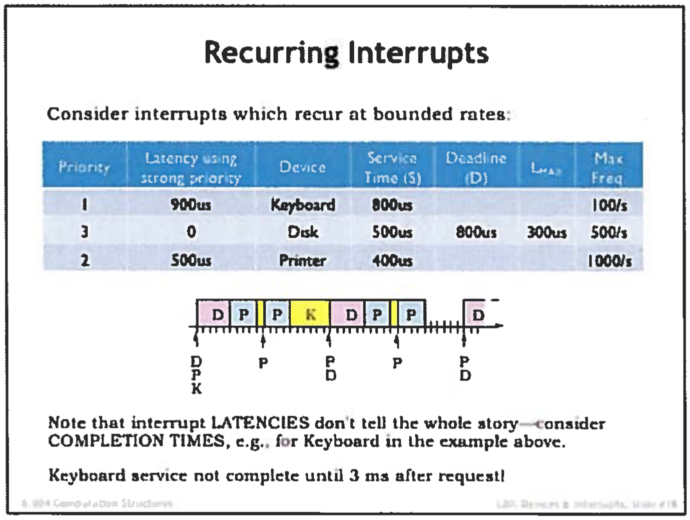
Recurring requests: often requests will occur at some fixed interval ≥ deadline.

Earliest deadline is a strategy for assigning priorities that is guaranteed to meet the deadlines if any priority assignment can meet the deadlines:

1. Sort the requests by their deadlines
2. Assign the highest priority to the earliest deadline, second priority to the next deadline, and so on.

Weak (non-preemptive) priorities: after current handler completes, look through the pending requests and execute the handler with the highest priority. Latencies with weak priorities: service of each device might be delayed by (1) service of one other (arbitrary) device whose request was just honored, and (2) service of all higher-priority tasks.

Strong (preemptive) priorities: always run the pending handler with the highest priority, possibly interrupting the execution of a lower-priority handler to do so. Latencies with weak priorities: service of each device might be delayed by service of all (possibly recurring) higher-priority tasks.



Recurring Interrupts

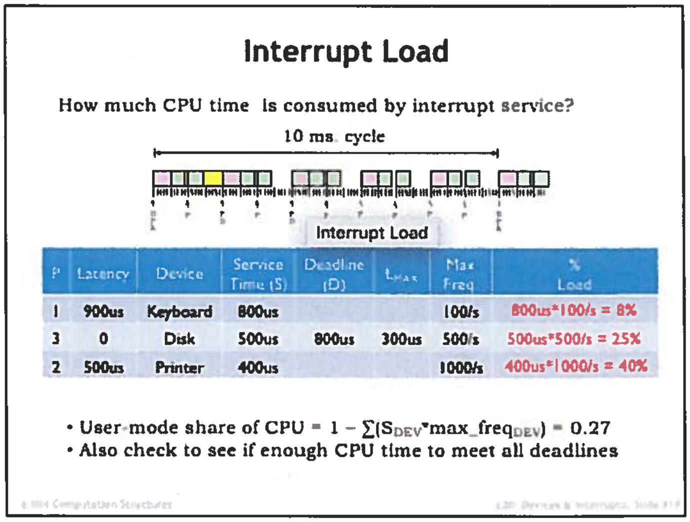
Consider interrupts which recur at bounded rates:

Priority	Latency using strong priority	Device	Service Time (S)	Deadline (D)	L_{MAX}	Max Freq
1	900us	Keyboard	800us			100/s
3	0	Disk	500us	800us	300us	500/s
2	500us	Printer	400us			1000/s

Note that interrupt LATENCIES don't tell the whole story—consider COMPLETION TIMES, e.g., for Keyboard in the example above.

Keyboard service not complete until 3 ms after request!

6.004 Computation Structures   L20: Devices & Interrupts, Slide #18



Interrupt Load

How much CPU time is consumed by interrupt service?

P	Latency	Device	Service Time (S)	Deadline (D)	L_{MAX}	Max Freq	% Load
1	900us	Keyboard	800us			100/s	800us*100/s = 8%
3	0	Disk	500us	800us	300us	500/s	500us*500/s = 25%
2	500us	Printer	400us			1000/s	400us*1000/s = 40%

- User-mode share of CPU = 1 – $\sum(S_{DEV} * max\_freq_{DEV})$ = 0.27
- Also check to see if enough CPU time to meet all deadlines

6.004 Computation Structures   L20: Devices & Interrupts, Slide #19

Problem 1.

Ben Bitdiddle has designed a wrist device called the BenBit to measure how long you've been tooling away without getting up and moving around. The BenBit runs a real-time operating system supporting three tasks whose handlers are executed in response to periodic requests. Each task has a period (time between requests), a service time (time it takes to run its handler), and a deadline (maximum time allowed to elapse between request and completion of the handler).

Task	*Period (ms)*	*Service time (ms)*	*Deadline (ms)*
Check Accelerometer (CA)	80	20	40
Update Display (UD)	200	?	200
Determine heart rate (DHR)	60	10	50

Ben is trying to figure out whether to use a weak or strong priority system to manage task execution. For each of the questions below, please fill the answers for both types of priority system. For both the weak and strong priority system **assume the task priority is CA > DHR > UD**, i.e., CA has the highest priority and UD the lowest. If a calculation requires the service time for UD, use your answer for part (A).

	Using Weak Priorities	**Using Strong Priorities**
(A) What is the maximum service time for UD handler that still allows all deadlines to be met (in ms)?	20 ms otherwise CA will miss deadline	100ms $= 200 - 3 * 20 - 4 * 10$
(B). What fraction of the time will the processor spend idle? (%)	48.33% $= 1 - (\frac{20}{80}) - (\frac{20}{200}) - (\frac{10}{60})$	8.33% $= 1 - (\frac{20}{80}) - (\frac{100}{200}) - (\frac{10}{60})$
(C) What is the worst-case completion time for the CA task (in ms)?	40 ms $= UD_{ST} + CA_{ST}$	20 ms $= CA_{ST}$
(D) What is the worst-case completion time for the UD task (in ms)?	50 ms $= DHR_{ST} + CA_{ST} + UD_{ST}$	200 ms $= 3 \cdot CA_{ST} + 4 \cdot DHR_{ST} + UD_{ST}$
(E) What is the worst-case completion time for the DHR task (in ms)?	50 ms $= UD_{ST} + CA_{ST} + DHR_{ST}$	30 ms $= CA_{ST} + DHR_{ST}$

Problem 2.

A particular real-time system has three interrupt handlers. The following table shows the maximum rate at which each interrupt occurs (rate), the time taken to execute each handler (service time), and the maximum allowable interval between the interrupt and completion of the handler (deadline). In your analysis, assume that A, B, and C interrupts can arrive at any time.

% CPU

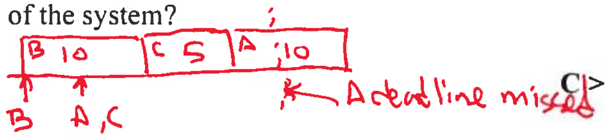
% CPU	*Task*	*Rate*	*Service time*	*Deadline*
10/20 = 50	A	1/20ms	10ms	20ms
10/80 = 12.5	B	1/80ms	10ms	80ms
5/25 = 20	C	1/25ms	5ms	25ms

(A) What is the percentage idle time for this system?

100% - (50+12.5+20)%

Percentage Idle Time (%): 17.5%

(B) Assuming a **weak** priority system. Can the priority ordering C > A > B satisfy **all** the constraints of the system?



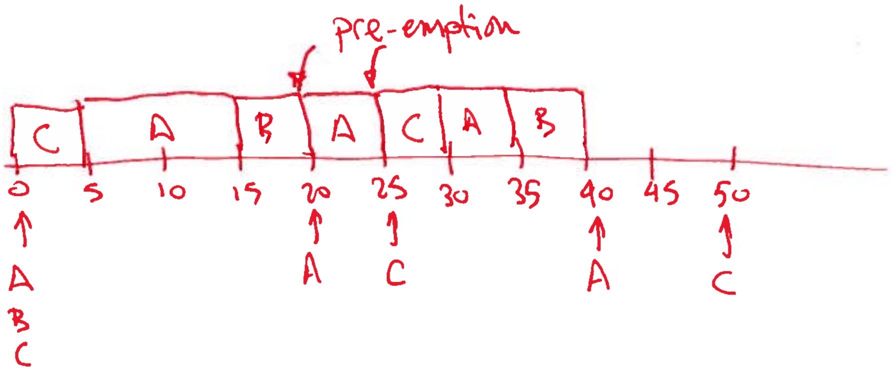
C > A > B (Yes/No): NO

(C) Assuming a **strong** priority system with priority **C > A > B**, what is the worst-case delay between the task's interrupt and **completion** of the task's interrupt handler?

Worst-case delay for task A (ms): 15

Worst-case delay for task B (ms): 40

Worst-case delay for task C (ms): 5



Problem 3.

A particular real-time system has three interrupt handlers. The following table shows the maximum rate at which each interrupt occurs (rate), the time taken to execute each handler (service time), and the maximum allowable interval between the interrupt and completion of the handler (deadline). In your analysis, assume that A, B, and C interrupts can arrive at any time.

Task	*Rate*	*Service time*	*Deadline*
A	1/40ms	5ms	30ms
B	1/100ms	?ms	100ms
C	1/50ms	10ms	25ms

Please answer the following two questions assuming the system has a **weak** priority system.

(A) (1 point) What is the maximum service time for task B that still allows all constraints to be met?

Task C has max latency of 15 ms → limits service time of B.

Maximum service time for task B (ms): { 15 if C>A>B in port(B)
10 if A>C>B inport(B)

(B) (1 point) Assuming a suitable service time for B, give a weak priority order for the tasks that meets the constraints. List the task with the highest priority first, the task with the lowest priority last.

earliest deadline first!

Weak priority order for the tasks: C > A > B

Please answer the following two questions assuming the system has a **strong** priority system where task C has the highest priority and task B has the lowest priority.

(C) (2 points) What is the maximum service time for task B that still allows all constraints to be met?

in 100 ms worst case:
3 task A requests (15 ms total)
2 task C requests (20 ms total)
35 ms service time ⟹ 100 - 35 = 65 ms available for B

Maximum service time for task B (ms): 65

(D) (3 points) Assume B's sevice time is 10ms. For each task, what is the worst-case delay between the task's interrupt and **completion** of the task's interrupt handler?

priority C > A > B

wait for C + A service time = 10 + 5

Worst-case delay for task A (ms): 15

Worst-case delay for task B (ms): 25

wait for C, A + B service time = (10 + 5) + 10 = 25

Worst-case delay for task C (ms): 10

↳ starts immediately = C service time

Problem 4.

A computer system has three devices whose characteristics are summarized in the following table:

Device	Service time	Interrupt Frequency	Deadline
D1	400us	1/(800us)	800us
D2	250us	1/(1000us)	300us
D3	100us	1/(800us)	400us

Service time indicates how long it takes to run the interrupt handler for each device. The maximum time allowed to elapse between an interrupt request and the end of the execution of the interrupt handler is indicated by the deadline.

A. If a user-mode program P takes 100 seconds to execute when interrupts are disabled, approximately how long will P take to run when interrupts are enabled?

Approximate time for P to run with interrupts enabled (seconds): 8·100 = 800

D1: 400/800 = 50% of CPU time
D2: 250/1000 = 25%
D3: 100/800 = 12.5%

only 12.5% = 1/8 of cpu time available for running P.

B. Can the requirements given in the table above be met using a **weak** priority ordering among the interrupt requests? If so give priority ordering for D1, D2, D3 or list device(s) whose deadlines cannot be met.

Weak priority ordering or list device(s) with missed deadlines: D2, D3

if D1 is running, D2 and D3 may miss deadlines since D2 max latency is 50us and D3 max latency is 300us.

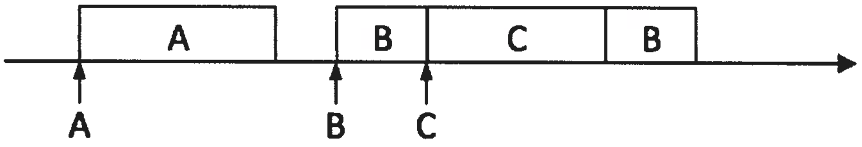
(C) Can the requirements given in the table above be met using a **strong** priority ordering among the interrupt requests? If so give priority ordering for D1, D2, D3 or list device(s) whose deadlines cannot be met.

Strong priority ordering or list device(s) with missed deadlines: D2 > D3 > D1

use earliest deadline strategy!

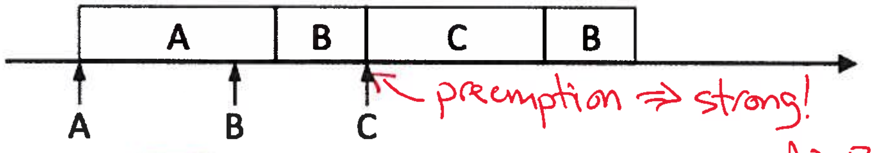
Problem 5.

A real-time operating system with priority interrupts has three interrupt handlers – A, B, C – each running at a different priority level. The handlers are invoked by the A, B, and C interrupts, marked as ↑ in the execution timelines. For example, the following execution timeline shows the A handler running to completion after an A interrupt request, followed by execution of the B handler, which is interrupted by execution of the C handler.



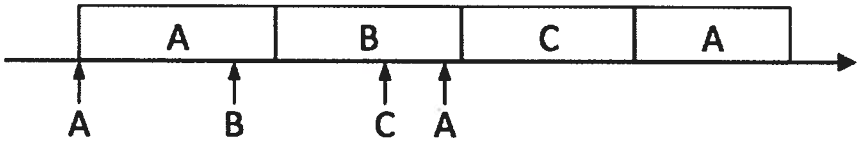
For each of the following execution timelines, please indicate whether the system is using a WEAK or STRONG priority scheme, or CAN'T TELL if the timeline is consistent with either WEAK or STRONG. Also, if WEAK or STRONG, indicate any relative priorities that can be deduced from the timeline (there may be more than one), expressed as inequalities. For example, A > B indicates A has a higher priority than B.

(A)



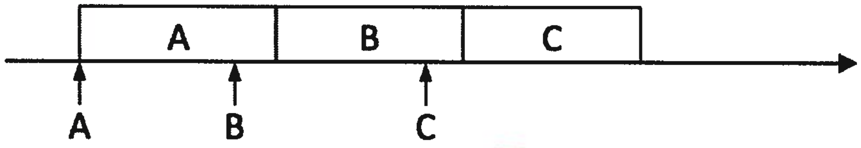
Circle one: [✓ **STRONG**] **… WEAK … CAN'T TELL, Priorities:** A > B, C > B

(B)



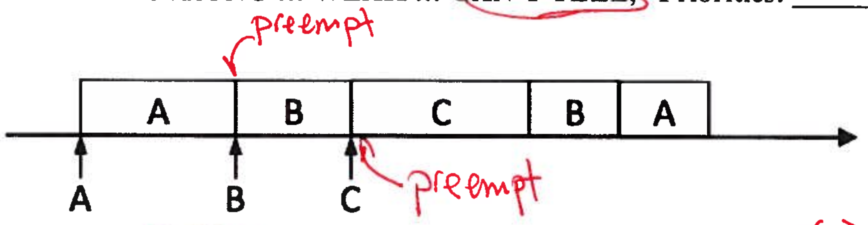
Circle one: STRONG … [✓ **WEAK**] **… CAN'T TELL, Priorities:** C > A

(C)



Circle one: STRONG … WEAK … [✓ **CAN'T TELL**]**, Priorities:** —

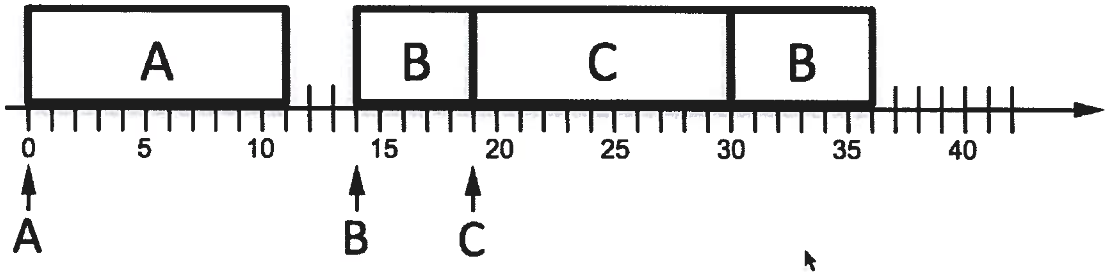
(D)



Circle one: [✓ **STRONG**] **… WEAK … CAN'T TELL, Priorities:** C > B > A

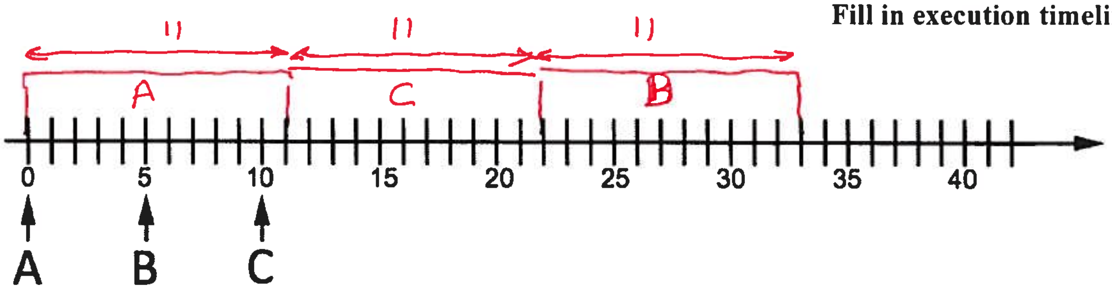
Problem 6.

A real-time operating system with priority interrupt has three interrupt handlers (A, B, C), each of which, when invoked by the appropriate interrupt request (marked as ↑ in the execution timelines), takes 11 time units to execute. For example, the following execution timeline shows the A handler running to completion after an A interrupt request, followed by execution of the B handler, which is itself interrupted by execution of the C handler.



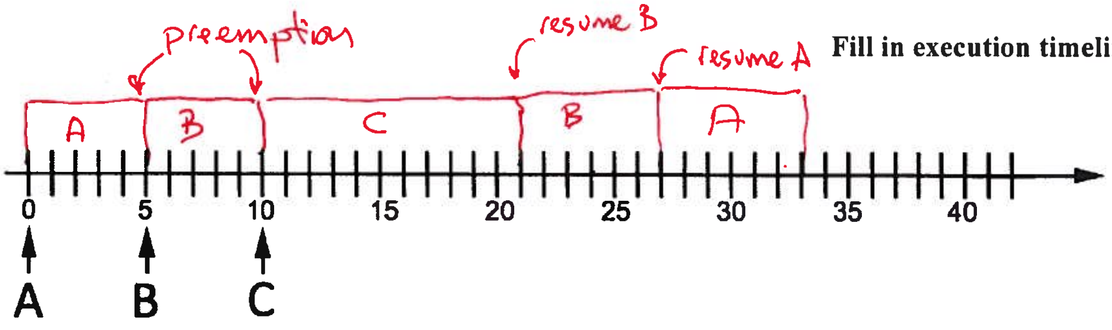
(A) The execution timeline below shows the arrival times of interrupt requests for A, B, and C. Diagram the execution of the A, B, and C handlers assuming a **weak** priority system with the priorities **C > B > A**. Remember to show the complete execution (all 11 time units) for each handler, labeling each block with the handler that is running.

Fill in execution timeline



(B) The execution timeline below shows the arrival times of interrupt requests for A, B, and C. Diagram the execution of the A, B, and C handlers assuming a **strong** priority system with the priorities **C > B > A**. Remember to show the complete execution (all 11 time units) for each handler, labeling each block with the handler that is running.

Fill in execution timeline


preemption
resume B
resume A
A
B
C
B
A
0
5
10
15
20
25
30
35
40
A
B
C