For this problem, assume that we have a computer system that has three devices D1, D2, and D3.

Each of these devices can cause interrupts in our system.

This table summarizes the interrupt characteristics of our three devices.

For each device, we are given its service time which is the amount of time it takes to service an interrupt for that particular device.

We are given an interrupt frequency which tells us how frequently the interrupts for that device arrive.

You can assume that the first interrupt of each device can arrive at any time.

The deadline is the longest amount of time that is allowed between the interrupt request and the completion of the interrupt handler.

Assume we have a program P that takes 100 seconds to execute when interrupts are disabled.

We would like to figure out how long it would take to execute this program when interrupts are enabled.

To answer this question, we need to determine the amount of cpu time that is dedicated to the handling of each of the three devices.

D1 has a service time of 400us and it runs every 800us so it is using 400/800 or 50% of the cpu time.

D2 has a service time of 250us and it runs every 1000us so it is using 250/1000 or 25% of the cpu time.

D3 uses 100/800 or 12.5% of the cpu time.

This means that the user programs have the remaining cpu time available to them.

The remaining cpu time is 12.5% or 1/8 of the cpu time.

If the user program can only run for one eighth of the time, that means that a program that takes 100 seconds without interrupts will take 800 seconds to run with interrupts enabled.

We want to consider whether there is a weak priority ordering that could satisfy all of the constraints for this system? Recall that with a weak priority ordering, there is no preemption, so if an interrupt handler has begun running it runs to completion even if another interrupt of higher priority arrives before its completion.

Upon completion, all interrupts that have arrived, regardless of their order of arrival, are processed in priority

order.

If there is a weak priority ordering that satisfies our system, then we should determine the priority ordering.

If there is no such ordering, then we should identify the devices for which a weak priority ordering cannot guarantee meeting all the constraints.

Returning to our device characteristics table and comparing our deadlines to the device service times, we see that in a weak priority system if the D1 handler which has a service time of 400us happens to be running when a D2 or D3 interrupt arrives, then the D2 or D3 devices could miss their deadlines because the service time of D1 plus their own service time is greater than their deadline.

In other words, if D2 or D3 have to wait up to 400us before beginning to be serviced then their completion time won't be until 650us for D2 which is greater than its deadline of 300us, and 500us for D3 which is greater than its deadline of 400us.

Thus, there is no weak priority system ordering which is guaranteed to satisfy all of our system constraints.

Now, lets reconsider the same question assuming a strong priority ordering.

Recall that with a strong priority ordering, the handler for a device with a higher priority will pre-empt a running handler of a lower priority device.

In other words if the priority of A is greater than B and an A interrupt arrives midway through the handling of a B interrupt, then the B interrupt handler will get interrupted, the A handler will be run, and upon completion of the A handler, the B handler will be resumed.

If there is a strong priority ordering that satisfies our system, then we should specify what it is.

If there is no such ordering, then we should identify the devices for which even a strong priority ordering cannot guarantee meeting all the constraints.

Since we now allow preemption of lower priority device handlers in order to satisfy the requirements of a higher priority handler, we are no longer faced with the issue that devices D2 and D3 can't meet their deadlines if D1 happens to be running first.

In addition, since at the beginning of our problem we determined that there is enough time to service all of our interrupts given their service times and interrupt frequencies, that means that there must exist a strong priority ordering that can satisfy all the constraints of our system.

You can use the scheme that a device with a shorter deadline should have a higher priority than one with a longer deadline to arrive at a valid strong priority ordering.

A valid strong priority ordering is D2 has the highest priority, then D3 and then D1.

Another way of expressing this is D2 > D3 > D1.

Note that for this example, this priority ordering is the only valid ordering that will satisfy all the constraints of our strong priority system.

To convince ourselves of this, let's take a closer look at other priority possibilities and determine what would happen in those situations.

If D1 had a higher priority than either D2 or D3, then the deadlines for D2 and D3 would not be guaranteed to be satisfied.

This means that D1 must have the lowest priority.

Now between D2 and D3, if D3 had a higher priority than D2, then if D3 was being serviced when a D2 interrupt arrived, the D2 interrupt may not complete until 350us which is beyond its deadline.

So this shows us that D2 must have the highest priority, then D3 and finally D1.