Lecture 18: Interrupt analysis

Exception handling – interrupt handling of error conditions such as task scheduling problems
Fault tolerance – ability of system to function in presence of faults
Fault latency – time between onset of (h/w or s/w) fault and error occurrence in the system
Error latency – time between error occurrence and its effect on the rest of the system
Watchdog timer – process that generates interrupt when not reset in a specified period; VxWorks routines for implementing watchdog are wdCreate(), wdDelete(), wdStart(), and wdCancel()
NMI (non-maskable interrupt) – usually highest priority interrupt (eg clock)

Examples: overwriting memory, array out-of-bounds, divide by 0, max run time, starting run time, context switching
O/S Exception sequence:
-updates status register and makes internal copy
-determines priority and masks lower priority interrupts
-acknowledges interrupt priority level
-saves program counter status register on stack
-puts ISR at head of program counter and executes
-returns from Exception and restores previously blocked task program counter and continues with task

Task interference due to interrupt:
Analysis allows one to set time bounds on real-time/embedded system. For example, one can do an analysis on clock function limitations.

Let t(s) be the time for task s to run by itself
Let t'(s) be the time for task s to run in presence of interrupt I
Let t(I) be interrupt handling time
Let f be the frequency for interrupt I to occur
Then:
$t'(s) = t(s) + t'(s) * f * t(I) \Rightarrow t'(s) = t(s)/[1 - f * t(I)]$ (assuming many interrupts during t(s))

example:
If internal interrupt frequency for refreshing the cache/RAM is 1% of the CPU time, then:
$f * t(I) = .01$, so $t'(s) = t(s)/.99$

exercise:
If an interrupt occurs 1% of the time and t(I) = 1 sec, but a task s normally runs for 10 sec without interrupts, what are the best/worst case execution times when interrupts are enabled.
So f = 1 occurrence per 100 sec = > best case scenario for task s is NO interrupt during its 10 sec run time so that f = 0, or $t'(s) = t(s) = 10$; worst case scenario is that an interrupt DOES happen during task s so that f = 1/10 or .1, and $t'(s) = 10/(1 - .1) = 10/.9$.

Lecture 15: Interrupt analysis (cont)

Kernel system performance is measured by:
-interrupt latency
-task context switch latency
-dispatch latency
-worst case interrupt response time
-worst case interrupt task response time

Interrupt latency (IL):
-The time between the arrival of the hardware interrupt and the start of the execution of the corresponding ISR. This time is statistical in nature whose major sources of contribution are hardware processing time, higher priority handling, task context switching to handler, interrupt disabling time. It is 5 to 15 usec with VxWorks or QNX, but up to 50 usec with Windows CE on a Pentium platform.

Task context switch latency (CSL):
-The time between the end instruction of one task and the first instruction of the next task on the scheduler queue. It is similar in time to the Interrupt latency of the environment minus 1 or 2 usec.

Dispatch latency (DL):
-The time between the end of the ISR routine and the first instruction of the routine pointed to by the ISR.
-Sum of kernel rescheduling time, interrupt re-enabling time, and task context switching time.

Worst case interrupt response time (WIR):
-The sum of interrupt latency time, and worst case ISR execution time.

Worst case interrupt task response time (WITR):
-The sum of worst case interrupt response time and dispatch latency. Task response time of best performance kernels like QNX or VxWorks is 100-150 usec. For Windows NT, it’s unpredictable (along with priority inversion problems).

Exercise: Express these last two definitions in terms of the above defined times and the worst case ISR execution time.