ECE 471 – Embedded Systems
Lecture 19

Vince Weaver

http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

19 October 2020

Announcements

- HW#6 was not posted yet. Maybe delayed.
- Info on project coming soon
- Midterms not quite graded yet
- No class on Wednesday due to Engineering Career Fair

Real Time Constraints

What are real time constraints?

- Time deadlines that hardware needs to respond in.
- Goal not performance, but response time

Types of Real Time Constraints

- Hard – miss deadline, total failure of system. Minor or major disaster (people may die?)
  Antilock brakes?

- Firm – result no longer useful after deadline missed
  lost frames in video, missed frames in video game

- Soft – results gradually less useful as deadline passes.
  Caps lock LED coming on?

Uses of Real Time

Who uses realtime?

- Timing critical situations. Cars, medical equipment, space probes, etc.
- Industrial automation. SCADA. Stuxnet.
- Musicians, important to have low-latency when recording
- High-speed trading

Why isn't everything Real-time?

- It's hard to do right
- It's expensive to do right
- It might take a lot of testing
- It's usually not necessary

Constraints depend on the Application

Try not to over-think things.

Can almost always come up with a scenario where a soft constraint could become hard.

For example: Unlocking a car door taking an extra second? Not hard real-time, except maybe if your car is about to crash and you need to escape quickly.

What can cause problems with real-time?

Sources of "Jitter"

- Interrupts. Taking too long to run; being disabled (cli)
- Unpredictable nature of modern CPUs (see following slide).
- Operating system. Scheduler. Context-switching.
- Dynamic memory allocation, garbage collection.
- Slow/unpredictable hardware (hard disks, network access)
- Memory refresh (LPDDR burst refresh can avoid this a

bit)

Worst Case Behavior – Hardware

- Easier on older and simple hardware
- Old chips like 6502 – fixed clock, each instruction takes an exact number of cycles. Deterministic. With interrupts disabled you can perfectly predict how long code will take.

  Steve Wozniak famously wrote disk firmware on 6502 that more or less cycle-accurate bit-banged stepper motors.

  Also video games, racing the beam.

- Modern hardware more complex:
  - Memory accesses unpredictable with caches – may take 2 cycles or 1000 cycles
  - Interrupts can take unknown amount of time
  - Page faults
  - Branch prediction
  - Power-save may change clock frequency
  - Even in manuals instructions can take a range of cycles

Do a video game keyboard latency example

See Dan Luu's Paper "Computer Latency: 1977-2017"
`https://danluu.com/input-lag/`

- 1977 computers can have less latency to getting keypress on screen than fastest 2010s computers
- Having a fast processor only helps so much
- Slow hardware (keyboards, LCD displays), layers of abstraction in the way

- Apple II (1977) 30ms, modern machines 60-100+ms