CprE 488 – Embedded Systems Design

Exam 2 Review

Joseph Zambreno
Electrical and Computer Engineering
Iowa State University

www.ece.iastate.edu/~zambreno
rcl.ece.iastate.edu

This is the end. My only friend, the end. – The Doors

Announcements

- Exam 2: Thursday 4/26, in class
  - 1 page of notes (8.5x11", single sided, hand-written)
  - 75 minutes
  - No electronic devices, except calculator
  - 15% of your overall grade

- Final demo day: Tuesday 5/1, Noon-2pm (Coover 2041)
  - Submit any and all final materials by 11:59pm on 4/30
  - Demo format:
    - Show off your design (platform and code), have some slides to explain the big picture and finer points of the project
    - Live demo
    - Self-evaluation of your rubric
  - Schedule and timing:
    - 10 minutes x 12 groups = 120 minutes (likely scenario)
    - 20 minutes x 12 groups = 240 minutes (can't do it)

Exam Focus

- Main topics:
  - Interfacing Technologies (Lect-04, first half)
  - Embedded Operating Systems (Lect-05)
  - Software Optimization (Lect-06)
  - Embedded Control Systems (Lect-07)
  - Hardware Acceleration (Lect-08)

- Questions will cover:
  - Conceptual topics (textbook readings and lecture topics)
  - In-class exercises
  - Understanding of lab experiences

Textbook

- Section 9.3: Interfacing
- Section 3.5: Memory System Mechanisms (mostly 3.5.2)
- Chapter 6: Processes and Operating Systems
- Chapter 5: Program Design and Analysis
- Sections 10.4-10.5: Accelerators and Coprocessors

I just know I left it around here, somewhere…

Relevant Labs

- MP-2: Digital Camera Design
  - Color space conversion
  - Image processing pipeline
  - Image framebuffers and VDMA
  - Embedded software and optimization
- MP-3: Target Acquisition
  - Embedded Linux bring-up
  - The ARM device tree
  - Virtual memory and I/O
  - USB driver development
- MP-4: UAV Control
  - Intuition and practicalities for PID control
  - Working with accelerometers and gyroscopes
  - Understanding Bluetooth

Lectures (Week 7)

- Embedded Operating Systems
  - Processes and scheduling
  - Atomic operations and inter-process communication
  - Virtual memory
  - ARM architecture support

EX: Embedded Operating Systems

1. What is the purpose of a context switch? Using assembly (pseudocode is acceptable), describe the main steps required to perform a context switch on an ARM processor running Linux. Do not use any non-conventional instructions unless you can clearly explain what they are doing.
2. For the periodic processes and deadlines given below:
   - Schedule the processes using RMS
   - Schedule using EDF and compare the number of context switches required for EDF and RMS

Process	Cost (ms)	Deadline (ms)
P1	2	30
P2	5	40
P3	7	120
P4	5	60
P5	1	15

Lectures (Week 9)

- Software Optimization
  - Understanding processor performance
  - Early optimizations (redundancy elimination, operator simplification)
  - Loop restructuring
  - Data representation

EX: Software Optimization

- Explain how loop unrolling, loop unswitching, and loop fusion can be applied by a compiler to the C code segment below. Describe the potential benefit and any possible disadvantages from applying these optimizations aggressively

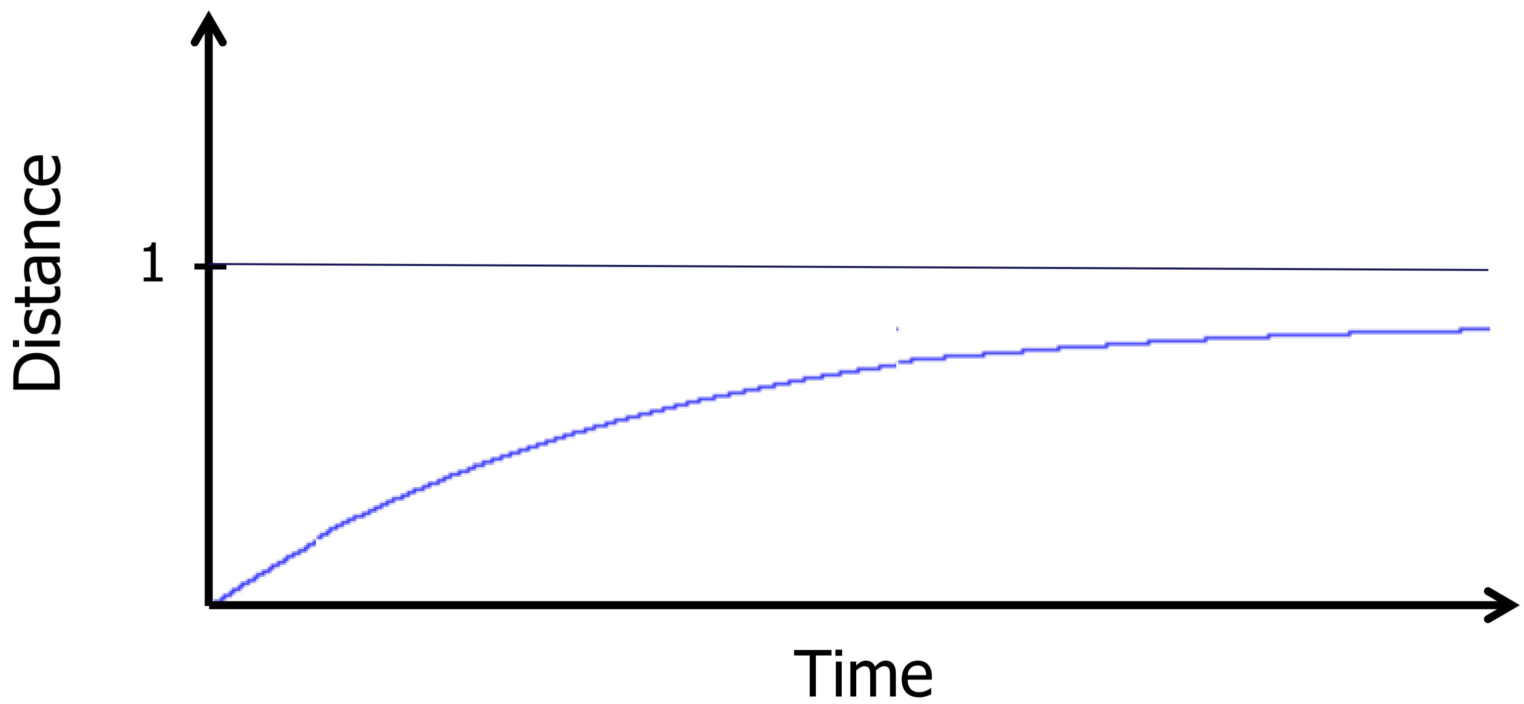
```
for (i = 0; i < N; i++) {
  if (j == 23) {
    A[i] = B[i] + 1;
  }
  else {
    A[i] = B[i] + 2;
  }
}

for (i = 0; i < N; i++) {
  C[i] = A[i] / 2;
}

for (i = 0; i < N; i++) {
  D[i] = C[i+1] * B[i];
}
```

Lectures (Week 11)

- Embedded Control Systems
  - Open-loop vs. closed-loop
  - PID control (continuous-time and discrete-time form)
  - Understanding P vs I vs D components
  - Nested PID (inverted pendulum example)
  - Model-based control

EX: Control Systems

- Assuming a PID controller is used to apply a force to move an object to a desired location. For the plot below (showing the response of the object when moved from 0 to 1), what PID settings would likely result in the observed behavior?



Lectures (Week 14)

- Hardware Acceleration
  - Motivation: Moore vs. Dennard vs. Amdahl
  - Performance analysis
  - Coprocessors vs. accelerators
  - Some common techniques and challenges
  - Case studies

EX: Hardware Acceleration

- For a randomly generated 32-bit value, how long (in terms of ARM instructions) would the following code take?
- What would a corresponding HW accelerator look like? What would the potential performance savings be?

```
int32_t hamming_distance(uint32_t x, uint32_t y) {
 int32_t dist = 0; uint32_t val;
  val = x ^ y; // XOR

  // Count the number of bits set
  while (val != 0) {
    if(val & 1)
    {
      dist++;
    }
    val = val >> 1;    // ?? val &= val - 1; ??
  }
  return dist;
}
```