23. Wrap Up & Exam Prep

18-342: Fundamentals of Embedded Systems

Rajeev Gandhi

Recommended Readings and Lecture Slides
are available on CMU's BlackBoard

Carnegie Mellon

Electrical & Computer ENGINEERING

Announcements

- Schedule for the final exam
  - Monday Dec 17th **1.00pm-4.00pm** (Pittsburgh time), DH 2315
  - Will include everything covered in the course
  - All clarifications will be posted on the smartboard
  - Open book, open notes
  - Bring your calculators (no laptops will be allowed)

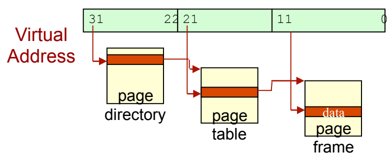
- Make sure that your scores are correct on Blackboard

Course Objectives

- Understand the low-level concepts in programming embedded systems
  - Assembly programming, details of linkers and loaders, debugging embedded systems, performance optimization, integrating peripheral devices in embedded systems, task scheduling, inter-task communication & synchronization, real-time systems, virtual memory…
- Obtain hands-on experience in programming embedded systems
  - Lab 1
    - Familiarizing you with development environment, code optimization exercises
  - Lab 2
    - Write wrappers for some of C library provided syscalls
    - Write your own C library and use SWI hijacking mechanism to install your own SWI handlers
  - Lab 3
    - Use the Gumstix timer in interrupt mode to add time related syscalls to your library
  - Lab 4
    - Gravelv2 scheduler and inter-task synchronization
    - Real-time task scheduling and preventing unbounded priority inversion

Memory Management

- How do programs run in memory?
  - What happens on "overflows"?
- **Virtual Memory** (VM)
  - How does VM work?
  - Virtual address to physical address mappings
  - Page faults
  - VM Schemes
    - page tables
    - virtual pages and physical page frames
    - page table entries
    - interactions with the rest of the system
    - two-level page tables



Virtual Memory Problem

Problem Statement –

- You are analyzing an extremely simple virtual memory system that has 256 total physical addresses, each the size of a single char, i.e. 8 bits. While the virtual address is 8 bits wide, we're going to ignore the two most significant bits, effectively making each virtual address 6 bits wide for your consideration.
  - The system's page directory has a maximum of 4 entries.
    - The page directory is stored starting at address 0x0.
  - The page tables are laid out contiguously.
    - Each page table has at most 4 entries.
    - Each page table points to the bottom of the page frame, and the offset into the frame is also calculated from the bottom of the page frame.
  - Each entry in the page directory and the page table is 16 bits wide.
    - The first encountered byte of every entry contains an address (the other byte contains status information, and can be ignored).

Addr	**[Addr + 0]**	**[Addr + 1]**	**[Addr + 2]**	**[Addr + 3]**
0x0	0x8	0x9	0x20	0x9
0x4	0x10	0x9	0x18	0x9
0x8	0x60	0x9	0x90	0x9
0xC	0x68	0x11	0x74	0x11
0x10	0x70	0x9	0x7C	0x9
0x14	0x78	0x9	0x6C	0x11
0x18	0x8C	0x11	0x64	0x11
0x1C	0x98	0x9	0x94	0x9
0x20	0x80	0x9	0x84	0x9
0x24	0x88	0x9	0x9C	0x11
…				
…				
0x60	‘a’	‘b’	‘c’	‘d’
0x64	‘e’	‘f’	‘g’	‘h’
0x68	‘i'	‘j’	‘k’	‘l’
0x6C	‘m’	‘n’	‘o’	‘p’
0x70	‘q	‘r’	‘s’	‘t’
0x74	‘u’	‘v’	‘w’	‘x’
0x78	‘y’	‘z’	‘1’	‘2’
0x7C	‘3’	‘4’	‘5’	‘6’
0x80	‘7’	‘8’	‘9’	‘0’
0x84	‘+’	‘-‘	‘=’	‘*’
0x88	‘!’	‘@’	‘#’	‘$’
0x8C	‘%’	‘^’	‘&’	‘(‘
0x90	‘)’	‘<’	‘>’	‘;’
0x94	‘a’	‘b’	‘c’	‘d’
0x98	‘e’	‘f’	‘g’	‘h’
0x9C	‘i'	‘j’	‘k’	‘l’
…				

Virtual Memory Problem – continued

- Your software performs *read* operations on the following addresses. What character would each yield?

0x24	
0x65	
0xeb	
0x16	
0x1d	
0x2e	
0x68	

Real-Time Synchronization Protocols

- Synchronization in real-time systems
  - **Priority inversion**
  - Unbounded priority inversion
  - Protocols to bound priority inversion
    - **Basic Priority Inheritance Protocol**
    - **Priority Ceiling Protocol**
    - **Highest Locker protocol**

Priority Inversion & Blocking Time

- Consider the following task set
  - T1 = (40, 100, 100)
  - T2 = (60, 200, 200)
  - T3 = (150, 600, 600)
  - T4 = (240, 1200, 1200)

- There are critical sections guarded by 3 mutexes M1, M2 and M3
- The critical sections are not nested
- Assume that we use rate-monotonic scheduling

Priority Inversion & Blocking Time

- Within its execution time, T1 accesses a critical section guarded by M1 for 15 ms and after normal execution, it enters a critical section guarded by M3 for 18 ms. In other words, these two critical sections are not nested.
- During its execution, T2 accesses a critical section guarded by M2 for 10 ms.
- During its execution, T3 accesses a critical section guarded by M1 for 20 ms and after normal execution, it enters a critical section guarded by M3 for 15 ms. In other words, these two critical sections are not nested.
- During its execution, T4 accesses a critical section guarded by M2 for 30 ms and after normal execution, it enters a critical section guarded by M3 for 25 ms. In other words, these two critical sections are not nested.

Priority Inversion & Blocking Time

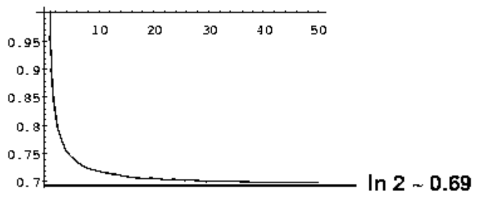
- What if we used the priority inheritance protocol?

- What is the *maximum* priority inversion encountered by each of the tasks T1-, T2, T3 and T4 using the basic priority inheritance protocol?

Analyzing Periodic Tasks

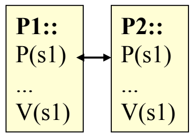
- Key concepts in real-time computing.
  - How to analyze the schedulability of **independent periodic tasks**

$$\frac{C_1}{T_1} + \dots + \frac{C_n}{T_n} \leq U(n) = n(2^{1/n} - 1)$$

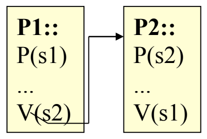


Synchronization and Deadlocks

- Process synchronization
- **Deadlocks**
  - necessary conditions for deadlocks
  - dealing with deadlocks
- Some **classical synchronization problems**
  - producer/consumer problems
  - the "dining philosophers" problem



Mutex



Barrier Synchronization

Concurrency and Deadlocks

- In our system, we have *n* processes competing for a number of resources, say, 6 identical video cards and 6 identical cameras. Each process might need 1 video-card and 1 camera to complete its job. Assume that we do not number the cameras and the video cards, and that we do not require a specific order in which they are to be acquired. <u>Please show all your work and justify your answer.</u>
  - If each process is well-behaved, i.e., it does not hang onto either the video card or the camera forever, list all the values of *n* for which the system is guaranteed to be deadlock-free.

Concurrency & Deadlocks

- The following is a set of three interacting processes that can access two shared semaphores S1 and S2
- Within each process the statements are executed sequentially, but statements from different processes can be interleaved (subject to the semaphores, of course). Assume that once execution starts, the processes will execute until all 3 processes are stuck waiting on a semaphore, at which point execution stops.

```
semaphore S1 = 3
semaphore S2 = 0

Process 1              Process 2              Process 3

while (1) {            while (1) {            while (1) {
wait(S1);              wait(S2);              wait(S2);
printf("C");           printf("A");           printf("D");
signal(S2);            printf("B");           }
}                      signal(S2);
                       }
```

Concurrency & Deadlocks – Answers

- Assuming all three processes eventually stop executing, how many C's are printed when the set of processes runs?
- Assuming all three processes eventually stop executing, how many D's are printed when this set of processes runs?

Concurrency & Deadlocks – Answers

- What is the smallest number of A's that might be printed when this set of processes runs?

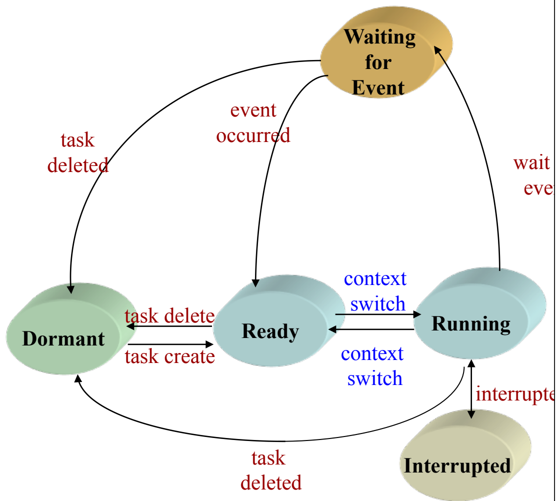
- Is CABABDDCABCABD a valid output sequence when this set of processes runs? Please show the sequence of actions that could or could not lead to this output.

Concurrency

- **Race Conditions and Critical sections**
  - the atomicity requirement
- Techniques to enforce critical sections
  - Solution 1: too hard-wired
  - Solution 2: does not work!
  - Solution 3: deadlock!
  - Solution 4: starvation!
  - Solution 5: Peterson's algorithm
  - **Semaphores**: good…
- Implementing Semaphores
  - test-and-set instructions
  - ARM's atomic swap instruction

Process Scheduling

- OS "Scheduling"
  - Processes/Tasks
    - Context Switching
    - Process State
    - Ready list
- **Process Scheduling algorithms**
  - FCFS scheduling
  - Shortest-Job-First Scheduling
  - Priority Scheduling
  - Multi-Level Round-Robin Scheduling



Serial Comm. & Interrupts

- **Serial Communications**
  - Data communications and modulation
  - Asynchronous protocols
  - Serial port and bit transmission
  - Serial I/O from device drivers
  - Parity
- Interrupts
  - interrupt handlers
  - nested interrupts
  - interrupt timing and metrics
  - installing and writing interrupt handle
  - Concurrency issues

Memory-mapped I/O & Timers

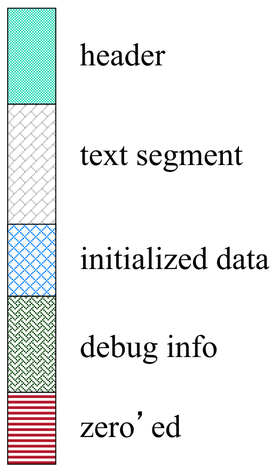
- Timers
  - What is a timer?
  - A peek inside the timers
  - Using the timer on Gumstix
- Difference between normal memory locations and memory-mapped I/O
- The use of volatile to make sure compilers don't cache memory mapped registers in processor registers

Practice Problems

- Are register contents saved by an interrupt handler?
- Can interrupt handlers be interrupted by other interrupts?
- How do serial communications handle error detection?
- If two devices interrupted the processor simultaneously, will two interrupt handlers will begin to execute simultaneously?

ARM Architecture, Assembly, SWIs, ELF & Memory

- ARM Programmer's Model
- Introduction to ARM Assembly Language
- Software Interrupts (SWIs)
  - What is a SWI?
  - What happens on an SWI?
  - What happens on SWI completion?
  - What do SWIs do?
  - A Full SWI Handler
  - A C_SWI_Handler (written in C)
- Linking and Loading
- ELF Image Format



In-memory layout

Practice Problems

- ARM performs instruction *prefetching* (i.e., fetching instructions from memory before the previous instruction has completed execution). Prefetching does not commit the processor to execute the prefetched instruction. Name two cases where the prefetched instruction is not executed, and might be discarded.
- Why is FIQ faster than IRQ ?
- Name one advantage of static linking and one of dynamic linking?
- **Be prepared to interpret assembly code in the final exam**

Short Answers

- Which register(s), apart from r3, is/are involved when the following assembly code is executed?

```
CMP r3, #0
BNE next
```

- Among r0 to r14, which one's value will change after STMFD sp!, {r0-r2, lr} and by how much from its old value?
- Explain TWO circumstances under which loop unrolling provides little or modest improvement.

Topics Covered

- What are embedded systems?
- Calling conventions, C/asm Interfacing
- System Calls (SWIs)
- Program Monitor and Loading
- Code Optimization
- I/O Basics
- Gumstix board
- Serial Communications
- Interrupts and Device Drivers

- Memory Management
- Basic Concurrency
- Processes and CPU Scheduling
- Mutual Exclusion
- Deadlocks

- Real-Time Resource Management
  - Independent Periodic Tasks
  - Task Synchronization

- Real-Time Operating Systems

We covered a wide spectrum.

Topics Not Covered

- Designing embedded systems (18-549: Embedded System Design)
- Distributed Embedded Systems (18-649)
- Fault-Tolerant Embedded Systems (18-749: Fault-Tolerant Distributed Systems)
- Dependable Embedded Systems (18-849)
- Software engineering issues
  - Courses offered by Software Engineering Institute and/or ISRI
- Security issues in embedded systems
- Applications
  - Signal Processing (18-551)
  - Embedded Control Systems (18-474)

Tips for the Final Exam

- Please make sure to **manage your time** during the exam
  - Use the first 5-10 minutes (extra time) to scan the exam and allocate time wisely across various questions
- State all your assumptions in answering each question. It's possible that your set of assumptions lead to a different answer, and we might have overlooked this.
  - Of course, no credit will be given if you simplify the problem because of your assumptions
- Answer every question, even if it is to show us how you were planning to approach the problem.
  - Partial credit is provided for starting on the right approach, even if you don't get the final/correct answer.
- Always raise your hand, and ask questions to clarify your doubts. Please take the time to read the clarifications posted.
- Don't panic or freeze – we will not ask anything beyond what we have taught in the lectures. If you have attended all the lectures, and followed the material, you should be just fine!

And Finally ….

- The best news of all
  - The course is almost over!
- Your feedback matters
  - Please let us know what you liked and what you did not
  - I would appreciate a note so that we know how to improve
- Will be looking for TAs for Fall 2013 in April

Good Luck With Finals