Advanced

操作系统 Operating System

Yu Chen

Tsinghua University

2015

Who ?

- **Lecturer: Chen Yu**
  - Assciate Professor of Computer Science&Technology
  - yuchen@tsinghua.edu.cn
  - Interests:
    - **Operating Sytems**

- TA: Qixue Xiao, Junjie Mao,

Why Study OS?

- The Operating System (OS) I use has already been written, and I doubt it will be my job to write another one. For example, Windows, Linux.
- Haven't OS developers figured everything out already? What more is there to do?
- Why should I study this as a graduate student?

Why Study OS?

- The Operating System (OS) I use has already been written, and I doubt it will be my job to write another one. For example, Windows, Linux.
- Haven't OS developers figured everything out already? What more is there to do?
- Why should I study this as a graduate student?

OS is cool!

Why Study OS?

- The Operating System (OS) I use has already been written, and I doubt it will be my job to write another one. For example, Windows, Linux.
- Haven't OS developers figured everything out already? What more is there to do?
- Why should I study this as a graduate student?

OS is cool!

OS is important!

Why Study OS?

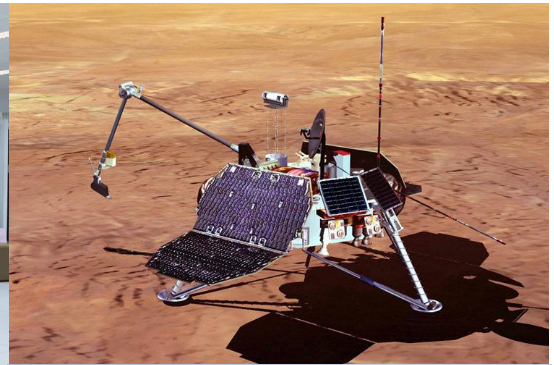
- The Operating System (OS) I use has already been written, and I doubt it will be my job to write another one. For example, Windows, Linux.
- Haven't OS developers figured everything out already? What more is there to do?
- Why should I study this as a graduate student?

OS is cool!

OS is important!

OS is challenging!

You want to be involved!



Objectives

- Gain experience in doing OS research
  - Know how to read/write papers/reports

Objectives

- Gain experience in doing OS research
  - Know how to read/write papers/reports
  - Know current OS hot topics

Objectives

- Gain experience in doing OS research
  - Know how to read/write papers/reports
  - Know current OS hot topics
  - Develop and test OS projects

Course Materials

- Lecture notes, Papers & Projects
- No required textbooks

Course Materials

- Lecture notes, Papers & Projects
- No required textbooks

Reference Books

- Wolfgang Mauerer, Professional Linux Kernel Architecture
- Uresh Vahalia, UNIX Internals-- The New Frontiers
- Daniel P. Bovet, Marco Cesati, Understanding the Linux Kernel,
- Mark E. Russinovich, David A. Solomon, Microsoft Windows? Internals
- Tanenbaum, Modern Operating Systems
- Andrew S. Tanenbaum,Distributed Operating Systems

Scheduling

Week	Lectures (includes some invited talks, Interviews)
1	Course overview + Speed+Security+Correct
2	OS Architecture
3	Virtual Machine
4	Building Real OS—uCore, Linux
5	Overview of OS Optimization for Multicore
6	Paper&Project: The Scalable Commutativity Rule & Commuter
7	Paper&Project: TCP/IP & FastSocket
8	Midterm Exam
9	Overview of OS Security
10	Paper&Project: Integer Security for System & Kint
11	Paper&Project: Lang, Library, Device Driver & RUST ,KLEE, PF-Miner
12	Overview of Correction on System
13	Paper&Project: Verified OS & SeL4
14	Paper&Project: Verified Component & JITK
15-16	Final Exam

Scheduling

Week	Lectures (includes some invited talks, Interviews)
1	Course overview + Speed+Security+Correct
2	OS Architecture
3	Virtual Machine
4	Building Real OS—uCore, Linux

Scheduling

Week	Lectures (includes some invited talks, Interviews)
5	Overview of OS Optimization for Multicore
6	Paper&Project: The Scalable Commutativity Rule & Commuter
7	Paper&Project: TCP/IP & FastSocket
8	Midterm Exam

Scheduling

Week	Lectures (includes some invited talks, Interviews)
9	Overview of OS Security
10	Paper&Project: Integer Security for System & Kint
11	Paper&Project: Lang, Library, Device Driver & RUST ,KLEE, PF-Miner

Scheduling

Week	Lectures (includes some invited talks, Interviews)
12	Overview of Correction on System
13	Paper&Project: Verified OS & SeL4
14	Paper&Project: Verified Component & JITK
15-16	Final Exam

Reading Grading

- 6 Papers Summaries and critiques 60%
  - include report doc(4+ or 4000 words inA4 pages) & presentation (15+ pages)
  - Research Areas
    - OS/VMM Architectures
    - Performance/Multicore
    - Security/Find Bug
    - Correction/Verification

Reading Grading

- 6 Papers Summaries and critiques 60%
  - include report doc(4+ or 4000 words inA4 pages) & presentation (15+ pages)
  - Research Areas
    - OS/VMM Architectures
    - Performance/Multicore
    - Security/Find Bug
    - Correction/Verification
  - The best papers are from below conferences: SOSP, OSDI, EuroSys, RTSS, NSDI, FAST, USENIX ATC, VEE, MobiSys , PPoPP, SC , ICS, PLDI, SenSys , SIGMETRICS , ISCA, HPCA, HPDC, Micro, ASPLOS, IPDPS, SIGMOD, SIGCOMM, PerCom, WWW , Cluster, CCGrid, S&P, CCS, SECURITY, NDSS,… (1996~2015)

Reading Critiques

- Need to address the following:
  - Summary of major innovations
  - What the problems the paper mentioned?
  - How about the important related works/papers?
  - What are some intriguing aspects of the paper?

Reading Critiques

- Need to address the following:
  - Summary of major innovations
  - What the problems the paper mentioned?
  - How about the important related works/papers?
  - What are some intriguing aspects of the paper?
  - How to test/compare/analyze the results?
  - How can the research be improved?
  - If you write this paper, then how would you do?

Project Grading

- One Project 40%
  - didn't finish ucore/xv6/jos labs
    - Analyze/Testing/Improving ucore OS
    - https://github.com/chyyuu/ucore

Project Grading

- One Project 40%
  - didn't finish ucore/xv6/jos labs
    - Analyze/Testing/Improving ucore OS
    - https://github.com/chyyuu/ucore_lab
  - already finished ucore/xv6/jos labs
    - Do one OS related project and write the final report
    - research areas
      - There are a project lists
      - Discuss with me

Project Proposal

- Due on the 2nd~3rd week
- 2-page written proposal
  - team members (<=3)
  - Motivation
  - The state-of-the-art Methodology
  - Expected results
  - Timeline
  - Division of labor among teams
  - Some references

Project Midterm/Final Exam

- Needed
  - 10-15 minutes Presentation
  - Midterm
    - 3~10 page written paper/report (double column, single-space, 10-pt font)
  - Final
    - 5~15 page written paper/ project report (double column, single-space, 10-pt font)

Other Choose

- I can not understand the paper, my coding ability is poor
  - Need to talk with me ASAP

OS Overview

History of OS: Change!

		1980	2009	Factor
Speed	CPU	1 MIPS	77,000 MIPS	7.7×10^5
	Memory	500 ns	0.9 ns	1.8×10^3
	Disk	18 ms	4 ms	2.2×10^1
	Network	300 b/sec	10 Gb/sec	3.3×10^7
Capacity	Memory	64 KB	8 GB	1.3×10^5
	Disk	1 MB	2 TB	2.0×10^6
Cost	Per MIP	$100K/MIP	$.007899	1.0×10^7
Other	Address bits	8	64	8
	Users/machine	10s	0.1	1.0×10^{-2}

Changing Roles of the OS

- What OS does depends on
  - available hardware and software
  - changing uses of machines
  - changing expectations of users

OS Concepts

- Single-Machine OS/VMM
  - Memory management
  - Process management
  - Synchronization
  - File systems and device support
- Security
- Correction
- Distributed System

Single-Machine OS/VMM

Purposes

- Clean virtual machine
- Hardware independence
- Resource sharing and management
- Persistent Data Storage
- Protection
- Real time support
- Parallelism

Single-Machine OS/VMM

Purposes

- Clean virtual machine
- Hardware independence
- Resource sharing and management
- Persistent Data Storage
- Protection
- Real time support
- Parallelism

Strategy

- How do we organize the OS effectively for development, evolution, performance, and security?
- How do we use multi-processor machines effectively?

Memory Management

Purposes

- Virtual memory: provides the illusion of infinite physical memory
- Swapping: moves processes to disk as necessary
- Paging: allows processes to run with only the active pages in memory
- Buffer Cache: speedup the IO access

Memory Management

Purposes

- Virtual memory: provides the illusion of infinite physical memory
- Swapping: moves processes to disk as necessary
- Paging: allows processes to run with only the active pages in memory
- Buffer Cache: speedup the IO access

Strategy

- How do we coordinate machines to share memory?
- How can we simplify memory management as memory becomes abundant?

Process Management

- Thread:
- Address space:
- Process:

Process Management

- Thread: A sequential execution stream
- Address space: Chunks of memory and everything needed to run a program
- Process: An address space + thread(s)

Process Management

Purposes

- Thread: A sequential execution stream
- Address space: Chunks of memory and everything needed to run a program
- Process: An address space + thread(s)

Strategy

- How do processes communicate and share states efficiently and securely on the same machine?
- How do we improve the computing process model?

Process Scheduling

Purposes

- Provides the illusion of multiple processes running at the same time on a single processor
- Context switching: changing the attention of the processor
  - Involves saving and restoring states
  - Necessary to cross kernel boundary

Process Scheduling

Purposes

- Provides the illusion of multiple processes running at the same time on a single processor
- Context switching: changing the attention of the processor
  - Involves saving and restoring states
  - Necessary to cross kernel boundary

Strategy

- How do we achieve fairness, high throughput, and responsiveness at the same time?
- How do we reduce or avoid the cost of context switching?

Memory Management

Purposes

- Virtual memory: provides the illusion of infinite physical memory
- Swapping: moves processes to disk as necessary
- Paging: allows processes to run with only the active pages in memory
- Buffer Cache: speedup the IO access

Memory Management

Purposes

- Virtual memory: provides the illusion of infinite physical memory
- Swapping: moves processes to disk as necessary
- Paging: allows processes to run with only the active pages in memory
- Buffer Cache: speedup the IO access

Strategy

- How do we coordinate machines to share memory?
- How can we simplify memory management as memory becomes abundant?

Synchronization

Purposes

- Provides correct execution or coordinating threads in the face of arbitrary context switching

Synchronization

Purposes

- Provides correct execution or coordinating threads in the face of arbitrary context switching

Strategy

- Atomic actions: all or nothing
- Mutual exclusion: one thread in the critical section at a time
- Semaphores: atomic, counter-based locks
- Avoid Deadlock: circular waiting on resources

File Systems

Purposes

- File: data + attributes
- File system services:
  - Organization
  - Naming
  - Access
  - Synchronization
  - Protection and security

File Systems

Purposes

- File: data + attributes
- File system services:
  - Organization
  - Naming
  - Access
  - Synchronization
  - Protection and security

Strategy

- How do we make different file systems work together, even across machines?
- How do we provide consistency, availability, and reliability to copies of a file across multiple machines?
- How do we handle very large data sets?

I/O Device

Purposes

- I/O devices tend to be a lot slower than memory speed
- Caching: stores extra data in memory in hope of near-term reuse

I/O Device

Purposes

- I/O devices tend to be a lot slower than memory speed
- Caching: stores extra data in memory in hope of near-term reuse

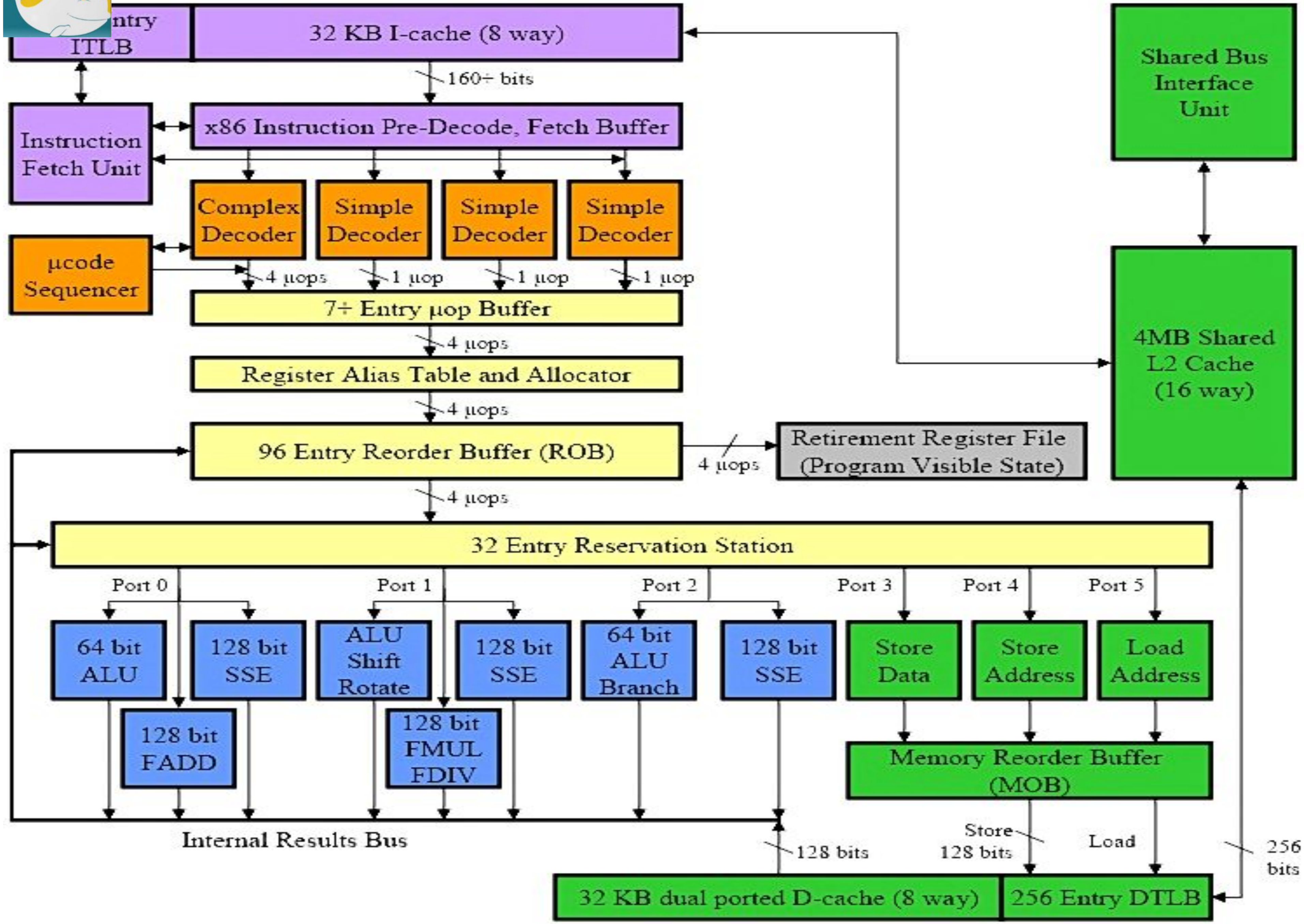
Strategy

- How do we coordinate the memory resources across machines to enhance performance?
- How do we handle new devices with new characteristics?

Tendency

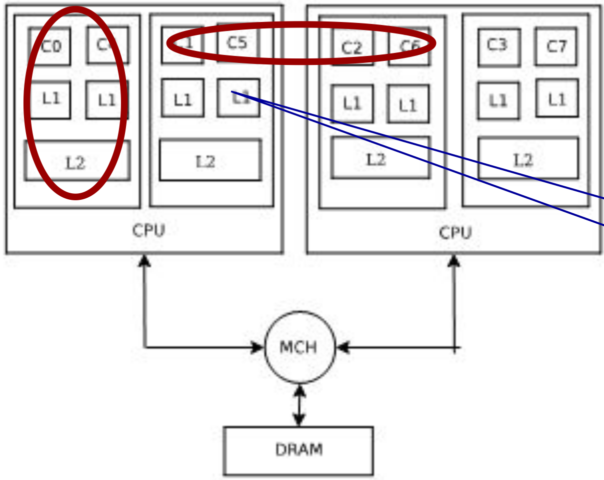
- MultiCore
- Security
- Correction

Core Microarchitecture



Multi-Core Challenges

- Today's Commodity Multi-Cores



SMP+CMP

Fig. 1. The architecture of Intel's multi-core platform used in experiments. Squares with *C* tag represent cores, squares with *L1* tag represent L1 instruction cache and L1 data cache, rectangles with *L2* represent L2 unified cache, rectangle with *DRAM* represents memory, circle with *MCH* represents memory controller hub, lines with arrow represent directly accessible relationship and rectangles with *CPU* represent chips. Each cache line is 64 bytes, both L1 instruction cache and L1 data cache are 32K bytes 8-way set associative, and the L2 cache is 4M bytes 16-way set associative.

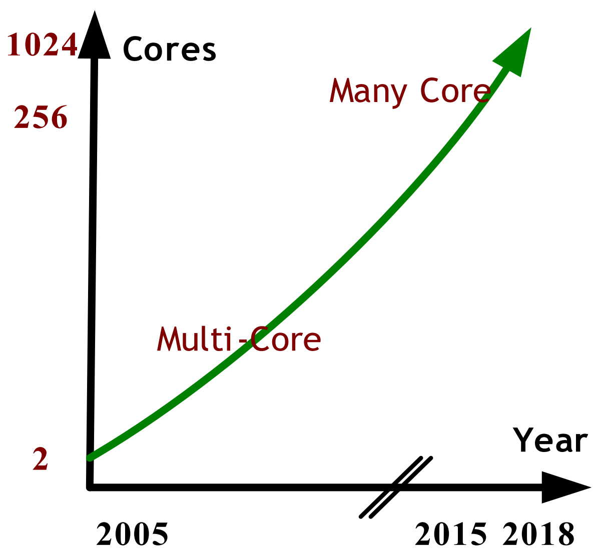
Applications should avoid the memory wall

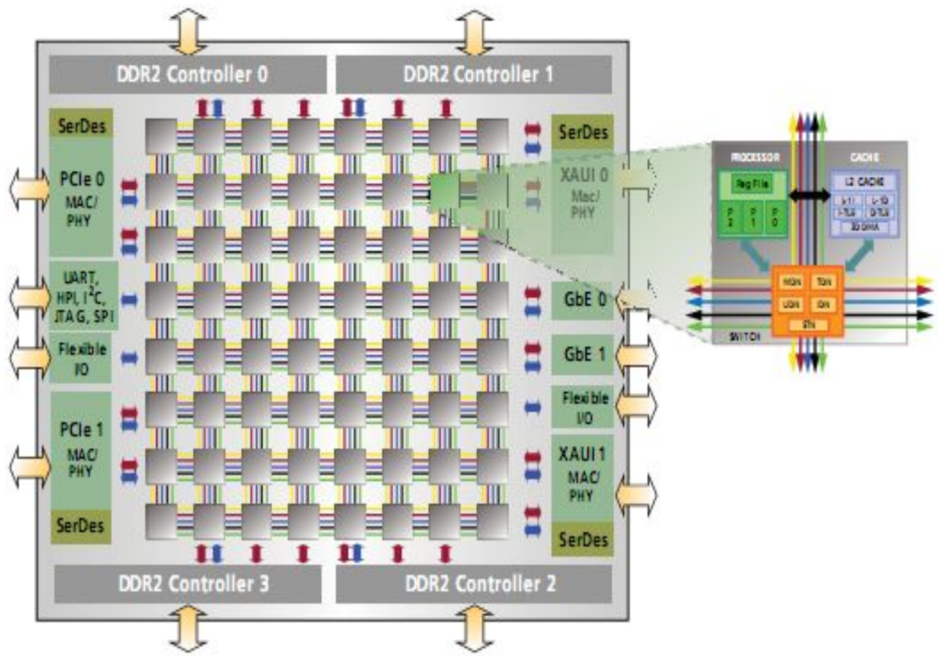
Memory References	Latency
memory read	212 cycles
L1 data cache hit	3 cycles
L2 unified cache hit	14 cycles
local cache-to-cache transfer	46 cycles
remote cache-to-cache transfer	50 cycles

Cache-to-cache transfer should be avoided

Multi-Core Challenges

- CMP V.S. SMP
  - **More cores can be integrated**
    - SMP: Low(2CPU),Middle(4~8CPU),High(>16CPU)
    - CMP: 4~8 cores systems, **1000+ cores (<10years)**
      E.g. Intel's 80 cores chip & Tilera's 64 core chip





Multi-Core Challenge

- However, can operating systems and applications use these cores effectively?

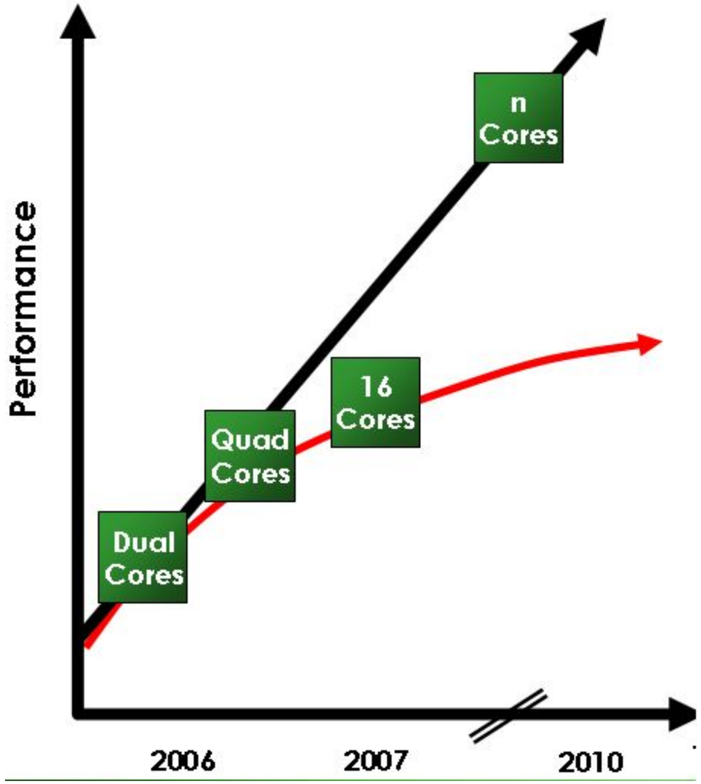
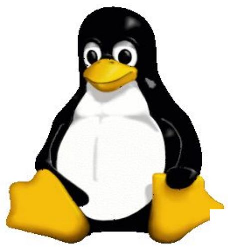


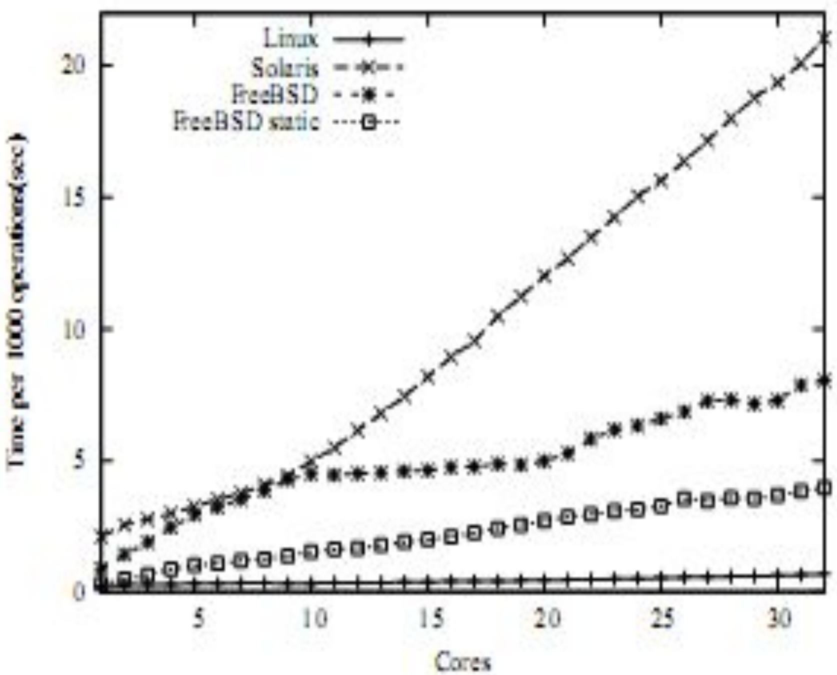
Fig 2. Performance bound by bottlenecks

OS supporting Multicore

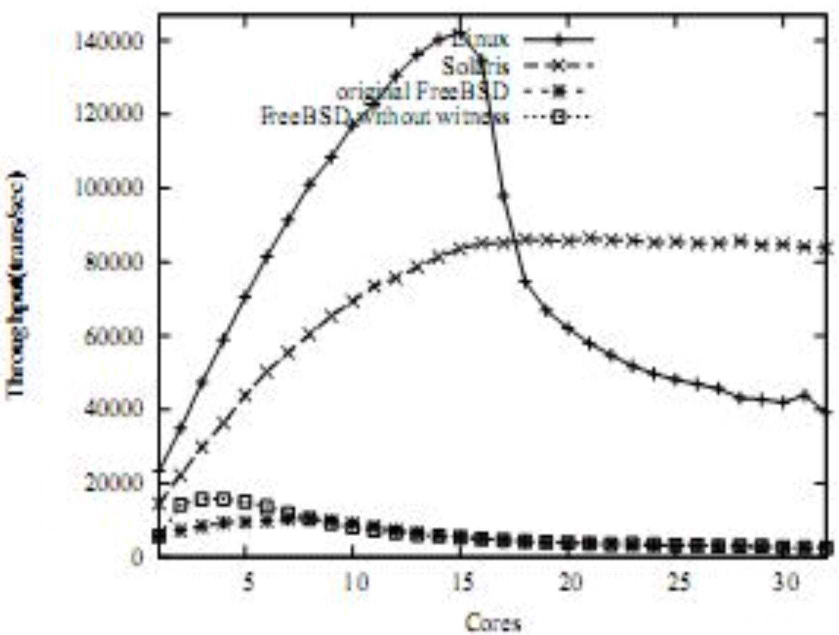
- Linux
- Solaris
- FreeBSD
- Windows
- VxWorks



Benchmarks



Linux
Solaris
FreeBSD
FreeBSD static
Time per 1000 operations(sec)
Cores



Linux
Solaris
original FreeBSD
FreeBSD without witness
Throughput(trans/sec)
Cores

Some Conclusions

- No system scales clearly better than another in all aspects for micro-benchmark test
- Linux and Solaris are competitive in application benchmark test, FreeBSD loses both in performance and scalability
- Kernel synchronizations protecting the shared data structure are the main bottlenecks on multi-core platform

Tendency

- MultiCore
- Security
- Correction

History of Security Problem

- Originally, there was no security problem
- Later, there was a problem, but nobody cared
- Now, there are increasing problems, and people are beginning to care
  - Automation
  - Action at a distance
  - Technique propagation

Threat Analysis

- What are we trying to protect? (and why?)
- What are the vulnerabilities of those assets?
- Who might exploit a vulnerability?
  - Either on purpose or by accident

Threat Analysis

- What are we trying to protect? (and why?)
- What are the vulnerabilities of those assets?
- Who might exploit a vulnerability?
  - Either on purpose or by accident
- How can we prevent a specific threat?
- How much is it worth to us to prevent it?
- How much will it cost to prevent it?

The Core Technical Problem

- Controlling access to machine and data resources
- Controlling the way *access rights* are passed from holder to holder
  - person to person
  - program to program
- Preventing maliciousness and errors from subverting the controls

Access Rights

- In general case, need triplet for every possible combination of *right*, protected *asset,* and potential user
  ⇓ (U, A, R)
- …and some entity must be responsible for checking and enforcing any limitation...
- The 3-D matrix is hard to manage…We need a simpler approach!

ACL&Capabilities

	file1	*file 2*	*file 3*	*device*	*domain*
User/Domain 1	*r*	*rx*	*rwx*	–	*enter*
User/Domain 2	*r*	*x*	*rx*	*rwx*	–
User/Domain 3	*rw*	–	–	–	–
…					

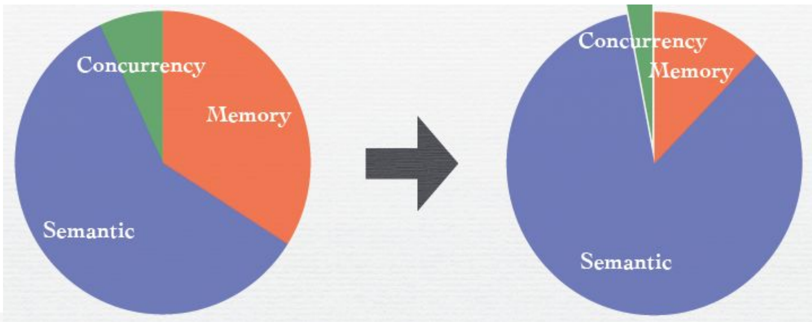
- Columns are *access control lists* (ACLs)
  - Associated with each object
- Rows are *capabilities*
  - Associated with each user or each domain

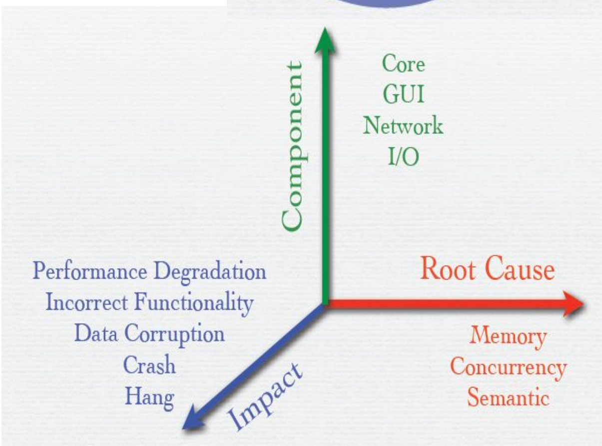
What about efficiency?

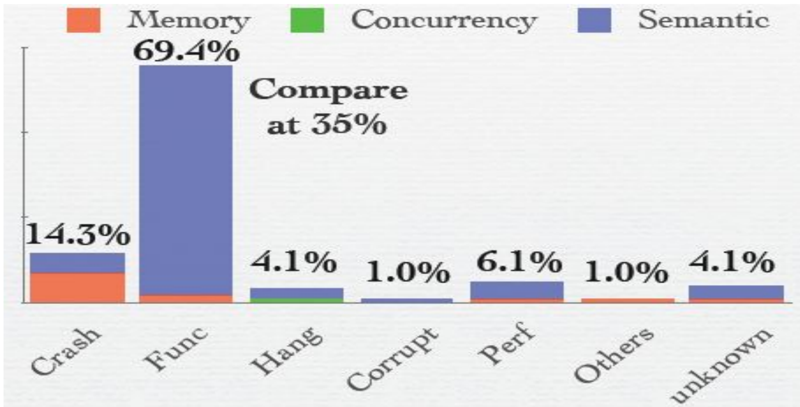
- At run-time…
  - What does the OS know about the user?
  - What does the OS know about the resources?
- What is the cost of checking and enforcing?
  - Access to the data
  - Cost of searching for a match

Current Status (1)

- 对当前Android漏洞的理解
  - Sematic Vulnerability 越来越多
  - 数据泄漏漏洞的威胁越来越大

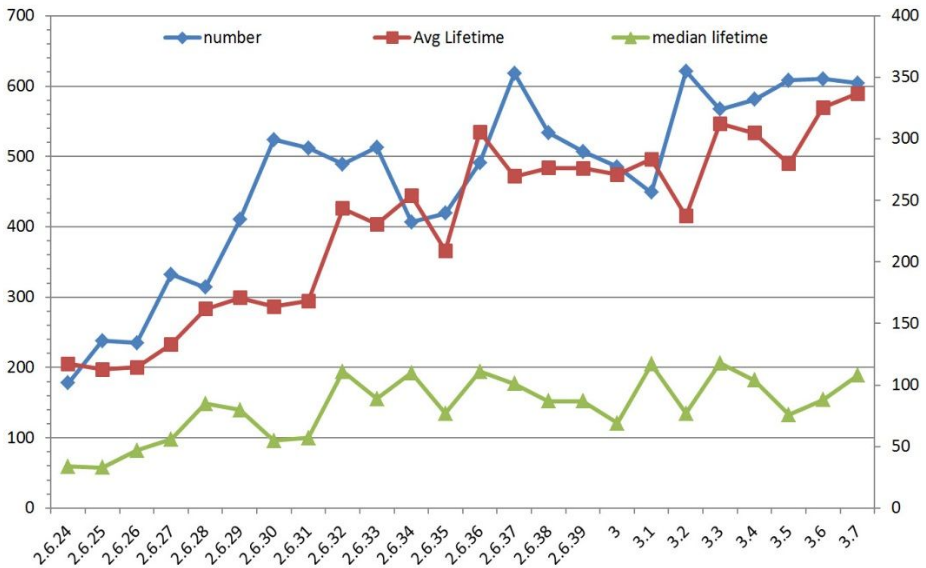


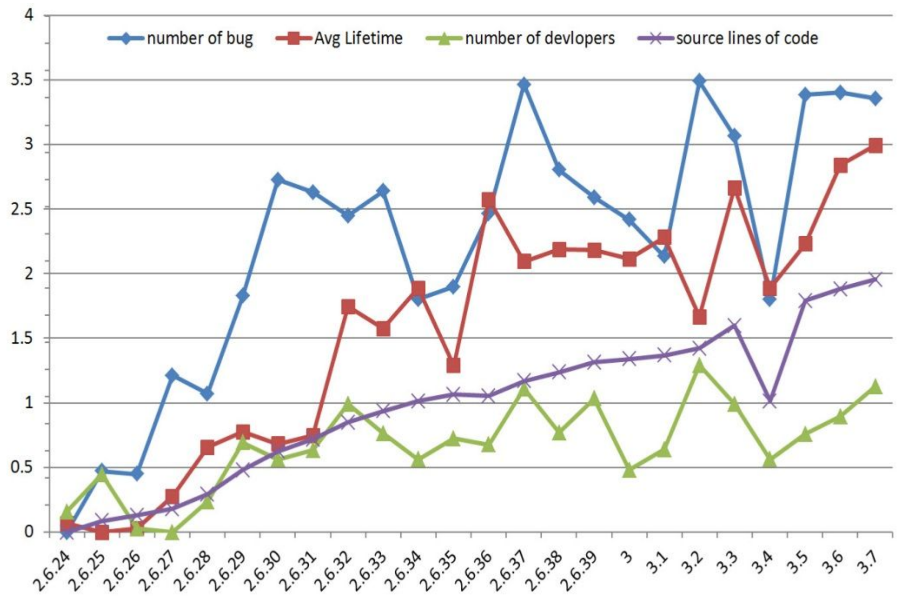




Current Status (2)

- 对当前Linux Kernel漏洞的理解
  - Linux漏洞有扩大化的趋势
  - 但发现Linux漏洞难度加大

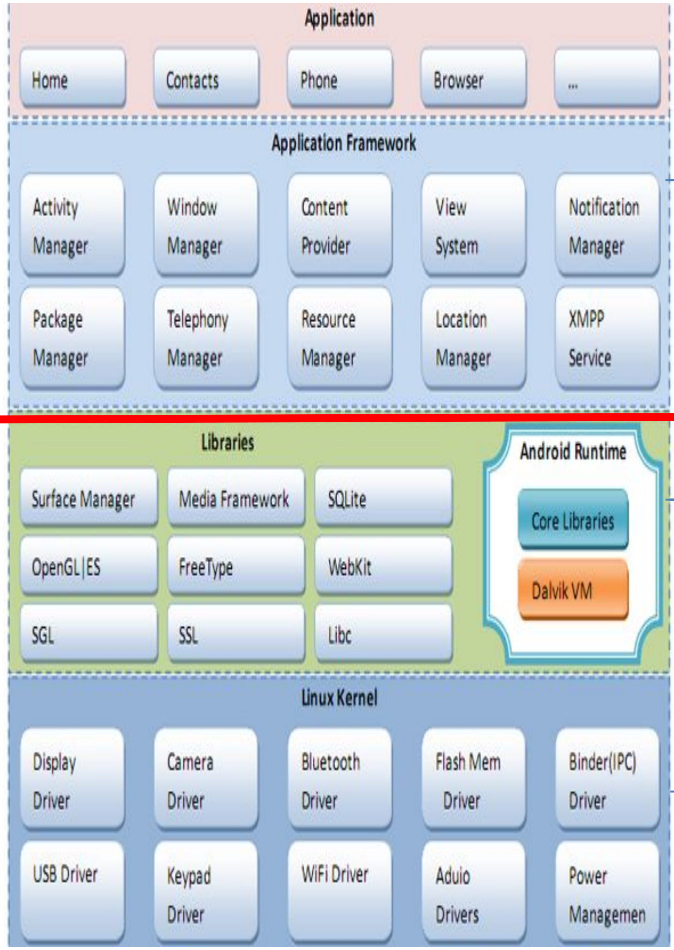




System Security Technology

安全分析工具

- OVALdi
- Metasploit
- WALA
- Soot
- Droidscope
- KINT
- KLEE
- S2E
- TEMU



系统安全机制

- 权限模型
- Binder -Intent机制
- Content Provider机制
- 应用签名验证机制
- 用户隐私保护机制

- SQLite数据库机制
- SSL机制

- 内核级整数溢出
- 内核级内存溢出
- 内核数据流分析
- 内核驱动缺陷分析

核心分析技术

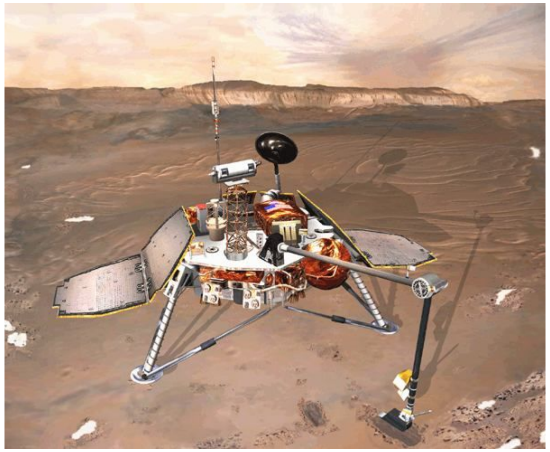
- 符号执行
- 数据流分析
- 动态执行分析
- 静态控制流分析
- Model Check

Tendency

- MultiCore
- Security
- Correction

Standard Motivating Slide for Verification

Ariane 5


Mars Polar Lander

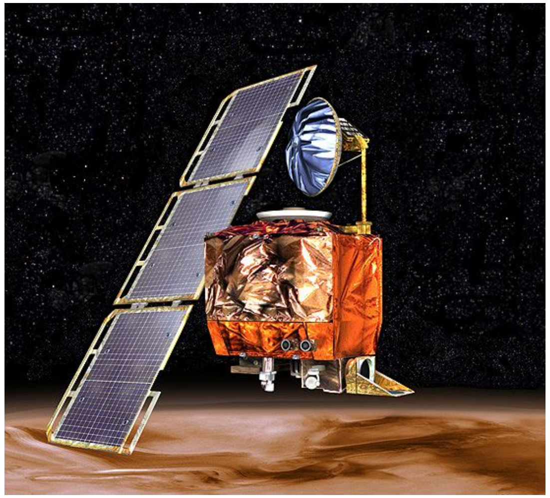
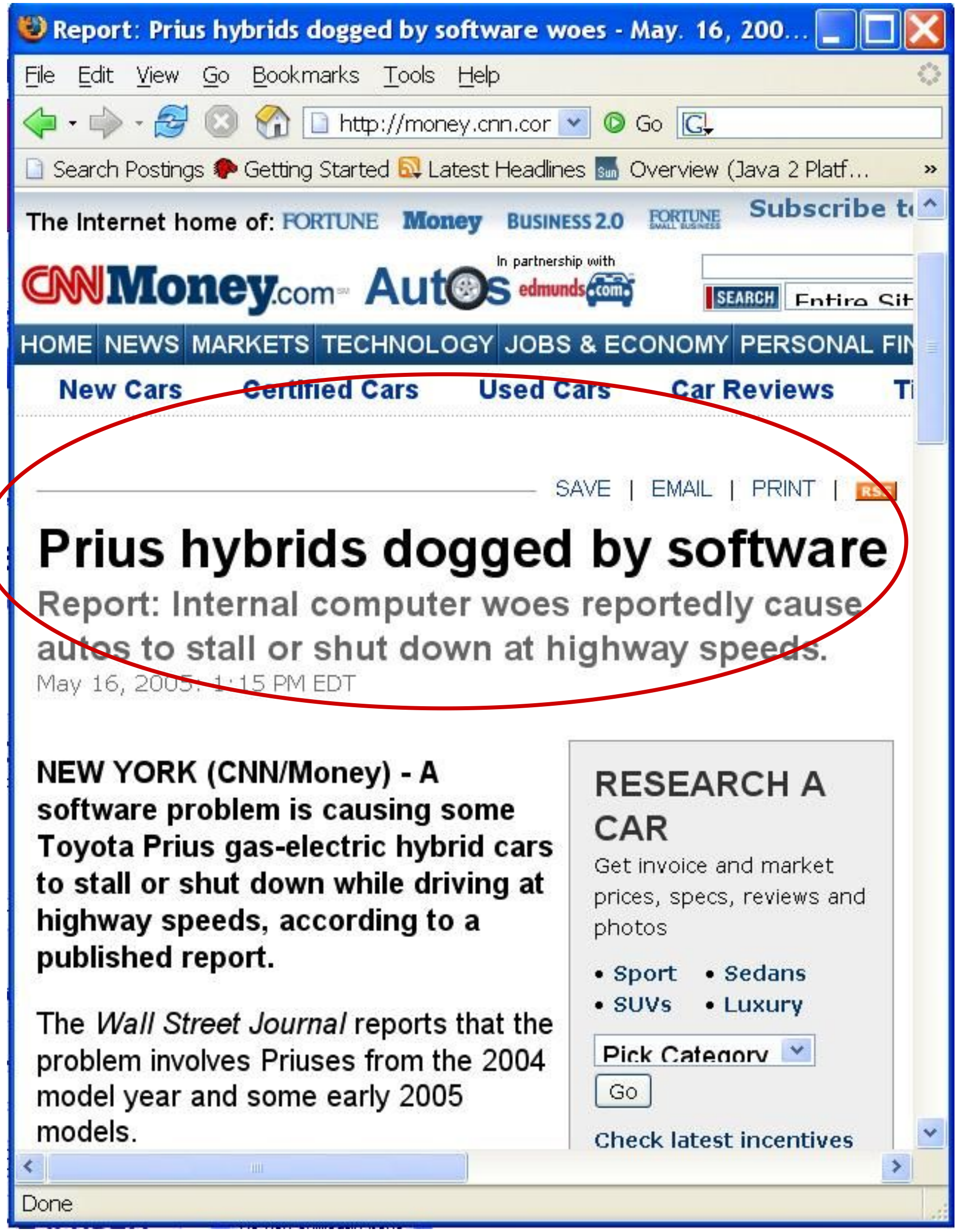

Mars climate orbiter

Photo from edmunds.com



Report: Prius hybrids dogged by software woes - May. 16, 200...

File Edit View Go Bookmarks Tools Help

http://money.cnn.cor Go

Search Postings Getting Started Latest Headlines Overview (Java 2 Platf...

The Internet home of: FORTUNE Money BUSINESS 2.0 FORTUNE SMALL BUSINESS Subscribe t

CNNMoney.com Autos In partnership with edmunds.com SEARCH Entire Sit

HOME NEWS MARKETS TECHNOLOGY JOBS & ECONOMY PERSONAL FIN

New Cars Certified Cars Used Cars Car Reviews Ti

SAVE | EMAIL | PRINT | RSS

Prius hybrids dogged by software

Report: Internal computer woes reportedly cause autos to stall or shut down at highway speeds.

May 16, 2005: 1:15 PM EDT

NEW YORK (CNN/Money) - A software problem is causing some Toyota Prius gas-electric hybrid cars to stall or shut down while driving at highway speeds, according to a published report.

The *Wall Street Journal* reports that the problem involves Priuses from the 2004 model year and some early 2005 models.

RESEARCH A CAR

Get invoice and market prices, specs, reviews and photos

- Sport
- Sedans
- SUVs
- Luxury

Pick Category

Go

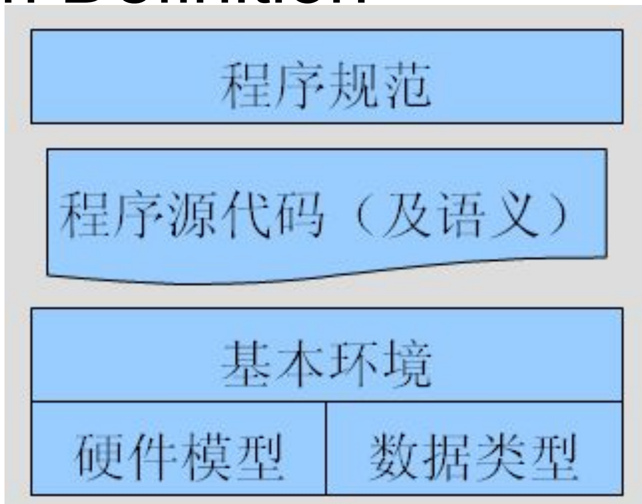
Check latest incentives

Done

Toyota recalled its 160,000 Prius cars in Oct 2005, because of bugs in the software controlling the hybrid gas-electric engine system...

Basic Idea

Certified Software: Problem Definition



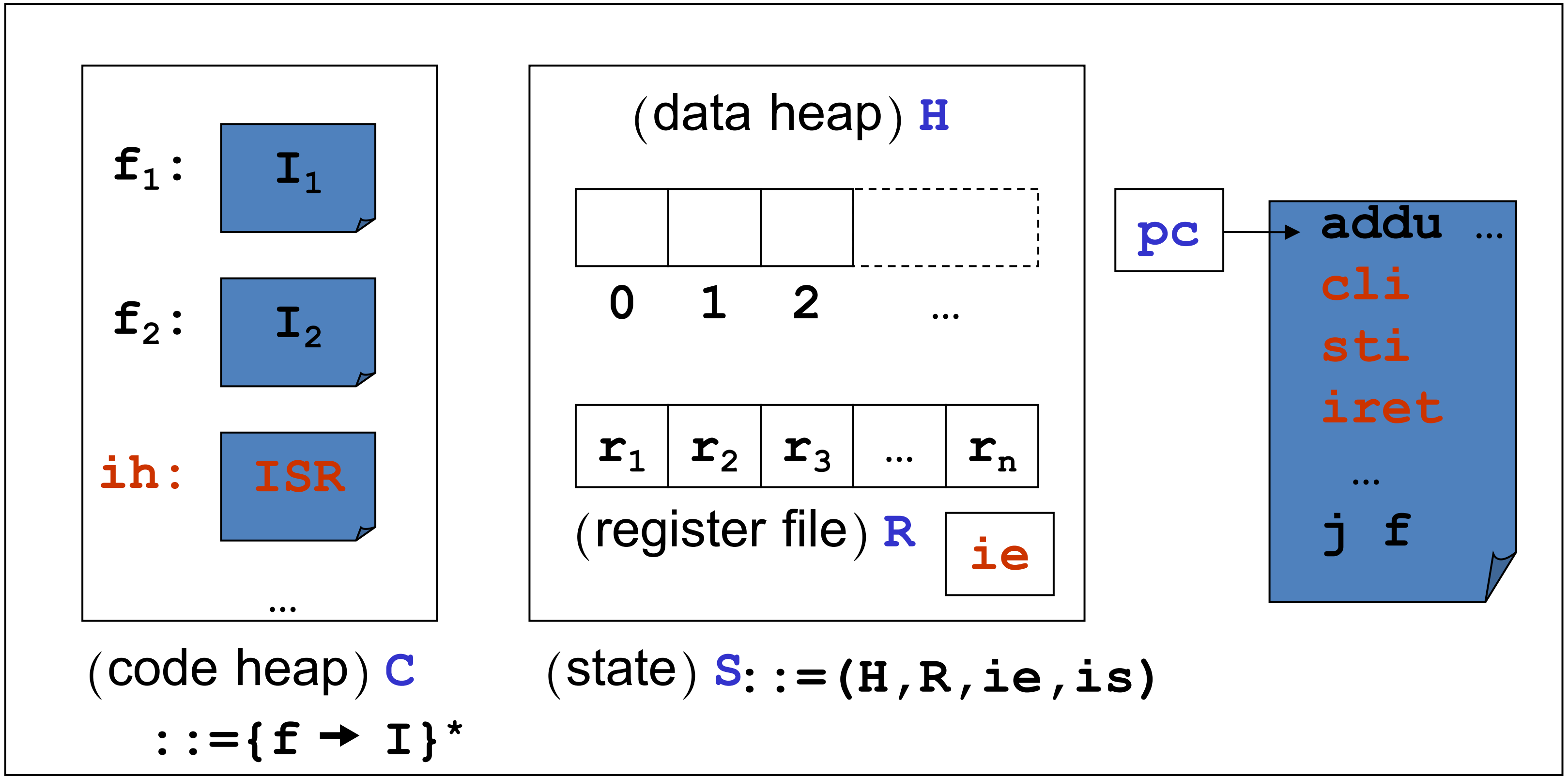
- ***Hardware***
  – processors, memory, storage, devices, …
- ***Software***
  – bootloader, device drivers, OS, runtime, applications, …
- Need a mathematical ***proof*** showing that
  as long as the ***hardware*** works, the ***software***
  always work according to its ***specification***

Challenges

- Components come from different sources
  - Manually written assembly
  - C/C++
  - Type safe languages
    - Java, C# …
- Many different features
  - Code loading
  - Control abstractions
    - jmp (goto)
    - functions
    - exceptions
    - threads
    - interrupts

AIM : The Machine

Abstract Interrupt Machine



(program) P ::=(C,S,pc)

Challenges

- Components come from different sources
  - Manually written assembly
  - C/C++
  - Type safe languages
    - Java, C# …
- Many different features
  - Memory update
    - type-preserving update
    - type-changing update
    - pointer arithmetic
  - Device drivers and I/O
  - …

Some Conclusions

- AIM machine
  - low-level
  - can implement interrupt handlers and thread libraries
- A program logic
  - following local reasoning in separation logic
  - modeling **`cli/sti, switch, block/unblock`** in terms of memory ownership transfer
  - can certify different implementation of locks and C.V.s
- mini-OS proof
  - Interrupt handler, scheduler and others

Thanks

- Some materials are from Andy Wang
- CS-502 Operating Systems from WPI
- Compiler/Program Research Group in TH
- ……