Lecture 12

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Multicore computers
Course Summary

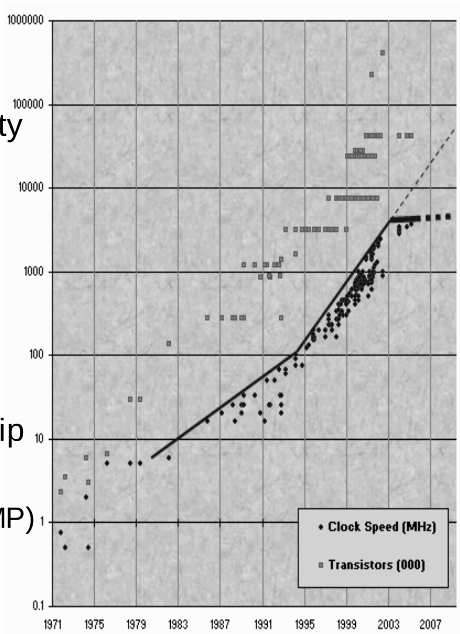
Ch 18 [Sta10]

Why Multicore?

- Current trend by processor manufacturers, because older improvements are no longer that promising
  - Clock frequency
  - Pipeline, superscalar,
  - Simultaneous multithreading, SMT (or hyperthreading)
- Enough transistors available on one chip to put two or more whole cores on the chip
  - Symmetric multiprocessor on one chip only
- But ... diminishing returns
  - More complexity requires more logic
  - Increasing chip area for coordinating and signal transfer logic

Real Problem

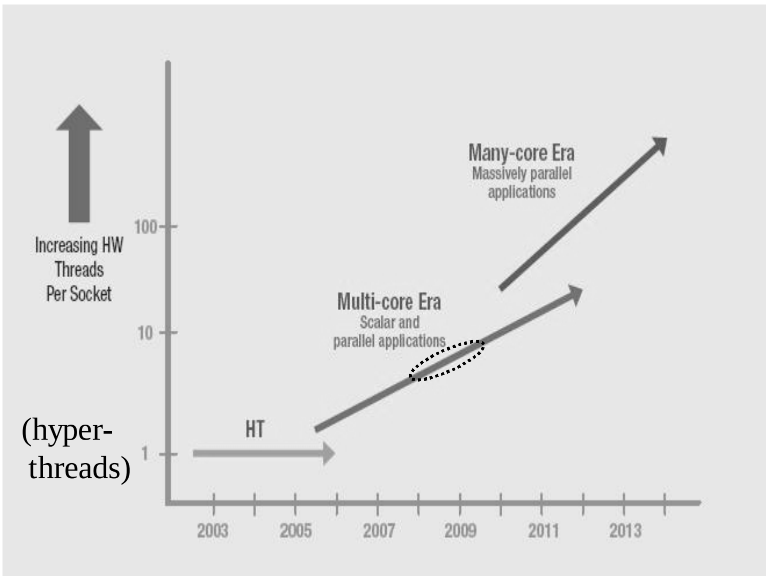
- Moore's Law: transistor density doubles every 1.5 years
  - Processor speed doubles also
  - True 1980-2003?
- Heat barrier: can not pack processors so thick
  - No more faster processors
- Now: more processors per chip
  - Multicore CPU
  - Chip-level multiprocessor (CMP)



Herb Sutter, "A Fundamental Turn Toward Concurrency in SW", Dr. Dobb's Journal, 2005.

http://www.ddj.com/web-development/184405990;jsessionid=BW05DMMAOT3ZGQSNDLPCKH0CJUNN2JVN?_requestid=1416784

Computer Organization II, Autumn 2010, Teemu Kerola 7.12.2010 3

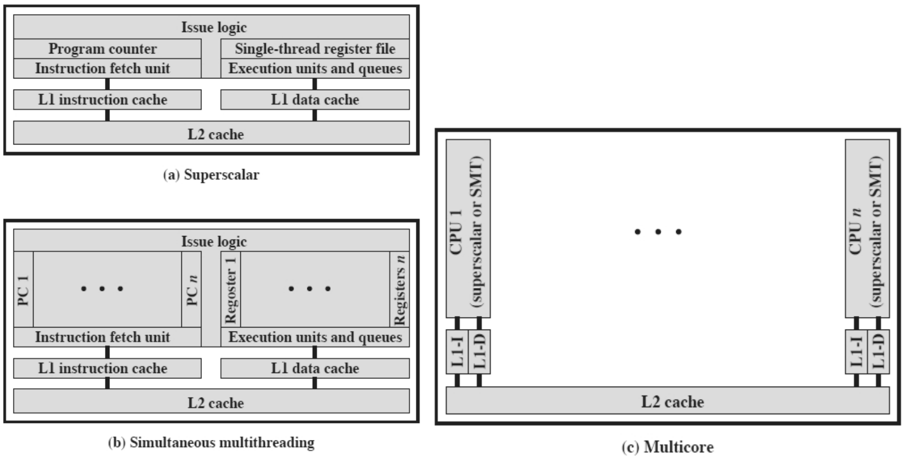


Borkar, Dubey, Kahn, et al. "Platform 2015." Intel White Paper, 2005.

http://download.intel.com/technology/computing/archinnov/platform2015/download/Platform_2015.pdf

Computer Organization II, Autumn 2010, Teemu Kerola 7.12.2010 4

What is Multicore?

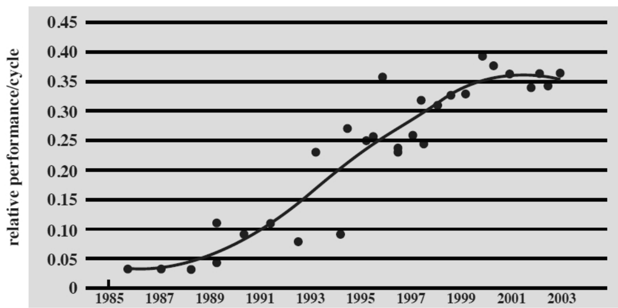


Sta10 Fig 18.1

Performance Gains

(Sta10 Fig 18.2 b)

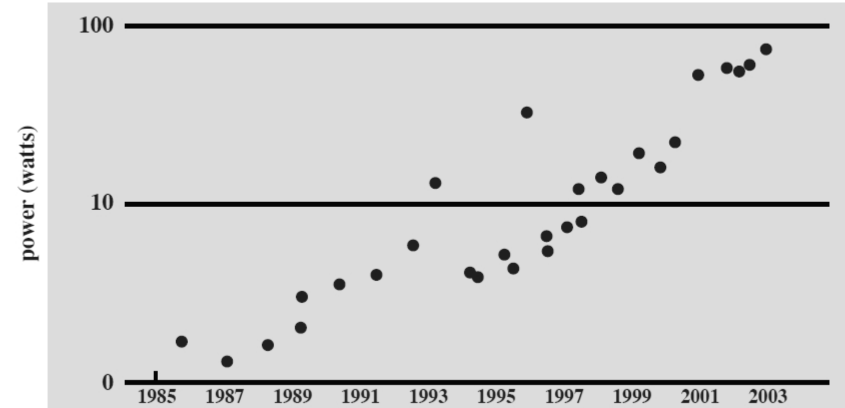
- Figure shows relative performance in Intel processors
- Dots calculated as a ratio of published SPEC CPU figures divided by clock frequency of that processor



- Late 1980’s no parallelism yet – flat curve
- Steep rise of the curve with improvements in instruction-level parallelism
  - pipelines, superscalar, SMT
- Flat again around 2000 -> limit of instruction-level parallelism reached

Power Consumption

(Sta10 Fig 18.2 c)

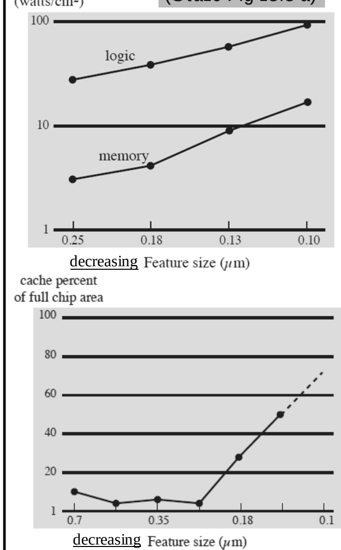


- Power consumption of Intel processors
- Notice the power requirement has grown exponentially

Computer Organization II, Autumn 2010, Teemu Kerola 7.12.2010 7

How to Use All the Transistors Available?

(Sta10 Fig 18.3 a)



- Reduce power intensity by increasing the ratio of memory transistors to logic transistors
  - Memory transistors used mainly for cache
  - Logic transistors used for everything else
- Increased complexity in logic follows Pollack's rule
  - On a single core the increased complexity of structure means that more of the logic is needed just for coordination and signal transfer logic

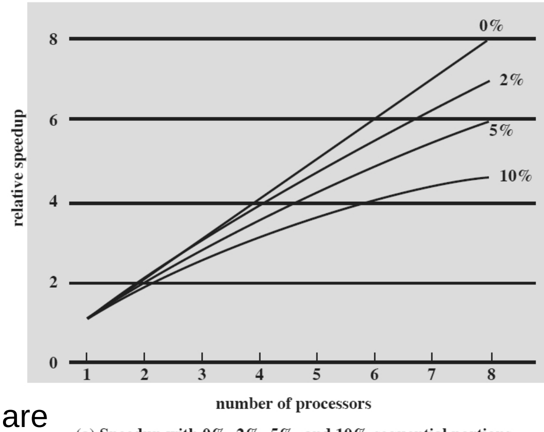
Performance increase is roughly proportional to [the] square root of [the] increase in complexity

Fred Pollack

Computer Organization II, Autumn 2010, Teemu Kerola 7.12.2010 8

Software Performance on Multicore

- Amdahl’s law: speedup is proportional to the fraction of time enhancement is used
- Thus, even a small portion of sequential code has noticeable impact with larger number of processors!
- Software improvements are not covered in this course

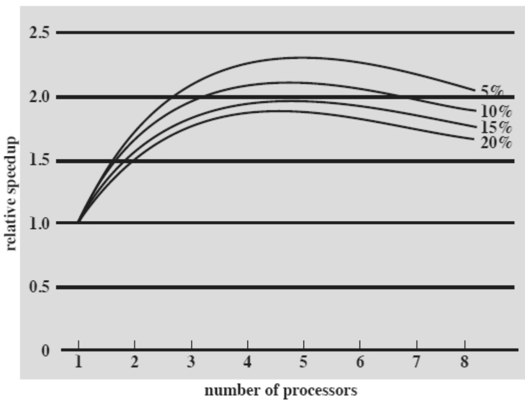


(a) Speedup with 0%, 2%, 5%, and 10% sequential portions

Sta10 Fig 18.5 a

Overhead Effect on Multicore Efficiency

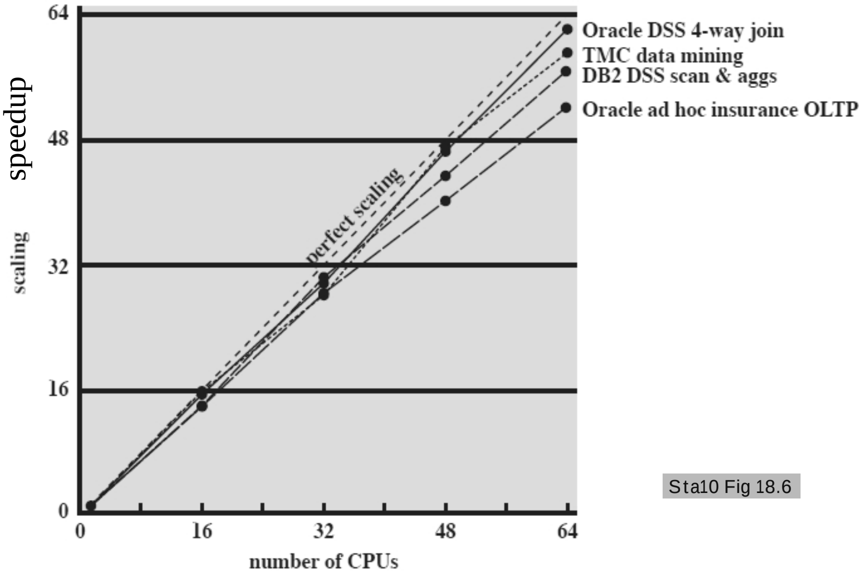
- Communication
- Distribution of work
- Cache coherence



(b) Speedup with overheads

Sta10 Fig 18.5 b

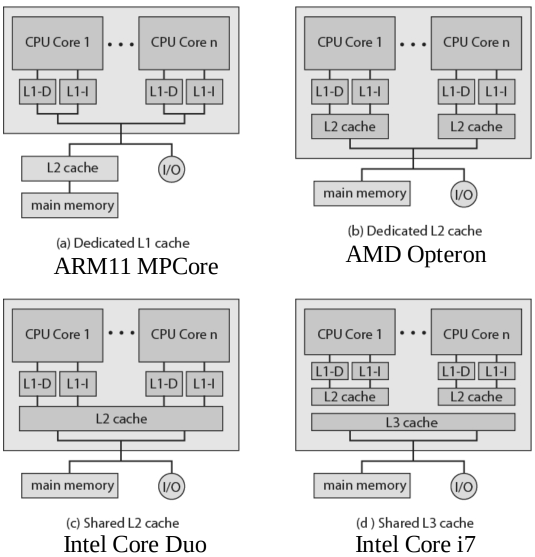
Application Specific Multicore Efficiency



Sta10 Fig 18.6

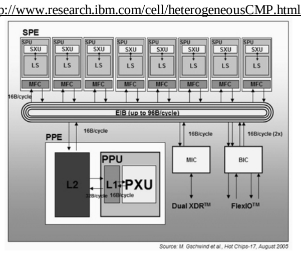
Multicore Organization

(Sta10 Fig 18.8)



- Key difference: Cache
- L1 always dedicated
  - Split for instructions and data
- L2 shared or dedicated (or mixed)
  - Active research on this issue
- L3 shared, if exists
- Cache coherence
  - Varies

http://www.research.ibm.com/cell/heterogeneousCMP.html



STI Cell

Shared L2 Cache vs. Dedicated ones

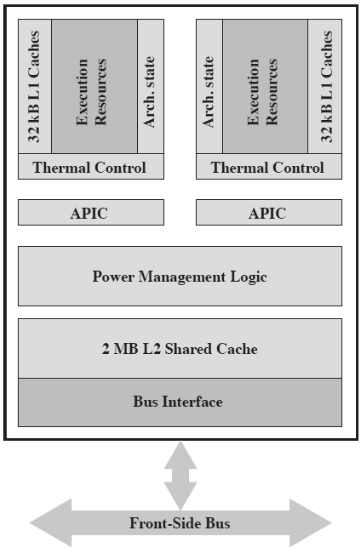
- Constructive interference
  - One core may fetch a cache line that is soon needed by another code – already available in shared cache
- Single copy
  - Shared data is not replicated, so there is just one copy of it.
- Dynamic allocation
  - The thread that has less locality needs more cache and may occupy more of the cache area
- Shared cache – no cache coherence solution needed
  - The shared data element already in the shared cache. With dedicated caches, the shared data must be invalidated from other caches before using
- Slower access
  - Larger cache area is slower to access, small dedicated cache would be faster

Computer Organization II

Intel Core Duo and Core i7

Intel Core Duo, 2006

Sta10 Fig 18.9



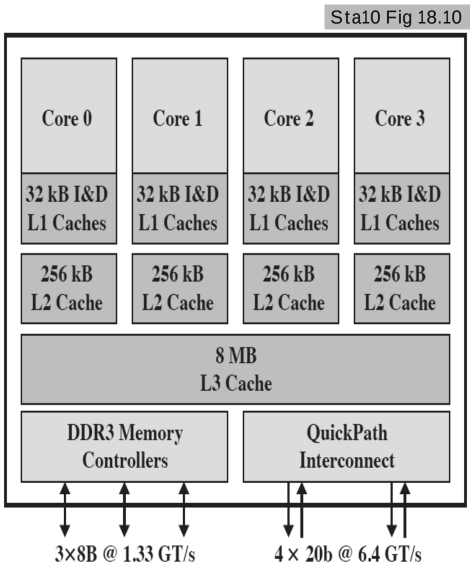
- Two x86 superscalar, shared L2 cache
  - MESI support for L1 caches
  - L2 data shared between local cores or external
- Thermal control unit per core
  - Manages chip heat dissipation
  - Maximize performance within constraints
- Advanced Programmable Interrupt Controlled (APIC)
  - Inter-process interrupts between cores
  - Routes interrupts to appropriate core
  - Includes timer so OS can interrupt core
- Power Management Logic
  - Adjusts voltage and power consumption
  - Can switch individual processor logic subsystems on and off

Computer Organization II, Autumn 2010, Teemu Kerola 7.12.2010 15

Intel Core i7 Block Diagram

- Four x86 SMT processors each with two simultaneous threads
- Dedicated L2, shared L3 cache
- Speculative pre-fetch for caches
- On chip DDR3 memory controller
  - Three 8 byte channels (192 bits) giving 32GB/s
  - No front side bus (mem access through cache)
- QuickPath Interconnection
  - Cache coherent point-to-point link
  - High speed communications between processor chips
  - 6.4G transfers per second, 16 bits per transfer
  - Dedicated bi-directional pairs
  - Total bandwidth 25.6GB/s

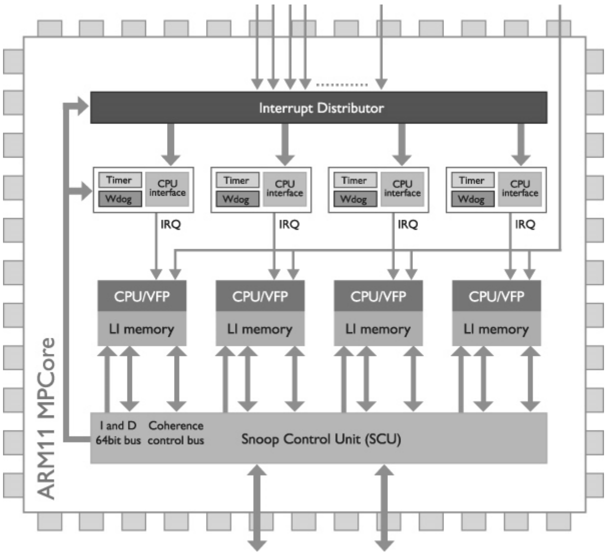
Sta10 Fig 18.10



Computer Organization II, Autumn 2010, Teemu Kerola 7.12.2010 16

Computer Organization II

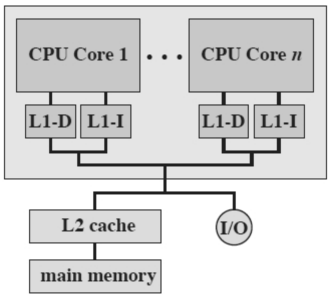
ARM11 MPCore



http://www.arm.com/products/processors/classic/arm11/arm11-mpcore.php

Computer Organization II, Autumn 2010, Teemu Kerola 7.12.2010 17

ARM11 MPCore



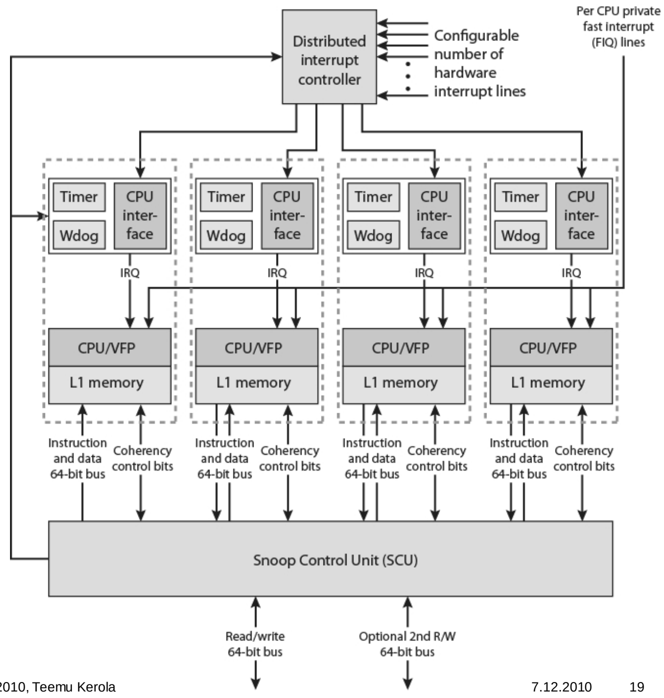
(a) Dedicated L1 cache

Sta10 Fig 18.8

- Up to 4 processors per chip
- Distributed interrupt controller
- Timer per CPU
- Watchdog
  - Warning alerts for software failures
  - Counts down from predetermined values, issues warning at zero
- CPU interface
  - Interrupt acknowledgement, masking and completion acknowledgement
- MP11 – Single ARM11 core (CPU)
- Vector floating-point (VFP) co-processor
- Dedicated split snoopy L1 cache
- Shared unified L2 cache, off-chip

Computer Organization II, Autumn 2010, Teemu Kerola 7.12.2010 18

ARM11 MPCore



Sta10 Fig 18.11

Computer Organization II, Autumn 2010, Teemu Kerola 7.12.2010 19

ARM11 MPCore Interrupt Control

- Distributed Interrupt Controller (DIC)
  - collates interrupts from many sources
  - Masking, prioritization
  - Distribution to target MP11 CPUs
  - Status tracking (Interrupt states: pending, active, inactive)
  - Software interrupt generation
- Number of interrupts independent of MP11 CPU design
- Accessed by CPUs via private interface through SCU
- Can route interrupts to single or multiple CPUs
  - OS can generate interrupts: all-but-self, self, or specific CPU
- Provides inter-process communication (16 intr. ids)
  - Thread on one CPU can cause activity by thread on another CPU

Computer Organization II, Autumn 2010, Teemu Kerola 7.12.2010 20

ARM11 MPCore L1 Cache Coherency

- MESI
- Direct Data Intervention (DDI)
  - Copy (clean) cache lines directly between caches
- Duplicated tag RAM (tag fields)
  - Copies of tag RAM in many CPU's
  - Cache knows who has the data needed
- Migratory lines
  - Copy dirty cache lines directly to other caches
    - No need to go to L2 cache, or to memory
  - Modified MESI protocol

Course Summary

Course Structure

- Week 1
  - Overview (Ch 1 – 8)
  - Digital logic (Ch 20)
  - Bus (Ch 3)
- Week 2
  - Memory, Cache (Ch 4, 5)
  - Virtual memory(Ch 8)
- Week 3
  - Computer arithmetic (Ch 9)
  - Instruction set (Ch 10, 11)
- Week 4
  - CPU struc.& func. (Ch 12)
  - RISC-architecture (Ch 13)
- Week 5
  - Instruction-level parallelism, Superscalar Processor (Ch 14)
  - Control Unit (Ch 15-16)
- Week 6
  - Parallel Processing (Ch 17)
  - Multicore (Ch 18)
  - **Summary**

Computer Organization II, Autumn 2010, Teemu Kerola 7.12.2010 23

Course Exam Tue 14.12.2010 9-12 (A111)

- 2,5 hours – three or four questions
  - Questions are in English
  - You may answer in English, Finnish, or Swedish
- Question try to assess your deeper understanding of relevant topics, not superficial facts
  - More of applying what you have learned
  - Some of understanding relevant concepts
  - Less of rephrasing topics from text book or lectures
  - No details on example architectures
- You can write on all answers on the same paper using pencil or pen
  - No need to write answer to each question to separate sheet
- There is <u>no need for a calculator</u>, but a simple one is allowed
  - If there is math needed, you can just write the formula and you do not need to write the result number without a calculator

Computer Organization II, Autumn 2010, Teemu Kerola 7.12.2010 24

For the Exam

- Go through the exercises
  - If you did all homeworks and understand them well, you should do fine in the exam
- Read the book and lecture slides
  - If there is nothing on the slides about the subsection, then there very probably is not a question in the exam
- The review questions in the slides are good hints!
- Old exams are in web
  - Many exams only in Finnish
  - See https://www.cs.helsinki.fi/courses/581365/2010/s/k/1
    - ”Basic Information” sub-page (tab)
    - ”Kerola’s CO-II home page”, and ”Previous Exams” there
  - Exam questions have high temporal locality!

Digital logic (Ch 20)

- What is the problem and how is it solved?
- Boolean algebra, gates and flip-flops
- Basic ideas on optimization, no Carnaugh maps
- Circuit description with Boolean tables, gates, and graphs
- Flip-flops and basic circuits, basic functionality
  - Understand, how S-R flip-flop works
- Combination circuits vs. sequential circuits
- How to implement memory?
- How to implement functions?
  - How to implement 32-bit add?

Bus (Ch 3)

- What is the problem and how is it solved?
- Instruction cycle, interrupts
- Bus characteristics
  - Speed, width, asynch/synch timing,
  - Signaling, centralized/distributed arbitration,
  - Events or transactions
- PCI
  - Arbitration, read & write sequences
- Can read and explain timing diagrams

Cache (Ch 4)

- What is the problem? How is it solved?
- Principle of locality,temporal & spatial locality
- Design features
  - Size, line size, split/unified, levels (L1, L2, L3)
  - Mapping: direct mapping, fully-associative, set-associative
  - Replacement policy
  - Write policy: write-through, write-back, write-once
- Cache coherency problem for multiprocessors

Main Memory (Ch 5)

- Basic ideas, no details
- DRAM implementation principles
  - Memory address split row and column access select fields
- How to build larger memory from smaller chips

Memory Management (Ch 8)

- Focus on virtual memory
  - What problem is solved? How is it solved?
  - Solution is based on locality
  - Solve protection problems at the same time?
- Virtual memory organization
  - page table, inverted page table, segment table,
  - hierarchical tables
  - Disk organization to support VM
- Address translation,
  - What is the problem, what is the solution, how is it done?
  - TLB, how does it work, how is it implemented?
- TLB and cache, how do they work together
  - How do you locate referenced data (in cache or in memory)

Computer Arithmetic (Ch 9)

- What is the problem and how is it solved?
- Integers
  - Representation
  - Add & subtract, multiply, divide
  - Booth's algorithm for multiplication
- Floating-point
  - IEEE representation, unnormalized, NaN, ∞
  - Principles of add, sub, mul, div – overflows/underflows
  - Accuracy
    - In representation
    - In computation
    - Loss of accuracy in certain math ops

Instruction Sets (Ch 10, 11)

- What is the problem and how is it solved?
- Characteristics
  - Data types, register sets
  - Addressing modes
- Architecture types
  - Accumulator, stack, register, load-store
- Instruction formats
  - Pentium cisc vs. Arm risc
  - Can explain basic differences
  - No need to study details

CPU Structure and Function (Ch 12)

- What is the problem and how is it solved?
- Structural elements: regs, internal regs, pws
- Pipelined implementation of fetch-exec cycle
  - What is the problem and how is it solved?
  - Performance gains: when and how much?
  - Hazards & dependencies
    - Types: structural, control, data
    - Solution methods: bubbles, compiler, more HW
  - How to solve RAW data dependency problems?
    - Bubble (hw), NOP (sw bubble), instr order (sw), by-pass circuits (hw)
  - How to solve control dependencies?
    - Clear pipeline (hw), delay slots (sw), mult. conditional instr. streams
    - Prefetch target, loop buffer
    - Static and dynamic branch prediction, branch history table

RISC (Ch 13)

- What is the problem and how is it solved?
- What is CISC, what is RISC?
  - RISC vs CISC
- RISC features
  - Lots of regs, few data types,
  - Few operands and memory addressing types
  - Simple instructions optimized for pipeline
  - Load/Store architecture
- Register files
  - What is the problem and how is it solved?
  - Registers windows, register optimization
- Register allocation problem
  - What is the problem and how is it solved (graph coloring)?

Superscalar (Ch 14)

- What is the problem and how is it solved?
- Implementation strategies
  - In-order or out-of-order issue
  - In-order or out-of-order complete
  - Instruction selection window, window of execution
- Name dependencies
  - What is the problem and how is it solved?
  - New dependency types to worry about: WAR, RAW
  - Register renaming
- Hyperthreading or multithreading
  - What is the problem and how is it solved?
  - Use larger register set to better utilize pipelines

Control-Unit (Ch 15, 16)

- What is the problem and how is it solved?
- Micro-ops
  - Micro-op sequences in different phases of the execution cycle
- How control signals make things happen?
  - Control signal state machine
- Hardwired control
  - Direct implementation of control state machine
  - Requires lots of optimiztion to reduce state space
- Microprogrammed control
  - Structure: control memory, control address, control buffer
  - Horizontal or vertical (functional & resource encoding)
  - Sequencing, i.e., which microinstruction next?

Parallel Processing (Ch 17)

- What is the problem and how is it solved?
- Classification, SIMD, SMP, etc
- Cache coherency
  - Snoopy-cache
  - MESI
- Clusters
  - NUMA, CC-NUMA
- Vector computation

Multicore (Ch 18)

- What is the problem and how is it solved?
- Multicore vs. SMP
- Different multicore organizations
- Multicore with CC-NUMA

-- The End --

http://researchweb.watson.ibm.com/journal/rd/494/kahle2.gif

STI Cell Power processor element
(a) major units and
(b) pipeline