CS252
Graduate Computer Architecture
Lecture 1

Introduction

January 22, 2002
Prof. David E Culler
Computer Science 252
Spring 2002

Outline

- Why Take CS252?
- Fundamental Abstractions & Concepts
- Instruction Set Architecture & Organization
- Administrivia
- Pipelined Instruction Processing
- Performance
- The Memory Abstraction
- Summary

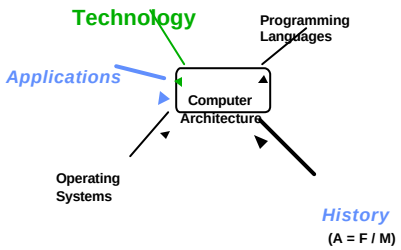
Why take CS252?

- To design the next great instruction set?...well...
  - instruction set architecture has largely converged
  - especially in the desktop / server / laptop space
  - dictated by powerful market forces
- Tremendous organizational innovation relative to established ISA abstractions
- Many New instruction sets or equivalent
  - embedded space, controllers, specialized devices, ...
- Design, analysis, implementation concepts vital to all aspects of EE & CS
  - systems, PL, theory, circuit design, VLSI, comm.
- Equip you with an intellectual toolbox for dealing with a host of systems design challenges

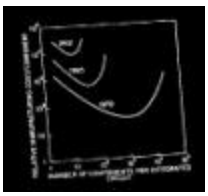
Example Hot Developments ca. 2002

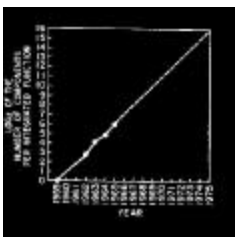
- Manipulating the instruction set abstraction
  - itanium: translate ISA64 -> micro-op sequences
  - transmeta: continuous dynamic translation of IA32
  - tinsilica: synthesize the ISA from the application
  - reconfigurable HW
- Virtualization
  - vmware: emulate full virtual machine
  - JIT: compile to abstract virtual machine, dynamically compile to host
- Parallelism
  - wide issue, dynamic instruction scheduling, EPIC
  - multithreading (SMT)
  - chip multiprocessors
- Communication
  - network processors, network interfaces
- Exotic explorations
  - nanotechnology, quantum computing

Forces on Computer Architecture

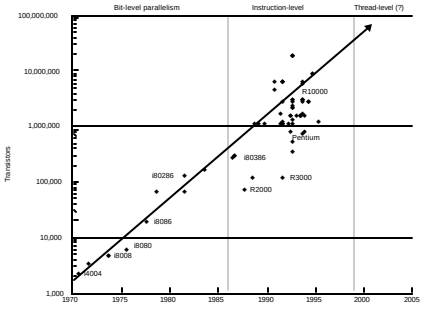


Amazing Underlying Technology Change





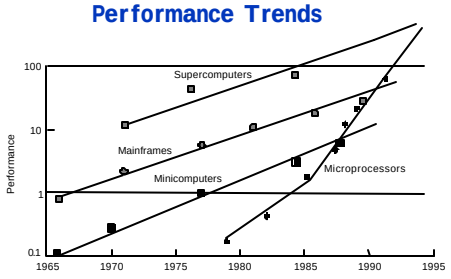
A take on Moore's Law



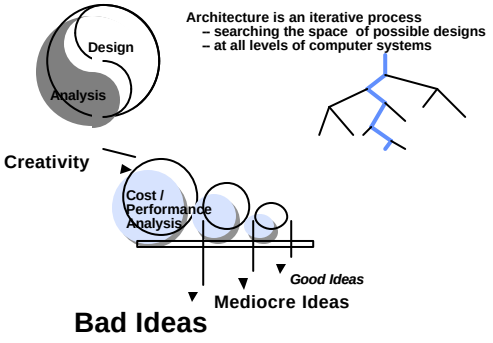
Technology Trends

- Clock Rate: ~30% per year
- Transistor Density: ~35%
- Chip Area: ~15%
- Transistors per chip: ~55%
- Total Performance Capability: ~100%
- by the time you graduate...
  - 3x clock rate (3-4 GHz)
  - 10x transistor count (1 Billion transistors)
  - 30x raw capability
- plus 16x dram density, 32x disk density

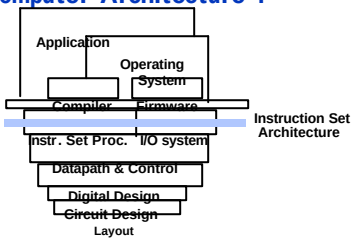
Performance Trends



Measurement and Evaluation



What is "Computer Architecture"?



- Coordination of many *levels of abstraction*
- Under a rapidly changing set of forces
- Design, Measurement, *and* Evaluation

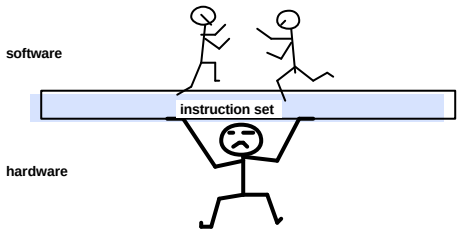
Coping with CS 252

- Students with too varied background?
  - In past, CS grad students took written prelim exams on undergraduate material in hardware, software, and theory
  - 1st 5 weeks reviewed background, helped 252, 262, 270
  - Prelims were dropped => some unprepared for CS 252?
- In class exam on Tues Jan. 29 (30 mins)
  - Doesn't affect grade, only admission into class
  - 2 grades: Admitted or audit/take CS 152 1st
  - Improve your experience if recapture common background
- Review: Chapters 1, CS 152 home page, maybe "Computer Organization and Design (COD)2/e"
  - Chapters 1 to 8 of COD if never took prerequisite
  - If took a class, be sure COD Chapters 2, 6, 7 are familiar
  - Copies in Bechtel Library on 2-hour reserve
- FAST review this week of basic concepts

Review of Fundamental Concepts

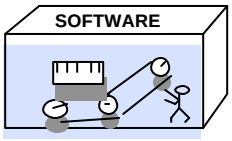
- Instruction Set Architecture
- Machine Organization
- Instruction Execution Cycle
- Pipelining
- Memory
- Bus (Peripheral Hierarchy)
- Performance Iron Triangle

The Instruction Set: a Critical Interface



Instruction Set Architecture

... the attributes of a [computing] system as seen by the programmer, *i.e.* the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls the logic design, and the physical implementation.
– Amdahl, Blaaw, and Brooks, 1964



- Organization of Programmable Storage
- Data Types & Data Structures: Encodings & Representations
- Instruction Formats
- Instruction (or Operation Code) Set
- Modes of Addressing and Accessing Data Items and Instructions
- Exceptional Conditions

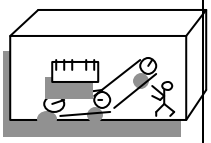
Organization

- Capabilities & Performance Characteristics of Principal Functional Units
  - (e.g., Registers, ALU, Shifters, Logic Units, ...)
- Ways in which these components are interconnected
- Information flows between components
- Logic and means by which such information flow is controlled.
- Choreography of FUs to realize the ISA
- Register Transfer Level (RTL) Description

Logic Designer's View

ISA Level

FUs & Interconnect



Review: MIPS R3000 (core)

Programmable storage
2^32 x bytes
31 x 32-bit GPRs (R0=0)
32 x 32-bit FP regs (paired DP)
HI, LO, PC

Data types ?

Format ?

Addressing Modes?

Arithmetic logical

Add, AddU, Sub, SubU, And, Or, Xor, Nor, SLT, SLTU,
AddI, AddIU, SLTI, SLTIU, AndI, OrI, XorI, *LUI*
SLL, SRL, SRA, SLLV, SRLV, SRAV

Memory Access

LB, LBU, LH, LHU, LW, LWL,LWR
SB, SH, SW, SWL, SWR

Control

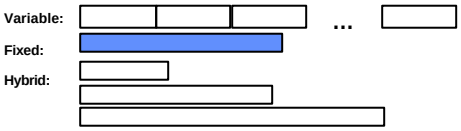
32-bit instructions on word boundary

J, JAL, JR, JALR
BEq, BNE, BLEZ,BGTZ,BLTZ,BGEZ,BLTZAL,BGEZAL

Review: Basic ISA Classes

Accumulator:		
1 address	add A	acc ← acc + mem[A]
1+x address	addx A	acc ← acc + mem[A + x]
Stack:		
0 address	add	tos ← tos + next
General Purpose Register:		
2 address	add A B	EA(A) ← EA(A) + EA(B)
3 address	add A B C	EA(A) ← EA(B) + EA(C)
Load/Store:		
3 address	add Ra Rb Rc	Ra ← Rb + Rc
	load Ra Rb	Ra ← mem[Rb]
	store Ra Rb	mem[Rb] ← Ra

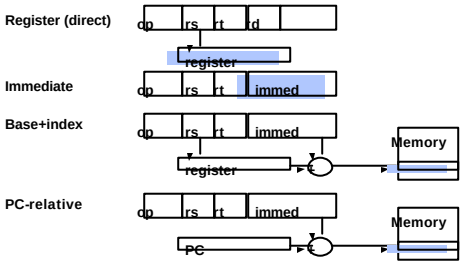
Instruction Formats



- Addressing modes
  - –each operand requires addess specifier => variable format
- code size => variable length instructions
- performance => fixed length instructions
  - –simple decoding, predictable operations
- With load/store instruction arch, only one memory address and few addressing modes
- => simple format, address mode given by opcode

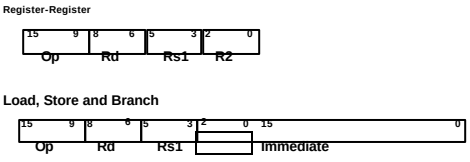
MIPS Addressing Modes & Formats

- Simple addressing modes
- All instructions 32 bits wide



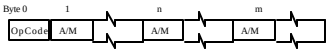
- Register Indirect?

Cray-1: the original RISC



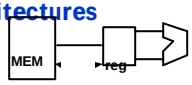
VAX-11: the canonical CISC

Variable format, 2 and 3 address instruction



- Rich set of orthogonal address modes
  - immediate, offset, indexed, autoinc/dec, indirect, indirect+offset
  - applied to any operand
- Simple and complex instructions
  - synchronization instructions
  - data structure operations (queues)
  - polynomial evaluation

Review: Load/Store Architectures

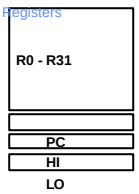


- 3 address GPR
- Register to register arithmetic
- Load and store with simple addressing modes (reg + immediate)
- Simple conditionals
  - compare ops + branch z
  - compare&branch
  - condition code + branch on condition
- Simple fixed-format encoding

- Substantial increase in instructions
- Decrease in data BW (due to many registers)
- Even more significant decrease in CPI (pipelining)
- Cycle time, Real estate, Design time, Design complexity

MIPS R3000 ISA (Summary)

- Instruction Categories
  - Load/Store
  - Computational
  - Jump and Branch
  - Floating Point
    - » coprocessor
  - Memory Management
  - Special



3 Instruction Formats: all 32 bits wide

CS 252 Administrivia

- TA: Jason Hill, jhill@cs.berkeley.edu
- All assignments, lectures via WWW page: http://www.cs.berkeley.edu/~culler/252S02/
- 2 Quizzes: 3/21 and ~14th week (maybe take home)
- Text:
  - Pages of 3rd edition of Computer Architecture: A Quantitative Approach
    - » available from Cindy Palwick (MWF) or Jeanette Cook ($30 1-5)
  - "Readings in Computer Architecture" by Hill et al
- In class, prereq quiz 1/29 last 30 minutes
  - Improve 252 experience if recapture common background
  - Bring 1 sheet of paper with notes on both sides
  - Doesn't affect grade, only admission into class
  - 2 grades: Admitted or audit/take CS 152 1st
  - Review: Chapters 1, CS 152 home page, maybe "Computer Organization and Design (COD)2/e"
  - If did take a class, be sure COD Chapters 2, 5, 6, 7 are familiar
  - Copies in Bechtel Library on 2-hour reserve

Research Paper Reading

- As graduate students, you are now researchers.
- Most information of importance to you will be in research papers.
- Ability to rapidly scan and understand research papers is key to your success.
- So: 1-2 paper / week in this course
  - Quick 1 paragraph summaries will be due in class
  - Important supplement to book.
  - Will discuss papers in class
- Papers "Readings in Computer Architecture" or online
- Think about methodology and approach

First Assignment (due Tu 2/5)

- Read
  - Amdahl, Blaauw, and Brooks, Architecture of the IBM System/360
  - Lonergan and King, B5000
- Four each prepare for in-class debate 1/29
- rest write analysis of the debate
- Read "Programming the EDSAC", Cambell-Kelly
  - write subroutine sum(A,n) to sum an array A of n numbers
  - write recursive fact(n) = if n==1 then 1 else n*fact(n-1)

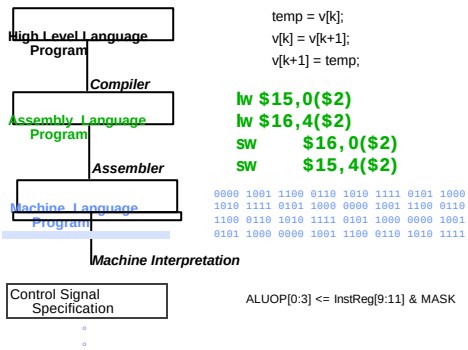
Grading

- 10% Homeworks (work in pairs)
- 40% Examinations (2 Quizzes)
- 40% Research Project (work in pairs)
  - Draft of Conference Quality Paper
  - Transition from undergrad to grad student
  - Berkeley wants you to succeed, but you need to show initiative
  - pick topic
  - meet 3 times with faculty/TA to see progress
  - give oral presentation
  - give poster session
  - written report like conference paper
  - 3 weeks work full time for 2 people (over more weeks)
  - Opportunity to do "research in the small" to help make transition from good student to research colleague
- 10% Class Participation

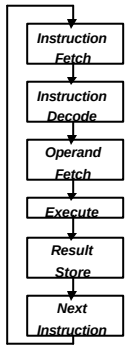
Course Profile

- 3 weeks: basic concepts
  - instruction processing, storage
- 3 weeks: hot areas
  - latency tolerance, low power, embedded design, network processors, NIs, virtualization
- Proposals due
- 2 weeks: advanced microprocessor design
- Quiz & Spring Break
- 3 weeks: Parallelism (MPs, CMPs, Networks)
- 2 weeks: Methodology / Analysis / Theory
- 1 weeks: Topics: nano, quantum
- 1 week: Project Presentations

Levels of Representation (61C Review)



Execution Cycle



Obtain instruction from program storage

Determine required actions and instruction size

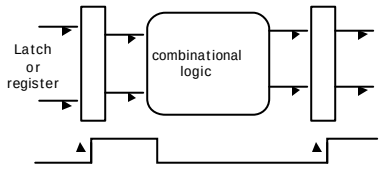
Locate and obtain operand data

Compute result value or status

Deposit results in storage for later use

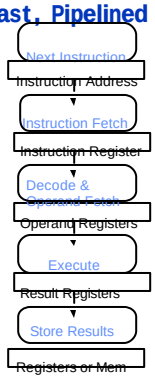
Determine successor instruction

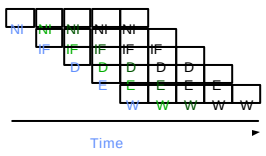
What's a Clock Cycle?



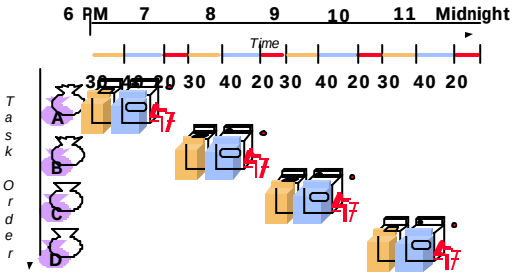
- Old days: 10 levels of gates
- Today: determined by numerous time-of-flight issues + gate delays
  - clock propagation, wire lengths, drivers

Fast, Pipelined Instruction Interpretation



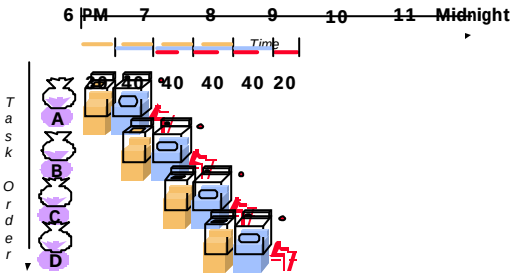


Sequential Laundry



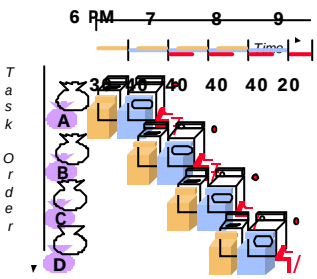
- Sequential laundry takes 6 hours for 4 loads
- If they learned pipelining, how long would laundry take?

Pipelined Laundry Start work ASAP



- Pipelined laundry takes 3.5 hours for 4 loads

Pipelining Lessons



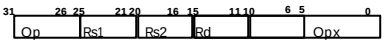
- Pipelining doesn't help latency of single task, it helps throughput of entire workload
- Pipeline rate limited by slowest pipeline stage
- Multiple tasks operating simultaneously
- Potential speedup = Number pipe stages
- Unbalanced lengths of pipe stages reduces speedup
- Time to "fill" pipeline and time to "drain" it reduces speedup

Instruction Pipelining

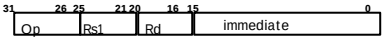
- Execute billions of instructions, so *throughput* is what matters
  - except when?
- What is desirable in instruction sets for pipelining?
  - Variable length instructions vs. all instructions same length?
  - Memory operands part of any operation vs. memory operands only in loads or stores?
  - Register operand many places in instruction format vs. registers located in same place?

Example: MIPS (Note register location)

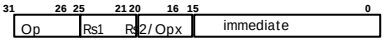
Register - Register



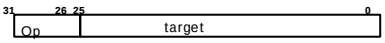
Register - Immediate



Branch

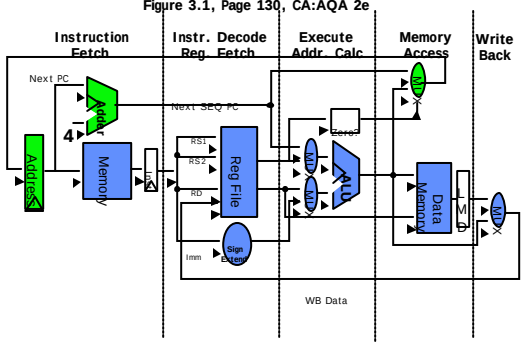


Jump / Call



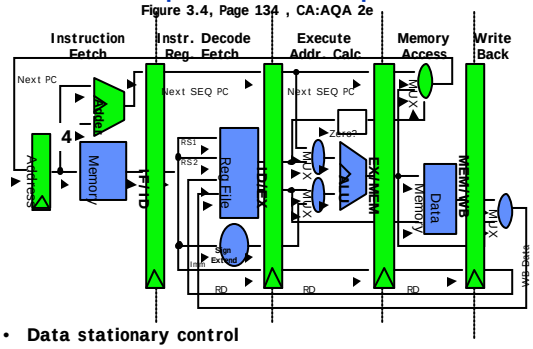
5 Steps of MIPS Datapath

Figure 3.1, Page 130, CA:AQA 2e



5 Steps of MIPS Datapath

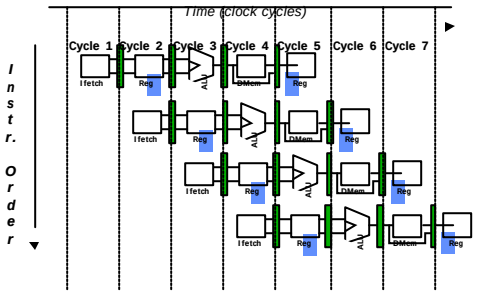
Figure 3.4, Page 134 , CA:AQA 2e



- Data stationary control
  - local decode for each instruction phase / pipeline stage

Visualizing Pipelining

Figure 3.3, Page 133 , CA:AQA 2e



Its Not That Easy for Computers

- Limits to pipelining: Hazards prevent next instruction from executing during its designated clock cycle
  - Structural hazards: HW cannot support this combination of instructions (single person to fold and put clothes away)
  - Data hazards: Instruction depends on result of prior instruction still in the pipeline (missing sock)
  - Control hazards: Caused by delay between the fetching of instructions and decisions about changes in control flow (branches and jumps).

Review of Performance

Which is faster?

Plane	DC to Paris	Speed	Passengers	Throughput (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
BAD/Sud Concorde	3 hours	1350 mph	132	178,200

- Time to run the task (ExTime)
  - Execution time, response time, latency
- Tasks per day, hour, week, sec, ns … (Performance)
  - Throughput, bandwidth

Definitions

- Performance is in units of things per sec
  - bigger is better
- If we are primarily concerned with response time
  - $\text{performance}(x) = \frac{1}{\text{execution\_time}(x)}$

" X is n times faster than Y" means

$$n = \frac{\text{Performance}(X)}{\text{Performance}(Y)} = \frac{\text{Execution\_time}(Y)}{\text{Execution\_time}(Y)}$$

Computer Performance

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

	Inst Count	CPI	Clock Rate
Program	X		
Compiler	X	(X)	
Inst. Set.	X	X	
Organization		X	X
Technology			X

Cycles Per Instruction (Throughput)

"Average Cycles per Instruction"

CPI = (CPU Time * Clock Rate) / Instruction Count
= Cycles / Instruction Count

$$\text{CPU time} = \text{Cycle Time} \times \sum_{j=1}^{n} \text{CPI}_j \times I_j$$

$$\text{CPI} = \sum_{j=1}^{n} \text{CPI}_j \times F_j \quad \text{where } F_j = \frac{I_j}{\text{Instruction Count}}$$

"Instruction Frequency"

Example: Calculating CPI bottom up

Base Machine (Reg / Reg)

Op	Freq	Cycles	CPI(i)	(% Time)
ALU	50%	1	.5	(33%)
Load	20%	2	.4	(27%)
Store	10%	2	.2	(13%)
Branch	~~20%~~	2	.4	(27%)
			1.5	

Typical Mix of instruction types in program

Example: Branch Stall Impact

- Assume CPI = 1.0 ignoring branches (ideal)
- Assume solution was stalling for 3 cycles
- If 30% branch, Stall 3 cycles on 30%

Op	Freq	Cycles	CPI(i)	(% Time)
Other	70%	1	.7	(37%)
Branch	30%	4	1.2	(63%)

- => new CPI = 1.9
- New machine is 1/1.9 = 0.52 times faster (i.e. slow!)

Speed Up Equation for Pipelining

$$CPI_{pipelined} = \text{Ideal CPI} + \text{Average Stall cycles per Inst}$$

$$\text{Speedup} = \frac{\text{Ideal CPI} \times \text{Pipeline depth}}{\text{Ideal CPI} + \text{Pipeline stall CPI}} \times \frac{\text{Cycle Time}_{unpipelined}}{\text{Cycle Time}_{pipelined}}$$

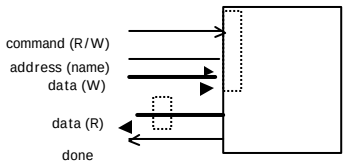
For simple RISC pipeline, CPI = 1:

$$\text{Speedup} = \frac{\text{Pipeline depth}}{1 + \text{Pipeline stall CPI}} \times \frac{\text{Cycle Time}_{unpipelined}}{\text{Cycle Time}_{pipelined}}$$

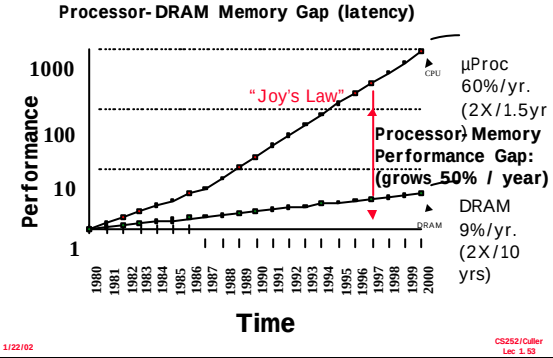
Now, Review of Memory Hierarchy

The Memory Abstraction

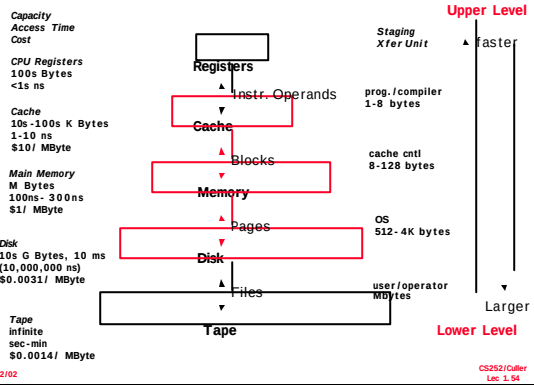
- Association of <name, value> pairs
  - typically named as byte addresses
  - often values aligned on multiples of size
- Sequence of Reads and Writes
- Write binds a value to an address
- Read of addr returns most recently written value bound to that address



Recap: Who Cares About the Memory Hierarchy?



Levels of the Memory Hierarchy

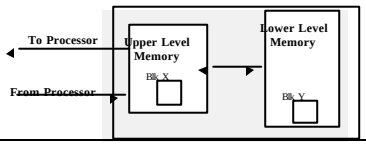


The Principle of Locality

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon (e.g., straightline code, array access)
- Last 15 years, HW (hardware) relied on locality for speed

Memory Hierarchy: Terminology

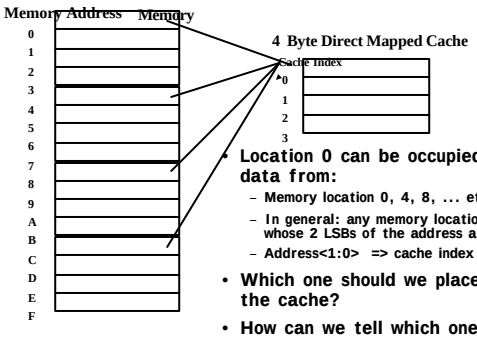
- Hit: data appears in some block in the upper level (example: Block X)
  - Hit Rate: the fraction of memory access found in the upper level
  - Hit Time: Time to access the upper level which consists of RAM access time + Time to determine hit/miss
- Miss: data needs to be retrieve from a block in the lower level (Block Y)
  - Miss Rate = 1 - (Hit Rate)
  - Miss Penalty: Time to replace a block in the upper level + Time to deliver the block the processor
- Hit Time << Miss Penalty (500 instructions on 21264!)



Cache Measures

- *Hit rate*: fraction found in that level
  - So high that usually talk about *Miss rate*
  - Miss rate fallacy: as MIPS to CPU performance, miss rate to average memory access time in memory
- Average memory-access time = Hit time + Miss rate x Miss penalty (ns or clocks)
- *Miss penalty*: time to replace a block from lower level, including time to replace in CPU
  - *access time*: time to lower level = f(latency to lower level)
  - *transfer time*: time to transfer block =f(BW between upper & lower levels)

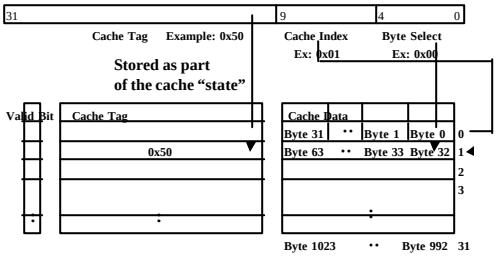
Simplest Cache: Direct Mapped



- Location 0 can be occupied by data from:
  - Memory location 0, 4, 8, ... etc.
  - In general: any memory location whose 2 LSBs of the address are 0s
  - Address<1:0> => cache index
- Which one should we place in the cache?
- How can we tell which one is in the cache?

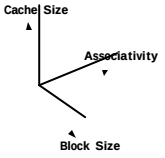
1 KB Direct Mapped Cache, 32B blocks

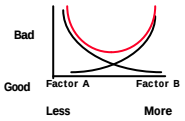
- For a 2 ** N byte cache:
  - The uppermost (32 - N) bits are always the Cache Tag
  - The lowest M bits are the Byte Select (Block Size = 2 ** M)



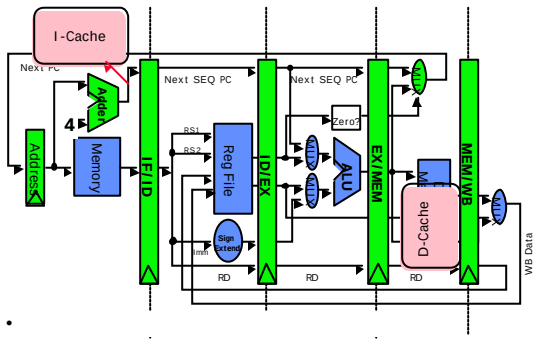
The Cache Design Space

- Several interacting dimensions
  - cache size
  - block size
  - associativity
  - replacement policy
  - write-through vs write-back
- The optimal choice is a compromise
  - depends on access characteristics
    - » workload
    - » use (I-cache, D-cache, TLB)
  - depends on technology / cost
- Simplicity often wins

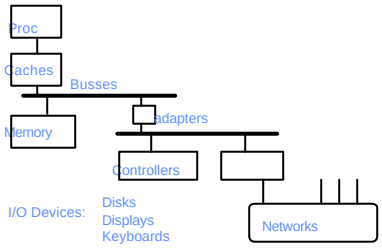




Relationship of Caching and Pipelining



Computer System Components

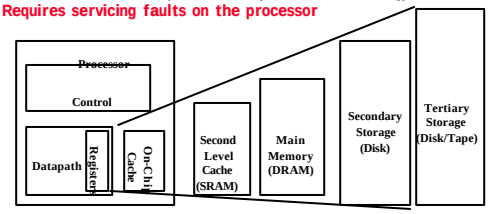


- All have interfaces & organizations
- Bus & Bus Protocol is key to composition

=> perhipheral hierarchy

A Modern Memory Hierarchy

- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.
- Requires servicing faults on the processor



Speed (ns):	1s	10s	100s	10,000,000s (10s ms)	10,000,000,000s (10s sec)	
Size (bytes):	100s	Ks	Ms	Gs	Ts	

TLB, Virtual Memory

- Caches, TLBs, Virtual Memory all understood by examining how they deal with 4 questions: 1) Where can block be placed? 2) How is block found? 3) What block is repalced on miss? 4) How are writes handled?
- Page tables map virtual address to physical address
- TLBs make virtual memory practical
  - Locality in data => locality in addresses of data, temporal and spatial
- TLB misses are significant in processor performance
  - funny times, as most systems can't access all of 2nd level cache without TLB misses!
- Today VM allows many processes to share single memory without having to swap all processes to disk; today VM protection is more important than memory hierarchy

Summary

- Modern Computer Architecture is about managing and optimizing across several levels of abstraction wrt dramatically changing technology and application load
- Key Abstractions
  - instruction set architecture
  - memory
  - bus
- Key concepts
  - HW/SW boundary
  - Compile Time / Run Time
  - Pipelining
  - Caching
- Performance Iron Triangle relates combined effects
  - Total Time = Inst. Count x CPI + Cycle Time