18-548
Memory Systems Architecture

Prof. Philip Koopman

Lecture: Monday & Wednesday 9:00-10:20 AM in DH 2122

Recitation: Friday 10:30-11:20 AM in HH B131

12 Units

Preview

- **Course information**
  - Goals
  - Administrative info
  - Materials
  - Grading
- **Computer trends**
  - Why is memory hierarchy such a big deal?
- **Preview of course**

Course Goals

- **Deep understanding of data flows & storage within computer systems**
  - Key architectural principles:
    - Latency / Bandwidth / Replication / Balance / Hierarchy
  - Applied to:
    - Cache memory; Main memory; Buses; Vector processing; Virtual memory Multiprocessor coherence
- **Demonstrate and apply principles**
  - Design tradeoffs: prediction & measurement
  - Software speedup using advanced memory architecture understanding
- **Practice research/architecture skills**
  - Running simulations & interpreting experimental data
  - Focus more on analysis+experimentation than on design synthesis

Administrative Notes

- **People:**
  - Instructor: Prof. Phil Koopman koopman@cmu.edu HH D-202
  - TA: Erik Riedel riedel+@CMU.EDU WeH 8114
    Greg Mann gm3g+@andrew.cmu.edu
  - Secretary: Karen Lindenfelser karen@ece.cmu.edu HH D-204
- **Course web page:**
  - http://www.ece.cmu.edu/~ece548
  - Contains office hours and other important information
- **Course communications:**
  - cmu.ece.class.ece548
    - Mandatory reading for announcements
    - Unmoderated; OK to use for discussion of class-related issues

Required Textbook: Cragon

- ***Memory Systems and Pipelined Processors***
  Cragon
  - Newer textbook that concentrates on memory first
  - Good details, but better read as a "reference" than as a novel
  - If there is some obscure issue, Cragon probably discusses it...

- **Coverage:**
  1) Memory Systems
  2) Caches
  3) Virtual Memory
  4) Memory Addressing and I/O Coherency
  5) Interleaved Memory and Disk Systems
  11) Vector Processors

Recommended Text: Hennessy & Patterson

- ***Computer Architecture: a quantitative approach***
  Hennessy & Patterson (2nd. edition)
  - A update of a classic in the field; based on quantitative simulation
  - For our purposes, more breadth than depth
  - Use for orientation to area; but skips many details
  - You ought to have read the first half of the book (or have equivalent knowledge) as a pre-requisite for this course
  - We're *not* going to go into the DLX architecture

- **Coverage:**
  5) Memory-Hierarchy Design
  6) Storage Systems
  8) Multiprocessors
  App. B) Vector Processors

Simulation Tools

- **Dinero -- cache simulator**
  - Special version will be used
    - Not limited to 32-bit operation
    - Modified for multi-level cache simulations
  - Compiles & runs on most Unix platforms
    - Only supported versions will be on IBM PPC workstations in the "HP Lab" & DEC Alphas we will provide accounts on
    - Must run on a 64-bit processor to handle 64-bit address traces
- **Atom -- program annotation tool**
  - Annotates programs to record/generate information
    - We'll be using primarily to generate address traces for cache simulations
    - Does many other nifty things (see the man page...)
  - Only works on DEC Alphas -- we will provide accounts
    - ECE students must have an ECE account
    - We're working with ECE facilities to create accounts for CS students

Grading

- **Grade distribution:**
  - 20% first test
  - 25% second test
  - 25% third test
  - 10% distributed among approximately 11 weekly homework sets
  - 20% distributed among 5 lab assignments (note: no physical "lab" room)
  - 5-point grading system per question
- **Assignments must be handed in on time**
  - Homeworks due Wednesday ***in class*** on due date; solutions handed out Fridays
  - Labs due Friday afternoons at *3 PM* to course secretary; solutions next Friday
  - Late materials accepted until solutions handed out; 10% penalty/day late
  - Don't run computer simulations at the last minute!
    - Expect machines to be overloaded the night before an assignment is due
- **All products must be a result of your own efforts**
  - However, you may ask for general guidance from staff & fellow students

1
Course Introduction

18-548 Memory System Architecture
Philip Koopman
August 26, 1998

Assignments

- **By next class read about Key Concepts:**
  - Hennessy & Patterson: page 7, Section 1.7
  - Cragon, Chapter 1
  - Supplemental reading: All of Hennessy & Patterson Chapter 1
- **Homework due by next class:**
  Send e-mail information to:
  riedel+@cmu.edu, koopman@cmu.edu
  - Full name & preferred nickname + pronunciation
  - Preferred e-mail address (not necessarily ECE or even CMU)
  - ECE account name if ECE student
  - CS and Andrew account name if CS student
  - One sentence on area of graduate research (if any)

Where Are We Now?

- **Where we're going today:**
  - Sampling of material for entire course
- **Where we're going next:**
  - Key concepts
  - Physical memory *vs.* virtual memory
  - Caches

COMPUTER TRENDS

Capacity Growth

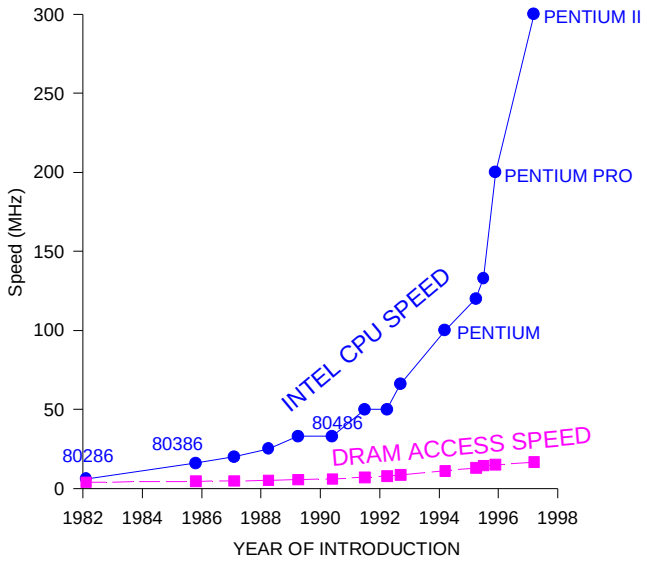
- **Good news -- exponential growth in hardware capacity**
  - Logic growth: 60% to 80% per year
  - DRAM growth: 60% per year (in 400% increments/3 years)
  - Disk growth: 50% per year (was 25%/year until 1990)
- **Bad news -- exponential growth in hardware usage**
  - Program size: 50% to 100% per year
    - Increase software productivity -- standard components/interface layers
    - Feature list increases -- "bloatware"
    - Memory used to enhance user interface -- GUI
- **Results**
  - Storage may be approaching "free" on a per-bit basis, but somehow it always seems to be full

Computer Performance

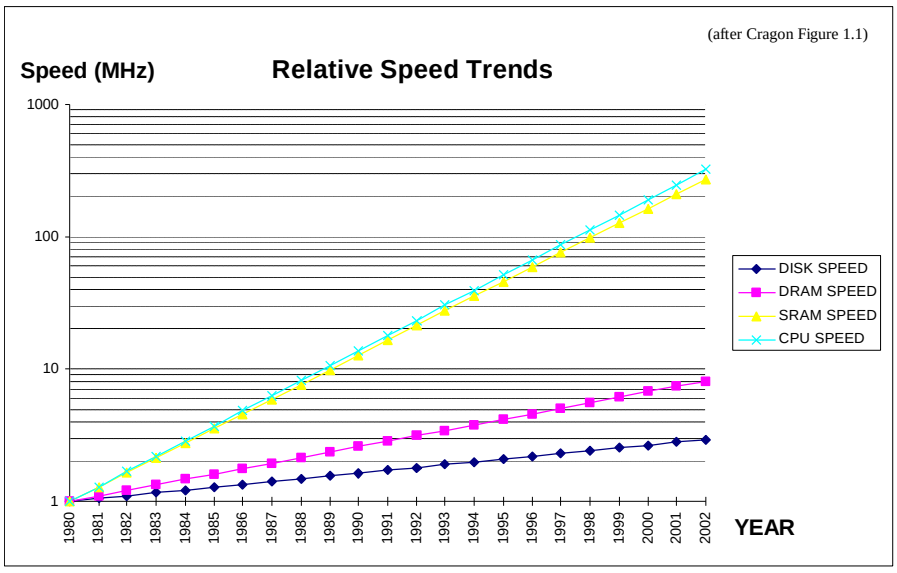
- **task time = number of instructions executed**
  *** clocks per instruction (CPI)**
  *** clock period**
- **Number of instructions depends on ISA, language, compiler**
- **Overall clocks per instruction:**
  - Instruction complexity (usually 1 clock, but can be longer)
  - Instruction issue rate (superscalar may be > 1 issue/clock)
  - Data dependence & resource stalls
  - ***Instruction fetch latency***
  - ***Data fetch/store latency***
- **Clock period**
  - Logic critical path (e.g., hardware multiplier) & clock distribution tree
  - ***First level cache cycle time***

*** = emphasized in this course***

Memory/CPU Speed Gap



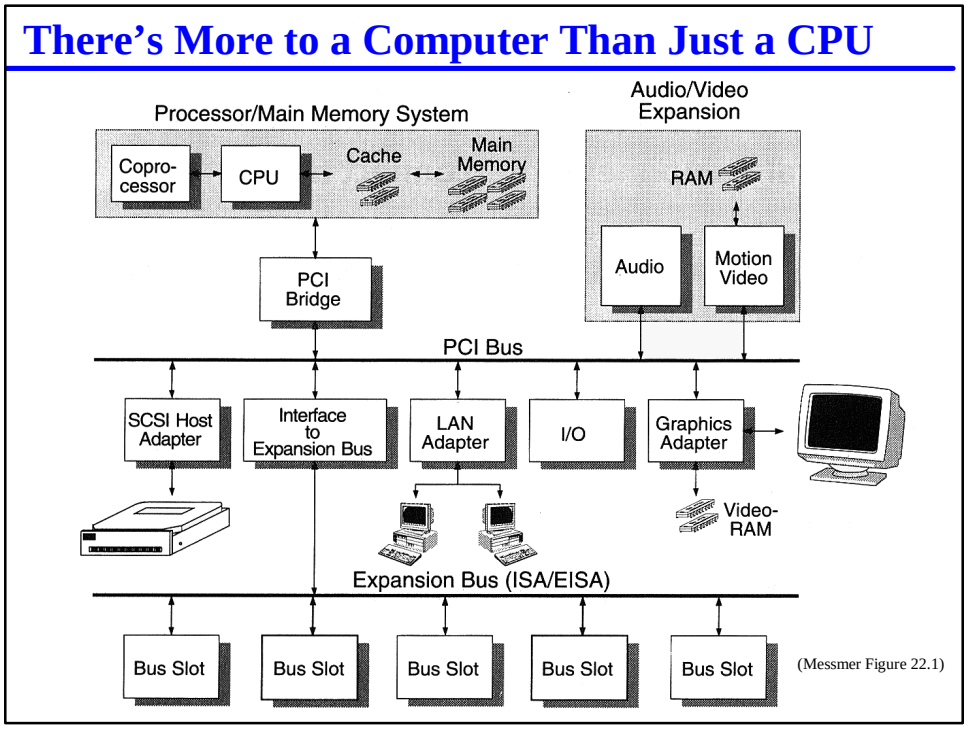
Raw Speed Gap Projection

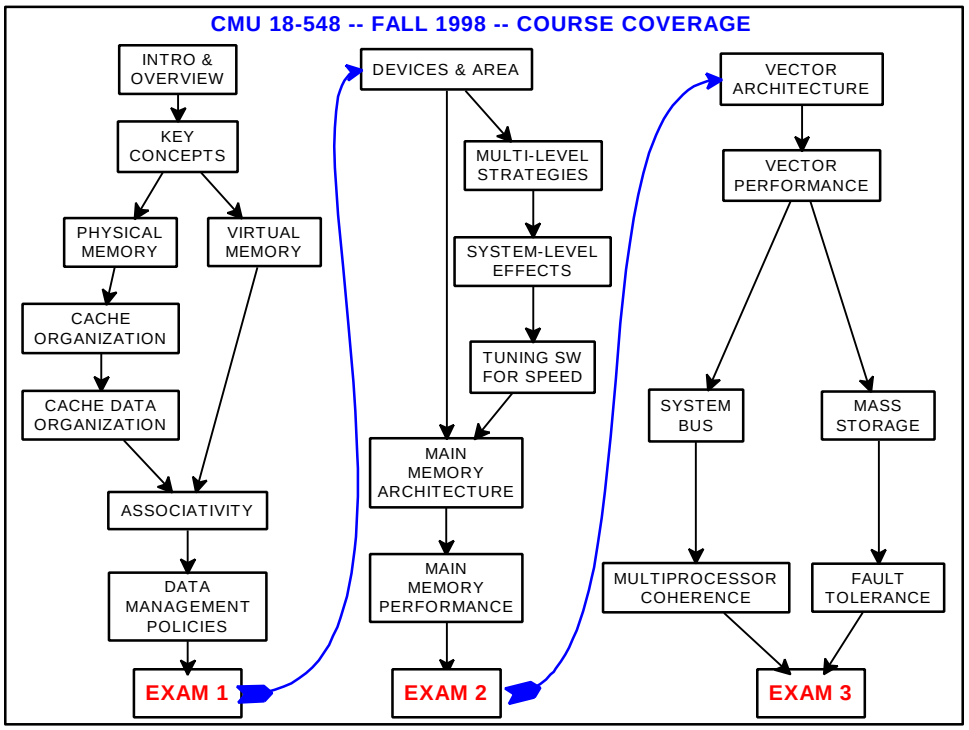


Why Not Just Use SRAM?

- **Off-chip vs. on-chip access limitations**
  - Delay -- pins are (in general) slower than on-chip access (more recently, across-chip is slower than nearby-on-chip)
  - Bandwidth -- a few hundred pins vs. thousands of traces
- **Density**
  - *e.g.*, 15 ns 256 MB of SRAM using 256 KB SIMMS = 1024 SIMMS
  - Address/data line fan-out & chip select slowdown ~ log N
  - Physical propagation delay
- **Heat/power**
  - 1024 SIMMS @ ~1000 mW (standby for async SRAM) =>1 KW power
- **Cost**
  - $30/SIMM (December 1996) => $30,000 (less quantity discount...)
- **Slower technology tends to be more cost effective**
  - 60 ns 32 MB DRAM SIMM $270 => 8 SIMMS, $2160
  - 256 MB is too small a disk to buy (about $0.32 / MB 8 ms => $82)

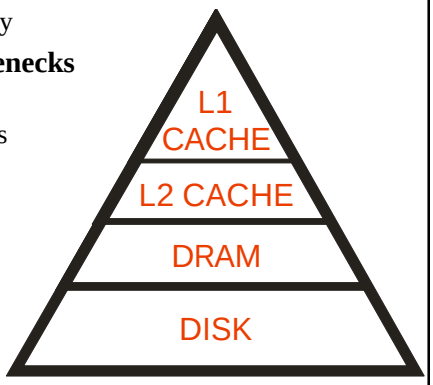
COURSE SAMPLER


There's More to a Computer Than Just a CPU
Processor/Main Memory System
Audio/Video Expansion
Copro-cessor
CPU
Cache
Main Memory
RAM
Audio
Motion Video
PCI Bridge
PCI Bus
SCSI Host Adapter
Interface to Expansion Bus
LAN Adapter
I/O
Graphics Adapter
Video-RAM
Expansion Bus (ISA/EISA)
Bus Slot
Bus Slot
Bus Slot
Bus Slot
Bus Slot
(Messmer Figure 22.1)

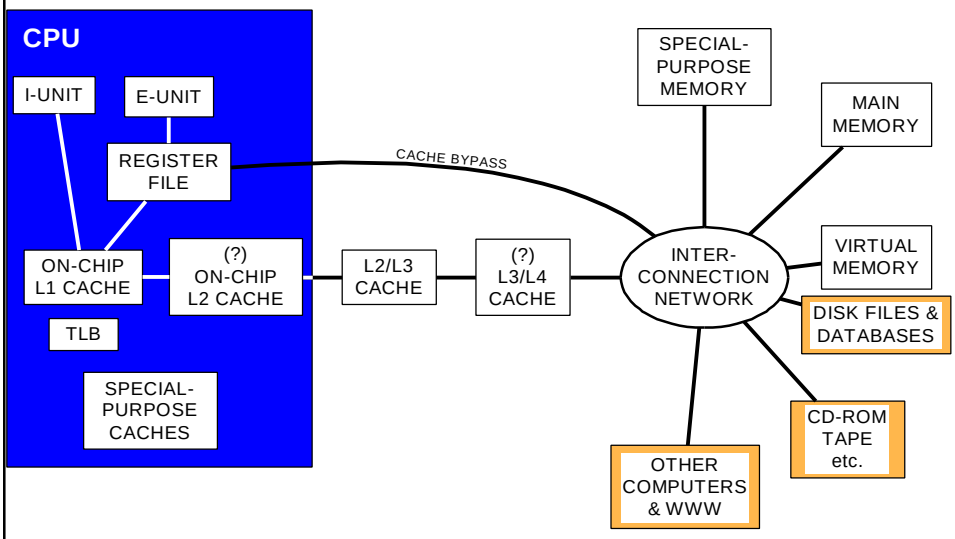

CMU 18-548 -- FALL 1998 -- COURSE COVERAGE
INTRO & OVERVIEW
KEY CONCEPTS
PHYSICAL MEMORY
VIRTUAL MEMORY
CACHE ORGANIZATION
CACHE DATA ORGANIZATION
ASSOCIATIVITY
DATA MANAGEMENT POLICIES
EXAM 1
DEVICES & AREA
MULTI-LEVEL STRATEGIES
SYSTEM-LEVEL EFFECTS
TUNING SW FOR SPEED
MAIN MEMORY ARCHITECTURE
MAIN MEMORY PERFORMANCE
EXAM 2
VECTOR ARCHITECTURE
VECTOR PERFORMANCE
SYSTEM BUS
MASS STORAGE
MULTIPROCESSOR COHERENCE
FAULT TOLERANCE
EXAM 3

Key Concepts

- **Latency -- delay**
  - Minimize delay to limit length of stalls waiting for data
- **Bandwidth -- data moved per unit time**
  - Maximize data path widths & clock rates
- **Concurrency -- multiple units (or pipelining to re-use a single unit)**
  - Adding resources can increase bandwidth
  - Exploiting concurrency can reduce latency
- **Balance -- avoiding performance bottlenecks**
  - Bottlenecks limit system performance
    balanced systems have no real bottlenecks
- **Memory hierarchy -- a way to balance speed with cost by exploiting locality (re-use/nearness) of memory accesses**



Reference Memory Hierarchy



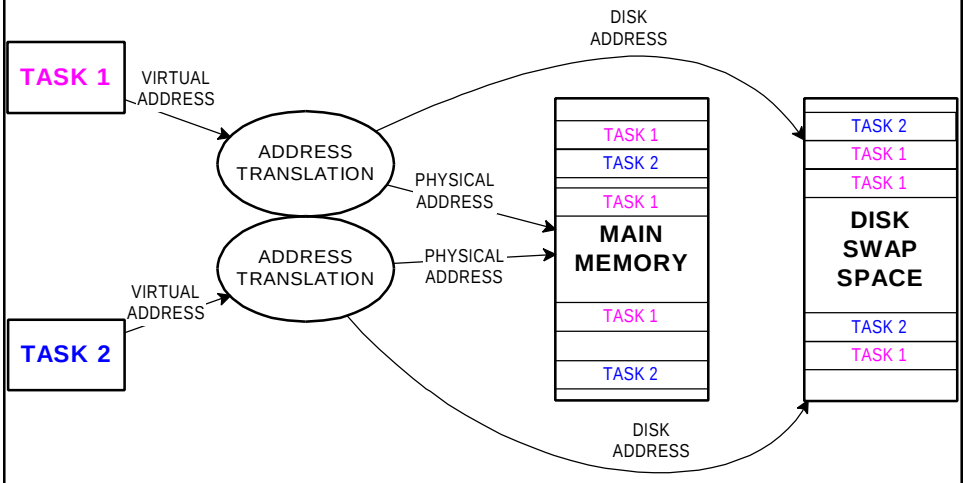
Example Memory Hierarchy

- **Digital AlphaServer 4000 5/600 (Alpha 21164 CPU)**
  - Dual 600 MHz CPUs
    - Registers
    - On-chip 16 KB data cache + 16 KB instruction cache
    - On-chip 96 KB level 2 cache
    - Off-chip 8 MB level 3 cache
  - 1024 MB main memory
  - 40 GB disk storage
  - Large networked file system (afs)

  - Program loading device (CD ROM)
  - Backup device (tape, network, WORM CD)
  - Internet connection to WWW (via 100 Mb/sec switched port)

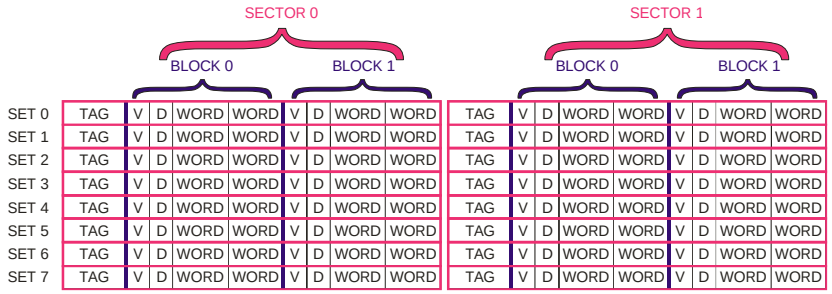
Address Space Hierarchy

- **Virtual address translated to hardware address**
  - Physical address used if page is in main memory; disk address on page fault
  - TLB (Translation Lookaside Buffer) caches address translation information



Cache Organization

- **Cache organized as sectors, blocks, and sets**
  - Each sector corresponds to a location in memory (tag holds address)
  - Cache lookup searches set to find matching address
    - Hit -- found a tag match within set -- provide fast access to data
    - Miss -- didn't find a tag match -- perform slow access to next level in hierarchy
    - Miss penalty -- clock cycles to process a miss



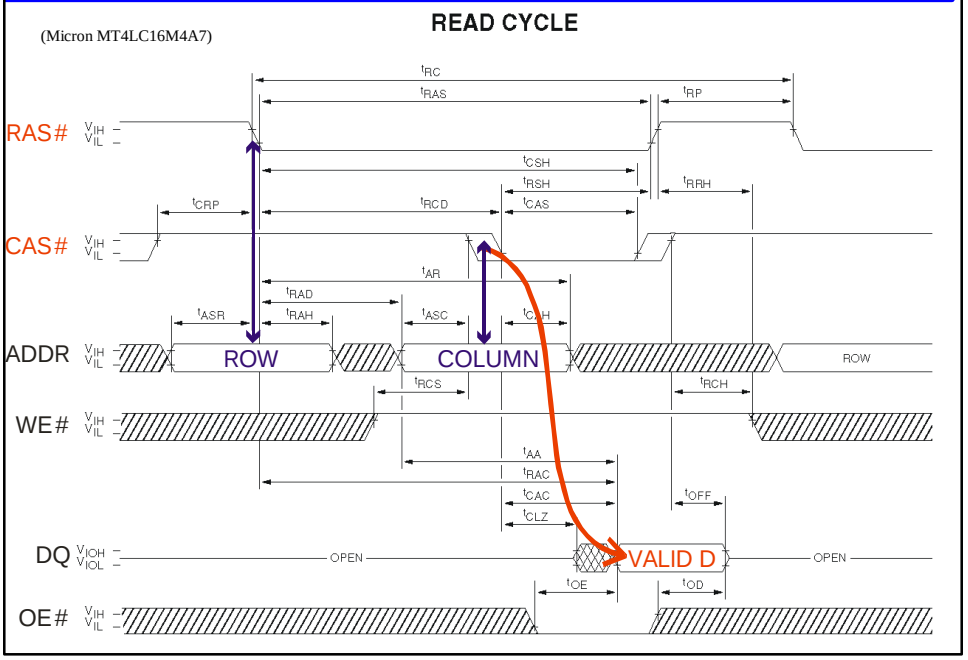
Major Cache Design Decisions

- **Cache Size -- in bytes**
- **Split/Unified -- instructions and data in same cache?**
- **Associativity -- how many sectors in each set?**
- **Sector/Block size -- how many bytes grouped together as a unit?**
- **Management policies**
  - Choosing victim for replacement on cache miss
  - Handling writes (when is write accomplished?; is written data cached?)
- **How many levels of cache?**
  - Perhaps L1 cache on-chip; L2 cache on-module; L3 cache on motherboard

System-Level Effects -- Context Switching

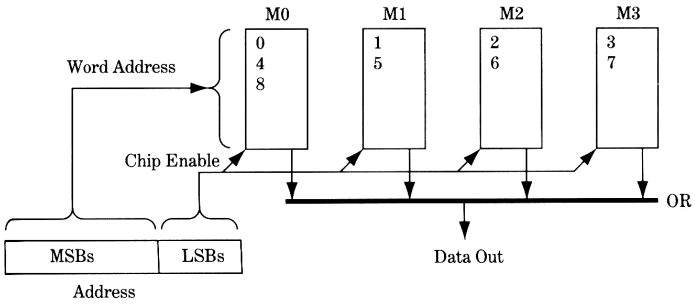
- **Traditionally, cost of context switching is considered to be:**
  - Processing timer interrupt
  - OS scheduling
  - Register save
  - Register restore
- **But, the TRUE process state that might be lost includes:**
  - CPU information (branch prediction, buffers, speculative execution results)
  - Cache contents
  - TLB contents
  - Virtual memory pages resident in memory
  - Disk cache contents
- **Important result: single-task simulations are typically very optimistic compared to real-world performance**

DRAM Read Cycle



Improving Main Memory Performance

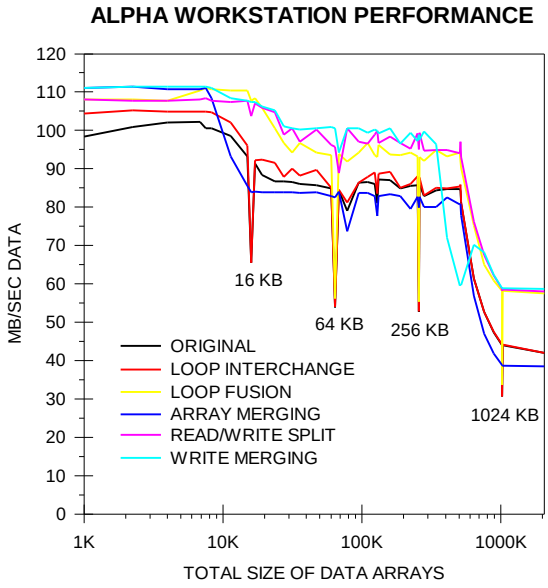
- **Interleaved memory -- multiple banks for concurrent accesses**



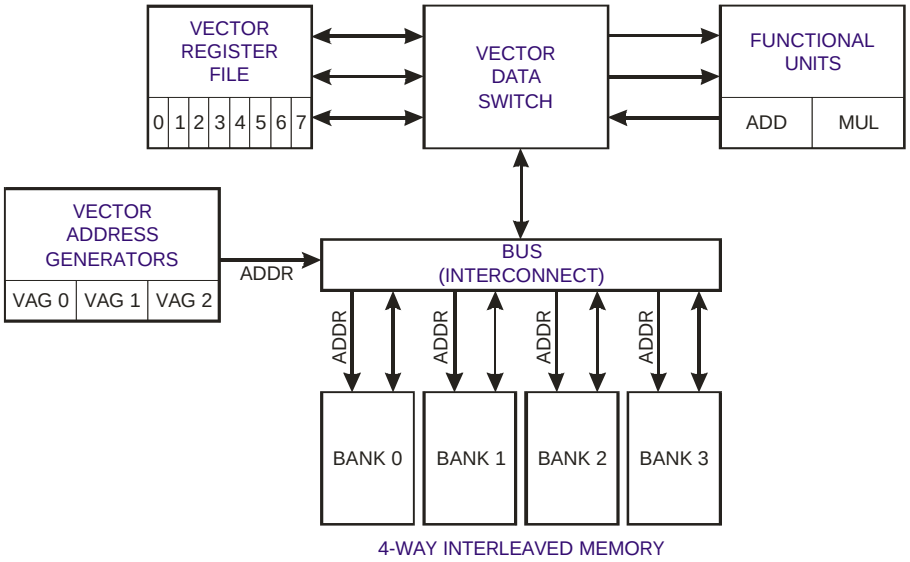
(Cragon Figure 5.1)

- **Use exotic DRAM technology**
  - EDO DRAM
  - SDRAM
  - Cached DRAM
  - Rambus
  - ...

Optimization of Matrix Multiplication



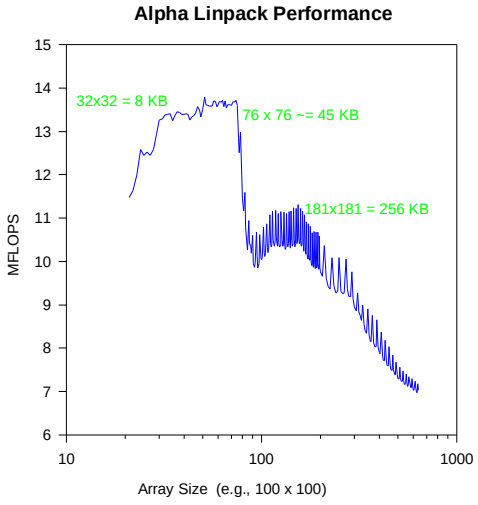
Generic Vector Architecture



- **Vector Address Generators perform concurrent prefetches & stores**
- **Vector Register File holds collated data sets to feed to pipelined functional units**

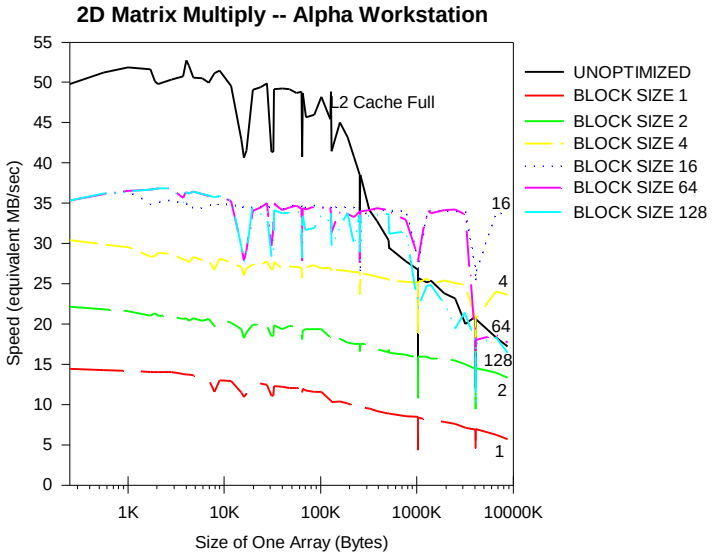
Vector Computers Illustrate Performance Issues

- **Code with a working set larger than available cache will have performance problems**
- **Vector computers teach us techniques to deal with this problem:**
  - Hardware prefetching of regular data access patterns
  - Techniques to segment accesses to memory and reduce working set to fit in available cache



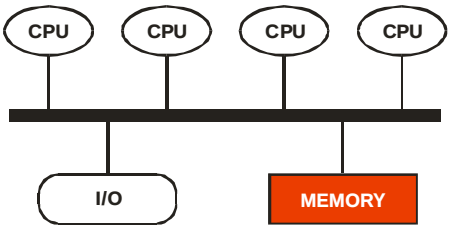
Blocked Matrix Multiply Performance

- **"Blocked" algorithm uses small blocks to fit working set into cache**

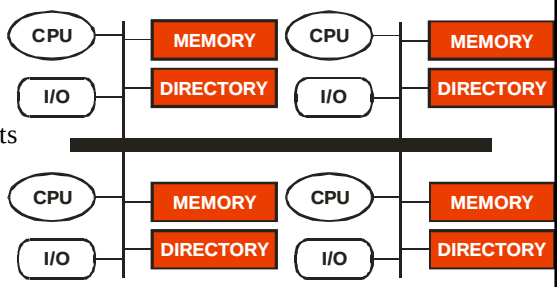


Multiprocessor Memory

- **Bandwidth is an key issue**
- **So is ensuring coherence among shared memory locations**



- **Centralized shared memory**
  - Single memory shared over interconnect



- **Distributed memory**
  - Numerous processing elements organized as clusters interconnected

Important Lessons in System Architecture

- **Every level of memory hierarchy employs similar principles**
  - Registers
  - Cache
  - Main memory
  - Disks
  - Multiprocessors
- **Address translation permits automatic management of memory**
  - Caches
  - Virtual memory
  - Multiprocessing
- **A balance between latency and bandwidth is crucial**
- **Concurrency is an effective tool for improving performance**
  - Replication of resources
  - Pipelining of individual resources
  - Prefetching and delayed storing of data

REVIEW

Review

- **CPU / memory speed gap is increasing in size**
  - Clock speed & superscalar CPUs require more from memory
  - But, memory isn't keeping up with the demands
- **Memory hierarchies are the obvious solution**
  - Registers
  - Cache
  - Main memory
  - Disks
  - Multiprocessor memories
- **Key concepts apply at all levels**
  - Latency -- delay
  - Bandwidth -- data moved per unit time
  - Concurrency -- multiple units (or pipelining to re-use a single unit)
  - Balance -- avoiding performance bottlenecks