3
Physical Memory Architecture

18-548 Memory System Architecture

Philip Koopman

August 31, 1998

Required Reading:

Hennessy & Patterson: skim 5.1, 6.1-6.2, pp. 496-497, 7.1, pp. 635-642

Cragon: 2.0

http://www.isdmag.com/Editorial/1996/CoverStory9610.html

Supplemental Reading:

Hennessy & Patterson: 2.0-2.3, 2.7

Assignments

- **By next class read about cache organization and access:**
  - Hennessy & Patterson 5.1, 5.2, pp. 390-393
  - Cragon 2.1, 2.1.2, 2.1.3, 2.2.2
  - Supplemental Reading: Flynn 1.6, 5.1
- **Homework 1 due Wednesday September 2**
- **Lab 1 due Friday September 4 at 3 PM.**

Where Are We Now?

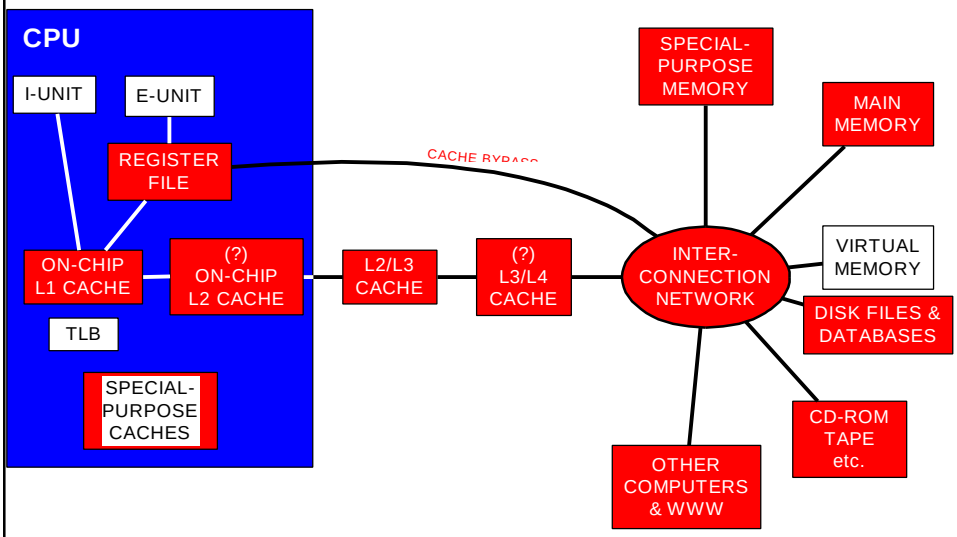
- **Where we've been:**
  - Key concepts applied to memory hierarchies
    - Latency
    - Bandwidth
    - Concurrency
    - Balance
- **Where we're going today:**
  - Physical memory architecture -- a trip through the memory hierarchy
- **Where we're going next:**
  - Cache organization and access
  - Virtual memory

Preview

- **Principles of Locality**
- **Physical Memory Hierarchy:**
  - CPU registers
  - Cache
  - Bus
  - Main Memory
  - Mass storage
- **Bandwidth "Plumbing" diagrams**

Physical Memory Architecture

- **Emphasis on providing low latency and high bandwidth**



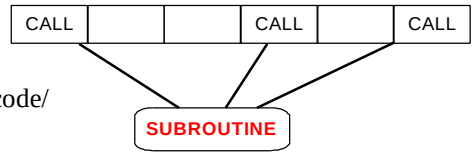
LOCALITY

Locality

- **Typically, one can predict future memory access addresses from past ones**
- **Temporal Locality**
  - A future memory access is likely to be identical to a past one
- **Spatial Locality**
  - A future memory access is likely to be near a past one
- **Sequentiality**
  - A future memory access is likely to be in sequential order (*e.g.*, a program counter)

Temporal Locality

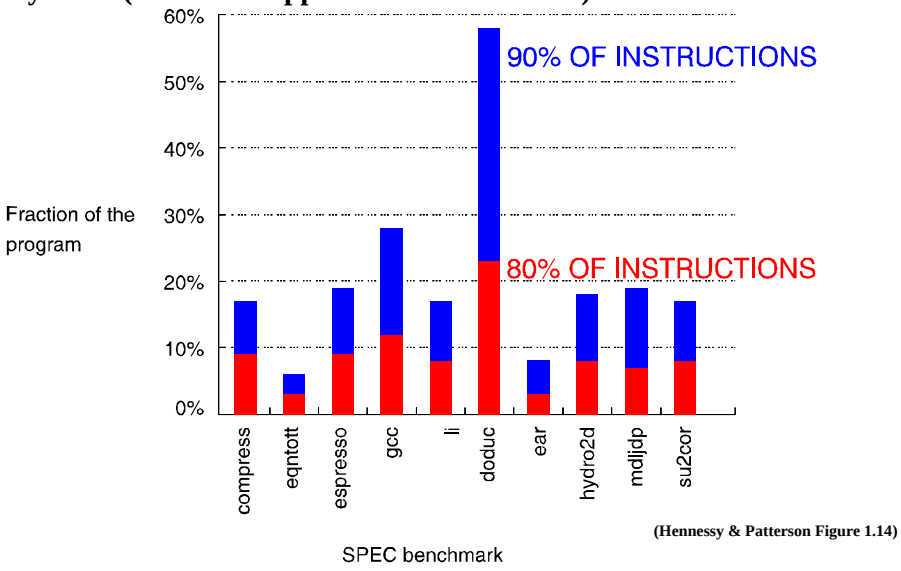
- ***Once a memory word has been accessed, it is likely to be accessed again***
- **Program structures**
  - Loops
  - Frequently used subroutines/ object methods (calls)
  - Frequently used interrupt service code/ TLB miss code



- **Application data structures**
  - Frequently used variables that aren't register-resident
  - "Directory" information (*e.g.*, linked list pointer chain)
- **System-induced temporal locality**
  - Local variables on stack frame
  - Generational garbage collection

Temporal Locality of Instructions

- **80/20 rule of systems: 80% of cost or benefit attributable to 20% of system (sometimes appears as the 90/10 rule)**



(Hennessy & Patterson Figure 1.14)

Spatial Locality

- ***Once a word has been accessed, neighboring words are likely to be accessed***

- **Program structures**
  - Short relative branches
  - Related methods in same object (*e.g.*, constructor followed by operator)
- **Data structures**
  - Records (C language *struct)*
  - Operations on 1-dimensional arrays of data
  - Image processing (*e.g.*, 2-D convolution)
- **Coincidental spatial locality**
  - Putting important routines together for virtual memory locality
  - Overlays (manual management of pages with small address spaces)

Instruction Spatial Locality

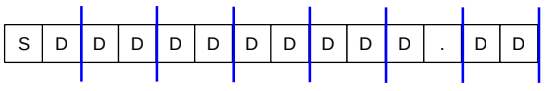
- **Branch offset coverage** (Flynn Table 3.12)

Offset	% Branches
± 8	13%
± 16	33%
± 32	37%
± 64	46%
± 128	55%
± 256	61%
± 512	68%
± 1K	73%
± 2K	79%
± 4K	83%
± 8K	87%
± 16K	93%
± 32K	99%

Sequentiality

- ***Memory words tend to be accessed in sequential order*** **(a special case of spatial locality)**

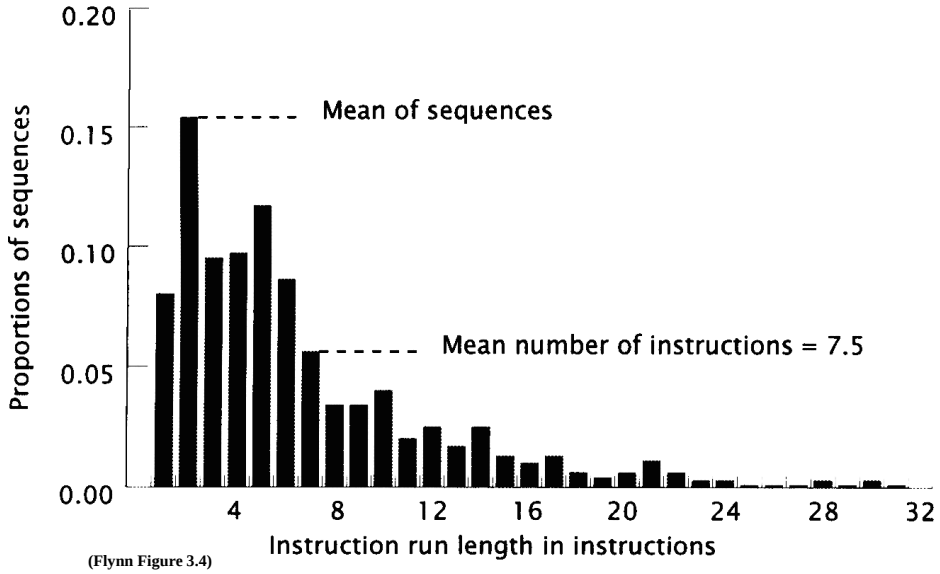
- **Program structures**
  - Program counter
  - Compiler output arranged to avoid branch delay penalties
- **Data structures**
  - Strings, BCD, multi-precision numbers
  - Vector/matrix operations with unit stride (or, sometimes, any fixed stride)
  - Multimedia playback



7-BYTE/12-DIGIT BCD NUMBER

Basic Block Lengths: Sequentiality

- **A few large blocks raise mean length**



(Flynn Figure 3.4)

CPU-BASED PHYSICAL MEMORY ELEMENTS

CPU Resources Form Top of Memory Hierarchy

- **Registers**
  - Register file
  - Top-of-stack "cache" (actually more like a buffer)
- **Buffers**
  - Instruction pre-fetch buffer
  - Data write buffer
  - Branch destination buffer (specialized instruction cache)
- **Programs with good locality make effective use of limited resources**
  - Only a handful of active values in register set (grows with superscalar CPUs)
- **Bandwidth -- limited by number of ports in storage**
  - For example, 3-ported 32-bit register file at 500 MHz is ~6 GB/sec
- **Latency -- cycled once or twice per clock cycle**
  - Latency is effectively "zero" -- CPU pipeline takes it into account

Registers Are Software-Managed Cache

- **Registers offer highest bandwidth and shortest latency in system**
  - Built into the CPU
  - Multi-ported fast access register file
- **Compiler/programmer manages registers**
  - Explicit load and store instructions
  - No hardware interlocks for dependencies
- **Very large register set can be effective in the right circumstances**
  - Vector computing where registers store intermediate vector results
  - Embedded systems with multiple register sets for context switching speed
    - But may not be so great for general purpose computing

CACHE MEMORY

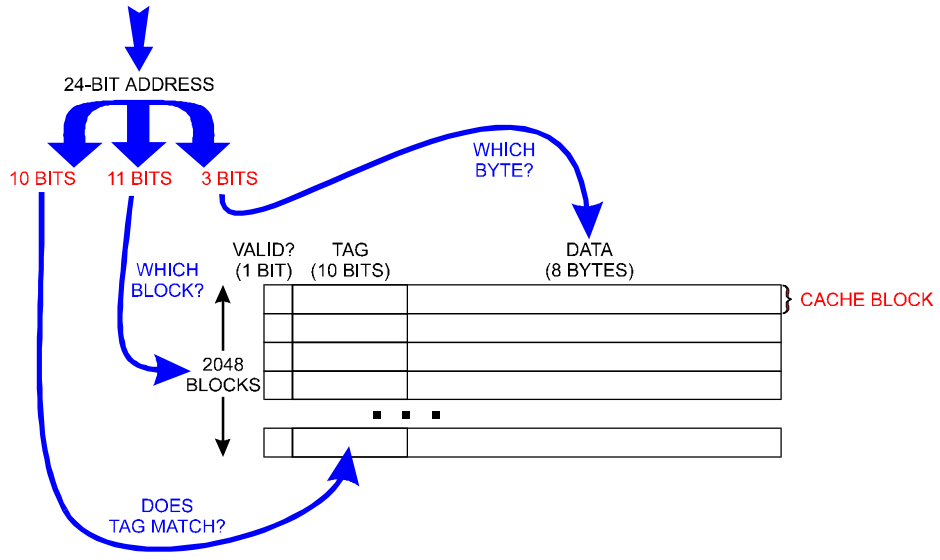
Cache Memory

- **A small memory that provides the CPU with low latency and high bandwidth access**
  - Typically hardware management of which memory locations reside in cache at any point in time
  - Can be entirely software managed, but this not commonly done today (may become more common on multiprocessor systems for managing coherence)
- **Multiple levels of cache memory possible**
  - Each level is typically bigger but slower; faster levels SRAM, slower levels may be DRAM
- **Bandwidth -- determined by interconnection technology**
  - On-chip limited by number of bits in memory cell array
  - Off-chip limited by number of pins and speed at which they can be clocked
- **Latency -- determined by interconnection technology & memory speed**
  - On-chip can be one or more clocks, depending on connect/cycle delays
  - Off-chip is likely to be 3-10 clocks, depending on system design

Major Cache Design Decisions

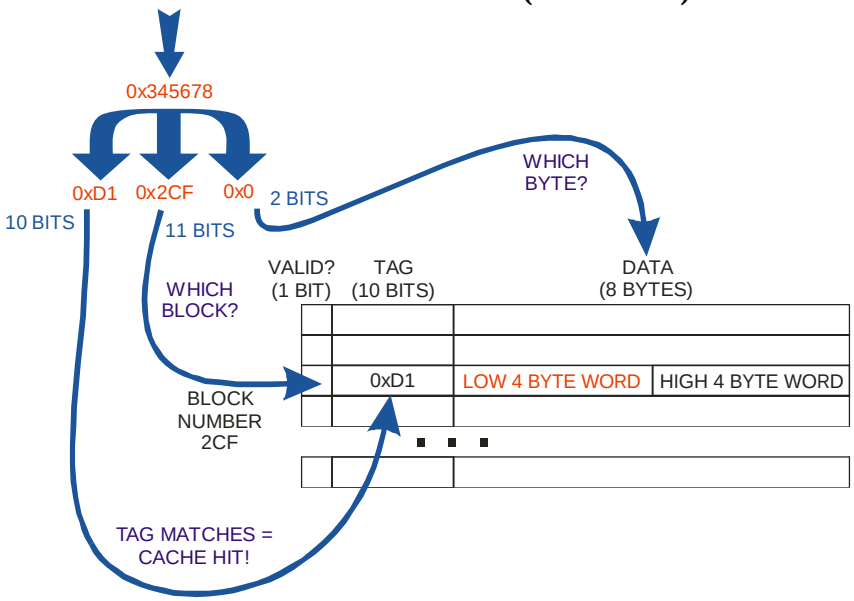
- **Cache Size -- in bytes**
- **Split/Unified -- instructions and data in same cache?**
- **Associativity -- how many sectors in each set?**
- **Sector/Block size -- how many bytes grouped together as a unit?**
- **Management policies**
  - Choosing victim for replacement on cache miss
  - Handling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

Cache Addressing



Example Cache Lookup Operation

- **Assume cache "hit" at address 0x345678 (word access)**

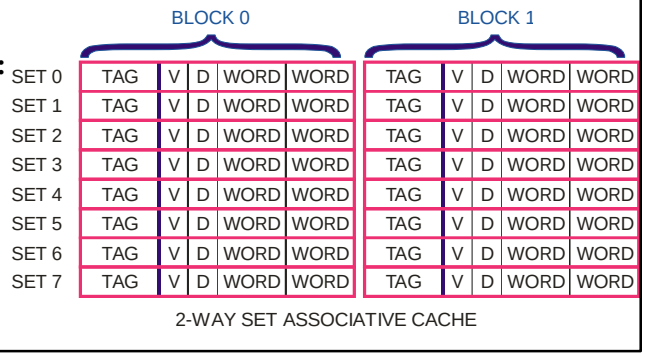


Block size

- **Cache is organized into blocks of data that are moved as a whole**
  - Blocks can range from 4 bytes to 256 bytes or more in size
  - Blocks are loaded and stored as a single unit to/from the cache
- **# blocks = cache size / block size**
- **example: 8 KB cache with 64-byte blocks:**
  - # blocks = 8192 / 64 = 128 blocks

Cache Associativity

- **When a cache block must be replaced with a new one, the cache logic determines which block to replace from a set of candidate blocks**
  - Fully associative: the set consists of all blocks in the cache
  - *n*-way set associative: the set consists of n blocks
    - For example, 4-way set associative means that there are 4 candidates
  - Direct mapped: there is only one block that can be replaced (1-way set associative)
- **# sets = # blocks / *n***
- **example: 256 B cache: 16-byte blocks, 2-way set associative, 16 blocks**
  - #sets = (256/16) / 2 = 8 sets



BUS / INTERCONNECT

Bus

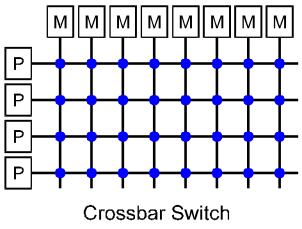
- **Bus is a shared system interconnection**
  - CPU to cache
    - Might be on system bus
    - As CPUs get more pins they move to a dedicated cache bus or all within package
  - Cache to main memory
  - CPU or main memory to I/O
  - Typically high-speed connection, and often carries processor/memory traffic
  - Typically accessed transparently via a memory address

Bus Performance

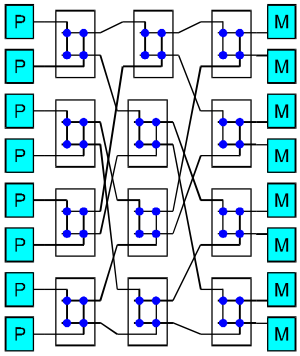
- **Bandwidth -- limited by cost and transmission line effects**
  - 64-bit or 128-bit data bus common (but, fewer bits on cost-sensitive systems)
    - Why was the 8088 used instead of the 8086 in the original IBM PC?
  - Bus speed often limited to 50 - 66 MHz due to transmission line effects
  - Example: Pentium Pro -- up to 528 MB/sec for 64-bit bus at 66 MHz
- **Latency -- limited by distance and need for drivers**
  - Multiple clock latency, but can pipeline and achieve 1 clock/datum throughput
  - Be careful about “bus clocks” *vs.* “processor clocks”
    - Many current processors clocked at a multiple of the bus frequency

Interconnect

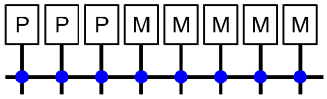
- **Interconnect is a generalization of a bus**
  - Crossbar switch permits connecting, for example, *n* CPUs to *m* memory banks for simultaneous accesses
    - Cost is O(*n***m*) switches
  - Multi-stage interconnection is a heavily research area (hypercubes, omega networks, mesh connections)
    - Cost is O(*n* log *n*) switches
  - Serial interconnections such as Ethernet are widely used for between-box communications
    - Cost is O(n) connections



Crossbar Switch



Omega Network



Serial Bus "Switch"

Interconnect Performance

- **Bandwidth -- usually limited by cost to fast serial connection**
  - Crossbar provides very high bandwidth (n simultaneous connections); but costs $O(n^2)$ in terms of switching nodes
  - Omega network provides potentially high bandwidth, but suffers from blocking/congestion
  - 10 Mbit/sec common for Ethernet; 100 Mbit/sec being introduced
    - Also limited by cost of switch to avoid sharing of high-speed line
- **Latency -- limited by routing**
  - Crossbar offers lowest latency, but at high cost
  - Each “hop” on network requires passing through an embedded processor/switch
    - Can be many seconds for the Internet
    - Omega network provides high potential bandwidth, but at cost of latency of log *n* switching stages

MAIN MEMORY

Main Memory

- **Main Memory is what programmers (think they) manipulate**
  - Program space
  - Data space
  - Commonly referred to as “physical memory” (as opposed to “virtual memory”)
- **Typically constructed from DRAM chips**
  - Multiple clock cycles to access data, but may operate in a “burst” mode once data access is started
  - Optimized for capacity, not necessarily speed
- **Latency -- determined by DRAM construction**
  - Shared pins for high & low half of address to save on packaging costs
  - Typically 2 or 3 bus cycles to begin accessing data
  - Once access initiated can return multiple data at rate of datum per bus clock

Main Memory Capacities

- **Main memory capacity is determined by DRAM chip**
  - At least 1 “bank” of DRAM chips is required for minimum memory size
  - Multiple banks (or bigger chips) used to increase memory capacity
- **Bandwidth -- determined by memory word width**
  - Memory words typically same width as bus
  - Peak memory bandwidth is usually one word per bus cycle
  - Sustained memory bandwidth varies with the complexity of the design
    - Sometimes multiple banks can be activated concurrently, exploiting “interleaved” memory
  - Representative main memory bandwidth is 500 MB/sec peak; 125 MB/sec sustained

Special-Purpose Memory

- **Memory dedicated to special hardware use**
  - Graphics frame buffer
    - Special construction to support high-speed transfers to video screen
    - Dual-ported access for modify-while-display
  - Video capture buffer
- **Perhaps could consider memory embedded in other devices as special-purpose system memory**
  - Postscript printer memory
  - Audio sample tables

DISK / BACKING STORE

"Backing Store"

- **Used for data that doesn't fit into main memory**
  - "Virtual memory" uses it to emulate a larger physical memory
  - File systems use it for nonvolatile storage of information
- **Hard disk technology is prevalent, but don't forget:**
  - Flash memory (especially for mobile computing) based on IC technology
  - Floppy disks (up to 100 MB)
  - CD-ROM (and WORM, and magneto-optical, and ...)
  - Novel storage (bar codes, optical paper tape...)
- **Magnetic disks have killed off:**
  - Magnetic drums
  - Punched cards
  - Paper tape
  - Bubble memory
  - Magnetic tape (not quite dead yet)

Disk Construction

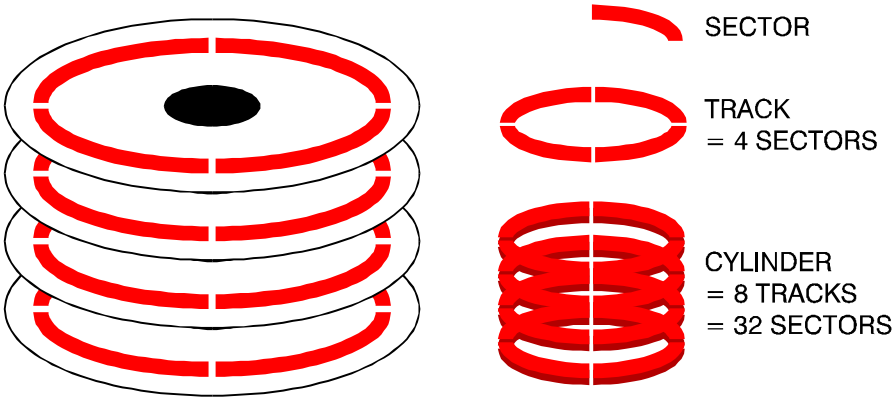
- **One or more platters with one or more heads**
  - Details of sectors/tracks/cylinders discussed at a later lecture

Seagate Technology

- **Latency: disk controller + physics of disk**
  - Seek latency for head to move into position
  - Rotational latency for disk to spin to correct position under the head
  - Minor additional processing latency
  - Typical latency is a handful or so of milliseconds; but can vary considerably from access to access
- **Bandwidth: disk controller + physics of disk**
  - Raw bandwidth is determined by data density on disk and rotational rate
  - Electronics have to be able to handle the data to achieve peak bandwidth
  - Transfer rate may be 1-10 MB/sec

Physical Disk Layout Example

- **Example: 4 platters, 2-sided**



PLUMBING DIAGRAMS

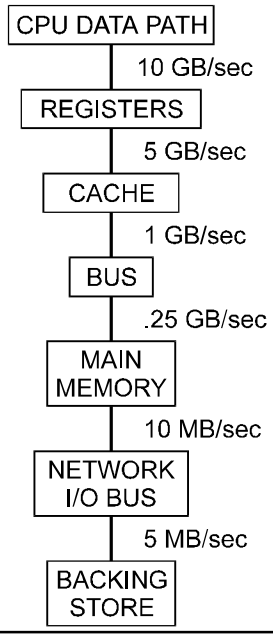
Physical Memory Characteristics

- **Approximate & representative numbers for 1998 technology**

	Capacity	Latency	Bandwidth
Registers	128 Bytes+	1 CPU clock (2-5 ns)	3-10 GB/sec
Cache	8 KB - 1 MB	1-10 clocks	3-5 GB/sec
Bus	---	10+ clocks	0.5 - 1 GB/sec
Main Memory	8 MB - 1 GB	25 - 100 clocks	0.1 - 1 GB/sec
Interconnect	---	1-10 msec	1-20 MB/sec
Backing Store	8 GB+	5-10 msec	1-10 MB/sec

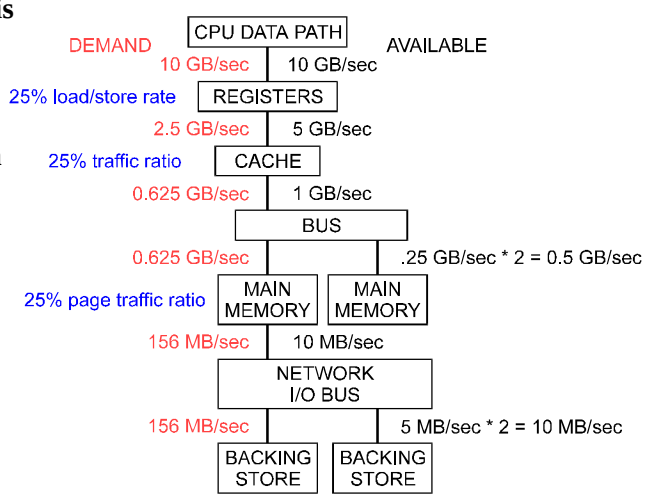
Example Baseline Plumbing Diagram

- **Example for bandwidth computations**
  - Uses representative numbers; not necessarily corresponding to any particular real system
- **Bandwidths may or may not be sufficient depending on attenuation factor at each stage**



An Example That Is Memory-Bandwidth-Bound

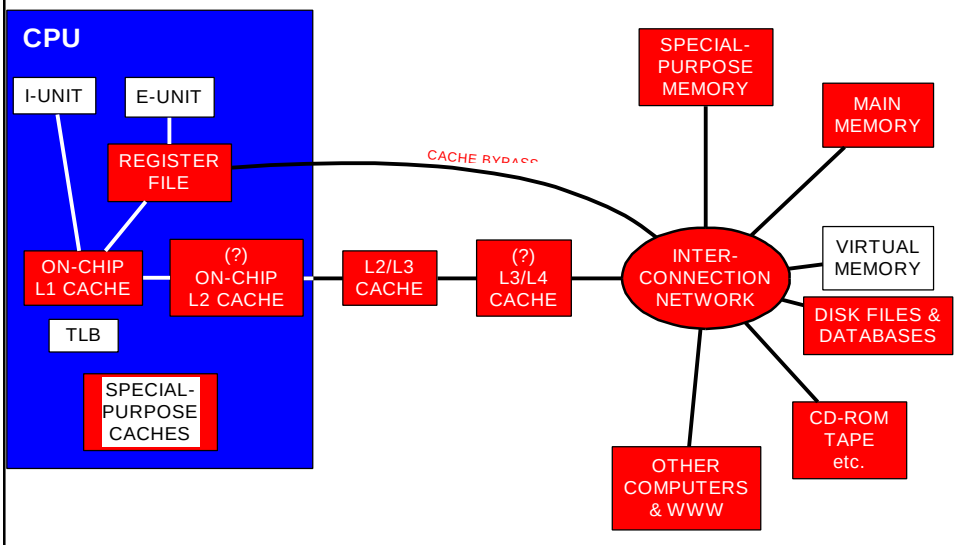
- **Assume register file is used at full bandwidth**
- **Assume 25% of necessary bandwidth gets through each level of memory filtering**
  - This is not really a miss rate in that it is by bandwidth, not simply number of accesses
- **Result:**
  - For this example, not enough bandwidth beyond the bus



REVIEW

Physical Memory Architecture

- **Emphasis on providing low latency and high bandwidth**



Key Concepts

- **Latency**
  - Higher levels in hierarchy have lower latency because there are shorter and less complicated interconnections
- **Bandwidth**
  - Generally higher levels in hierarchy have greater bandwidth because it's cheaper to run wider data paths with short "distances" and low fanout
- **Concurrency**
  - Replication of resources can improve bandwidth for a cost
    - Split caches
    - Concurrent interconnection paths
    - Multiple memory banks
    - Multiple mass storage devices
- **Balance**
  - Plumbing diagrams can give a first-order approximation of system balance for throughput

Review

- **Principles of Locality**
  - Temporal, spatial, sequential
- **Physical Memory Hierarchy:**
  - CPU registers -- smallest & fastest; measured in Bytes
  - Cache -- almost as fast as CPU; measured in KB
  - Bus/Interconnect -- bandwidth costs money
  - Main Memory -- slow but large; measured in MB
  - Mass storage -- slower and larger; measured in GB
- **Bandwidth "Plumbing" diagrams**
  - Back-of-envelope calculations can demonstrate bottlenecks