Chapter 2: Memory Hierarchy Design (Part 3)

Introduction
Caches
Main Memory (Section 2.3)
Virtual Memory (Section 2.4)
Memory Technologies

Dynamic Random Access Memory (DRAM)
  Optimized for density, not speed
  One transistor cells
Multiplexed address pins
  Row Address Strobe (RAS)
  Column Address Strobe (CAS)
Cycle time roughly twice access time
  Destructive reads
Must refresh every few ms
  Access every row
Sold as dual inline memory modules (DIMMs)
Static Random Access Memory (SRAM)

- Optimized for speed, then density
  - 4 to 6 transistors per cell
  - Separate address pins
- Static $\Rightarrow$ No Refresh
- Greater power dissipation than DRAM
- Access time = cycle time
DRAM Organization

DIMM

Rank

Bank

Array

Row buffer
DRAM Organization

Rank: chips needed to respond to a single request

Assume 64 bit data bus

For 8 bit DRAM, need 8 chips in a rank
For 4 bit DRAM, need 16 chips in a rank

Can have multiple ranks per DIMM

Bank: A chip is divided into multiple independent banks for pipelined access

Array: A bank consists of many arrays, 1 array per bit of output, for parallel access

Row buffer: A “cache” that preserves the last row read from a bank
DRAM Organization

See figure 1.5 in

The Memory System: You Can’t Avoid It, You Can’t Ignore It, You Can’t Fake It
By Bruce Jacob
Series editor: Mark Hill

Downloadable from U of I accounts
Internals of a DRAM Array

See Figure 1.6 of the synthesis lecture

Steps to access a bit

Pre-charge bit lines

Activate row: turn on word line for the row, brings data to sense amps

Column read: send subset of data (columns)

(Restore data)
DRAM Optimizations – Page Mode

Unoptimized DRAM

First read entire row
Then select column from row
Stores entire row in a buffer

Page Mode

Row buffer acts like an SRAM
By changing column address, random bits can be accessed within a row.
DRAM Optimizations – Synchronous DRAM

Previously, DRAM had asynchronous interface
Each transfer involves handshaking with controller

Synchronous DRAM (SDRAM)
  Clock added to interface
  Register to hold number of bytes requested
    Send multiple bytes per request

Double Data Rate (DDR)
  Send data on rising and falling edge of clock
Consider a memory with these parameters:

- 1 cycle to send address
- 6 cycles to access each word
- 1 cycle to send word back to CPU/Cache

What's the miss penalty for a 4word block?

\[(1 + 6 \text{ cycles} + 1 \text{ cycle}) \times 4 \text{ words} \]
\[= 32 \text{ cycles} \]

How can we speed this up?
Wider Main Memory

Make the memory wider

Read out 2 (or more) words in parallel

Memory parameters:

1 cycle to send address
6 cycles to access each *doubleword*
1 cycle to send doubleword back to CPU/Cache

Miss penalty for a 4 word block:

\[(1 + 6 \text{ cycles} + 1 \text{ cycle}) \times 2 \text{ doublewords} \]
\[= 16 \text{ cycles}\]

Cost

Wider bus
Larger expansion size
Interleaved Main Memory

Organize memory in banks

Subsequent words map to different banks
Word A in bank \((A \mod M)\)
Within a bank, word A in location \((A \div M)\)

Word address

How many banks to include?
Virtual Memory

User operates in a virtual address space, mapping between virtual space and main memory is determined at runtime

Original Motivation

- Avoid overlays
- Use main memory as a cache for disk

Current motivation

- Relocation
- Protection
- Sharing
- Fast startup

Engineered differently than CPU caches

- Miss access time $O(1,000,000)$
- Miss access time $>>$ miss transfer time
Blocks, called *pages*, are 512 to 16K bytes.

**Page placement**
- Fully-associative -- avoid expensive misses

**Page identification**
- Address translation -- virtual to physical address
  - Indirection through one or two page tables
  - Translation cached in translation buffer

**Page replacement**
- Approx. LRU

**Write strategy**
- Writeback (with page dirty bit)
Logical Path

Two memory operations
Often two or three levels of page tables
TOO SLOW!
Address Translation

Fast Path

Translation Lookaside Buffer (TLB, TB)
A cache w/ PTEs for data
Number of entries 32 to 1024
Address Translation / Cache Interaction

Address Translation

Cache Lookup
Sequential TLB Access

Address translation before cache lookup

Problems
Slow
May increase cycle time, CPI, pipeline depth
Parallel TLB Access

Address translation in parallel with cache lookup
Parallel TLB Access

Address translation in parallel with cache lookup

Large Cache

VPN PO
IDX BO

TLB

PFN PO
TAG

read tags

Index taken from virtual page number
Virtual Address Synonyms
Solutions to Synonyms
Virtual Address Cache

Address translation after cache miss
  Implies fast lookup even for large caches

Must handle
  Virtual address synonyms (aliases)
  Virtual address space changes
  Status and protection bit changes
Protection

Goal:

One process should not be able to interfere with the execution of another

Process model

Privileged kernel
Independent user processes

Primitives vs. Policy

Architecture provides the primitives
Operating system implements the policy
Problems arise when hardware implements policy
Protection Primitives

User vs. Kernel

At least one privileged mode
Usually implemented as mode bit(s)

How do we switch to kernel mode?

Change mode and continue execution at predetermined location

Hardware to compare mode bits to access rights

Access certain resources only in kernel mode
Protection Primitives, cont.

Base and Bounds

- Privileged registers
- \( \text{Base} \leq \text{Address} \leq \text{Bounds} \)

Pagelevel protection

- Protection bits in page table entry
- Cache them in TLB