Main Memory

- Main memory generally utilizes Dynamic RAM (DRAM), which use a single transistor to store a bit, but require a periodic data refresh by reading every row (~every 8 msec).
- Static RAM may be used if the added expense, low density, power consumption, and complexity is feasible (e.g. Cray Vector Supercomputers).

- Main memory performance is affected by:
  - **Memory latency**: Affects cache miss penalty. Measured by:
    - **Access time**: The time it takes between a memory access request is issued to main memory and the time the requested information is available to cache/CPU.
    - **Cycle time**: The minimum time between requests to memory (greater than access time in DRAM to allow address lines to be stable)
  - **Memory bandwidth**: The sustained data transfer rate between main memory and cache/CPU.
Classic DRAM Organization

- **bit (data) lines**
- **Each intersection represents a 1-T DRAM Cell**
- **word (row) select**
- **Row and Column Address together:**
  - Select 1 bit a time

Row decoder

Column selector & I/O circuits

Column address

Data
Logical Diagram of A Typical DRAM

- Control Signals (RAS_L, CAS_L, WE_L, OE_L) are all active low

- Din and Dout are combined (D):
  - WE_L is asserted (Low), OE_L is disasserted (High)
    - D serves as the data input pin
  - WE_L is disasserted (High), OE_L is asserted (Low)
    - D is the data output pin

- Row and column addresses share the same pins (A)
  - RAS_L goes low: Pins A are latched in as row address
  - CAS_L goes low: Pins A are latched in as column address
Four Key DRAM Timing Parameters

- $t_{RAC}$: Minimum time from RAS (Row Access Strobe) line falling to the valid data output.
  - Usually quoted as the nominal speed of a DRAM chip
  - For a typical 4Mb DRAM $t_{RAC} = 60$ ns

- $t_{RC}$: Minimum time from the start of one row access to the start of the next.
  - $t_{RC} = 110$ ns for a 4Mbit DRAM with a $t_{RAC}$ of 60 ns

- $t_{CAC}$: minimum time from CAS (Column Access Strobe) line falling to valid data output.
  - 15 ns for a 4Mbit DRAM with a $t_{RAC}$ of 60 ns

- $t_{PC}$: minimum time from the start of one column access to the start of the next.
  - About 35 ns for a 4Mbit DRAM with a $t_{RAC}$ of 60 ns
DRAM Performance

• A 60 ns (t_{RAC}) DRAM chip can:
  – Perform a row access only every 110 ns (t_{RC})
  – Perform column access (t_{CAC}) in 15 ns, but time between column accesses is at least 35 ns (t_{PC}).

• In practice, external address delays and turning around buses make it 40 to 50 ns

• These times do not include the time to drive the addresses off the CPU or the memory controller overhead.
DRAM Write Timing

° Every DRAM access begins at:
  • The assertion of the RAS_L
  • 2 ways to write: early or late v. CAS

RAS_L

CAS_L

RAS_L CAS_L WE_L OE_L

A 9

256K x 8 DRAM

D 8

A Row Address Cbl Address Junk Row Address Cbl Address Junk

OE_L

WE_L

D Junk Data In Junk Data In Junk

WR Access Time

Early Wr Cycle: WE_L asserted before CAS_L

Late Wr Cycle: WE_L asserted after CAS_L
DRAM Read Timing

° Every DRAM access begins at:
  • The assertion of the RAS_L
  • 2 ways to read: early or late v. CAS

DRAM Read Cycle Time

RAS_L

CAS_L

A  Row Address  Col Address  Junk  Row Address  Col Address  Junk

WE_L

OE_L

D  High Z  Junk  Data Out  High Z  Data Out

Read Access Time

Early Read Cycle: OE_L asserted before CAS_L

Late Read Cycle: OE_L asserted after CAS_L
Page Mode DRAM: Motivation

- Regular DRAM Organization:
  - $N$ rows x $N$ column x M-bit
  - Read & Write M-bit at a time
  - Each M-bit access requires a RAS / CAS cycle

- Fast Page Mode DRAM
  - $N \times M$ “register” to save a row
Page Mode DRAM: Operation

- Fast Page Mode DRAM
  - \( N \times M \) “SRAM” to save a row

- After a row is read into the register
  - Only CAS is needed to access other M-bit blocks on that row
  - RAS_L remains asserted while CAS_L is toggled
Synchronous Dynamic RAM, SDRAM Organization
Memory Bandwidth Improvement Techniques

• **Wider Main Memory:**
  Memory width is increased to a number of words (usually the size of a second level cache block).
  ⇒ Memory bandwidth is proportional to memory width.
  
  e.g. Doubling the width of cache and memory doubles memory bandwidth

• **Simple Interleaved Memory:**
  Memory is organized as a number of banks each one word wide.
  – Simultaneous multiple word memory reads or writes are accomplished by sending memory addresses to several memory banks at once.
  – Interleaving factor: Refers to the mapping of memory addressees to memory banks.
  
  e.g. using 4 banks, bank 0 has all words whose address is:
  
  \[(\text{word address}) \mod 4 = 0\]
Three examples of bus width, memory width, and memory interleaving to achieve higher memory bandwidth

Simplest design: Everything is the width of one word

Wider memory, bus and cache

Narrow bus and cache with interleaved memory
Memory Interleaving

Access Pattern without Interleaving:

- D1 available
- Start Access for D1
- Start Access for D2

Access Pattern with 4-way Interleaving:

- Access Bank 0
- Access Bank 1
- Access Bank 2
- Access Bank 3

We can Access Bank 0 again
Four way interleaved memory

Three memory banks address interleaving:
- Sequentially interleaved addresses on the left, address requires a division
- Right: Alternate interleaving requires only modulo to a power of 2
Increasing the cache block size tends to decrease the miss rate due to increased use of spatial locality:

- Miss Rate Vs. Cache Block Size

### Graph Details
- **Y-axis**: Miss Rate (0%, 5%, 10%, 15%, 20%, 25%)
- **X-axis**: Block Size (bytes): 16, 32, 64, 128, 256
- **Legend**:
  - 1K
  - 4K
  - 16K
  - 64K
  - 256K
Memory Width, Interleaving: An Example

Given a base system with following parameters:

- Cache Block size = 1 word, Memory bus width = 1 word, Miss rate = 3%
- Miss penalty = 32 cycles, broken down as follows:
  - 4 cycles to send address, 24 cycles access time/word, 4 cycles to send a word
- Memory access/instruction = 1.2
- Ideal execution CPI (ignoring cache misses) = 2
- Miss rate (block size=2 word) = 2%
- Miss rate (block size=4 words) = 1%

- The CPI of the base machine with 1-word blocks = 2 + (1.2 x .03 x 32) = 3.15

- Increasing the block size to two words gives the following CPI:
  - 32-bit bus and memory, no interleaving = 2 + (1.2 x .02 x 2 x 32) = 3.54
  - 32-bit bus and memory, interleaved = 2 + (1.2 x .02 x (4 + 24 + 8)) = 2.86
  - 64-bit bus and memory, no interleaving = 2 + (1.2 x .02 x 1 x 32) = 2.77

- Increasing the block size to four words; resulting CPI:
  - 32-bit bus and memory, no interleaving = 2 + (1.2 x 1% x 4 x 32) = 3.54
  - 32-bit bus and memory, interleaved = 2 + (1.2 x 1% x (4 + 24 + 16)) = 2.53
  - 64-bit bus and memory, no interleaving = 2 + (1.2 x 2% x 2 x 32) = 2.77
Computer System Components

SDRAM
PC100/PC133
100-133MHZ
64-128 bits wide
2-way interleaved
~ 900 MBYTES/SEC

Double Date Rate (DDR) SDRAM
PC266
266MHZ
64-128 bits wide
4-way interleaved
~ 2.1 GBYTES/SEC
(second half 2000)

RAMbus DRAM (RDRAM)
400-800MHZ
16 bits wide
~ 1.6 GBYTES/SEC

CPU
500MHZ - 1GHZ

Caches

System Bus

Examples: Alpha, AMD K7: EV6, 200MHZ
Intel PII, PIII: GTL+ 100MHZ

Memory Controller

Memory

Controllers

I/O Buses

Examples: PCI, 33MHZ
32 bits wide 133 MBYTES/SEC

NICs

Networks

I/O Devices:

Disks
Displays
Keyboards

L1 ➔
L2 ➔
L3 ➔

L1 ➔
L2 ➔
L3 ➔

L1 ➔
L2 ➔
L3 ➔

L1 ➔
L2 ➔
L3 ➔

L1 ➔
L2 ➔
L3 ➔

L1 ➔
L2 ➔
L3 ➔

L1 ➔
L2 ➔
L3 ➔

L1 ➔
L2 ➔
L3 ➔
A Typical Memory Hierarchy

Processor

Control

Datapath

Registers

On-Chip Level One Cache

L1

Second Level Cache (SRAM)

L2

Main Memory (DRAM)

Virtual Memory, Secondary Storage (Disk)

Tertiary Storage (Tape)

Faster

Larger Capacity

Speed (ns): 1s 10s 100s 10,000,000s (10s ms) 10,000,000,000s (10s sec)

Size (bytes): 100s Ks Ms Gs Ts
Virtual Memory

- Virtual memory controls two levels of the memory hierarchy:
  - Main memory (DRAM)
  - Mass storage (usually magnetic disks)

- Main memory is divided into blocks allocated to different running processes in the system:
  - Fixed size blocks: Pages (size 4k to 64k bytes).
  - Variable size blocks: Segments (largest size 216 up to 232)

- At a given time, for any running process, a portion of its data/code is loaded in main memory while the rest is available only in mass storage.

- A program code/data block needed for process execution and not present in main memory results in a page fault (address fault) and the block has to be loaded into main main memory from disk by the OS handler.

- A program can be run in any location in main memory or disk by using a relocation mechanism controlled by the operating system which maps the address from the virtual address space (logical program address) to physical address space (main memory, disk).
Virtual Memory

Benefits

- Illusion of having more physical main memory
- Allows program relocation
- Protection from illegal memory access

Virtual address

<table>
<thead>
<tr>
<th>Virtual page number</th>
<th>Page offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27</td>
<td>...........</td>
</tr>
<tr>
<td>15 14 13 12</td>
<td>11 10 9 8</td>
</tr>
<tr>
<td>...........</td>
<td>...........</td>
</tr>
<tr>
<td>3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

Translation

<table>
<thead>
<tr>
<th>Physical page number</th>
<th>Page offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>29 28 27</td>
<td>...........</td>
</tr>
<tr>
<td>15 14 13 12</td>
<td>11 10 9 8</td>
</tr>
<tr>
<td>...........</td>
<td>...........</td>
</tr>
<tr>
<td>3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

Physical address
# Paging Versus Segmentation

<table>
<thead>
<tr>
<th></th>
<th>Page</th>
<th>Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Words per address</td>
<td>One</td>
<td>Two (segment and offset)</td>
</tr>
<tr>
<td>Programmer visible?</td>
<td>Invisible to application programmer</td>
<td>May be visible to application programmer</td>
</tr>
<tr>
<td>Replacing a block</td>
<td>Trivial (all blocks are the same size)</td>
<td>Hard (must find contiguous, variable-size, unused portion of main memory)</td>
</tr>
<tr>
<td>Memory use inefficiency</td>
<td>Internal fragmentation (unused portion of page)</td>
<td>External fragmentation (unused pieces of main memory)</td>
</tr>
<tr>
<td>Efficient disk traffic</td>
<td>Yes (adjust page size to balance access time and transfer time)</td>
<td>Not always (small segments may transfer just a few bytes)</td>
</tr>
</tbody>
</table>
Virtual → Physical Addresses Translation

Contiguous virtual address space of a program
Mapping Virtual Addresses to Physical Addresses Using A Page Table
Virtual Address Translation

Virtual page number

Page table
Physical page or disk address

Valid

Physical memory

Disk storage

Virtual page number

Page table
Physical page or disk address

Valid

Physical memory

Disk storage
Two memory accesses needed:
First to page table
Second to item
## Typical Parameter Range For Cache and Virtual Memory

<table>
<thead>
<tr>
<th>Parameter</th>
<th>First-level cache</th>
<th>Virtual memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block (page) size</td>
<td>16–128 bytes</td>
<td>4096–65,536 bytes</td>
</tr>
<tr>
<td>Hit time</td>
<td>1–2 clock cycles</td>
<td>40–100 clock cycles</td>
</tr>
<tr>
<td>Miss penalty</td>
<td>8–100 clock cycles</td>
<td>700,000–6,000,000 clock cycles</td>
</tr>
<tr>
<td>(Access time)</td>
<td>(6–60 clock cycles)</td>
<td>(500,000–4,000,000 clock cycles)</td>
</tr>
<tr>
<td>(Transfer time)</td>
<td>(2–40 clock cycles)</td>
<td>(200,000–2,000,000 clock cycles)</td>
</tr>
<tr>
<td>Miss rate</td>
<td>0.5–10%</td>
<td>0.00001–0.001%</td>
</tr>
<tr>
<td>Data memory size</td>
<td>0.016–1MB</td>
<td>16–8192 MB</td>
</tr>
</tbody>
</table>
Virtual Memory Issues/Strategies

- **Main memory block placement:** Fully associative placement is used to lower the miss rate.
- **Block replacement:** The least recently used (LRU) block is replaced when a new block is brought into main memory from disk.
- **Write strategy:** Write back is used and only those pages changed in main memory are written to disk (dirty bit scheme is used).
- To locate blocks in main memory a **page table** is utilized. The page table is indexed by the virtual page number and contains the physical address of the block.
  - In paging: Offset is concatenated to this physical page address.
  - In segmentation: Offset is added to the physical segment address.
- To limit the size of the page table to the number of physical pages in main memory a hashing scheme is used.
- Utilizing address locality, a **translation look-aside buffer (TLB)** is usually used to cache recent address translations and prevent a second memory access to read the page table.
Speeding Up Address Translation:
Translation Lookaside Buffer (TLB)

- TLB: A small on-chip fully-associative cache used for address translations.
- If a virtual address is found in TLB (a TLB hit), the page table in main memory is not accessed.
Operation of The Alpha AXP 21064
Data TLB During Address Translation

Virtual address

TLB = 32 blocks
Data cache = 256 blocks
TLB access is usually pipelined
TLB & Cache Operation

Virtual address

TLB access

TLB miss use page table

TLB hit?

Physical address

Cache operation

Write?

Write access bit on?

Write protection bit on?

Write data into cache, update the tag, and put the data and the address into the write buffer

Try to read data from cache

Cache miss stall

Cache hit?

Deliver data to the CPU

Cache is physically-addressed
### Event Combinations of Cache, TLB, Virtual Memory

<table>
<thead>
<tr>
<th>Cache</th>
<th>TLB</th>
<th>Virtual Memory</th>
<th>Possible?</th>
<th>When?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miss</td>
<td>Hit</td>
<td>Hit</td>
<td>Possible, no need to check page table</td>
<td></td>
</tr>
<tr>
<td>Hit</td>
<td>Miss</td>
<td>Hit</td>
<td>TLB miss, found in page table</td>
<td></td>
</tr>
<tr>
<td>Miss</td>
<td>Miss</td>
<td>Hit</td>
<td>TLB miss, cache miss</td>
<td></td>
</tr>
<tr>
<td>Miss</td>
<td>Miss</td>
<td>Miss</td>
<td>Page fault</td>
<td></td>
</tr>
<tr>
<td>Miss</td>
<td>Hit</td>
<td>Miss</td>
<td>Impossible, cannot be in TLB if not in memory</td>
<td></td>
</tr>
<tr>
<td>Hit</td>
<td>Hit</td>
<td>Miss</td>
<td>Impossible, cannot be in TLB or cache if not in memory</td>
<td></td>
</tr>
<tr>
<td>Hit</td>
<td>Miss</td>
<td>Miss</td>
<td>Impossible, cannot be in cache if not in memory</td>
<td></td>
</tr>
</tbody>
</table>