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.
Logical DRAM Organization (4 Mbit)

- Square root of bits per RAS/CAS
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 \( t_{RAC} = 60 \text{ ns} \)

- \( t_{RC} \): Minimum time from the start of one row access to the start of the next.
  - \( t_{RC} = 110 \text{ ns} \) 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

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

Diagram showing the timing and signals involved in a DRAM read operation. The signals include RAS_L, CAS_L, WE_L, OE_L, A, 9, D, 8, A (Row Address), Col Address, Junk, Row Address, Col Address, Junk, D (High Z), Junk, Data Out, High Z, Data Out, Read Access Time, Output Enable Delay, 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 x 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

![Diagram of Page Mode DRAM Operation]
Synchronous Dynamic RAM, SDRAM Organization
## DRAM “Near” Future: 1 Gbit Chips

<table>
<thead>
<tr>
<th></th>
<th>Mitsubishi</th>
<th>Samsung</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocks</td>
<td>512 x 2 Mbit</td>
<td>1024 x 1 Mbit</td>
</tr>
<tr>
<td>Clock</td>
<td>200 MHz</td>
<td>250 MHz</td>
</tr>
<tr>
<td>Data Pins</td>
<td>64</td>
<td>16</td>
</tr>
<tr>
<td>Die Size</td>
<td>24 x 24 mm</td>
<td>31 x 21 mm</td>
</tr>
<tr>
<td></td>
<td>Sizes will be much smaller in production</td>
<td></td>
</tr>
<tr>
<td>Metal Layers</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Technology</td>
<td>0.15 micron</td>
<td>0.16 micron</td>
</tr>
</tbody>
</table>
Memory Bandwidth Improvement Techniques

• **Wider Main Memory:**
  Memory width is increased to a number of words (usually the size of a 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:
    (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
Memory Width, Interleaving: An Example

Given the following system parameters with single cache level $L_1$:

Block size=1 word  Memory bus width=1 word  Miss rate =3%  Miss penalty=32 cycles
(4 cycles to send address  24 cycles access time/word,  4 cycles to send a word)
Memory access/instruction = 1.2  Ideal 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 = $2+(1.2 \times 3\% \times 32) = 3.15$

• Increasing the block size to two words gives the following CPI:
  – 32-bit bus and memory, no interleaving = $2 + (1.2 \times 2\% \times 2 \times 32) = 3.54$
  – 32-bit bus and memory, interleaved = $2 + (1.2 \times 2\% \times (4 + 24 + 8) = 2.86$
  – 64-bit bus and memory, no interleaving = $2 + (1.2 \times 2\% \times 1 \times 32) = 2.77$

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

CPU Core
- 600 MHZ - 1.2 GHZ
- 4-way Superscaler

RISC or RISC-core (x86):
- Deep Instruction Pipelines
- Dynamic scheduling
- Multiple FP, integer FUs
- Dynamic branch prediction
- Hardware speculation

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
- 2-way interleaved
- ~ 2.1 GBYTES/SEC (second half 2000)

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

I/O Devices:
- Memory
- Controllers
- NICs
- Networks
- Disks
- Displays
- Keyboards
- Networks

CPU
- L1 16-64K 1-2 way set associative (on chip), separate or unified
- L2 128K-1M 4-16 way set associative (on chip) unified
- L3 1-16M 8-16 way set associative (off chip) unified

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

Example: PCI, 33MHZ
- 32 bits wide
- 133 MBYTES/SEC
A Typical Memory Hierarchy

Control

Datapath

Processor

On-Chip Level

One Cache

L1

On-Chip Level

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

EECC551 - Shaaban
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 any 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 result in a page fault (address fault) and the block has to be loaded into main memory from disk.

• 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 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
## Paging Versus Segmentation

<table>
<thead>
<tr>
<th>Page</th>
<th>Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Words per address</td>
<td>One</td>
</tr>
<tr>
<td>Programmer visible?</td>
<td>Invisible to application programmer</td>
</tr>
<tr>
<td>Replacing a block</td>
<td>Trivial (all blocks are the same size)</td>
</tr>
<tr>
<td>Memory use inefficiency</td>
<td>Internal fragmentation (unused portion of page)</td>
</tr>
<tr>
<td>Efficient disk traffic</td>
<td>Yes (adjust page size to balance access time and transfer time)</td>
</tr>
</tbody>
</table>
Virtual $\rightarrow$ Physical Address Translation

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

Virtual address

Virtual page number
Page offset

Page table

Physical address

Main memory
Two memory accesses needed:

- First to page table.
- Second to item.
## Typical Parameter Range For Cache & 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.

![Diagram of TLB and address translation process]

**TLB (on-chip)**
128-256 Entries

**Virtual Page Number**

**Physical Page Address**

**Valid**

**Tag**

**Page Table (in main memory)**

**Physical Page or Disk Address**

**Physical Memory**

**Disk Storage**
Operation of The Alpha AXP 21064
Data TLB During Address Translation

Virtual address

Valid
Read Permission
Write Permission

TLB = 32 blocks
Data cache = 256 blocks

TLB access is usually pipelined
**TLB & Cache Operation**

- **Virtual address**
  - TLB access
  - TLB hit? (Yes)
    - Physical address
      - Write? (Yes)
        - Write access bit on? (Yes)
          - Write data into cache, update the tag, and put the data and the address into the write buffer
        - Write access bit on? (No)
          - Write protection exception
      - Write? (No)
        - Try to read data from cache
          - Cache hit? (Yes)
            - Deliver data to the CPU
          - Cache hit? (No)
            - Cache miss stall
  - TLB hit? (No)
    - TLB miss use page table

**Cache operation**
## 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>