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**
- **Column Selector & I/O Circuits**
- **Column Address**
- **row address**
- **row decoder**

° Row and Column Address together:
  - Select 1 bit a time
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

- $t_{\text{RAC}}$: 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 $t_{\text{RAC}} = 60$ ns

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

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

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

256K x 8 DRAM

RAS_L | CAS_L | WE_L | OE_L

A | 9

D | 8

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

```
RAS_L
CAS_L
A  Row Address  Col Address  Junk
WE_L
OE_L
D  High Z  Junk  Data Out
```

- DRAM Read Cycle Time
- Early Read Cycle: OE_L asserted before CAS_L
- Late Read Cycle: OE_L asserted after CAS_L

256K x 8 DRAM
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 x 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 addresses 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

#### Table: Address Interleaving

<table>
<thead>
<tr>
<th>Address</th>
<th>Bank 0</th>
<th>Address</th>
<th>Bank 1</th>
<th>Address</th>
<th>Bank 2</th>
<th>Address</th>
<th>Bank 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>5</td>
<td></td>
<td>6</td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>9</td>
<td></td>
<td>10</td>
<td></td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>13</td>
<td></td>
<td>14</td>
<td></td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

#### Four way interleaved memory

<table>
<thead>
<tr>
<th>Address within bank</th>
<th>Memory bank</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sequentially interleaved</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
</tr>
</tbody>
</table>

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:
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
- L1
- L2
- L3
- 500MHz - 1GHz

#### Caches

#### System Bus
- Examples: Alpha, AMD K7: EV6, 200MHz
- Intel PII, PIII: GTL+ 100MHz

#### Memory Controller

#### Memory

#### Controllers
- Disks
- Displays
- Keyboards

#### I/O Buses
- Example: PCI, 33MHz
  - 32 bits wide
  - 133 MBYTES/SEC

- NICs

#### Networks

#### I/O Devices:
A Typical Memory Hierarchy

- Processor
  - Control
  - Datapath
    - Registers
    - On-Chip Level One Cache $L_1$
    - Second Level Cache (SRAM) $L_2$

- Main Memory (DRAM)

- Virtual Memory, Secondary Storage (Disk)
  - Tertiary Storage (Tape)

- Speed (ns): 1s $\rightarrow$ 10s $\rightarrow$ 100s $\rightarrow$ 10,000,000s (10s ms) $\rightarrow$ 10,000,000,000s (10s sec)

- Size (bytes): 100s $\rightarrow$ Ks $\rightarrow$ Ms $\rightarrow$ Gs $\rightarrow$ Ts

- Faster
  - Larger Capacity
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 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

Virtual page number | Page offset

Translation

Physical page number | Page offset

Physical address
**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 → Physical Addresses Translation

Virtual memory

Contiguous virtual address space of a program

Physical location of blocks A, B, C

Disk

Physical main memory

Virtual address:

0
4K
8K
12K

Physical address:

0
4K
8K
12K
16K
20K
24K
28K

A
B
C
D

A
C
B
D
Mapping Virtual Addresses to Physical Addresses Using A Page Table
Virtual Address Translation

Virtual page number

Page table
Physical page or disk address

Valid

1 1 1 1 1 1
0 1 1 1 1
0 1 1 1
0 1
1

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.

<table>
<thead>
<tr>
<th>Virtual Page Number</th>
<th>Tag</th>
<th>Physical Page Address</th>
<th>TLB (on-chip)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>128-256 Entries</td>
</tr>
</tbody>
</table>

128-256 TLB Entries

Page Table (in main memory)

Valid  Tag  Physical Page Address  TLB (on-chip)  128-256 Entries

Physical Memory

Valid  Physical Page or Disk Address

Physical Page or Disk Address

Disk Storage
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

No

Yes

TLB hit?

No

TLB miss use page table

Yes

Physical address

Write?

No

Cache miss stall

Yes

Cache hit?

No

Try to read data from cache

Write access bit on?

No

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

Yes

Write protection bit on?

No

Deliver data to the CPU

Yes

Cache operation

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>