Mainstream Computer System Components
(Desktop/Low-end Server)

CPU Core
1 GHz - 3.8 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 (64bit)

Double Date Rate (DDR) SDRAM
PC3200
200 MHz DDR
64-128 bits wide
4-way interleaved
~3.2 GBYTES/SEC (one 64bit channel)
~6.4 GBYTES/SEC (two 64bit channels)

RAMbus DRAM (RDRAM)
400 MHz DDR
16 bits wide (32 banks)
~ 1.6 GBYTES/SEC

Current Standard

System Bus = CPU-Memory Bus = Front Side Bus (FSB)
The Memory Hierarchy: Main & Virtual Memory

- The Motivation for The Memory Hierarchy:
  - CPU/Memory Performance Gap
  - The Principle Of Locality

- Cache Concepts:
  - Organization, Replacement, Operation
  - Cache Performance Evaluation: Memory Access Tree

- Main Memory:
  - Performance Metrics: Latency & Bandwidth
    - Key DRAM Timing Parameters
  - DRAM System Memory Generations
  - Basic Techniques for Memory Bandwidth Improvement/Miss Penalty (M) Reduction

- Virtual Memory
  - Benefits, Issues/Strategies
  - Basic Virtual → Physical Address Translation: Page Tables
  - Speeding Up Address Translation: Translation Look-aside Buffer (TLB)

Cache exploits access locality to:
- Lower AMAT by hiding long main memory access latency.
- Lower demands on main memory bandwidth.

(In Chapter 7.3)

(In Chapter 7.4)
A Typical Memory Hierarchy

- **Faster**
- **Larger Capacity**

### Components:
- Processor
  - Control
  - Datapath
    - Registers
    - Level One Cache
      - \( L_1 \)
  - Level Two Cache
    - \( L_2 \)

### Memory Levels:
- Level One Cache (SRAM)
- Secondary Storage (Disk)
- Main Memory (DRAM)
- Virtual Memory, Secondary Storage (Disk)
- Tertiary Storage (Tape)

### Speed (ns):
- \(< 1s\)
- \(1s\)
- \(10s\)
- \(10,000,000s\) (10s ms)
- \(10,000,000,000s\) (10s sec)

### Size (bytes):
- \(100s\)
- \(Ks\)
- \(Ms\)
- \(Gs\)
- \(Ts\)
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 increasing cycle time.
- Static RAM may be used for main memory if the added expense, low density, high power consumption, and complexity is feasible (e.g. Cray Vector Supercomputers).
- Main memory performance is affected by:
  - Memory latency: Affects cache miss penalty, \( M \). Measured by:
    - Memory 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.
    - Memory Cycle time: The minimum time between requests to memory (greater than access time in DRAM to allow address lines to be stable)
  - Peak Memory bandwidth: The maximum sustained data transfer rate between main memory and cache/CPU.
    - In current memory technologies (e.g Double Data Rate SDRAM) published peak memory bandwidth does not take account most of the memory access latency.
    - This leads to achievable realistic memory bandwidth \(<\) peak memory bandwidth.
Typical DRAM access time = 80 ns or more (non ideal)

Logical Dynamic RAM (DRAM) Chip Organization
(16 Mbit)

Control Signals:
1 - Row Access Strobe (RAS): Low to latch row address
2 - Column Address Strobe (CAS): Low to latch column address
3 - Write Enable (WE) or Output Enable (OE)
4 - Wait for data to be ready

Basic Steps:
1 - Supply Row Address
2 - Supply Column Address
3 - Get Data

A periodic data refresh is required by reading every bit

D, Q share the same pins

(Single transistor per bit)

EECC550 - Shaaban
Four Key DRAM Timing Parameters

- **t\textsubscript{RAC}**: Minimum time from RAS (Row Access Strobe) line falling (activated) to the valid data output.
  - Used to be quoted as the nominal speed of a DRAM chip
  - For a typical 64Mb DRAM t\textsubscript{RAC} = 60 ns

- **t\textsubscript{RC}**: Minimum time from the start of one row access to the start of the next (memory cycle time).
  - t\textsubscript{RC} = t\textsubscript{RAC} + RAS Precharge Time
  - t\textsubscript{RC} = 110 ns for a 64Mbit DRAM with a t\textsubscript{RAC} of 60 ns

- **t\textsubscript{CAC}**: Minimum time from CAS (Column Access Strobe) line falling to valid data output.
  - 12 ns for a 64Mbit DRAM with a t\textsubscript{RAC} of 60 ns

- **t\textsubscript{PC}**: Minimum time from the start of one column access to the start of the next.
  - t\textsubscript{PC} = t\textsubscript{CAC} + CAS Precharge Time
  - About 25 ns for a 64Mbit DRAM with a t\textsubscript{RAC} of 60 ns
Simplified Asynchronous DRAM Read Timing

Memory Cycle Time = $t_{RC} = t_{RAC} + \text{RAS Precharge Time}$

$t_{RAC}$: Minimum time from RAS (Row Access Strobe) line falling to the valid data output.
$t_{RC}$: Minimum time from the start of one row access to the start of the next (memory cycle time).
$t_{CAC}$: Minimum time from CAS (Column Access Strobe) line falling to valid data output.
$t_{PC}$: Minimum time from the start of one column access to the start of the next.

Peak Memory Bandwidth = Memory bus width / Memory cycle time

Example: Memory Bus Width = 8 Bytes  Memory Cycle time = 200 ns
Peak Memory Bandwidth = $8 / 200 \times 10^{-9} = 40 \times 10^6$ Bytes/sec

Source: http://arstechnica.com/paedia/r/ram_guide/ram_guide.part2-1.html
Simplified DRAM Speed Parameters

- **Row Access Strobe (RAS)Time**: (similar to $t_{RAC}$):
  - Minimum time from RAS (Row Access Strobe) line falling (activated) to the first valid data output.
  - A major component of memory latency.
  - Only improves ~ 5% every year.

- **Column Access Strobe (CAS) Time/data transfer time**: (similar to $t_{CAC}$)
  - The minimum time required to read additional data by changing column address while keeping the same row address.
  - Along with memory bus width, determines peak memory bandwidth.

  - E.g For SDRAM Peak Memory Bandwidth = Bus Width /($0.5 \times t_{CAC}$)  
    For PC100 SDRAM Memory bus width = 8 bytes $t_{CAC} = 20$ns  
    Peak Bandwidth = $8 \times 100 \times 10^6 = 800 \times 10^6$ bytes/sec
#9 Lec # 9  Winter 2005  2-21-2006

## DRAM Generations

<table>
<thead>
<tr>
<th>Year</th>
<th>Size</th>
<th>RAS (ns)</th>
<th>CAS (ns)</th>
<th>Cycle Time</th>
<th>Memory Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1980</td>
<td>64 Kb</td>
<td>150-180</td>
<td>75</td>
<td>250 ns</td>
<td>Page Mode</td>
</tr>
<tr>
<td>1983</td>
<td>256 Kb</td>
<td>120-150</td>
<td>50</td>
<td>220 ns</td>
<td>Page Mode</td>
</tr>
<tr>
<td>1986</td>
<td>1 Mb</td>
<td>100-120</td>
<td>25</td>
<td>190 ns</td>
<td>Fast Page Mode</td>
</tr>
<tr>
<td>1989</td>
<td>4 Mb</td>
<td>80-100</td>
<td>20</td>
<td>165 ns</td>
<td>Fast Page Mode</td>
</tr>
<tr>
<td>1992</td>
<td>16 Mb</td>
<td>60-80</td>
<td>15</td>
<td>120 ns</td>
<td>EDO</td>
</tr>
<tr>
<td>1996</td>
<td>64 Mb</td>
<td>50-70</td>
<td>12</td>
<td>110 ns</td>
<td>PC66 SDRAM</td>
</tr>
<tr>
<td>1998</td>
<td>128 Mb</td>
<td>50-70</td>
<td>10</td>
<td>100 ns</td>
<td>PC100 SDRAM</td>
</tr>
<tr>
<td>2000</td>
<td>256 Mb</td>
<td>45-65</td>
<td>7</td>
<td>90 ns</td>
<td>PC133 SDRAM</td>
</tr>
<tr>
<td>2002</td>
<td>512 Mb</td>
<td>40-60</td>
<td>5</td>
<td>80 ns</td>
<td>PC2700 DDR SDRAM</td>
</tr>
</tbody>
</table>

- **Asynchronous DRAM**
- **Synchronous DRAM**

8000:1 (Capacity)  
15:1 (~bandwidth)  
3:1 (Latency)  

A major factor in cache miss penalty M  

PC3200 DDR (2003)  
DDR2 SDRAM (2004)
Asynchronous DRAM:

Page Mode DRAM (Early 80s)

- 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

![Diagram of Page Mode DRAM](image)
Asynchronous DRAM:

Fast Page Mode DRAM (FPM)

- **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

- The first “burst mode” DRAM

![Diagram of Fast Page Mode DRAM](image)

A read burst of length 4 shown
Simplified Asynchronous Fast Page Mode (FPM) DRAM Read Timing

Typical timing at 66 MHz: 5-3-3-3 (burst of length 4)
For bus width = 64 bits = 8 bytes cache block size = 32 bytes
It takes = 5 + 3 + 3 + 3 = 14 memory cycles or 15 ns x 14 = 210 ns to read 32 byte block
Miss penalty for CPU running at 1 GHz = M = 15 x 14 = 210 CPU cycles

One memory cycle at 66 MHz = 1000/66 = 15 CPU cycles at 1 GHz
Simplified Asynchronous Extended Data Out (EDO) DRAM Read Timing

- Extended Data Out DRAM operates in a similar fashion to Fast Page Mode DRAM except putting data from one read on the output pins at the same time the column address for the next read is being latched in.

**EDO Read**

<table>
<thead>
<tr>
<th>RAS Active</th>
<th>Pre</th>
<th>Pre</th>
<th>Pre</th>
<th>Pre</th>
<th>Pre</th>
<th>Pre</th>
<th>Pre</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAS Precharge</td>
<td>CAS</td>
<td>Pre</td>
<td>Pre</td>
<td>Pre</td>
<td>Pre</td>
<td>Pre</td>
<td>Pre</td>
</tr>
<tr>
<td>Address Bus Row</td>
<td>Col. 1</td>
<td>Col. 2</td>
<td>Col. 3</td>
<td>Col. 4</td>
<td>Col. 5</td>
<td>Col. 6</td>
<td></td>
</tr>
<tr>
<td>WE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Bus tRAC</td>
<td>tCAC</td>
<td>Data 1</td>
<td>D2</td>
<td>D3</td>
<td>D4</td>
<td>D5</td>
<td></td>
</tr>
</tbody>
</table>

EDO DRAM speed rated using tRAC ~ 40-60ns

Typical timing at 66 MHz: 5-2-2-2 (burst of length 4)

For bus width = 64 bits = 8 bytes  Max. Bandwidth = 8 x 66 / 2 = 264 Mbytes/sec

It takes = 5+2+2+2 = 11 memory cycles or 15 ns x 11 = 165 ns to read 32 byte cache block

Minimum Read Miss penalty for CPU running at 1 GHz = M = 11 x 15 = 165 CPU cycles

One memory cycle at 66 MHz = 1000/66 = 15 CPU cycles at 1 GHz

Basic Memory Bandwidth Improvement/Miss Penalty (M) Reduction Techniques

• **Wider Main Memory (CPU-Memory Bus):**
  Memory bus width is increased to a number of words (usually up to the size of a cache block).
  – Memory bandwidth is proportional to memory bus width.
    • e.g. Doubling the width of cache and memory doubles potential memory bandwidth available to the CPU.
    – The miss penalty is reduced since fewer memory bus accesses are needed to fill a cache block on a miss.

• **Interleaved (Multi-Bank) Memory:**
  Memory is organized as a number of independent banks.
  – Multiple interleaved memory reads or writes are accomplished by sending memory addresses to several memory banks at once or pipeline access to the banks.
  – **Interleaving factor:** Refers to the mapping of memory addresses to memory banks. **Goal reduce bank conflicts.**
    e.g. using 4 banks (width one word), 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 (lowest performance)

Wider memory, bus and cache (highest performance)

Narrow bus and cache with interleaved memory banks
Four Way (Four Banks) Interleaved Memory

Memory Bank Number

<table>
<thead>
<tr>
<th>Bank 0</th>
<th>Bank 1</th>
<th>Bank 2</th>
<th>Bank 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
</tbody>
</table>

Bank Width = One Word
Bank Number = (Word Address) Mod (4)
Memory Bank Interleaving

Can be applied at: 1- DRAM chip level (e.g. SDRAM, DDR) 2- DRAM module level 3- DRAM channel level

Access Pattern without Interleaving: (One Bank)

- Memory Bank Cycle Time
- D1 available
- Start Access for D1
- Start Access for D2

Very long memory bank recovery time shown here

Pipeline access to different memory banks to increase effective bandwidth

Access Pattern with 4-way Interleaving:

- Memory Bank Cycle Time

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

We can Access Bank 0 again

Number of banks \( \geq \) Number of cycles to access word in a bank

Bank interleaving does not reduce latency of accesses to the same bank
## Synchronous DRAM Characteristics Summary

<table>
<thead>
<tr>
<th></th>
<th>SDRAM</th>
<th>DDR (Double Data Rate) SDRAM</th>
<th>RAMbus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Potential Bandwidth</strong></td>
<td>PC100</td>
<td>DDR266 (PC2100)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.8 GB/s</td>
<td>2.133 GB/s</td>
<td>1.6 GB/s</td>
</tr>
<tr>
<td></td>
<td>.1 x 8 = .8</td>
<td>.133 x 2 x 8 = 2.1</td>
<td>.4 x 2 x 2 = 1.6</td>
</tr>
<tr>
<td><strong>Interface Signals</strong></td>
<td>64(72) data</td>
<td>64(72) data</td>
<td>16(18) data</td>
</tr>
<tr>
<td></td>
<td>168 pins</td>
<td>168 pins</td>
<td>184 pins</td>
</tr>
<tr>
<td><strong>Interface Frequency</strong></td>
<td>100 MHz</td>
<td>133 MHz</td>
<td>400MHz</td>
</tr>
<tr>
<td><strong>Latency Range</strong></td>
<td>30-90 nS</td>
<td>18.8-64 nS</td>
<td>35-80 nS</td>
</tr>
<tr>
<td><strong># of Banks per DRAM Chip</strong></td>
<td>2</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td><strong>Bus Width Bytes</strong></td>
<td>8</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

The latencies given only account for memory module latency and do not include memory controller latency or other address/data line delays. Thus realistic access latency is longer.
**Synchronous Dynamic RAM, (SDRAM) (mid 90s)**

**Organization**

SDRAM speed is rated at max.
clock speed supported:
100MHz = PC100
133MHz = PC133

**DDR SDRAM**

(late 90s - current)

organization is similar but **four banks** are used in each DDR SDRAM chip instead of two.

Data transfer on both **rising and falling edges of the clock**

DDR SDRAM rated by maximum memory bandwidth
PC3200 = 8 bytes x 200 MHz x 2
= 3200 Mbytes/sec

**SDRAM**

Peak Memory Bandwidth =
= Bus Width / (0.5 x \( t_{CAC} \))
= Bus Width x Clock rate

**DDR SDRAM**

Peak Memory Bandwidth =
= Bus Width / (0.25 x \( t_{CAC} \))
= Bus Width x Clock rate x 2
Simplified SDRAM/DDR SDRAM Read Timing

SDRAM

Typical timing at 133 MHz (PC133 SDRAM) : 5-1-1-1
For bus width = 64 bits = 8 bytes
Max. Bandwidth = 133 x 8 = 1064 Mbytes/sec
It takes = 5+1+1+1 = 8 memory cycles or 7.5 ns x 8 = 60 ns to read 32 byte cache block
Minimum Read Miss penalty for CPU running at 1 GHz = $M = 7.5 \times 8 = 60$ CPU cycles

DDR SDRAM:
Possible timing at 133 MHz (DDR x2) (PC2100 DDR SDRAM) : 5-.5-.5-.5
For bus width = 64 bits = 8 bytes
Max. Bandwidth = 133 x 2 x 8 = 2128 Mbytes/sec
It takes = 5+.5+.5+.5 = 6.5 memory cycles or 7.5 ns x 6.5 = 49 ns to read 32 byte cache block
Minimum Read Miss penalty for CPU running at 1 GHz = $M = 7.5 \times 6.5 = 49$ CPU cycles

In this example for SDRAM: $M = 60$ cycles for DDR SDRAM: $M = 49$ cycles
Thus accounting for access latency DDR is $60/49 = 1.22$ times faster
Not twice as fast ($2128/1064 = 2$) as indicated by peak bandwidth!
The Impact of Larger Cache Block Size on Miss Rate

- A larger cache block size improves cache performance by taking better advantage of spatial locality. However, for a fixed cache size, larger block sizes mean fewer cache block frames.

- Performance keeps improving to a limit when the fewer number of cache block frames increases conflicts and thus overall cache miss rate.

For SPEC92

- Improves spatial locality reducing compulsory misses.
Memory Width, Interleaving: Performance Example

Given the following system parameters with single unified cache level L1 (ignoring write policy):

Block size = 1 word  Memory bus width = 1 word  Miss rate = 3%  M = Miss penalty = 32 cycles
(4 cycles to send address  24 cycles access time,  4 cycles to send a word to CPU)

<table>
<thead>
<tr>
<th>Block Size</th>
<th>CPI Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 word</td>
<td>CPI = 2 + (1.2 x 0.03 x 32) = 3.15</td>
</tr>
<tr>
<td>2 words</td>
<td>CPI = 2 + (1.2 x 0.02 x 36) = 2.86</td>
</tr>
<tr>
<td>4 words</td>
<td>CPI = 2 + (1.2 x 0.01 x 36) = 2.43</td>
</tr>
</tbody>
</table>

Increasing the block size to two words (64 bits) gives the following CPI:

<table>
<thead>
<tr>
<th>Block Size</th>
<th>CPI Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 words</td>
<td>CPI = 2 + (1.2 x 0.02 x 64) = 3.54</td>
</tr>
<tr>
<td>4 words</td>
<td>CPI = 2 + (1.2 x 0.01 x 44) = 2.53</td>
</tr>
<tr>
<td>8 words</td>
<td>CPI = 2 + (1.2 x 0.01 x 32) = 2.38</td>
</tr>
</tbody>
</table>

Increasing the block size to four words (128 bits); resulting CPI:

<table>
<thead>
<tr>
<th>Block Size</th>
<th>CPI Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 words</td>
<td>CPI = 2 + (1.2 x 0.01 x 128) = 3.54</td>
</tr>
<tr>
<td>8 words</td>
<td>CPI = 2 + (1.2 x 0.01 x 44) = 2.53</td>
</tr>
<tr>
<td>16 words</td>
<td>CPI = 2 + (1.2 x 0.01 x 32) = 2.38</td>
</tr>
</tbody>
</table>

Miss Penalty = M = Number of CPU stall cycles for an access missed in cache and satisfied by main memory.
**X86 CPU Dual Channel PC3200 DDR SDRAM**

**Sample (Realistic?) Bandwidth Data**

<table>
<thead>
<tr>
<th>Processor</th>
<th>Write</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Athlon 64 FX-51</td>
<td>1937.2</td>
<td>3621.5</td>
</tr>
<tr>
<td>Pentium 4 3.2GHz</td>
<td>1663.4</td>
<td>3507.9</td>
</tr>
<tr>
<td>P4 3.2GHz Extreme Edition</td>
<td>1953.9</td>
<td>3468.7</td>
</tr>
<tr>
<td>Opteron 146</td>
<td>1903.8</td>
<td>3441.9</td>
</tr>
<tr>
<td>Pentium 4 'C' 2.8GHz</td>
<td>1524.4</td>
<td>3351.1</td>
</tr>
<tr>
<td>Pentium 4 'C' 2.4GHz</td>
<td>1511.3</td>
<td>3163.7</td>
</tr>
<tr>
<td>Athlon S4 3400+</td>
<td>1252.4</td>
<td>2892.7</td>
</tr>
<tr>
<td>Athlon S4 3200+</td>
<td>1251.2</td>
<td>2878.4</td>
</tr>
<tr>
<td>Athlon S4 3000+</td>
<td>1285.9</td>
<td>2875.9</td>
</tr>
<tr>
<td>Athlon XP 3200+</td>
<td>1862.0</td>
<td>4282.7</td>
</tr>
<tr>
<td>Athlon XP 2000+</td>
<td>1203.1</td>
<td>2406.2</td>
</tr>
<tr>
<td>Athlon XP 2500+</td>
<td>1464.8</td>
<td>3020.8</td>
</tr>
</tbody>
</table>

Dual (64-bit) Channel PC3200 DDR SDRAM has a theoretical peak bandwidth of

$$400 \text{ MHz} \times 8 \text{ bytes} \times 2 = 6400 \text{ MB/s}$$

**Is memory bandwidth still an issue?**

Source: The Tech Report 1-21-2004
X86 CPU Dual Channel PC3200 DDR SDRAM

Sample (Realistic?) Latency Data

PC3200 DDR SDRAM has a theoretical latency range of 18-40 ns (not accounting for memory controller latency or other address/data line delays).

Is memory latency still an issue?

X86 CPU Cache/Memory Performance Example:
AMD Athlon XP/64/FX Vs. Intel P4/Extreme Edition

Main Memory: Dual (64-bit) Channel PC3200 DDR SDRAM
peak bandwidth of 6400 MB/s

Source: The Tech Report 1-21-2004
A Typical Memory Hierarchy

Larger Capacity → Faster

Processor

Control

Datapath

Registers

Level One Cache (SRAM) L₁

Second Level Cache (SRAM) L₂

Main Memory (DRAM)

Managed by Hardware

Virtual Memory, Secondary Storage (Disk)

Managed by OS with hardware assistance

Tertiary Storage (Tape)

Speed (ns):

< 1s 1s 10s

Size (bytes):

100s Ks Ms Gs Ts

Virtual Memory: Chapter 7.4
Virtual Memory: Overview

- 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 by the OS:
  - **Fixed size blocks:** Pages (size 4k to 64k bytes). (Most common)
  - **Variable size blocks:** Segments (largest size $2^{16}$ up to $2^{32}$).
  - **Paged segmentation:** Large variable/fixed size segments divided into a number of fixed size pages (X86, PowerPC).

- At any given time, for any running process, a portion of its data/code is loaded (allocated) 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 page has to be loaded into main memory by the OS from disk (demand paging).

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

• **Original Motivation:**
  – Illusion of having more physical main memory (using demand paging)
  – Allows program and data address relocation by automating the process of code and data movement between main memory and secondary storage.

• **Additional Current Motivation:**
  – Fast process start-up.
  – **Protection** from illegal memory access.
    • Needed for multi-tasking operating systems.
  – Controlled code and data sharing among processes.
    • Needed for multi-threaded programs.
  – Uniform data access
    • Memory-mapped files
    • Memory-mapped network communication
Paging Versus Segmentation

Fixed-size blocks (pages)

Paging

Segmentation

Variable-size blocks (segments)

<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 memory stores only the most often used portions of a process address space in main memory and retrieves other portions from a disk as needed (demand paging).

The virtual-memory space is divided into pages identified by virtual page numbers (VPNs), shown on the far left, which are mapped to page frames or physical page numbers (PPNs) or page frame numbers (PFNs), in physical memory as shown on the right.

Virtual Address Space Vs. Physical Address Space
(logical)

Virtual address to physical address mapping or translation
Using a page table

Paging is assumed here

Virtual Address Space = Process Logical Address Space
Basic Virtual Memory Management

- Operating system makes decisions regarding which virtual (logical) pages of a process should be allocated in real physical memory and where (demand paging) assisted with hardware Memory Management Unit (MMU)
- On memory access -- If no valid virtual page to physical page translation (i.e page not allocated in main memory)
  - Page fault to operating system (e.g system call to handle page fault))
  - Operating system requests page from disk
  - Operating system chooses page for replacement
    - writes back to disk if modified
  - Operating system allocates a page in physical memory and updates page table w/ new page table entry (PTE).

Paging is assumed

Then restart faulting process
## 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 ( M )</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>

- Program assumed in steady state
- Paging is assumed here

- i.e page fault
Virtual Memory Basic Strategies

• **Main memory page placement(allocation):** Fully associative placement or allocation (by OS) is used to lower the miss rate.

• **Page replacement:** The least recently used (LRU) page is replaced when a new page 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).

• **Page Identification and address translation:** To locate pages in main memory a page table is utilized to translate from virtual page numbers (VPNs) to physical page numbers (PPNs). The page table is indexed by the virtual page number and contains the physical address of the page.
  – **In paging:** Offset is concatenated to this physical page address.
  – **In segmentation:** Offset is added to the physical segment address.

• Utilizing address translation locality, a translation look-aside buffer (TLB) is usually used to cache recent address translations (PTEs) and prevent a second memory access to read the page table.

PTE = Page Table Entry
Virtual \(\rightarrow\) Physical Address Translation

Contiguous virtual address (or logical) space of a program

Virtual address:  
0  
4K  
8K  
12K

Physical address:  
0  
4K  
8K  
12K  
16K  
20K  
24K  
28K

Virtual memory

Virtual address to physical address translation using page table

Page Fault: D in Disk  
(not allocated in main memory)  
OS allocates a page in physical main memory

Paging is assumed

Physical location of blocks A, B, C

Disk

EECC550 - Shaaban
Virtual to Physical Address Translation:

Page Tables

- Mapping information from virtual page numbers (VPNs) to physical page numbers is organized into a page table which is a collection of page table entries (PTEs).
- At the minimum, a PTE indicates whether its virtual page is in memory, on disk, or unallocated and the PPN (or PFN) if the page is allocated.
- Over time, virtual memory evolved to handle additional functions including data sharing, address-space protection and page level protection, so a typical PTE now contains additional information including:
  - A valid bit, which indicates whether the PTE contains a valid translation;
  - The page’s location in memory (page frame number, PFN) or location on disk (for example, an offset into a swap file);
  - The ID of the page’s owner (the address-space identifier (ASID), sometimes called Address Space Number (ASN) or access key);
  - The virtual page number (VPN);
  - A reference bit, which indicates whether the page was recently accessed;
  - A modify bit, which indicates whether the page was recently written; and
  - Page-protection bits, such as read-write, read only, kernel vs. user, and so on.
Basic Mapping Virtual Addresses to Physical Addresses Using A Direct Page Table

Paging is assumed
Virtual to Physical Address Translation

Virtual address

31 30 29 28 27 ............ 15 14 13 12 11 10 9 8 ........ 3 2 1 0

Virtual page number (VPN) | Page offset

PTE (Page Table Entry)

Translation

Physical page number (PPN) | Page offset

Physical address

physical page numbers (PPN) or page frame numbers (PFN)

Paging is assumed

Here page size = $2^{12} = 4096$ bytes = 4K bytes

Cache is normally designed to be physically addressed

EECC550 - Shaaban
Direct Page Table Organization

Two memory accesses needed:
- **First** to page table.
- **Second** to item.

- Page table usually in main memory.

How to speedup virtual to physical address translation?

Paging is assumed

Cache is normally designed to be physically addressed

---

Two memory accesses needed:
- **First** to page table.
- **Second** to item.

Page table usually in main memory.

How to speedup virtual to physical address translation?

Paging is assumed

Cache is normally designed to be physically addressed
Virtual Address Translation Using
A Direct Page Table

Paging is assumed
Speeding Up Address Translation:
Translation Lookaside Buffer (TLB)

- Translation Lookaside Buffer (TLB): Utilizing address reference locality, a small on-chip cache that contains recent address translations (PTEs).
  - TLB entries usually 32-128
  - High degree of associativity usually used
  - Separate instruction TLB (I-TLB) and data TLB (D-TLB) are usually used.
  - A unified larger second level TLB is often used to improve TLB performance and reduce the associativity of level 1 TLBs.

- If a virtual address is found in TLB (a TLB hit), the page table in main memory is not accessed.

- TLB-Refill: If a virtual address is not found in TLB, a TLB miss (TLB fault) occurs and the system must search (walk) the page table for the appropriate entry and place it into the TLB this is accomplished by the TLB-refill mechanism.

- Types of TLB-refill mechanisms:
  - Hardware-managed TLB: A hardware finite state machine is used to refill the TLB on a TLB miss by walking the page table. (PowerPC, IA-32)
  - Software-managed TLB: TLB refill handled by the operating system. (MIPS, Alpha, UltraSPARC, HP PA-RISC, …)
Speeding Up Address Translation:

Translation Lookaside Buffer (TLB)

- TLB: A small on-chip cache that contains recent address translations (PTEs).
- If a virtual address is found in TLB (a TLB hit), the page table in main memory is not accessed.

Paging is assumed

TLB (on-chip) 32-128 Entries

Single-level Unified TLB shown
Operation of The Alpha 21264 Data TLB (DTLB) During Address Translation

Virtual address

Address Space Number (ASN) Identifies process similar to PID (no need to flush TLB on context switch)

Protection Permissions Valid bit

Address space number (VPN) Virtual page number <8>  Page offset <13>

8Kbytes pages

DTLB = 128 entries

PID = Process ID  PTE = Page Table Entry

EECC550 - Shaaban
Basic TLB & Cache Operation

TLB Operation

Virtual address

TLB access

TLB miss use page table

TLB hit?

Yes

No

TLB Refill

Normal Cache operation

Try to read data from cache

Cache miss stall

Cache hit?

Yes

No

Physical address

Write?

No

Yes

Write access bit on?

No

Write protection exception

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

Deliver data to the CPU

Cache is usually physically-addressed
CPU Performance with Real TLBs

When a real TLB is used with a TLB miss rate and a TLB miss penalty (time needed to refill the TLB) is used:

\[
CPI = CPI_{\text{execution}} + \text{mem stalls per instruction} + \text{TLB stalls per instruction}
\]

Where:

- Mem Stalls per instruction = Mem accesses per instruction \times \text{mem stalls per access}
- TLB Stalls per instruction = Mem accesses per instruction \times \text{TLB stalls per access} \times \text{TLB miss rate} \times \text{TLB miss penalty}

Example: (For unified single-level TLB)

Given: CPI_{\text{execution}} = 1.3 \quad \text{Mem accesses per instruction} = 1.4
\text{Mem stalls per access} = 0.5 \quad \text{TLB miss rate} = 0.3\% \quad \text{TLB miss penalty} = 30 \text{ cycles}

What is the resulting CPU CPI?

- Mem Stalls per instruction = 1.4 \times 0.5 = 0.7 \text{ cycles/instruction}
- TLB stalls per instruction = 1.4 \times (0.3\% \times 30) = 0.126 \text{ cycles/instruction}

\[
CPI = 1.3 + 0.7 + 0.126 = 2.126
\]
### 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>Hit</td>
<td>Hit</td>
<td>Hit</td>
<td>TLB/Cache Hit</td>
<td></td>
</tr>
<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 main 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 main 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>