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.

• 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:
    • **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 maximum sustained data transfer rate between main memory and cache/CPU.

(In Chapter 5.8 - 5.10)
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 64Mb DRAM $t_{RAC} = 60$ ns

- $t_{RC}$: Minimum time from the start of one row access to the start of the next (memory cycle time).
  - $t_{RC} = 110$ ns for a 64Mbit DRAM with a $t_{RAC}$ of 60 ns

- $t_{CAC}$: Minimum time from CAS (Column Access Strobe) line falling to valid data output.
  - 12 ns for a 64Mbit 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 25 ns for a 64Mbit DRAM with a $t_{RAC}$ of 60 ns
Simplified DRAM Speed Parameters

• **Row Access Strobe (RAS)Time:** (similar to $t_{RAC}$)
  – Minimum time from RAS (Row Access Strobe) line falling 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.
## 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></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>40-65</td>
<td>5</td>
<td>80 ns</td>
<td>PC2700 DDR SDRAM</td>
</tr>
</tbody>
</table>

- **Capacity**: 8000:1
- **Bandwidth**: 15:1
- **Latency**: 3:1
Simplified Asynchronous DRAM Read Timing

Source: http://arstechnica.com/paedia/r/ram_guide/ram_guide.part2-1.html

(late 70s)
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
Simplified Asynchronous Fast Page Mode (FPM) DRAM Read Timing

Typical timing at 66 MHz: 5-3-3-3
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
Read Miss penalty for CPU running at 1 GHz = M = 15 x 14 = 210 CPU cycles

FPM DRAM speed rated using tRAC ~ 50-70ns (late 80s)
Extended Data Out DRAM operates in a similar fashion to Fast Page Mode DRAM except the data from one read is on the output pins at the same time the column address for the next read is being latched in.

**EDO Read**

EDO DRAM speed rated using $t_{RAC} \sim 40-60\text{ns}$

Typical timing at 66 MHz: 5-2-2-2

For bus width = 64 bits = 8 bytes  Max. Bandwidth = $8 \times 66 / 2 = 264 \text{ Mbytes/sec}$

It takes = $5+2+2+2 = 11$ memory cycles or $15 \text{ ns} \times 11 = 165 \text{ ns}$ to read 32 byte cache block

Minimum Read Miss penalty for CPU running at 1 GHz = $M = 11 \times 15 = 165 \text{ CPU cycles}$

**Source:** http://arstechnica.com/paedia/r/ram_guide/ram_guide.part2-1.html
### Synchronous DRAM Interface Characteristics Summary

<table>
<thead>
<tr>
<th></th>
<th>SDRAM</th>
<th>DDR SDRAM</th>
<th>RAMbus</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>PC100</strong></td>
<td><strong>DDR266</strong> (PC2100)</td>
<td><strong>DDR2</strong> (Late 2003)</td>
</tr>
<tr>
<td>Potential Bandwidth</td>
<td>0.8 GB/s</td>
<td>2.133 GB/s</td>
<td>3.2 GB/s</td>
</tr>
<tr>
<td></td>
<td>(Similar to PC3200)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface Signals</td>
<td>64(72) data</td>
<td>64(72) data</td>
<td>64(72) data</td>
</tr>
<tr>
<td></td>
<td>168 pins</td>
<td>168 pins</td>
<td>184 pins</td>
</tr>
<tr>
<td>Interface Frequency</td>
<td>100 MHz</td>
<td>133 MHz</td>
<td>200 MHz</td>
</tr>
<tr>
<td>Latency Range</td>
<td>30-90 nS</td>
<td>18.8-64 nS</td>
<td>17.5-42.6 nS</td>
</tr>
<tr>
<td># of Banks per DRAM Chip</td>
<td>2</td>
<td>4</td>
<td>4?</td>
</tr>
</tbody>
</table>

- **DDR SDRAM** is similar to **PC3200**.
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
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 x 8 = 60 CPU cycles
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:
    
    \[(\text{word address mod } 4) = 0\]
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

Number of banks ≥ Number of cycles to access word in a bank
Memory Width, Interleaving: Performance Example

Given the following system parameters with single unified cache level L\textsubscript{1} (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)

Memory access/instruction = 1.2

$\text{CPI}_{\text{execution}}$ (ignoring cache misses) = 2

Miss rate (block size = 2 word = 8 bytes) = 2%

Miss rate (block size = 4 words = 16 bytes) = 1%

- The CPI of the base machine with 1-word blocks = $2 + (1.2 \times 0.03 \times 32) = 3.15$

Increasing the block size to two words gives the following CPI:

- 32-bit bus and memory, no interleaving,  
  $M = 2 \times 32 = 64$ cycles 
  $\text{CPI} = 2 + (1.2 \times 0.02 \times 64) = 3.54$

- 32-bit bus and memory, interleaved, 
  $M = 4 + 24 + 8 = 36$ cycles 
  $\text{CPI} = 2 + (1.2 \times 0.02 \times 36) = 2.86$

- 64-bit bus and memory, no interleaving,  
  $M = 32$ cycles 
  $\text{CPI} = 2 + (1.2 \times 0.02 \times 32) = 2.77$

Increasing the block size to four words; resulting CPI:

- 32-bit bus and memory, no interleaving,  
  $M = 4 \times 32 = 128$ cycles 
  $\text{CPI} = 2 + (1.2 \times 0.01 \times 128) = 3.54$

- 32-bit bus and memory, interleaved,  
  $M = 4 + 24 + 16 = 44$ cycles 
  $\text{CPI} = 2 + (1.2 \times 0.01 \times 44) = 2.53$

- 64-bit bus and memory, no interleaving,  
  $M = 2 \times 32 = 64$ cycles 
  $\text{CPI} = 2 + (1.2 \times 0.01 \times 64) = 2.77$

- 64-bit bus and memory, interleaved,  
  $M = 4 + 24 + 8 = 36$ cycles 
  $\text{CPI} = 2 + (1.2 \times 0.01 \times 36) = 2.43$

- 128-bit bus and memory, no interleaving,  
  $M = 32$ cycles 
  $\text{CPI} = 2 + (1.2 \times 0.01 \times 32) = 2.38$
Three-Level Cache Example

- CPU with $\text{CPI}_{\text{execution}} = 1.1$ running at clock rate = 500 MHz
- 1.3 memory accesses per instruction.
- $L_1$ cache operates at 500 MHz with a miss rate of 5%
- $L_2$ cache operates at 250 MHz with a local miss rate 40%, $(T_2 = 2$ cycles$)$
- $L_3$ cache operates at 100 MHz with a local miss rate 50%, $(T_3 = 5$ cycles$)$
- Memory access penalty, $M = 100$ cycles. Find CPI.

With No Cache, $\text{CPI} = 1.1 + 1.3 \times 100 = 131.1$

With single $L_1$, $\text{CPI} = 1.1 + 1.3 \times 0.05 \times 100 = 7.6$

With $L_1$, $L_2$ $\text{CPI} = 1.1 + 1.3 \times (0.05 \times 0.6 \times 2 + 0.05 \times 0.4 \times 100) = 3.778$

$$\text{CPI} = \text{CPI}_{\text{execution}} + \text{Mem Stall \ cycles per instruction}$$

Mem Stall cycles per instruction = Mem accesses per instruction $\times$ Stall cycles per access

Stall cycles per memory access $= (1-H1) \times H2 \times T2 + (1-H1) \times (1-H2) \times H3 \times T3 + (1-H1)(1-H2) (1-H3) \times M$

$= 0.05 \times 0.6 \times 2 + 0.05 \times 0.4 \times 0.5 \times 5 + 0.05 \times 0.4 \times 0.5 \times 100$

$= 0.097 + 0.0075 + 0.00225 = 1.11$

CPI $= 1.1 + 1.3 \times 1.11 = 2.54$

Speedup compared to $L_1$ only $= 7.6/2.54 = 3$

Speedup compared to $L_1$, $L_2$ $= 3.778/2.54 = 1.49$
3-Level (All Unified) Cache Performance
Memory Access Tree (Ignoring Write Policy)
CPU Stall Cycles Per Memory Access

CPU Memory Access

\[ L_1 \]
L1 Hit:
Stalls = H1 x 0 = 0
(No Stall)

L1 Miss:
\[ \% = (1-H1) \]

\[ L_2 \]
L2 Hit:
\[ (1-H1) x H2 x T2 \]

L2 Miss:
\[ \% = (1-H1)(1-H2) \]

\[ L_3 \]
L3 Hit:
\[ (1-H1) x (1-H2) x H3 x T3 \]

L3 Miss:
\[ (1-H1)(1-H2)(1-H3) x M \]

Stall cycles per memory access = \((1-H1) x H2 x T2 + (1-H1) x (1-H2) x H3 x T3 + (1-H1)(1-H2)(1-H3)x M\)

AMAT = 1 + Stall cycles per memory access
Program Steady-State Main Memory Bandwidth-Usage Example

• In the previous example with three levels of cache (all unified, ignore write policy)
• CPU with \( \text{CPI}_{\text{execution}} = 1.1 \) running at clock rate = 500 MHz
• 1.3 memory accesses per instruction.
• \( L_1 \) cache operates at 500 MHZ with a miss rate of 5%
• \( L_2 \) cache operates at 250 MHZ with a local miss rate 40%, \( (T_2 = 2 \text{ cycles}) \)
• \( L_3 \) cache operates at 100 MHZ with a local miss rate 50%, \( (T_3 = 5 \text{ cycles}) \)
• Memory access penalty, \( M = 100 \text{ cycles} \) (to deliver 32 bytes to CPU)

• We found the CPI:
  With No Cache, \( \text{CPI} = 1.1 + 1.3 \times 100 = 131.1 \)
  With single \( L_1 \), \( \text{CPI} = 1.1 + 1.3 \times 0.05 \times 100 = 7.6 \)
  With \( L_1, L_2 \) \( \text{CPI} = 1.1 + 1.3 \times (0.05 \times 0.6 \times 2 + 0.05 \times 0.4 \times 100) = 3.778 \)
  With \( L_1, L_2, L_3 \) \( \text{CPI} = 1.1 + 1.3 \times 1.11 = 2.54 \)

Assuming:
  instruction size = data size = 4 bytes, all cache blocks are 32 bytes

For each of the three cases with cache:
  What is the total number of memory accesses generated by the CPU per second?
  What is the percentage of these memory accesses satisfied by main memory?
  Percentage of main memory bandwidth used by the CPU?
**Program Steady-State Main Memory Bandwidth-Usage Example**

- Memory requires 100 CPU cycles = 200 ns to deliver 32 bytes, thus total main memory bandwidth = 32 bytes / (200 ns) = 160 x 10^6 bytes/sec

- The total number of memory accesses generated by the CPU per second = (memory access/instruction) x clock rate / CPI = 1.3 x 500 x 10^6 / CPI = 650 x 10^6 / CPI
  - With single L1 = 650 x 10^6 / 7.6 = 85 x 10^6 accesses/sec
  - With L1, L2 = 650 x 10^6 / 3.778 = 172 x 10^6 accesses/sec
  - With L1, L2, L3 = 650 x 10^6 / 2.54 = 255 x 10^6 accesses/sec

- The percentage of these memory accesses satisfied by main memory:
  - With single L1 = L1 miss rate = 5%
  - With L1, L2 = L1 miss rate x L2 miss rate = .05 x .4 = 2%
  - with L1, L2, L3 = L1 miss rate x L2 miss rate x L3 miss rate = .05 x .4 x .5 = 1%

- Memory Bandwidth used
  - With single L1 = 32 bytes x 85x10^6 accesses/sec x .05 = 136 x 10^6 bytes/sec
    or 136/160 = 85% of total memory bandwidth
  - With L1, L2 = 32 bytes x 172x10^6 accesses/sec x .02 = 110 x 10^6 bytes/sec
    or 110/160 = 69% of total memory bandwidth
  - With L1, L2, L3 = 32 bytes x 255x10^6 accesses/sec x .01 = 82 x 10^6 bytes/sec
    or 82/160 = 51% of total memory bandwidth
Computer System Components

CPU Core
1 GHz - 3.0 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)
400MHZ DDR
16 bits wide (32 banks)
~ 1.6 GBYTES/SEC

CPU
Caches
Memory Controller
Memory

System Bus

All Non-blocking caches
L1 16-128K 1-2 way set associative (on chip), separate or unified
L2 256K-2M 4-32 way set associative (on chip) unified
L3 2-16M 8-32 way set associative (off or on chip) unified

Examples:
Alpha, AMD K7: EV6, 200-400 MHZ
Intel PII, PIII: GTL+, 133 MHZ
Intel P4 800 MHZ

North Bridge
South Bridge
Chipset

I/O Buses
NICs

I/O Devices:
Disks
Displays
Keyboards
Networks
X86 CPU Cache/Memory Performance Example
AMD Athlon T-Bird Vs. Intel PIII

AMD Athlon T-Bird 1GHZ
L1: 64K INST, 64K DATA (3 cycle latency), both 2-way
L2: 256K 16-way 64 bit bus
    Latency: 7 cycles
    L1,L2 on-chip

Intel PIII GHZ
L1: 16K INST, 16K DATA (3 cycle latency), both 2-way
L2: 256K 8-way 256 bit, Latency: 7 cycles
    L1,L2 on-chip (32 byte blocks)

Main Memory:
PC2100
133MHZ DDR SDRAM 64bit
Peak bandwidth: 2100 MB/s
Latency Range: 19ns - 64ns

PC133
133MHZ SDRAM 64bit
Peak bandwidth: 1000 MB/s
Latency Range: 25ns - 80ns

PC800
Rambus DRDRAM
400 MHZ DDR 16-bit
Peak bandwidth: 1600 MB/s
(1 channel)
Latency Range: 35ns - 80ns


Intel 840 uses two PC800 channels
X86 CPU Cache/Memory Performance Example:
AMD Athlon T-Bird Vs. Intel PIII, Vs. P4

**AMD Athlon T-Bird 1GHz**
- L1: 64K INST, 64K DATA (3 cycle latency), both 2-way
- L2: 256K 16-way 64 bit bus, Latency: 7 cycles, L1,L2 on-chip

**Intel P4, 1.5 GHz**
- L1: 8K DATA (2 cycle latency), 4-way 64 byte blocks, 96KB Execution Trace Cache
- L2: 256K 8-way 256 bit blocks, 128 byte blocks, Latency: 7 cycles, L1,L2 on-chip

**Intel PIII 1GHz**
- L1: 16K INST, 16K DATA (3 cycle latency), both 2-way 32 byte blocks
- L2: 256K 8-way 256 bit bus, 128 byte blocks, Latency: 7 cycles, L1,L2 on-chip

AMD Athlon T-Bird
750MHZ-1GHZ
L1: 64K INST, 64K DATA, both 2-way
L2: 256K 16-way 64 bit
Latency: 7 cycles
L1,L2 on-chip

Memory:
PC2100
133MHZ DDR SDRAM 64bit
Peak bandwidth: 2100 MB/s

PC1600
100MHZ DDR SDRAM 64bit
Peak bandwidth: 1600 MB/s

AMD Athlon Duron
750MHZ-1GHZ
L1: 64K INST, 64K DATA both 2-way
L2: 64K 16-way 64 bit
Latency: 7 cycles
L1,L2 on-chip

Source: http://www1.anandtech.com/showdoc.html?i=1345&p=10
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)

<table>
<thead>
<tr>
<th>Speed (ns)</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1s</td>
<td>100s</td>
</tr>
<tr>
<td>10s</td>
<td>Ks</td>
</tr>
<tr>
<td>100s</td>
<td>Ms</td>
</tr>
<tr>
<td>10,000,000s (10s ms)</td>
<td>Gs</td>
</tr>
<tr>
<td>10,000,000,000s (10s sec)</td>
<td>Ts</td>
</tr>
</tbody>
</table>

- Faster
- Larger Capacity

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 $2^{16}$ up to $2^{32}$ bytes).

- 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 main memory from disk (demand paging).

- 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

**Virtual address**

![Diagram of virtual memory translation](image)

**Virtual page number**

**Page offset**

**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 \(\rightarrow\) Physical Address Translation

Contiguous virtual address space of a program

Page Fault: D in Disk (not allocated in main memory)

Physical location of blocks A, B, C

Virtual address:
- 0
- 4K
- 8K
- 12K

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

Virtual memory

Physical main memory

Disk
Mapping Virtual Addresses to Physical Addresses Using A Page Table
Virtual Address Translation

- Virtual page number
- Page table
  - Valid
  - Physical page or disk address
- Physical memory
- Disk storage
Page Table Organization

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

If 0 then page is not present in memory.
## 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 (Access time)</td>
<td>8–100 clock cycles</td>
<td>700,000–6,000,000 clock cycles</td>
</tr>
<tr>
<td></td>
<td>(6–60 clock cycles)</td>
<td>(500,000–4,000,000 clock cycles)</td>
</tr>
<tr>
<td></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 21264 Data TLB (DTLB) During Address Translation

Address Space Number (ASN) Identifies process similar to PID (no need to flush TLB on context switch)
A Memory Hierarchy With TLB & Two Levels of Cache

TLB: Direct Mapped 256 entries
L1 direct mapped 8KB
L2 direct mapped 4MB
Virtual address 64 bits
Physical address 41 bits
TLB & Cache Operation

Virtual address

TLB access

TLB miss
use page table

No

Yes

TLB hit?

Physical address

No

Yes

Write?

Try to read data
from cache

Cache miss stall

No

Yes

Cache hit?

Write access
bit on?

Write protection
bit on?

No

Yes

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 physically-addressed

Cache operation
CPU Performance with Real TLBs

When a real TLB is used with a TLB miss rate and a TLB miss penalty is used:

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

Where:

\[
\text{Mem Stalls per instruction} = \text{Mem accesses per instruction} \times \text{mem stalls per access}
\]

Similarly:

\[
\text{TLB Stalls per instruction} = \text{Mem accesses per instruction} \times \text{TLB stalls per access}
\]

\[
\text{TLB Stalls per access} = \text{TLB miss rate} \times \text{TLB miss penalty}
\]

Example:

Given: \(\text{CPI}_{\text{execution}} = 1.3\) \(\text{Mem accesses per instruction} = 1.4\)

\(\text{Mem stalls per access} = .5\) \(\text{TLB miss rate} = .3\%\) \(\text{TLB miss penalty} = 30\) cycles

What is the resulting CPU CPI?

\(\text{Mem Stalls per instruction} = 1.4 \times .5 = .7\) cycles/instruction

\(\text{TLB stalls per instruction} = 1.4 \times (\text{TLB miss rate} \times \text{TLB miss penalty})\)

\[= 1.4 \times .003 \times 30 = .126\] cycles/instruction

\(\text{CPI} = 1.3 + .7 + .126 = 2.126\)
# 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>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>