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. 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.
Logical DRAM Organization (16 Mbit)

Control Signals:
Row Access Strobe (RAS): Low to latch row address
Column Address Strobe (CAS): Low to latch column address
Write Enable (WE)
Output Enable (OE)
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 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
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 12 ns, but time between column accesses is at least 25 ns ($t_{PC}$).
    • In practice, external address delays and turning around buses make it 30 to 40 ns

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

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

256K x 8 DRAM
Every DRAM access begins at:

- The assertion of the RAS_L
- 2 ways to read: early or late v. CAS

Early Read Cycle: OE_L asserted before CAS_L
Late Read Cycle: OE_L asserted after CAS_L

256K x 8 DRAM

Read Access Time

Output Enable Delay
Simplified Asynchronous DRAM Read Timing

Source: http://arstechnica.com/paedia/r/ram_guide/ram_guide.part2-1.html
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 = 15 x 14 = 210 CPU cycles
Simplified Asynchronous Extended Data Out (EDO) DRAM Read Timing

- 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 tRAC ~ 40-60ns

Typical timing at 66 MHZ: 5-2-2-2
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 = 11 x 15 = 165 CPU cycles

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

<table>
<thead>
<tr>
<th></th>
<th>PC100</th>
<th>DDR266 (PC2100)</th>
<th>DDR2</th>
<th>DRDRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Potential Bandwidth</td>
<td>0.8 GB/s</td>
<td>2.133 GB/s</td>
<td>3.2 GB/s</td>
<td>1.6 GB/s</td>
</tr>
<tr>
<td>Interface Signals</td>
<td>64(72) data 168 pins</td>
<td>64(72) data 168 pins</td>
<td>64(72) data 184 pins</td>
<td>16(18) data 184 pins</td>
</tr>
<tr>
<td>Interface Frequency</td>
<td>100 MHz</td>
<td>133 MHz</td>
<td>200 MHz</td>
<td>400 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>
<td>35-80 nS</td>
</tr>
</tbody>
</table>
Synchronous Dynamic RAM, SDRAM Organization

DDR SDRAM 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.
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 = 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:

- 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
### Four way interleaved memory

<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>

#### 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

<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>
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 words) = 2%
Miss rate (block size = 4 words) = 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 = $2 + (1.2 \times .02 \times 2 \times 32) = 3.54$
  - 32-bit bus and memory, interleaved = $2 + (1.2 \times .02 \times (4 + 24 + 8) = 2.86$
  - 64-bit bus and memory, no interleaving = $2 + (1.2 \times 0.02 \times 1 \times 32) = 2.77$

- Increasing the block size to four words; resulting CPI:
  - 32-bit bus and memory, no interleaving = $2 + (1.2 \times 0.01 \times 4 \times 32) = 3.54$
  - 32-bit bus and memory, interleaved = $2 + (1.2 \times 0.01 \times (4 + 24 + 16) = 2.53$
  - 64-bit bus and memory, no interleaving = $2 + (1.2 \times 0.01 \times 2 \times 32) = 2.77$
Three-Level Cache Example

• CPU with CPI\textsubscript{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 \(L1, L2\) \[\text{CPI} = 1.1 + 1.3 \times (0.05 \times 0.6 \times 2 + 0.05 \times 0.4 \times 100) = 3.778\]

\[\text{CPI} = \text{CPI}\textsubscript{execution} + \text{Mem Stall cycles per instruction}\]

Mem Stall cycles per instruction = Mem accesses per instruction \(\times\) Stall cycles per access

Stall cycles per memory access \[= (1-H_1) \times H_2 \times T_2 + (1-H_1) \times (1-H_2) \times H_3 \times T_3 + (1-H_1)(1-H_2)(1-H_3) \times M\]
\[= 0.05 \times 0.6 \times 2 + 0.05 \times 0.4 \times 0.5 \times 5 + 0.05 \times 0.4 \times 0.5 \times 100\]
\[= 0.097 + 0.0075 + 0.00225 = 1.11\]

\[\text{CPI} = 1.1 + 1.3 \times 1.11 = 2.54\]

Speedup compared to \(L1\) only \[= 7.6/2.54 = 3\]
Speedup compared to \(L1, L2\) \[= 3.778/2.54 = 1.49\]

Repeated from lecture 8
3-Level Cache Performance

Memory Access Tree

CPU Stall Cycles Per Memory Access

CPU Memory Access

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

L1 Miss:
% = (1-H1)

L2
L2 Hit:
(1-H1) x H2 x T2

L2 Miss:
% = (1-H1)(1-H2)

L3
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

Repeated from lecture 8
Program Steady-State Main Memory Bandwidth-Usage Example

- In the previous example with three levels of cache (all unified, ignore write policy).
- CPU with CPI_{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 (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 x10^6 bytes/sec
    or 136/160 = 85 % of total memory bandwidth
  - With L1, L2 = 32 bytes x 172 x10^6 accesses/sec x .02 = 110 x10^6 bytes/sec
    or 110/160 = 69 % of total memory bandwidth
  - With L1, L2, L3 = 32 bytes x 255 x10^6 accesses/sec x .01 = 82 x10^6 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 Data Rate (DDR) SDRAM**
- PC2100
- 133MHZ DDR
- 64-128 bits wide
- 4-way interleaved
- ~2.1 GBYTES/SEC (64bit)

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

**CPU**

**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 chip) unified

**System Bus**

**Memory Controller**

**Memory**

**Controllers**
- Disks
- Displays
- Keyboards

**Adapters**

**I/O Buses**
- Example: PCI, 33-66MHZ
- 32-64 bits wide
- 133-528 MBYTES/SEC

**Networks**

**I/O Devices:**
- Example: Alpha, AMD K7: EV6, 200-333MHZ
- Intel PII, PIII: GTL+ 133 MHZ
- Intel P4 533 MHZ
X86 CPU Cache/Memory Performance Example

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

64K 320K


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

This Linpack data size range causes L2 misses and relies on main memory

**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 bus, 128 byte blocks
  - Latency: 7 cycles
  - L1,L2 on-chip

Intel PIII 1 GHz
- 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

X86 CPU Cache/Memory Performance Example:
AMD Athlon T-Bird Vs. Duron

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?id=1345&p=10
A Typical Memory Hierarchy

Faster
Larger Capacity

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)

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 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

Virtual address

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

Virtual page number  Page offset

Translation

Physical address

Physical page number  Page offset
### Paging Versus Segmentation

#### Table Comparison

<table>
<thead>
<tr>
<th>Page</th>
<th>Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Words per address</td>
<td>Two (segment and offset)</td>
</tr>
<tr>
<td>Programmer visible?</td>
<td>May be visible to application programmer</td>
</tr>
<tr>
<td>Replacing a block</td>
<td>Hard (must find contiguous, variable-size, unused portion of main memory)</td>
</tr>
<tr>
<td>Memory use inefficiency</td>
<td>External fragmentation (unused pieces of main memory)</td>
</tr>
<tr>
<td>Efficient disk traffic</td>
<td>Not always (small segments may transfer just a few bytes)</td>
</tr>
</tbody>
</table>

- **Paging**
  - Code
  - Data

- **Segmentation**
  - Code
  - Data
Virtual → Physical Address 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
Page Table Organization

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 (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: <8>
- Virtual page number: <35>
- Page offset: <13>

Virtual address

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

Protection Permissions

Valid bit

128:1 mux

DTLB = 128 entries

(Low-order 13 bits of address) 44- or 41-bit physical address

(High-order 31/28 bits of address)

Valid bit

Protection Permissions

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 hit?
   ↓
No
   ↓
TLB miss use page table
   ↓
Yes
   ↓
Physical address
   ↓
Write?
   ↓
No
   ↓
Write protection bit on?
   ↓
No
   ↓
Cache hit?
   ↓
No
   ↓
Cache miss stall
   ↓
Yes
   ↓
Deliver data to the CPU

Cache is physically-addressed

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

Cache operation

Write access bit on?
   ↓
Yes
   ↓
Write access bit on?
   ↓
No
   ↓
Try to read data from cache
   ↓
No
   ↓
Cache miss stall
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:

Mem Stalls per instruction = Mem accesses per instruction \times mem stalls per access

Similarly:

TLB Stalls per instruction = Mem accesses per instruction \times TLB stalls per access

TLB stalls per access = TLB miss rate \times TLB miss penalty

Example:

Given: CPI_{\text{execution}} = 1.3 \quad \text{Mem accesses per instruction} = 1.4

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 (\text{TLB miss rate} \times \text{TLB miss penalty})

= 1.4 \times 0.003 \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>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>