MEMORY SYSTEM ORGANIZATION

MEMORY TECHNOLOGY

• Every computer system contains a variety of devices to store the instructions and data
• The storage devices + the algorithms (HW or SW implemented) needed to control or manage the stored information constitute the memory system of the computer
• Desirable: processors should have immediate and uninterrupted access to memory
  – maximum speed to transfer information between a processor and memory
• Memories that operate at speeds comparable to processor speeds are relatively costly
  – It is not feasible to employ a single memory using just one type of technology
  – The stored information is distributed over a variety of different memory units with very different physical characteristics

DESIGNING MEMORY SYSTEM

• The goal in designing any memory system is to provide
  – An adequate storage capacity
  – An acceptable level of performance
  – A reasonable cost

SUMMARY

• Memory technology
• Hierarchical memory systems
• Characteristics of the storage-device
• Main memory organization
• SRAM
• DRAM
• Cache memory
DESIGNING MEMORY SYSTEM

1. The use of a number of different memory devices with different cost-performance ratios organized to provide a high average performance at a low average cost per bit
   - The individual memories form a hierarchy of storage devices
2. The development of automatic storage-allocation methods to make more efficient use of the available memory space.
4. The development of virtual-memory concepts
   - to free the ordinary user from memory management
   - to make programs largely independent of the physical memory configurations used.
5. The design of communication links to the memory system so that all processors connected to it can operate at or near their maximum rates
   - increasing the effective memory processor bandwidth
   - providing protection mechanisms to prevent programs from accessing or altering one another’s storage areas.

MAIN COMPONENTS OF THE MEMORY

1. Internal processor memory
   - A small set of high-speed registers used for temporary storage of instructions and data.
2. Main memory (or primary memory)
   - A relatively large & fast memory used for program and data storage during computer operation
   - Locations in main memory can be accessed directly and rapidly by the CPU instruction set
   - Semiconductor technology
3. Secondary memory (or auxiliary memory)
   - Large memory capacity & much slower than main memory
   - Store system programs, large data files, and the like which are not continually required by the CPU
   - Information in secondary storage is accessed indirectly via input-output programs
   - Representative technologies used for secondary memory are magnetic and optic disks

BLOCK DIAGRAM OF A MEMORY HIERARCHY (main components)

LOCALITY OF REFERENCES PRINCIPLES

- Programs commonly localize their memory references
  - Only a small portion of the memory is likely to be accessed at a given time
  - This characteristic is not a natural law; it may be violated
    - But it is a behavior pattern exhibited by many programs at most times
    - Exploiting this behavior is key to designing an effective memory hierarchy
TWO TYPES OF LOCALITY

- **Temporal Locality (Locality in Time)**
  - The location of a memory reference is likely to be the same as another recent reference
  - Programs tend to reference the same memory locations at a future point in time
  - Due to loops and iteration, programs spending a lot of time in one section of code

- **Spatial Locality (Locality in Space)**
  - The location of a memory reference is likely to be near another recent reference
  - Programs tend to reference memory locations that are near other recently-referenced memory locations

---

MEMORY HIERARCHY PRINCIPLES

- Increasing distance from the CPU represents also an increasing for the access time

```
CPU → Memory Level 1 → Memory Level i → Memory Level i+1
```

- **Inclusion property**: All information located on a upper memory level it is also stored on a lower memory level (ILi represents the information stored in memory level i):
  
  \[
  IL_1 \subset IL_2 \subset \ldots \subset IL_n
  \]

  - word miss / word hit

- **Coherence property**: An information stored at a specific address in level i must be the same on all lower memory levels
  - Maintain coherence:
    - propagation of the modified value to the lower levels ("write-through")
    - updating information on lower levels when replaced from level i ("write-back")
**Hierarchical Terminology**

- **Hit**: Accessed data is found in upper level (for example Block A in level i)
  - Hit Rate = fraction of accesses found in upper level
  - Hit Time = Time to access the upper level
    = Memory access time + Time to determine hit/miss
- **Miss**: Accessed data found only in lower level (for example Block B in level i+1)
  - Processor waits until data is fetched then restarts the access
  - Miss Rate = 1 – (Hit Rate)
  - Miss Penalty
    - Time to get block from lower level
    + time to replace in upper level
    + time to deliver the block to the processor
- **Hit Time << Miss Penalty**

---

**Memory-Device Functional Characteristics**

- **1. Storage capacity (S)**. Expressed in multiple of bits, or bytes.
  - \(2^{10}\) bits = 1024 bits = 1 Kb; 1024 Kb = 1 Mb
  - 8 bits = 1 Byte = 1 B
  - 1024 Mb = 1 Gb; 1024 Gb = 1 Tb
- **2. Cost**: The price include not only the cost of the information storage cells themselves but also the cost of the peripheral equipment or access circuitry essential to the operation of the memory
  - Let \(C\) be the price in dollars of a complete memory system with \(S\) bits of storage capacity, and specific cost \(c\) of the memory:
    \[
    c = \frac{C}{S} \text{ dollars / bit}
    \]
- **3. Access time**
  - The average time required to read a fixed amount of information, e.g., one word, from the memory - read access time or, more commonly, the access time of the memory \(t_A\)
  - The write access time is defined similarly - it is typically, but not always, equal to the read access time
  - Access time depends on the physical characteristics of the storage medium, and also on the type of access mechanism used
  - \(t_A\) is usually calculated from the time a read request is received by the memory unit to the time at which all the requested information has been made available at the memory output terminals
  - The access rate \(b_A\) of the memory defined as \(1/t_A\) and measured in words per second
- **4. Access modes** - the order of sequence in which information can be accessed
  - **Random-access memory** (RAM) - if locations may be accessed in any order and access time is independent of the location being accessed
    - Semiconductor memories
    - Each storage location has a separate access (addressing) mechanism
Memory-Device Characteristics

- **Serial-access memories** - storage locations can be accessed only in certain predetermined sequences
  - Magnetic-tape, optical memories.
  - The access mechanism is shared among different locations, and must be assigned to different locations at different times - moving the stored information, the read-write head, or both
  - Serial access tends to be slower than random access
- Magnetic disks contain a large number of independent rotating tracks
- If each track has its own read-write head, the tracks may be accessed randomly, although access within each track is serial
- The access mode is sometimes called **semi-random** or, rather misleadingly, **direct access**

---

Memory-Device Characteristics

- **5. Alterability** of information contained
  - ROMs - memories whose contents cannot be altered on-line
  - ROMs whose contents can be changed (usually off-line and with some difficulty) are called programmable read-only memories (PROMs).
  - RWM - memories in which reading or writing can be done on-line

---

Memory-Device Characteristics

- **6. Permanence of storage** - the stored information may be lost over a period of time unless appropriate action is taken
- **Destructive readout** (reading the memory destroys the stored information)
  - Each read operation must be followed by a write operation that restores the original state of the memory
  - The restoration is usually carried out automatically, by write back into the location originally addressed from a buffer register
- **Dynamic storage**
  - Over a period of time, a stored charge tends to leak away, causing a loss of information unless the charge is restored
  - The process of restoring is called **refreshing** (dynamic memories)
- **Volatility**
  - A memory is said to be volatile if the stored information can be destroyed by a power failure

---

Memory-Device Characteristics

- **7. Cycle time and data transfer rate**
  - Cycle time \( t_M \) is the time needed to complete any read or write operation in the memory
  - This means that the minimum time that must elapse between the initiation of two different accesses by the memory can be greater than \( t_M \)
  - Data transfer rate or bandwidth \( b_M \) - the maximum amount of information that can be transferred to or from the memory every second (= \( 1/t_M \))
    \[
    b_M = \frac{1}{t_M} \quad \text{[words/sec]}
    \]
    \[
    b_M = \frac{w}{t_M} \quad \text{[bits/sec]}
    \]
    \[
    w = \text{memory bus width}
    \]
Memory-Device Characteristics

• Cycle time vs access time

<table>
<thead>
<tr>
<th>Memory cycle time</th>
<th>Access time</th>
<th>Transfer time</th>
<th>Latency time</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>t₂</td>
<td>t₃</td>
<td>t₄</td>
</tr>
</tbody>
</table>

• In cases where $t_A \neq t_M$ both are used to measure memory speed

Memory-Device Characteristics

• 8. Power: specific power

\[ p = \frac{P_{ac}}{S} \quad [W/\text{bit}] \]

• 9. Geometry: only for semiconductor memories
  – $N$ lines x $M$ columns

Main memory devices

• SRAM
• DRAM
• Cache memory

MAIN MEMORY

• Main memory (primary or internal memory):
  – Used for program and data storage during computer operation
  – Directly accessed by the CPU (microprocessor)
  – Semiconductor technology
  – Volatile memory
  – Alterable memory (RWM)
  – A random accessed memory
  – Every cell is connected to selection lines (bit line for RD / WR)
  – Every cell is connected to data lines (column bit lines for RD / WR)
Example cell memory

• ROM-MOS

Example cell memory

• Static RAM (SRAM) cell

Example cell memory

• Dynamic RAM

Example: RAM selection cells
**Simple example: 4 x 4 bits RAM**

- **Register 1**
- **Register 2**
- **Data In**
- **2 to 4 Decoder**
- **Register 3**
- **Register 4**
- **Data Out**
- **Chip Select**
- **Address**
- **A1**
- **A0**
- **Address input**
- **Control signals**
- **Output Data pins**
- **Input Data pins**
- **Read/Write**
- **A1**
- **A0**
- **CS**
- **COMPUTER ARCHITECTURE**

### Static RAM (SRAM)

- Bipolar or unipolar technology
- Operating cycles:
  - Read cycle
  - Write cycle
- Every cell contains a latch (1 bit)
- Access circuitry:
  - Decode the address word
  - Control the cell selection lines
  - Read and write cells
  - Control internal logic

### SRAM

- A general approach to reducing the access circuitry cost in random-access memories is called matrix (array) organization.
- The storage cells are physically arranged as rectangular arrays of cells:
  - This is primarily to facilitate layout of the connections between the cells and the access circuitry.
- The memory address is partitioned into $d$ components so that the address $A_i$ of cell $C_i$ becomes a $d$-dimensional vector $(A_{i,1}, A_{i,2}, ..., A_{i,d}) = A_i$.
- Each of the $d$ parts of an address word goes to a different address decoder and a different set of address drivers.
- A particular cell is selected by simultaneously activating all $d$ of its address lines.
- A memory unit with this kind of addressing is said to be a $d$-dimensional memory.

### SRAM

- The simplest array organizations have $d = 1$ and are called **one-dimensional**, or 1-D, memories.
- If the storage capacity of the unit is $N \times w$ bits, then the access circuitry typically contains a one-out-of-$N$ address decoder and $N$ address drivers.
- For example, for a $8K \times 8$ RAM memory, for this 1-D organization:
  
  $$a=13, \ N = 2^{13} \text{ and } w = 8.$$
**Principle of 1-d addressing scheme**

- **Address Bus**
- **Address Register**
- **Address Decoder**
- **Storage locations**
- **Internal Control Signals**
- **Timing and control circuits**
- **CS**
- **R/W**
- **OE**
- **Data drivers and registers**
- **Data Bus**

**Explanation**

- \( N = 2^a \)
- The CS (chip select) input is used to enable the device
  - This is an active low input, so it will only respond to its other inputs if the CS line is low.
- The OE (output enable) input is used to cause the memory device to place the contents of the location specified by the address inputs onto the data pins
- The WR (write enable) input is used to cause the memory device to write the internal location specified by the address inputs with the data currently appearing on the data pins
  - This is an active low signal, and the actual write occurs on the rising edge when WR goes from low back to the high state

**Principle of 2-D addressing**

- The address field is divided into two components, called X and Y, which consist of ax and ay bits, respectively

- **X Address Decoder**
- **Array of memory storage locations**
- **Y Address Decoder**

**Explanation**

- The cells are arranged in a rectangular array of rows and columns
- The total number of cells is \( N = N_x \times N_y \)
- A cell is selected by the coincidence of signals on its X and Y address lines
- The 2-D organization requires substantially less access circuitry than the 1-D for a fixed amount of storage
- For example, if \( N_x = N_y = \sqrt{N} \) the number of address drivers needed is \( 2\sqrt{N} \)
- For \( N >> 4 \) the difference between 1-D and 2-D organizations is significant
- If the 1-bit storage cells are replaced by w-bit registers, then an entire word can be accessed in each read or write cycle but the bits within a word are not individually addressable
- For the example of 8K×8 memory the 2-D organization, for 8-bits on a word can look theoretically like in next slide
Theoretical example of a 2-d organization for a 8K×8 bits memory

Real structure example of a 2-d organization for a 8K×8 bits memory

Read Cycle Timing Diagram

Write Cycle Timing Diagram

- $t_{CR}$ = read cycle time
- $t_{AA}$ = access time (from address change)
- $t_{AC}$ = access time (from CS active)
- $t_{CW}$ = write cycle time
- $t_{DH}$ = data hold time
- $t_{DS}$ = data set-up time
- $w_{W}$ = write width
- $t_{DW}$ = delay for input data to write
- $t_{AW}$ = delay between address change and write control
DYNAMIC MEMORIES

- DRAMs are currently the primary memory device of choice because they provide the lowest cost per bit and greatest density among solid-state memory technologies.
- Every cell contains MOS capacitor(s).
- DRAM - dynamic since needs to be refreshed periodically (≈8 ms, 1% time).
- Addresses divided (address pins multiplexed) into 2 halves (Memory as a 2D matrix):
  - RAS or Row Address Strobe
  - CAS or Column Address Strobe
  - E.g. $2^{22}$ cells = 4 Mbit, memory array 2048 x 2048 cells, number of address IC pins = 11
- Operating cycles:
  - Read cycle
  - Write cycle
  - Refreshment cycle

The Conventional DRAM

- The multiplexed address bus uses two control signals - the row and column address strobe signals, (RAS and CAS respectively).
- The row address causes a complete row in the memory array to propagate down the bit lines to the sense amps.
- The column address selects the appropriate data subset from the sense amps and causes it to be driven to the output pins.
- Access transistors called 'sense amps' are connected to each column and provide the read and restore operations of the chip.
- Since the cells are capacitors that discharge for each read operation, the sense amp must restore the data before the end of the access cycle.
- A refresh controller determines the time between refresh cycles, and a refresh counter ensures that the entire array (all rows) is refreshed.

The Conventional DRAM
(typical memory access)

1. The row address bits are placed onto the address pins.
2. After a period of time the RAS signal falls, which activates the sense amps and causes the row address to be latched into the row address buffer.
3. The column address bits are set up.
4. The column address is latched into the column address buffer when CAS falls, at which time the output buffer is also turned on.
5. When CAS stabilizes, the selected sense amp feeds its data onto the output buffer.
- /WE = 0 write, = 1 read. If became active (LOW) before /CAS ("advance write cycle") data outputs remains in HiZ state.
- If /WE became active (LOW) after /CAS the cycle is a write-read cycle.
Conventional DRAM read cycles

![Diagram of Conventional DRAM read cycles]

Key DRAM Timing Parameters

- $t_{RAC}$: minimum time from RAS line falling to the valid data output (quoted as the speed of a DRAM when buy)
  - A typical 4Mb DRAM $t_{RAC} = 60$ ns
- $t_{RC}$: minimum time from the start of one row access to the start of the next.
  - $t_{RC} = 110$ ns for a 4Mbit DRAM with a $t_{RAC}$ of 60 ns
- $t_{CAC}$: minimum time from CAS line falling to valid data output.
  - 15 ns for a 4Mbit DRAM with a $t_{RAC}$ of 60 ns
- $t_{PC}$: minimum time from the start of one column access to the start of the next.
  - 35 ns for a 4Mbit DRAM with a $t_{RAC}$ of 60 ns
- A 60 ns ($t_{RAC}$) DRAM 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 ($t_{PC}$).

DRAM - fast speed op. modes

- Modes: change the access to columns, to reduce the average access time
- Fast page mode
  - Page = All bits on the same ROW (spatial locality)
  - Row-address is held constant and data from multiple columns is read from the sense amplifiers
  - Don’t need to wait for word-line to recharge
  - Toggle CAS with new column address
- Nibble mode
  - Four successive column addresses generated internally
- Burst mode
  - More successive column addresses (in the same page) generated internally

Fast Page Mode DRAM (FPM DRAM)

![Diagram of Fast Page Mode DRAM (FPM DRAM)]
Extended Data Out DRAM (EDO DRAM)

- EDO DRAM adds a latch between the sense-amps and the output pins. This latch holds output pin state and permits the CAS to rapidly de-assert, allowing the memory array to begin precharging sooner.
- The latch in the output path also implies that the data on the outputs of the DRAM circuit remain valid longer into the next clock phase.

Nibble mode (serial four)

Burst mode
Refreshment modes

- RAS only refresh (conventional refresh)
- CAS-before-RAS refresh (internal logic for refresh)
- Hidden refresh (refresh cycle is hidden in a read/write access cycle)
**Synchronous DRAM (SDRAM)**

- SDRAM exchanges data with the processor synchronized to an external clock signal.
- SDRAM latches information to and from the controller based on a clock signal.
- SDRAM employs a burst mode.
- In burst mode, a series of data bits can be clocked out rapidly after the first bit has been accessed. This mode is useful when all the bits to be accessed are in sequence and in the same row of the array as the initial access.
- SDRAM has a dual-bank internal architecture that improves opportunities for on-chip parallelism (interleaved memory banks).

**SDRAM Read Operation Clock Diagram**

**DRAM vs SDRAM**

**DRAM**
- No clock
- RAS control by change level
- One bank (array) of memory
- A transfer for every column address (or CAS pulse)
- Read delay (latency) no programmable

**SDRAM**
- Operate at the clock frequency
- RAS control at the clock impulse
- Two interleaved memory banks
- Sometimes static cache at the interface
- Burst transfer programmable (1, 2, 4, 8, or 256 transfers for a single provided column address, in the same row)
- Read latency programmable
Cache principles

- Cache memory is an intermediate temporary memory unit positioned between the processor registers and main memory.
- Large and slow main memory ↔ smaller and faster cache memory.
- The cache contains a copy of portions of main memory.
- When the processor attempts to read a word of memory, a check is made to determine if the word is in the cache:
  - If so, the word is delivered to the processor.
  - If not, a block of main memory, consisting of some fixed number of words, is read into the cache and then the word is delivered to the processor.
- Because of the phenomenon of locality of reference, when a block of data is fetched into the cache to satisfy a single memory reference, it is likely that future references will be to other words in the block.

**Cache and Main Memory**

![Diagram of CPU, Cache, and Main Memory]

**Structure of a Cache/Main-Memory System**

- M = \(2^a/K\) blocks
- \(m << M\)
**Hit Ratio**

- The performance of cache memory can be given by a synthetic parameter, called **hit ratio (HR)**
- Hit ratio determined running benchmarks
- HR represents the ratio between the total numbers of hits in cache and the total number of memory accesses (hit + miss numbers)
- The value of HR must be greater than 0.9
- From HR value we can compute the average memory access time. For example if HR = 0.9, the access time to main memory (for misses) is 100 nsec. and access time to cache memory is 10 ns, the average memory access time is:

\[
t_{acm} = \frac{9 \times 10 + 100}{10} = 19\text{ns}
\]

**Dynamic Mapping**

- Direct mapping is the simplest technique
- Direct mapping maps each block of main memory into only one possible cache line
- The mapping is expressed as: \( i = j \mod m \)
  - where \( i \) = cache line number; \( j \) = main memory block number; \( m \) = number of lines in the cache
- For purposes of cache access, each main memory address can be viewed as consisting of three fields
  - The least significant \( w \) bits identify a unique word or byte within a block of main memory (in most contemporary machines, the address is at the byte level)
  - The remaining \( a-w \) bits specify one of the \( 2^{a} \) blocks of main memory
  - The cache logic interprets these \( a-w \) bits as a tag of \( t \) bits (most significant portion) plus a cache line field of \( r \) bits \((a = t+r+w)\)
Direct mapping

- Sometime the word and block fields are called index field, because the index is used to address data in cache.
- The use of a portion of the address as a line number provides a unique mapping of each block of main memory into the cache.
- When a block is actually read into its assigned line, it is necessary to tag the data to distinguish it from other blocks that can fit into that line.
- The effect of this mapping is that blocks of main memory are assigned to lines of the cache as follows:

<table>
<thead>
<tr>
<th>Cache line</th>
<th>Main memory blocks assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0, m, 2m, ..., 2^m-m</td>
</tr>
<tr>
<td>1</td>
<td>1, m+1, 2m+1, ..., 2^m+1-m+1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>m-1</td>
<td>m-1, 2m-1, ..., 2^m-1</td>
</tr>
</tbody>
</table>

Example for address 1D00002E hex

- 32 bits memory address
- 8 bits tag
- 20 bits block address
- 4 bits word addr.

Advantages for direct mapping:
- Simple and cheap
- The tag field is short
  - Only those bits have to be stored which are not used to address the cache
- Access is very fast.

Disadvantage:
- There is a fixed cache location for any given block
- If a program happens to reference words repeatedly from two different blocks that map into the same line, then the blocks will be continually swapped in the cache, and the hit ratio will be low.
Associative mapping

- Each main memory block can be loaded into any line of the cache
- The cache control logic interprets a memory address simply as a tag and a word field
- In the cache is stored also data and the corresponding address
- The associative mapping is implemented with associative memories (content addressable memories) as cache memories
  - more expensive than a random access memory
  - each cell must have storage capacity as well comparison logic circuits for matching its content with an external argument

Fully associative mapping

Advantages:
- provides the highest flexibility concerning the line to be replaced when a new block is read into the cache
  \[ i = j \mod 1 \]
- where \( i \) = cache line number; \( j \) = main memory block number;

Disadvantages:
- complex
- the tag field is long
- fast access can be achieved only using high performance associative memories for the cache, which is difficult and expansive.
Set associative mapping

- Set associative mapping is a compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages.
- In this case, the cache is divided into \( v \) sets, each of which consists of \( k \) lines.
- This is referred to as \( k \)-way set associative mapping.
- With set associative mapping, block \( Bj \) can be mapped into any of the lines of set \( i \).
- In this case, the cache control logic interprets a memory address simply as three fields: tag, set, and word.
  - With fully associative mapping, the tag in a memory address is quite large and must be compared to the tag of every line in the cache.
  - With \( k \)-way set associative mapping, the tag in a memory address is much smaller and is only compared to the \( k \) tags within a single set.

**Example two-way set associative**

<table>
<thead>
<tr>
<th>Block Address</th>
<th>Tags</th>
<th>Words</th>
<th>Tags</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>000011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Set associative mapping**

- The mapping relationships are:
  \[ i = j \mod v \]
  \[ m = v \times k \]
- where \( i \) = cache set number; \( j \) = main memory block number; \( m \) = number of lines in the cache; \( v \) = number of sets in cache; \( k \) = number of cache lines per set.
- In the extreme case of \( v = m, k = 1 \), the set associative technique reduces to direct mapping.
- In the extreme case of \( v = 1, k = m \), it reduces to associative mapping.
- The use of two lines per set \((v = m/2, k = 2)\) is the most common set associative organization (two-way set associative mapping).
- Four-way set associative \((v = m/4, k = 4)\) makes a modest additional improvement for a relatively small additional cost.
  - Further increases in the number of lines per set have little effect.

**Example**

- Assume that:
  - Main memory is divided in 32 blocks.
  - Cache cache has 8 block frames (lines).
  - The set-associative organization has 4 sets with 2 blocks per set, called two-way set associative.
- Where in cache main memory block 12 can be placed, for the three categories of cache organization?
  - Fully associative: block 12 can go into any of the eight block frames of the cache.
  - Direct mapped: block 12 can only be placed into block frame 4 (12 modulo 8).
  - Two-way set associative: allows block 12 to be placed anywhere in set 0 (12 modulo 4). Line 0 or line 1.
Replacement Algorithms

- **LRU (least recently used):** Replace that block in the set that has been in the cache longest with no reference to it
  - For two-way set associative each line includes a USE bit
  - When a line is referenced, its USE bit is set to 1 and the USE bit of the other line in that set is set to 0
  - When a block is to be read into the set, the line whose USE bit is 0 is used
  - Because we are assuming that more recently used memory locations are more likely to be referenced, LRU should give the best hit ratio.
- **FIFO (first-in-first-out):** Replace that block in the set that has been in the cache longest
  - FIFO is easily implemented as a round-robin or circular buffer technique.
- **LFU (least frequently used):** Replace that block in the set that has experienced the fewest references
  - LFU could be implemented by associating a counter with each line
- **Random replacement:** is the simplest to implement and results are surprisingly good.

Write Policy

- **Write through:** all write operations are made to main memory as well as to the cache
  - Any other processor-cache module can monitor traffic to main memory to maintain consistency within its own cache
  - The main disadvantage of this technique is that it generates substantial memory traffic and may create a bottleneck
- **Write back:** updates are made only in the cache
  - Minimizes memory writes
  - When an update occurs, an UPDATE bit associated with the line is set
  - When a block is replaced, it is written back to main memory if and only if the UPDATE bit is set
  - The problem with write back is that portions of main memory are invalid, and hence accesses by I/O modules can be allowed only through the cache
Write Policy

• Write-through with buffered write
  – The same as write-through, but instead of slowing the processor down by writing directly to main memory, the write address and data are stored in a high-speed write buffer; the write buffer transfers data to main memory while the processor continues its task.
• Dirty bits: this status bit indicates whether the block is dirty (modified while in the cache) or clean (not modified)
  – If it is clean, the block is not written on a miss, since the lower level has identical information to the cache.
• Both write back and write through have their advantages
  – With write back, writes occur at the speed of the cache memory, and multiple writes within a block require only one write to the lower-level memory
  – write back uses less memory bandwidth, making write back attractive in multiprocessors
  – With write through, read misses never result in writes to the lower level, and write through is easier to implement than write back
    • Write through also has the advantage that the next lower level has the most current copy of the data
    • This is important for I/O and for multiprocessors

Write Policy

• In a bus organization in which more than one device (typically a processor) has a cache and main memory is shared:
• If data in one cache is altered:
  ⇒ This invalidates the corresponding word in main memory
  ⇒ This invalidates the corresponding word in other caches (if any other cache happens to have that same word)
  ⇒ Even if a write-through policy is used, the other caches may contain invalid data
• A system that prevents this problem is said to maintain cache coherency

Write Policy

• Possible approaches to cache coherency include the following:
  – Bus watching with write through: Each cache controller monitors the address lines to detect write operations to memory by other bus masters
  – If another master writes to a location in shared memory that also resides in the cache memory, the cache controller invalidates that cache entry
  – Hardware transparency: Additional hardware is used to ensure that all updates to main memory via cache are reflected in all caches
  – Thus, if one processor modifies a word in its cache, this update is written to main memory. In addition, any matching words in other caches are similarly updated.
  – Noncachable memory: Only a portion of main memory is shared by more than one processor, and this is designated as noncachable
  – All accesses to shared memory are cache misses, because the shared memory is never copied into the cache
  – The noncachable memory can be identified using chip-select logic or high-address bits

Write Policy for write miss

• Since the data are not needed on a write, there are two common options on a write miss:
  • Write allocate (also called fetch on write)
    – The block is loaded on a write miss, followed by the write-hit actions above
    – This is similar to a read miss.
  • No-write allocate (also called write around)
    – The block is modified in the lower level and not loaded into the cache.
  • Write-back caches generally use write allocate (hoping that subsequent writes to that block will be captured by the cache)
  • Write-through caches often use no-write allocate (since subsequent writes to that block will still have to go to memory)