Computer Organization and Assembly Language (COAL)

Lecture 8

Dr. Naveed Anwar Bhatti

Webpage: naveedanwarbhatti.github.io

Locality

Principle of Locality:

- Programs tend to **reuse data and instructions near those used recently**, or that were recently referenced.
- ***Temporal locality:*** Recently referenced items are likely to be referenced in the near future.
- ***Spatial locality:*** Items with nearby addresses tend to be referenced close together in time.
- *Programs with good locality run faster than programs with poor locality*
- At the hardware level, the principle of locality allows computer designers to speed up main memory accesses by introducing small fast memories known as ***cache memories*** that hold blocks of the most recently referenced instructions and data items
- At the operating system level, the principle of locality allows the system to use the main memory as a cache of the most recently referenced chunks of the virtual address space.

Locality

- The principle of locality also plays a crucial role in the design of application programs.
- **Web browsers** exploit temporal locality by caching recently referenced documents on a local disk.
- **High-volume Web servers** hold recently requested documents in front-end disk caches that satisfy requests for these documents without requiring any intervention from the server

Locality Example

```
1   int sumvec(int v[N])
2   {
3       int i, sum = 0;
4
5       for (i = 0; i < N; i++)
6           sum += v[i];
7       return sum;
8   }
```

(a)

Address	0	4	8	12	16	20	24	28
Contents	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
Access order	1	2	3	4	5	6	7	8

(b)

Data:

Reference array elements in succession (stride-1): Spatial

Reference sum each iteration: Temporal

Locality Example

Claim: Being able to look at code and get a qualitative sense of its locality is a key skill for a professional programmer.

Question: Does this function have good locality?

```
1  int sumarrayrows(int a[M][N])
2  {
3      int i, j, sum = 0;
4
5      for (i = 0; i < M; i++)
6          for (j = 0; j < N; j++)
7              sum += a[i][j];
8      return sum;
9  }
```

(a)

Address	0	4	8	12	16	20
Contents	a_{00}	a_{01}	a_{02}	a_{10}	a_{11}	a_{12}
Access order	1	2	3	4	5	6

(b)

Locality Example

Question: Does *this* function have good locality? How does it compare to the previous version?

```
1   int sumarraycols(int a[M][N])
2   {
3       int i, j, sum = 0;
4
5       for (j = 0; j < N; j++)
6           for (i = 0; i < M; i++)
7               sum += a[i][j];
8       return sum;
9   }
```

(a)

Address	0	4	8	12	16	20
Contents	a_{00}	a_{01}	a_{02}	a_{10}	a_{11}	a_{12}
Access order	1	3	5	2	4	6

(b)

Summary of Locality

- Programs that repeatedly reference the same variables enjoy good temporal locality.
- For programs with stride-k reference patterns, the smaller the stride the better the spatial locality. Programs with stride-1 reference patterns have good spatial locality. Programs that hop around memory with large strides have poor spatial locality.
- Loops have good temporal and spatial locality with respect to instruction fetches. The smaller the loop body and the greater the number of loop iterations, the better the locality.

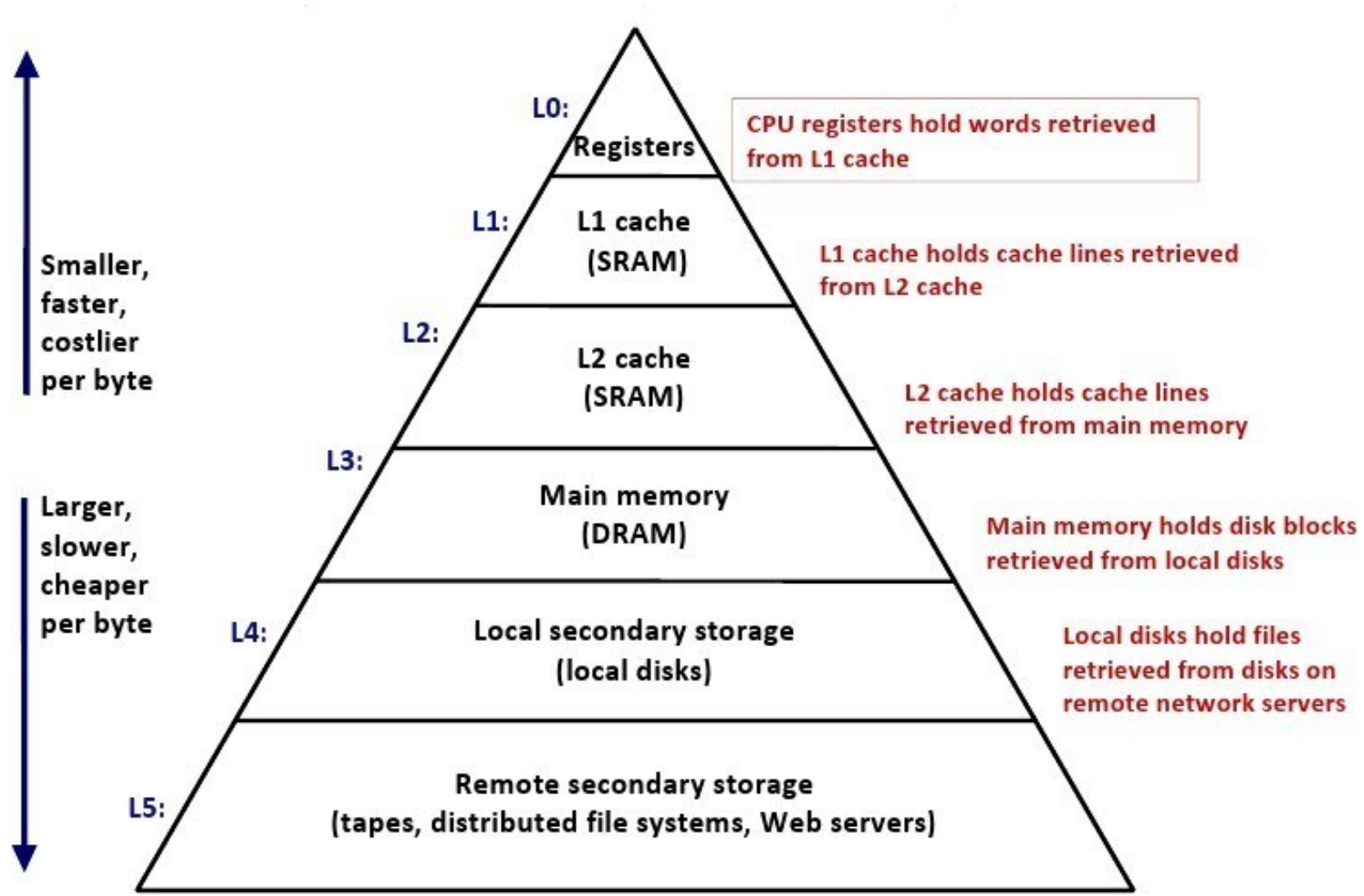
Memory Hierarchies

Some fundamental and enduring properties of hardware and software:

- Fast storage technologies cost more per byte and have less capacity.
- The gap between CPU and main memory speed is widening.
- Well-written programs tend to exhibit good locality.

These fundamental properties complement each other **beautifully**.

They suggest an approach for organizing memory and storage systems known as a *memory hierarchy.*


Smaller, faster, costlier per byte
Larger, slower, cheaper per byte
L0:
Registers
CPU registers hold words retrieved from L1 cache
L1:
L1 cache (SRAM)
L1 cache holds cache lines retrieved from L2 cache
L2:
L2 cache (SRAM)
L2 cache holds cache lines retrieved from main memory
L3:
Main memory (DRAM)
Main memory holds disk blocks retrieved from local disks
L4:
Local secondary storage (local disks)
Local disks hold files retrieved from disks on remote network servers
L5:
Remote secondary storage (tapes, distributed file systems, Web servers)

Cache

Cache: A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.

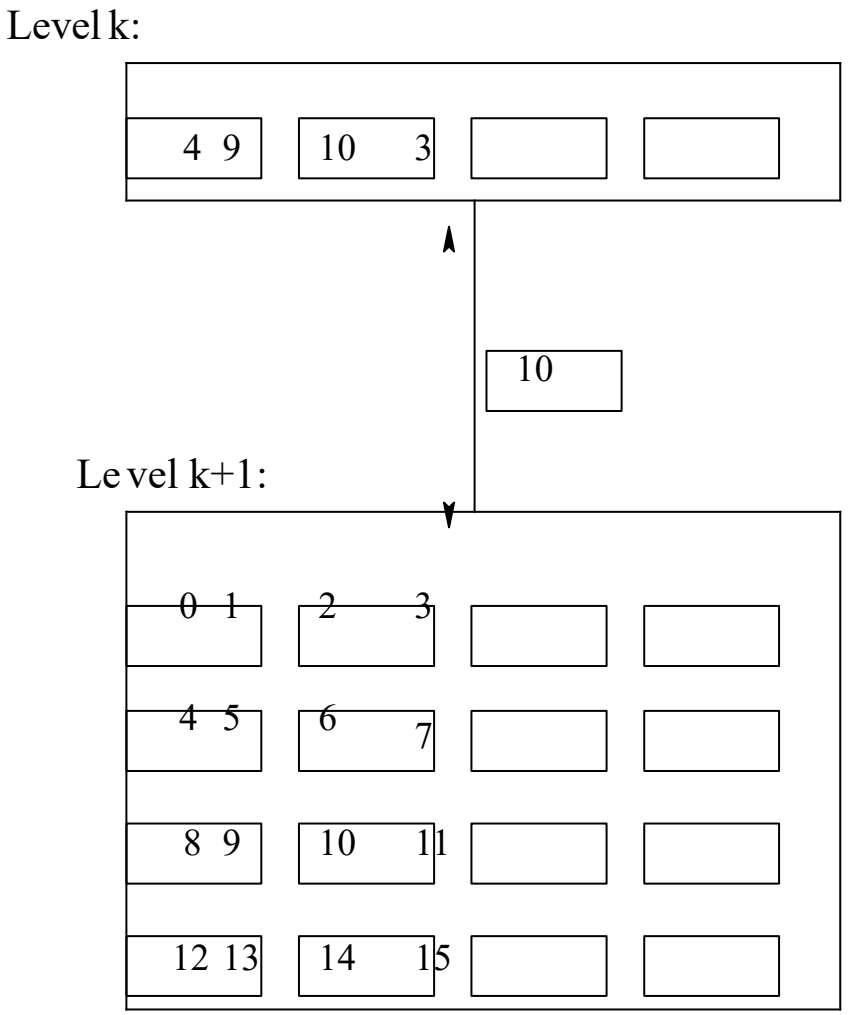
The fundamental idea of a memory hierarchy: For each k, the faster, smaller device at level k serves as a cache for the larger, slower device at level k+1.

Why Memory Hierarchies?

Why do memory hierarchies work?

- Programs tend to access the data at level k more often than they access the data at level k+1.
- Thus, the storage at level k+1 can be slower, and thus larger and cheaper per bit.
- *Net effect:* A large pool of memory that costs as much as the cheap storage near the bottom, but that serves data to programs at the rate of the fast storage near the top.
- We use a combination of small fast memory and big slow memory to give the illusion of **big fast memory**.

Caching in a Memory Hierarchy



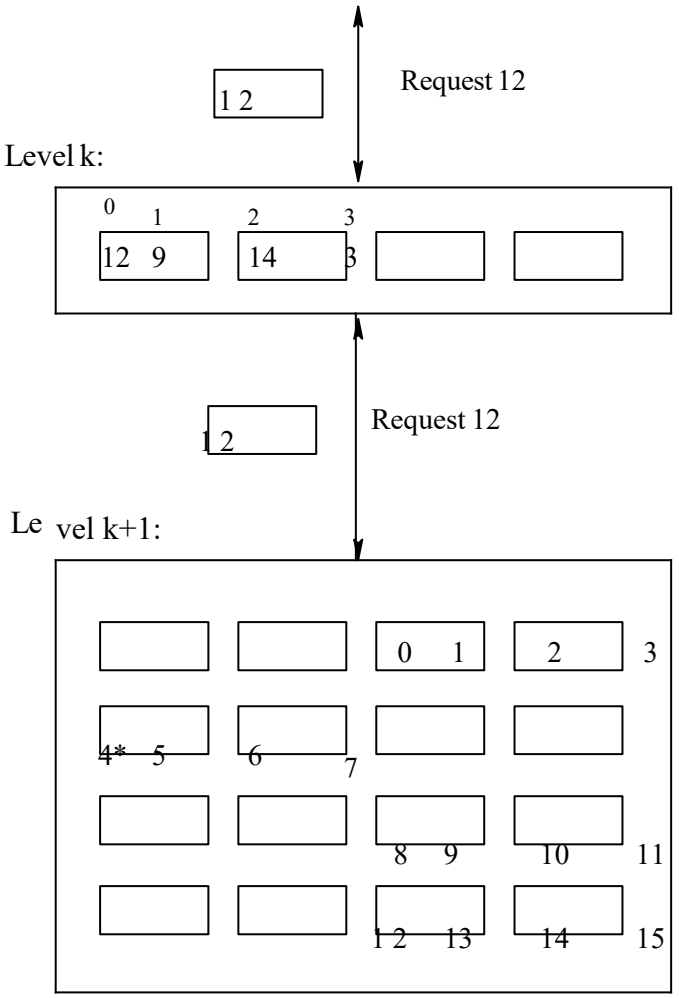
Smaller, faster, more expensive device at level k caches a subset of the blocks from level k+1.

Data is copied between levels in block-sized transfer units.

Larger, slower, cheaper storage device at level k+1 is partitioned into blocks.

General Caching Concepts

Program needs object d, stored in some block b.



Cache hit: program finds b in the level k cache, e.g., block 14.

Cache miss: b is not at level k, so must fetch it from level k+1, e.g., block 12.

- If level k cache is full, then some current block must be replaced (evicted). Which one is the "victim"?
- *Placement policy:* where can the new block go? E.g., b mod 4.
- *Replacement policy:* Which block should be evicted? E.g., LRU.

General Caching Concepts

Types of cache misses:

***Cold (compulsary) miss*:** the cache is empty.

Conflict miss: all available positions at level k are occupied.

- Most caches limit blocks at level k+1 to a small subset (sometimes only one) of the block positions at level k.
- E.g., Block i at level k+1 must be placed in block (i mod 4) at level k+1.
- Conflict misses occur when multiple data objects all map to the same level k block. Note: there still may be empty slots in the cache.

Capacity miss: the set of active cache blocks (working set) is larger than the cache.

Computer Organization and Assembly Language (COAL)

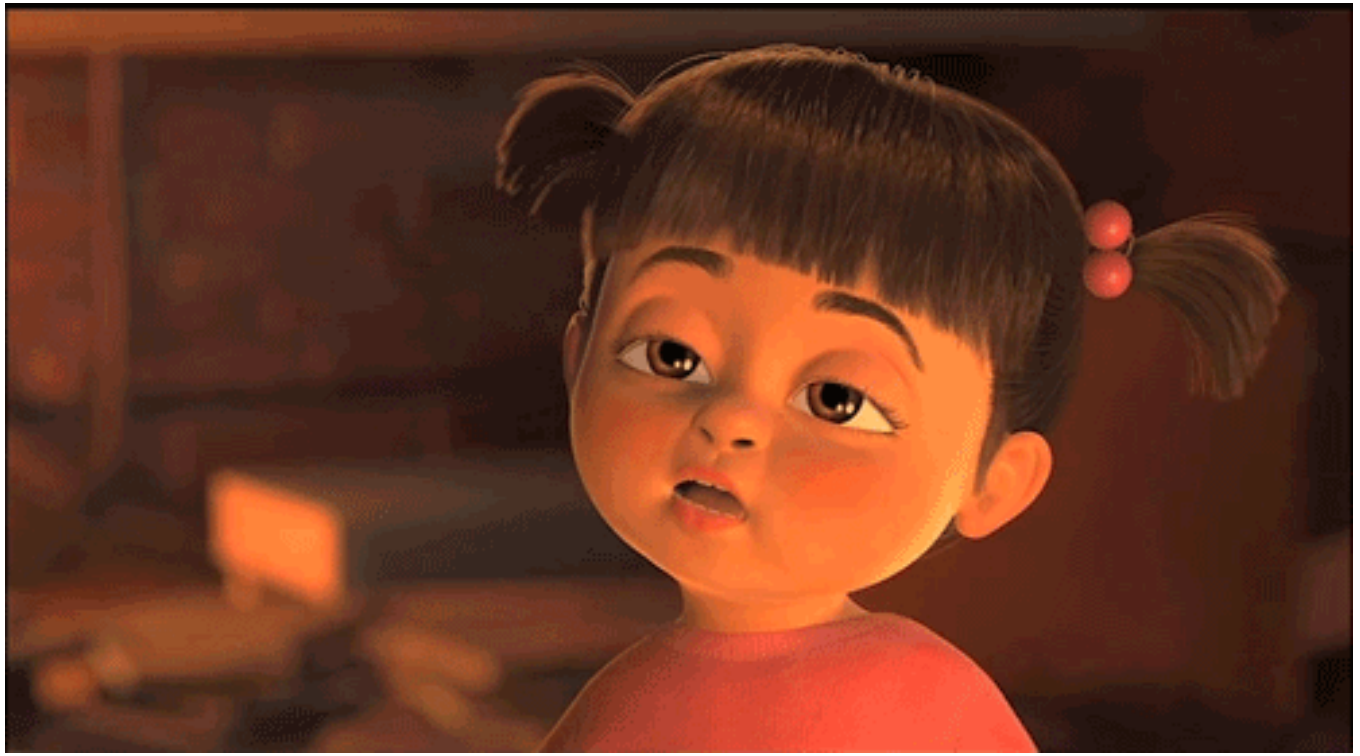
The End

Dr. Naveed Anwar Bhatti

Webpage: naveedanwarbhatti.github.io

Khatam, Tata, Bye-bye

Thanks a lot



If you are taking a Nap, **wake up**.....Lecture Over