COL 216

Holi Semester (Feb-May), 2020-21

Mondays and Thursdays 9:30-10:50 AM, Online

COMPUTER ARCHITECTURE

THE MEMORY HIERARCHY

Preeti Ranjan Panda
Dept. of Computer Science and Engineering, IIT Delhi

Acknowledgment
Minor adaptation of lecture slides provided by publishers of: *Computer Organization and Design: The Hardware/Software Interface*
David A. Patterson and John L. Hennessy
Morgan Kaufmann Publisher

The Memory Problem

- Programs need larger and larger amounts of memory
- Larger the memory, longer the access time
  - Why?
  - All programs suffer! (even those that don’t need large memories)
- **Solution: Cache Memory (Hierarchy of Memories)**
  - Small (fast) memory closer to the pipeline
  - Larger (slow) backup memory further away

Principle of Locality

- Programs access a small proportion of their address space at any time
- **Temporal locality**
  - Items accessed recently are likely to be accessed again soon
  - Why?
- **Spatial locality**
  - Items near those accessed recently are likely to be accessed soon
  - Why?

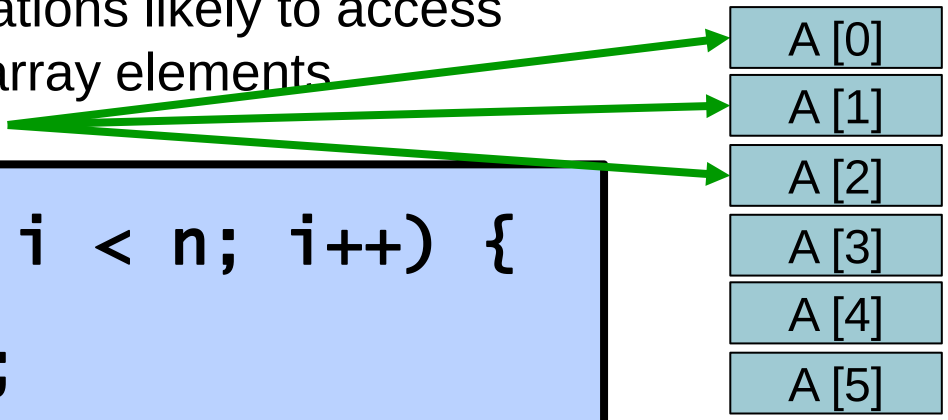
Temporal Locality Example

Instructions in LOOP accessed repeatedly

```
Loop: sll  $t1, $s3, 2
      add  $t1, $t1, $s6
      lw   $t0, 0($t1)
      bne  $t0, $s5, Exit
      addi $s3, $s3, 1
      j    Loop
Exit: …
```

Spatial Locality Example

Successive loop iterations likely to access consecutive array elements

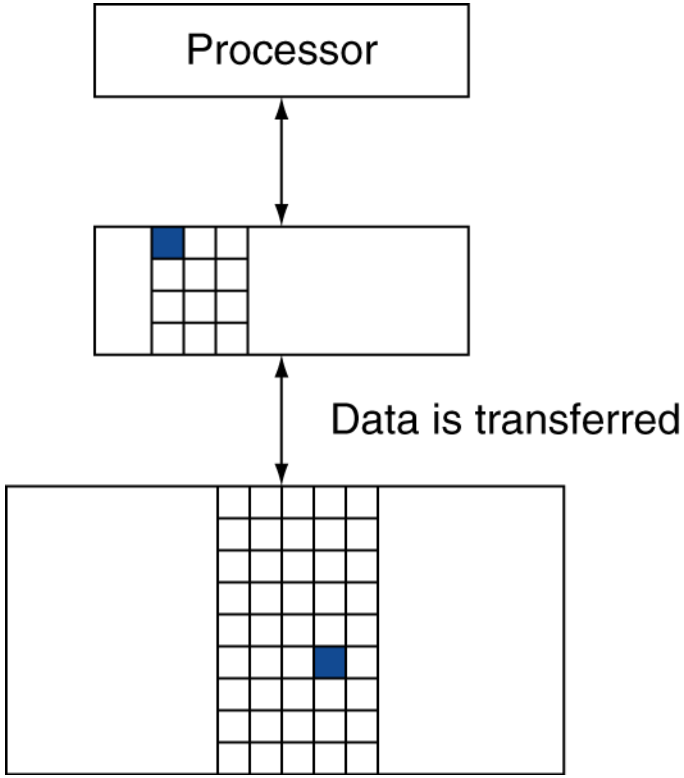


```
for (i = 0; i < n; i++) {
    A[i] = 0;
}
```

Taking Advantage of Locality

- Memory hierarchy
- Store everything on disk
- Copy recently accessed (and nearby) items from disk to smaller DRAM memory
  - Main memory
- Copy more recently accessed (and nearby) items from DRAM to smaller SRAM memory
  - Cache memory attached to CPU

Memory Hierarchy Levels



- Block (Line): unit of copying
  - May be multiple words
- If accessed data is present in upper level
  - Hit: access satisfied by upper level
    - Hit ratio: hits/accesses
- If accessed data is absent
  - Miss: block copied from lower level
    - Time taken: miss penalty
    - Miss ratio: misses/accesses
      = 1 – hit ratio
  - Then accessed data supplied from upper level

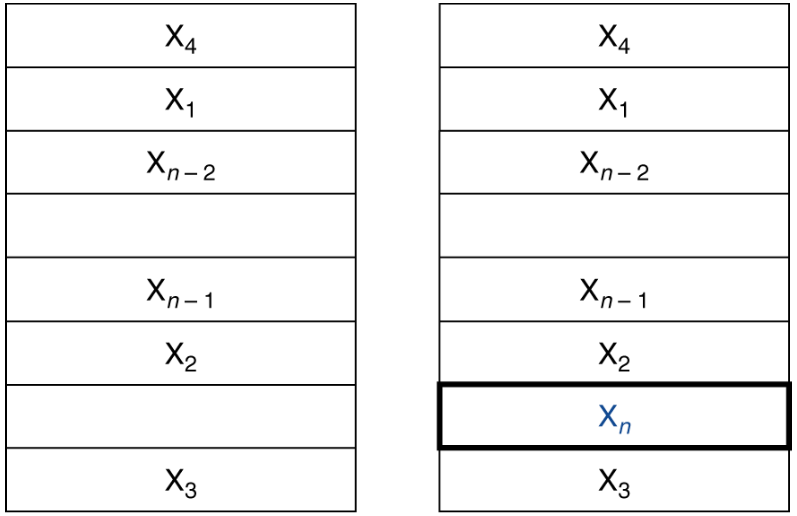
Memory Technology

- Static RAM (SRAM)
  - 0.5ns – 2.5ns, $2000 – $5000 per GB
- Dynamic RAM (DRAM)
  - 50ns – 70ns, $20 – $75 per GB
- Magnetic disk
  - 5ms – 20ms, $0.20 – $2 per GB
- Ideal memory
  - Access time of SRAM
  - Capacity and cost/GB of disk

§5.3 The Basics of Caches

Cache Memory

- Cache memory
  - The level of the memory hierarchy closest to the CPU
- Given accesses $X_1, \ldots, X_{n-1}, X_n$

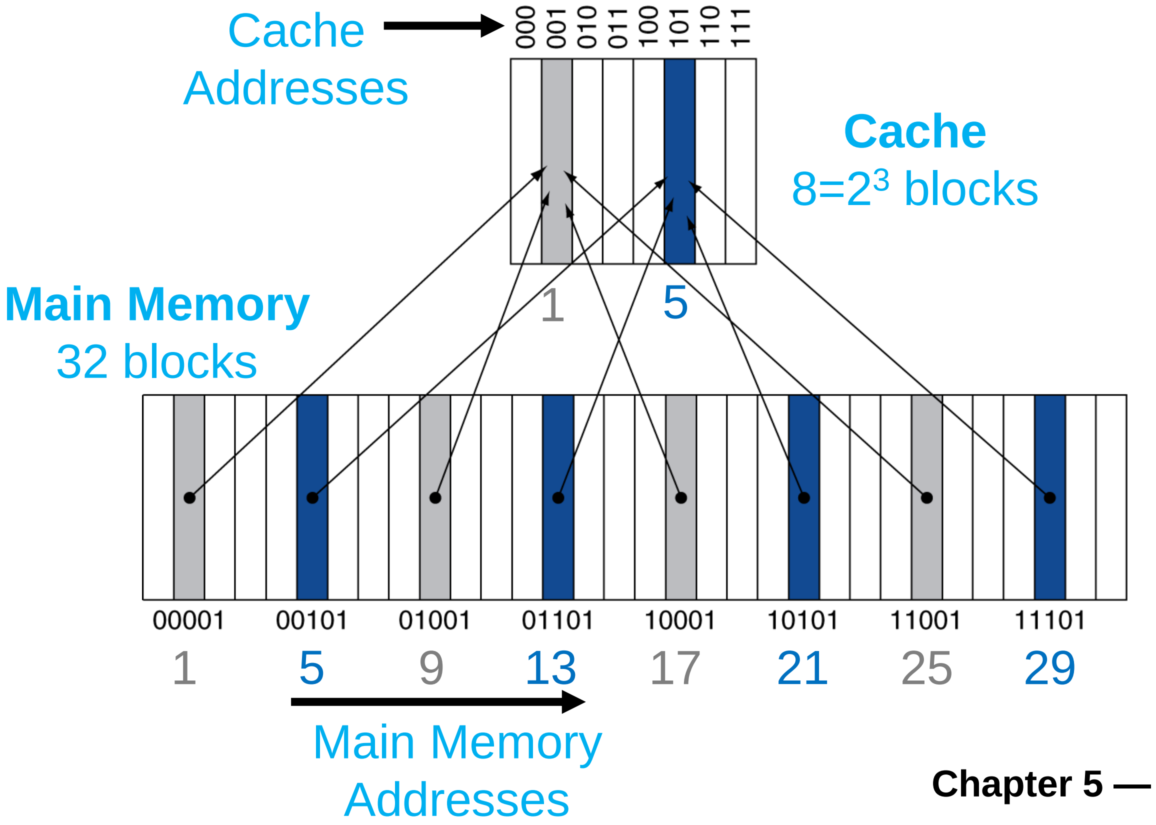


a. Before the reference to X_n b. After the reference to X_n

- How do we know if the data is present?
- Where do we look?

Direct Mapped Cache

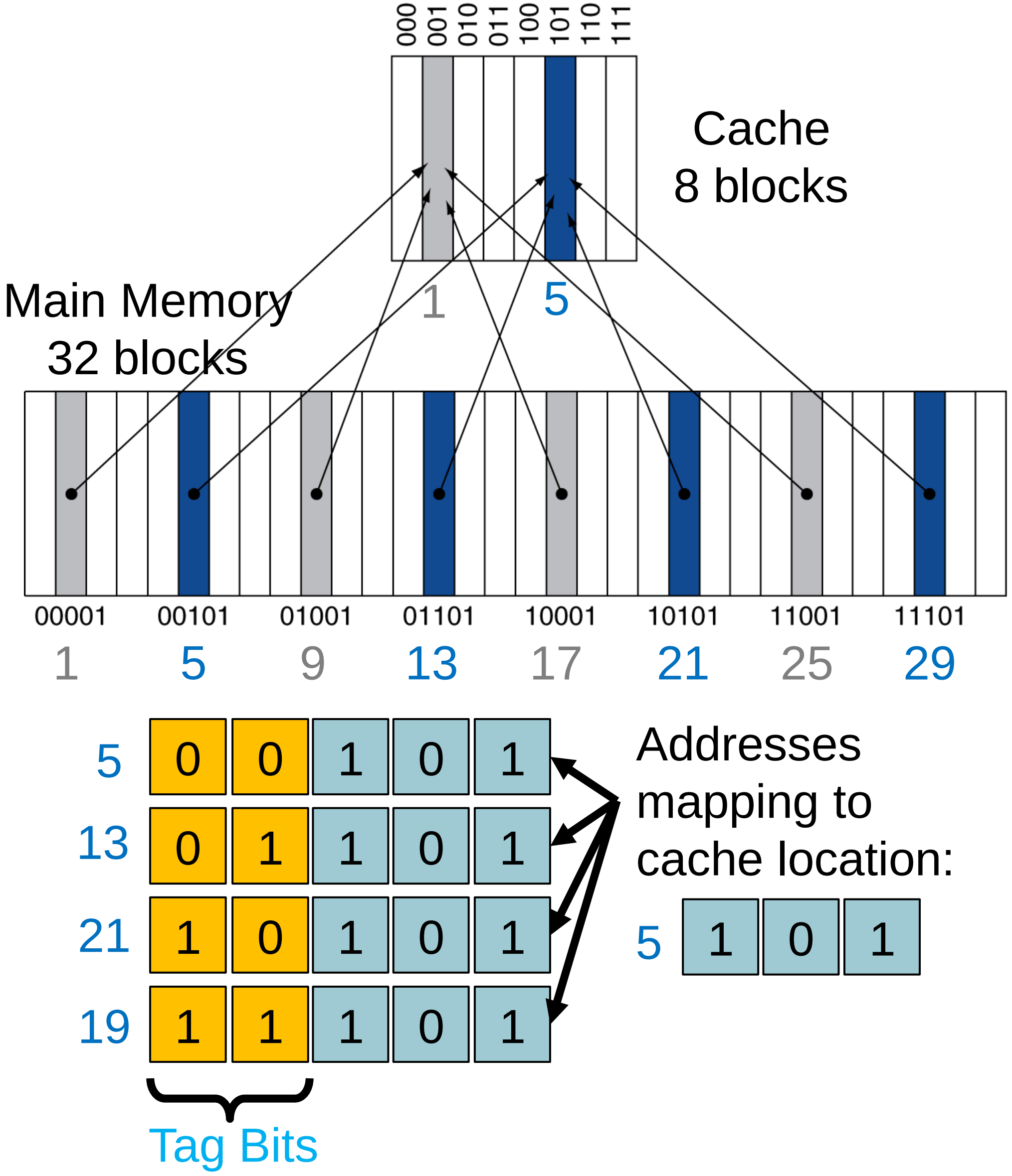
- Location determined by address
- Direct mapped: only one choice
  - (Block address) modulo (#Blocks in cache)



- #Blocks is a power of 2
- Use low-order address bits

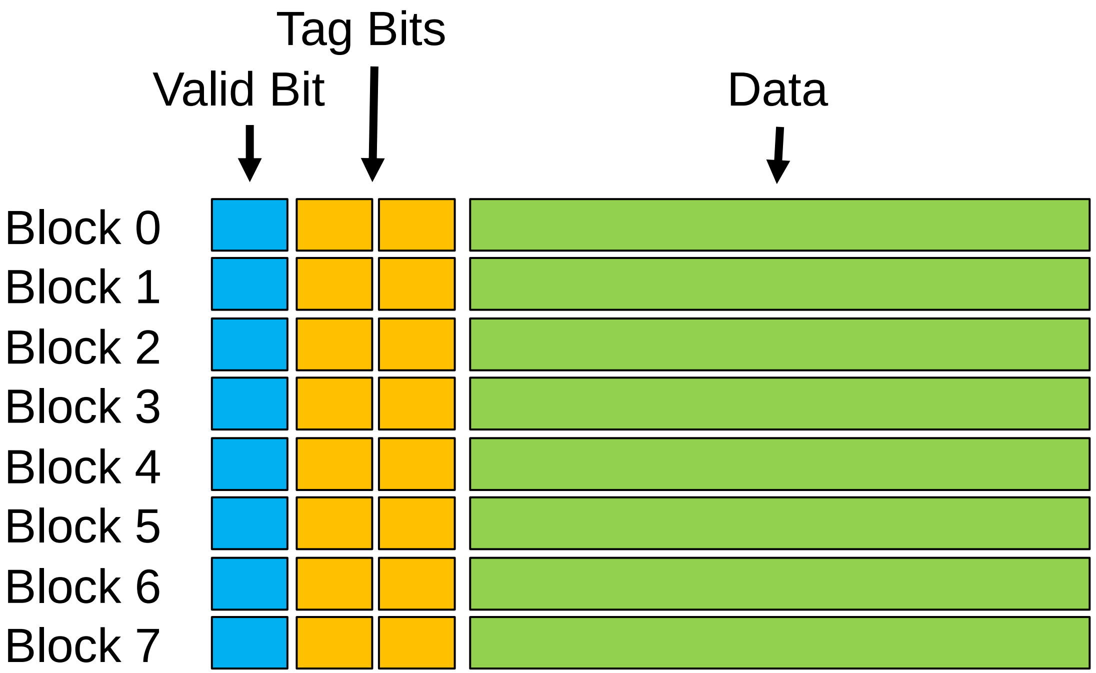
Tag Bits

- How do we know which particular block is stored in a cache location?
  - Store block address as well as the data
  - Actually, only need the high-order bits (**Tag**)
- What if there is no data in a location?
  - **Valid bit**: 1 = present, 0 = not present
  - Initially 0



5	0	0	1	0	1
13	0	1	1	0	1
21	1	0	1	0	1
19	1	1	1	0	1

8 Block Cache



Cache Example

- 8-blocks, 1 word/block, direct mapped
- Initial state

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Miss	110

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	**Y**	**10**	**Mem[10110]**
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
26	11 010	Miss	010

Index	V	Tag	Data
000	N		
001	N		
010	**Y**	**11**	**Mem[11010]**
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Hit	110
26	11 010	Hit	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
16	10 000	Miss	000
3	00 011	Miss	011
16	10 000	Hit	000

Index	V	Tag	Data
000	**Y**	**10**	**Mem[10000]**
001	N		
010	Y	11	Mem[11010]
011	**Y**	**00**	**Mem[00011]**
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Cache Example

Word addr	Binary addr	Hit/miss	Cache block
18	10 010	Miss	010

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	**Y**	**11**	**Mem[11010] (Address 26)**
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Valid data already present
(from different address)

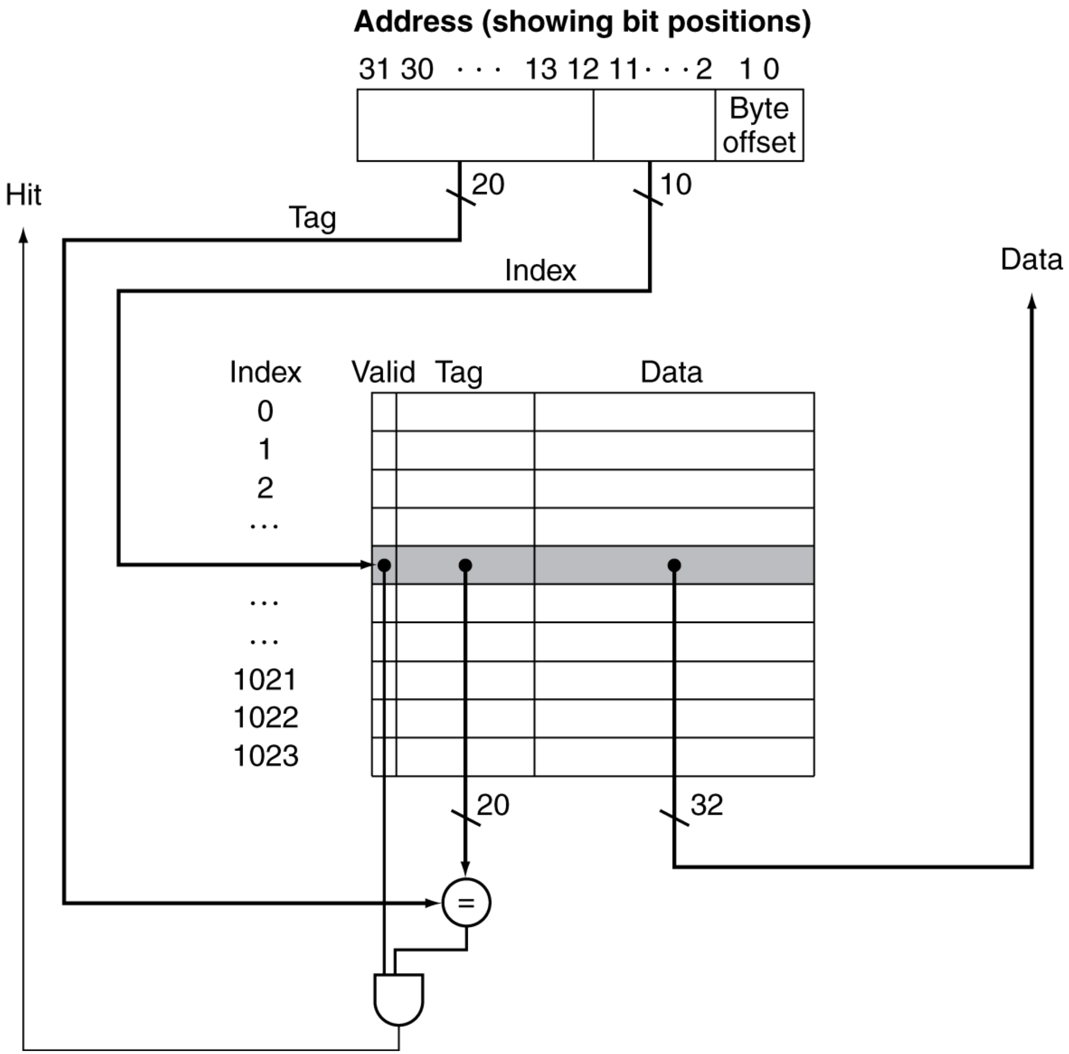
Cache Example

Word addr	Binary addr	Hit/miss	Cache block
18	10 010	Miss	010

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	**Y**	**10**	**Mem[10010]**
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

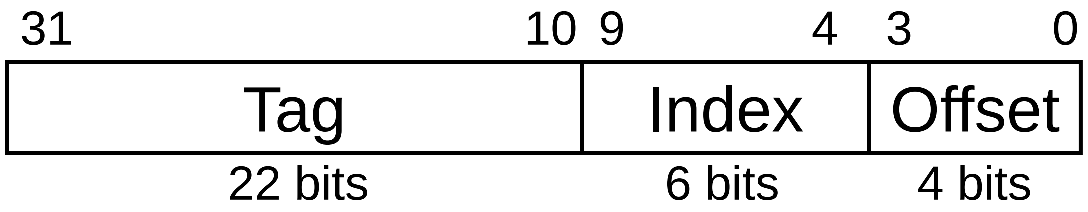
Replace

Address Subdivision



Example: Larger Block Size

- 64 blocks, 16 bytes/block
  - To what block number does address 1200 map?
- Block address (in main memory) = $\lfloor 1200/16 \rfloor = 75$
- Block number (in cache) = 75 modulo 64 = 11



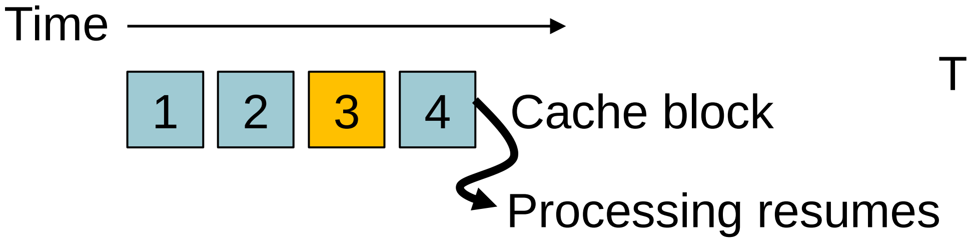
31 10	9 4	3 0
Tag	Index	Offset
22 bits	6 bits	4 bits

Block Size Considerations

- Larger blocks should reduce miss rate
  - Due to spatial locality
- But in a fixed-sized cache
  - Larger blocks ⇒ fewer of them
    - More competition ⇒ increased miss rate
  - Larger blocks ⇒ pollution
- Larger miss penalty
  - Can override benefit of reduced miss rate
  - Early restart and critical-word-first can help

Early Restart and Critical-word-first

Default: wait for block



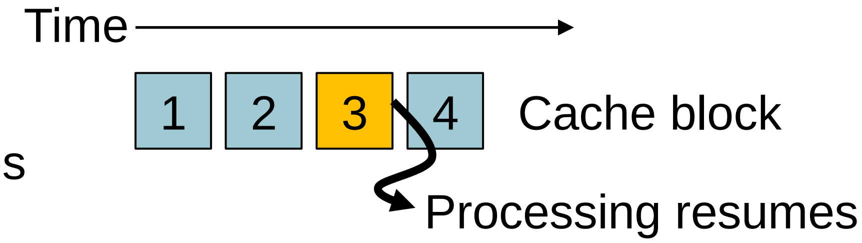
Time

1 2 3 4

Cache block

Processing resumes

Early Restart: start as soon as requested word arrives



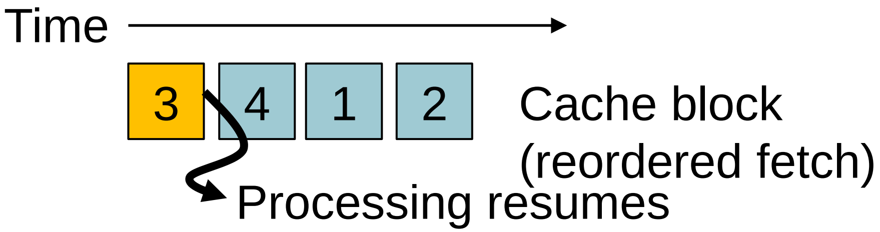
Time

1 2 3 4

Cache block

Processing resumes

Critical-word-first: start fetching from requested word



Time

3 4 1 2

Cache block (reordered fetch)

Processing resumes