CMPE 200
Computer Architecture & Design

Lecture 4.
Memory Hierarchy (2)

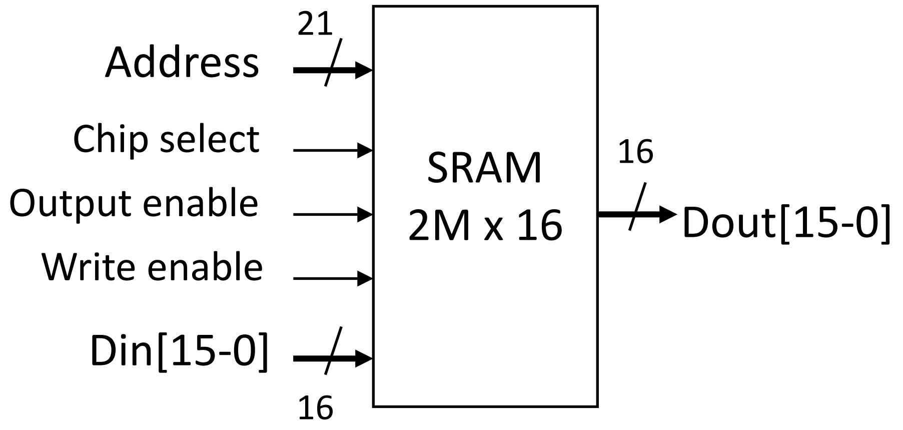
Haonan Wang

SJSU SAN JOSÉ STATE UNIVERSITY

Cache Design

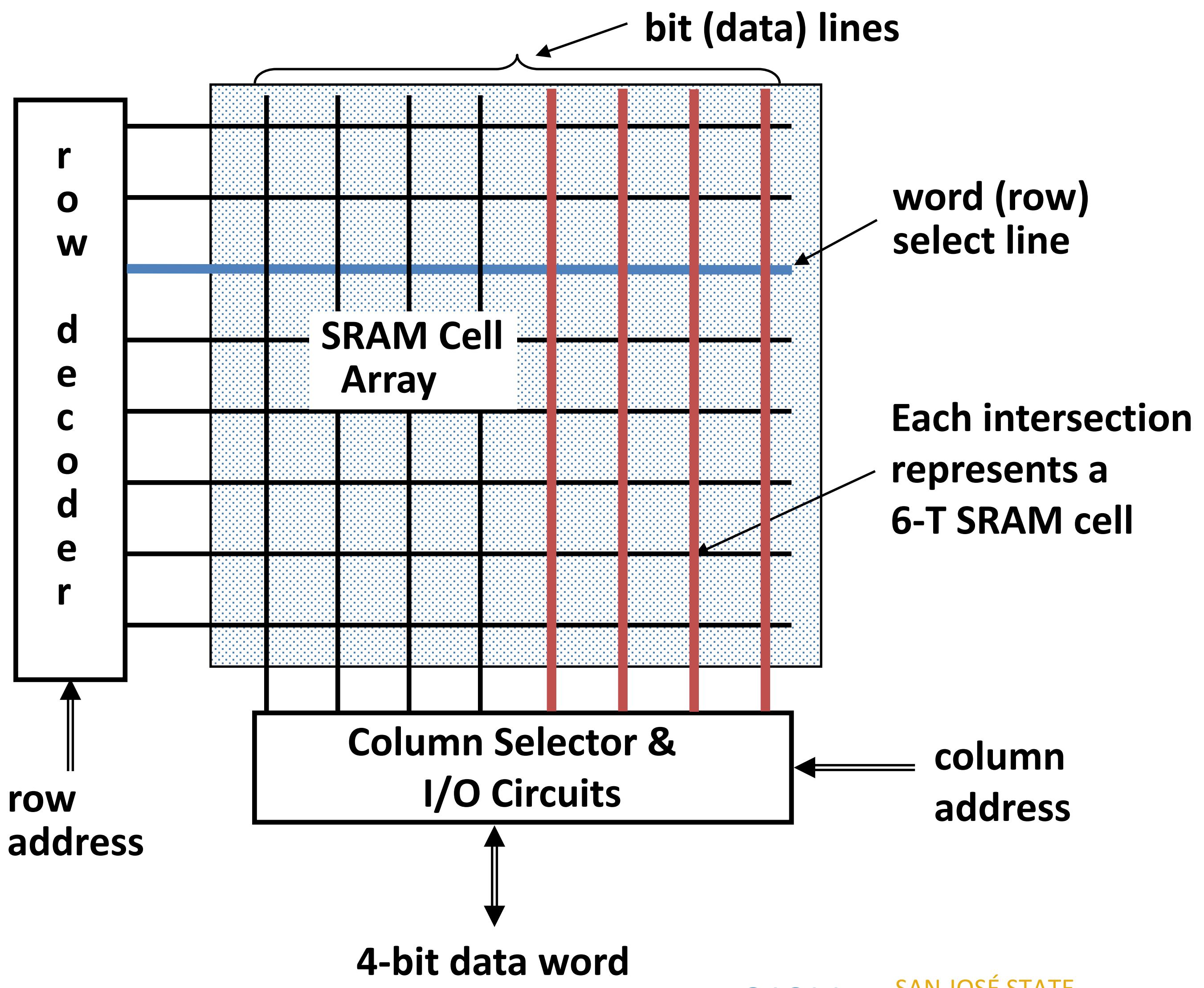
- **Caches use *SRAM* for speed and technology compatibility**
  - Low density (6 transistor cells), high power, expensive, fast
  - Static: content will last “forever” (until power turned off)

- **Example: A (2M X 16 bit) SRAM logic**



SRAM Cache Design

- Each row holds a data block
- Column address selects the requested word from block

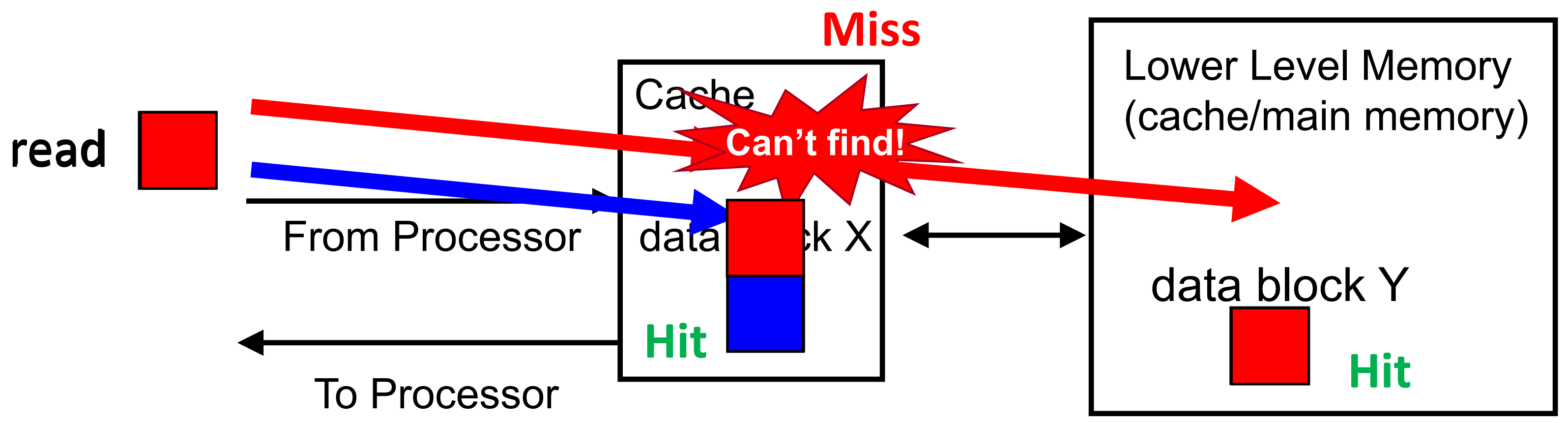


SJSU SAN JOSÉ STATE UNIVERSITY

Cache Hit and Miss

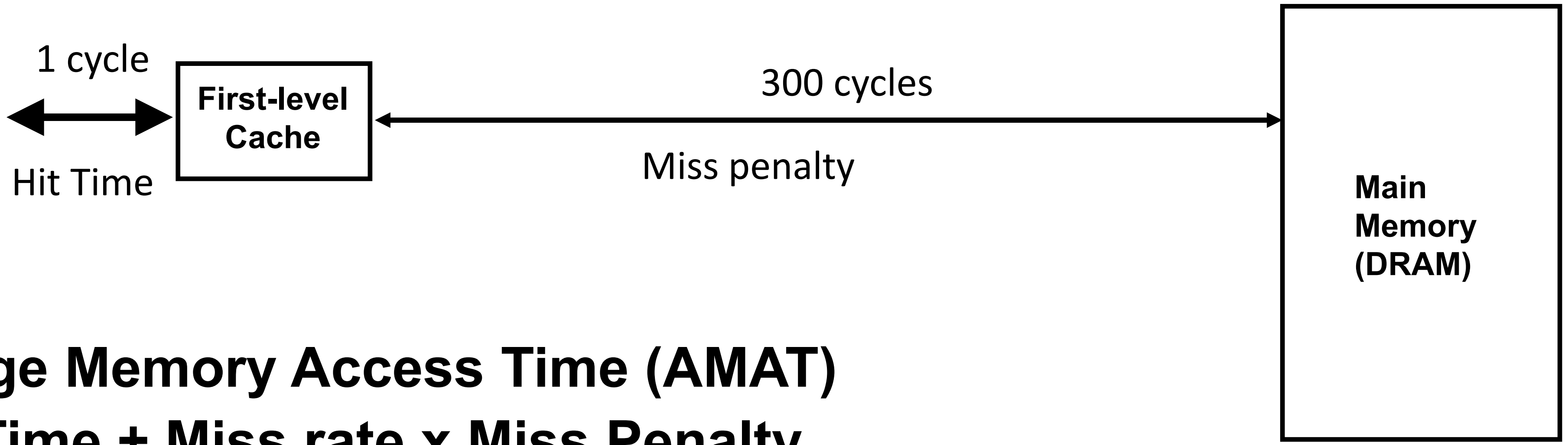
- **Hit:** Data appears in some block of the cache
  - **Hit Rate:** # hits / total accesses on the cache
  - **Hit Time:** Time to access the cache

- **Miss:** Data needs to be retrieved from the lower level (and stored in cache)
  - **Miss Rate:** 1 - (Hit Rate)
  - **Miss Penalty:** Average delay in the processor caused by each miss



SJSU SAN JOSÉ STATE UNIVERSITY

Memory Hierarchy Performance

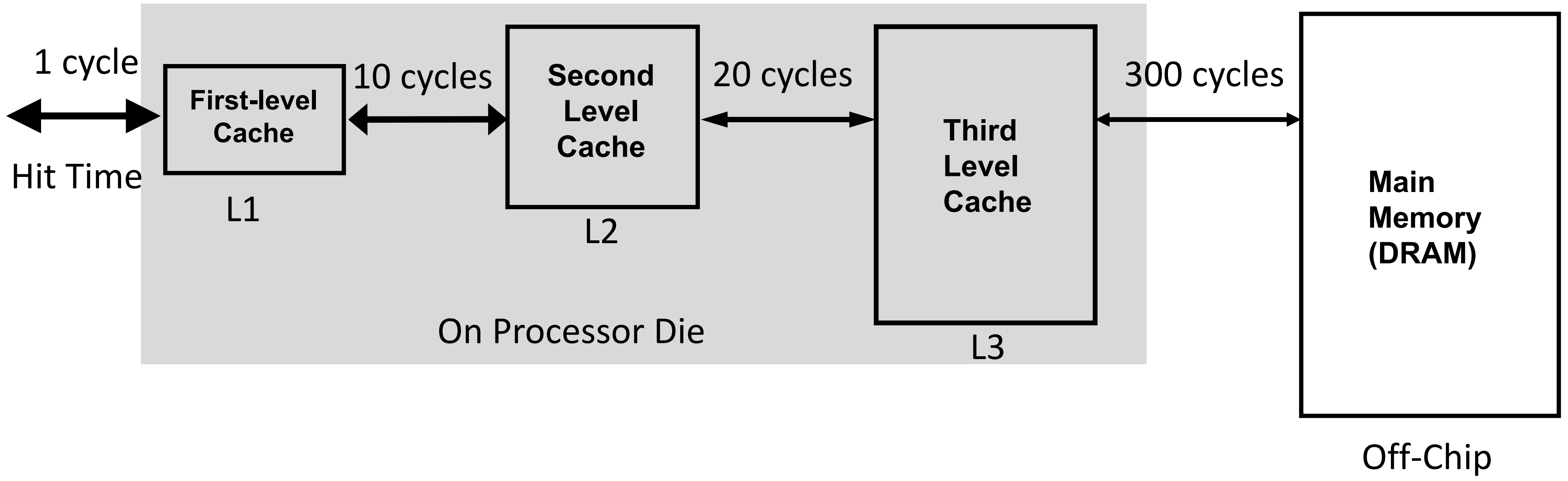


- **Average Memory Access Time (AMAT)**
  = Hit Time + Miss rate x Miss Penalty

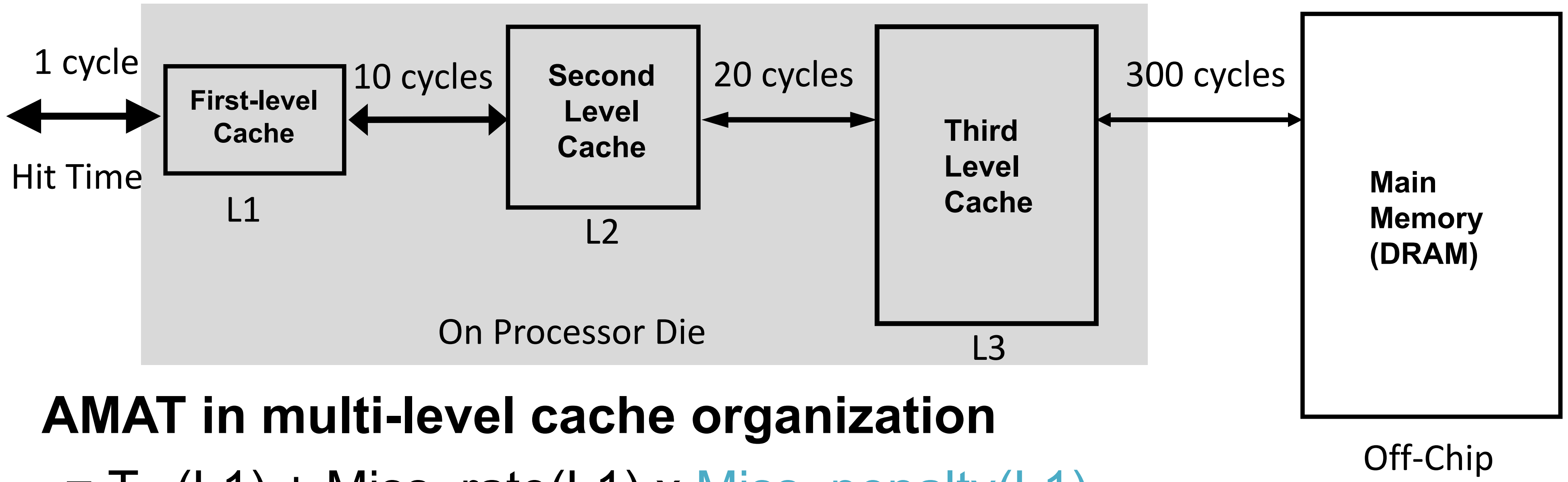
- **Example:**
  - Cache Hit = 1 cycle
  - Miss rate = 10% = 0.1
  - Miss penalty = 300 cycles
  - AMAT = T_{hit}(L1) + Miss_rate(L1) x T(Memory) = 1 + 0.1 x 300 = 31 cycles

Due to long-latency memory, AMAT is 30 cycles longer than cache latency.
Can we reduce the overhead?

Reducing Penalty: Multi-Level Cache



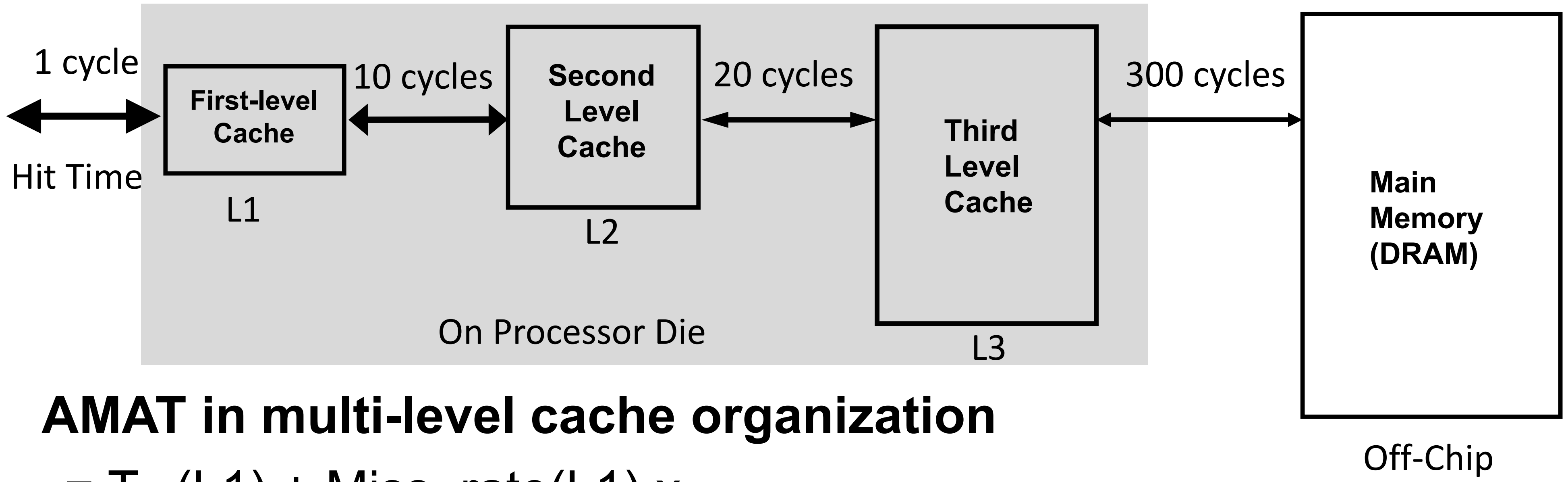
Reducing Penalty: Multi-Level Cache



- **AMAT in multi-level cache organization**

  $= T_{hit}(L1) + \text{Miss\_rate}(L1) \times \text{Miss\_penalty}(L1)$

Reducing Penalty: Multi-Level Cache

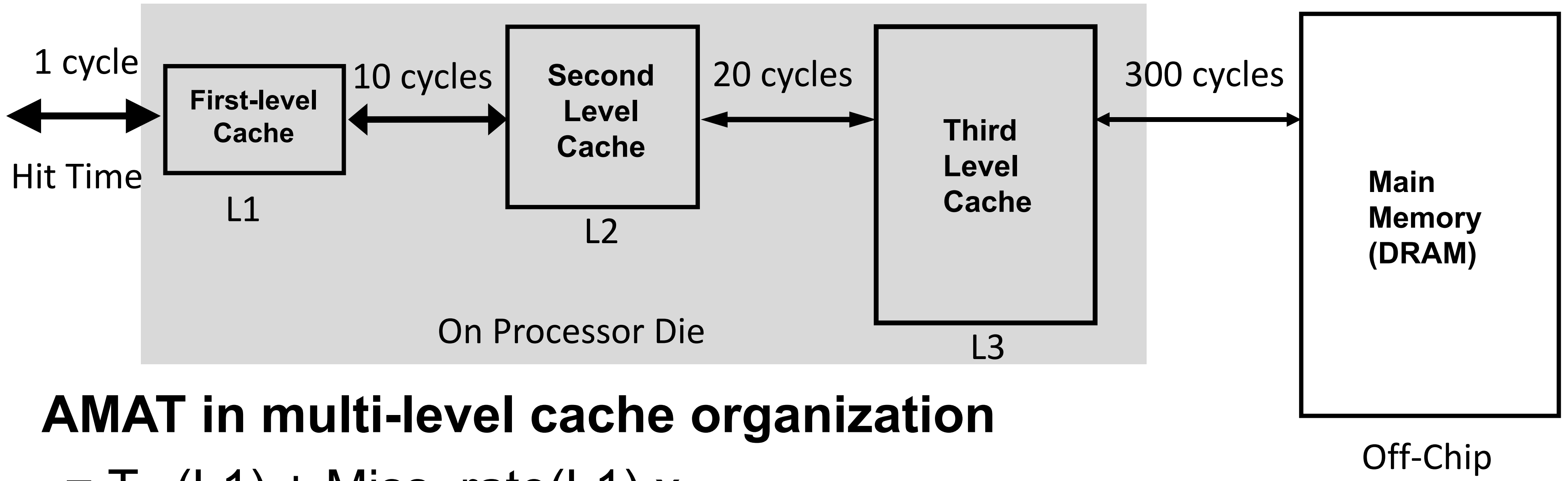


- **AMAT in multi-level cache organization**

  $= T_{hit}(L1) + Miss\_rate(L1) \times$

  $[\ T_{hit}(L2) + Miss\_rate(L2) \times Miss\_penalty(L2)\ ]$

Reducing Penalty: Multi-Level Cache



- **AMAT in multi-level cache organization**

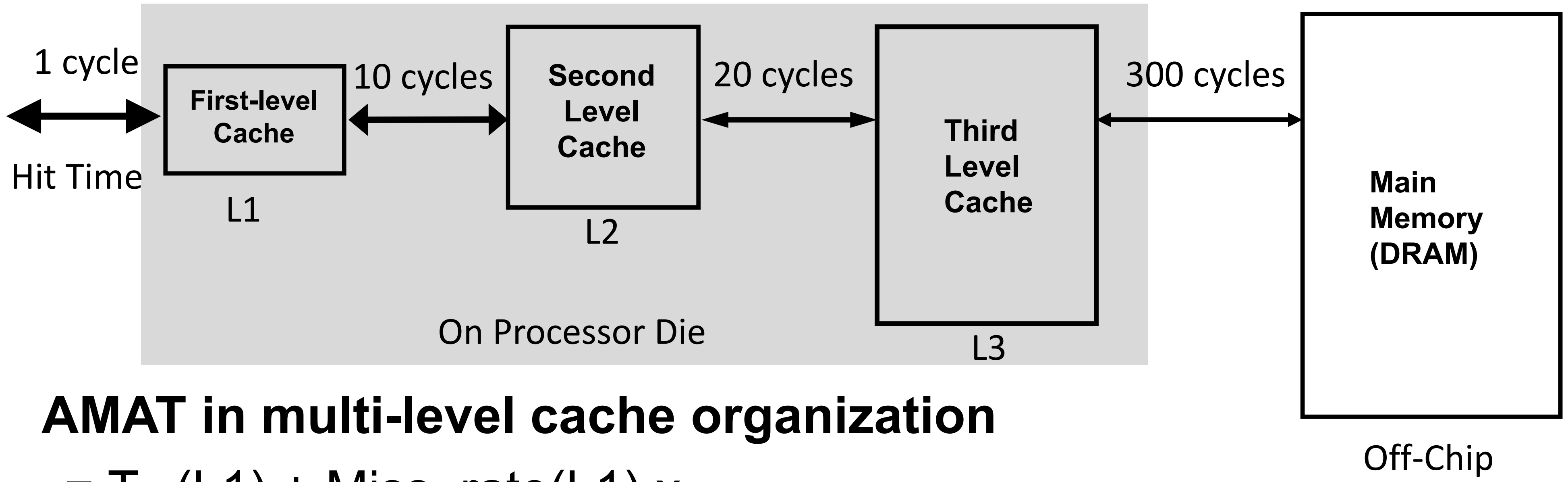
  $= T_{hit}(L1) + Miss\_rate(L1) \times$

  $[\ T_{hit}(L2) + Miss\_rate(L2) \times$

  $\{\ T_{hit}(L3) + Miss\_rate(L3) \times Miss\_penalty(L3)\ \}\ ]$

SJSU SAN JOSÉ STATE UNIVERSITY

Reducing Penalty: Multi-Level Cache



- **AMAT in multi-level cache organization**

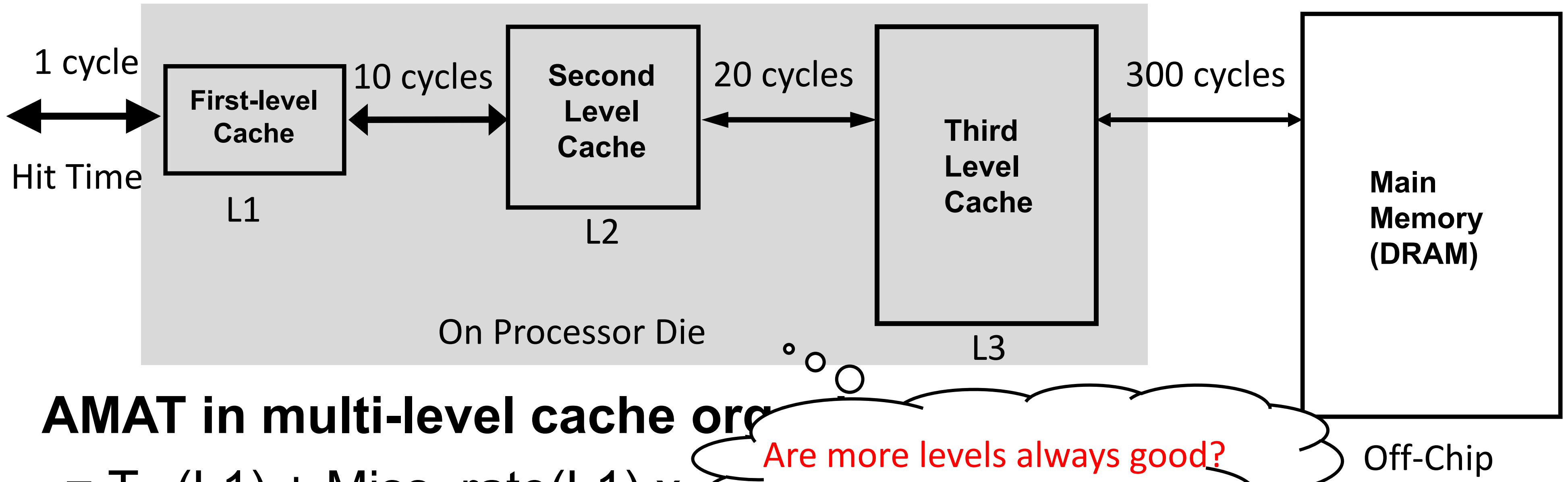
  $= T_{hit}(L1) + \text{Miss\_rate}(L1) \times$

  $[\ T_{hit}(L2) + \text{Miss\_rate}(L2) \times$

  $\{\ T_{hit}(L3) + \text{Miss\_rate}(L3) \times T(\text{memory})\ \}\ ]$

SJSU SAN JOSÉ STATE UNIVERSITY

Reducing Penalty: Multi-Level Cache



Are more levels always good?

- **AMAT in multi-level cache org**
  = $T_{hit}(L1)$ + Miss_rate(L1) x
  [$T_{hit}(L2)$ + Miss_rate(L2) x
  { $T_{hit}(L3)$ + Miss_rate(L3) x T(memory) }]
- **Example:**
  - Miss rate of L1, L2, L3 = 10%, 5%, 1%, respectively
  - AMAT = 1 + 0.1 x [10 + 0.05 x { 20 + 0.01 x 300 }] = 2.115 cycles

Vs. 31 cycles
14.7x speedup!

Discussion

Background: The cache space is limited. We can only keep a subset of data in cache.

Question: What happens when the cache is full?

Question: What to keep in the cache (which block should be evicted)?

What to Keep in Caches (1)?

- **It depends on the cache organization and replacing policy.**

	Way 0	Way 1	…
Set 0	block 0	block 2	
Set 1	block 1	block 3	
⋮			

- **Cache organization:**
  - **Cache line (block):** The basic unit of data replacement. A longer cache line fetches and replaces more data per miss.
  - **Set:** An entry that one cache line is mapped to according to certain bits of its address.
  - **Way:** A slot within a cache set to hold one history cache line.

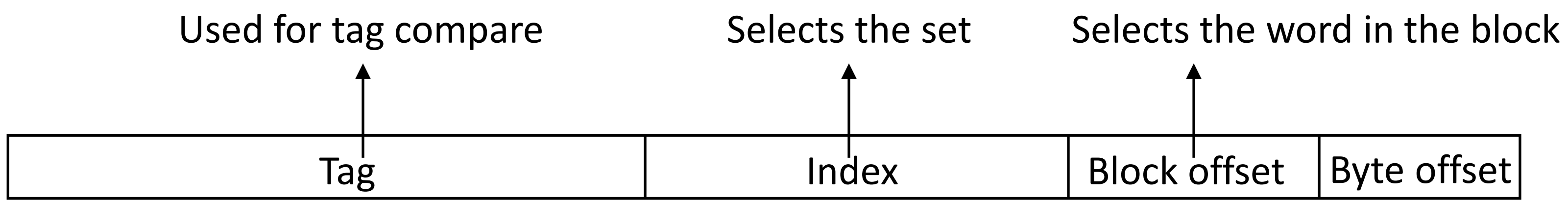
What to Keep in Caches (2)?

- **It depends on the cache organization and replacing policy**

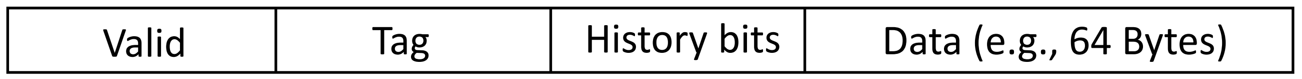
- **Cache line replacing policy:** which history line should be replaced?
  - In a set entry, each cache line can be identified with a **Tag** (a portion of its address).

  - Least recently used (LRU), First-in first-out (FIFO), Random, etc.

Address decoding:



Cache line:

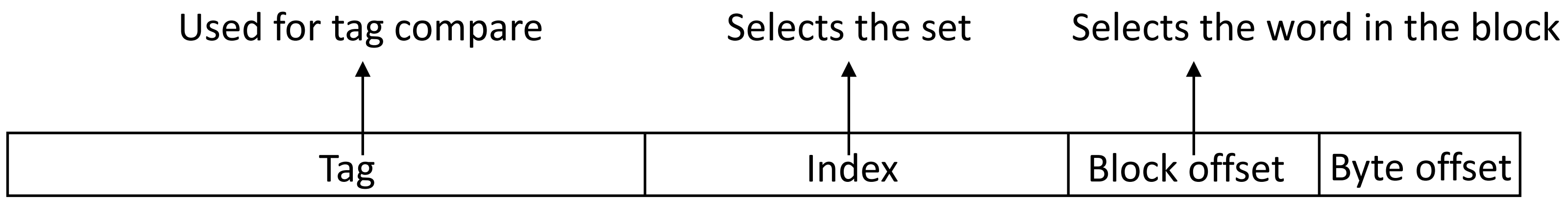


Cache Indexing

Example: a simple 2-set x 2-way cache

	Way 0	**Way 1**
Set 0	block 0	block 2
Set 1	block 1	block 3

Address decoding:



SJSU SAN JOSÉ STATE UNIVERSITY

Cache Types

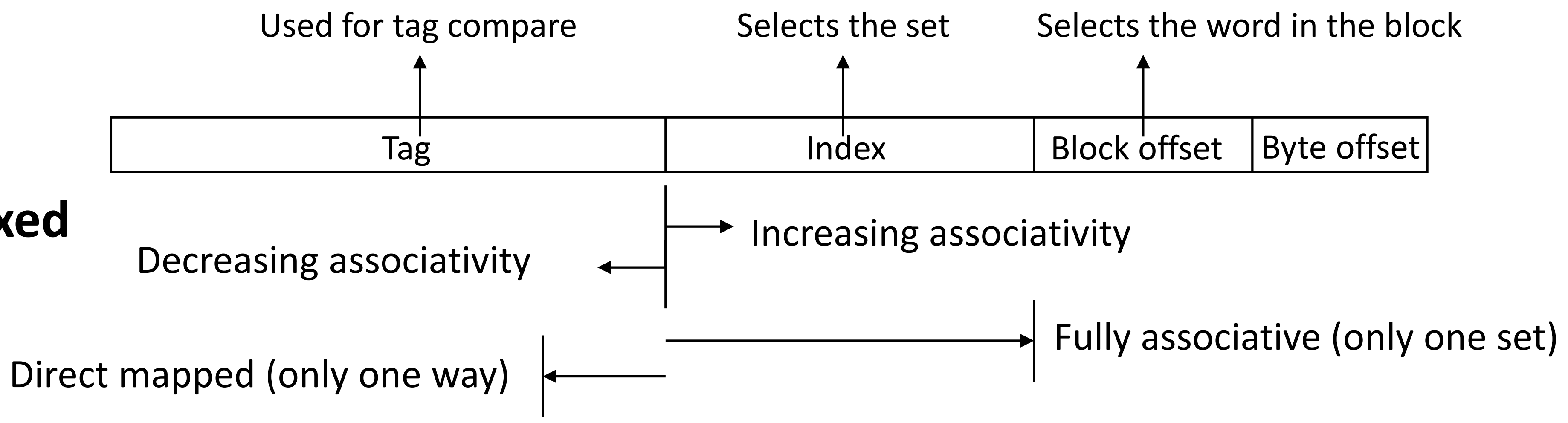
- **N-way Set-Associative:** Number of ways > 1 & Number of sets > 1
  - Slightly complex searching mechanism
- **Direct Mapped:** Number of ways = 1
  - Fast indexing mechanism
- **Fully-Associative:** Number of sets = 1
  - Extensive hardware resources required to search

	Way 0	**Way 1**	…
Set 0	block 0	block 2	
Set 1	block 1	block 3	
⋮			

Cache Types

	Way 0	**Way 1**	**…**
Set 0	block 0	block 2	
Set 1	block 1	block 3	
⋮			

Assuming fixed sized cache:



SAN JOSÉ STATE UNIVERSITY *powering* SILICON VALLEY