دانشگاه کردستان
University of Kurdistan
زانکۆی کوردستان

Department of Computer Engineering
University of Kurdistan

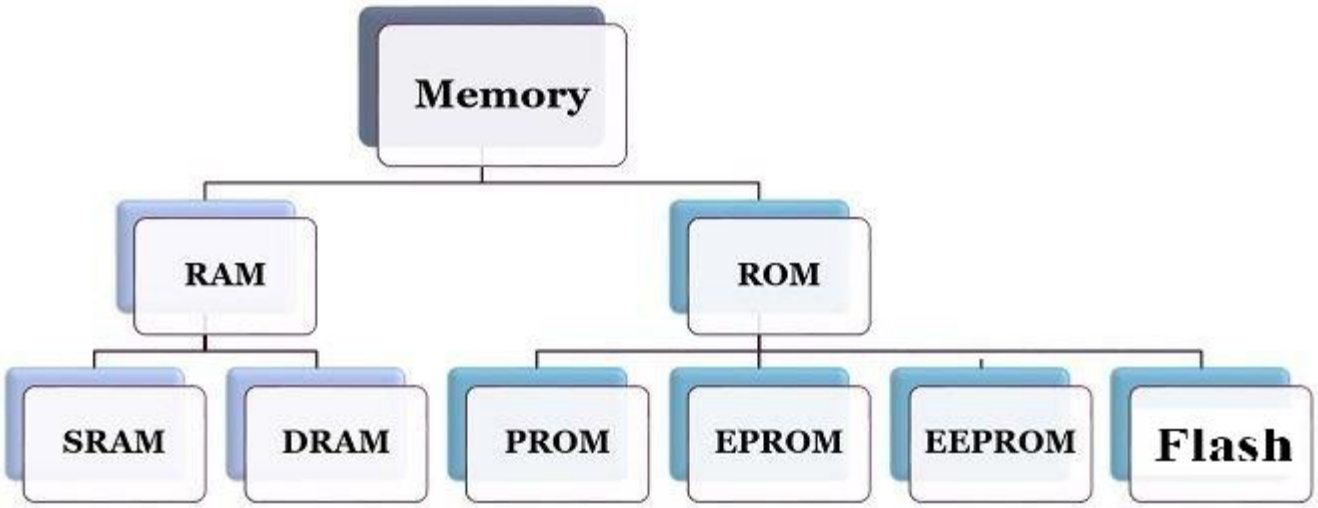
Computer Architecture

Memory Hierarchy

By: Dr. Alireza Abdollahpouri

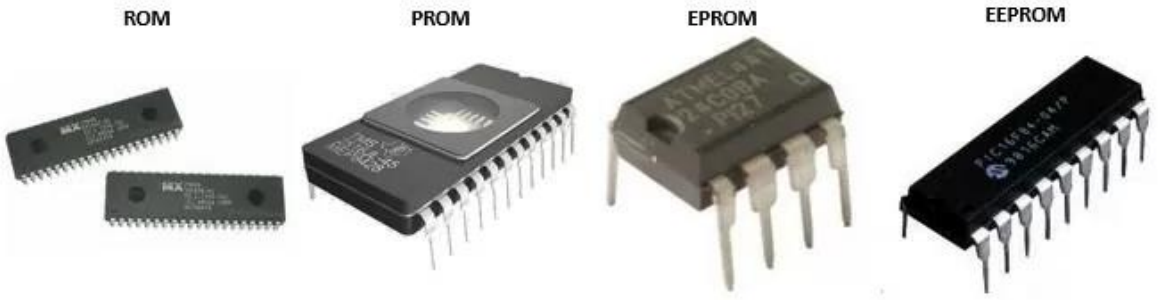
حافظه اصلی

- حافظه اصلی از مدارات سریعی ساخته میشود که برنامه‌ها و داده‌های مورد نیاز را در هنگام اجرا نگهداری مینماید.
- بر دو نوع است:
  ۱. ROM
  ۲. RAM



Read Only Memory (ROM)

- Permanent storage
  - Nonvolatile
- Used in:
  - Microprogramming
  - Library subroutines
  - Systems programs (BIOS)
  - Function tables

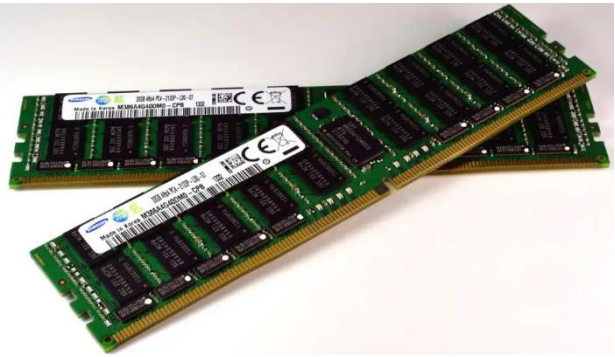


Types of ROM

- Written during manufacture
  - Very expensive for small runs
- Programmable (once)
  - PROM
  - Needs special equipment to program
- Read “mostly”
  - Erasable Programmable (EPROM)
    - Erased by UV
  - Electrically Erasable (EEPROM)
    - Takes much longer to write than read
  - Flash memory
    - Erase whole memory electrically

RAM Technology

- Static RAM (SRAM) for Cache
  - Requires 6 transistors per bit
  - Requires low power to retain bit
- Dynamic RAM (DRAM) for Main Memory
  - One transistor + capacitor per bit
  - Must be re-written after being read
  - Must also be periodically refreshed
    - Each row can be refreshed simultaneously
  - Address lines are multiplexed
    - Upper half of address: Row Access Strobe (RAS)
    - Lower half of address: Column Access Strobe (CAS)

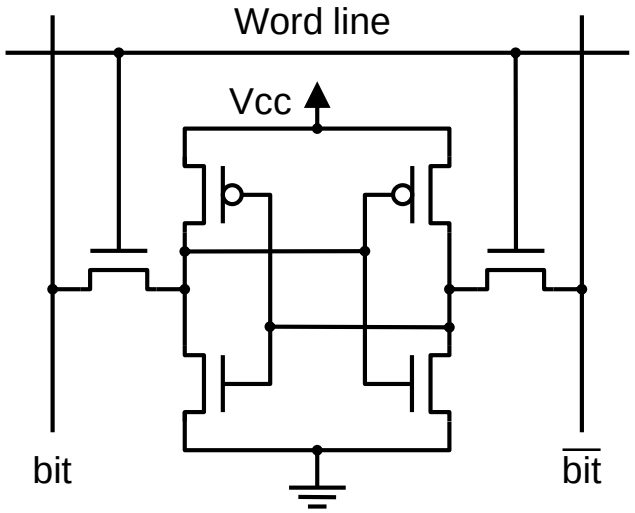


DRAM vs. SRAM

- **DRAM**
  - Slower access (capacitor)
  - Higher density (1T 1C cell)
  - Lower cost
  - Requires refresh (power, performance, circuitry)
  - Manufacturing requires putting capacitor and logic together
- **SRAM**
  - Faster access (no capacitor)
  - Lower density (6T cell)
  - Higher cost
  - No need for refresh
  - Manufacturing compatible with logic process (no capacitor)

Static RAM Storage Cell

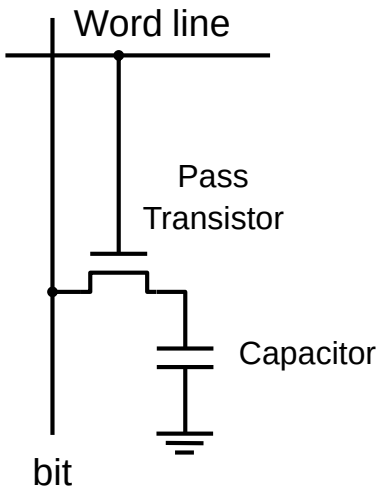
- Static RAM (SRAM): fast but expensive RAM
- 6-Transistor cell with no static current
- Typically used for **caches**
- Provides fast access time
- Cell Implementation:
  - Cross-coupled inverters store bit
  - Two pass transistors
  - Row decoder selects the word line
  - Pass transistors enable the cell to be read and written



Typical SRAM cell

Dynamic RAM Storage Cell

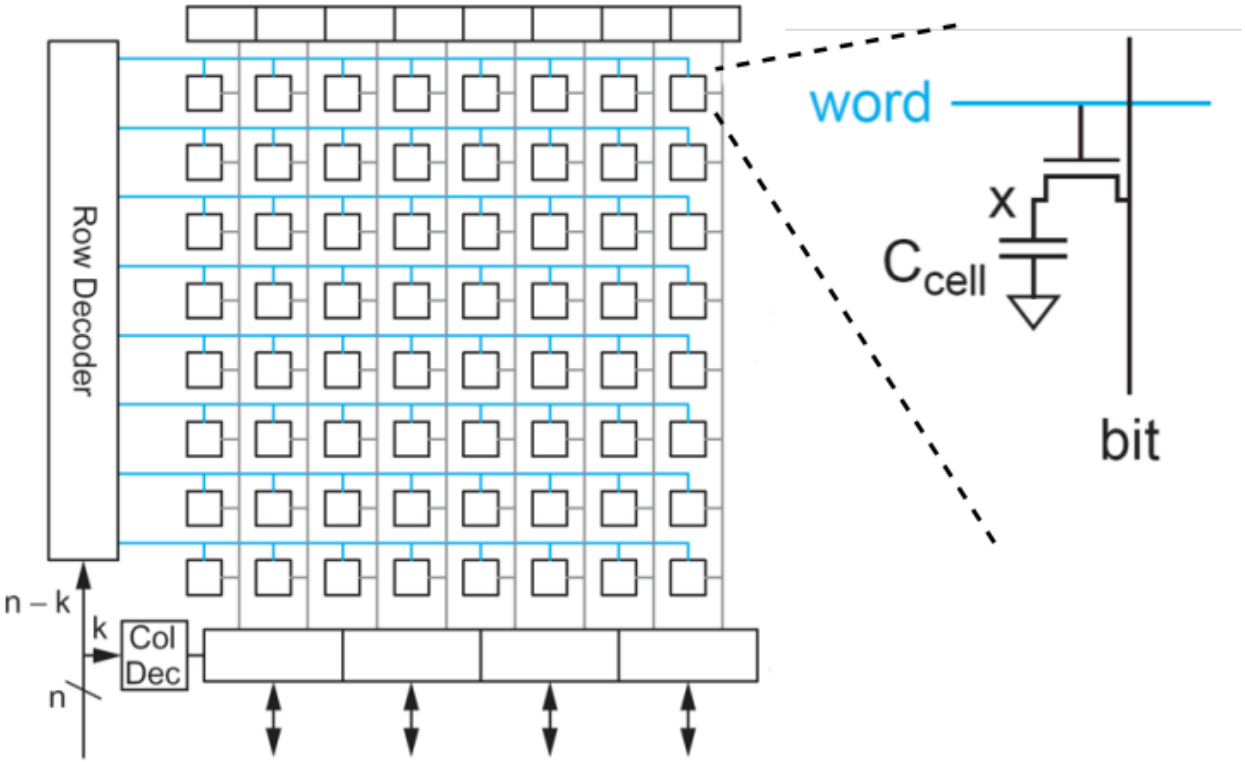
- Typical choice for **main memory**
- Cell Implementation:
  - 1-Transistor cell (pass transistor)
  - Trench capacitor (stores bit)
- Bit is stored as a charge on capacitor
- Must be refreshed periodically
  - Because of leakage of charge from tiny capacitor
- Refreshing for all memory rows
  - Reading each row and writing it back to restore the charge



Typical DRAM cell

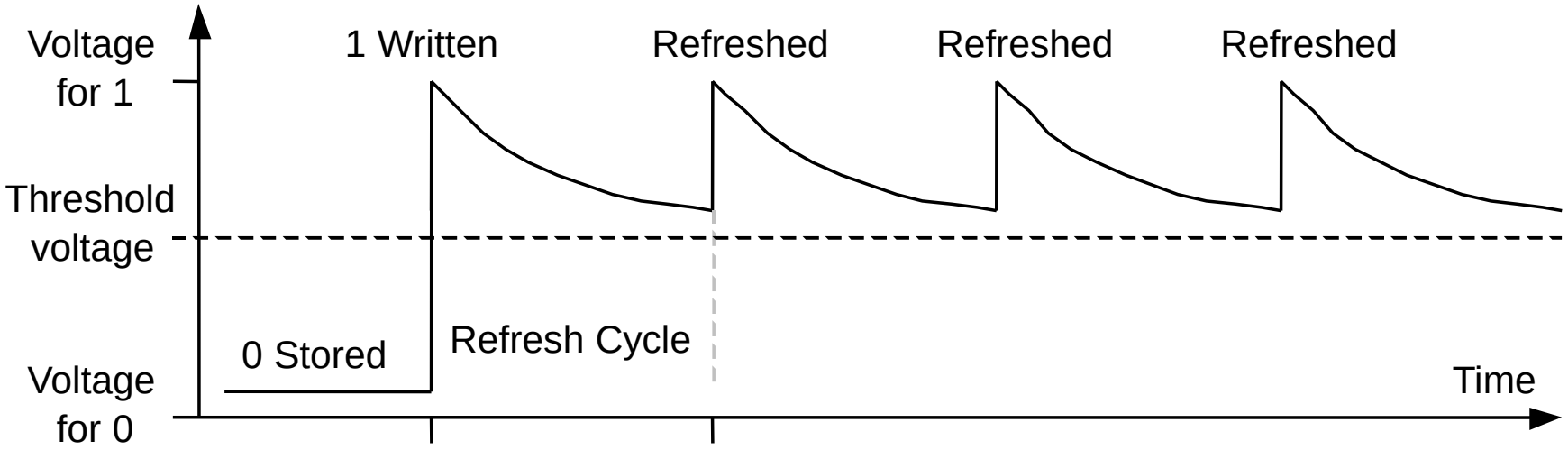
DRAM Structure

Memory Arrays: DRAM

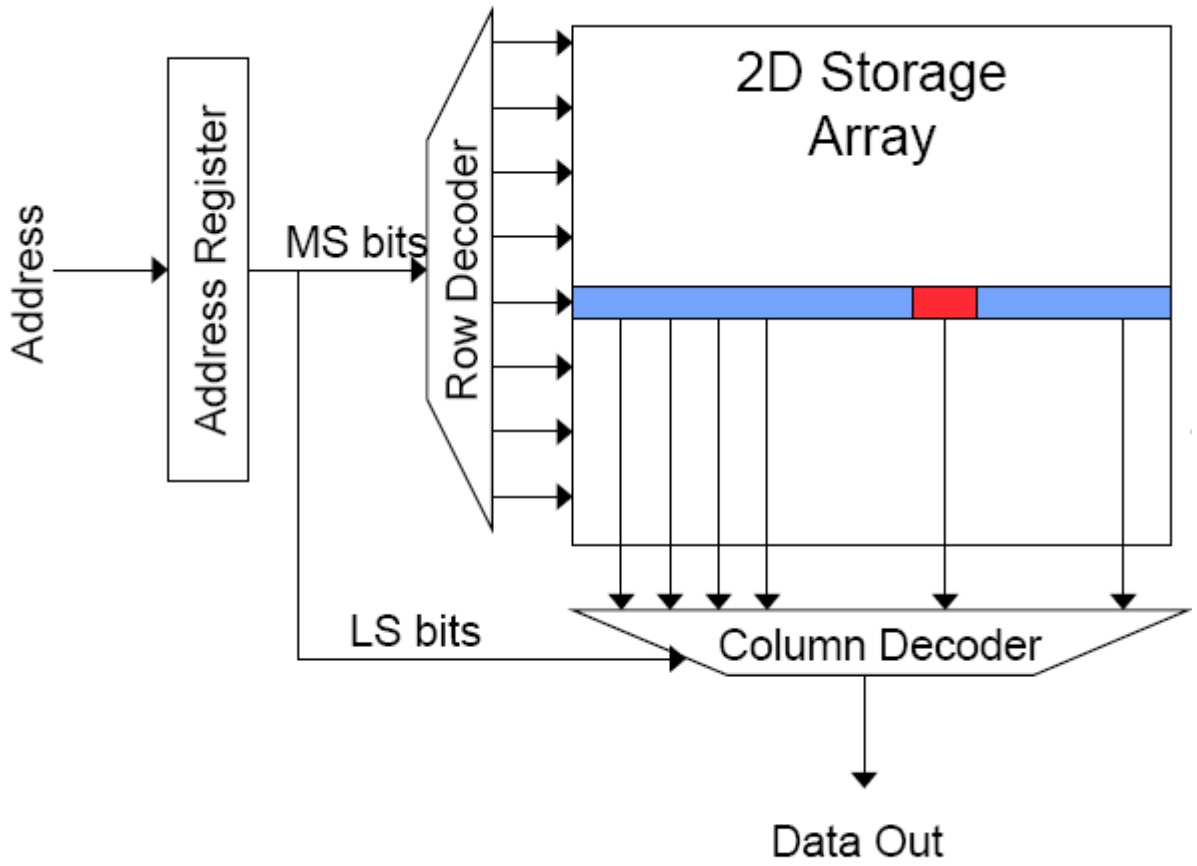


DRAM Refresh Cycles

- Refresh cycle is about tens of milliseconds
- Refreshing is done for the entire memory
- Each row is read and written back to restore the charge
- Some of the memory bandwidth is lost to refresh cycles



Memory Bank Organization and Operation



- Read access sequence:

1. Decode row address & drive word-lines

2. Selected bits drive bit-lines
   - Entire row read

3. Amplify row data

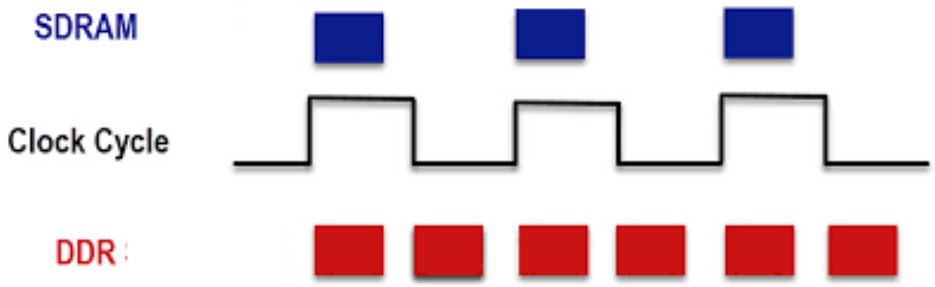
4. Decode column address & select subset of row
   - Send to output

5. Precharge bit-lines
   - For next access

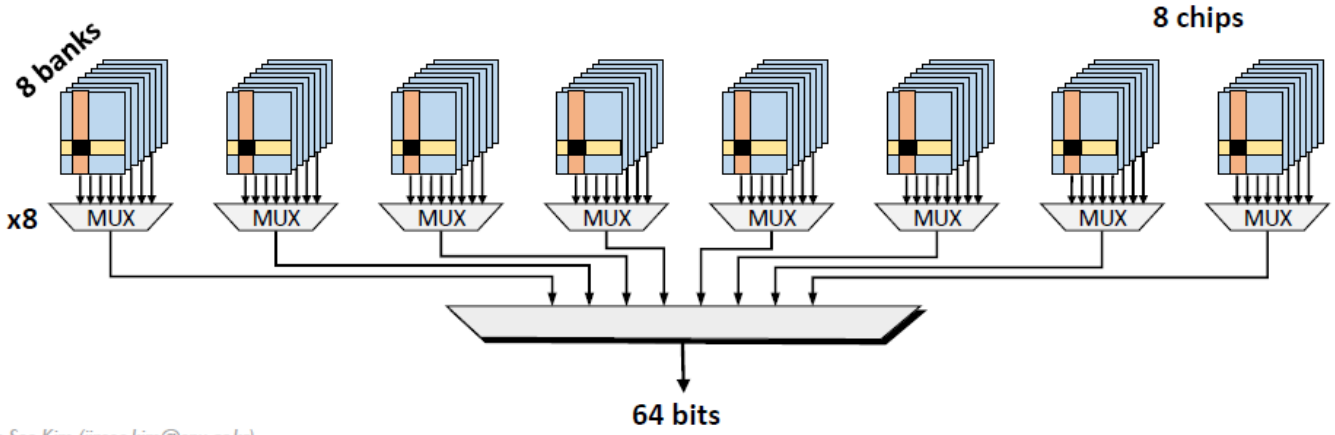
SDRAM and DDR SDRAM

- SDRAM is Synchronous Dynamic RAM
  - Added clock to DRAM interface
- SDRAM is synchronous with the system clock
  - Older DRAM technologies were asynchronous
  - As system bus clock improved, SDRAM delivered higher performance than asynchronous DRAM
- DDR is Double Data Rate SDRAM
  - Like SDRAM, DDR is synchronous with the system clock, but the difference is that DDR reads data on both the rising and falling edges of the clock signal

DDR



- Double data rate (DDR) DRAM
  - Transfer on rising and falling clock edges
  - DDR4-3200 (PC25600): 64bits * 1600MHz * 2 = 25600MB/s



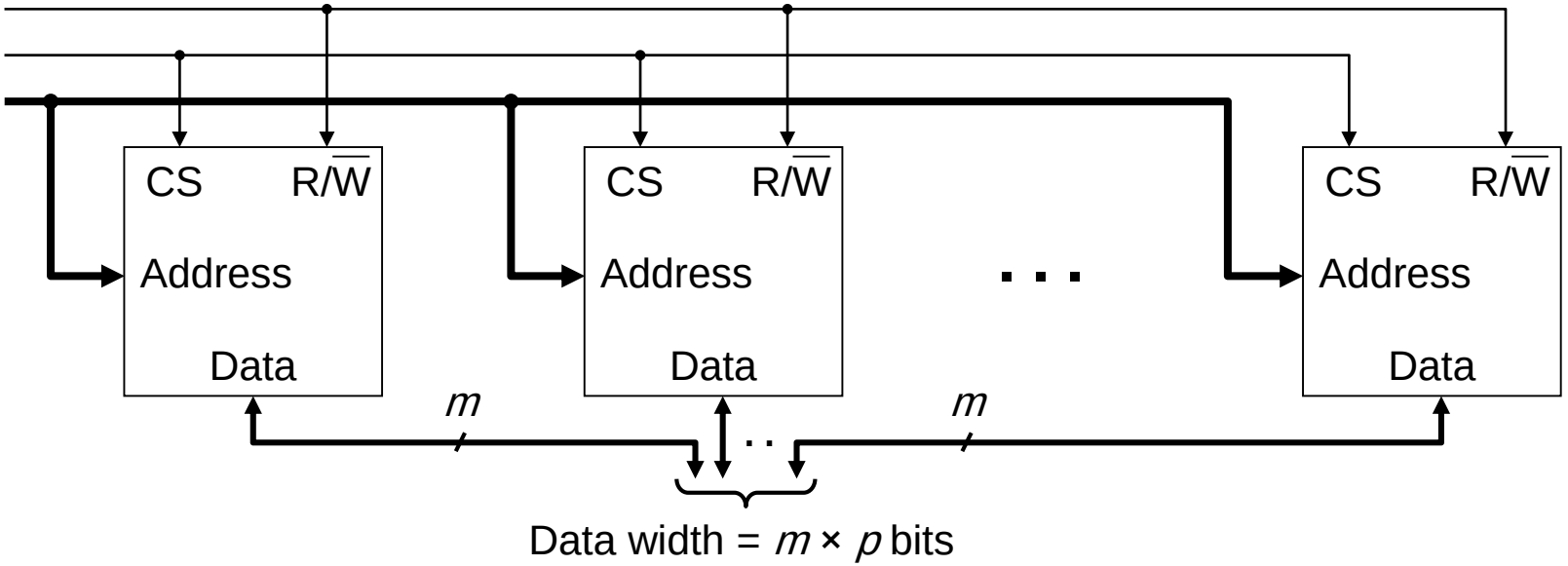
DDR Transfer Rates & Peak Bandwidth

Standard Name	Memory Bus Clock	Millions Transfers per second	Release Year	Peak Bandwidth
DDR-200	100 MHz	200 MT/s	1998	1600 MB/s
DDR-400	200 MHz	400 MT/s	1998	3200 MB/s
DDR2-667	333 MHz	667 MT/s	2003	5333 MB/s
DDR2-800	400 MHz	800 MT/s	2003	6400 MB/s
DDR3-1066	533 MHz	1066 MT/s	2007	8533 MB/s
DDR3-1600	800 MHz	1600 MT/s	2007	12800 MB/s
DDR4-3200	1600 MHz	3200 MT/s	2014	25600 MB/s
DDR5-3600	1800 MHz	3600 MT/s	2020	28800 MB/s
DDR5-7200	3600 MHz	7200 MT/s	2020	57600 MB/s

- 1 Transfer = 64 bits = 8 bytes of data

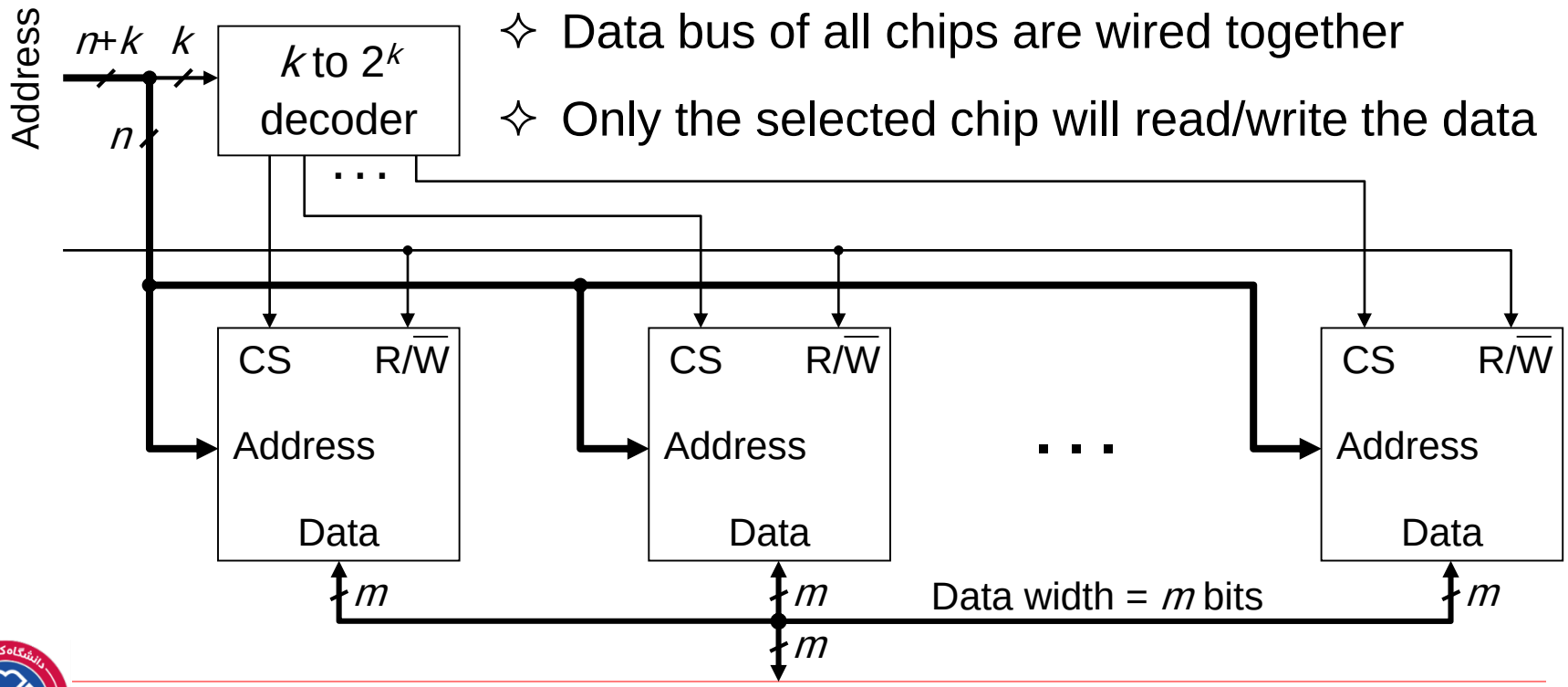
Expanding the Data Bus Width

- Memory chips typically have a narrow data bus
- We can expand the data bus width by a factor of p
  - Use p RAM chips and feed the same address to all chips
  - Use the same Chip Select and Read/Write control signals



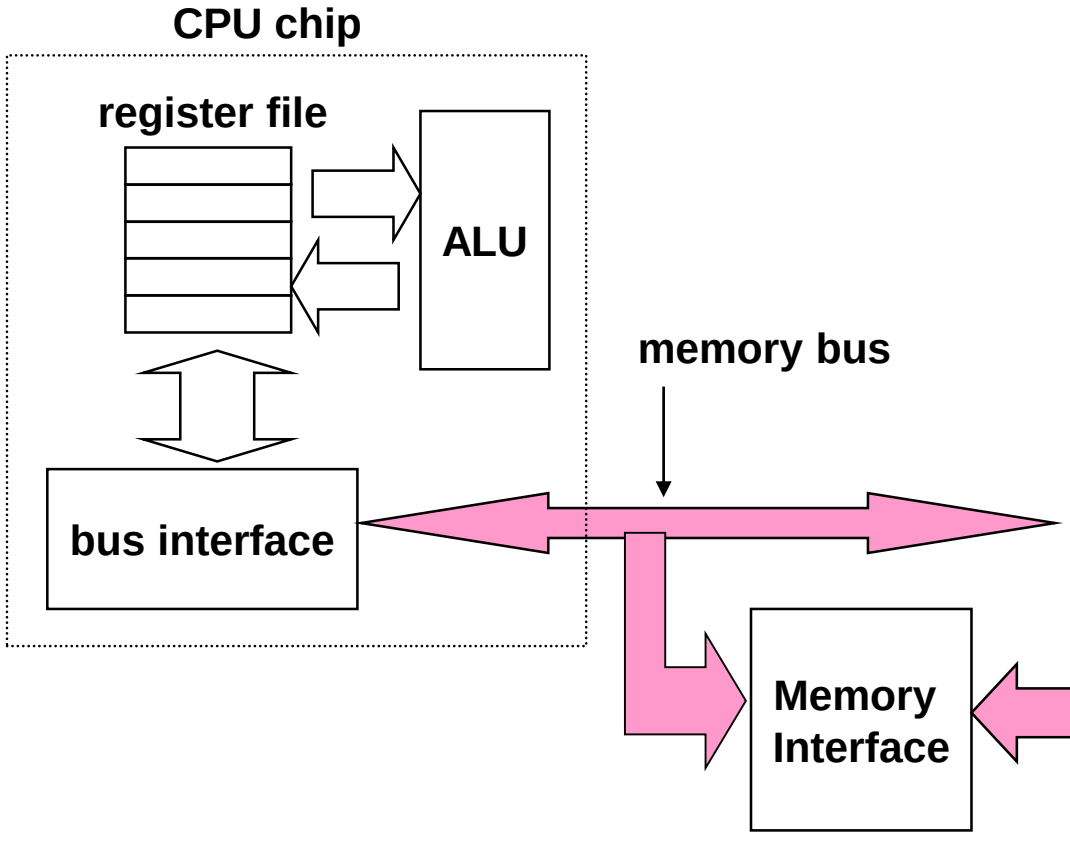
Increasing Memory Capacity by 2^k

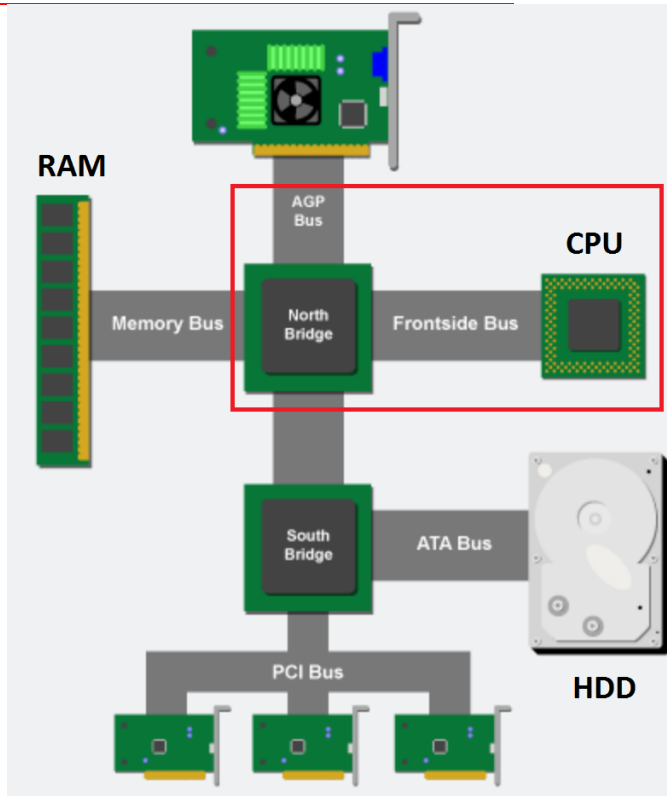
- A k to 2^k decoder is used to select one of the 2^k chips
  - Upper n bits of address is fed to all memory chips
  - Lower k bits of address are decoded to select one of the 2^k chips



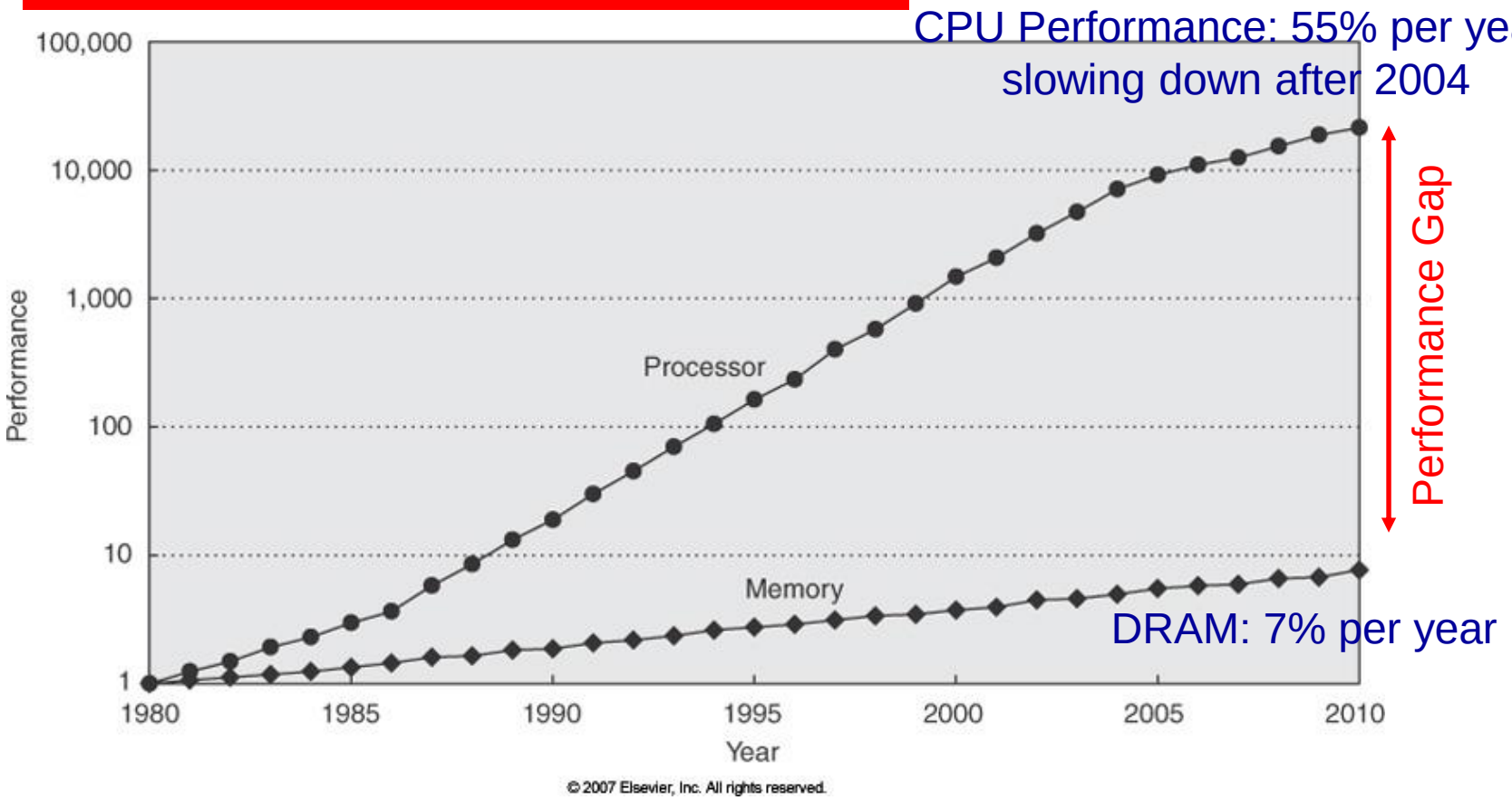
- Data bus of all chips are wired together
- Only the selected chip will read/write the data

ارتباط حافظه با پردازنده





Processor-Memory Performance Gap



- 1980 – No cache in microprocessor
- 1995 – Two-level cache on microprocessor

Technology Trends

- Latency (Cycle Time, Access Time to Memory) doesn't improve (or very slowly improves) overtime compared to DRAM size (capacity)

Year	DRAM Size		Cycle Time
1980	64 Kb	2:1!	250 ns
1983	256 Kb		220 ns
1986	1 Mb		190 ns
1989	4 Mb		165 ns
1992	16 Mb		145 ns
1995	64 Mb		120 ns
1998	256 Mb		100 ns
2001	1 Gb		80 ns
2012	4 Gb		35 ns

1000:1!

صورت مسئله

- ویژگیهای حافظه:
- حافظه های بزرگ کند هستند
- حافظه های سریع کوچک هستند

- سوال :چگونه میتوان حافظه بزرگ، سریع و ارزانی داشت؟
  ۱. استفاده از سلسله مراتب
  ۲. دسترسی موازی

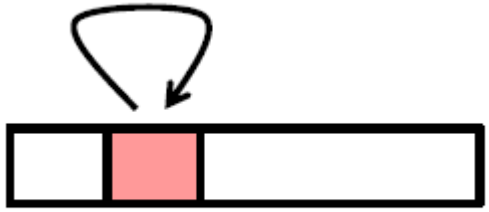
اصل محلی بودن رجوع به حافظه

- با آنالیز برنامه های کامپیوتری مشخص شده است که در یک فاصله زمانی رجوع به حافظه برای دسترسی به داده یا دستورالعمل معمولا در اطراف یک ناحیه حول داده ها و دستوراتی است که اخیرا مورد رجوع قرار گرفته اند.
- این امر به این دلیل است که برنامه ها دستورات را از محل های پشت سر هم اجرا میکنند و همچنین در هنگام اجرای حلقه ها دستورات آن حلقه بطور مدام تکرار میشوند.

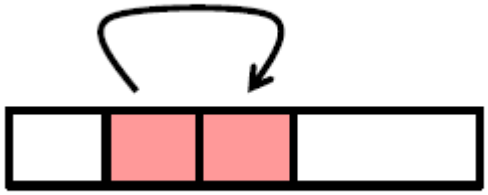
```
sum = 0;
for (i = 0; i < n; i++)
        sum += a[i];
return sum;
```

اصل محلی بودن رجوع به حافظه

- **محلی بودن رجوع به حافظه بدو صورت است:**



- **محلیت زمانی:Temporal locality**
- داده ها و دستوراتی که اخیرا مورد استفاده قرار گرفته اند، مجددا مورد رجوع قرار خواهند گرفت
- **محلیت مکانی:Spatial locality**



- داده ها یا دستوراتی که در خانه های نزدیک به هم حافظه قرار دارند با فاصله زمانی اندکی از هم مورد رجوع قرار خواهند گرفت.

مثال

کدامیک از برنامه های زیر دارای خاصیت رجوع محلی بهتری هستند؟

```
int sumarrayrows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum
}
```

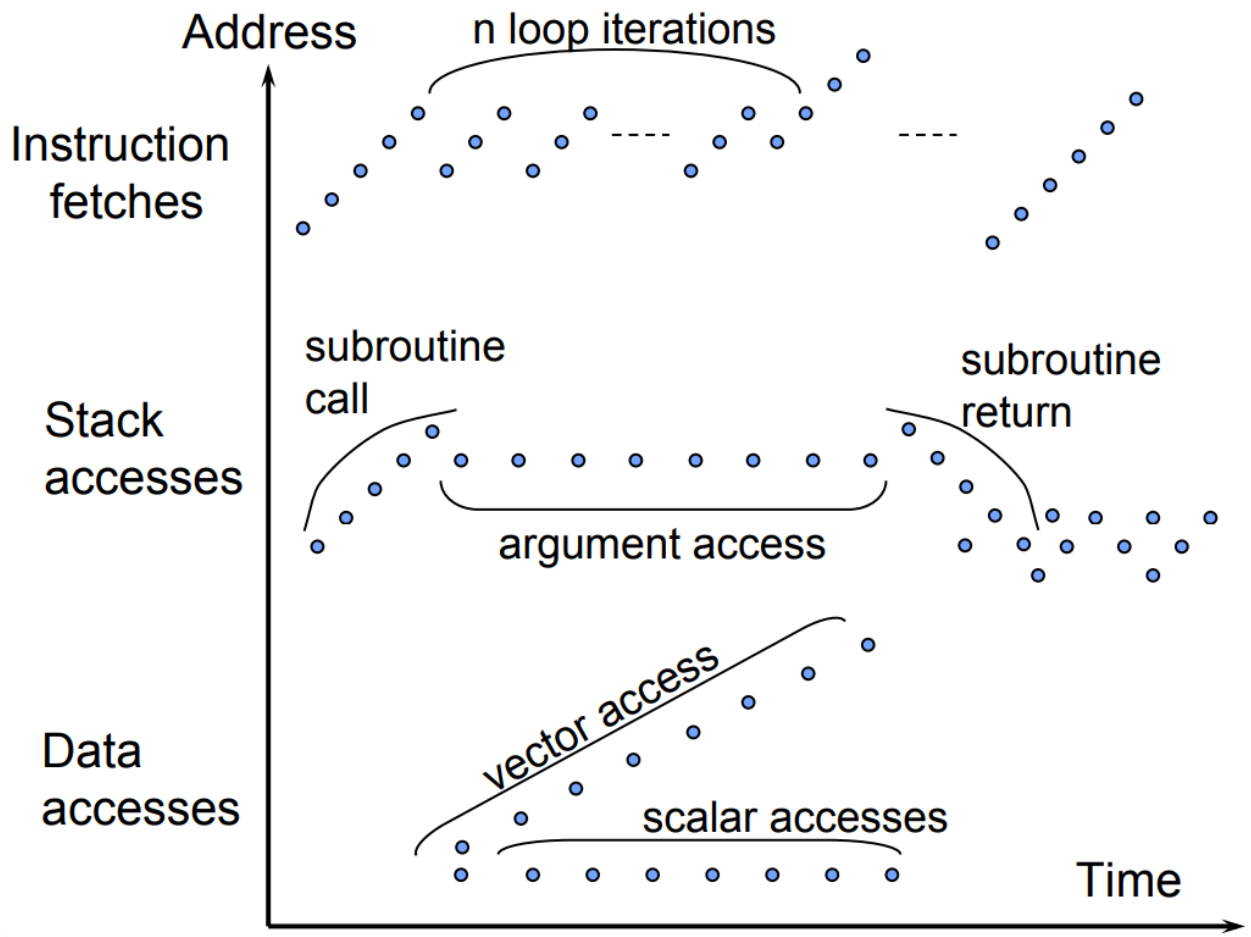
```
int sumarraycols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum
}
```

	1	2	3	4
1	A[1,1]	A[1,2]	A[1,3]	A[1,4]
2	A[2,1]	A[2,2]	A[2,3]	A[2,4]
3	A[3,1]	A[3,2]	A[3,3]	A[3,4]

Concept:
Row Major Order (RMO)

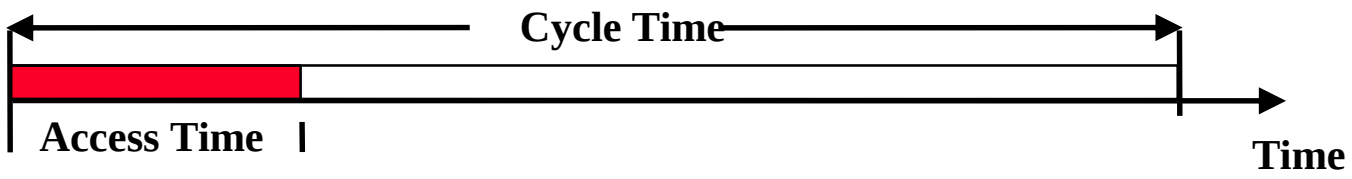
ELEMENTS	ADDRESS	
A[1,1]	200	Row 1
A[1,2]	201	
A[1,3]	202	
A[1,4]	203	
A[2,1]	204	Row 2
A[2,2]	205	
A[2,3]	206	
A[2,4]	207	
A[3,1]	208	Row 3
A[3,2]	209	
A[3,3]	210	
A[3,4]	211	

محلی بودن دسترسی به حافظه



زمان دسترسی به حافظه

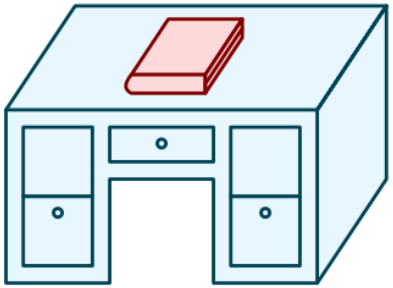
- زمان دسترسی به حافظه (Access Time) :عبارت است از زمانی که بین تقاضای داده و آماده شدن آن طول میکشد
- زمان سیکل (Cycle Time) :عبارت است از زمان بین دو تکرار متوالی



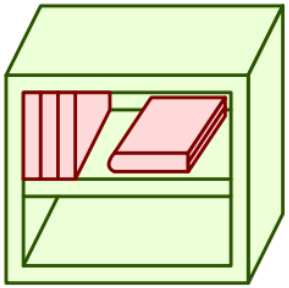
- برای مثال در حالیکه زمان دسترسی به یک حافظه DRAM ممکن است 60ns باشد با احتساب زمان های لازم برای انتقال آدرس از پردازنده به حافظه، تاخیر باس ها، و ... زمان سیکل آن ممکن است به 180-250 ns برسد.

مثال– مطالعه

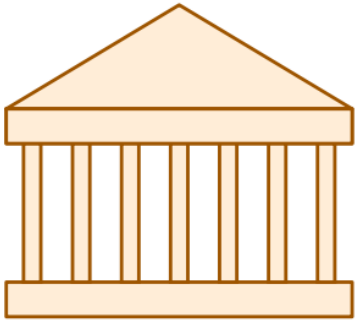
Desk
(can hold one book)



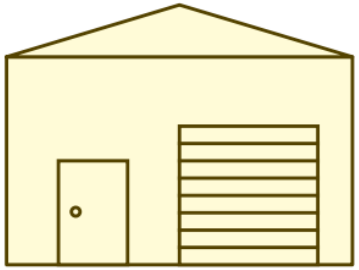
Book Shelf
(can hold a few books)



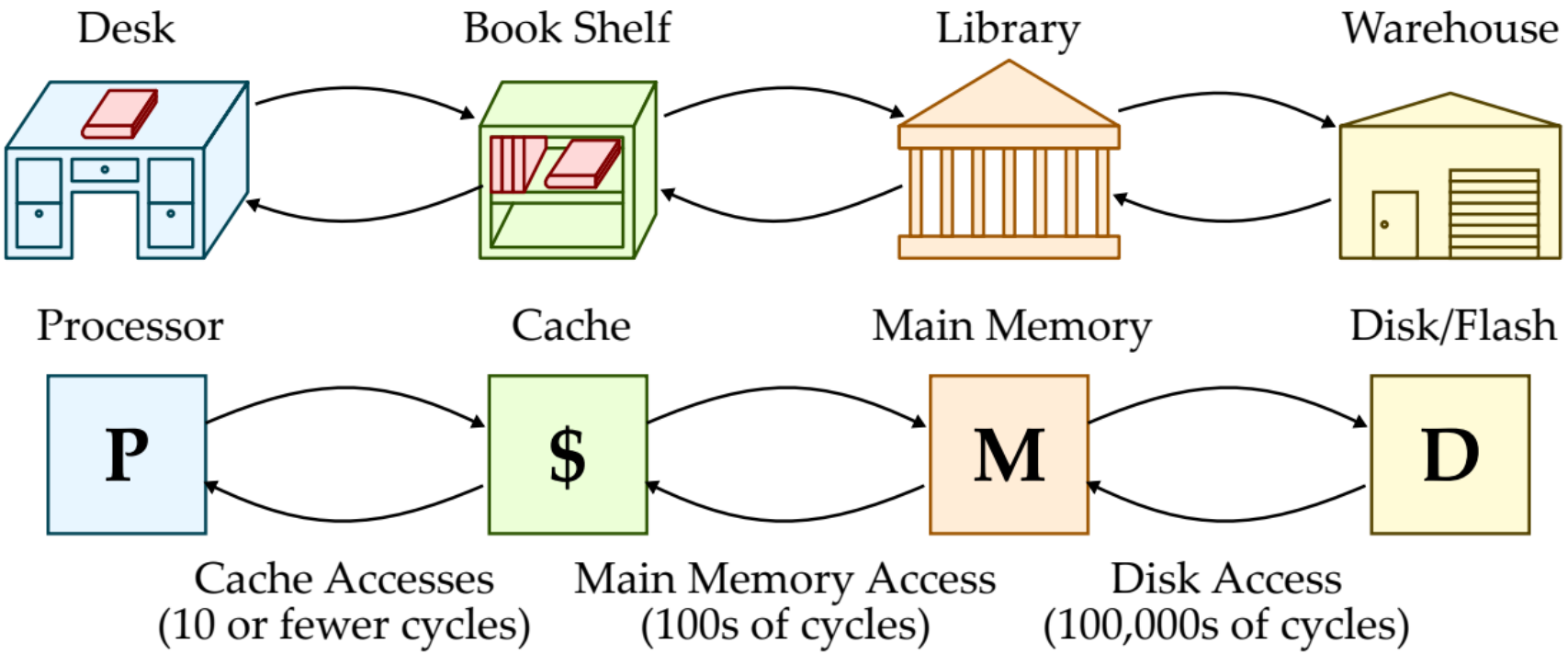
Library
(can hold many books)



Warehouse
(long-term storage)



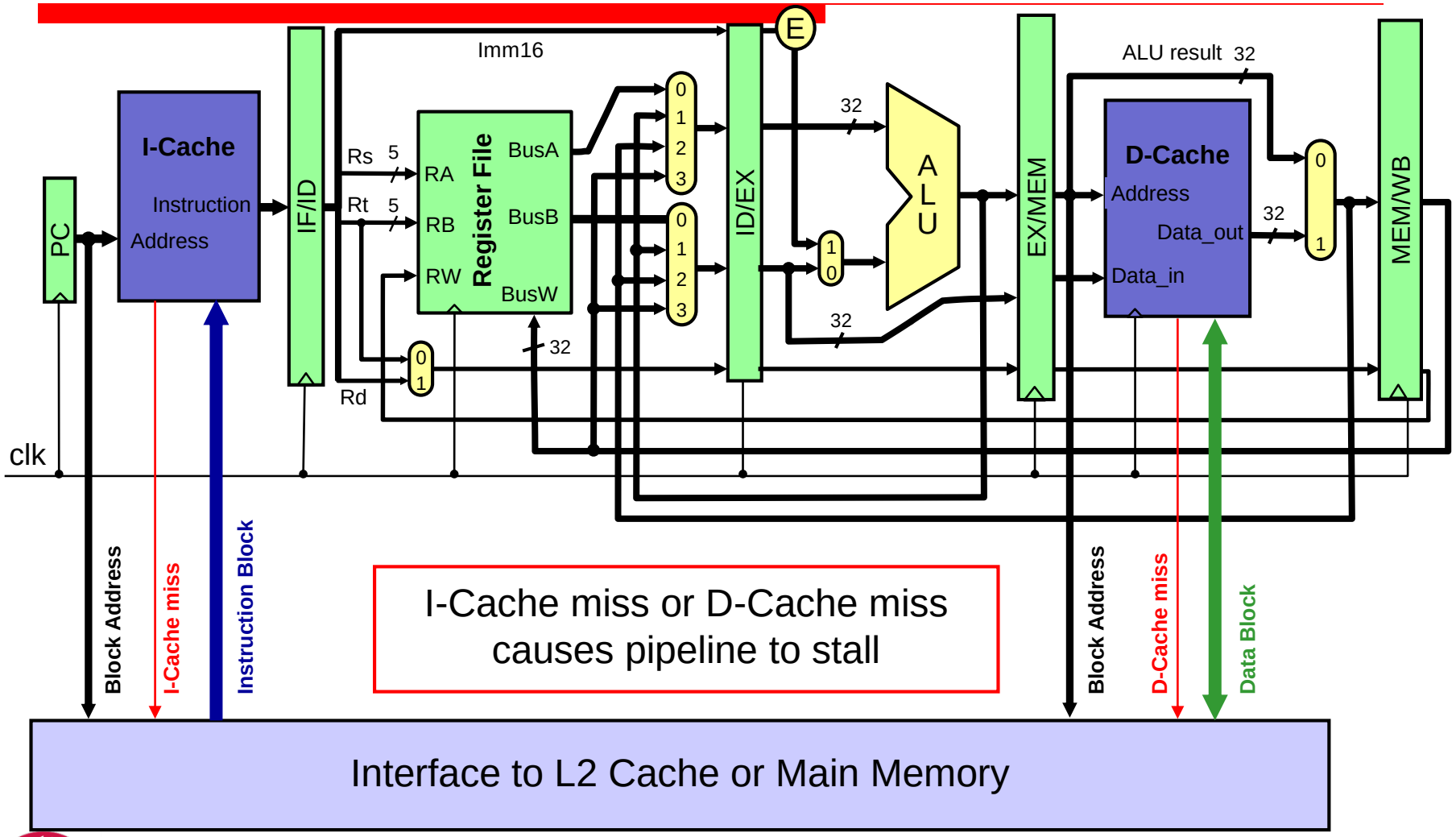
مثال – مطالعه


Desk
Book Shelf
Library
Warehouse
Processor
Cache
Main Memory
Disk/Flash
P
$
M
D
Cache Accesses
(10 or fewer cycles)
Main Memory Access
(100s of cycles)
Disk Access
(100,000s of cycles)

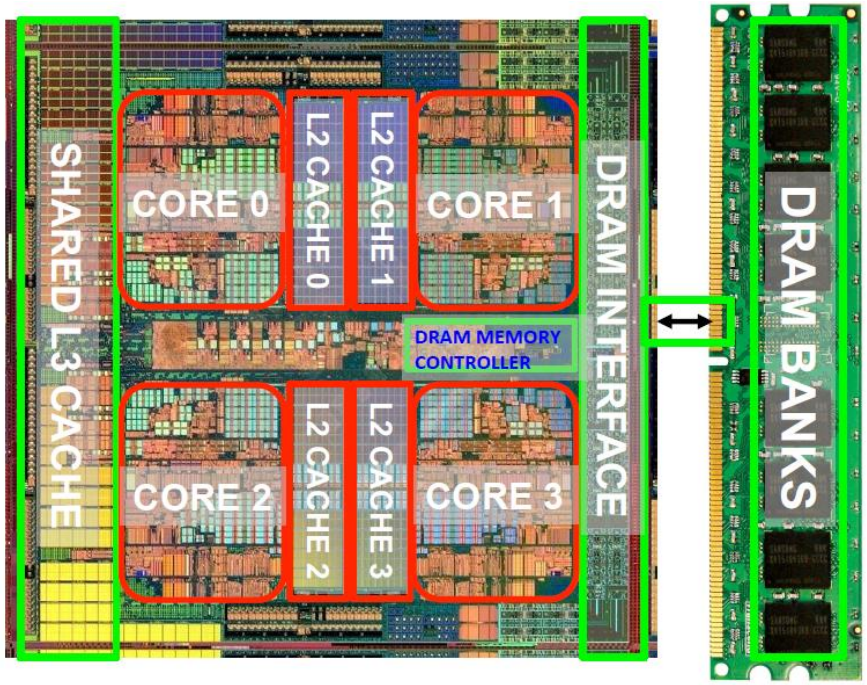
حافظه cache

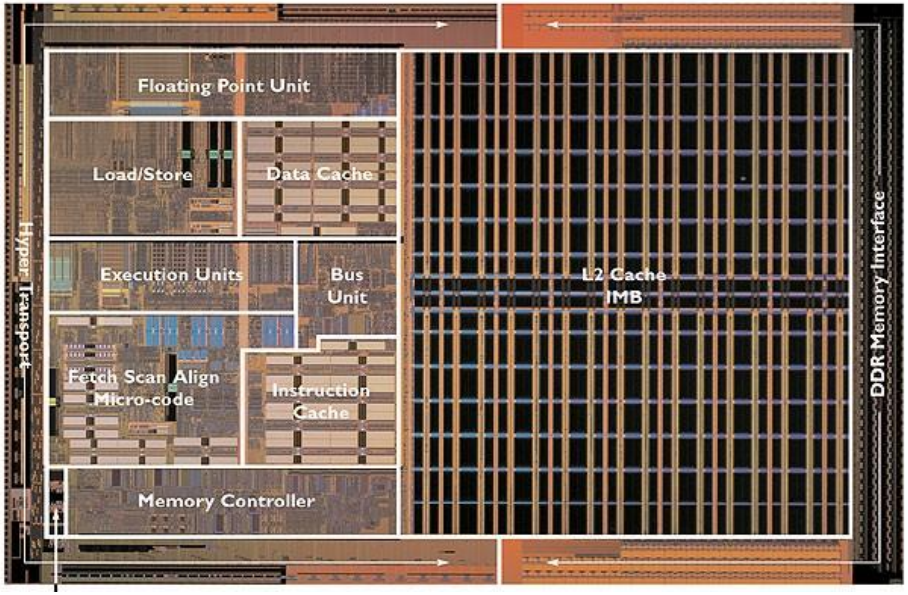
- اگر قسمت فعال برنامه و داده ها را در حافظه سریع و کوچکی قرار دهیم، میتوانیم با کم کردن میانگین زمان دسترسی به حافظه زمان اجرای برنامه را کاهش دهیم.
- این حافظه سریع و کوچک را حافظه cache می نامند.
- معمولا حافظه cache زمان دسترسی بسیارکمتری نسبت به حافظه اصلی دارد(5 تا 10 برابر کمتر)
- دریک کامپیوتر ممکن است عمل cache تا چند سطح تکرار گردد.

Cache Memories in the Datapath

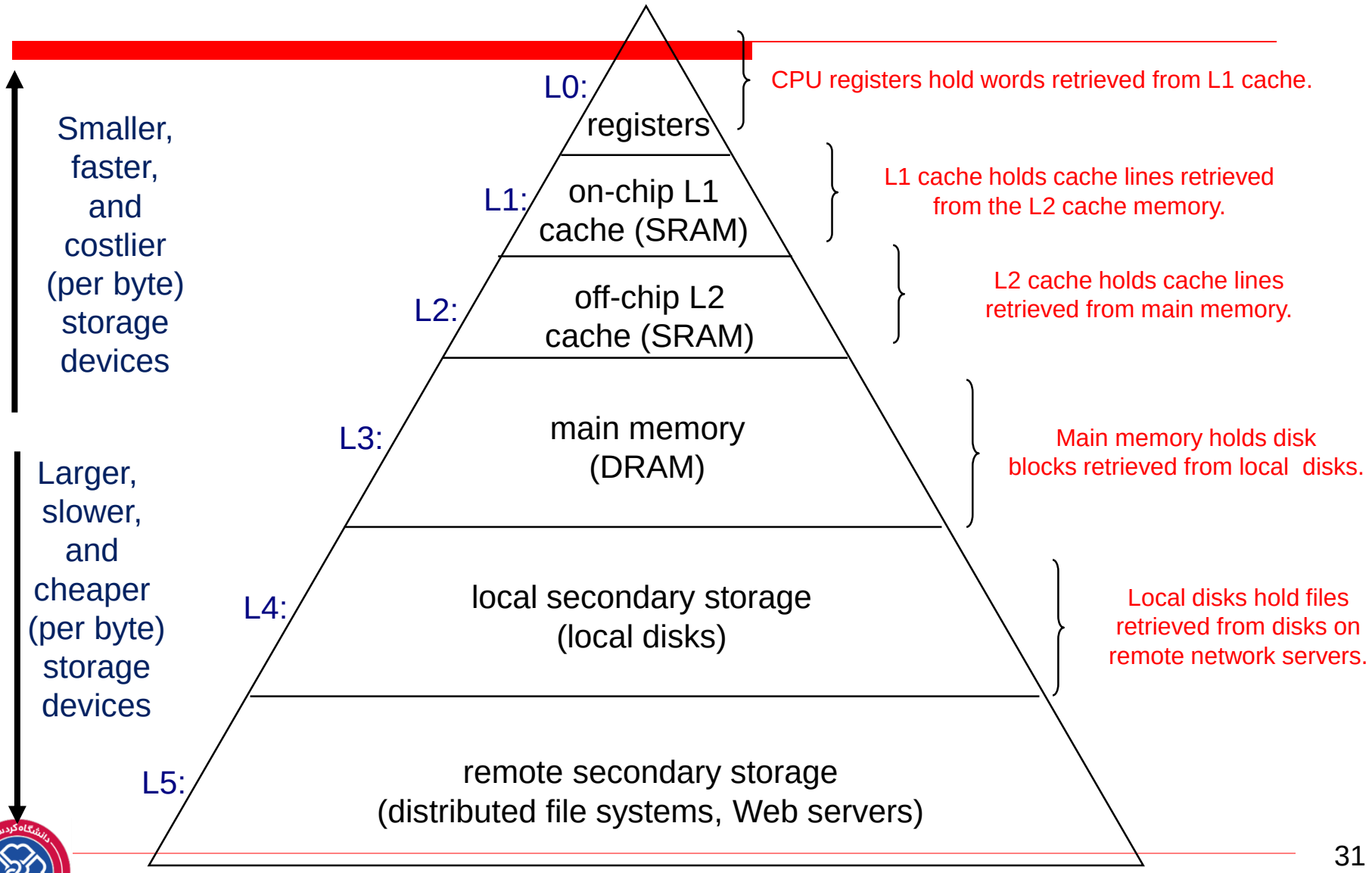


Cache introduction



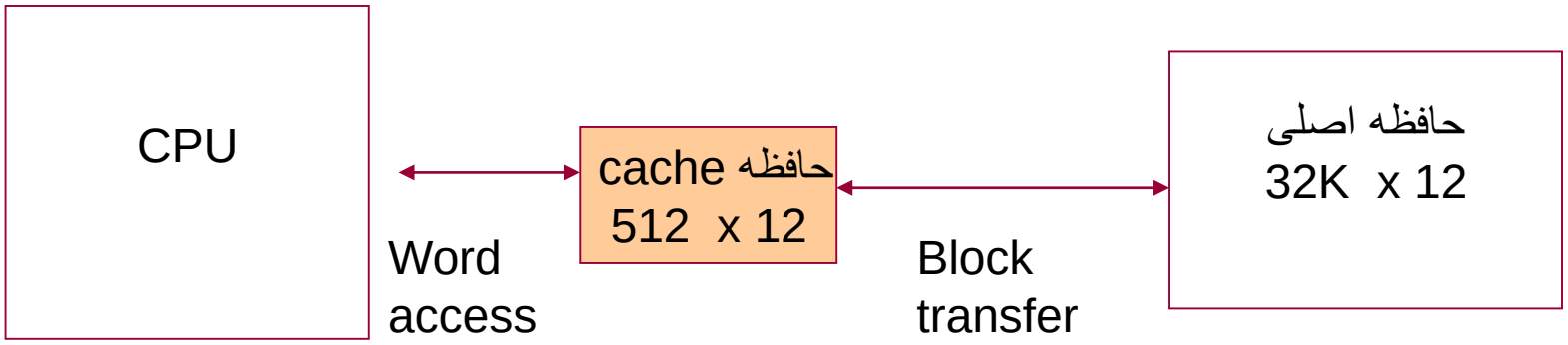


سلسله مراتب حافظه



نحوه عمل cache

- وقتی که cpu نیاز به دسترسی به یک داده را دارد ابتدا حافظه cache را جستجو مینماید. اگر داده در این حافظه سریع موجود بود از آن استفاده میشود در غیر اینصورت با رجوع به حافظه بلوکی از داده که شامل داده مورد نیاز cpu میشود از حافظه اصلی به حافظه cache منتقل میگردد.



Four Basic Questions on Caches

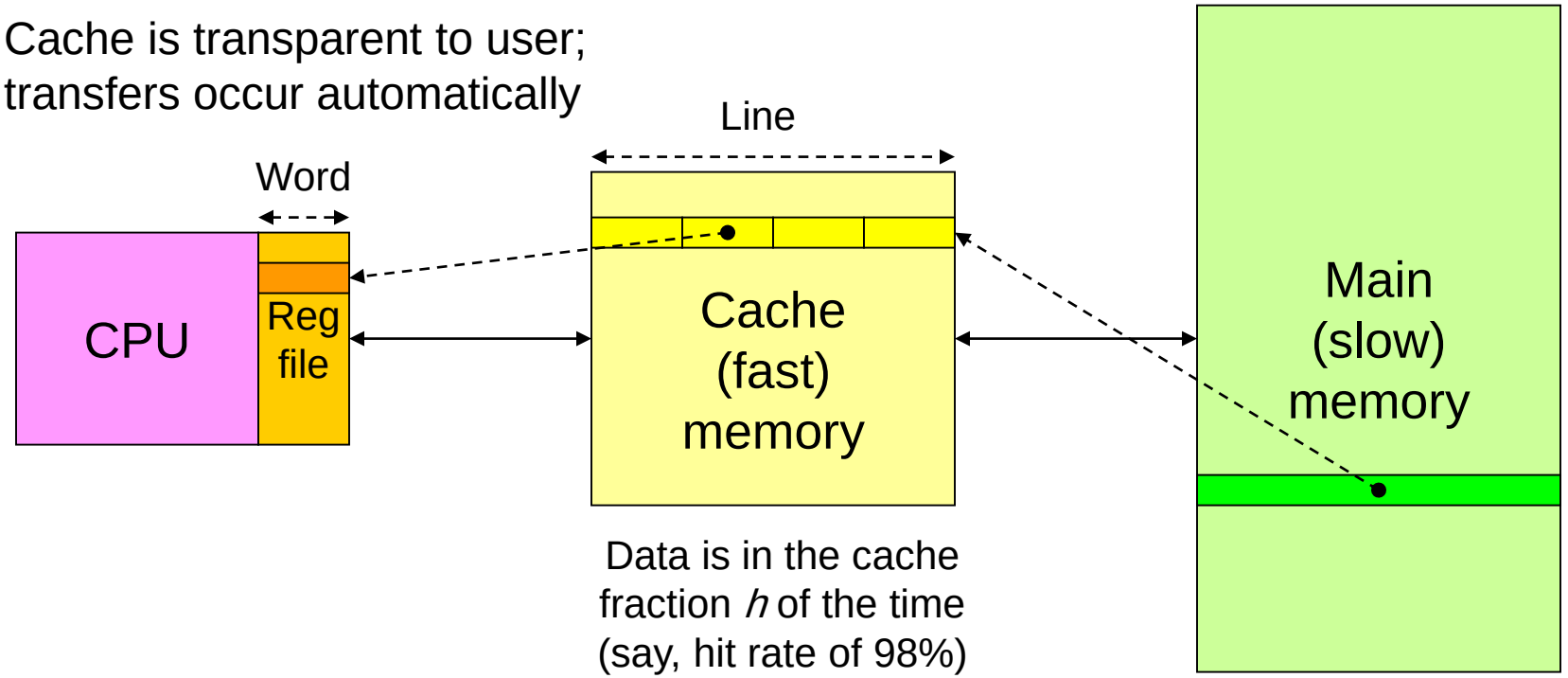
- **Q1: Where can a block be placed in a cache?**
  - Block placement (Mapping)
  - Direct Mapped, Set Associative, Fully Associative
- **Q2: How is a block found in a cache?**
  - Block identification
  - Block address, tag, index
- **Q3: Which block should be replaced on a miss?**
  - Block replacement
  - FIFO, Random, LRU
- **Q4: What happens on a write?**
  - Write strategy
  - Write Back or Write Through (with Write Buffer)

درصد موفقیت (hit ratio)

- کارایی حافظه cache با ضریبی با نام درصد موفقیت hit ratio اندازه گیری میشود.
- این درصد عبارت است از نسبت تعداد دفعاتی که داده در cache یافت شده به تعداد کل دفعات رجوع به حافظه کش.
- هر چه مقدار این درصد بیشتر باشد سرعت دسترسی به حافظه به سرعت cache نزدیکتر میشود.
- مثال:
- یک کامپیوتر با زمان دسترسی 100ns برای حافظه cache و1000nsبرای حافظه اصلی در صورت داشتن درصد موفقیت0.9 زمان دسترسی برابر با 200ns خواهد داشت.

Cache Hit/Miss Rate, and Effective Access Time

Cache is transparent to user;
transfers occur automatically



One level of cache with hit rate h

$$C_{\text{eff}} = hC_{\text{fast}} + (1 - h)(C_{\text{slow}} + C_{\text{fast}}) = C_{\text{fast}} + (1 - h)C_{\text{slow}}$$

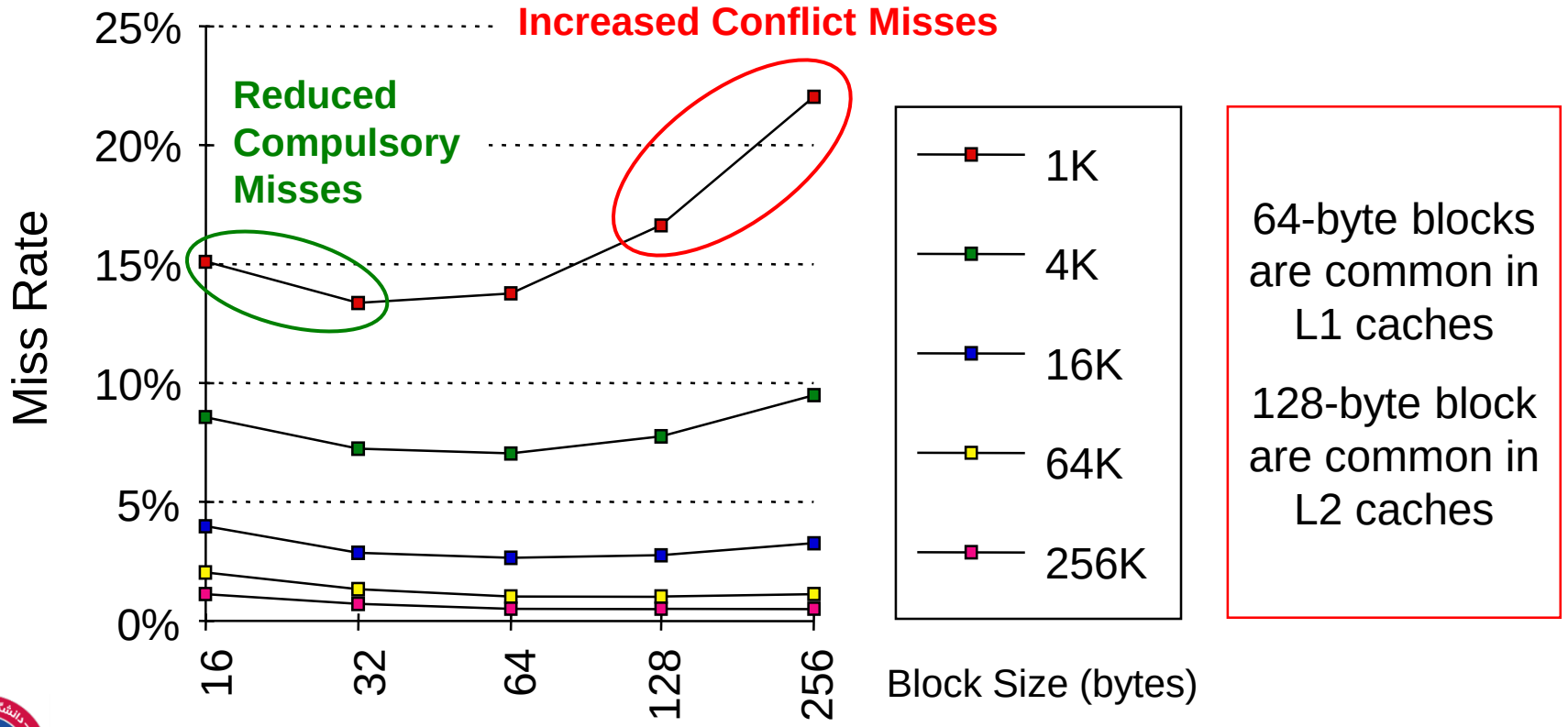
مثال

- در یک کامپیوتر دارای cache دو سطحی (L2, L1) تاخیر دستیابی به L1 برابر 1 ns و برای L2 برابر با 10 ns است. زمان دستیابی به حافظه اصلی برای یک بلوک 100 nsمی باشد. اگر در صد موفقیت برای L1 و L2 به ترتیب ۹۰٪و ۵۰٪ باشد، متوسط زمان رجوع به حافظه چیست؟

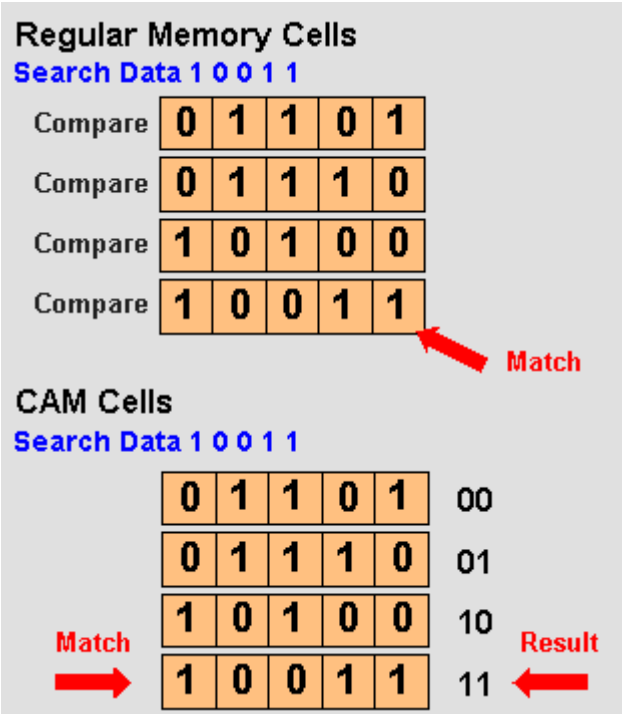
$$t_{av} = t_{L1} + (1-h_{L1}).t_{L2} + (1-h_{L1})(1-h_{L2}).t_m = 1+ 1+ 5 = 7 \text{ ns}$$

Block Size Considerations

- Simplest way to reduce miss rate is to increase block size (Spatial Locality)
- However, it increases conflict misses if cache is small



حافظه انجمنی (آدرس دهی با محتوا)

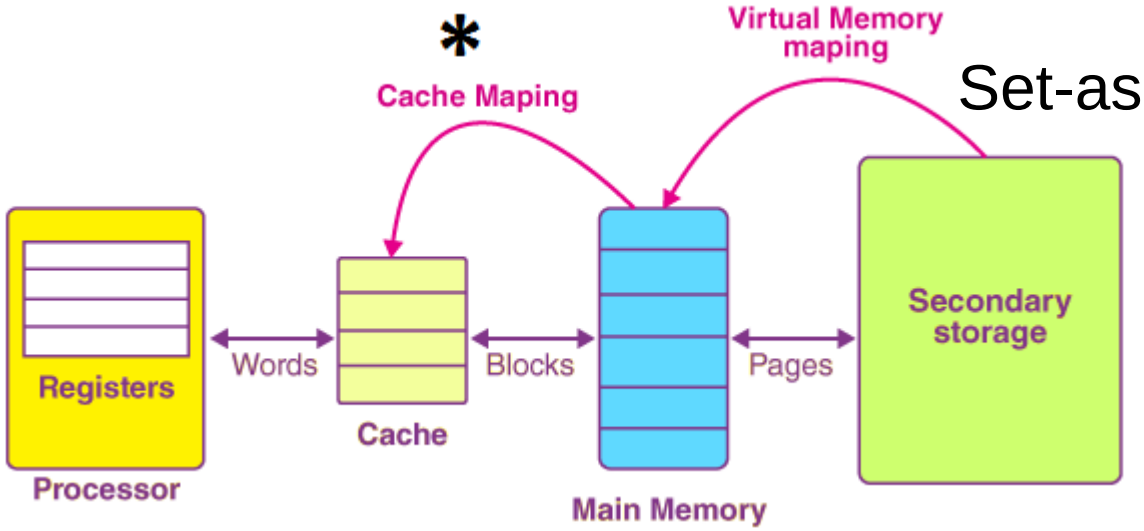


- از این حافظه برای تسریع عمل جستجو در بین داده های ذخیره شده استفاده میشود.
- برای کاهش زمان جستجو بجای استفاده از آدرس، از محتوی خود داده استفاده میشود. این روش را گاهی Content Addressable Memory (CAM) هم مینامند.
- هنگام نوشتن داده در این حافظه نیازی به آدرس نیست!
- این حافظه قادر است تا جای خالی را پیدا نموده و داده را در آنجا ذخیره مینماید.
- هنگام خوان داده از حافظه، مقدار داده و یا بخشی از آن به حافظه ارائه شده و حافظه تمامی کلمات ذخیره شده ای را که با داده مورد نظر یکسان هستند را علامت گذاری کرده و آماده خواندن مینماید.

نگاشت (Mapping)

- عمل انتقال داده از حافظه اصلی به حافظه cache را نگاشت مینامند. این کار به سه طریق انجام میشود:

۱. Associative Mapping
۲. Direct Mapping
۳. Set-associative mapping

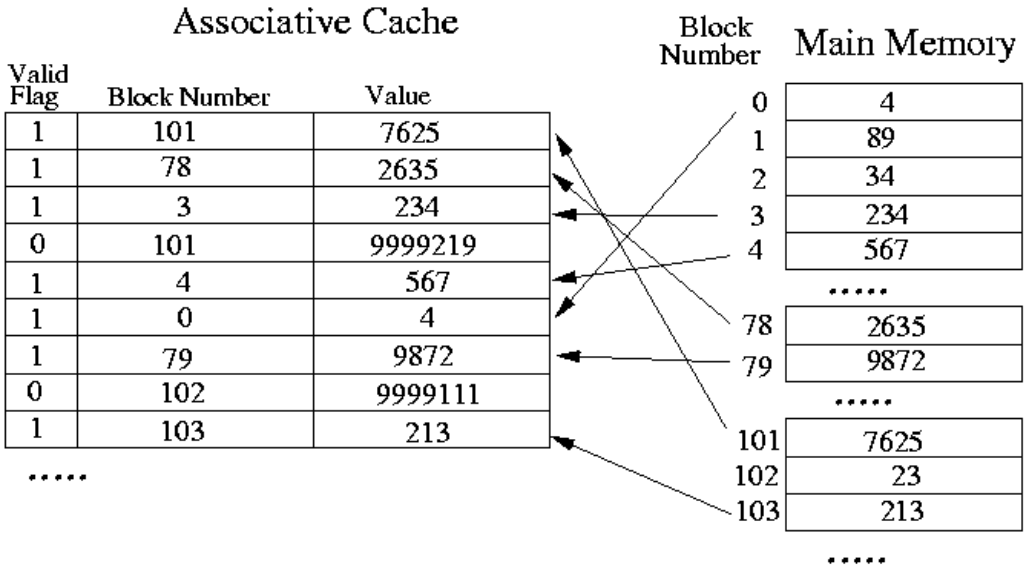


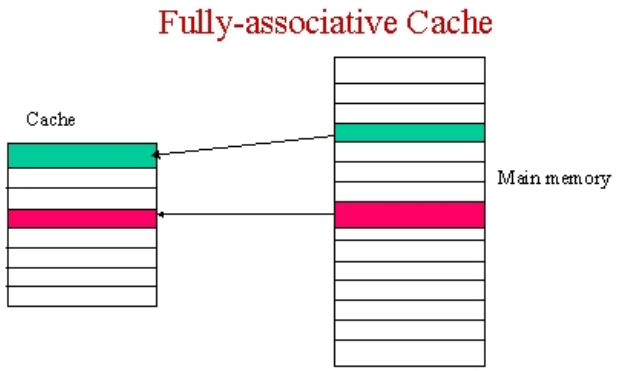
Fully Associative mapping

- در **نگاشت انجمنی کامل** که سریعترین راه پیاده سازی یک cache است از یک حافظه associative استفاده میشود.
- در این حافظه هم آدرس و هم محتوی یک کلمه ذخیره میشوند. در نتیجه cache میتواند محتوی هر محل از حافظه را ذخیره نماید.
- هنگام جستجو برای یک داده آدرس آن به حافظه associative عرضه میشود. در صورتیکه ورودی متناظری در حافظه باشد، داده مربوطه در خروجی ظاهر میگردد. در غیر اینصورت هم آدرس و هم داده در حافظه associative ذخیره خواهند شد.

Fully Associative mapping

Any memory block can map to any cache block

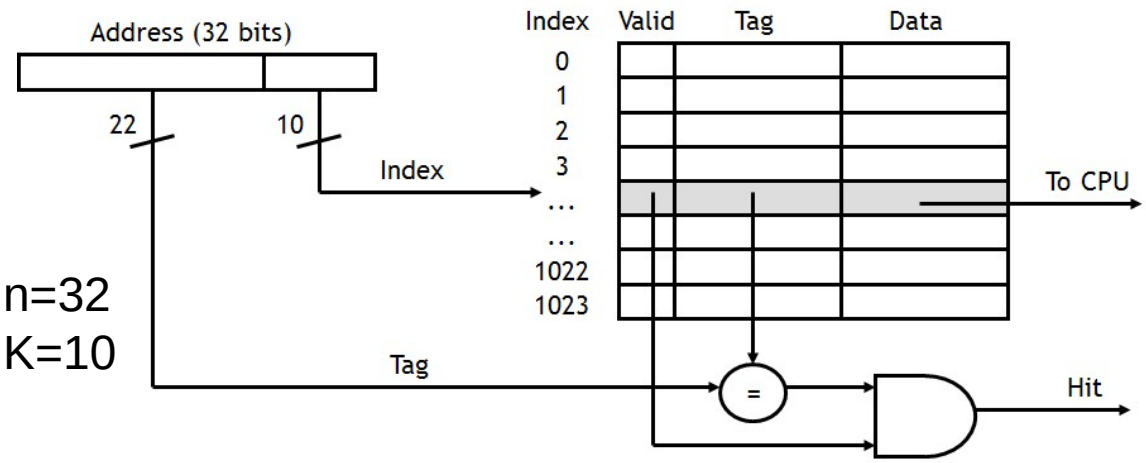




نگاشت مستقیم Direct Mapping

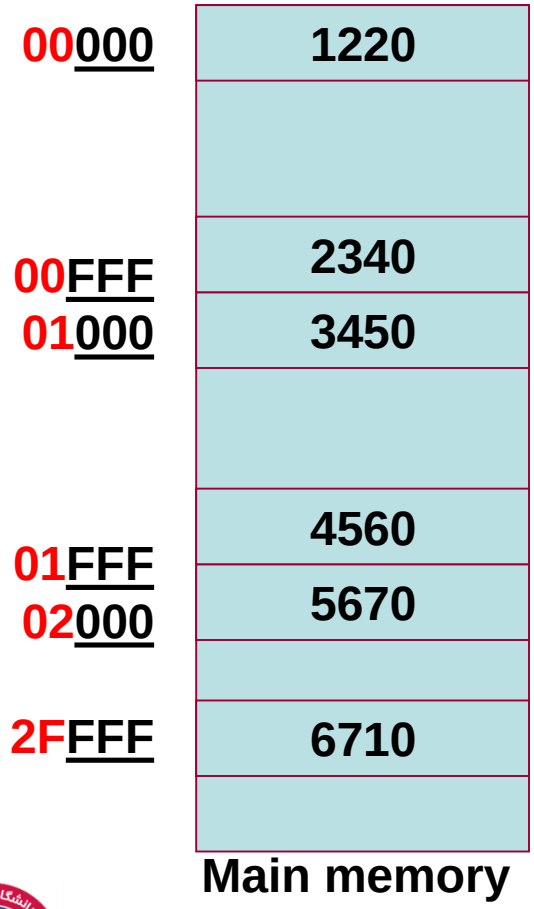
- اگر حافظه اصلی دارای 2^n کلمه و حافظه cache دارای 2^k کلمه باشند در نتیجه به ترتیب به n و k بیت آدرس نیاز خواهند داشت.
- آدرس n بیتی حافظه اصلی بصورت زیر تقسیم میشود:

Tag	Index
n-k	k



Direct Mapping نگاشت مستقیم

- در حافظه cache علاوه بر داده اطلاعات مربوط به Tag هم ذخیره میشود:



Index	Tag	Data
000	00	1220
FFF	2F	6710

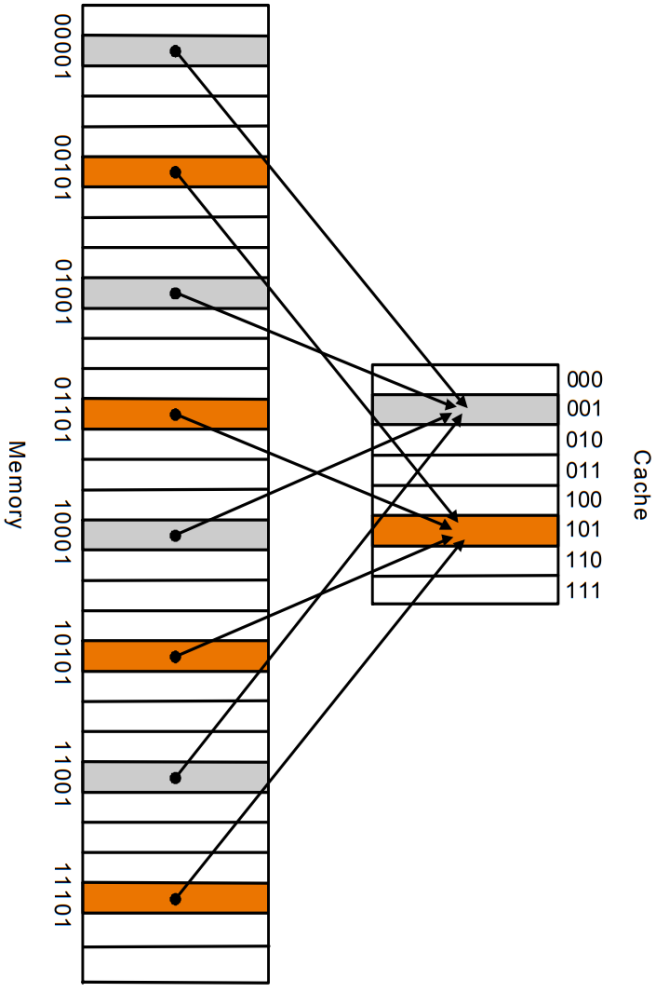
Cache

نگاشت مستقیم Direct Mapping

- وقتی که آدرسی توسط CPU تولید میشود با استفاده از قسمت Index آن یک کلمه از حافظه cache خوانده شده و مقدار Tag ذخیره شده در آن با مقدار Tag آدرس مقایسه میگردد.
- اگر دو Tag یکسان بودند دادهcache مورد استفاده قرار میگیرد (hit)
- در غیر اینصورت (miss) داده از حافظه خوانده شده و همراه با Tag جدید در cache نوشته میشود.
- در این روش اگر رجوع به آدرسهائی با index یکسان زیاد اتفاق بیافتد، درصد موفقیت پائین میاید.

Direct mapped

22	10110_{two}	miss	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	miss	$(11010_{two} \bmod 8) = 010_{two}$
22	10110_{two}	hit	$(10110_{two} \bmod 8) = 110_{two}$
26	11010_{two}	hit	$(11010_{two} \bmod 8) = 010_{two}$
16	10000_{two}	miss	$(10000_{two} \bmod 8) = 000_{two}$
3	00011_{two}	miss	$(00011_{two} \bmod 8) = 011_{two}$
16	10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$
18	10010_{two}	miss	$(10010_{two} \bmod 8) = 010_{two}$

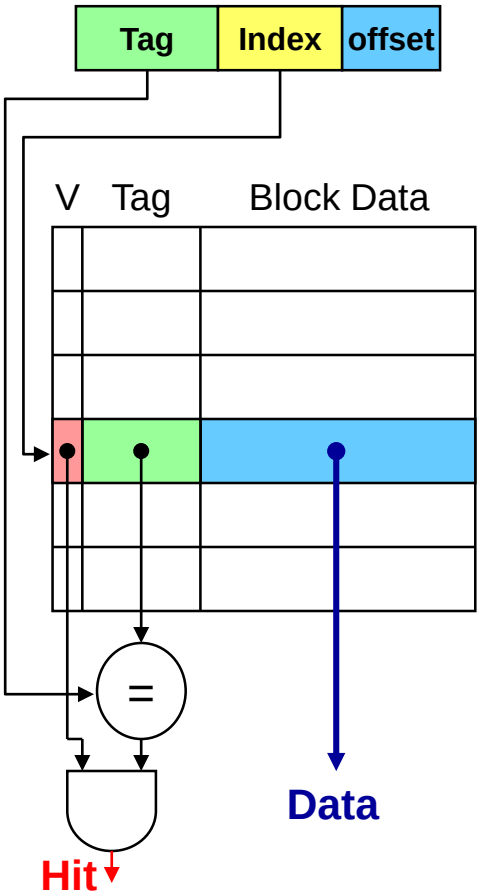


نگاشت مستقیم با اندازه بلوک بزرگتر

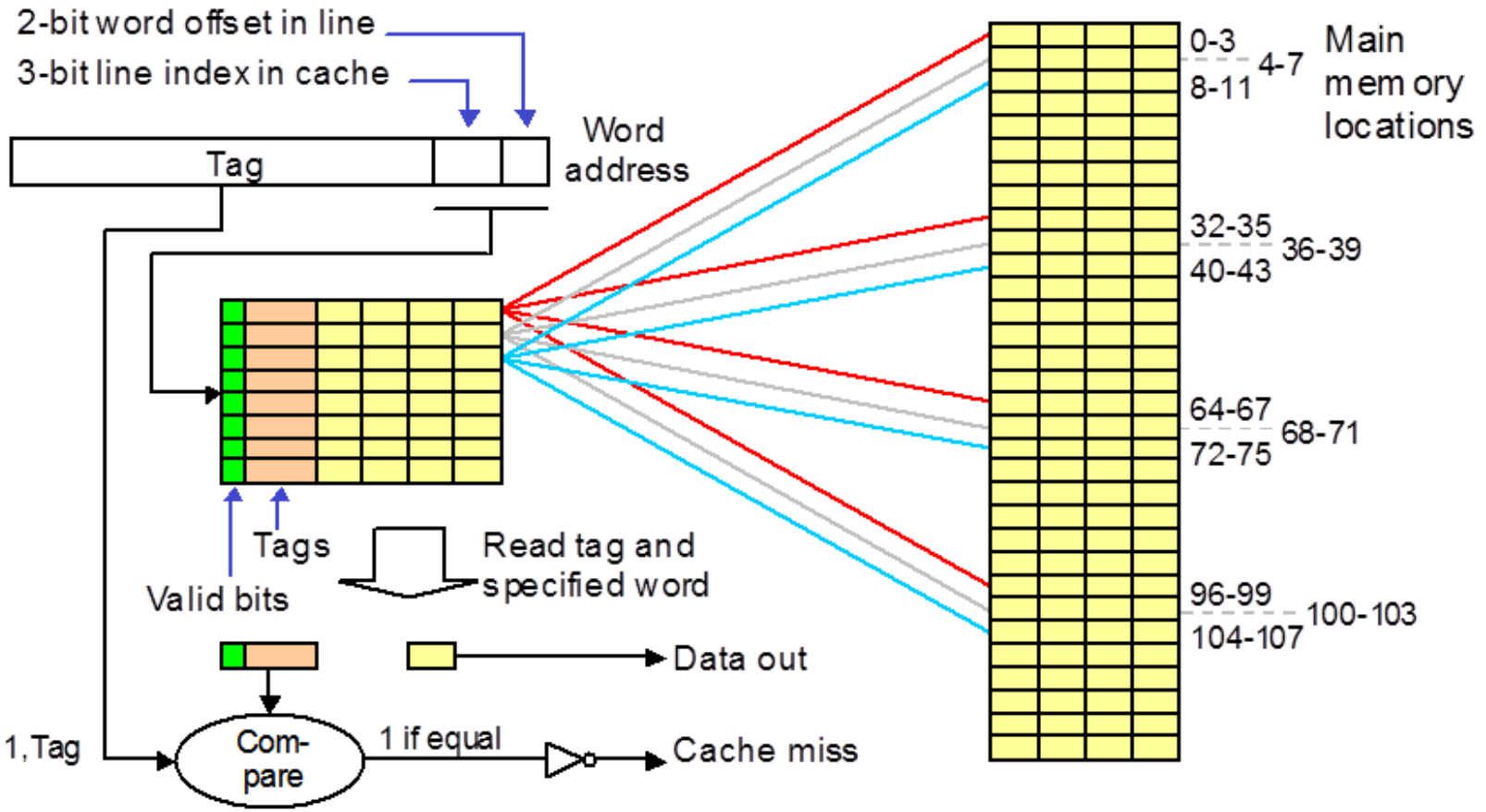
	Index	Tag	Data
Block 0	000	01	3450
	007	01	6578
Block 1	010	00	1340
	017	00	1658

Cache

Tag	Index (Block)	Word offset
n-k	k	



Direct-Mapped Cache (Block size=4)



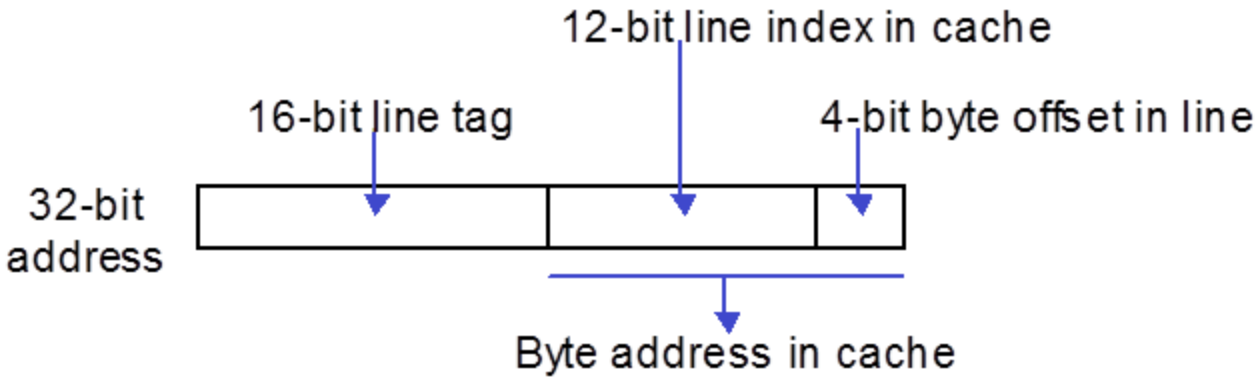
Direct-mapped cache holding 32 words within eight 4-word lines. Each line is associated with a tag and a valid bit.

Accessing a Direct-Mapped Cache - example

Show cache addressing for a byte-addressable memory with 32-bit addresses. Cache line W = 16 B. Cache size L = 4096 lines (64 KB).

Solution

Byte offset in line is $\log_2 16 = 4$ b. Cache line index is $\log_2 4096 = 12$ b. This leaves 32 – 12 – 4 = 16 b for the tag.

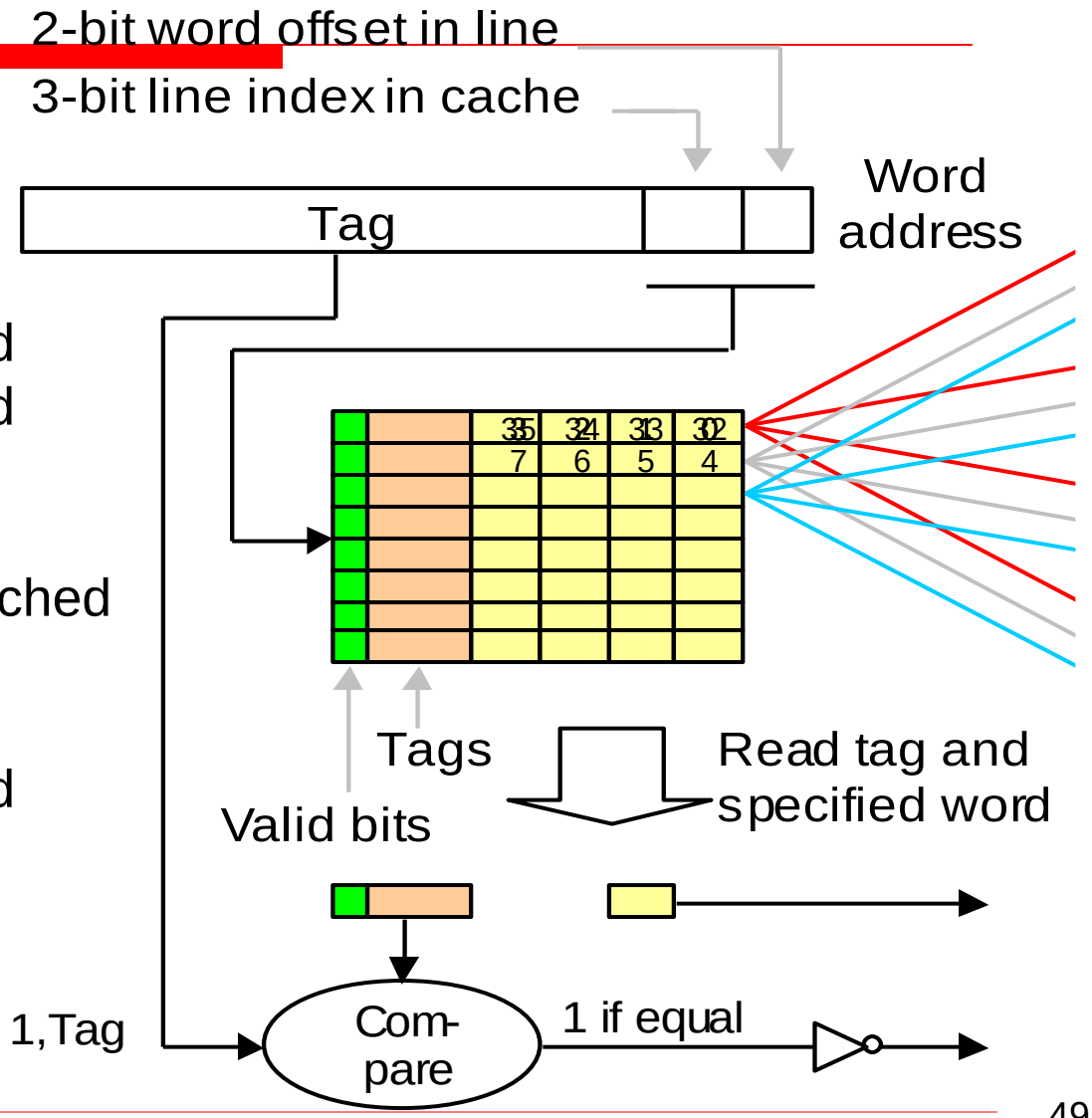


Components of the 32-bit address in an example direct-mapped cache with byte addressing.

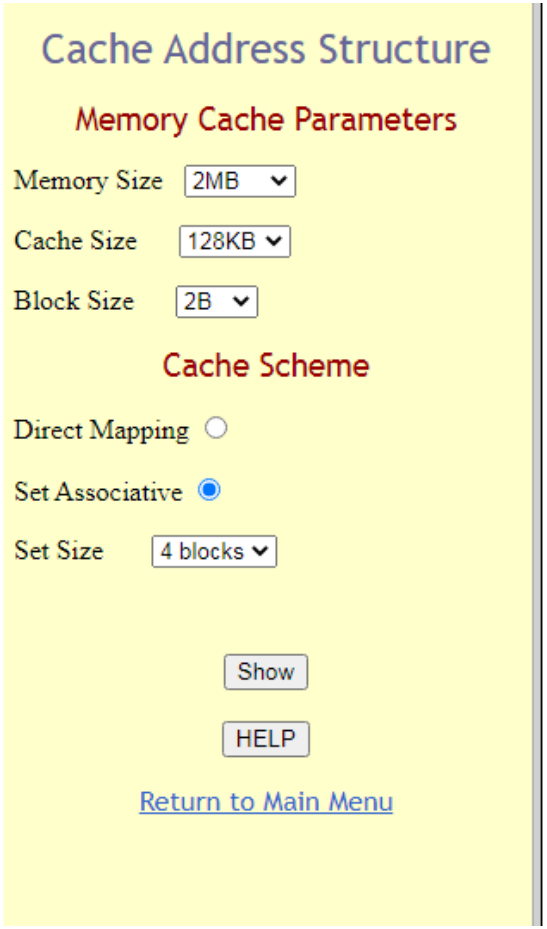
Direct-Mapped Cache Behavior

Address trace:
1, 7, 6, 5, 32, 33, 1, 2, . . .

1: miss, line 3, 2, 1, 0 fetched
7: miss, line 7, 6, 5, 4 fetched
6: hit
5: hit
32: miss, line 35, 34, 33, 32 fetched
(replaces 3, 2, 1, 0)
33: hit
1: miss, line 3, 2, 1, 0 fetched
(replaces 35, 34, 33, 32)
2: hit
... and so on



شبیه ساز Cache



Address Bit Partitioning

TAG						INDEX												OFFSET
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Compare Bits						Set Select Bits														Byte Select Bits

The Compare Bits are compared with the corresponding Tag Bits in the Cache Directory.
The Set Select Bits are used to select a particular Set in the Cache.
The Byte Select Bits are used to select a particular byte in the accessed block.

Memory size = 2MB = 2^{21}

Block size = 2Bytes = 2^{1}

Number of sets in cache = Cache size/(Set size * Block size) = 128KB/(4 blocks * 2B) = $2^{17}/(2^2 * 2^1) = 2^{14}$

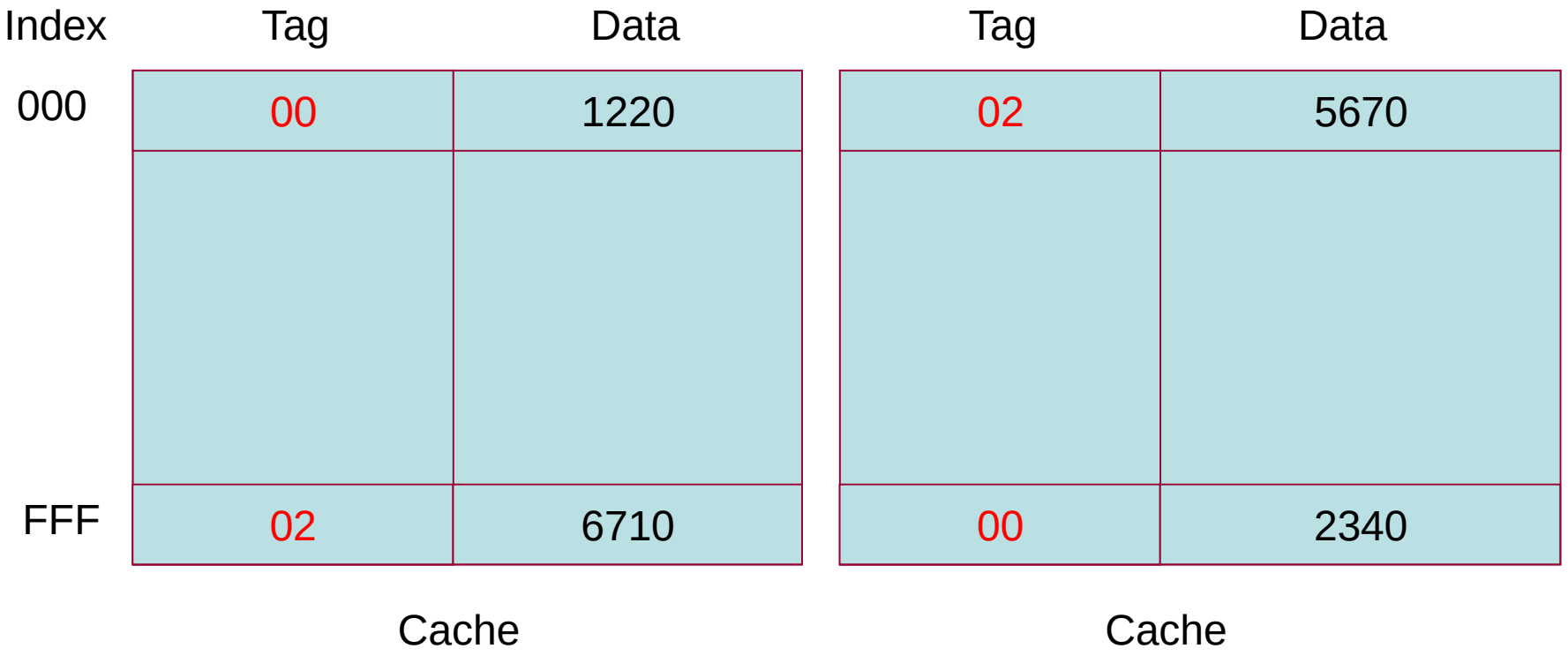
Number of bits in Tag = Total bits - Index bits - Offset bits = 21-14-1 = 6

http://www.ecs.umass.edu/ece/koren/architecture/Cache/default.htm

Set-associative mapping

- در این روش هر محل از حافظه cache میتواند چندین کلمه با آدرس Index یکسان را ذخیره نماید.
- تعداد data-tag های ذخیره شده در حافظه یک set خوانده میشود.
- برای مقایسه tag آدرس تولید شده با مقادیر ذخیره شده ازحافظه associative استفاده میشود.
- با بزرگ شدن set ها درصد موفقیت cache افزایش می یابد.

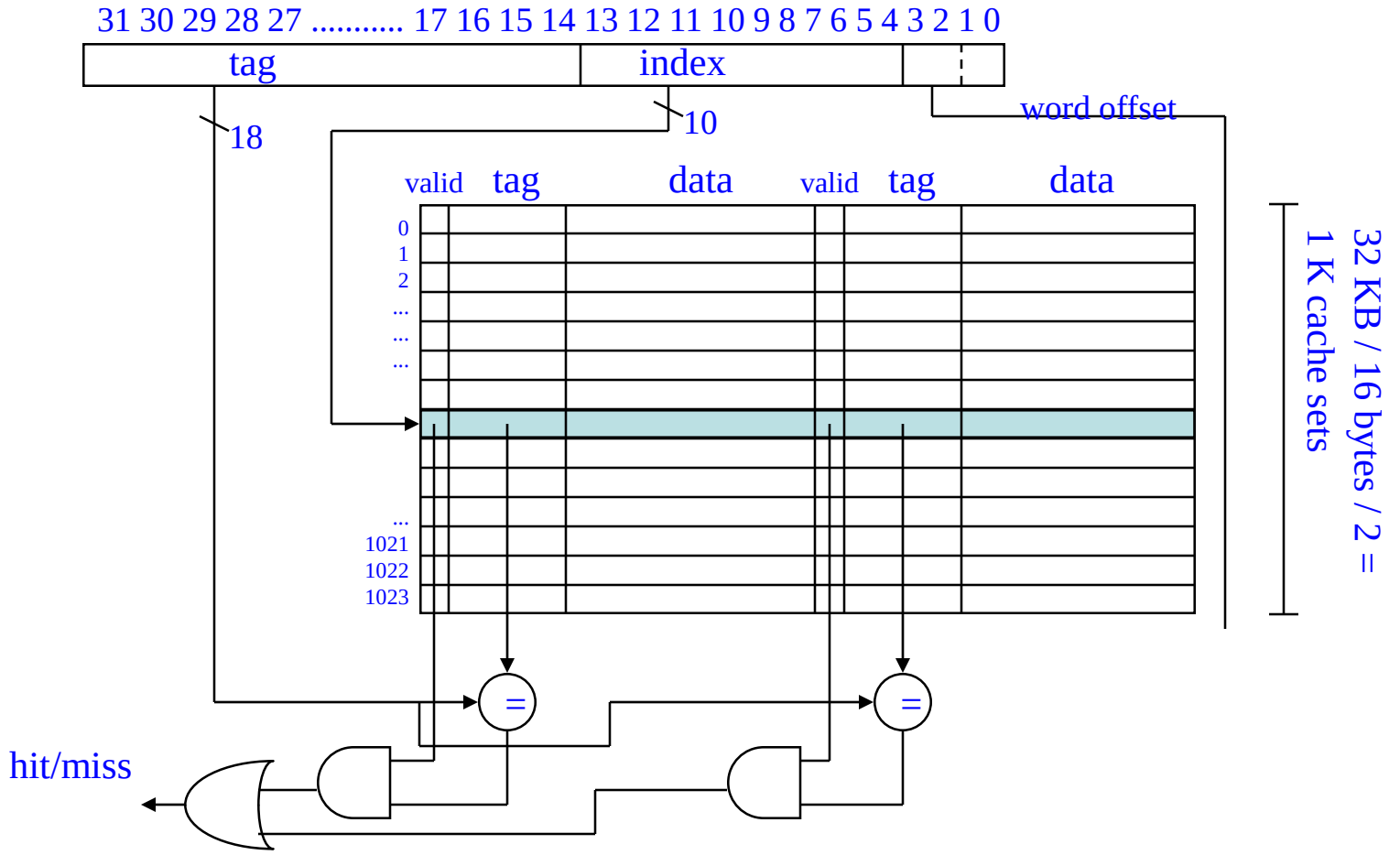
Set-associative mapping



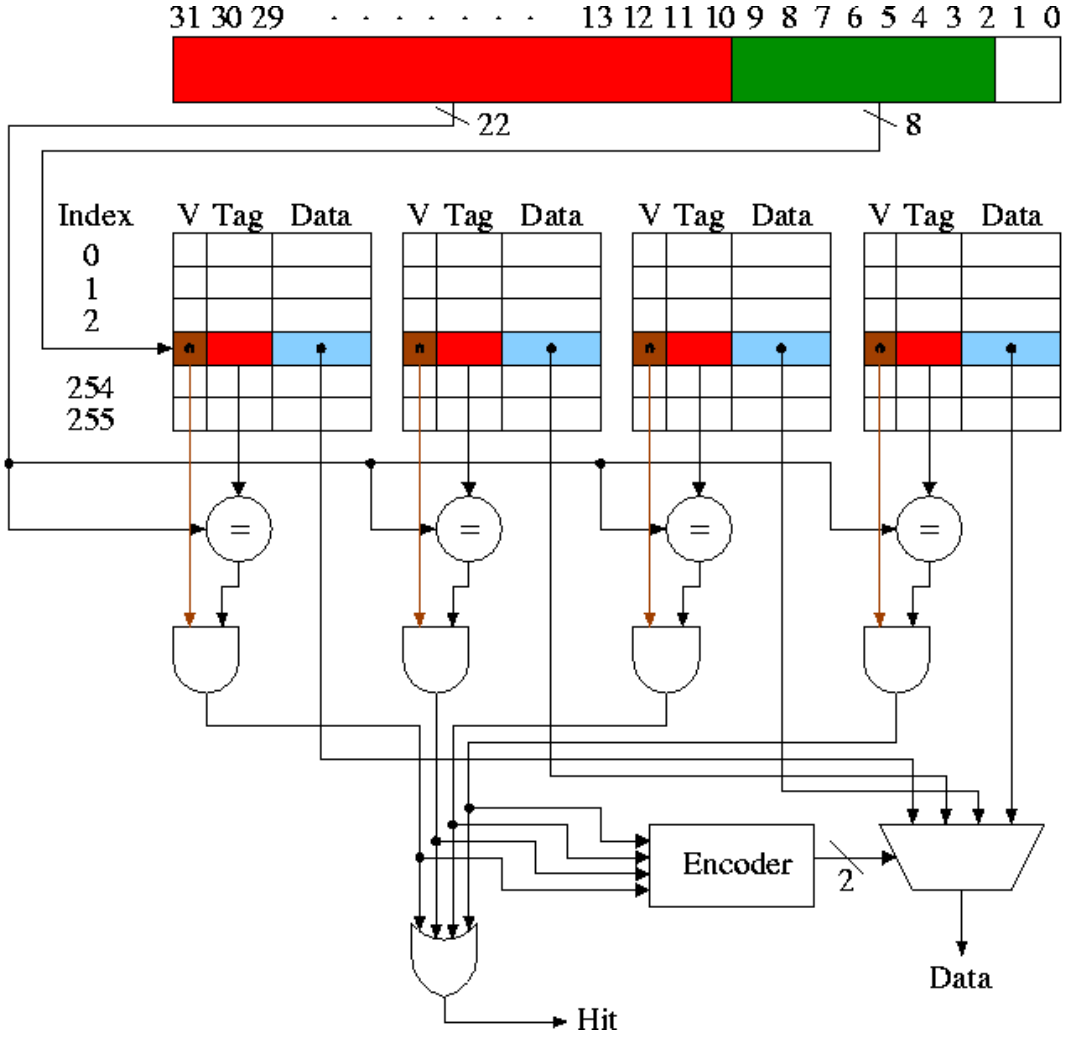
2-way Set associative Cache

Accessing a Sample Cache

- 32 KB cache, 2-way set-associative, 16-byte block size



4-way Set-associative Cache



Cache Address Mapping - example

A 64 KB four-way set-associative cache is byte-addressable and contains 32 B lines. Memory addresses are 32 b wide.

a. How wide are the tags in this cache?

b. Which main memory addresses are mapped to set number 5?

Solution

a. Address (32 b) = 5 b byte offset + 9 b set index + 18 b tag

b. Addresses that have their 9-bit set index equal to 5. These are of the general form $2^{14}a + 2^5 \times 5 + b$; e.g., 160-191, 16 554-16 575, . . .

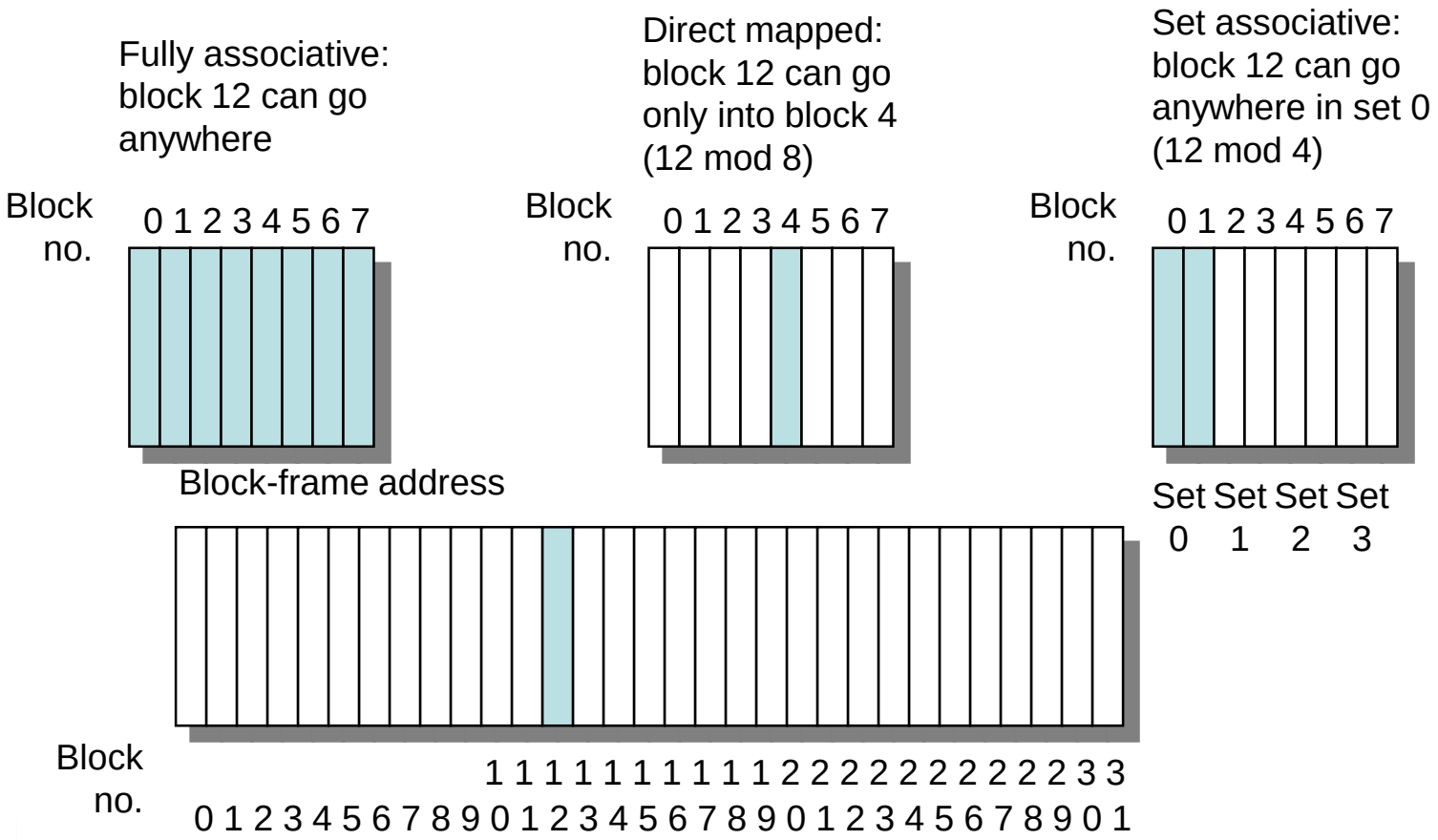
Tag width = 32 – 5 – 9 = 18

Set size = 4 × 32 B = 128 B
Number of sets = $2^{16}/2^7 = 2^9$

Line width = 32 B = 2^5 B

Mapping- Comparison

- Block 12 placed in 8 block cache:
  - Fully associative, direct mapped, 2-way set associative



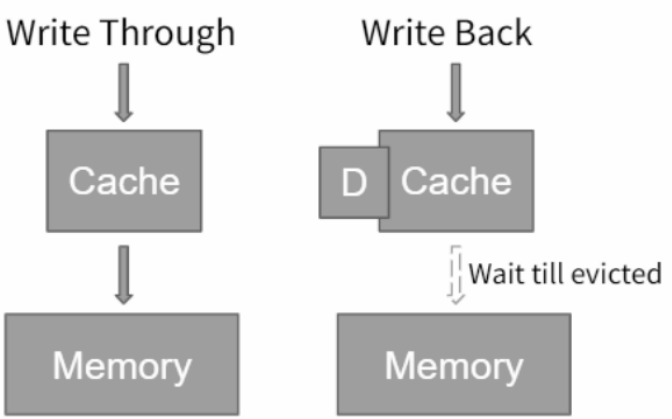
الگوریتم های جایگزینی

- وقتی که یک miss اتفاق می افتد، در صورتیکه یک set پر باشد لازم میشود تا یکی از داده های آن خالی شده و داده جدید وارد cache شود.
- روشهای مختلفی برای اینکار وجود دارد:

1. First-In First-Out (FIFO)
2. Random replacement
3. Least recently used (LRU)

نوشتن در حافظه cache

یکی از مسایل مهم در ارتباط با حافظه cache نحوه عمل در هنگام نوشتن اطلاعات در حافظه است.



- هنگام خواندن داده ها وقتی داده در cache وجود داشته باشد، نیازی به رجوع به حافظه اصلی نیست اما وقتی داده را در حافظه مینویسیم ممکن است به دو طریق عمل شود:
  ۱. write through :در این روش درهر بار نوشتن در حافظه داده هم در cache و هم در حافظه اصلی نوشته میشود.
  ۲. write back :در این روش داده فقط در cache نوشته شده و با یک پرچم set میشود .تا زمانیکه این داده در cache قرار دارد از این داده استفاده خواد شد، اما در صورت انتقال داده ازcache مقدار آن در حافظه اصلی نیز update میگردد.

Improving Cache Performance

Average Memory Access Time (AMAT)

AMAT = Hit time + Miss rate * Miss penalty

Used as a framework for optimizations

- Reduce the Hit time
  - Small and simple caches
- Reduce the Miss Rate
  - Larger cache size, higher associativity, and larger block size
- Reduce the Miss Penalty
  - Multilevel caches

Small and Simple Caches

- Hit time is critical: affects the processor clock cycle
  - Fast clock rate demands small and simple L1 cache designs
- Small cache reduces the indexing time and hit time
  - Indexing a cache represents a time consuming portion
  - Tag comparison also adds to this hit time
- Size of L1 caches has not increased much
  - L1 caches are the same size on Alpha 21264 and 21364
  - Same also on UltraSparc II and III, AMD K6 and Athlon
  - Reduced from 16 KB in Pentium III to 8 KB in Pentium 4

Classifying Misses – Three Cs

- Conditions under which misses occur
- Compulsory: program starts with no block in cache
  - Also called cold start misses
  - Misses that would occur even if a cache has infinite size
- Capacity: misses happen because cache size is finite
  - Blocks are replaced and then later retrieved
  - Misses that would occur in a fully associative cache of a finite siz
- Conflict: misses happen because of limited associativity
  - Limited number of blocks per set
  - Non-optimal replacement algorithm

Average Memory Access Time

- Average Memory Access Time (AMAT)

  AMAT = Hit time + Miss rate × Miss penalty

- Time to access a cache for both hits and misses
- Example: Find the AMAT for a cache with
  - Cache access time (Hit time) of 1 cycle = 2 ns
  - Miss penalty of 20 clock cycles
  - Miss rate of 0.05 per access
- Solution:

  AMAT = 1 + 0.05 × 20 = 2 cycles = 4 ns

  Without the cache, AMAT will be equal to Miss penalty = 20 cycles

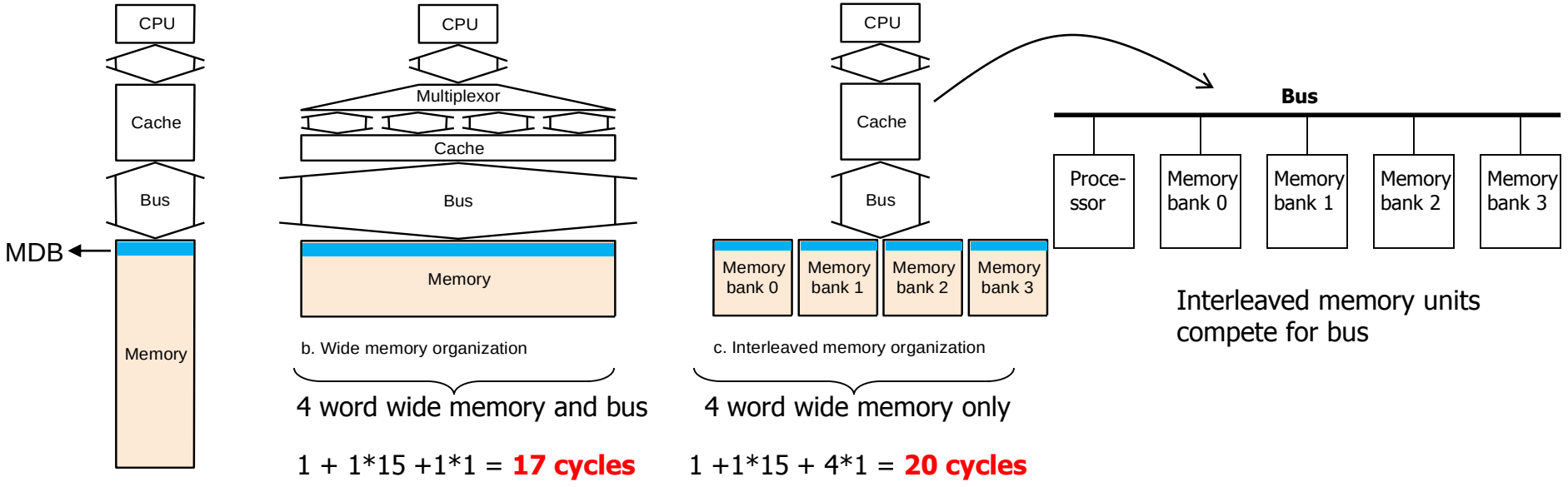
Average Memory Access Time

- AMAT(IC) = Hit time(IC) + Miss rate(IC) × Miss penalty
- AMAT(DC) = Hit time(DC) + Miss rate(DC) × Miss penalty
- AMAT = $1/(1+P_{LS})$ * AMAT(IC) + $P_{LS}/(1+P_{LS})$ * AMAT(DC)

- P_{LS} is the probability of Load/Store instructions
- $1/(1+P_{LS})$ is the probability of accessing the instruction cache
- $P_{LS}/(1+P_{LS})$ is the probability of accessing the data cache

Improving Cache Performance by Increasing Bandwidth

- Assume:
  - Cache block of 4 words
  - 1 clock cycle to send address to memory address buffer (1 bus trip)
  - 15 clock cycles for each memory data access
  - 1 clock cycle to send data to memory data buffer (1 bus trip)



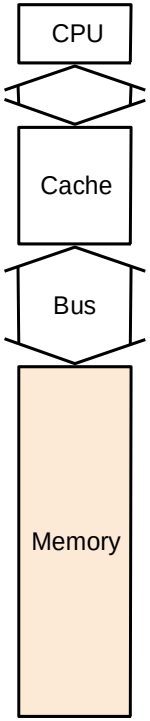
1 + 4*15 + 1 = **62 cycles**

1 + 1*15 +1*1 = **17 cycles**

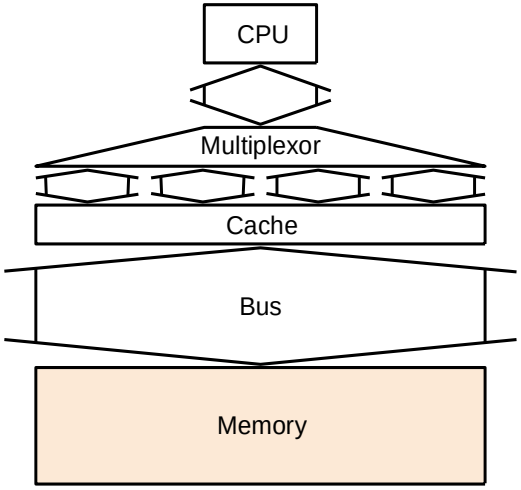
1 +1*15 + 4*1 = **20 cycles**

Miss penalties

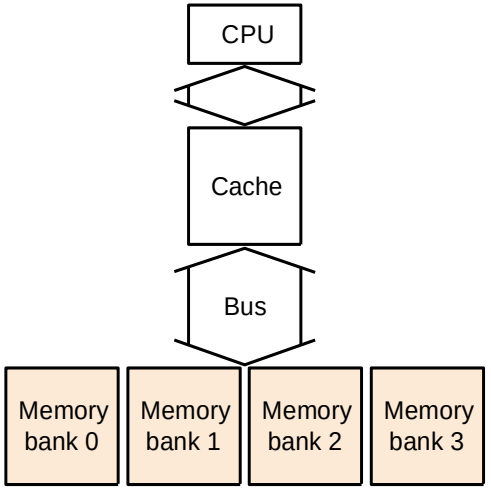
Improving Cache Performance by Increasing Bandwidth



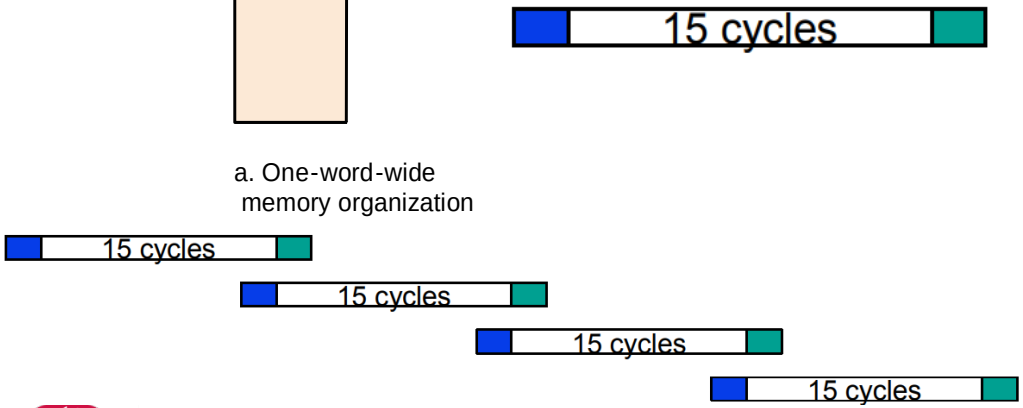
a. One-word-wide memory organization

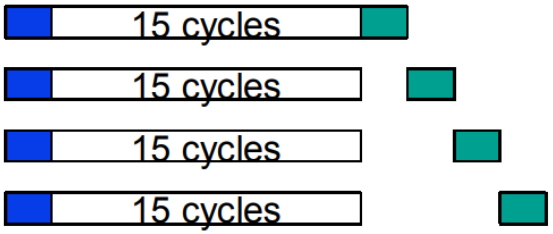


b. Wide memory organization



c. Interleaved memory organization





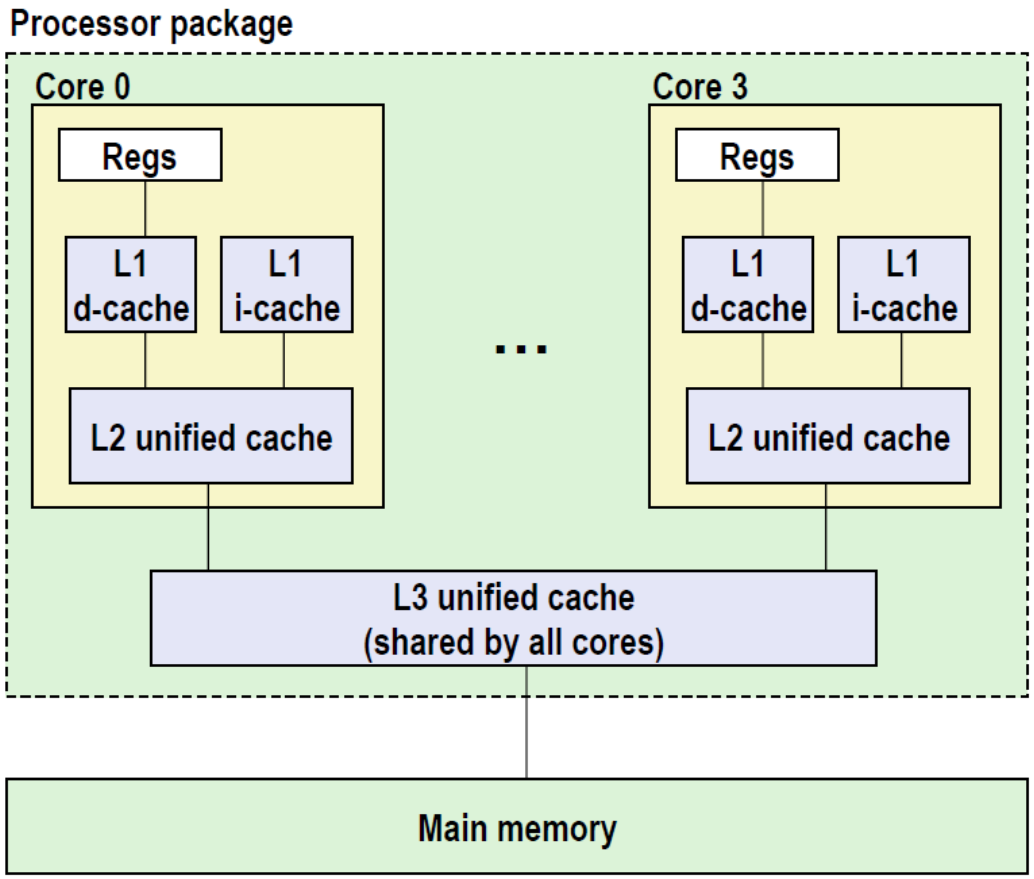
سلسله مراتب حافظه در پردازنده های جدید

Intel Pentium 4, 2.2 GHz Processor.

Component	Access Speed (Time for data to be returned)	Size of Component
Registers	1 cycle = 0.5 nanoseconds	32 registers
L1 Cache	3 cycles = 1.5 nanoseconds	Separate Data and Instruction Caches: 8 Kbytes each
L2 Cache	20 cycles = 10 nanoseconds	256 Kbytes, 8-way set associative
L3 Cache	30 cycles = 15 nanoseconds	512 Kbytes, 8-way set associative
Memory	400 cycles = 200 nanoseconds	16 Gigabytes

سلسله مراتب حافظه در پردازنده های جدید

Intel Core i7 Cache Hierarchy



L1 i-cache and d-cache:
32 KB, 8-way,
Access: 4 cycles

L2 unified cache:
256 KB, 8-way,
Access: 11 cycles

L3 unified cache:
8 MB, 16-way,
Access: 30-40 cycles

Block size: 64 bytes for all caches.

مقایسه Cache در دو پردازنده

	Intel Nehalem	AMD Barcelona
L1 cache organization & size	Split I$ and D$; 32KB for each per core; 64B blocks	Split I$ and D$; 64KB for each per core; 64B blocks
L1 associativity	4-way (I), 8-way (D) set assoc.; ~LRU replacement	2-way set assoc.; LRU replacement
L1 write policy	write-back, write-allocate	write-back, write-allocate
L2 cache organization & size	Unified; 256KB (0.25MB) per core; 64B blocks	Unified; 512KB (0.5MB) per core; 64B blocks
L2 associativity	8-way set assoc.; ~LRU	16-way set assoc.; ~LRU
L2 write policy	write-back	write-back
L2 write policy	write-back, write-allocate	write-back, write-allocate
L3 cache organization & size	Unified; 8192KB (8MB) shared by cores; 64B blocks	Unified; 2048KB (2MB) shared by cores; 64B blocks
L3 associativity	16-way set assoc.	32-way set assoc.; evict block shared by fewest cores
L3 write policy	write-back, write-allocate	write-back; write-allocate

Almost Everything is a Cache !

- In computer architecture, almost everything is a cache!
- Registers: a cache on variables – software managed
- First-level cache: a cache on second-level cache
- Second-level cache: a cache on memory
- Memory: a cache on hard disk
  - Stores recent programs and their data
  - Hard disk can be viewed as an extension to main memory
- Branch target and prediction buffer
  - Cache on branch target and prediction information


Questions