Computer Architecture (CS F342)

Memory/Storage Hierarchy

&

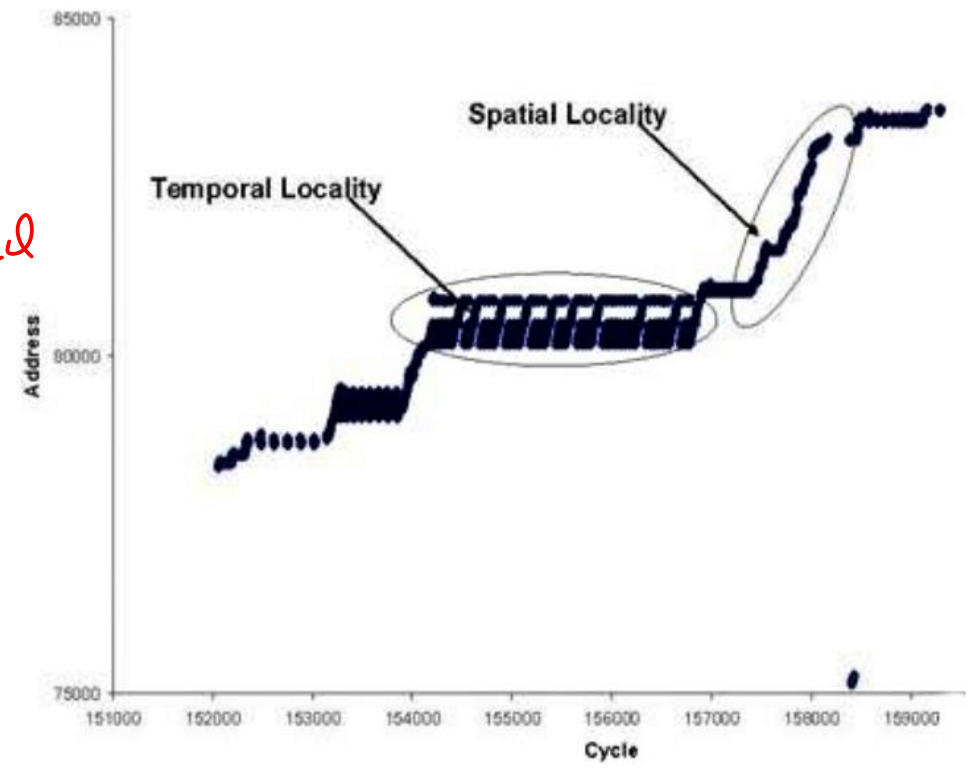
Fast Storage Unit: Cache Memory Architecture & Organization

Why do we need to study memory/storage hierarchy?

- CPU is a component in the computer systems
- Others components: Memory and I/O systems
- Programs exhibit <u>principle of locality</u>
  - Temporal → The same element being accessed
  - Spatial → Neighboring elements being accessed
  - A rule of thumb (the 90/10 rule):

  90% of the execution of programs spends in only 10% of the code

So, both the localities are present in cache



Behavior of a program

Matrix multiplication

- Data stored in row-major order
- Data of A, B & C can be used in near future
- Data neighboring to previously accessed data
- Instructions are also to be used in near future
- Principle of locality

```
for (i=0; i<l; i++)
    for (j=0; j<m; j++)
        for (k=0; k<n; k++)
            A[i][j] += B[i][k] x C[k][j];
```

fast access

slow access

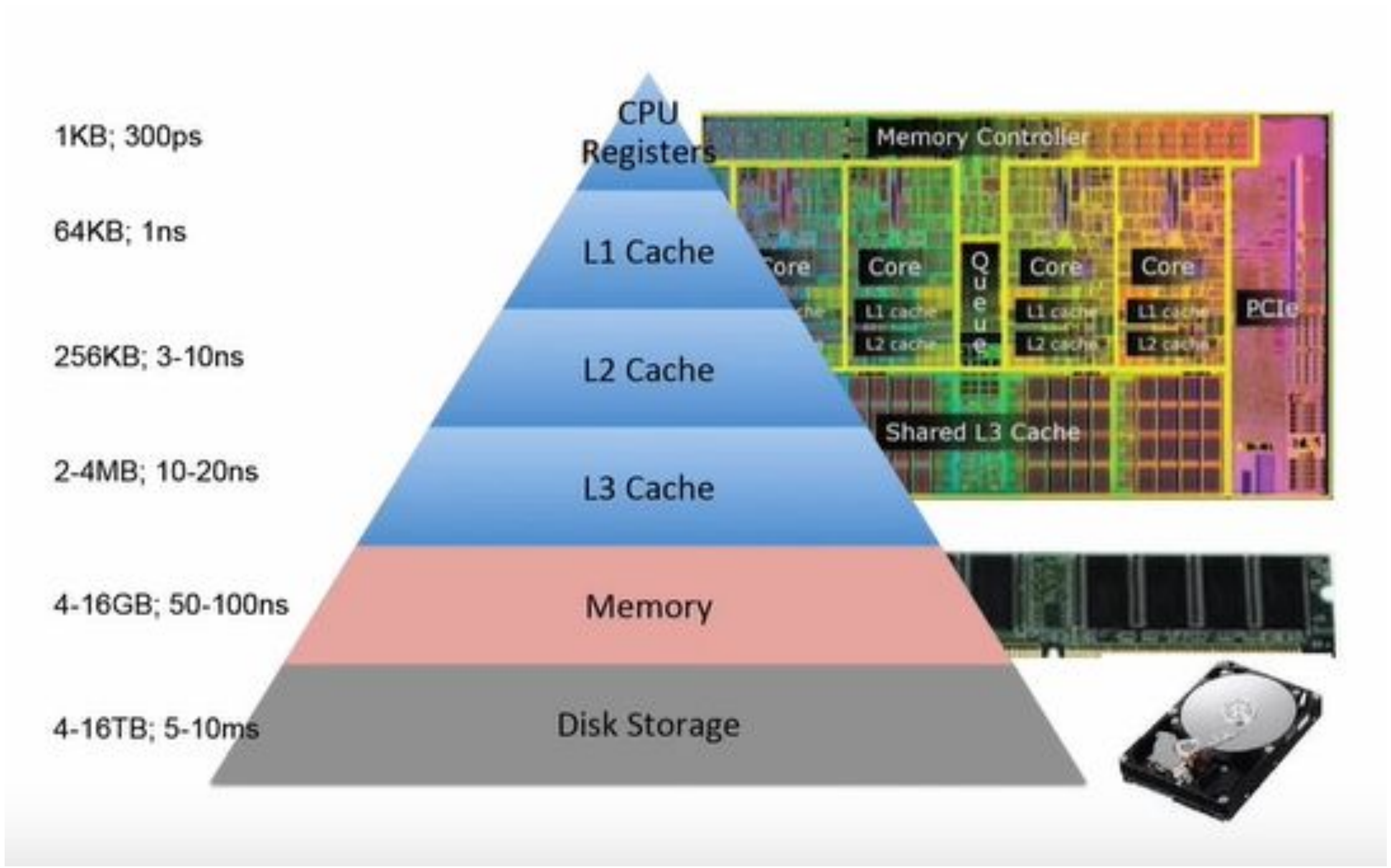
Since neighbors are put in cache.

Why do we need to study memory/storage hierarchy?

- Principle of locality can be found in most of the programs
- To exploit such principle we need memory hierarchy
  - Keep the repeatedly accessed data & instruction near to CPU
  - Not the entire code
- Memory hierarchy
  - Faster but smaller memory closer to CPU
  - Slower but larger memory faraway from the CPU

Memory hierarchy

Access time and space increase



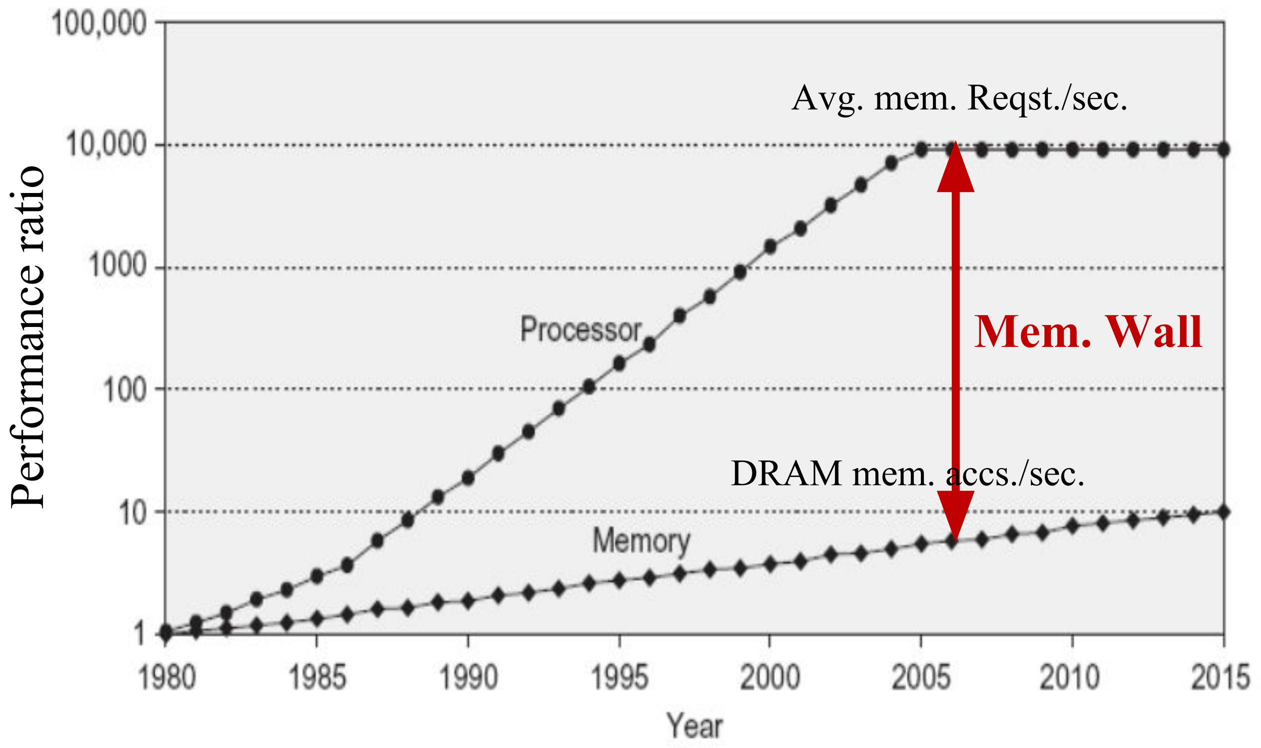
Cost increase

What makes improvement in the storage access time?

- Static Random Access Memory (SRAM) Technology → Costly
  - Registers, L1, L2 & L3 cache
- Dynamic Random Access Memory (DRAM) Technology
  - Main memory →
    - Capacitor is used
    - So DRAM must be changed because change leaks if memory not read for a long time
    - Also after every r/w operation, the change is used up. So it will have to be refreshed.
- Magnetic Technology
  - Hard disk takes longer access time because of mechanical components

Do we really need memory hierarchy?

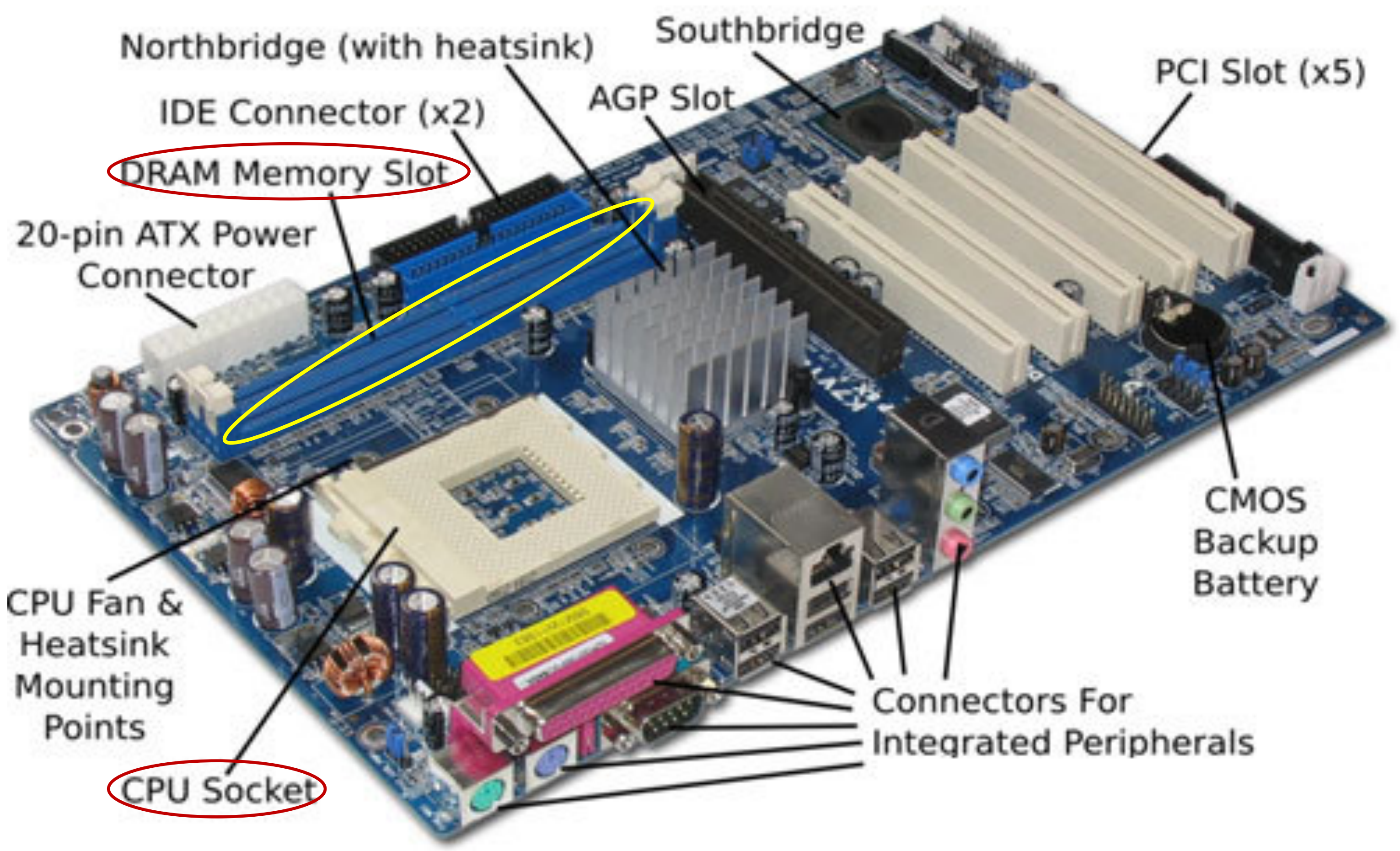
- Memory Wall Problem:
  - Significant increase in processor performance over the years
  - Not significant increase in main-memory performance over the years



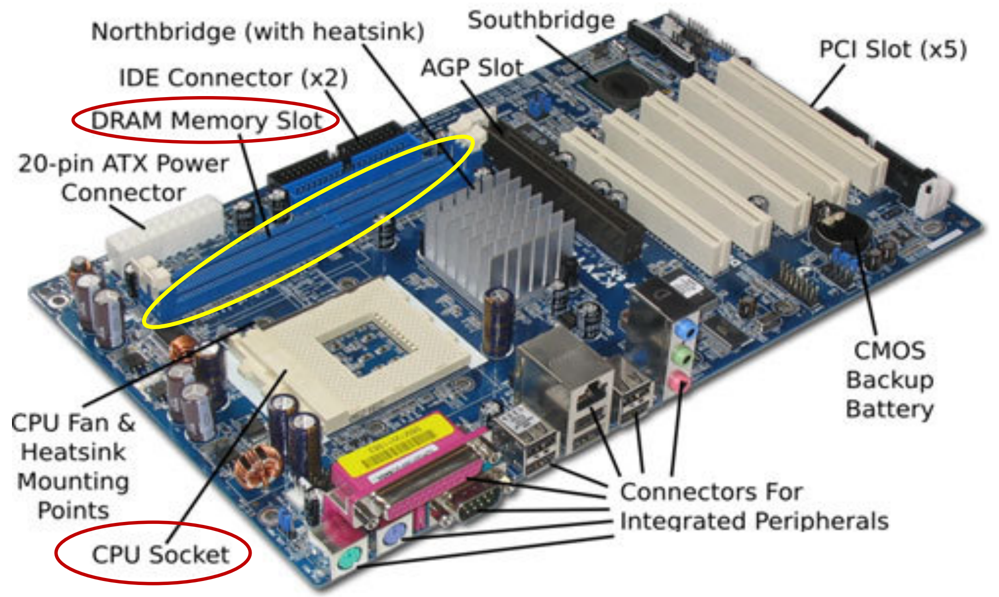
Necessity of memory hierarchy

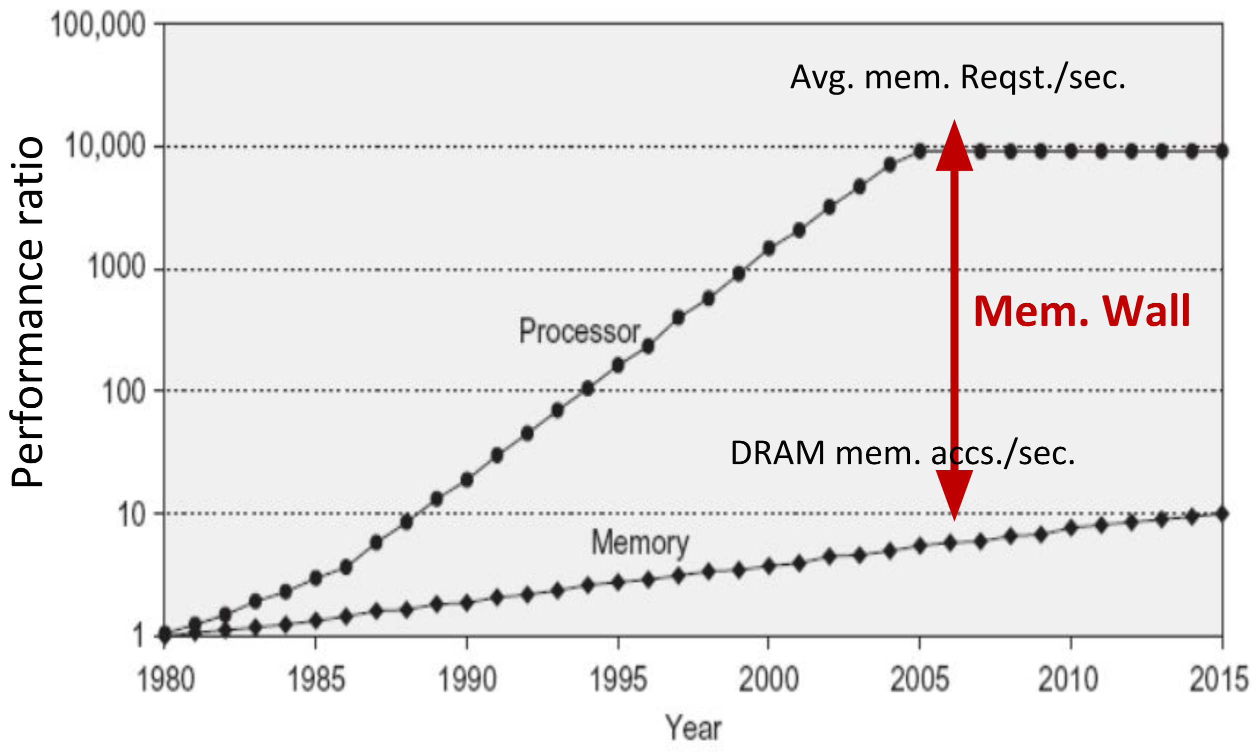
- From the previous graph
  - The gap is increasing
- The previous graph did not include the multiprocessors
  - The aggregated peak bandwidth requirement increases with no. cores/processors
- How does one deal with this increasing gap?
  - Need memory hierarchy: multi-levels of cache hierarchy

Why do we need cache memory?


Northbridge (with heatsink)
Southbridge
PCI Slot (x5)
AGP Slot
IDE Connector (x2)
DRAM Memory Slot
20-pin ATX Power Connector
CMOS Backup Battery
CPU Fan & Heatsink Mounting Points
Connectors For Integrated Peripherals
CPU Socket

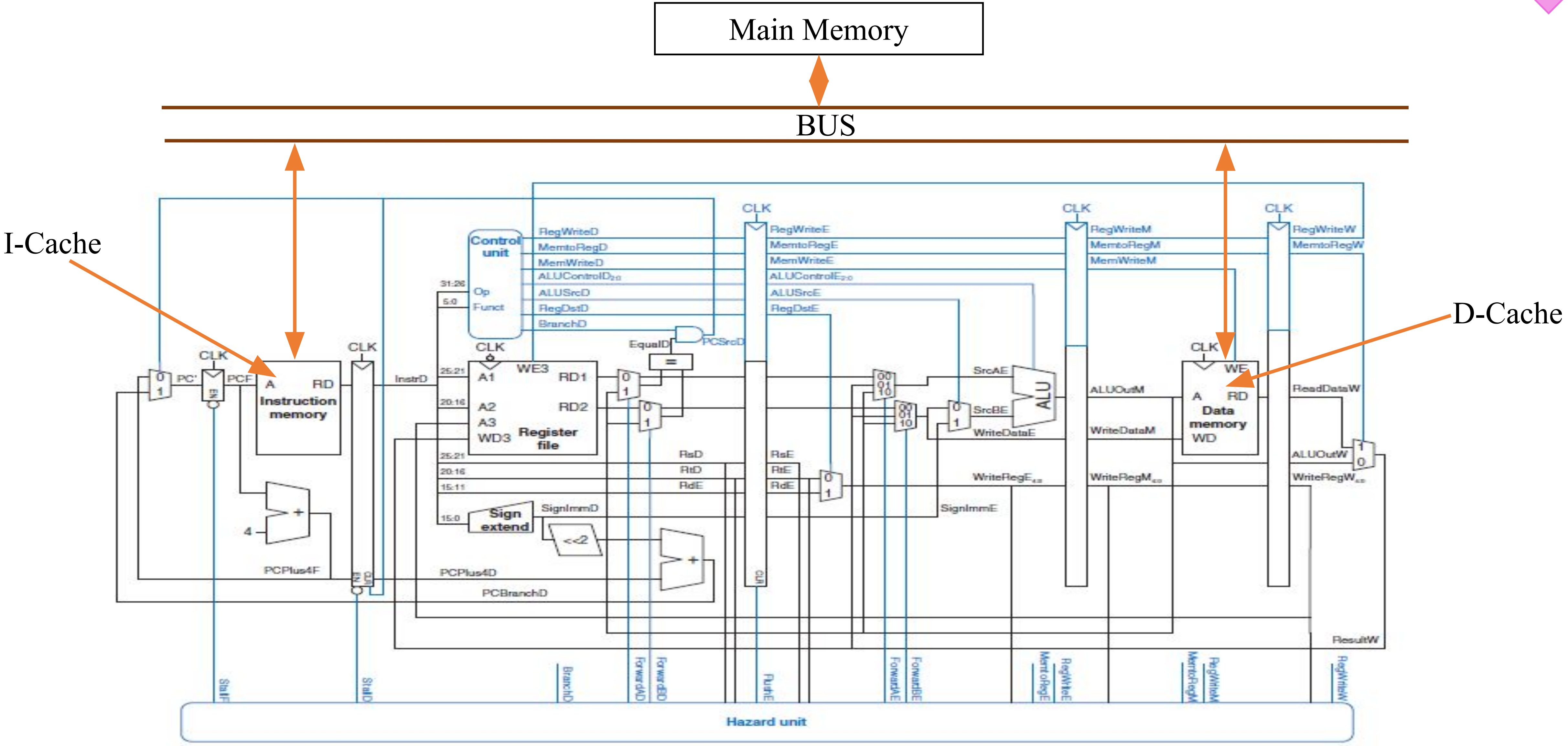
Memory Wall Problem and Necessity of Cache Memory


Northbridge (with heatsink)
Southbridge
IDE Connector (x2)
AGP Slot
PCI Slot (x5)
DRAM Memory Slot
20-pin ATX Power Connector
CMOS Backup Battery
CPU Fan & Heatsink Mounting Points
Connectors For Integrated Peripherals
CPU Socket



- Performance difference between processor (CPU) and memory by technology and memory is placed far away (*nm* scale) from CPU (off-chip)
- There is a gap in CPU's request (rate) for the memory accesses and the service (rate) for those request by memory
-
- Cache memory technique can <u>speed up</u> the performance of the memory accesses time

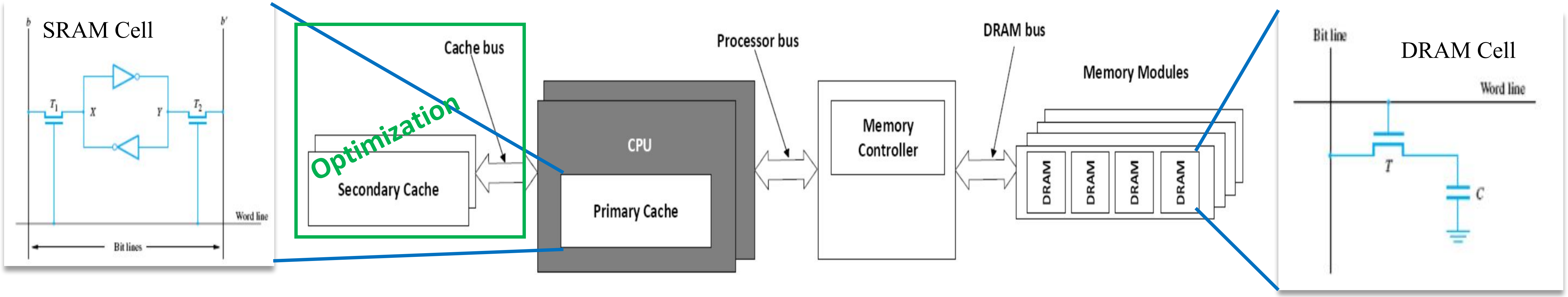
Caches are in MIPS-processor



Cache Memory Architecture

- Cache memory is a high-speed storage unit
- **What makes it faster as compare to the memory in question?**
  - 1) Position 2) SRAM-based memory technology
- **What would be the size and characteristic of the Cache memory?**
  - Smaller in size and bit access time must be faster

If Cache gets bigger, access time becomes slower



Cache size is smaller than the main memory & its associated challenges

- Which data are to be stored/kept in cache? → OS decides
- What could be the granularity of the data-size? → Arc. Based
- Who will decide such granularity?
- Where do we place the data?
- Which data are to be evict out, if cache (line) is full?
- How to update data?
- <u>How to organize the cache for the programs to run efficiently?</u>

- <u>Do we know a related problem?</u>
- How does one organize the cloths in a closet or an Almira?
- Are we able to answer all the unknown in the left part from this problem?
- Which behavior to be used for organizing the closet?
- Can we use such method to organize the cache memory?
  - Directly (?)
  - Or modified one (?)

Can we imagine a case in which we could utilize this method or result?

Program’s Behavior and Characteristics of Cache Memory

Which memory request should be keep in the cache?

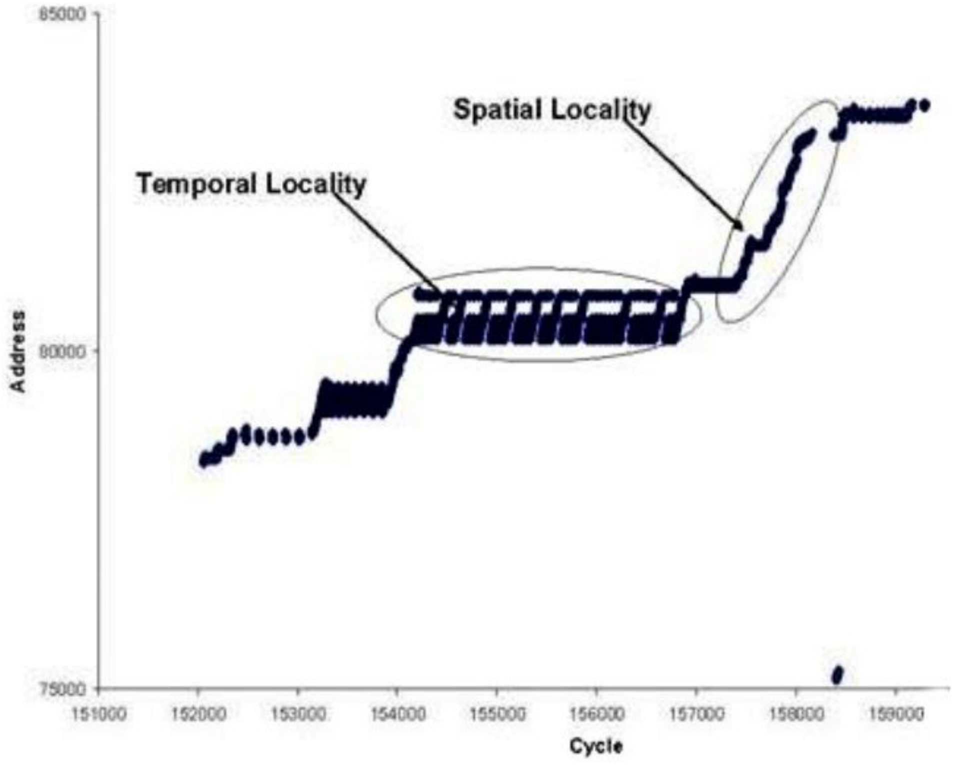
- Program’s behavior: programs tend to reuse data and instructions they have used recently.
- Spatial locality and Temporal locality

How many such instructions and/or data one keep in the cache?

- More than one or a block of instructions and/or data

How does one decide the block size?

- Processor’s cache is managed by its own set of heuristics or rules
- For example, in Intel i7, block size of the primary cache is 64 bytes



Behavior of a program

Characteristics of Cache Memory

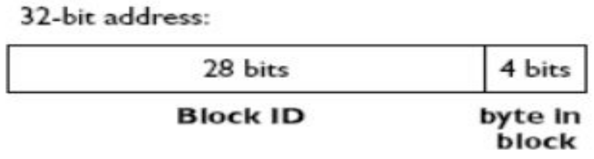
The following items are embedded in the cache:

- **Organization**
  - The logical arrangement of storage unit/data
- **Content-management heuristics**
  - Decide the best possible items for caching and evict out the candidate to make room for more important data not yet cached
- **Consistency-management heuristics**
  - Ensure that the instructions and data that the program expects to receive are the ones the program does, indeed, receive
  - Consistency with 1) self, 2) main memory 3) other caches

Whenever an upate is done in the cache, it should reflect wherever required.

Cache Organization: Blocks, Tags and Set

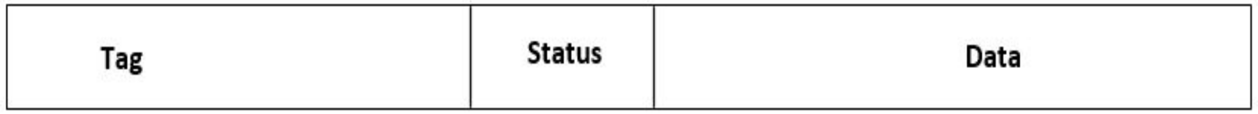
A cache stores chunks of data (called cache blocks or cache lines) that come from the memory.



A cache is typically much smaller than the memory:

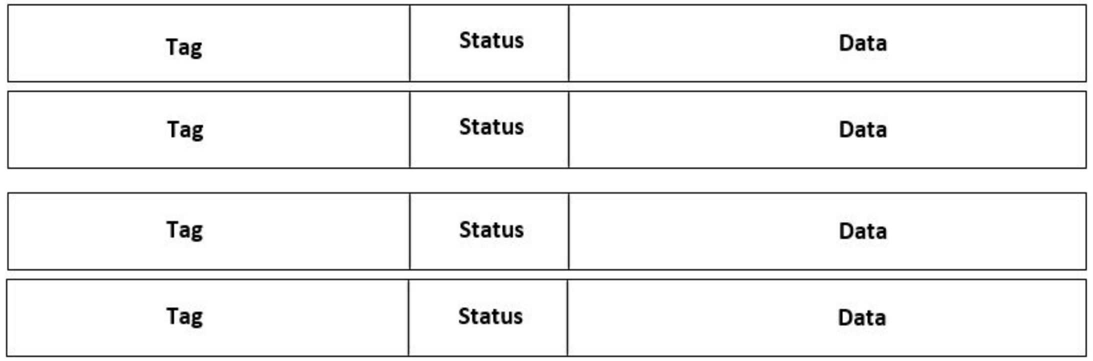
How does CPU know whether any particular datum is present in the cache or not?

Cache tags fulfil this necessity.



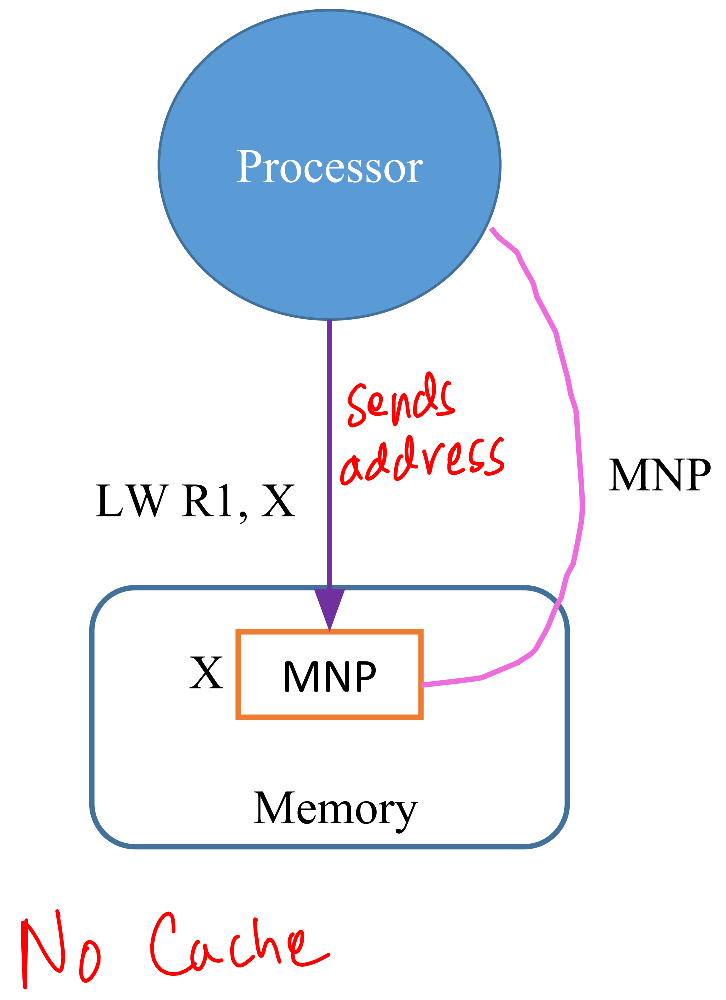
What if the cache contains more than one line?

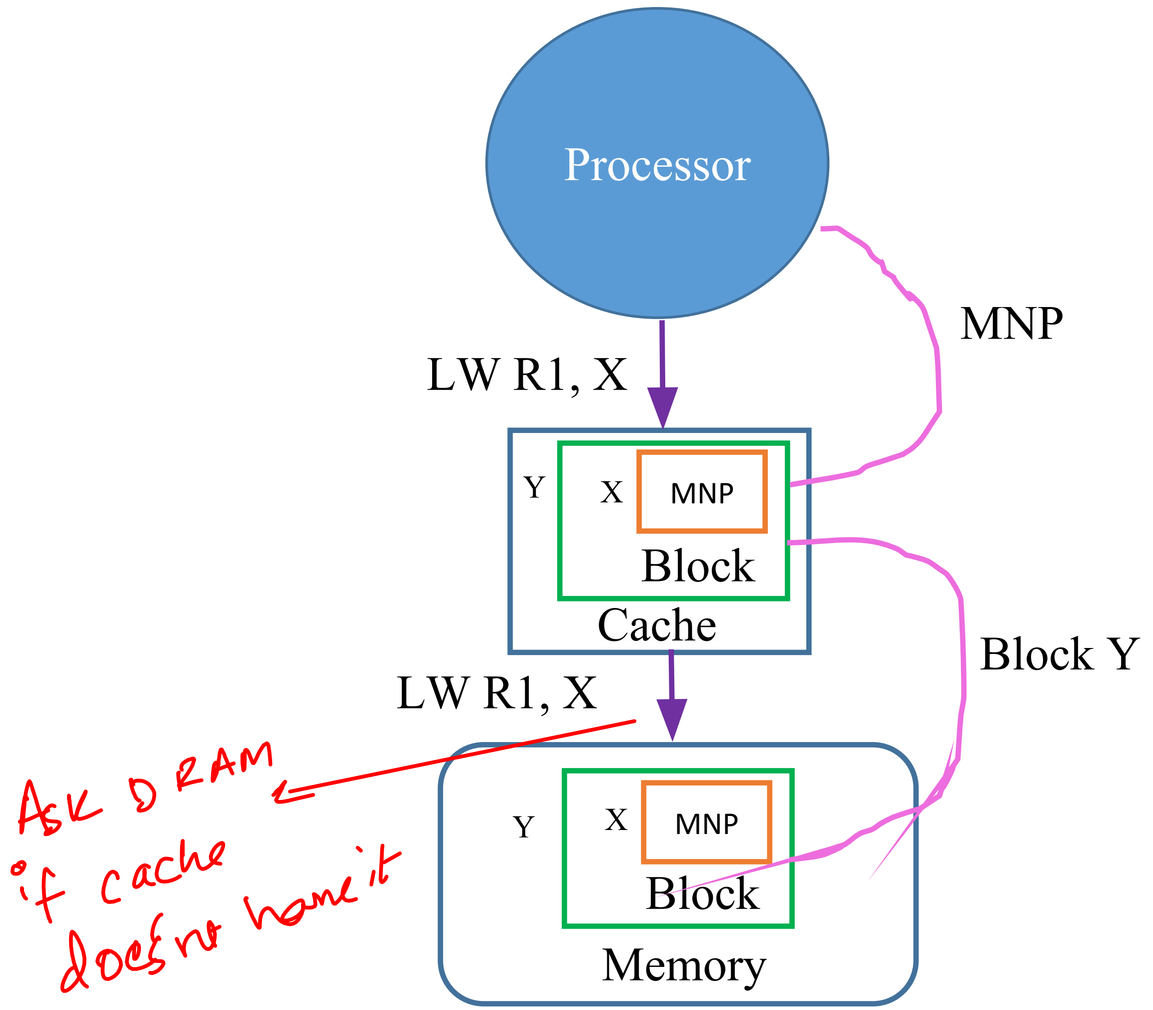
Cache can have the set of choices for the incoming blocks



Cache Memory

- Cache Operations
- What could be the size of the block in cache & in main memory?





How many ways one map the incoming block from main memory onto the Cache?

- A block can be placed
  - Anywhere
  - A fixed position
  - A set of positions

3 types

Tag	Status	Data
Tag	Status	Data
Tag	Status	Data
Tag	Status	Data

- Is there any similar problem available?
  - Container & items
  - Taking notes during class & notebooks
  - Can we use this strategy?

Can we imagine a case in which we could utilize this method or result?

How does one measure the performance of these methods?

- Matrices to measure the performance of the methods
  - Cache hit rate = $\frac{\#\ of\ hits}{\#\ of\ total\ memory\ request} = (1 - miss\ rate)$
  - Cache miss rate = $\frac{\#\ of\ misses}{\#\ of\ total\ memory\ request} = (1 - hit\ rate)$
  - Average memory access time (AMAT)

    $= Hit\ time + Miss\ rate \times Miss\ penalty$
  - Usage of resources: # of components & energy

Example

- Suppose a program has 2000 data access instructions (loads or stores), and 1250 of these requested data values are found in the cache. The other 750 data values are supplied to the processor by main memory or disk memory. What are the miss and hit rates for the cache?

- The miss rate is 750/2000 = 0.375 = 37.5%
- The hit rate is 1250/2000 =0.625 = 1 − 0.375 = 62.5%

- Can we imagine a case in which we could utilize this method or result?

Example

- Suppose a computer system has a memory organization with only two levels of hierarchy, a cache and main memory. What is the average memory access time given the access times and miss rates in the given Table?

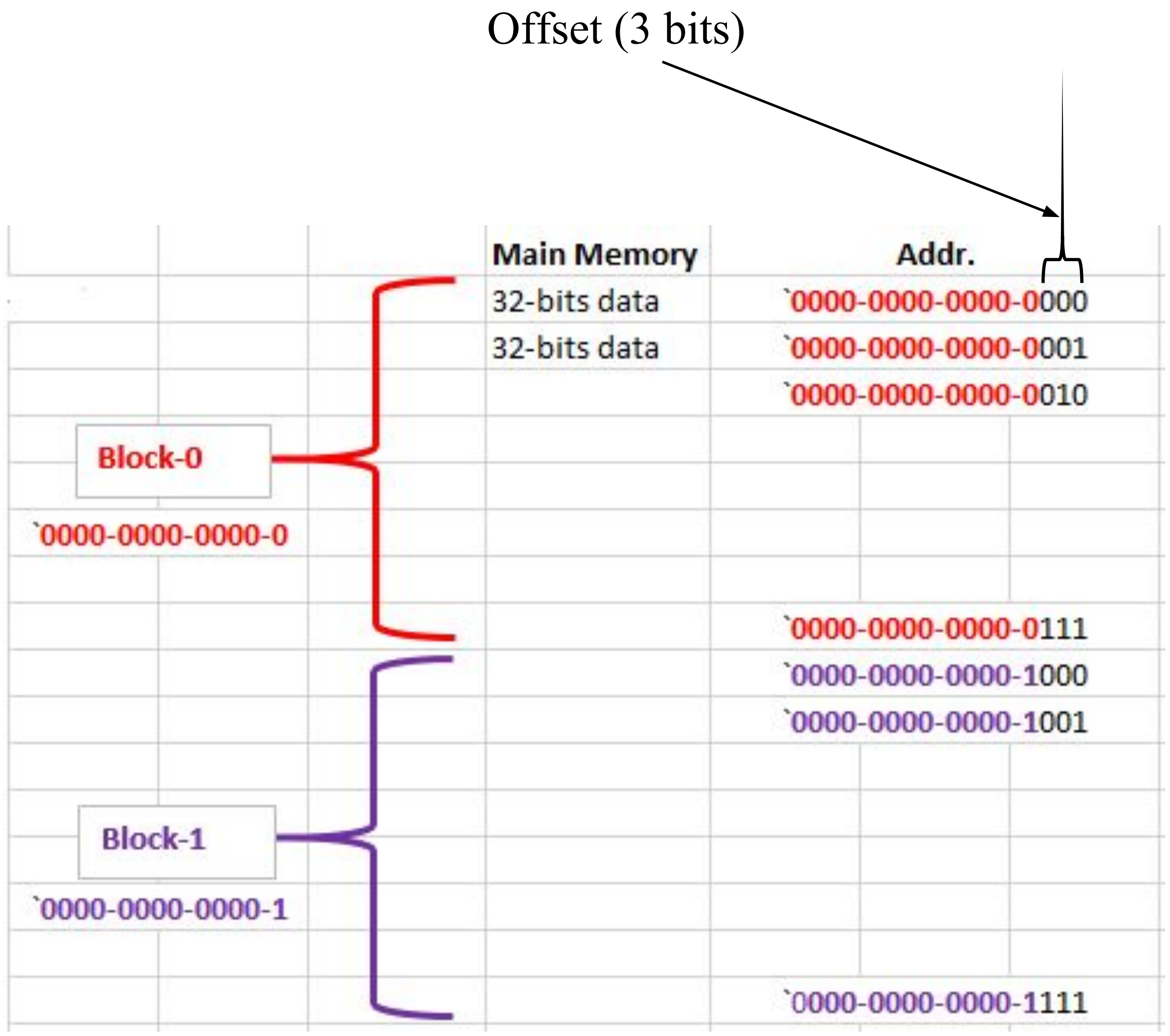
Memory level	Access time (cycles)	Miss rate
Cache	1	10%
Main memory	100	0%

- The average memory access time is $1 + 0.1(100) = 11$ cycles

- Can we imagine a case in which we could utilize this method or result?

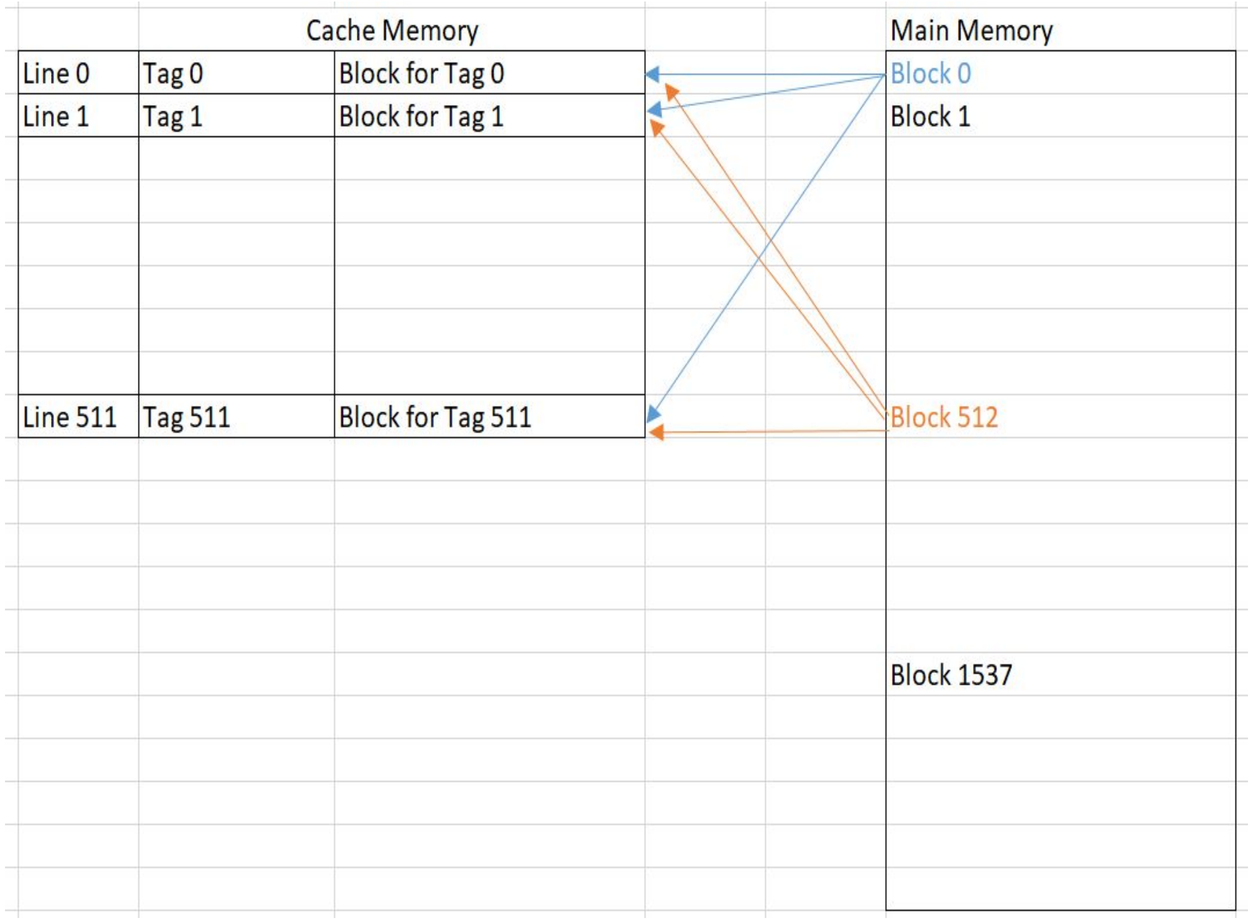
Concept of Block, Cache line & Address

- What is the block size here?
  - Block size is 8 x 32 bits
- CPU generates memory address
  - Block & offset
- Block size in main memory is equal as the cache line



How does one map a incoming block onto anywhere in the cache?

- Fully Associative Cache Organization
  - Block size in main memory is equal as in cache line
  - CPU generates memory address
    - Block & offset
  - Tag will contribute to identify a block
    - Tag contains block address
  - Any block of the MM can be mapped onto any cache line
- Example & Architecture
  - see in the CacheArchitecute.xlsx



Summary of the example

- # of cache line is less that the number of blocks in main memory
- Tag line is as in number of cache line
- Size of Tag memory is (# of Tag line × Tag data)
- # of comparator is (# of tag lines)
- Data Cache size is (# of cache line × Data size)
- How about cache hit rate?
- How does one implement this method?

Example:1

- A cache has the following parameters: b, block size given in numbers of words; S, number of blocks; and A, number of address bits.
- In terms of the parameters described, what is the cache capacity, C?
- C=b*S
- In terms of the parameters described, what is the total number of bits required to store the tags?
- No of bits for Tag = A – $\log_2$ (b)
- What is S for a fully associative cache of capacity C words with block size b?
- S=C/b
- Can we imagine a case in which we could utilize this method or result?

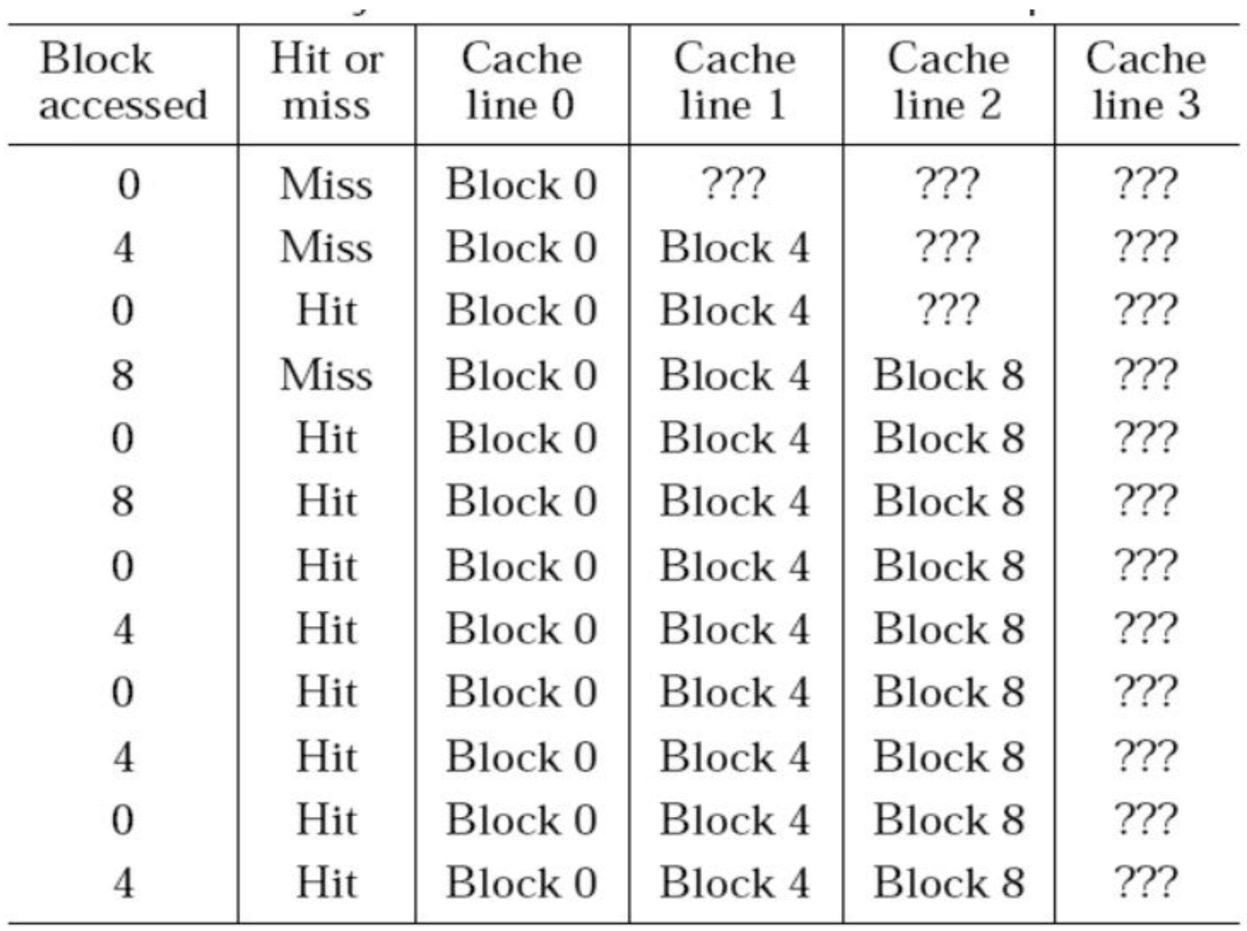
Example:2

- The table below represents three lines from a cache that uses fully associative mapping with a block size of eight. Identify the address of the red data ($C9_{16}$).
- 0100011010101001
- Can we imagine a case in which we could utilize this method or result?

TAG	Word id bits (in binary)							
	000	001	010	011	100	101	110	111
0110110110010_2	16	36	66	28	A1	3B	D6	78
0100011010101_2	54	C9	6A	63	54	32	00	D3
0001000111011_2	29	8C	ED	FD	29	54	12	F3

Example:3

- Consider the program’s reference pattern of accessing the blocks 0, 4, 0, 8, 0, 8, 0, 4, 0, 4, 0, 4. Assuming that the cache uses associative mapping, find the hit ratio with a cache size of four cache lines.



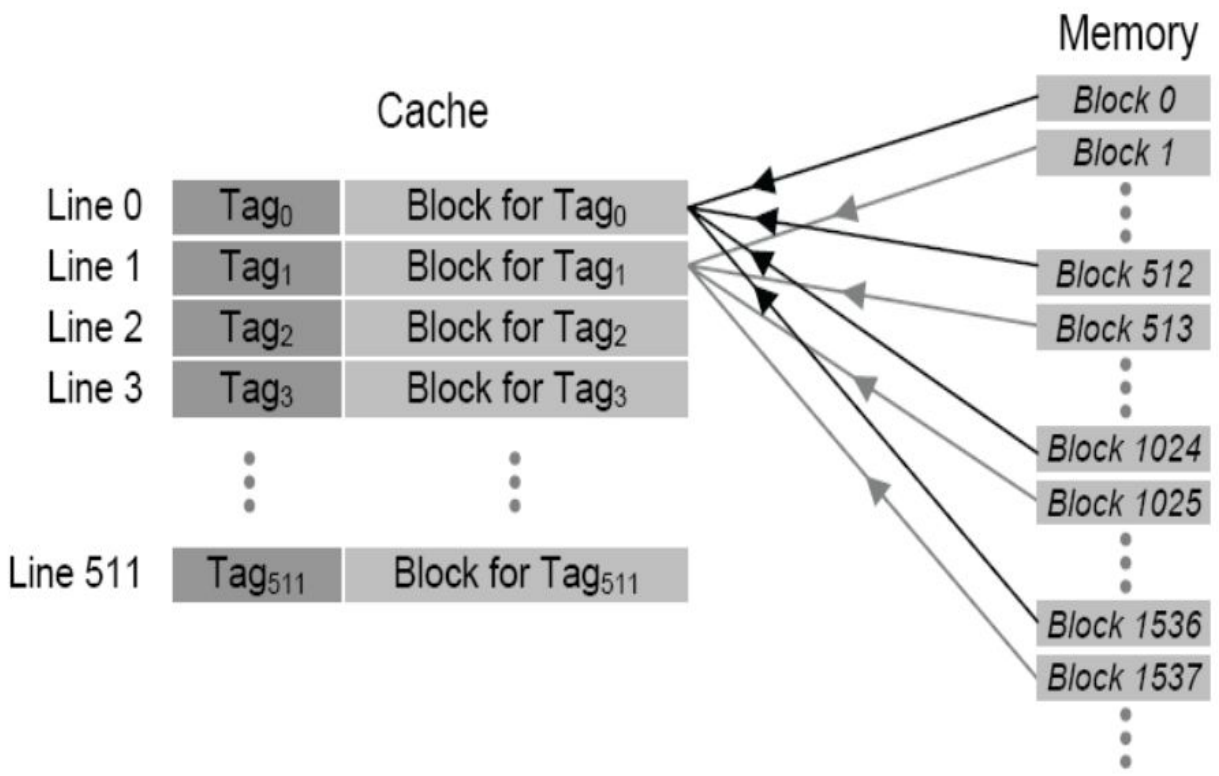
Block accessed	Hit or miss	Cache line 0	Cache line 1	Cache line 2	Cache line 3
0	Miss	Block 0	???	???	???
4	Miss	Block 0	Block 4	???	???
0	Hit	Block 0	Block 4	???	???
8	Miss	Block 0	Block 4	Block 8	???
0	Hit	Block 0	Block 4	Block 8	???
8	Hit	Block 0	Block 4	Block 8	???
0	Hit	Block 0	Block 4	Block 8	???
4	Hit	Block 0	Block 4	Block 8	???
0	Hit	Block 0	Block 4	Block 8	???
4	Hit	Block 0	Block 4	Block 8	???
0	Hit	Block 0	Block 4	Block 8	???
4	Hit	Block 0	Block 4	Block 8	???

Total Ref.=12
Hit ratio = 9/12=75%
Miss ratio = 3/12=25%

How does one map a incoming block onto a fixed location in the cache?

- Direct Cache Organization
  - Block size in main memory is equal as in cache
  - CPU generates memory address
    - Block & offset
  - Set/Group will contribute to identify the location in the cache
  - Tag will contribute to identify a block
    - Tag contains block address

- Example & Architecture
  - see in the CacheArchitecute.xlsx
- Can we imagine a case in which we could utilize this method or result?



Summary of the example

- # of cache line is less that the number of blocks in main memory
- Tag line is as in number of cache line
- Size of Tag memory is (# of Tag line × Tag data)
- # of comparator is one
- Data Cache size is (# of cache line × Data size)
- How about cache hit rate?
- How about cache hit rate as compared to fully associative cache organization?
- How does one implement this method?

Example:1

- A cache has the following parameters: b, block size given in numbers of words; S, number of blocks; and A, number of address bits.
- In terms of the parameters described, what is the cache capacity, C?
- C=b*S
- In terms of the parameters described, what is the total number of bits required to store the tags?
- No of bits for Tag = A – $\log_2(b)$ – $\log_2(S)$
- What is S for a direct mapped cache of size C words and block size b?
- S=C/b
- Can we imagine a case in which we could utilize this method or result?

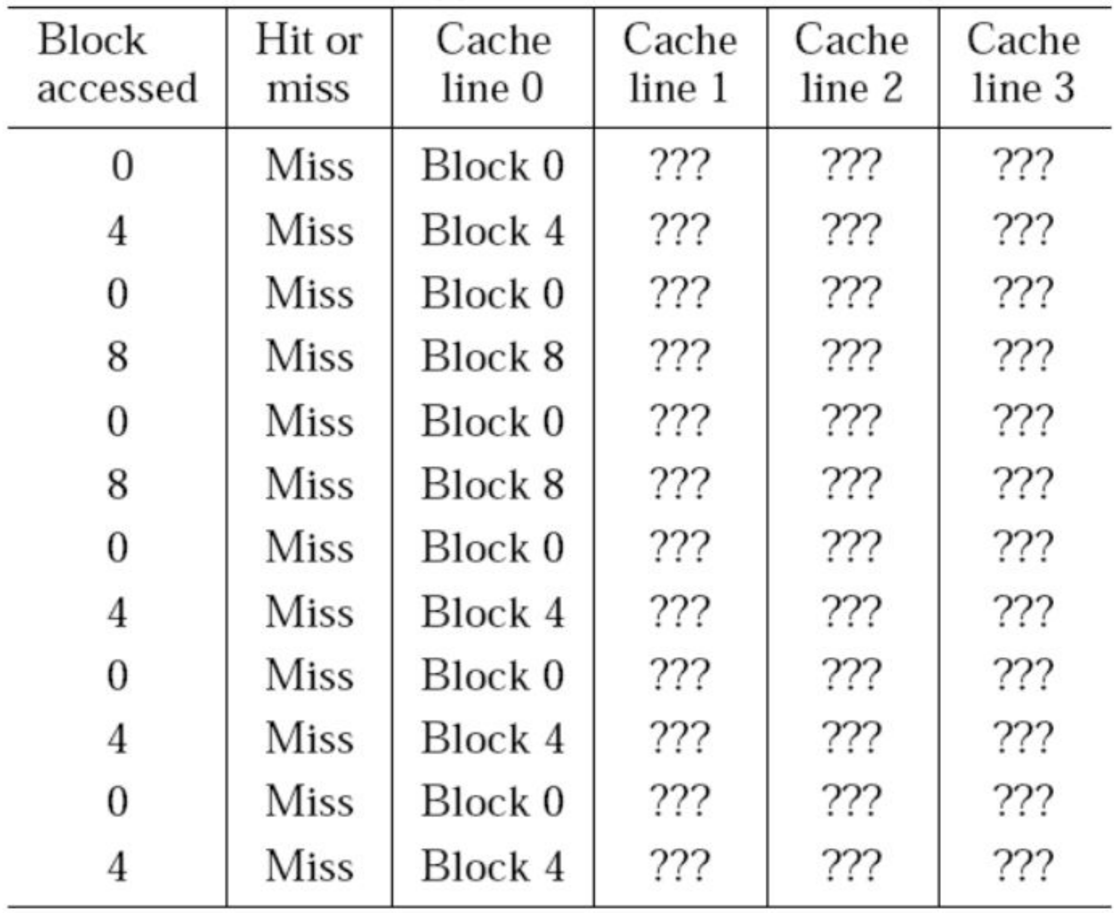
Example:2

- The table below represents 5 lines from a cache. A block contains 4 words. Identify the address of the red data ($D8_{16}$). Cache has 256 line.
- 11001100000010010
- Can we imagine a case in which we could utilize this method or result?

Line	Tag	Word-00	Word-01	Word-10	Word-11
0	110101	12	34	56	78
1	010101	54	32	6A	D3
2	000111	29	8C	ED	F3
3	001100	33	A2	2C	C8
4	110011	9A	BC	D8	F0

Example:3

- Consider the program's reference pattern of accessing the blocks 0, 4, 0, 8, 0, 8, 0, 4, 0, 4, 0, 4. Assuming that the cache uses direct-mapped cache, find the hit ratio with a cache size of four cache lines.



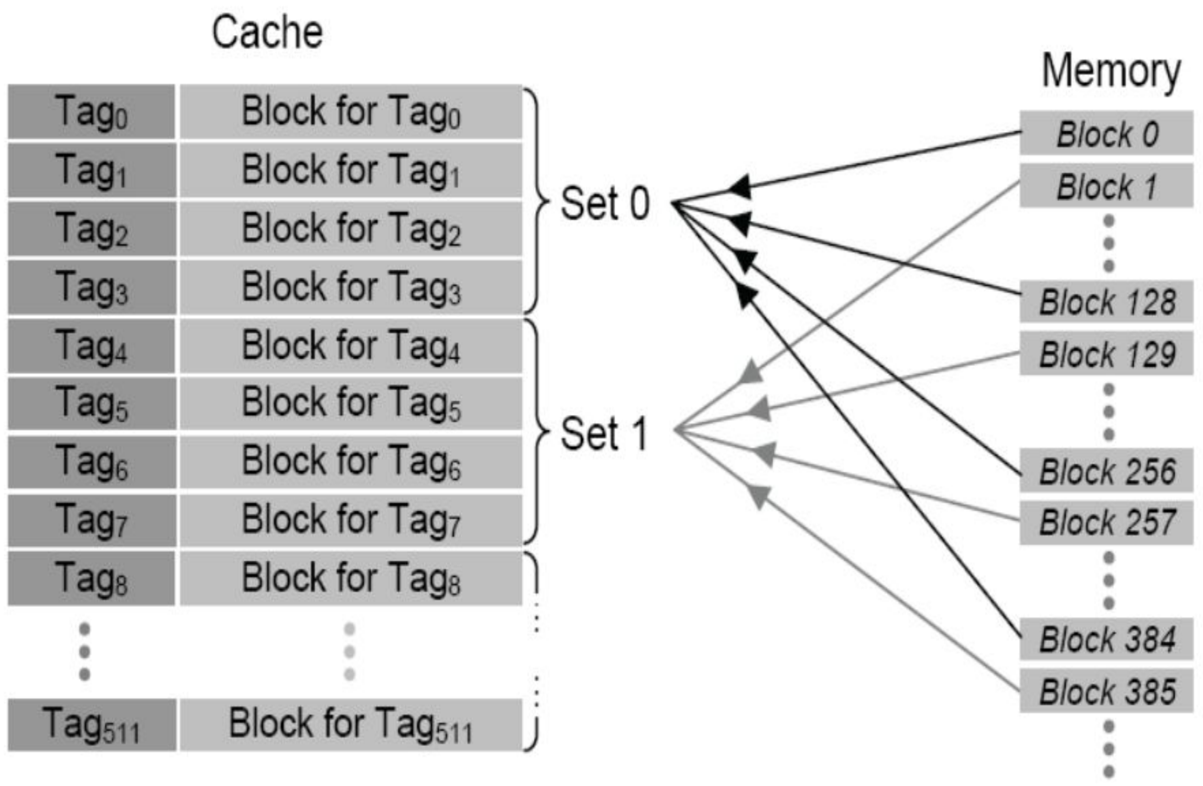
Block accessed	Hit or miss	Cache line 0	Cache line 1	Cache line 2	Cache line 3
0	Miss	Block 0	???	???	???
4	Miss	Block 4	???	???	???
0	Miss	Block 0	???	???	???
8	Miss	Block 8	???	???	???
0	Miss	Block 0	???	???	???
8	Miss	Block 8	???	???	???
0	Miss	Block 0	???	???	???
4	Miss	Block 4	???	???	???
0	Miss	Block 0	???	???	???
4	Miss	Block 4	???	???	???
0	Miss	Block 0	???	???	???
4	Miss	Block 4	???	???	???

Total Ref.=12
Hit ratio = 0/12=0%
Miss ratio = 12/12=100%

Cache thrashing

How does one map a incoming block onto the fixed set of locations in the cache?

- Set Associative Cache Organization
  - Block size in main memory is equal as in cache
  - CPU generates memory address
    - Block & offset
  - Set/Group will contribute to identify the location in the cache
  - Tag will contribute to identify a block
    - Tag contains block address
  - Combination of Fully associative & Direct cache organization
- Example & Architecture
  - see in the CacheArchitecute.xlsx
- Can we imagine a case in which we could utilize this method or result?


Cache
Tag$_0$ Block for Tag$_0$
Tag$_1$ Block for Tag$_1$
Tag$_2$ Block for Tag$_2$
Tag$_3$ Block for Tag$_3$
Tag$_4$ Block for Tag$_4$
Tag$_5$ Block for Tag$_5$
Tag$_6$ Block for Tag$_6$
Tag$_7$ Block for Tag$_7$
Tag$_8$ Block for Tag$_8$
Tag$_{511}$ Block for Tag$_{511}$
Set 0
Set 1
Memory
Block 0
Block 1
Block 128
Block 129
Block 256
Block 257
Block 384
Block 385

Summary of the example

- # of cache line is less that the number of blocks in main memory
- Tag line is as in number of cache line
- Size of Tag memory is (# of Tag line × Tag data)
- # of comparator is (size of the set)
- Data Cache size is (# of cache line × Data size)
- How about cache hit rate?
- How about cache hit rate as compared to fully associative & direct cache organization?
- How does one implement this method?

Example:1

- A cache has the following parameters: b, block size given in numbers of words; S, number of blocks; N, number of ways, and A, number of address bits.
- In terms of the parameters described, what is the cache capacity, C?
- C=b*S
- In terms of the parameters described, what is the total number of bits required to store the tags?
- No of bits for Tag = $A - \log_2(b) - \log_2 (S/N)$
- What are S and N for a direct mapped and fully associative cache of size C words and block size b?
- S=C/b, N=S for fully associative cache
- S=C/b, N=1 for direct mapped cache
- Can we imagine a case in which we could utilize this method or result?

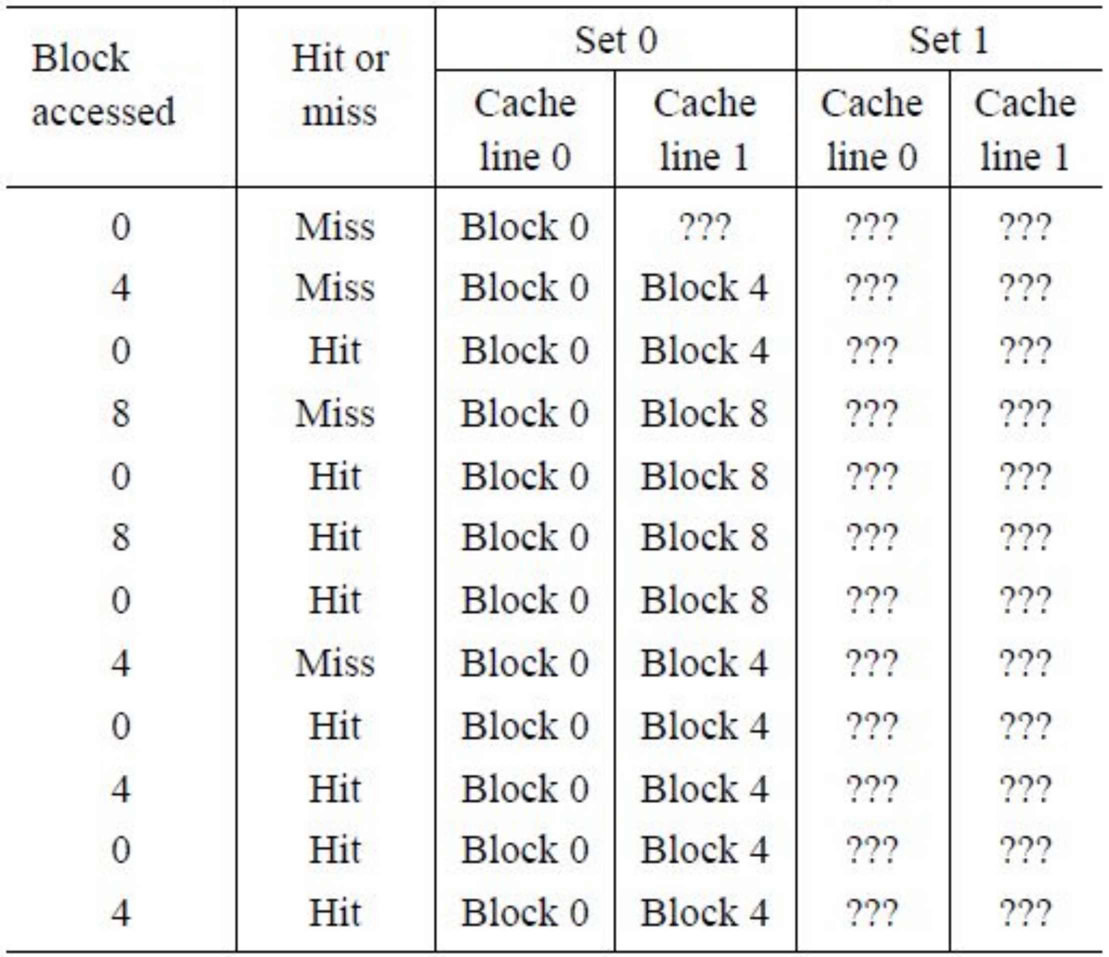
Example:2

- The table below represents 4 lines from a cache. A block contains 4 words. Identify the address of the red data ($D8_{16}$). Cache has 256 line. 2-ways associative mapping technique is used.
- 110011000000110
- Can we imagine a case in which we could utilize this method or result?

Set	Line	Tag	Word-00	Word-01	Word-10	Word-11
0	0	1101011	12	34	56	78
	1	0101010	54	32	6A	D3
1	2	0001110	29	8C	ED	F3
	4	1100110	9A	BC	D8	F0

Example:3

- Consider the program’s reference pattern of accessing the blocks 0, 4, 0, 8, 0, 8, 0, 4, 0, 4, 0, 4. Assuming that the set size is 2, find the hit ratio with a cache size of four cache lines.



Block accessed	Hit or miss	Set 0		Set 1	
		Cache line 0	Cache line 1	Cache line 0	Cache line 1
0	Miss	Block 0	???	???	???
4	Miss	Block 0	Block 4	???	???
0	Hit	Block 0	Block 4	???	???
8	Miss	Block 0	Block 8	???	???
0	Hit	Block 0	Block 8	???	???
8	Hit	Block 0	Block 8	???	???
0	Hit	Block 0	Block 8	???	???
4	Miss	Block 0	Block 4	???	???
0	Hit	Block 0	Block 4	???	???
4	Hit	Block 0	Block 4	???	???
0	Hit	Block 0	Block 4	???	???
4	Hit	Block 0	Block 4	???	???

All blocks with an even number are mapped to set 0, and odd numbered blocks to set 1.

What if a set is full: need to make a decision as to which block should be replaced. This is determined by the replacement policy. Replacement policies will be discussed next. <u>For now, we use a policy that replaces a block that has not been accessed recently</u>.

Total Ref.=12
Hit ratio = 8/12=67%
Miss ratio = 4/12=33%

Is the hit ratio better than direct mapping and fully associative mapping?

Access mechanism of cache

→ Data should be ready while the matching takes place

- Parallel mode of access—access both the tag and data in parallel
  - Access time fast; but power consumption is high
  - Applicable to the cache (L1) nearer to CPU
- Serial mode of access—access the tag first and then the data part
  - Access time more; but less power consumption
  - Applicable to the cache (L2 and L3) far away from CPU

Different types of cache misses

- *Compulsory misses*: Initial access to a block, independent of the cache size
  - What if we increase the block size?
- *Capacity misses*: Occurs as there is no space in the cache (fully associative cache)
  - Increase the cache size to reduce the misses
- *Conflict misses*: Consider Set Asso. Cache, cache misses occur due to lack of space in the specific set. Although, there can be space in other sets.
  - Increase the associativity

Different types of cache misses

- *Compulsory misses* occurs less as compare to *conflict misses*. We have to optimize the caches in such a way that to reduce the conflict misses, for improving the performance.
- *Direct cache* --- search is simple; but high conflict miss rate
- *Fully Associative Cache* --- Complex search, low conflict misses
- *Set Associative Cache* --- Less complex search, reduces conflict miss rate
- Modern processors uses either Set Associative or Direct Cache architecture

$\longrightarrow$ Don't want Bulky Processors.

Optimization Techniques to Improve the Memory Access Time (or Miss Rate)

How does following items affect the cache miss rate?:

- Cache size (higher or smaller)
- Cache block size (higher or smaller)
- Set or associativity size (higher or smaller)
- Cache hierarchies

  Issues in hierarchy management

Summary

- Necessity of Memory Hierarchy
- Necessity of Cache memory
- Characteristic of Cache memory
- Program’s behavior
- Elements of Cache Organization
- Cache Mapping techniques
  - Fully Associative
  - Direct
  - Set Associative
- Elements of Cache Optimization