CSARCH Lecture Series: Cache Memory: Basic Concepts

Sensei RL Uy

College of Computer Studies

De La Salle University

Manila, Philippines

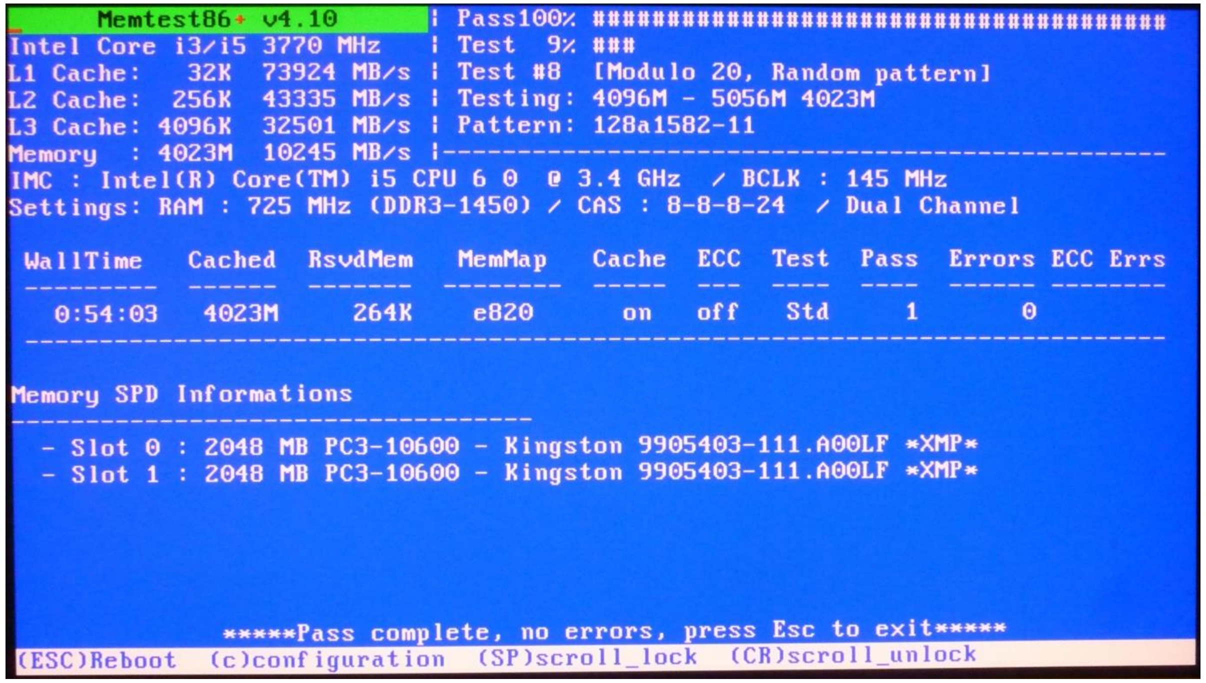
'2110

Copyright Notice

Overview

Reflect on the following question:

- You may have heard of cache memory when you come across processor specification. So what is cache memory?



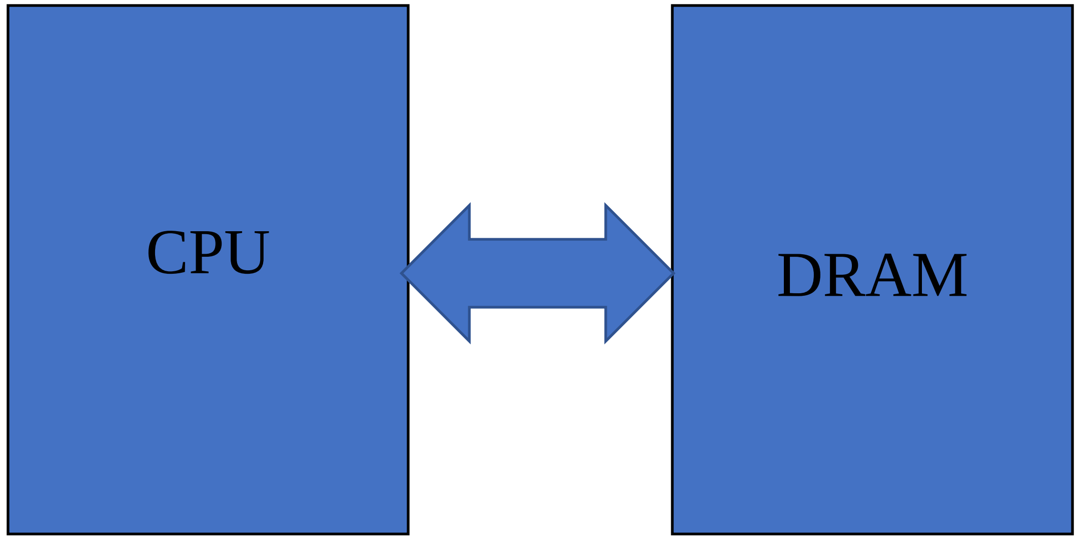
```
      Memtest86+ v4.10         | Pass100% ##########################################
Intel Core i3/i5 3770 MHz      | Test   9% ###
L1 Cache:    32K  73924 MB/s   | Test #8  [Modulo 20, Random pattern]
L2 Cache:   256K  43335 MB/s   | Testing: 4096M - 5056M 4023M
L3 Cache: 4096K  32501 MB/s    | Pattern: 128a1582-11
Memory  : 4023M  10245 MB/s    |------------------------------------------------
IMC : Intel(R) Core(TM) i5 CPU 6 0  @ 3.4 GHz  / BCLK : 145 MHz
Settings: RAM : 725 MHz (DDR3-1450) / CAS : 8-8-8-24  / Dual Channel

 WallTime   Cached  RsvdMem   MemMap   Cache  ECC  Test  Pass  Errors ECC Errs
 ---------  ------  -------  --------  -----  ---  ----  ----  ------ --------
   0:54:03   4023M     264K     e820      on   off   Std     1       0
 -----------------------------------------------------------------------------

Memory SPD Informations
 ----------------------------------
  - Slot 0 : 2048 MB PC3-10600 - Kingston 9905403-111.A00LF *XMP*
  - Slot 1 : 2048 MB PC3-10600 - Kingston 9905403-111.A00LF *XMP*



              *****Pass complete, no errors, press Esc to exit*****
(ESC)Reboot  (c)configuration  (SP)scroll_lock  (CR)scroll_unlock
```

Source: Lklundin via wikipedia (CC BY-SA 4.0)

Overview

- This sub-module introduces the concept of cache memory
- The objective is as follows:
  - ✓Describe the basic concept of cache memory

Memory Hierarchy



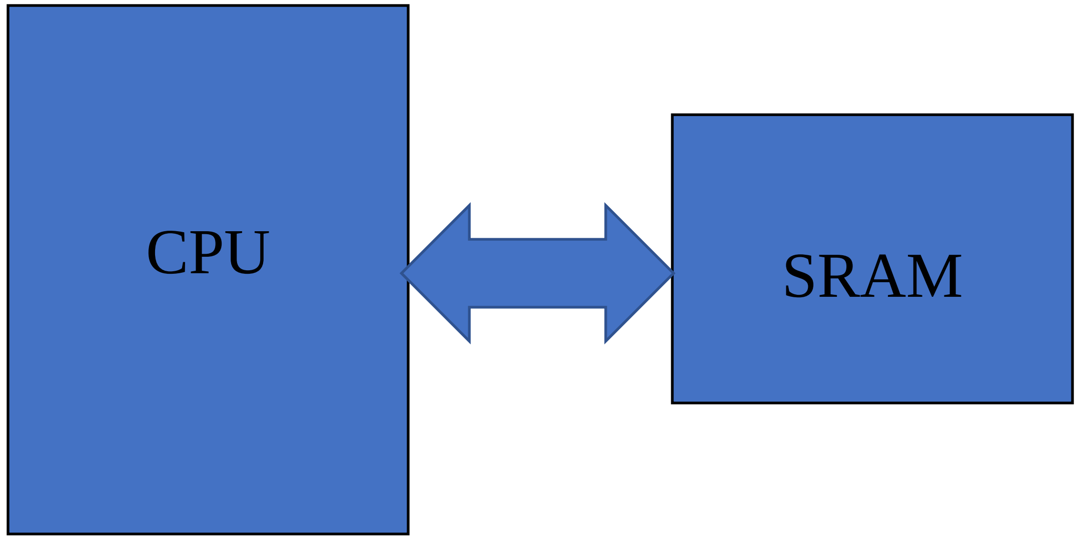
Technology	Usage	Cost/ GiB (2012)	Typical access time
Static RAM (SRAM)	Cache memory	$500-1000	0.5-2.5ns
Dynamic RAM (DRAM)	Main memory	$10-20	50-70ns
Flash semiconductor	SSD/Flash drive Secondary storage	$0.75-$1.00	5,000-50,000ns
Magnetic disk	Secondary storage	$0.05-$0.10	5,000,000ns-20,000,000ns

Processor speed: 2GHz
~ 5ns per clock cycle

DRAM (memory):
~ 50-70ns per clock cycle

Slow compare to processor!

Memory Hierarchy



Processor speed: 2GHz
~ 5ns per clock cycle

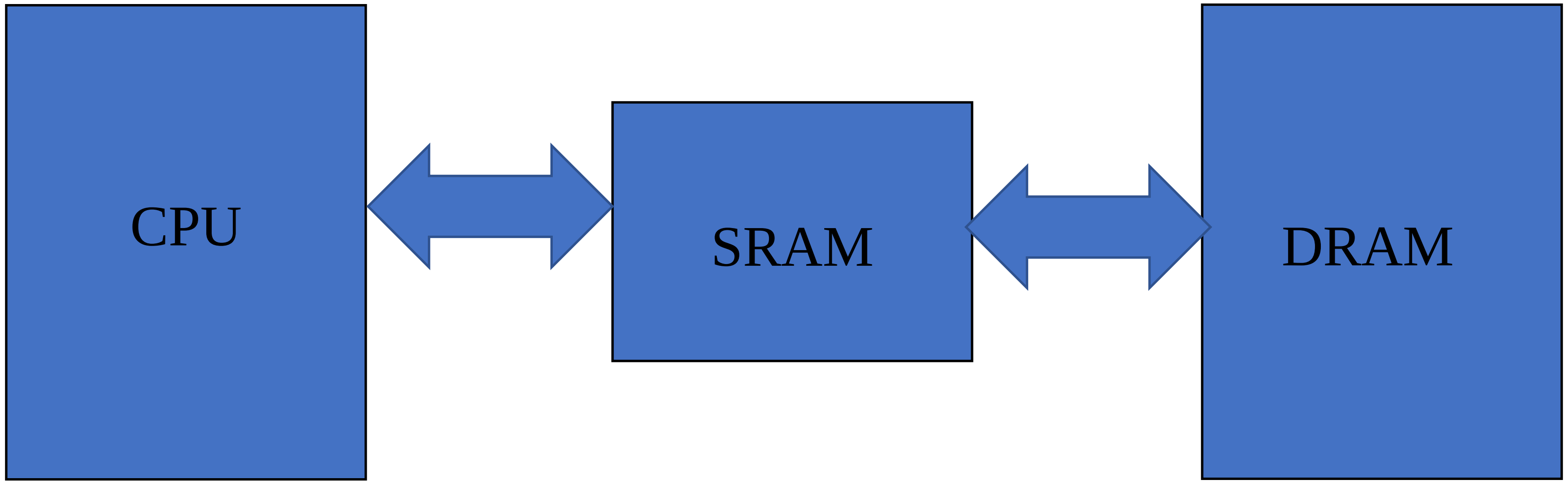
SRAM (cache memory):
~ 0.5-2.5 per clock cycle

fast but very expensive!

Technology	Usage	Cost/ GiB (2012)	Typical access time
Static RAM (SRAM)	Cache memory	\$500-1000	0.5-2.5ns
Dynamic RAM (DRAM)	Main memory	\$10-20	50-70ns
Flash semiconductor	SSD/Flash drive Secondary storage	\$0.75-\$1.00	5,000-50,000ns
Magnetic disk	Secondary storage	\$0.05-\$0.10	5,000,000ns-20,000,000ns

Memory Hierarchy

- Memory hierarchy – a structure that uses multiple levels of memories; as the distance from the processor increases, the size of the memories and the access time both increase



Cache Memory

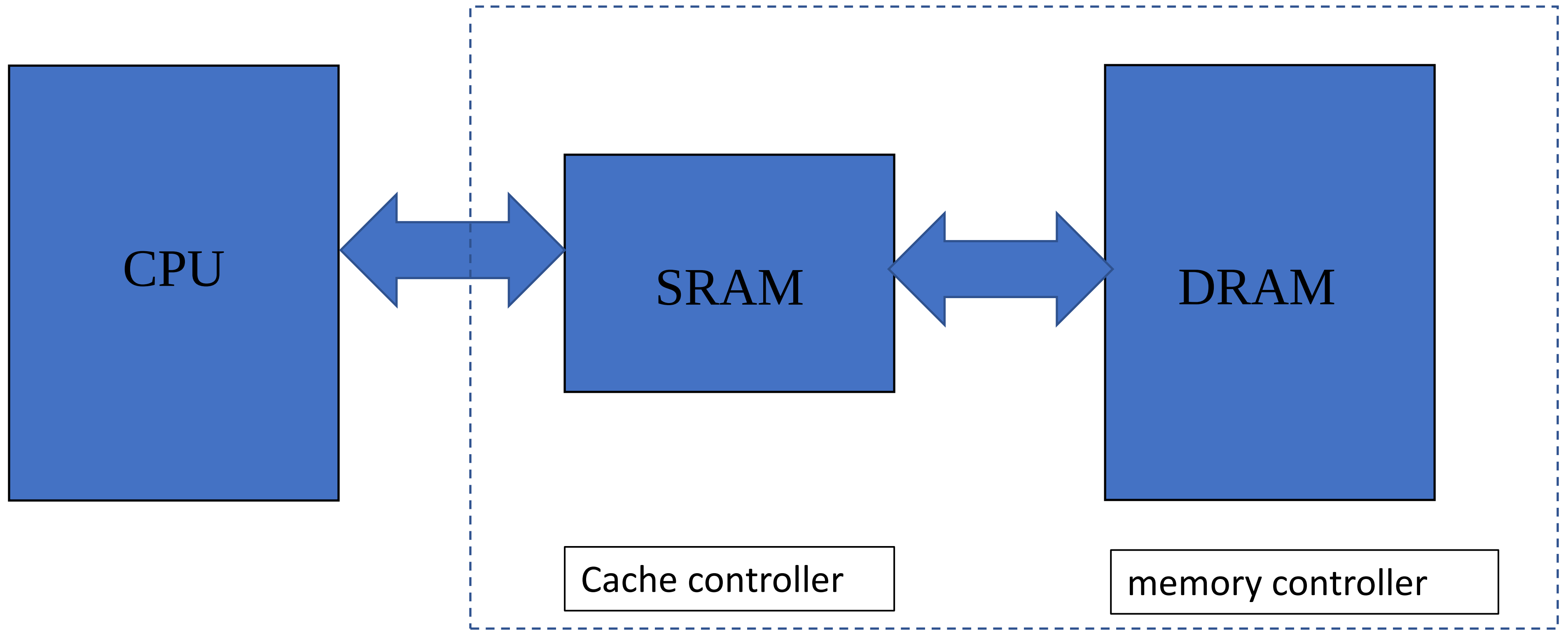
- One trade-off encountered in the design of the main memory is between cost/size and speed.
- SRAM are faster than DRAM but much more expensive and bulkier.
- Typically, the cheaper and denser DRAM is used in main memory.
- Flash memory is used as secondary memory for personal mobile devices
- Magnetic disk is used as secondary memory

Cache Memory

	Library analogy	Instruction and Data
Temporal locality(locality in time): if the item is referenced, it will tend to be referenced again soon.	Get a book from the shelf to the desk to look at, you will probably need to look at it again soon	Most programs contain loops, so instruction and data are like to be accessed repeatedly
Spatial locality (locality in space): if an item is referenced, items whose addresses are closed by will tend to be referenced soon.	Get not only the book on a related topic but also other books of similar topics. In the library the related book are shelved together	Since instructions are normally accessed sequentially thus has high spatial locality Sequential accesses to elements of an array will naturally have high degrees of spatial locality

Cache Memory

- ***Memory caching*** attempts to reduce the access time of the main memory by placing a cache memory (handled by cache controller) between the processor and the main memory.



Cache Organization

- The size of the cache memory is a lot smaller than main memory, *e.g.*, 4 GiB of main memory with 6MiB of cache $\Rightarrow$ <1% only!
- ***Cache organization*** refers to how portions of main memory are located in cache memory.

Cache Organization

- The cache and main memories are divided into ***blocks*** also known as cache block or **cache line**.
- Each block is composed of 2^n contiguous memory locations
- Each cache block is assigned a ***tag*** to distinguish memory blocks contained in the cache.
- Additionally, each cache block has valid bits to determine if the cache block entry is valid or not.

Cache Organization

- Cache has 4 blocks
- Each block (known as cache block or cache line): 4 words
- Assume memory access time is 10 τ
- Assume cache access time is 1 τ
- Valid bit is initially 0 (invalid)

Valid bit	TAG	Data
0		
0		
0		
0		

Cache Memory

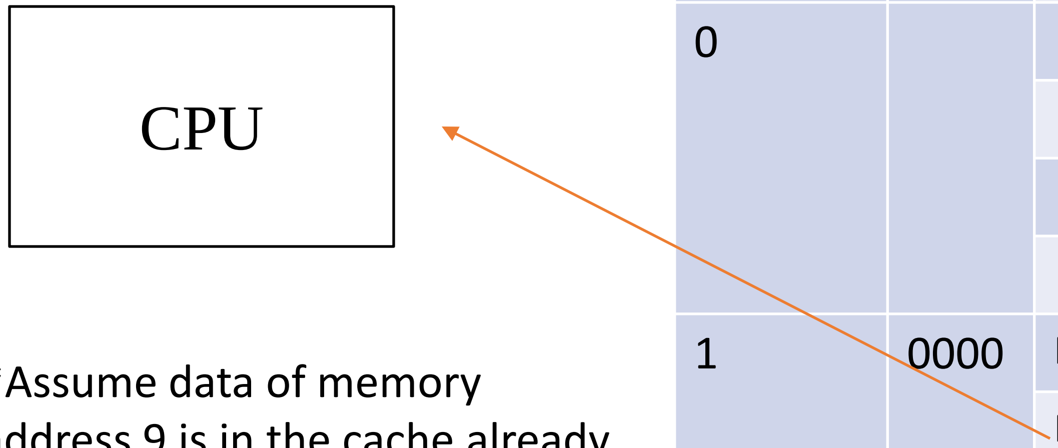
Address (binary)	Data (Hex)	Address (binary)	Data (hex)
00000000		00010000	
00000001		00010001	
00000010		00010010	
00000011		00010011	
00000100		00010100	
00000101		00010101	
00000110		00010110	
00000111		00010111	
00001000		00011000	
00001001		00011001	
00001010		00011010	
00001011		00011011	
00001100		00011100	
00001101		00011101	
00001110		00011110	
00001111		00011111	

Main Memory

Cache Hit

- ***Cache Hit*** – desired memory location has copy on cache memory
- **Hit rate** or *hit ratio* – fraction of memory accesses found in the cache memory.
- **Hit time** – the time required to access the cache memory. This includes the time needed to determine whether the access is a hit or miss.

Cache Hit (memory read)



*Assume data of memory address 9 is in the cache already.

*Situation: CPU request data from memory address 9
→ MOV AL, [0x09]

Cache Hit (read hit):
Access time is 1 τ

Valid bit	TAG	Data
0		
0		
1	0000	Memory (00001000_2) = 0x12
		Memory (00001001_2) = 0x4E
		Memory (00001010_2) = 0xCA
		Memory (00001011_2) = 0x99
0		

Cache Memory

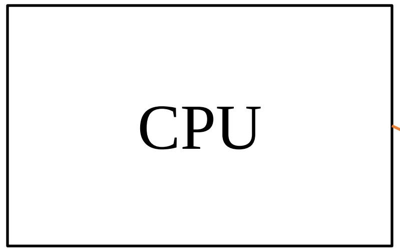
Address (binary)	Data (Hex)	Address (binary)	Data (hex)
00000000		00010000	
00000001		00010001	
00000010		00010010	
00000011		00010011	
00000100		00010100	
00000101		00010101	
00000110		00010110	
00000111		00010111	
00001000	12	00011000	
00001001	4E	00011001	
00001010	CA	00011010	
00001011	99	00011011	
00001100		00011100	
00001101		00011101	
00001110		00011110	
00001111		00011111	

Main Memory

Cache Hit

- Cache hit during memory read:
  - the information is read from the cache.
- Cache hit during memory write:
  - ***Write-Through*** or ***Store-Through*****:** Write on both the main memory and cache memory
  - ***Write-Back*** or ***Write-Behind*** **:** Write to cache and update main memory later, *e.g.*, when the processor is not performing any memory operations or when a copy of the cache block is to be removed from the cache

Cache Hit (memory write: Write-Through)



*Assume data of memory address 9 is in the cache already.

*Situation: CPU request memory write to from memory address 9
→ MOV byte ptr [0x09], 0x1F

Cache Hit :
Access time is 1 τ (write cache) + 10 τ (memory write) = 11 τ

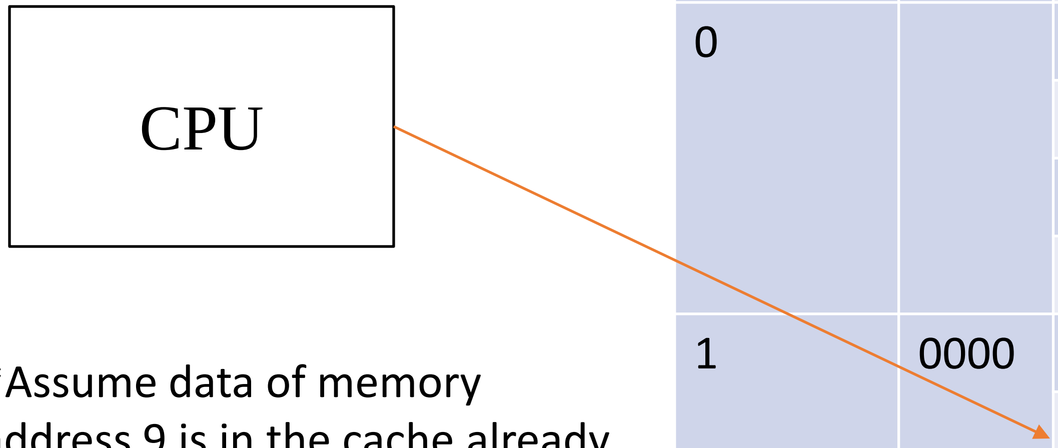
Valid bit	TAG	Data
0		
0		
1	0000	Memory (00001000_2) = 0x12
		Memory (00001001_2) = 0x~~4E~~1F
		Memory (00001010_2) = 0xCA
		Memory (00001011_2) = 0x99
0		

Cache Memory

Address (binary)	Data (Hex)	Address (binary)	Data (hex)
00000000		00010000	
00000001		00010001	
00000010		00010010	
00000011		00010011	
00000100		00010100	
00000101		00010101	
00000110		00010110	
00000111		00010111	
00001000	12	00011000	
00001001	~~4E~~1F	00011001	
00001010	CA	00011010	
00001011	99	00011011	
00001100		00011100	
00001101		00011101	
00001110		00011110	
00001111		00011111	

Main Memory

Cache Hit (memory write: Write-Back)



*Assume data of memory address 9 is in the cache already.

*Situation: CPU request memory write to memory address 9
→ MOV byte ptr [0x09], 0x1F

Cache Hit :
Access time is 1 τ (write cache)

Valid bit	TAG	Data
0		
0		
1	0000	Memory (00001000_2) = 0x12
		Memory (00001001_2) = 0x~~4E~~1F
		Memory (00001010_2) = 0xCA
		Memory (00001011_2) = 0x99
0		

Cache Memory

Address (binary)	Data (Hex)	Address (binary)	Data (hex)
00000000		00010000	
00000001		00010001	
00000010		00010010	
00000011		00010011	
00000100		00010100	
00000101		00010101	
00000110		00010110	
00000111		00010111	
00001000	12	00011000	
00001001	4E	00011001	
00001010	CA	00011010	
00001011	99	00011011	
00001100		00011100	
00001101		00011101	
00001110		00011110	
00001111		00011111	

Main Memory

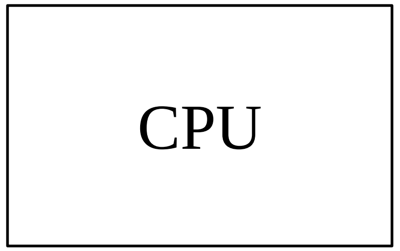
Cache Miss

- ***Cache Miss*** – desired location has no copy on cache memory
- ***Miss rate*** (1-hit rate): fraction of memory accesses not found in the cache
- ***Miss penalty*** – the time required to fetch a block (i.e., *cache line*) from the main memory into the cache memory. This includes the time to access the block, transmit it from main memory to cache, insert it into the cache and then pass the block to the requestor.

Cache Miss

- Cache miss during memory read:
  - ***No Load-through***: the information is read from the main memory and a ***block*** or ***cache line*** is copied from the main memory onto the cache (***cache line fill***). The data is then transferred from the cache to the CPU
  - ***Load-through*****:** The desired information can optionally be sent to the processor prior to completion of the cache line fill.

Cache Miss (memory read) (no Load-Through)



*Assume data of memory address 16 is NOT in the cache

*Situation: CPU request data from memory address 16
→ MOV AL, [0x10]

Cache Miss (read miss- no LT): miss penalty is 1 τ (initial cache check) + 40 τ (transfer cache line)+ τ (cache read) = 42 τ

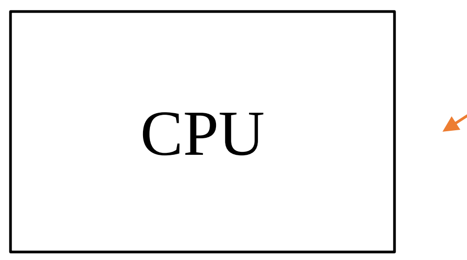
Valid bit	TAG	Data
1	0001	Memory (00010000_2) = 0x78
		Memory (00010001_2) = 0x56
		Memory (00010010_2) = 0x34
		Memory (00010011_2) = 0x12
0		
1	0000	Memory (00001000_2) = 0x12
		Memory (00001001_2) = 0x4E
		Memory (00001010_2) = 0xCA
		Memory (00001011_2) = 0x99
0		

Cache Memory

Address (binary)	Data (Hex)	Address (binary)	Data (hex)
00000000		00010000	78
00000001		00010001	56
00000010		00010010	34
00000011		00010011	12
00000100		00010100	
00000101		00010101	
00000110		00010110	
00000111		00010111	
00001000	12	00011000	
00001001	4E	00011001	
00001010	CA	00011010	
00001011	99	00011011	
00001100		00011100	
00001101		00011101	
00001110		00011110	
00001111		00011111	

Main Memory

Cache Miss (memory read) (Load Through)



*Assume data of memory address 16 is NOT in the cache

*Situation: CPU request data from memory address 16
→ MOV AL, [0x10]

Cache Miss (read miss – LT): miss penalty is 1 τ (initial cache check) + 10 τ (transfer address 0x16 from cache and CPU) = 11 τ

Valid bit	TAG	Data
1	0001	Memory (00010000_2) = 0x78
		Memory (00010001_2) = 0x56
		Memory (00010010_2) = 0x34
		Memory (00010011_2) = 0x12
0		
1	0000	Memory (00001000_2) = 0x12
		Memory (00001001_2) = 0x4E
		Memory (00001010_2) = 0xCA
		Memory (00001011_2) = 0x99
0		

Cache Memory

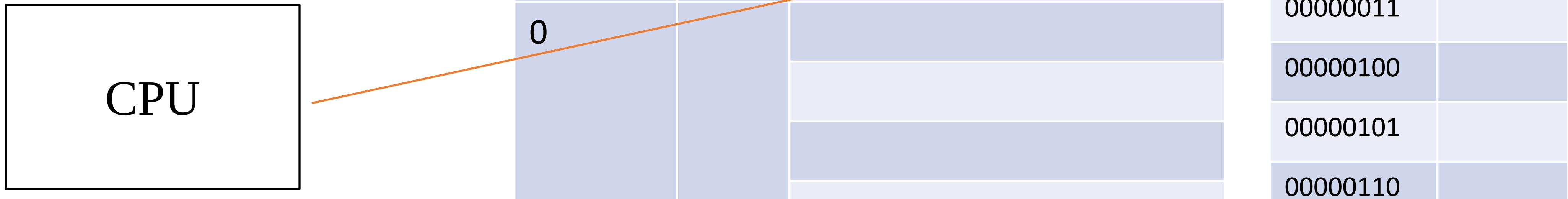
Address (binary)	Data (Hex)	Address (binary)	Data (hex)
00000000		00010000	78
00000001		00010001	56
00000010		00010010	34
00000011		00010011	12
00000100		00010100	
00000101		00010101	
00000110		00010110	
00000111		00010111	
00001000	12	00011000	
00001001	4E	00011001	
00001010	CA	00011010	
00001011	99	00011011	
00001100		00011100	
00001101		00011101	
00001110		00011110	
00001111		00011111	

Main Memory

Cache Miss

- Cache miss during memory write:
  - ***No write-allocate*** (also called *write around*): data at the missed-write location is not loaded to cache, and is written directly to the main memory
  - ***Write allocate*** (also called *fetch on write*): data at the missed-write location is loaded to cache, followed by a write-hit operation.

Cache Miss (memory write) (No write-allocate)



*Assume data of memory address 16 is NOT in the cache

*Situation: CPU request memory write to memory address 16
→ MOV byte ptr [0x10], AL

Cache Miss (write miss – NO WA): miss penalty is 1 τ (initial cache check) + 10 τ (write data to MM) = 11 τ

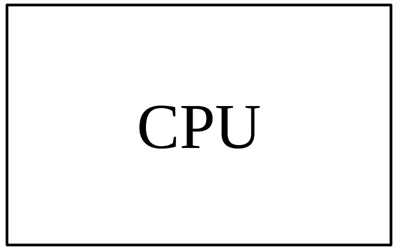
Valid bit	TAG	Data
0		
0		
1	0000	Memory (00001000_2) = 0x12
		Memory (00001001_2) = 0x4E
		Memory (00001010_2) = 0xCA
		Memory (00001011_2) = 0x99
0		

Cache Memory

Address (binary)	Data (Hex)	Address (binary)	Data (hex)
00000000		00010000	~~78~~ 45
00000001		00010001	56
00000010		00010010	34
00000011		00010011	12
00000100		00010100	
00000101		00010101	
00000110		00010110	
00000111		00010111	
00001000	12	00011000	
00001001	4E	00011001	
00001010	CA	00011010	
00001011	99	00011011	
00001100		00011100	
00001101		00011101	
00001110		00011110	
00001111		00011111	

Main Memory

Cache Miss (memory write) (Write Allocate)



Valid bit	TAG	Data
1	0001	Memory (00010000_2) = 0x45
		Memory (00010001_2) = 0x56
		Memory (00010010_2) = 0x34
		Memory (00010011_2) = 0x12
0		
1	0000	Memory (00001000_2) = 0x12
		Memory (00001001_2) = 0x4E
		Memory (00001010_2) = 0xCA
		Memory (00001011_2) = 0x99
0		

Cache Memory

Address (binary)	Data (Hex)	Address (binary)	Data (hex)
00000000		00010000	45
00000001		00010001	56
00000010		00010010	34
00000011		00010011	12
00000100		00010100	
00000101		00010101	
00000110		00010110	
00000111		00010111	
00001000	12	00011000	
00001001	4E	00011001	
00001010	CA	00011010	
00001011	99	00011011	
00001100		00011100	
00001101		00011101	
00001110		00011110	
00001111		00011111	

Main Memory

*Assume data of memory address 16 is NOT in the cache

*Situation: CPU request data from memory address 16
→ MOV AL, [0x10]

Cache Miss (write miss- WA): miss penalty is 1 τ (initial cache check) + 40 τ (transfer cache line)+ τ (cache write) + 10 τ (memory write) = 52 τ

Cache Miss

- Average memory access time experience by the processor is:

$$T_{avg} = hC + (1-h)*M$$

Where:

h is the hit rate

C is cache access time

M is the miss penalty

Cache Miss

Example: Consider a computer that has the following parameters. Access times to the cache and the main memory are τ and 10τ , respectively. When a cache miss occurs, a block of 8 words is transferred from the main memory to the cache. It takes 10τ to transfer a word from memory to cache. The miss penalty also includes a delay of τ for the initial access to the cache, which misses, and another delay of τ to transfer the word to the processor after the block is loaded into the cache (assuming no load-through).

Assume that 30 percent of the instructions in a typical program perform a Read or a Write operation. Assume that there are 100 instructions in a program. Assume that the hit rates in the cache are 0.95 for instructions and 0.9 for data. Assume further that the miss penalty is the same for both read and write accesses. Compute for (a) time to fetch without cache and (b) time to fetch with cache

Cache Miss

Assume that 30 percent of the instructions in a typical program perform a Read or a Write operation. Assume that there are 100 instructions in a program. Assume that the hit rates in the cache are 0.95 for instructions and 0.9 for data. Assume further that the miss penalty is the same for both read and write accesses. Compute for (a) time without cache and (b) time with cache

Solution:

100 instructions and 30% of the instructions perform memory operation = total 130 (100 instruction fetch + 30 data fetch) memory operations

Time w/o cache = $130 * 10\tau = 1300\tau$

Cache Miss

Assume that 30 percent of the instructions in a typical program perform a Read or a Write operation. Assume that there are 100 instructions in a program. Assume that the hit rates in the cache are 0.95 for instructions and 0.9 for data. Assume further that the miss penalty is the same for both read and write accesses. Compute for (a) time without cache and (b) time with cache

Solution:

Miss penalty = $\tau + 80\tau + \tau = 82\tau$

Time with cache = $100(0.95\tau + 0.05*82\tau) + 30(0.9\tau + 0.1*82\ \tau)$

$= 100(5.05\tau) + 30(9.1\tau)$

$= 778\ \tau$

To recall …

- What have we learned:
  - ✓Describe the basic concept of cache memory