

21. Cache Problems

Assignment Project Exam Help

EECS 370 – Introduction to Computer Organization – Fall 2020

<https://powcoder.com>

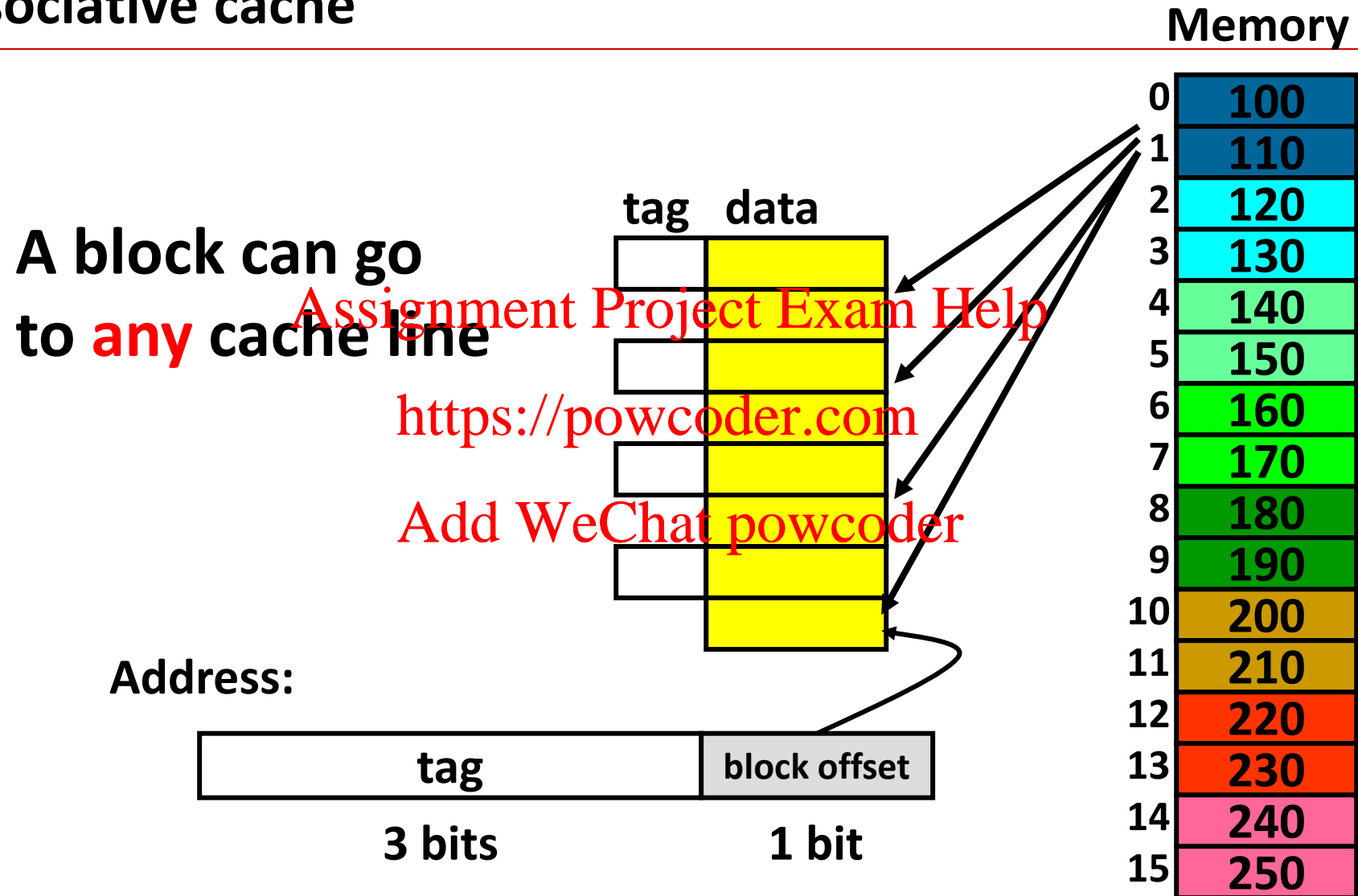
Satish Narayanasamy
Add WeChat powcoder

EECS Department
University of Michigan in Ann Arbor, USA

© Narayanasamy 2020

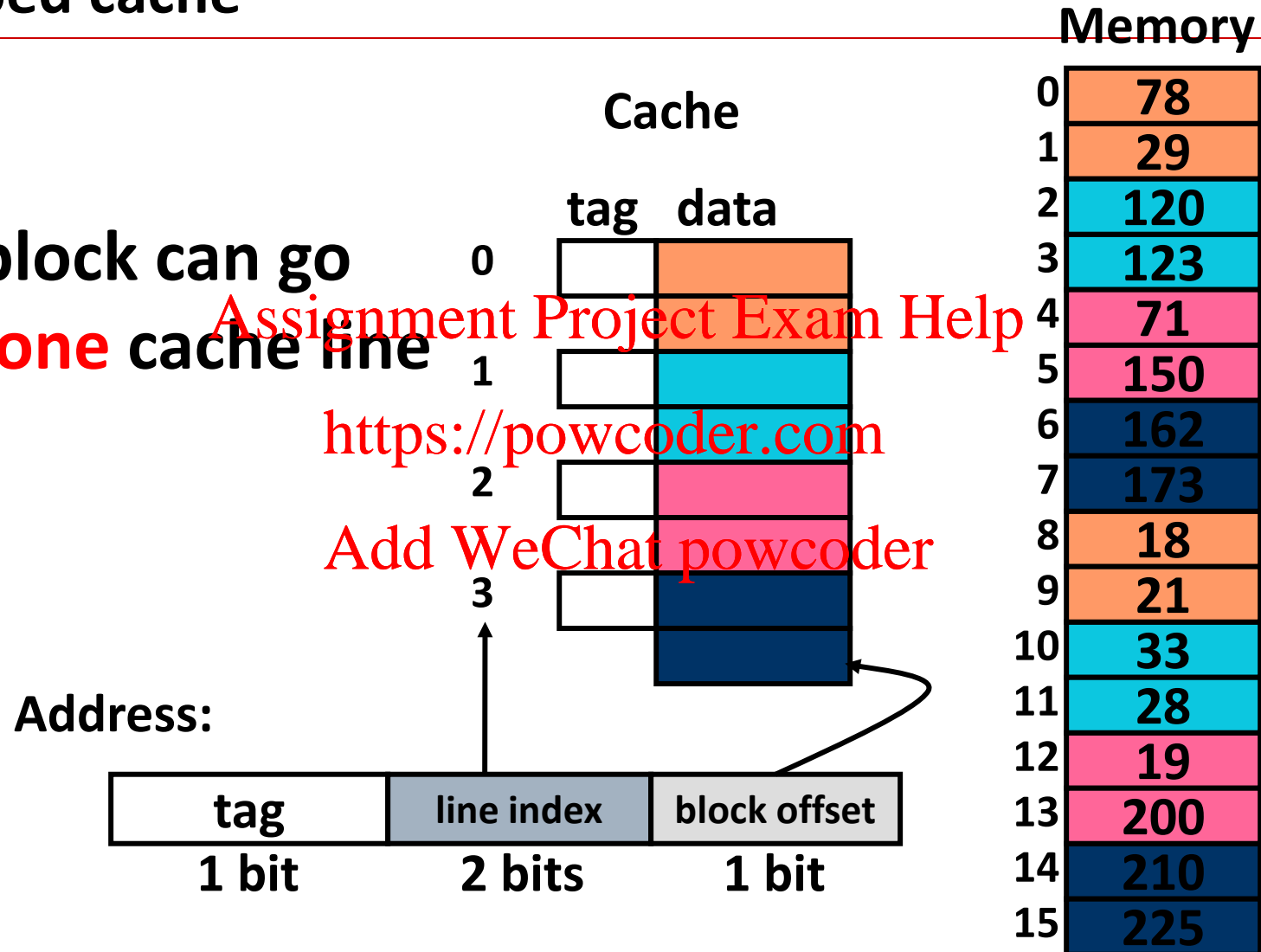
The material in this presentation cannot be
copied in any form without written permission

Fully-associative cache



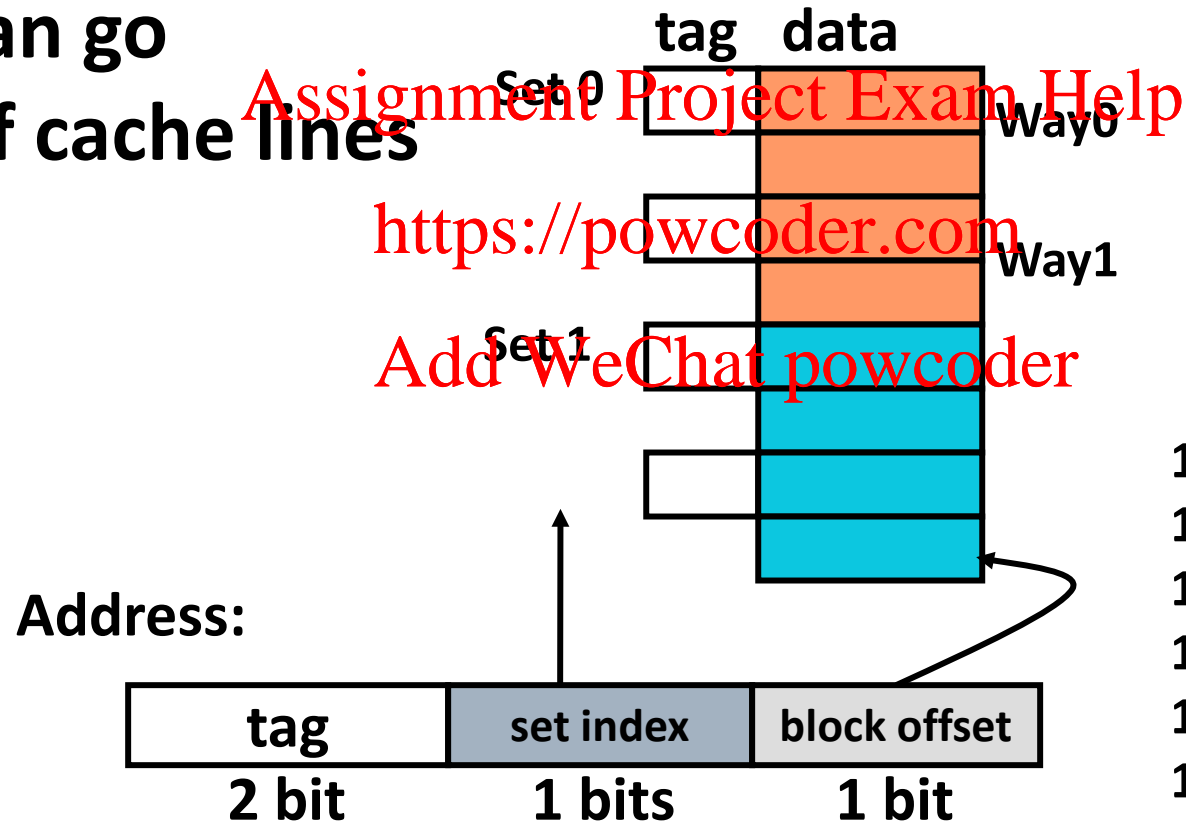
Direct-mapped cache

A block can go
to **one** cache line



Set-associative cache

A block can go
to **a set** of cache lines



Memory

0	78
1	29
2	120
3	123
4	71
5	150
6	162
7	173
8	18
9	21
10	33
11	28
12	19
13	200
14	210
15	225

How to solve problems on caches?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Programmer's view

Address size: defined by ISA

e.g., 64-bit ISA has 64-bit addresses

Cache design defines how address is split into the following (hidden from the programmers/compiler):



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Byte vs word addressable

Byte addressable ISA: each address refers to a **byte** in memory

Word addressable ISA: each address refers to a **word** in memory.
(cannot address individual bytes in a word)

Common practice:

In 64-bit ISA, a “word” is 64-bits (8 bytes)

In 32-bit ISA, a “word” is 32-bits (4 bytes)

<https://powcoder.com>
Add WeChat powcoder

This difference impacts how we determine block offset size:

Example: Block-offset for a 256-byte block:

In 64-bit **byte** addressable ISA: a 256-byte block has **256 bytes**. So block_offset_size is 8 bits

In 64-bit **word** addressable ISA: a 256-byte block has **32 words**. So block_offset_size is 5 bits

tag	line index	block offset
-----	------------	--------------

Cache Organization: Equations

Block

$$\#blocks = \text{cache size} / \text{block_size}$$

$$\#cache\ lines = \#blocks$$

$$\text{block_offset_size} = \log_2(\text{block_size})$$

Assignment Project Exam Help

Set

$$\#sets = \frac{\#lines}{\#ways} = \frac{\#lines}{(\text{lines per set})}$$

Direct-mapped: $\#sets = \#lines / 1$

2-way associative: $\#sets = \#lines / 2$

n-way associative: $\#sets = \#lines / n$

fully-associative: $\#sets = 1$ (all lines are in 1 set)

$$\text{set_index_size} = \log_2(\#sets)$$

$$\text{Tag size} = \text{address size} - \text{set_index_size} - \text{block_offset_size}$$

Calculating Average Memory Access Time (AMAT)

If we assume memory latency includes time to cache access and determine hit/miss:

$$\text{AMAT} = \text{cache latency} \times \text{hit rate} + \text{memory latency} \times \text{miss rate}$$

Assignment Project Exam Help

<https://powcoder.com>

If we **do not** assume memory latency includes time to cache access and determine hit/miss:

Add WeChat powcoder

$$\text{AMAT} = \text{cache latency} + \text{memory latency} \times \text{miss rate}$$

3C problem

Simulate three different caches:

Infinite cache (same block size)

Fully associative cache (same block size and cache size)

Given cache

Assignment Project Exam Help

<https://powcoder.com>

How to classify a miss in the given cache?

Is it also a miss in the infinite cache?

Yes, then Compulsory miss

No. Is it a miss in the fully associative cache?

Yes, then capacity miss

No, then conflict miss

Simulating caches

Figure out fields in the address:

tag, set index, block-offset

Determine (tag, set-index) of given sequence of addresses.

<https://powcoder.com>

Sequence of (tag, set-index) is sufficient to simulate a cache. Data is not necessary.

Add WeChat powcoder

Keep track of LRU rank of cache lines within each cache set for set-associative caches.
(replacement policies other than LRU exist)

Sample problems on caches Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Problem 0: Cache Block Size

Assume 32-bit word-addressable ISA, and a fully-associative cache of size 64 KB. If the block size 32 bytes. Determine the size of block offset and tag.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Problem 0: Cache Block Size

Assume 32-bit word-addressable ISA, and a fully-associative cache of size 64 KB. If the block size 32 bytes. Determine the size of block offset and tag.

A 32-byte block contains $32 / 4 = 8$ words. That is, $32 / 4 = 8$ words.

⇒ block_offset is $\log(8) = 3$ bits. <https://powcoder.com>

Tag = 32 (address size) – 3 bits = 29 bits. Add WeChat powcoder

No set-index, because the cache is fully-associative (1 set).

Practice Problem 1: CPI with caches

The *grinder* application is run on the LC2k with full data forwarding. All branches are predicted not-taken. Assume the following.

45% R-type

20% Branches (of these, 40% of branches are taken.)

15% Loads (of these, 50% of LWs are followed by an immediate use.)

20% Stores

Cache performance:

Assume cache hit is 1 cycle

I-cache miss rate: 3%

D-cache miss rate: 6%

(assume misses do not overlap)

Main memory latency: 100 ns

Processor clock frequency: 500 MHz

What is the CPI of *grinder* on the LC2k?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Practice Problem 1: Solution

What is the CPI of *grinder* on the LC2k?

Cycle time: 2 ns per cycle (500 MHz)

Stalls per cache miss = 100 ns / 2ns = 50 cycles

CPI = 1 + data hazard stalls + control hazard stalls + icache stalls + dcache stalls

CPI = 1 + 0.15*0.50*1 + 0.20*0.40*3 + 1*0.03*50 + 0.35*0.06*50

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Practice Problem 2: CPI with caches

The 370 application is run on the LC2k. Assume the following:

30% of instructions are loads/stores

Cache :

I-cache and D-cache: 8 KB size; 2-way associative; 1-cycle access latency

I-cache hit rate: 95%

D-cache hit rate: 90%

(assume misses do not overlap)

Main memory latency: 20 cycles

Processor clock frequency: 200 MHz

Assume CPI is 1, if cache has 100% hit rate and 1 cycle latency.

continued....

Practice Problem 2: CPI w/ Caches 2

Suppose you have 2 options for the next generation processor, which do you pick?

Option 1: Double the clock frequency—assume this will increase your memory latency to 40 cycles. Also assume a base CPI of 1 can still be achieved after this change.

<https://powcoder.com>

Option 2: Double the size of your caches, this will increase the instruction cache hit rate to 98% and the data cache hit rate to 95%. Assume the hit latency is still 1 cycle.

Practice Problem 2: Solution

200 MHz → clock cycle = 5 ns.

Option 1: (double clock freq to 400 MHz, so new cycle time is 2.5 ns)

$$\text{CPI} = \text{baseCPI} + \text{IcacheStallCPI} + \text{DcacheStallCPI}$$
$$\text{CPI} = 1.0 + 0.05 * 40 + 0.3 * 0.1 * 40 = 4.2$$

$$\text{Execution time} = 4.2 * \text{NumInstrs} * 2.5\text{ns} = 10.5\text{ns} * \text{NumInstrs}$$

Add WeChat powcoder

Option 2 (icache/dcache miss rates lowered to 2% and 5%)

$$\text{CPI} = \text{baseCPI} + \text{IcacheStallCPI} + \text{DcacheStallCPI}$$
$$\text{CPI} = 1.0 + 0.02 * 20 + 0.3 * 0.05 * 20 = 1.7$$

$$\text{Execution time} = 1.7 * \text{NumInstrs} * 5\text{ns} = 8.5\text{ns} * \text{NumInstrs}$$

Therefore, Option 2 is the better choice

Practice Problem 3: Write-back vs Write-through

Say you have the following:

A program that generates 2 Billion loads and 1 Billion stores, each 4 bytes in size.

A cache with a 32-byte block which gets a 95% hit rate on that program.

Assignment Project Exam Help

How many bytes of memory would be read and written if:

1. We had no cache?

<https://powcoder.com>

2. We had a write-through cache with a no-write allocate policy?

Add WeChat powcoder

3. We had a write-back cache with a write-allocate policy? (Assume 25% of all misses result in a dirty eviction)

Practice Problem 3: Solution (1/3)

No-cache

All stores go to memory and are 4 bytes each

Writes: $1 \text{ billion stores} * 4 \text{ bytes} = 4 \text{ billion bytes}$

All loads go to memory and are 4 bytes each.

Reads: $2 \text{ billion loads} * 4 \text{ bytes} = 8 \text{ billion bytes}$

Assignment Project Exam Help

<https://powcoder.com>

Cache: Write-through, no write-allocate

All stores still go to memory. Cache block not read from memory, due to no write-allocate

Writes: $1 \text{ billion stores} * 4 \text{ bytes} = 4 \text{ billion bytes}$

Only loads that miss in the cache go to memory. But they read the full cache block.

Reads: $2 \text{ billion loads} * 0.05 * 32 \text{ bytes} = 3.2 \text{ billion bytes}$

Add WeChat powcoder

Practice Problem 3: Solution (2/3)

Cache: Write-back, write-allocate

Total memory accesses: 1 billion stores + 2 billion loads = 3 billion memory accesses

Data read from memory

Store (due to write-allocate policy) and *load* misses result in a cache block being read.

Data read from memory: $3 \text{ billion memory accesses} * 0.05 * 32 \text{ bytes} = 4.8 \text{ billion bytes}$

Data written to memory

Store and load cache misses result in a cache block getting evicted.

Given: 25% of cache misses result in dirty eviction (write 32-byte block to memory).

Data written to memory = $3 \text{ billion memory accesses} * 0.05 * 0.25 * 32 \text{ bytes} = 1.2 \text{ billion bytes}$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Practice Problem 3: Solution (3/3)

Write-back, write-allocate (data writes)

When do we write to memory?

Only on dirty evictions. Can be done by both loads and stores.

Recall we are assuming 25% of all evictions are of dirty data.

<https://powcoder.com>

Total memory accesses: 1 billion stores + 2 billion loads = 3 billion memory accesses

25% of cache misses result in dirty eviction (write 32-byte block to memory).

3 billion memory accesses * 0.05 * 0.25 * 32 bytes = 1.2 billion bytes

Practice Problem 4: Reverse engineer the cache!

Here is the series of address references (in hex) to a cache of size 512 bytes. You are asked to **determine the configuration of the cache**. Assume 12-bit addresses.

0x310 – Miss

0x30f – Miss

0x510 – Miss

0x31f – Hit

0x72d – Miss

0x72f – Hit

0x320 – Miss

0x520 – Miss

0x720 - Miss

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Block size: ?

Associativity: ?

Number of sets: ?

Practice Problem 4: Solution

Here is the series of address references (in hex) to a cache of size 512 bytes. You are asked to **determine the configuration of the cache**. Assume 12-bit addresses

0x310 – Miss ←

0x30f – Miss ←

0x510 – Miss

0x31f – Hit ←

0x72d – Miss

0x72f – Hit

0x320 – Miss

0x520 – Miss

0x720 – Miss

Assignment Project Exam Help

Determine block size

<https://powcoder.com>

Add WeChat powcoder

Practice Problem 4: Solution

Here is the series of address references (in hex) to a cache of size 512 bytes. You are asked to **determine the configuration of the cache**. Assume 12-bit addresses

0x310 – Miss ←

0x30f – Miss ←

0x510 – Miss

0x31f – Hit ←

0x72d – Miss

0x72f – Hit

0x320 – Miss

0x520 – Miss

0x720 – Miss

Assignment Project Exam Help

Determine block size

<https://powcoder.com>

First hit must be brought in by another miss

Add WeChat powcoder

Take closest address: 0x310, so know block size must be at least 16 bytes so 0x31f brought in when 0x310 miss occurs

Now, is the block size larger? Know that 0x30f was a miss, thus 0x310 and 0x30f not in the same block.

Thus, block size must be ≤ 16 bytes

Thus Block Size = 16 bytes


Practice Problem 4 : Solution (2)

Here is the series of address references (in hex) to a cache of size 512 bytes. You are asked to **determine the configuration of the cache**. Assume 12-bit addresses Determine associativity


Assignment Project Exam Help


<https://powcoder.com>

Add WeChat powcoder


0x310 – Miss 


0x30f – Miss


0x510 – Miss 

0x31f – Hit 

0x72d – Miss

0x72f – Hit 

0x320 – Miss 

0x520 – Miss 

0x720 – Miss 

Practice Problem 4 : Solution (2)

Here is the series of address references (in hex) to a cache of size 512 bytes. You are asked to **determine the configuration of the cache**. Assume 12-bit addresses

Determine associativity

Assignment Project Exam Help

Assume direct mapped: 3-bit tag, 5-bit index, 4-bit offset.

If direct mapped 0x310 and 0x510 would both map to index 9,

this 0x31f could not be a hit. So, not direct mapped.


Assume 2-way associative: 4-bit tag, 4-bit index, 4-bit offset

This fixes the green accesses, and allows 0x31f to be a hit.


What about > 2-way associative?


Now we also know that 0x720 is a miss even though 3 accesses earlier 0x72f was a hit, and thus it is in the cache. The intervening 2 accesses must kick it out, 0x320 and 0x520. Both go to set 2. If the associativity was > 2, then 0x720 would be a hit. So, must conclude that cache is 2-way associative.

Lastly, number of sets = $512 / (2 * 16) = 16$


0x310 – Miss 


0x30f – Miss


0x510 – Miss 

0x31f – Hit 

0x72d – Miss

0x72f – Hit 

0x320 – Miss 

0x520 – Miss 

0x720 – Miss 