Data cache & Locality

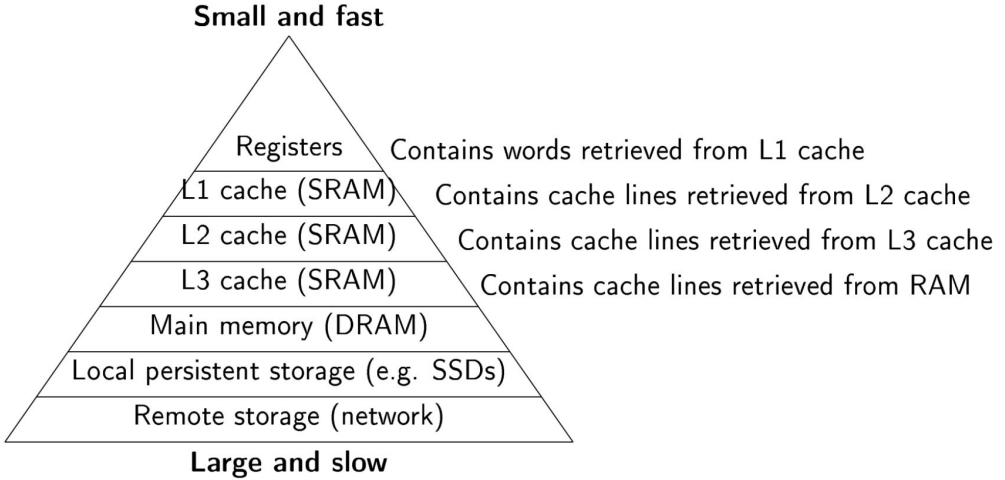
Based on slides from the lectures
Lea

Content

- Theory (just a little bit)
- Locality
  - Example (only with C code)
- Cache task
  - How to read the task
  - Formulas
  - Unit conversion
- Walkthrough of cache task from exam 2023-2024

The memory hierarchy

- **Cache definition:** A smaller, faster storage device that acts as a staging area for the subset of the data in a larger, slower device.
- **Idea:** The smaller and faster device at level *k* acts as a cache for the larger slower device at level *k+1*



Found on slide 16 in “Memory hierarchy and Caching”

Array layouts (row- and column-major order)

- **Row-major order:** Is used in C

- **Column major order:** Is used in MatLab

Example:

```
A[3][3] =
{{1,2,3}, {4,5,6}, {7,8,9}}
```

Row-major order:

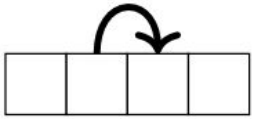
Row 1			Row 2			Row 3		
1	2	3	4	5	6	7	8	9

Column-major order:

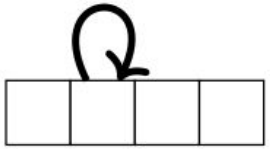
Column 1			Column 2			Column 3		
1	4	7	2	5	8	3	6	9

Locality

- **Spatial locality:** Accessing data that is close to the data that was recently accessed.



- **Temporal locality:** Accessing data that was recently accessed.



- Generally, code that is written normally has good locality by default, so the focus tends to be only on data locality

Locality, exercise 5.1, COD from 23/09-2024

Exercise 5.1.2: Which variable references exhibit temporal locality?

Answer: i, j, B[i][0]

Why? : Since temporal locality means to access data at the same location again in the near future, we can see that **i, j** and **B[i][0]** uses temporal locality.

i uses temporal locality because it remains constant each time loop **j** runs.

j uses temporal locality since we access **j** a lot at each iteration of the inner loop.

B[i][0] uses temporal locality, because we access this 8000 times, since we run the whole **j** loop for each iteration of the **i** loop.

```
for (I = 0; I < 8; I++) {
  for J = 0; J < 8000; J++) {
    A[I][J] = B[I][0] + A[I][J];
  }
}
```

Note: This code is written in C, which means that it uses row major order.

Locality, exercise 5.1, COD from 23/09-2024 (continue)

Exercise 5.1.3: Which variable references exhibit spatial locality?

Answer: A[i][j]

Why? : Spatial locality is when we will access data at a nearby location in the near future. Here, the spatial locality is in A[i][j], because as **j** increases in the loop, and the code accesses A[i][0], A[i][1], ... and so on to A[i][j], the elements are then accessed consecutively in the same row for A.

```
for (I = 0; I < 8; I++) {
  for J = 0; J < 8000; J++) {
    A[I][J] = B[I][0] + A[I][J];
  }
}
```

Note: This code is written in C, which means that it uses row major order.

Formulas (Address translation)

- Order in bits (left to right) = Tag, Set index, Block offset
- Offset bits (b) : how many bits to represent a block

$$b = log_2(block\_size\_in\_bytes)$$

- How many sets are in n-way associative cache?

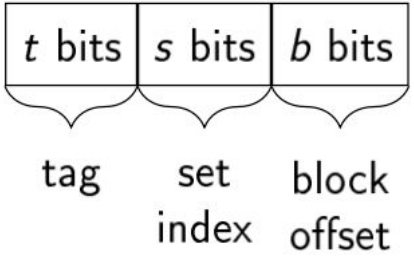
$$set\_count = \frac{cachesize\_in\_bytes}{n*blocksize\_in\_bytes}$$

- Set index in bits (s) : How many bits needed to represent the sets

$$s = log_2(set\_count)$$

- Tag: the rest

$$t = bit\_in\_address - (s + b)$$



From Memory Hierarchy, slide 39

Unit conversion

Memory is byte-addressed, so it is a good idea to convert everything to bytes.

Units:

- Kibibyte (KiB) = 1024 bytes
- Mebibyte (MiB) = 1024 KiB
- kb = kbit = kilobit = 125 bytes
- To find how many bits it takes to represent something:
  - log_2(n)

How to read the cache task

NB: Read the text carefully!!

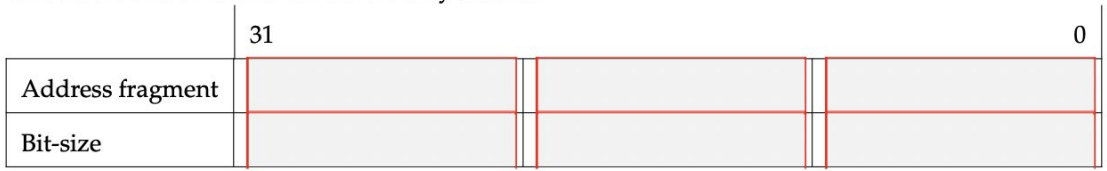
1.1 Data Cache and Locality (about 8 %)

A byte-addressed machine with 32-bit addresses is equipped with a 8 kibibyte 2-way set-associative cache with a block size of 16 bytes. The cache uses "least-recently-used replacement policy".

Question 1.1.1: The address is separated into the following fragments when the cache is accessed

- block offset,
- cache tag, and
- set index.

What is bit-size of each and how are they ordered?



The task usually looks like so

How to read the cache task

NB: Read the text carefully!!

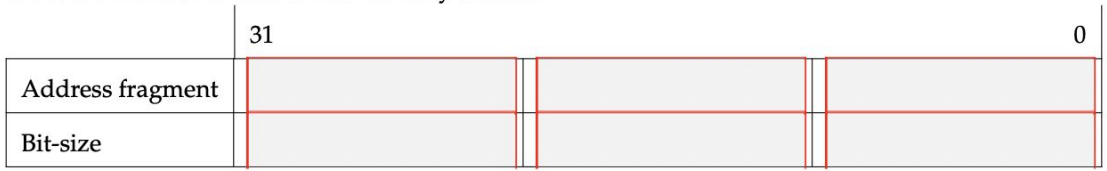
1.1 Data Cache and Locality (about 8 %)

A byte-addressed machine with 32-bit addresses is equipped with a 8 kibibyte 2-way set-associative cache with a block size of 16 bytes. The cache uses "least-recently-used replacement policy".

Question 1.1.1: The address is separated into the following fragments when the cache is accessed

- block offset,
- cache tag, and
- set index.

What is bit-size of each and how are they ordered?



Cache size = 8KIB

n = 2

Block size = 16 bytes

LRU (useful later)

Exam 2023-2024, question 1.1.1

1.1 Data Cache and Locality (about 8 %)

A byte-addressed machine with 32-bit addresses is equipped with a 8 kibibyte 2-way set-associative cache with a block size of 16 bytes. The cache uses "least-recently-used replacement policy".

Question 1.1.1: The address is separated into the following fragments when the cache is accessed

- block offset,
- cache tag, and
- set index.

What is bit-size of each and how are they ordered?

	31		0
Address fragment	Cache Tag	Set index	Block offset
Bit-size	20	8	4

Now, lets calculate using the formulas from earlier!

$$b = log_2(block\_size\_in\_bytes) = log_2(16) = 4$$

$$cachesize\_in\_bytes = 8KIB * 1024 = 8192$$

$$set\_count = \frac{cachesize\_in\_bytes}{n*blocksize\_in\_bytes} = \frac{8192}{2*16} = 256$$

$$s = log_2(set\_count) = log_2(256) = 8$$

$$t = bit\_in\_address - (s + b) = 32 - (8 + 4) = 20$$

Exam 2023/2024, question 1.1.2

Question 1.1.2: Calculate for the following addresses the value of block offset, cache tag, and set index.
Address: 0x76543210

Now, from the other question, we know the block offset, cache tag and set index, which makes this question straight forward

Rest = Tag = 0x76543

Block offset = 0x0

0x76543210 = 0111 0110 0101 0100 0011 0010 0001 0000

Set index = 0x21

Exam 2023/2024, question 1.1.3

Question 1.1.3: On the machine, the following stream of cache accesses are performed (read from top to bottom). Indicate for each of the address references:

- the set index,
- if the cache access is a miss or a hit, and
- the cache tags in LRU-order that are in the affected set after the access.

Was also mentioned earlier

Addresses are given in hexa-decimal notation. Assume the cache is cold on entry.

LRU (Least Recently Used): The block that is replaced in the set, is the one that has been unused for the longest

With LRU:

- First, find set index of the address
- Search for the block in the set by comparing tags
- If found = cache hit, and update the block so that it is visible that it was just used
- Not found = cache miss,
  - If one or more blocks are not valid (check valid bit), replace one of these with the new block
  - If all blocks in the set are valid, replace the least recently used block, und update the new block so that it is visible that it was just used

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010			
0x0010008			
0x0110004			
0x0000008			
0x0300004			
0x0300010			
0x0110008			
0x0410004			
0x0A00008			
0x0110004			
0x0A00000			
0x0A00004			
0x0300010			

Remember from before:

- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Lets start!

First, translate the first address to bits:

0x0100004 =

0000 0000 0001 0000 0000 | 0000 0000 | 0100

Tag = 0x100

Set index = 0x0

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008			
0x0110004			
0x0000008			
0x0300004			
0x0300010			
0x0110008			
0x0410004			
0x0A00008			
0x0110004			
0x0A00000			
0x0A00004			
0x0300010			

Remember from before:

- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Next:

0x0100010 =
0000 0000 0001 0000 0000 0000 0001 0000

Tag = 0x100

Set index = 0x1

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008	0x0	Miss	0x10, 0x100
0x0110004			
0x0000008			
0x0300004			
0x0300010			
0x0110008			
0x0410004			
0x0A00008			
0x0110004			
0x0A00000			
0x0A00004			
0x0300010			

Remember from before:

- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Next:

0x0010008 =
0000 0000 0000 0001 0000 | 0000 0000 | 1000

Tag = 0x10

Set index = 0x0

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008	0x0	Miss	0x10, 0x100
0x0110004	0x0	Miss	0x110, 0x10
0x0000008			
0x0300004			
0x0300010			
0x0110008			
0x0410004			
0x0A00008			
0x0110004			
0x0A00000			
0x0A00004			
0x0300010			

Remember from before:
- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Next:

0x0110004 =
0000 0000 0001 0001 0000 0000 0000 0100

Tag = 0x110

Set index = 0x0

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008	0x0	Miss	0x10, 0x100
0x0110004	0x0	Miss	0x110, 0x10
0x0000008	0x0	Miss	0x0, 0x110
0x0300004			
0x0300010			
0x0110008			
0x0410004			
0x0A00008			
0x0110004			
0x0A00000			
0x0A00004			
0x0300010			

Remember from before:

- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Next:

0x0000008 =
0000 0000 0000 0000 0000 0000 0000 1000

Tag = 0x0

Set index = 0x0

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008	0x0	Miss	0x10, 0x100
0x0110004	0x0	Miss	0x110, 0x10
0x0000008	0x0	Miss	0x0, 0x110
0x0300004	0x0	Miss	0x300, 0x0
0x0300010			
0x0110008			
0x0410004			
0x0A00008			
0x0110004			
0x0A00000			
0x0A00004			
0x0300010			

Remember from before:

- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Next:

0x0300004 =
0000 0000 0011 0000 0000 0000 0000 0100

Tag = 0x300

Set index = 0x0

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008	0x0	Miss	0x10, 0x100
0x0110004	0x0	Miss	0x110, 0x10
0x0000008	0x0	Miss	0x0, 0x110
0x0300004	0x0	Miss	0x300, 0x0
0x0300010	0x1	Miss	0x300, 0x100
0x0110008			
0x0410004			
0x0A00008			
0x0110004			
0x0A00000			
0x0A00004			
0x0300010			

Remember from before:
- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Next:

0x0300010 =
0000 0000 0011 0000 0000 | 0000 0001 | 0000

Tag = 0x300

Set index = 0x1

OBS! Notice that the state of tags depends of the set index! Meaning right now we have two state of tags, one for 0x0 and one for 0x1

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008	0x0	Miss	0x10, 0x100
0x0110004	0x0	Miss	0x110, 0x10
0x0000008	0x0	Miss	0x0, 0x110
0x0300004	0x0	Miss	0x300, 0x0
0x0300010	0x1	Miss	0x300, 0x100
0x0110008	0x0	Miss	0x110, 0x300
0x0410004			
0x0A00008			
0x0110004			
0x0A00000			
0x0A00004			
0x0300010			

Remember from before:

- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Next:

0x0110008 =
0000 0000 0001 0001 0000 | 0000 0000 | 1000

Tag = 0x110

Set index = 0x0

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008	0x0	Miss	0x10, 0x100
0x0110004	0x0	Miss	0x110, 0x10
0x0000008	0x0	Miss	0x0, 0x110
0x0300004	0x0	Miss	0x300, 0x0
0x0300010	0x1	Miss	0x300, 0x100
0x0110008	0x0	Miss	0x110, 0x300
0x0410004	0x0	Miss	0x410, 0x110
0x0A00008			
0x0110004			
0x0A00000			
0x0A00004			
0x0300010			

Remember from before:

- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Next:

0x0410004 =
0000 0000 0100 0001 0000 0000 0000 1000

Tag = 0x410

Set index = 0x0

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008	0x0	Miss	0x10, 0x100
0x0110004	0x0	Miss	0x110, 0x10
0x0000008	0x0	Miss	0x0, 0x110
0x0300004	0x0	Miss	0x300, 0x0
0x0300010	0x1	Miss	0x300, 0x100
0x0110008	0x0	Miss	0x110, 0x300
0x0410004	0x0	Miss	0x410, 0x110
0x0A00008	0x0	Miss	0xA00, 0x410
0x0110004			
0x0A00000			
0x0A00004			
0x0300010			

Remember from before:

- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Next:

0x0A00008 =
0000 0000 1010 0000 0000 0000 0000 1000

Tag = 0xA00

Set index = 0x0

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008	0x0	Miss	0x10, 0x100
0x0110004	0x0	Miss	0x110, 0x10
0x0000008	0x0	Miss	0x0, 0x110
0x0300004	0x0	Miss	0x300, 0x0
0x0300010	0x1	Miss	0x300, 0x100
0x0110008	0x0	Miss	0x110, 0x300
0x0410004	0x0	Miss	0x410, 0x110
0x0A00008	0x0	Miss	0xA00, 0x410
0x0110004	0x0	Miss	0x110, 0xA00
0x0A00000			
0x0A00004			
0x0300010			

Remember from before:

- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Next:

0x0110004 =
0000 0000 0001 0001 0000 0000 0000 0100

Tag = 0x110

Set index = 0x0

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008	0x0	Miss	0x10, 0x100
0x0110004	0x0	Miss	0x110, 0x10
0x0000008	0x0	Miss	0x0, 0x110
0x0300004	0x0	Miss	0x300, 0x0
0x0300010	0x1	Miss	0x300, 0x100
0x0110008	0x0	Miss	0x110, 0x300
0x0410004	0x0	Miss	0x410, 0x110
0x0A00008	0x0	Miss	0xA00, 0x410
0x0110004	0x0	Miss	0x110, 0xA00
0x0A00000	0x0	Hit	0xA00, 0x110
0x0A00004			
0x0300010			

Remember from before:

- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Next:

0x0A00000 =
0000 0000 1010 0000 0000 0000 0000 0000

Tag = 0xA00

Set index = 0x0

Yes! the first hit!

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008	0x0	Miss	0x10, 0x100
0x0110004	0x0	Miss	0x110, 0x10
0x0000008	0x0	Miss	0x0, 0x110
0x0300004	0x0	Miss	0x300, 0x0
0x0300010	0x1	Miss	0x300, 0x100
0x0110008	0x0	Miss	0x110, 0x300
0x0410004	0x0	Miss	0x410, 0x110
0x0A00008	0x0	Miss	0xA00, 0x410
0x0110004	0x0	Miss	0x110, 0xA00
0x0A00000	0x0	Hit	0xA00, 0x110
0x0A00004	0x0	Hit	0xA00, 0x110
0x0300010			

Remember from before:

- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Next:

0x0A00004 =
0000 0000 1010 0000 0000 0000 0000 0100

Tag = 0xA00

Set index = 0x0

Another hit!

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008	0x0	Miss	0x10, 0x100
0x0110004	0x0	Miss	0x110, 0x10
0x0000008	0x0	Miss	0x0, 0x110
0x0300004	0x0	Miss	0x300, 0x0
0x0300010	0x1	Miss	0x300, 0x100
0x0110008	0x0	Miss	0x110, 0x300
0x0410004	0x0	Miss	0x410, 0x110
0x0A00008	0x0	Miss	0xA00, 0x410
0x0110004	0x0	Miss	0x110, 0xA00
0x0A00000	0x0	Hit	0xA00, 0x110
0x0A00004	0x0	Hit	0xA00, 0x110
0x0300010	0x1	Hit	0x300, 0x100

Remember from before:

- set index = bit 5-12 (8 bits)
- Tag = bit 13-32 (20 bits)

Cache is cold = empty

Next:

0x0300010 =
0000 0000 0011 0000 0000 | 0000 0001 | 0000

Tag = 0x300

Set index = 0x1

And lastly, a hit!

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0x0	Miss	0x100
0x0100010	0x1	Miss	0x100
0x0010008	0x0	Miss	0x10, 0x100
0x0110004	0x0	Miss	0x110, 0x10
0x0000008	0x0	Miss	0x0, 0x110
0x0300004	0x0	Miss	0x300, 0x0
0x0300010	0x1	Miss	0x300, 0x100
0x0110008	0x0	Miss	0x110, 0x300
0x0410004	0x0	Miss	0x410, 0x110
0x0A00008	0x0	Miss	0xA00, 0x410
0x0110004	0x0	Miss	0x110, 0xA00
0x0A00000	0x0	Hit	0xA00, 0x110
0x0A00004	0x0	Hit	0xA00, 0x110
0x0300010	0x1	Hit	0x300, 0x100

Reference	Set index	Hit/Miss	State of Tags
0x0100004	0	miss	0x100
0x0100010	1	miss	0x100
0x0010008	0	miss	0x10,0x100
0x0110004	0	miss	0x110,0x10
0x0000008	0	miss	0x0,0x110
0x0300004	0	miss	0x300,0x0
0x0300010	1	miss	0x300,0x100
0x0110008	0	miss	0x110,0x300
0x0410004	0	miss	0x410,0x110
0x0A00008	0	miss	0xA00,0x410
0x0110004	0	miss	0x110,0xA00
0x0A00000	0	hit	0xA00,0x110
0x0A00004	0	hit	0xA00,0x110
0x0300010	1	hit	0x300,0x100

Good luck on the exam!!!