

EECS
2021

Computer
Organization

CACHES

The Fast-Slow Interface

Prof. H. Roumani,
Dept of EECS, York University

1

CACHES

- End-to-End Performance
- DRAM and SRAM
- The Idea & Basic Concepts
- Implementation
- Improving Performance: Multi-Word
- Improving Performance: Associative
- Other Caches: L2, Data, and TLBs

2

Example

```
do
{
    int el = ar[i];
    sum += el;
    sha ^= el;
    i++;
} while (el != last);
```

```
loop: lw    $t5, ar($t0)
      addu  $s0, $s0, $t5
      xor   $s2, $s2, $t5
      addi  $t0, $t0, 4
      bne   $t5, $a0, loop
```

RF=50, ALU=100, and MEM (both IM and DM)=200 ps

CPU	Clock GHz	Time/Loop ns
S/C	1.66 GHz	3.00
M/C	variable	2.15
P/L	5.00 GHz	1.00 – 1.60


3

End-to-End Performance

RF=50, ALU=100, and MEM (both IM and DM)=200 ps

TYPE	\$/GB	~SIZE	LATENCY *
SRAM	5000	MB	200-500 ps
DRAM	25	GB	50-100 ns
FLASH	1	GB	0.05-0.1 ms
DISK	0.01	TB	5-50 ms

* End-to-End to access of a random byte



4

DRAM

- The Bus Clock and its physical limit
Speed of light in conductors
- DRAM Physical Structure
Capacitors, destructive reads, refresh rate
- Latency Per Byte
The storage matrix
- Interleaving
Parallel performance at a price = alignment

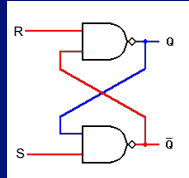
5

SRAM

- The Magic Circuit
- Building SRAM: from R-S to F-F
- Transistors per bit
- Heat Dissipation

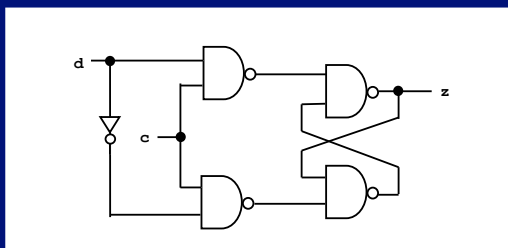
6

R-S Latch



7

D Flip Flop



8

D Flip Flop –Edge Triggered

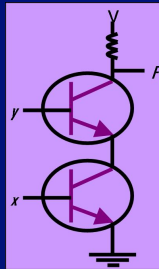
Enhance the D FF so that:

1. It only takes input on the rising edge
2. It has an enable

9

CU2021/HR

Transistor per bit



10
EECS2021/RR

Heat Dissipation

- CPU Size
datapath + control + few MB SRAM = G Transistor
- How small
Currently 32 nm per transistor (about 100 Si atoms)
1 blood cell 8500 nm, 1 grain of powder 10,000 ns
- CPU Heat Density
At 100W $\rightarrow 10^8$ W/m²

To contextualize, compute the Sun's heat density using Stefan-Boltzmann's Law.

11

The Cache Idea

When you fetch a word from DRAM, store a copy of it in an on-board, static memory (the cache).

In general, caching the result of a lengthy process has the potential of improving performance.

Can you think of examples of this strategy in other contexts?

12

The Idea ...

Caching can improve performance provided:

1. The word is re-requested
Temporal Locality
2. The word is still in the cache
High Hit Rate (= low miss rate)
3. Cache search faster than re-fetching
Hit Penalty << Miss Penalty

Achievable?

Yes, for instructions

Yes, for a big cache

Yes, for a small cache and a fast search

13

Cache Terminology

- Storage & Transfer "Chunks"
Blocks, Cache Lines, Words/Block
- Hit/Miss Ratio; Hit/Miss Penalty
Effective DRAM access time
- Cache Organization
Directly Mapped – Set – Fully Associative
- Policies
Fetch, Placement, Replacement, Write
- Cache Hierarchy
Instruction/Data, L1, L2, L3; Cache Consistency

14

Basic Concepts

- Instruction versus Data Caching
Temporal versus Spatial Locality
- The cache must be big yet small
Keep L1 small and make L2, L3 big
- Cache search must be fast
Use Hashing Techniques + Parallel hardware
- Flush the cache on a context switch
Use a valid bit to flag a dirty block

15

A Basic Example

16B (4 blocks, 1 W/block) I-only cache.
 Assume 100ns/DRAM, 1 ns/ SRAM.
 Compute DRAM time / iteration.

```

loop: lw    $t5, ar($t0)
      addu  $s0, $s0, $t5
      xor   $s2, $s2, $t5
      addi  $t0, $t0, 4
      bne   $t5, $a0, loop
    
```

Answer: 505 ns for the first iteration, 205 ns for subsequent ones (exc data)

Explorations:

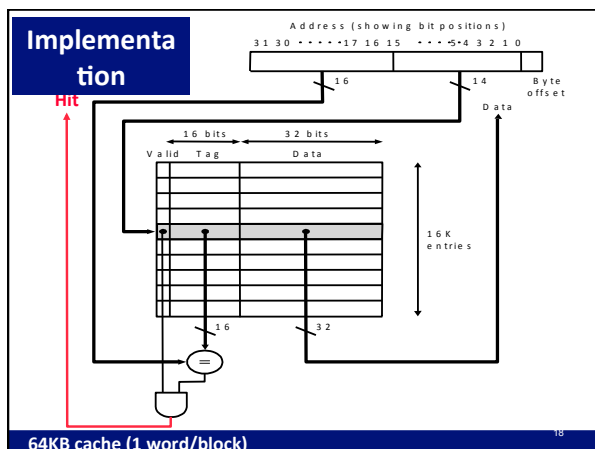
- Compute the exact cache size in bits and the bit utilization rate.
- What if we also cached data (in addition to instructions)?
- What if the loop had 4, 6, 7, or 8 instructions in its body?

What is the bit utilization rate? 50%

A Realistic Example

A 64KB cache implemented as:

- 16K blocks
- 1 word per block



Improving Performance: Multi-Word Blocks

The Basic Example

Four blocks, 1 word per block, 244 bits total size. Make it support bigger loops **without** increasing its size too much. Try these:

1. Eight Blocks with 1 instruction per block...
2. Four Blocks of two **consecutive** instructions each...
3. Two blocks of four words each...

Does this increase hit rate? How about miss penalty?

It utilizes **spatial locality**. It is a form of **pre-fetching**. Manifested in the "Block Offset" redundant bits

19

Multi-Word Blocks

- Keeps the cache size small (why?)
Increases the utilization rate
- Tends to increase the miss penalty (why?)
Multi-fetch unless wider memory channel
- May increase the miss rate (why?)
Due to conflict (not capacity) misses (the 3C's)
- Does it increase the hit penalty?
Not if parallel word search is done

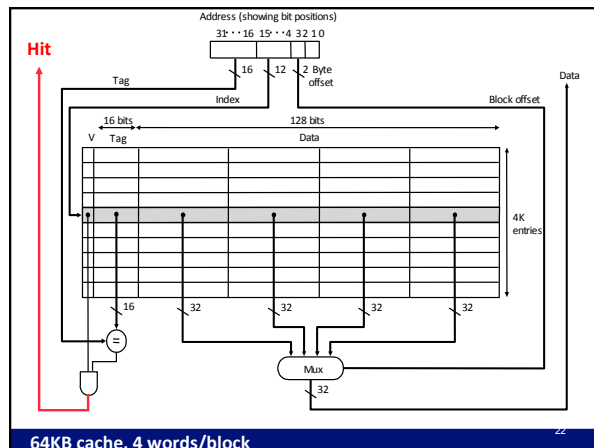
20
CSE202 L/HK

Multi-Word Blocks: A Realistic Example

A 64KB cache implemented as:

- 4K blocks
- 4 instructions per block

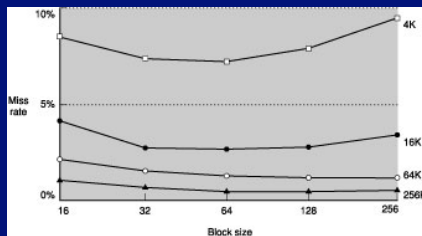
21



Multi-Word Blocks: A Critique

As the #of words per block increases:

- The miss rate increases due to "edge effects"
- The miss penalty increases due to bus/DRAM width



Source: COAD, 3rd Ed, Patterson & Hennessy, Fig. 7.8

23

Exercise

Draw the structural diagram and compute the exact size of a 256 KB direct-mapped cache with 16 words per block. Can the following two addresses coexist in this cache:

Address1 = 0111 1000 1111 0001 1101 1010 0000 0000
Address2 = 0011 1100 1111 0001 1101 1010 0010 1000

The answer must indicate where each word is stored in the cache and why there is (or isn't) a conflict between the two words.

24

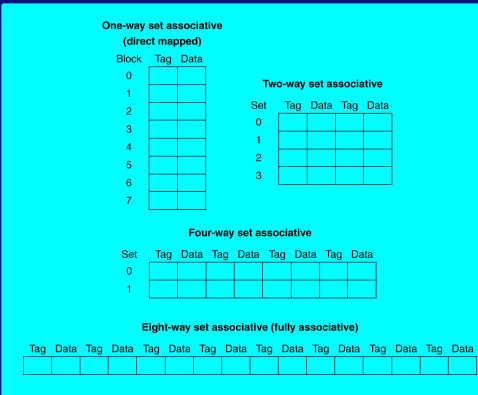
Improving Performance: Associative

To avoid the heavy miss penalty incurred when a large block is to be replaced, keep the words within it de-coupled.

- Direct-Mapped is one extreme
- Fully Associative is another
- Set Associative is a good compromise

The flexibility of replacing just one word yet still benefiting from bit redundancy allows us to reduce the miss penalty as well as the miss rate. The only problem is that search is slower (and/or hotter), and hence, the hit penalty can be higher.

25



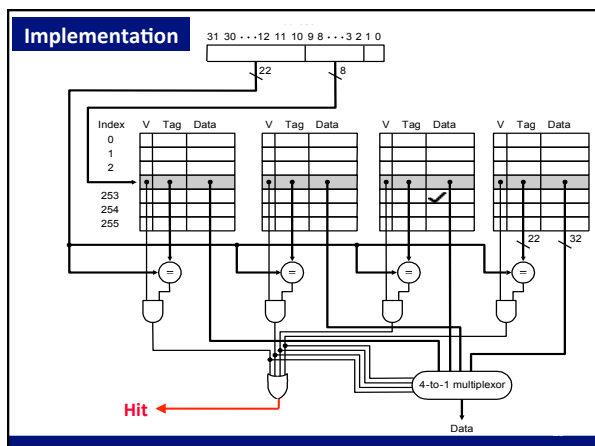
26

A Realistic Example

A 4KB cache implemented as:

- 4-Way Set Associative (1 word/block)
- Parallel Search Hardware

27



Exercise

Draw the structural diagram of a 64KB cache organized as two-way set associative with one word per block and compute its exact size in kilobits. The diagram must indicate the name and size of each field in the cache.

29

Exercise

A loop requests the following sequence of data addresses:

```
Address1 = 0111 1000 1111 0001 1101 1010 0001 1000
Address2 = 0111 1000 1111 0001 1101 1010 0010 1100
Address3 = 0111 1000 1111 0001 1101 1010 0011 1000
Address4 = 0111 1000 1111 0001 1101 1010 0100 1100
Address5 = 0111 1000 1111 0001 1101 1010 0110 1000
Address6 = 0111 1000 1111 0001 1101 1010 0111 0100
```

The CPU is equipped with a 32B cache which can be organized in one of two ways: direct-mapped with four words per block, or set-associative with two sets each of which is four-way associative. Which organization is better for the loop?

30
