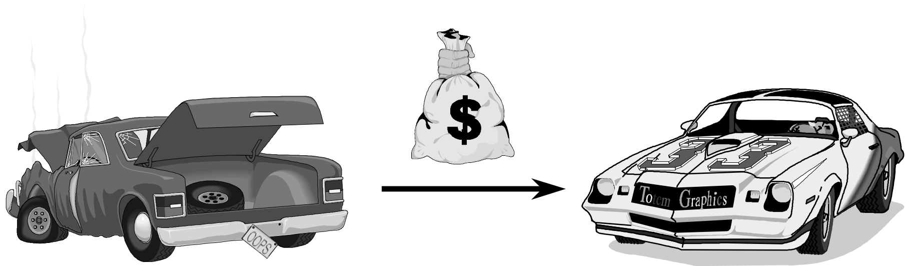
Multilevel Memories
(Improving performance using a little "cash")



1

Memory Systems: Present Day Numbers

Type	Capacity	Cost	Latency
Tape	16 GB	$0.10/MB	120 s
DISK	2-8 GB	$1.00/MB	10 ms
DRAM	256 MB	$10/MB	50 ns
SRAM (off-chip)	16 MB	$100/MB	10 ns
SRAM (on-chip)	1 MB	??	2 ns

2

6004 Spring 1998: L36

Fundamental Issue in Computer Design

Latency of memory reads.

How do we build a large and fast memory?

Aside: Pipelining a memory improves throughput not latency.

3

Answer: Multilevel Memory

Strategy: Hide latency using small, fast memories called caches.

Caches are a mechanism to hide memory latency based on the empirical observation that the stream of memory references made by a processor exhibits __________.

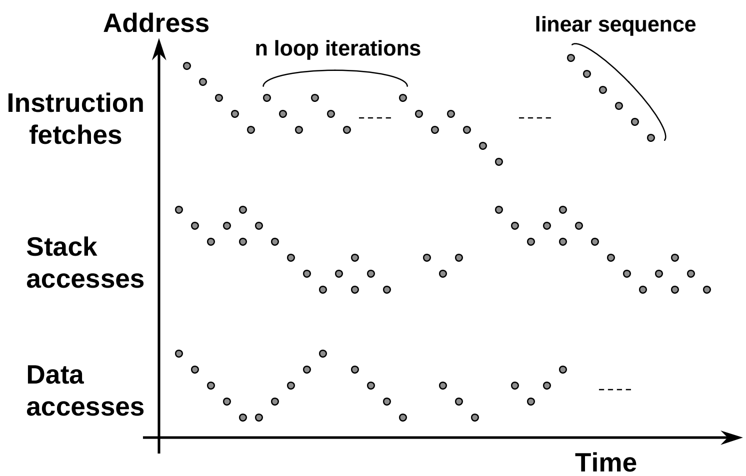
	PC
...	96
loop: MUL(r1, r0, r1)	100
SUB(r0, 1, r0)	104
BNE(r0, loop)	108
...	112

What is the pattern of instruction memory addresses?

4

6004 Spring 1998: L36

Typical Memory Reference Patterns



5

Locality Properties of Patterns

Two locality properties of memory references:

- **If a location is referenced it is likely to be referenced again in the near future. This is called temporal locality.**
- **If a location is referenced it is likely that locations near it will be referenced in the near future. This is called spatial locality.**

6

6004 Spring 1998: L36

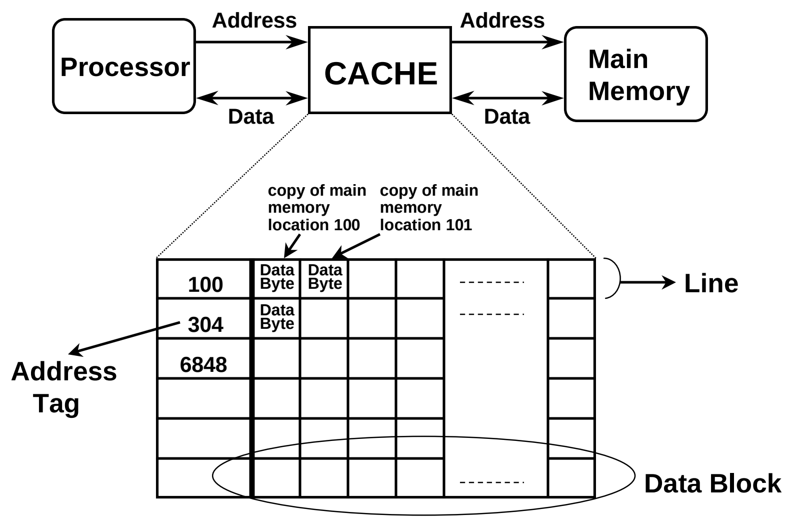
Caches

Caches exploit both properties of patterns.

- **Exploit <u>temporal locality</u> by remembering the contents of recently accessed locations.**
- **Exploit <u>spatial locality</u> by remembering blocks of contents of recently accessed locations.**

7

Inside a Cache



8

Cache Algorithm (Read)

Look at Processor Address, <u>search cache tags</u> to find match. Then either

Found in cache
a.k.a. _____

Return copy of data from cache

Not in cache
a.k.a. _____

Read <u>block</u> of data from Main Memory

Wait …

Which line do we replace? → **Return data to processor and update cache**

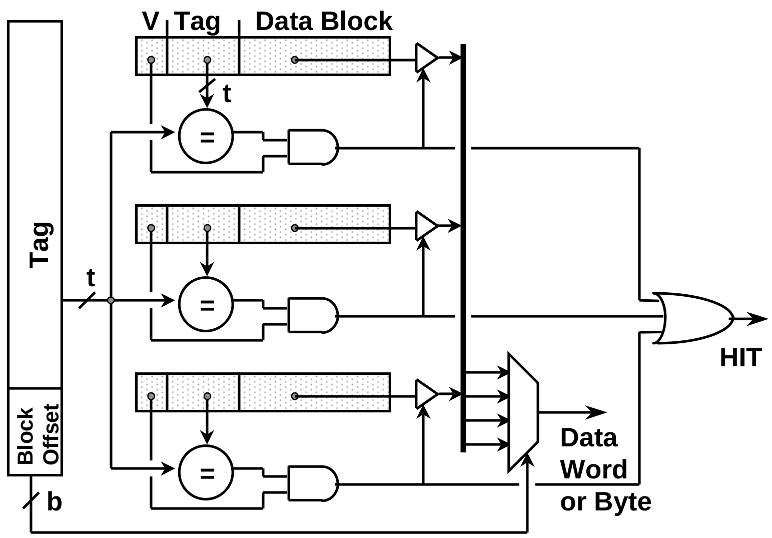
9

Cache Organization

- **Cache has to be FAST and therefore small.**
- **Have to perform search in parallel and/or limit the number of places in the cache that a particular address (block) will be found.**
  - **Fully associative cache: Block can be anywhere in cache!**
  - **Direct-mapped cache: Block can be in only one place in the cache.**
  - **Set-Associative cache: Block can be in a few (usually 2 to 8) places in the cache.**
- **A <u>set</u> is a collection of cache locations in which a given memory block may be placed.**

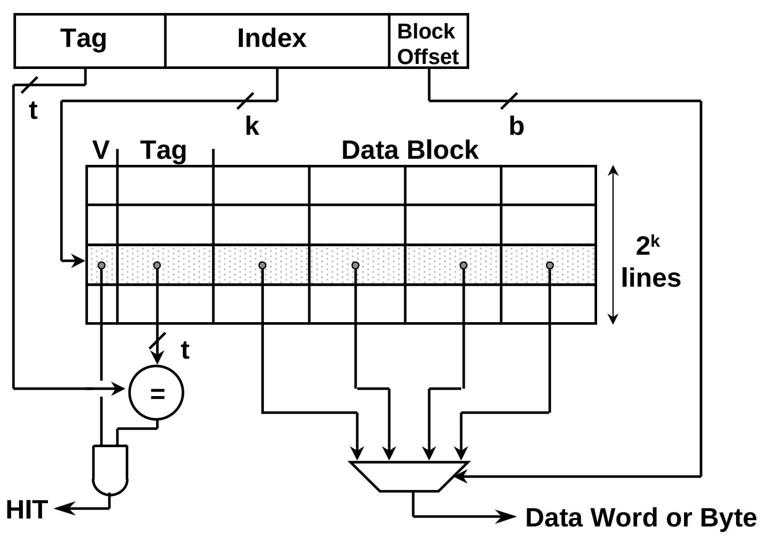
10

Fully Associative Cache



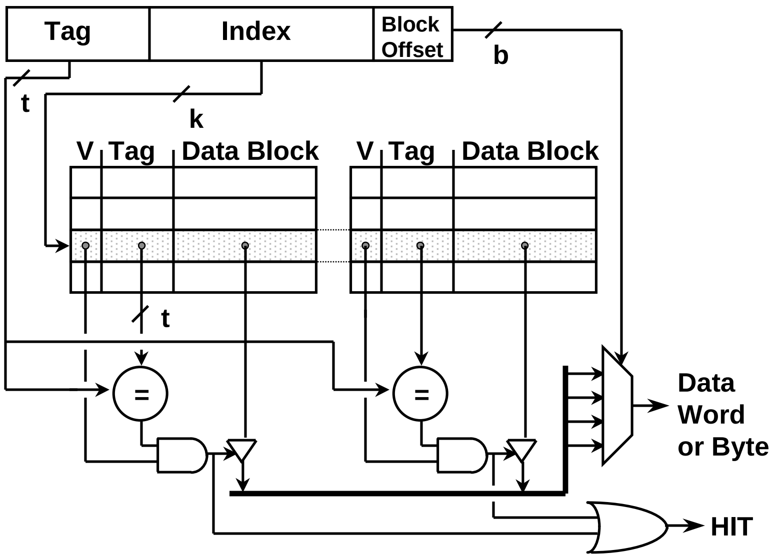
11

Direct-Mapped Cache



12

2-Way Set-Associative Cache



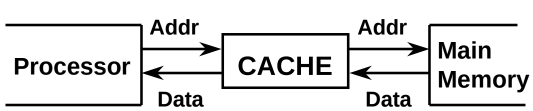
13

Average Read Latency using a Cache

α is HIT RATIO: Fraction of references in cache

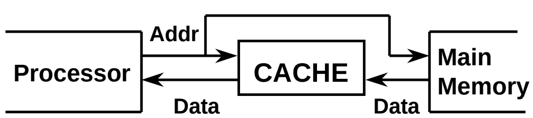
1 - α is MISS RATIO: Remaining references

Average access time for serial search:



$t_c + (1 - \alpha)\, t_m$

Average access time for parallel search:

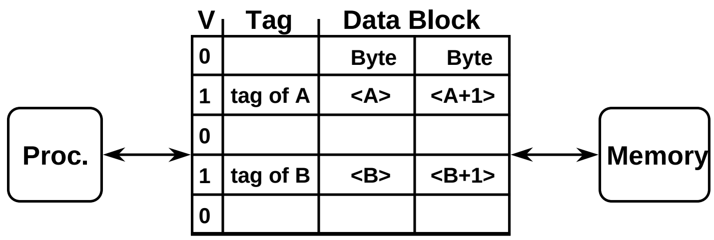


____

14

6004 Spring 1998: L36

Valid Bits



- Valid bit must be 1 for cache line to HIT.
- At power-up or reset, we set all valid bits to 0.
- Set valid bit to 1 when cache line is first replaced.
- Flush cache by setting all valid bits to 0, under external program control.

15

Block Replacement: Selecting the Victim

- **After a read miss, what cache block should we replace with the data block read from main memory?**

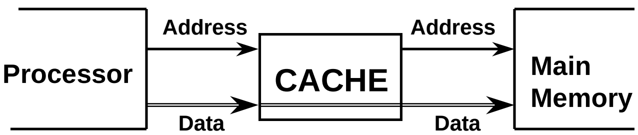
- **Direct-mapped cache: Unique line.**
- **Associative caches:**
  - Least recently used (LRU): Replace block in set corresponding to address which has not been read or written for the longest time.
  - Random: Select block in set randomly.
  - First-In First-Out (FIFO): Select block which has been in the set for the longest time.

16

6004 Spring 1998: L36

Write Policy

- What happens when we want to write data into a particular address?



- Possibilities are:
  – No-Write: Writes go to main memory and flush out cache.
  – Write-Through: Writes go to main memory and cache.
  – Write-Back: Write cache, write main memory only when block is replaced.

17

Dirty Bits for Write-Back Caches

Proc.

D	V	Tag	Data Block	
0	0		Byte	Byte
1	1	tag of A	<A>	<A+1>
0	0			
0	1	tag of B	<B>	<B+1>
0	0			
0	0			

Memory

When the line corresponding to A is replaced, its data block has to be written to main memory since its dirty bit is set.

18

Cache Issues

- What should the size of blocks in a cache be?
- What should the degree of associativity be?
- How do we choose a replacement strategy?
- What should the write policy be?
- Separate data and instruction caches?
- How many levels of caches in a multilevel memory? What size for each cache?

19