Slide 1

The Memory Hierarchy

gelato.org

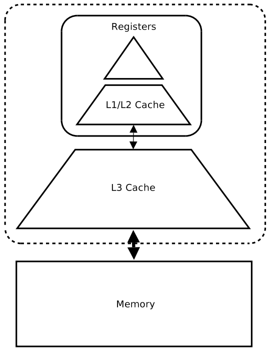
This work supported by UNSW and HP through the Gelato Federation

Slide 2

Memory Hierarchy Overview

Why a memory hierarchy?:

- ➜ Fast == Expensive
- ➜ Exploit locality
  - Spatial
  - Temporal



Slide 3

Cache Design

- ➜ Cache Levels
  - L1/L2 usually on chip
  - L2 usually unified
  - L3 **much** larger
  - L1 tied to clock rate, lower levels tied to miss cost of L1
- ➜ Cache Lines
- ➜ Split cache
  - Instruction cache
  - Data cache

Slide 4

Cache Organisation

Where can a cache line live?:

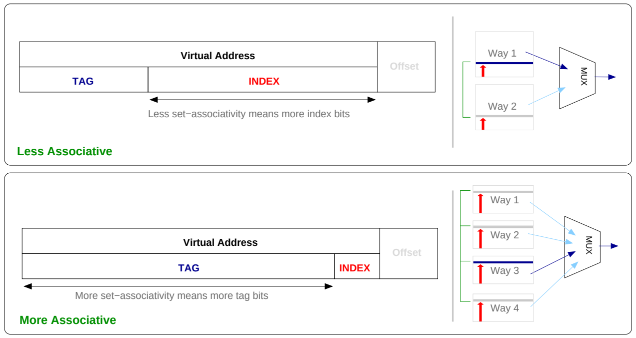
- ➜ Direct Mapped
- ➜ Fully Associative
- ➜ Set Associative
  - *n-way set associative* is to divide total cache into n compartments.
  - Line may live anywhere in set $total\_blocks \bmod n$

How do we find a cache line?:

- ➜ Index
- ➜ Tag
- $index\_bits = log_2(\frac{\frac{cache\_size}{associativity}}{line\_size})$

Slide 5

Where is that line - Tags and Indexes



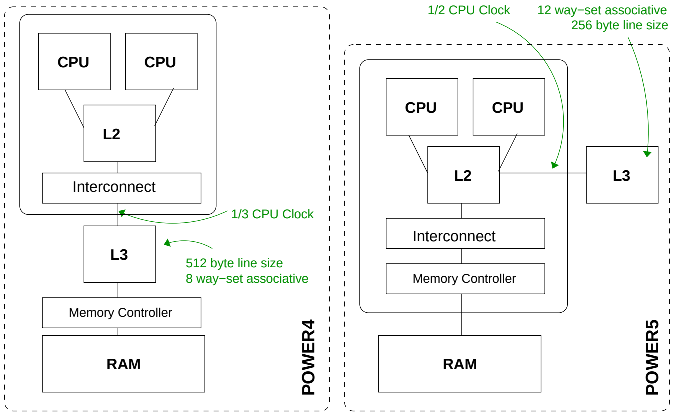
Slide 6

Quickly - other cache parameters

- ➜ Replacement Policy
  - ● LRU/Random/FIFO
- ➜ Write policy
  - ● Through
  - ● Back
- ➜ Inclusive or Exclusive?

Slide 7

POWER



Slide 8

Chronology of a cache hit

① Processor loads from an address

② Address is requested in the cache

③ Index selects offset within (all) ways

④ Tag selects correct entry from set

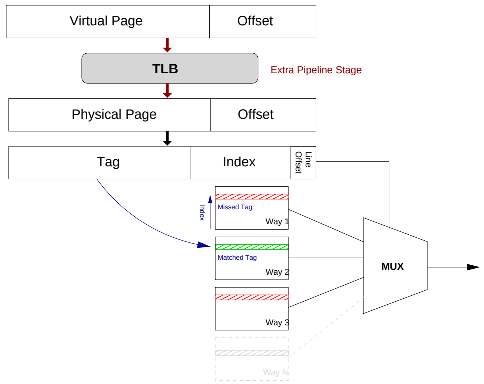
⑤ Data is retrieved from selected line

Cache addressing: This address is a **virtual address**

- ➜ Virtual addresses may *alias*
- ⛔ Cache must be *coherent*
- ✗ ***Synonyms*** : Δ VA, ≡ page
- ✗ ***Homonyms*** : ≡ page, Δ VA

Slide 9

Physically Indexed and Tagged Cache

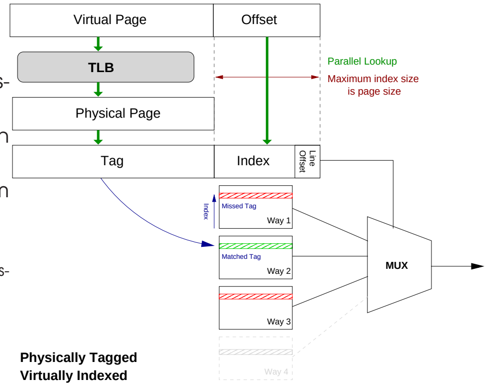


- ➜ TLB translates VA to PA
- ☑ No aliases
- ☒ Extra overhead

Slide 10

Physically Tagged, Virtually Indexed

- ➜ Tag is based on PA
- ➜ Index based on untranslated offset bits
- ☑ TLB lookup happens in parallel
- ☒ Index limited to system page size
  - ➜ Unless bits are shared. . .
  - ☒ . . . which introduces aliasing



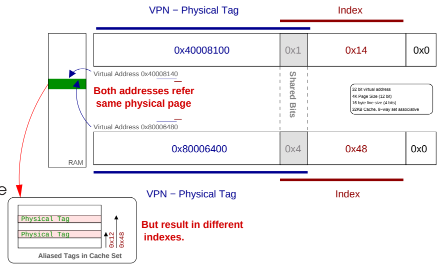
Slide 11

Virtually Tagged, Virtually Indexed

- ☑ TLB less involved
- ☑ No size limitations
- ☒ Synonyms **and** homonyms

Protection:

- ➜ TLB still required for protection
- ➜ Active research area
  - ➜ Protection details in cache
  - ➜ PLB
  - ➜ Capabilities
  - ➜ Segmentation



VPN − Physical Tag
Index
0x40008100
0x1
0x14
0x0
Virtual Address 0x40008140
Both addresses refer same physical page
Shared Bits
32 bit virtual address
4K Page Size (12 bit)
16 byte line size (4 bits)
32KB Cache, 8−way set associative
Virtual Address 0x80006480
RAM
0x80006400
0x4
0x48
0x0
VPN − Physical Tag
Index
Physical Tag
Physical Tag
0x12
0x48
Aliased Tags in Cache Set
But result in different indexes.

Slide 12

Dealing with Aliasing

- ☒ Flush cache on context switch (Sledgehammer)

Software Approaches:

- ➜ SASOS
  - ➜ Mungi
- ➜ Colouring
  - ➜ Make shared items align in the cache (SunOS)
  - ➜ Globally visible shared region (OS/2)

Slide 13

Dealing with Aliasing

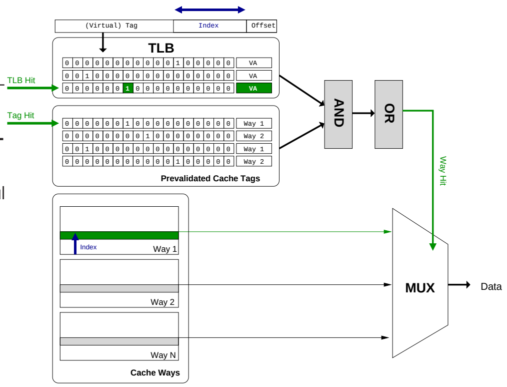
Hardware Approaches:

- ➜ Reverse Maps
  - ➜ Back Pointers (MIPS R6000, Alpha?)
  - ➜ Dual Directories
- ➜ ASID or segmentation
  - ✓ Add bits to distinguish VAs
  - ✗ Makes sharing harder
  - ● Itanium Region Registers

Slide 14

Itanium2 - Prevalidated Cache

- ➜ Tie TLB and L1 closely together
- ➜ Cache tagged with **TLB location**
- ➜ No need to find physical tag
- ➜ Simple AND
- ➜ 1 cycle latency


(Virtual) Tag
Index
Offset
TLB
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 VA
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 VA
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 VA
TLB Hit
Tag Hit
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 Way 1
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 Way 2
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Way 1
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 Way 2
Prevalidated Cache Tags
AND
OR
Way Hit
Index
Way 1
Way 2
Way N
Cache Ways
MUX
Data

Slide 15

Dealing with Misses

So far, everything has been about hit latency

Types of misses:

① Compulsory
② Capacity
③ Conflict

Slide 16

More Cache?

- ✓ Less compulsory misses
- ✗ $$$
- ● $cache\_size = line\_size * set\_index * assocativity$
- ● ⇑ cache means more *what?*
  - ➜ Greater line size
  - ➜ Greater associativity
  - ➜ Greater index size

Slide 17

Other miss penalty reduction schemes

- ➜ Critical word first
- ➜ Victim caches
- ➜ Way prediction
- ➜ Trace Cache

Slide 18

Prefetching

- ➜ Requires non-blocking cache

Concept Hide latency by overlapping execution with fetching.

- ➜ Very useful for loops
- ➜ *stride* is distance a loop jumps in memory
- ➜ Hardware or Software based

Slide 19

Prefetching Example

Walk an array in cache sized lines

Metric	ICC	GCC	Description
Run Time (seconds)	0.824	1.507	Program execution time
L1D_READ_MISSES_ALL	12,514,832	12,505,200	L1 Data Cache Read Misses
BE_EXE_BUBBLE_GRALL	129,629,838	737,891,717	Full Pipe Bubbles in Main Pipe due to Execution Unit Stalls
BE_EXE_BUBBLE_GRGR	0	0	Back-end was stalled by exe due to GR/GR dependency

Why?:

```
4000000000000940:       [MII] (p16) ld4 r32=[r3],64
4000000000000946:             (p17) add r34=r35,r33
400000000000094c:                   nop.i 0x0
4000000000000950:       [MMB] (p16) lfetch.nt1 [r2],64
4000000000000956:                   nop.m 0x0
400000000000095c:                   br.ctop.sptk.few 4000000000000940 <walk+0x80>;;
```

Slide 20

If you are still awake, I owe you a beer

Questions?