Hardware Optimization

Cache hierarchy requirements

+ decrease effective memory access time
+ different cache designs for different purposes

- level 1 -- **goal: fast access**
  - small cache
  - virtually-addressed
  - direct-mapped or set-associative?
  - contain instructions or separate
  - write-through
- level last -- **goal: keep traffic off the system bus**
  - big cache
  - physically-addressed
  - direct-mapped
  - large block size
  - unified
  - write-back

Inclusion property

- lower level cache data in next level cache
- external accesses only need to check highest level cache
- different caches in the hierarchy have different block sizes: is there a problem here?

Hardware Optimization

Cache hierarchy (cont'd)

$$\textbf{EAT} = \text{hit}_{L1} + \text{mr}_{L1} \bullet \text{misspenalty}_{L1}$$
$$(\text{hit}_{L2} + \text{mr}_{L2} \bullet \text{misspenalty}_{L2})$$

$$\textbf{local MR} = \frac{\#\text{misses}}{\#\text{accesses}} \text{ for that cache!}$$

$$\textbf{global MR} = \frac{\#\text{ misses in cache}}{\#\text{ references generated by CPU}}$$
$$\text{missrate}_{L1} \bullet \text{missrate}_{L2}$$

Example:

1000 refs	local MR (L1)
40 L1 misses	global MR (L1)
10 L2 misses	local MR (L2)
	global MR (L2)

Which cache will normally have the higher local hit ratio and why?

Hardware Optimization

Subblocks (A S)

tag	1	subblock	1	subblock
tag	0	invalid data	1	subblock

+ less transfer time on a miss
- valid bit per subblock
+ fewer tags of larger blocks
+ possibly decrease write time

Other techniques to reduce effective transfer time

- **early restart**:
  - send requested word to the CPU as soon as it arrives
- **requested word first**: (P)
  - request the load address first
  - other words in the block arrive later

Hardware Optimization

Non-blocking cache (lockup-free cache)

+ hides memory latency

- CPU continues executing instructions while miss is handled
  - **nonblocking loads**: execute instructions until the load use
  - **out-of-order execution**: execute any independent instructions, even if follow the load use
- "hit under miss" (A)
- "miss under miss": multiple misses outstanding (A, M, S, X)
- **miss status holding registers** (MSHR)
  - physical address of the data
  - destination register number
  - mechanism to merge requests to the same block (A)
  - mechanism to insure accesses to the same location execute in program order

Hardware Optimization

Victim cache (Jouppi 1990) (A)

- small fully-associative cache
- contains the most recently replaced blocks
- check it on a cache miss
- alternative to 2-way set-associative cache
    - used with direct-mapped caches

+ reduces miss penalty/conflict misses: why does it work?

Hardware Optimization

Pseudo-set associative cache (M S)

- prediction bit for which set to check
- second access if miss on first block
- how might you implement it?

+ miss rate of 2-way set associative cache
+ access time of direct-mapped cache

Hardware Optimization

Pipeline cache access

+ hides memory latency

- for example:
  - arbitrate for bus to cache
  - tag check
  - hit/miss logic
  - data access 1
  - data access 2, data transfer 1
  - data transfer 2
  - optimization: pipeline writes (A)

Hardware Optimization

Prefetching

+ hides memory latency

- instructions
  - next sequential instructions (A M)
  - multiple paths
  - prefetch instruction (S)
- data
  - stride-based prefetching (Baer & others)

- **stream buffers** (Jouppi 90, Farkas & Jouppi 94) (A M)
  - where prefetched instructions/data held
  - if requested block in the stream buffer, then cache access is cancelled for a prefetch

Compiler Optimization

Compiler-controlled prefetching

- **nonbinding** (data to the cache, not the registers)
- **nonfaulting** (causes no exceptions)
- used with lockup-free (or non blocking) caches

- usually for data in loops
  - loop unrolling
  - software pipelining
  - cache miss analysis (Lam & Mowry)

+ hides memory latency
- additional instructions
- additional conflict misses

Compiler Optimizations

Improve spatial locality

- **group & transpose**
  - portions of vectors that are accessed together lie in memory together
- **loop interchange**
  - so inner loop follows memory layout

Improve temporal locality, decrease capacity misses

- **loop fusion**
  - do multiple computations on the same portion of an array
- **blocking or tiling**
  - do all computation on a small block of memory

Blocking Example

```
/* before */
for (i=0; i<n; i=i+1)
    for (j=0; j<n; j=j+1){
        r = 0;
        for (k=0; k<n; k=k+1) {
            r = r + y[i,k] * z[k,j]; }
        x[i,j] = r;
        };

/* after */
for (jj=0; jj<n; jj=jj+B)
for (kk=0; kk<n; kk=kk+B)
for (i=0; i<n; i=i+1)
    for (j=jj; j<min(jj+B-1,n); j=j+1) {
        r = 0;
        for (k=kk; k<min(kk+B-1,n); k=k+1)
            {r = r + y[i,k] * z[k,j]; }
        x[i,j] = x[i,j] + r;
        };
```


Machine Comparison

L1 on-chip instruction cache

Alpha 21164	MIPS R10000	P6 Pentium Pro	UltraSparc 1
8KB direct- mapped 32B block virtual index physical tags prefetching with stream buffers	32KB 2-way LRU 64B block critical word first prefetching with stream buffers	8KB 4-way 32B virtual index physical tags nonblocking	16KB pseudo 2-way 32B block virtual index

Machine Comparison

L1 on-chip data cache

Alpha 21164	MIPS R10000	P6 Pentium Pro	UltraSparc 1
8KB direct- mapped 32B block	32KB 2-way LRU 32B block	8KB 2-way 32B block	16KB direct- mapped 32B block 16B subblock
virtual index physical tags TLB in parallel	physical	virtual index physical tags TLB in parallel	virtual index physical tags TLB in parallel
write-through no-write-allocate	write-back		write-through store compres- sion
miss under miss	nonblocking	nonblocking (hit under miss?)	miss under miss (4)
dual read ported	2 banks requested word first	2 ports/4 banks	
	stream buffers		

Machine Comparison

L2 data cache

Alpha 21164 on-chip	P6 Pentium Pro same package
96KB 3-way 64B block/32B sub- block valid bits/16B	256KB 4-way 32B block
write-back (write allocate)	
physical index	physical index physical tags
banked	requested work first
miss under miss (2)	nonblocking
victim cache (2)	
pipelined	

Machine Comparison

External caches

Alpha 21164	MIPS R10000	UltraSparc 1
1MB-64MB direct-mapped	512KB-16MB 2-way pseudo LRU	512KB-4MB direct-mapped
32/64B blocks	64/128B blocks	8B block
write-back (write alloc)		write-back (write alloc)
physical		physical
hit under miss (1)		
		pipelined access
inclusion		inclusion