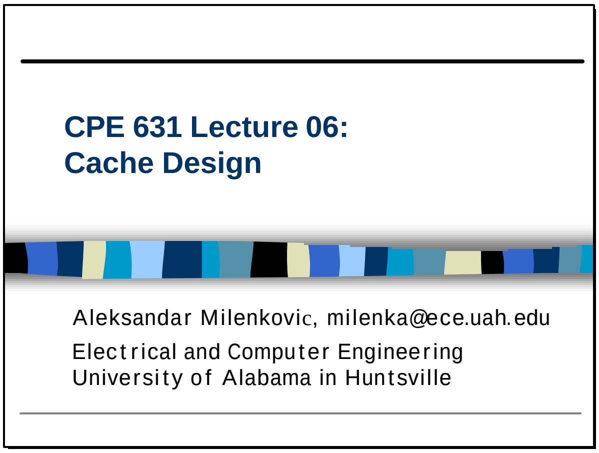
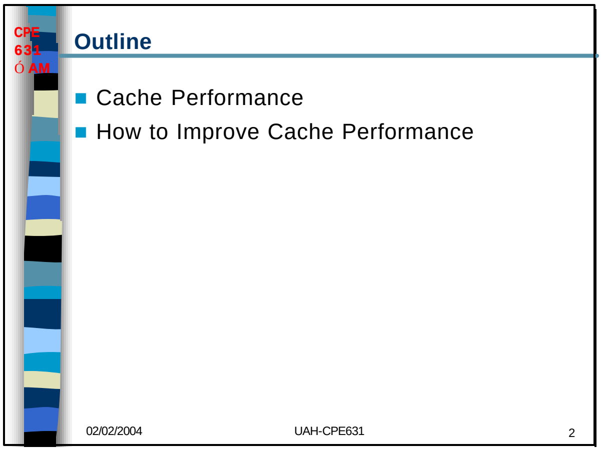


CPE 631 Lecture 06: Cache Design

Aleksandar Milenkovic, milenka@ece.uah.edu

Electrical and Computer Engineering
University of Alabama in Huntsville

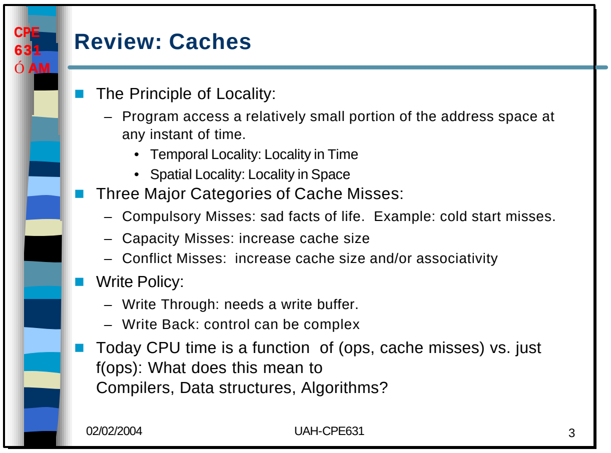


CPE 631 Ó AM

Outline

- Cache Performance
- How to Improve Cache Performance

02/02/2004 UAH-CPE631 2

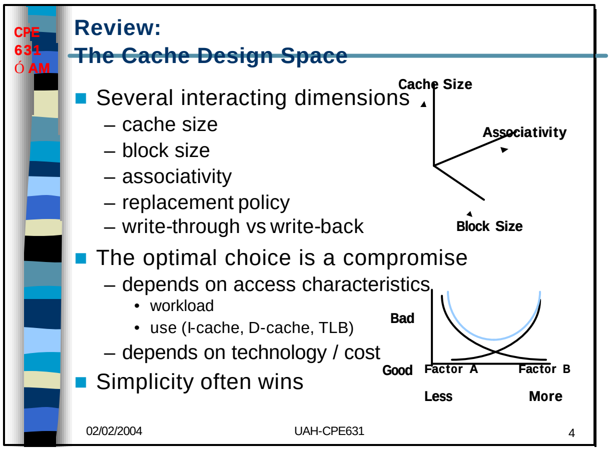


CPE 631 Ó AM

Review: Caches

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
    - Temporal Locality: Locality in Time
    - Spatial Locality: Locality in Space
- Three Major Categories of Cache Misses:
  - Compulsory Misses: sad facts of life. Example: cold start misses.
  - Capacity Misses: increase cache size
  - Conflict Misses: increase cache size and/or associativity
- Write Policy:
  - Write Through: needs a write buffer.
  - Write Back: control can be complex
- Today CPU time is a function of (ops, cache misses) vs. just f(ops): What does this mean to Compilers, Data structures, Algorithms?

02/02/2004 UAH-CPE631 3

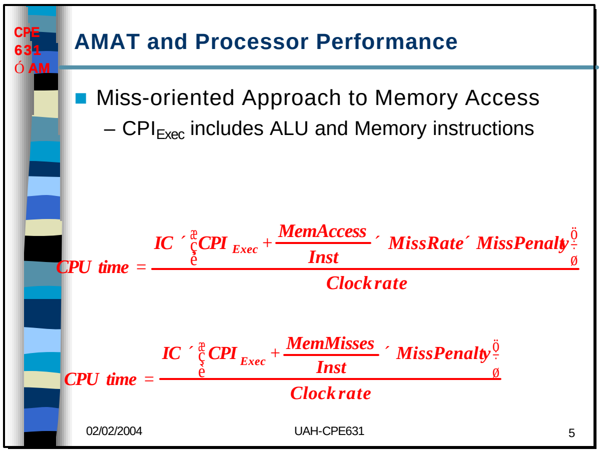


CPE 631 Ó AM

Review: The Cache Design Space

- Several interacting dimensions
  - cache size
  - block size
  - associativity
  - replacement policy
  - write-through vs write-back
- The optimal choice is a compromise
  - depends on access characteristics
    - workload
    - use (I-cache, D-cache, TLB)
  - depends on technology / cost
- Simplicity often wins

02/02/2004 UAH-CPE631 4



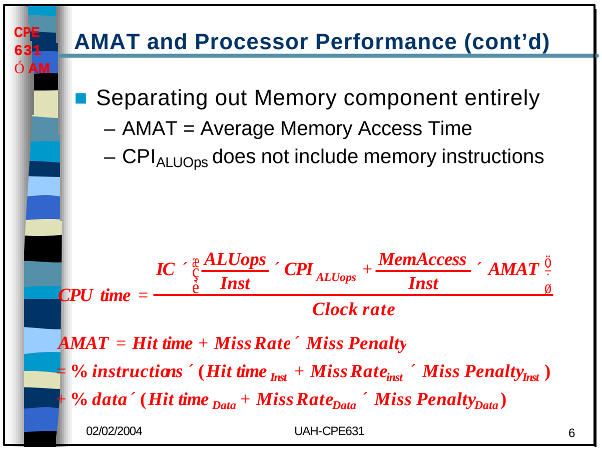
CPE 631 Ó AM

AMAT and Processor Performance

- Miss-oriented Approach to Memory Access
  - CPI_{Exec} includes ALU and Memory instructions

$$CPU\ time = \frac{IC \times \left(CPI_{Exec} + \frac{MemAccess}{Inst} \times MissRate \times MissPenalty\right)}{Clock\,rate}$$

$$CPU\ time = \frac{IC \times \left(CPI_{Exec} + \frac{MemMisses}{Inst} \times MissPenalty\right)}{Clock\,rate}$$



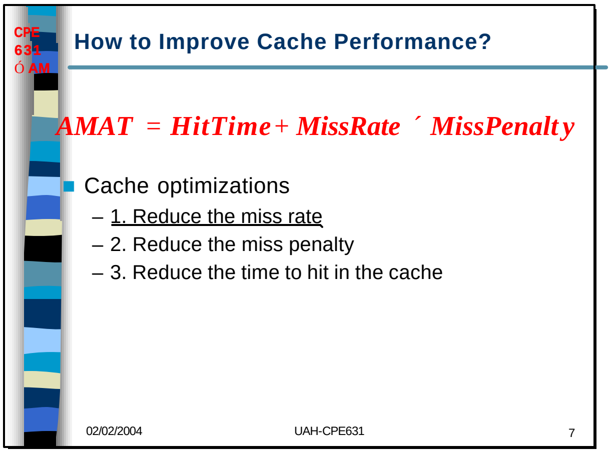
CPE 631 Ó AM

AMAT and Processor Performance (cont'd)

- Separating out Memory component entirely
  - AMAT = Average Memory Access Time
  - $\text{CPI}_{\text{ALUOps}}$ does not include memory instructions

$$CPU\ time = \frac{IC \times \left(\frac{ALUops}{Inst} \times CPI_{ALUops} + \frac{MemAccess}{Inst} \times AMAT\right)}{Clock\ rate}$$

$$\begin{aligned} AMAT &= Hit\ time + Miss\ Rate \times Miss\ Penalty \\ &= \%\ instructions \times (Hit\ time_{Inst} + Miss\ Rate_{inst} \times Miss\ Penalty_{Inst}) \\ &+ \%\ data \times (Hit\ time_{Data} + Miss\ Rate_{Data} \times Miss\ Penalty_{Data}) \end{aligned}$$

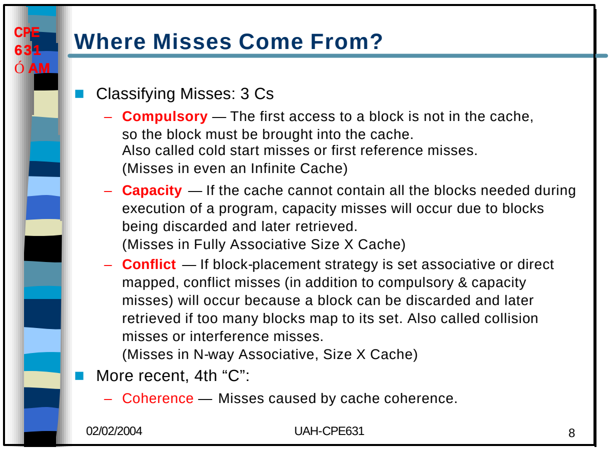


CPE 631 Ó AM

How to Improve Cache Performance?

$$AMAT = HitTime + MissRate \times MissPenalty$$

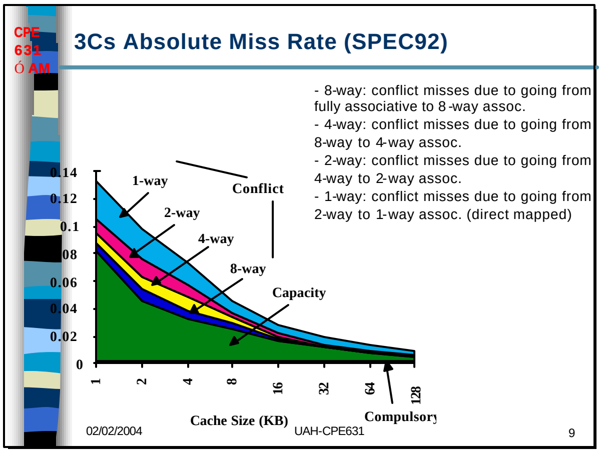
- Cache optimizations
  - 1. Reduce the miss rate
  - 2. Reduce the miss penalty
  - 3. Reduce the time to hit in the cache



CPE 631 Ó AM

Where Misses Come From?

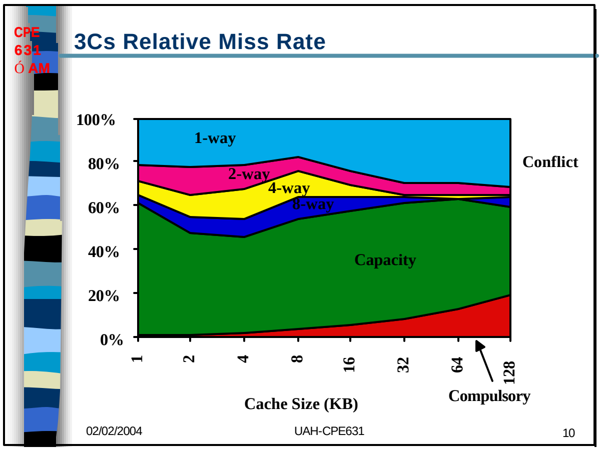
- Classifying Misses: 3 Cs
  - **Compulsory** — The first access to a block is not in the cache, so the block must be brought into the cache. Also called cold start misses or first reference misses. (Misses in even an Infinite Cache)
  - **Capacity** — If the cache cannot contain all the blocks needed during execution of a program, capacity misses will occur due to blocks being discarded and later retrieved. (Misses in Fully Associative Size X Cache)
  - **Conflict** — If block-placement strategy is set associative or direct mapped, conflict misses (in addition to compulsory & capacity misses) will occur because a block can be discarded and later retrieved if too many blocks map to its set. Also called collision misses or interference misses. (Misses in N-way Associative, Size X Cache)
- More recent, 4th "C":
  - Coherence — Misses caused by cache coherence.



3Cs Absolute Miss Rate (SPEC92)

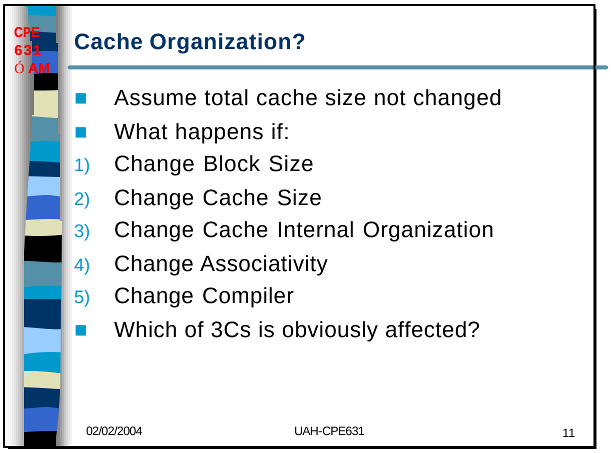
- 8-way: conflict misses due to going from fully associative to 8-way assoc.
- 4-way: conflict misses due to going from 8-way to 4-way assoc.
- 2-way: conflict misses due to going from 4-way to 2-way assoc.
- 1-way: conflict misses due to going from 2-way to 1-way assoc. (direct mapped)

02/02/2004 UAH-CPE631 9



3Cs Relative Miss Rate

02/02/2004 UAH-CPE631 10



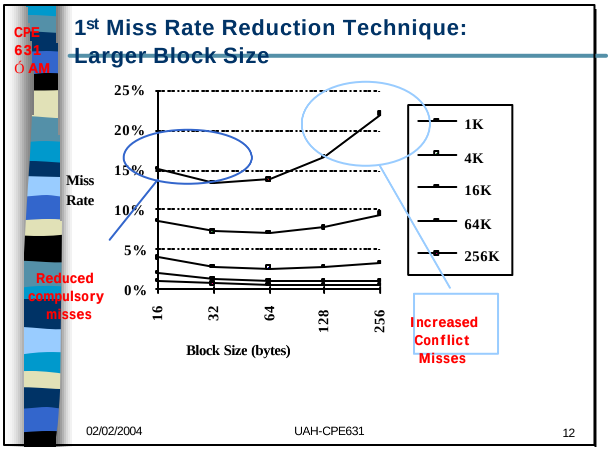
Cache Organization?

- Assume total cache size not changed
- What happens if:

1) Change Block Size
2) Change Cache Size
3) Change Cache Internal Organization
4) Change Associativity
5) Change Compiler

- Which of 3Cs is obviously affected?

02/02/2004 UAH-CPE631 11

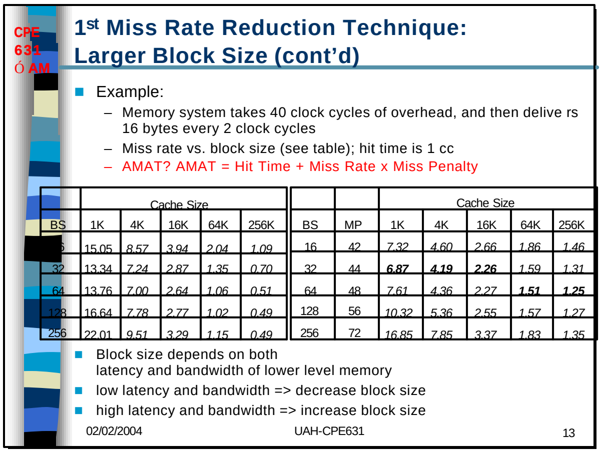


1st Miss Rate Reduction Technique: Larger Block Size

Reduced compulsory misses

Increased Conflict Misses

02/02/2004 UAH-CPE631 12



1st Miss Rate Reduction Technique: Larger Block Size (cont’d)

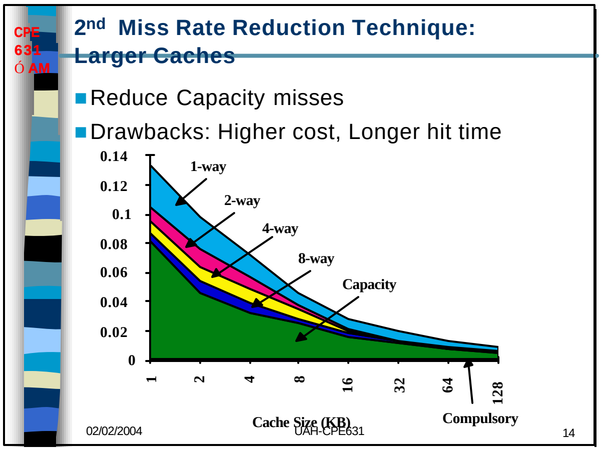
- Example:
  - Memory system takes 40 clock cycles of overhead, and then delive rs 16 bytes every 2 clock cycles
  - Miss rate vs. block size (see table); hit time is 1 cc
  - AMAT? AMAT = Hit Time + Miss Rate x Miss Penalty

	Cache Size				
BS	1K	4K	16K	64K	256K
16	*15.05*	*8.57*	*3.94*	*2.04*	*1.09*
32	*13.34*	*7.24*	*2.87*	*1.35*	*0.70*
64	*13.76*	*7.00*	*2.64*	*1.06*	*0.51*
128	*16.64*	*7.78*	*2.77*	*1.02*	*0.49*
256	*22.01*	*9.51*	*3.29*	*1.15*	*0.49*

		Cache Size				
BS	MP	1K	4K	16K	64K	256K
16	42	*7.32*	*4.60*	*2.66*	*1.86*	*1.46*
32	44	***6.87***	***4.19***	***2.26***	*1.59*	*1.31*
64	48	*7.61*	*4.36*	*2.27*	***1.51***	***1.25***
128	56	*10.32*	*5.36*	*2.55*	*1.57*	*1.27*
256	72	*16.85*	*7.85*	*3.37*	*1.83*	*1.35*

- Block size depends on both latency and bandwidth of lower level memory
- low latency and bandwidth => decrease block size
- high latency and bandwidth => increase block size

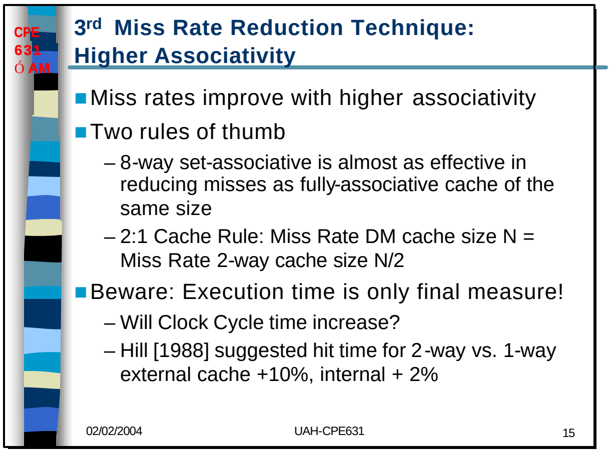
02/02/2004 UAH-CPE631 13



2nd Miss Rate Reduction Technique: Larger Caches

- Reduce Capacity misses
- Drawbacks: Higher cost, Longer hit time

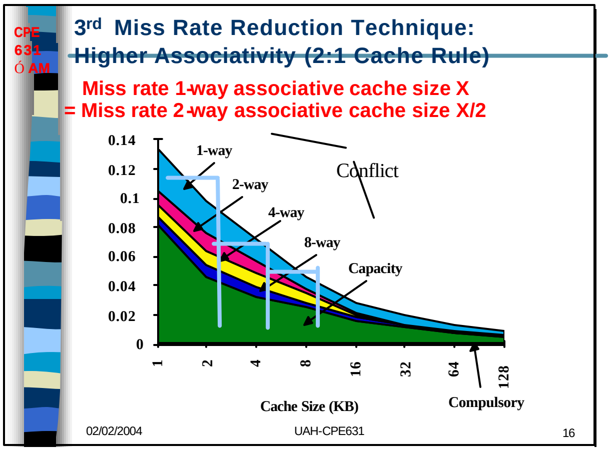
02/02/2004 UAH-CPE631 14



3rd Miss Rate Reduction Technique: Higher Associativity

- Miss rates improve with higher associativity
- Two rules of thumb
  - 8-way set-associative is almost as effective in reducing misses as fully-associative cache of the same size
  - 2:1 Cache Rule: Miss Rate DM cache size N = Miss Rate 2-way cache size N/2
- Beware: Execution time is only final measure!
  - Will Clock Cycle time increase?
  - Hill [1988] suggested hit time for 2-way vs. 1-way external cache +10%, internal + 2%

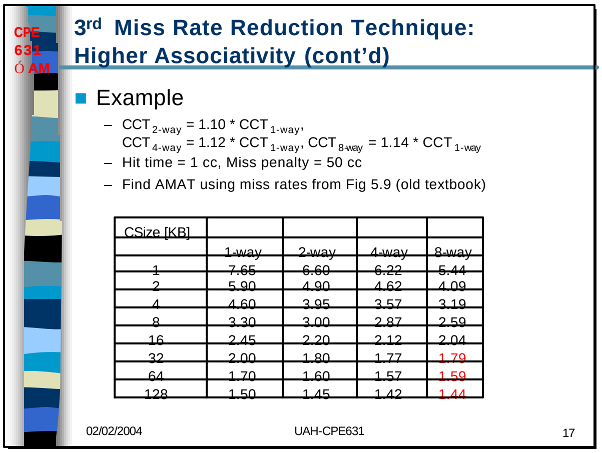
02/02/2004 UAH-CPE631 15



3rd Miss Rate Reduction Technique: Higher Associativity (2:1 Cache Rule)

Miss rate 1-way associative cache size X = Miss rate 2-way associative cache size X/2

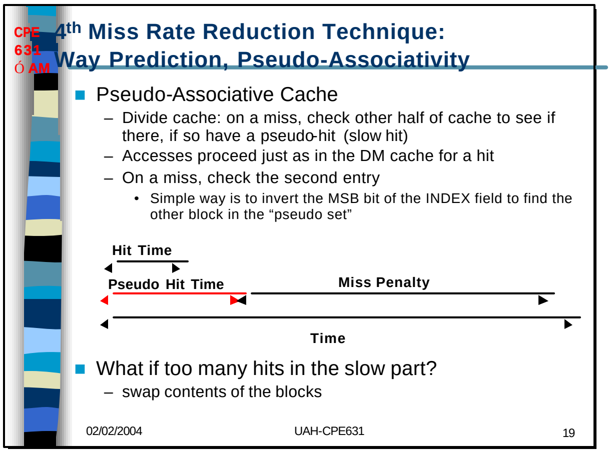
02/02/2004 UAH-CPE631 16



3rd Miss Rate Reduction Technique: Higher Associativity (cont'd)

- Example
  - $CCT_{2\text{-way}} = 1.10 * CCT_{1\text{-way}}$, $CCT_{4\text{-way}} = 1.12 * CCT_{1\text{-way}}$, $CCT_{8\text{-way}} = 1.14 * CCT_{1\text{-way}}$
  - Hit time = 1 cc, Miss penalty = 50 cc
  - Find AMAT using miss rates from Fig 5.9 (old textbook)

CSize [KB]				
	1-way	2-way	4-way	8-way
1	7.65	6.60	6.22	5.44
2	5.90	4.90	4.62	4.09
4	4.60	3.95	3.57	3.19
8	3.30	3.00	2.87	2.59
16	2.45	2.20	2.12	2.04
32	2.00	1.80	1.77	1.79
64	1.70	1.60	1.57	1.59
128	1.50	1.45	1.42	1.44



4th Miss Rate Reduction Technique: Way Prediction, Pseudo-Associativity

- Pseudo-Associative Cache
  - Divide cache: on a miss, check other half of cache to see if there, if so have a pseudo-hit (slow hit)
  - Accesses proceed just as in the DM cache for a hit
  - On a miss, check the second entry
    - Simple way is to invert the MSB bit of the INDEX field to find the other block in the "pseudo set"

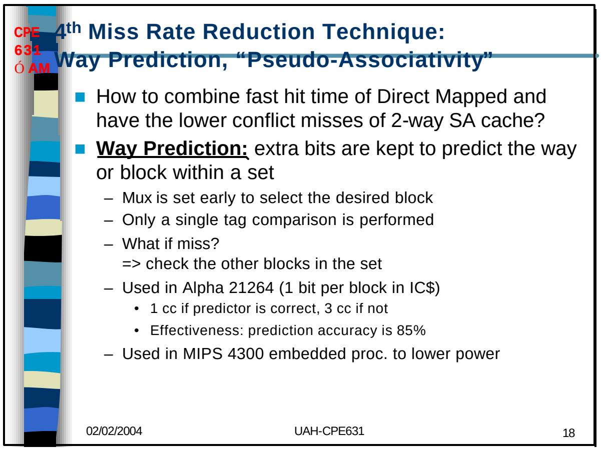
Hit Time

Pseudo Hit Time

Miss Penalty

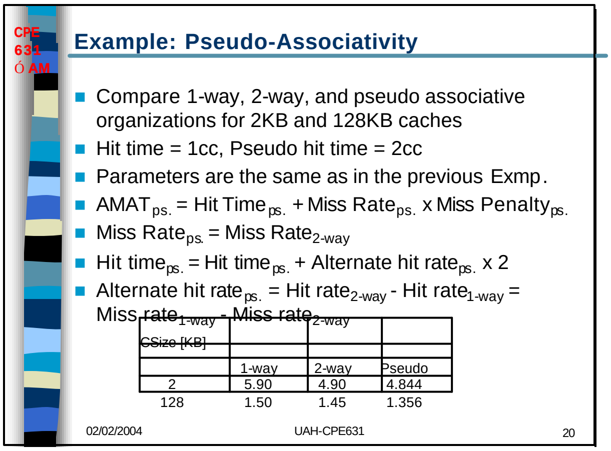
Time

- What if too many hits in the slow part?
  - swap contents of the blocks



4th Miss Rate Reduction Technique: Way Prediction, "Pseudo-Associativity"

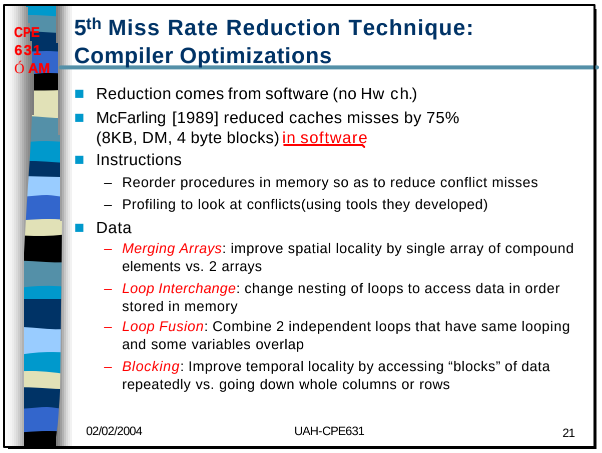
- How to combine fast hit time of Direct Mapped and have the lower conflict misses of 2-way SA cache?
- **Way Prediction:** extra bits are kept to predict the way or block within a set
  - Mux is set early to select the desired block
  - Only a single tag comparison is performed
  - What if miss?
    => check the other blocks in the set
  - Used in Alpha 21264 (1 bit per block in IC$)
    - 1 cc if predictor is correct, 3 cc if not
    - Effectiveness: prediction accuracy is 85%
  - Used in MIPS 4300 embedded proc. to lower power



Example: Pseudo-Associativity

- Compare 1-way, 2-way, and pseudo associative organizations for 2KB and 128KB caches
- Hit time = 1cc, Pseudo hit time = 2cc
- Parameters are the same as in the previous Exmp.
- $AMAT_{ps.} = \text{Hit Time}_{ps.} + \text{Miss Rate}_{ps.} \times \text{Miss Penalty}_{ps.}$
- $\text{Miss Rate}_{ps.} = \text{Miss Rate}_{2\text{-way}}$
- $\text{Hit time}_{ps.} = \text{Hit time}_{ps.} + \text{Alternate hit rate}_{ps.} \times 2$
- $\text{Alternate hit rate}_{ps.} = \text{Hit rate}_{2\text{-way}} - \text{Hit rate}_{1\text{-way}} = \text{Miss rate}_{1\text{-way}} - \text{Miss rate}_{2\text{-way}}$

CSize [KB]			
	1-way	2-way	Pseudo
2	5.90	4.90	4.844
128	1.50	1.45	1.356

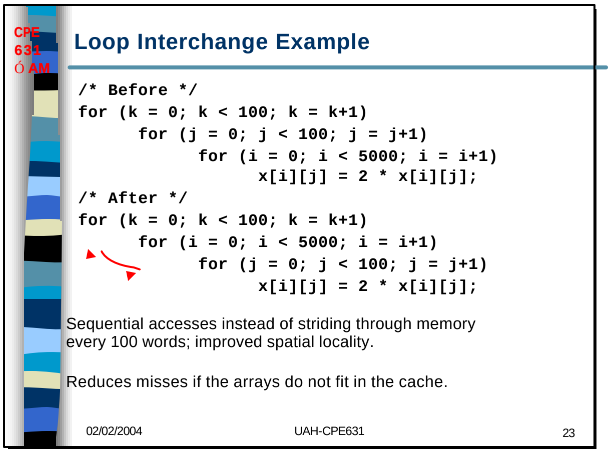


CPE 631 Ó AM

5th Miss Rate Reduction Technique: Compiler Optimizations

- Reduction comes from software (no Hw ch.)
- McFarling [1989] reduced caches misses by 75% (8KB, DM, 4 byte blocks) in software
- Instructions
  - Reorder procedures in memory so as to reduce conflict misses
  - Profiling to look at conflicts(using tools they developed)
- Data
  - *Merging Arrays*: improve spatial locality by single array of compound elements vs. 2 arrays
  - *Loop Interchange*: change nesting of loops to access data in order stored in memory
  - *Loop Fusion*: Combine 2 independent loops that have same looping and some variables overlap
  - *Blocking*: Improve temporal locality by accessing "blocks" of data repeatedly vs. going down whole columns or rows

02/02/2004 UAH-CPE631 21



CPE 631 Ó AM

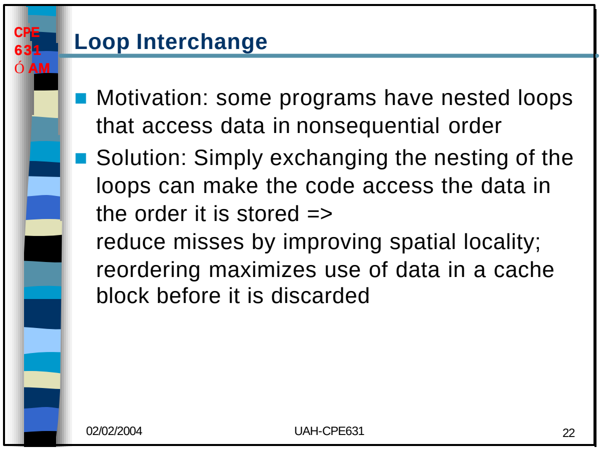
Loop Interchange Example

```
/* Before */
for (k = 0; k < 100; k = k+1)
     for (j = 0; j < 100; j = j+1)
          for (i = 0; i < 5000; i = i+1)
               x[i][j] = 2 * x[i][j];
/* After */
for (k = 0; k < 100; k = k+1)
     for (i = 0; i < 5000; i = i+1)
          for (j = 0; j < 100; j = j+1)
               x[i][j] = 2 * x[i][j];
```

Sequential accesses instead of striding through memory every 100 words; improved spatial locality.

Reduces misses if the arrays do not fit in the cache.

02/02/2004 UAH-CPE631 23

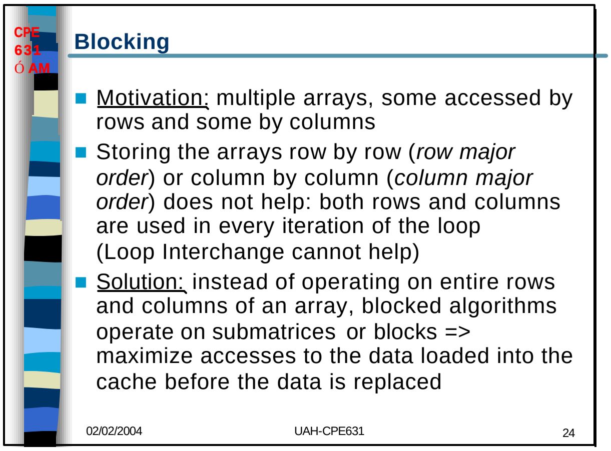


CPE 631 Ó AM

Loop Interchange

- Motivation: some programs have nested loops that access data in nonsequential order
- Solution: Simply exchanging the nesting of the loops can make the code access the data in the order it is stored => reduce misses by improving spatial locality; reordering maximizes use of data in a cache block before it is discarded

02/02/2004 UAH-CPE631 22

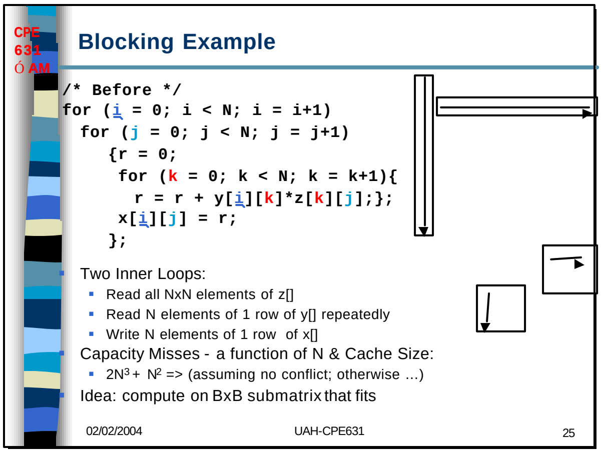


CPE 631 Ó AM

Blocking

- Motivation: multiple arrays, some accessed by rows and some by columns
- Storing the arrays row by row (*row major order*) or column by column (*column major order*) does not help: both rows and columns are used in every iteration of the loop (Loop Interchange cannot help)
- Solution: instead of operating on entire rows and columns of an array, blocked algorithms operate on submatrices or blocks => maximize accesses to the data loaded into the cache before the data is replaced

02/02/2004 UAH-CPE631 24

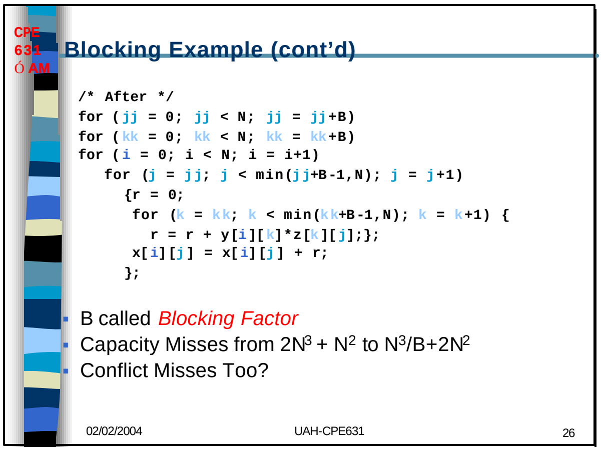


CPE 631 Ó AM

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;
     };
```

- Two Inner Loops:
  - Read all NxN elements of z[]
  - Read N elements of 1 row of y[] repeatedly
  - Write N elements of 1 row of x[]
- Capacity Misses - a function of N & Cache Size:
  - $2N^3 + N^2$ => (assuming no conflict; otherwise …)
- Idea: compute on BxB submatrix that fits

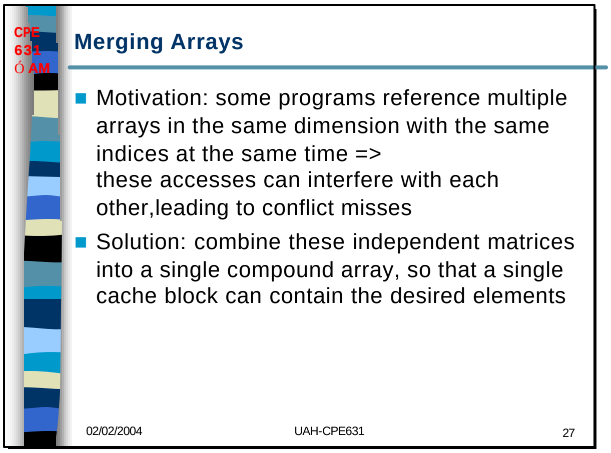


CPE 631 Ó AM

Blocking Example (cont'd)

```
/* 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;
      };
```

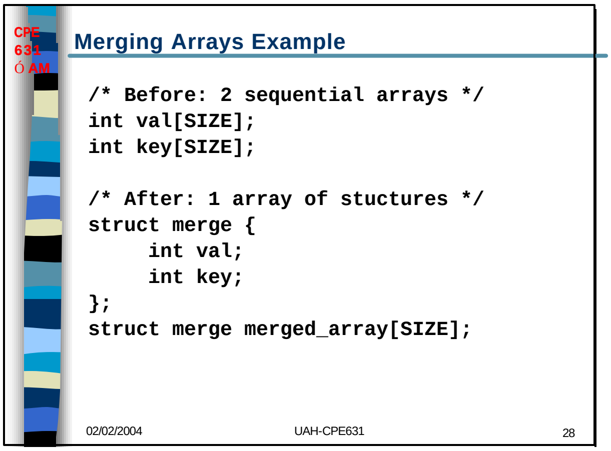
- B called *Blocking Factor*
- Capacity Misses from $2N^3 + N^2$ to $N^3/B+2N^2$
- Conflict Misses Too?



CPE 631 Ó AM

Merging Arrays

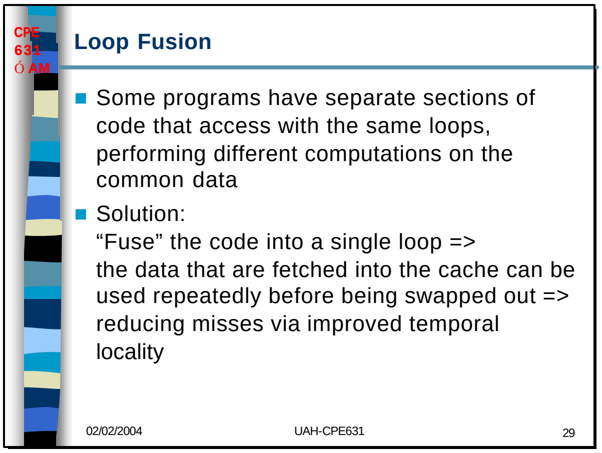
- Motivation: some programs reference multiple arrays in the same dimension with the same indices at the same time =>
  these accesses can interfere with each other,leading to conflict misses
- Solution: combine these independent matrices into a single compound array, so that a single cache block can contain the desired elements



CPE 631 Ó AM

Merging Arrays Example

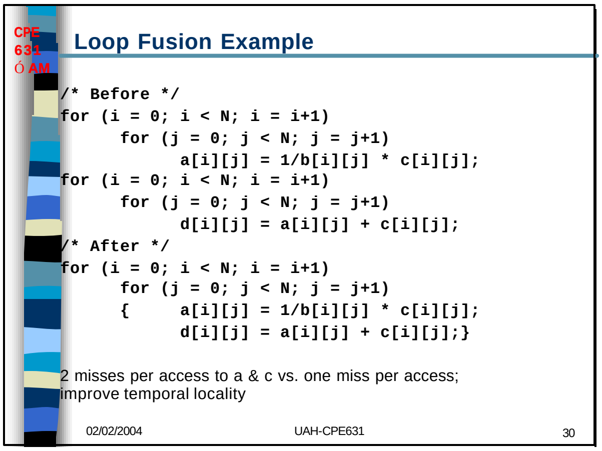
```
/* Before: 2 sequential arrays */
int val[SIZE];
int key[SIZE];

/* After: 1 array of stuctures */
struct merge {
     int val;
     int key;
};
struct merge merged_array[SIZE];
```



CPE 631 Ó AM

Loop Fusion

- Some programs have separate sections of code that access with the same loops, performing different computations on the common data
- Solution:
  “Fuse” the code into a single loop =>
  the data that are fetched into the cache can be used repeatedly before being swapped out => reducing misses via improved temporal locality

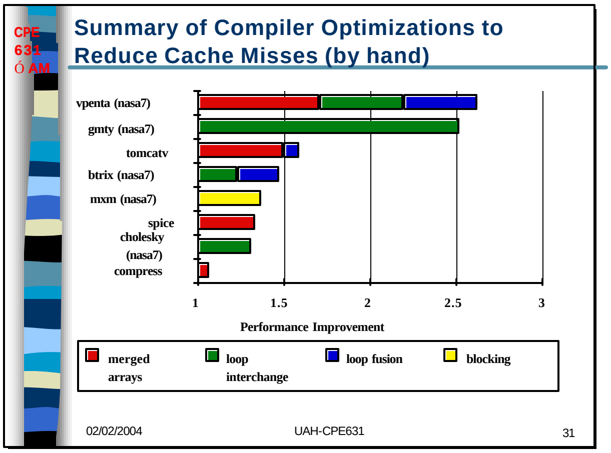


CPE 631 Ó AM

Loop Fusion Example

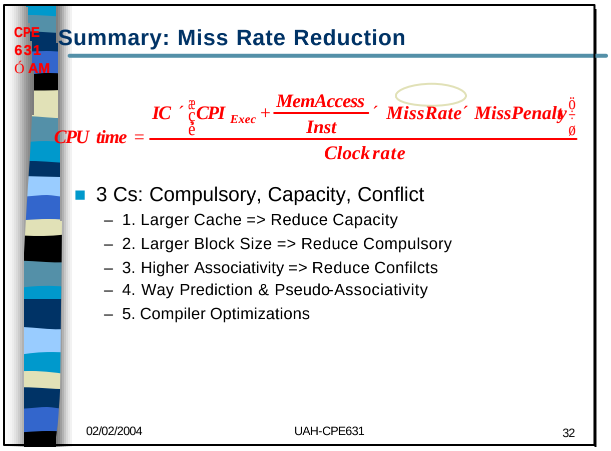
```
/* Before */
for (i = 0; i < N; i = i+1)
      for (j = 0; j < N; j = j+1)
            a[i][j] = 1/b[i][j] * c[i][j];
for (i = 0; i < N; i = i+1)
      for (j = 0; j < N; j = j+1)
            d[i][j] = a[i][j] + c[i][j];
/* After */
for (i = 0; i < N; i = i+1)
      for (j = 0; j < N; j = j+1)
      {     a[i][j] = 1/b[i][j] * c[i][j];
            d[i][j] = a[i][j] + c[i][j];}
```

2 misses per access to a & c vs. one miss per access; improve temporal locality



CPE 631 Ó AM

Summary of Compiler Optimizations to Reduce Cache Misses (by hand)



CPE 631 Ó AM

Summary: Miss Rate Reduction

$$CPU\ time = \frac{IC \times \left(CPI_{Exec} + \frac{MemAccess}{Inst} \times MissRate \times MissPenalty\right)}{Clockrate}$$

- 3 Cs: Compulsory, Capacity, Conflict
  - 1. Larger Cache => Reduce Capacity
  - 2. Larger Block Size => Reduce Compulsory
  - 3. Higher Associativity => Reduce Confilcts
  - 4. Way Prediction & Pseudo-Associativity
  - 5. Compiler Optimizations

CPE 631 Ó AM

Reducing Miss Penalty

- Motivation
  - AMAT = Hit Time + Miss Rate x Miss Penalty
  - Technology trends => relative cost of miss penalties increases over time
- Techniques that address miss penalties
  - 1. Multilevel Caches
  - 2. Critical Word First and Early Restart
  - 3. Giving Priority to Read Misses over Writes
  - 4. Merging Write Buffer
  - 5. Victim Caches

CPE 631 Ó AM

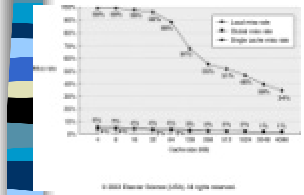
1st Miss Penalty Reduction Technique: Multilevel Caches

- Architect's dilemma
  - Should I make the cache faster to keep pace with the speed of CPUs
  - Should I make the cache larger to overcome the widening gap between CPU and main memory
- L2 Equations
  - AMAT = Hit Time$_{L1}$ + Miss Rate$_{L1}$ x Miss Penalty$_{L1}$
  - Miss Penalty$_{L1}$ = Hit Time$_{L2}$ + Miss Rate$_{L2}$ x Miss Penalty$_{L2}$
  - AMAT = Hit Time$_{L1}$ + Miss Rate$_{L1}$ x (Hit Time$_{L2}$ + Miss Rate$_{L2}$ + Miss Penalty$_{L2}$)
- Definitions:
  - Local miss rate— misses in this cache divided by the total number of memory accesses to this cache (Miss rate$_{L2}$)
  - Global miss rate—misses in this cache divided by the total number of memory accesses generated by the CPU (Miss Rate$_{L1}$ x Miss Rate$_{L2}$)
  - Global Miss Rate is what matters

CPE 631 Ó AM

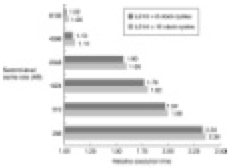
1st Miss Penalty Reduction Technique: Multilevel Caches

- Global vs. Local Miss Rate



© 2003 Elsevier Science (USA). All rights reserved.

- Relative Execution Time
  - 1.0 is 8MB L2, 1cc hit

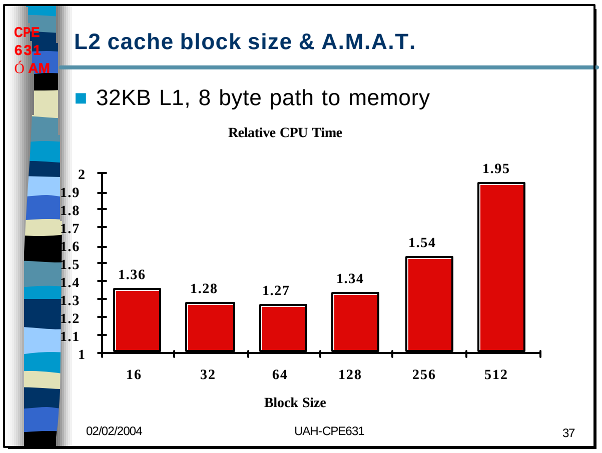


© 2003 Elsevier Science (USA). All rights reserved.

CPE 631 Ó AM

Reducing Misses: Which apply to L2 Cache?

- Reducing Miss Rate
  - 1. Reduce Capacity Misses via Larger Cache
  - 2. Reduce Compulsory Misses via Larger Block Size
  - 3. Reduce Conflict Misses via Higher Associativity
  - 4. Reduce Conflict Misses via Way Prediction & Pseudo-Associativity
  - 5. Reduce Conflict/Capac. Misses via Compiler Optimizations

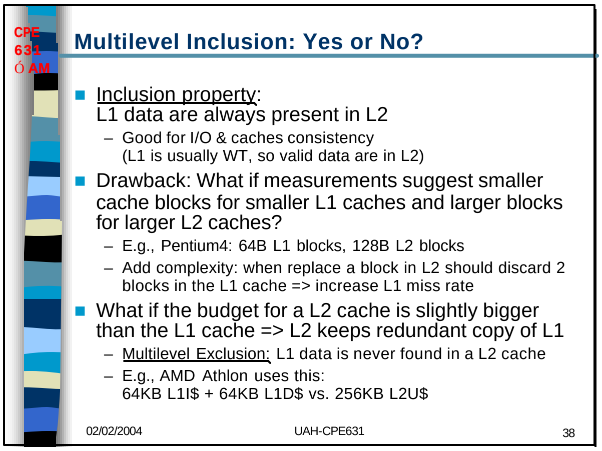


CPE 631 Ó AM

L2 cache block size & A.M.A.T.

- 32KB L1, 8 byte path to memory

02/02/2004 UAH-CPE631 37

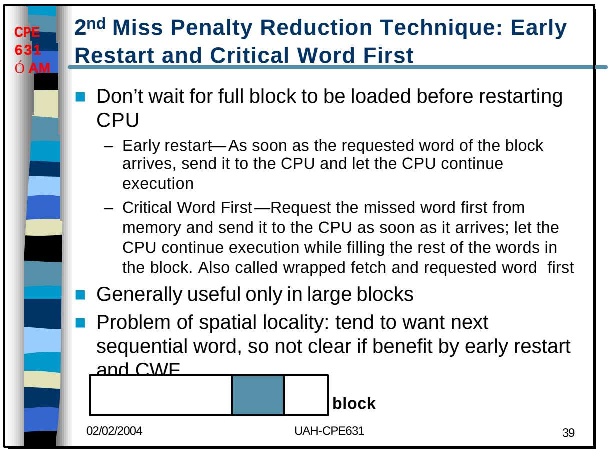


CPE 631 Ó AM

Multilevel Inclusion: Yes or No?

- Inclusion property: L1 data are always present in L2
  - Good for I/O & caches consistency (L1 is usually WT, so valid data are in L2)
- Drawback: What if measurements suggest smaller cache blocks for smaller L1 caches and larger blocks for larger L2 caches?
  - E.g., Pentium4: 64B L1 blocks, 128B L2 blocks
  - Add complexity: when replace a block in L2 should discard 2 blocks in the L1 cache => increase L1 miss rate
- What if the budget for a L2 cache is slightly bigger than the L1 cache => L2 keeps redundant copy of L1
  - Multilevel Exclusion: L1 data is never found in a L2 cache
  - E.g., AMD Athlon uses this: 64KB L1I$ + 64KB L1D$ vs. 256KB L2U$

02/02/2004 UAH-CPE631 38

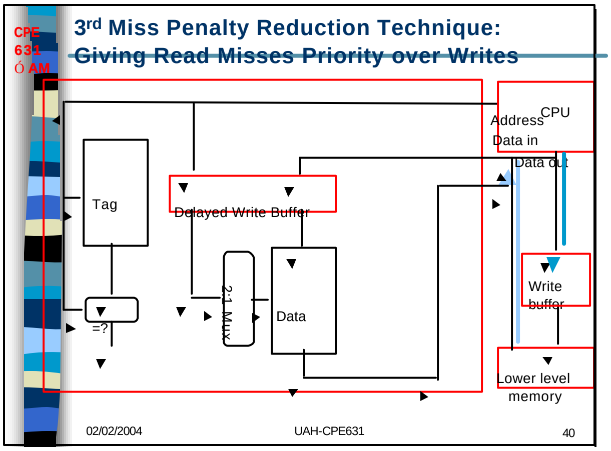


CPE 631 Ó AM

2nd Miss Penalty Reduction Technique: Early Restart and Critical Word First

- Don’t wait for full block to be loaded before restarting CPU
  - Early restart—As soon as the requested word of the block arrives, send it to the CPU and let the CPU continue execution
  - Critical Word First—Request the missed word first from memory and send it to the CPU as soon as it arrives; let the CPU continue execution while filling the rest of the words in the block. Also called wrapped fetch and requested word first
- Generally useful only in large blocks
- Problem of spatial locality: tend to want next sequential word, so not clear if benefit by early restart and CWF

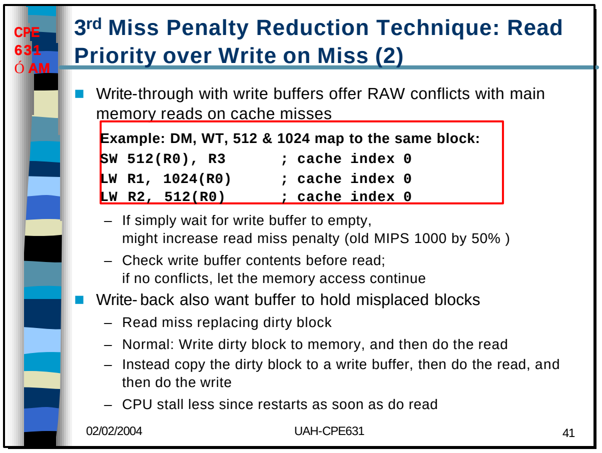
02/02/2004 UAH-CPE631 39



CPE 631 Ó AM

3rd Miss Penalty Reduction Technique: Giving Read Misses Priority over Writes

02/02/2004 UAH-CPE631 40



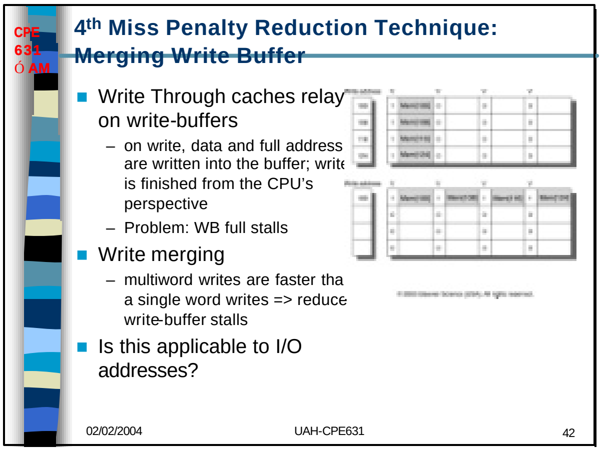
CPE 631 Ó AM

3rd Miss Penalty Reduction Technique: Read Priority over Write on Miss (2)

- Write-through with write buffers offer RAW conflicts with main memory reads on cache misses

```
Example: DM, WT, 512 & 1024 map to the same block:
SW 512(R0), R3      ; cache index 0
LW R1, 1024(R0)     ; cache index 0
LW R2, 512(R0)      ; cache index 0
```

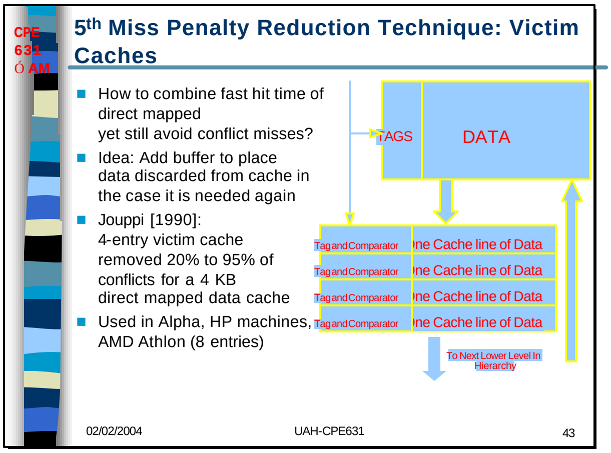
  - If simply wait for write buffer to empty, might increase read miss penalty (old MIPS 1000 by 50%)
  - Check write buffer contents before read; if no conflicts, let the memory access continue
- Write-back also want buffer to hold misplaced blocks
  - Read miss replacing dirty block
  - Normal: Write dirty block to memory, and then do the read
  - Instead copy the dirty block to a write buffer, then do the read, and then do the write
  - CPU stall less since restarts as soon as do read



CPE 631 Ó AM

4th Miss Penalty Reduction Technique: Merging Write Buffer

- Write Through caches relay on write-buffers
  - on write, data and full address are written into the buffer; write is finished from the CPU's perspective
  - Problem: WB full stalls
- Write merging
  - multiword writes are faster tha a single word writes => reduce write-buffer stalls
- Is this applicable to I/O addresses?



CPE 631 Ó AM

5th Miss Penalty Reduction Technique: Victim Caches

- How to combine fast hit time of direct mapped yet still avoid conflict misses?
- Idea: Add buffer to place data discarded from cache in the case it is needed again
- Jouppi [1990]: 4-entry victim cache removed 20% to 95% of conflicts for a 4 KB direct mapped data cache
- Used in Alpha, HP machines, AMD Athlon (8 entries)

TAGS

DATA

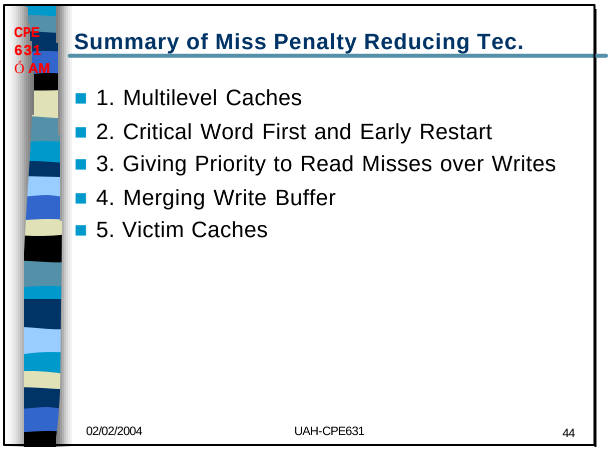
Tag and Comparator | ne Cache line of Data

Tag and Comparator | ne Cache line of Data

Tag and Comparator | ne Cache line of Data

Tag and Comparator | ne Cache line of Data

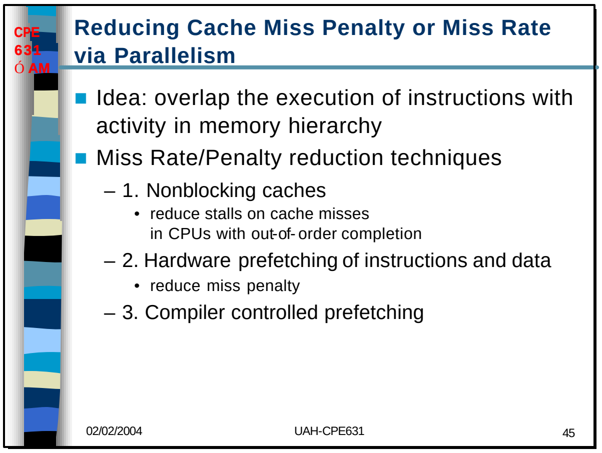
To Next Lower Level In Hierarchy



CPE 631 Ó AM

Summary of Miss Penalty Reducing Tec.

- 1. Multilevel Caches
- 2. Critical Word First and Early Restart
- 3. Giving Priority to Read Misses over Writes
- 4. Merging Write Buffer
- 5. Victim Caches

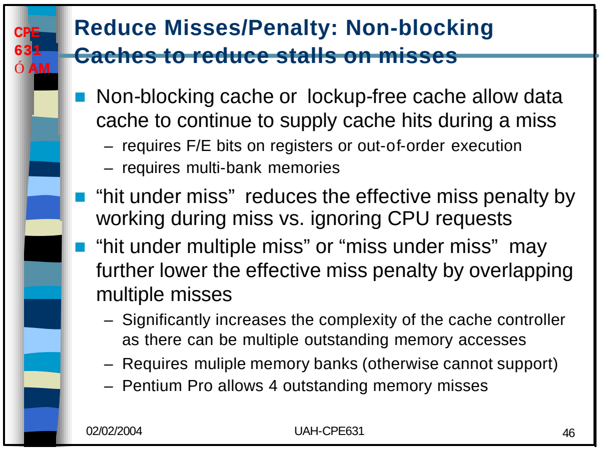


CPE 631 Ó AM

Reducing Cache Miss Penalty or Miss Rate via Parallelism

- Idea: overlap the execution of instructions with activity in memory hierarchy
- Miss Rate/Penalty reduction techniques
  - 1. Nonblocking caches
    - reduce stalls on cache misses in CPUs with out-of-order completion
  - 2. Hardware prefetching of instructions and data
    - reduce miss penalty
  - 3. Compiler controlled prefetching

02/02/2004 UAH-CPE631 45

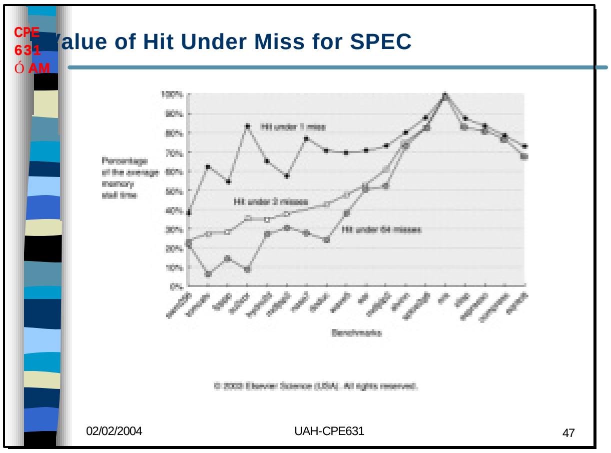


CPE 631 Ó AM

Reduce Misses/Penalty: Non-blocking Caches to reduce stalls on misses

- Non-blocking cache or lockup-free cache allow data cache to continue to supply cache hits during a miss
  - requires F/E bits on registers or out-of-order execution
  - requires multi-bank memories
- "hit under miss" reduces the effective miss penalty by working during miss vs. ignoring CPU requests
- "hit under multiple miss" or "miss under miss" may further lower the effective miss penalty by overlapping multiple misses
  - Significantly increases the complexity of the cache controller as there can be multiple outstanding memory accesses
  - Requires muliple memory banks (otherwise cannot support)
  - Pentium Pro allows 4 outstanding memory misses

02/02/2004 UAH-CPE631 46

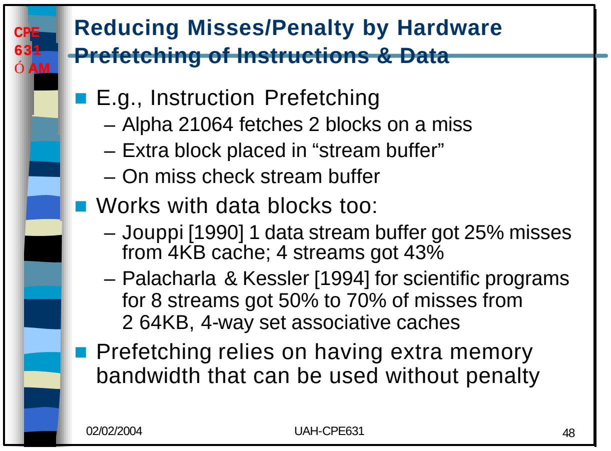


CPE 631 Ó AM

alue of Hit Under Miss for SPEC

© 2003 Elsevier Science (USA). All rights reserved.

02/02/2004 UAH-CPE631 47

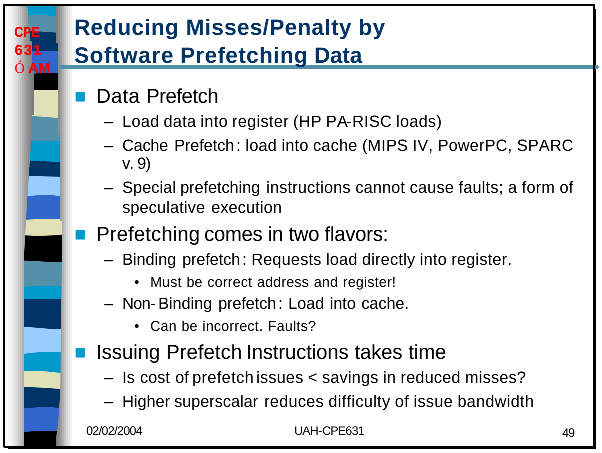


CPE 631 Ó AM

Reducing Misses/Penalty by Hardware Prefetching of Instructions & Data

- E.g., Instruction Prefetching
  - Alpha 21064 fetches 2 blocks on a miss
  - Extra block placed in "stream buffer"
  - On miss check stream buffer
- Works with data blocks too:
  - Jouppi [1990] 1 data stream buffer got 25% misses from 4KB cache; 4 streams got 43%
  - Palacharla & Kessler [1994] for scientific programs for 8 streams got 50% to 70% of misses from 2 64KB, 4-way set associative caches
- Prefetching relies on having extra memory bandwidth that can be used without penalty

02/02/2004 UAH-CPE631 48

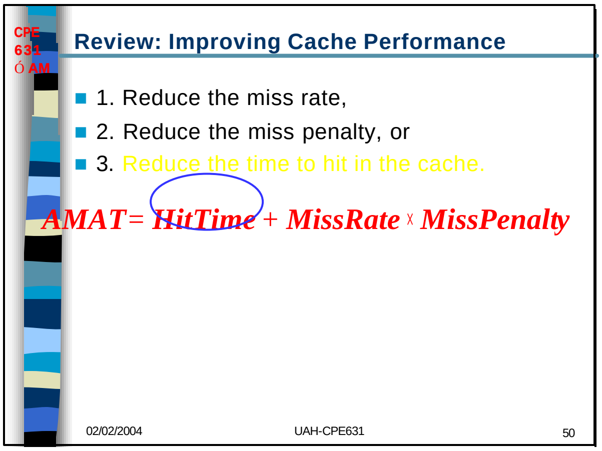


CPE 631 Ó AM

Reducing Misses/Penalty by Software Prefetching Data

- Data Prefetch
  - Load data into register (HP PA-RISC loads)
  - Cache Prefetch: load into cache (MIPS IV, PowerPC, SPARC v. 9)
  - Special prefetching instructions cannot cause faults; a form of speculative execution
- Prefetching comes in two flavors:
  - Binding prefetch: Requests load directly into register.
    - Must be correct address and register!
  - Non-Binding prefetch: Load into cache.
    - Can be incorrect. Faults?
- Issuing Prefetch Instructions takes time
  - Is cost of prefetch issues < savings in reduced misses?
  - Higher superscalar reduces difficulty of issue bandwidth

02/02/2004 UAH-CPE631 49



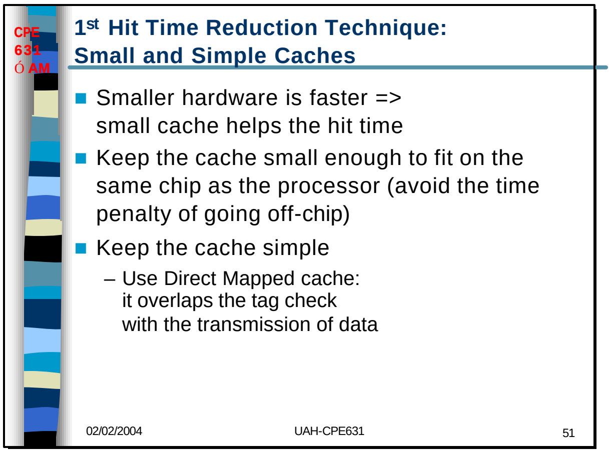
CPE 631 Ó AM

Review: Improving Cache Performance

- 1. Reduce the miss rate,
- 2. Reduce the miss penalty, or
- 3. Reduce the time to hit in the cache.

$$AMAT = HitTime + MissRate \times MissPenalty$$

02/02/2004 UAH-CPE631 50

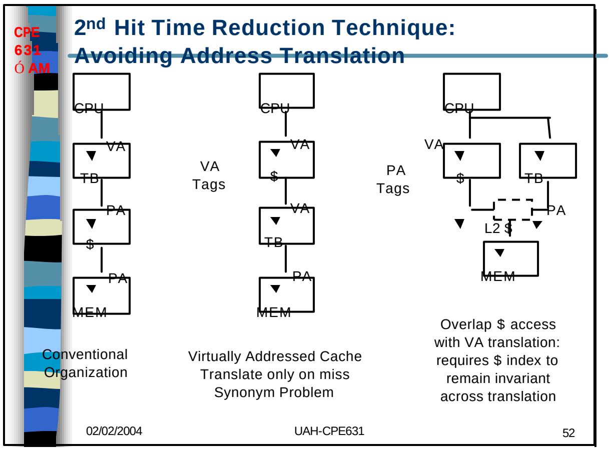


CPE 631 Ó AM

1st Hit Time Reduction Technique: Small and Simple Caches

- Smaller hardware is faster => small cache helps the hit time
- Keep the cache small enough to fit on the same chip as the processor (avoid the time penalty of going off-chip)
- Keep the cache simple
  - Use Direct Mapped cache: it overlaps the tag check with the transmission of data

02/02/2004 UAH-CPE631 51



CPE 631 Ó AM

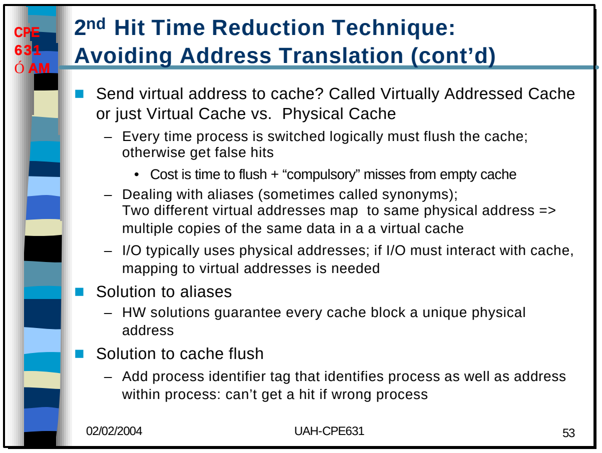
2nd Hit Time Reduction Technique: Avoiding Address Translation

Conventional Organization

Virtually Addressed Cache Translate only on miss Synonym Problem

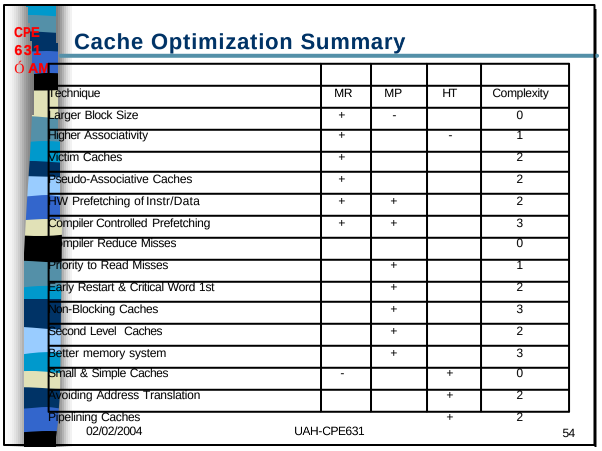
Overlap $ access with VA translation: requires $ index to remain invariant across translation

02/02/2004 UAH-CPE631 52



2nd Hit Time Reduction Technique: Avoiding Address Translation (cont’d)

- Send virtual address to cache? Called Virtually Addressed Cache or just Virtual Cache vs. Physical Cache
  - Every time process is switched logically must flush the cache; otherwise get false hits
    - Cost is time to flush + “compulsory” misses from empty cache
  - Dealing with aliases (sometimes called synonyms); Two different virtual addresses map to same physical address => multiple copies of the same data in a a virtual cache
  - I/O typically uses physical addresses; if I/O must interact with cache, mapping to virtual addresses is needed
- Solution to aliases
  - HW solutions guarantee every cache block a unique physical address
- Solution to cache flush
  - Add process identifier tag that identifies process as well as address within process: can’t get a hit if wrong process



Cache Optimization Summary

Technique	MR	MP	HT	Complexity
Larger Block Size	+	-		0
Higher Associativity	+		-	1
Victim Caches	+			2
Pseudo-Associative Caches	+			2
HW Prefetching of Instr/Data	+	+		2
Compiler Controlled Prefetching	+	+		3
Compiler Reduce Misses				0
Priority to Read Misses		+		1
Early Restart & Critical Word 1st		+		2
Non-Blocking Caches		+		3
Second Level Caches		+		2
Better memory system		+		3
Small & Simple Caches	-		+	0
Avoiding Address Translation			+	2
Pipelining Caches			+	2