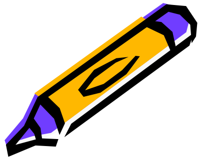
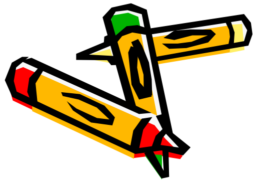
# 5008: Computer Architecture

## Chapter 5 - Memory Hierarchy Design

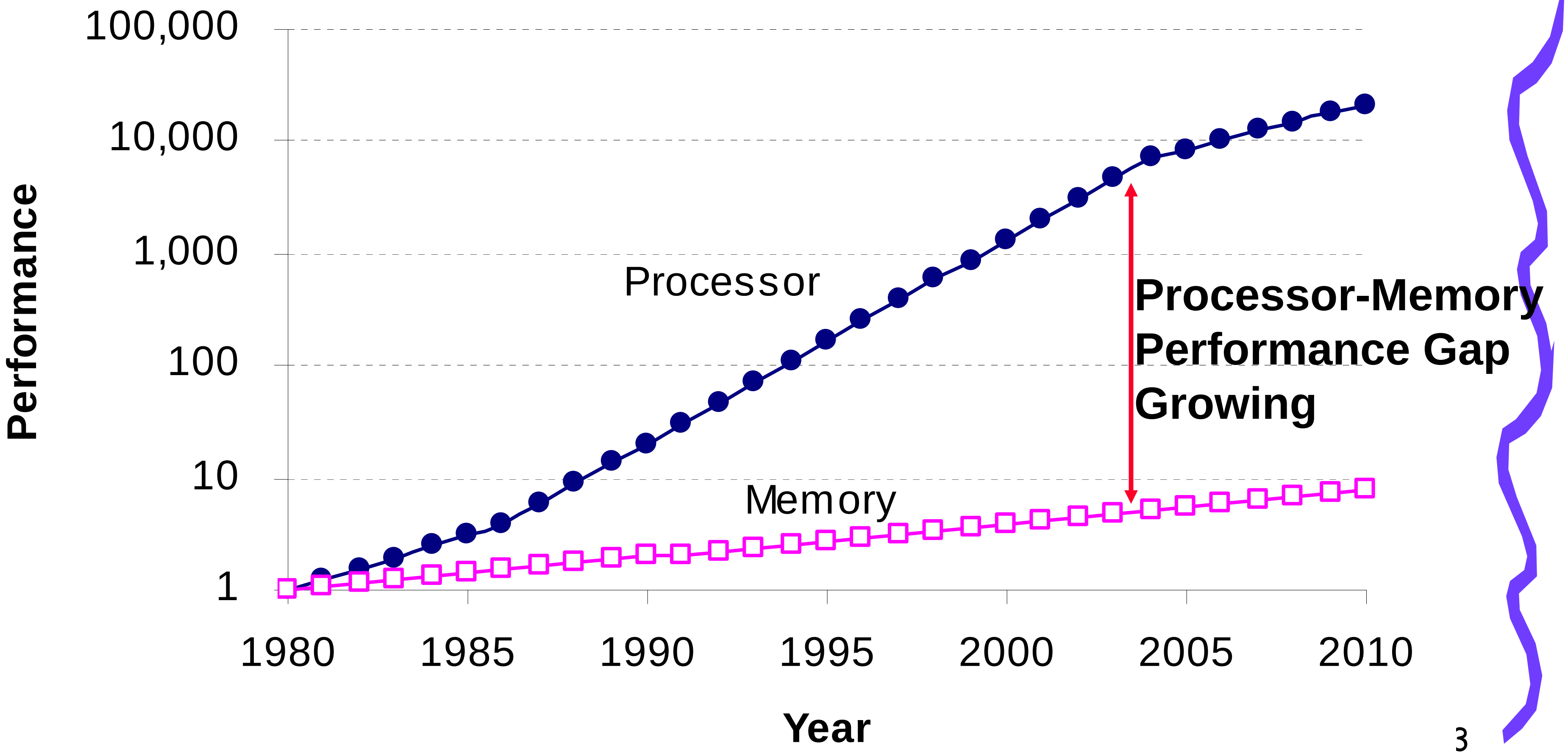


# Outline

- 11 Advanced Cache Optimizations
- Memory Technology and DRAM Optimizations
- Virtual Machines
- Conclusion



# Why More on Memory Hierarchy?



# Review: 6 Basic Cache Optimizations

- Reducing hit time

1. Giving Reads Priority over Writes
   - E.g., Read complete before earlier writes in write buffer
2. Avoiding Address Translation during Cache Indexing

- Reducing Miss Penalty

3. Multilevel Caches

- Reducing Miss Rate

4. Larger Block size (Compulsory misses)
5. Larger Cache size (Capacity misses)
6. Higher Associativity (Conflict misses)

# 11 Advanced Cache Optimizations

- Reducing hit time

1. Small and simple caches
2. Way prediction
3. Trace caches

- Increasing cache bandwidth

4. Pipelined caches
5. Multibanked caches
6. Nonblocking caches

- Reducing Miss Penalty

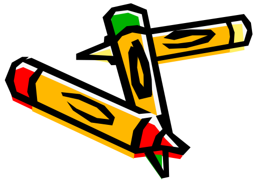
7. Critical word first
8. Merging write buffers

- Reducing Miss Rate

9. Compiler optimizations

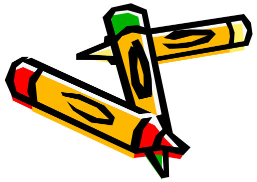
- Reducing miss penalty or miss rate via parallelism

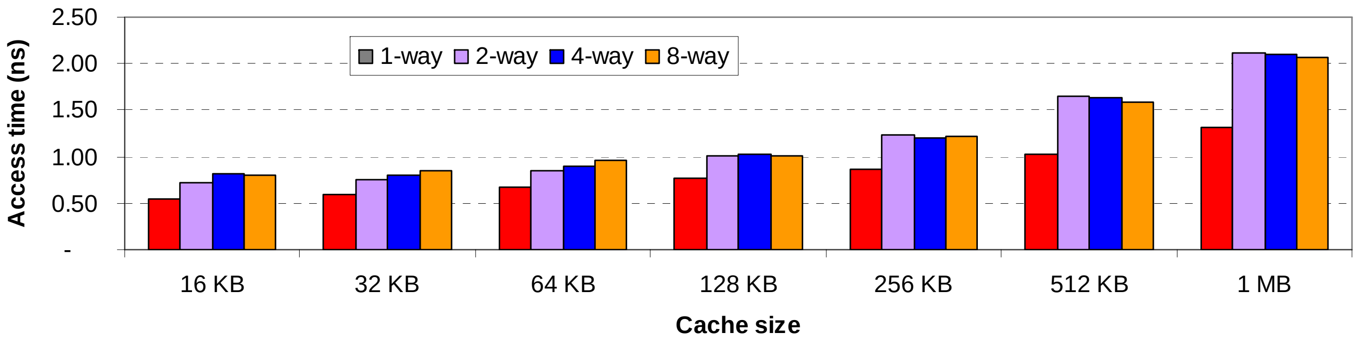
10. Hardware prefetching
11. Compiler prefetching



# 1. Fast Hit times via Small and Simple Caches

- Index tag memory and then compare takes time
- ⇒ Small cache can help hit time since smaller memory takes less time to index
  - E.g., L1 caches same size for 3 generations of AMD microprocessors: K6, Athlon, and Opteron
  - Also L2 cache small enough to fit on chip with the processor avoids time penalty of going off chip
- Simple ⇒ direct mapping
  - Can overlap tag check with data transmission since no choice
- Access time estimate for 90 nm using CACTI model 4.0
  - Median ratios of access time relative to the direct-mapped caches are 1.32, 1.39, and 1.43 for 2-way, 4-way, and 8-way caches

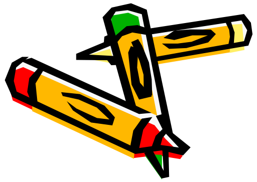




# 2. Fast Hit times via Way Prediction

- How to combine fast hit time of Direct Mapped and have the lower conflict misses of 2-way SA cache?
- Way prediction: keep extra bits in cache to predict the "way," or block within the set, of next cache access.
  - Multiplexor is set early to select desired block, only 1 tag comparison performed that clock cycle in parallel with reading the cache data
  - Miss ⇒ 1st check other blocks for matches in next clock cycle

- Accuracy ≈ 85%
- Drawback: CPU pipeline is hard if hit takes 1 or 2 cycles
  - Used for instruction caches vs. data caches



# 3. Fast Hit times via Trace Cache

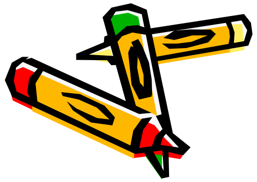
- Find more instruction level parallelism?
  How avoid translation from x86 to microops?
- Trace cache in Pentium 4

1. Dynamic traces of the executed instructions vs. static sequences of instructions as determined by layout in memory
   - Built-in branch predictor
2. Cache the micro-ops vs. x86 instructions
   - Decode/translate from x86 to micro-ops on trace cache miss

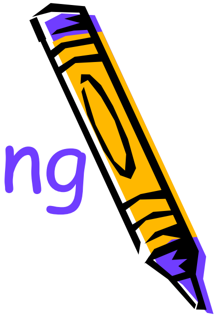
\+ 1. ⇒ better utilize long blocks (don't exit in middle of block, don't enter at label in middle of block)

\- 1. ⇒ complicated address mapping since addresses no longer aligned to power-of-2 multiples of word size

\- 1. ⇒ instructions may appear multiple times in multiple dynamic traces due to different branch outcomes



# 4: Increasing Cache Bandwidth by Pipelining



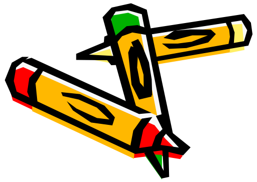
- Pipeline cache access to maintain bandwidth, but higher latency
- Instruction cache access pipeline stages:

  1: Pentium

  2: Pentium Pro through Pentium III

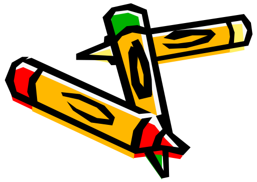
  4: Pentium 4

- ⇒ greater penalty on mispredicted branches
- ⇒ more clock cycles between the issue of the load and the use of the data



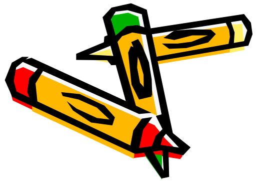
# 5. Increasing Cache Bandwidth: Non-Blocking Caches

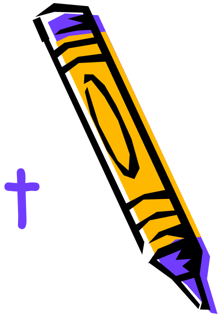
- *Non-blocking cache* or *lockup-free cache* allow data cache to continue to supply cache hits during a miss
  - requires F/E bits on registers or out-of-order execution
  - requires multi-bank memories
- "*hit under miss*" reduces the effective miss penalty by working during miss vs. ignoring CPU requests
- "*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)
  - Penium Pro allows 4 outstanding memory misses



# 6: Increasing Cache Bandwidth via Multiple Banks

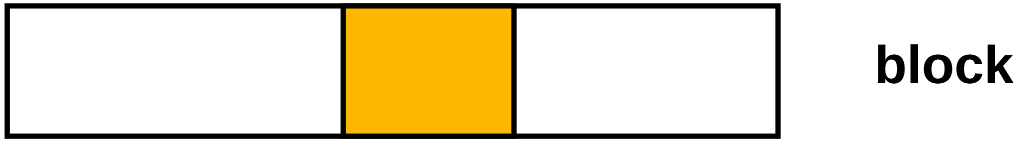
- Rather than treat the cache as a single monolithic block, divide into independent banks that can support simultaneous accesses
  - E.g.,T1 (“Niagara”) L2 has 4 banks
- Banking works best when accesses naturally spread themselves across banks ⇒ mapping of addresses to banks affects behavior of memory system
- Simple mapping that works well is “sequential interleaving”
  - Spread block addresses sequentially across banks
  - E,g, if there 4 banks, Bank 0 has all blocks whose address modulo 4 is 0; bank 1 has all blocks whose address modulo 4 is 1; …

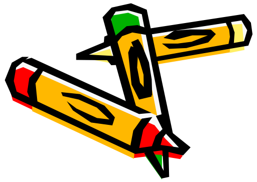




# 7. Reduce Miss Penalty: Early Restart and Critical Word First

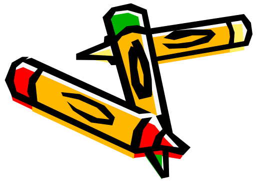
- Don’t wait for full block before restarting CPU
- *Early restart*—As soon as the requested word of the block arrives, send it to the CPU and let the CPU continue execution
  - Spatial locality ⇒ tend to want next sequential word, so not clear size of benefit of just early restart
- *Critical Word First*—Request 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
  - Long blocks more popular today ⇒ Critical Word 1st Widely used





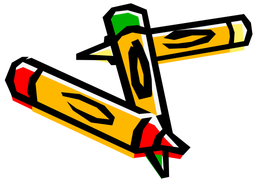
# 8. Merging Write Buffer to Reduce Miss Penalty

- Write buffer to allow processor to continue while waiting to write to memory
- If buffer contains modified blocks, the addresses can be checked to see if address of new data matches the address of a valid write buffer entry
- If so, new data are combined with that entry
- Increases block size of write for write-through cache of writes to sequential words, bytes since multiword writes more efficient to memory
- The Sun T1 (Niagara) processor, among many others, uses write merging



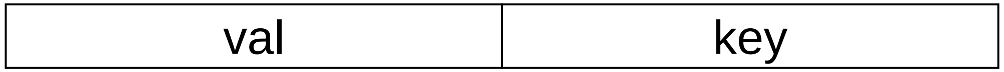
# 9. Reducing Misses by Compiler Optimizations

- McFarling [1989] reduced caches misses by 75%
  on 8KB direct mapped cache, 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



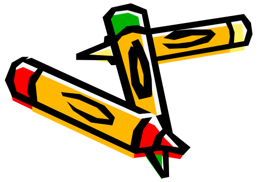
# Merging Arrays Example

```
/* 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];
```

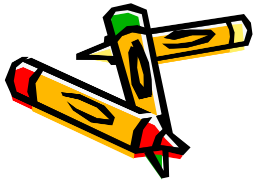
Reducing conflicts between val & key; improve spatial locality



# 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

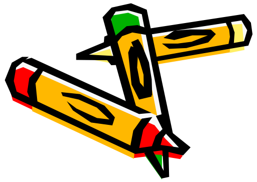


# Loop Fusion Example

```
/* 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];}
```

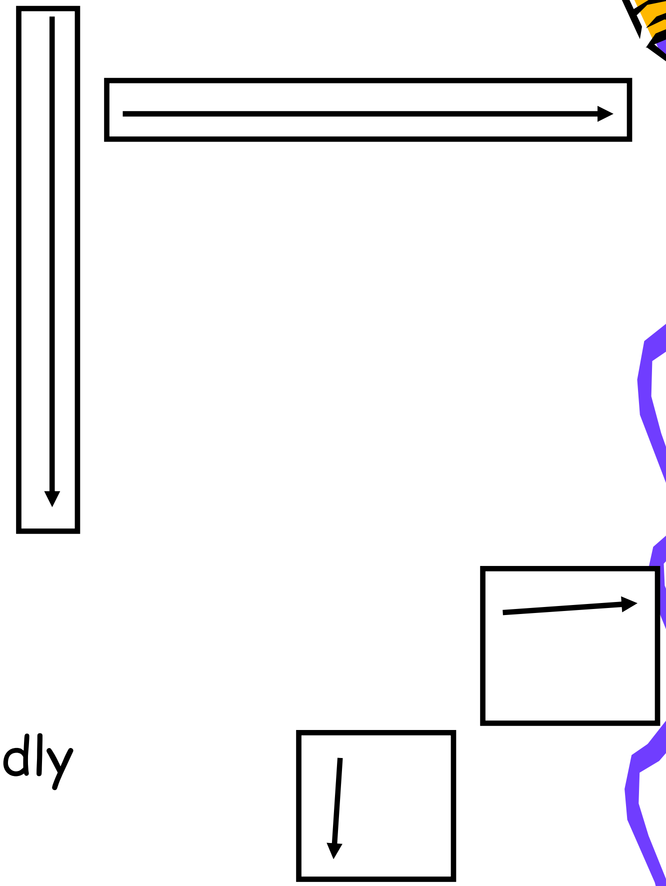
Perform different computations on the common data in two loops ➔ fuse the two loops

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



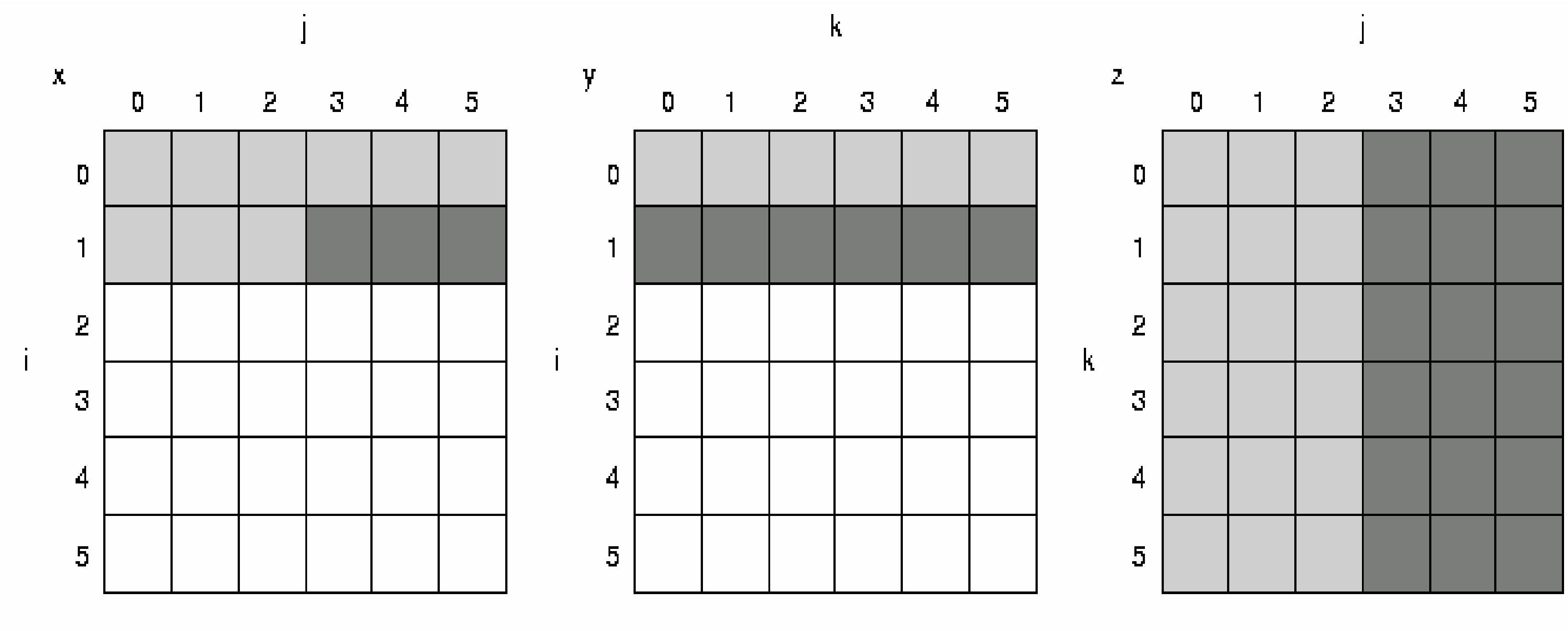
# Blocking Example

```
/* 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;
     };
```



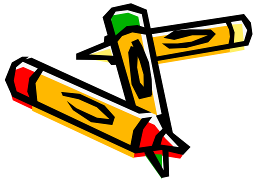
- Two Inner Loops:
  - Read all NxN elements of z[]
  - Read N elements of 1 row of y[] repeatedly
  - Write N elements of 1 row of x[]
- Capacity Misses a function of N & Cache Size:
  - $2N^3 + N^2$ => (assuming no conflict; otherwise …)
- Idea: compute on BxB submatrix that fits

# Snapshot of x, y, z when i=1



White: not yet touched
Light: older access
Dark: newer access

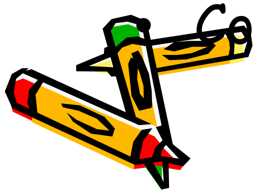
Before….



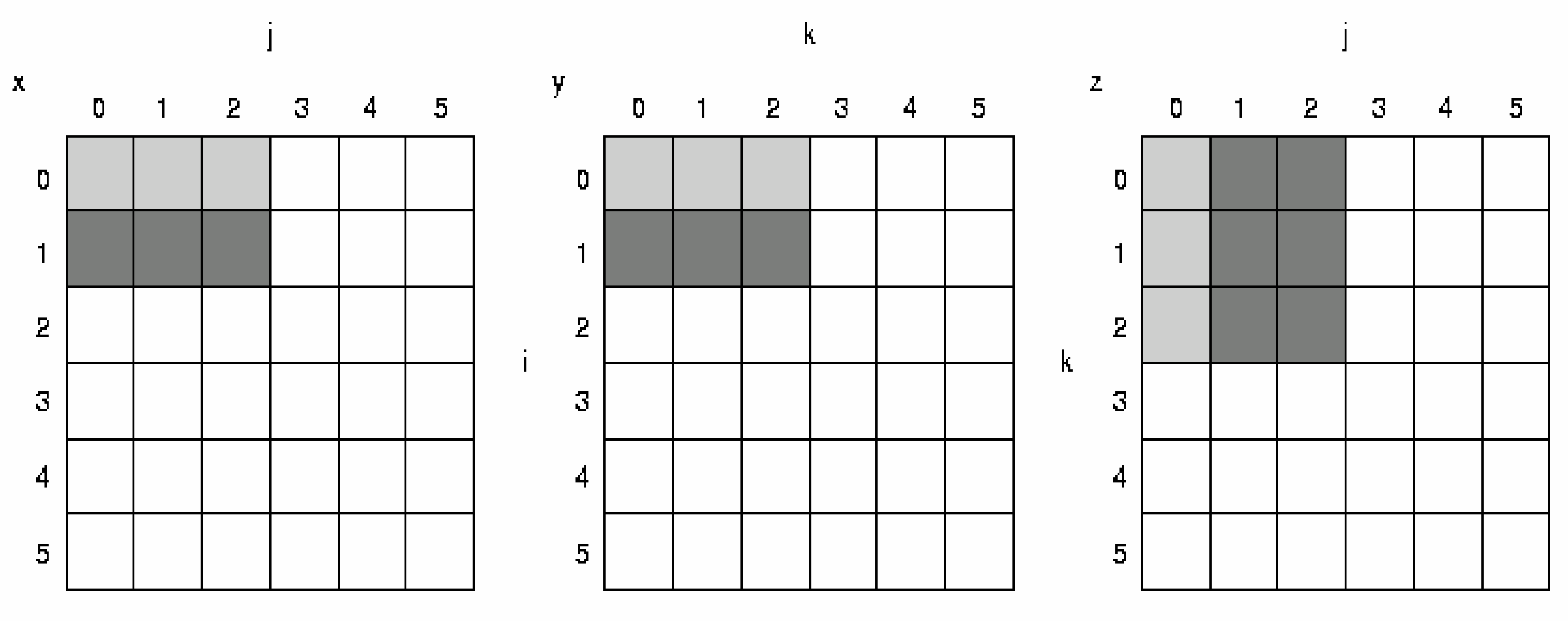
# Blocking Example

```
/* 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;
      };
```

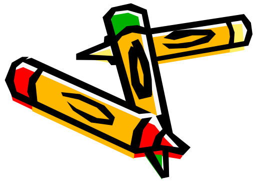
- B called *Blocking Factor*
- Capacity Misses from $2N^3 + N^2$ to $2N^3/B + N^2$
- Conflict Misses Too?



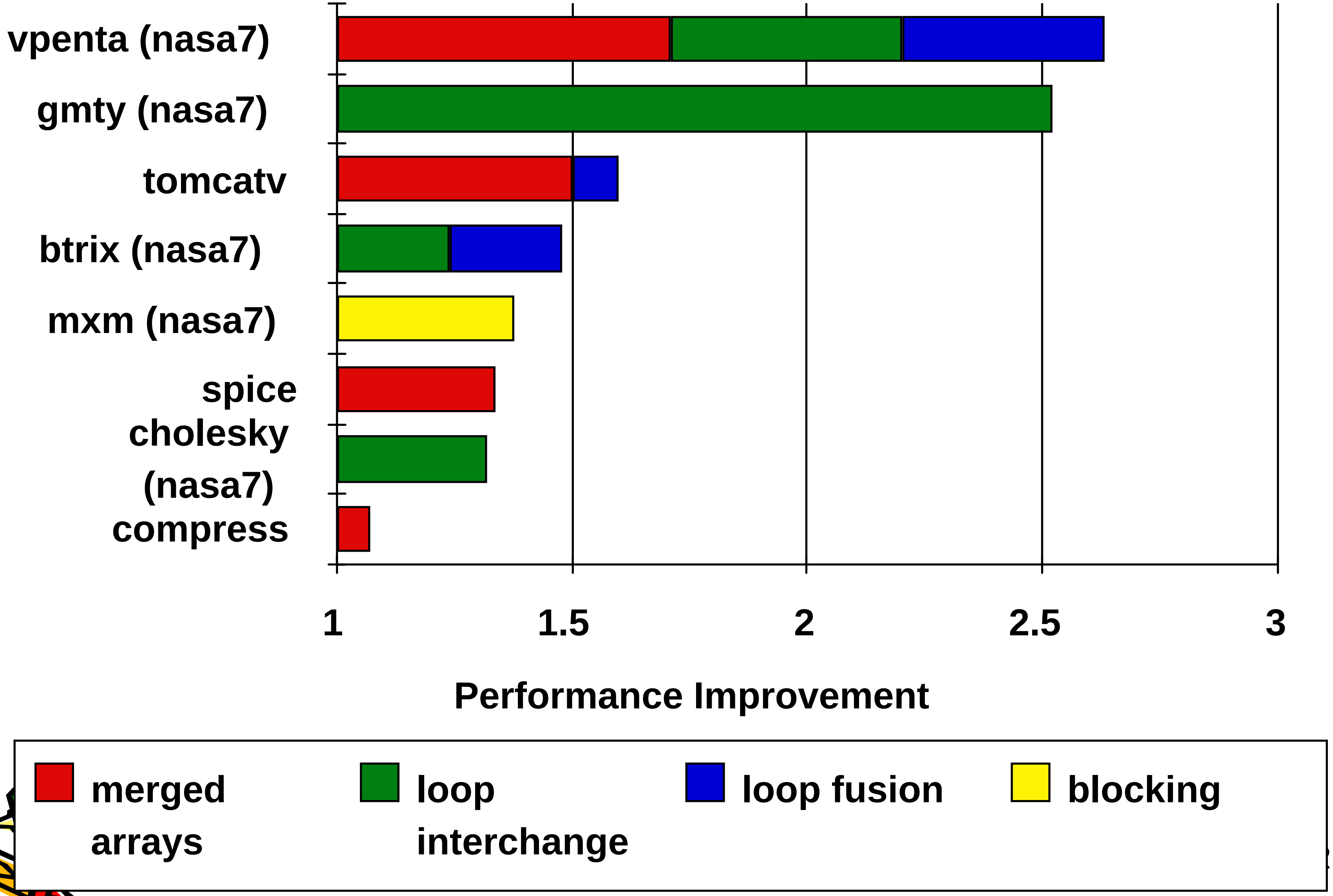
# The Age of Accesses to x, y, z



Note in contrast to previous Figure, the smaller number of elements accessed

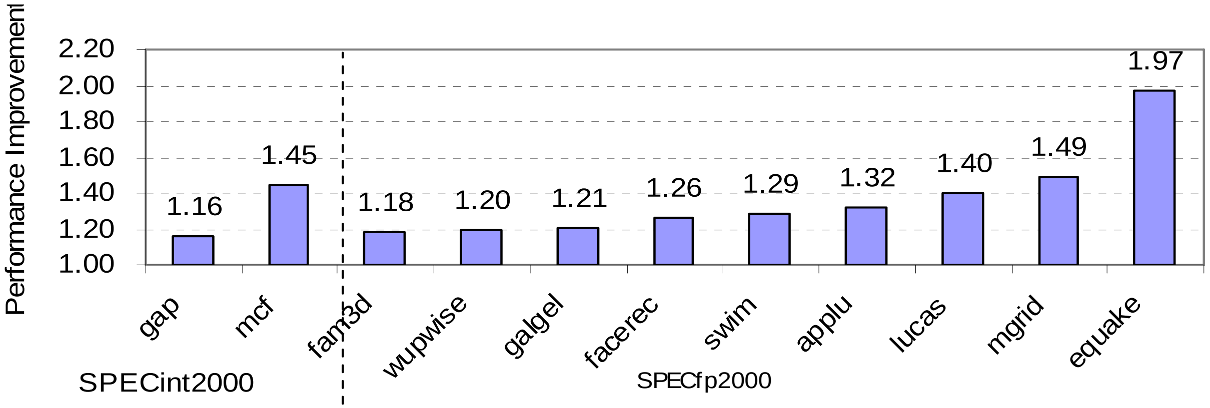


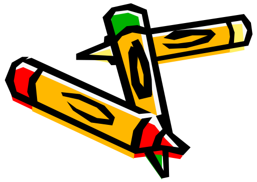
# Summary of Compiler Optimizations to Reduce Cache Misses (by hand)



# 10. Reducing Misses by Hardware Prefetching of Instructions & Data

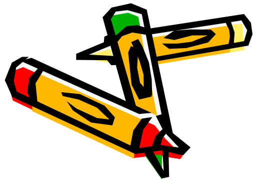
- Prefetching relies on having extra memory bandwidth that can be used without penalty
- Instruction Prefetching
  - Typically, CPU fetches 2 blocks on a miss: the requested block and the next consecutive block.
  - Requested block is placed in instruction cache when it returns, and prefetched block is placed into instruction stream buffer
- Data Prefetching
  - Pentium 4 can prefetch data into L2 cache from up to 8 streams from 8 different 4 KB pages
  - Prefetching invoked if 2 successive L2 cache misses to a page, if distance between those cache blocks is < 256 bytes





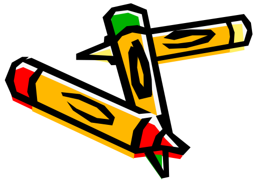
# 11. Reducing Misses by Software Prefetching Data

- 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
- Issuing Prefetch Instructions takes time
  - Is cost of prefetch issues < savings in reduced misses?
  - Higher superscalar reduces difficulty of issue bandwidth



# Compiler Optimization vs. Memory Hierarchy Search

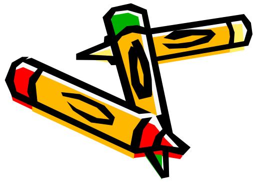
- Compiler tries to figure out memory hierarchy optimizations
- New approach: "Auto-tuners" 1st run variations of program on computer to find best combinations of optimizations (blocking, padding, …) and algorithms, then produce C code to be compiled for *that* computer
- "Auto-tuner" targeted to numerical method
  - E.g., PHiPAC (BLAS), Atlas (BLAS), Sparsity (Sparse linear algebra), Spiral (DSP), FFT-W



| Technique | Hit Time | Band-width | Miss penalty | Miss rate | HW cost/ complexity | Comment |
|---|---|---|---|---|---|---|
| Small and simple caches | + | | | – | 0 | Trivial; widely used |
| Way-predicting caches | + | | | | 1 | Used in Pentium 4 |
| Trace caches | + | | | | 3 | Used in Pentium 4 |
| Pipelined cache access | – | + | | | 1 | Widely used |
| Nonblocking caches | | + | + | | 3 | Widely used |
| Banked caches | | + | | | 1 | Used in L2 of Opteron and Niagara |
| Critical word first and early restart | | | + | | 2 | Widely used |
| Merging write buffer | | | + | | 1 | Widely used with write through |
| Compiler techniques to reduce cache misses | | | | + | 0 | Software is a challenge; some computers have compiler option |
| Hardware prefetching of instructions and data | | | + | + | 2 instr., 3 data | Many prefetch instructions; AMD Opteron prefetches data |
| Compiler-controlled prefetching | | | + | + | 3 | Needs nonblocking cache; in many CPUs |

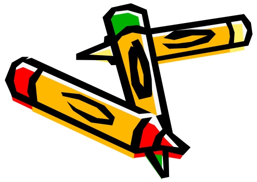
# Outline

- 11 Advanced Cache Optimizations
- Memory Technology and DRAM Optimizations
- Virtual Machines
- Conclusion



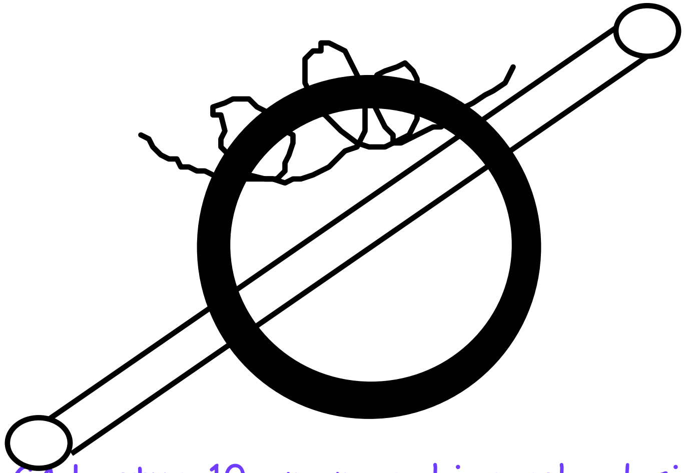
# Main Memory Background

- Performance of Main Memory:
  - Latency: Cache Miss Penalty
    - *Access Time*: time between request and word arrives
    - *Cycle Time*: time between requests
  - Bandwidth: I/O & Large Block Miss Penalty (L2)
- Main Memory is *DRAM*: Dynamic Random Access Memory
  - Dynamic since needs to be refreshed periodically (8 ms, 1% time)
  - Addresses divided into 2 halves (Memory as a 2D matrix):
    - *RAS* or *Row Access Strobe*
    - *CAS* or *Column Access Strobe*
- Cache uses *SRAM*: Static Random Access Memory
  - No refresh (6 transistors/bit vs. 1 transistor
    *Size*: DRAM/SRAM - *4-8*,
    *Cost/Cycle time*: SRAM/DRAM - *8-16*

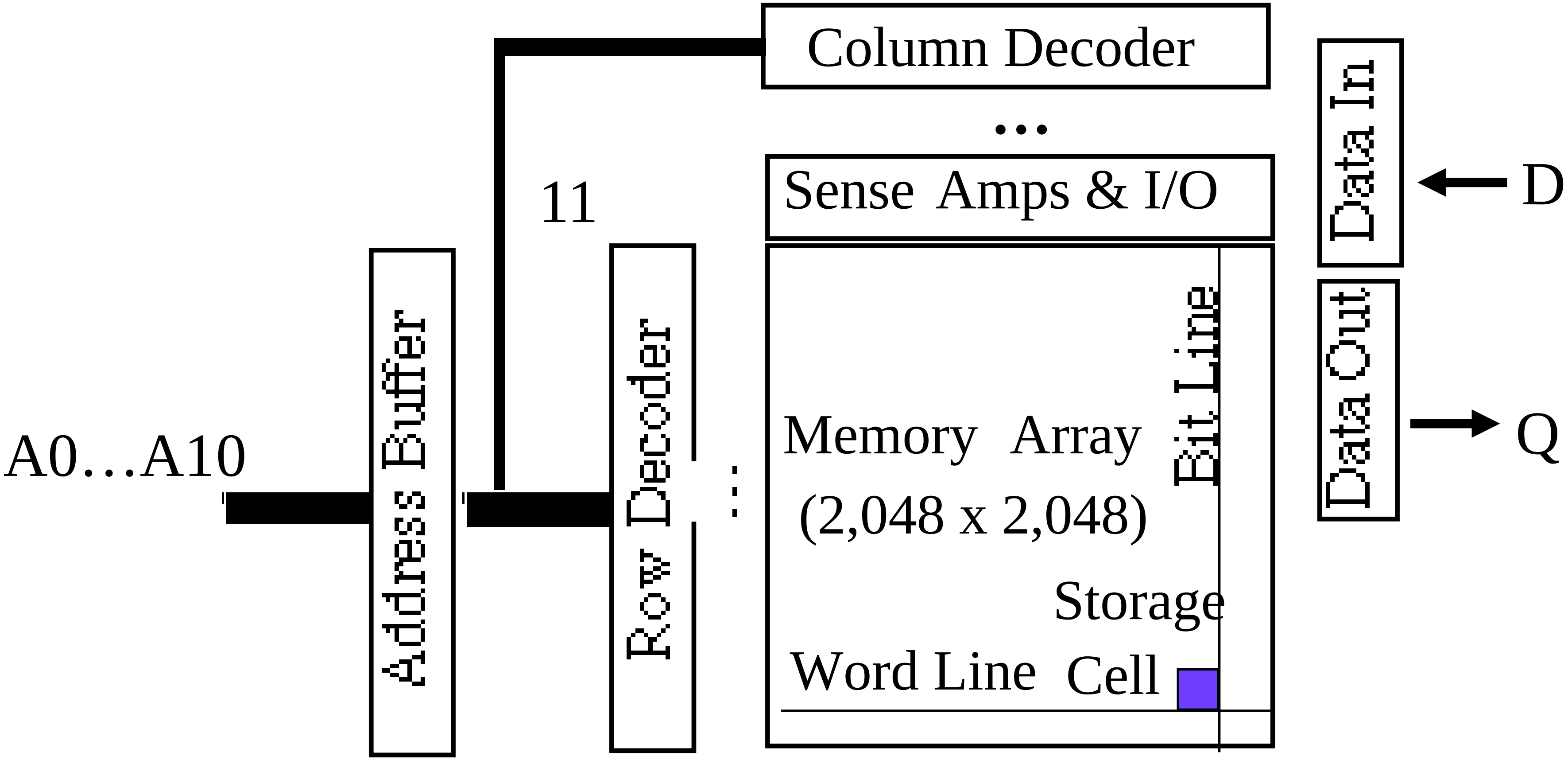


# Main Memory Deep Background

- “Out-of-Core”, “In-Core,” “Core Dump”?
- “Core memory”?
- Non-volatile, magnetic
- Lost to 4 Kbit DRAM (today using 512Mbit DRAM)
- Access time 750 ns, cycle time 1500-3000 ns



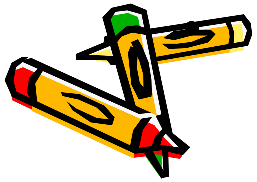
# DRAM logical organization (4 Mbit)



- Square root of bits per RAS/CAS

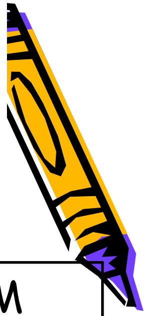
# Quest for DRAM Performance

1. Fast Page mode
   - Add timing signals that allow repeated accesses to row buffer without another row access time
   - Such a buffer comes naturally, as each array will buffer 1024 to 2048 bits for each access
2. Synchronous DRAM (SDRAM)
   - Add a clock signal to DRAM interface, so that the repeated transfers would not bear overhead to synchronize with DRAM controller
3. Double Data Rate (DDR SDRAM)
   - Transfer data on both the rising edge and falling edge of the DRAM clock signal ⇒ doubling the peak data rate
   - DDR2 lowers power by dropping the voltage from 2.5 to 1.8 volts + offers higher clock rates: up to 400 MHz
   - DDR3 drops to 1.5 volts + higher clock rates: up to 800 MHz



Improved Bandwidth, not Latency

# DRAM name based on Peak Chip Transfers / Sec
# DIMM name based on Peak DIMM MBytes / Sec



| Standard | Clock Rate (MHz) | M transfers / second | DRAM Name | Mbytes/s/ DIMM | DIMM Name |
|---|---|---|---|---|---|
| DDR | 133 | 266 | DDR266 | 2128 | PC2100 |
| DDR | 150 | 300 | DDR300 | 2400 | PC2400 |
| DDR | 200 | 400 | DDR400 | 3200 | PC3200 |
| DDR2 | 266 | 533 | DDR2-533 | 4264 | PC4300 |
| DDR2 | 333 | 667 | DDR2-667 | 5336 | PC5300 |
| DDR2 | 400 | 800 | DDR2-800 | 6400 | PC6400 |
| DDR3 | 533 | 1066 | DDR3-1066 | 8528 | PC8500 |
| DDR3 | 666 | 1333 | DDR3-1333 | 10664 | PC10700 |
| DDR3 | 800 | 1600 | DDR3-1600 | 12800 | PC12800 |

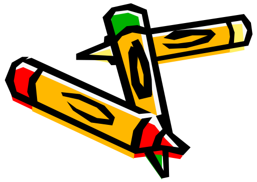
Fastest for sale 4/06 ($125/GB)

x 2

x 8

# Need for Error Correction!

- Motivation:
  - Failures/time *proportional* to number of bits!
  - As DRAM cells shrink, more vulnerable
- Went through period in which failure rate was low enough without error correction that people didn't do correction
  - DRAM banks too large now
  - Servers always corrected memory systems
- Basic idea: add redundancy through parity bits
  - Common configuration: Random error correction
    - SEC-DED (single error correct, double error detect)
    - One example: 64 data bits + 8 parity bits (11% overhead)
  - Really want to handle failures of physical components as well
    - Organization is multiple DRAMs/DIMM, multiple DIMMs
    - Want to recover from failed DRAM and failed DIMM!
    - "Chip kill" handle failures width of single DRAM chip



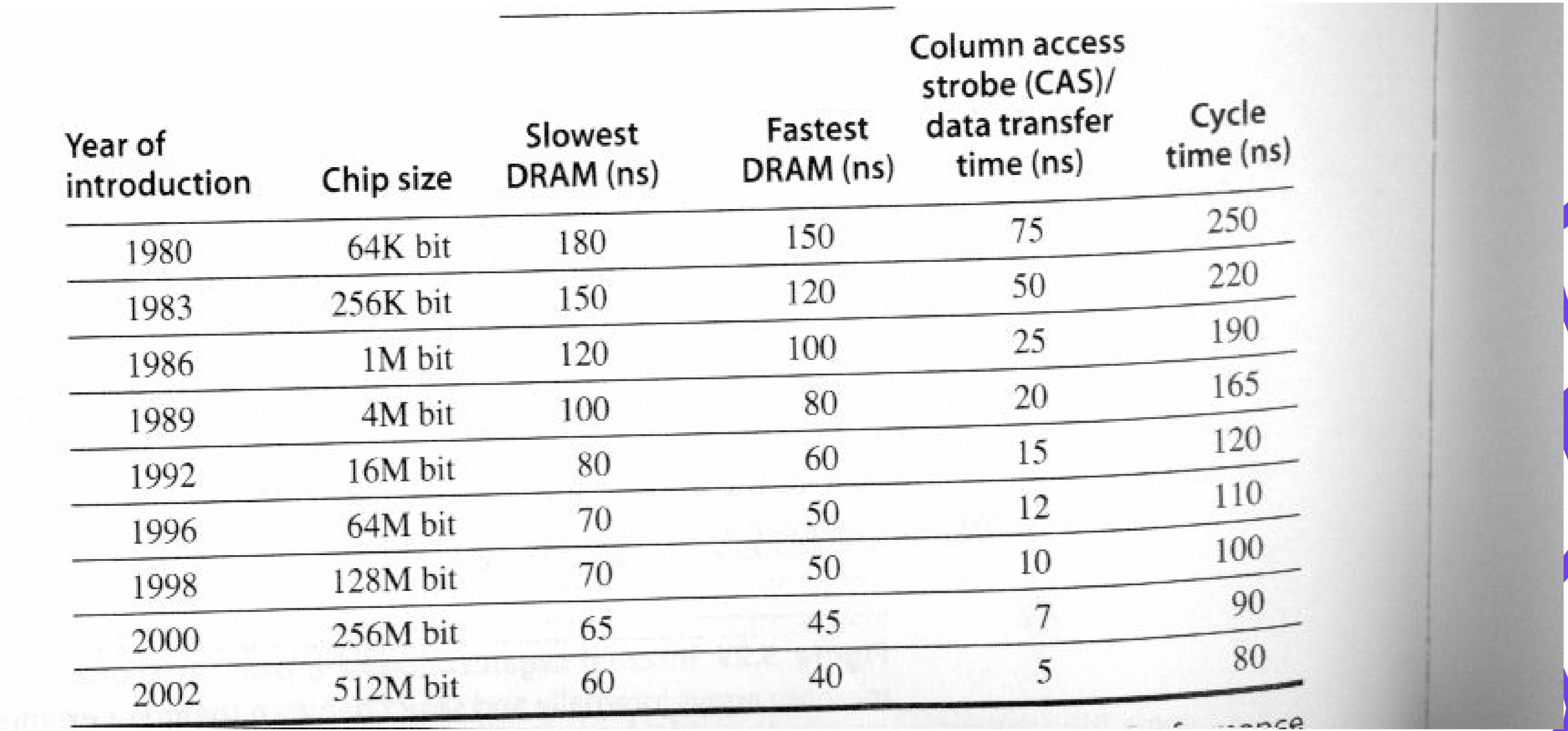
# DRAM Technology

- Semiconductor Dynamic Random Access Memory
- Emphasize on cost per bit and capacity
- Multiplex address lines ➔ cutting # of address pins in half
  - Row access strobe (RAS) first, then column access strobe (CAS)
  - Memory as a 2D matrix – rows go to a buffer
  - Subsequent CAS selects subrow
- Use only a single transistor to store a bit
  - Reading that bit can destroy the information
  - Refresh each bit periodically (ex. 8 milliseconds) by writing back
    - Keep refreshing time less than 5% of the total time
- DRAM capacity is 4 to 8 times that of SRAM

# DRAM Technology (Cont.)

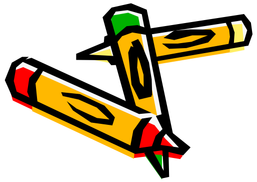
- DIMM: Dual inline memory module
  - DRAM chips are commonly sold on small boards called DIMMs
  - DIMMs typically contain 4 to 16 DRAMs
- Slowing down in DRAM capacity growth
  - Four times the capacity every three years, for more than 20 years
  - New chips only double capacity every two year, since 1998
- DRAM performance is growing at a slower rate
  - RAS (related to latency): 5% per year
  - CAS (related to bandwidth): 10%+ per year

# RAS improvement



| Year of introduction | Chip size | Slowest DRAM (ns) | Fastest DRAM (ns) | Column access strobe (CAS)/ data transfer time (ns) | Cycle time (ns) |
|---|---|---|---|---|---|
| 1980 | 64K bit | 180 | 150 | 75 | 250 |
| 1983 | 256K bit | 150 | 120 | 50 | 220 |
| 1986 | 1M bit | 120 | 100 | 25 | 190 |
| 1989 | 4M bit | 100 | 80 | 20 | 165 |
| 1992 | 16M bit | 80 | 60 | 15 | 120 |
| 1996 | 64M bit | 70 | 50 | 12 | 110 |
| 1998 | 128M bit | 70 | 50 | 10 | 100 |
| 2000 | 256M bit | 65 | 45 | 7 | 90 |
| 2002 | 512M bit | 60 | 40 | 5 | 80 |

A performance improvement in RAS of about 5% per year



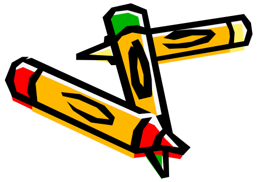
# SRAM Technology

- Cache uses SRAM: Static Random Access Memory
- SRAM uses six transistors per bit to prevent the information from being disturbed when read

  ➔ no need to refresh
  - SRAM needs only minimal power to retain the charge in the standby mode ➔ good for embedded applications
  - No difference between access time and cycle time for SRAM
- Emphasize on speed and capacity
  - SRAM address lines are not multiplexed
- SRAM speed is 8 to 16x that of DRAM

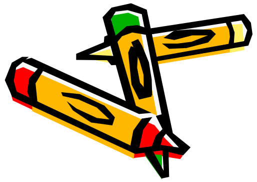
# ROM and Flash

- Embedded processor memory
- Read-only memory (ROM)
  - Programmed at the time of manufacture
  - Only a single transistor per bit to represent 1 or 0
  - Used for the embedded program and for constant
  - Nonvolatile and indestructible
- Flash memory:
  - Nonvolatile but allow the memory to be modified
  - Reads at almost DRAM speeds, but writes 10 to 100 times slower
  - DRAM capacity per chip and MB per dollar is about 4 to 8 times greater than flash



# Improving Memory Performance in a Standard DRAM Chip

- Fast page mode: time signals that allow repeated accesses to buffer without another row access time
- Synchronous RAM (SDRAM): add a clock signal to DRAM interface, so that the repeated transfer would not bear overhead to synchronize with the controller
  - Asynchronous DRAM involves overhead to sync with controller
  - Peak speed per memory module 800—1200MB/sec in 2001
- Double data rate (DDR): transfer data on both the rising edge and falling edge of DRAM clock signal
  - Peak speed per memory module 1600—2400MB/sec in 2001



# RAMBUS

- RAMBUS optimizes the interface between DRAM and CPU
- RAMBUS makes a single chip act more like a memory system than a memory component
  - Each chip has interleaved memory and high-speed interface
- 1st generation RAMBUS: RDAM
  - Replace RAS/CAS with a bus that allows other accesses over it between the sending of the address and return of the data
  - Each chip has four banks, each with their own row buffer
  - A chip can return a variable amount of data from a single request, and even perform its refresh
  - Clock signal and transfer on both edges of its clock
  - 300 MHz clock

# RAMBUS (Cont.)

- 2nd generation RAMBUS: direct RDRAM (DRDRAM)
  - Offer up to 1.6GB/sec of bandwidth
  - Separate row- and column-command buses
  - 18-bit data bus; 16 internal banks; 8 row buffers; 400 MHz
- RAMBUS are sold in RIMMs: one RAMBUS chip per RIMM
- RAMBUS vs. DDR SDRAM
  - DIMM bandwidth (multiple DRAM chips) is closer to RAMBUS
  - RDRAM and DRDRAM have a price premium over traditional DRAM
    - Larger chips
    - In 2001, it is factor of 2