Lecture 18: Memory Hierarchy—Main Memory and Enhancing its Performance

Professor Randy H. Katz

Computer Science 252

Spring 1996

Review: Reducing Miss Penalty Summary

- **Five techniques**
  - **Read priority over write on miss**
  - **Subblock placement**
  - **Early Restart and Critical Word First on miss**
  - **Non-blocking Caches (Hit Under Miss)**
  - **Second Level Cache**
- **Can be applied recursively to Multilevel Caches**
  - **Danger is that time to DRAM will grow with multiple levels in between**

Review: Improving Cache Performance

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

   Fast hit times by small and simple caches

   Fast hits via avoiding virtual address translation

   Fast hits via pipelined writes

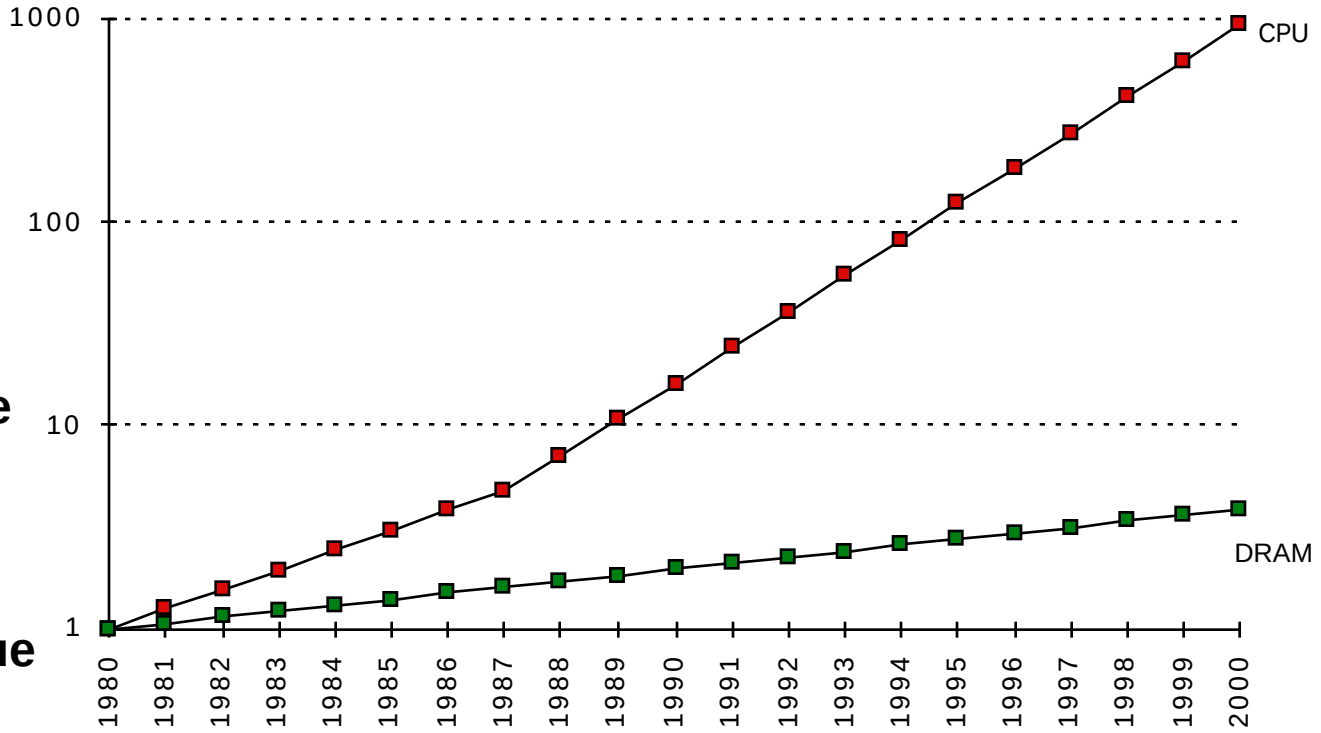
   Fast writes on misses via small subblocks

Review: 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
Subblock Placement		+	+	1
Early Restart & Critical Word 1st		+		2
Non-Blocking Caches		+		3
Second Level Caches		+		2
Small & Simple Caches	–		+	0
Avoiding Address Translation			+	2
Pipelining Writes			+	1

Review: What is the Impact of What You've Learned About Caches?

- 1960-1985: Speed = *f*(no. operations)
- 1995
  - Pipelined Execution & Fast Clock Rate
  - Out-of-Order completion
  - Superscalar Instruction Issue
- 1995: Speed = *f*(non-cached memory accesses)
- What does this mean to
  - Compilers?,Operating Systems?, Algorithms? Data Structures?

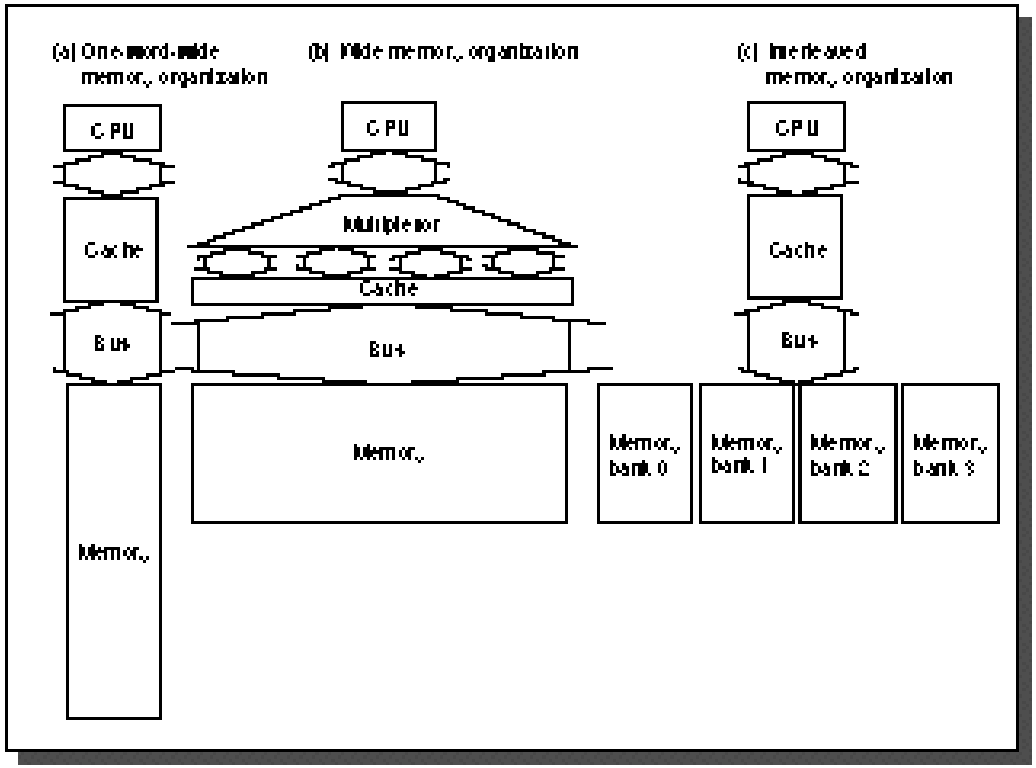


Main Memory Background

- **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)**
  - **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/bit)**
  - **Address not divided**
- ***Size*: DRAM/SRAM ≈ *4-8*,**
  Cost/Cycle time*: SRAM/DRAM ≈ *8-16

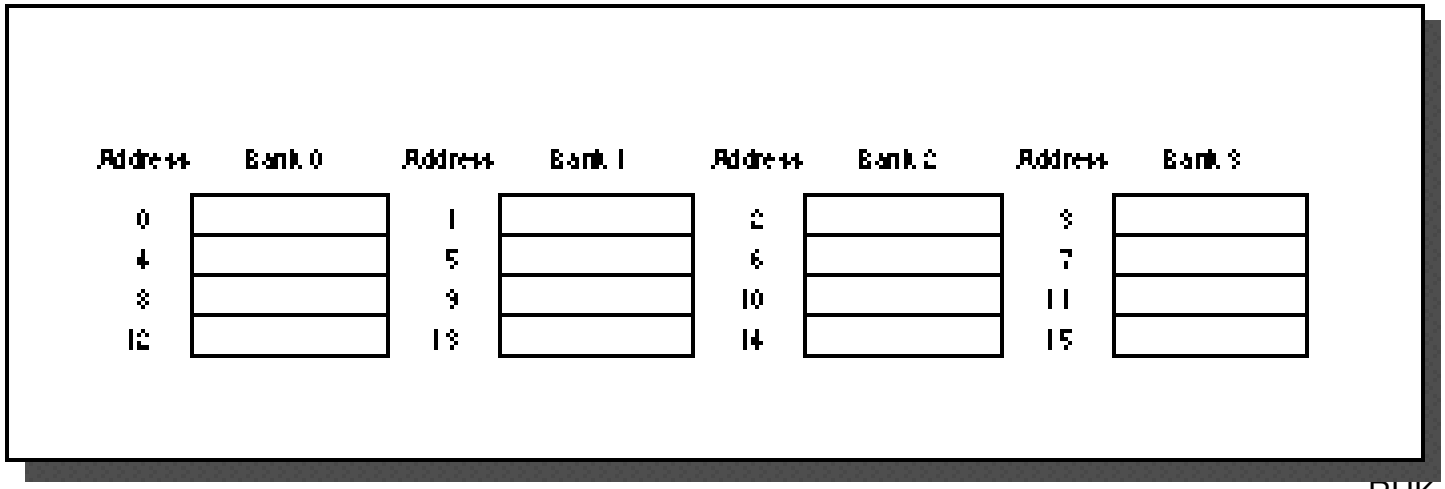
Main Memory Performance

- ***Simple***:
  - CPU, Cache, Bus, Memory same width (32 bits)
- ***Wide***:
  - CPU/Mux 1 word; Mux/ Cache, Bus, Memory N words (Alpha: 64 bits & 256 bits)
- ***Interleaved***:
  - CPU, Cache, Bus 1 word: Memory N Modules (4 Modules); example is *word interleaved*



Main Memory Performance

- **Timing model**
  - **1 to send address,**
  - **6 access time, 1 to send data**
  - **Cache Block is 4 words**
- ***Simple M.P.*** **= 4 x (1+6+1) = 32**
- ***Wide M.P.*** **= 1 + 6 + 1 = 8**
- ***Interleaved M.P.*** **= 1 + 6 + 4x1 = 11**



Independent Memory Banks

- **Memory banks for independent accesses vs. faster sequential accesses**
  - **Multiprocessor**
  - **I/O**
  - **Miss under Miss, Non-blocking Cache**
- ***Superbank*: all memory active on one block transfer**
- ***Bank*: portion within a superbank that is word interleaved**

Independent Memory Banks

- **How many banks?**

  number banks ≥ number clocks to access word in bank

  – **For sequential accesses, otherwise will return to original bank before it has next word ready**

- **Increasing DRAM => fewer chips => harder to have banks**

Avoiding Bank Conflicts

- **Lots of banks**

```
int x[256][512];
     for (j = 0; j < 512; j = j+1)
            for (i = 0; i < 256; i = i+1)
                  x[i][j] = 2 * x[i][j];
```

- **Even with 128 banks, since 512 is multiple of 128, conflict**
- **SW: loop interchange or declaring array not power of 2**
- **HW: Prime number of banks**
  - **bank number = address mod number of banks**
  - **address within bank = address / number of banks**
  - **modulo & divide per memory access?**
  - **address within bank = address mod number words in bank (3, 7, 31)**
  - **bank number? easy if 2^N words per bank**

Fast Bank Number

- **Chinese Remainder Theorem**

  As long as two sets of integers ai and bi follow these rules

  $$b_i = x \bmod a_i, 0 \le b_i < a_i, 0 \le x < a_0 \times a_1 \times a_2 \times$$

  and that ai and aj are co-prime if i ≠ j, then the integer x has only one solution (unambiguous mapping):

  - **bank number = b_0, number of banks = a_0 (= 3 in example)**
  - **address within bank = b_1, number of words in bank = a_1 (= 8 in example)**
  - **N word address 0 to N-1, prime no. banks, words power of 2**

		Seq. Interleaved			Modulo Interleaved		
Bank Number:		0	1	2	0	1	2
Address within Bank:	0	0	1	2	0	16	8
	1	3	4	5	9	1	17
	2	6	7	8	18	10	2
	3	9	10	11	3	19	11
	4	12	13	14	12	4	20
	5	15	16	17	21	13	5
	6	18	19	20	6	22	14
	7	21	22	23	15	7	23

Fast Memory Systems: DRAM specific

- **Multiple RAS accesses: several names (page mode)**
  - **64 Mbit DRAM: cycle time = 100 ns, page mode = 20 ns**
- **New DRAMs to address gap; what will they cost, will they survive?**
  - ***Synchronous DRAM*: Provide a clock signal to DRAM, transfer synchronous to system clock**
  - ***RAMBUS*: startup company; reinvent DRAM interface**
    - » **Each Chip a module vs. slice of memory**
    - » **Short bus between CPU and chips**
    - » **Does own refresh**
    - » **Variable amount of data returned**
    - » **1 byte / 2 ns (500 MB/s per chip)**
- **Niche memory or main memory?**
  - **e.g., Video RAM for frame buffers, DRAM + fast serial output**

Main Memory Summary

- **Wider Memory**
- **Interleaved Memory: for sequential or independent accesses**
- **Avoiding bank conflicts: SW & HW**
- **DRAM specific optimizations: page mode & Specialty DRAM**