Parallelism at the instruction-level

- Simple idea: CPU allowed to do two things at once:
  - Method: During memory stall perform other operations
  - Result: overlap stall time with useful work
  - Consequence: additional complexity (Append A, Ch 3-4, later)
  - Impact:
    - 1970-1985 EX time a function of # ops
    - 1985-1995 EX time a function of stall time due to misses
    - Circa 2002 EX time a function of non-overlapped stall time

Other ways to reduce hit time...

- Pipelined cache access
  - Break hit into multiple cycles and increase throughput
- Traces caches
  - Cache entries under dynamic trace of instruction stream
  - More complicated address mapping, multiple copies of instructions (branches)
  - Better utilization of space

Non-blocking caches

- To reduce stalls on a miss
  - Data cache allowed to service hits to the cache while waiting on a miss
  - A.k.a. hit under miss
  - Variation: hit under multiple miss (requires specially designed memory subsystem)
- Drawback: Memory controller complexity increases significantly

Miss Penalty/Rate Reduction via Parallelism

- Hardware Prefetching of Instructions and Data
  - Instruction prefetch get two blocks on a miss
    - First block put in cache
    - Second block put in instruction stream buffer
  - Check stream buffer on next miss
    - If found, cancel miss, repeat prefetch
  - Hardware supported to provide overlap
  - Joseph reduced misses by 20% (1 buff), 72% (16 buff)
  - For data prefetch, use multiple stream buffers

例

- 2-way set associativity vs hit under miss
  - 8KB data cache
    - DM: 11.4% miss rate for FP / 7.4% miss rate for Int programs
    - 2-way SA: 10.7% miss rate for FP / 6% miss rate for Int programs
    - Assume hit time = 0, miss penalty=16
    - AMAT = miss rate * miss penalty
    - DM+FP: AMAT = 11.4% x 16 = 1.84
    - 2-way+FP: AMAT = 10.7% x 16 = 1.71
    - For FP: 2-way has 93% of stalls for DM (blocking cache)
    - For FP: hit under miss has 76% of stalls of blocking cache (pg. 437)
  - DM+Int: AMAT = 7.4% x 16 = 1.18
  - 2-way+Int: AMAT = 6% x 16 = .96
    - For Int: 2-way has 81% of stalls for DM (blocking cache)
    - For Int: hit under miss has 81% of stalls of blocking cache (pg. 437)
  - Adv = hit under miss since it won't affect hit time

Example: Calculate effective miss rate

- What is effective miss rate with prefetching?
  - 64 KB data cache
  - Prefetching reduces dmiss rate 20%
  - Prefetch hit rate 20%
  - 36.9 dmisses per 1000 instr (22% data refs)
  - 1 cycle penalty for data hit in prefetch buffer
  - Hit time = 1 cycle, Miss penalty = 15 cycles
  - AMAT = Hit Time_DC + Miss Rate_DC x Miss Penalty_DC
  - For prefetch, modify formula
    - AMAT = Hit Time_DC + Miss Rate_DC x [Hit Rate x 1 + Miss Rate x Miss penalty]

1
Example: Calculate effective miss rate
- Calculate Miss Rate DC = 36.9/220 = 16.7%
- AMAT = 1 + Miss Rate DC x Miss penalty
  AMAT = 1 + 16.7% x (20% x 1) + (1-20%) x 15 = 3.046
- The effective miss rate (or resulting miss rate) = Miss rate for this AMAT:
  - Solve AMAT equation for Miss Rate DC
  Miss Rate DC = (AMAT - Hit Time DC) / Miss Penalty DC
  Miss Rate DC = (3.046 - 1.0) / 15 = 13.6%

How big of cache would be need to match prefetch performance?
- Find number of misses per 1000 instructions:
  X/220 = 13.6%; X = 29.92 misses per 1000 instructions
- From pg. 406, 256 KB Dcache has 32.6 misses per 1000 instructions
- Cache must be at least this size to out perform prefetching

Example: Interleaving + Wide Memory
- Block size = 1 word
- Mem bus width = 1 word
- Miss rate = 3% for current block size
- Block size = 2 words; miss rate = 1.2%
- Block size = 4 words; miss rate = 0.6%
- Mem accesses per instr = 1.2
- Miss penalty = 64 cycles
- Average CFU w/out misses = 2
- Calculate speedup for interleaving 2-ways and 4-ways versus doubling width of memory and the bus?
- CFU = (64 + 64 x 64) / (64 + 64 + 64 x 64) = 4.3
- Cycle time and instr count unaffected by change to memory system
- Cycle time and instr count unexploited by change to memory system
- Calculate speedup using CFU values at
  - Black size = 2 words -- Mem bus = double; can use 64-bit or 128-bit
  - For 64-bit, interleaving: CFU = 2 + (1.2 x 1.2 x 64) = 5.07
  - For 128-bit, interleaving: CFU = 2 + (1.2 x 1.2 x 64) = 1.54
  - For 64-bit, no interleaving: CFU = 2 + (1.2 x 64) = 3.54
  - For 128-bit, no interleaving: CFU = 2 + (1.2 x 64) = 3.54
- 2-ways slows initial implementation; interleaving = 1.19 or 1.12 speedup
  - Block size = 4 words -- Mem bus = double; can use 64-bit or 128-bit
  - For 64-bit, no interleaving: CFU = 2 + (1.2 x 1.2 x 64) = 5.07
  - For 128-bit, no interleaving: CFU = 2 + (1.2 x 64) = 3.54
  - 4-ways slows initial implementation; interleaving = 1.19 or 1.12 speedup

Example: Interleaving + Wide Memory
- Calculate speedup for interleaving 2-ways and 4-ways versus doubling width of memory and the bus?
- CFU = (64 + 64 x 64) / (64 + 64 + 64 x 64) = 4.3
- Cycle time and instr count unaffected by change to memory system
- Cycle time and instr count unexploited by change to memory system
- Calculate speedup using CFU values at
  - Black size = 2 words -- Mem bus = double; can use 64-bit or 128-bit
  - For 64-bit, interleaving: CFU = 2 + (1.2 x 1.2 x 64) = 5.07
  - For 128-bit, interleaving: CFU = 2 + (1.2 x 1.2 x 64) = 1.54
  - For 64-bit, no interleaving: CFU = 2 + (1.2 x 64) = 3.54
  - For 128-bit, no interleaving: CFU = 2 + (1.2 x 64) = 3.54
- 2-ways slows initial implementation; interleaving = 1.19 or 1.12 speedup
  - Block size = 4 words -- Mem bus = double; can use 64-bit or 128-bit
  - For 64-bit, no interleaving: CFU = 2 + (1.2 x 1.2 x 64) = 5.07
  - For 128-bit, no interleaving: CFU = 2 + (1.2 x 64) = 3.54
  - 4-ways slows initial implementation; interleaving = 1.19 or 1.12 speedup
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
- Independent Banks
  - Allow independent accesses (multiple mem controllers)

Review: 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

Crib Sheets

- Prepare a crib sheet. Follow the rules or it may not be used during the exam.
- You are only allowed one index card crib sheet.
- Maximum size: 5 inches x 8 inches (I will bring a ruler and scissors)
- Writing may only appear on one side. You may use either side.
- All writing MUST BE HANDWRITTEN on the card itself.
- No cutting and pasting printouts or copies of other material on your card.
- No sharing crib sheets.

I have the right to inspect and collect crib sheets before or after the exam (if I wish).
- You do not have the “right” to a crib sheet. If you do not follow these rules, you may be kept from using one during the exam.
- I have final ruling on all crib sheet issues.
- Not having a crib sheet does not preclude you (nor excuse you) from taking the exam.
- I reserve the right to disallow crib sheets on any exams.

Grade Distribution Fall '02