LECTURE - 22

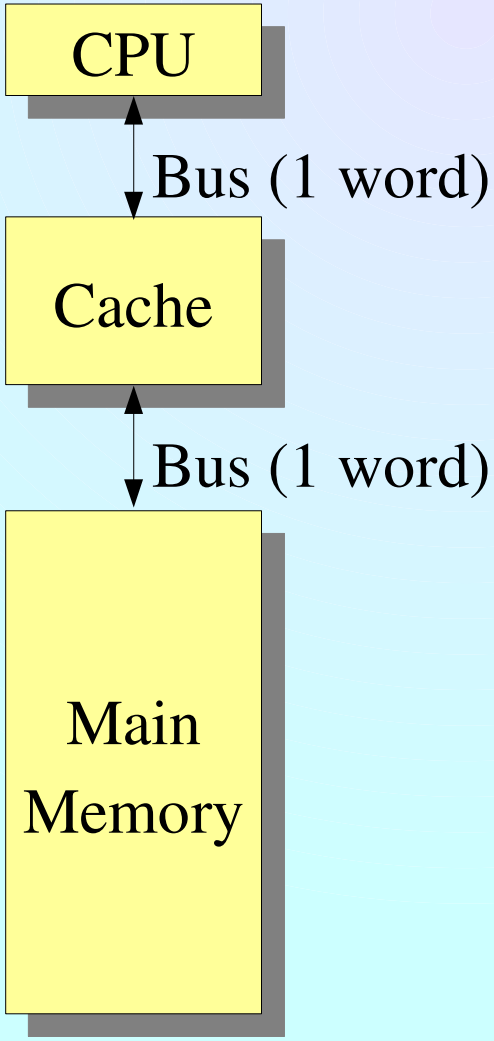
Topics for Today

- Main memory
-
- ***Scribe for today?***

Main Memory

- DRAM versus SRAM
  - DRAM is cheaper, but slower
- Reducing the number of pins
  - At the cost of some performance
  - Address = RAS + CAS
- Performance metrics: latency and bandwidth
  - #cycles to send address
  - #cycles to access a word
  - #cycles to send the data word

Main Memory Performance: One-Word Wide Memory



Suppose,

#cycles to send address = 4

#cycles to access 1 word = 24

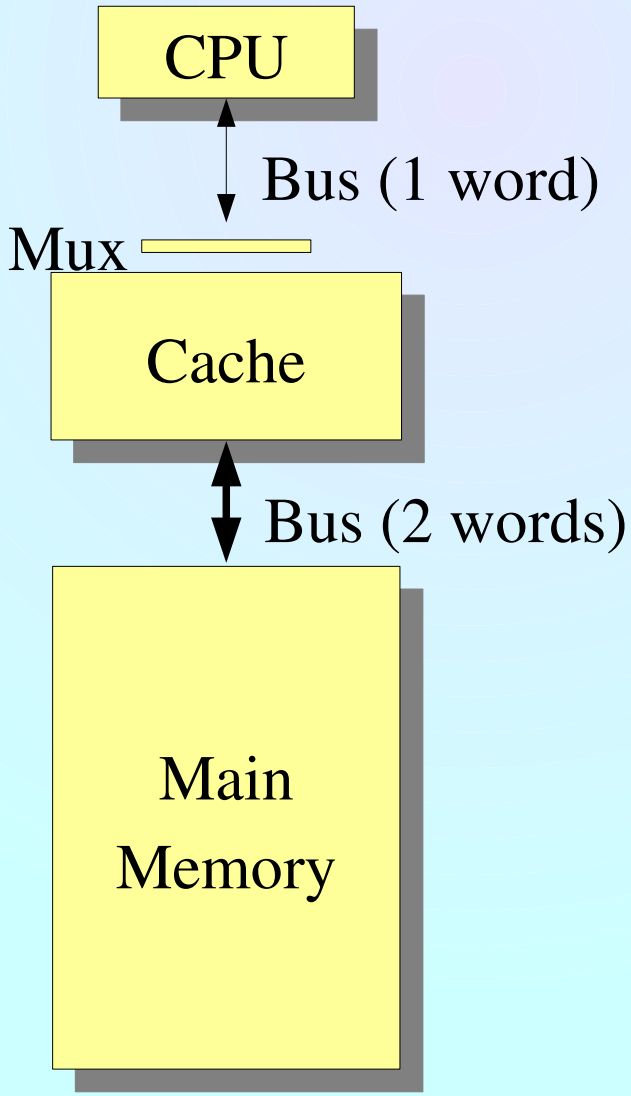
#cycles to send data word = 4

Cache line = 4 words

What is the miss penalty?

4 x (4 + 24 + 4) = 128 cycles

Technique-1: Wider Memory



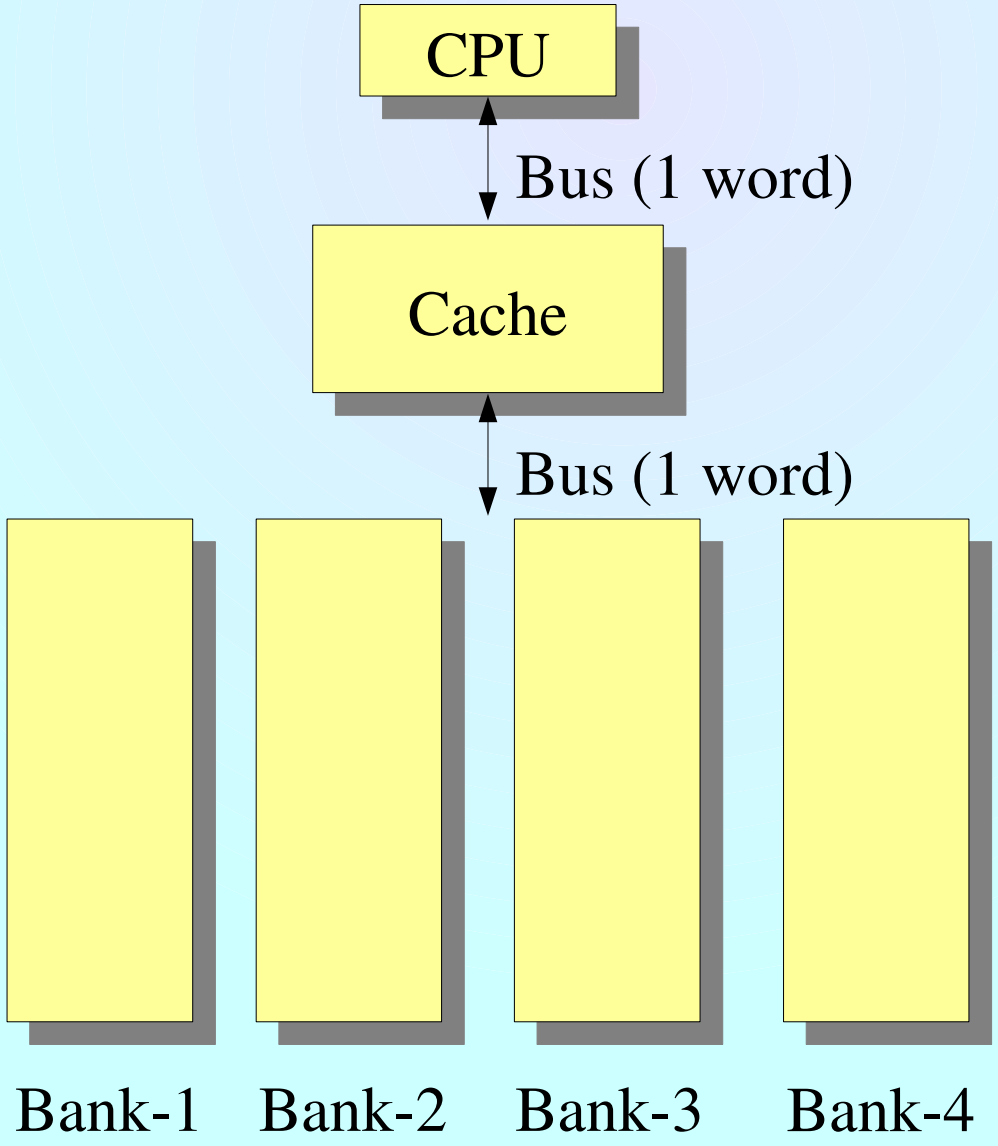
What is the miss penalty now?

2 x (4 + 24 + 4) = 64 cycles

Disadvantages?

- Larger bus width (cost)
- Unit of memory addition is larger
- Read-modify-write for single-byte write, if error-correction present

Technique-2: Interleaved-Memory



What is the miss penalty now?

4 + 24 + 4x4 =44 cycles

Notion of *interleaving factor*

Can the interleaving factor be anything?

Technique-3: Independent Memory Banks

- Multiple independent accesses
  - Separate address and data lines
- Needed for miss-under-miss scheme
- Also, parallel I/O with CPU
- Each independent bank may itself be interleaved
  - Super-bank number and bank number

Memory-Bank Conflicts

- Code can often be such that memory-bank conflicts occur
  - No use of independent memory bank organization under such conflicts
- Example:

```
int x[2][512];
for(j = 0; j < 512; j++) {
    for(i = 0; i < 2; i++) {
        x[i][j]++;
    }
}
```

Technique-4: Avoiding Memory-Bank Conflicts

- Software solutions:
  - Loop interchange (works for this example)
  - Expand array size so that it is not a power of two
- Hardware solution:
  - Use *prime number* of banks

$$Bank\ num = Addr\ \%\ \#banks$$

$$Addr\ within\ bank = Addr\ /\ \#banks$$

$$Addr\ within\ bank = Addr\ \ \#words\ within\ bank$$

if #words within bank, and #banks are co-prime

Technique-5: DRAM-Specific Interleaving

- DRAM has RAS and CAS
  - Usually RAS and CAS are given one after another
  - Same RAS can be used to read multiple columns
  - DRAMs come with separate signals to allow such access

Now, various remarks before finishing up with memory-hierarchy design

Virtual Memory and Protection

- OS requires support in terms of:
  - Two modes (at least) of execution: *user*, *supervisor/kernel*
  - Some CPU state which is readable but not writable in user mode
    - TLB
    - User/supervisor mode bit
  - Mechanisms to switch between the modes
    - System calls

ILP and Caching

- Superscalar execution:
  - Cache must have enough ports to match the peak bandwidth
  - Hit-under-miss, Miss-under-miss required
- Speculative execution:
  - Suppress exception on speculative instructions
  - Don't stall the cache on a speculative instruction cache miss

ILP vs. Caching: Compiler Choices

```
int x[32][512];
for(j = 0; j < 512; j++) {
    for(i = 0; i < 32; i++) {
        x[i][j] = 2*x[i][j-1];
    }
}
```

```
int x[32][512];
for(i = 0; i < 32; i++) {
    for(j = 0; j < 512; j++) {
        x[i][j] = 2*x[i][j-1];
    }
}
```

Caches and Consistency

- I/O using caches?
  - Interferes with CPU, may throw useful blocks
- I/O using main memory
  - Write-through ==> No problem for CPU output
  - What about input?
    - Approach-1: OS marks memory block as non-cacheable
    - Approach-2: OS flushes the cache block after input
    - Approach-3: h/w checks if block is present in cache, invalidate if cached (parallel set of tags for perf.)
- Multi-processors -w ant same data in many caches: cache-coherence problem