Lecture 15: Memory Design

• Topics: virtual memory, DRAMs (Sections 5.8-5.10)
for (jj=0; jj<N; jj+= B)
for (kk=0; kk<N; kk+= B)
for (i=0;i<N;i++)
  for (j=jj; j< min(jj+B,N); j++) {
    r=0;
    for (k=kk; k< min(kk+B,N); k++)
      r = r + y[i][k] * z[k][j];
    x[i][j] = x[i][j] + r;
  }
Exercise

- Original code could have $2N^3 + N^2$ memory accesses, while the new version has $2N^3/B + N^2$

```plaintext
for (i=0; i<N; i++)
  for (j=0; j<N; j++) {
    r = 0;
    for (k=0; k<N; k++)
      r = r + y[i][k] * z[k][j];
    x[i][j] = r;
  }
```

```plaintext
for (jj=0; jj<N; jj+=B)
  for (kk=0; kk<N; kk+=B)
    for (i=0; i<N; i++)
      for (j=jj; j<min(jj+B, N); j++) {
        r = 0;
        for (k=kk; k<min(kk+B, N); k++)
          r = r + y[i][k] * z[k][j];
        x[i][j] = x[i][j] + r;
      }
```
Tolerating Miss Penalty

• Out of order execution: can do other useful work while waiting for the miss – can have multiple cache misses -- cache controller has to keep track of multiple outstanding misses (non-blocking cache)

• Hardware and software prefetching into prefetch buffers – aggressive prefetching can increase contention for buses
DRAM Access

1M DRAM = 1024 x 1024 array of bits

10 row address bits arrive first
Row Access Strobe (RAS)

Column decoder

1024 bits are read out

10 column address bits arrive next
Column Access Strobe (CAS)

Subset of bits returned to CPU
DRAM Properties

• The RAS and CAS bits share the same pins on the chip

• Each bit loses its value after a while – hence, each bit has to be refreshed periodically – this is done by reading each row and writing the value back (hence, *dynamic* random access memory) – causes variability in memory access time

• Dual Inline Memory Modules (DIMMs) contain 4-16 DRAM chips and usually feed eight bytes to the processor
Technology Trends

• Improvements in technology (smaller devices) → DRAM capacities double every two years

• Time to read data out of the array improves by only 5% every year → high memory latency (the memory wall!)

• Time to read data out of the column decoder improves by 10% every year → influences bandwidth
Increasing Bandwidth

• The column decoder has access to many bits of data – many sequential bits can be forwarded to the CPU without additional row accesses (fast page mode)

• Each word is sent asynchronously to the CPU – every transfer entails overhead to synchronize with the controller – by introducing a clock, more than one word can be sent without increasing the overhead – synchronous DRAM
Increasing Bandwidth

• By increasing the memory width (number of memory chips and the connecting bus), more bytes can be transferred together – increases cost

• Interleaved memory – since the memory is composed of many chips, multiple operations can happen at the same time – a single address is fed to multiple chips, allowing us to read sequential words in parallel
Virtual Memory

• Processes deal with virtual memory – they have the illusion that a very large address space is available to them

• There is only a limited amount of physical memory that is shared by all processes – a process places part of its virtual memory in this physical memory and the rest is stored on disk

• Thanks to locality, disk access is likely to be uncommon

• The hardware ensures that one process cannot access the memory of a different process
Address Translation

- The virtual and physical memory are broken up into pages

8KB page size

Virtual address

- virtual page number
- page offset

Translated to physical page number

Physical address
Memory Hierarchy Properties

• A virtual memory page can be placed anywhere in physical memory (fully-associative)

• Replacement is usually LRU (since the miss penalty is huge, we can invest some effort to minimize misses)

• A page table (indexed by virtual page number) is used for translating virtual to physical page number

• The memory-disk hierarchy can be either inclusive or exclusive and the write policy is writeback
TLB

• Since the number of pages is very high, the page table capacity is too large to fit on chip

• A translation lookaside buffer (TLB) caches the virtual to physical page number translation for recent accesses

• A TLB miss requires us to access the page table, which may not even be found in the cache – two expensive memory look-ups to access one word of data!

• A large page size can increase the coverage of the TLB and reduce the capacity of the page table, but also increases memory wastage
TLB and Cache

- Is the cache indexed with virtual or physical address?
  - To index with a physical address, we will have to first look up the TLB, then the cache → longer access time
  - Multiple virtual addresses can map to the same physical address – can we ensure that these different virtual addresses will map to the same location in cache? Else, there will be two different copies of the same physical memory word

- Does the tag array store virtual or physical addresses?
  - Since multiple virtual addresses can map to the same physical address, a virtual tag comparison can flag a miss even if the correct physical memory word is present
Virtually Indexed Caches

- 24-bit virtual address, 4KB page size → 12 bits offset and 12 bits virtual page number
- To handle the example below, the cache must be designed to use only 12 index bits – for example, make the 64KB cache 16-way
- Page coloring can ensure that some bits of virtual and physical address match
Cache and TLB Pipeline

Virtually Indexed; Physically Tagged Cache
Title

• Bullet