Big Picture

- Issue 1: Fundamental concepts and principles
  - What is computer, CSE, computer architecture?
- Issue 2: ISA (HW-SW interface) design
  - Ch. 1: computer performance
  - Ch. 2: language of computer; ISA
  - Ch. 3: data representation and ALU
- Issue 3: implementation of ISA (internal design)
  - Ch. 4: processor (data path, control, pipelining)
  - Ch. 5: memory system (cache memory)
- Ch. 6: short introduction to parallel processors

Chapter 6

Parallelism and Parallel Processors (optional)

Advanced Computer Architecture Textbook:

Computer Architecture: A Quantitative Approach, Hennessy and Patterson

Some of authors' slides are modified

Preview of Topics

- Instruction-level parallelism (ILP): pipelining
- Data-level parallelism (DLP)
  - 6.3: SIMD and vector
  - 6.6: Graphic Processing Units (GPUs)
- Thread-level parallelism (TLP)
  - 6.5: Shared memory multiprocessors (SMPs)
  - 6.7: Message passing multiprocessors

이 용어들을 개념적으로 이해

SIMD Extensions in Intel x86 (반복)

- SSE2 (2001)

  Moore's law

  - Generalize SSE (8 × 128-bit registers)
  - Can be used for multiple FP/INT operands
    - 2 × 64-bit FP, 4 × 32-bit FP
    - 2 × 64-bit, 4 × 32-bit, 8 ×16-bit,16×8-bit INT
- AVX (advanced vector extension; 2008)
  - Double the width of registers
    - e.g., 4 × 64-bit double precision FP

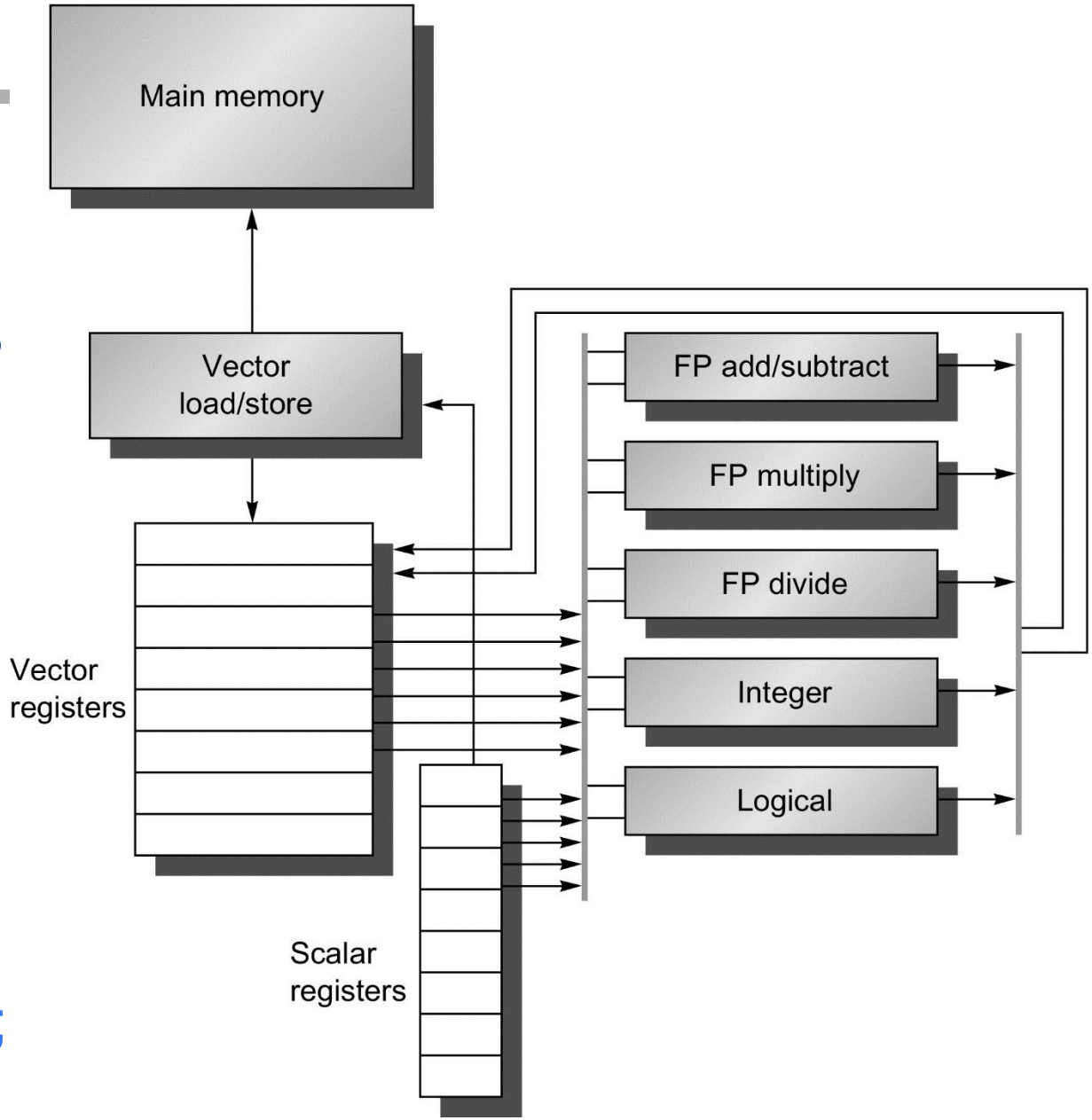
Data-Level (or SIMD) Parallelism

- SIMD architectures can exploit significant data-level parallelism for:
  - Matrix-oriented scientific computing
  - Media-oriented image and sound processors
- SIMD is more energy efficient than MIMD
  - Only fetch one instruction per data operation
  - Makes SIMD attractive for mobile devices
- SIMD allows programmer to continue to think sequentially

Vector Processors

- More elegant interpretation of SIMD
  - Seymour Cray, starting in the 1970s
- Keys
  - Set of vector registers
  - Pipelined load/store, pipelined EX units
- Basic idea:
  - Read sets of data elements into “vector registers”
  - Operate on those registers
  - Disperse the results back into memory

Vector Processors



Vector register stores 64 values

```
for (i = 0; i < 64; i++)
    Y[i] = a * X[i] + Y[i];
```

Example: DAXPY (Y = a * X + Y)

- Vector MIPS code

```
for (i = 0; i < 64; i++)
    Y[i] = a * X[i] + Y[i];
```

```
l.d      $f0,a($sp)    ;load scalar a
lv       $v1,0($s0)    ;load vector x
mulvs.d  $v2,$v1,$f0   ;vector-scalar
                        ; multiply
lv       $v3,0($s1)    ;load vector y
addv.d   $v4,$v2,$v3   ;add y to product
sv       $v4,0($s1)    ;store the result
```

6 vs. almost 600 instructions

Example: DAXPY (Y = a * X + Y)

- Conventional MIPS code

```
for (i = 0; i < 64; i++)
    Y[i] = a * X[i] + Y[i];
```

```
      l.d   $f0,a($sp)      ;load scalar a
      addiu r4,$s0,#512     ;upper bound of what
                            ; to load
loop: l.d   $f2,0($s0)      ;load x(i)
      mul.d $f2,$f2,$f0     ;a × x(i)
      l.d   $f4,0($s1)      ;load y(i)
      add.d $f4,$f4,$f2     ;a × x(i) + y(i)
      s.d   $f4,0($s1)      ;store into y(i)
      addiu $s0,$s0,#8      ;increment index to x
      addiu $s1,$s1,#8      ;increment index to y
      subu  $t0,r4,$s0      ;compute bound
      bne   $t0,$zero,loop  ;check if done
```

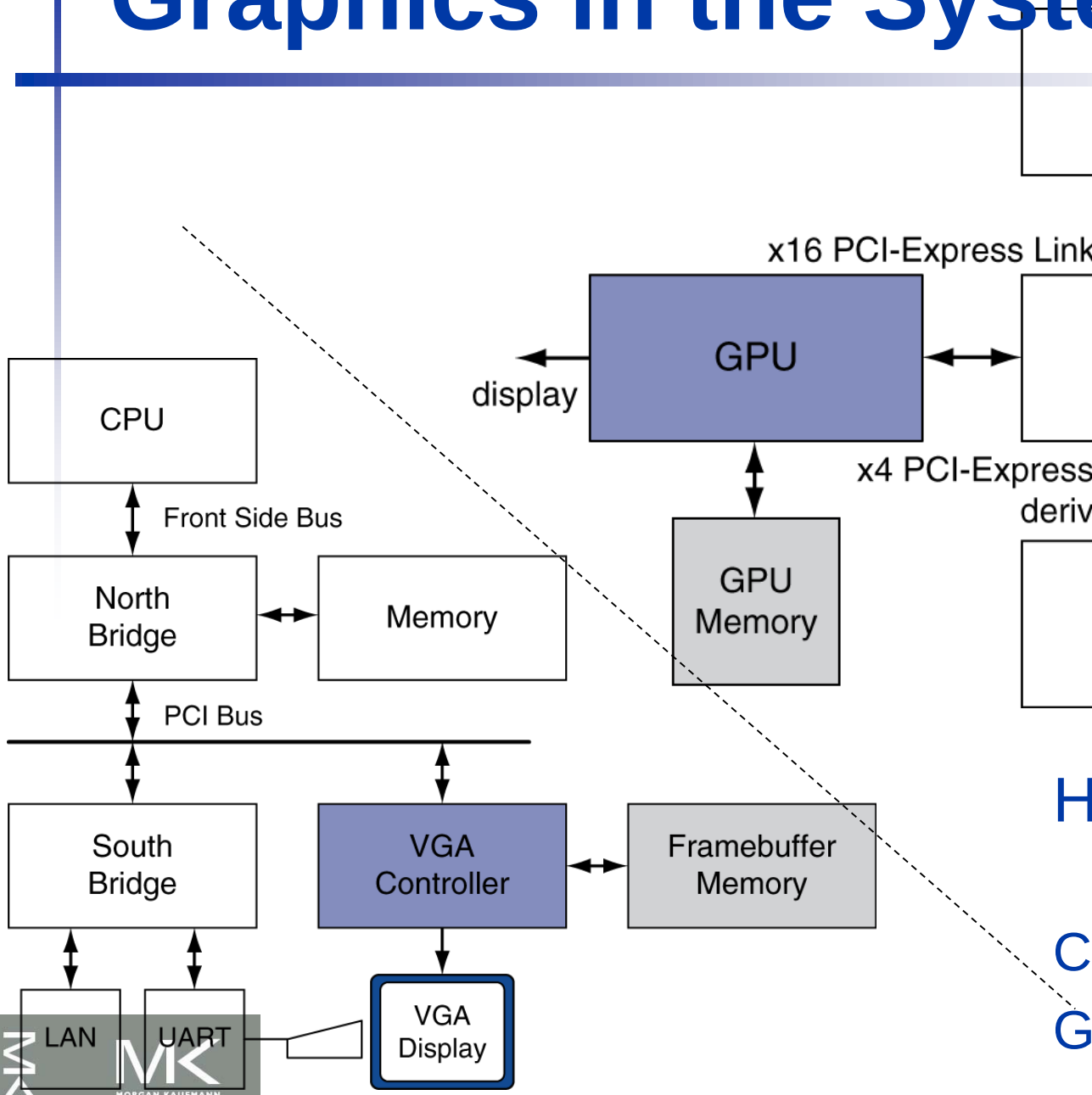
Vector vs. Scalar

- Vector architectures and compilers
  - Simplify data-parallel programming
  - HW check for hazards once per vector operation
  - Regular access patterns benefit from interleaved and burst memory
  - Avoid control hazards by avoiding loops
- More general than SIMD extensions
  - Better match with compiler technology
- Much easier than with MIMD multiprocessors

History of GPUs

- Original justification for adding SIMD instructions
  - Graphic display in early video cards
    - Increasing processing time for graphics
- Driving force for 3D graphic processing
  - Computer games (PCs and game consoles)
  - Improve faster than general-purpose processing
- Graphic processing units (GPUs)
  - Evolved its own style of processing/terminology
  - Extensive data-level parallelism

Graphics in the System



Heterogeneous computing

CPU: sequential code

GPU: parallel code

GPU Computing

- Scientific applications want to tap the performance of GPU
  - For a few hundred dollars, can buy GPU with hundreds of parallel floating-point units
  - Tired of writing programs using the graphic APIs and language (DirectX, OpenGL)
- Developed C-inspired programming environments
  - e.g., NVIDIA's CUDA
    - Can write C programs to run on GPUs
- OpenCL is a multi-company initiative

Preview of Topics

- Instruction-level parallelism (ILP): pipelining
- Data-level parallelism (DLP)
  - 6.3: SIMD and vector
  - 6.6: Graphic Processing Units (GPUs)
- Thread-level parallelism (TLP)
  - 6.5: Shared memory multiprocessors (SMPs)
  - 6.7: Message passing multiprocessors

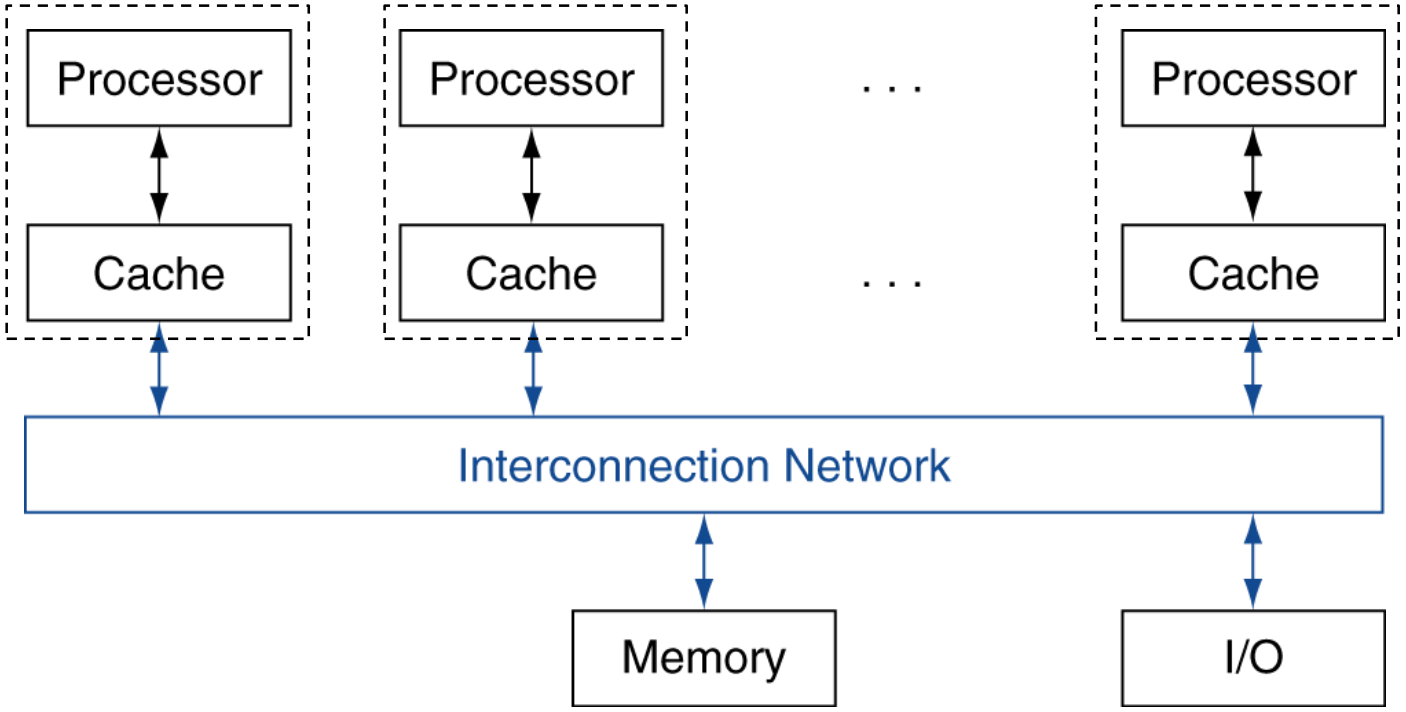
이 용어들을 개념적으로 이해

Shared Memory Multiprocessors (SMPs)

- Processors may run a single job or multiple independent jobs
- Difficult problem
  - Rewriting old programs for parallel hardware
- What computer designers did to simplify the task
  - Hardware provides single physical address space that all processors share
    - Sharing and protection
  - Synchronize shared variables using locks

Symmetric Multiprocessors (SMPs)

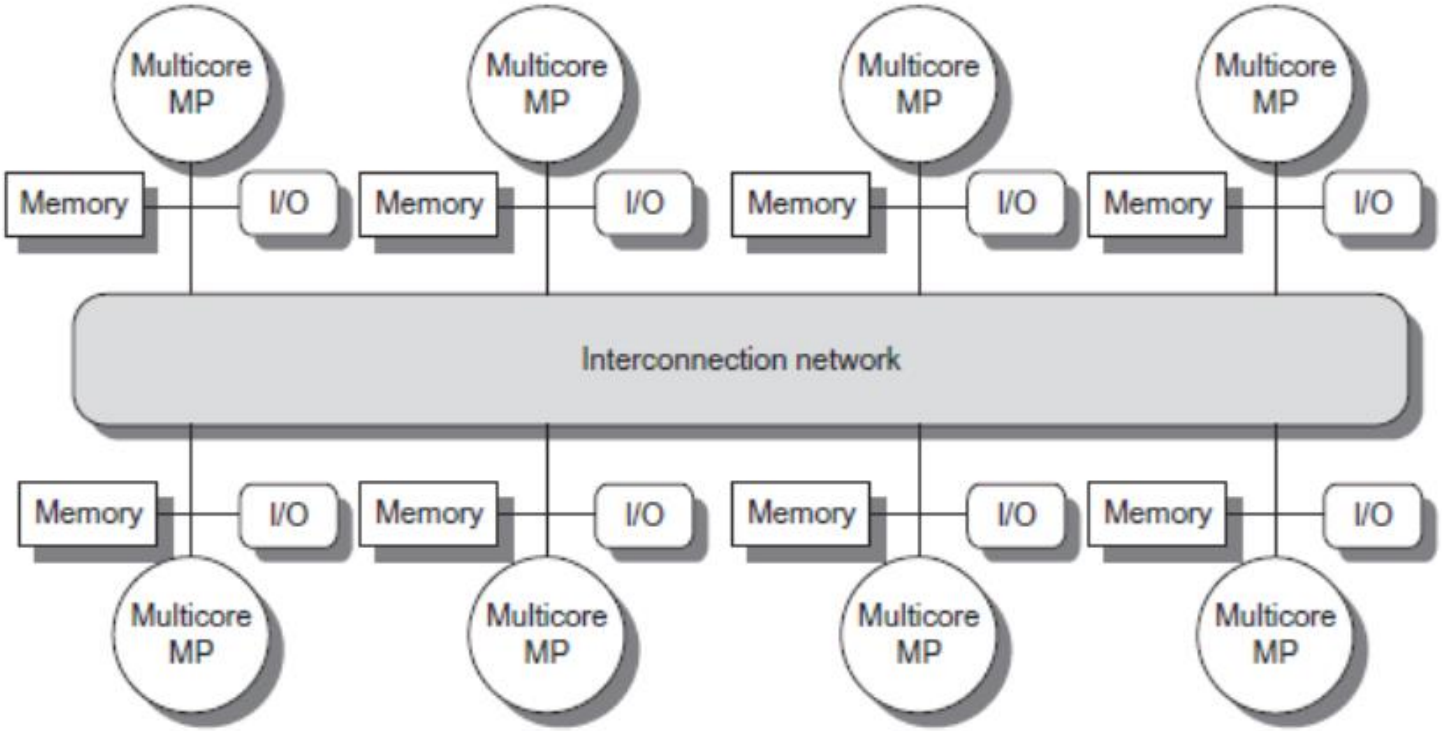
- Small number of cores
- Share single memory with uniform memory latency (UMA)



Distributed Shared memory (DSM)

- Memory distributed among processors
  - Non-uniform memory access latency (NUMA)
    - Scalable but programming harder

But market say no!



Cache Coherence Problem

- Suppose two CPU cores share a physical address space
  - Write-through caches

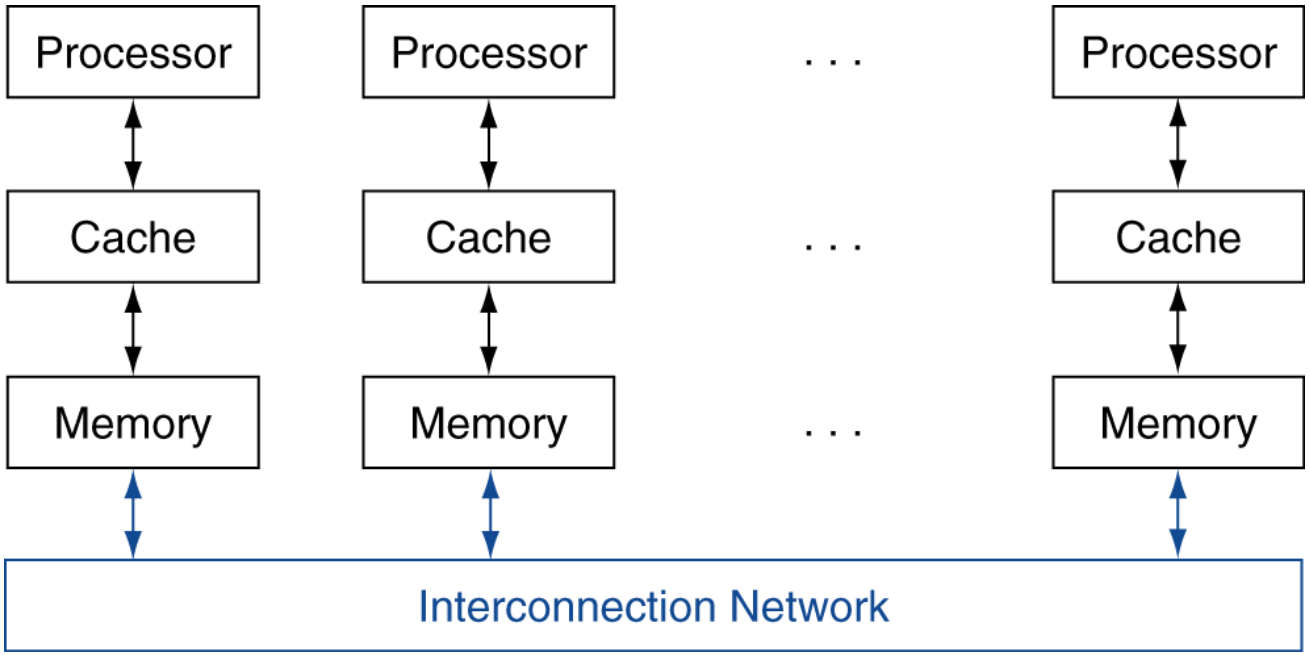
Time step	Event	CPU A's cache	CPU B's cache	Memory
0				0
1	CPU A reads X	0		0
2	CPU B reads X	0	0	0
3	CPU A writes 1 to X	1	0	1

Cache Coherence Protocols

- Snooping protocols
  - Broadcast medium
  - All cache controllers monitor the medium
  - Cache blocks have associated sharing status
- Directory-based protocols for DSM
  - Caches and memory record sharing status of blocks in a directory

Message Passing Machines

- Upside: much easier to build hardware
- Downside: harder to port sequential program to message passing computers



Message Passing Machines

- Each processor has private physical address space
- Hardware sends/receives messages between processors (thus the name)
  - Coordination is built in with explicit message passing
- Interconnection networks
  - Many supercomputers: high-performance custom networks, expensive (parallel programming)
  - Clusters: standard network switches

    (ISP data centers; natural parallelism; loosely coupled)

Loosely Coupled Clusters

- Data centers: task-level parallelism not require shared address to run well
- Dependability, scalability, rapid/incremental expandability, lower cost
  - Attractive for Internet Service Providers (ISPs)
    - Amazon, Facebook, Google, Microsoft, …
    - Web search, mail/file servers, databases, …
- Clusters have become the most widespread example of message-passing parallel computers

Warehouse-Scale Computers

- Internet services necessitate construction of new buildings
  - May be classified as large clusters
    - Act as one giant computer
  - Order of $150M for building, electrical and cooling infrastructure, servers, network equipment that connect and house 50,000 to 100,000 servers
  - Energy, power, cooling: 30% of costs

Cloud Computing

- Volume discounts (100,000 servers with infra)
- This economy of scale realized long dreamed goal of computing as a utility (*cloud computing)*
  - With lower per-unit costs of WSC, cloud companies can rent servers at a profitable rate
- Cloud computing mean anyone anywhere with good ideas, a business model and a credit card can tap thousands of servers to deliver their vision almost instantly around the world

Preview of Topics

- Instruction-level parallelism (ILP): pipelining
- Data-level parallelism (DLP)
  - 6.3: SIMD and vector
  - 6.6: Graphic Processing Units (GPUs)
- Thread-level parallelism (TLP)
  - 6.5: Shared memory multiprocessors (SMPs)
  - 6.7: Message passing multiprocessors

이 용어들을 개념적으로 이해

Domain-Specific Architectures

- Deep neural networks
  - Convolutional neural network
    - Computer vision
  - Recurrent neural network
    - Speech recognition & language translation
- TPU (Tensor Processing Unit)
- NPU (Neural Processing Unit)

AI processors

Big Picture

- Issue 1: Fundamental concepts and principles
  - What is computer, CSE, computer architecture?
- Issue 2: ISA (HW-SW interface) design
  - Ch. 1: computer performance
  - Ch. 2: language of computer; ISA
  - Ch. 3: data representation and ALU
- Issue 3: implementation of ISA (internal design)
  - Ch. 4: processor (data path, control, pipelining)
  - Ch. 5: memory system (cache memory)
- Ch. 6: short introduction to parallel processors