Lecture 07

병렬성 및 멀티스레디드 아키텍처

이재진

서울대학교 컴퓨터공학부

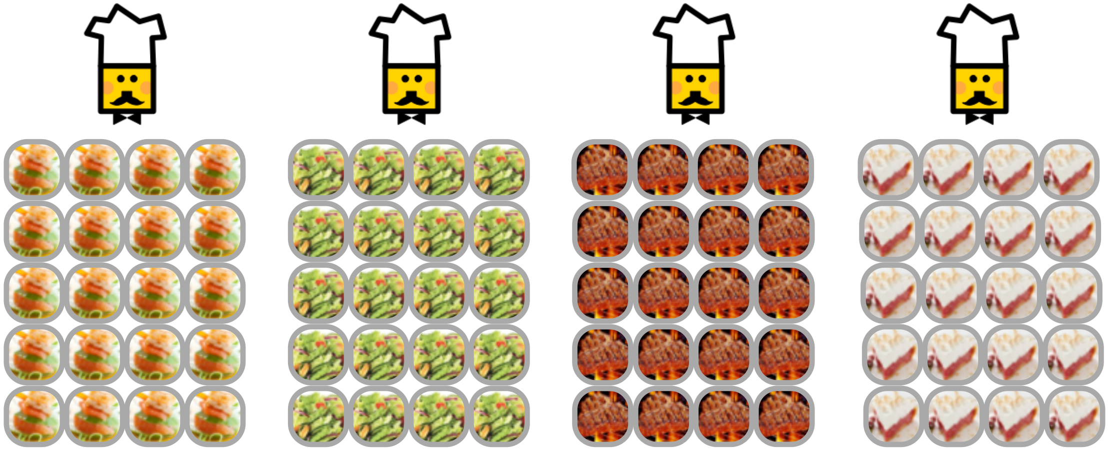
http://aces.snu.ac.kr

Types of Parallelism

- ILP
- Task parallelism
- Data parallelism

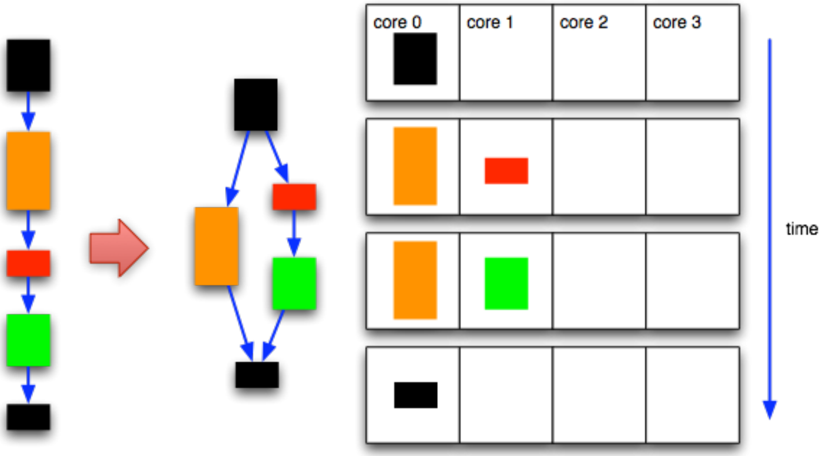
Task Parallelism

- The job of preparing a banquet
- Meal preparation consists of tasks
  - Preparing appetizer, salad, main dish, and desert
- Four different chefs
  - Each focus on one of the four tasks



Task Parallelism (cont'd)

- Performing distinct tasks at the same time
- Dividing an application into different parallel tasks (functions)
  - Most applications only have a few parallel tasks



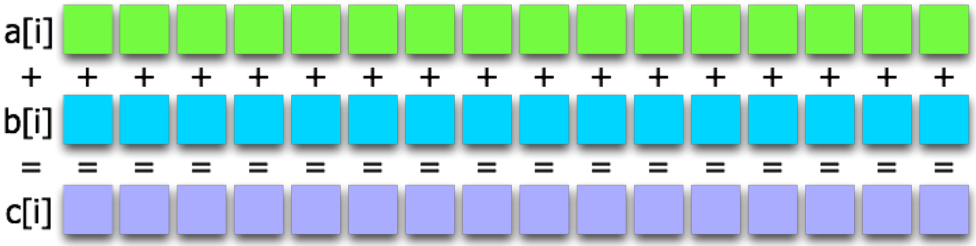
Data Parallelism

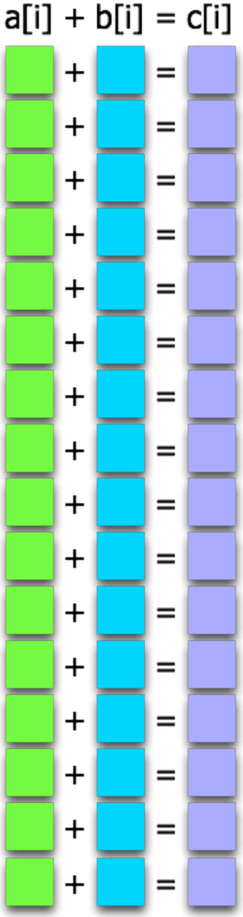
- The job of preparing a banquet
- P chefs prepare N meals
    - Each producing N/P complete meals
- As N increases, P can be increased if there are sufficient resources, such as stoves, cutting boards, etc.

Data Parallelism (cont'd)

- Also known as loop-level parallelism
- Performing the same operation to different items of data at the same time
- More data, more parallelism

```
for(i=0; i<16; i++)
{
    c[i] = a[i] + b[i];
}
```





Data Parallelism (cont'd)

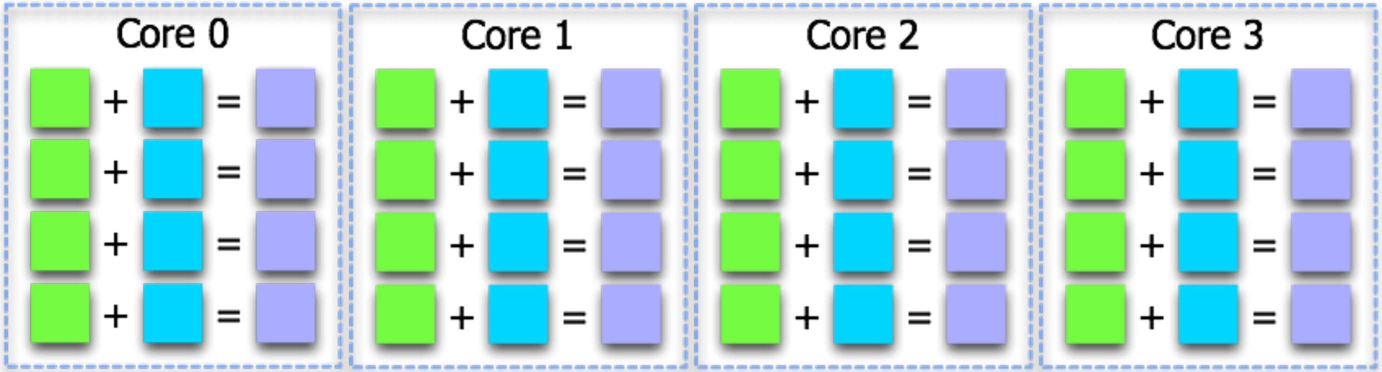
```
for(i=0; i<16; i++)
{
    c[i] = a[i] + b[i];
}
```

```
for(i=0;i<4;i++)
{
    c[i]=a[i]+b[i];
}
```

```
for(i=4;i<8;i++)
{
    c[i]=a[i]+b[i];
}
```

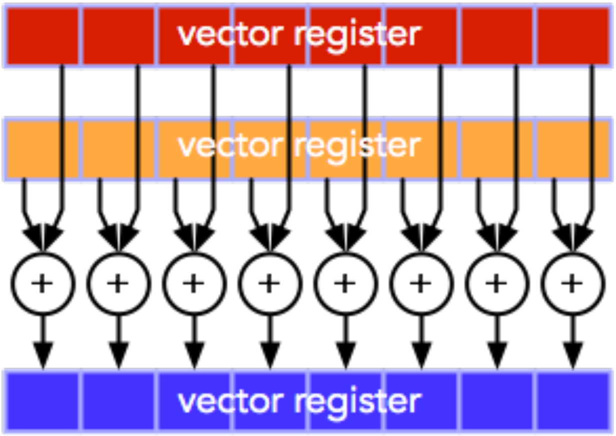
```
for(i=8;i<12;i++)
{
    c[i]=a[i]+b[i];
}
```

```
for(i=12;i<16;i++)
{
    c[i]=a[i]+b[i];
}
```



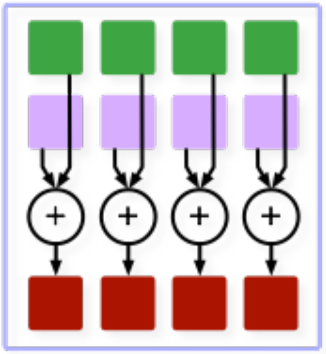
SIMD

- Single Instruction, Multiple Data
- A single instruction (or copies of a single instruction) performs the same operation in parallel on multiple data items of the same type and size
  - A single program counter
- Compilers typically support auto-vectorization
- Multiple parallel execution units are called lanes



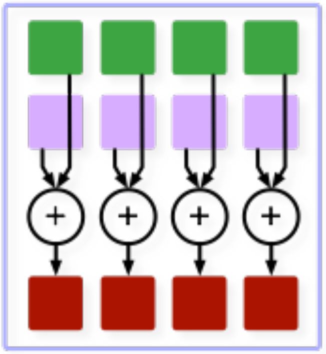
SIMD (cont'd)

```
for(i = 0;i < 4; i++)
{
  c[i] = a[i] + b[i];
}
```



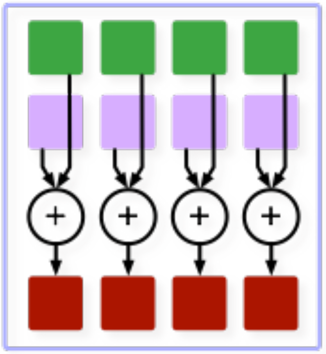
Core 0

```
for(i = 4;i < 8; i++)
{
  c[i] = a[i] + b[i];
}
```



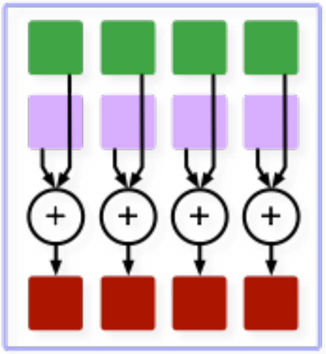
Core 1

```
for(i = 8; i < 12; i++)
{
  c[i] = a[i] + b[i];
}
```



Core 2

```
for(i = 12; i < 16; i++)
{
  c[i] = a[i] + b[i];
}
```



Core 3

SPMD

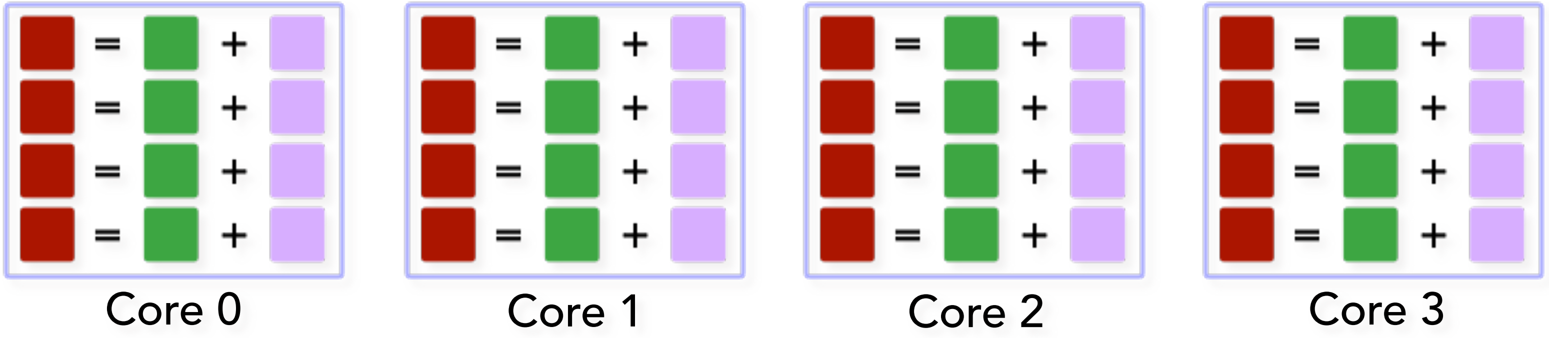
- Single Program, Multiple Data
- 같은 코드를 다른 데이터 아이템에 동시에 실행
- 개념적으로 한 개의 제어 스레드(thread of control)

```
LL = myid() * 4;
UL = LL + 4
for(i = LL; i < UL; i++)
{
    c[i] = a[i] + b[i];
}
```

```
LL = myid() * 4;
UL = LL + 4
for(i = LL; i < UL; i++)
{
    c[i] = a[i] + b[i];
}
```

```
LL = myid() * 4;
UL = LL + 4
for(i = LL; i < UL; i++)
{
    c[i] = a[i] + b[i];
}
```

```
LL = myid() * 4;
UL = LL + 4
for(i = LL; i < UL; i++)
{
    c[i] = a[i] + b[i];
}
```



Multithreaded Processors

- Issue instructions from multiple threads of control for the pipeline
  - To guarantee no dependences between instructions in a pipeline
- Exploit ILP from multiple threads that are executing simultaneously
- Functional units remain unchanged

- The lack of ILP in a single thread
  - However, ILP enabled the rapid increase in processor speed

Thread-level Parallelism (TLP)

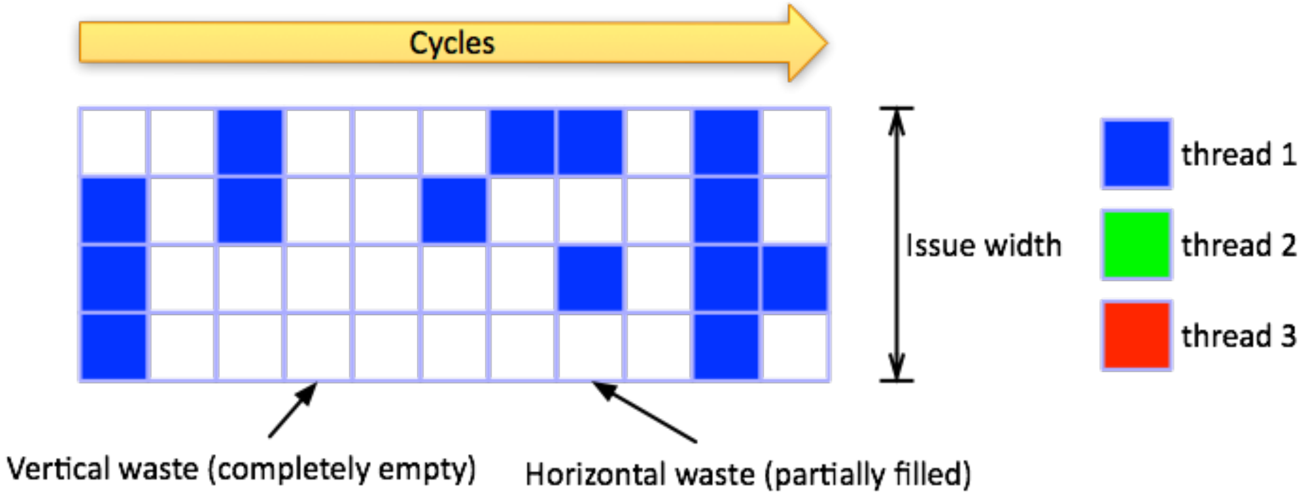
- TLP is explicitly represented by the use of multiple threads of execution
  - To improve throughput
- TLP could be more cost-effective to exploit than ILP

Issue Width of Superscalar Processors

- The goal of the instruction pipeline
    - To issue an instruction on every clock cycle
- Issuing an instruction
    - The instruction proceeds into the reservation station
- Scheduling (dispatching) an instruction
    - The instruction proceeds into the execution unit from the reservation station
- Issue-width is the maximum number of instructions that can be issued by a processor
    - When the hardware can issue up to n instructions on every cycle:
        - The processor has n issue slots
        - The processor is an n-issue processor
    - Fetch n instructions simultaneously
    - There are n decode units

Superscalar Processors

- Inefficiency
  - Vertical waste and horizontal waste

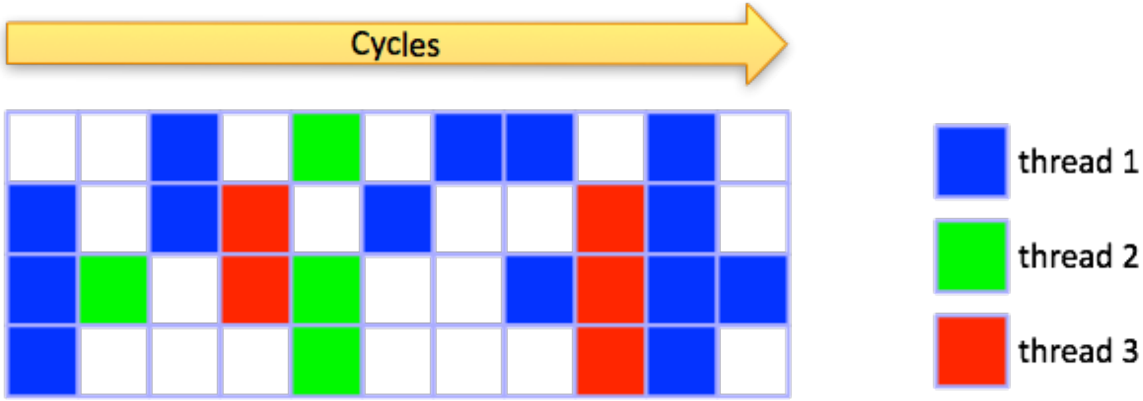


Vertical Multithreading

- Reduce vertical waste by scheduling threads to hide long latency
  - Switching different thread contexts each cycle
  - Tolerate long latency operations (remove vertical waste)
- Still waste unused issue slots (horizontal waste)
- Scheduling
  - Fine-grained multithreading - context switch among threads every cycle
  - Coarse-grained multithreading - context switch among threads every few cycles on data hazards, cache misses, etc.

- CDC 6600 (Cray, 1964)
  - For peripheral processing unit
- HEP (Burton Smith, 1982)
  - First commercial hardware-threading for CPU
- Tera MTA (1990)

Vertical Multithreading (contd.)

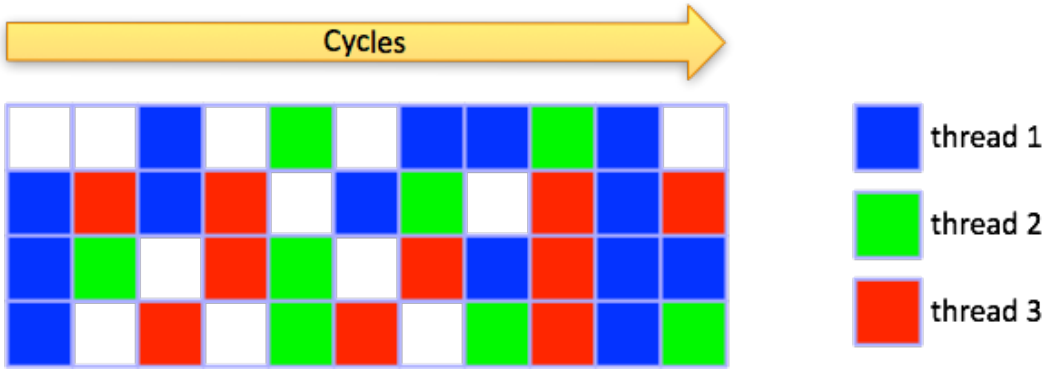


		🟦		🟩		🟦	🟦		🟦	
🟦		🟦	🟥		🟦			🟥	🟦	
🟦	🟩		🟥	🟩			🟦	🟥	🟦	🟦
🟦				🟩				🟥	🟦	

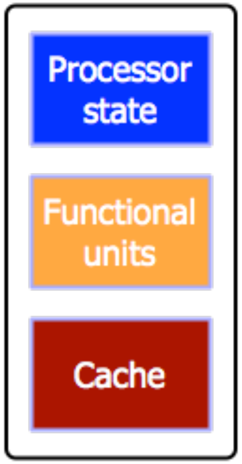
🟦 thread 1
🟩 thread 2
🟥 thread 3

Simultaneous Multithreading (SMT)

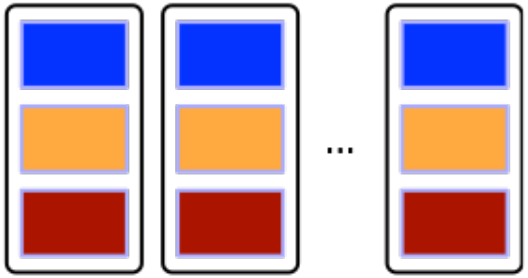
- Selects instructions for execution from all threads on each cycle
  - Remove both horizontal and vertical waste
  - To more fully utilize the issue width
- Superscalar processors already have many HW mechanisms to support multithreading
- Hyper-threading (Intel)
- IBM Power 5



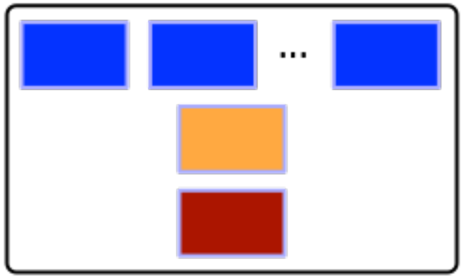
Homogeneous Multicores



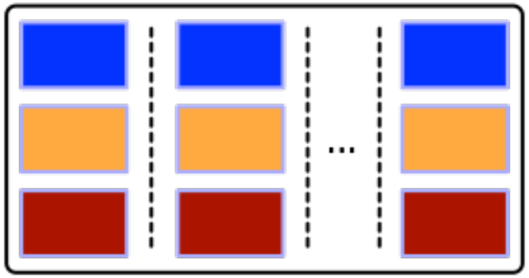
Uniprocessor CPU



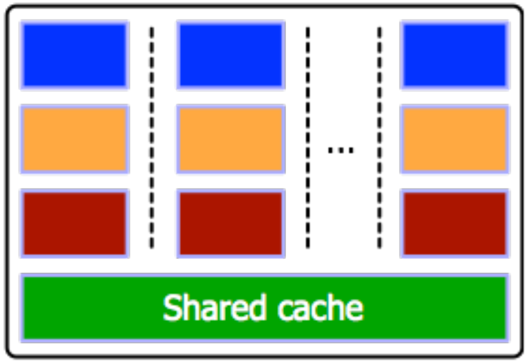
Multiprocessor



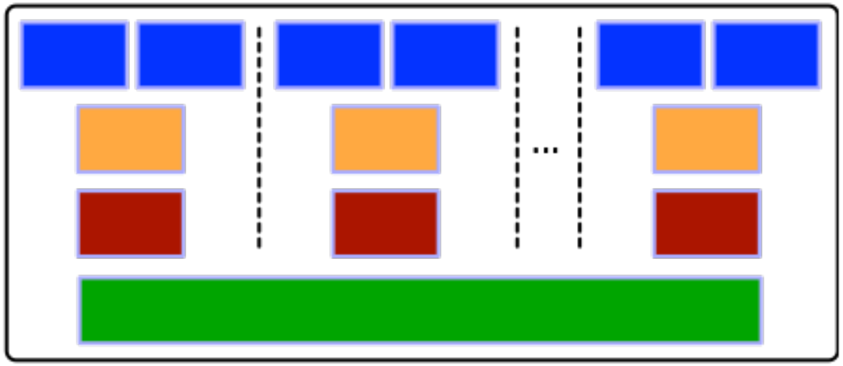
Uniprocessor CPU with simultaneous multithreading



Multicore CPU with private caches



Multicore CPU with a shared cache



Multicore CPU with simultaneous multithreading