Concurrency vs. Parallelism

To make programmes a little bit more efficient

Contents

- Parallelism?

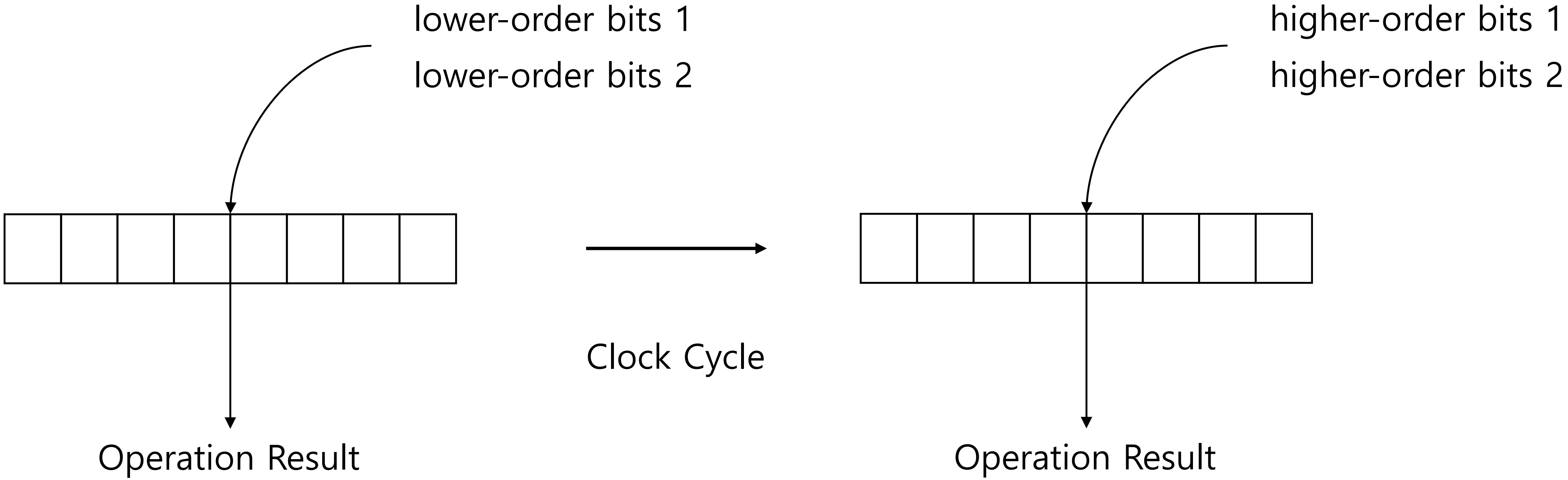
- Concurrency?

Parallelism?

- In Korean : 병렬성
- To **physically** do more than one thing at once
- Requires hardware support

Bit-level parallelism

Ex) Operating two 16-bit integer with 8-bit processor



lower-order bits 1
lower-order bits 2

Clock Cycle

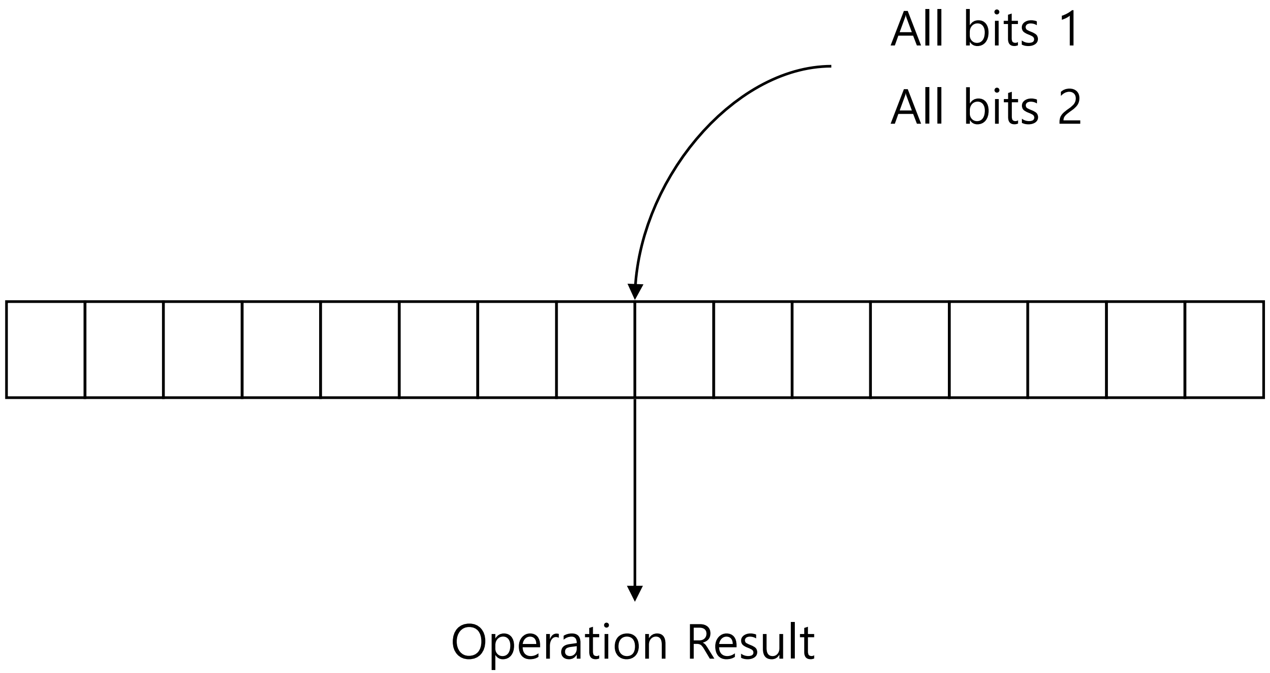
higher-order bits 1
higher-order bits 2

Operation Result

Operation Result

Bit-level parallelism

Ex) Operating two 16-bit integer with 16-bit processor



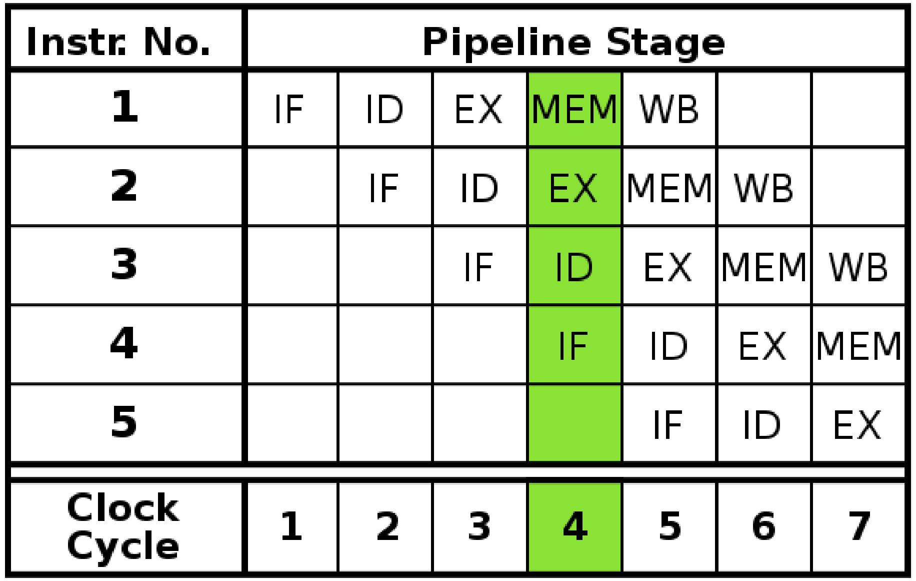
All bits 1

All bits 2

Operation Result

Instruction-level parallelism

```
e = a + b
f = c + d
g = e + f
```



Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

Instruction pipelining, Superscalar, etc...

Task parallelism

```
program:
...
if CPU = "a"
   do task "A"

else if CPU = "b"
   do task "B"
```

```
program:
   do "A"
```

```
program:
   do "B"
```

Parallelism

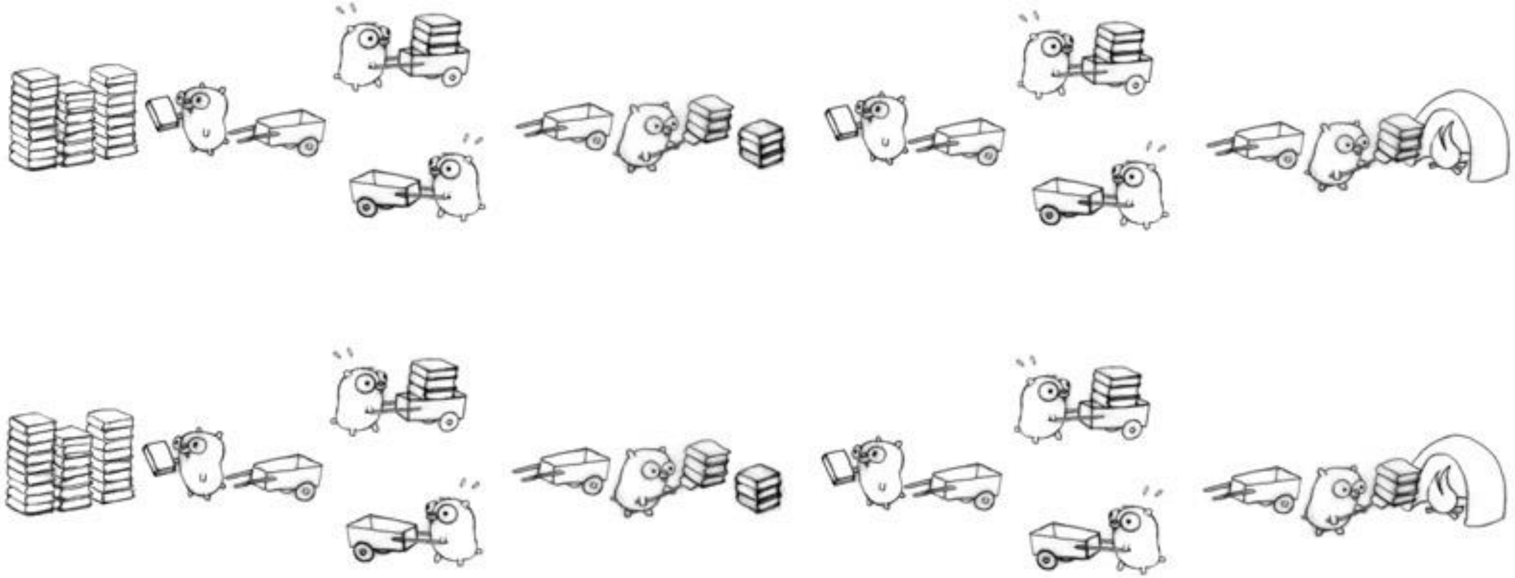
NVIDIA®
CUDA®

Concurrency?

- In Korean : 병행성 / 동시성
- Let "→" as "happened before"
- If a → b, then it is possible for a to casually affect b
- Two events are concurrent if…
    - !(a → b) && !(b → a)
    - neither can casually affect the other

So what?

- Concurrency is programming as the composition of independently executing processes!

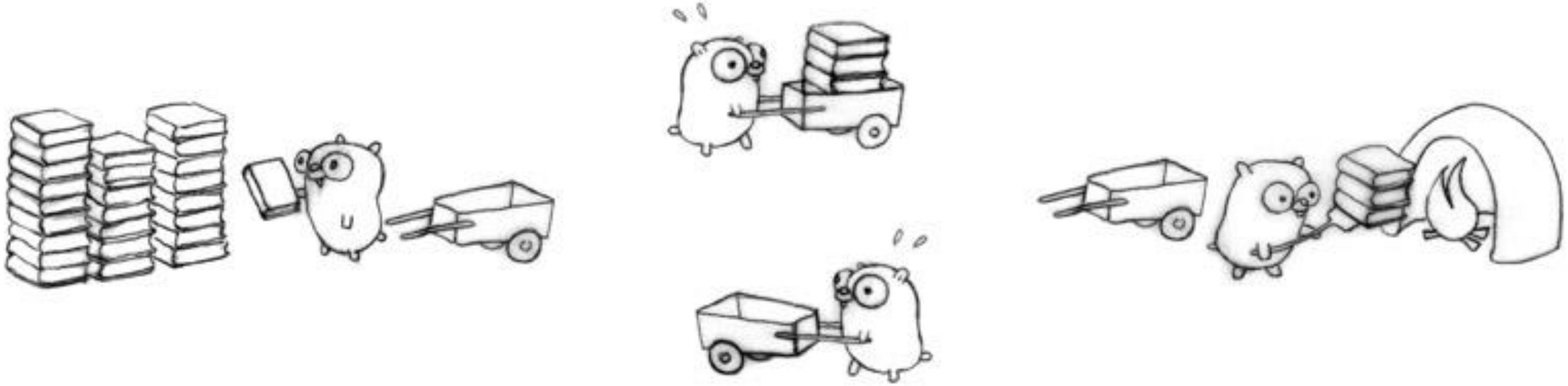


Concurrency

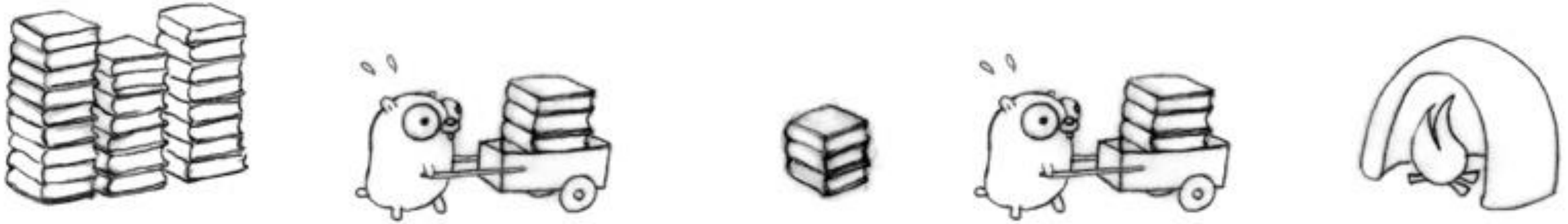
Concurrency

Concurrency

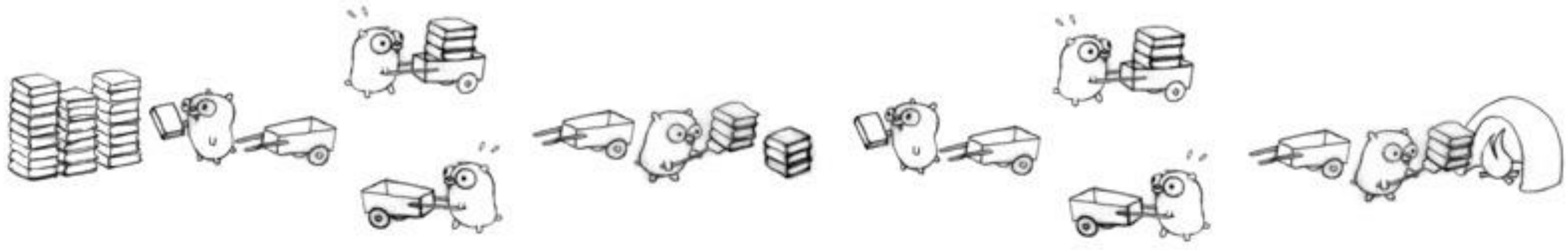
Concurrency



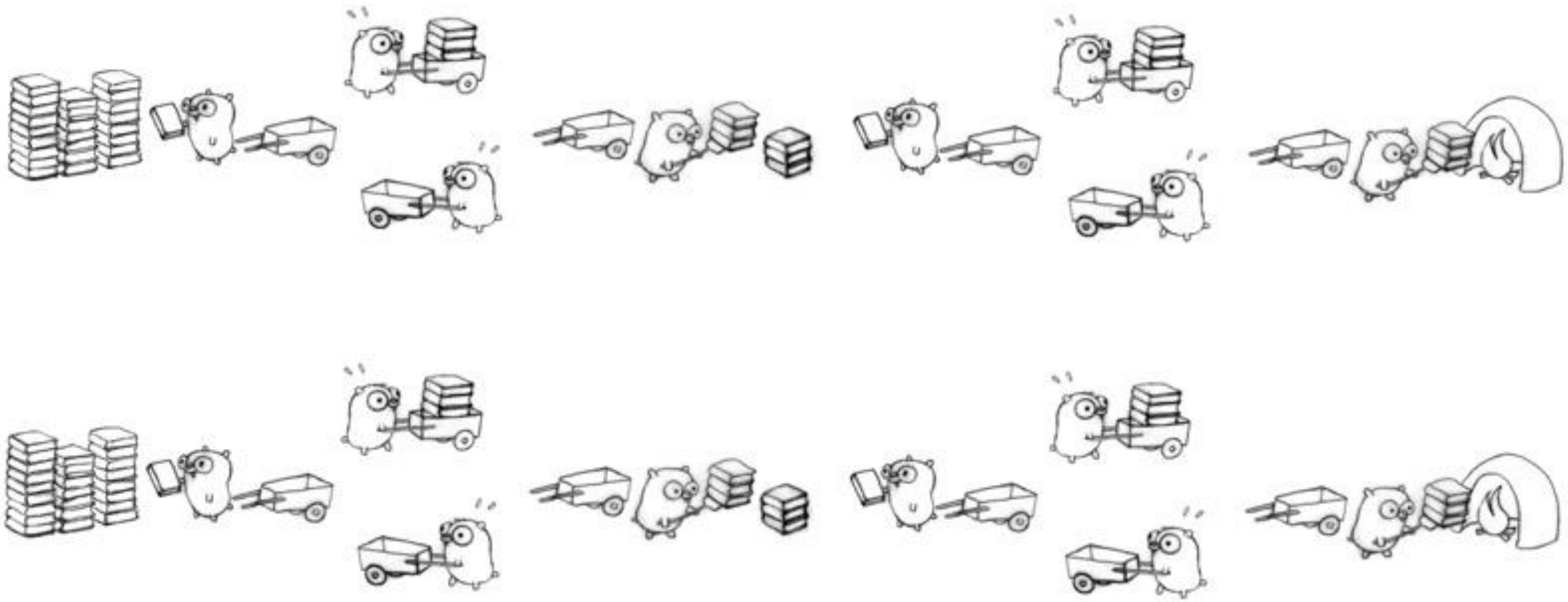
Concurrency



Concurrency



Concurrency



Todo

- MutEx?
- Semaphore?
- Maybe example codes?