15 Pipelining and Flynn – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	**中文**	**Write it 3 times** (English)		
pipeline	流水线			
hazard	冒险			
Flynn's taxonomy	弗林分类			
SIMD	单指令多数据			

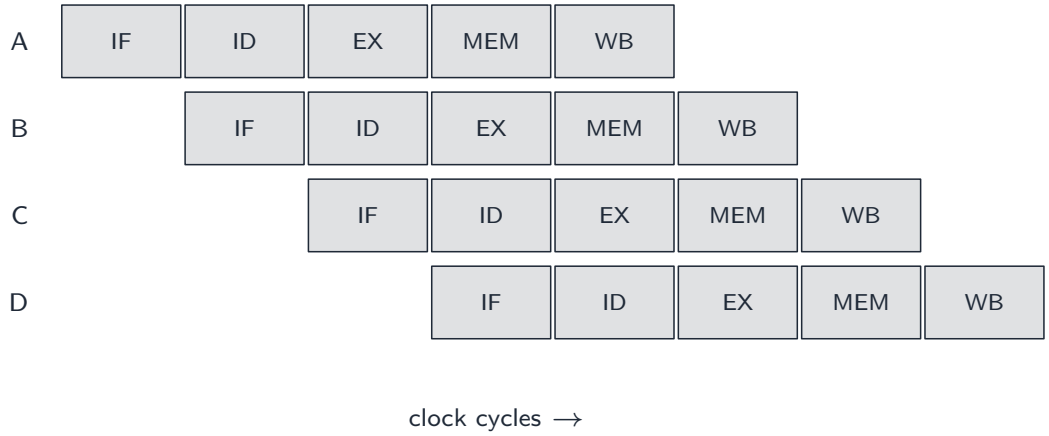
Recall

Modern CPUs run instructions in overlapping stages, like an assembly line, and come in different parallel styles.

New ideas

Read these first. Each idea has a short worked example. Then do the Practice.

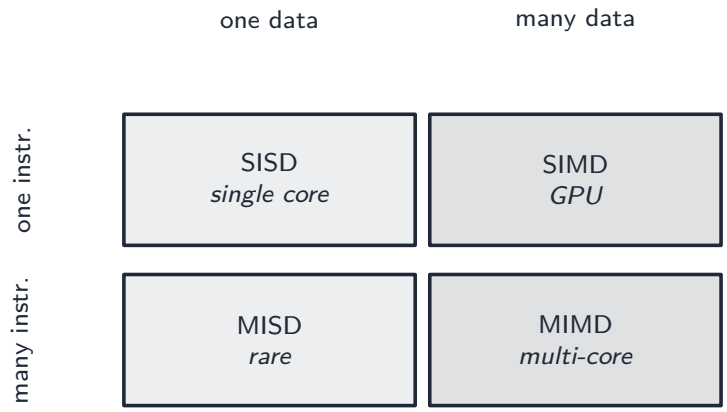
■ 1 Pipelining



A **pipeline** 流水线 overlaps the stages (fetch, decode, execute, memory, write-back) of different instructions, so once it is full, one instruction finishes per cycle. A **hazard** 冒险 can stall it.

Worked example. Without pipelining, 4 instructions of 5 stages each take about 20 cycles (finish one, then start the next). With a full 5-stage pipeline they overlap, so the 4 finish in about 8 cycles — once full, one completes every cycle.

2 Flynn's taxonomy



Flynn's taxonomy 弗林分类 sorts computers by instruction and data streams. **SIMD** 单指令多数据 does one instruction on many data items at once (GPUs).

Worked example. A GPU brightening a photo applies the same "add to brightness" instruction to millions of pixels at the same time — one instruction, many data items: that is SIMD.

Watch out: a pipeline does *not* make a single instruction finish sooner — it raises *throughput* (how many finish per second) by overlapping them. A hazard, such as a

> branch, can stall the pipeline and lose that gain.

Practice

Try these in order.

15.1 What does a *pipeline* do? [1]

15.2 Once the pipeline is full, how many instructions finish per cycle? [1]

15.3 What is it called when a pipeline has to stall? [1]

15.4 What does *SIMD* do (in idea)? [1]

15.5 Which kind of hardware is good at SIMD? [1]

15.6 Explain why a pipeline raises the number of instructions completed per second, even though each single instruction still takes the same time to pass through. [2]

15.7 Challenge. Editing every pixel of a large image with the same operation is much faster on a GPU than on an ordinary CPU. Explain why, using the idea of SIMD. [2]