B38DF
Computer Architecture and Embedded Systems

Alexander Belyaev

Heriot-Watt University
School of Engineering & Physical Sciences
Electrical, Electronic and Computer Engineering

E-mail: a.belyaev@hw.ac.uk
Office: EM2.29

Based on the slides prepared by Dr. Mustafa Suphi Erden

Parallelism – Improving Performance

Two forms of parallelism:

- **Instruction-level parallelism**

  Parallelism within individual instructions to get more instructions/sec

- **Processor-level parallelism**

  Multiple CPUs work together on the same problem

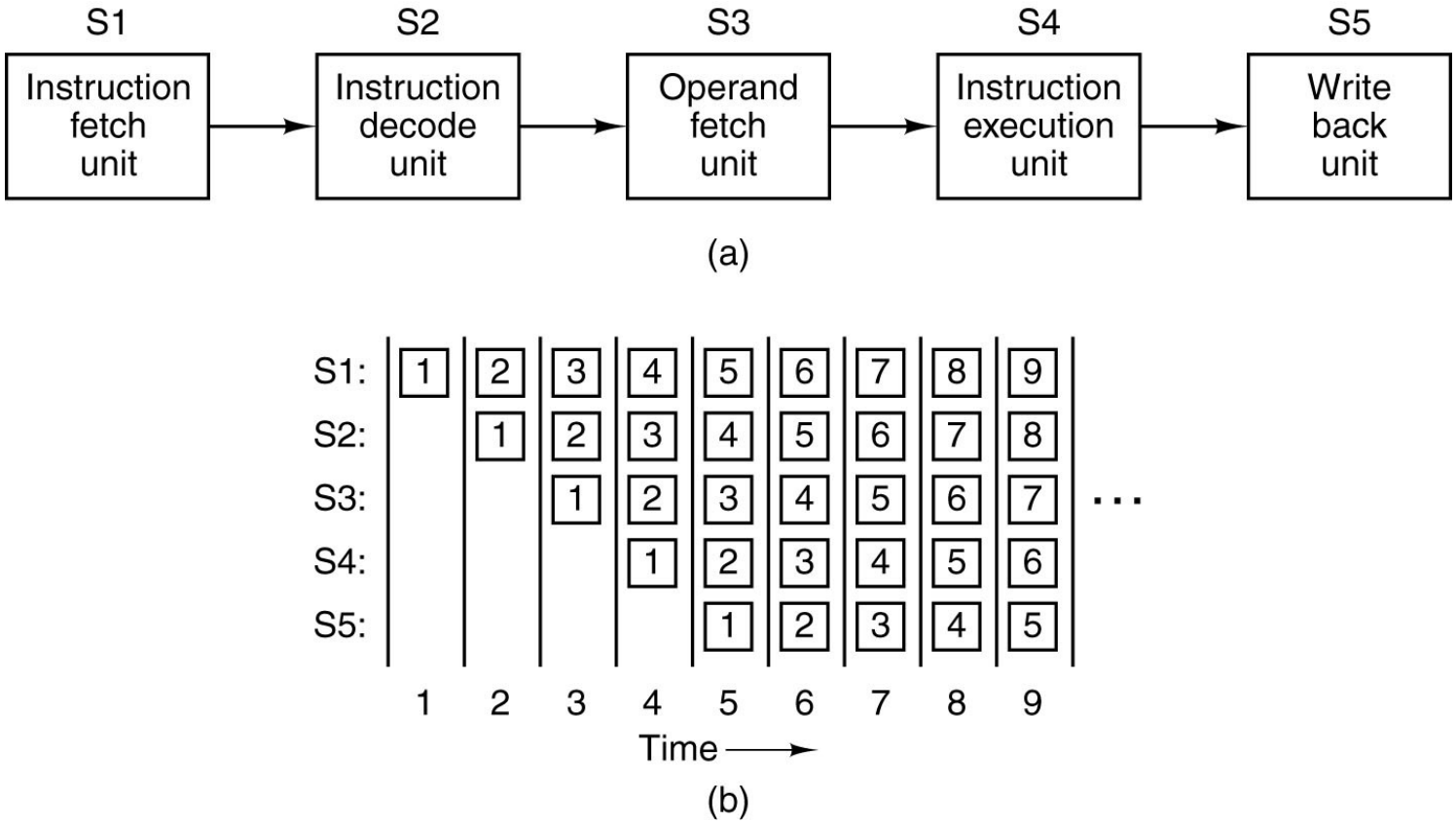
Instruction-level Parallelism

- A major bottle-neck in instruction execution speed:

  Fetching of instructions from memory

- **Prefetch** buffer registers (→ first step solution)
  - Fetch instruction from memory in advance to the prefetch registers
  - Instruction execution in two parts: fetching + actual execution
- **Pipelining** (→ profound solution)
  - Instruction execution in many (a dozen or more) parts

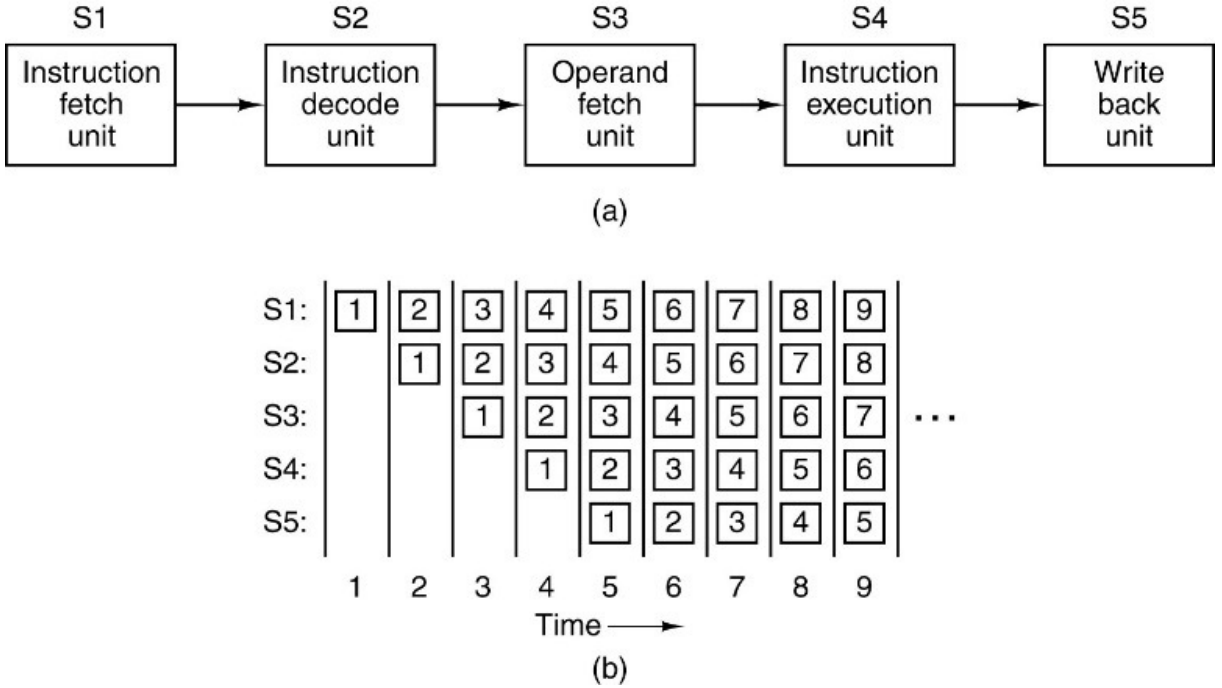
Pipelining



a) A **five-stage** pipeline

b) The state of each stage as a function of time. **Nine clock cycles** are illustrated

Pipelining – Processing Rate Computation



- Clock cycle time:

  2 nsec (assume)

- Instruction processing time

  5 × 2nsec = 10 nsec

- **Without pipelining:**

  Processing Rate = 1/10 nsec

  = 100 MIPS (Millions of Instructions Per Second)

- **With pipelining:**

  Processing Rate = 5 × 1/10 nsec

  = 500 MIPS (Millions of Instructions Per Second)

Latency and Processor Bandwidth

- **Latency:**

  How long it takes to execute an instruction

- **Processor bandwidth**

  How many MIPS (millions of instructions per second) the CPU has

---

(Clock) Cycle time *T* nanoseconds (nsec)

n stages in pipeline

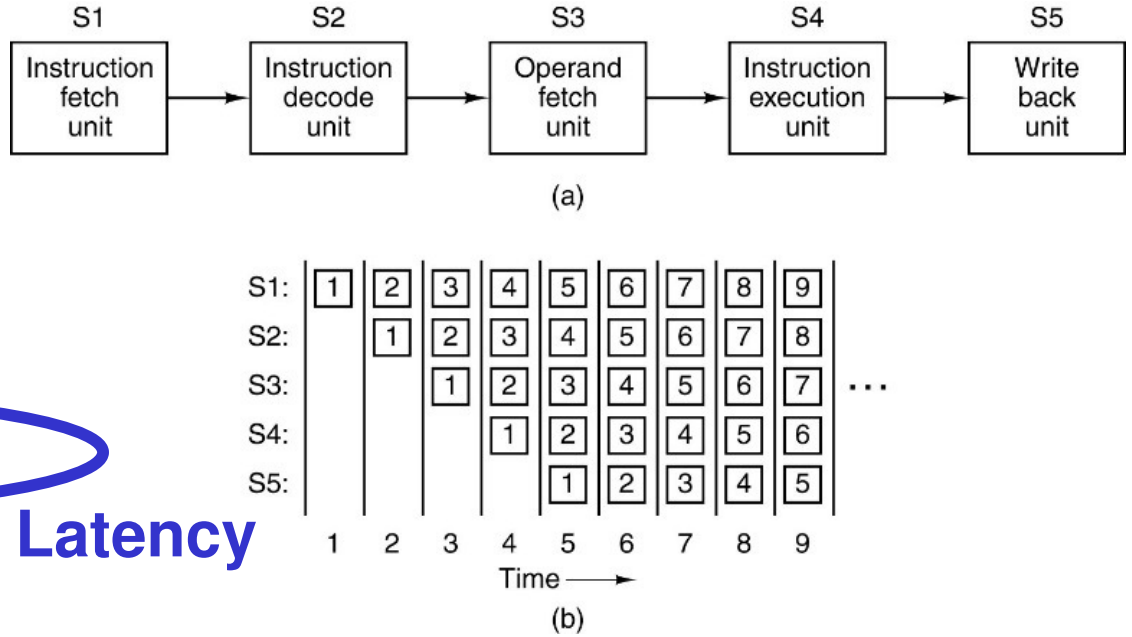
Latency = *nT*

➢ *One instruction completes every clock cycle*

Number of instructions executed per second = $10^9 / T$

Bandwidth = $(10^9 / T) / 10^6$ MIPS = $1000/T$ MIPS

Pipelining – Processing Rate Computation



- Clock cycle time:

  2 nsec (assume)

- Instruction processing time

  5 × 2nsec = 10 nsec

Latency

- **Without pipelining:**

  Processing Rate = 1/10 nsec

  = 100 MIPS (Millions of Instructions Per Second)

- **With pipelining:**

  Processing Rate = 5 × 1/10 nsec

  = 500 MIPS (Millions of Instructions Per Second)

Throughput (Processor bandwidth)

Performance Analysis

- **How much does the performance of a computing system improve when its components are improved?**

- A task executed by a system whose resources are improved compared to an initial similar system can be split up into two parts:
  - a part that does not benefit from the improvement;
  - a part that benefits from the improvement.

Source: Wikipedia

Performance Analysis

Example:

- A computer program that processes files from disk.
- A part of that program may scan the directory of the disk and create a list of files internally in memory.
- Another part of the program passes each file to a separate thread for processing.
- The part that scans the directory and creates the file list cannot be speed up on a parallel computer, but the part that processes the files can.

Source: Wikipedia

Amdahl's Law and its Derivation

Amdahl's law is a formula which gives the theoretical speedup in latency of the execution of a task at fixed workload that can be expected of a system whose resources are improved. It is named after computer scientist Gene Amdahl who presented it 1967.

Amdahl's law is often used in parallel computing to predict the theoretical speedup when using multiple processors..

$$S_{\text{latency}}(s) = \frac{1}{(1-p)+p/s}$$

Source: Wikipedia

Amdahl's Law and its Derivation

For example, if a program needs 20 hours using a single processor core, and a particular part of the program which takes one hour to execute cannot be parallelized, while the remaining 19 hours ($p = 0.95$) of execution time can be parallelized, then regardless of how many processors are devoted to a parallelized execution of this program, the minimum execution time cannot be less than that critical one hour. Hence, the theoretical speedup is limited to at most 20 times ($1/(1 - p) = 20$). For this reason, parallel computing with many processors is useful only for highly parallelizable programs.

$$S_{\text{latency}}(s) = \frac{1}{(1-p)+p/s}$$

Source: Wikipedia

Amdahl's Law and its Derivation

- S_{latency} is the theoretical speedup of the execution of the whole task;
- s is the speedup of the part of the task that benefits from improved system resources;
- p is the proportion of execution time that the part benefiting from improved resources originally occupied.

$$S_{\text{latency}}(s) = \frac{1}{(1-p) + p/s}$$

$$S_{\text{latency}}(s) < S_{\text{latency}}(\infty) = \frac{1}{(1-p)}$$

Source: Wikipedia

Amdahl's Law and its Derivation

- T is the execution time of the whole task before the improvement.
- p is the proportion of execution time that the part benefiting from improved resources originally occupied.

$$T = (1-p)T + pT$$ ← Before speedup

$$T(s) = (1-p)T + \frac{p}{s}T$$ ← After speedup

$$T(s) = \left(1 - p + \frac{p}{s}\right)T$$

$$S_{\text{latency}}(s) = \frac{T}{T(s)} = \frac{1}{1 - p + p/s}$$

Source: Wikipedia

Example 1

- If 30% of the execution time may be the subject of a speedup, p will be 0.3.

- If the improvement makes the affected part twice as fast: $s=2$.

- Amdahl's law states that the overall speedup of applying the improvement will be:

$$S_{\text{latency}}(s) = \frac{1}{1-0.3+0.3/2} = \frac{1}{0.85} = 1.18$$

Source: Wikipedia

Example 2

We are given a serial task which is split into four consecutive parts, whose percentages of execution time are

$$p1 = 0.11, \quad p2 = 0.18, \quad p3 = 0.23, \quad p4 = 0.48.$$

We are told that:

1st part is not sped up, so $s1 = 1$,
2nd part is sped up 5 times, so $s2 = 5$,
3rd part is sped up 20 times, so $s3 = 20$,
4th part is sped up 1.6 times, so $s4 = 1.6$.

$$T = T_1 + T_2 + T_3 + T_4 = p_1 T + p_2 T + p_3 T + p_4 T$$

$$T(s) = \frac{p_1}{s_1} T + \frac{p_2}{s_2} T + \frac{p_3}{s_3} T + \frac{p_4}{s_4} T$$

$$T/T(s) = 1/(p_1/s_1 + p_2/s_2 + p_3/s_3 + p_4/s_4)$$

By using Amdahl's law, the overall speedup is

$$S_{\text{latency}} = \frac{1}{\frac{p_1}{s_1} + \frac{p_2}{s_2} + \frac{p_3}{s_3} + \frac{p_4}{s_4}} = 2.2$$

Notice how the 5 times and 20 times speedup on the 2nd and 3rd parts respectively don't have much effect on the overall speedup when the 4th part (48% of the execution time) is accelerated by only 1.6 times.

Source: Wikipedia

(run Maple code)

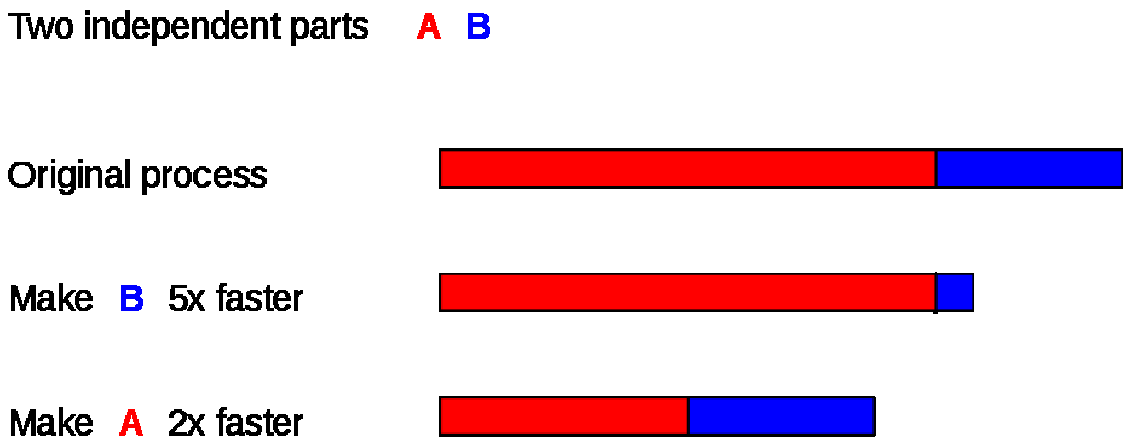
Example 3

A serial program consists of two parts A and B for which $T_A = 3\text{s},\ T_B = 1\text{s}$.

If part B is made to run 5 times faster, then $S_{\text{latency,B}} = \dfrac{1}{0.75 + 0.25/5} = 1.25$

If part A is made to run 2 times faster, then $S_{\text{latency,B}} = \dfrac{1}{0.75/2 + 0.25} = 1.6$

Therefore, making
part A to run 2 times faster
is better than making
part B to run 5 times faster.



Source: Wikipedia

Percentage Improvement

The percentage improvement in speed can be calculated as

$$P_I = 100\left(1 - \frac{1}{S_l}\right)$$

Improving part A **by a factor 2** will increase overall program speed by factor 1.60, which makes it

$$P_I = 100\left(1 - \frac{1}{1.60}\right) = 37\%$$

faster than the original computation

Improving part B **by a factor 5** will increase overall program speed by factor 1.25, which makes it

$$P_I = 100\left(1 - \frac{1}{1.25}\right) = 20\%$$

faster than the original computation

Source: Wikipedia

Example 4

Amdahl's Law gives us a handy way to estimate the performance improvement we can expect when we upgrade a system component.

- On a large system, suppose we can upgrade a CPU to make it 50% faster for £10,000 or upgrade its disk drives for £7,000 to make them 150% faster.

- Processes spend 70% of their time running in the CPU and 30% of their time waiting for disk service.

- An upgrade of which component would offer the greater benefit for the lesser cost?

Example 4

- The processor option offers a 30% speedup:

$$S = \frac{1}{1-0.7+0.7/1.5} \approx 1.30$$

- And the disk drive option gives a 22% speedup:

$$S = \frac{1}{1-0.3+0.3/2.5} \approx 1.22$$

- Each 1% of improvement for the processor costs £333, and for the disk a 1% improvement costs £318.

Should price/performance be your only concern?

Example 4

- The processor option offers a 30% speedup:

$$S = \frac{1}{1-0.7+0.7/1.5} = 1.30$$

- And the disk drive option gives a 22% speedup:

$$S = \frac{1}{1-0.3+0.3/2.5} \approx 1.22$$

- Each 1% of improvement for the processor costs £333, and for the disk a 1% improvement costs £318.

Should price/performance be your only concern?

If your disks are nearing the end of their expected life, or if you're running out of disk space, you might consider the disk upgrade even if it were to cost more than the processor upgrade.