The von Neumann Machine Model

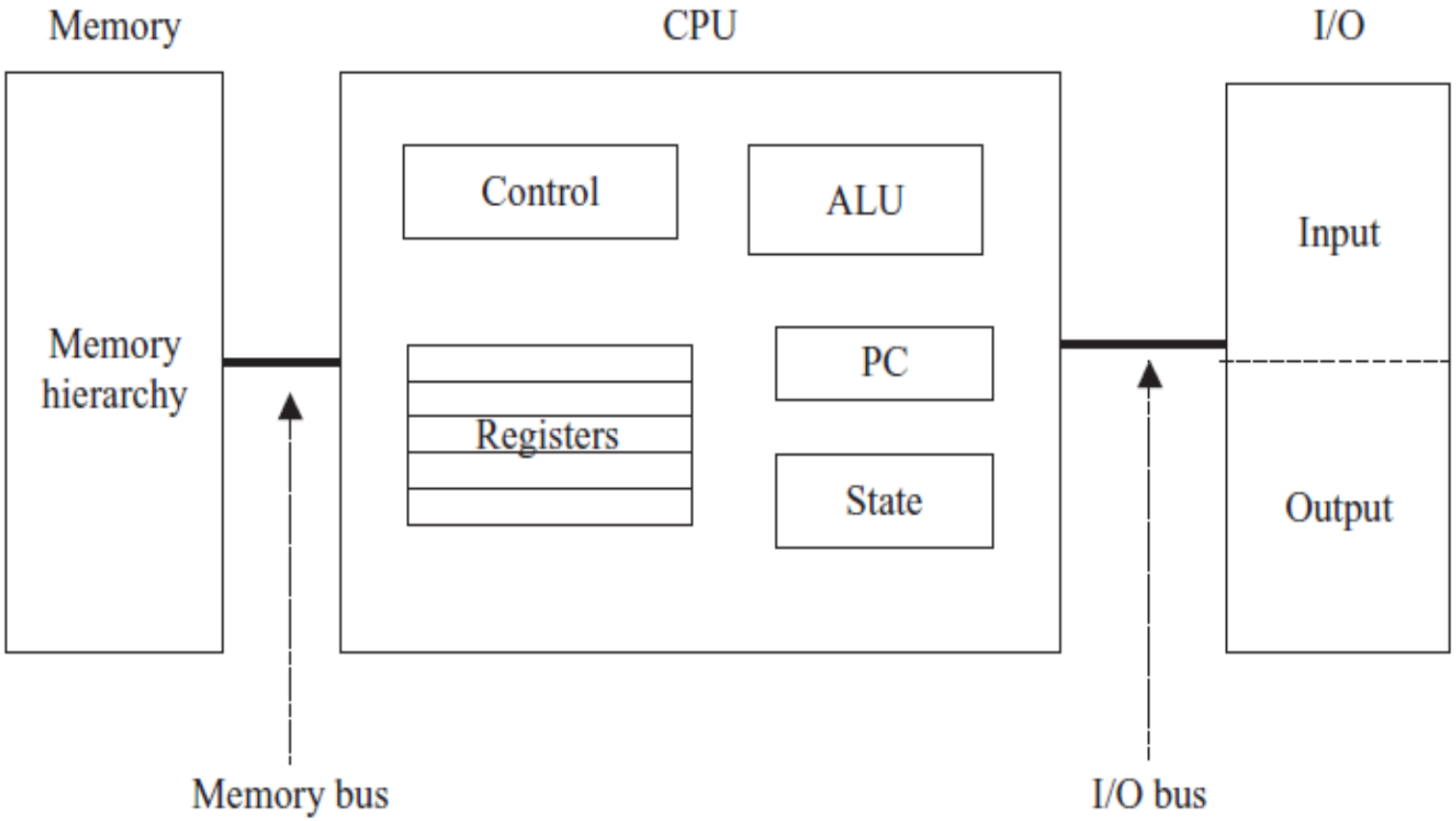


Figure 1.1. The von Neumann machine model.

Plateau shape: Freq.es stabilized

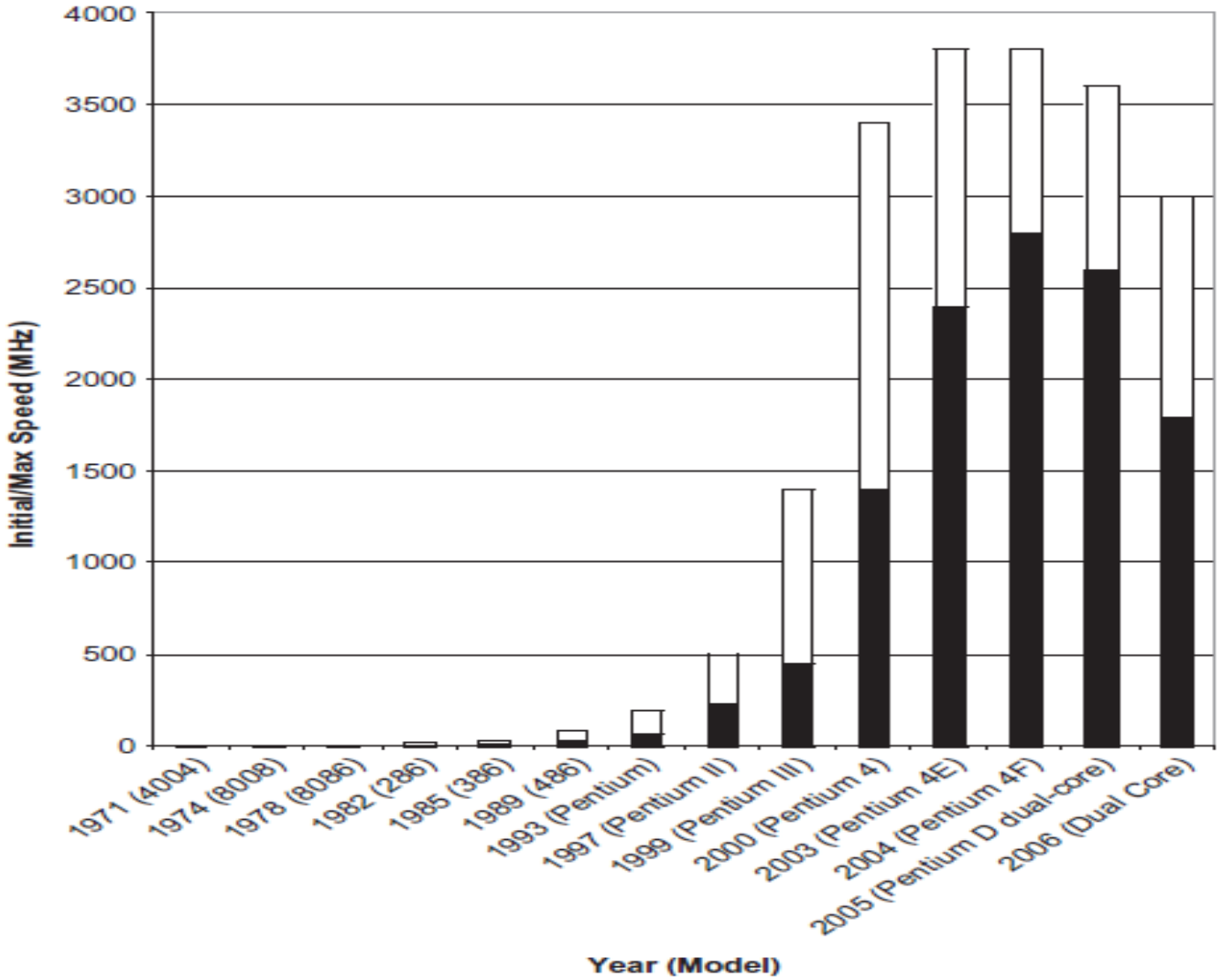


Figure 1.2. Evolution of Intel microprocessors speeds (black bars: frequency at introduction; white bars: peak frequency).

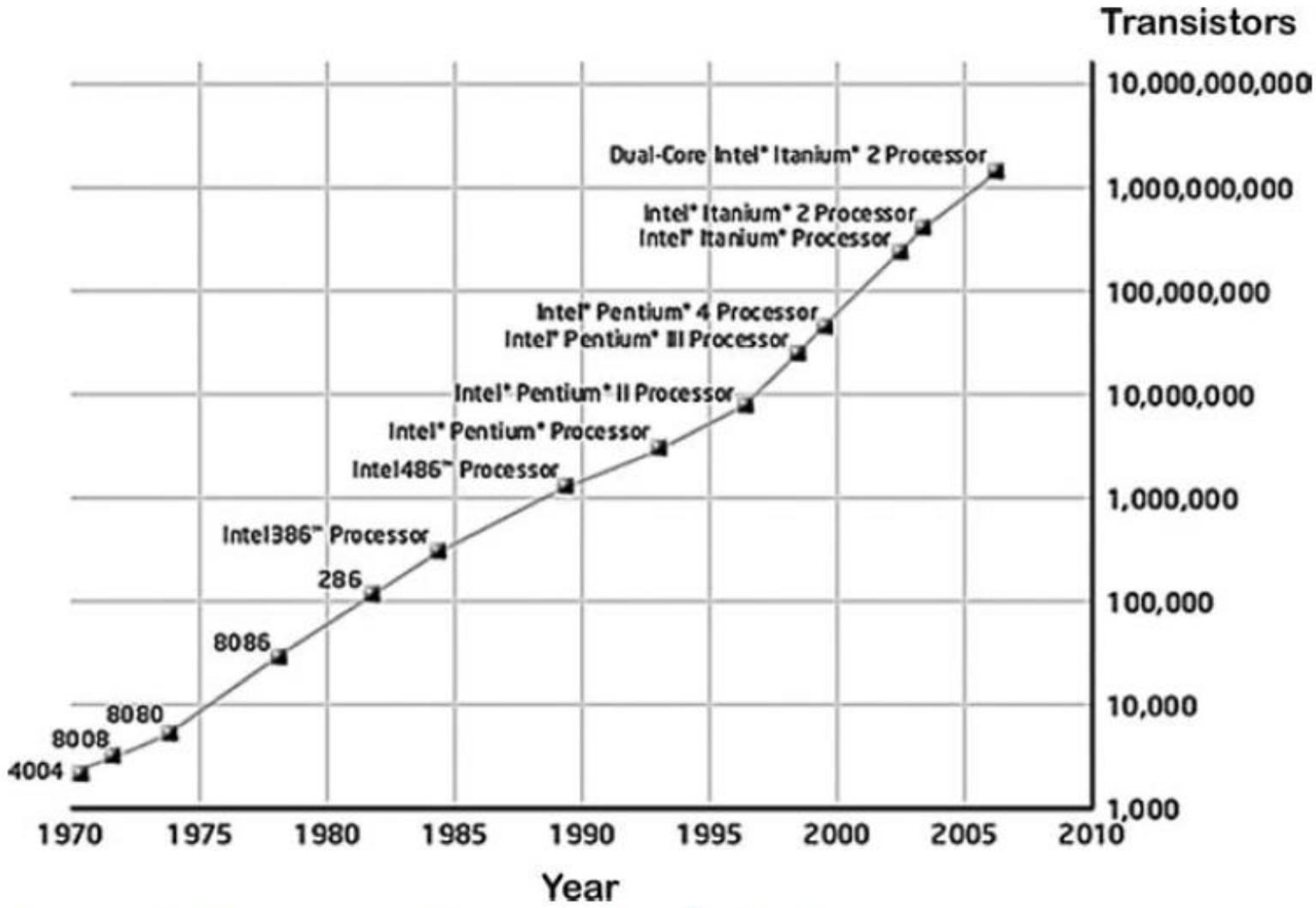


Figure 1.3. Illustration of Moore's law for the Intel microprocessors.

Performance Metrics

It depends ☺ H/w . S/w N/w Compiler , Programs of interest ,etc .Thus, we need:

Metrics that reflect the underlying architecture in a program independent fashion.

A suite of programs, or *benchmarks*.

Execution time & MIPS

- *EXCPU = Number of instructions × Time to execute one instruction*
- *EXCPU = Number of instructions(IC) × CPI × cycle time*
- *For Pipeline : CPI = 1 + CPIload latency + CPIbranches + CPIcache*
- *IPC= 1/CPI*
- *MIPS= IPC * F / 10^6*

Pipelining

- *CPI* = 1 + *CPIload latency* + *CPIbranches* + *CPIcache* (4)
- Mix of Instruction : can put as (instead of 1)

$$CPI = \sum_{i=1}^{c} f_i \times CPI_i + \sum_{x} CPI_x \qquad (6)$$

Example 1

- Determine CPI , IPC for a program that

Contains 15% load, 20% instructions following a load depend on its results and are stalled for 1 cycle

a) Assume further that 20% of instructions are branches, with 60% of them being taken. The penalty is 2 cycles if the branch is not taken, and it is 3 cycles if the branch is taken

- Then, 1 cycle is lost for 20% of the loads, 2 cycles are lost when a conditional branch is not taken, and 3 cycles are lost for taken branches. This can be written as
- $CPIload\ latency = 0.15 \times 0.2 \times 1 = 0.03$
- and
- $CPIbranches = 0.2 \times 0.6 \times 3 + 0.2 \times 0.4 \times 2 = 0.52$
- Thus
- $CPI = 1.55$ and $IPC = 0.65$.

b) A very simple optimization implementation for branches is to assume that they are not taken. There will be no penalty if indeed the branch is not taken, and there will still be a 3 cycle penalty if it is taken. In this case, we have $CPIbranches = 0.2 \times 0.6 \times 3 = 0.36$, yielding $CPI = 1.39$ and $IPC = 0.72$

Based on b, let us assume now that the hit ratio of loads in the first level cache is 95%. On a miss, the penalty is 20 cycles. For this case, we have: $CPIcache = (1 - 0.95) \times 20 = 1$ yielding (in the case of the branch-not-taken optimization) $CPI = 2.39$ and $IPC = 0.42$.

Performance Equations

$$\frac{Performance_A}{Performance_B} = \frac{EX_{CPU-B}}{EX_{CPU-A}} \qquad (7)$$

$$Speedup = \frac{Enhanced\ performance}{Original\ performance} = \frac{EX_{CPU-original}}{EX_{CPU-enhanced}} \qquad (8)$$

For parallel processors

$$Speedup_n = \frac{T_1}{T_n} \qquad Efficiency_n = Speedup_n / n \qquad (10)$$

$$Efficiency_n = Speedup_n / n \qquad (10)$$

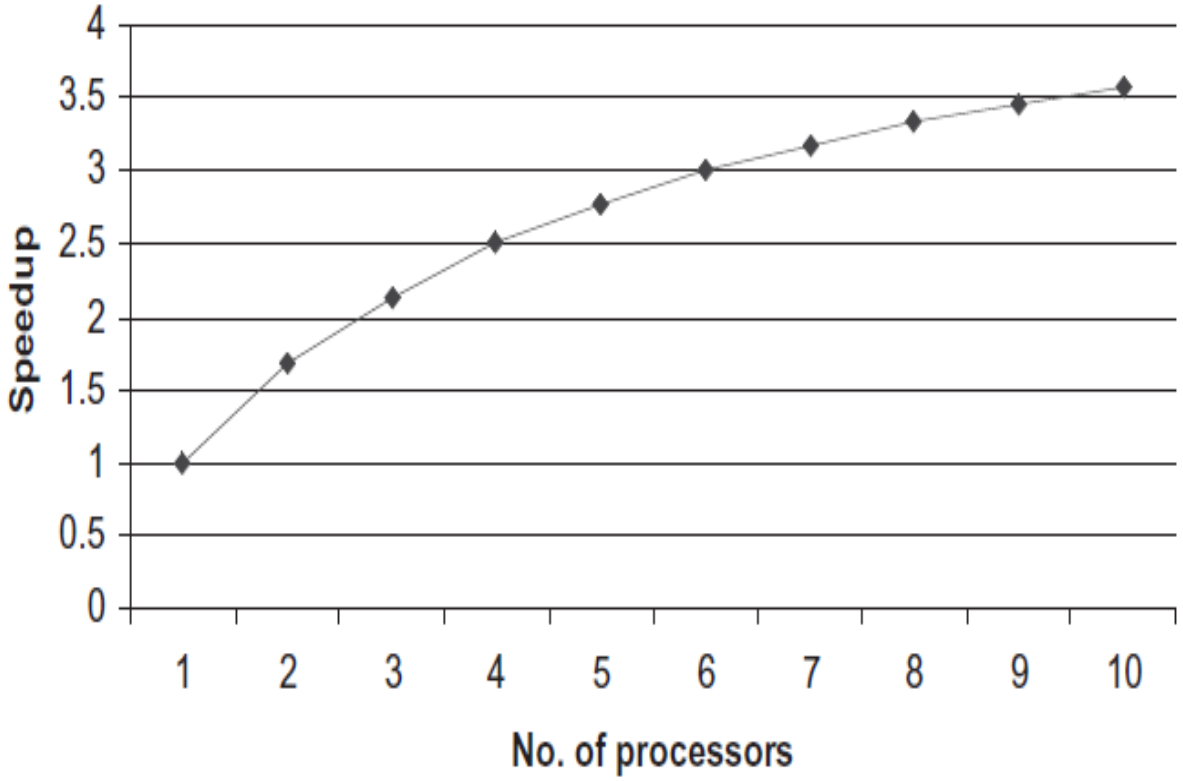
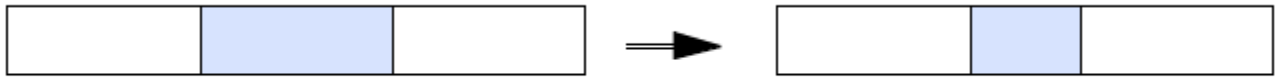


Figure 1.5. Illustration of Amdahl's law. The sequential portion of the program is 20%.

Amdahl ‘s law



$$\text{Execution time}_{new} = \text{Execution time}_{old} \times \left((1 - \text{Fraction}_{enhanced}) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}} \right)$$

$$\text{Speedup}_{overall} = \frac{\text{Execution time}_{old}}{\text{Execution time}_{new}} = \frac{1}{(1 - \text{Fraction}_{enhanced}) + \frac{\text{Fraction}_{enhanced}}{\text{Speedup}_{enhanced}}}$$

Suppose that we want to enhance the processor used for Web serving. The new processor is 10 times faster on computation in the Web serving application than the original processor. Assuming that the original processor is busy with computation 40% of the time and is waiting for I/O 60% of the time, what is the overall speedup gained by incorporating the enhancement?

Answer $\text{Fraction}_{\text{enhanced}} = 0.4, \text{Speedup}_{\text{enhanced}} = 10, \text{Speedup}_{\text{overall}} = \dfrac{1}{0.6 + \dfrac{0.4}{10}} = \dfrac{1}{0.64} \approx 1.56$

Example

A common transformation required in graphics processors is square root. Implementations of floating-point (FP) square root vary significantly in performance, especially among processors designed for graphics. Suppose FP square root (FPSQR) is responsible for 20% of the execution time of a critical graphics benchmark. One proposal is to enhance the FPSQR hardware and speed up this operation by a factor of 10. The other alternative is just to try to make all FP instructions in the graphics processor run faster by a factor of 1.6; FP instructions are responsible for half of the execution time for the application. The design team believes that they can make all FP instructions run 1.6 times faster with the same effort as required for the fast square root. Compare these two design alternatives.

Answer

$$\text{Speedup}_{\text{FPSQR}} = \frac{1}{(1 - 0.2) + \frac{0.2}{10}} = \frac{1}{0.82} = 1.22$$

$$\text{Speedup}_{\text{FP}} = \frac{1}{(1 - 0.5) + \frac{0.5}{1.6}} = \frac{1}{0.8125} = 1.23$$

Example

Suppose we have made the following measurements:

Frequency of FP operations = 25%

Average CPI of FP operations = 4.0

Average CPI of other instructions = 1.33

Frequency of FPSQR= 2%

CPI of FPSQR = 20

Assume that the two design alternatives are to decrease the CPI of FPSQR to 2 or to decrease the average CPI of all FP operations to 2.5. Compare these two design alternatives using the processor performance equation.

$$\text{CPI}_{\text{original}} = \sum_{i=1}^{n} \text{CPI}_i \times \left(\frac{\text{IC}_i}{\text{Instruction count}} \right)$$

$$= (4 \times 25\%) + (1.33 \times 75\%) = 2.0$$

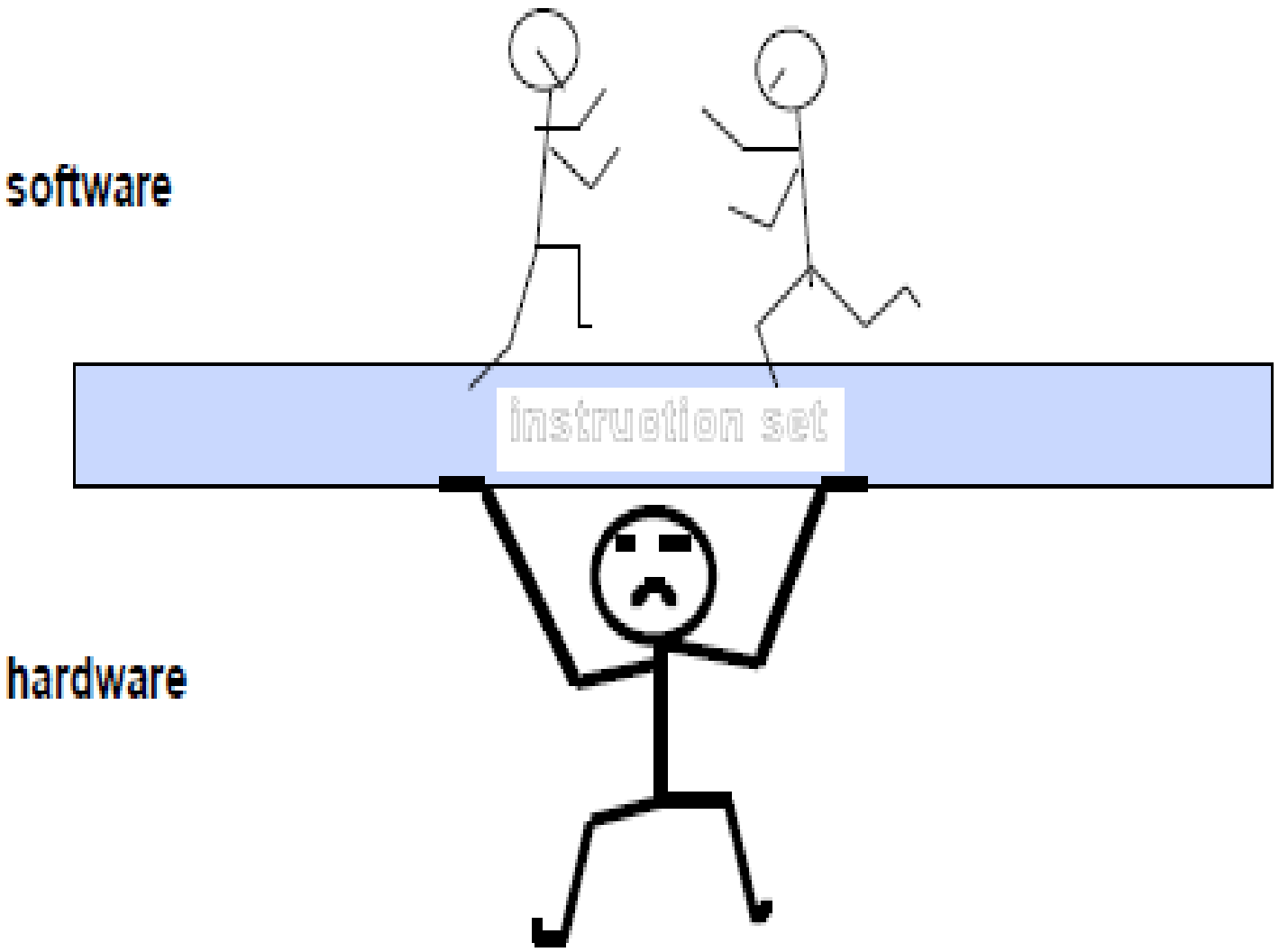
$$\text{CPI}_{\text{with new FPSQR}} = \text{CPI}_{\text{original}} - 2\% \times (\text{CPI}_{\text{old FPSQR}} - \text{CPI}_{\text{of new FPSQR only}})$$

$$= 2.0 - 2\% \times (20 - 2) = 1.64$$

CPInew FP = (75% × 1.33) + (25% × 2.5) = 1.62

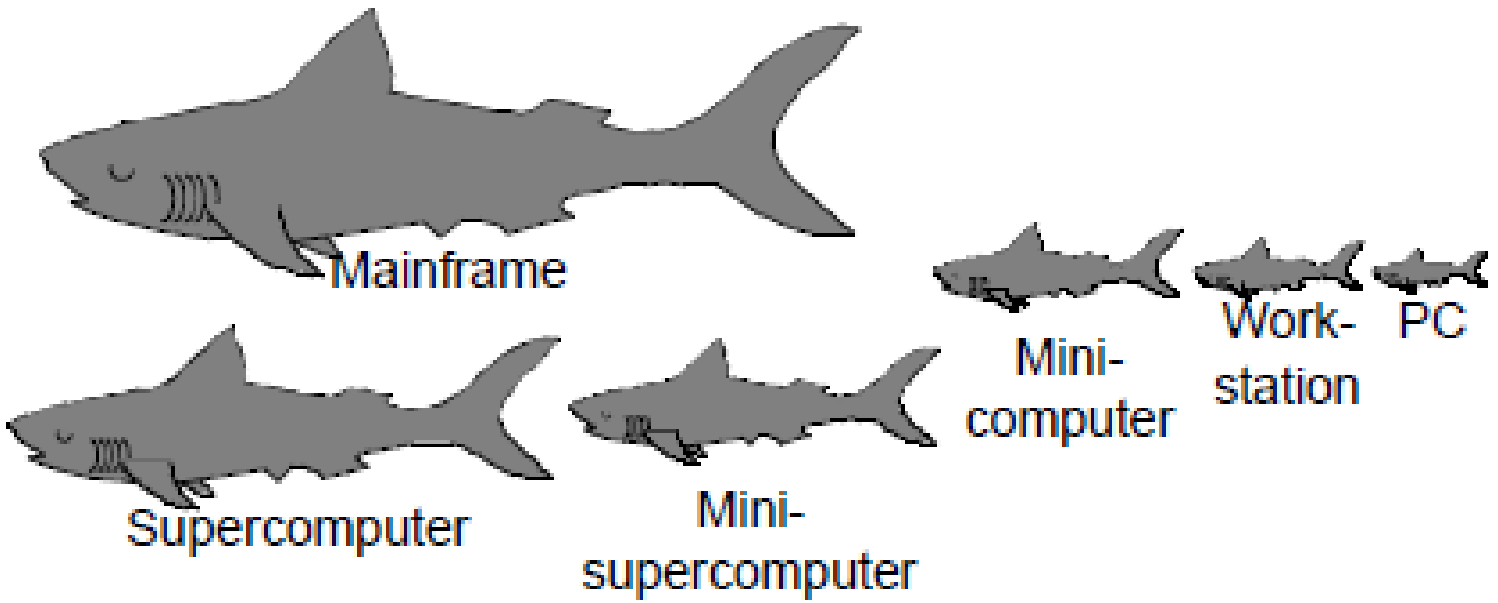
$$\text{Speedup}_{\text{new FP}} = \frac{\text{CPU time}_{\text{original}}}{\text{CPU time}_{\text{new FP}}} = \frac{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{original}}}{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{new FP}}}$$

$$= \frac{\text{CPI}_{\text{original}}}{\text{CPI}_{\text{new FP}}} = \frac{2.00}{1.625} = 1.23$$


software
instruction set
hardware

Big Fishes Eating Little Fishes

1985 Computer Food Chain



Mini-supercomputer

Mini-computer

1995 Computer Food Chain

Mainframe

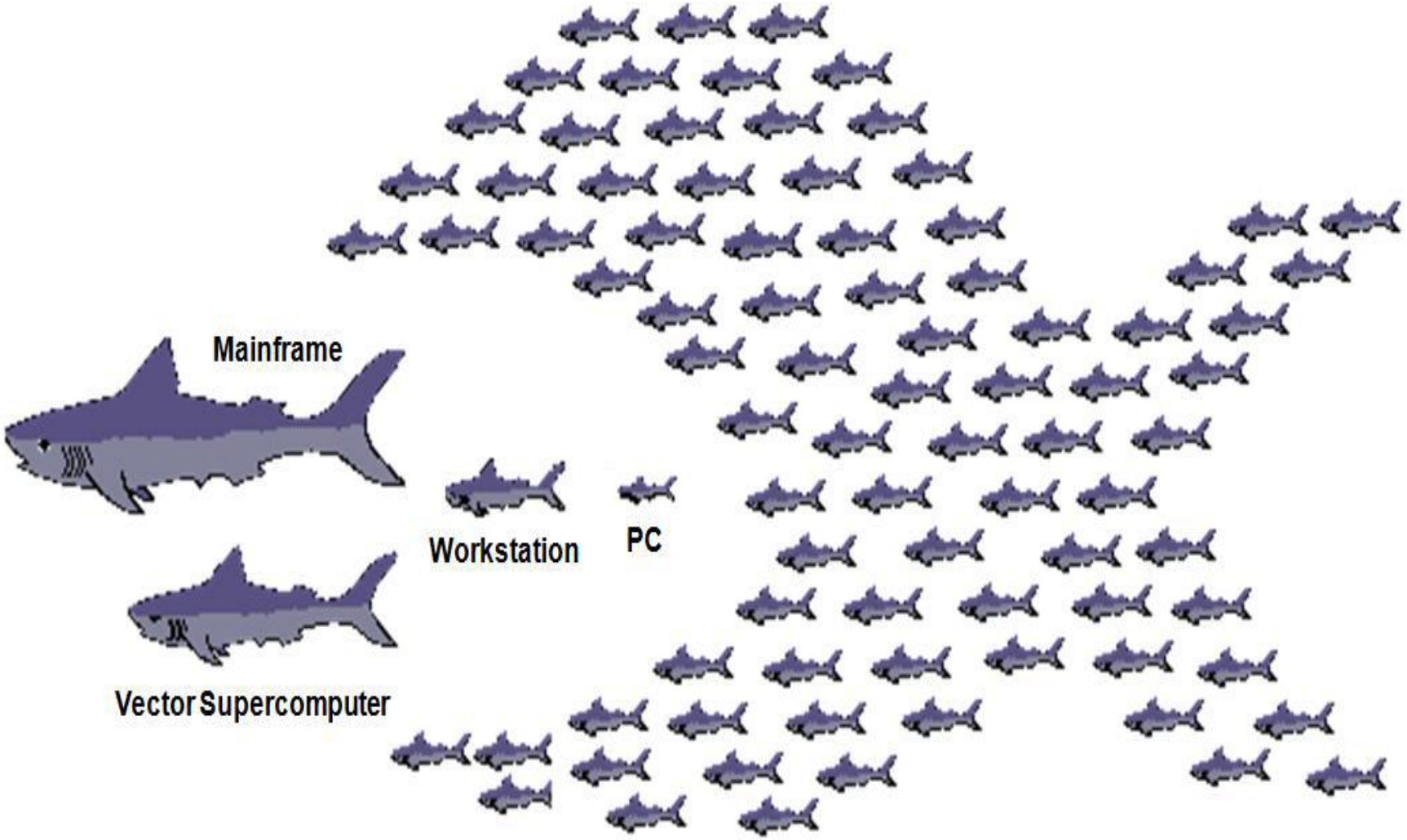
Supercomputer

Work-station

PC

Massively Parallel Processors

Now who is eating whom?


Mainframe
Workstation
PC
Vector Supercomputer
Cluster