2.2. Benchmark Suites

- real programs
  - × synthetic benchmarks
  - × toy programs
  - × Kernels
- geometric mean
- reference machine

$$\text{stddev} = \sqrt{\sum_{i=1}^{n} (\text{sample}_i - \text{mean})^2}$$

✓ SPEC,

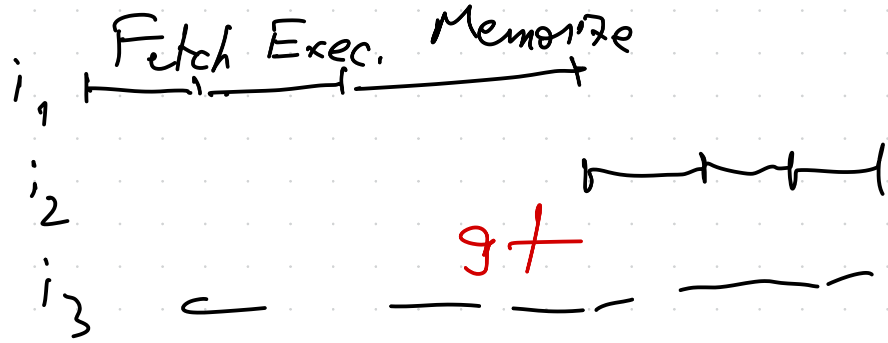
$$\text{Geom. mean} = \exp\left(\frac{1}{N} \times \sum_{i=1}^{N} \ln \text{Sample}_i\right)$$

$$\text{gstddev} = \exp\left(\sqrt{\frac{1}{N} \sum_{i=1}^{N} (\ln \text{Sample}_i - \ln \text{gmean})^2}\right)$$

2.3. Principles of Computer System Design

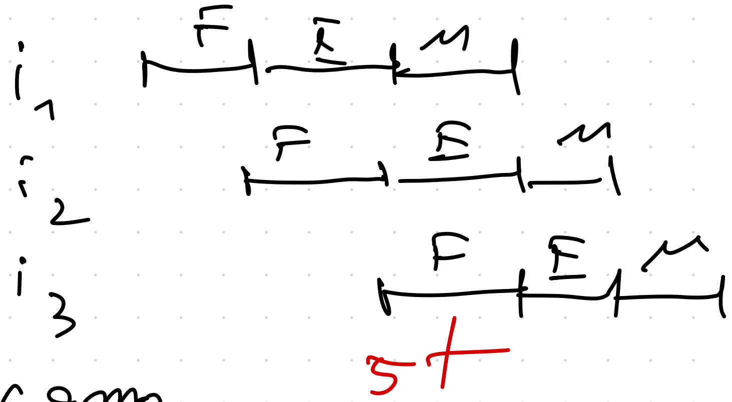
① Use parallelism when possible

- CLA
- Pipelining



Instruction Level Parallelism

ILP

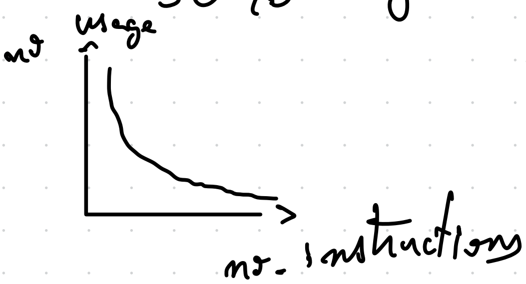


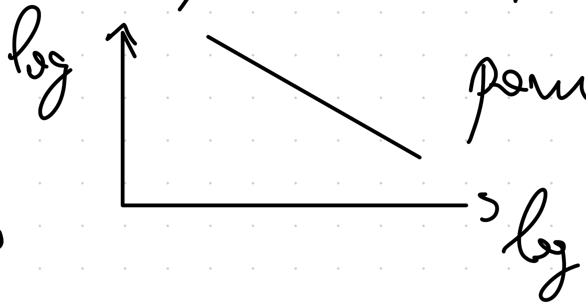
- multicore comp.

---

② Locality principle

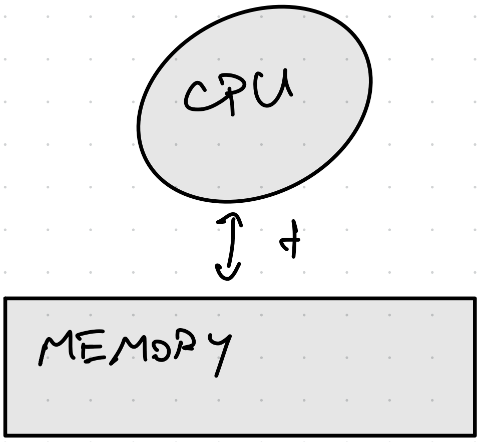
90% of the time, the comp. runs 10% of the code

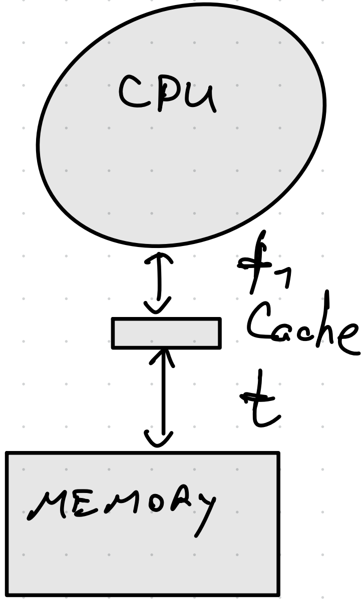




power law

Cache Memory → pt. cele mai utilizate operații





Cache Hit $\quad H = t_1$

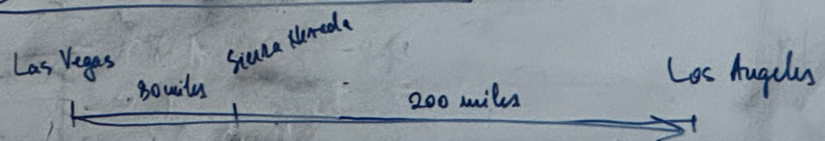
Cache Miss $\quad M = t + t_1$

---

(3) Make the common case faster

2.3 Principles of computer system design

(3) Make the common case faster



Foot 4 miles/hr.

20 hr.

Bike - 10 miles/hr

Hyundai - 50 miles/hr

Ferrari → 120 miles/hr

Rocket car → 200 miles/hr.

Second section vehicle	Time in second section (hrs.)	Speedup second section	Total Time (hrs.)	Overall speedup
Foot	50	1	70	1
Bike	20	2.5	40	1.75
Hyundai	4	12.5	24	2.92
Ferrari	1.7	29.4	21.7	3.225
Rocket car	1	50	21	3.333

$\frac{50}{20} = 2.5$

5 ÷ 3 = 1.6
3
20
18

$\frac{50}{4} = \frac{25}{2} = 12.5$

$$\text{Overall speedup} = \frac{\text{Time without optimization}}{\text{Time with optimization when possible}} =$$

$$= \frac{\cancel{\text{Time without optimization}}}{\cancel{\text{Time without optimization}}\,(1 - \text{Fraction optimized}) + \frac{\cancel{\text{Time without optimization}}}{\text{Speedup optimized}} \cdot \text{Fraction optimized}} =$$

$$= \frac{1}{(1 - \text{Fraction optimized}) + \frac{\text{Fraction optimized}}{\text{Speedup optimized}}}$$

Amdahl's Law

$$\lim_{\text{Speedup-optimized} \to \infty} \text{Overall-speedup} = \frac{1}{1 - \text{Fraction optimized}}$$

2.4. Computer Performance Equation

$$CPU_{time} = \text{Total no. of clock cycles} \times \text{clk cycle time}$$

$$\text{Clk frequency} = \frac{1}{\text{time}}$$

total number of clk cycles =

$$\to \underbrace{\text{Instruction Count}}_{IC} \times \underbrace{\text{Clk cycles per Instr}}_{CPI}$$

$$CPU_{time} = IC \times CPI \times \text{Clk cycle time}$$

$$\frac{\text{Instruction}}{\text{Program}} \times \frac{\text{Clk cycles}}{\text{Instr.}} \times \frac{\text{Time}}{\text{Clk cycle}} = \frac{\text{Time}}{\text{Program}}$$

Instruction types → ALU, Compare, Branch, MOV

no of type i instr.

$$IC = IC_1 + IC_2 + IC_3 \ldots IC_n = \sum_{i=1}^{n} IC_i$$

$$CPI = \frac{IC_1 \times CPI_1 + IC_2 \ldots}{IC}$$

$$= \sum_{i=1}^{n} \underbrace{\frac{IC_i}{IC}}_{\text{fraction of type } i \text{ instructions}} \times CPI_i$$

$$CPU_{time} = IC \times CPI \times \text{clk cycle time}$$

RISC ⟷ CISC

Example 1

FLOPS FP instr. freq = 25%

$CPI_{FP} = 4.0$ c.c.

$CPI_{other} = 1.33$ c.c.

Freq. of FP SQRT = 2%

$CPI_{FP\,SQRT} = 20$ c.c.

(A) Decrease $CPI_{FPSQRT} = 2$

(B) Decrease $CPI_{FP} = 2.5$

$CPU_{time} = IC_o \times CPI_o \times$ Clk cycle time$_{original}$

$CPI_o = 0.25 \times 4 + 0.75 \times 1.33 = 2$ c.c.

$CPU_{time\,A} = IC_o \times CPI_A \times$ Clk cycle time$_o$

$CPI_A = 2 - (20-2) \times 0.02 = \underline{\underline{1.64}}$ cc

$CPI_B = 0.25 \times 2.5 + 0.75 \times 1.33 = \underline{\underline{1.625}}$ c.c.

$2 - (4 - 2.5) \times 0.25$

Example 2 Conditional branch

CPU_A

$CMP \quad r_1, r_2$

$BNE \quad address$ → modifica program counter

Branch not equal ($\neq$)

20% CPU Branch

addr.

$CPU_B \quad BNE \quad r_1, r_2, addr.$

$CPI_B = 2 \, c.c.$

$CPI_{other} = 1 \, c.c.$

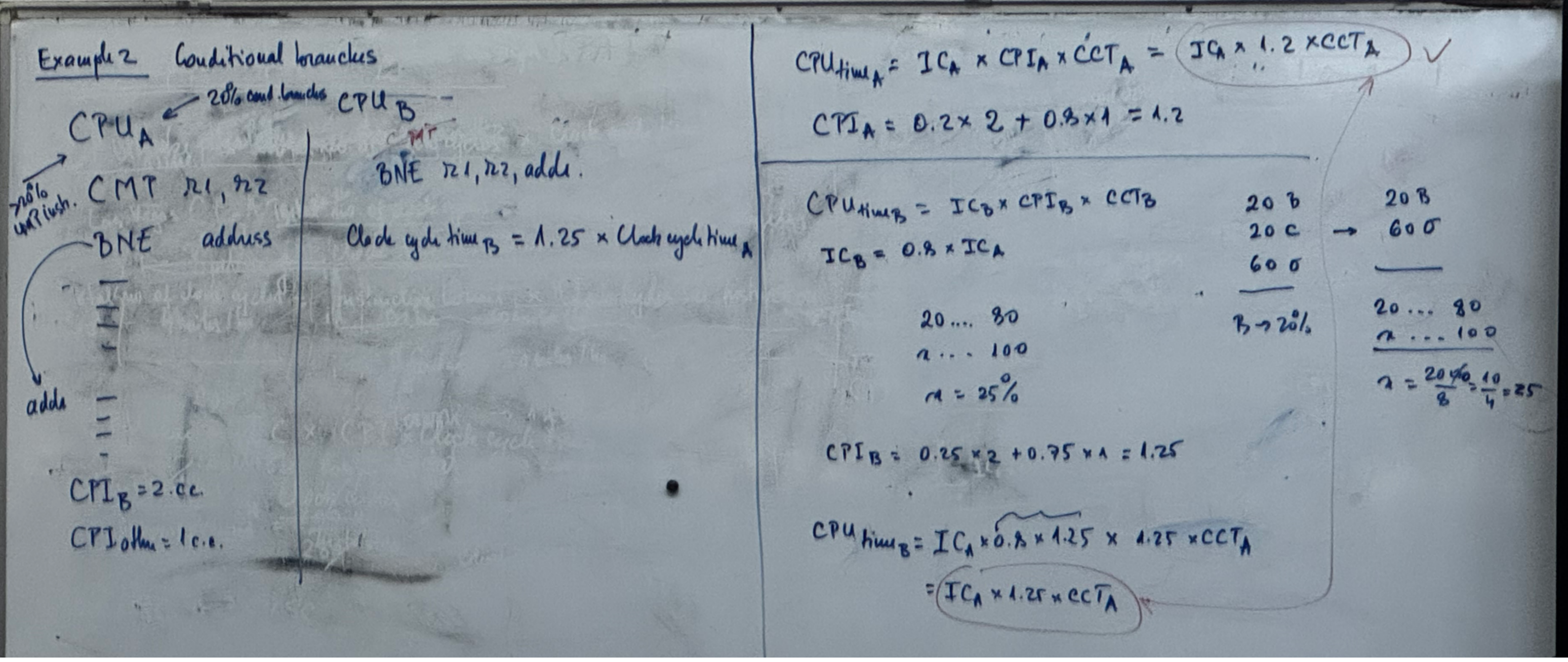
$\text{clk cycle time } B = 1.25 \times \text{clk cycle time } A$

$$CPU_{time\,A} = IC_A * CPI_A * CCT_A$$

$$CPI_A = \underset{(Branch)}{0.2 \times 2} + \underset{(no\ branch)}{0.8 \times 1} = 1.2$$

$$CPU_{time\,B} = IC_B * CPI_B * CCT_B$$

$$IC_B = 0.8 * IC_A$$



Exemplu 3 Load / Store machine

Instr.	Freq	CPI
ALU	43%	1
Load	21%	2
store	12%	2
Branch	24%	2

25% of all ALU have 1 operand from mem.

LDR r_1, [r_2]

ADD r_3, r_3, r_1 ; $r_3 \leftarrow r_3 + r_1$

new ALU 2 c.c.

new CPU → $CPI_{Branch} = 3$