Quantitative Performance Theory

Amdahl Equation for Multiprocessors

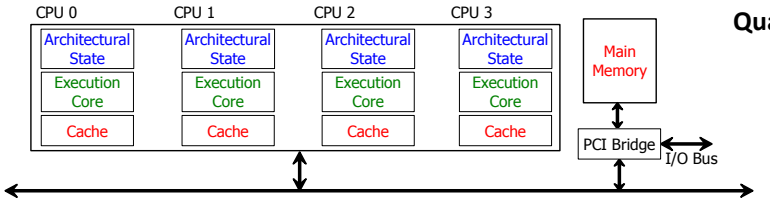
Symmetric Multiprocessor (SMP)

N equivalent microprocessors
Communication network between processors
Operating system runs on one+ processor
OS assigns tasks to processors by some scheduling system

Amdahl equation for SMP

$$S = \frac{1}{(1-F_P)+\frac{F_P}{N}}$$

F_P = fraction of work that can be enhanced (parallelized)
N = speedup for part to be enhanced (number of processors)



Quad Core CPU

Example of Amdahl Equation

For multiprocessor system

$$S = \frac{1}{(1-F_P)+\frac{F_P}{N}}$$

F_P = fraction of work that can be enhanced (parallelized)
N = speedup for part to be enhanced (number of processors)

Typical small Dell file server

N = 8 Xeon processors
F_p = 80% of work can be parallelized

$$S = \frac{1}{(1-0.80)+\frac{0.80}{8}} = \frac{1}{0.20+0.10} \cong 3.33$$

If number of processors were unlimited

$$S = \frac{1}{(1-F_P)+\frac{F_P}{N}} \xrightarrow{N\to\infty} \frac{1}{1-F_P} = \frac{1}{1-0.80} = 5$$

Maximum speedup is 5
Future enhancements require more parallelization F_p

Basic Performance Measures

Run Time (זמן ריצה)

Elapsed time T from start to finish of a defined program task

Latency (זמן המתנה)

Excess response time — depends on context

Throughput (תפוקה)

Number of defined tasks performed per unit time

$$\text{Throughput} = \frac{1}{\text{T} + \text{ latency between tasks}}$$

Enhancement (שינוי מבנה)

Change to system ⇒ new run time T '

Speedup (שיפור)

$$S = \frac{T}{T'} \qquad S > 1 \Rightarrow T' < T$$

Processor Performance Enhancements

Hardware Enhancements

- Clock rate
- Instruction implementation
- Memory organization
- Number of processing elements (CPUs, ALUs, registers)

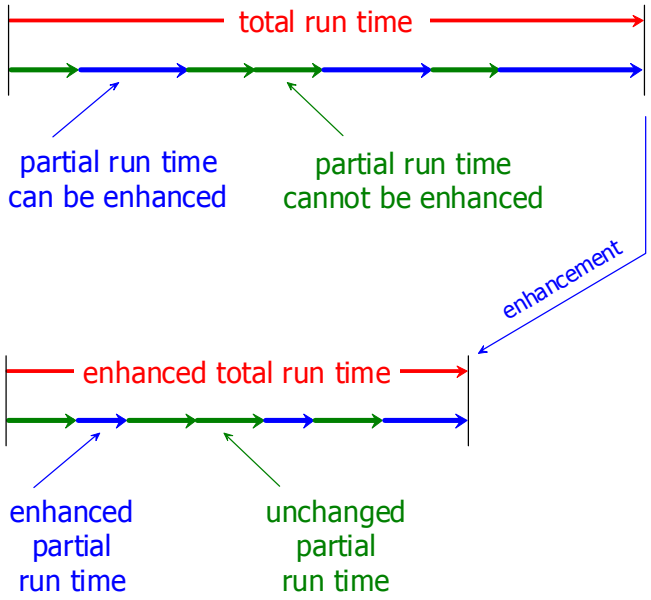
Software Enhancements

- Run time optimizations
- Compiler
- Operating system

Enhanced Run Time

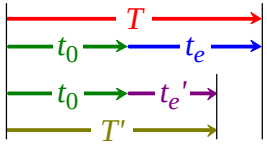
- Run time = sum of partial run times
- Enhancement $\Rightarrow$ partial run times are longer, shorter, or unchanged
- S > 1 $\Rightarrow$ Sum of new partial run times < sum of old partial run times

Run Time Enhancements



Amdahl Equation

Definitions



- T total run time of a task
- T' total run time of a task after enhancement
- t_e partial run time that can be enhanced
- t_e' partial run time that can be enhanced, after enhancement
- t_0 partial run time that cannot be enhanced
- F_e fraction of run time that can be enhanced = t_e / T
- S_e Speedup of portion of run time that can be enhanced = t_e / t_e'

$$S = \frac{T}{T'} = \frac{T}{t_0 + t'_e} = \frac{1}{\frac{t_0}{T} + \frac{t'_e}{T}} = \frac{1}{\frac{T - t_e}{T} + \frac{t'_e}{t_e}\frac{t_e}{T}} = \frac{1}{1 - F_e + \frac{1}{S_e}F_e}$$

Amdahl equation expresses speedup in terms of relative quantities
Actual run-times not needed if **RELATIVE ENHANCEMENTS** are known

Example of Amdahl Equation

Program partial run times

Integer Instructions	t_{INT}	**100 ms**
Float Instructions	t_{FP}	**300 ms**
Total Run Time	T	**400 ms**

Enhance partial run time of Float Instructions

Integer Instructions	t_{INT}	**100 ms**
Float Instructions	t_{FP}'	**200 ms**
Total Run Time	T'	**300 ms**

Speedup from actual run times

$$S = \frac{T}{T'} = \frac{400\ \text{ms}}{300\ \text{ms}} = \frac{4}{3}$$

$$F_e = \frac{t_{FP}}{T} = \frac{300\ \text{ms}}{400\ \text{ms}} = \frac{3}{4} = 75\%$$

$$S_e = \frac{t_{FP}}{t_{FP}'} = \frac{300\ \text{ms}}{200\ \text{ms}} = \frac{3}{2} = 1.50$$

Speedup from relative enhancements

$$S = \frac{1}{1 - F_e + \frac{F_e}{S_e}} = \frac{1}{1 - \frac{3}{4} + \frac{3/4}{3/2}} = \frac{1}{\frac{1}{4} + \frac{1}{2}} = \frac{4}{3}$$

Application of Amdahl Equation

On some CPU

Float (FP) instructions account for 50% of total run time

Square root (FP) accounts for 20% of total run time

Choose between two alternative enhancements

Speedup of $S_e = 2$ for all FP instructions

Speedup of $S_e = 10$ for square root instruction

Enhancement 1

$$S_1 = \frac{1}{1 - F_e + \frac{1}{S_e} F_e} = \frac{1}{1 - 0.50 + \frac{0.50}{2}} = \frac{1}{0.50 + 0.25} \cong 1.33 \Rightarrow \boxed{33\% \text{ speedup}}$$

Enhancement 2

$$S_2 = \frac{1}{1 - F_e + \frac{1}{S_e} F_e} = \frac{1}{1 - 0.20 + \frac{0.20}{10}} = \frac{1}{0.80 + 0.02} \cong 1.22 \Rightarrow \boxed{22\% \text{ speedup}}$$

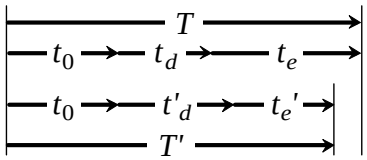
Generalized Amdahl Equation

Result of reasonable architectural change

Enhancements to most features

Degradations to some features

Overall enhancement



New definitions

t_d portion of run time that is degraded

t_d' portion of run time that is degraded, after degradation

F_d fraction of run time that is degraded = t_d / T

S_d Speedup of portion of run time that is degraded = t_d / t_d'

$$S = \frac{T}{T'} = \frac{T}{t_0 + t'_e + t'_d} = \frac{1}{\frac{T - t_e - t_d}{T} + \frac{t'_e}{t_e}\frac{t_e}{T} + \frac{t'_d}{t_d}\frac{t_d}{T}} = \frac{1}{1 - (F_e + F_d) + \frac{F_e}{S_e} + \frac{F_d}{S_d}}$$

Amdahl's "Law"

To make good architectural improvements

Focus on enhancements that positively affect most features

Ignore degradations that negatively affect few features

Example — simple "RISC" processor

94% of run time is 5 times faster than CISC processor

1% of run time is 10 times slower than CISC processor

5% of run time is same as for CISC processor

$$S = \frac{1}{1 - (F_e + F_d) + \frac{F_e}{S_e} + \frac{F_d}{S_d}}$$

$$= \frac{1}{1 - (0.94 + 0.01) + \frac{0.94}{5} + \frac{0.01}{1/10}}$$

$$= \frac{1}{0.05 + 0.19 + 0.10} \cong \boxed{2.94}$$

This RISC processor is (overall) about 3 times faster than CISC

Even though some operations are slower

Detailed Analysis of CPU Run Times

Amdahl equation requires relative run time data

Run time data requires measurements on running programs

Measurements on running programs require CPU implementation

CPU analysis predicts run time without building CPU

Assumptions:

Instructions can be grouped together according to resource usage

Example — **`ADD R1, R2, R3`** and **`SUB R1, R2, R3`**

All instructions in a group run in same number of clock cycles

Every clock cycle measures same unit of time

Instruction run time = clock cycle time × number of clock cycles

Group run time = instruction run time × instructions in group

Total run time = sum of instruction group run times

Definitions

T = total run time of program

t_i = total run time of instructions in group i

IC_i = number of instructions in group i (Instruction Count)

CPI_i = number of clock cycles to run 1 instruction in group i (Cycles Per Instruction)

N_i = number of clock cycles to run all instructions in group i

τ = seconds per clock cycle

R = clock rate = clock frequency = clock cycles per second = Hertz (Hz) = $1/\tau$

IC = total number of instructions in program

N = total number of clock cycles to run program

CPI = average number of clock cycles per instruction for the program

$quantity'$ = new value of $quantity$ after architectural change

CPU Equation

Clock cycles to run all instructions of type i

$$N_i = \text{instructions of type } i \times \frac{\text{clock cycles}}{\text{instruction of type } i} = IC_i \times CPI_i$$

Total clock cycles to run all instructions in program

$$N = \sum_{\text{all groups } i} N_i = \sum_i IC_i \times CPI_i$$

Average number of clock cycles per instruction for program

$$CPI = \frac{\text{total number of clock cycles to run program}}{\text{total number of instructions in program}} = \frac{N}{IC}$$

$$CPI = \frac{1}{IC}\sum_i N_i = \frac{1}{IC}\sum_i IC_i \times CPI_i = \sum_i CPI_i \times \frac{IC_i}{IC}$$

weighted average $\sum_i \frac{IC_i}{IC} = 1$

Ratio $\frac{IC_i}{IC}$ is proportion (percent) of instructions in group i

Example of CPU Equation

Program distribution

Instruction Type i	IC_i	CPI_i
Integer	5,000	4
Branch	1,000	12
Load / Store	4,000	8

$$IC = IC_{\text{int}} + IC_{\text{branch}} + IC_{\text{load/store}} = 5,000 + 1,000 + 4,000 = 10,000 \text{ instructions}$$

Instruction Type i	IC_i / IC	$N_i = IC_i \times CPI_i$
Integer	5000/10000 = 50%	4 × 5000 = 20,000 cycles
Branch	1000/10000 = 10%	12 × 1000 = 12,000 cycles
Load / Store	4000/10000 = 40%	8 × 4000 = 32,000 cycles

$N = 20,000$ cycles $+ 12,000$ cycles $+ 32,000$ cycles $= 64,000$ cycles

$CPI = N / IC = 64,000$ cycles $/ 10,000$ instuctions $= 6.4$ cycles per instruction

$$CPI = \sum_i CPI_i \times \frac{IC_i}{IC} = 4 \times 0.50 + 12 \times 0.10 + 8 \times 0.40 = 6.4 \text{ cycles per instruction}$$

CPU Run Time

Run time of one instruction of type i

$$\frac{\text{clock cycles}}{\text{instruction of type } i} \times \frac{\text{seconds}}{\text{clock cycle}} = CPI_i \times \tau$$

Run time for all instructions of type i

$$t_i = \text{instructions of type } i \times \frac{\text{clock cycles}}{\text{instruction of type } i} \times \frac{\text{seconds}}{\text{clock cycle}}$$

$$= IC_i \times CPI_i \times \tau$$

Total run time for program

$$T = \sum_{\text{all groups } i} t_i = \sum_i CPI_i \times IC_i \times \tau = \left(\sum_i CPI_i \times \frac{IC_i}{IC}\right) \times IC \times \tau$$

So

$$T = CPI \times IC \times \tau$$

$$= \text{clock cycles per instruction} \times \text{number of instructions} \times \text{clock cycle}$$

CPU Run Time — Example

For a certain CPU

- Instructions in a typical programs can be grouped as
  - 50% integer ALU instructions that run in 8 clock cycles
  - 10% float ALU instructions that run in 20 clock cycles
  - 20% load instructions that run in 10 clock cycles
  - 10% store instructions that run in 15 clock cycles
  - 10% branch instructions that run in 10 clock cycles
- The clock speed is 100 MHz
- A typical program runs 1,000,000 instructions
  - Running 500,000 ALU instructions, 100,000 FP instructions, 100,000 loads, …

The average number of cycles per instruction is

$$CPI = \sum_i CPI_i \times \frac{IC_i}{IC} = 8 \times 0.5 + 20 \times 0.1 + 10 \times 0.2 + 15 \times 0.1 + 10 \times 0.1 = 10.5$$

The typical program runs in

$$T = CPI \times IC \times \tau = \frac{CPI \times IC}{R} = \frac{10.5 \text{ cycles/instruction} \times 10^6 \text{ instructions}}{10^8 \text{ Hz}} = 0.105 \text{ seconds}$$

C Code to Runtime Example — 1

High level code

```
int x = 0 , n = 0 , a[5] ;
while ( n < 5 ){
    x = x + a[n] ;
    n++ ;
}
```

compile + optimize

Assembly program

```
1000    MOV R1, 0              load    1
1002    MOV R2, 2000           load    1      13%
1004    ADD R1, R1, (R2)+      ALU_AI  5      29%
1008    CMP R2, 2020           ALU     5      29%
1012    JL 1004                JMP     5      29%
                               IC =    17     100%
```


C Code to Runtime Example — 2

Assembly code

```
ADD R1, R1, (R2)+
```

Microprogram

```
ALU_IN, MAR ← R2
ALU ← 4
ADD
R2 ← ALU_OUT
READ
ALU_IN ← MDR
ALU ← R1
ADD
R1 ← ALU_OUT
```

interpret to microcode

$\text{CPI}_{\text{ALU-autoinc}} = 9$

C Code to Runtime Example — 3

Assembly program

```
1000    MOV R1, 0              IC_i    CPI_i
1002    MOV R2, 2000           load    13%    2
1004    ADD R1, R1, (R2)+      ALU_AI  29%    9
1008    CMP R2, 2020           ALU     29%    3
1012    JL 2004                JMP     29%    12
                       IC =    17      100%
```

Average CPI

$$CPI = \sum_i CPI_i \times \frac{IC_i}{IC} = 2 \times 0.13 + 9 \times 0.29 + 3 \times 0.29 + 12 \times 0.29 = 7.2$$

Total clock cycles

$$N = CPI \times IC = 7.2 \times 17 = 122$$

Run time with 1 GHz clock rate

$$T = N \times \tau = 122 \times 10^{-9} \text{seconds} = 1.22 \times 10^{-7} \text{seconds} = 0.122 \text{ microseconds}$$

Applying CPU Equation

1. Run time before enhancement

$$T = CPI \times IC \times \tau$$

2. Characterize enhancement

Calculate IC'

Calculate CPI'

Calculate τ'

3. Run time after enhancement

$$T' = CPI' \times IC' \times \tau'$$

4. Speedup

$$S = \frac{T}{T'} = \frac{CPI \times IC \times \tau}{CPI' \times IC' \times \tau'}$$

CPU Equation — Example 2

For a certain CPU

25% of all instructions in programs are float ($IC_{FP} / IC = 0.25$)

FP group includes ADD, SUB, MULT, DIV, SQRT

Average FP instruction runs in 4 clock cycles ($CPI_{FP} = 4$)

2% of all instructions in program are square root ($IC_{SQRT} / IC = 0.02$)

SQRT (FP) instruction runs in 20 cycles ($CPI_{SQRT} = 20$)

Average CPI for all other instructions in program is 4/3 clock cycles

($IC_{other} / IC = 1 - 0.25 = 0.75$ $\quad CPI_{other} = 4/3$)

Average cycles per instruction

$$CPI = \sum_i CPI_i \times \frac{IC_i}{IC} = 4 \times 0.25 + \frac{4}{3} \times (1 - 0.25) = 2.00$$

Example 2 — 2

Two possible enhancements

1. Improve performance of all FP instructions

Enhance average CPI_{FP} = 4 cycles to CPI_{FP}' = 2 cycles

No change in program $\Rightarrow IC_i' = IC_i$ for all instruction types

No change to clock rate $\Rightarrow \tau' = \tau$

2. Improve performance of SQRT (FP) instruction

Enhance CPI_{SQRT} = 20 cycles to CPI_{SQRT}' = 2 cycles

No change in program $\Rightarrow IC_i' = IC_i$ for all instruction types

No change to clock rate $\Rightarrow \tau' = \tau$

To evaluate enhancements, must find *CPI'*

Example 2 — 3

Enhancement 1

Improve average FP from CPI_{FP} = 4 cycles to CPI_{FP}' = 2 cycles

$$\begin{aligned} CPI' &= \sum_i CPI'_i \times \frac{IC_i'}{IC'} \\ &= CPI'_{FP} \times \frac{IC_{FP}'}{IC'} + CPI'_{\text{other}} \times \frac{IC_{\text{other}}'}{IC'} \\ &= 2 \times 0.25 + \frac{4}{3} \times (1 - 0.25) \\ &= 1.50 \end{aligned}$$

$$S = \frac{T}{T'} = \frac{CPI \times IC \times \tau}{CPI' \times IC' \times \tau'} = \frac{CPI \times IC \times \tau}{CPI' \times IC \times \tau} = \frac{CPI}{CPI'} = \frac{2.00}{1.50} \cong 1.33$$

Example 2 — 4

Enhancement 2

Improve square root (FP) from CPI_{SQRT} = 20 cycles to CPI_{SQRT}' = 2 cycles

Must separate into 3 instruction groups

FP/SQRT = FP group without SQRT = ADD, SUB, MULT, DIV

SQRT

All other instructions

$$IC_{FP} / IC = 25\%,\ \ IC_{SQRT} / IC = 2\% \ \Rightarrow\ \ IC_{FP/SQRT} / IC = 25\% - 2\% = 23\%$$

$$CPI' = \sum_i CPI'_i \times \frac{IC_i'}{IC'}$$

$$= CPI'_{FP/SQRT} \times \frac{IC_{FP/SQRT}'}{IC'} + CPI'_{SQRT} \times \frac{IC_{SQRT}'}{IC'} + CPI'_{\text{other}} \times \frac{IC_{\text{other}}'}{IC'}$$

$$= CPI_{FP/SQRT} \times \frac{IC_{FP/SQRT}}{IC} + CPI'_{SQRT} \times \frac{IC_{SQRT}}{IC} + CPI_{\text{other}} \times \frac{IC_{\text{other}}}{IC}$$

First calculate $CPI_{FP/SQRT}$ from CPI_{FP}, CPI_{SQRT}, IC_{FP} / IC, IC_{SQRT} / IC

Example 2 — 5

$$CPI_{FP} = \text{Average } CPI \text{ for } FP \text{ group with } SQRT = 4$$

$$= \frac{\text{total } FP \text{ cycles}}{FP \text{ instructions}} = \frac{N_{FP}}{IC_{FP}} = \sum_{k \in FP} \frac{N_k}{IC_{FP}}$$

$$= \sum_{k \in FP} \frac{CPI_k \times IC_k}{IC_{FP}} = \sum_{k \in FP} CPI_k \times \frac{IC_k}{IC_{FP}}$$

$$CPI_{FP/SQRT} = \text{Average } CPI \text{ for } FP \text{ group without } SQRT$$

$$= \frac{\text{total } FP / SQRT \text{ cycles}}{FP / SQRT \text{ instructions}} = \frac{N_{FP/SQRT}}{IC_{FP/SQRT}} = \sum_{k \in FP/SQRT} \frac{N_k}{IC_{FP/SQRT}}$$

$$= \sum_{k \in FP/SQRT} \frac{CPI_k \times IC_k}{IC_{FP/SQRT}} = \sum_{k \in FP/SQRT} CPI_k \times \frac{IC_k}{IC_{FP/SQRT}}$$

Example 2 — 6

$$CPI_{FP} = \sum_{k \in FP} CPI_k \times \frac{IC_k}{IC_{FP}}$$

$$= \sum_{k \in FP/SQRT} CPI_k \times \frac{IC_k}{IC_{FP}} + CPI_{SQRT} \times \frac{IC_{SQRT}}{IC_{FP}}$$

$$= \left[\sum_{k \in FP/SQRT} CPI_k \times \frac{IC_k}{IC_{FP}} \right] \times \frac{IC_{FP/SQRT} / IC}{IC_{FP/SQRT} / IC} + \left[CPI_{SQRT} \times \frac{IC_{SQRT}}{IC_{FP}} \right] \times \frac{1/IC}{1/IC}$$

$$= \left[\sum_{k \in FP/SQRT} CPI_k \times \frac{IC_k}{IC_{FP/SQRT}} \right] \frac{IC_{FP/SQRT} / IC}{IC_{FP} / IC} + CPI_{SQRT} \times \frac{IC_{SQRT} / IC}{IC_{FP} / IC}$$

$$= CPI_{FP/SQRT} \times \frac{IC_{FP/SQRT} / IC}{IC_{FP} / IC} + CPI_{SQRT} \times \frac{IC_{SQRT} / IC}{IC_{FP} / IC}$$

$$4 = CPI_{FP/SQRT} \times \frac{0.25 - 0.02}{0.25} + 20 \times \frac{0.02}{0.25} \ \Rightarrow\ \boxed{CPI_{FP/SQRT} = 2.61}$$

Example 2 — 7

Speedup for Enhancement 2

$$CPI' = \sum_i CPI'_i \times \frac{IC_i'}{IC'}$$

$$= CPI_{FP/SQRT} \times \frac{IC_{FP/SQRT}}{IC} + CPI'_{SQRT} \times \frac{IC_{SQRT}}{IC} + CPI_{\text{other}} \times \frac{IC_{\text{other}}}{IC}$$

$$= 2.61 \times 0.23 + 2 \times 0.02 + \frac{4}{3} \times (1 - 0.25)$$

$$= 1.64$$

$$S = \frac{T}{T'} = \frac{CPI \times IC \times \tau}{CPI' \times IC' \times \tau'} = \frac{CPI \times IC \times \tau}{CPI' \times IC \times \tau} = \frac{CPI}{CPI'} = \frac{2.00}{1.64} \cong 1.22$$

Example 2 — 8

Trick — technique to avoid calculating $CPI_{FP/\text{SQRT}}$

$$
\begin{aligned}
CPI' &= CPI - (CPI - CPI') \\
&= CPI - \left(\sum_i CPI_i \times \frac{IC_i}{IC} - \sum_i CPI'_i \times \frac{IC'_i}{IC'} \right) \\
&= CPI - \sum_i \left(CPI_i \times \frac{IC_i}{IC} - CPI'_i \times \frac{IC'_i}{IC'} \right)
\end{aligned}
$$

If $IC'_i = IC_i$ then combine terms as

$$
\begin{aligned}
CPI' &= CPI - \sum_i \left[(CPI_i - CPI'_i) \times \frac{IC_i}{IC} \right] \\
&= CPI - \left[(CPI_{SQRT} - CPI'_{SQRT}) \times \frac{IC_{SQRT}}{IC} \right] \\
&= 2.00 - \left[(20 - 2) \times 0.02 \right] \\
&= 1.64
\end{aligned}
$$

Example 2 — 9

5. Speedups

Enhancement to square root — S = 1.22

Enhancement to all FP — S = 1.33

Results identical to analysis by Amdahl equation

Can derive inputs to Amdahl equation from CPU analysis

1.
$$
\begin{aligned}
F_e^{FP} &= \frac{t_{FP}}{T} = \frac{CPI_{FP} \times IC_{FP} \times \tau}{CPI \times IC \times \tau} = \frac{4 \times (0.25 \times IC) \times \tau}{2 \times IC \times \tau} = 50\% \\
S_e^{FP} &= \frac{CPI_{FP} \times IC_{FP} \times \tau}{CPI'_{FP} \times IC'_{FP} \times \tau'} = \frac{CPI_{FP}}{CPI'_{FP}} = \frac{4}{2} = 2
\end{aligned}
$$

2.
$$
\begin{aligned}
F_e^{SQRT} &= \frac{t_{SQRT}}{T} = \frac{CPI_{SQRT} \times IC_{SQRT} \times \tau}{CPI \times IC \times \tau} = \frac{20 \times (0.02 \times IC) \times \tau}{2 \times IC \times \tau} = 20\% \\
S_e^{SQRT} &= \frac{CPI_{SQRT} \times IC_{SQRT} \times \tau}{CPI'_{SQRT} \times IC'_{SQRT} \times \tau'} = \frac{CPI_{SQRT}}{CPI'_{SQRT}} = \frac{20}{2} = 10
\end{aligned}
$$

Changing Instruction Mix — 1

Program distribution

Instruction Type i	IC_i	IC_i / IC	CPI_i
Integer	5,000	5000/10000 = 50%	4
Branch	1,000	1000/10000 = 10%	12
Load / Store	4,000	4000/10000 = 40%	8
Total	10,000	10000/10000 = 100%	

$$CPI = \sum_i CPI_i \times \frac{IC_i}{IC} = 4 \times 0.50 + 12 \times 0.10 + 8 \times 0.40 = 6.4$$ cycles per instruction

New program distribution

Instruction Type i	IC_i	IC_i / IC	CPI_i
Integer	3,000	3000/8000 = 37.5%	4
Branch	1,000	1000/8000 = 12.5%	12
Load / Store	4,000	4000/8000 = 50%	8
Total	8,000	8000/8000 = 100%	

$$CPI = \sum_i CPI_i \times \frac{IC_i}{IC} = 4 \times 0.375 + 12 \times 0.125 + 8 \times 0.50 = 7.0$$ cycles per instruction

Changing Instruction Mix — 2

Speedup

$$
\begin{aligned}
S &= \frac{T}{T'} \\
&= \frac{CPI \times IC \times \tau}{CPI' \times IC' \times \tau'} \\
&= \frac{6.4 \times 10000 \times \tau}{7.0 \times 8000 \times \tau} \\
&= 1.14
\end{aligned}
$$

The Instructions Per Second Myth

Measures often used to describe computer power

MIPS = million instructions per second

FLOPS = floating point operations per second

Neither gives fair comparison

$$\text{MIPS} = \frac{\text{instructions} / 10^6}{\text{run time}} = \frac{IC}{10^6 \times T} = \frac{IC}{10^6 \times CPI \times IC \times \tau} = \frac{R}{10^6 \times CPI}$$

Example

CPU-1 and CPU-2

Run ALU instructions in 1 cycles and others in 2 cycles

Have clock speed of 1 GHz

CPU-1 compiler produces 50% ALU instructions and 50% other

CPU-2 compiler produces 25% fewer ALU instructions than CPU-1

$$CPI_1 = 1 \times 0.50 + 2 \times 0.50 = 1.50 \Rightarrow MIPS_1 = \frac{10^9\, Hz}{10^6 \times 1.50} \cong 667 \text{ million instructions / sec}$$

MIPS — 2

For CPU-2

$$IC_1^{ALU} = 0.50 \times IC_1$$

$$IC_2^{ALU} = 0.75 \times IC_1^{ALU} = 0.75 \times (0.50 \times IC_1) = 0.375 \times IC_1$$

$$IC_2 = IC_2^{ALU} + IC_2^{other} = 0.375 \times IC_1 + 0.50 \times IC_1 = 0.875 \times IC_1$$

$$\frac{IC_2^{ALU}}{IC_2} = \frac{0.375 \times IC_1}{0.875 \times IC_1} = 0.43 \qquad \frac{IC_2^{other}}{IC_2} = \frac{0.50 \times IC_1}{0.875 \times IC_1} = 0.57$$

$$CPI_2 = 1 \times 0.43 + 2 \times 0.57 = 1.57$$

$$MIPS_2 = \frac{10^9\, Hz}{10^6 \times 1.57} \cong 637 \text{ million instructions / sec} < MIPS_1$$

$$\boxed{\frac{MIPS_2}{MIPS_1} = \frac{637}{667} = 0.955}$$

MIPS — 3

Run time comparison

$$S = \frac{T_1}{T_2} = \frac{CPI_1 \times IC_1 \times \tau_1}{CPI_2 \times IC_2 \times \tau_2} = \frac{CPI_1 \times IC_1}{CPI_2 \times IC_2} = \frac{1.50 \times IC_1}{1.57 \times (0.875 \times IC_1)} \cong 1.09$$

MIPS is about 5% lower for CPU-2 than CPU-1

CPU-2 is about 9% faster than CPU-1

Replacing Instruction Types

Instruction count

IC = IC_1 + IC_2 + ... + IC_n

Examples

Type 1 = ALU

Type 2 = Conditional Branch

New Instruction count

Replace 2 ALU instructions + 1 Branch

```
DEC CX
CMP CX, 0
JNZ target
```

New instruction

```
LOOP target
```

IC' = IC_1' + IC_2' + ... + IC_n

Example — Replacing Instructions

A certain CPU has no floating point unit (FPU)

Performs FP calculations by **EMULATION**

Converts FP operations to integer operations

Example

$(2.165 \times 10^4) \times (3.247 \times 10^{-3}) \rightarrow 2165 \times 3247$

exp = (4 – 3) + (-3 – 3)

Instruction distribution

Type i	IC_i / IC	CPI_i
ALU	75%	1
Load	10%	2
Store	5%	2
Branch	10%	2

$$CPI = \sum_i CPI_i \times \frac{IC_i}{IC}$$
$$= 1 \times 0.75 + 2 \times (0.10 + 0.05 + 0.10)$$
$$= 1.25$$

Replacing Instructions — 2

Enhance CPU with ALU

Replace ALU instructions that emulate FP with new FP instructions

2/3 of old ALU instructions emulate FP instructions

1 new FPU instruction replaces 10 old ALU emulation instructions

New FPU instructions run in 4 clock cycles

Type i	IC_i / IC	CPI_i
ALU	75%	1
Load	10%	2
Store	5%	2
Branch	10%	2

} →

ALU^{int}	1/3 × 75% = 25%
$ALU^{emulation}$	2/3 × 75% = 50%

$IC'_{ALU} = 0.25\ IC$

$IC'_{FPU} = 1/10 \times 0.50\ IC = 0.05\ IC$

$IC'_{load} = IC_{load} = 0.10\ IC$

$IC'_{store} = IC_{store} = 0.05\ IC$

$IC'_{branch} = IC_{branch} = 0.10\ IC$

$$IC' = 0.25\ IC + 0.05\ IC + 0.10\ IC + 0.05\ IC + 0.10\ IC = 0.55\ IC$$

Replacing Instructions — 3

New instruction distribution

Type i	IC_i / IC	CPI_i
ALU	0.25/0.55	1
FPU	0.05 /0.55	4
Load	0.10 /0.55	2
Store	0.05 /0.55	2
Branch	0.10 /0.55	2

$$CPI' = \sum_i CPI'_i \times \frac{IC'_i}{IC}$$
$$= 1 \times \frac{0.25}{0.55} + 4 \times \frac{0.05}{0.55} + 2 \times \left(\frac{0.10}{0.55} + \frac{0.05}{0.55} + \frac{0.10}{0.55} \right)$$
$$= \frac{0.95}{0.55}$$

$$CPI' = 1.73 \quad > \quad CPI = 1.25$$

$$S = \frac{T}{T'} = \frac{CPI \times IC \times \tau}{CPI' \times IC' \times \tau'} = \frac{1.25 \times IC}{\frac{0.95}{0.55} \times (0.55 IC)} = \frac{1.25}{0.95} \cong 1.32$$

Load-Store versus Register-Memory

CPU-1 is a load-store machine

ALU operands must come from register

Memory operand

Loaded to register before ALU operation

Stored to memory after ALU operation

Instruction distribution

$$CPI = \sum_i CPI_i \times \frac{IC_i}{IC}$$
$$= 5 \times 0.25 + 4 \times 0.75$$
$$= 4.25$$

Type i	IC_i / IC	CPI_i
ALU	40%	4
Load	25%	5
Store	15%	4
Branch	20%	4

Possible enhancement

25% of ALU memory operands used in only 1 ALU operation

Can register-memory ALU operations improve performance?

Load-Store versus Register-Memory

CPU-2 is an "ideal" register-memory machine

- ALU operands may come from register or memory
- 75% of memory operands
  - Used in multiple ALU operations
  - Perfect compiler loads "multiple" memory operands to registers
- 25% of ALU memory operands
  - Used in only a single ALU operation
  - Perfect compiler never loads "single" memory operands to registers

Convert CPU-1 to CPU-2

- Split ALU operations into ALU^{multi} and ALU^{single}
- Replace ALU^{single} with $ALU^{register\text{-}memory}$
- Cancel 1 register load for every ALU^{single}

Detailed Instruction Distribution

Type i	IC_i / IC	CPI_i
ALU	40%	4
Load	25%	5
Store	15%	4
Branch	20%	4

ALU^{single}	10%
ALU^{multi}	30%

$Load^{single}$	10%
$Load^{multi}$	25% - 10% =15%

Type i	IC_i / IC	CPI_i
ALU^{single}	10%	4
ALU^{multi}	30%	4
$Load^{single}$	10%	5
$Load^{multi}$	15%	5
Store	15%	4
Branch	20%	4

$ALU^{register\text{-}memory}$	10%	9

$$CPI_{ALU^{reg-mem}} = CPI_{ALU} + CPI_{Load}$$

New Instruction Distribution and Speedup

Type i	IC_i / IC	CPI_i
ALU^{single}	10%	4
ALU^{multi}	30%	4
$Load^{single}$	10%	5
$Load^{multi}$	15%	5
Store	15%	4
Branch	20%	4

$ALU^{register\text{-}memory}$	10%	9

$$\begin{aligned} IC' &= IC'_{ALU^{reg-mem}} + IC_{ALU^{multi}} + IC_{Load^{multi}} + IC_{Store} + IC_{Branch} \\ &= 0.10IC + 0.30IC + 0.15IC + 0.15IC + 0.20IC \\ &= 0.90IC \end{aligned}$$

$$\begin{aligned} CPI' &= \sum_i CPI'_i \times \frac{IC'_i}{IC} \\ &= 4 \times \frac{65}{90} + 5 \times \frac{15}{90} + 9 \times \frac{10}{90} \\ &= \frac{425}{90} = 4.72 \end{aligned}$$

Type i	IC_i / IC	CPI_i
$ALU^{register\text{-}memory}$	10/90	9
ALU^{multi}	30/90	4
$Load^{multi}$	15/90	5
Store	15/90	4
Branch	20/90	4

$$S = \frac{T}{T'} = \frac{CPI \times IC \times \tau}{CPI' \times IC' \times \tau'} = \frac{4.25 \times IC}{\frac{425}{90} \times (0.90IC)} = 1 \Rightarrow$$ No Change in Performance

Analysis of 8086 Example

8086 program compiled from C source

	Instruction	Clock Cycles	Runs	Type
	MOV WORD PTR [BP-02],0000	13	1	Store
start:	CMP WORD PTR [BP-02],N	10	N	$ALU^{imm\text{-}mem}$
	JGE stop	4/13	N-1 / 1	Conditional Branch
	MOV AX,[BP-02]	9	N-1	Load
	SHL AX,1	2	N-1	ALU^{reg}
	MOV [BP-04],AX	12	N-1	Store
	INC WORD PTR [BP-02]	15	N-1	$ALU^{reg\text{-}mem}$
	JMP start	JMP 14	N-1	Unconditional Branch
stop:	RET	RET 16	1	Return

CPI for Store and ALU

JGE

Runs N-1 times in 4 clock cycles and 1 time in 13 clock cycles

$$CPI_{JGE} = \frac{JGE \text{ cycles}}{JGE \text{ instructions}} = \frac{4\times(N-1)+13\times 1}{(N-1)+1} = \frac{4\times(N-1)+13}{N}$$

STORE

Runs N-1 times in 12 clock cycles and 1 time in 13 clock cycles

$$CPI_{STORE} = \frac{STORE \text{ cycles}}{STORE \text{ instructions}} = \frac{12\times(N-1)+13\times 1}{(N-1)+1} = \frac{12\times(N-1)+13}{N}$$

Instruction Distribution

Type i	IC_i / IC	CPI_i
Load	$\frac{N-1}{7N-3}$	9
Store	$\frac{1+(N-1)}{7N-3}$	$[12(N–1) +13] / N$
ALU$^{\text{reg-mem}}$	$\frac{N-1}{7N-3}$	15
ALU$^{\text{reg}}$	$\frac{N-1}{7N-3}$	2
ALU$^{\text{imm-mem}}$	$\frac{N}{7N-3}$	10
Conditional Branch	$\frac{N}{7N-3}$	$[4(N–1) + 13] / N$
Unconditional Branch	$\frac{N-1}{7N-3}$	14
Return	$\frac{1}{7N-3}$	16

Instruction Distribution for Loop Length = 100

Type i	IC_i / IC	CPI_i
Load	14.20%	9
Store	14.34%	12.01
ALU$^{\text{reg-mem}}$	14.20%	15
ALU$^{\text{reg}}$	14.20%	2
ALU$^{\text{imm-mem}}$	14.34%	10
Conditional Branch	14.34%	4.09
Unconditional Branch	14.20%	14
Return	0.14%	16

$$N = 100$$

$$IC = 7N - 3 = 697$$

$$CPI = \sum_i CPI_i \times \frac{IC_i}{IC}$$

$$= (9+15+2+14)\times 0.1420 + (12.01+10+4.09)\times 0.1434 + 16\times 0.0014 = 9.45$$

$$T = \frac{CPI \times IC}{R} = \frac{(9.45 \text{ cycles / instruction})\times(697 \text{ instructions})}{4 \text{ MHz}} = 1.646\times 10^{-3} \text{ sec}$$

Estimated run time for
296 MHz UltraSPARC II = 4.71×10^{-7} sec

$$S = \frac{1.646\times 10^{-3}}{4.71\times 10^{-7}} = 3494$$

Register Variables

Improved program

Memory variables replaced with register variables

	Instruction	Clock Cycles	Runs	Type
	MOV SI,0000	4	1	MOV$^{\text{imm-reg}}$
start:	CMP SI,+0A	3	N	ALU$^{\text{imm-reg}}$
	JGE stop	4/13	N-1 / 1	Conditional Branch
	MOV AX,SI	2	N-1	MOV$^{\text{reg-reg}}$
	SHL AX,1	2	N-1	ALU$^{\text{reg}}$
	MOV DI,AX	2	N-1	MOV$^{\text{reg-reg}}$
	INC SI	3	N-1	ALU$^{\text{reg-reg}}$
	JMP start	JMP 14	N-1	Unconditional Branch
stop:	RET	RET 16	1	Return

New Instruction Distribution

Type i	IC_i / IC	CPI_i
MOV$^{\text{imm-reg}}$	$\frac{1}{7N-3}$	4
MOV$^{\text{reg-reg}}$	$\frac{2(N-1)}{7N-3}$	2
ALU$^{\text{reg-reg}}$	$\frac{N-1}{7N-3}$	3
ALU$^{\text{reg}}$	$\frac{N-1}{7N-3}$	2
ALU$^{\text{imm-reg}}$	$\frac{N}{7N-3}$	3
Conditional Branch	$\frac{N}{7N-3}$	$[4(N–1) + 13] / N$
Unconditional Branch	$\frac{N-1}{7N-3}$	14
Return	$\frac{1}{7N-3}$	16

New Distribution for Loop Length = 100

Type i	IC_i' / IC'	CPI_i'
MOV$^{\text{imm-reg}}$	0.14%	4
MOV$^{\text{reg-reg}}$	28.41%	2
ALU$^{\text{reg-reg}}$	14.20%	3
ALU$^{\text{reg}}$	14.20%	2
ALU$^{\text{imm-reg}}$	14.35%	3
Conditional Branch	14.35%	4.09
Unconditional Branch	14.20%	14
Return	0.14%	16

$$N = 100$$

$$IC = 7N - 3 = 697$$

$$CPI' = \sum_i CPI_i' \times \frac{IC_i'}{IC'}$$

$$= (4+16)\times 0.0014 + 2\times 0.2840 + (3+2+14)\times 0.1420 + (3+4.09)\times 0.1435 = 4.31$$

$$T = \frac{CPI' \times IC'}{R} = \frac{(4.31 \text{ cycles / instruction}) \times (697 \text{ instructions})}{4 \text{ MHz}} = 7.515\times 10^{-4} \text{ sec}$$

Run time with memory variables = 1.646×10^{-3} sec

$$S = \frac{1.646\times 10^{-3}}{7.515\times 10^{-4}} = 2.19$$

Register Variables — Two Threads

Data decomposition

Take N = 100 data points

Each thread runs on 50 data points (loop length)

	Thread 0	Thread 1
	`MOV SI,0000`	`MOV SI,0032`
`start:`	`CMP SI,+32 ; 0x32 = 50₁₀`	`CMP SI,+64 ; 0x64 = 100₁₀`
	`JGE stop`	`JGE stop`
	`MOV AX,SI`	`MOV AX,SI`
	`SHL AX,1`	`SHL AX,1`
	`MOV DI,AX`	`MOV DI,AX`
	`INC SI`	`INC SI`
	`JMP start`	`JMP start`
`stop:`	`RET`	`RET`

New Distribution for Loop Length = 50 For Each Thread

Type i	IC_i' / IC'	CPI_i'
MOV$^{\text{imm-reg}}$	0.28%	4
MOV$^{\text{reg-reg}}$	28.41%	2
ALU$^{\text{reg-reg}}$	14.12%	3
ALU$^{\text{reg}}$	14.12%	2
ALU$^{\text{imm-reg}}$	14.41%	3
Conditional Branch	14.41%	4.09
Unconditional Branch	14.12%	14
Return	0.28%	16

$$N = 50$$

$$IC = 7N - 3 = 347$$

$$CPI' = \sum_i CPI_i' \times \frac{IC_i'}{IC'}$$

$$= (4+16)\times 0.0028 + 2\times 0.2841 + (3+2+14)\times 0.1412 + (3+4.09)\times 0.1441 = 4.34$$

$$T = \frac{CPI' \times IC'}{R} = \frac{(4.34 \text{ cycles / instruction}) \times (347 \text{ instructions})}{4 \text{ MHz}} = 3.765\times 10^{-4} \text{ sec}$$

Run time on one thread = 7.525×10^{-4} sec

$$S = \frac{3.765\times 10^{-4}}{7.515\times 10^{-4}} = 1.998$$

Software Content — Instruction Distribution

Type i	CPI_i	IC_i
ALU	2	400
Branch	8	200
Load	4	250
Store	4	150

CPI	Cycles Per Instruction
IC	Instruction Count

Software Content — Instruction Distribution

Type i	CPI_i	IC_i
ALU	2	400
Branch	8	200
Load	4	250
Store	4	150
IC		1000

$$IC = \sum_i IC_i$$

Software Content — Instruction Distribution

Type i	CPI_i	IC_i	IC_i/IC
ALU	2	400	40%
Branch	8	200	20%
Load	4	250	25%
Store	4	150	15%
IC		1000	

Software Content — Instruction Distribution

Type i	CPI_i	IC_i	IC_i/IC	$CPI_i * IC_i/IC$
ALU	2	400	40%	0.8
Branch	8	200	20%	1.6
Load	4	250	25%	1.0
Store	4	150	15%	0.6
IC		1000		

Software Content — Instruction Distribution

Type i	CPI_i	IC_i	IC_i/IC	$CPI_i * IC_i/IC$
ALU	2	400	40%	0.8
Branch	8	200	20%	1.6
Load	4	250	25%	1.0
Store	4	150	15%	0.6
IC		1000		
CPI				4.0

$$CPI = \sum_i CPI_i \times \frac{IC_i}{IC}$$

Software Content — Instruction Distribution

Type i	CPI_i	IC_i	IC_i/IC	$CPI_i * IC_i/IC$
ALU	2	400	40%	0.8
Branch	8	200	20%	1.6
Load	4	250	25%	1.0
Store	4	150	15%	0.6
IC		1000		
CPI				4.0

$$N = CPI \times IC = 4.0 \times 1000 = 4000$$

$$T = CPI \times IC \times \tau = 4000\tau$$

Software Content — Instruction Distribution

Type i	CPI_i	IC_i	IC_i/IC	$CPI_i * IC_i/IC$	N_i
ALU	2	400	40%	0.8	800
Branch	8	200	20%	1.6	1600
Load	4	250	25%	1.0	1000
Store	4	150	15%	0.6	600
IC		1000			
CPI				4.0	

$$N_i = CPI_i \times IC_i$$

Software Content — Instruction Distribution

Type i	CPI_i	IC_i	IC_i/IC	$CPI_i * IC_i/IC$	N_i
ALU	2	400	40%	0.8	800
Branch	8	200	20%	1.6	1600
Load	4	250	25%	1.0	1000
Store	4	150	15%	0.6	600
IC		1000			
CPI				4.0	

$$N = CPI \times IC = 4.0 \times 1000 = 4000$$

$$N = \sum_i N_i = 800 + 1600 + 100 + 600 = 4000$$

Software Content — Instruction Distribution

Type i	CPI_i	IC_i	IC_i/IC	$CPI_i * IC_i/IC$	N_i	F_i
ALU	2	400	40%	0.8	800	20%
Branch	8	200	20%	1.6	1600	40%
Load	4	250	25%	1.0	1000	25%
Store	4	150	15%	0.6	600	15%
IC		1000				
CPI				4.0		

$$F_i = \frac{t_i}{T} = \frac{CPI_i \times IC_i \times \tau}{CPI \times IC \times \tau} = \frac{CPI_i \times IC_i}{CPI \times IC} = \frac{N_i}{N}$$

Enhancement

Type i	CPI_i'	IC_i	IC_i/IC	$CPI_i' * IC_i/IC$	N_i'	F_i'
ALU	2	400	40%	0.8	800	25%
Branch	8 → 4	200	20%	1.6 → 0.8	800	25%
Load	4	250	25%	1.0	1000	31%
Store	4	150	15%	0.6	600	19%
IC		1000				
CPI'				3.2		

CPU Equation $$S = \frac{T}{T'} = \frac{CPI \times IC \times \tau}{CPI' \times IC' \times \tau'} = \frac{CPI}{CPI'} = \frac{4.0}{3.2} = 1.25$$

Amdahl Equation $$S = \frac{T}{T'} = \frac{1}{1 - F_e + \frac{F_e}{S_e}} = \frac{1}{1 - 0.4 + \frac{0.4}{2}} = \frac{1}{0.6 + 0.2} = 1.25$$

$$S_e = \frac{t_e}{t_e'} = \frac{8 \times IC \times \tau}{4 \times IC \times \tau} = 2$$

Instruction Distributions

CPU analysis

Permits performance analysis of machine design "on drawing board"

Evaluate proposed design without building CPU implementation

Summary of procedure

Specify Instruction Set Architecture (ISA)

- Describes machine language for proposed CPU
- Provides human-readable assembly language
- Determines CPI_i for each instruction group i
  - Count clock cycles to implement a single instruction in ISA

Write C, C++, Fortran compilers for proposed machine language

Compile representative programs to machine language

- Can use programs from SPEC CINT and CFP

Sort instructions into groups to find relative instruction count IC_i/IC

Calculate average CPI and run time T

Compare run time with reference machine