CSL7070: Computer Architecture

Lecture 9, 23rd February 2022

Dip Sankar Banerjee

Indian Institute of Technology, Jodhpur

January-April 2022

CPU Performance Evaluation (CPI) – tradeoffs

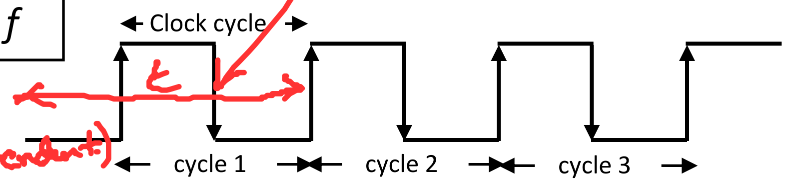
DVFS

- Most computers run synchronously utilizing a CPU clock running at a constant clock rate:

  Or clock frequency: f

  where: Clock rate = 1 / clock cycle

  $f = 1/C$ (system independent)



- The CPU clock rate depends on the specific CPU organization (design) and hardware implementation technology (VLSI) used.
- A computer machine (ISA) instruction is comprised of a number of elementary or micro operations which vary in number and complexity depending on the the instruction and the exact CPU organization (Design).
  - A micro operation is an elementary hardware operation that can be performed during one CPU clock cycle.
  - This corresponds to one micro-instruction in microprogrammed CPUs.
  - Examples: register operations: shift, load, clear, increment, ALU operations: add , subtract, etc.

CPI → time
IPC → throughput

- Thus: A single machine instruction may take one or more CPU cycles to complete termed as the Cycles Per Instruction (CPI). // the most imp.

  Instructions Per Cycle = IPC = 1/CPI

- Average (or effective) CPI of a program: The average CPI of all instructions executed in the program on a given CPU design.

  Cycles/sec = Hertz = Hz
  MHz = 10^6 Hz GHz = 10^9 Hz

CPI (≈ time synonyms)

6 MHz → $\frac{1}{6\times10^6} = \frac{1}{6}\times10^{-6}$ s.

Generic CPU Machine Instruction Processing Steps

Instruction Fetch 1 4 — Obtain instruction from program memory

The Program Counter (PC) points to the instruction to be processed

What is the CPI of a program?

Instruction Decode 2 — Determine required actions and instruction size → 10

R-types → 4
J-types → ?
I-types → 8

Operand Fetch 4 6 — Locate and obtain operand data

From data memory or registers

6/14 → 4/14

Execute 1 20 — Compute result value or status

Result Store 2 — Deposit results in storage (data memory or register) for later use

CPI is an average!

Next Instruction 1 — Determine successor or next instruction

(i.e Update PC to fetch next instruction to be processed)

In 1 pc → instr.

CPI = Cycles per instruction

14 4/14 → Efficiency

→ single cycle = 1 cycle

Computer Performance Measures: Program Execution Time

- For <u>a specific program</u> compiled to run on <u>a specific machine (CPU)</u> "A", has the following parameters:
  - The total executed instruction count of the program. $\boxed{I}$
  - The average number of cycles per instruction (average CPI). $\boxed{CPI}$
    Or effective CPI
  - Clock cycle of machine "A" $\boxed{C/CC}$
- How can one measure the performance of this machine (CPU) running this program?
  - Intuitively the machine (or CPU) is said to be faster or has better performance running this program if the total execution time is shorter.
  - Thus the inverse of the total measured program execution time is a possible performance measure or metric:

Programs/second

$$\boxed{\text{Performance}_A = 1 / \text{Execution Time}_A}$$

Seconds/program

How to compare performance of different machines?

What factors affect performance? How to improve performance?

Comparing Computer Performance Using Execution Time

- To compare the performance of two machines (or CPUs) "A", "B" running <u>a given specific program</u>:

  $\text{Performance}_A = 1 \;/\; \text{Execution Time}_A$

  $\text{Performance}_B = 1 \;/\; \text{Execution Time}_B$

> The two CPUs may target different ISAs provided the program is written in a high level language (HLL)

- Machine A is n times faster than machine B means (or slower? if n < 1) :

$$\boxed{\text{Speedup} = n = \frac{\text{Performance}_A}{\text{Performance}_B} = \frac{\text{Execution Time}_B}{\text{Execution Time}_A}}$$

- Example: (i.e Speedup is ratio of performance, no units)

  For a given program:

  Execution time on machine A: $\text{Execution}_A = 1$ second

  Execution time on machine B: $\text{Execution}_B = 10$ seconds

Speedup= $\text{Performance}_A \;/\; \text{Performance}_B = \text{Execution Time}_B \;/\; \text{Execution Time}_A$

$= 10 / 1 = 10$

The performance of machine A is 10 times the performance of machine B when running this program, or: Machine A is said to be 10 times faster than machine B when running this program.

Proc. A is 1.2(x) faster than B

A's Proc. 20% faster

CPU Execution Time: The CPU Equation

- A program is comprised of <u>a number of instructions executed , I</u>
  - Measured in: instructions/program

AKA Dynamic Executed Instruction Count

- The average instruction executed takes a number of *cycles per instruction (CPI)* to be completed.
  - Measured in: cycles/instruction, CPI

Or Instructions Per Cycle (IPC):
IPC = 1/CPI

- CPU has a fixed clock cycle time <u>C = 1/clock rate</u>
  - Measured in: seconds/cycle

$C = 1/f$

- CPU execution time is the product of the above three parameters as follows:

Avg.

Executed

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

$$T = I \times CPI \times C$$

$\frac{\text{Total no. of cycles}}{\text{no. of Instr}}$

execution Time per program in seconds

Number of instructions executed

Average CPI for program

CPU Clock Cycle

(This equation is commonly known as the <u>CPU performance equation</u>)

CPU Average CPI/Execution Time

For a given program executed on a given machine (CPU):

CPI = Total program execution cycles / Instructions count Executed (I)

(i.e average or effective CPI)

→ CPU clock cycles = Instruction count x CPI

(executed, I)

CPU execution time =

= CPU clock cycles x Clock cycle

= Instruction count x CPI x Clock cycle

T = I x CPI x C

execution Time per program in seconds

Number of instructions executed

Average or effective CPI for program

CPU Clock Cycle

(This equation is commonly known as the CPU performance equation)

CPI = Cycles Per Instruction

CPU Execution Time: Example

- A Program is running on a specific machine (CPU) with the following parameters: I
  - Total executed instruction count: 10,000,000 instructions
  - Average CPI for the program: 2.5 cycles/instruction.
  - CPU clock rate: 200 MHz. (clock cycle = C = 5×10^{-9} seconds) i.e 5 nanoseconds

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

- What is the execution time for this program:

CPU time = Instruction count x CPI x Clock cycle

= 10,000,000 x 2.5 x 1 / clock rate

= 10,000,000 x 2.5 x 5×10^{-9}

= 0.125 seconds

Nanosecond = nsec = ns = 10^{-9} second
MHz = 10^{6} Hz

$T = I \times CPI \times C$

Factors Affecting CPU Performance

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

$$T = I \times \underset{}{\overset{\text{Average}}{CPI}} \times C$$

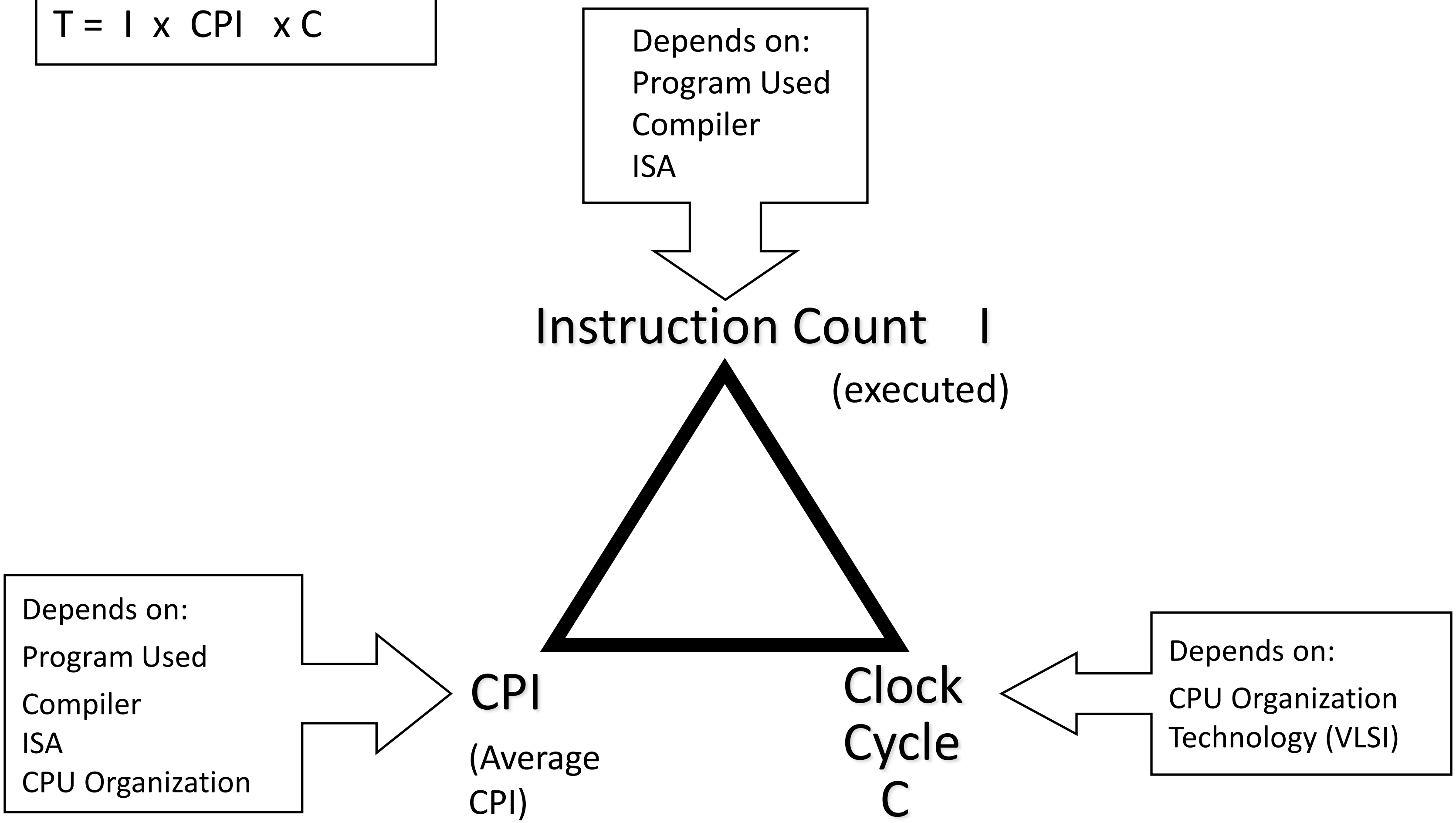
	Instruction Count	Cycles per Instruction	Clock Rate (1/C)
Program	✓	✓	✗
Compiler	✓	✗	✗
Instruction Set Architecture (ISA)	✓	✓	✗
Organization (CPU Design)	✗	✓	✗
Technology (VLSI)	✗	~~✓~~ ✗	✓

$$\boxed{T = I \times CPI \times C}$$

Aspects of CPU Execution Time

CPU Time = Instruction count executed x CPI x Clock cycle

T = I x CPI x C



Performance Comparison: Example

- From the previous example: A Program is running on a specific machine (CPU) with the following parameters:
  - Total executed instruction count, I: 10,000,000 instructions
  - Average CPI for the program: 2.5 cycles/instruction.
  - CPU clock rate: 200 MHz.

Thus: $C = 1/(200\text{x}10^6) = 5\text{x}10^{-9}$ seconds

- Using the same program with these changes:
  - A new compiler used: New executed instruction count, I: 9,500,000

    New CPI: 3.0
  - Faster CPU implementation: New clock rate = 300 MHz

Thus: $C = 1/(300\text{x}10^6) = 3.33\text{x}10^{-9}$ seconds

- What is the speedup with the changes?

$$\text{Speedup} = \frac{\text{Old Execution Time}}{\text{New Execution Time}} = \frac{I_{old} \text{ x } CPI_{old} \text{ x Clock cycle}_{old}}{I_{new} \text{ x } CPI_{new} \text{ x Clock Cycle}_{new}}$$

Speedup = $(10{,}000{,}000 \text{ x } 2.5 \text{ x } 5\text{x}10^{-9}) / (9{,}500{,}000 \text{ x } 3 \text{ x } 3.33\text{x}10^{-9})$

= .125 / .095 = 1.32

or 32 % faster after changes.

Clock Cycle = C = 1/ Clock Rate

T = I x CPI x C

Instruction Types & CPI

- Given a program with n types or classes of instructions executed on a given CPU with the following characteristics:

e.g ALU, Branch etc.

C_i = Count of instructions of type$_i$ executed

i = 1, 2, ... n

CPI_i = Cycles per instruction for type$_i$

Depends on CPU Design

Then:

CPI = CPU Clock Cycles / Instruction Count I

i.e average or effective CPI

Where:

Executed

$$CPU\ clock\ cycles = \sum_{i=1}^{n}\left(CPI_i \times C_i\right)$$

Executed Instruction Count I = $\Sigma\ C_i$

T = I x CPI x C

Instruction Types & CPI: An Example

- An instruction set has three instruction classes:

e.g ALU, Branch etc.

Instruction class	CPI
A	1
B	2
C	3

For a specific CPU design

- Two code sequences have the following instruction counts:

Program Code Sequence	Instruction counts for instruction class A	B	C
1	2	1	2
2	4	1	1

- CPU cycles for sequence 1 = 2 x 1 + 1 x 2 + 2 x 3 = 10 cycles

  CPI for sequence 1 = clock cycles / instruction count

  = 10 /5 = 2

i.e average or effective CPI

- CPU cycles for sequence 2 = 4 x 1 + 1 x 2 + 1 x 3 = 9 cycles

  CPI for sequence 2 = 9 / 6 = 1.5

$$CPU\ clock\ cycles = \sum_{i=1}^{n} \left(CPI_i \times C_i \right)$$

CPI = CPU Cycles / I

Instruction Frequency & CPI

- Given a program with n types or classes of instructions with the following characteristics:

$i = 1, 2, \ldots, n$

C_i = Count of instructions of type_i executed

CPI_i = Average cycles per instruction of type_i

F_i = Frequency or fraction of instruction type_i executed

= C_i/ total executed instruction count = C_i / I

Where: Executed Instruction Count $I = \Sigma C_i$

Then:

$$\underbrace{CPI}_{\text{i.e average or effective CPI}} = \sum_{i=1}^{n} \underbrace{\left(CPI_i \times F_i \right)}$$

Fraction of total execution time for instructions of type i = $\dfrac{CPI_i \times F_i}{CPI}$

$T = I \times CPI \times C$

Instruction Type Frequency & CPI

Program Profile or Executed Instructions Mix

$\dfrac{CPI_i \times F_i}{CPI}$

Depends on CPU Design

Given

Base Machine (Reg / Reg)

Op	Freq, F_i	CPI_i	$CPI_i \times F_i$	% Time
ALU	50%	1	.5	23% = .5/2.2
Load	20%	5	1.0	45% = 1/2.2
Store	10%	3	.3	14% = .3/2.2
Branch	20%	2	.4	18% = .4/2.2

Sum = 2.2

Typical Mix

$$CPI = \sum_{i=1}^{n} \left(CPI_i \times F_i \right)$$

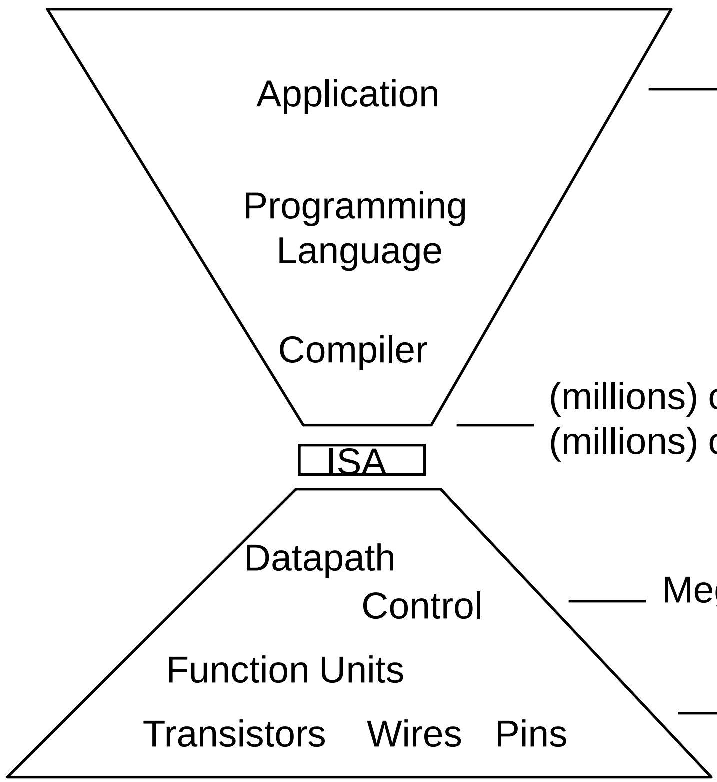
i.e average or effective CPI

CPI = .5 x 1 + .2 x 5 + .1 x 3 + .2 x 2 = 2.2
= .5 + 1 + .3 + .4

T = I x CPI x C

Metrics of Computer Performance

(Measures)



Execution time: Target workload, SPEC, etc.

(millions) of Instructions per second – MIPS
(millions) of (F.P.) operations per second – MFLOP/s

Megabytes per second.

Cycles per second (clock rate).

Each metric has a purpose, and each can be misused.

Choosing Programs To Evaluate Performance

Levels of programs or benchmarks that could be used to evaluate performance:

- Actual Target Workload: Full applications that run on the target machine.
- Real Full Program-based Benchmarks:
  - Select a specific mix or suite of programs that are typical of targeted applications or workload (e.g SPEC95, SPEC CPU2000).
- Small "Kernel" Benchmarks: Also called synthetic benchmarks
  - Key computationally-intensive pieces extracted from real programs.
    - Examples: Matrix factorization, FFT, tree search, etc.
  - Best used to test specific aspects of the machine.
- Microbenchmarks:
  - Small, specially written programs to isolate a specific aspect of performance characteristics: Processing: integer, floating point, local memory, input/output, etc.

Types of Benchmarks

Pros		Cons
• Representative	Actual Target Workload	• Very specific. • Non-portable. • Complex: Difficult to run, or measure.
• Portable. • Widely used. • Measurements useful in reality.	Full Application Benchmarks	• Less representative than actual workload.
• Easy to run, early in the design cycle.	Small "Kernel" Benchmarks	• Easy to "fool" by designing hardware to run them well.
• Identify peak performance and potential bottlenecks.	Microbenchmarks	• Peak performance results may be a long way from real application performance

SPEC: System Performance Evaluation Corporation

The most popular and industry-standard set of CPU benchmarks.

Programs application domain: Engineering and scientific computation

- SPECmarks, 1989:
  - 10 programs yielding a single number ("SPECmarks").
- SPEC92, 1992:
  - SPECInt92 (6 integer programs) and SPECfp92 (14 floating point programs).
- SPEC95, 1995:
  - SPECint95 (8 integer programs):
    - go, m88ksim, gcc, compress, li, ijpeg, perl, vortex
  - SPECfp95 (10 floating-point intensive programs):
    - tomcatv, swim, su2cor, hydro2d, mgrid, applu, turb3d, apsi, fppp, wave5
  - Performance relative to a Sun SuperSpark I (50 MHz) which is given a score of SPECint95 = SPECfp95 = 1
- SPEC CPU2000, 1999:
  - CINT2000 (11 integer programs). CFP2000 (14 floating-point intensive programs)
  - Performance relative to a Sun Ultra5_10 (300 MHz) which is given a score of SPECint2000 = SPECfp2000 = 100
- SPEC CPU2006, 2006:
  - CINT2006 (12 integer programs). CFP2006 (17 floating-point intensive programs)
  - Performance relative to a Sun Ultra Enterprise 2 workstation with a 296-MHz UltraSPARC II processor which is given a score of SPECint2006 = SPECfp2006 = 1

All based on execution time and give speedup over a reference CPU

Example Problem

Op	Freq	Cycles	CPI(i)	Time
ALU	50%	1	0.5	23%
Load	20%	5	0.5	45%
Store	10%	3	0.3	14%
Branch	20%	2	0.4	18%

Average CPI: 2.2

- **How much faster would the machine be of reduce load time to 2 cycle?**
- **How does it compare with branch prediction to reduce 1 cycle off branch?**
- **What if 2 ALU instructions can be executed at once?**

Example Problem

- Compiler has to decide between two code sequences for a single machine. Based on the hardware implementation there are 3 classes of instructions A, B, C. They require 1, 2 and 3 cycles respectively.
- First code sequence has 5 instr. 2 of A, 1 of B, 2 of C.
- Second has 6 instr. 4 of A, 1 of B, 1 of C

- Which sequence is faster? By how much? What is CPI for each sequence?

Example Problem

- A given application written in Java runs in 15 sec on a machine. A new Java compiler requires only 0.6 as many instr. As old. Unfortunately it raises CPI by 1.1 times.
- How fast can we expect the application to run using this fast compiler?

Example Problem

- Two compilers are being tested for a 4 GHz machine with three different classes of instr. A, B, C requiring 1, 2, 3 cycles respectively. Both compilers are used to produce code for a large software.
- First compiler uses 5 billion Class A, 1 bn class B, and 1 bn Class C.
- Second uses 10 bn A, 1 bn B, 1 bn C.
- Which sequence is faster in mips?
- Which sequence faster according to exec. time?

Example Problem

- A 1 GHz processor takes 100 seconds to execute a program, while consuming 70 W of dynamic power and 30 W of leakage power. Does the program consume less energy in Turbo boost mode when the frequency is increased to 1.2 GHz?

  Normal mode energy = 100 W x 100 s = 10,000 J
  Turbo mode energy = (70 x 1.2 + 30) x 100/1.2 = 9,500 J

  Note:
  Frequency only impacts dynamic power, not leakage power.
  We assume that the program's CPI is unchanged when frequency is changed, i.e., exec time varies linearly with cycle time.