EE 457 Unit 4

Computer System Performance

Motivation

- An individual user wants to:
  - Minimize single program execution time
- A datacenter owner wants to:
  - Maximize number of compute jobs performed per unit time
  - Minimize cost (power, # of servers, etc.)

http://e-telligentinternetmarketing.com/website/frustrated-computer-user-2/
http://www.intomobile.com/2010/11/02/opera-iceland-clean/

Performance Depends on View Point?!

- What's faster:
  - A 777 passenger airliner
  - An F-22 fighter jet
- If you are an individual interested in getting from point A to point B, then the F-22
  - This is known as latency [units of time]
  - Time from the start of an operation until it completes
- If you are trying to evacuate a large number of people, the airliner looks much better
  - This is known as throughput or bandwidth [jobs/time]

Throughput vs. Latency

- If Latency is the Time it takes for a Job to complete & Throughput = Jobs / Time…
- …Is Throughput = 1 / Latency?
  - No!
  - Latency is from the perspective of a single job
  - Throughput is from the perspective of many jobs
  - Parallelism is the great friend of throughput!
- We will see many times in this course (pipelining, memory org., etc.) that there is often not much we can do about latency but there are lots of ways to improve throughput
  - Hopefully without degrading latency too much, if at all

Metrics

- What are the metrics?
  - Clock speed (GHz),
  - IPS/OPS = Instructions/Operations Per Second
  - FLOPS = Floating Point Ops. Per Second
  - CPI, IPC = Clocks per Instruction (vice versa)
  - Memory Latency
  - Memory Bandwidth
  - Network bandwidth
  - FLOPS/Watt

Execution Time

- Key Point: When comparing different systems, *absolute execution time* is the ultimate criterion (metric)
- Using a rate as a metric can often be misleading metrics
  - Often not comparing apples to apples
  - Often not normalized

What's Wrong with Rates

- Two trains take two different routes from City A to City B and leave at the same time. Train 1 travels at 60 MPH, while train 2 travels at 75 MPH. Which one arrives first?
- Need to know how far each route is?
- Example 1 (MIPS):
  - You may hear that Computer 1 executes 500 MIPS while Computer 2 executes 750 MIPS. Which one executes a given program faster?
  - Train speed = MIPS & Routes = Program (how many instructions)
  - MIPS is only useful for the same compiled program run on 2 CPU's
- Example 2 (Clock Rate):
  - You may hear that CPU1 runs at 2 GHz and CPU2 runs at 3 GHz, which one executes a program faster (assume same instruction set)
  - CPU1 may have CPI=2 while CPU2 has CPI=4
  - CPU1 Time = 2/2GHz < CPU2 Time = 4/3GHz

Wall Clock Time vs. CPU Time

- Even execution time can be hard to measure accurately because the OS may allocate a percentage of compute cycles to other programs (also, part of a programs execution is spent in OS calls for I/O, etc.)
  - Wall Clock Time: Real time it took from when the user submitted the job until it was completed
  - CPU Time (User Time + System Time): Actual time the program used the CPU either in the application code (User Time) or in the OS (System Time)
    - Doesn't include I/O time
  - Linux/Unix: % time *executable*
    - real 0m16.019s
    - user 0m12.840s
    - sys 0m0.180s

Performance

- Performance is defined as the inverse of execution time

$$\text{Performance} = \frac{1}{\text{Execution Time}}$$

- Often want to compare relative performance or speedup (how many times faster is a new system than an old one)

$$\text{Speedup} = \frac{\text{Performance}_{\text{New}}}{\text{Performance}_{\text{Old}}} = \frac{\text{Execution}_{\text{Old}}}{\text{Execution}_{\text{New}}}$$

Performance Equation

- Execution time can be modeled using three components
  - Instruction Count: Total instructions executed by the program
    - IC = Dynamic Instruction Count not Static Instruction Count
  - Clocks Per Instruction (CPI): Average number of clock cycles to execute each instruction
  - Cycle Time: Clock period (1 / Freq.)

Compiler / Instruction Set

Microarchitecture

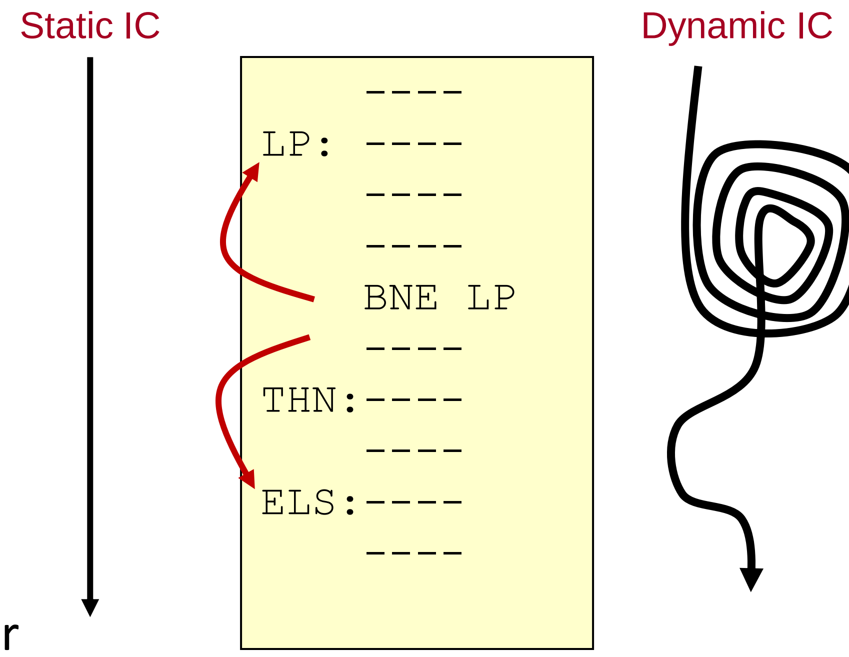
Technology (VLSI design)

$$\text{Exec. Time} = \text{Instruc. Count} * \frac{\text{Clocks}}{\text{Instruction}} * \frac{\text{Time}}{\text{Clock}}$$

$$= \text{Instruc. Count} * \text{CPI} * \text{Cycle Time}$$

Dynamic vs. Static Instruction Count

- Static instruction count is the number of written instructions
- Dynamic instruction count (or “trace” count) is how many instruction were executed at run time
- Would you prefer either:
  - Small Static IC & Large Dynamic IC … or …
  - Large Static IC & Small Dynamic IC

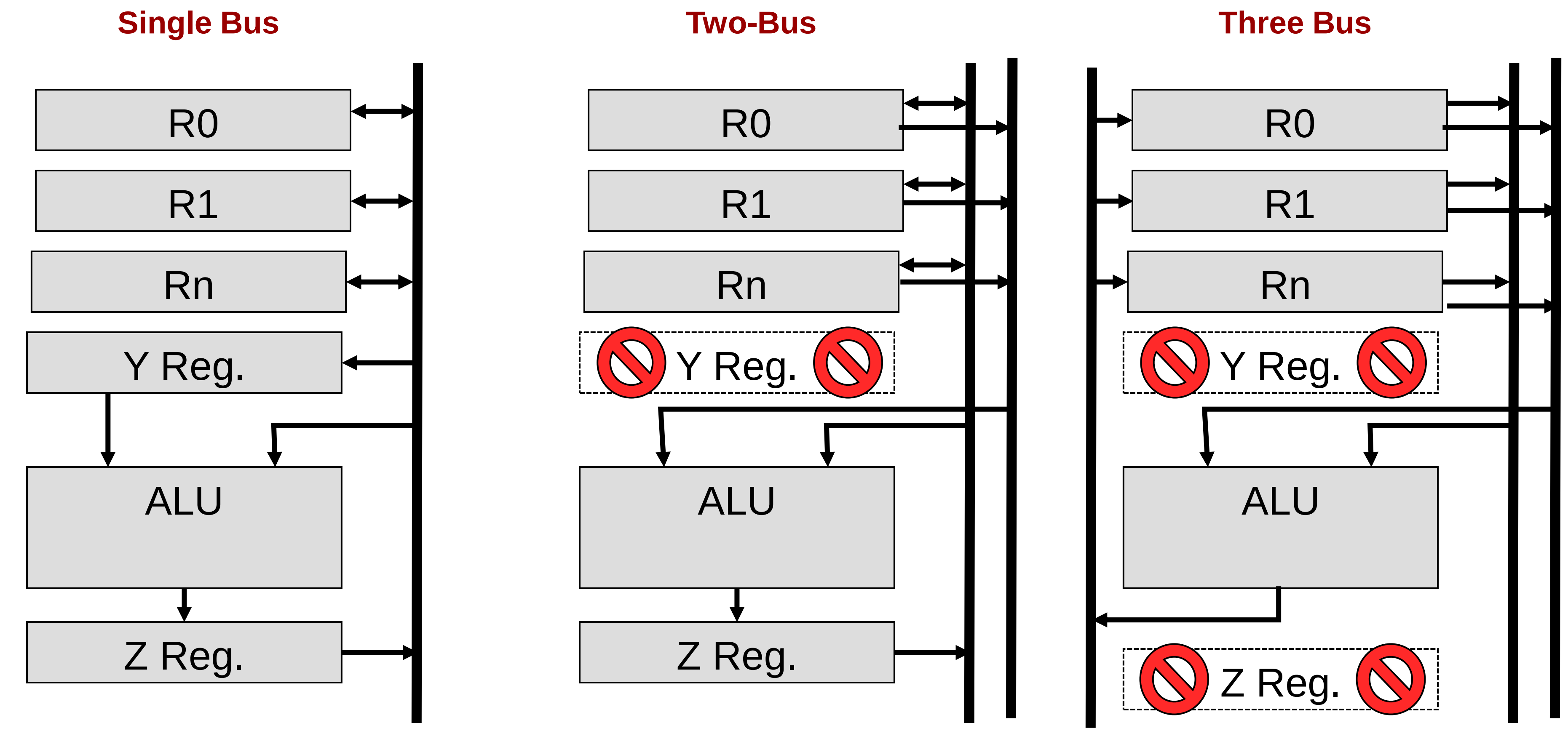


What Affects Performance

Component	SW/HW	Affects	Description
Algorithm	SW	Instruc. Count & CPI	Determines how many instructions & which kind are executed
Programming Language	SW	Instruc. Count & CPI	Determines constructs that need to be translated and the kind of instructions
Compiler	SW	Instruc. Count & CPI	Efficiency of translation affects how many and which instructions are used
Instruction Set	HW	Instruc. Count, CPI, Clock Cycle	Determines what instructions are available and what work each instruction performs
Microarchitecture	HW	CPI, Clock Cycle	Determines how each instruction is executed (CPI, clock period)

Source: H&P, Computer Organization & Design, 3rd Ed.

Different Architectures



Clock 1: Y = Rsrc1
Clock 2: Z = Rsrc2 + Y
Clock 3: Rdst = Z

Clock 1: Z = Rsrc1 + Rsrc2
Clock 2: Rdst = Z

Clock 1: Rdst = Rsrc1 + Rsrc2

General Implications: Less Resources => More Clock Cycles (Time)

Example

- Processor A runs at 200 MHz and executes a 40 million instruction program at a sustained 50 MIPS
- Processor B runs at 400 MHz and executes the same program (w/ a different compiler) which yields a count of 60 million instructions and a CPI of 6
- What is the CPI of the program on Proc. A?
- Which processor executes the program faster and by what factor?
- What is the MIPS rate of Proc. B?

$$CPI_A = \frac{200*10^6\,cycles}{second} * \frac{second}{50*10^6\,instrucs}$$

$$ExecTime_A = 40*10^6\,instrucs.* \frac{second}{50*10^6\,instrucs.} = 0.8\,\text{sec}$$

$$ExecTime_B = 60*10^6\,instrucs.* \frac{6cycles}{instruc.} * \frac{second}{400*10^6\,cycles} = 0.9\,\text{sec}$$

$$Speedup = \frac{ExecTime_B}{ExecTime_A} = \frac{0.9}{0.8} = 1.125$$

$$MIPS_B = \frac{60*10^6\,instrucs}{0.9 seconds} = 66.67\,MIPS$$

Calculating CPI

- CPI can be found by taking the expected value (weighted average) of each instruction type's CPI [i.e. CPI for each type * frequency (probability) of that type of instruction]

$$CPI = \sum_i CPI_{Type\_i} * P(InstructionType_i)$$

- In practice, CPI is often hard too find analytically because in modern processors instruction execution is dependent on earlier instructions
  – Instead we run benchmark applications on simulators to measure average CPI.

Example

Instruction Type	CPI P1
A	1
B	2
C	3

If CLK=1 MHz what is PEAK Inst./Sec. = 1 MIPS

Average CPI = (1+2+3)/3 = 2

Instruction Type	CPI P1	Freq.
A	1	10%
B	2	40%
C	3	50%

Average CPI = 1*0.10 + 2*0.40 + 3*0.5
= .10+.80+1.5 = 2.40

Example

- Calculate CPI of this snippet of code using the following CPI's for each instruction type

```
        add     $s0,$zero,$zero
        addi    $t1,$zero,4
loop:   lw      $t2,0($t0)
        add     $t2,$t2,$t1
        addi    $t0,$t0,4
        addi    $t1,$t1,-1
        bne     $t1,$zero,loop
        sw      $t2,0($t2)
```

Instruction Type	CPI
add	1
lw / sw	4
bne	2

Dynamic Instruction Count = 4*5 + 3 = 23

$$CPI = \sum_i CPI_{Type\_i} * P(InstructionType_i)$$

Instruction Type	Dynamic Count
add	14
lw / sw	5
bne	4

$$CPI = \frac{1}{23}\sum (1*14)+(4*5)+(2*4) = \frac{42}{23} = 1.826$$

Other Performance Measures

- OPS/FLOPS = (Floating-Point) Operations/Sec.
  – Maximum number of arithmetic operations per second the processor can achieve
  – Example: 4 FP ALU's on a processor running @ 2 GHz => 8 GFLOPS
- Memory Bandwidth (Bytes/Sec.)
  – Maximum bytes of memory per second that can be read/written
- Programs are either memory bound or computationally bound
- Performance/Watt, Energy Proportionality, etc.

Energy Proportional Computing

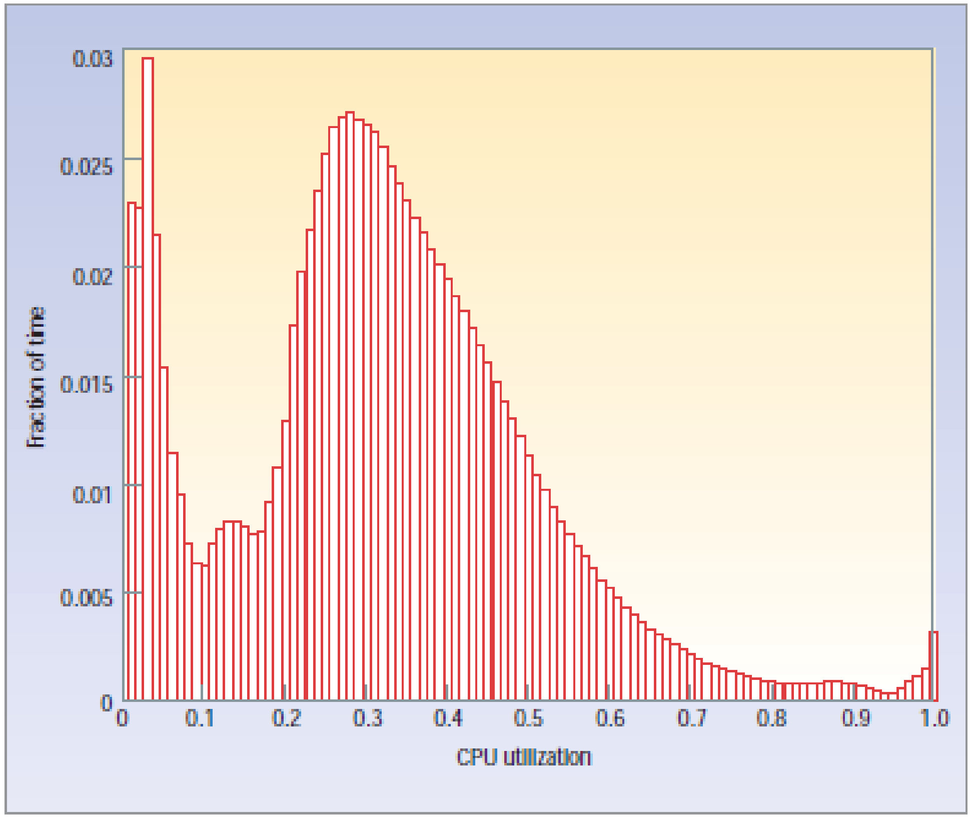
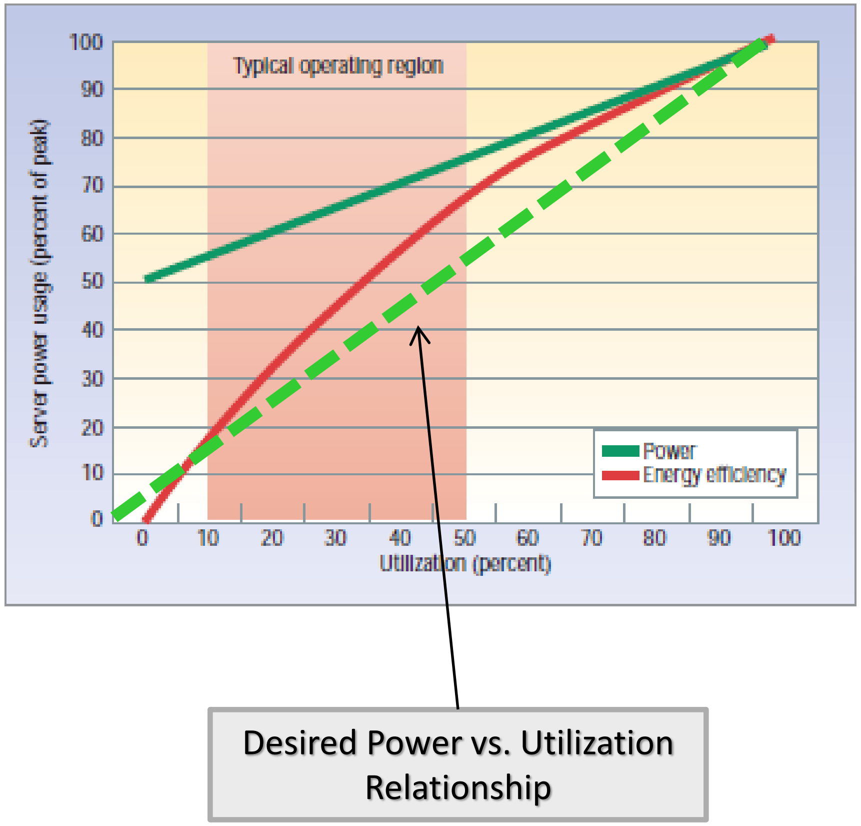


Figure 1. Average CPU utilization of more than 5,000 servers during a six-month period. Servers are rarely completely idle and seldom operate near their maximum utilization, instead operating most of the time at between 10 and 50 percent of their maximum utilization levels.



"The Case for Energy-Proportional Computing", Luiz André Barroso, Urs Hölzle, *IEEE Computer*, vol. 40 (2007).

What should I optimize?

AMDAHL'S LAW

Amdahl's Law

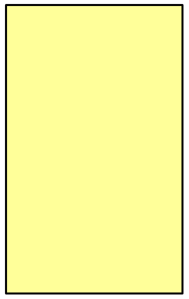
- Where should we put our effort when trying to enhance performance of a program
- Amdahl's Law = How much performance gain do we get by improving only a part of the whole

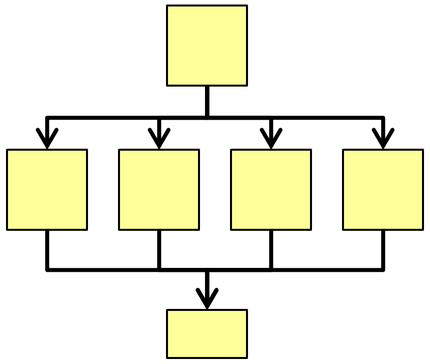
$$ExecTimeNew = ExecTimeUnaffected + \frac{ExecTimeAffected}{ImprovementFactor}$$

$$Speedup = \frac{ExecTimeOld}{ExecTimeNew} = \frac{1}{Percent_{Unaffected} + \frac{Percent_{Affected}}{ImprovementFactor}}$$

Amdahl's Law

- Holds for both HW and SW
  - HW: Which instructions should we make fast? The most used (executed) ones
  - SW: Which portions of our program should we work to optimize
- Holds for parallelization of algorithms (converting code to run multiple processors)


Original Sequential Program


Parallelized Program

Parallelization Example

- A programmer parallelizes a function in his program to be run on 8 cores. The function accounted for 40% of the runtime of the overall program. What is the speedup of the enhancement?

$$Speedup = \frac{1}{0.6 + \frac{0.4}{8}} = \frac{1}{0.65} = 1.53$$

Example

- What if we improve only class B instrucs.

Instruction Type	CPI P1	Freq.
A	1	30%
B	~~2~~ => 1	20%
C	3	50%

$$Speedup = \frac{1}{Percent_{Unaffected} + \frac{Percent_{Affected}}{ImprovementFactor}}$$

$$Speedup = \frac{1}{2/3 + (1/3/2)} = \frac{1}{5/6} = 6/5 = 1.2$$

…OR…

Must put percentages in terms of time

$$Speedup = \frac{1}{\frac{9}{11} + (\frac{2}{11}/2)} = \frac{1}{\frac{10}{11}} = 11/10 = 1.1$$

Profiling

- How do you know where time is being spent?
- From a software (programming for performance) perspective, profilers are handy tools
  - Instrument your code to take statistics as it runs and then can show you what percentage of time each function or even line of code was responsible for
  - Common profilers
    - 'gprof' (usually standard with Unix / Linux installs) and 'g++'
    - Intel VTune
    - MS Visual Studio Profiling Tools
- From a hardware perspective, simulators can help
  - SimpleScalar
  - Simics
  - Your own simulation model developed in Verilog/SystemC/etc.

gprof Output

```
%   cumulative   self              self     total
time   seconds   seconds    calls   s/call   s/call  name
42.96      4.48     4.48 56091649     0.00     0.00  Board::operator<(Board const&) const
 6.43      5.15     0.67  2209524     0.00     0.00  std::_Rb_tree<...>::_M_lower_bound(...)
 5.08      5.68     0.53 108211500     0.00     0.00  __gnu_cxx::__normal_iterator<...>::operator+(...)
 4.51      6.15     0.47  4419052     0.00     0.00  Board::Board(Board const&)
 4.32      6.60     0.45  1500793     0.00     0.00  void std::__adjust_heap<...>(...)
 3.84      7.00     0.40 28553646     0.00     0.00  PuzzleMove::operator>(PuzzleMove const&) const
```

Credits

- These slides were derived from Gandhi Puvvada’s EE 457 Class Notes

BACKUP

An Opening Question

- An Intel and a Sun/SPARC computer measure their respective rates of instruction execution on the same application written in C
  - Computer A achieves 160 MIPS (Millions of Instructions Per Second)
  - Computer B achieves 200 MIPS
- Which computer executes the program faster?
  - It depends on the instruction set and compiler (ultimately, the instruction count). Computer B and its compiler may use many more simpler (faster) instructions to implement the program thereby increasing its instruction execution rate but saying nothing of overall execution time

Another Question

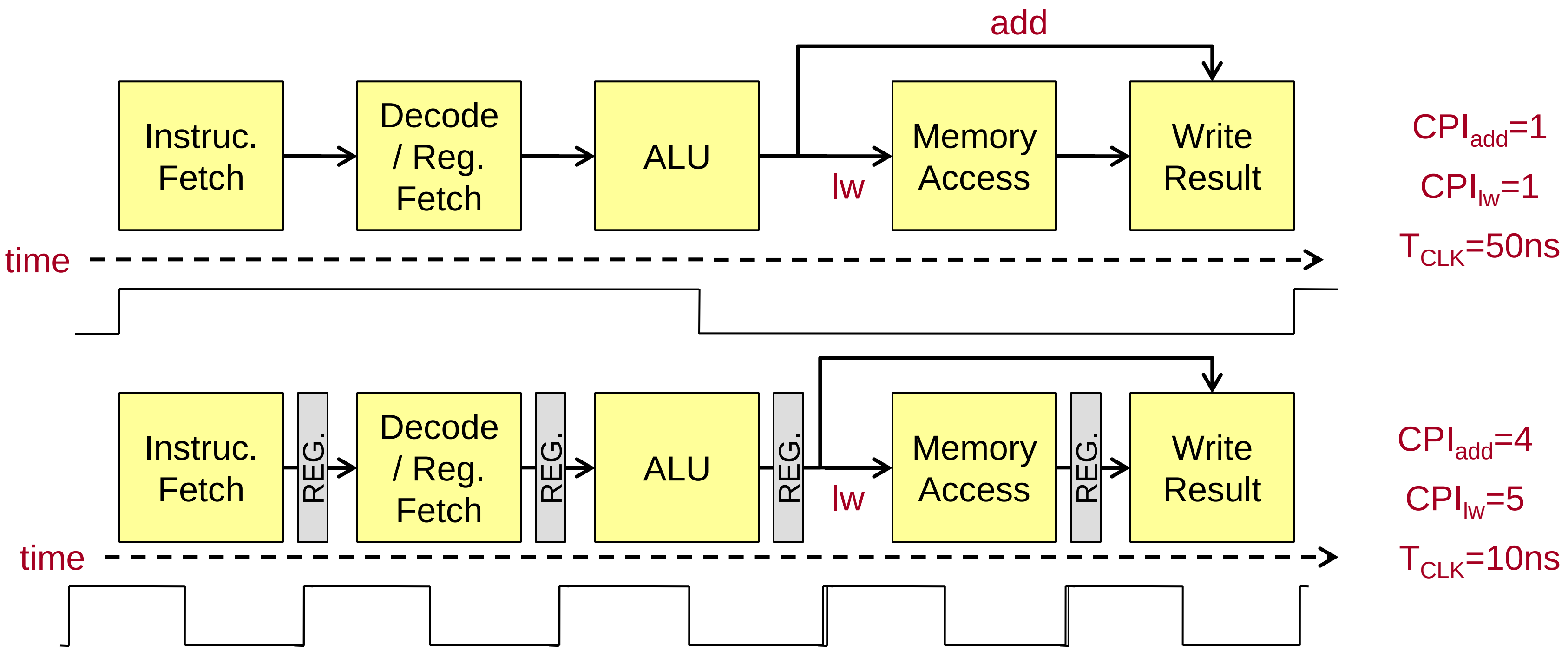
- A Pentium 3 has a clock rate of 1 GHz while a Pentium 4 has a clock rate of 2 GHz.
  - They implement the same instruction set
  - They are tested on the same executable program.
- Is the Pentium 4 twice as fast as the Pentium 3?
  - Since they both use the same instructions and the same instruction count (same executable), we may think that the Pentium 4 would be twice as fast
  - However, the microarchitectural implementation of the processor may mean that the Pentium 3 executes instructions in 2 clocks on average while the Pentium 4 executes instruction in 4 clocks on average thus making the execution time exactly the same.

What Affects Speed

HW/SW Design	**Corresponding Metric**
• Compiler & Instruction Set – What instructions we support, how high level programs are translated, what instructions the compiler uses	• Instruction Count – Static vs. dynamic
• Microarchitecture – Clocks per Instruction (CPI) or its reciprocal, IPC	• CPI (Clocks per instruction) or its reciprocal, IPC
• IC Technology – Gate/wire delay – Clock	• Clock frequency / clock period

Microarchitecture's Effect

- Micro-architecture affects performance via the CPI and clock period
  - Consider LW and ADD instruction
    - LW (Fetch, Decode & Fetch Base Reg., Add Base+Offset, Read Mem., Write Result
    - ADD (Fetch, Decode & Fetch Src. Reg's., Add, Write Result)



Example

- Two different processors implement the same instruction set (such as Intel and AMD processors).
- There are four instruction classes (types) with the given CPI's. P1 has a clock rate of 2 GHz and P2 has a clock rate of 3 GHz.
- A certain program has an instruction mix in which classes A and B are executed twice as often as C and D
- Which computer is faster and by how much?

Average CPI (P1): 11/6

Average CPI (P2): 14/6

Exec. Time (P1): IC * 11/6 * 0.5 ns

Exec. Time (P2): IC * 14/6 * 0.333 ns

Speedup = 11/12 / 14/18 = 1.17

Instruction Type	CPI P1	CPI P2
A	1	2
B	2	2
C	2	2
D	3	4