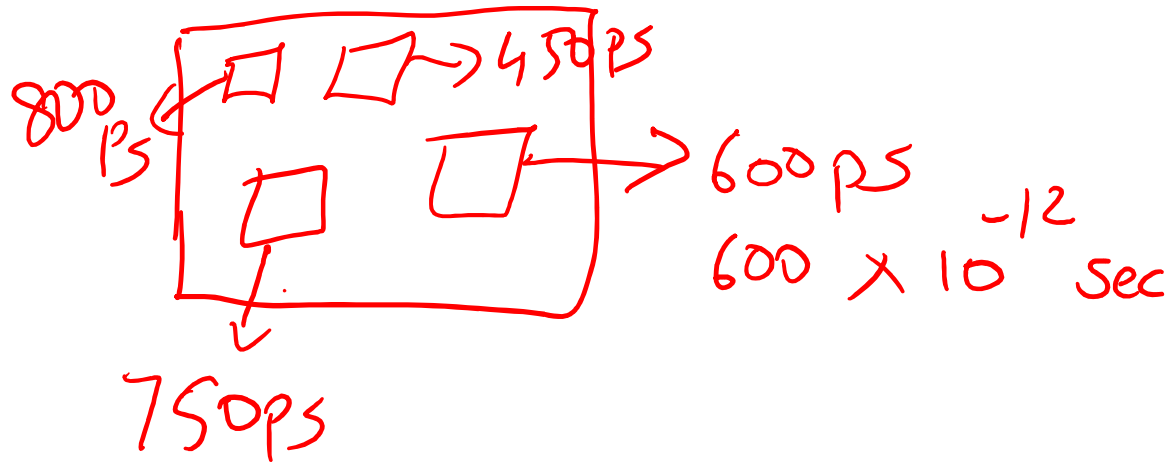


Lecture 3: Performance/Power, MIPS Instructions

- Today's topic:
 - More performance/power equations, examples
 - MIPS instructions
- HW1 is due on Thursday (+ 1.5 days)
- TA office hours (CADE Lab, TA queue)

Piazza to be added

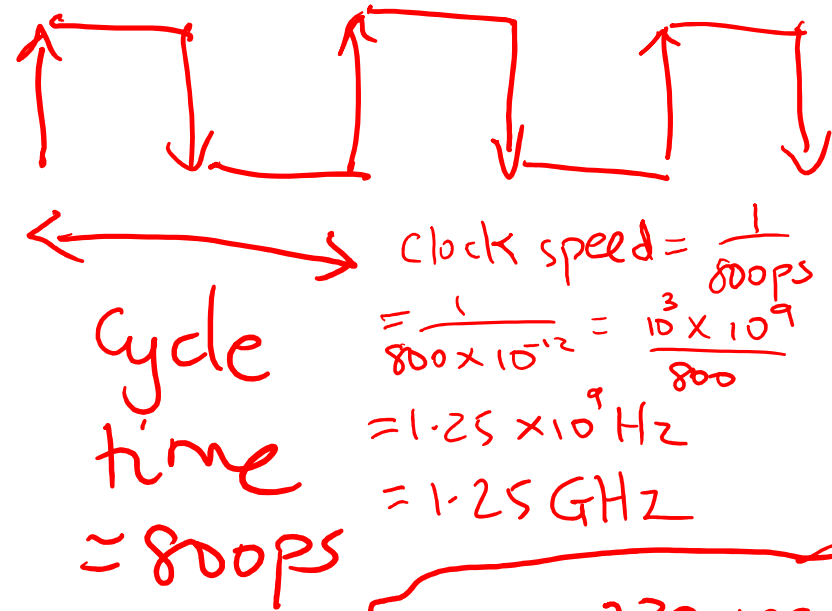
10^{-9} nano ; 10^{-6} micro ; 10^{-3} milli Dyn power \propto freq
A Primer on Clocks and Cycles



synchronous

instr per cycle = IPC
 Typically, say $IPC = 0.5$, i.e., the adder is able to productively use half its opportunities to do work.

1.25 billion cycles (opportunities) to do work in a sec
 0.625 billion instrs completed in 1 sec
 Throughput (instrs per second) = $IPC \times \text{clock speed}$



330 ms

cycle time = 0.33s

clock frequency = $\frac{1}{\text{cycle time}}$
 = how many cycles in 1 sec
 $= \frac{1}{0.33 \text{ s}}$
 $\approx 3 \text{ Hz}$

Performance Equation - I

$\text{CPU execution time} = \text{CPU clock cycles} \times \text{Clock cycle time}$
 $\text{Clock cycle time} = 1 / \text{Clock speed}$

If a processor has a frequency of 3 GHz, the clock ticks 3 billion times in a second – as we'll soon see, with each clock tick, one or more/less instructions may complete

If a program runs for 10 seconds on a 3 GHz processor, how many clock cycles did it run for?

$3 \times 10^9 \text{ cycles} \times 10 \text{ sec} = 30 \times 10^9 \text{ cycles}$

If a program runs for 2 billion clock cycles on a 1.5 GHz processor, what is the execution time in seconds?

$\text{exec time} = 2 \times 10^9 \times \frac{1}{1.5 \times 10^9} = 1.33 \text{ secs}$

cycle time
 $= \frac{1}{3 \text{ GHz}}$
 $= \frac{1}{3 \times 10^9}$
 $= 0.33 \times 10^{-9} \text{ s}$
 $= 0.33 \text{ ns}$

Performance Equation - II

$$IPC = \frac{1}{CPI}$$

CPU clock cycles = number of instrs x avg clock cycles
per instruction (CPI)

Substituting in previous equation,

(in secs)

Execution time = clock cycle time x number of instrs x avg ^{cycles} CPI

If a 2 GHz processor graduates an instruction every third cycle, ^{CPI=3}
how many instructions are there in a program that runs for
10 seconds?

$$10 \text{ sec} = \frac{1}{2 \times 10^9} \times \text{instrs} \times 3$$

$$\text{instrs} = \frac{10 \times 2 \times 10^9}{3} = 6.66 \times 10^9 \text{ B instrs}$$

Factors Influencing Performance

Execution time = clock cycle time x number of instrs x avg CPI

- Clock cycle time: manufacturing process (how fast is each transistor), how much work gets done in each pipeline stage (more on this later)
- Number of instrs: the quality of the compiler and the instruction set architecture
- CPI: the nature of each instruction and the quality of the architecture implementation

Example

$$Exec_{MIPS} = \frac{1}{1 \times 10^9} \times 4 \times 10^9 \times 1.5 = 6 \text{ secs}$$

$$Exec_{x86} = \frac{1}{1.5 \times 10^9} \times 2 \times 10^9 \times 6 = 8 \text{ secs}$$

Execution time = clock cycle time x number of instrs x avg CPI

Which of the following two systems is better?

- A program is converted into 4 billion MIPS instructions by a compiler ; the MIPS processor is implemented such that each instruction completes in an average of 1.5 cycles and the clock speed is 1 GHz
- The same program is converted into 2 billion x86 instructions; the x86 processor is implemented such that each instruction completes in an average of 6 cycles and the clock speed is 1.5 GHz

Intel

Power and Energy

80% / v

20% / v

- Total power = dynamic power + leakage power
- Dynamic power \propto activity x capacitance x voltage² x frequency
 \downarrow proportional to
- Leakage power \propto voltage

- Energy = power x time
(joules) (watts) (sec)

$$300 \text{ J} = 150 \text{ W} \times 2 \text{ s}$$

- For a CPU-bound program,
Execution time \propto cycle time \propto 1 / clock speed

→ doing only number crunching

[mem-bound]

Example Problem

- A 1 GHz processor takes 100 seconds to execute a CPU-bound 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?

Nominal system

$$\text{Power} = \text{Dyn} + \text{Lkg} = 70 + 30 = 100 \text{ W}$$

$$\text{Exec time} = 100 \text{ s}$$

$$\text{Energy} = \text{Power} \times \text{time} = 100 \text{ W} \times 100 \text{ s} = 10 \text{ KJ}$$

Turbo system : 1 GHz \rightarrow 1.2 GHz

$$\text{Exec time} : 100 \text{ s} \rightarrow \frac{100 \times 1}{1.2} = 83.3 \text{ secs}$$

$$\text{Dyn power} : 70 \text{ W} \times 1.2 = 84 \text{ W}$$

$$\text{Energy} = (84 + 30) \times 83.3 = 9500 \text{ J}$$

Example Problem

- A 1 GHz processor takes 100 seconds to execute a CPU-bound 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 = $100 \text{ W} \times 100 \text{ s} = 10,000 \text{ J}$

Turbo mode energy = $(70 \times 1.2 + 30) \times 100/1.2 = \underline{\underline{9,500 \text{ J}}}$

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.

Benchmark Suites

- Each vendor announces a SPEC rating for their system
 - a measure of execution time for a fixed collection of programs
 - is a function of a specific CPU, memory system, IO system, operating system, compiler
 - enables easy comparison of different systems

The key is coming up with a collection of relevant programs

SPEC CPU

- SPEC: System Performance Evaluation Corporation, an industry consortium that creates a collection of relevant programs
- SPEC 2006 includes 12 integer and 17 floating-point applications
- The SPEC rating specifies how much faster a system is, compared to a baseline machine – a system with SPEC rating 600 is 1.5 times faster than a system with SPEC rating 400
- Note that this rating incorporates the behavior of all 29 programs – this may not necessarily predict performance for your favorite program!
- Latest version: SPEC 2017

Deriving a Single Performance Number

How is the performance of 29 different apps compressed into a single performance number?

- SPEC uses geometric mean (GM) – the execution time of each program is multiplied and the N^{th} root is derived
- Another popular metric is arithmetic mean (AM) – the average of each program's execution time
- Weighted arithmetic mean – the execution times of some programs are weighted to balance priorities

Amdahl's Law

- Architecture design is very bottleneck-driven – make the common case fast, do not waste resources on a component that has little impact on overall performance/power
- Amdahl's Law: performance improvements through an enhancement is limited by the fraction of time the enhancement comes into play
- Example: a web server spends 40% of time in the CPU and 60% of time doing I/O – a new processor that is ten times faster results in a 36% reduction in execution time (speedup of 1.56) – Amdahl's Law states that maximum execution time reduction is 40% (max speedup of 1.66)

Common Principles

- Amdahl's Law
- Energy: performance improvements typically also result in energy improvements – less leakage
- 90-10 rule: 10% of the program accounts for 90% of execution time
- Principle of locality: the same data/code will be used again (temporal locality), nearby data/code will be touched next (spatial locality)

Recap

- Knowledge of hardware improves software quality: compilers, OS, threaded programs, memory management
- Important trends: growing transistors, move to multi-core and accelerators, slowing rate of performance improvement, power/thermal constraints, long memory/disk latencies
- Reasoning about performance: clock speeds, CPI, benchmark suites, performance and power equations
- Next: assembly instructions

Instruction Set

- Understanding the language of the hardware is key to understanding the hardware/software interface
- A program (in say, C) is compiled into an executable that is composed of machine instructions – this executable must also run on future machines – for example, each Intel processor reads in the same x86 instructions, but each processor handles instructions differently
- Java programs are converted into portable bytecode that is converted into machine instructions during execution (just-in-time compilation)
- What are important design principles when defining the instruction set architecture (ISA)?

A Basic MIPS Instruction

C code: `a = b + c ;`

Assembly code: (human-friendly machine instructions)
`add a, b, c # a is the sum of b and c`

Machine code: (hardware-friendly machine instructions)
`00000010001100100100000000100000`

Translate the following C code into assembly code:
`a = b + c + d + e;`