

Evaluating processor performance

The iron law of performance

Introduction

Estimating performance


Comparing microarchitectures

Conclusion

Performance metrics

- There are many ways to measure performance
 - Marketing in the 90s and 2000s focused on frequency
- Some performance metrics hide the actual performance we care about
- What do we care about?
 - How long will it take me to run this program on this machine?
 - Execution time!

Estimating execution time



A back-of-the-envelope model

| The Iron Law of Performance

$$\textit{Execution Time} = \left(\frac{\textit{instructions}}{\textit{program}} \right) \left(\frac{\textit{cycles}}{\textit{instruction}} \right) \left(\frac{\textit{seconds}}{\textit{cycle}} \right)$$

I Instructions per program

- Static instruction count
 - The number of instructions in the program executable
 - Impacts the size of the executable
- Dynamic instruction count
 - The number of instructions executed by the program from start to end
 - Impacts performance
 - Influenced by, e.g., algorithms, compiler, ISA, and the input(s) to the program
- Architects typically assume this is constant per program

Cycles per instruction (CPI)

- Number of cycles to execute the average instruction
 - Heavily influenced by the microarchitecture
- A measure of latency
 - Inverse is throughput: Instructions per cycle (IPC)
- For now, assume an ideal memory system
 - i.e., instructions can access memory within a cycle
 - Not a realistic assumption

I Seconds per cycle (T_c)

- Microarchitecture
 - The critical path through one micro-architecture is not the same as another
 - Improving combinational elements is also helpful
- Technology
 - Advances in manufacturing
 - Trade secrets in how transistors are laid out and connected

Comparing microarchitectures



Single-cycle vs. pipelining

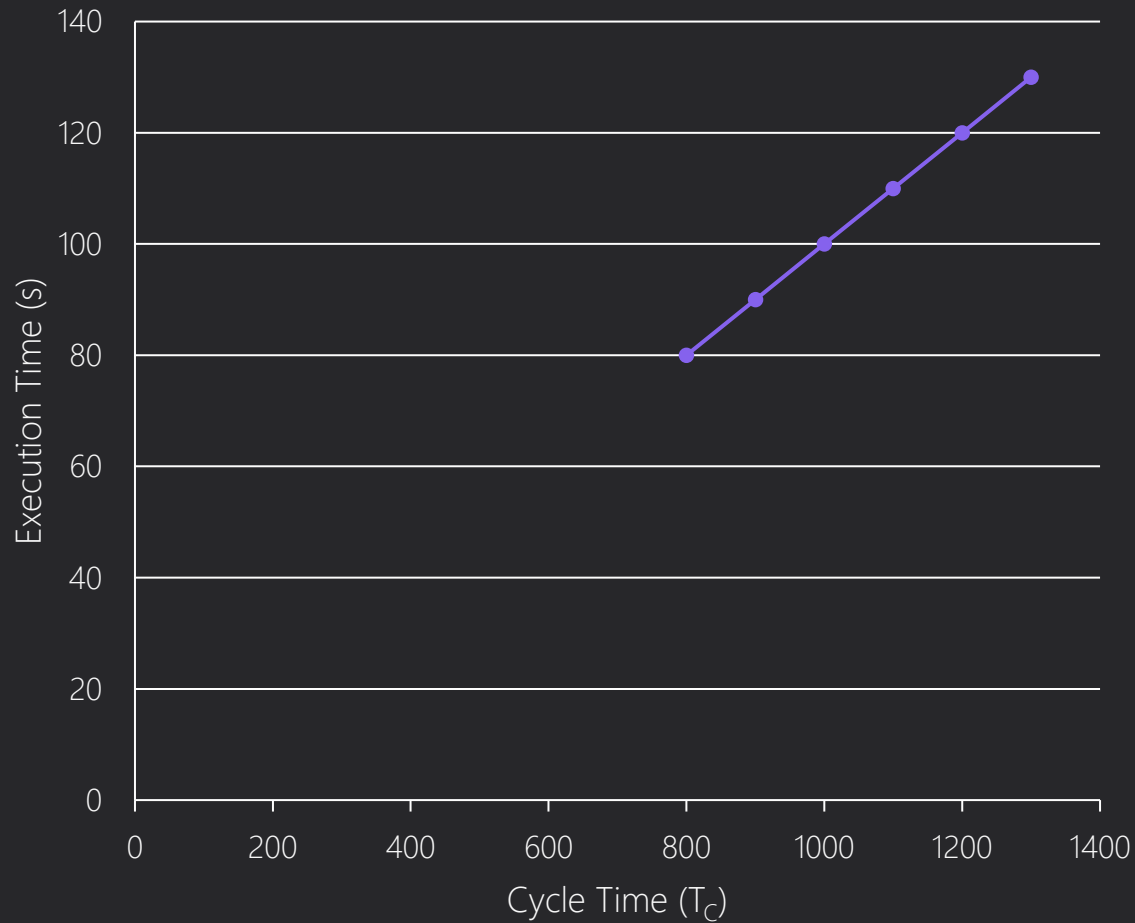
Single-cycle processor

- One instruction takes one cycle
 - So, $CPI = 1$
- Cycle time (period) limited to slowest instruction
- So, $Execution\ Time = \left(\frac{instructions}{program} \right) (T_c)$

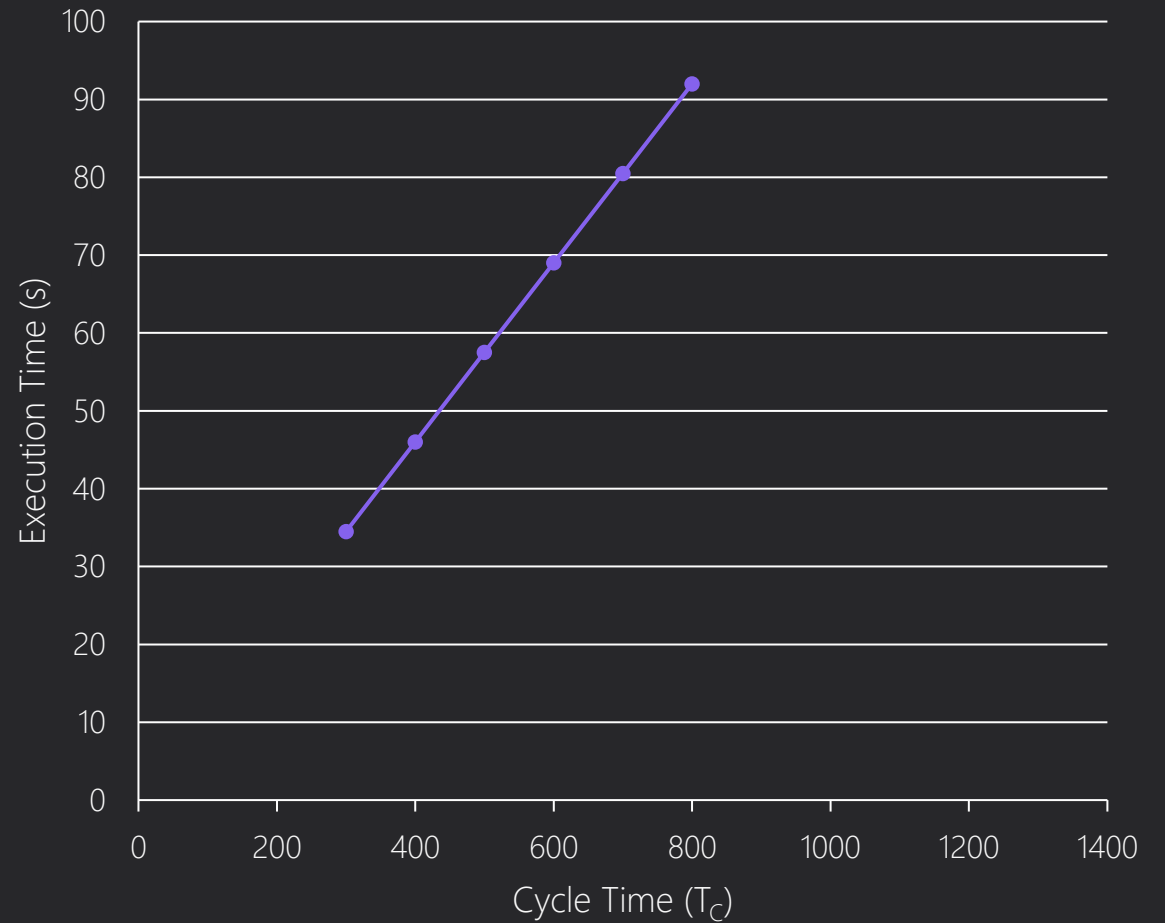
Pipelined processor

- The ideal CPI is 1, but the ideal is improbable
 - Dependencies between instructions
 - Branch instructions
 - So, $CPI = 1 + (overhead)$
- Cycle time limited to slowest pipeline stage
 - Recall a stage only executes one part of the instruction

Sweeping period



- Assume 100 billion instructions (per program)
- Single-cycle: $CPI = 1$; $T_C = 800$ ps to 1400 ps
- Pipelined processor: $CPI = 1.15$; $T = 300$ to 800 ps



Conclusion



Recapping the important points

Performance analysis of computer systems

- Performance analysis is a complex art and science
 - We have only scraped the surface
 - New performance metric: CPI
- Even with sequencing overhead, pipelined architectures can operate at very high frequencies
- A proper performance analysis needs to evaluate more than one program
 - The CPI of a pipelined processor is heavily tied to the program it is running