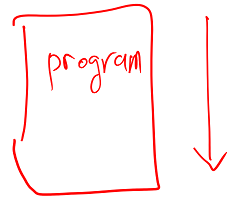
CPU Performance

- Execution time for a program.
  - CPU time
- Number of cycles for a program. ← instructions
  - CPU clock cycle
- Number of cycles in one second.
  - CPU clock rate $= \frac{1}{\text{Clock cycle time}}$
- Time for a single cycle.
  - Clock cycle time $= \frac{1}{\text{CPU clock rate}}$



CPU Time

= CPU clock cycle

X clock cycle time

Instruction Count and CPI

- Instruction Count for a program
  - Determined by program, ISA and compiler
- Average Cycles Per Instruction
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

$$CPI = \frac{\text{clock cycles}}{\text{\# of instruction}}$$

clock cycles = CPI x # of instruction

$$\text{CPU time} = \underbrace{\text{CPU clock cycles}}_{\text{CPI x \# of instruction}} \times \text{clock cycle time}$$

CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

$Exc_A = CPI_A * \#\text{inst} * CCT_A$
$= 2.0 * \#\text{inst} * 250$
$= 50 * \#\text{inst}$

$Exc_B = CPI_B * \#\text{inst} * CCT_B$
$= 1.2 * \#\text{inst} * 500$
$= 60 * \#\text{inst}$

$\frac{60 \cancel{* \#\text{inst}}}{50 \cancel{* \#\text{inst}}} = 1.2 \Rightarrow 20\%$ faster

CSULB

CPI in More Detail

- If different instruction classes take different numbers of cycles

$$\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)$$

- Weighted average CPI

$$\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left(\text{CPI}_i \times \frac{\text{Instruction Count}_i}{\text{Instruction Count}} \right)$$

CSULB

CPI Example

IC = # of instruction

- Alternative compiled code sequences using instructions in classes A, B, C

ALU ← A; Load or Save ← B

Class	A	B	C
CPI for class	1	2	3
IC in sequence 1	2	1	2
IC in sequence 2	4	1	1

← Program 1 (IC in sequence 1)

← Program 2 (IC in sequence 2)

$$CPI_1 = \frac{1}{5}(1\times2 + 2\times1 + 3\times2) = \frac{10}{5} = 2$$

$$CPI_2 = \frac{1}{6}(4 + 2 + 3) = \frac{9}{6} = 1.5$$

CSULB

Performance Summary

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, Tc

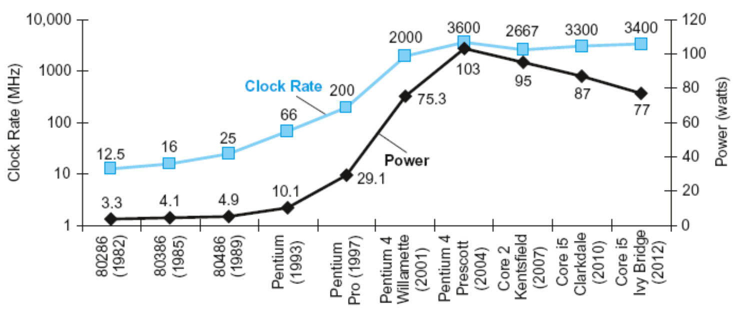
$$\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

CSULB

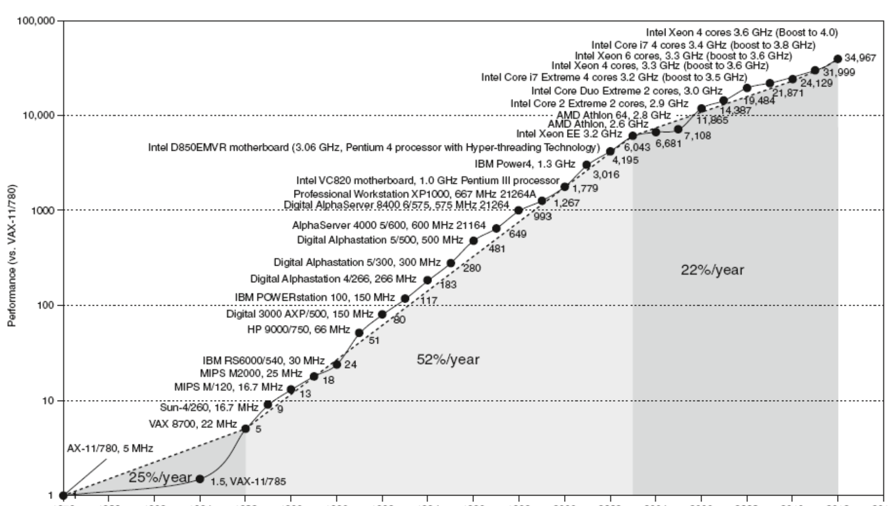
Power Usage Trends

- High power IC leads to high heat system.
- In CMOS IC technology

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$



Uniprocessor Performance



Multiprocessors

- Multicore microprocessors
  - More than one processor per chip
- Requires explicitly parallel programming
  - Compare with instruction level parallelism
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Hard to do
    - Programming for performance
    - Load balancing
    - Optimizing communication and synchronization

CSULB

Workloads and Benchmark

- Image you want to purchase a new laptop.
- Workload
  - Programs used to measure performance
- Standard Performance Evaluation Corp (SPEC)
  - Develops benchmarks for CPU, I/O, Web, …
- SPEC CPU2006
  - Elapsed time to execute a selection of programs
    - Negligible I/O, so focuses on CPU performance
  - Normalize relative to reference machine

CSULB

CINT2006 for Intel Core i7 920

Description	Name	Instruction Count x 10^9	CPI	Clock cycle time (seconds x 10^{-9})	Execution Time (seconds)	Reference Time (seconds)
Interpreted string processing	perl	2252	0.60	0.376	508	9770
Block-sorting compression	bzip2	2390	0.70	0.376	629	9650
GNU C compiler	gcc	794	1.20	0.376	358	8050
Combinatorial optimization	mcf	221	2.66	0.376	221	9120
Go game (AI)	go	1274	1.10	0.376	527	10490
Search gene sequence	hmmer	2616	0.60	0.376	590	9330
Chess game (AI)	sjeng	1948	0.80	0.376	586	12100
Quantum computer simulation	libquantum	659	0.44	0.376	109	20720
Video compression	h264avc	3793	0.50	0.376	713	22130
Discrete event simulation library	omnetpp	367	2.10	0.376	290	6250
Games/path finding	astar	1250	1.00	0.376	470	7020
XML parsing	xalancbmk	1045	0.70	0.376	275	6900
Geometric mean	–	–	–	–	–	–

CSULB

Pitfall: Amdahl's Law

- Improving an aspect of a computer and expecting a proportional improvement in overall performance.

$$T_{improved} = \frac{T_{affected}}{\text{improvement factor}} + T_{unaffected}$$

	A	B	C	
Tunaf	1	2	3	⇐ original
Taff	1→1.2	2	3	

- Example
  - A program runs in 100 seconds with multiply operations for 80 seconds. Can I run the program in 20 seconds?

$$\cancel{20} = \frac{80}{n} + \cancel{20}$$

$$0 = \frac{80}{n} \qquad n \to \infty$$

$$100 = \underbrace{80}_{\uparrow\ \text{multiply}} + \underbrace{20}_{\uparrow\ \text{others}}$$

CSULB

Fallacy: Low Power at Idle

- Computer at low utilization use little power.
- Example
  - Look back at i7 power benchmark
    - At 100% load: 258W
    - At 50% load: 170W (66%)
    - At 10% load: 121W (47%)

CSULB

Concluding Remarks

- Cost/performance is improving
  - Due to underlying technology development
- Hierarchical layers of abstraction
  - In both hardware and software
- Instruction set architecture
  - The hardware/software interface
- Execution time: the best performance measure
- Power is a limiting factor
  - Use parallelism to improve performance

CSULB