CS4617 Computer Architecture

Lecture 2

Dr J Vaughan

September 10, 2014

Amdahl's Law

- $Speedup = \frac{Execution\ time\ for\ entire\ task\ without\ using\ enhancement}{Execution\ time\ for\ entire\ task\ using\ enhancement\ when\ possible}$
- $Speedup_{overall} = \frac{Execution\ time_{old}}{Execution\ time_{new}}$
- $Speedup_{overall} = \frac{1}{(1 - Fraction_{enhanced}) + \frac{Fraction_{enhanced}}{Speedup_{enhanced}}}$

Example

- Processor enhancement: New CPU ten times faster
- If original CPU is busy 40% of time and waits for I/O 60% of time, what is overall speedup?
- $Fraction_{enhanced} = 0.4$
- $Speedup_{enhanced} = 10$
- $Speedup_{overall} = \frac{1}{0.6 + \frac{0.4}{10}}$
- ≈ 1.56

Example

- Floating point square root (FPSQR) enhancement
- Suppose FPSQR responsible for 20% of a graphics benchmark.
- Suppose FP instructions responsible for 50% of execution time benchmark
- Proposal 1: Speed up FPSQR H/W by 10
- Proposal 2: make all FP instruction run 1.6 times faster
- $Speedup_{FPSQR} = \frac{1}{(1-0.2)+\frac{0.2}{10}} \approx 1.22$
- $Speedup_{FP} = \frac{1}{(1-0.5)+\frac{0.5}{1.6}} \approx 1.23$

The Processor Performance Equation

- CPU time = CPU clock cycles for a program $\times$ clock cycle time
- Number of instructions executed = Instruction count (IC)
- CPI = $\frac{\textit{CPU clock cycles for a program}}{\textit{Instruction count}}$
- Thus, clock cycles = CPI $\times$ IC
- CPU time = $\textit{CPI} \times \textit{IC} \times \textit{clock cycle time}$
- CPU clock cycles = $\sum_{i=1}^{n} IC_i \times CPI_i$
  Where IC_i is the number of times instruction i is executed in a program, CPI_i is the average number of clock cycles for instruction i and the sum gives the total processor clock cycles in a program
- Therefore CPU time = $\textit{Clock cycle time} \times \sum_{i=1}^{n} IC_i \times CPI_i$
-

$$\textit{CPI} = \frac{\sum_{i=1}^{n} IC_i \times CPI_i}{\textit{Instruction count}} = \sum_{i=1}^{n} \frac{IC_i}{\textit{Instruction count}} \times CPI_i$$

Example

- Frequency of FP operations = 25%
- Average CPI of FP operations = 4.0
- Average CPI of other instructions = 1.33
- Frequency of FPSQR = 2%
- CPI of FPSQR = 20
- Proposal 1: Decrease CPI of FPSQR to 2
- Proposal 2: Decrease average CPI of all FP operations to 2.5

Comparing the proposals

- $$CPI_{original} = \sum_{i=1}^{n} \frac{IC_i}{Instruction\ count} \times CPI_i$$
$$= (4 \times 25\%) + (1.33 \times 75\%) = 2.0$$

- $$CPI_{new\ FPSQR} = CPI_{original} - 2\% \times (CPI_{old\ FPSQR} - CPI_{new\ FPSQR\ only})$$
$$= 2.0 - 2\% \times (20 - 2) = 1.64$$

- $CPI_{newFP} = (75\% \times 1.33) + (25\% \times 2.5) = 1.625$
- So the FP enhancement gives marginally better performance

Addressing modes

- MIPS: Register, Immediate, Displacement (Constant offset + Reg content)
- 80x86: Absolute, Base + index + displacement, Base + scaled index + displacement, etc.
- ARM: MIPS addressing, PC-relative, Sum of two registers, autoincrement, autodecrement

Types and sizes of operands

- 80x86, ARM, MIPS
- 8-bit ASCII character
- 16-bit Unicode character or half-word
- 32-bit integer or word
- 64-bit double work or long integer
- IEEE 754 floating point 32-bit (single precision) and 64-bit (double precision)
- 80x86: 80-bit floating point (extended double precision)

Operations

- Data transfer
- Arithmetic and logic
- Control
- Floating point

Control flow

- Conditional jumps
- Unconditional jumps
- Procedure call and return
- PC-relative addressing
- MIPS tests contents of registers
- 8086/ARM test condition flags
- ARM/MIPS put return address in a register
- 8086 call puts return address on stack in memory

Encoding an ISA

- Fixed vs. Variable length instructions
- 80x86 variable, 1 to 18 bytes
- ARM/MIPS fixed, 32 bits
- ARM/MIPS reduced instruction size 16 bits
  - ARM: Thumb
  - MIPS: MIPS16

Computer Architecture

- ISA
- Organisation or Microarchitecture
- Hardware

Five rapidly-changing technologies

1. IC Logic
   - Transistor count on a chip doubles every 18 to 24 months (Moore's Law)
2. Semiconductor DRAM
   - Capacity per DRAM chip doubles every 2-3 years, but this rate is slowing
3. Semiconductor Flash (EEPROM)
   - Standard for personal mobile devices (PMDs)
   - Capacity per chip doubles every 2 years approximately
   - 15-20 times cheaper per bit than DRAM
4. Magnetic disk technology
   - Density doubles every 3 years approximately.
   - 15-20 times cheaper per bit than flash
   - 300-500 times cheaper than DRAM
   - Central to server and warehouse-scale storage
5. Network technology
   - Depends on performance of switches
   - Depends on performance of the transmission system

Technology

- Continuous technology improvement can lead to step-change in effect
- Example: MOS density reached 25K-50K transistors/chip
  - Possible to design single-chip 32-bit microprocessor
  - ...then microprocessors + L1 cache
  - ...then multicores + caches
- Cost and energy savings can occur for a given performance

Energy and Power in a Microprocessor

- For transistors used as switches, dynamic energy dissipated is $Energy_{dynamic} \propto Capacitive\ Load \times Voltage^2$
- The power dissipated in a transistor is $Power_{dynamic} \propto Capacitive\ Load \times Voltage^2 \times Switching\ Frequency$
- Slowing the clock reduces power, not energy
- Reducing voltage decreases energy and power, so voltages have dropped from 5V to under 1V
- Capacitive load is a function of the number of transistors, the transistor and interconnection capacitance and the layout

Example

- 15% reduction in voltage
- Dynamic energy change is

$$\frac{Energy_{new}}{Energy_{old}} = \frac{(Voltage \times 0.85)^2}{Voltage^2}$$
$$= 0.85^2$$
$$= 0.72$$

- Some microprocessors are designed to reduce switching frequency when voltage drops, so

$$Dynamic\ power\ change = \frac{Power_{new}}{Power_{old}}$$
$$= 0.72 \times \frac{frequency\ switched \times 0.85}{frequency\ switched}$$
$$= 0.61$$

Power

- Power consumption increases as processor complexity increases
- Number of transistors increases
- Switching frequency increases
- Early microprocessors consumed about 1W
- 80386 microprocessors consumed about 2W
- 3.3GHz Intel Core i7 consumes about 130W
- Must be dissipated from a chip that is about $1.5cm \times 1.5cm$

Managing power for further expansion

- Voltage cannot be reduced further
- Power per chip cannot be increased because the air cooling limit has been reached
- Therefore, clock frequency growth has slowed
- Heat dissipation is now the major constraint on using transistors

Energy efficiency strategies

1. *Do nothing well*
   - Turn off clock of inactive modules, e.g., FP unit, idle cores to save energy
2. *Dynamic Voltage-Frequency Scaling (DVFS)*
   - Reduce clock frequency and/or voltage when highest performance is not needed.
   - Most μPs now offer a range of operating frequencies and voltages.
3. *Design for typical case*
   - PMDs and laptops are often idle
   - Use low power mode DRAM to save energy
   - Spin disk at lower rate
   - PCs use emergency slowdown if program execution causes overheating

Energy efficiency strategies (continued)

4. *Overclocking*
   - Run at higher clock rate on a few cores until temperature rises
   - 3.3 GHz Core i7 can run in short bursts at 3.6 GHz
5. *Power gating*
   - $Power_{static} \propto Current_{static} \times Voltage$
   - Current flows in transistors even when idle: *leakage current*
   - Leakage ranges from 25% to 50% of total power
   - Power Gating turns off power to inactive modules
6. *Race-to-halt*
   - Processor is only part of system cost
   - Use faster, less energy-efficient processor to allow the rest of the system to halt

Effect of power on performance measures

- Old
  - Performance per mm^2 of Si
- New
  - Performance per Watt
  - Tasks per Joule
- Approaches to parallelism are affected

Cost of an Integrated Circuit

- PMDs rely on systems on a chip (SOC)
- Cost of PMD $\propto$ Cost of IC
- Si manufacture: Wafer, test, chop into die, package, test
- $Cost\ of\ IC = \frac{Cost\ of\ die + Cost\ of\ testing\ die + Cost\ of\ packaging\ and\ final\ test}{Final\ test\ yield}$
- $Cost\ of\ die = \frac{Cost\ of\ wafer}{Dies\ per\ wafer \times Die\ yield}$
- This cost equation is sensitive to die size

Cost of an Integrated Circuit (2)

- $Dies\ per\ wafer = \frac{\pi \times (Wafer\ \ diameter/2)^2}{Die\ \ area} - \frac{\pi \times Wafer\ \ diameter}{\sqrt{2 \times Die\ \ area}}$
  - The first term is the wafer area divided by die area
  - However, the wafer is circular and the die is rectangular
  - So the second term divides the circumference ($2\pi R$) by the diagonal of a square die to give the approximate number of dies along the rim of the wafer
  - Subtracting the partial dies along the rim gives the maximum number of dies per wafer

Die yield

- Fraction of good dies on wafer = die yield
- *Die yield* = $\textit{Wafer yield} \times 1/(1 + \textit{Defects per unit area} \times \textit{Die area})^N$
- This is the Bose-Einstein formula: an empirical model
  - *Wafer yield* accounts for wafers that are completely bad, with no need for testing
  - *Defects per unit area* accounts for random manufacturing defects = 0.016 to 0.057 per cm^2
  - N = *process complexity factor, measures manufacturing difficulty*
    = *11.5 to 15.5 for a 40nm process (in 2010)*

Yield

- Example
  - Find the number of dies per 300mm wafer for a die that is 1.5 cm square.
- Solution

$$Die\ area = 1.5 \times 1.5 = 2.25 cm^2$$

$$Dies\ per\ wafer = \frac{\pi \times (30/2)^2}{2.25} - \frac{\pi \times 30}{\sqrt{2 \times 2.25}}$$

$$= 270$$