Lecture 10: FP, Performance Metrics

- Today's topics:
  - IEEE 754 representations
  - FP arithmetic
  - Evaluating a system
- Reminder: assignment 4 due in a week – start early!

Examples

Final representation: $(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$

- Represent -0.75_{ten} in single and double-precision formats

  Single: (1 + 8 + 23)

  Double: (1 + 11 + 52)

- What decimal number is represented by the following single-precision number?

  1 1000 0001 01000…0000

Examples

Final representation: $(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$

- Represent -0.75_{ten} in single and double-precision formats

  Single: (1 + 8 + 23)
  1 0111 1110 1000…000

  Double: (1 + 11 + 52)
  1 0111 1111 110 1000…000

- What decimal number is represented by the following single-precision number?
  1 1000 0001 01000…0000
  -5.0

FP Addition

- Consider the following decimal example (can maintain only 4 decimal digits and 2 exponent digits)

$9.999 \times 10^{1} + 1.610 \times 10^{-1}$

Convert to the larger exponent:

$9.999 \times 10^{1} + 0.016 \times 10^{1}$

Add

10.015×10^{1}

Normalize

1.0015×10^{2}

Check for overflow/underflow

Round

1.002×10^{2}

Re-normalize

FP Addition

- Consider the following decimal example (can maintain only 4 decimal digits and 2 exponent digits)

$9.999 \times 10^{1} + 1.610 \times 10^{-1}$

Convert to the larger exponent:

$9.999 \times 10^{1} + 0.016 \times 10^{1}$

Add

10.015×10^{1}

Normalize

1.0015×10^{2}

Check for overflow/underflow

Round

1.002×10^{2}

Re-normalize

If we had more fraction bits, these errors would be minimized

FP Multiplication

- Similar steps:
  - Compute exponent (careful!)
  - Multiply significands (set the binary point correctly)
  - Normalize
  - Round (potentially re-normalize)
  - Assign sign

MIPS Instructions

- The usual add.s, add.d, sub, mul, div
- Comparison instructions: c.eq.s, c.neq.s, c.lt.s….
  These comparisons set an internal bit in hardware that is then inspected by branch instructions: bc1t, bc1f
- Separate register file $f0 - $f31 : a double-precision value is stored in (say) $f4-$f5 and is referred to by $f4
- Load/store instructions (lwc1, swc1) must still use integer registers for address computation

Code Example

```
float  f2c (float fahr)
{
    return ((5.0/9.0) * (fahr – 32.0));
}
```

(argument fahr is stored in $f12)

```
 lwc1   $f16, const5($gp)
 lwc1   $f18, const9($gp)
 div.s   $f16, $f16, $f18
 lwc1   $f18, const32($gp)
 sub.s  $f18, $f12, $f18
 mul.s  $f0, $f16, $f18
 jr       $ra
```

Performance Metrics

- Possible measures:
  - response time – time elapsed between start and end of a program
  - throughput – amount of work done in a fixed time
- The two measures are usually linked
  - A faster processor will improve both
  - More processors will likely only improve throughput
- What influences performance?

Execution Time

Consider a system X executing a fixed workload W

$\text{Performance}_X = 1 / \text{Execution time}_X$

Execution time = response time = wall clock time
- Note that this includes time to execute the workload as well as time spent by the operating system co-ordinating various events

The UNIX "time" command breaks up the wall clock time as user and system time

Speedup and Improvement

- System X executes a program in 10 seconds, system Y executes the same program in 15 seconds
- System X is 1.5 times faster than system Y
- The speedup of system X over system Y is 1.5 (the ratio)
- The performance improvement of X over Y is
  1.5 -1 = 0.5 = 50%
- The execution time reduction for the program, compared to Y is (15-10) / 15 = 33%
  The execution time increase, compared to X is
  (15-10) / 10 = 50%

Performance Equation - I

CPU execution time = CPU clock cycles x Clock cycle time
Clock cycle time = 1 / 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?

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

Performance Equation - II

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

Substituting in previous equation,

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

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

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

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

Benchmark Suites

- Measuring performance components is difficult for most users: average CPI requires simulation/hardware counters, instruction count requires profiling tools/hardware counters, OS interference is hard to quantify, etc.

- 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

- The 2006 version 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!

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)

Title

- Bullet