# Review Problem 13

- Write assembly to compute X1 = X0*5 without using a multiply or divide instruction.

```
LSL  X1, X0, #2     // X1 = X0*4
ADD  X1, X1, X0
```

```
ADD X1, X0, X0
ADD X1, X1, X0
ADD X1, X1, X0
ADD X1, X1, X0
```

```
ADD X1, X0, X0     // X1 = 2*X0
ADD X1, X1, X1     // X1 = 4*X0
ADD X1, X1, X0
```

# Computer Performance

Primary goal: execution time (time from program start to program completion)

$$Performance = \frac{1}{ExecutionTime}$$

To compare machines, we say "X is n times faster than Y"

$$n = \frac{Performance_x}{Performance_y} = \frac{ExecutionTime_y}{ExecutionTime_x}$$

Example: Machine *Orange* and *Grape* run a program

Orange takes 5 seconds, Grape takes 10 seconds

$$\frac{Perf_{orange}}{Perf_{grape}} = \frac{Exec_{grape}}{Exec_{orange}} = \frac{10}{5} = 2$$

Orange is 2x times faster than Grape

# Execution Time

Elapsed Time

- counts everything *(disk and memory accesses, I/O , etc.)*
- a useful number, but often not good for comparison purposes

CPU time

- doesn't count I/O or time spent running other programs
- can be broken up into system time, and user time

Example: Unix "time" command

```
linux15.ee.washington.edu> time javac CircuitViewer.java
3.370u 0.570s 0:12.44 31.6%
```

$\underbrace{3.370u}_{\text{User}}\ \underbrace{0.570s}_{\text{System}}\ \underbrace{0{:}12.44}_{\text{elapsed}}\ \underbrace{31.6\%}_{\frac{u+s}{\text{elapsed}}}$

Our focus: user CPU time

- time spent executing the lines of code that are "in" our program

# CPU Time

$MHz = 10^6$ cycles $GHz \Rightarrow 10^9$ cycles

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} * \text{Clock period}$$

$$\text{CPU execution time for a program} = \text{CPU clock cycles for a program} * \frac{1}{\text{Clock rate}}$$

Application example:

A program takes 10 seconds on computer *Orange*, with a 400MHz clock. Our design team is developing a machine *Grape* with a much higher clock rate, but it will require 1.2 times as many clock cycles. If we want to be able to run the program in 6 second, how fast must the clock rate be?

$$10 = Cycles_{orange} * \frac{1}{0.4 \times 10^9} \qquad Cycles_{orange} = 0.4 \times 10^9 \times 10 = 4 \times 10^9 \text{ cycles}$$

$$6 = 1.2 \times (4 \times 10^9) \times \frac{1}{Rate_{grape}} \qquad Rate_{grape} = \frac{1.2 \times 4 \times 10^9}{6} = 0.2 \times 4 \times 10^9$$

$$= 0.8 \times 10^9$$

0.8GHz / 800MHz

# of issued instructions

# CPI

How do the # of instructions in a program relate to the execution time?

$$\text{CPU clock cycles for a program} = \text{Instructions for a program} * \text{Average Clock Cycles per Instruction } \mathbf{(CPI)}$$

$$\text{CPU execution time for a program} = \text{Instructions for a program} * \text{CPI} * \frac{1}{\text{Clock rate}}$$

# CPI Example

Suppose we have two implementations of the same instruction set (ISA).

For some program

Machine A has a clock cycle time of 10 ns. and a CPI of 2.0
Machine B has a clock cycle time of 20 ns. and a CPI of 1.2

What machine is faster for this program, and by how much?

$$\frac{Perf_A}{Perf_B} = \frac{Exec_B}{Exec_A} = \frac{Instr_B \times CPI_B \times Period_B}{Instr_A \times CPI_A \times Period_A} = \frac{1.2 \times 20}{2.0 \times 10} = \frac{24}{20} = 1.2\times$$

# Computing CPI

Different types of instructions can take very different amounts of cycles

- Memory accesses, integer math, floating point, control flow

$$CPI = \sum_{types} \left( Cycles_{type} * Frequency_{type} \right)$$

| Instruction Type | Type Cycles | Type Frequency | Cycles * Freq |
|---|---|---|---|
| ALU | 1 × | 50% = | 0.5 |
| Load | 5 × | 20% = | 1.0 |
| Store | 3 × | 10% = | 0.3 |
| Branch | 2 × | 20% = | 0.4 |
| | | Σ CPI: | 2.2 |

$$\frac{Perf_{new}}{Perf_{old}} = \frac{Exec_{old}}{Exec_{new}} = \frac{\cancel{Inst_{old}} \times CPI_{old} \times \cancel{Period_{old}}}{\cancel{Inst_{new}} \times CPI_{new} \times \cancel{Period_{new}}} = \frac{2.2}{CPI_{new}}$$

# CPI & Processor Tradeoffs

| Instruction Type | Type Cycles | Type Frequency |
|---|---|---|
| ALU | 1 | 50% |
| Load | 5 | 20% |
| Store | 3 | 10% |
| Branch | 2 | 20% |

2.2

How much faster would the machine be if:

1. A data cache reduced the average load time to 2 cycles?

$$2.2/(1\times0.5 + 2\times0.2 + 3\times0.1 + 2\times0.2) = \frac{2.2}{.5+.4+.3+.4} = \frac{2.2}{1.6} = 1.375\times$$

2. Branch prediction shaved a cycle off the branch time?

$$2.2/(1\times0.5 + 5\times0.2 + 3\times0.1 + 1\times0.2) = \frac{2.2}{0.5+1.0+0.3+0.2} = \frac{2.2}{2.0} = 1.1\times$$

3. Two ALU instructions could be executed at once?

$$2.2/(0.5\times0.5 + 5\times0.2 + 3\times0.1 + 2\times0.2) = \frac{2.2}{0.25+1.0+0.3+0.4} = \frac{2.2}{1.95} = 1.13\times$$

# Warning 1: Amdahl's Law

The impact of a performance improvement is limited by what is NOT improved:

$$\text{Execution time after improvement} = \text{Execution time of unaffected} + \text{Execution time affected} * \frac{1}{\text{Amount of improvement}}$$

Example: Assume a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to speed up multiply to make the program run 4 times faster?

$$\frac{100}{4} = 20 + 80 \times \frac{1}{\text{improve}}$$

$$25 = 20 + 80/\text{improve}$$

$$5 = 80/\text{improve} \qquad \text{improve} = \frac{80}{5} = 16\times$$

5 times faster?

$$\frac{100}{5} = 20 + \frac{80}{\text{improve}}$$

$$20 = 20 + \frac{80}{\text{improve}}$$

$$\text{improve} = \infty$$

# Warning 2: BIPs, GHz ≠ Performance

Higher MHz (clock rate) doesn't always mean better CPU

Orange computer: 1000 MHz, CPI: 2.5, 1 billion instruction program

$Exec_{orange} = instr \times CPI \times \frac{1}{Rate} = 1\times10^9 \times 2.5 \times \frac{1}{1\times10^9} = 2.5$ seconds

Grape computer: 500MHz, CPI: 1.1, 1 billion instruction program

$Exec_{grape} = instr \times CPI \times \frac{1}{Rate} = 1\times10^9 \times 1.1 \times \frac{1}{0.5\times10^9} = 2.2$ seconds

Higher MIPs (million instructions per second) doesn't always mean better CPU

1 GHz machine, with two different compilers

Compiler A on program X: 10 Billion ALU, 1 Billion Load

Compiler B on program X: 5 Billion ALU, 1Billion Load

A: $10\times1 + 1\times5 = 15$ sec

B: $5\times1 + 1\times5 = 10$ sec

Execution Time: A 15 sec B 10 sec

A: $\frac{11B}{15sec} =$ B: $\frac{6B}{10sec}$

~~B~~ MIPS: A 733 B 600

| Instruction Type | Type Cycles |
|---|---|
| ALU | 1 |
| Load | 5 |
| Store | 3 |
| Branch | 2 |