Review Problem 8

- We goofed, and wrote: ADD X0, X1, X4, when we meant to write SUB X0, X1, X4. The instruction is at location Mem[0]. What's the simplest program to fix this?

ADD is opcode 0x458
LDUR x7, [x31, #0]
ADDI x8, x31, #1
LSL x9, x8, #30
ORR x10, x9, x7
STUR x10, [x31, #0]

SUB is 0x658
set bit 30 to 1

// x7 = MEM[7..0]
// x8 = 1
// All zeroes, but bit 30 is true
// x10 has correct instruction
// Save it back.
ALU: Arithmetic Logic Unit

Computes arithmetic & logic functions based on controls
Add, subtract
XOR, AND, NAND, OR, NOR
==, <, overflow, ...

Diagram:
- Inputs: A, B (64 bits each)
- Outputs: R (64 bits), Zero, Negative, Overflow
- Function Selector
- Controls
Bit Slice ALU Design

Add, Subtract, AND, OR

A<sub>i</sub>

B<sub>i</sub>

4:1 Mux

S<sub>0</sub>

S<sub>1</sub>

ALU Output Bus
Bit Slice ALU Design (cont.)

Route Carries
Overflow, zero, negative

1-bit Slice

1-bit Slice

1-bit Slice

1-bit Slice

ALU[3:0]

A3 A2 B3
A2 B2
A1 B1
A0 B0

Carry
Overflow
Negative
Zero
Shifter

Support shift operations: \((A \ll 001101)\)

Optional shift by one: \((A \ll b_0)\)

Optional shift by two: \((A \ll b_1)\)
Shifter (cont.)

\[ B = \Sigma b_5 b_4 b_3 b_2 b_1 b_0? \]

A
+64

\[ \ll 1 \quad \ll 32 \]

\[ \ll 2 \quad \ll 16 \]

\[ \ll 4 \quad \ll 8 \]

\[ d \]

\[ \ll 32 \]

\[ e \]
Multiplication

Example

Multiplicand: 0 1 1 0 6
Multiplier: 0 1 0 1 5

4 partial products

+ 0 0 0 0 0

30

Repeat n times:
Compute partial product; shift; add

NOTE: Each bit of partial products is just an AND operation
Parallel Multipliers
Review Problem 12

- How would the ALU be used to help with each of the following branches? The first is filled in for you:
  - B.EQ: SUBS X31, <val1>, <val2>; use zero flag
  - B.NE: ; use $\overline{0}$
  - B.GE: ; use $(\text{negative} \oplus \text{overflow})$
  - B.GT: ; use $(\text{negative} \oplus \text{overflow}) \ast \overline{0}$
  - B.LE: ; use $(\text{negative} \oplus \text{overflow}) + \overline{0}$
  - B.LT: ; use $(\text{negative} \oplus \text{overflow})$
Computer "Performance"

Readings: 1.6-1.8

BIPS (Billion Instructions Per Second) vs. GHz (Giga Cycles Per Second)

Throughput (jobs/seconds) vs. Latency (time to complete a job)

Measuring "best" in a computer

The PowerBook G4 outguns Pentium III-based notebooks by up to 30 percent.*

* Based on Adobe Photoshop tests comparing a 500MHz PowerBook G4 to 850MHz Pentium III-based portable computers
### Performance Example: Homebuilders

<table>
<thead>
<tr>
<th>Builder</th>
<th>Time per House</th>
<th>Houses Per Month</th>
<th>House Options</th>
<th>Dollars Per House</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-build</td>
<td>24 months</td>
<td>1/24</td>
<td>Infinite</td>
<td>$200,000</td>
</tr>
<tr>
<td>Contractor</td>
<td>3 months</td>
<td>1</td>
<td>100</td>
<td>$400,000</td>
</tr>
<tr>
<td>Prefab</td>
<td>6 months</td>
<td>1,000</td>
<td>1</td>
<td>$250,000</td>
</tr>
</tbody>
</table>

Which is the “best” home builder?
- Homeowner on a budget? **Self-build**
- Rebuilding Haiti?
- Moving to wilds of Alaska?

Which is the “speediest” builder?
- Latency: how fast is one house built?
- Throughput: how long will it take to build a large number of houses?
Computer Performance

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

\[
\text{Performance} = \frac{1}{\text{ExecutionTime}}
\]

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

\[
n = \frac{\text{Performance}_x}{\text{Performance}_y} \cdot \frac{\text{ExecutionTime}_y}{\text{ExecutionTime}_x}
\]

Example: Machine Orange and Grape run a program
Orange takes 5 seconds, Grape takes 10 seconds

\[
\frac{\text{Performance}_{\text{orange}}}{\text{Performance}_{\text{grape}}} = \frac{\text{Execution}_{\text{grape}}}{\text{Execution}_{\text{orange}}} = \frac{10}{5} = 2
\]

Orange is \(2\) 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%
```

Our focus: user CPU time
time spent executing the lines of code that are "in" our program
CPU Time

\[
\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \frac{1}{\text{Clock period}}
\]

\[
\text{CPU execution time for a program} = \text{CPU clock cycles for a program} \times \frac{1}{\text{Clock rate}}
\]

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?

\[
10 = \text{CPU clock cycles} \times \frac{1}{0.4 \times 10^9} \quad \text{Clock cycles} = 4 \times 10^9
\]

\[
6 = (1.2 \times 4 \times 10^9) \times \frac{1}{\text{Rate}} \quad \text{Rate} = \frac{1.2 \times 4 \times 10^9}{6} = 0.2 \times 4 \times 10^9 = 0.8 \times 10^9 \quad = 800 \text{ MHz}
\]
CPI

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

CPU clock cycles for a program = Instructions for a program * Average Clock Cycles per Instruction (CPI)

CPU execution time for a program = Instructions for a program * 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?

\[
\frac{\text{Perf}_A}{\text{Perf}_B} = \frac{\text{ExecTime}_B}{\text{ExecTime}_A} = \frac{\text{instr}_B \times \text{CPI}_B \times \text{Period}_B}{\text{instr}_A \times \text{CPI}_A \times \text{Period}_A}
\]

\[
= \frac{1.2 \times 20}{2.0 \times 10} = \frac{2.4}{2.0} = 1.2
\]
Computing CPI

Different types of instructions can take very different amounts of cycles. Memory accesses, integer math, floating point, control flow

\[ CPI = \sum_{\text{types}} \left( \text{Cycles}_{\text{type}} \times \text{Frequency}_{\text{type}} \right) \]

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Type Cycles</th>
<th>Type Frequency</th>
<th>Cycles * Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>1</td>
<td>X</td>
<td>0.5</td>
</tr>
<tr>
<td>Load</td>
<td>5</td>
<td>X</td>
<td>1.0</td>
</tr>
<tr>
<td>Store</td>
<td>3</td>
<td>X</td>
<td>0.3</td>
</tr>
<tr>
<td>Branch</td>
<td>2</td>
<td>X</td>
<td>0.4</td>
</tr>
</tbody>
</table>

CPI: 2.2