Midterm Solutions

CS230 Tutorial 06
Midterm Q2ab - Floating Point

Use the following 10-bit floating point format: 1 sign bit, 4 exponent bits (with a bias of 7) and 5 fraction bits.

a) Express 7.125 in this format.

This is a positive number, so the sign bit is 0
Now, convert this value into binary.

\[
7 = 1 + 2 + 4 = 111 \quad 0.125 = \frac{1}{8} = 001
\]

We have binary 111.001 so we move the point 2 places left: \(1.11001 \times 2^2\)

Now we add in the bias: \(2 + 7 = 9 = 1001\)
And now we can fill in the bits: \(S=0 \quad E=1001 \quad F=11001 \rightarrow 0100111001\)

b) What is the smallest number, larger than 7.125, that is representable in this format? Answer in decimal.

We increase the fraction bits by the smallest value possible, so \(F=11001 \rightarrow F=11010\)

Convert back to binary: \(1.11010 \times 2^2 \rightarrow 111.010\)

And then to decimal: \(111 = 7 \quad 010 = \frac{1}{4} \quad \text{Combining we have: } 7.25\)
Midterm Q2c - Floating Point

Use the following 10-bit floating point format: 1 sign bit, 4 exponent bits (with a bias of 7) and 5 fraction bits.

c) Write 1101000011 in the form \((-1)^s x 1.F x 2^E\)

First we split into S,E,F:

\[ S = 1 \quad E = 1010 \quad F = 00011 \]

Recall the full form is \((-1)^s x 1.F x 2^{E-B}\) so we must apply the bias to E:

\[ E = E - B = 10 - 7 = 3 \]

Now we fill in the formula:

\[ (-1)^1 x 1.00011 x 2^3 \]
Midterm Q3 - Boolean Circuits

- Write an equivalent boolean expression to this circuit. Your expression must contain exactly as many operators as this expression does gates.

- We have three gates, so our expression has three operators. Just connect them along the wires: \((A \text{ XOR } B) \text{ OR } (A \text{ AND } B)\)
Midterm Q3 - Boolean Circuits

● Fill in the truth table for this circuit, your truth table should include the inputs A and B and the output out.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A XOR B</th>
<th>A AND B</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

● This circuit can be simplified to use less gates, provide the simplified circuit diagram. It must use less gates than the diagram above.
  ○ We notice that the “out” column of the the above table exactly matches the result of: A OR B

\[ \text{out} = \text{A OR B} \]
Describe what happens when you call the following MIPS instructions:

1. `sw $8, 12($10)`
   - The value in register 8 is stored at the memory address calculated by adding 12 to the value in register 10.

2. `addi $10, $8, 12`
   - The value in register 8 is added to the immediate value 12 and the sum is put in register 10.
At the end of execution of the following program, the value in $8 is 1, explain why.

1| addi $8, $0, 0
2| lis $10
3| .word -1
4| lis $11
5| .word 4294967295
6| bne $10, $11, end
7| addi $8, $8, 1
8| end:
9| jr $31

- Line 2-3: We set register 10 to -1.
- Line 4-5: We set register 11 to 4294967295
  - This is exactly $2^{32} - 1$.
  - $2^{32} - 1$ has the same representation as -1 in two’s complement.
- Line 6: Therefore $10 == 11$ on line 6
  - So the branch is *not* taken.
- Line 7: adds one to $8$.
- Line 9: ends the program.
- The program ends with 1 in register 8.
- Line 1 does not matter.
  - The program must end with 1 in register 8.
  - Regardless of line 1 the program must begin with 0 in register 8 so line 7 sets it to 1.
Midterm Q7 - Calculating Execution Time

How long does execution take (CPU time) for a program with 150 instructions when run on a computer with a CPI of 2.0 and a clock rate of 2GHz? Recall: 1GHz = 10^9 Hz.

**Solution:**

Remember the formula for CPU time:

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

We know: instruction count = 150 (given)
- cycles per instruction = 2.0 (given)
- seconds/clock cycle = 1/clock rate = 1/(2 * 10^9)

So, CPU time = 150 * 2 * (1 / (2 * 10^9)) = 1.5 * 10^{-7} seconds
Midterm Q8 - MIPS pipeline

Consider the following execution diagram for a set of MIPS instructions with no pipelining:

<table>
<thead>
<tr>
<th>Cycle</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>sub $8, $3, $2</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add $3, $6, $7</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add $9, $3, $2</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a) Provide the execution diagram of this same set of instructions fully pipelined, without any forwarding.

**Solution:** First, identify any hazards. There are no conflicts between instr1 and instr2. But instr3 uses the value in $3, which is computed in instr2. So, there is a data hazard, and since there is no forwarding, ID of instr3 is delayed until the WB stage of instr2.
Solution: First, identify any hazards. There are no conflicts between instr1 and 2. But instr3 uses the value in $3, which is computed in instr2. So, there is a data hazard, and since there is no forwarding, ID of instr3 is delayed until the WB stage of instr2.

Cycle | 1  2  3  4  5  6  7  8  9
------|---------------------------
sub $8, $3, $2 | IF  ID  EX  MEM  WB
add $3, $6, $7 | IF  ID  EX  MEM  WB
add $9, $3, $2 | IF  -----  -----  ID  EX  MEM  WB
Data hazards are a common problem for pipelines. Explain what a data hazard is, and give an example of one.

**Solution:**
A data hazard is when a value is needed for the execution of an instruction but it is not ready yet. One example is the code from part a of this problem:

```
add $3, $6, $7
add $9, $3, $2
```

Here the second instruction relies on the value computed in the first instruction.

There are also many other examples of code with data hazards!
Assignment reminders

- Always test your code on the university servers before you hand it in
- Hand in the .asm file (not the .mips file)
That's all Folks!