**Problem No. 1 (8 points)**

For each of the following statements, indicate whether the statement is **TRUE** or **FALSE**. Each correct answer carries 2 points.

(a) Profit margins for low-end consumer devices (such as smartphones) are higher than the profit margins for more specialized products (such as HPC, servers). **FALSE**

(b) Reducing the clock frequency is an effective way to reduce energy consumption. **FALSE**

(c) Code size for a program written with the RISC ISA is usually larger than when written with the CISC ISA. **TRUE**

(d) The CALLS instruction in the VAX ISA is an example of “make the common case fast”. **FALSE**

**Problem No. 2 (9 points)**

(a) Consider two different enhancements to the design of a microprocessor. The first enhancement (E1) speeds up 5% of the program execution time by 10x. The second enhancement (E2) speeds up 10% of the program execution time by 5x. Which optimization provides higher performance improvement? Mark the correct answer.
   - i. E1 provides more improvement than E2
   - ii. **E2 provides more improvement than E1**
   - iii. Both E1 and E2 provide similar improvements
   - iv. Problem statement does not have enough data to conclude which option is better

(b) The static power consumed by a microprocessor can be reduced by:
   - i. Using the *Turbo* mode
   - ii. Using a low power DRAM state
   - iii. **Turning off the power supply to inactive modules**

(c) Write down the equivalent MIPS instruction sequence for the following instruction: LOAD R7, (R2 + R3)

**Solution:**

ADD R4, R2, R3  
LOAD R7, 0(R4)
Problem No. 3 (14 points)

Intel's Sandy Bridge die, implemented in a 32nm process is 20.7mm x 10.5mm. Assume 300mm wafers, a defect density of 0.03/cm², a process complexity factor of 12.5 and a wafer yield of 100%.

(a) (6 points) How many dies per wafer can Intel expect for Sandy Bridge on 32nm?

Solution:
Number of dies per wafer is given by:

$$Dies \ per \ wafer = \pi \times \left(\frac{Wafer \ diameter}{2}\right)^2 \div \frac{Die \ Area}{\sqrt{2} \times Die \ area}$$

Wafer diameter = 300mm, Die area = 20.7 mm * 10.5 mm = 217.35 mm²
Therefore:

$$Dies \ per \ wafer = \pi \times \left(\frac{300}{2}\right)^2 \div \frac{300}{\sqrt{2} \times 217.35} = 280$$

(b) (4 points) What percentage of the wafer area is wasted due to the difference in wafer and die geometries (round wafers as opposed to rectangular dies)?

Solution:
Total area of a wafer = π * \left(\frac{300}{2}\right)^2 = 70685.83 mm²
Area used up by dies = Number of dies * die area = 280 * 217.35 = 60858 mm²
Percentage of wafer area wasted = (70685.83 – 60858) / 70685.83 = 13.9%

(c) (4 points) How many dies per wafer does Intel have to discard due to defects?

Solution

$$Die \ yield = Wafer \ yield \times \frac{1}{(1 + Defects \ per \ unit \ area \times \ Die \ area)^N}$$

Wafer yield = 100%, Defects per unit area = 0.03/cm² = 0.0003/mm²
Die area = 217.35 mm², N = 12.5
Therefore:

$$Die \ yield = 100\% \times \frac{1}{(1+0.0003\times217.35)^{12.5}} = 45.4\%$$

Number of good dies per wafer = 280 * 0.454 = 127
Number of dies discarded per wafer = 280 – 127 = 153
Problem No. 4 (13 points)

Consider a processor with the following CPI for each instruction type and a typical instruction mix as shown:

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Clock Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loads/Stores</td>
<td>2</td>
</tr>
<tr>
<td>Branches</td>
<td>3</td>
</tr>
<tr>
<td>Integer ALU (including shifts)</td>
<td>1</td>
</tr>
<tr>
<td>Integer Multiply</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loads/Stores</td>
<td>40%</td>
</tr>
<tr>
<td>Branches</td>
<td>10%</td>
</tr>
<tr>
<td>Shifts</td>
<td>4%</td>
</tr>
<tr>
<td>Multiples</td>
<td>6%</td>
</tr>
<tr>
<td>Other Integer Arithmetic</td>
<td>40%</td>
</tr>
</tbody>
</table>

The compiler writer is considering using strength-reduction optimization to convert multiplies into a series of adds and shifts. Assume that 60% of multiply instructions occurring in the source program can be converted to shift-add sequences with an average length of three instructions.

(a) **5 points** Compute the percentage change in number of instructions, after the compiler applies the strength reduction transformation.

**Solution**

Whenever strength reduction transforms a multiply into a shift-add sequence, it removes 1 instruction from the program and adds 3 instructions, causing a net increase of two instructions. Therefore:

\[
\text{Change in number of instructions} = (\text{Original Frequency of Multiply Instructions}) \times (\text{Percentage of multiply instructions removed by strength reduction}) \times 2
\]

\[
= 6\% \times 60\% \times 2 = 0.06 \times 0.6 \times 2 = 0.072
\]

Therefore, strength reduction increases the number of instructions by 7.2%.

(b) **8 points** Does strength reduction optimization improve performance? If yes, by how much?

**Solution**

Before strength reduction:

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Clock Cycles</th>
<th>Frequency</th>
<th>Effective CPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loads/Stores</td>
<td>2</td>
<td>40%</td>
<td>2*0.4 = 0.8</td>
</tr>
<tr>
<td>Branches</td>
<td>3</td>
<td>10%</td>
<td>3*0.1 = 0.3</td>
</tr>
<tr>
<td>Integer ALU + Shifts</td>
<td>1</td>
<td>44%</td>
<td>1*0.44 = 0.44</td>
</tr>
<tr>
<td>Integer Multiply</td>
<td>8</td>
<td>6%</td>
<td>8*0.06 = 0.48</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td></td>
<td><strong>2.02</strong></td>
</tr>
</tbody>
</table>

After strength reduction:

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Cycles</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loads/Stores</td>
<td>2</td>
<td>40/107.2 = 37.3%</td>
</tr>
<tr>
<td>Branches</td>
<td>3</td>
<td>10/107.2 = 9.3%</td>
</tr>
<tr>
<td>Integer ALU + Shifts</td>
<td>1</td>
<td>(44+(0.06)(0.6)(3))/107.2 = 51.1%</td>
</tr>
<tr>
<td>Integer Multiply</td>
<td>8</td>
<td>(6 – (6)(0.6))/107.2 = 2.2%</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{Speedup} = \frac{(\text{Instruction count} \times \text{Average CPI} \times \text{Clock period})_{\text{before strength reduction}}}{(\text{Instruction count} \times \text{Average CPI} \times \text{Clock period})_{\text{after strength reduction}}} = \frac{1+2.02}{1.072+1.712} = 1.098
\]

\[
\text{%age Improvement in Performance} = 1 - \frac{1}{\text{Speedup}} = 1 - \frac{1}{1.098} = 8.9\%
\]

Therefore, strength reduction improves performance by 8.9%.
Problem No. 5 (12 points)

Consider the following MIPS code fragment:

```
ADDI R1, R0, #1
ADD R2, R0, R0
ADDI R3, R0, #128
```

```
loop:
MUL R1, R1, #2
ADDI R2, R2, #1
BNE R2, R3, loop
```

(a) (6 points) How many MIPS instructions are executed dynamically to complete the above code fragment?

**Solution**

This code fragment has 3 instructions before the loop and 3 instructions within the loop. The loop counter (R2) is initialized to 0, gets incremented by 1 every iteration and then compared to 128 (R3). The loop terminates, when the loop counter reaches a value of 128. Therefore the loop has 128 iterations.

Total number of dynamic instructions = 3 + (128 * 3) = 387

(b) (6 points) After completion, what are the final contents of registers R1 and R2?

**Solution**

Final contents of R2 = 128
R1 is initialized to 1, gets multiplied by 2 every iteration. Since there are 128 iterations:

Final contents of R1 = $1 \times 2^{128}$

**Note:** Since each register has 32 bits, R1 will overflow in 32nd iteration. Therefore the final contents of R1 may be undefined (e.g., zero)
Problem No. 6 (14 points)

(a) (6 points) Consider the following sequence of instructions being processed on the pipelined 5-stage RISC processor discussed in class:

Add R4, R2, R3
Store R2, #100(R6)
Load R5, #200(R4)
Subtract R5, R8, R6

Identify all the data dependencies in the above instruction sequence. For each dependency, indicate the dependency type (RAW, WAR or WAW), the two instructions and the register that causes the dependency.

Solution:
1. RAW hazard between Add and Load instructions for register R4
2. WAW hazard between Load and Subtract instructions for register R5

(b) (8 points) Consider a non-pipelined processor using the 5-stage datapath with 2 GHz clock speed. Assume that due to clock skew and pipeline registers, pipelining the processor lengthens the clock cycle period by 10%. Also, assume that the processor uses a unified single-ported cache for data and instruction accesses, resulting in a structural hazard between IF and MEM stages. Suppose that data references represent 30% of the instructions executed and that the ideal CPI of the pipelined processor, ignoring the structural hazard is 1. How much speedup can we gain from pipelining? Assume a balanced pipeline and ignore the pipeline fill and drain overheads.

Solution:

Speedup from pipelining = \( \frac{(Average \: time \: per \: instruction)_{\text{non-pipelined}}}{(Average \: time \: per \: instruction)_{\text{pipelined}}} \)

Without pipelining:
Clock period = 1 / 2 GHz = 0.5 ns
CPI = 5

With pipelining:
Clock period is lengthened by 10%. Therefore:
Clock period = 0.5 + (10% * 0.5) = 0.55 ns
30% of instructions access memory during the MEM stage. Each of these instructions results in a structural hazard with a stall penalty of 1 cycle. Therefore:
CPI = Ideal CPI + Stall CPI = 1 + (0.3)(1) = 1.3

Speedup from pipelining = (0.5 ns * 5) / (0.55 ns * 1.3) = 3.5