Solution

1. (a) Amdahl’s Law, \( f = 25\% \), \( s = 1.8 \)

\[
\text{speedup} = \frac{1}{1 - f + \frac{f}{s}} = \frac{1}{1 - 0.25 + \frac{0.25}{1.8}} \approx 1.12
\]

(b) Amdahl’s Law, \( f_1 = 25\% \), \( s_1 = 1.8 \), \( f_2 = 15\% \), \( s_2 = 0.83 \)

\[
\text{speedup} = \frac{1}{1 - f_1 - f_2 + \frac{f_1}{s_1} + \frac{f_2}{s_2}} = \frac{1}{1 - 0.25 - 0.15 + \frac{0.25}{1.8} + \frac{0.15}{0.83}} \approx 1.09
\]

(c) \( t_{fp} = \frac{25}{1.8} = 0.14 \)

\( \text{t}_{dc} = \frac{15}{83} = 0.18 \)

\( \text{t}_{total} = 1 - 0.25 - 0.15 + 0.14 + 0.18 = 0.92 \)

% time in floating point = \( \frac{t_{fp}}{t_{total}} = \frac{0.14}{0.92} = 15.2\% \)

% time in data cache = \( \frac{t_{dc}}{t_{total}} = \frac{0.18}{0.92} = 19.5\% \)

2. (a) \( CPI_{M1} = 1 \times 0.6 + 2 \times 0.15 + 4 \times 0.25 = 1.9 \)

\( CPI_{M2} = 2 \times 0.6 + 3 \times 0.15 + 4 \times 0.25 = 2.65 \)

(b) \( MIPS_{M1} = \frac{CLK_{M1}}{CPI_{M1}} \times 10^{-6} = \frac{10^9}{1.9} \times 10^{-6} = 526 \)

\( MIPS_{M2} = \frac{CLK_{M2}}{CPI_{M2}} \times 10^{-6} = \frac{2 \times 10^9}{2.65} \times 10^{-6} = 754 \)

(c) M1 has a smaller MIPS rating. To reduce the average number of cycles of instruction B or C on Machine M1:

i. B in 1 cycle: \( MIPS_{M1}' = \frac{10^9}{1 \times 0.6 + 1 \times 0.15 + 4 \times 0.25} \times 10^{-6} = 571 \)

ii. C in 3 cycles: \( MIPS_{M1}' = \frac{10^9}{1 \times 0.6 + 2 \times 0.15 + 3 \times 0.25} \times 10^{-6} = 606 \)

iii. C in 2 cycles: \( MIPS_{M1}' = \frac{10^9}{1 \times 0.6 + 2 \times 0.15 + 2 \times 0.25} \times 10^{-6} = 714 \)

iv. C in 1 cycle: \( MIPS_{M1}' = \frac{10^9}{1 \times 0.6 + 2 \times 0.15 + 1 \times 0.25} \times 10^{-6} = 869 \)

Changing the CPI of B from 2 to 1, or the CPI of C from 4 to 3, 2 cannot achieve a higher MIPS rating for M1 than M2. Only changing the CPI of C from 4 to 1 can achieve a higher MIPS rating for M1 than M2.

3. Loads-stores: \( \frac{25.1\% + 13.2\% + 30.3\% + 4.3\%}{2} = 36.45\% \)

Conditional Branches: \( \frac{12.1\% + 17.5\%}{2} = 14.8\% \)

Jumps, calls, returns: \( \frac{0.7\% + 0.6\% + 0.6\% + 0.7\% + 3.2\% + 3.2\%}{2} = 4.5\% \)

ALU instructions: \( 1 - 36.45\% - 14.8\% - 4.5\% = 44.25\% \)

CPI: \( 1.0 \times 44.25\% + 2 \times 36.45\% + (3.0 \times 60\% + 1.5 \times 40\%) \times 14.8\% + 1.3 \times 4.5\% \approx 1.59 \)

4. The length of the instructions are 12 bits. There are 32 registers so the length of the register field will be 5 bits for each register operand. We use \( \text{addr}[11] \) to \( \text{addr}[0] \) to represent the 12 bits of the address as shown in the tables below.
(a) In the first case, we need to support 3 two-address instructions. These can be encoded as follows:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3 two-addr inst.</td>
<td>'00' - '10'</td>
<td>'00000' - '11111'</td>
</tr>
<tr>
<td>other inst.</td>
<td>'11'</td>
<td>'00000' - '11111'</td>
</tr>
</tbody>
</table>

Hence, for the one-address and two-address instructions must be mapped to the remaining 10 bits with the upper two bits encoded as '11'. The one-address instructions are then encoded with the addr[9:5] field using '00000' - '11101' for the 30 instruction types, leaving the addr[4:0] field for the register operand. This leaves the patterns with '11' followed by '11111' in the upper seven bits and '00000' - '11111' in the lower five bits to encode 32 of the zero-address instructions. The remaining zero-address instructions can be encoded using '11' followed by '11111' in the upper seven bits and '00000' to '10100' in the lower five bits to encode the other 20 zero-address instructions.


| 3 two-addr inst. | '00' - '10' | '00000' - '11111' | '00000' - '11111' |
| 30 one-addr inst. | '11' | '00000' - '11111' | '00000' - '11111' |
| 52 zero-addr inst. | '11' | '11110' | '00000' - '11111' |
| x zero-addr inst. | '11' | '11111' | '00000' - '10100' |

Hence, it is possible to have the above instruction encodings.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2 two-addr inst.</td>
<td>'00' - '01'</td>
<td>'00000' - '11111'</td>
</tr>
<tr>
<td>32 one-addr inst.</td>
<td>'10'</td>
<td>'00000' - '11111'</td>
</tr>
<tr>
<td>29 one-addr inst.</td>
<td>'11'</td>
<td>'00000' - '11111'</td>
</tr>
<tr>
<td>x zero-addr inst.</td>
<td>'11'</td>
<td>'00000' - '11111'</td>
</tr>
</tbody>
</table>

Because x includes 96 patterns (there are 32 possible values for addr[9:5], and 3 non-overlapping values in addr[4:0]), we can support up to 96 zero-address instructions.

5. Assuming there are s million instructions in a program:

(a) \[
T = \frac{\alpha s}{n x} + \frac{(1-\alpha)s}{x} \\
\text{MIPS: } \frac{s}{T} = \frac{nx}{\alpha + (1-\alpha) \cdot n}
\]

(b) \[
T_{\text{new}}: \frac{\alpha s + \beta s}{n x} + \frac{(1-\alpha)s}{x} \\
\text{Speedup: } \frac{T}{T_{\text{new}}} = \frac{\alpha s}{\alpha s + \beta s} + \frac{(1-\alpha)s}{x} = \frac{\alpha + (1-\alpha) n}{\alpha + (1-\alpha) n + \beta}
\]

Note: changing \( \beta \) to \((\beta + 1)^{n-1}\) or \((n - 1) \cdot \beta\) is also okay.