would not be able to use the new instruction form in this case. In general, the values will not be equal, and a computer

The result written to R1 in this case is \( Y + Y \), however, the register-memory form of the code sequence is

\[
\text{ADD R1, R1, R0}
\]

\[
\text{LOAD R1, R0}
\]

Consider the code sequence be replaced when performance of the design is better. The loads must be replaced with the new load operation. For the old and new performance to be the same. It must be

\[
(1 - \text{CPI of } \text{old instruction}) \times \frac{\text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

For the modified machine,

\[
(1 - \text{CPI of } \text{mod} \text{if instruction}) \times \frac{\text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

For the original machine,

\[
\text{new CPI of } \text{IC} \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
\text{IC \times new CPI} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]

\[
(1 - \text{CPI of } \text{IC}) \times \frac{\text{CPI of } \text{old instruction}}{\text{CPI of } \text{old instruction}} = \frac{\text{CPI of } \text{old instruction} \times \text{CPI of } \text{new instruction}}{\text{CPI of } \text{old instruction}}
\]
Instructions in the sim of the frequencies of load and store for a local of 37.6%. The frequency of load/store data to move to the destination register(s) itself, and, in a local and other locals for a local of 48.7%. The frequency of load/store with the destination register in the instruction itself, and more (implemented as an OR instruction between a condition register and the register with the destination register) are exhibited in the instruction's mix. The instruction's frequency is the sum of the frequencies of the add, sub, and...

With the above categories defined, the frequency of ALU instructions is the sum of the frequencies of the add, sub, and...

To complete the average CPI we need to aggregate the instruction frequencies in Figure 12.1 to match these categories. This is done in the above table is the average from 68% and 70%. The exercise statement gives CPI information in terms of load/store instruction categories with two suggestions for conditional branches.

<table>
<thead>
<tr>
<th>other</th>
<th>load</th>
<th>store</th>
<th>compare</th>
<th>shift</th>
<th>ret</th>
<th>ult</th>
<th>eq</th>
<th>jmp</th>
<th>int</th>
<th>pred</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.8</td>
<td>0.2</td>
<td>0.3</td>
<td>0.3</td>
<td>0.1</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Where 0.15 rounds to 0.2 and 0.27 rounds to 0.3. For 99% and 98% the average instruction frequencies are shown below.

When rounds to 0.2. For a summation of terms round to even will not substitute an even or odd, but rounds to 0.2. If 0.25 rounds to 0.2, which makes the least significant fraction even. For example, 0.175 rounds to 0.2. However, to use in our next example, suppose the frequency of the load classic instructions is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format. However, in our next example, suppose the frequency of the load is high as close to 100 as possible. One such scenario is to use an indexed format.
the negative of the instruction that caused the jump. This is done to ensure that the negative instruction is executed before the jump. Finally, the result of the jump is recorded in the instruction register.

(a) The value of \( P \) after executing the second instruction is:

\[
P = (2\cdot1) (0.01) + ((0.02) (0.06 - 1)) + (0.07) (1.03) + (0.04) (1.04) + (0.09) (1.04) + (0.04) (0.36) + (0.06) (0.03)
\]

(b) The clock cycles for the file transfer are:

\[
\text{Clock Cycles} = \sum \text{Instructions} \times \text{Clock Cycles per Instruction}
\]

(c) Modify the above equations to account for clock cycles for other operations.
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Clock Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD R1 (R2)</td>
<td>1</td>
</tr>
<tr>
<td>ADD R1,R2</td>
<td>2</td>
</tr>
<tr>
<td>MOV R1,R2</td>
<td>3</td>
</tr>
<tr>
<td>SUB R1,R2</td>
<td>4</td>
</tr>
</tbody>
</table>

- Since the PC is incremented after each cycle, the next PC in the memory following PENDZ is the next instruction to be fetched.
- If the PENDZ instruction is fetched during the clock cycle, it is not fetched until the next cycle. However, if the PENDZ instruction is fetched during the clock cycle, it is not fetched until the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle.

**Explanation:**

The PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruction is fetched after the current instruction. Depending on the implementation, the PENDZ instruction may be fetched during the current cycle or the next cycle. This is because the PENDZ instruc...
3. How would you determine the effectiveness of a new product in the market?

(a) Correlation analysis: Do the revenues grow when the price increases?

(b) Comparative analysis: Is the price of the new product comparable to its competitors?

3. What is the significance of the CPX in determining the profitability of a company?

4. How would you calculate the effective CPI for a processor with the following instruction mix for an application to be executed on the processor:

- Jumps: 7.2 cycles
- Conditional branches: 2.2 cycle
- Load/Store: 1.5 cycles
- ALU and Load: 1.2 cycles

Assume that a processor X-GUL has the following average CPI for instructions:

CPI = 0.2 * 1 + 0.2 * 1.5 + 0.3 * 1.7 + 0.1 * 1.1 = 1.42

The CPI for the processor is 1.42.

5. What is the role of the CPI in evaluating the performance of a computer system?

6. How would you determine the performance of a computer system for a given application?

7. What is the significance of the CPI in comparing the performance of different computer systems?
8: the XOR control dependent on BNEZ

7: the second DADD control dependent on BNEZ

6: the second DADD is data dependent on DISP due to Rs

5: DNEZ is data dependent on DADD due to R7

4: the XOR instruction's name/mask-dependent on LD due to Register R2

3: OR data dependent on the LD instruction due to Store/Location R1

2: DSB data dependent on the LD instruction due to Store/Location R1

1: the Instruction DADD is data dependent on the LD instruction due to Store/Location R1

(Question 3.2 Text (a)

The performance due to a hardware design option can be affected both positively and negatively by compiler actions.

Issue opportunity: The available ILP is affected by the compiler technology. Compiler optimization may reduce the available ILP.

Multiple opportunity: The available ILP is subject to compiler technology. Compiler optimization may reduce the available ILP, yet this was the only multiple of calculation. Statement 4 produces a value that does not contribute to the live ranges and is hence not live. Thus this statement may be deleted.

Only the code segment so that are not dominated by other code segments, therefore are not code segment, which are not dedicated to T, and therefore, T are not code segment, whereas A and F are code segment.

The group of statements that can be issued together are

\{7\} \{6\} \{5\} \{4\} \{3\} \{2\} \{1\} \{0\}
ADD.D F0',F2 D2 (R2)
ADD.D F2',F2 D2 (R2)

ADD.D F0',P0',P2
ADD.D F0',P2,0 (R2)
ADD.D F0',P4,0 (R2)

1 2 3 from T.D P0',P0',P2
1 2 3 from T.D P0',P0',P2
1 2 3 from T.D P0',P0',P2

Loop:

**Problem D2:** What if the instruction must follow the specified load by more than one clock cycle?

- The Gantt Loop takes multiple layers of execution and must be updated based on recognition of instructions and determination of the instruction that needs to be updated.

The Gantt Loop takes multiple layers of execution and must be updated based on recognition of instructions and determination of the instruction that needs to be updated.

Schedule that does not violate data dependencies.

two instructions with a name-and-dependence can never replace the MIPS pipeline to shift to preserve correct access order. Thus:

In the MIPS pipeline, register values are read early, at stage ID, before register values are written into the WID stage. Thus:

**Problem 2:** The register file in the MIPS pipeline is not a hazard.

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Would not be a hazard.

The MIPS stage and not available to the ID stage is the DSB instruction. With forwarding, this dependence would not be a hazard.

Would not be a hazard.

The register file in the MIPS pipeline is not a hazard.

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Would not be a hazard.

The MIPS stage and not available to the ID stage is the DSB instruction. With forwarding, this dependence would not be a hazard.

Would not be a hazard.

The register file in the MIPS pipeline is not a hazard.

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Would not be a hazard.

The MIPS stage and not available to the ID stage is the DSB instruction. With forwarding, this dependence would not be a hazard.

Would not be a hazard.

The register file in the MIPS pipeline is not a hazard.

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Would not be a hazard.

The MIPS stage and not available to the ID stage is the DSB instruction. With forwarding, this dependence would not be a hazard.

Would not be a hazard.

The register file in the MIPS pipeline is not a hazard.

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Would not be a hazard.

The MIPS stage and not available to the ID stage is the DSB instruction. With forwarding, this dependence would not be a hazard.

Would not be a hazard.

The register file in the MIPS pipeline is not a hazard.

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Would not be a hazard.

The MIPS stage and not available to the ID stage is the DSB instruction. With forwarding, this dependence would not be a hazard.

Would not be a hazard.

The register file in the MIPS pipeline is not a hazard.

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Before the ID instruction completes its memory access in the WID stage, the previous calculation is complete, and the ID stage needs the new value of R1 for the DD instruction. 

Would not be a hazard.

The MIPS stage and not available to the ID stage is the DSB instruction. With forwarding, this dependence would not be a hazard.
Load has not been speculated, so if there is speculation check this no.

Load speculations: check this no.

Use of the speculation check is not possible in this case because the position of the instruction is first instruction in the basic block. Use of the speculation check instruction allows the load to be moved above the branches. Because the load may have long latency, it should be moved as early in the program as possible. In this case, the position of the instruction is first instruction in the basic block. Use of the speculation check instruction allows the load to be moved above the branches. Because the load may have long latency, it should be moved as early in the program as possible.

The speculation check instruction detects the performance response to a memory access fault, which occurs if the processor loads a data element that is not present in the memory. Normally, the processor cannot speculative the load before the branch because of potential complications.

For a memory access fault, the processor automatically loads the load before the branch because of potential complications.

Question 4.32(a): The branch in the given code checks if the conditions are full ported. The code then loads P2 by determining the ported and then uses the loaded value. Normally, the compiler cannot speculative the load before the branch because of potential complications.

Question 4.33(a): The branch in the given code checks if the conditions are full ported. The code then loads P2 by determining the ported and then uses the loaded value. Normally, the compiler cannot speculative the load before the branch because of potential complications.

Question 4.34(a): The predicted code becomes:

Compared to the original code, the conditions are true when there is no duplication values in the elements of $x[i]$. There is no need to make checks from arrays $a[i]$ and $b[i]$. Thus, these can be no loop control dependence. The elements of $a[i]$ are only used to make checks from arrays $a[i]$ and $b[i]$. Thus, there can be no loop control dependence. The elements of $a[i]$ provide a sequence of values used to make checks from arrays $a[i]$ and $b[i]$. Thus, these can be no loop control dependence. The elements of $a[i]$ provide a sequence of values used to make checks from arrays $a[i]$ and $b[i]$. Thus, these can be no loop control dependence.

Question 4.15: Analyse the loop we observe that the loop kernel from $i=1$ to $n$ in the index array $x[i]$ provides a sequence of values used to make checks from arrays $a[i]$ and $b[i]$. Thus, there can be no loop control dependence. The elements of $a[i]$ provide a sequence of values used to make checks from arrays $a[i]$ and $b[i]$. Thus, these can be no loop control dependence. The elements of $a[i]$ provide a sequence of values used to make checks from arrays $a[i]$ and $b[i]$. Thus, these can be no loop control dependence.

This unrolled code has 13 instructions and takes 13 cycles to produce two results. For an average of 6.5 cycles per iteration.

Weekend delay:

Step 1: and this branch delay:

S.D +8(R2),Rs
ADD D.Ps,Ps,Ps

Step 2: from ADD D.Ps,Ps,Ps

S.D +16(R2),Po
ADD D.R2,P2,

Step 3: from ADD D.R2,P2

S.D +8(R2),Rs
ADD D.Ps,Ps,Ps

Step 4: from ADD D.Ps,Ps,Ps

S.D +8(R2),Rs
ADD D.Ps,Ps,Ps

Step 5: from ADD D.Ps,Ps,Ps

S.D +8(R2),Rs
ADD D.Ps,Ps,Ps

Step 6: from ADD D.Ps,Ps,Ps

S.D +8(R2),Rs
ADD D.Ps,Ps,Ps

Step 7: from ADD D.Ps,Ps,Ps

S.D +8(R2),Rs
ADD D.Ps,Ps,Ps

Step 8: from ADD D.Ps,Ps,Ps

S.D +8(R2),Rs
ADD D.Ps,Ps,Ps

Step 9: from ADD D.Ps,Ps,Ps

S.D +8(R2),Rs
ADD D.Ps,Ps,Ps

Step 10: from ADD D.Ps,Ps,Ps

S.D +8(R2),Rs
ADD D.Ps,Ps,Ps

Step 11: from ADD D.Ps,Ps,Ps

S.D +8(R2),Rs
ADD D.Ps,Ps,Ps

Step 12: from ADD D.Ps,Ps,Ps

S.D +8(R2),Rs
ADD D.Ps,Ps,Ps

Step 13: from ADD D.Ps,Ps,Ps

S.D +8(R2),Rs
ADD D.Ps,Ps,Ps
null: ....

SL.D F2,0(R1)

; speculative load

; check for exception deferred by SL.D on F2

null: ....

SPECK F2

ADD D F4,F0,F2

null: ....

BNEQZ R1,null