Note: Solution is given in RED.
Q. No. 1 (15 Marks)

Answer True or False by circling one. (RED Correct.)

1. WAR hazards occur because of out-of-order execution
   T / F

2. Name dependencies are false dependencies
   T / F

3. In scoreboard two instructions can execute at the same time
   T / F

4. Scoreboard can issue instructions out of order.
   T / F

5. In cancel branching, instruction in delay slot is always cancelled
   T / F

6. Modern processors can eliminate RAW hazard.
   T / F

7. In Scoreboarding WAW hazards are checked in the Write Result stage
   T / F

8. Scoreboard was first introduced in early 80’s
   T / F

9. Register renaming can fully eliminate WAW hazard.
   T / F

10. SD R5, 34(R4) changes either the value of R4 or R5.
    T / F

11. WAW is detected by scoreboard using register result status.
    T / F

12. MIPS is a register-register architecture.
    T / F

13. Early Intel architectures were Register-Memory architectures.
    T / F

14. MIPS does not use flags for branch decision.
    T / F

15. No exception can occur during instruction fetch (IF) cycle.
    T / F.
Q.No.2 (12)
Suppose we have two processors P1 and P2. Processor P1 adopts Predict Not Taken policy and has a penalty of 2 stalls for branches that are not taken and 4 for branches that are taken. Processor P2 incorporates Predict Taken and has a penalty of 3 for branches that are taken and 4 for branches that are not taken. Now suppose we have two benchmarks B1 and B2 both having 20% branch instructions. Benchmark B1, however has 30% branches that are taken and benchmark B2 has 10% branches that are not taken. Compare the performance of both processors on both benchmarks due to control hazards only, assuming all other conditions remain the same.

Let's assume INB = Non-Branch Instructions, TB = Taken Branch Instructions, NTB = Not Taken Branch Instructions

**Benchmark 1**
Processor 1: (INB CPI = 1, Taken branch penalty = 5, Not Taken branch penalty = 3)
\[
\text{CPI} = \text{INB} \times 1 + \text{TB} \times 5 + \text{NTB} \times 3 = (0.8 \times 1) + (0.2 \times 0.3 \times 5) + (0.2 \times 0.7 \times 3) = 1.52
\]
Processor 2: (INB CPI = 1, Taken branch penalty = 4, Not Taken branch penalty = 5)
\[
\text{CPI} = \text{INB} \times 1 + \text{TB} \times 4 + \text{NTB} \times 5 = (0.8 \times 1) + (0.2 \times 0.3 \times 4) + (0.2 \times 0.7 \times 5) = 1.74
\]

**Benchmark 2**
Processor 1:
\[
\text{CPI} = \text{INB} \times 1 + \text{TB} \times 5 + \text{NTB} \times 3 = (0.8 \times 1) + (0.2 \times 0.9 \times 5) + (0.2 \times 0.1 \times 3) = 1.76
\]
Processor 2:
\[
\text{CPI} = \text{INB} \times 1 + \text{TB} \times 4 + \text{NTB} \times 5 = (0.8 \times 1) + (0.2 \times 0.9 \times 4) + (0.2 \times 0.1 \times 5) = 1.62
\]

**Comparison**
One benchmark 1: Processor 1 is \(1.74/1.52 = 1.144\) times faster (14.4%)
One benchmark 2: Processor 2 is \(1.76/1.62 = 1.086\) times faster (8.6%)

Note that a general statement on performance comparison is NOT practical while considering both benchmarks at the same time unless we have total number of instructions in both benchmarks.
Q.No.3 (20 marks, all parts carry equal marks)
This description applies to part (a) and (b) only. The following program has been already rearranged to reduce branch hazards with a processor having 1 slot delayed branching.

```assembly
DADD    R1, R0, #5 ; Initialize R1 with value 5
Foo:   LD   R5, 2000(R2)
       SD   R5, 1000(R3)
       DADD R2, R2, #4
       DSUBUI R1, R1, #1 ; Decrement R1
       BNEQZ R1, foo ; Branch until R1 is not equal to 0.
       DADD R3, R3, #4
       OR
```

(a) Assume all registers initially have a zero value. What is final value of R3 with delayed branching

Following two instructions are always executed together.

BNEQZ   R1, foo
DADD    R3, R3, #4

Before Loop: R1 = 5: R2, R3 = 0, then after each iteration we have values as follows:
R1 = 4: R2, R3 = 4
R1 = 3: R2, R3 = 8
R1 = 2: R2, R3 = 12
R1 = 1: R2, R3 = 16
R1 = 0: R2, R3 = 20 (Last Iteration)

Ans: 20

(b) Repeat part (b) if cancelled branching is used.

Before Loop: R1 = 5: R2, R3 = 0, then after each iteration we have values as follows:
R1 = 4: R2, R3 = 4
R1 = 3: R2, R3 = 8
R1 = 2: R2, R3 = 12
R1 = 1: R2, R3 = 16
R1 = 0: R2=20, R3 = 16 (Last Iteration, branch not taken hence instruction following the branch is cancelled)

Ans: 16

(c) For part c-f, assume 5stage Integer MIPS pipeline. Give example of a program where forwarding is required between MEM to MEM stage. Do not need to explain.

(d) Give example of a program where forwarding is required between EXE to EXE stage.
(e) Give example of a program where forwarding is required between WB and ID stage.

(f) Explain two hazards that cause stall in Aggressive Integer MIPS pipeline and give examples. Get 1 bonus point if you name three.

(g) Explain Amdahl’s LAW.

(h) What are the checks that are done in the ISSUE stage of the pipeline in a scoreboard?

(i) What exceptions are possible in ID stage of the pipeline?
Q.No.4 (8 Marks)
(a) Explain direct addressing mode with example

(b) Explain Indirect addressing mode with example

(c) Explain how MIPS implements Direct and Indirect mode with example.

(d) Give example of a PC relative addressing mode.
Q.No. 5. (12+2+1 Marks)
For the floating point (useless) program shown in the figure below, describe how a scoreboard will advance the instruction with each clock cycle with following assumptions:

1. Execution Stage Latencies and number of units are as given in the table below.
2. Assume Read Operand can be performed in same clock cycle as Write Result.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Execution Cycles</th>
<th>Number of FUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>FP Add</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>DIV</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>MUL</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Issue</th>
<th>Read Operand</th>
<th>Execute</th>
<th>Write Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB.D F5, F6, F7</td>
<td>1</td>
<td>2</td>
<td>3-7</td>
<td>8</td>
</tr>
<tr>
<td>MUL.D F1, F2, F3</td>
<td>2</td>
<td>3</td>
<td>4-11</td>
<td>12</td>
</tr>
<tr>
<td>DIV.D F3, F1, F5</td>
<td>3</td>
<td>4-12 (RAW on F1)</td>
<td>13-23</td>
<td>24</td>
</tr>
<tr>
<td>ADD.D F5, F6, F5</td>
<td>9(WAW on F5)</td>
<td>10</td>
<td>11-15</td>
<td>16</td>
</tr>
<tr>
<td>MUL.D F6, F1, F5</td>
<td>13 (Only one MUL Unit)</td>
<td>14-16 (RAW on F5)</td>
<td>17-24</td>
<td>25</td>
</tr>
<tr>
<td>ADD.D F5, F7, F9</td>
<td>17(WAW on F5 again)</td>
<td>18</td>
<td>19-23</td>
<td>26 (cannot write result in cc 24, 25, structural hazard)</td>
</tr>
</tbody>
</table>

(b) Write values of Result Status Register when second MUL is about to write its result.

<table>
<thead>
<tr>
<th>Register</th>
<th>F0</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
<th>F6</th>
<th>F7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(c) In which clock cycle, pipeline is running with maximum efficiency?
Q.No 6 (10 +5 Marks)

Write all hazards in following codes

1. **SUB.D** F1, F6, F7
2. **MUL.D** F1, F2, F1
3. **DIV.D** F3, F1, F5
4. **ADD.D** F6, F3, #5
5. **MUL.D** F6, F6, #5
6. **S.D** F6, 34(R2)

<table>
<thead>
<tr>
<th>Hazard Type</th>
<th>Due to register</th>
<th>Between Instructions (Numbers 1 to 6)</th>
<th>Comments if any</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Re-write the program after removing name dependencies. Use registers F10 onwards for register renaming.
For the following program write the pipeline timing diagram, assuming number of unit and latencies are same as given in problem 5, however, full forwarding is available.

| Instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
|-------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| SUB.D F5, F6, F7 | F | D | E | E | E | E | E | M | W |
| MUL.D F1, F2, F3 | F | D | E | E | E | E | E | E | E | E | E | W | M |
| DIV.D F3, F1, F5 | F | D | D | D | D | D | D | D | E | E | E | E | E | E | E | E | E | E | E | E | E | E | M | W |
| ADD.D F5, F6, F8 | F | F | F | F | F | F | F | F | D | E | E | E | E | E | M | W |
| MUL.D F6, F1, F5 | F | D | D | D | D | D | E | E | E | E | E | E | E | E | M | W |
| ADD.D F5, F7, F9 | F | F | F | F | F | D | E | E | E | E | E | M | W |