1 (20 points) The following loop has multiple types of dependencies. Find all the true   
dependencies, output dependencies, and anti-dependencies.   
for (i=1; i<100; i++)   
{   
y[i] = x[i]/a; S1   
x[i] = x[i] + b; S2   
z[i] = y[i] - c; S3   
y[i] = c - y[i]; S4   
}3

Two true dependencies, RAW hazards on S1 and S3 (S3 ->T S1) also on S4 and S1 (S4 ->T S1)

Two anti-dependencies, WAR hazard on S1 and S2 (S1 ->A S2) as well as S3 and S4 (S3 ->A S4)

One output dependency, WAW hazard on S1 and S4 with both instructions trying to write to y[i] (S1 ->O S4)

2 (60 pts) Schedule the following code segment by applying Tomasulo’ algorithm.   
Assume the floating point addition takes 2 cycles, the floating point multiplication takes 3   
cycles and other operations take only 1 cycle, and two instructions can be dispatched per   
cycle. Figure 1 shows the initial states of the reservation station for the adder and   
multiplier, and the initial values of the floating point registers (FLR). Please schedule the   
instructions and update the states of the reservation stations and FLR for each clock cycle,   
until this code segment is finished.

i: R0 ⇐ R0 \* R2   
j: R2 ⇐ R4 + R8   
k: R4 ⇐R0 \* R8

cycle 1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS** | **Tag** | **Sink** | **Tag** | **Src** |
| **1** | 0 | 10.0 | 0 | 7.8 |
| **2** |  |  |  |  |
| **3** |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS** | **Tag** | **Sink** | **Tag** | **Src** |
| **4** | 0 | 6.0 | 0 | 3.5 |
| **5** |  |  |  |  |

Adder Mult/Div

|  |  |  |  |
| --- | --- | --- | --- |
| **FLR** | **Busy** | **Tag** | **Data** |
| **0** | X | 4 | -- |
| **2** | X | 1 | -- |
| **4** |  |  | 10.0 |
| **8** |  |  | 7.8 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS** | **Tag** | **Sink** | **Tag** | **Src** |
| **4** | 0 | 6.0 | 0 | 3.5 |
| **5** | 4 | -- | 0 | 7.8 |

cycle 2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS** | **Tag** | **Sink** | **Tag** | **Src** |
| **1** | 0 | 10.0 | 0 | 7.8 |
| **2** |  |  |  |  |
| **3** |  |  |  |  |

Adder Mult/Div

|  |  |  |  |
| --- | --- | --- | --- |
| **FLR** | **Busy** | **Tag** | **Data** |
| **0** | X | 4 | -- |
| **2** | X | 1 | -- |
| **4** |  |  | 10.0 |
| **8** | X | 5 | -- |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS** | **Tag** | **Sink** | **Tag** | **Src** |
| **4** | 0 | 6.0 | 0 | 3.5 |
| **5** | 4 | -- | 0 | 7.8 |

cycle 3:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS** | **Tag** | **Sink** | **Tag** | **Src** |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |

Adder Multi/Div

|  |  |  |  |
| --- | --- | --- | --- |
| **FLR** | **Busy** | **Tag** | **Data** |
| **0** | X | 4 | -- |
| **2** |  |  | 17.8 |
| **4** |  |  | 10.0 |
| **8** | X | 5 | -- |

cycle 4:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS** | **Tag** | **Sink** | **Tag** | **Src** |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS** | **Tag** | **Sink** | **Tag** | **Src** |
| **4** |  |  |  |  |
| **5** | 0 | 21.0 | 0 | 7.8 |

Adder Mult/Div

|  |  |  |  |
| --- | --- | --- | --- |
| **FLR** | **Busy** | **Tag** | **Data** |
| **0** |  |  | 21.0 |
| **2** |  |  | 17.8 |
| **4** |  |  | 10.0 |
| **8** | X | 5 | -- |

cycle 5:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS** | **Tag** | **Sink** | **Tag** | **Src** |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS** | **Tag** | **Sink** | **Tag** | **Src** |
| **4** |  |  |  |  |
| **5** | 0 | 21.0 | 0 | 7.8 |

Adder Mult/Div

|  |  |  |  |
| --- | --- | --- | --- |
| **FLR** | **Busy** | **Tag** | **Data** |
| **0** |  |  | 21.0 |
| **2** |  |  | 17.8 |
| **4** |  |  | 10.0 |
| **8** | X | 5 | -- |

cycle 6:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS** | **Tag** | **Sink** | **Tag** | **Src** |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS** | **Tag** | **Sink** | **Tag** | **Src** |
| **4** |  |  |  |  |
| **5** |  |  |  |  |

Adder Mult/Div

|  |  |  |  |
| --- | --- | --- | --- |
| **FLR** | **Busy** | **Tag** | **Data** |
| **0** |  |  | 21.0 |
| **2** |  |  | 17.8 |
| **4** |  |  | 10.0 |
| **8** |  |  | 163 |

3 (20 pts) In your own words, summarize how are RAW, WAR, and WAR dependences   
handled by the Tomasulo’s algorithm.

Tomasulo’s algorithm handles all of these dependency hazards by using tags to keep track of data in registers even if they need to be used in different instructions. For RAW it can use tags to label a register as busy and keep it from being used in any instruction until it has had all previous instructions completed. This shows that the tags can help to keep registers from being used with bad data and keeps register updates made by instructions in order. For WAR hazards the algorithm can use its tags to get the data needed from a register by keeping it in the Sink. This data can be held sperate from the register which frees up the register to be able to change or written to. This means that data inside a register does not need to be kept track of once an instruction has been read using that data. Finally, a WAW hazard is handled by the algorithm using tags again in a different way. When a register is written to a tag is assigned to it so that the data that is will be computed by the instruction knows where to go once it has been completed, but if another instruction writes to the register before an earlier instruction has finished its computation, then a new tag will be written to the register that uses the most recent instruction. This gets around the hazard because even if a different instruction used the register at any given time before or after the newest instruction issued it a new tag, the instruction getting data from that register will have the appropriate tag of what the data was supposed to be in that register at the time the instruction tried to retrieve data from it. So even if a WAW hazard occurred the instructions using the data from the register would be able to get what the data inside that register should have been at that time.