Homework 9

Problem 1

Suppose the order of the third and fourth cases (the two forwarding sources from the memory stage) in the HCL code for d_valA were reversed, as follows in table 1. Describe the resulting behavior of the rrmovl instruction (line5) for the following program:

```
int d_valA=[
     D_icode in { ICALL, IJXX } : D_valP;
     d_srcA == e_dstE : e_valE;
     d_srcA == M_dstM : m_valM;
     d_srcA == M_dstE : M_valE;
     d_srcA == W_dstM : W_valM;
     d_srcA == W_dstE :W_valE;
     1 : d_rvalA;
];
```

Table 1

```
1    irmovl $5, %edx
2    irmovl $0x100, %esp
3    rmmovl %edx, 0(%esp)
4    popl %esp
5    rrmovl %esp, %eax
```

Problem 2

Considering the codes below and answer the following questions:

Prog1	**Prog2**
irmovl $1, %ebx irmovl $2, %ecx addl %ebx, %ecx halt	rrmovl %eax, %ebx rrmovl %edx, %eax

1. How to solve above data hazard in **Prog1** by Forwarding ? You can draw a picture to illustrate yourself.
2. In **Prog2**, Is there a Data Hazard? If yes, how to solve it. If no, explain why.

Problem 3

```
#demo-retB.ys
0x000:      irmovl Stack,%esp      #Intialize stack pointer
0x006:      call p         #Procedure call
0x00b:      irmovl $5,%esi      #Return point
0x011:      halt
0x020: .pos 0x20
0x020: p:   irmovl $-1,%edi     #procedure
0x026:      ret
0x027:      irmovl $1,%eax      #Should not be executed
0x02d:      irmovl $2,%ecx      #Should not be executed
0x033:      irmovl $3,%edx      #Should not be executed
0x039:      irmovl $4,%ebx      #Should not be executed
0x100: .pos 0x100
0x100: Stack:       #Stack: Stackpointer
```

1. During the above example's executing, how many hazards will happen? Please pointed them out.
2. How will the data hazard in the above code be handled? (detail required)
3. What's the difference between **stall** and **bubble**?