1.

Field	rcall rB valC	rret rB
Fetch	icode:ifun ← M1[PC] rA:rB ← M1[PC+1]= F:rB valC ← M8[PC+2] valP ← PC + 10	icode:ifun ← M1[PC] rA:rB ← M1[PC+1]= F:rB valP ← PC + 2
Decode	--	valB ← R[rB]
Execute	--	valE ← 0 + valB
Memory	--	--
Write Back	R[rB] ← valP	--
PC update	PC ← valC	PC ← valE

2.

Condition	Trigger
Target-addr Hazard	IRRET in { D_icode,E_icode,M_icode }

Condition	Pipeline register				
	F	D	E	M	W
Target-addr Hazard	S	B	-	-	-

3.

Condition	Trigger
Target-addr Hazard	IRRET in { D_icode }

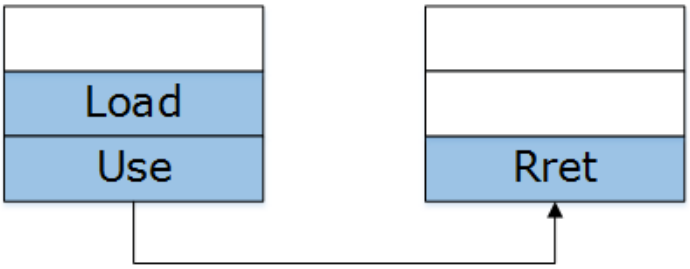
Condition	Pipeline register				
	F	D	E	M	W
Target-addr Hazard	S	B	-	-	-

4.

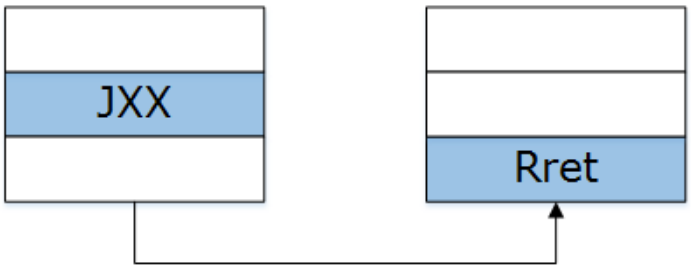
```
int f_PC = [
        M_icode == IJXX && !M_Cnd : M_valA;
        E_icode == IRRET : E_valB;
        1: F_predPC
];
 bool F_stall =
      # Conditions for a load/use hazard
      E_icode in { IMRMOVL, IPOPL } &&
      E_dstM in { d_srcA, d_srcB } ||
      # Stalling at fetch
      while ret passes through pipeline
      D_icode == IRRET;

bool D_stall =
      # Conditions for a load/use hazard
      E_icode in { IMRMOVL, IPOPL } &&
      E_dstM in { d_srcA, d_srcB };
```

```
bool D_bubble =
        # Mispredicted branch
        (E_icode == IJXX && !e_Bch) ||
        # Stalling at fetch
        while ret passes through pipeline
        D_icode == IRRET
        # but not condition for a load/use hazard
            && !(E_icode in { IMRMOVL, IPOPL }
                  && E_dstM in { d_srcA, d_srcB });
```

5.



Condition	F	D	E	M	W
Load	Stall	Stall	Bubble	Normal	Normal
RRET	Stall	Bubble	Normal	Normal	Normal
Combination	Stall	B+S	Bubble	Normal	Normal
Desired	Stall	Stall	Bubble	Normal	Normal



Condition	F	D	E	M	W
Mispredicted branch	Normal	Bubble	Bubble	Normal	Normal
RRET	Stall	Bubble	Normal	Normal	Normal
Combination	Stall	Bubble	Bubble	Normal	Normal

6. 25/8 21/4

7. 这里和用M_cnd而不用e_cnd一样，直接用d_rvalB会导致fetch周期过长。
具体来说，Instruction memory 以及 PC increment必须等待Select PC信号稳定之后才能执行（包含内部时序电路），如果select PC信号依赖于Decode阶段结束时的d_rvalB信号，那么Fetch Stage一定会比Decode stage长很多（本身就长很多），导致pipeline低效。