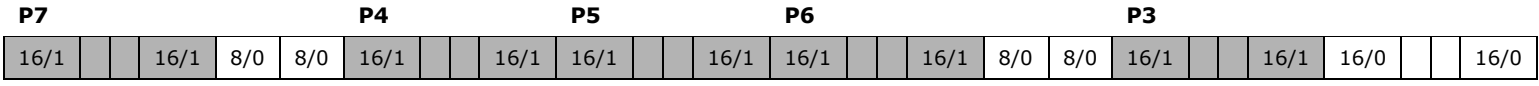
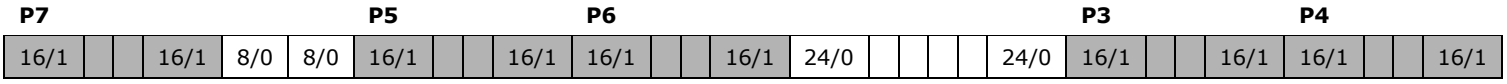
Problem 1: Memory Allocation (16points)

1.



(16-7)+(16-1)+(16-3)+(16-5) = 48 bytes

2.



(16-7)+(16-1)+(16-3)+(16-5) = 48 bytes

3

first-fit: Yes, (24-7)+(32-1)+(16-3)+(24-5) = 80 bytes
best-fit: No, fail on the fourth step

Problem 2: Linking (18points)

1. [1] yes [2] global
   [4] yes [5] global
   [7] yes [8] local
   [10] no [11] --
   [12] yes [10] global
2. a=0x0 b=0x2 c=0x0 d=0x4
3. 1) 0x08048302
   2) 5

Problem 3: Linking (21points)

1 [1] LOCAL [2] GLOBAL
   [3] -- [4] .data [5] .bss
2 [1] R_386_32 [2] 0000001b
   [3] 00000023 [4] a
   [5] R_386_32 [6] 00000022
   [7] R_386_PC32 [8] foo/.text
3 [1] 2c 97 04 08 [2] a1 30 97 04 08
   [3] e8 ad ff ff ff [4] a3 88 97 04 08

Problem 4: Processor (45points)

1. (12′)

Field	rret rB	rcall rB valC
Fetch	icode:ifun ← M1[PC] rA:rB ← M1[PC+1]= F:rB valP ← PC + 2	icode:ifun ← M1[PC] rA:rB ← M1[PC+1]= F:rB valC ← M4[PC+1] valP ← PC + 6
Decode	valB ← R[rB]	--
Execute	valE ← 0 + valB	valE ← valP+0
Memory	--	--
Write Back	--	R[rB] ← valP
PC update	PC ← valE	PC ← valC

2. (6′)

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. (6′)

Condition	Trigger
Target-addr Hazard	IRRET in { D_icode }

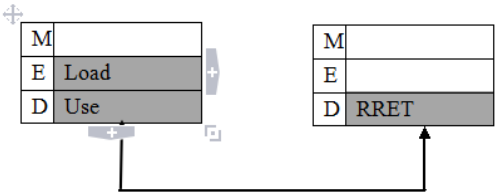
Condition	Pipeline register				
	F	D	E	M	W
Target-addr Hazard	S	B	-	-	-

4. (8′)

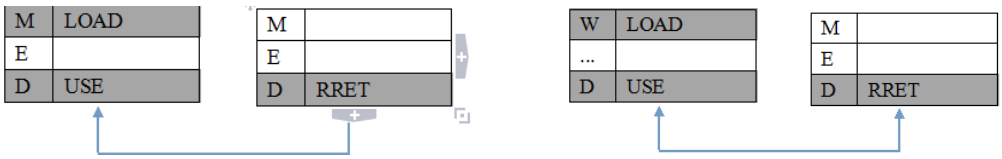
```
int f_pc = [
    E_icode == IRRET  : E_valB;
];

bool F_stall =
    #Stalling at fetch before ret forwarding the E_valB
    D_icode == IRRET  ||
    #load/use hazard (optional)
    (E_icode in { IMRMOVL, IPOPL } && E_dstM == d_srcB);
    or
    d_srcB != RNONE && d_srcB in {E_dstM,e_dstE,M_dstM,M_dstE,W_dstM,W_dstE};
    #other expression about the load/use hazard is also right!
```

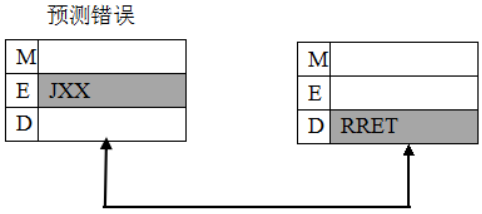
```
bool D_stall =
    #load/use hazard (optional)
    E_icode in { IMRMOVL, IPOPL } && E_dstM == d_srcB ;
    or
    d_srcB!=RNONE && d_srcB in {E_dstM,e_dstE,M_dstM,M_dstE,W_dstM,W_dstE}
    #other expression about the load/use hazard is also right!

bool D_bubble =
    # Stalling at fetch while ret forwarding the E_valB
    IRET in { D_icode,}
    # but not condition for a load/use hazard (optional)
    && !(E_icode in { IMRMOVL, IPOPL } && E_dstM != d_srcB)
    or
    && !(d_srcB!=RNONE && d_srcB in
    {E_dstM,e_dstE,M_dstM,M_dstE,W_dstM,W_dstE})
```

5. (6')



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. [1] 25/38 [2] 9/22
[3] 23/32 [4] 4/16

7. 不能正确 forwarding。
Decode 阶段，需要一段时间才能读取寄存器的值，而在 fetch 阶段一开始便需要 select pc，无法等到 decode 读完寄存器后再 forward 过来。