I. Control Hazard

Assume we have a PIPE implementation of Y86-64, and we run the following program on it:

```
0x000     main:
0x000         irmovq Stack, %rsp
0x00a         irmovq $1, %rax
0x014         irmovq $2, %rbx
0x01e         call   f
0x027         hlt
0x028     f:
0x028         irmovq $3, %rcx
0x032         ret
0x033         irmovq $4, %rdx
0x03d         irmovq $5, %rdx
0x047         irmovq $6, %rdx
0x051         irmovq $7, %rdx
          .pos 0x100
0x100     Stack:
```

1. Please fill the following table of the status **AFTER** CPU stop:
   (All registers are initialized to **0** before CPU start)

Register	Value
%rax	
%rbx	
%rcx	
%rdx	
%rsp	

2. Please fill the table that showing the `ret` instruction passes through the pipeline.
   (You can write `---` for unknown value. e.g. `valC` for `NOP` instruction)

Register	Cycle X	X + 1	X + 2	X + 3	X + 4
F_predPC	0x32				
f_pc	0x32				
D_icode	IIRMOVQ				
D_valA	---				
D_valC	3				
E_icode	ICALL				
E_valA					
E_valC					
M_icode					
M_valE					
W_icode					
W_valM					

3. Assume we make a mistake in the HCL code, so the HCL of D_bubble looks like this:

```
D_bubble = [
        # Handle mispredicted branch
        …
        # Stalling at fetch while ret passes through pipeline
        IRET in { D_icode, M_icode };
]
```

Please fill the above table again:

Register	Value
%rax	
%rbx	
%rcx	
%rdx	
%rsp	

II. Exception handling

If we want to modify the PIPE implementation so that it will reboot after any exception, please write the HCL changes. For simplicity, you should only reboot the CPU when the exception reach W stage, even if it is detected earlier. And you **DO NOT** need to consider the combination between reboot hazard and other hazards.

(You can reboot the CPU by reseting PC back to 0. You **MUST** preserve other registers, CC and memory)

III. Pipeline Performance

The following program will calculate `%rdx = |*%rax| + |*%rbx| + |*%rcx|`.

```
f:
      irmovq     $0, %rdx
      mrmovq     (%rax), %rax
      andq       %rax, %rax
      jle        sub_rax
      addq       %rax, %rdx
      jmp        rax_done
sub_rax:
      subq       %rax, %rdx
rax_done:

      mrmovq     (%rbx), %rbx
      andq       %rbx, %rbx
      jle        sub_rbx
      addq       %rbx, %rdx
      jmp        rbx_done
sub_rbx:
      subq       %rbx, %rdx
rbx_done:

      mrmovq     (%rcx), %rcx
      andq       %rcx, %rcx
      jle        sub_rcx
      addq       %rcx, %rdx
      jmp        rcx_done
sub_rcx:
      subq       %rcx, %rdx
rcx_done:
```

We run this program on the following three pipeline implementation:

A. No prediction (Stall until conditional jump reach E stage)
B. Always taken prediction
C. Always not-taken prediction

Assume `*%rax=1, *%rbx=2, *%rcx=-1`, please show the CPI of the three implementations.