

Problem 1:

Please write down the HCL code for the following signals in the SEQ implementation.

1. need_valC
2. aluB
3. mem_write

Problem 2:

Please fill the blanks below. Assume the value of PC is 0x10.

Stage	Generic	Specific
	irmovq V, rB	Irmovq 0x123, %rbx
Fetch	icode:ifun <- M1[PC] rA:rB <- M1[PC+1] valC <- M8[PC+2] valP <- PC+10	
Decode		
Execute	valE <- 0 + valC	
Memory		
Write back	R[rB] <- valE	
PC update	PC <- valP	

Please fill the blanks below. Assume the value of PC is 0x10 and the value of %rsp is 0x200. The 8-byte value stored in 0x200 is 0x4000.

Stage	Generic	Specific
	popq rA	popq %rax
Fetch	icode:ifun <- M1[PC] rA:rB <- M1[PC+1] valP <- PC+2	
Decode	valA <- R[%rsp] valB <- R[%rsp]	
Execute	valE <- valB + 8	
Memory	valM <- M8[valA]	
Write back	R[%rsp] <- valE R[rA] <- valM	
PC update	PC <- valP	

Problem 3:

Please explain the pros and cons of RISC compared with CISC.

Call and ret instructions access stack to save and load return addresses. Please consider a different set of these instructions:

callr Dest, which saves the return address in %rsi and jumps to the instruction at Dest;

retr, which jumps to the instruction with the address saved in %rsi.

1. Please fill in the blanks below to describe these two instructions which are executed in sequential Y86 implementation. You only need to write the generic logic for each instruction.

Stage	Generic	Specific
	callr Dest	retr
Fetch		
Decode		
Execute		
Memory		
Write back		
PC update		

2. For any correctly implemented Y86 program, if we simply replace all call instructions with callr and all ret with retr, will the program run correctly? Why or why not?