HW 12

Problem 1.

The following four problems are based on the following code snippet:

```
# int Sum(int *Start, int Count)


0x042: a05f          Sum:    pushl %ebp
0x044: 2045                  rrmovl %esp,%ebp
0x046: 501508000000          mrmovl 8(%ebp),%ecx      # ecx = Start
0x04c: 50250c000000          mrmovl 12(%ebp),%edx     # edx = Count
0x052: 6300                  xorl %eax,%eax           # sum = 0
0x054: 6222                  andl %edx,%edx           # Set condition codes
0x056: 7378000000            je End
0x05b: 506100000000 Loop:    mrmovl (%ecx),%esi       # get *Start
0x061: 6060                  addl %esi,%eax           # add to sum
0x063: 30f304000000          irmovl $4,%ebx
0x069: 6031                  addl %ebx,%ecx           # Start++
0x06b: 30f3ffffffff          irmovl $-1,%ebx
0x071: 6032                  addl %ebx,%edx           # Count--
0x073: 745b000000            jne Loop                 # Stop when 0
0x078: 2054          End:    rrmovl %ebp,%esp
0x07a: b05f                  popl %ebp
0x07c: 90                    ret
```

Note: Here are some values of the registers you maybe use before the corresponding instructions. The value stored in %esp is 128, value in %eax is 1, value in %ecx is 256. Fill in the blank of the following table (if there should be nothing, using "----").

1.1

Stage	**Generic**	**Specific**
	rrmovl rA, rB	rrmovl %esp,%ebp
Fetch	icode : ifun ← M1[PC] rA:rB←M1[PC+1] valP←PC+2	
Decode	valA ← R[rA]	
Execute	valE ← 0 + valA	
Memory		
Write Back	R[rB] ← valE	
Update PC	PC ← valP	

1.2

Stage	**Generic**	**Specific**
	OPl rA, rB	xorl %eax,%eax
Fetch	icode : ifun ← M1[PC] rA:rB←M1[PC+1] valP←PC+2	
Decode	valA ← R[rA] valB ← R[rB]	
Execute	valE←valB OP valA Set CC	
Memory		
Write Back	R[rB] ← valE	
Update PC	PC ← valP	

1.3

Stage	**Generic**	**Specific**
	irmovl V, rB	irmovl $4,%ebx
Fetch		
Decode		
Execute		
Memory		
Write Back		
Update PC		

1.4

Stage	**Generic**	**Specific**
	OPl rA, rB	addl %ebx,%ecx
Fetch	icode : ifun ← M1[PC] rA:rB ← M1[PC+1]	

	valP ← PC+2	
Decode	valA ← R[rA] valB ← R[rB]	
Execute	valE←valB OP valA Set CC	
Memory		
Write Back	R[rB] ← valE	
Update PC	PC ← valP	

Problem 2

As described in Section 3.7.2, the IA32 instruction 'leave' can be used to prepare the stack for returning. It is equivalent to the following Y86 code sequence:

rrmovl %ebp, %esp Set stack pointer to beginning of frame

popl %ebp Restore saved %ebp and set stack ptr to end of caller's frame

Suppose we add this instruction to the Y86 instruction set, using the following encoding:

Byte 0 1 2 3 4 5

Leave

D	0

Describe the computations performed to implement this instruction.

Stage	leave
Fetch	icode: ifun ← M1[PC] valP ← PC+1
Decode	
Execute	
Memory	
Write Back	
Update PC	