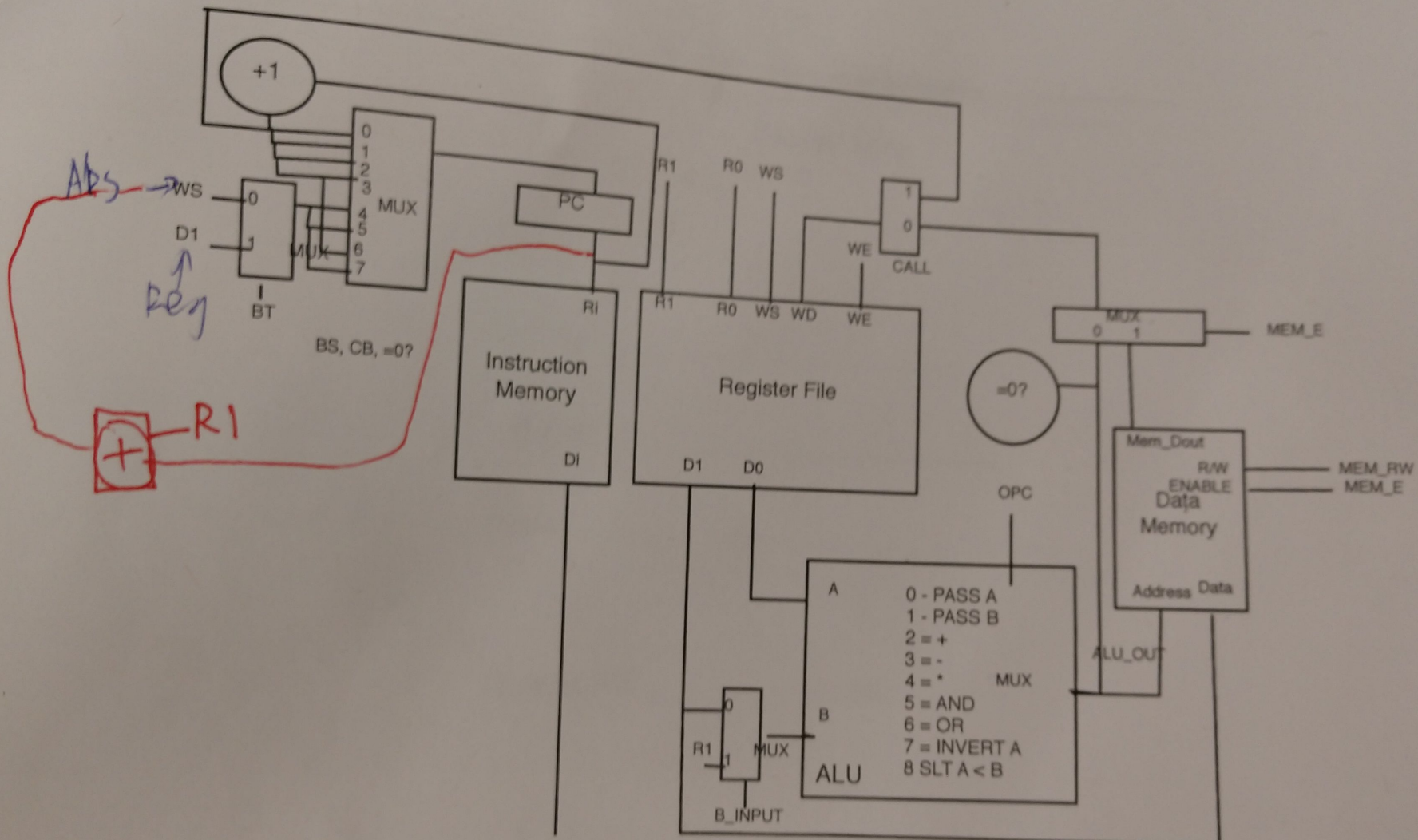
```
sum(unsigned char*, int):
        sub     sp, sp, #16  //allocate space in stack pointer(r13)
        stur    r4, [sp, #8]  //save r4 into stack(callee saved)
        stur    r5, [sp, #16]  //save r5 into stack(callee saved)
        mov     r3, #0  //s = 0
        mov     r2, #0  //i = 0
        mov     r5, #0  //zeros out r5(loaded value is 4 bit, so the upper 4 bits should be zeroed out)
.LOOP:
        cmp     r2, r3  //i  < length?
        bge     .END
        add     r4, r0, r2  //calculate address
        ldrb    r5, [r4]    //load the value to r5
        add     r3, r3, r5  // s+= array[i]
        add     r2, r2, #1  //i++
        b       .LOOP
.END:
        mov     r0, r3
        ldur    r4, [sp, #8]  //load r4 into stack
        ldur    r5, [sp, #16]  //load r5 into stack
        add     sp, sp, #16
        bx      lr  //branch back to link register to return
```

Place your work here. Assume the first line of your code is executing at address 0.

	MEM_RW	MEM_E	B_INPUT,CALL	BT,CB,BS	OPC	WE	WS	R1	R0	Description
00000	X	0	10	XXO	2	1	2	0	2	ADDI X2,X2,#0 (5)
00001 LOOP:	X	0	10	XXO	2	1	3	0	3	ADDI X3,X3,#0 (i)
00010	X	0	0X	011	3	0	01001	2	1	B.EQ X1,X2 GND
00011	X	0	00	XXO	2	1	4	0	3	ADD X4,X0,X3
00100	1	1	10	XXO	2	1	4	0	5	LDUR X5,[X4,#0]
00101	X	0	10	XXO	5	1	5	FF	5	ANDI X5,X5,#FF
00110	X	0	00	XXO	2	1	2	5	2	ADD X2,X5,X2
00111	X	0	10	XXO	2	1	3	1	3	ADDI X3,X3,#1
01000	X	0	XX	001	X	0	00010	X	X	B LOOP
01001 END:	X	0	XX	101	X	0	X	254	X	B X254
01010										
01100										
01101										

Question 3:



Modify this architecture from question 2 to support a PC relative branch. Specifically support branching of the form PC + constant for both conditional and unconditional branches. Your solution can drop support for jumping to an absolute address (constant) that is specified in the architecture.