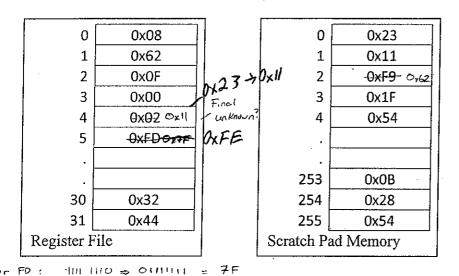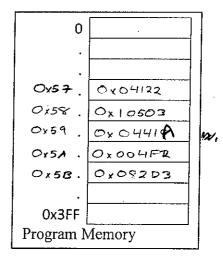### 4. RAT ASM and Memory (15)

Figures 1 and 2 show the current contents of the register file and of the scratch pad memory. The following program is executed.

```
;------------------------------------------------------
;- Memory Designation Constants
;------------------------------------------------------
.DSEG
.ORG     0x00
;------------------------------------------------------

;------------------------------------------------------           05A =  00  0101  1010
;- Main program
;------------------------------------------------------
.CSEG
.ORG          0x57

        ST     R1, (R4)    ;  0001000010010010 = 0x04122
        ROR    R5          ;  0100001011000011 = 0x10503
        LD     R4, (R3)    ;  0010010000001010 = 0x04419
label:  EXOR   R4, R30     ;  0000010011110010 = 0x004F2
        BRNE   label       ;  0010001010110011 = 0x08203
```

1. Write the machine code for each instruction and fill in the prog_rom using 5 hex values. If there are any don't cares in the instruction, use '0' for the don't care. Label the address of each row you fill in.
2. Modify the register contents in the figures to show how the register contents and memory contents change as a result of execution.

	Register File			Scratch Pad Memory			Program Memory
0	0x08		0	0x23		0	
1	0x62		1	0x11			
2	0x0F	0x23→0x11	2	0xF9 0x62			
3	0x00	Final	3	0x1F		0x57	0x04122
4	0x02 0x11	unknown?	4	0x54		0x58	0x10503
5	0xFD 0xFF 0xFE		.			0x59	0x04419
			.			0x5A	0x004F2
.			253	0x0B		0x5B	0x08203
30	0x32		254	0x28		.	
31	0x44		255	0x54		0x3FF	

for FD :   1111 1110 ⇒ 01111111 = 7F

Exor R4

1st: 0x23 ⊕ 0x32 :   00100011
                    ⊕ 00110010
                    ─────────
                      00010001 → R4 = 0x11

2nd: 00010001
     ⊕ 00110010
     ─────────
       00100011 → R4 = 0x23   infinite loop ?

— 🔟 ok

5. RAT ASM Interpretation (10)

Assume that the register R3 is initialized to ⑩ and the register R2 is initialized to zero. What is the value in R2 when the code reaches done? Explain your answer.

```
        MOV    R1, 0x00
loop:   CMP    R1, R3
        BRCS   else
        BRN    done
else:   ADD    R2, 0x02
        XOR    R2, 0x0A
        SUB    R3, 0x01
        BRN    loop
done:
```

	Initial	Loop 1	Loop 2	Loop 3	Loop 4
R1	0x00	0x00	0x00	0x00	0x00
R2	0x00	0x02 → 0x08	0x04 → 0x00	0x02 → 0x08	0x02 → 0x00
R3	0x13	0x12	0x11	0x10	0x09

The loop runs 13 times (ie, until ⓪ − R3 = ⊘)

R2 toggles between 0x00 and 0x08
                   Even loop   Odd loops

So after ~~13~~ 19 loops (an odd loop), R2 = 0x08 ✓