5.10

A:BRnzp -171

B:JSP -171

Both A and B result in PC being changed to (PC+1)-171.

However, B saves the linkage information to R7, but A does not affect R7.

5.15

R1=x3121

R2=x4566

R3=xABCD

R4=xABCD

5.33 R5 has exactly 5 “ones” of the lower 8 bits.

5.34 The Register file and the ALU in figure 5.18 implement the NOT instruction, alongwith NZP and the logic which goes with it.

5.41

a. Y is the P condition code.

b. Yes. The problem is that the logic should not have the logic gate A. Because only the instruction which can change the condition code should be in the given logic. It is obvious that BR(0000) shouldn’t be in the given logic.

6.3

(0000) 3000 0011000000000000 ( 1) .ORIG x3000

(3000) A40D 1010010000001101 ( 2) LDI R2 cur

(3001) 5260 0101001001100000 ( 3) AND R1 R1 #0

(3002) 1261 0001001001100001 ( 4) ADD R1 R1 #1

(3003) 1241 0001001001000001 ( 5) loop ADD R1 R1 R1

(3004) 14BF 0001010010111111 ( 6) ADD R2 R2 #-1

(3005) 0401 0000010000000001 ( 7) BRZ work

(3006) 0FFC 0000111111111100 ( 8) BRNZP loop

(3007) A607 1010011000000111 ( 9) work LDI R3 a

(3008) 96FF 1001011011111111 ( 10) NOT R3 R3

(3009) 927F 1001001001111111 ( 11) NOT R1 R1

(300A) 5243 0101001001000011 ( 12) AND R1 R1 R3

(300B) 927F 1001001001111111 ( 13) NOT R1 R1

(300C) B202 1011001000000010 ( 14) STI R1 a

(300D) F025 1111000000100101 ( 15) TRAP x25

(300E) 4000 0100000000000000 ( 16) cur .FILL x4000

(300F) 4001 0100000000000001 ( 17) a .FILL x4001

6.7

This program adds together the corresponding elements of two lists. The first lists begins at x300E, and the other begins at x3013. The length of the two lists are stored in x3018. The first element of the first list is added to the first element of the second list, and the result is stored back to the first element of the first list; the second element of the first list is added to the second element of the second list, and the result is stored back to the second element of the first list; and so on.

6.9

(0000) 3000 0011000000000000 ( 1) .ORIG x3000

(3000) 2005 0010000000000101 ( 2) LD R0 number

(3001) 2205 0010001000000101 ( 3) LD R1 time

(3002) F021 1111000000100001 ( 4) loop TRAP x21

(3003) 127F 0001001001111111 ( 5) ADD R1 R1 #-1

(3004) 03FD 0000001111111101 ( 6) BRP loop

(3005) F025 1111000000100101 ( 7) TRAP x25

(3006) 005A 0000000001011010 ( 8) number .FILL x005A

(3007) 0064 0000000001100100 ( 9) time .FILL x0064

6.10

The program output 1 to the screen if the number stored in R2 is odd, else output 0.

(0000) 3000 0011000000000000 ( 1) .ORIG x3000

(3000) 50A1 0101000010100001 ( 2) AND R0 R2 #1

(3001) 2203 0010001000000011 ( 3) LD R1 ASCII

(3002) 1001 0001000000000001 ( 4) ADD R0 R0 R1

(3003) F021 1111000000100001 ( 5) TRAP x21

(3004) F025 1111000000100101 ( 6) TRAP x25

(3005) 0030 0000000000110000 ( 7) ASCII .FILL x0030