5.15

R1: 0x3121

R2: 0x4566

R3: 0xABCD

R4: 0xABCD

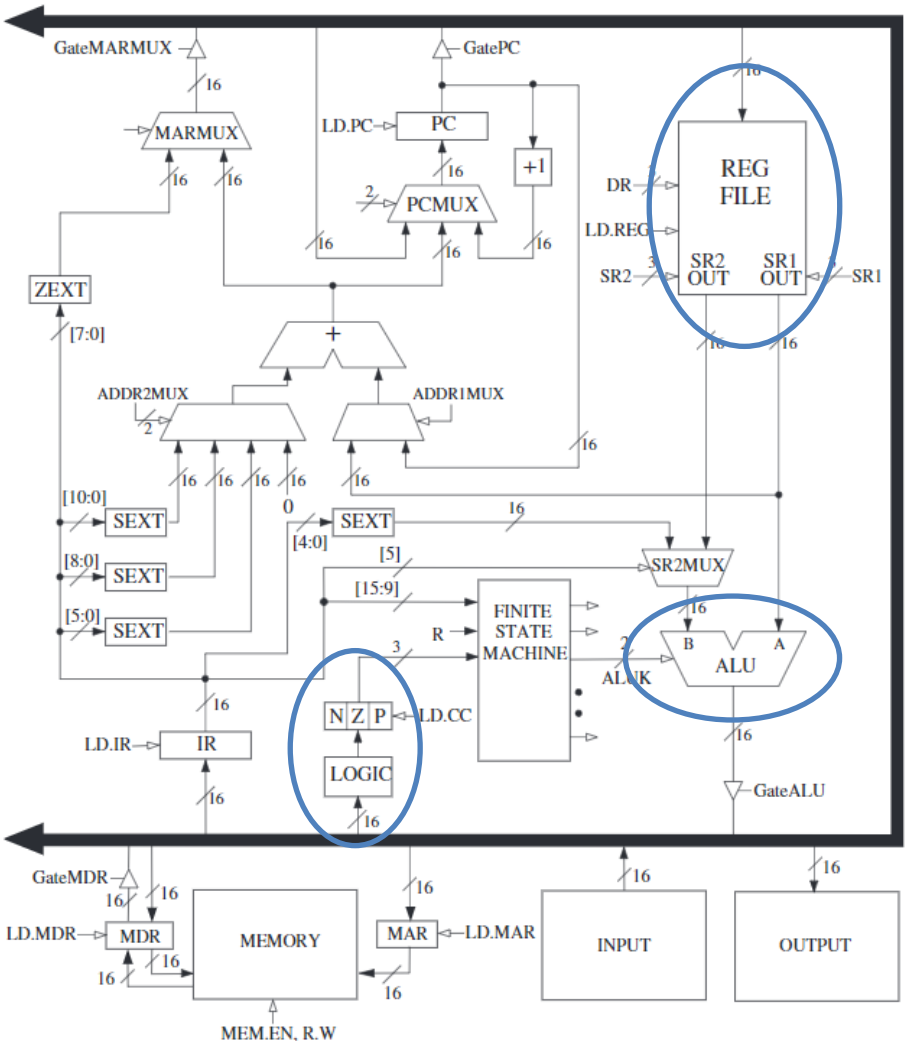
5.25

```
        AND     R1, R1, #0
        NOT     R4, R3
        ADD     R4, R4, #1
        ADD     R5, R2, R4
        BRz     END
        BRn     NEG
        BRp     POS
NEG     ADD     R1, R3, #0
        BRnzp   END
POS     ADD     R1, R2, #0
END     TRAP    0x25
```

5.33

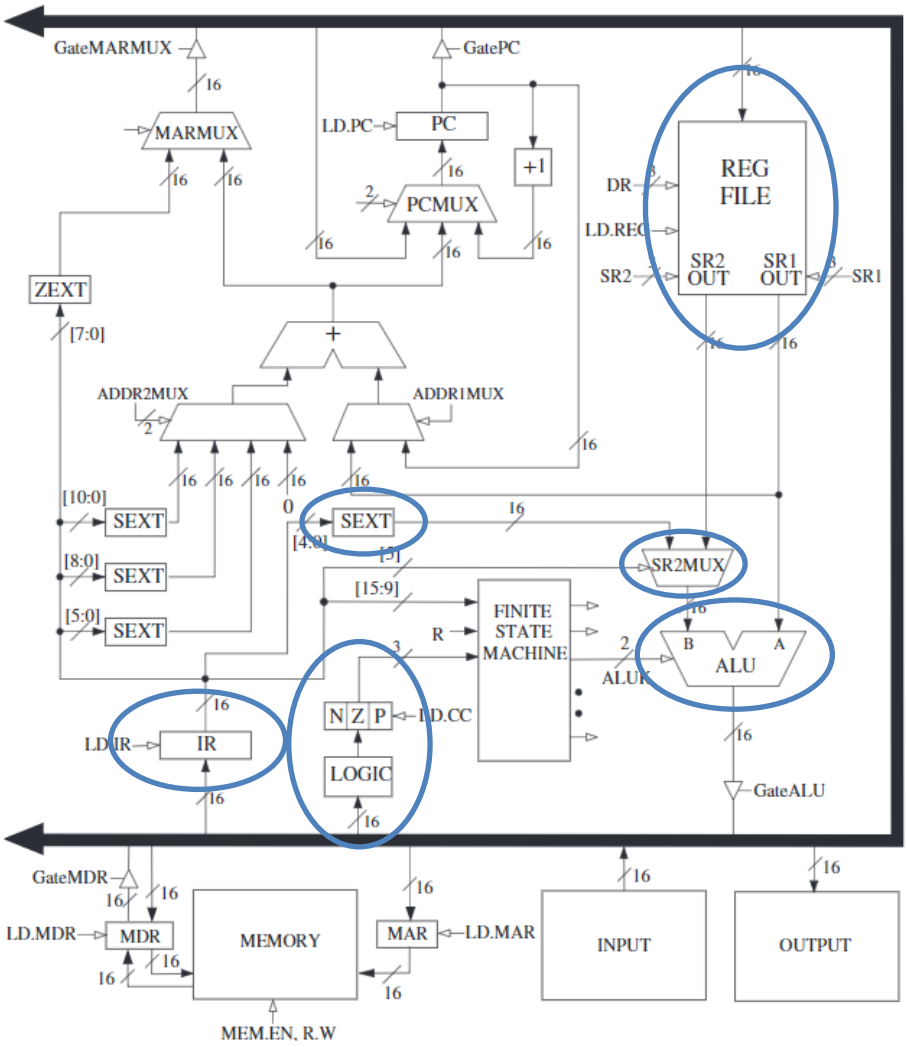
It can be inferred that R5 has 5 1s of its lower 8 bits.

5.34



(Figure 5.4 in Edition 3 has NZP and the logic, but Figure 5.4 in Edition 2 does not. I identify NZP and the logic anyway.)

5.35



(Figure 5.5 in Edition 3 has NZP and the logic, but Figure 5.5 in Edition 2 does not. I identify NZP and the logic anyway.)

5.40

To signal whether the instruction in IR is a BR(Conditional Branch) instruction with at least one condition code that is 1.

5.41

a. Y is the P condition code that signals whether the value is

positive.

b. Yes. The gate A should not be included in the logic that produces X, because the BR instruction is not an instruction that changes the condition codes.

6.9

```
x3000   0010 000 001100011      ; load ‘Z’ to R0
x3001   0010 001 001100011      ; load 100 to R1
x3002   1111 0000 00100001      ; display ‘Z’
x3003   0001 001 001 1 1111     ; add -1 to R1
x3004   0000 001 111111101      ; continue to display
x3005   1111 0000 00100101      ; halt
x3100   0000 0000 0101 1010     ; the ASCII of ‘Z’
x3101   0000 0000 0110 0100     ; 100
```

6.10

If the number stored in R2 is odd, the routine will store 1 in R3. Otherwise, it will store 0 in R3.

```
x3000   0101 011 011 1 00000    ; clear R3
x3001   0101 000 010 1 00001    ; check the lowest bit
x3002   0000 010 000000001      ; even
x3003   0001 011 011 1 00001    ; add 1 to R3 when R2 is odd
```

x3004 1111 0000 00100101 ; halt