Bitwise and Bitfield Instructions

Prerequisite Reading: Chapter 7
Revised: March 26, 2020

Objectives: *To use the web-based simulator ("CPUlator") to better understand …*

1. The operation of the bitwise instructions (MVN, AND, ORR, EOR and BIC)
2. The operation of the bitfield instructions (BFC, BFI, UBFX, and SBFX)
3. The use of the C left-shift operator to create constants.

To do offline: Answer the questions that follow the listing below. (Numbers at far left are memory addresses.)

```
                .syntax     unified
                .global     _start

00000000  _start:  LDR      R0,=0xFFFF << 16        // *** EXECUTION STARTS HERE ***
00000004           MVN      R0,R0
00000008           EOR      R0,R0,0xFF << 12
0000000C           BIC      R0,R0,0xFF << 0
00000010           ORR      R0,R0,0xFF << 12
00000014           AND      R0,R0,0xFF << 12

00000018           LDR      R1,=0x23456789
0000001C           BFI      R0,R1,24,8
00000020           BFC      R0,12,8
00000024           UBFX     R1,R0,24,8
00000028           SBFX     R1,R0,24,8

0000002C  done:    B        done                   // Infinite loop

                .end
```

What is left in register R0 after executing the LDR instruction at 00000000_{16}?

R0 (as hexadecimal)
FFFF0000

What is left in register R0 after executing the MVN instruction at 00000004_{16}?

R0 (as hexadecimal)
0000FFFF

What is left in register R0 after executing the EOR instruction at 00000008_{16}?

R0 (as hexadecimal)
000F0FFF

What is left in register R0 after executing the BIC instruction at $0000000C_{16}$?

R0 (as hexadecimal)
000F0F00

What is left in register R0 after executing the ORR instruction at 00000010_{16}?

R0 (as hexadecimal)
000FFF00

What is left in register R0 after executing the AND instruction at 00000014_{16}?

R0 (as hexadecimal)
000FF000

What is left in register R1 after executing the LDR instruction at 00000018_{16}?

R1 (as hexadecimal)
23456789

What is left in register `R0` after executing the `BFI` instruction at $0000001C_{16}$?

R0 (as hexadecimal)
890FF000

What is left in register `R0` after executing the `BFC` instruction at 00000020_{16}?

R0 (as hexadecimal)
89000000

What is left in register `R1` after executing the `UBFX` instruction at 00000024_{16}?

R0 (as hexadecimal)
00000089

What is left in register `R1` after executing the `SBFX` instruction at 00000028_{16}?

R1 (as hexadecimal)
FFFFFF89

Getting ready: Now use the simulator to collect the following information and compare to your earlier answers.

1. Click here to open a browser for the ARM instruction simulator with pre-loaded code.

Step 1: *Press **F2** exactly **once** to execute the LDR instruction at 00000000_{16}*

What is left in register `R0` after executing the `LDR` instruction at 00000000_{16}?

R0 (as hexadecimal)
FFFF0000

Step 2: *Press **F2** exactly **once** to execute the MVN instruction at 00000004_{16}*

What is left in register `R0` after executing the `MVN` instruction at 00000004_{16}?

R0 (as hexadecimal)
0000FFFF

Step 3: *Press **F2** exactly **once** to execute the EOR instruction at 00000008_{16}*

What is left in register `R0` after executing the `EOR` instruction at 00000008_{16}?

R0 (as hexadecimal)
000F0FFF

Step 4: *Press **F2** exactly **once** to execute the BIC instruction at $0000000C_{16}$*

What is left in register `R0` after executing the `BIC` instruction at $0000000C_{16}$?

R0 (as hexadecimal)
000F0F00

Step 5: *Press **F2** exactly **once** to execute the ORR instruction at 00000010_{16}*

What is left in register `R0` after executing the `ORR` instruction at 00000010_{16}?

R0 (as hexadecimal)
000FFF00

Step 6: *Press **F2** exactly **once** to execute the AND instruction at 00000014_{16}*

What is left in register `R0` after executing the `AND` instruction at 00000014_{16}?

R0 (as hexadecimal)
000FF000

Step 7: *Press **F2** exactly **once** to execute the LDR instruction at 00000018_{16}*

What is left in register `R1` after executing the `LDR` instruction at 00000018_{16}?

R1 (as hexadecimal)
23456789

Step 8: *Press **F2** exactly **once** to execute the BFI instruction at $0000001C_{16}$*

What is left in register `R0` after executing the `BFI` instruction at $0000001C_{16}$?

R0 (as hexadecimal)
890FF000

Step 9: *Press **F2** exactly **once** to execute the BFC instruction at 00000020_{16}*

What is left in register `R0` after executing the `BFC` instruction at 00000020_{16}?

R0 (as hexadecimal)
89000000

Step 10: *Press **F2** exactly **once** to execute the UBFX instruction at 00000024_{16}*

What is left in register `R1` after executing the `UBFX` instruction at 00000024_{16}?

R1 (as hexadecimal)
00000089

Step 11: *Press **F2** exactly **once** to execute the SBFX instruction at 00000028_{16}*

What is left in register `R1` after executing the `SBFX` instruction at 00000028_{16}?

R1 (as hexadecimal)
FFFFFF89