ABS – Solving the sign extension problem

LC3SignExt:

C:\Users\Alifa\Desktop\mystuff\faiz.dev.root\gitdir\github-fizaan\temp\vsc-java\java\

Note: SEXT is **not the same as** **or doesn’t represent** a Two’s complement negative number.

/\*

    30BE/12478 0000111111110011: 0FF3 00    br <> 12478 - 499 (65523) JUMP

    30B2/12466 0010000001011001: 2059 02    LD r0 12466 + 89 (89)

\*/

public class LC3SignExt {

    public static void main(String args[]) {

        int pc = 0x30bf; //incremented PC

        int y = 0xfff3;

        pc += y;

        char tochar = (char) pc;

        System.out.printf("%04X %04X\n", pc, (int) tochar);

        //correct? No

        pc = 0x30bf; //incremented PC

        y = 0x1f3;

        pc -= y;

        System.out.printf("%04X\n", pc);

        //correct? Yes

        pc = 0x30bf; //incremented PC

        y = 0xFFF9; //-7

        pc += y;

        System.out.printf("%04X\n", pc);

        /\*

            Problem with sign extension:

            pc = 0x3000

            lea r6 stack

            pc = 0x3001

            stack PC relative addr is 271

            LEA instruction takes a 9-bit

            offset. If 271, a positive #,

            is put within 9-bits, then it

            is represented as 100001111

            but the LSBit of 100001111 is

            1! So SEXT(271) to 16-bits =

            1111111100001111 which is FF0F

            0x3001 + 0xFF0F = 12F10

            Hence why you see this which is

            clearly wrong!

            3000/12288 1110110100001111: ED0F 0E    lea r6[0000]  12288(3000) - 271(FF0F)

        \*/

        pc = 0x3001; //incremented PC from 0x3000

        y = 0xFF0F; // SEXT(271) to 16-bits

        pc += y;

        System.out.printf("%04X\n", pc);

        pc = 0x3001; //incremented PC from 0x3000

        y = 0x10F; // 271 no sign extension

        pc += y;

        System.out.printf("%04X\n", pc);

    }

}