Review:

- for exam: - representing information (binary, hex, ASCII, floating pt.)
  - ARM (all topics related to lectures / labs)
  - memories & caches
  - C code (like in Lab 7)

- convert -800 to 2's complement.

$$+800 = 8\times100 = 8(64+32+4) = 2^3(2^6+2^5+2^2)$$
$$= 2^9 + 2^8 + 2^5$$
$$= 0\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0$$

$$-800 = (1\,0\,0\,1\,1\,1\,0\,0\,0\,0\,0)_2$$

- convert -512 (use min # of bits)

$$+512 = 2^9 = 0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0$$

$$-512 = 1\,0\,0\,0\,0\,0\,0\,0\,0\,0$$

- What's the decimal value of the 2's complement number of 1?

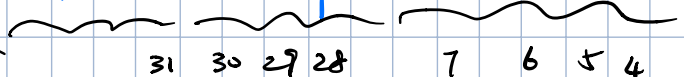
with 1 bit $\begin{cases} 0 \\ 1 \end{cases}$ (-1)

- Addition:

$$\begin{array}{r} 0\,1\,1\,1\,0\,0\,1\,1 \\ 0\,1\,0\,0\,1\,1\,1\,1 \\ 1\,1\,0\,1\,1\,0\,0\,0 \\ +\quad 1\,1\,1\,1\,1\,1\,1\,1 \\ \hline 1\,0\,1\,0\,0\,1\,0\,0\,1 \end{array}$$

Interrupts (Exception) Example

// Example uses KEY interrupts. The ISR shows on HEX2-0 the name of the
// inst. that will get executed next on return from the interrupt.



```
.global _start
_start:
        MOV   R1, #0b11010010 // I=1 (disabled), mode = IRQ
        MSR   CPSR, R1
        LDR   SP, =0x40000 // set IRQ sp
        MOV   R1, #0b11010011 // I=1 (disabled), mode = SVC
        MSR   CPSR, R1
        LDR   SP, =0x20000 // set SVC sp
        BL    CONFIG_GIC //enable interrupts from KEYS in GIC
        LDR   R0, =0xFF200000 // KEY port
        MOV   R1, #0xF
        STR   R1, [R0, #8] // write to KEY interrupt mask register.
        MOV   R0, #0b01010011 // I=0, mode = SVC
        MSR   CPSR, R0 //enable IRQ exception for ARM
LOOP:   AND   ..
        EOR   ..
        ORR   ..
        AND   ..
        EOR   ..
        ORR   ..
        B     LOOP
```

Some random operand.

CPSR

N	Z	C	V	//////	I	F	T	Mode.

user mode	10000
FIQ mode	10001
IRQ mode	10010
Supervisor mode	10011
Abort mode	10111
undefined mode	11011

- we need to provide an IRQ handler that checks for interrupts from the KEYs, and an interrupt service routine for the KEYs. The ISR has to check which instruction in the loop will be executed on returned from the exception (AND, EOR, ORR, B). Display on HEX_{2-0}

Requirement : exception vector table, IRQ handler, KEY_ISR

```
.section  .vectors, "ax"
    B       _start          // reset.
    .word   0               // undefined inst.
    .word   0               // SVC exception. (maybe on exam)
    .word   0               // abort instruction
    .word   0               // abort data
    .word   0               // unused.
    B       SERVICE_IRQ
    .word   0               // FIQ

SERVICE_IRQ: PUSH   {R0~R5, LR}
    // Read from GIC to get the interrupt ID
    LDR     R4, =0xFFFEC100
    LDR     R5, [R4, 0xC]   // read ICCIAR.
```

GIC Registers used above

Address	31 ... 10	9	8	7 ... 1	0	Register name
0xFFFEC100	Unused				E	ICCICR
0xFFFEC104	Unused			Priority		ICCPMR
0xFFFEC10C	Unused			Interrupt ID		ICCIAR
0xFFFEC110	Unused			Interrupt ID		ICCEOIR

```
    CMP     R5, #73         // check if it's KEYs ?
CH_NO:  BNE     CH_NO
    LDR     R0, [LR, #-4]   // read machine code of the
                            // inst. that will be executed
                            // on return from IRQ
    BL      KEY_ISR         // print to HEX-0
    STR     R5, [R4, 0x10]  // clear interrupt from ICCEOIR
    POP     {R0-R5, LR}
    SUBS    PC, LR, #4      // return from exception.
```

OP_CODE:
0000 AND
0001 EOR
1100 ORR

```
//machine code is in R0, we "print" to HEX2-0
KEY_ISR:    PUSH    {LR}
            LSR     R0, #21
            AND     R0, #0xF    //isolated the Opcode.
            BL      DISPLAY_OP
END_ISR:    LDR     R0, =0xFF200050 //KEY port
            mov     R2, #0xF
            STR     R2, [R0, #0xC] //clear interrupt by writing
                                   //to Edge_capture.
            POP     {PC}    // return to SERVICE_IRQ

//Op-code is in R0
DISPLAY_OP: CMP     R0, #0b0000 //is it AND?
            BNE     NEXT
            LDR     R0, = 0b01110111010101010.... // "H, -, C"
            B       STORE
NEXT:       ... check for EOR, then ORR.....
            :
STORE:      LDR     R1, =0xFF200020 //HEX DISPLAY
            STR     R0,[R1]
            mov     PC, LR
```

- convert IEEE float to decimal: 0x42B20000

0 | 10000101 | 01100100000....

133 - 127 = 6

$\therefore f = 1.01100100... \times 2^6$

$= 1011001.000\ldots$

$= 89.0$

- convert from decimal to IEEE float : -44.5

$44 = 32+8+4 = 2^5+2^3+2^2 = 101100$ $\qquad 1/2 = 0.1$

$\therefore f = 101100.1$

$= 1.0110001 \times 2^5$

$\therefore f = 10000100\,01100010\ldots$

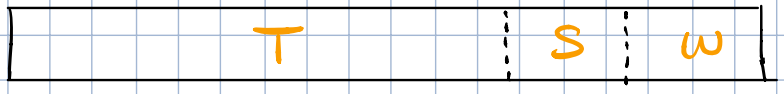
C 2 3 2

$E = 5+127 = 132$

$= 10000100$

$f = (C2320000)_{16}$

- Caches: 2^{32}-byte memory space, 48 KByte cache, 64 bytes/line, 3-way set associative.



- how many bits are needed for T, S, W.

W: 64 bytes/line ÷ 4 bytes/word = 16 words/line

$W = 4$ bits

S: 48 KByte ÷ 64 bytes/line $= 48 \times 2^4$ lines.

$48 \times 2^4 \div 3 = 16 \times 2^4 = 2^4 \times 2^4 = 2^8$ sets.

$S = 8$ bits.

T: $32 - 8 - 4 = 20$ bits.