Tomas Belmar

@20513-5338

1a. -49_{10} in base 2

49	1
24	0
12	0
6	0
3	1
1	1
0	0

$49_{10} = 0110001_2$

Flip bits and add 1

$-49_{10} = 1001111_2$

Sign extend

$-49_{10} = 1111\,1100\,1111_2$

$-49_{10} = FCF_{16}$

b. 113_{10} in base 2

113	1
56	0
28	0
14	0
7	1
3	1
1	1
0	0

$113_{10} = 01110001_2$

Zero extend

$113_{10} = 0000\,0111\,0000_2$

$113_{10} = 071_{16}$

2. int p = x & 0x3E000 ##preserves only the relevant bits
p = p >> 13 ##moves bits to correct position

z &= 0xFFFFFFE0 ##makes last 5 bits of z 0
z |= p ##copies last 5 bits of p to z

0x3

p = y & ~~0xFFFFFFE0~~ ##preserves relevant bits
p = p << 5 ##shifts bits to correct position
z &= 0xFFFFFF1F ##deletes irrelevant bits
z |= p ##replaces index 5 to 7 of z with p

Combines into ---

z = (z & 0xFFFFFF00) | ((x & 0x3E000) >> 13) | ((y & 0x3) << 5)

Tomer Belmar

@20573-5338

3. void adder(const int* x, const int* y, int * z) {
    *z = *y + *x;
}

int foo(int a){
    const int k=10;
    ~~[illegible]~~
    ~~[illegible]~~
    int c;
    ~~[illegible]~~
    adder(&a, &k, &c);
    return c;
}

4. 0x 2 d 2 8 f f f 9

0010 1101 0010 1000 1111 1111 1111 1001

op: 11 rs:9 rt:A imm: -7

sltiu, I type

sltiu rt, rs, imm

$8 = $10 $9 = $21

sltiu $10, $21, -7

5. op: 2 → 000010 ← 6 bits

address: 0x00400024 → divide by 4 → 0x00100009

~~[illegible]~~

0x00100009 → 0000.0100 0001 0000 0000 0000 0000 1010

26 bits

Instruction: 00001000 0001 0000 0000 0000 0000 1010

: 0x08100009

```
#include <math.h>

6. void square_reverse(double *x, double *y, const int len) {
       y += len-1;
       for(int i=0; i<len; i++)
           *(y--) = pow(*(x++), 2);
   }
```