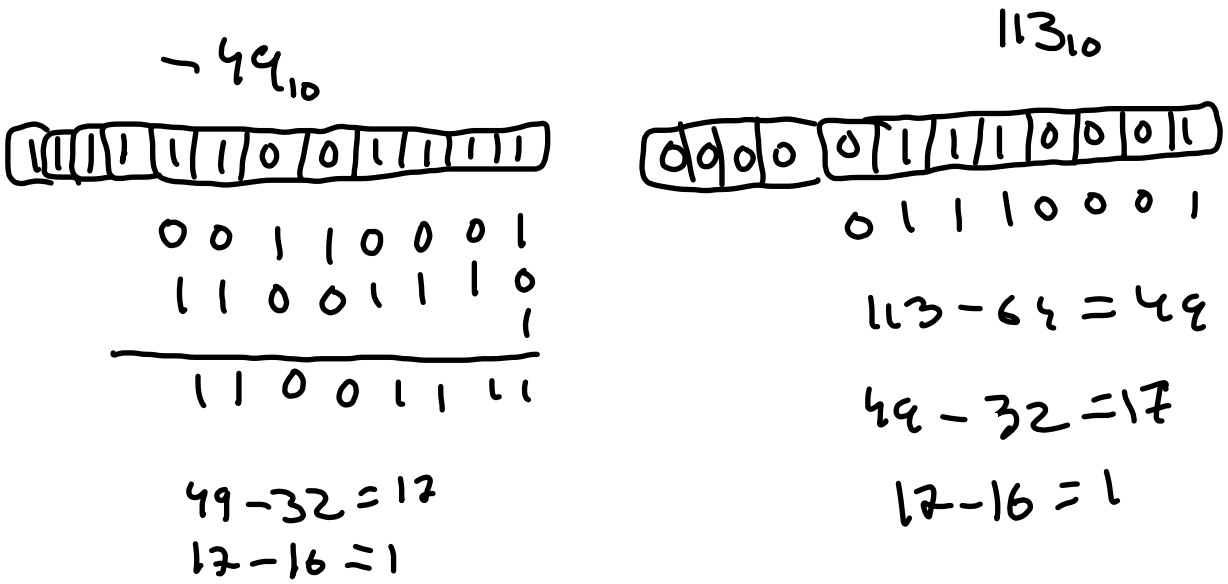
Exercises

1. Assume that the two numbers -49_{10} and 113_{10} are encoded as 8-bit signed values in two's complement form. Sign extend *and* zero extend each of them into 12-bit values. Do it by hand and answer in hexadecimal form. Show the main steps of your solution.

   Your solution:



2. Assume that you have a C program with signed integer (`int`) variables `x`, `y`, and `z`. All variables contain some arbitrary values. Write a C-statement that extracts the bits with index 17 to 13 from `x` and places them as the least significant bits in `z`, and extracts the least 3 significant bits of `y` and places them in the bits with index 7 to 5 in `z`. No other bits of `z` should be changed, besides the 8 bits that were extracted from `x` and `y`. Note that the bit index 0 is the least significant bit. Your answer should contain one single C statement together with short notes of what the different parts of the statement do.

   Your solution:

```
int main()
{
  int x=0x0003E000, y=0x00000001, z=0x00000000;
  z=(z & 0xFFFFFF00)(((x & 0x0003E000)>>13)
    |((y & 0x00000007)<<5));
Printf("Storleken på int är: %d, z);
```

för att lägga in de bitarna i z

Stifftar x, y 13 respektive 5 steg

3. Write down the function body of the two following C functions. Function `adder` should add together the two integer values that the pointers `x` and `y` points to, and then write the result to where `z` points to. Function `foo` should use function `adder` to add together `a` and `k` and then return the resulting value. For instance, if expression `foo(7)` is executed, value `17` should be returned.

   Your solution:

```
void adder(const int *x, const int *y, int *z){
*z = *x + *y

}

int foo(int a){
  const int k = 10;
int z = 0;
adder(&a, &k, &z)
return z;

}
```

4. Write out the MIPS assembly instruction that has the machine code `0x2d28fff9`. You should include the main steps of how you computed your solution.

   Your solution:

0x2d28fff9

rs rt

0010 | 1101 | 0010 | 1000 | 1111 | 1111 | 1111 | 1001

13 9 8 -7

Sltiu rt rs Imm

$var:

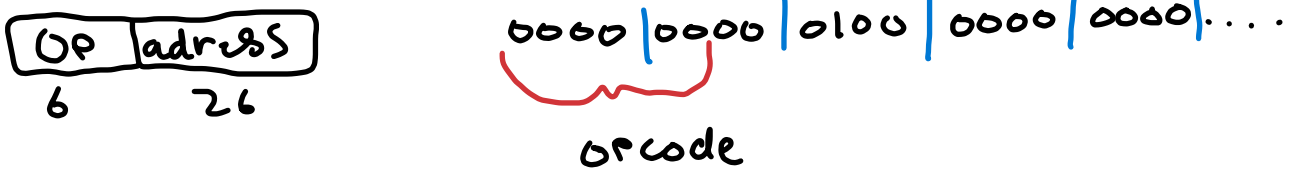
Sltiu $t0, $t1, -7

5. Assume that the MIPS machine encoded word of the following instruction is located at address `0x00400000` in the program code memory.

```
j       foo
```

Assume further that label `foo` is located at address `0x0040002c`. What is then the machine encoding of the jump instruction? Include a short explanation of the different parts of encoding.

Your solution: 0x00400000



6. Create a C function named `square_reverse` with three parameters. The two first parameters are 64-bit floating-point pointers `x` and `y`, and the third parameter is an integer parameter called `len`. The function must not return any value. Pointer `x` points to an array of length `len` of floating-point values. The function reads out each element of the array, computes the square value of the element (x^2) and then writes back the result into the array `y` in reverse order. That is, the output array `y` has also the length `len`.

For example, if we have the following declarations

```
double in[] = {11.0, 20.0, 100.0};
double out[3];
```

a function call `square_reverse(in,out,3);` should result in that the three elements 10000.0, 400.0, and 121.0 are the content of `out`. Note that your function should also declare parameters as `const` when appropriate.

Your solution:

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

Corrected by _____. Total number of points: _____