```c
// lab00, lab procedure, tasks 1 & 3

// source file

/*
 * this file contains the implementation of the functions relevant for
 * lab00.
 * The declarations of the functions are in the corresponding header
 * file (functions.h).
*/

// the header file needs to be included
#include "functions.h"

/* task 1
 * The function print_bits() accepts an input of type uint16_t
 * (arg_word) and has no return value (void).
 * The function simply writes the binary and hexadecimal number of the
 * input to the terminal.
*/
void print_bits(uint16_t arg_word)
{
  // only a simple implementation of a for loop is given
  // you are free to use it or solve the problem in another way
  uint16_t one = 1;
  uint16_t array[16];
  uint16_t arg_word2 = arg_word;
  int i;
  for (i = 0; i <= 15; i++)
  {
      if (arg_word & one == 1)
      {
        array[15-i] = 1;
      }
      else
      {
        array[15-i] = 0;
      }

      arg_word = arg_word >> 1;
  }

  printf("hex: 0x%x, bin: ",arg_word2);

  for(i=0;i<4;i++)
  {
    int k = 4*i;
    printf("%d%d%d%d ", array[k],array[k+1],array[k+2],array[k+3]);
  }


  /*int i;
  for(i = 0; i <= 0; i++)
  {
    printf("i = 0x%x\n", i);
  }*/
}

/* task 3
 * The function bit_merge() accepts two uint16_t as inputs (lsb and msb)
* and combines them to a uint32_t number by merging them.
* The return value is a uint32_t number.
*/
uint32_t bit_merge(uint16_t lsb, uint16_t msb)
{
  // for the moment, the function only returns 0
  uint32_t lsbb = lsb;
  uint32_t msbb = msb;
  msbb = msbb << 16;
  uint32_t result = lsbb | msbb ;
  return result;
}

```

```c
// lab00, lab procedure, tasks 1 & 3

// header file

/*
 * This file contains the declarations of the relevant functions.
 * The implementation of each function can be found in the source file
 * (functions.c)
*/

// some standard libraries are already inluded
#include <stdio.h>
#include <stdint.h>

/* task 1
 * The function print_bits() accepts an input of type uint16_t and has
 * no return value (void).
 * The function simply writes the binary and hexadecimal number of the
 * input to the terminal.
*/
void print_bits(uint16_t arg_word);

/* task 3
 * The function bit_merge() accepts two uint16_t as inputs and combines
 * them to a uint32_t number by merging them.
 * The return value is a uint32_t number.
 */
 uint32_t bit_merge(uint16_t lsb, uint16_t msb);

```

```c
// lab00, lab procesure, task 4



//This solution also contains the prelab Bonus exercise 5

#include "functions.h"


int main(int argc, char *argv[])
{

  while(1)
  {
    uint16_t num1;
    uint16_t num2;


    printf("write first hexadecimal number");
    scanf("%x",& num1);

    printf("write second hexadecimal number");
    scanf("%x",& num2);

    if (num1 == 0 && num2 == 0){
      break;
    }
    uint16_t sum=num1+num2;

    uint32_t merge=bit_merge(num1,num2);
    printf("merging 0x%x and 0x%x results in 0x%x\n",num1,num2,merge);

    printf("the sum is ");
    print_bits(sum);
    printf("\n \n");

}
  return 0;
}

```

```c
// lab00, lap procedure, task 2

// source file for the program sum_numbers

/*
 * This file is used to generate an executable program that prints the
 * sum of the two numbers in binary and hexadecimal format.
*/

// Since we will be using our own functions, we need to add the header
// file where the functions are declared.
// Keep in mind that the header file already includes standard libraries
// so they do not to be included here.
#include "functions.h"

 // main function
int main()
{
  uint16_t num1;
  uint16_t num2;
  uint16_t sum;

  printf("write first hexadecimal number");
  scanf("%x",& num1);

  printf("write second hexadecimal number");
  scanf("%x",& num2);

  sum = num1 + num2;

  print_bits(sum);

  return 0;
}

```

```
# when running make, all programs are generated
all: sum_numbers hello_world manipulate_two_numbers

# compile hello_world.c and make executable program
hello:
  gcc -o hello_world hello_world.c

# linking of sum_numbers.o with functions.o
sum_numbers: sum_numbers.o functions.o
  gcc sum_numbers.o functions.o -o sum_numbers

# compile sum_numbers.c
sum_numbers.o: sum_numbers.c
  gcc -c sum_numbers.c

# linking of manipulate_two_numbers.o with functions.o
manipulate_two_numbers: manipulate_two_numbers.o functions.o
  gcc manipulate_two_numbers.o functions.o -o manipulate_two_numbers

# compile manipulate_two_numbers.c
manipulate_two_numbers.o: manipulate_two_numbers.c
  gcc -c manipulate_two_numbers.c

# compile functions.c
functions.o: functions.h functions.c
  gcc -c functions.c

# remove generated files and programs
clean:
  rm functions.o sum_numbers.o sum_numbers hello_world

```