Lab 1:

Decimal to binary and binary to decimal conversion using c language

Program:

// program to convert binary into decimal and decimal into binary

#include<stdio.h>

#include<math.h>

void binary\_to\_decimal(int);

void decimal\_to\_binary(int);

int main()

{

int n, bnum, dnum;

do {

printf("Please select the number 1 or 2 \n");

printf("1. Number conversion form binary to decimal. \n");

printf("2. Number conversion form decimal to binary. \n");

scanf("%d", &n);

switch (n)

{

case 1:

printf("Enter the binary number: ");

scanf("%d", &bnum);

binary\_to\_decimal(bnum);

break;

case 2:

printf("Enter the decimal number: ");

scanf("%d", &dnum);

decimal\_to\_binary(dnum);

break;

defult:

printf("Please select the valid number.\n");

break;

}

} while(1);

return 0;

}

void binary\_to\_decimal(int n)

{

int dec =0;

int power = 0;

int digit;

while (n > 0)

{

digit = n%10;

dec= dec+ digit\*pow(2,power);

n=n/10;

power++;

}

printf("Decimal number is : %d \n", dec);

}

void decimal\_to\_binary(int n)

{

int bin[64];

int index=0;

while (n!=0)

{

bin[index] = n%2;

n=n/2;

index++;

}

//display

printf("Binary number is : ");

for(int i= index-1; i >= 0 ; i--)

{

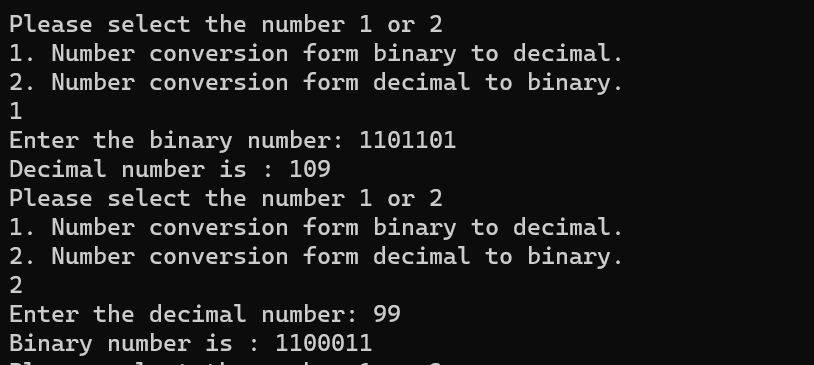
printf("%d",bin[i]);

}

printf("\n");

}

Output:



LAB 2:

Addition and subtraction of two unsigned interger binary number using c language.

Program:

#include <stdio.h>

#include <stdlib.h>

unsigned int bin\_addition(unsigned int bin1, unsigned int bin2)

{

unsigned int carry = 0, result = 0, bit\_pos = 1;

while (bin1 != 0 || bin2 != 0)

{

unsigned int bit1 = bin1 % 2;

unsigned int bit2 = bin2 % 2;

unsigned int sum = bit1 + bit2 + carry;

carry = sum / 2;

result = result + (sum % 2) \* bit\_pos;

bin1 = bin1 / 2;

bin2 = bin2 / 2;

bit\_pos = bit\_pos \* 2;

}

if (carry)

{

result = result + bit\_pos;

}

return result;

}

unsigned int bin\_subtraction(unsigned int bin1, unsigned int bin2)

{

unsigned int borrow = 0, result = 0, bit\_pos = 1;

while (bin1 != 0 || bin2 != 0)

{

unsigned int bit1 = bin1 % 2;

unsigned int bit2 = bin2 % 2;

bit1 = bit1 - borrow;

if (bit1 < bit2)

{

bit1 += 2;

borrow = 1;

}

else

{

borrow = 0;

}

unsigned int difference = bit1 - bit2;

result = result + difference \* bit\_pos;

bin1 = bin1 / 2;

bin2 = bin2 / 2;

bit\_pos = bit\_pos \* 2;

}

return result;

}

void decimalToBinary(unsigned int num)

{

if (num == 0)

{

printf("0");

return;

}

int binary[32];

int index = 0;

while (num > 0)

{

binary[index++] = num % 2;

num /= 2;

}

for (int i = index - 1; i >= 0; i--)

{

printf("%d", binary[i]);

}

}

int main()

{

char bin1\_str[32], bin2\_str[32];

printf("Enter the first binary number: ");

scanf("%s", bin1\_str);

printf("Enter the second binary number: ");

scanf("%s", bin2\_str);

unsigned int bin1 = strtoul(bin1\_str, NULL, 2);

unsigned int bin2 = strtoul(bin2\_str, NULL, 2);

unsigned int sum = bin\_addition(bin1, bin2);

printf("Sum: ");

decimalToBinary(sum);

printf("\n");

unsigned int diff = bin\_subtraction(bin1, bin2);

printf("Difference: ");

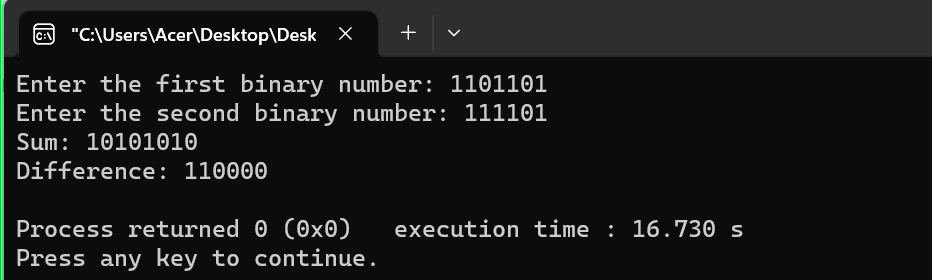
decimalToBinary(diff);

printf("\n");

return 0;

}

Output:



Lab 3:

Multiplication and division of two unsigned integers binary number using c language.

#include <stdio.h>

#include <stdlib.h>

unsigned int bin\_multiplication(unsigned int bin1, unsigned int bin2) {

unsigned int multiplier = bin2;

unsigned int result = 0;

while (multiplier > 0) {

if (multiplier & 1) {

result += bin1;

}

bin1 <<= 1;

multiplier >>= 1;

}

return result;

}

unsigned int bin\_division(unsigned int dividend, unsigned int divisor) {

if (divisor == 0) {

printf("Error: Division by zero is not allowed.\n");

return 0;

}

unsigned int quotient = 0;

unsigned int temp = 1;

unsigned int temp\_divisor = divisor;

while (dividend >= divisor) {

divisor <<= 1;

temp <<= 1;

}

while (temp > 1) {

divisor >>= 1;

temp >>= 1;

if (dividend >= divisor) {

dividend -= divisor;

quotient += temp;

}

}

return quotient;

}

void decimalToBinary(unsigned int num) {

if (num == 0) {

printf("0");

return;

}

int binary[32];

int index = 0;

while (num > 0) {

binary[index++] = num % 2;

num /= 2;

}

for (int i = index - 1; i >= 0; i--) {

printf("%d", binary[i]);

}

}

int main() {

char bin1\_str[32], bin2\_str[32];

printf("Enter the first binary number: ");

scanf("%s", bin1\_str);

printf("Enter the second binary number: ");

scanf("%s", bin2\_str);

unsigned int bin1 = strtoul(bin1\_str, NULL, 2);

unsigned int bin2 = strtoul(bin2\_str, NULL, 2);

unsigned int product = bin\_multiplication(bin1, bin2);

printf("Product: ");

decimalToBinary(product);

printf("\n");

unsigned int quotient = bin\_division(bin1, bin2);

printf("Quotient: ");

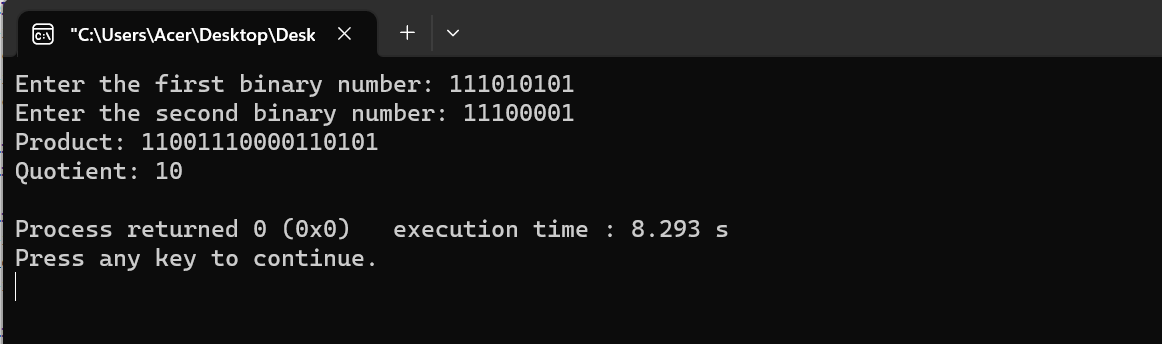
decimalToBinary(quotient);

printf("\n");

return 0;

}

Output:



Lab 4:

Familiarizing with the syntax, data types, and operators of VHDL.

Program:

Objective:

To familiarize with the syntax, data types and operators of VHDL.

Description:

In this lab, we will explore the basic syntax, data type and operators used in VHDL ( Very high speed integration circuit Hardware Description Language ). It is hardware description language used for designing and modeling digital system.

Tasks:

1. VHDL code snippet to declare a signal name “input\_signal” of type std\_logic.

* Signal input\_signal.std\_logic;

1. VHDL code snippet to declare constant name “clock period” of type time and initialize with a value 10ns.

* Constant clock\_period:time =10ns;

1. VHDL code snippet to declare a variable named “counter” of type integer and initialize it with a value of 0.

* Variable counter:integer =0;

1. VHDL code snippet to define an entity “and\_gate” with two input ports (A & B) of type std\_logic and an output port name “out” of type std\_logic.

* entity and\_gate is

port (

A,B: in std\_logic;

Out: out std\_logic;

);

end entity and\_gate;

1. VHDL code snippet to instantiate “and\_gate” entity and connect its input ports A and B to other signal or constants.

* and\_gate\_inst: entity word and\_gate

port map (

A=>input\_signal,

B=> ‘I’ ---Example connection to a constant value.

out => output\_signal

);

1. VHDL code snippets to define an architecture name “behav” for “and\_gate” entity, where the output “out” is assigned the logic AND operation of inputs A and B.

* architecture behav of and\_gate is

begin

out=> A and B

end of architecture behav;

conclusion:

By completing the above tasks, we have gained familiarity with VHDL syntax, data types and operators.

Lab 6:

Design half adder and full adder.

Aim:

To design and implement half adder and full adder using VHDL.

Description:

In digital electronic, adder circuits are fundamentals building blocks used for performing arithmetic operations. The half adder is combinational circuit that add two single bit binary number. On the other hand, full adder is an extension of half adder capable of adding three single bit binary number( two inputs and a carry in) to produce two outputs i.e sum and carry out. The concept of both half adder and full adder are described below with aid of block diagram, truth table and circuit implementation.

Block diagram:

Truth table:

Circuit implementation:

Program for (half adder)

library IEEE;

use IEEE.STD\_LOGIC\_116H.all;

entity half\_adder is

Port (

a: in STD\_LOGIC;

b: in STD\_LOGIC;

sum : out STD\_LOGIC;

carry: out STD\_LOGIC;

);

end half\_adder;

architecture Behaviorial of half\_adder is

begin

sum <= a XOR b;

carry <= a and b;

end Behaviorial;