Readme File:

**PES PROJECT - 1 README FILE**

Team Members : Atharv Desai ([atharv.desai@colorado.edu](mailto:atharv.desai@colorado.edu))

Suraj Thite ([suraj.thite@colorado.edu](mailto:suraj.thite@colorado.edu))

This is a readme file for the first project assignment in the Principles of Embedded Software Course for FALL '19.

The below enumerated files are contained in the repository

1 ( Executable File for the First Program Assigned).

1.c ( C Source Code File for the First Program Assigned).

Program1Output.out ( Piped output from Executable file from the bash terminal). 4 2 ( Executable File for the First Program Assigned).

2.c ( C Source Code File for the First Program Assigned).

Program2Output.out ( Piped output from Executable file from the bash terminal). 7 3 ( Executable File for the First Program Assigned).

3.c ( C Source Code File for the First Program Assigned).

Program3Output.out ( Piped output from Executable file from the bash terminal). 10.Project1.pdf ( Project Objectives Assigned in pdf Format). 11.Project1-PES2019.pdf ( Codes in pdf format with their console output).

**INSTALLATION AND EXECTUION NOTES**

This code is build and tested on enviornment mentioned below

gcc (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0 Copyright (C) 2017 Free Software Foundation, Inc. This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

The editor used to build the code is gedit version 2.3.8.1 on Linux Mint Machine.

To execute the executable file simply type ./(filename).

To compile and create executable file , type gcc (filename).c -o (filename) -Wall - Werror –lm

Code 1:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Author : 1) Atharv Desai(atharv.desai@colorado.edu)

2) Suraj Thite(suraj.thite@colorado.edu)

Problem Statement 1: Create a C program that will take as input a numeric value, a radix, and an operand size.

Your program will need to error check for invalid inputs. Radix values are limited to 8, 10, 16.

Operand size values are limited to bit sizes of 4,8,16. Numeric input must be valid for the operand size (that is, a number larger than the operand size should throw an error).

For each input line read, return the following output to the console (for capture into an output file) – the following

output is for the input -6, 10, 4 (numeric value, radix of the value, operand bit size). Any values that cannot be calculated

should show the word Error for a result in the table. Note that the first four outputs in the table display absolute values,

the last three are signed

Input Set {Value, Radix, Operand Size} {-6, 10, 4}, {-6, 9, 4}, {-6, 10, 5}, {-9, 10, 4}, {237, 10, 8}, {0354, 8, 8},

{0xEB, 16, 8}, {-125, 10, 8}, {65400, 10, 8}, {65400, 10, 16}, {-32701, 10, 16}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<math.h>

#include <stdio.h>

#include<string.h>

#include<stdint.h>

#include<stdlib.h>

// Declaration of all the functions

void print\_input(int,int,int);

void hex\_to\_dec(int,int,int);

void decimal\_bin(int num, int n);

void compute\_max(int n);

void compute\_min(int n);

void compute\_max\_decimal(int n);

void compute\_max\_hex(int n);

void compute\_min\_hex(int n);

void compute\_max\_oct(int n);

void compute\_min\_oct(int n);

void compute\_min\_dec();

void compute\_oct(int);

void compute\_hex(int);

void compute\_ones(int);

void compute\_twos(int);

void sign\_magni(int,int);

void compute\_signed\_max(int);

void compute\_signed\_min1(int);

void compute\_signed\_min2(int);

void compute\_signed\_min3(int);

char bin\_prefix[] = "ob";

int bin[16] = {0};

int ones[16] = {0};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: Main Parameters: void Description: Function from where execution of any C program begins. \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void)

{

const int dataset[11][3] = {{-6, 10, 4}, {-6, 9, 4}, {-6, 10, 5}, {-9, 10, 4}, {237, 10, 8}, {125, 8, 8}, {0xEB, 16, 8}, {-125, 10, 8}, {65400, 10, 8}, {65400, 10, 16}, {-32701, 10, 16} };

int i=0;

int deci = 0;

int bin[16] = {0};

int err\_flag;

int negative\_flag;

for (i=0;i<11;i++)

{ negative\_flag = 0;

if( dataset[i][0]<0) // Set a flag If value is negative

{ negative\_flag =1;

}

print\_input(dataset[i][0],dataset[i][1],dataset[i][2]); //Printing Input values

if(dataset[i][1]!=8 && dataset[i][1]!=10 && dataset[i][1]!=16 )

{ //Error handling for checking radix values

printf("\n Error: Radix value is wrong for the input set %d %d %d \nRadix value should be 8,10 or 16 but here it's %d \n",dataset[i][0],dataset[i][1],dataset[i][2],dataset[i][1]);

continue;

}

if(dataset[i][2]!=4 && dataset[i][2]!=8 && dataset[i][2]!=16 )

{ //Error handling for checking Operand values

printf("\n Error: Operand value is wrong for the input set %d %d %d \nOperand value should be 4,8 or 16 but here it's %d \n",dataset[i][0],dataset[i][1],dataset[i][2],dataset[i][2]);

continue;

}

deci= abs(dataset[i][0]); // To convert all the values in input datset to decimal form

if (deci> pow(2, dataset[i][2])){ //Error handling to check if the value is greater than operand size

printf("\n Error since operand size limit exceeded\n");

continue;

}

if ((dataset[i][0]> pow(2, (dataset[i][2] - 1))) // Error handling to check if value greater than operand size considering signed bit

||(dataset[i][0] < - pow(2, (dataset[i][2] - 1))))

err\_flag = 1;

else

err\_flag = 0;

printf("Output:\t\tValue\t\t\tMaximum\t\t\tMinimum\t\t\n"); //Printing Output Coloumn headings

// Calling all the functions starts here. Functions created for several tasks to keep the code modular in structure

decimal\_bin(deci,dataset[i][2]); // to convert our decimal value to binary

compute\_max(dataset[i][2]); // to calculate the maximum value in binary for the given operand size

compute\_min(dataset[i][2]); // to calculate the minimum value in binary for the given operand size

printf("\nDec(abs): %d",deci);

compute\_max\_decimal(dataset[i][2]); // to calculate the maximum value in decimal for the given operand size

compute\_min\_dec(dataset[i][2]); // to calculate the minimum value in decimal for the given operand size

compute\_oct(deci); // to convert our decimal value to octal

compute\_max\_oct(dataset[i][2]); // to calculate the maximum value in octal for the given operand size

compute\_min\_oct(dataset[i][2]); // to calculate the minimum value in octal for the given operand size

compute\_hex(deci); // to convert our decimal value to hexadecimal

compute\_max\_hex(dataset[i][2]); // to calculate the maximum value in hexadecimal for the given operand size

compute\_min\_hex(dataset[i][2]); // to calculate the minimum value in hexadecimal for the given operand size

printf("\n1's Complement\t");

if(err\_flag == 0)

compute\_ones(dataset[i][2]); // Compute one's complement if value in limits

else{

printf("Error"); // Throw error if the value is out of limits

if(dataset[i][2]==8) { printf("\t"); } //spacing for binary for given operand size 8 in table

if(dataset[i][2]==16) { printf("\t\t"); } //spacing for binary for given operand size 16 in table

}

compute\_signed\_max(dataset[i][2]); // compute signed max ones complement value for given operand size

compute\_signed\_min1(dataset[i][2]); // compute signed min ones complement value for given operand size

printf("\n2's Complement\t");

if(err\_flag == 0)

compute\_twos(dataset[i][2]); // Compute two's complement if value in limits

else

{ printf("Error"); // Throw error if the value is out of limits

if(dataset[i][2]==8) { printf("\t"); } //spacing for binary for given operand size 8 in table

if(dataset[i][2]==16) { printf("\t\t"); } //spacing for binary for given operand size 16 in table

}

compute\_signed\_max(dataset[i][2]); // compute signed max two's complement value for given operand size

compute\_signed\_min2(dataset[i][2]); // compute signed min two's complement value for given operand size

printf("\nSign magnitude\t");

if(err\_flag == 0)

sign\_magni(negative\_flag,dataset[i][2]); // Compute sign magnitude if value in limits

else

{ printf("Error"); // Throw error if the value is out of limits

if(dataset[i][2]==8) { printf("\t"); } //spacing for binary for given operand size 8 in table

if(dataset[i][2]==16) { printf("\t\t"); } //spacing for binary for given operand size 16 in table

}

compute\_signed\_max(dataset[i][2]); // compute signed max signed magnitude value for given operand size

compute\_signed\_min3(dataset[i][2]); // compute signed min signed magnitude value for given operand size

printf("\n");

}

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: print \_input Parameters: Value,radix,operand Description:Function for Printing Input values \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void print\_input(int val,int radix,int operand)

{

if(radix==8){

printf("\nInput:\t\tValue:%o\t\tRadix:%d\t\tOperand Size:%d\t\t\n",val,radix,operand); //print octal input values

}

if(radix==16){

printf("\nInput:\t\tValue:%x\t\tRadix:%d\t\tOperand Size:%d\t\t\n",val,radix,operand); //print hexadecimal input values

}

if(radix==10){

printf("\nInput:\t\tValue:%d\t\tRadix:%d\t\tOperand Size:%d\t\t\n",val,radix,operand); //print other input values

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: decimal\_bin Parameters: Decimal value,operand Description:Function to convert decimal to binary \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void decimal\_bin(int num, int n) // Function to convert decimal values to binary

{

int binnum[n];

int i, j;

for(i=0;i<n;i++)

{

binnum[i]=num%2;

num=num/2;

}

printf("Binary(abs)\t0b");

for(i = n-1, j= 0; i>=0, j < n;i--,j++)

{

bin[j] = binnum[i]; // storing binary value s in global array

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

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_max Parameters: operand Description:to calculate the max binary value for the given operand size \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_max(int n) //Function to calculate the maximum value in binary for the given operand size

{

int i;

int max\_value[n];

for(i=0;i<n;i++)

{

max\_value[i]=1;

}

if(n==4) { printf("\t\t\t0b"); } //spacing for binary

if(n==8) { printf("\t\t0b"); } //spacing for binary

if(n==16) { printf("\t0b"); } //spacing for binary

for ( i=0;i<n;i++)

{

printf("%d",max\_value[i]);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_min Parameters: operand Description:to calculate the min binary value for the given operand size \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_min(int n) //Function to calculate the minimum value in binary for the given operand size

{

int i;

int min\_value[n];

for(i=0;i<n;i++)

{

min\_value[i]=0;

}

if(n==4) { printf("\t\t\t0b"); } //spacing for binary

if(n==8) { printf("\t\t0b"); } //spacing for binary

if(n==16) { printf("\t0b"); } //spacing for binary

for ( i=0;i<n;i++)

{

printf("%d",min\_value[i]);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_max\_decimal Parameters: operand Description:to calculate the max value for the given operand size \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_max\_decimal(int n) //Function to calculate the maximum value in decimal for the given operand size

{

int i;

uint16\_t max\_value = 0;

for(i=0;i<n;i++)

{

max\_value = max\_value + pow(2,i);

}

printf("\t\t\t%-5d",max\_value);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_max\_hex Parameters: operand Description:to calculate the max hex value for the given operand size \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_max\_hex(int n) //Function to calculate the maximum value in hex for the given operand size

{

int i;

uint16\_t max\_value = 0;

for(i=0;i<n;i++)

{ max\_value = max\_value + pow(2,i);

// printf("%d\n",pow(2,i));

}

printf("\t\t\t0x%-1X",max\_value);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_min\_hex Parameters: operand Description:to calculate the min hex value for the given operand size \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_min\_hex(int n) //Function to calculate the minimum value in hex for the given operand size

{ int i;

unsigned short int min\_value=0;

printf("\t\t\t0x%-1X",min\_value);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_max\_oct Parameters: operand Description:to calculate max octal value for the given operand size \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_max\_oct(int n) //Function to calculate the maximum value in octal for the given operand size

{ int i;

uint16\_t max\_value = 0;

for(i=0;i<n;i++)

{ max\_value = max\_value + pow(2,i);

}

printf("\t\t\t%1o",max\_value);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_min\_oct Parameters: operand Description:to calculate min octal value for the given operand size \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_min\_oct(int n) //Function to calculate the minimum value in octal for the given operand size

{ int i;

unsigned short int min\_value=0;

printf("\t\t\t%-1o",min\_value);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_min\_dec Parameters: None Description:to calculate min decimal value for the given operand size \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_min\_dec() //Function to calculate the minimum value in decimal for the given operand size

{ int i;

unsigned short int min\_value=0;

printf("\t\t\t%-1d",min\_value);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_oct Parameters: Decimal value Description: to print octal value for the given decimal value \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_oct(int n) //Function to print our decimal value in octal

{

printf("\nOctal(abs): %-1o",n);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_hex Parameters: Decimal value Description: to print hex value for the given decimal value \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_hex(int n) //Function to print our decimal value in hex

{

printf("\nHex(abs): 0x%-2X",n);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_ones Parameters: Operand Description:to calculate ones complement value by using array to store bits \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_ones(int n) // Function to compute ones complement

{

int i=0;

while(i<n) {

ones[i]= !bin[i]; // complementing each bit from global binary value array and store it in other global array

i++;

}

printf("0b");

for(i=0; i<n; i++)

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

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_twos Parameters: Operand Description:to calculate twos complement value by using array to store bits \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_twos(int n) // Function to compute twos complement

{ // Reference: codeforwin.org/2015/08/c-program-to-find-twos-complement-of-binary-number.html

int i;

int add=0;

int bintwo[n];

if(ones[n-1]==1){ // Checking cases for binary bit,addition and carry and storing bit and updated carry accordingly

bintwo[n-1]=0;

add=1;

}

else{

bintwo[n-1]=1;

add=0;

}

for(i=1;i<n;i++){

if (ones[n-i-1]==1 && add==1){

bintwo[n-i-1]=0;

add=1;

}

else if (ones[n-i-1]==0 && add==1){

bintwo[n-i-1]=1;

add=0;

}

else{

bintwo[n-i-1]=ones[n-i-1];

}

}

printf("0b");

for(i=0; i<n; i++){

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

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: sign\_magni Parameters: Negative flag,Operand Description:to calculate sign magnitude by checking negative flag \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void sign\_magni(int neg,int n) // Function to calculate signed magnitude

{

int i=0;

if(neg==0){

printf("0b");

for(i=0;i<n;i++){

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

}

}

else{

printf("0b");

bin[0]=1;

for(i=0;i<n;i++){

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

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_signed\_max Parameters: Operand Description:to calculate max signed value by setting the array bits high \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_signed\_max( int x) // Function to calculate signed max value

{

int max[x];

max[0]=0;

int i;

for (i=1;i<x;i++)

{

max[i]=1;

}

if(x==4) { printf("\t\t\t0b"); } //spacing for binary

if(x==8) { printf("\t\t0b"); } //spacing for binary

if(x==16) { printf("\t0b"); } //spacing for binary

for(i=0;i<x;i++)

{

printf("%-d",max[i]);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_signed\_min1 Parameters: Operand Description:to calculate min ones complement by setting the array bits low \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_signed\_min1( int x) // Function to calculate signed minimum value for ones complement

{

int min[x];

min[0]=1;

int i;

for (i=1;i<x;i++)

{

min[i]=0;

}

if(x==4) { printf("\t\t\t0b"); } //spacing for binary

if(x==8) { printf("\t\t0b"); } //spacing for binary

if(x==16) { printf("\t0b"); } //spacing for binary

for(i=0;i<x;i++)

{

printf("%-d",min[i]);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_signed\_min2 Parameters: Operand Description:to calculate min twos complement by setting the array bits low \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_signed\_min2( int x) // Function to calculate signed minimum value for twos complement

{

int min[x];

min[0]=1;

min[x]=1;

int i;

for (i=1;i<x-1;i++)

{

min[i]=0;

}

if(x==4) { printf("\t\t\t0b"); } //spacing for binary

if(x==8) { printf("\t\t0b"); } //spacing for binary

if(x==16) { printf("\t0b"); } //spacing for binary

for(i=0;i<x;i++)

{

printf("%-d",min[i]);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: compute\_signed\_min3 Parameters: Operand Description:to calculate min sign magnitude by setting the array bits low \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void compute\_signed\_min3( int x) // Function to calculate signed minimum value for signed magnitude

{

int min[x];

min[0]=1;

min[x]=1;

int i;

for (i=1;i<x-1;i++)

{

min[i]=1;

}

if(x==4) { printf("\t\t\t0b"); } //spacing for binary

if(x==8) { printf("\t\t0b"); } //spacing for binary

if(x==16) { printf("\t0b"); } //spacing for binary

for(i=0;i<x;i++)

{

printf("%-d",min[i]);

}

}

Code 2 :

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Author 1) Atharv Desai (atharv.desai@colorado.edu)

2) Suraj Thite (suraj.thite@colorado.edu)

Problem Statement 2: Write a program that uses a logical expression that tests whether a given character code is a

 lower case

 upper case

 digit

 white space (like null, backspace, space, tabs, etc.)

 or a special character (like ! or >) in ASCII.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <ctype.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: Main Parameters: void Description: Function from where execution of any C program begins. \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void)

{

int a[21]={66,114,117,99,101,32,83,97,121,115,32,72,105,33,7,9,50,48,49,57}; //Taking input

int i=0;

int Code;

for ( i=0;a[i]!=0;i++) {

printf("%c ", a[i]);

}

for (i=0;a[i]!=0;i++){

if( islower(a[i])) //to check if the ASCII char is in lower case

{

printf("\n Code: %d \t \t Type: Lower Case \t \t ASCII char =%c ",a[i],a[i]);

}

if( isupper(a[i])) //to check if the ASCII char is in upper case

{

printf("\n Code: %d \t \t Type: Upper Case \t \t ASCII char =%c ",a[i],a[i]);

}

if( isdigit(a[i])) //to check if the ASCII char is a number between 0-9

{

printf("\n Code: %d \t \t Type: digit \t \t \t ASCII char =%c ",a[i],a[i]);

}

if( ispunct(a[i])) //to check if the ASCII char is a special character

{

printf("\n Code: %d \t \t Type: special char \t \t ASCII char =%c ",a[i],a[i]);

}

if( isspace(a[i]) || a[i]==7) //to check if the ASCII char indicates space

{

printf("\n Code: %d \t \t Type: space \t \t \t ASCII char =space ",a[i]);

}

}

printf(" \n");

return 0;

}

Code 3 :

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Author 1) Atharv Desai (atharv.desai@colorado.edu)

2) Suraj Thite (suraj.thite@colorado.edu)

Problem Statement 3 :

Given the starting integer value 0xCAFE, perform each of these operations in

series, that is, each operation should be performed on the result of the previous function. Print the

results of each function to the command line (to capture as ProgramThree.out).

Question 1.Print the original input in hexadecimal

Question 2.Test if 3 of last 4 bits are on, and print the value in binary (along with the result of the test –

true/false)

Question 3.Reverse the byte order, print the value in hexadecimal

Question 4.Test if 3 of last 4 bits are on, and print the value in binary (along with the result of the test –

true/false)

Question 5.Rotate the value by four bits to the left, print the value in hexadecimal

Question 6.Test if 3 of last 4 bits are on, and print the value in binary (along with the result of the test –

true/false)

Question 7.Rotate the value by eight bits to the right, print the value in hexadecimal

Question 8.Test if 3 of last 4 bits are on, and print the value in binary (along with the result of the test –

true/false)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

void print\_hex(unsigned short int n);

void check\_bin(unsigned short int num);

unsigned int ReverseBytes(unsigned short int val);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: Main Parameters: void Description: Function from where execution of any C program begins. \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

unsigned short int x = 0xCAFE; //Declaration of the starting variable value.

print\_hex(x);

check\_bin(x);

unsigned short int y = ReverseBytes(x); // Reverse the bytes of the input variable

print\_hex(y);

check\_bin(y);

y= (y<<4 | y>>12); //Rotate the value by 4 bits to the left and move them to the rightmost 4 bits retaining all the input values while rotation.

print\_hex(y);

check\_bin(y);

y= (y>>8 | y<<8); ////Rotate the value by 8 bits to the right and move them to the rightmost 8 bits retaining all the input values while rotation.

print\_hex(y);

check\_bin(y);

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: print\_hex Parameters: Hex number Description: Function to print hex value \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void print\_hex(unsigned short int n)

{

printf("\n Hex value %X",n); // function to print the value in hexa decimal.

}

void check\_bin(unsigned short int num)

{ int bin[16];

int ct,i; // function to convert input value into binary array.

int flag=0;

ct=0;

while(num>0)

{

bin[15-ct]=num%2;

num=num/2;

ct++;

}

printf("\n Binary value is: "); //Print the binary value

for(i=0; i<16;i++)

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

for(i=12; i<16;i++)

{

if (bin[i]==1)

flag=flag+1;

}

if (flag ==3 || flag ==4)

printf("\n Condition--> 3 of last 4 bits high--True");

else

printf("\n Condition--> 3 of last 4 bits high--False");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Function name: ReverseBytes Parameters: Hex number Description: Function to reverse byte order by retaining the data \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned int ReverseBytes(unsigned short int val) //Function to reverse the byte order

{

return ((val) << 8 | (val) >> 8); //Swap the bytes implementing shift operators and retaining the input data using OR operator.

}