Base Converter Embedded in High level Language

Strengths of the Project:

- ✓ This program represents the best example of inline assembly.
- ✓ The program can handle 16 bit of input data properly and successfully.
- ✓ It can calculate results accurately.
- ✓ The program executable file can be run on any platform. Like Windows or Linux.

Shortcomings of the Project:

- ✓ The written code is not optimized as it should be.
- ✓ It can not convert signed numbers correctly.
- ✓ There are 6 total functions for conversion. Code to take input from user is implemented in each function separately according to function requirements. So, there are no function parameters. Which is not good (according to me) as I can not use that function again by providing only arguments. I must write whole code for that function again to use it, which is not optimized. So, in this regard a big future improvement can be made by taking input separately and provide them as parameters in functions.
- ✓ The intel syntax is used for inline assembly. So, to compile it successfully one must use MSVC (Microsoft Visual C++).
- ✓ The program can be improved by adding more conversions functions like we can add 6 more functions for Octal number system.

Testing:

- ✓ Every line of code was tested with the help of Disassembler, Registers panel and Memory viewer in debugger section of VS studio.
- ✓ After the completion of every function, it was tested against some random and predefined values. The result was crossed checked with the **online base converter's** results.
- ✓ After the completion of the project testing on whole project was performed. Switches were checked by giving random values as switches were used to navigate between different functions
- ✓ Restraint put on different types of inputs were checked by giving very large value greater than 16bit and very small value (negative number).
- ✓ All the results were cross checked by an online base converter.

Resources:

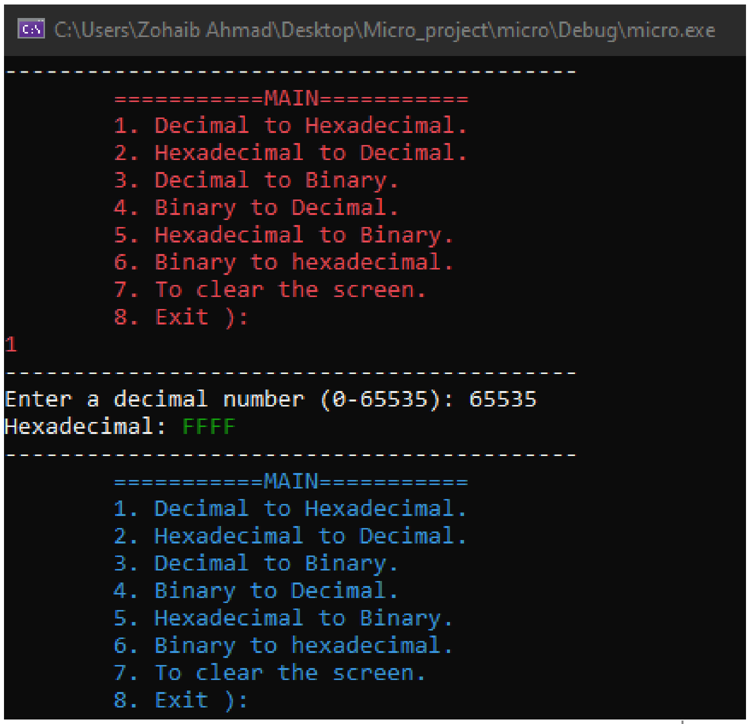
- https://learn.microsoft.com/documentation for resolving Errors
- Emu8086 official documentation.

To compile and test the code, use VS studio.

```
//===================Base Converter====================
//Zohaib Ahmad
//SP21-BCS-048

#include <iostream>
#include <string>
#include <windows.h>
#include <cstdlib>
using namespace std;
//------------------------------
char func_1_input[] = "0000";
int func_1_DecimalResult;
//------------------------------
string func_2_hexResult;
int func_2_input;
short func_2_counter;
//--------------------------------
int func_3_input;
string func_3_BinaryResult;
short func_3_counter;
//--------------------------------
unsigned short func_4_decimalResult;
short func_4_counter;
//------------------------------------
char func_5_input[] = "0000";
int func_5_decimalvalue;
uint8_t func_5_counter;
//--------------------
unsigned short func_6_decimalvalue;
string func_6_hexResult;
short func_6_counter;

//================================
int random_num() {
	int a = 1;
	int b = 15;
	int rn = (rand() % (b - a + 1)) + a;
	return rn;
}
//=============================
void color(int y)
{
	SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), y);
}

//1.==========================
void De_to_hex()
{
	//string
	func_2_hexResult = "0000";
	func_2_counter = 0;
	// Take input between 0 to 65535
	do
	{
	std::cout << "Enter a decimal number (0-65535): ";
		cin >> func_2_input;

	} while (func_2_input < 0 || func_2_input> 65535);


	__asm {
		//clearing registers
		xor EBX, EBX;
		xor EDX, EDX;
		xor EAX, EAX;
		xor EDI, EDI;
		xor ECX, ECX;

//Storing offset address of input variable in ESI register
		lea ESI, func_2_input;
//Storing offset address of output string in EDI register
		lea EDI, func_2_hexResult;
//Adding 4 in EDI to get exact address of first byte of the result string
			add EDI, 4;
			//Moving input number in AX register
			mov AX, [ESI];
			//moving 16 in EBX
			mov EBX, 16;

	divide_1:
//Dividing EAX by EBX
			div EBX;
//Pushing the remainder onto to the stack
			push EDX;
//Incrementing counter (which will be used to print the result)
			add func_2_counter, 1;
//Incrementing ECX to use it later in loop
			inc ECX;
//Compairing AX with 0
			cmp AX, 0;
//Checking if AX is not zero jump to lebal divide_1;
			jne divide_1;

	check_1:
//Poping remainder values in EBX
			pop EBX;
//Compairing with 9
			cmp BL, 9;
//If BL is greater than 9 jump to label greater_1
			ja greater_1;

	less_1:
//Adding 48 to change the value to Ascii to print
			add BL, 48;
//Storing Ascii value to result string
			mov[EDI], BL;
//Incrementing address to move it to next Byte of string
			add EDI, 1;

			loop check_1;
			jmp r;

	greater_1:
//If BL is greater than 9 adding addition 7 to get ascii values from A-F
			add BL, 7;
			jmp less_1;
	r:


	}
	std::cout << "Hexadecimal: ";
	color(2);
	for (int i = 0; i < func_2_counter; i++)
	{
			//Printing result string
			std::cout << func_2_hexResult[i];
	}
	color(15);
	std::cout << "" << endl;
}

//2.=====================================
void hex_to_de() {
	bool choice = false;

	//Will take 4 character hex input.
	//will check whether the input is between 1-9 an A-F.
	//if not ask again.

	do {
			std::cout << "Enter hexadecimal value : " << endl;
			cin >> func_1_input; //char
```

```
        for (int i = 0; i < 4; i++) {
            if ((func_1_input[i] >= '0' && func_1_input[i] <= '9') || (func_1_input[i] >= 'A' && func_1_input[i] <= 'F') || (func_1_input[i] >= 'a' && func_1_input[i] <= 'f')) {
                choice = true;

            }
            else {
std::cout << "Enter correct hex Value" << endl;
                choice = false;
                break;
            }
        }
    } while (choice == false);


    __asm
    {
        //Moving 4 in ECX to loop 4 times
        mov ECX, 4;
        //Clearing EBX
        xor EBX, EBX;
        mov EBX, 0;

    my1:
//Compairing input characters with the value of ascci 'a'
        cmp[func_1_input + EBX], 'a';
//jump if input character value is less than 'a'
        jl nocap;
//Compairing characters of input with the value of ascci 'z'
        cmp[func_1_input + EBX], 'z';
//jump if input character value is greater than 'z'
        ja nocap;
        //Else sub 32 to change it to capital
        sub[func_1_input + EBX], 32;
    nocap:
        add EBX, 1;
        loop my1;

//Converting ascci values to hex in memory
        xor ECX, ECX;
        xor EAX, EAX;
//Taking legth of variable 'func_1_input' and moving it to ECX
        mov ECX, length func_1_input;
//sub 1 for getting correrct length
        sub ECX, 1;
        xor EBX, EBX;
        mov EBX, 0;
    ToHex_1:
//Moving first ascii character in BL
        mov AL, [func_1_input + EBX];
//Compairing it with ascii '9'
        cmp AL, 39h;
//jump if value is above ascii '9'
        ja If_A_to_F_1;
        //Else sub 30h from it to change it to hex value
        sub AL, 30h;
//Store it back to its location
        mov[func_1_input + EBX], AL;
        add EBX, 1;
        jmp loop_1;


//if value is above the value of '9'
    If_A_to_F_1:
//Add 7h to move to ascii value 'A'
        sub AL, 7h;
        //sub 30h to change it into its hex value
        sub AL, 30h;
        //storing back to its location
        mov[func_1_input + EBX], AL;
        add EBX, 1;
        jmp loop_1;

    loop_1:
        loop ToHex_1;
    }



    __asm
    {
        //Clearing the registers
        xor EDX, EDX;
        xor EAX, EAX;
        xor EDI, EDI;
        xor ECX, ECX;
    //Storing Staring address of hex value in ESI
        lea ESI, func_1_input;
        //Storing address of result  in EDI
        lea EDI, func_1_DecimalResult;
        mov ECX, 4;
        //moving 1 in BX
        mov BX, 1d;
//Adjusting address of ESI (addressing last value)
        add ESI, 3;

    labl2:


        //Moving first Hex value in AL
        mov AL, [ESI];
        //Multiplying it with BX
        mul BX;
        //Pushing DX onto stack
        push DX;
        //Pushing AX onto stack
        push AX;
//Pop value of stack in 32bit register EDX
        pop  EDX;
//storing result value in 'func_1_DecimalResult' variable
        mov[EDI], EDX;
//Adding 2 in EDI to point to next word
        add EDI, 2;
        sub ESI, 1;

        //Moving value of BX in AX
        mov AX, BX;
        mov BX, 16d;
        mul BX;
        mov BX, AX;
        xor EAX, EAX;
        loop labl2;
    }

    __asm
    {xor EBX, EBX;
    xor EDX, EDX;
    xor EDI, EDI;

    lea ESI, func_1_DecimalResult;
    mov ECX, 4;

labl1:
    //Adding all the word size value in variable
    mov BX, [ESI];
    add ESI, 2;
    add EDX, EBX;
    loop labl1;

    lea EDI, func_1_DecimalResult;
    mov[EDI], EDX;
    xor ECX, ECX;

    }

    std::cout << "Decimal: ";
    color(2);
    std::cout << func_1_DecimalResult << endl;
    color(15);




}

//3.=================================
void de_to_bi() {

    //string
    func_3_BinaryResult = "$$$$$$$$$$$$$$$";
    func_3_counter = 0;

    // Take input between 0 to 65535
    do
    {
        std::cout << "Enter a decimal number (0-65535): " << endl;
        cin >> func_3_input;
    } while (func_3_input < 0 || func_3_input > 65535);

    __asm
    {
        //Clearing registers
        xor EBX, EBX;
        xor EDX, EDX;
        xor EAX, EAX;
        xor EDI, EDI;
        xor ECX, ECX;

        //Loading address in ESI
        lea ESI, func_3_input;
        //Loading address in EDI
        lea EDI, func_3_BinaryResult;
        //Adjusting address of EDI
        add EDI, 4;

        //Moving input value in AX
        mov AX, [ESI];
        mov EBX, 2;

    divid_3:
        //Dividing by EBX
        div EBX;
        //Push remainder onto stack
        push EDX;
        add func_3_counter, 1;
        inc ECX;
        //Compairing with zero
        cmp AX, 0;
        //if not equal loop 'divid_3'
        jne divid_3;


        //saves result
    save_3:
        //pop in EBX register
        pop EBX;
        //Convert to ascii to print
        add BX, 30h;
        mov[EDI], BL;
        inc EDI;
        loop save_3;
    }
    std::cout << "Binary: ";
    color(2);
    for (int i = 0; i < func_3_counter; i++) {
        std::cout << func_3_BinaryResult[i];
    }
    std::cout << "" << endl;
    color(15);

}

//4.==================================
void bi_to_de()
{
    func_4_decimalResult = 0;
    func_4_counter = 0;
    char func_4_input[] = "$$$$$$$$$$$$$$$$$$";

    //input
    std::cout << "Enter Binary value : " << endl;
    cin >> func_4_input;

    //Counting the 0 and 1
    for (int i = 0; i < 18; i++) {
        if (func_4_input[i] == '$')
        {
            break;
        }
        else {
            func_4_counter = func_4_counter + 1;

        }

    }
    func_4_counter = func_4_counter - 1;



    __asm
    {

        //Clearing registers
        xor EDX, EDX;
        xor ECX, ECX;
        xor EAX, EAX;
        xor EDI, EDI;

        //Loading address of 'func_4_input' in ESI
        lea ESI, func_4_input;
        //Loading address of 'func_4_decimalResult' in EDI
        lea EDI, func_4_decimalResult;
        mov BX, 1d;
        mov CX, func_4_counter;
        //for pointing the last byte of the char array
        add ESI, ECX;
        sub ESI, 1;

    labl2:

        mov AL, byte ptr[ESI];
        //coverting to decimal
        sub AL, 30h;
        mul BX;
        push DX;
        push AX;
        pop  EDX;
        add func_4_decimalResult, DX;
        sub ESI, 1;

        mov AX, BX;
        mov BX, 2d;
        mul BX;
        mov BX, AX;
        xor EAX, EAX;
        loop labl2;
    }

    //Result
    std::cout << "Decimal: ";
    color(2);
    std::cout << func_4_decimalResult << endl;
    color(15);
}

//5.=====================================
void hex_to_bi() {
    func_5_counter = 0;
    char e[] = "$$$$$$$$$$$$$$$";

    //Will take 4 character hex input.
    //will check whether the input is between 1-9 an A-F.
    //if not ask again.

    bool choice = false;
    do {
        std::cout << "enter hexadecimal value : " << endl;
        cin >> func_5_input;

        for (int i = 0; i < 2; i++) {
            if ((func_5_input[i] >= '0' && func_5_input[i] <= '9') || (func_5_input[i] >= 'A' && func_5_input[i] <= 'F') || (func_5_input[i] >= 'a' && func_5_input[i] <= 'f')) {
                choice = true;

            }
            else {
                std::cout << "enter correct" << endl;
                choice = false;
                break;
            }
        }
    } while (choice == false);



    __asm
    {
        mov ECX, 4;
        xor EBX, EBX;
        mov EBX, 0;

    my12:
//Compairing input characters with the value of ascci 'a'
        cmp[func_5_input + EBX], 'a';
//jump if input character value is less than 'a'
        jl nocap1;
//Compairing characters of input with the value of ascci 'z'
        cmp[func_5_input + EBX], 'z';
//jump if input character value is greater than 'z'
        ja nocap1;
        //Else sub 32 to change it to capital
        sub[func_5_input + EBX], 32;
    nocap1:
        add EBX, 1;
        loop my12;


        //Converting Ascii value to its hex values
        xor ECX, ECX;
        xor EAX, EAX;
        mov ECX, length func_5_input;
        sub ECX, 1;
        xor EBX, EBX;
        mov EBX, 0;
    ak1:
        mov AL, [func_5_input + EBX];
        cmp AL, 39h;
        ja delta1;
        sub AL, 30h;
        mov[func_5_input + EBX], AL;
        add EBX, 1;
        jmp alpha1;

    delta1:
        sub AL, 7h;
        sub AL, 30h;
        mov[func_5_input + EBX], AL;
        add EBX, 1;
        jmp alpha1;

    alpha1:
        loop ak1;


        //Converting Hex to Decimal
        __asm
        {
            xor EDX, EDX;
            xor EAX, EAX;
            xor EDI, EDI;
            xor ECX, ECX;
            lea ESI, func_5_input;
            lea EDI, func_5_decimalvalue;
            mov ECX, 4;
            mov BX, 1d;
            add ESI, 3;

        labl2:

            mov AL, [ESI];
            mul BX;
            push DX;
            push AX;
            pop  EDX;
            mov[EDI], EDX;
            add EDI, 2;
            sub ESI, 1;

            mov AX, BX;
            mov BX, 16d;
            mul BX;
            mov BX, AX;
            xor EAX, EAX;
            loop labl2;
        }
        __asm
        {
            xor EBX, EBX;
            xor EDX, EDX;
            xor EDI, EDI;
            lea ESI, func_5_decimalvalue;

            mov ECX, 4;

        labl12:
            mov BX, [ESI];
            add ESI, 2;
            add EDX, EBX;
            loop labl12;

            lea EDI, func_5_decimalvalue;
            mov[EDI], EDX;



            //Converting decimal value to Binary
            xor EBX, EBX;
            xor EDX, EDX;
            xor EAX, EAX;
            xor EDI, EDI;
            xor ECX, ECX;
            lea ESI, func_5_decimalvalue;
            lea EDI, e;


            mov AX, [ESI];
            mov EBX, 2;

        divid12:
            div EBX;
            push EDX;
            add  func_5_counter, 1;
            inc ECX;
            cmp AX, 0;
            jne divid12;


        save12:
            pop EBX;
            add BX, 30h;
            mov[EDI], BL;
            inc EDI;
            loop save12;
        }

    }

    std::cout << "Binary :";
    color(2);
    for (int i = 0; i < func_5_counter; i++) {
        std::cout << e[i];

    }
    std::cout << "" << endl;
    color(15);

}

//6.=======================================
void bi_to_hex() {

    func_6_hexResult = "0000";
    func_6_counter = 0;
    func_6_decimalvalue = 0;
    short n3 = 0;
    char func_6_input[] = "$$$$$$$$$$$$$$$$$$";

    //Input
    std::cout << "Enter Binary value : " << endl;
    cin >> func_6_input;

    //Counting the 0 and 1
    for (int i = 0; i < 18; i++) {
        if (func_6_input[i] == '$') {
            break;
        }
        else {
            n3 = n3 + 1;
        }

    }
    n3 = n3 - 1;


    //Converting binary To Decimal
    __asm
    {
        xor EDX, EDX;
        xor ECX, ECX;
        xor EAX, EAX;
        xor EDI, EDI;
        lea ESI, func_6_input;
        lea EDI, func_6_decimalvalue;
        mov BX, 1d;
        mov CX, n3;
        add ESI, ECX;
        sub ESI, 1;

    labl23:

        mov AL, byte ptr[ESI];
        sub AL, 30h;
        mul BX;
        push DX;
        push AX;
```

```
            pop  EDX;
            add func_6_decimalvalue, DX;
            sub ESI, 1;

            mov AX, BX;
            mov BX, 2d;
            mul BX;
            mov BX, AX;
            xor EAX, EAX;
            loop labl23;
    }
    std::cout << func_6_decimalvalue << endl;

    //Converting Decimal to Hex
    __asm {
            xor EBX, EBX;
            xor EDX, EDX;
            xor EAX, EAX;
            xor EDI, EDI;
            xor ECX, ECX;
            lea ESI, func_6_decimalvalue;
            lea EDI, func_6_hexResult;
            add EDI, 4;

            mov AX, [ESI];
            mov EBX, 16;

    Divide_6:
            div EBX;
            push EDX;
            add func_6_counter, 1;
            inc ECX;

            cmp AX, 0;
            jne Divide_6;

    check_6:
            pop EBX;
            cmp BL, 9;
            ja greater_6;
    less_6:
            add BL, 48;
            mov[EDI], BL;
            add EDI, 1;
            loop check_6;
            jmp r1;
    greater_6:
            add BL, 7;
            jmp less_6;
    r1:


    }

    std::cout << "Hexadecimal: ";
    color(2);
    for (int i = 0; i < func_6_counter; i++) {
```



C:\Users\Zohaib Ahmad\Desktop\Micro_project\micro\Debug\micro.exe

```
-------------------------------------------
        ===========MAIN===========
        1. Decimal to Hexadecimal.
        2. Hexadecimal to Decimal.
        3. Decimal to Binary.
        4. Binary to Decimal.
        5. Hexadecimal to Binary.
        6. Binary to hexadecimal.
        7. To clear the screen.
        8. Exit ):
1
-------------------------------------------
Enter a decimal number (0-65535): 65535
Hexadecimal: FFFF
-------------------------------------------
        ===========MAIN===========
        1. Decimal to Hexadecimal.
        2. Hexadecimal to Decimal.
        3. Decimal to Binary.
        4. Binary to Decimal.
        5. Hexadecimal to Binary.
        6. Binary to hexadecimal.
        7. To clear the screen.
        8. Exit ):
```

```
            std::cout << func_6_hexResult[i];
    }
    std::cout << "" << endl;
    color(15);

}
//============================================


int main()
{
    int option;
    do {
            std::cout << "------------------------------------------" << endl;
            color(random_num());
            std::cout << "\t===========MAIN===========" << endl;
            std::cout << "\t1. Decimal to Hexadecimal." << endl;
            std::cout << "\t2. Hexadecimal to Decimal." << endl;
            std::cout << "\t3. Decimal to Binary." << endl;
            std::cout << "\t4. Binary to Decimal." << endl;
            std::cout << "\t5. Hexadecimal to Binary." << endl;
            std::cout << "\t6. Binary to hexadecimal." << endl;
            std::cout << "\t7. To clear the screen." << endl;
            std::cout << "\t8. Exit ):" << endl;

            cin >> option;
            color(15);
            std::cout << "------------------------------------------" << endl;

            switch (option)
            {
            case 1:
                    De_to_hex();
                    break;
            case 2:
                    hex_to_de();

                    break;
            case 3:
                    de_to_bi();

                    break;
            case 4:
```

```
                                bi_to_de();

                                break;
                    case 5:
                                hex_to_bi();

                                break;
                    case 6:
                                bi_to_hex();
                                break;
                    case 7:
                                system("cls");
                                break;
                    default:
                                break;
                          }
      } while (option != 8);

      return 0;
}
```

Result: