FCR PL01

Juan José Vázquez Prieto -> UO282978

Martín Feito Rodríguez -> UO286333

```cpp
#include <iostream>
#include <bitset>
using namespace std;
using std::cout; using std::endl;
using std::string; using std::bitset;
extern "C" bool IsValidAssembly(int a, int b, int c);

char controlString(char* cadena,char* cadena2) { //Esta es la primera función que recibe dos cadenas definidas en main()
    if (strcmp(cadena, "2dBkF998Y") == 0) {  //Este condicional imprime en la consola si la primera cadena es 2dBkF998Y
        cout << "Acceso Permitido" << endl;
        return true;
    }
    else {
        cout << "Acceso Denegado" << endl; //En caso contrario da como resultado un mensaje de Acceso Denegado y exit.
        return false;
        exit(1);
    }
    if (strlen(cadena2) <= 10 && cadena2[0] == cadena2[9]) { //Este condicional comprueba si la longitud de la cadena 2 es <= que 10
        cout << "Acceso Permitido" << endl; //Ademas de comprobar si el elemento 0 (osea el primero) de la cadena2 es = al 9(el 10 en realidad).
        return true;
    }                                   //En caso afirmativo imprime Acceso Permitido y no nos echaria del programa.
    else {
        cout << "Hubo un fallo" << endl; //En caso contrario da como resultado un mensaje de Hubo un fallo y exit.
        return false;
        exit(1);
    }
}
```

```cpp
int ControlWithMask(bitset<32> bs1,bitset<32> bs2,int decimal) //Esta función de tipo entero recibe dos numeros binarios de 32 bits ademas de un decimal.
{                                                              //Este número decimal serían los bits 5,6 y 7 del numero bs2 convertidos a decimal.
    if (bs1[5] == 0) {   //Este condicional comprueba si el bit 6 o la 5 posicion del numero 1 es igual a 0.
        cout << "Correcto" << endl; //En caso afirmativo imprime un mensaje de correcto en la pantalla.
        return true;
    }
    else {
        cout << "Hubo algún fallo" << endl; //En caso contrario imprime un mensaje de error y llama al exit.
        return false;
        exit(1);
    }
    if (bs1[12] == bs2[10]) { //Este condicional comprueba si la posición 12 o bit 13 del numero 1 es igual al bit 11 o posicion 10 del número 2.
        cout << "Correcto" << endl; //En caso afirmativo imprime un mensaje de correcto en la pantalla.
        return true;
    }
    else {
        cout << "Hubo algún fallo" << endl; //En caso contrario imprime un mensaje de error y llama al exit.
        return false;
        exit(1);
    }
    if (decimal == 3) { //Este condicional comprueba si las posiciones 5,6 y 7 del número 2 convertidas a decimal son igual a 3.
        cout << "Correcto" << endl; //En caso afirmativo imprime un mensaje de correcto en la pantalla.
        return true;
    }
    else {
        cout << "Hubo algún fallo" << endl; //En caso contrario imprime un mensaje de error y llama al exit.
        return false;
        exit(1);
    }
    return 0;
}
```

```cpp
int CheckAsmControl(bitset<32> bs3,bitset<32> bs4,bitset<32> bs5) //Esta es la 3 función que recibe tres numeros binarios de 32 bits.
{
    int num180 = 10110100; //Declaramos num180 que es el 180 en binario.
    if ((bs3[0] == bs5[17]) && ((int)(bs3.to_ulong()) > num180)) //Este condicional comprueba si el bit 1 o posicion 0 del número 3 es igual del bit 18 o pos 17
    {                                                            //del numero binario5 ademas de cumplirse si y solo si se cumple que el numero 3 es mayor que 180.
        cout << "Bien" << endl; //En caso afirmativo imprime un mensaje de correcto en la pantalla.
        return 1;
        return true;
    }
    else
    {
        cout << "Mal" << endl; //En caso contrario imprime un mensaje de error y llama al exit.
        return 0;
        return false;
    }
}
```

```cpp
int CheckInlineAsmAccess(bitset<32> bs6, const char* bs) // Esta función de tipo entero recibe un número binario de 32 bits.
{
    if (num6 == 262144) {
        bs = "00000000000000100000000000000000";
        if (bs[19] == bs[18] && bs[18] != bs[13]) {
            cout << "Entrada correcta" << endl;
            return true;
        }
        else {
            cout << "Entrada incorrecta" << endl;
            return false;
            exit(1);
        }
    }
    if (num6 == 12288) {
        bs = "00000000000000000011000000000000";
        if (bs[19] == bs[18] && bs[18] != bs[13]) {
            cout << "Entrada correcta" << endl;
            return true;
        }
        else {
            cout << "Entrada incorrecta" << endl;
            return false;
            exit(1);
        }
    }
    else {
        if (bs6[14] == bs6[15] && bs6[15] != bs6[20]) { // Este condicional comprueba si el bit 13 del num6 es igual al 14 y si el bit 14 es igual al 19.
            cout << "Entrada correcta" << endl; //En caso afirmativo imprime un mensaje de correcto en la pantalla.
            return true;
        }
        else {
            cout << "Entrada incorrecta" << endl; //En caso contrario imprime un mensaje de error y llama al exit.
            return false;
            exit(1);
        } //Aqui esta __asm que es donde se implementa el ensamblador en linea el cual conseguimos al debuggear la función y acceder al desensamblado.
          //Lo tenemos comentado debido a que no funciona.
    }
//  /* __asm

// ...
    return 0;
}
```

```cpp
string Decimal_Binario(int n) //Esta función convierte los números enteros a números binarios.
{

    unsigned resto;
    unsigned int numBin = 0;
    unsigned int cont = 0;
    while (n > 0) {
        resto = n % 2;
        n = n / 2;
        numBin = numBin + resto * pow(10, cont);
        cont++;
    }
    return numBin;
}

int Binario_Decimal(int x,int decimal) { //Esta función convierte los números binarios a números decimales.
    int exp, digito;
    exp = 0;
    decimal = 0;
    while (((int)(x / 10)) != 0)
    {
        digito = (int)x % 10;
        decimal = decimal + digito * pow(2.0, exp);
        exp++;
        x = (int)(x / 10);
    }
    decimal = decimal + x * pow(2.0, exp);
    return decimal;
}
```

```cpp
int main() //Esta es la parte principal del programa donde se ejecuta todo el codigo.
{
    char cadena[10]; //Aquí se declaran todas las variables que se usan.
    char cadena2[10];
    unsigned int num1;
    unsigned int num2;
    int num3;
    int num4;
    int num5;
    int num6;
    int decimal;
    const char* bs;
    cout << "Introduce una cadena: "; //Aquí se solicitan todas las variables necesarias.
    cin >> cadena;
    cout << cadena << endl;

    cout << "Introduce otra cadena: ";
    cin >> cadena2;
    cout << cadena2 << endl;

    cout << "Introduce el num1 (entero positivo): ";
    cin >> num1;
    cout << num1 << endl;

    cout << "Introduce el num2 (entero positivo): ";
    cin >> num2;
    cout << num2 << endl;

    cout << "Introduce el num3 (entero): ";
    cin >> num3;
    cout << num3 << endl;

    cout << "Introduce el num4 (entero): ";
    cin >> num4;
    cout << num4 << endl;

    cout << "Introduce el num5 (entero): ";
    cin >> num5;
    cout << num5 << endl;

    cout << "introduce el num6 (entero): ";
    cin >> num6;
    cout << num6 << endl;
```

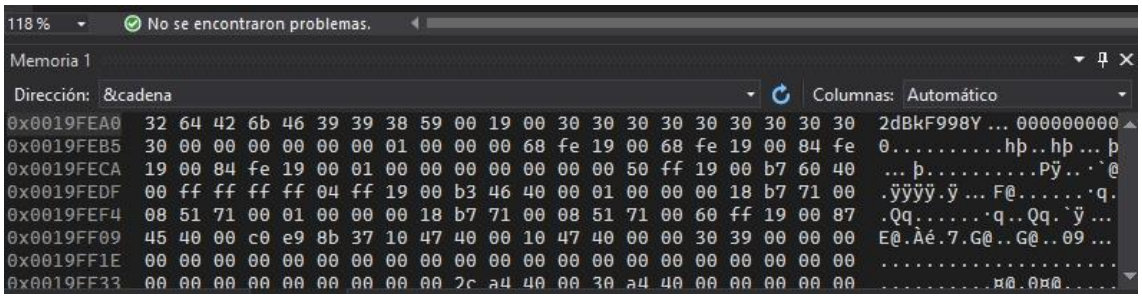
```cpp
    bitset<32> bs1(Decimal_Binario(num1)); //Aquí con la libreria bitset se convierten usando la función Decimal_Binario los numeros enteros a binarios de 32 bits.
    bitset<32> bs2(Decimal_Binario(num2));
    bitset<32> bs3(Decimal_Binario(num3));
    bitset<32> bs4(Decimal_Binario(num4));
    bitset<32> bs5(Decimal_Binario(num5));
    bitset<32> bs6(Decimal_Binario(num6));
    char x = bs2[4]+bs2[5]+bs2[6];

    Binario_Decimal(x,decimal);
    controlString(cadena,cadena2); //Aquí se hace llamada en main a la primera función que usa dos cadenas.
    ControlWithMask(bs1,bs2,decimal);//Aquí se hace llamada en main a la segunda función que usa dos num binarios y la conversion de tres bits de num2 a numero decimal.
    CheckAsmControl(bs3, bs4, bs5);//Aquí se hace llamada en main a la tercera función que usa tres números binarios.
    CheckInlineAsmAccess(bs6, num6, bs);//Aquí se hace llamada en main a la cuarta y ultima función que usa un número binario.

    if (controlString != false  && ControlWithMask != false && CheckAsmControl != false && CheckInlineAsmAccess != false) //Aquí se hace llamada a si las funciones retornaron o no falso.
        //En caso que no retornaran falso se imprime el mensaje de acceso permitido.
    {
        cout << "Acceso permitido" << endl;
    }
    else {
        cout << "Acceso bloqueado" << endl;
    }
    cout << endl;
    return 0;
}
```

Esta es una explicación del código comentado, he dejado dos archivos cpp unos comentado y otro sin comentar que hace más fácil su lectura de código, pero sin la explicación.

Respuesta a las preguntas propuestas:

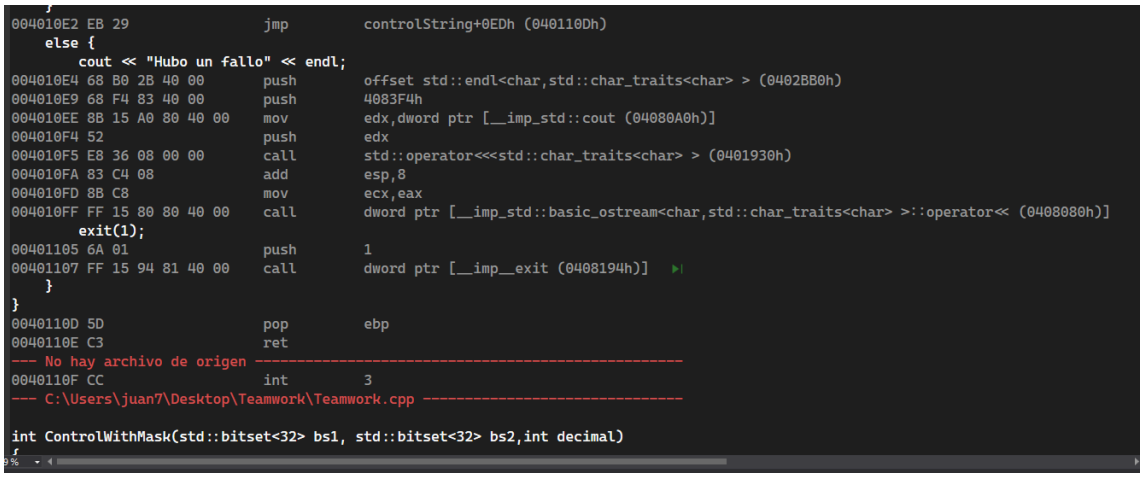
- (0,5 puntos) Dirección de la memoria en la que se encuentra la primera cadena que se lee en la primera función indicada en las instrucciones.

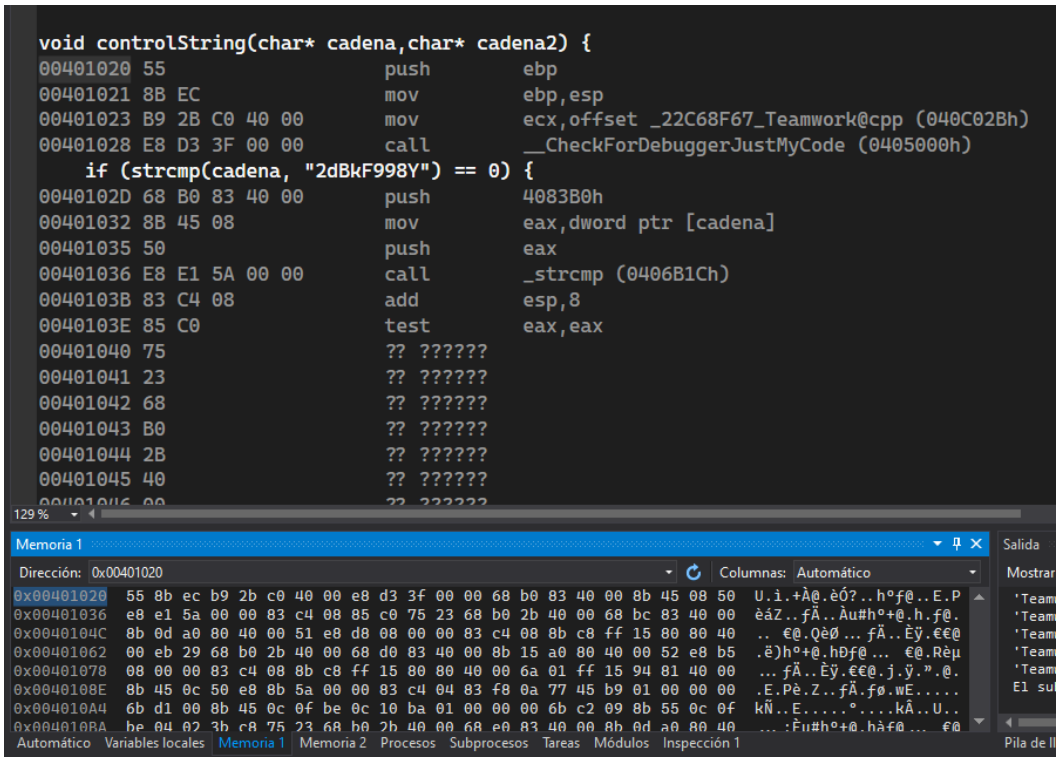
La respuesta es 0019FEA0, esto lo sabemos ya que al debuggear el programa poniendo puntos de ruptura para parar el programa programa para ver las variables y ver donde se ubican. Adjunto una imagen:



- (0,5 puntos) Dirección de la memoria en la que se sitúa el epílogo de la primera función indicada en las instrucciones y el propio código del epílogo, en forma de código máquina y de mnemónicos.

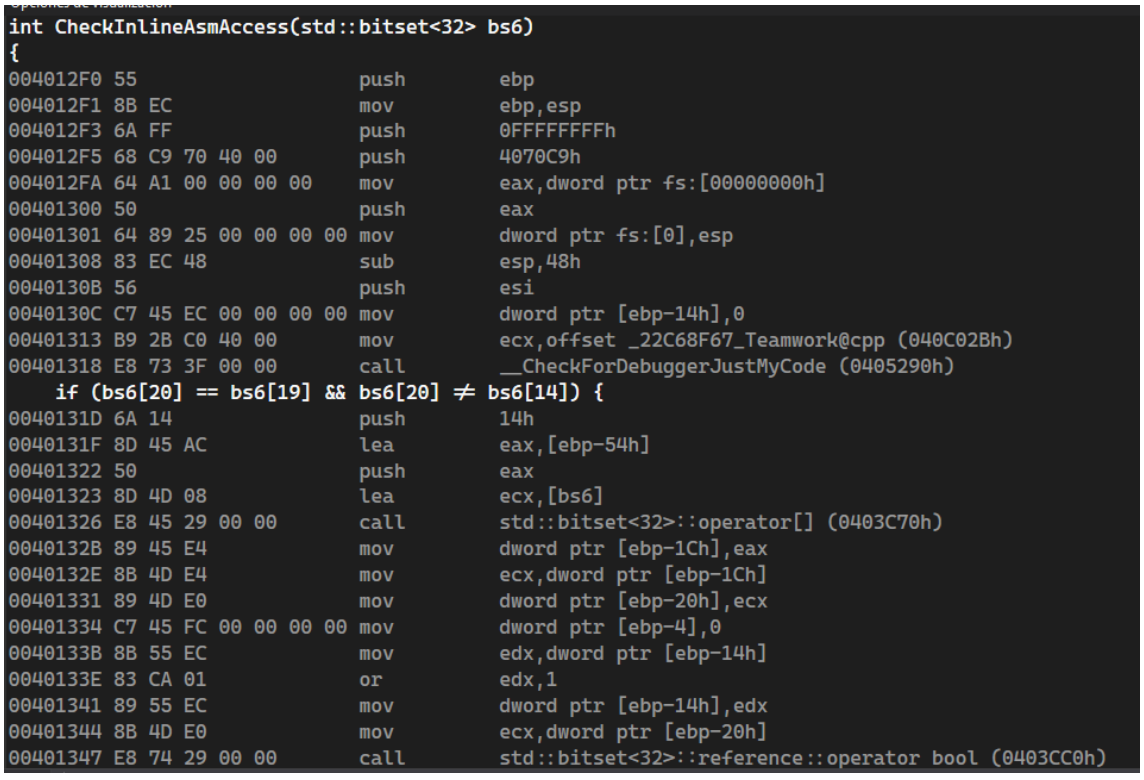
La respuesta creo que es 0040110E debido a que es lo ultimo en desensamblado de la primera función, si el epilogo se refiere a la primera en vez de a la última parte sería 00401020.





- (0,5 puntos) Código máquina de la primera instrucción de ensamblador introducida en el ensamblador en línea de la cuarta función indicada en las instrucciones.

Adjunto foto de la respuesta que se obtiene mediante el desensamblado:



Ejemplos de entradas validas e invalidas para las funciones:

Para la 1ª función una entrada valida en cadena 1 sería "2dBkF998Y" y cualquiera distinta a esta sería invalida. La cadena 2 sería diferente ya que sus condiciones es que tenga una longitud igual o menor a 10 y que su posición 2 sea igual a la 9, un ejemplo de una entrada valida sería 0000000000 siendo una cadena de longitud igual a 10 y todas sus posiciones iguales y una no valida por ejemplo 011100000000000 ya que su longitud es mayor a 10 y su posición 2 distinta a la 9.

Para la 2ª función una entrada valida sería el 48 en n2 y 0 en n1 ya que cumple todas las condiciones, el bit 6 del primer número sería 0, el bit 13 de n1 seria igual al bit 11 de n2 y los bits 5,6 y 7 de n2 formarían un 3 en decimal ("011"). Una entrada no valida sería 200 en n2 y 48 en n1 ya que no cumplirían las condiciones puestas e imprimiría un mensaje de error además de llamar al exit.

Para la 3ª función una entrada valida sería 181 en num3,0 en num4 y 0 en num5 ya que cumple la condición de igualdad de bits y de ser mayor que 180. Una entrada no valida seria 50 en num3, 1 en num4 y 0 en num5 ya que no se cumpliría ninguna condición.

Para la última función una entrada valida sería 12288 o el 262144 debido a que ambos cumplen las condiciones de igualdad y diferencia de bits, mientras que 0 sería una entrada invalida al no cumplirse la diferencia de bits o 266240 ya que cumple la diferencia, pero no la igualdad.

Reparto de trabajo:

Yo (Martin) me he encargado de la función 3 y 2. Además de arreglar algunos errores del programa.

Yo (Juan) por mi parte me he encargado de la función 1 y 4 además de arreglar errores y resolver los ejercicios de la fase 1.2 y hacer este pdf.

Miguel Nonide Miranda con UO286609 no participo en el trabajo por tanto solo fue realizado por las dos personas ya nombradas.

Entre los dos le hemos dedicado de media cada uno entre 15 y 20 horas a este trabajo.