6 de diciembre de 2020

Rogelio Daniel Gonzalez Nieto

FACULTAD DE INGENIERIA

AREA DE CIENCIAS DE LA COMPUTACION

Practica no. 11

LABORATORIO PROGRAMACION DE SISTEMAS

# OBJETIVO

Ensamblar las instrucciones de acuerdo con los formatos y modos de direccionamiento de la SIC extendida y generar el archivo objeto añadiendo los registros de modificación correspondientes.

# DESARROLLO

Para poder crear el código objeto se usó una lógica parecida a la del sic standard, con la diferencia de las nuevas instrucciones, en primera se buscan las instrucciones que no generan código objeto y a esas se les asignaba como código **“---“,** después se analizaban las instrucciones y se determinaba que formato de instrucción era.

Para las instrucciones byte y Word se seguía el mismo procedimiento del sic standard.

Para las instrucciones de formatos 1 y 2 no había problema y se ensamblaba la instrucción como se muestra a continuación:

else if (this.formatoInstruccion(linea.sCodigoOp) == 1)

{

linea.sCodigoObjeto = this.codInsXE(linea.sCodigoOp);

}

else if (this.formatoInstruccion(linea.sCodigoOp) == 2)

{

linea.sCodigoObjeto = this.codInsXE(linea.sCodigoOp);

if (linea.sDireccion.Length >= 1)

{

string trim = ",";

string[] aRegistros = linea.sDireccion.Split(trim[0]);

foreach (var a in aRegistros)

{

if (a == "A" || a == "B" || a == "F" || a == "L" || a == "S" || a == "T" || a == "X")

{

linea.sCodigoObjeto = linea.sCodigoObjeto + this.regresarRegistro(a);

}

else if (this.valorNumero(a) != "ERROR SINTACTICO")

{

linea.sCodigoObjeto = linea.sCodigoObjeto + a;

}

else if (this.valorNumero(a) == "ERROR SINTACTICO")

{

linea.sCodigoObjeto = "ERROR SINTACTICO";

}

}

if (linea.sCodigoObjeto.Length < 4)

{

switch (linea.sCodigoObjeto.Length)

{

case 1:

linea.sCodigoObjeto = linea.sCodigoObjeto + "000";

break;

case 2:

linea.sCodigoObjeto = linea.sCodigoObjeto + "00";

break;

case 3:

linea.sCodigoObjeto = linea.sCodigoObjeto + "0";

break;

}

}

}

}

Para los formatos 3 y 4 era un poco más complicado el asunto ya que se tenia que tomar en cuenta el valor del contador y la base con las mismas estructuras if y else se determinaba el tipo de instrucción, basándose en la estructura inicial del programa que separa las instrucciones del demás contendió como las direcciones de memoria y en base a esas se determinaban las banderas del código obj como se muestra a continuación:

else if (this.formatoInstruccion(linea.sCodigoOp) == 3)

{

if (linea.sDireccion.Contains("#"))

{

string nDireccion = linea.sDireccion.Substring(1);

if (this.esConstante(nDireccion))

{

string nixbpe = "0100000";

string sCodigo = this.codInsXE(linea.sCodigoOp) + nixbpe;

string sDireccion = linea.sDireccion.Substring(1);

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + sDireccion;

}

else if (buscarTABSIB(linea.sDireccion.Replace("#", "")) != null)

{

if (this.relativoContador(this.buscarTABSIB(linea.sDireccion), this.sContador(this.ensamblador.programa.IndexOf(linea))))

{

string nixbpe = "010010";

string sCodigo = this.codInsXE(linea.sCodigoOp) + nixbpe;

int iTA = Convert.ToInt32(this.buscarTABSIB(linea.sDireccion), 16);

int iPC = Convert.ToInt32(this.sContador(this.ensamblador.programa.IndexOf(linea)), 16);

int iDes = iTA - iPC;

string sDesplazamiento = iDes.ToString("X");

if (iDes < 0)

{

string sDes = this.desplazamiento(sDesplazamiento);

linea.sCodigoObjeto = sCodigo + sDes;

}

else

{

linea.sCodigoObjeto = sCodigo + sDesplazamiento;

}

}

else if (this.relativoBase(this.buscarTABSIB(linea.sDireccion), this.buscaBase()))

{

string nixbpe = "010100";

string sCodigo = this.codInsXE(linea.sCodigoOp) + nixbpe;

int iTA = Convert.ToInt32(this.buscarTABSIB(linea.sDireccion), 16);

int iBase = Convert.ToInt32(this.buscaBase(), 16);

int iDes = iTA - iBase;

string sDesplazamiento = iDes.ToString("X");

if (iDes < 0)

{

string sDes = this.desplazamiento(sDesplazamiento);

linea.sCodigoObjeto = sCodigo + sDes;

}

else

{

linea.sCodigoObjeto = sCodigo + sDesplazamiento;

}

}

else

{

linea.sCodigoObjeto = "ERROR la ins. no es relativa ni al contador ni a la base";

}

}

else if (buscarTABSIB(linea.sDireccion.Substring(1)) == null)

{

linea.sCodigoObjeto = "ERROR simbolo no existe";

}

}

else if (linea.sDireccion.Contains("@"))

{

string nDireccion = linea.sDireccion.Substring(1);

if (this.esConstante(nDireccion))

{

string nixbpe = "100000";

string sCodigo = this.codInsXE(linea.sCodigoOp) + nixbpe;

string sDireccion = linea.sDireccion.Substring(1);

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + sDireccion;

}

else if (buscarTABSIB(linea.sDireccion.Replace("@", "")) != null)

{

if (this.relativoContador(this.buscarTABSIB(nDireccion), this.sContador(this.ensamblador.programa.IndexOf(linea))))

{

string nixbpe = "100010";

string sCodigo = this.codInsXE(linea.sCodigoOp) + nixbpe;

int iTA = Convert.ToInt32(this.buscarTABSIB(linea.sDireccion.Replace("@", "")), 16);

int iPC = Convert.ToInt32(this.sContador(this.ensamblador.programa.IndexOf(linea)), 16);

int iDes = iTA - iPC;

string sDesplazamiento = iDes.ToString("X");

if (iDes < 0)

{

string sDes = this.desplazamiento(sDesplazamiento);

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + sDes;

}

else

{

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + sDesplazamiento;

}

}

else if (this.relativoBase(this.buscarTABSIB(nDireccion), this.buscaBase()))

{

string nixbpe = "100100";

string sCodigo = this.codInsXE(linea.sCodigoOp) + nixbpe;

int iTA = Convert.ToInt32(this.buscarTABSIB(linea.sDireccion), 16);

int iBase = Convert.ToInt32(this.buscaBase(), 16);

int iDes = iTA - iBase;

string sDesplazamiento = iDes.ToString("X");

if (iDes < 0)

{

string sDes = this.desplazamiento(sDesplazamiento);

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + sDes;

}

else

{

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + sDesplazamiento;

}

}

else

{

linea.sCodigoObjeto = "ERROR la ins. no es relativa ni al contador ni a la base";

}

}

else if (buscarTABSIB(linea.sDireccion.Substring(1)) == null)

{

linea.sCodigoObjeto = "ERROR simbolo no existe";

}

}

else if (linea.sDireccion.Contains(","))

{

int index = linea.sDireccion.IndexOf(",");

string nDireccion = linea.sDireccion.Substring(0, index);

if (this.esConstante(nDireccion))

{

string nixbpe = "111000";

string sCodigo = this.codInsXE(linea.sCodigoOp) + nixbpe;

string sDireccion = linea.sDireccion.Substring(1);

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + sDireccion;

}

else if (buscarTABSIB(nDireccion) != null)

{

if (this.relativoContador(this.buscarTABSIB(nDireccion), this.sContador(this.ensamblador.programa.IndexOf(linea))))

{

string nixbpe = "111010";

string sCodigo = this.codInsXE(linea.sCodigoOp) + nixbpe;

int iTA = Convert.ToInt32(this.buscarTABSIB(nDireccion), 16);

int iPC = Convert.ToInt32(this.sContador(this.ensamblador.programa.IndexOf(linea)), 16);

int iDes = iTA - iPC;

string sDesplazamiento = iDes.ToString("X");

if (iDes < 0)

{

string sDes = this.desplazamiento(sDesplazamiento);

linea.sCodigoObjeto = sCodigo + sDes;

}

else

{

linea.sCodigoObjeto = sCodigo + sDesplazamiento;

}

}

else if (this.relativoBase(this.buscarTABSIB(nDireccion), this.buscaBase()))

{

string nixbpe = "111100";

string sCodigo = this.codInsXE(linea.sCodigoOp) + nixbpe;

int iTA = Convert.ToInt32(this.buscarTABSIB(nDireccion), 16);

int iBase = Convert.ToInt32(this.buscaBase(), 16);

int iDes = iTA - iBase;

string sDesplazamiento = iDes.ToString("X");

if (iDes < 0)

{

string sDes = this.desplazamiento(sDesplazamiento);

linea.sCodigoObjeto = sCodigo + sDes;

}

else

{

linea.sCodigoObjeto = sCodigo + sDesplazamiento;

}

}

else

{

linea.sCodigoObjeto = "ERROR la ins. no es relativa ni al contador ni a la base";

}

}

else if (buscarTABSIB(nDireccion) == null)

{

linea.sCodigoObjeto = "ERROR simbolo no existe";

}

}

else

{

string nDireccion = linea.sDireccion;

if (this.esConstante(nDireccion))

{

string nixbpe = "110000";

string sCodigo = this.codInsXE(linea.sCodigoOp) + nixbpe;

string sDireccion = linea.sDireccion.Substring(1);

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + sDireccion;

}

else if (buscarTABSIB(linea.sDireccion) != null)

{

if (this.relativoContador(this.buscarTABSIB(linea.sDireccion), this.sContador(this.ensamblador.programa.IndexOf(linea))))

{

string nixbpe = "110010";

string sCodigo = this.codInsXE(linea.sCodigoOp) + nixbpe;

int iTA = Convert.ToInt32(this.buscarTABSIB(linea.sDireccion), 16);

int iPC = Convert.ToInt32(this.sContador(this.ensamblador.programa.IndexOf(linea)), 16);

int iDes = iTA - iPC;

string sDesplazamiento = iDes.ToString("X");

if (iDes < 0)

{

string sDes = this.desplazamiento(sDesplazamiento);

linea.sCodigoObjeto = sCodigo + sDes;

}

else

{

linea.sCodigoObjeto = sCodigo + sDesplazamiento;

}

}

else if (this.relativoBase(this.buscarTABSIB(linea.sDireccion), this.buscaBase()))

{

string nixbpe = "110100";

string sCodigo = this.codInsXE(linea.sCodigoOp) + nixbpe;

int iTA = Convert.ToInt32(this.buscarTABSIB(linea.sDireccion), 16);

int iBase = Convert.ToInt32(this.buscaBase(), 16);

int iDes = iTA - iBase;

string sDesplazamiento = iDes.ToString("X");

if (iDes < 0)

{

string sDes = this.desplazamiento(sDesplazamiento);

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + sDes;

}

else if (iDes > 0)

{

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + sDesplazamiento;

}

}

else

{

linea.sCodigoObjeto = "ERROR la ins. no es relativa ni al contador ni a la base";

}

}

else if (buscarTABSIB(linea.sDireccion.Substring(1)) == null)

{

linea.sCodigoObjeto = "ERROR simbolo no existe";

}

}

}

else if (this.formatoInstruccion(linea.sCodigoOp) == 4)

{

if (linea.sDireccion.Contains("#"))

{

//nixbpe

//010001

string nixbpe = "010001";

string direccion = linea.sDireccion.Replace("#", "");

string valorDireccion = this.buscarTABSIB(direccion);

if (valorDireccion != null)

{

string sCodigo = this.codInsXE(linea.sCodigoOp.Replace("+", "")) + nixbpe;

if (valorDireccion.Length < 6)

{

for (int i = 0; i <= 6 - valorDireccion.Length; i++)

{

valorDireccion = "0" + valorDireccion;

}

}

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + valorDireccion;

}

else

{

linea.sCodigoObjeto = "ERROR simbolo no existe";

}

}

else if (linea.sDireccion.Contains("@"))

{

//nixbpe

//100001

string nixbpe = "100001";

string direccion = linea.sDireccion.Replace('@', ' ');

string valorDireccion = this.buscarTABSIB(direccion);

string sCodigo = this.codInsXE(linea.sCodigoOp.Replace("+", "")) + nixbpe;

if (valorDireccion.Length <= 5)

{

for (int i = 0; i < 5 - valorDireccion.Length; i++)

{

valorDireccion = "0" + direccion;

}

}

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + valorDireccion;

}

else if (linea.sDireccion.Contains(","))

{

string nixbpe = "111001";

string trim = ",";

string[] aRegistros = linea.sDireccion.Split(trim[0]);

foreach (var a in aRegistros)

{

if (a != "X")

{

try

{

string direccion = a.Replace("H", "");

int iN1 = Convert.ToInt32(direccion, 16);

//string direccion = a.Replace("H", "");

string sCodigo = this.codInsXE(linea.sCodigoOp.Replace("+", "")) + nixbpe;

if (direccion.Length <= 6)

{

for (int i = 0; i <= 6 - direccion.Length; i++)

{

direccion = "0" + direccion;

}

}

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + direccion;

}

catch

{

string direccion = a.Replace("H", "");

string valorDireccion = this.buscarTABSIB(direccion);

string sCodigo = this.codInsXE(linea.sCodigoOp) + nixbpe;

if (direccion.Length <= 5)

{

for (int i = 0; i < 5 - direccion.Length; i++)

{

direccion = "0" + direccion;

}

}

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + direccion;

}

}

}

}

else if (this.esConstante(linea.sDireccion))

{

string nixbpe = "110001";

string sCodigo = this.codInsXE(linea.sCodigoOp.Replace("+", "")) + nixbpe;

string direccion = linea.sDireccion;

if (direccion.Length <= 5)

{

for (int i = 0; i < 5 - direccion.Length; i++)

{

direccion = "0" + direccion;

}

}

linea.sCodigoObjeto = this.regresarHexa(sCodigo) + direccion;

}

else if (this.esConstante(linea.sDireccion))

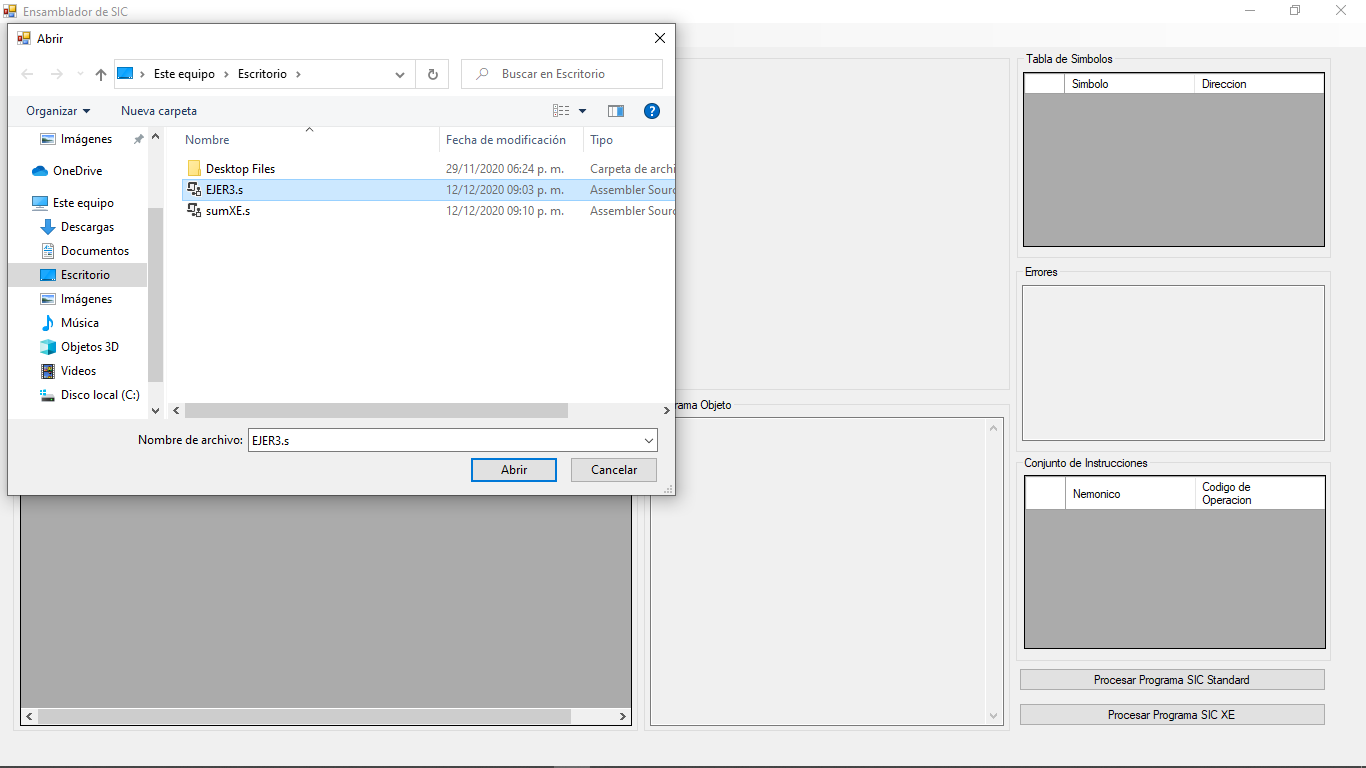
{

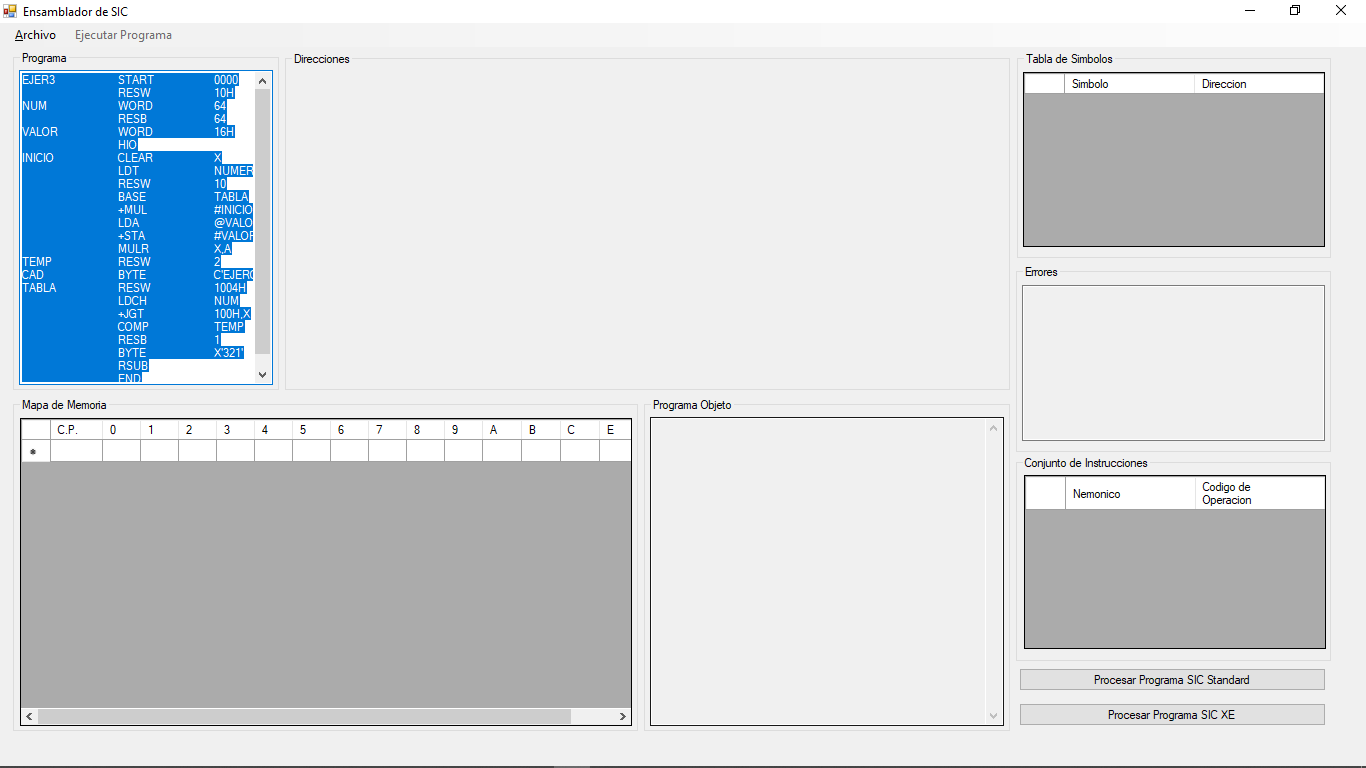
linea.sCodigoObjeto = "ERROR OPERANDO NO EXISTE";

}

}

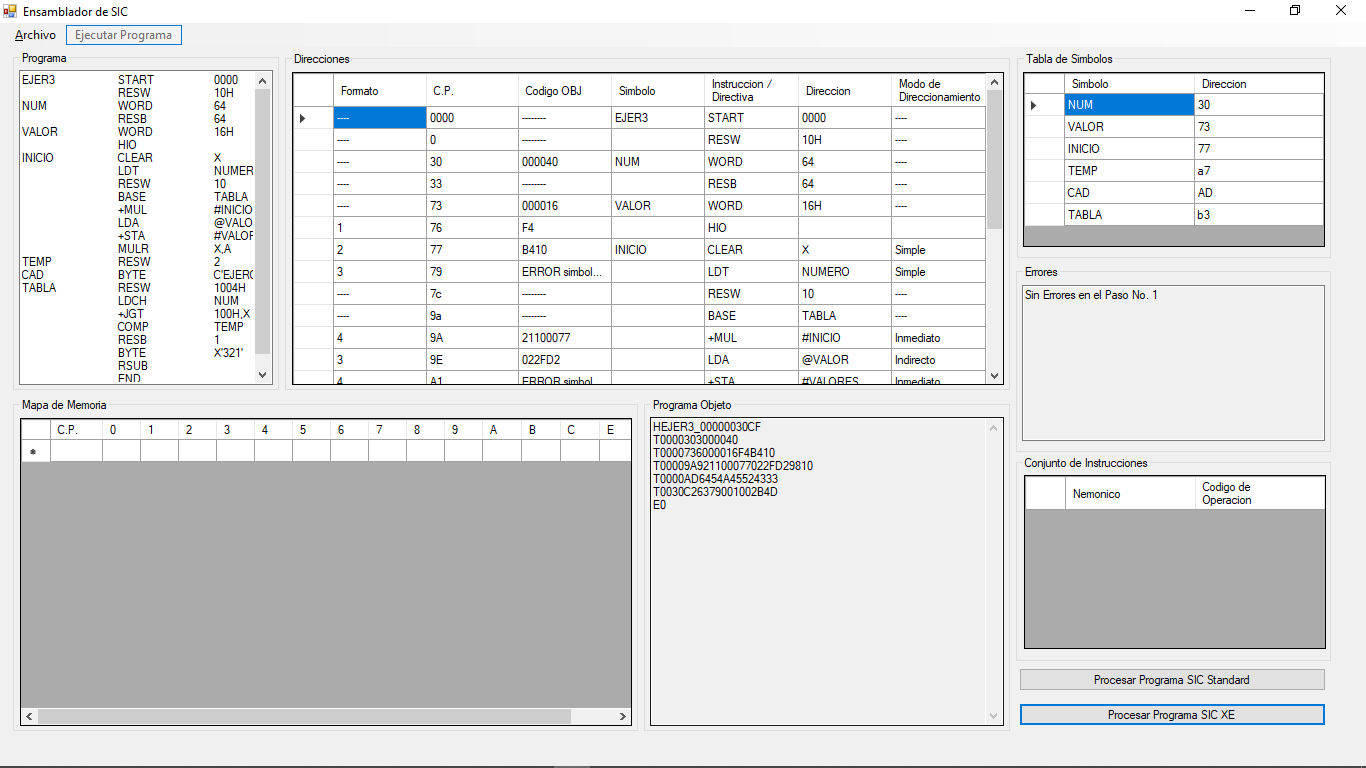
# EJECUCION DEL PROGRAMA





* Hacemos click en **PROCESAR PROGRAMA SIC XE**  para ejecutar nuestro programa con sic XE





Finalmente se crea todo el programa obj de la sic xe