PRACTICA NO. 9

Laboratorio Programación de Sistemas

16 DE NOVIEMBRE DE 2020

FACULTAD DE INGENIERIA
AREA DE CIENCIAS DE LA COMPUTACION

Objetivo

Ejecutar las instrucciones de un programa objeto en el mapa de memoria de una SIC Estándar.

ESTRUCTURA DEL PROGRAMA

Para la ejecución del programa en sic estándar se guarda el mapa de memoria en un diccionario de datos en el que cada registro es una posición del contador de programa, el programa inicia con la primera dirección registrada en ese diccionario y en cada recorrido aumenta en 3 el contador, y para las instrucciones se usa el siguiente código:

```
private int instruccion(string instruccion, string m, int cp, string sCodigo, int i)
    {
        string bytesInstruccion = "";
        if (instruccion == "ADD")
        {
            //A <- (A) + (m..m + 2)
            bytesInstruccion = sCodigo + m;
            if (sRegistroA == "")
                sRegistroA = "000000";
            iRegistroA = Convert.ToInt32(this.sRegistroA, 16);
            string sRegM = this.registroM(Convert.ToInt32(m, 16));
            if (sRegM == "")
                sRegM = "00";
            int iValorM = Convert.ToInt32(sRegM, 16);

            int iLimite = 6 - iValorM.ToString().Length;
           // for(int )

            iRegistroA = iRegistroA + iValorM;
            for (int iii = 0; iii < iLimite; iii++)
                sRegistroA = "0" + sRegistroA;
            m = $"{iRegistroA:X}";

            this.sRegistroA = $"{iRegistroA:X}";
            for (int iii = 0; iii < iLimite; iii++)
                sRegistroA = "0" + sRegistroA;
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "ADD m", "A <- " +
this.sRegistroA));
            sCP = $"{i + 3:X}";

            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "AND")
        {
            //A <- (A) & (m..m + 2)
            if (sRegistroA == "")
```

```
                sRegistroA = "000000";
            iRegistroA = Convert.ToInt32(this.sRegistroA, 16);
            string sRegM = this.registroM(Convert.ToInt32(m, 16));
            if (sRegM == "")
                sRegM = "00";
            int iValorM = Convert.ToInt32(sRegM, 16);
            iRegistroA = iRegistroA & iValorM;
            m = $"{iRegistroA:X}";
            bytesInstruccion = sCodigo + m;
            this.sRegistroA = $"{iRegistroA:X}";
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "AND m", "A <- " +
this.sRegistroA));
            sCP = $"{i + 3:X}";
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "COMP")
        {
            //A : (m..m+2)
            if (sRegistroA == "")
                sRegistroA = "000000";
            string sRegM = this.registroM(Convert.ToInt32(m, 16));
            iRegistroM = Convert.ToInt32(sRegistroM);
            iRegistroA = Convert.ToInt32(sRegistroA);
            if (iRegistroM > iRegistroA)
            {
                this.CC = ">";
            }
            else if (iRegistroM < iRegistroA)
            {
                this.CC = "<";
            }
            else
            {
                this.CC = "=";
            }
            bytesInstruccion = sCodigo + m;
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "COMP m", "CC <- " +
this.CC));
            sCP = $"{i + 3:X}";
            return Convert.ToInt32(sCP, 16);

        }
        if (instruccion == "DIV")
        {
            //A <- (A) / (m..m + 2)
            if (sRegistroA == "")
                sRegistroA = "000000";
```

```
            iRegistroA = Convert.ToInt32(this.sRegistroA, 16);
            string sRegM = this.registroM(Convert.ToInt32(m, 16));
            if (sRegM == "")
                sRegM = "00";
            int iValorM = Convert.ToInt32(sRegM, 16);
            iRegistroA = iRegistroA / iValorM;
            m = $"{iRegistroA:X}";
            bytesInstruccion = sCodigo + m;
            this.sRegistroA = $"{iRegistroA:X}";
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "DIV m", "A <- " + this.sRegistroA));
            sCP = $"{i + 3:X}";
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "J")
        {
            //CP <- m
            string sRegM = this.registroM(Convert.ToInt32(m, 16));
            this.sCP = sRegM;
            bytesInstruccion = sCodigo + m;
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "J m", "CP <- " + this.sCP));
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "JEQ")
        {
            //CP <- m si CC =
            if (CC == "=")
            {
                string sRegM = this.registroM(Convert.ToInt32(m, 16));
                this.sCP = sRegM;
                bytesInstruccion = sCodigo + m;
                this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "JEQ m", "CP <- " + this.sCP));
                return Convert.ToInt32(sCP, 16);
            }

            return i + 3;
        }
        if (instruccion == "JGT")
        {
            //CP <- m si CC >
            if (CC == ">")
            {
                string sRegM = this.registroM(Convert.ToInt32(m, 16));
                this.sCP = sRegM;
                bytesInstruccion = sCodigo + m;
```

```
                this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "JGT m", "CP <- " + this.sCP));
                return Convert.ToInt32(sCP, 16);
            }

            return i + 3;
        }
        if (instruccion == "JLT")
        {
            //CP <- m si CC >
            if (CC == "<")
            {
                string sRegM = this.registroM(Convert.ToInt32(m, 16));
                this.sCP = sRegM;
                bytesInstruccion = sCodigo + m;
                this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "JLT m", "CP <- " + this.sCP));
                return Convert.ToInt32(sCP, 16);
            }

            sCP = $"{i + 3:X}";
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "JSUB")
        {
            //L <- (CP):
            //CP <- m;
            this.sRegistroL = this.sCP;
            this.sCP = m;
            bytesInstruccion = sCodigo + m;
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "JSUB m", "L <- " + this.sRegistroL + "; CP <-" + this.sCP));
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "LDA")
        {
            //A <- (m)...(m+2)
            if (sRegistroA == "")
                sRegistroA = "000000";
            string sRegM = this.regresaM(Convert.ToInt32(m, 16), 3);
            if (sRegM == "")
                sRegM = "000000";
            sRegistroA = sRegM;
            bytesInstruccion = sCodigo + m;
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "LDA m", "A <- " + this.sRegistroA));
            sCP = $"{i + 3:X}";
```

```
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "LDCH")
        {
            //A <- [Byte mas a la derecha de m]
            if (sRegistroA == "")
                sRegistroA = "000000";
            string sRegM = this.registroM(Convert.ToInt32(m, 16));
            iRegistroA = Convert.ToInt32(this.sRegistroA, 16);
            //List<int> valorM = this.valorM(Convert.ToInt32(m, 16));
            sRegistroA = sRegM[0].ToString().Substring(1);
            bytesInstruccion = sCodigo + m;
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "LDCH m", "A <- " +
this.sRegistroA));
            sCP = $"{i + 3:X}";
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "LDL")
        {
            //L <- (m)...(m+2)
            string sRegM = this.registroM(Convert.ToInt32(m, 16));
            sRegistroL = sRegM;
            bytesInstruccion = sCodigo + m;
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "LDL m", "A <- " +
this.sRegistroA));
            sCP = $"{i + 3:X}";
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "LDX")
        {
            //X <- (m)...(m+2)
            if (sRegistroX == "")
                sRegistroX = "000001";
            string sRegM = this.regresaM(Convert.ToInt32(m, 16), 3);
            sRegistroX = sRegM;
            bytesInstruccion = sCodigo + m;
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "LDX m", "X <- " +
this.sRegistroX));
            sCP = $"{i + 3:X}";
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "MUL")
        {
            //A <- (A) * (m..m + 2)
            if (sRegistroA == "")
                sRegistroA = "000000";
            iRegistroA = Convert.ToInt32(this.sRegistroA, 16);
```

```
            string sRegM = this.registroM(Convert.ToInt32(m, 16));
            if (sRegM == "")
                sRegM = "00";
            int iValorM = Convert.ToInt32(sRegM, 16);
            iRegistroA = iRegistroA * iValorM;
            m = ${iRegistroA:X};
            bytesInstruccion = sCodigo + m;
            this.sRegistroA = ${iRegistroA:X};
            this.programa.Add(new Ejecucion(${cp:X}", bytesInstruccion, "MUL m", "A <- " +
this.sRegistroA));
            sCP = ${i + 3:X};
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "OR")
        {
            //A <- (A) | (m..m + 2)
            if (sRegistroA == "")
                sRegistroA = "000000";
            iRegistroA = Convert.ToInt32(this.sRegistroA, 16);
            string sRegM = this.registroM(Convert.ToInt32(m, 16));
            if (sRegM == "")
                sRegM = "00";
            int iValorM = Convert.ToInt32(sRegM, 16);
            iRegistroA = iRegistroA | iValorM;
            m = ${iRegistroA:X};
            bytesInstruccion = sCodigo + m;
            this.sRegistroA = ${iRegistroA:X};
            this.programa.Add(new Ejecucion(${cp:X}", bytesInstruccion, "OR m", "A <- " +
this.sRegistroA));
            sCP = ${i + 3:X};
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "RSUB")
        {
            //PC <- L
            this.sRegistroPC = sRegistroL;
            //List<int> valorM = this.valorM(Convert.ToInt32(m, 16));
            bytesInstruccion = sCodigo + m;
            this.programa.Add(new Ejecucion(${cp:X}", bytesInstruccion, "RSUB m", "PC <- " +
this.sRegistroPC));
            //sCP = ${i + 3:X};
            sCP = this.sRegistroPC;
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "STA")
        {
            //m .. m + 2 <- (A)
```

```
            string srm = this.regresaM(Convert.ToInt32(m, 16), 3);
            if (sRegistroA == "")
                sRegistroA = "000000";

            int iLimite = sRegistroA.Length;
            for (int yi = 0; yi < 6 - iLimite; yi++)
                sRegistroA = "0" + sRegistroA;

            int iRegM = Convert.ToInt32(m, 16);

            this.modificarM(iRegM, sRegistroA);
            bytesInstruccion = sCodigo + m;
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "STA m", "PC <- " +
this.sRegistroPC));
            sCP = $"{i + 3:X}";
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "STCH")
        {
            // m <- a[byte as a la derecha]
            int iRegM = Convert.ToInt32(m, 16);
            this.modificarM(iRegM, sRegistroA[sRegistroA.Length - 1].ToString());
            bytesInstruccion = sCodigo + this.valorM(iRegM);
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "STCH m", "m .. m + 2 <- " +
this.sRegistroA));
            sCP = $"{i + 3:X}";
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "STL")
        {
            //m .. m + 2 <- L
            int iRegM = Convert.ToInt32(m, 16);
            this.modificarM(iRegM, sRegistroL);
            bytesInstruccion = sCodigo + this.valorM(iRegM);
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "STL m", "m .. m + 2 <- " +
this.sRegistroL));
            sCP = $"{i + 3:X}";
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "STSW")
        {
            //m .. m + 2 <- sw
            int iRegM = Convert.ToInt32(m, 16);
            this.modificarM(iRegM, sRegistroW);
            bytesInstruccion = sCodigo + this.valorM(iRegM);
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "STSW m", "m .. m + 2 <- " +
this.sRegistroW));
```

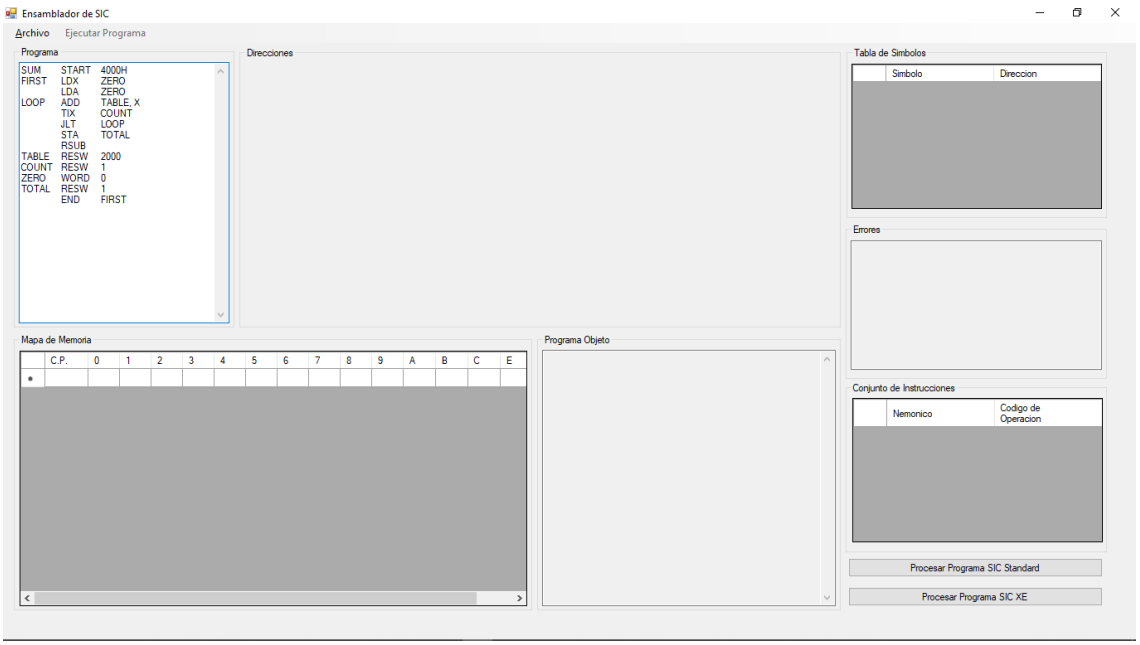
```
            sCP = $"{i + 3:X}";
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "STX")
        {
            //m .. m + 2 <- x
            if (sRegistroX == "")
                sRegistroX = "000000";
            int iRegM = Convert.ToInt32(m, 16);
            this.modificarM(iRegM, sRegistroX);
            bytesInstruccion = sCodigo + this.valorM(iRegM);
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "STSX m", "m .. m + 2 <- " +
this.sRegistroX));
            sCP = $"{i + 3:X}";
            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "SUB")
        {
            //A <- (A) + (m..m + 2)
            if (sRegistroA == "")
                sRegistroA = "000000";
            iRegistroA = Convert.ToInt32(this.sRegistroA, 16);
            string sRegM = this.registroM(Convert.ToInt32(m, 16));
            if (sRegM == "")
                sRegM = "00";
            int iValorM = Convert.ToInt32(sRegM, 16);
            iRegistroA = iRegistroA - iValorM;
            m = $"{iRegistroA:X}";
            bytesInstruccion = sCodigo + sRegM;
            this.sRegistroA = $"{iRegistroA:X}";
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "SUB m", "A <- " +
this.sRegistroA));
            sCP = $"{i + 3:X}";

            return Convert.ToInt32(sCP, 16);
        }
        if (instruccion == "TD")
        {
            // no se que hace
        }
        if (instruccion == "TIX")
        {
            //X <- X + 1; (X) : X : M .. M + 2
            if (sRegistroX == "")
                sRegistroX = "000000";
            iRegistroX = Convert.ToInt32(sRegistroX, 16) + 1;
            string srm = this.regresaM(Convert.ToInt32(m, 16),3);
```

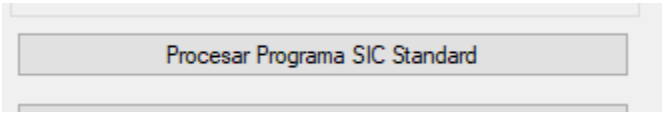
```
            if (srm == "")
                srm = "00";
            int im = Convert.ToInt32(srm, 16);
            if (iRegistroX < im)
            {
                this.CC = "<";
            }
            if (iRegistroX > im)
            {
                this.CC = ">";
            }
            if (iRegistroX == im)
            {
                this.CC = "=";
            }
            bytesInstruccion = sCodigo + m;
            this.programa.Add(new Ejecucion($"{cp:X}", bytesInstruccion, "TIX m", "X <- X + 2 : " + m));
            sCP = $"{i + 3:X}";
            return Convert.ToInt32(sCP, 16);
        }
        return 0;
    }
```

Cada contador de programa accede a una de estas instrucciones y ellas mismas realizan su respetico acción

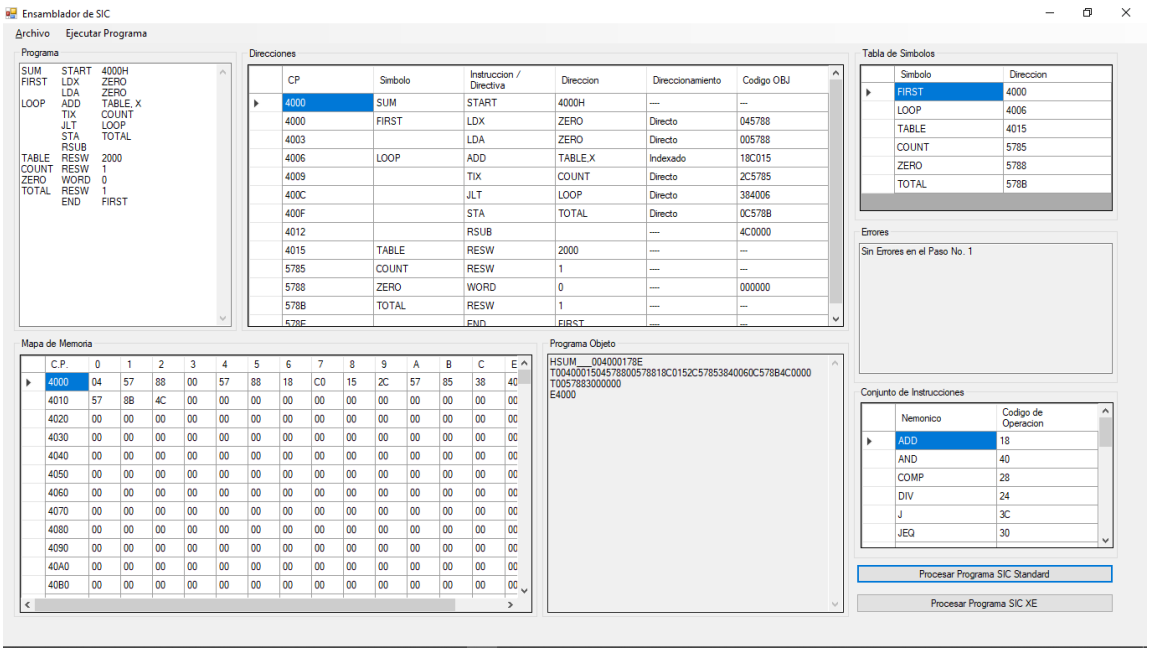
Ejecución del programa



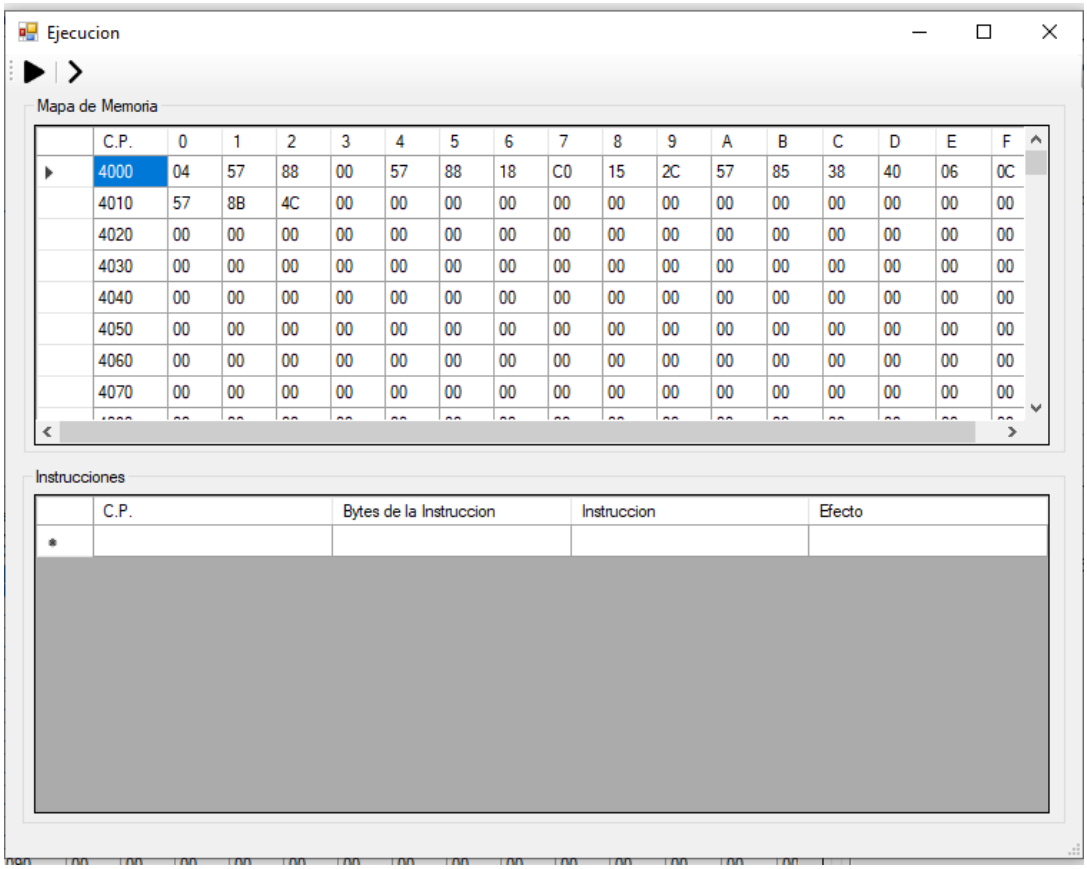
Hacemos click en **Procesar Programa SIC Standard**



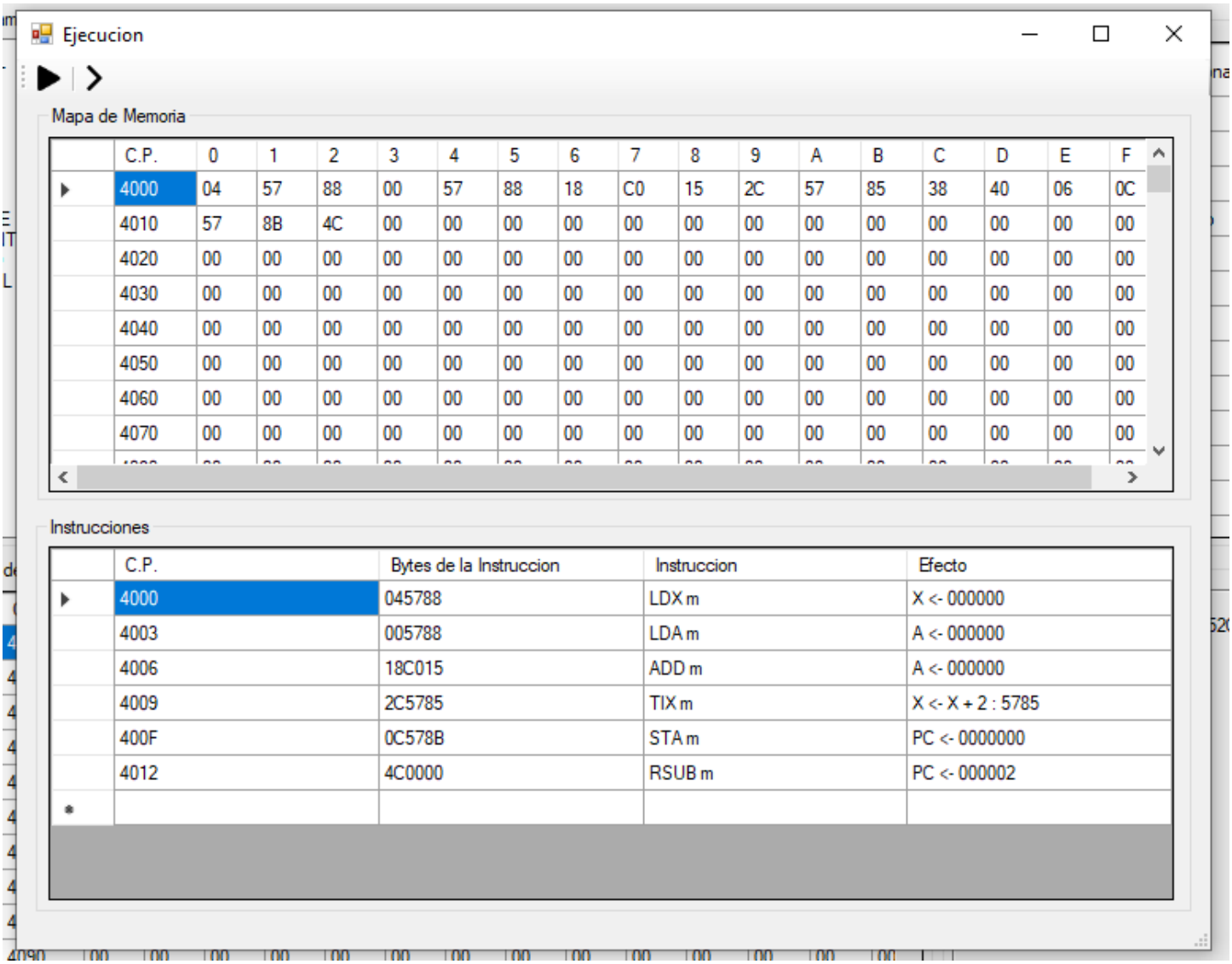
Se llenará el mapa de memoria, tabsim y se creará el código objeto del programa



Para ejecutar el programa haremos click en **Ejecutar Programa** y aparecerá una nueva ventana con el mapa de memoria



Solo falta hacer click en el botón ejecutar y el programa se ejecutará



C.P.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4000	04	57	88	00	57	88	18	C0	15	2C	57	85	38	40	06	0C
4010	57	8B	4C	00	00	00	00	00	00	00	00	00	00	00	00	00
4020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

C.P.	Bytes de la Instruccion	Instruccion	Efecto
4000	045788	LDX m	X <- 000000
4003	005788	LDA m	A <- 000000
4006	18C015	ADD m	A <- 000000
4009	2C5785	TIX m	X <- X + 2 : 5785
400F	0C578B	STA m	PC <- 0000000
4012	4C0000	RSUB m	PC <- 000002