PRACTICA 08 MAPA DE MEMORIA

Rogelio Daniel Gonzalez Nieto
FACULTAD DE INGENIERIA AREA DE CIENCIAS DE LA COMPUTACION

OBJETIVO

Crear el mapa de memoria en base al código objeto obtenido en la practica no. 6

DESARROLLO

Para el desarrollo de esta práctica primero se separa el código objeto del programa en grupos de 2 caracteres que representan 1 bit de memoria con la siguiente función:

```
private List<string> separarCodigo(string linea)
{
    string n = "";
    string separado = linea.Substring(9);
    List<string> lista = new List<string>();

    foreach (var s in separado)
    {
        n += s;
        if (n.Length == 2)
        {
            lista.Add(n);
            n = "";
        }
    }
}

return lista;
}
```

Se uso un diccionario como estructura de datos para el manejo de memoria, como key del diccionario se usó la dirección de inicio de escritura y como value se uso una lista de string con el código objeto ya separado, en base a la dirección en el key y a la dirección de inicio del código objeto se determina donde empezar a guardar los datos en el diccionario por medio del siguiente código:

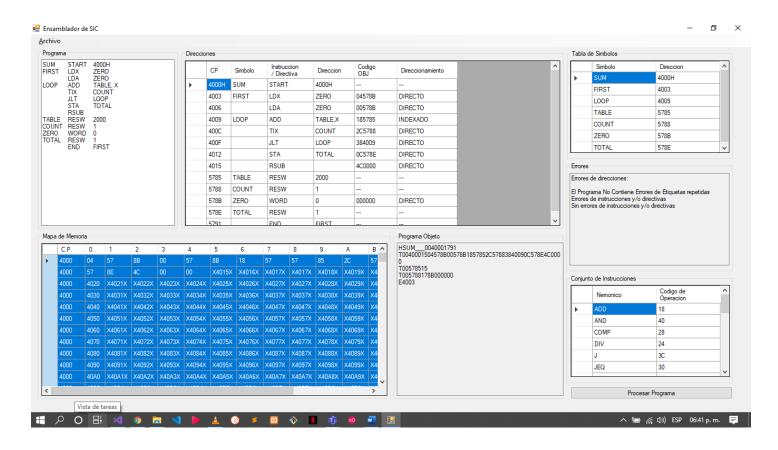
```
bandera = false;
}
}
}
```

Después de crearse el diccionario este se muestra en un datagrid con el siguiente código:

```
private void mapaMemoria()
    this.dataGridView4.Rows.Clear();
    int inicioProgrma = Convert.ToInt32(this.sTamaInicial, 16);
    int finalProgrma = Convert.ToInt32(this.sTamaFinal, 16);
    int inicio = inicioProgrma;
    int final = this.RoundUp(finalProgrma);
    string sDirIni = $"{inicio:X}";
    string sDirFin = $"{final:X}";
    string direccion = "";
    string dirInicio = "";
    List<string> lAux = new List<string>();
    for (int i = inicio; i < final; i += 16)</pre>
        direccion = $"{i:X}";
        lAux = new List<string>();
        lAux.Add(direccion);
        lAux.Add("X" + direccion.Substring(0, 3) + "1X");
        lAux.Add("X" + direccion.Substring(0, 3) + "2X");
        lAux.Add("X" + direccion.Substring(0, 3) + "3X");
        lAux.Add("X" + direccion.Substring(0, 3) + "4X");
        lAux.Add("X" + direccion.Substring(0, 3) + "5X");
        lAux.Add("X" + direccion.Substring(0, 3) + "6X");
        lAux.Add("X" + direccion.Substring(0, 3) + "7X");
        lAux.Add("X" + direccion.Substring(0, 3) + "8X");
        lAux.Add("X" + direccion.Substring(0, 3) + "9X");
        lAux.Add("X" + direccion.Substring(0, 3) + "AX");
        lAux.Add("X" + direccion.Substring(0, 3) + "BX");
        lAux.Add("X" + direccion.Substring(0, 3) + "CX");
        lAux.Add("X" + direccion.Substring(0, 3) + "DX");
        lAux.Add("X" + direccion.Substring(0, 3) + "EX");
        lAux.Add("X" + direccion.Substring(0, 3) + "FX");
        this.dMapaMemoria.Add(direccion, lAux);
    }
    foreach (var 1 in lRegistros)
        if (1 != lRegistros[0] && 1 != lRegistros[lRegistros.Count - 1] && l.Length > 10)
            List<string> objs = this.separarCodigo(1);
            string sDireccion = 1.Substring(1, 6);
            int iDireccion = Convert.ToInt32(sDireccion, 16);
            sDireccion = $"{iDireccion:X}";
            this.llenarMapa(sDireccion, objs);
    }
    int j = 0;
```

```
int k = 0;
List<string> sHeader = new List<string>();
for (int i = inicio; i < final; i += 16)
{
         direccion = $"{i:X}";
         sHeader.Add(direccion);
}

foreach (var d in this.dMapaMemoria)
{
         this.dataGridView4.Rows.Add(sHeader[j], d.Value[0], d.Value[1], d.Value[2], d.Value[3],
d.Value[4], d.Value[5], d.Value[6], d.Value[7], d.Value[7], d.Value[8], d.Value[9], d.Value[10],
d.Value[11], d.Value[12], d.Value[13], d.Value[14], d.Value[15]);
}
</pre>
```



Las celdas con valores con X son solo celdas de ayuda para la posición de memoria, las demás son el código objeto anteriormente obtenido