1 Código de las clases principales

1.1 CPU

CPU

```
    public static class CPU
    {
        public static Alu.Alu Alu { get; private set; }
        public static Banderas Banderas { get; private set; }
        public static Memoria Memoria { get; private set; }
        static CPU()
        {
            CPU.Alu = new Alu.Alu();
            CPU.Banderas = new Banderas();
            CPU.Memoria = new Memoria();
            Reset();
        }
        public static void Reset()
        {
            CPU.Banderas.Clear();
            CPU.Memoria.Clear();
            Registros.Registros.Reset();
        }
        public static void Ejecutar(bool[] Operacion, bool[] Modificador,
         bool[] Operador1, bool[] Operador2){...}
    }
```

1.2 ALU

ALU

```
    public class Alu
    {   public const int Byte = 16;
        public bool[] Resultado = new bool[Byte * 2 + 1];
        public void ADD(bool[] Operador1, bool[] Operador2){ ... }
        private bool HALF_ADD(bool A, bool B){ ... }
        private bool FULL_ADD(bool A, bool B){ ... }
        public void SUB(bool[] Operador1, bool[] Operador2){ ... }
        public bool[] COMPLEMENTO_2(bool[] Operador1){ ... }
        private bool AND(bool A, bool B){ ... }
        public void AND(bool[] Operador1, bool[] Operador2){ ... }
        public void OR(bool[] Operador1, bool[] Operador2){ ... }
        public void NAND(bool[] Operador1, bool[] Operador2){ ... }
        public void NOR(bool[] Operador1, bool[] Operador2){ ... }
        public void MUL(bool[] Operador2){ ... }
        public void NOT(bool[] Operador1){ ... }
        private bool XOR(bool Operador1, bool Operador2){ ... }
        public void XOR(bool[] Operador1, bool[] Operador2){ ... }
        public void XNOR(bool[] Operador1, bool[] Operador2){ ... }
        public void DIV(bool[] Divisor){ ... }
    }
```

1.3 Registros

Registros

```
public static class Registros
{
    public static Registro AX { get; private set; }
    public static Registro BX { get; private set; }
    public static Registro CX { get; private set; }
    public static Registro DX { get; private set; }
    public static Registro SI { get; private set; }
    public static Registro DI { get; private set; }
    public static Registro IP { get; private set; }
    public static Registro IA { get; private set; }
    public static Registro IR { get; private set; }
    static Registros()
    {
        Registros.AX = new Registro("AX");
        Registros.BX = new Registro("BX");
        Registros.CX = new Registro("CX");
        Registros.DX = new Registro("DX");
        Registros.SI = new Registro("SI");
        Registros.DI = new Registro("DI");
        Registros.IP = new Registro("IP");
        Registros.IA = new Registro("IA");
        Registros.IR = new Registro("IR");
    }
    internal static void Reset()
    {
        Registros.AX.Clear();
        Registros.BX.Clear();
        Registros.CX.Clear();
        Registros.DX.Clear();
        Registros.SI.Clear();
        Registros.DI.Clear();
        Registros.IP.Clear();
    }
}
```

1.4 Registro

Registro

```
public class Registro : Localidad
{
    public string Nombre { get; private set; }
    private ParteRegistro High;
    public ParteRegistro Low;
    public void SetHigh(bool[] High){ ... }
    public void SetLow(bool[] Low){ ... }
}
```

1.5 Memoria

Memoria

```
public class Memoria
{
    public bool[] this[bool[] direccion]
    {
        set
        {
            Escribir(direccion, value);
        }
    }
    private ObservableCollection<Celda> Real;

    public void Cargar(string CodigoMaquina) { ... }
    public void Cargar(bool[][] programa) { ... }
    public static bool[] CalcularDireccion(bool[] Numero) { ... }
    internal void Clear() { ... }
    public bool[] Leer(bool[] direccion) { ... }
    public void Escribir(bool[] direccion, bool[] Valor){ ... }
}
```

1.6 Banderas

Banderas

```
public class Banderas :
{
    private bool Carry;
    private bool Signo;
    private bool Zero;
    private bool OverFlow;
    internal void Clear()
    {
        Carry = false;
        Signo = false;
        Zero = false;
        OverFlow = false;
    }
}
```