PARADIGMAS 3 (1) 13-34 (2011)
Revista del Departamento Académico de Ingeniería.
Universidad Ricardo Palma. ISSN: 2220-2056

Velocidad de procesador y de proceso

Processor and Process speed

Francisco Santiago Aguilar Vásquez[1]

Resumen

Descripción organizacional de un procesador básico. Rol del reloj. Influencia de la velocidad de proceso de los hilos computacionales en su enseñanza - aprendizaje. Control de la velocidad de proceso de los hilos computacionales. Programación de un reloj de periodo definido por el usuario y de un hilo computacional controlado por este reloj.

Palabras claves

Procesador, reloj, hilo, velocidad de proceso, significatividad en enseñanza-aprendizaje

Abstract

Organizational description of a core processor. Role of the clock. Influence of processing speed of the processing threads in their teaching - learning. Control of the processing speed of the processing threads. Programming a clock, with a period defined by the user, and a processing thread, controlled by this clock.

Key words

Processor, clock, thread, processing speed, significance in teaching – learning.

Introducción

En la actualidad es una preocupación constante desarrollar computadores cada vez más veloces para satisfacer los requerimientos de proceso de información. Se destacan dos alternativas referenciales para lograr más velocidad en computadores: desarrollar procesadores (microprocesadores) más veloces para computadores monoprocesador y desarrollar computadores multiprocesador. En la práctica se combinan ambas alternativas, [ver 5], resultando computadores muy veloces (potentes) que incluyen microprocesadores que ejecutan, inclusive miles, de millones de operaciones (instrucciones) por segundo.

Paradójicamente existen situaciones en las cuales se desea que el computador funcione a velocidades menores, como sucede durante el proceso de enseñanza-aprendizaje de los hilos de proceso computacional. El proceso de enseñanza-aprendizaje en general, y de estos hilos en particular, se realiza a ciertas velocidades. La disponibilidad de hilos de proceso que se ejecuten a velocidades compatibles con estas ciertas velocidades facilitaría su enseñanza-aprendizaje.

1 Ingeniero. Magíster en Computación en Informática. Doctorando en Ingeniería. Docente URP

El presente trabajo, basado en aspectos tratados en [1], describe un hipotético procesador (microprocesador o núcleo) referencial que procesa instrucciones a su velocidad, presenta velocidades significativas de proceso de hilos computacionales para enseñanza-aprendizaje y materializa estas velocidades significativas por medio de un ejemplo de un reloj en software y de un ejemplo de un hilo computacional concurrente de proceso controlado por este reloj.

Velocidad de procesador

Según [6], el procesador, "cerebro" del computador, está compuesto por registros y arreglos combinacionales, intercomunicados y comunicados con los sistemas que constituyen el computador (memoria e interfaces de entrada/salida) por medio de buses, organizados en la unidad de proceso (UP) y la unidad de control (UC) tal como se muestra en la figura 1. La tecnología y la organización de estos registros, arreglos combinacionales y buses definen la velocidad de procesar instrucciones del procesador, dictada por los pulsos generados por el reloj (clock).

Para llevar a cabo los distintos tipos de operaciones, la UP cuenta con la unidad funcional o unidad aritmética - lógica (ALU), registros de propósito general y registros de estado o de control interconectados por el bus interno y sincronizados por las señales de control generadas por la UC.

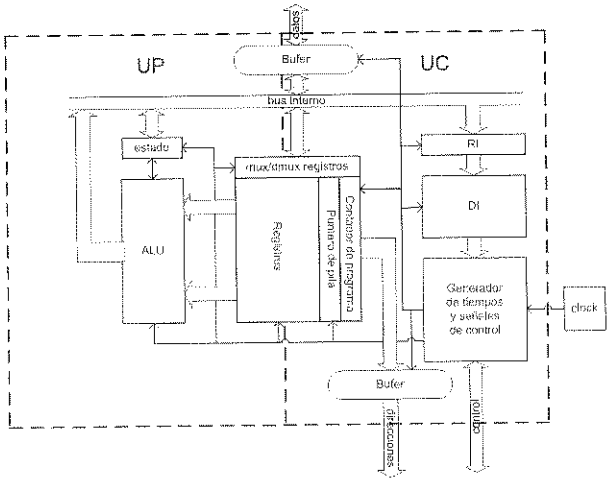

Figura 1. Un procesador (μP)

Las operaciones se realizan sobre un operando (operación unaria) o sobre dos operandos (operación binaria). El operando representa un dato en código binario. Los operandos se toman de registros, memoria o interfaz y el resultado de la operación se dirige a un registro, memoria o interfaz.

La ejecución de operaciones suele afectar a los indicadores de estado. Entre los indicadores más comunes de los procesadores se encuentran: modo de operación, cero, signo, paridad, acarreo, auxiliar, dirección, interrupción, desborde, desvío, parado. Estos indicadores condicionan el procesamiento de determinadas instrucciones del procesador, como los saltos condicionales o llamadas a subrutinas condicionales.

Para ejecutar el cálculo expresado en el siguiente fragmento de código fuente, expresado en un lenguaje de programación general:

```
a=b-c;
d=a+25;
```

el procesador, por ejemplo, deberá realizar las operaciones correspondientes a las instrucciones en lenguaje máquina (en lenguaje ensamblador) que se dan a continuación:

//instrucciones en lenguaje ensamblador para a=b-c

```
carga R1, b      //transferencia de memoria a registro de propósito general
carga R2, c      //transferencia de memoria a registro de propósito general
resta R1, R2     //ALU resta el valor de R2 del valor de R1 y dirige resultado a R1
almacena R1,a    //transferencia de registro de propósito general a memoria
```

//instrucciones en lenguaje ensamblador para d=a+25

```
carga R2,25      //transferencia de memoria primaria a registro de propósito general
suma R1, R2      /ALU suma el valor de R2 al valor de R1 y dirige resultado a R1
almacena R1, d   //transferencia de registro de propósito general a memoria primaria
```

La ALU es el motor de cómputo real del computador. Las operaciones de punto flotante son más complejas que las operaciones sobre enteros, razón por la cual algunas unidades funcionales no incorporan la aritmética de punto flotante. En estos casos las operaciones de punto flotante se realizan por software o en un procesador auxiliar dedicado.

La UC gestiona el procesamiento de instrucciones que constituyen los programas ejecutables almacenados en memoria: determinando la instrucción a procesar, obteniéndolo desde memoria, decodificándolo y ejecutando la operación correspondiente. Para cumplir con esta gestión, como se aprecia en la figura 1, la UC cuenta con un contador de programa (PC), un registro de instrucciones (IR), un decodificador de instrucciones (ID), un puntero de pila (SP), registros auxiliares (AR), un generador de tiempos y señales de control (BTS) y comparte los indicadores con la UP.

El BTS incorpora la base de tiempos y el secuenciador o generador de microinstrucciones que en conjunto generan las señales de temporización y control requeridas para ejecutar las diferentes fases del procesamiento de instrucciones.

El tiempo elemental de la base de tiempos y de funcionamiento del procesador y computador, que determina la velocidad de procesamiento de instrucciones (velocidad del procesador), está definido por el clock (reloj) externo dentro del margen que soporta el procesador. La frecuencia (velocidad) de este reloj alcanza hasta gigahercios. En consecuencia, las instrucciones se ejecutan, inclusive en nanosegundos; velocidades imperceptibles a la vista humana.

El procesador se caracteriza por un repertorio de instrucciones en lenguaje máquina con totalidad funcional. La totalidad funcional requiere de tipos básicos de instrucciones en lenguaje máquina, que incluyen:

- Aritméticas: suma, resta, multiplicación, división
- Lógicas: AND, OR, XOR NOT
- Relativas a registros: desplazamiento, circulación, incremento, decremento, complementos, borrado
- Relativas a indicadores: puesta a uno, puesta a cero
- Intercambio de información entre registros de procesador
- Intercambio de información entre registros y memoria
- Carga inmediata de información en registros o memoria
- Entrada/salida
- Ruptura de secuencia: saltos in/condicionales, llamadas a rutinas in/condicionales, retorno de interrupción.
- Especiales y de control: parar, puesta en marcha, no operación.

Estas instrucciones se codifican en bits (código binario), constituyendo una o más palabras (bytes), y se procesan en uno o más ciclos de máquina (memoria). Cada ciclo de máquina se implementa en unos cuantos ciclos o tiempos elementales definidos por el reloj (clock) del sistema conectado al procesador.

El procesador desde que se activa (se prende), hasta que se desactiva (se apaga) y mientras el indicador parado esté deshabilitado, se encuentra procesando, a su velocidad, instrucciones de su repertorio. A esta realización continua de instrucciones a nivel hardware por parte del procesador se suele denominar hilo de proceso hardware (HPH) y tiene el siguiente esquema lógico.

```
PC <== <dirección inicial de máquina>
parado <== FALSE
mientras (no parado)
{
        IR <== memoria[PC]      //obtiene instrucción desde memoria
        PC <== PC+1             //determina siguiente instrucción a procesar
        decodificar [IR]        //decodifica instrucción
        ejecutar[IR]            //ejecuta operación
}
```

Un programa en lenguaje máquina (ejecutable) de un computador es una secuencia de instrucciones del repertorio de su procesador, que es previamente almacenado en memoria, desde donde es procesado (ejecutado), a velocidad del procesador, en concordancia con el HPH presentado.

En procesamiento concurrente, la memoria contiene múltiples programas ejecutables, aplicativos y del sistema, que durante su procesamiento son multiplexados en el HPH. En este contexto

identificamos la velocidad de proceso de cada programa, que debe ser significativa en la enseñanza aprendizaje de hilo de proceso, y la necesidad de un reloj que produzca esa velocidad significativa de proceso.

Velocidad significativa de proceso.

Los conocimientos sobre hilos computacionales son compartidos por docentes y estudiantes durante su enseñanza - aprendizaje, a través de medios (y material), técnicas y herramientas en un contexto determinado. Gran parte de este material y herramientas lo constituyen los programas operativos y aplicativos pertinentes que el computador los procesa como hilos computacionales. Según [2] y [7], la organización y demostración (velocidad de proceso) de estos programas deben ser significativas. La velocidad significativa de proceso de los hilos computacionales se relaciona con los lenguajes de programación empleados en su construcción y ejecución, así como con el control de la velocidad y dinámica de su procesamiento en el computador por parte de los interesados.

Los programas de computadora suelen escribirse (editarse) en lenguajes de programación orientados al usuario - persona (desarrollador, programador, docente, estudiante) y se traducen al lenguaje de la máquina destino para su procesamiento (ejecución). Normalmente existe una distancia entre el lenguaje de programación (edición) elegido y el lenguaje máquina destino. La relación entre esta distancia y la significatividad para la enseñanza-aprendizaje de hilos computacionales se presenta en la figura 2. La mayor significatividad se logra si el lenguaje de programación coincide con el lenguaje de máquina (distancia 0).

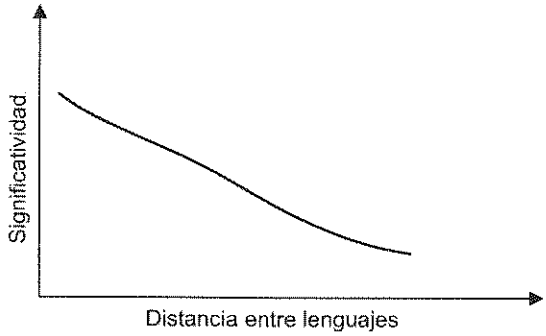

Figura 2. Relación entre distancia y significatividad

Un microprocesador de 3 GHz de computador personal, caracterizado por un juego de instrucciones con promedio de tiempo de ejecución de 10 periodos de reloj, procesa 300 millones de instrucciones de lenguaje máquina por segundo. Considerando 10 instrucciones de lenguaje máquina por instrucción del lenguaje de programación, un computador con este tipo de procesador ejecuta 30 millones de instrucciones de programa fuente por segundo. A esta velocidad, el interesado difícilmente captaría por algún medio (por ejemplo visual) la dinámica de ejecución de los hilos computacionales, especificados en el programa fuente.

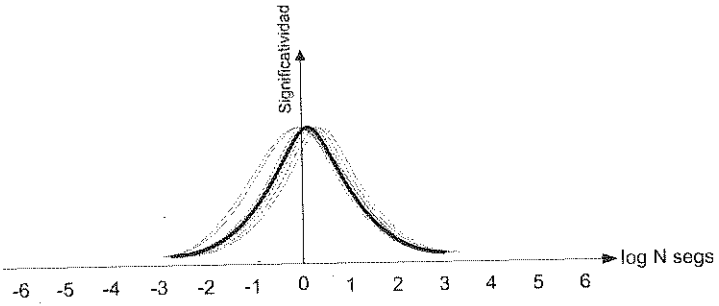

Figura 3. Velocidad de ejecución de programas y significatividad

El proceso de enseñanza – aprendizaje de los hilos computacionales es afectado por la velocidad del computador al ejecutar los programas usados para esta finalidad. Esta influencia se refleja en el gráfico de la figura 3, donde la ejecución referencial que permite acreditarse aceptablemente ronda alrededor de una instrucción de programa fuente por segundo. Tanto la alta velocidad como la muy baja velocidad del computador parecen inadecuadas para la enseñanza-aprendizaje de los hilos computacionales.

Reloj en software.

Para alcanzar una velocidad de proceso apropiada para la enseñanza-aprendizaje de los hilos computacionales construimos un reloj en software, usando el entorno de programación ms vs.Net 2005, o posteriores, con C#. Basado en un temporizador disponible en .NET, este reloj genera pulsos, con la precisión que el computador permite, en periodos desde 1 milisegundo hasta 2 minutos aproximadamente.

Tal como se muestra en la figura 4, la interfaz del reloj brinda al usuario las siguientes facilidades: arrancar y parar el reloj, indicar la pulsación del reloj y variar/ingresar el periodo en milisegundos a partir de un valor inicial.

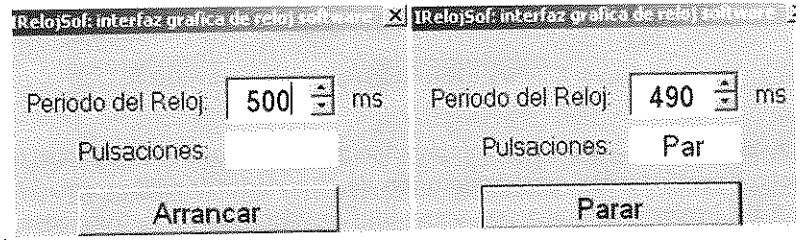

Figura 4. IGU del reloj en software

La organización de este programa se muestra en el diagrama de la figura 5, que incluye un delegado y las clases Program, IRelojSof y RelojSof, a partir de las cuales se genera el ejecutable RelojSof.exe.

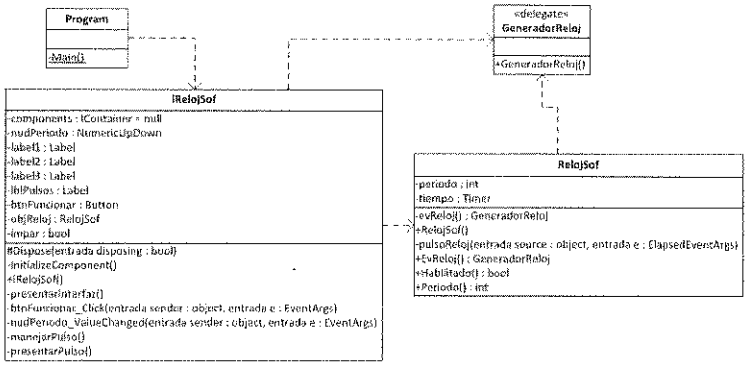

Figura 5. Diagrama de clases - diseño del reloj en software

Las clases Program y RelojSof se implementan en los archivos Program.cs y RelojSof.cs, respectivamente. La clase IRelojSof se implementa en dos archivos: IRelojSof.cs y IrelojSof.Disgner.cs.

//Programa.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace RelojSof
{
  static class Program
  {
    /// <summary>
    /// Punto de entrada principal para la aplicación.
    /// </summary>
    [STAThread]
    static void Main()
    {
      Application.EnableVisualStyles();
      Application.SetCompatibleTextRenderingDefault(false);
      Application.Run(new IRelojSof());
    }
  }
}
```

//IRelojSof.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace RelojSof
```

```
{
  public partial class IRelojSof : Form
  {
    private RelojSof objReloj; //referencia a la instancia del reloj en software
    private bool impar; //para presentar pulsaciones

    public IRelojSof()
    {
      InitializeComponent();
      objReloj = new RelojSof(); //instancia de reloj en software

      //registro para informarse del pulso del reloj
      //la temporización y manejarPulso se realizan asincronamente al hilo
      //principal del RelojSoft.-
      objReloj.EvReloj += new GeneradorReloj(this.manejarPulso);

      //visualizar el periodo vigente de del reloj
      nudPeriodo.Value = objReloj.Periodo;
      presentarInterfaz(); // alinear interfaz visual
    }

    private void presentarInterfaz()
    {
      if (objReloj.Habilitado)
      {
        btnFuncionar.Text = "Parar";
      }
      else
      {
        btnFuncionar.Text = "Arrancar";
        lblPulsos.Text = "";
      }
    }

    //arrancar o parar el reloj, en función de su estado
    private void btnFuncionar_Click(object sender, EventArgs e)
    {
      if (objReloj.Habilitado)
      {
        objReloj.Habilitado = false;
        presentarInterfaz();
      }
      else
      {
        objReloj.Periodo = (int)nudPeriodo.Value;
        objReloj.Habilitado = true;
        presentarInterfaz();
      }
    }

    //ajustar el periodo del reloj a pedido del ususario
    private void nudPeriodo_ValueChanged(object sender, EventArgs e)
    {
      if (objReloj.Habilitado)
        objReloj.Periodo = (int)nudPeriodo.Value;
    }

    void manejarPulso()
    {
      //sincroniza el metodo presentarPulso con el hilo principal de RelojSoft
      this.BeginInvoke(new GeneradorReloj(presentarPulso));
    }
```

```
    //mostrar en interfaz indicación de pulso de reloj en el hilo principal de
RelojSoft
    void presentarPulso()
    {
      if (impar)
      {
        lblPulsos.ForeColor = Color.Green;
        lblPulsos.Text = "Impar";
      }
      else
      {
        lblPulsos.ForeColor = Color.Blue;
        lblPulsos.Text = "Par";
      }
      impar = !impar;
    }
  }
}
```

//IRelojSof.Designer.cs

```
namespace RelojSof
{
  partial class IRelojSof
  {
    /// <summary>
    /// Variable del diseñador requerida.
    /// </summary>
    private System.ComponentModel.IContainer components = null;

    /// <summary>
    /// Limpiar los recursos que se estén utilizando.
    /// </summary>
    /// <param name="disposing">true si los recursos administrados se deben
eliminar; false en caso contrario, false.</param>
    protected override void Dispose(bool disposing)
    {
      if (disposing && (components != null))
      {
        components.Dispose();
      }
      base.Dispose(disposing);
    }

    #region Código generado por el Diseñador de Windows Forms

    /// <summary>
    /// Método necesario para admitir el Diseñador. No se puede modificar
    /// el contenido del método con el editor de código.
    /// </summary>
    private void InitializeComponent()
    {
      this.nudPeriodo = new System.Windows.Forms.NumericUpDown();
      this.label1 = new System.Windows.Forms.Label();
      this.label2 = new System.Windows.Forms.Label();
      this.label3 = new System.Windows.Forms.Label();
      this.lblPulsos = new System.Windows.Forms.Label();
      this.btnFuncionar = new System.Windows.Forms.Button();
      ((System.ComponentModel.ISupportInitialize)(this.nudPeriodo)).BeginInit();
      this.SuspendLayout();
      //
      // nudPeriodo
```

```
            //
            this.nudPeriodo.Font = new System.Drawing.Font("Microsoft Sans Serif", 14F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.nudPeriodo.Increment = new decimal(new int[] {
                  10,
                  0,
                  0,
                  0});
            this.nudPeriodo.Location = new System.Drawing.Point(163, 28);
            this.nudPeriodo.Maximum = new decimal(new int[] {
                  99999,
                  0,
                  0,
                  0});
            this.nudPeriodo.Minimum = new decimal(new int[] {
                  1,
                  0,
                  0,
                  0});
            this.nudPeriodo.Name = "nudPeriodo";
            this.nudPeriodo.Size = new System.Drawing.Size(75, 29);
            this.nudPeriodo.TabIndex = 0;
            this.nudPeriodo.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;
            this.nudPeriodo.Value = new decimal(new int[] {
                  1000,
                  0,
                  0,
                  0});
            this.nudPeriodo.ValueChanged += new
System.EventHandler(this.nudPeriodo_ValueChanged);
            //
            // label1
            //
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label1.Location = new System.Drawing.Point(12, 32);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(145, 23);
            this.label1.TabIndex = 1;
            this.label1.Text = "Periodo del Reloj:";
            this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
            //
            // label2
            //
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label2.Location = new System.Drawing.Point(244, 33);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(36, 23);
            this.label2.TabIndex = 2;
            this.label2.Text = "ms";
            //
            // label3
            //
            this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label3.Location = new System.Drawing.Point(26, 64);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(131, 23);
            this.label3.TabIndex = 3;
            this.label3.Text = "Pulsaciones:";
            this.label3.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
            //
```

```
      // lblPulsos
      //
      this.lblPulsos.BackColor = System.Drawing.SystemColors.ActiveCaptionText;
      this.lblPulsos.Font = new System.Drawing.Font("Microsoft Sans Serif", 14F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
      this.lblPulsos.Location = new System.Drawing.Point(163, 64);
      this.lblPulsos.Name = "lblPulsos";
      this.lblPulsos.Size = new System.Drawing.Size(75, 23);
      this.lblPulsos.TabIndex = 4;
      this.lblPulsos.Text = "par";
      this.lblPulsos.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
      //
      // btnFuncionar
      //
      this.btnFuncionar.BackColor = System.Drawing.SystemColors.Control;
      this.btnFuncionar.Font = new System.Drawing.Font("Microsoft Sans Serif", 14F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
      this.btnFuncionar.Location = new System.Drawing.Point(63, 100);
      this.btnFuncionar.Name = "btnFuncionar";
      this.btnFuncionar.Size = new System.Drawing.Size(175, 30);
      this.btnFuncionar.TabIndex = 5;
      this.btnFuncionar.Text = "Arrancar";
      this.btnFuncionar.UseVisualStyleBackColor = false;
      this.btnFuncionar.Click += new System.EventHandler(this.btnFuncionar_Click);
      //
      // IRelojSof
      //
      this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
      this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
      this.ClientSize = new System.Drawing.Size(288, 132);
      this.Controls.Add(this.btnFuncionar);
      this.Controls.Add(this.lblPulsos);
      this.Controls.Add(this.label3);
      this.Controls.Add(this.label2);
      this.Controls.Add(this.label1);
      this.Controls.Add(this.nudPeriodo);
      this.MaximizeBox = false;
      this.MinimizeBox = false;
      this.Name = "IRelojSof";
      this.Text = "IRelojSof: interfaz grafica de reloj software ";
      ((System.ComponentModel.ISupportInitialize)(this.nudPeriodo)).EndInit();
      this.ResumeLayout(false);

    }

    #endregion

    private System.Windows.Forms.NumericUpDown nudPeriodo;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.Label lblPulsos;
    private System.Windows.Forms.Button btnFuncionar;
  }
}
```

//RelojSof.cs

```
using System;

namespace RelojSof
{
  delegate void GeneradorReloj(); //delegado para procesar interrupciones del Timer
```

```
public class RelojSof
{
  //periodo del reloj en milisegundos, definido por interrupciones del Timer
  private int periodo;

  //temporizador disponible en .NET
  private System.Timers.Timer tiempo;

  private event GeneradorReloj evReloj; //evento de interrupcion del timer

  internal RelojSof()
  {
    periodo = 500; //periodo inicial
    tiempo = new System.Timers.Timer(periodo); //implementación del timer
    //vinculación de metodo que procesa interrupciones del timer
    tiempo.Elapsed += new System.Timers.ElapsedEventHandler(pulsoReloj);
  }

  private void pulsoReloj(object source, System.Timers.ElapsedEventArgs e)
  {
    if (evReloj != null)
      evReloj(); //producción de eventos para los interesados registrados
  }

  internal GeneradorReloj EvReloj //para registrar interesado en el evento
  {
    set
    {
      evReloj += value;
    }
    get
    {
      return evReloj;
    }
  }

  public bool Habilitado //acceso a estado del temporizador
  {
    get
    {
      return tiempo.Enabled;
    }
    set
    {
      tiempo.Enabled = value;
    }
  }

  public int Periodo //acceso al periodo del temporizador
  {
    get
    {
      return periodo;
    }
    set
    {
      periodo = value < 1 ? 1 : value;
      tiempo.Interval = periodo;
    }
  }
 }
}
```

Hilo computacional de proceso controlado por reloj en software.

La atención se centra en el hilo que calcula y presenta la sumatoria de un número dado, mostrando el avance de proceso del hilo a velocidad controlada por el usuario. Se emplea el reloj de periodo controlado del párrafo anterior. El entorno de programación ms vs.Net 2005, o posteriores, con C#. Sumatoria es un hilo computacional concurrente que se ejecuta dentro del proceso HPC, junto con el hilo principal de este proceso y el hilo concurrente que implementa la temporización para el reloj. El hilo principal y el hilo de temporización los brinda .NET.

La interfaz integrada de la aplicación, tanto del reloj en software como del hilo computacional, se muestra en la figura 6. En la interfaz del hilo se presenta un avance de su proceso.

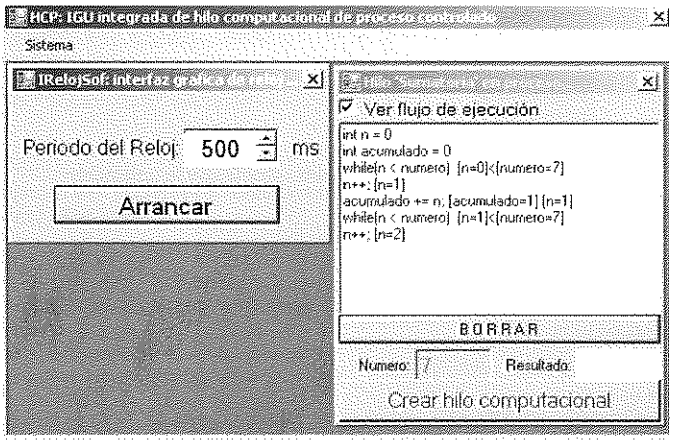

Figura 6. IGU integrada: de reloj y de hilo controlado

La estructura de la aplicación se muestra en el diagrama de clases – diseño de la figura 7. Las clases IRelojSof y RelojSof están a cargo del reloj implementado en software. Las clases ISumatoria y Sumatoria realizan el hilo computacional. La clase HCP es la interfaz, que a su vez integra las interfaces de administración del reloj en software y de gestión del hilo computacional. La clase Program es la entrada a la aplicación.

Las clases se implementan en sus respectivos archivos fuentes en C#. Los archivos que implementan el reloj y la clase Program se listaron en el párrafo anterior. El código de la clase HCP fácilmente se infiere de la interfaz de la aplicación y de la respectiva clase en el diagrama. A continuación se listan los archivos fuentes que contienen el código de las clases que están a cargo del hilo computacional "Sumatoria".

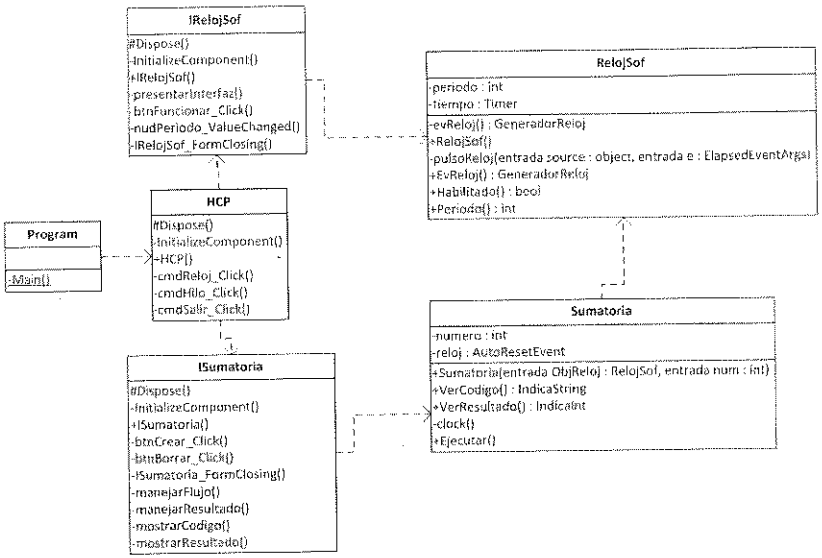

Figura 7. Diagrama de clases - diseño de la aplicación

//ISumatoria.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;

namespace HCP
{
  public partial class ISumatoria : Form
  {
    RelojSof objReloj;
    Thread hSumatoria;

    bool hiloActivo = false;
    public ISumatoria(RelojSof relo)
    {
      InitializeComponent();
      objReloj = relo;
      txtNumero.Select();
    }

    private void btnCrear_Click(object sender, EventArgs e)
    {
      //definicion de variables y objetos
      bool ver = true;
      int numero = 0;
      Sumatoria objSum;
```

```
  //captura del numero ingresado por el usuario
  try
  {
    numero = int.Parse(txtNumero.Text);
  }
  catch
  {
    ver = false;
  }

  if (ver)
  {
    //creacion y arranque del hilo Sumatoria
    btnCrear.Enabled = false;
    txtNumero.Enabled = false;
    lblResultado.Text = "";
    objSum = new Sumatoria(objReloj, numero);
    hSumatoria = new Thread(new ThreadStart(objSum.Ejecutar));
    //vinculación a eventos
    objSum.VerCodigo += new IndicaString(this.manejarFlujo);
    objSum.VerResultado += new IndicaInt(this.manejarResultado);
    hSumatoria.Start();
    hiloActivo = true;
    Text = "Hilo 'Sumatoria' en proceso...";
  }
  else
  {
    display.AppendText("Ingrese numero correcto...\n");
    display.ScrollToCaret();
  }
}

private void btnBorrar_Click(object sender, EventArgs e)
{
  display.Clear();
}

private void ISumatoria_FormClosing(object sender, FormClosingEventArgs e)
{
  if (hiloActivo)
  {
    hSumatoria.Abort();
  }
}

void manejarFlujo(string s)
{
  //invocación asincrona
  this.BeginInvoke(new IndicaString(mostrarCodigo), new Object[] {s });
}

void manejarResultado(int sum)
{
  //invocacion asincrona
  hiloActivo = false;
  this.BeginInvoke(new IndicaInt(mostrarResultado), new Object[] {sum });
}

void mostrarCodigo(string s)
{
  if (chbFlujo.Checked)
  {
```

```
        display.AppendText(s);
        display.ScrollToCaret();
      }
    }

    void mostrarResultado(int suma)
    {
      lblResultado.Text = suma.ToString();
      btnCrear.Enabled = true;
      txtNumero.Enabled = true;
      hiloActivo = false;
      Text = "ISumatoria";
      mostrarCodigo("<hilo terminado>\n");
    }
  }
}
```

//ISumatoria.designer.cs

```
namespace HCP
{
  partial class ISumatoria
  {
    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.IContainer components = null;

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
    protected override void Dispose(bool disposing)
    {
      if (disposing && (components != null))
      {
        components.Dispose();
      }
      base.Dispose(disposing);
    }

    #region Windows Form Designer generated code

    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
      this.splitContainer1 = new System.Windows.Forms.SplitContainer();
      this.display = new System.Windows.Forms.RichTextBox();
      this.chbFlujo = new System.Windows.Forms.CheckBox();
      this.btnBorrar = new System.Windows.Forms.Button();
      this.btnCrear = new System.Windows.Forms.Button();
      this.panel1 = new System.Windows.Forms.Panel();
      this.splitContainer3 = new System.Windows.Forms.SplitContainer();
      this.txtNumero = new System.Windows.Forms.TextBox();
      this.lblNumero = new System.Windows.Forms.Label();
      this.lblResultado = new System.Windows.Forms.Label();
      this.lblResultadolbl = new System.Windows.Forms.Label();
      ((System.ComponentModel.ISupportInitialize)(this.splitContainer1)).BeginInit();
```

```
        this.splitContainer1.Panel1.SuspendLayout();
        this.splitContainer1.Panel2.SuspendLayout();
        this.splitContainer1.SuspendLayout();
        this.panel1.SuspendLayout();
        ((System.ComponentModel.ISupportInitialize)(this.splitContainer3)).BeginInit();
        this.splitContainer3.Panel1.SuspendLayout();
        this.splitContainer3.Panel2.SuspendLayout();
        this.splitContainer3.SuspendLayout();
        this.SuspendLayout();
        //
        // splitContainer1
        //
        this.splitContainer1.Dock = System.Windows.Forms.DockStyle.Fill;
        this.splitContainer1.Location = new System.Drawing.Point(0, 0);
        this.splitContainer1.Name = "splitContainer1";
        this.splitContainer1.Orientation = System.Windows.Forms.Orientation.Horizontal;
        //
        // splitContainer1.Panel1
        //
        this.splitContainer1.Panel1.Controls.Add(this.display);
        this.splitContainer1.Panel1.Controls.Add(this.chbFlujo);
        this.splitContainer1.Panel1.Controls.Add(this.btnBorrar);
        //
        // splitContainer1.Panel2
        //
        this.splitContainer1.Panel2.Controls.Add(this.btnCrear);
        this.splitContainer1.Panel2.Controls.Add(this.panel1);
        this.splitContainer1.Size = new System.Drawing.Size(251, 256);
        this.splitContainer1.SplitterDistance = 195;
        this.splitContainer1.TabIndex = 0;
        //
        // display
        //
        this.display.Dock = System.Windows.Forms.DockStyle.Fill;
        this.display.Location = new System.Drawing.Point(0, 21);
        this.display.Name = "display";
        this.display.Size = new System.Drawing.Size(251, 151);
        this.display.TabIndex = 5;
        this.display.Text = "";
        //
        // chbFlujo
        //
        this.chbFlujo.AutoSize = true;
        this.chbFlujo.Checked = true;
        this.chbFlujo.CheckState = System.Windows.Forms.CheckState.Checked;
        this.chbFlujo.Dock = System.Windows.Forms.DockStyle.Top;
            this.chbFlujo.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.chbFlujo.Location = new System.Drawing.Point(0, 0);
        this.chbFlujo.Margin = new System.Windows.Forms.Padding(2);
        this.chbFlujo.Name = "chbFlujo";
        this.chbFlujo.Size = new System.Drawing.Size(251, 21);
        this.chbFlujo.TabIndex = 4;
        this.chbFlujo.Text = "Ver flujo de ejecución";
        this.chbFlujo.UseVisualStyleBackColor = true;
        //
        // btnBorrar
        //
        this.btnBorrar.AutoSize = true;
        this.btnBorrar.Dock = System.Windows.Forms.DockStyle.Bottom;
        this.btnBorrar.Location = new System.Drawing.Point(0, 172);
        this.btnBorrar.Margin = new System.Windows.Forms.Padding(2);
        this.btnBorrar.Name = "btnBorrar";
```

```
        this.btnBorrar.Size = new System.Drawing.Size(251, 23);
        this.btnBorrar.TabIndex = 3;
        this.btnBorrar.Text = "B O R R A R";
        this.btnBorrar.UseVisualStyleBackColor = true;
        this.btnBorrar.Click += new System.EventHandler(this.btnBorrar_Click);
        //
        // btnCrear
        //
         this.btnCrear.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(255)))),
((int)(((byte)(192)))), ((int)(((byte)(192)))));
        this.btnCrear.Dock = System.Windows.Forms.DockStyle.Fill;
            this.btnCrear.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.btnCrear.Location = new System.Drawing.Point(0, 24);
        this.btnCrear.Margin = new System.Windows.Forms.Padding(2);
        this.btnCrear.Name = "btnCrear";
        this.btnCrear.Size = new System.Drawing.Size(251, 33);
        this.btnCrear.TabIndex = 5;
        this.btnCrear.Text = "Crear hilo computacional";
        this.btnCrear.UseVisualStyleBackColor = false;
        this.btnCrear.Click += new System.EventHandler(this.btnCrear_Click);
        //
        // panel1
        //
        this.panel1.Controls.Add(this.splitContainer3);
        this.panel1.Dock = System.Windows.Forms.DockStyle.Top;
        this.panel1.Location = new System.Drawing.Point(0, 0);
        this.panel1.Name = "panel1";
        this.panel1.Size = new System.Drawing.Size(251, 24);
        this.panel1.TabIndex = 0;
        //
        // splitContainer3
        //
        this.splitContainer3.Dock = System.Windows.Forms.DockStyle.Fill;
        this.splitContainer3.Location = new System.Drawing.Point(0, 0);
        this.splitContainer3.Name = "splitContainer3";
        //
        // splitContainer3.Panel1
        //
        this.splitContainer3.Panel1.Controls.Add(this.txtNumero);
        this.splitContainer3.Panel1.Controls.Add(this.lblNumero);
        //
        // splitContainer3.Panel2
        //
        this.splitContainer3.Panel2.Controls.Add(this.lblResultado);
        this.splitContainer3.Panel2.Controls.Add(this.lblResultadolbl);
        this.splitContainer3.Size = new System.Drawing.Size(251, 24);
        this.splitContainer3.SplitterDistance = 119;
        this.splitContainer3.TabIndex = 0;
        //
        // txtNumero
        //
        this.txtNumero.Dock = System.Windows.Forms.DockStyle.Fill;
            this.txtNumero.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.txtNumero.Location = new System.Drawing.Point(59, 0);
        this.txtNumero.Margin = new System.Windows.Forms.Padding(2);
        this.txtNumero.Name = "txtNumero";
        this.txtNumero.Size = new System.Drawing.Size(60, 26);
        this.txtNumero.TabIndex = 3;
        //
        // lblNumero
        //
```

```
      this.lblNumero.Dock = System.Windows.Forms.DockStyle.Left;
          this.lblNumero.Font = new System.Drawing.Font("Microsoft Sans Serif", 8F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
      this.lblNumero.Location = new System.Drawing.Point(0, 0);
      this.lblNumero.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
      this.lblNumero.Name = "lblNumero";
      this.lblNumero.Size = new System.Drawing.Size(59, 24);
      this.lblNumero.TabIndex = 1;
      this.lblNumero.Text = "Numero:";
      this.lblNumero.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
      //
      // lblResultado
      //
      this.lblResultado.BackColor = System.Drawing.SystemColors.ControlLightLight;
      this.lblResultado.Dock = System.Windows.Forms.DockStyle.Fill;
        this.lblResultado.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
      this.lblResultado.Location = new System.Drawing.Point(62, 0);
      this.lblResultado.Margin = new System.Windows.Forms.Padding(2);
      this.lblResultado.Name = "lblResultado";
      this.lblResultado.Size = new System.Drawing.Size(66, 24);
      this.lblResultado.TabIndex = 4;
      this.lblResultado.Text = "?";
      this.lblResultado.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
      //
      // lblResultadolbl
      //
      this.lblResultadolbl.Dock = System.Windows.Forms.DockStyle.Left;
       this.lblResultadolbl.Font = new System.Drawing.Font("Microsoft Sans Serif", 8F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
      this.lblResultadolbl.Location = new System.Drawing.Point(0, 0);
      this.lblResultadolbl.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
      this.lblResultadolbl.Name = "lblResultadolbl";
      this.lblResultadolbl.Size = new System.Drawing.Size(62, 24);
      this.lblResultadolbl.TabIndex = 2;
      this.lblResultadolbl.Text = "Resultado:";
      this.lblResultadolbl.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
      //
      // ISumatoria
      //
      this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
      this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
      this.ClientSize = new System.Drawing.Size(251, 256);
      this.Controls.Add(this.splitContainer1);
      this.MaximizeBox = false;
      this.MinimizeBox = false;
      this.Name = "ISumatoria";
      this.Text = "ISumatoria";
                  this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.ISumatoria_FormClosing);
      this.splitContainer1.Panel1.ResumeLayout(false);
      this.splitContainer1.Panel1.PerformLayout();
      this.splitContainer1.Panel2.ResumeLayout(false);
      ((System.ComponentModel.ISupportInitialize)(this.splitContainer1)).EndInit();
      this.splitContainer1.ResumeLayout(false);
      this.panel1.ResumeLayout(false);
      this.splitContainer3.Panel1.ResumeLayout(false);
      this.splitContainer3.Panel1.PerformLayout();
      this.splitContainer3.Panel2.ResumeLayout(false);
      ((System.ComponentModel.ISupportInitialize)(this.splitContainer3)).EndInit();
      this.splitContainer3.ResumeLayout(false);
      this.ResumeLayout(false);
    }
```

```
    #endregion

    private System.Windows.Forms.SplitContainer splitContainer1;
    private System.Windows.Forms.Panel panel1;
    private System.Windows.Forms.SplitContainer splitContainer3;
    private System.Windows.Forms.Button btnCrear;
    private System.Windows.Forms.Label lblNumero;
    private System.Windows.Forms.TextBox txtNumero;
    private System.Windows.Forms.Label lblResultadolbl;
    private System.Windows.Forms.Label lblResultado;
    private System.Windows.Forms.Button btnBorrar;
    private System.Windows.Forms.CheckBox chbFlujo;
    private System.Windows.Forms.RichTextBox display;
  }
}
```

//Sumatoria.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;

namespace HCP
{
  public delegate void IndicaString(string s);
  public delegate void IndicaInt(int n);

  public class Sumatoria
  {
    int numero;
    AutoResetEvent reloj;//reloj de ejecucion

    public Sumatoria(RelojSof ObjReloj , int num)
    {
      reloj = new AutoResetEvent(false);
      ObjReloj.EvReloj += new GeneradorReloj(clock);
      numero = num;
    }

    public event IndicaString VerCodigo;
    public event IndicaInt VerResultado;

    private void clock()
    {
      reloj.Set();
    }

    public void Ejecutar()
    {
      reloj.WaitOne();
      int n = 0;
      VerCodigo("int n = 0\n");
      reloj.WaitOne();

      int acumulado = 0;
      VerCodigo("int acumulado = 0\n");
      reloj.WaitOne();

      while (n < numero)
      {
```

```
                    VerCodigo(string.Format("while(n < numero)
[n={0}]<[numero={1}]\n",n,numero));
        reloj.WaitOne();

        n++;
        VerCodigo(string.Format("n++; [n={0}]\n",n));
        reloj.WaitOne();

        acumulado += n;
                    VerCodigo(string.Format("acumulado += n; [acumulado={0}]
[n={1}]\n",acumulado,n));
        reloj.WaitOne();
      }
      VerCodigo(string.Format("while(n < numero) [n={0}]<[numero={1}]\n", n, numero));
      reloj.WaitOne();

          VerCodigo(string.Format("VerResultado(acumulado); [acumulado={0}]\n",
acumulado));
      VerResultado(acumulado);
    }
  }
}
```

CONCLUSIONES

- La velocidad de proceso de los procesadores es diferente a la velocidad de proceso de los hilos computacionales.
- El control, con reloj de periodo definido por el usuario, de la velocidad de proceso de los hilos computacionales permite usarlos en su enseñanza y aprendizaje significativos.

REFERENCIAS BIBLIOGRÁFICAS

[1] AGUILAR, V. F. S. Gestión de sub procesos en la enseñanza-aprendizaje de sub procesos y procesos de los sistemas operativos. Perú: Universidad Nacional Mayor de San Marcos. 2011. Tesis defendida en Marzo-2011.

[2] AUSUBEL, D.P.; NOVAK, J.; HANESIAN H. Psicología educativa. Un punto de vista cognoscitivo. Trillas, México D.F., 1976.

[3] BOOCH G.; RUMBAUGH J.; JACOBSON I. El lenguaje unificado de modelado. 2ª ed. Madrid (España). 2006.

[4] DEITEL H.M.; et al. C# how to program. Prentice Hall, New Jersey (USA), 2002.

[5] HENNESSY, J. L.; PATTERSON, D. A. Computer Architecture: A Quantitative Approach. 4ª ed. Morgan Kaufman, CA, USA. 2007

[6] MANO M.M. Computer system architecture. 3rd ed. Prentice Hall, New Jersey (USA), 1993.

[7] MENESES B.G. NTIC, interacción y aprendizaje en la Universidad. [en línea]. [España]: Universidad Rovira I Virgili. 2008. [consulta: -04-2010]. Documento: El Proceso De Enseñanza. Tesis: defendica el 26-06-2007. Formato pdf. Disponible en web: http://www.tdx.cat/TDX-1207107-161635. DL/ISBN: T.2183-2007/978-84-691-0359-3.

[8] NUTT, G. Sistemas Operativos. 3a ed. Pearson: Addison Wesley, Madrid (España), 2004.