**ÍNDICE GENERAL**

[**Generalidades 4**](#_Toc406955342)

[**Introduccion 4**](#_Toc406955343)

[**Antecedentes 4**](#_Toc406955344)

[**Antecedentes organizacionales 4**](#_Toc406955345)

[**Antecedentes tecnológicos 7**](#_Toc406955346)

[**Problema 8**](#_Toc406955347)

[**Situación Problemática 8**](#_Toc406955348)

[**Formulación del Problema 9**](#_Toc406955349)

[**Objetivo 9**](#_Toc406955350)

[**Objetivo General 9**](#_Toc406955351)

[**Objetivos Específicos y Acciones 10**](#_Toc406955352)

[**Alcances 11**](#_Toc406955353)

[**Limitaciones 12**](#_Toc406955354)

[**Justificación 13**](#_Toc406955355)

[**Justificación Social 13**](#_Toc406955356)

[**Justificación Técnica 13**](#_Toc406955357)

[**Cronograma 14**](#_Toc406955358)

[**Bibliografía 14**](#_Toc406955359)

[**Capítulo 1. Marco Teórico 15**](#_Toc406955360)

[**1.1. Arquitectura de Computadoras 15**](#_Toc406955361)

[**1.1.1. Ciclo de instrucción 15**](#_Toc406955362)

[**1.1.2. Arquitectura del Procesador 16**](#_Toc406955363)

[**1.1.3. Interacción entre Procesador y Memoria Principal 18**](#_Toc406955364)

[**1.2. Software educativo 19**](#_Toc406955365)

[**1.2.1. Definición 19**](#_Toc406955366)

[**1.2.2. Tipos de software didáctico 20**](#_Toc406955367)

[**1.3. Metodología de desarrollo de Software 23**](#_Toc406955368)

[**1.3.1. Generalidades 23**](#_Toc406955369)

[**1.3.2. Selección de metodología. 24**](#_Toc406955370)

[**1.3.3. Prototipado 25**](#_Toc406955371)

[**1.3.4. Etapas del prototipado 27**](#_Toc406955372)

[**1.4. Tecnologías de desarrollo 27**](#_Toc406955373)

[**1.4.1 Visual Studio 27**](#_Toc406955374)

[**1.4.2. C# 28**](#_Toc406955375)

[**1.4.3. Librerías utilizadas en el proyecto: 29**](#_Toc406955376)

[**Capítulo 2. Marco Práctico 32**](#_Toc406955377)

[**2.1. Diseño de componentes de interfaz que permitan visualizar arquitectura del CPU. 32**](#_Toc406955378)

[**2.1.1. Prototipo 1 32**](#_Toc406955379)

[**2.1.2. Prototipo 2 33**](#_Toc406955380)

[**2.2. Desarrollo del módulo interactivo para trabajar con lenguaje binario. 35**](#_Toc406955381)

[**2.2.1. Prototipo 3 35**](#_Toc406955382)

[**2.2.2. Prototipo 4 38**](#_Toc406955383)

[**2.2.3. Prototipo 5 42**](#_Toc406955384)

[**2.2.4. Prototipo 6 44**](#_Toc406955385)

[**2.2.5. Prototipo 7 50**](#_Toc406955386)

[**2.2.6. Prototipo 8 52**](#_Toc406955387)

[**2.2.7. Prototipo 9 54**](#_Toc406955388)

[**Anexos 57**](#_Toc406955389)

[**Anexo 1 57**](#_Toc406955390)

[**Anexo 2 59**](#_Toc406955391)

[**Anexo 3 60**](#_Toc406955392)

# Generalidades

## Introduccion

En la actualidad el proceso de enseñanza aprendizaje ha ido evolucionando a medida que la tecnología ha ingresado a este campo y le ha proporcionado herramientas para que la educación sea más eficiente y logre mejores resultados.

Hoy en día las herramientas interactivas orientadas a apoyar el proceso de enseñanza-aprendizaje han tomado mucha fuerza, esto debido a la posibilidad que proveen estas herramientas de aplicar los conocimientos teóricos avanzados en clases en ejercicios prácticos, facilitando la visualización del problema.

Una ventaja que otorgan las herramientas interactivas en el área de software educativo es que estas herramientas suelen ser fácilmente adaptables a las necesidades del usuario final, debido a que pueden tomar funcionalidades y características de otros tipos de software educativos como ser simuladores o sistemas con bases de datos.

A lo largo de los años, en la materia de Arquitectura de Computadoras perteneciente a la rama de la informática, el estudio del Ciclo de Instrucción del computador ha traído consigo serios problemas en el proceso de enseñanza aprendizaje y se han probado varios métodos de enseñanza y optado por distintas herramientas para llevarlo a cabo, pero los problemas persisten.

## Antecedentes

### Antecedentes organizacionales

En la actualidad, muchas universidades cuentan con la materia de Arquitectura de Computadoras dentro del pensum establecido para la carrera de Ingeniería de Sistemas. La Universidad Católica Boliviana “San Pablo” no es la excepción,

Dentro del contenido de esta materia, uno de los temas en los que se hace mayor énfasis es el Ciclo de Instrucción del Computador.

El ***objetivo principal*** de este tema es que el estudiante comprenda la secuencia exacta de pasos que debe realizar el CPU, dependiendo de su arquitectura y formato de instrucción (número de direcciones, CO, tipo de direccionamiento), y cómo este influye en la velocidad de ejecución de un programa.

Un ***ciclo de instrucción*** es el período que tarda la unidad central de proceso (CPU) en ejecutar una instrucción en lenguaje máquina. Este comprende una secuencia de acciones que se deben llevar a cabo para ejecutar cada instrucción en un programa.

Para que un CPU realice una tarea, primero debe buscar cada instrucción en la memoria principal y luego ejecutarla. A estos pasos se los conoce respectivamente como ciclo de captación y ciclo de ejecución, y el último depende de varios factores descritos a continuación.

En el ciclo de captación se realiza la búsqueda de la instrucción en la memoria principal y su respectiva decodificación, posteriormente en el ciclo de ejecución se ejecuta la instrucción y finalmente se almacena el resultado.

El número de pasos en el ciclo de captación generalmente es similar, sin importar la arquitectura del CPU, ni la instrucción que se haya dado. En cambio, el número de pasos que hay en el ciclo de ejecución sí depende de la arquitectura del CPU y del formato de instrucción, es decir, del código de operación (CO), del número de direcciones de la instrucción, del tipo de de direccionamiento y de los registros que contenga (AR y DR o banco de registros).

Los ejercicios de ciclo de instrucción pueden ser resueltos, ya sea en binario o en hexadecimal, así como también en lenguaje ensamblador.

En el desarrollo de la enseñanza del tema, se dedica una cierta cantidad de clases para explicar este proceso detalladamente al alumno, primeramente mostrándole la interacción de los componentes del CPU durante el ciclo de instrucción en binario. Posteriormente, para facilitar y acortar la tarea, cuando ya se ha captado el concepto en binario, se procede a utilizar la notación hexadecimal y finalmente se empieza a aplicar el lenguaje ensamblador en todo el proceso.

En binario se enseña la distribución de los bits en los registros (bits utilizados para CO y para direccionamiento), y dependiendo de estos se calcula el rango de valores que se podrán utilizar y el espacio de memoria que podrá accederse. Esto da una idea clara respecto a que los recursos son limitados.

En hexadecimal empieza a mostrarse a más profundidad la interacción entre CPU y memoria principal. En esta parte se ve como se utilizan los recursos del procesador destinados a la interacción con memoria, así como también con los dispositivos de E/S, como por ejemplo los registros MAR (registro de direcciones de memoria) y MBR (registro de datos de memoria), buses internos y externos del CPU (buses con memoria y con los dispositivos de E/S).

En ensamblador se aborda el objetivo principal del tema: que el estudiante comprenda la secuencia exacta de pasos que debe realizar el CPU, dependiendo de su arquitectura y formato de instrucción (número de direcciones, CO, tipo de direccionamiento), y como este influye en la velocidad de ejecución de un programa.

En el desarrollo de las clases para entender el proceso que implica el ciclo de instrucción del computador se elaboran ejercicios que ponen en práctica los conceptos mencionados anteriormente, lo cual toma bastante tiempo dependiendo de la complejidad de ejercicio (aplicar la lógica de funcionamiento, además de la cantidad de instrucciones requeridas en el proceso a seguir) y la cantidad de diagramas a realizar. Por ejemplo, en el inicio del tema, para realizar el primer ejercicio en binario, la docente necesita dedicarle dos clases enteras buscando que los alumnos entiendan el proceso, lo cual implica que se gasta para ello un tercio del tiempo dedicado a esta parte del tema.

Al desarrollar ejercicios se emplea tiempo considerable, alrededor de 5 minutos por cada etapa, en la diagramación de las estructuras necesarias: registros, buses, posiciones de memoria, etc.

Existen otros factores que llevan a que al estudiante le tome más tiempo resolver un ejercicio, como haber cometido errores en el desarrollo del problema, como ser el confundir registros, errores de transcripción de un diagrama al siguiente o en el manejo de direcciones de memoria.

Un dato importante a considerar es que a las dos semanas de iniciado el tema, la docente toma una prueba periódica, en la cual todos los semestres aproximadamente el 50% de los estudiantes reprueban.

Cuando la docente asigna ejercicios para que el estudiante los resuelva en casa, el estudiante los resuelve por su cuenta, por lo que si desea cerciorarse de que realizó un trabajo correcto solo puede revisarlo el mismo hasta la siguiente clase.

En algunas ocasiones la resolución de ejercicios se torna algo mecánica, por lo que cuando se cambia la arquitectura del CPU o el formato de instrucción en los ejercicios en clase los estudiantes no saben cómo resolver la situación planteada.

### Antecedentes tecnológicos

Actualmente existen herramientas de ayuda para la comprensión del ciclo de instrucción del procesador, como por ejemplo:

**Microprocessor Tutorial:** es un tutorial en línea en el cual se muestra paso a paso el desarrollo del ciclo de instrucción, utilizando lenguaje ensamblador. Permite observar, de forma general, los pasos que sigue el procesador para ejecutar una instrucción, con ejemplos ya definidos. El tutorial a su vez hace una explicación tanto de este tema, como de otros relacionados, como por ejemplo la estructura del CPU (ALU, unidad de control, registros, bus del sistema), pipelining y arquitecturas de computadora. También cuenta con pequeñas pruebas al finalizar cada tema para asegurar la comprensión de cada uno.

**Pep/7 Simulator:** es un applet en Java que simula el ciclo de instrucción, manejando lógica de instrucciones en ensamblador. Este programa es utilizado para explicar el funcionamiento del procesador, contando con siete registros y cuatro bits de estado (flags: negative, zero, overflow y carry). Entre sus opciones permite ver los registros en binario, decimal y hexadecimal. Se enfoca principalmente en el manejo de registros del procesador y de la memoria principal. Así mismo, cabe recalcar que cuenta con un tutorial en línea disponible para los usuarios.

**Dan! 71 CPU Simulator:** este simulador muestra lo que sucede dentro una CPU durante el ciclo de instrucción. Permite crear, guardar y cargar programas que puede ejecutar. Maneja los datos en lenguaje maquina y está diseñado para ser manejado por usuarios que conocen a profundidad el tema. Muestra el contenido de todos sus registros en binario y permite ejecutar un programa paso por paso, sin mostrar el procedimiento a seguir en cada uno de ellos, o mostrar directamente el resultado. Así mismo, cabe recalcar que el simulador cuenta con un manual en línea disponible para que el usuario aprenda a utilizarlo.

El uso de simuladores aplicados al proceso de enseñanza-aprendizaje del ciclo de instrucción trae consigo una gran cantidad de ventajas, especialmente visuales, pero pone a los estudiantes en un estado pasivo al realizar todo el trabajo por ellos.

## Problema

### Situación Problemática

* Al realizar los esquemas de los registros de procesador y de memoria principal de manera entendible y ordenada se incrementa el tiempo para resolución del problema en sí, así mismo desvía la atención del estudiante, que en vez de enfocarse en la dependencia entre cambios arquitectónicos del procesador y la velocidad de ejecución de un programa, se dedica más a un trabajo mecánico de dibujo de componentes del CPU, de memoria y de sus formas de interconexión.
* Al realizar el estudiante los ejercicios de ciclo de instrucción realizando varios diagramas por separado no puede ver claramente lo que está pasando en el proceso de interacción entre procesador y memoria principal.
* La elaboración manual del proceso conlleva a que el estudiante cometa errores en la transcripción del contenido de los registros entre un diagrama y otro.
* Durante la elaboración de un ejercicio si se comete un error, el resto del flujo de datos e instrucciones es incorrecto y uno se da cuenta de eso solamente en la siguiente clase cuando se hace la revisión del mismo con el docente, lo cual es inadecuado para el proceso de auto-aprendizaje.
* Los simuladores existentes relacionadas al tema están dedicados a una sola arquitectura del CPU y a la vez no son fáciles de utilizar, lo cual exige tiempo extra para su aprendizaje, al mismo tiempo realizan todo el trabajo por el estudiante y disminuye el aprovechamiento deseado. Tal es el caso del Pep/7 Simulator y del Dan! 71 CPU Simulator.
* El Microprocessor Tutorial, es muy simple, y a pesar de ser un tutorial didáctico maneja datos pre-definidos para la explicación. A su vez, no se muestra de manera clara la instrucción que se está ejecutando en el procesador ya que utiliza una notación abreviada para ello, en lugar de utilizar el CO.

### Formulación del Problema

La elaboración manual de los diagramas y las limitaciones conceptuales y técnicas de las herramientas aplicadas al proceso de enseñanza-aprendizaje del ciclo de instrucción del procesador, dificultan1 el proceso de autoaprendizaje2 respecto a la dependencia entre las prestaciones del procesador y su diseño arquitectónico.

## Objetivo

### Objetivo General

Desarrollar una herramienta interactiva para el apoyo al proceso de enseñanza-aprendizaje del ciclo de instrucción del procesador.

1 Hacer difícil una cosa, introduciendo obstáculos o inconvenientes que antes no tenía.

2 es la forma de [aprender](http://definicion.de/aprendizaje/) por uno mismo. Se [trata](http://definicion.de/autoaprendizaje/) de un proceso de adquisición de conocimientos, habilidades, valores y actitudes, que la [persona](http://definicion.de/persona)  realiza por su cuenta ya sea mediante el estudio o la [experiencia](http://definicion.de/autoaprendizaje/).

### Objetivos Específicos y Acciones

**Tabla 1. Objetivos específicos y acciones.**

|  |  |
| --- | --- |
| **Objetivos Específicos** | **Acciones** |
| Diseñar componentes de interfaz que permitan visualizar arquitectura del CPU. | Elegir metodología de desarrollo de software. |
| Elegir arquitectura de desarrollo. |
| Elegir lenguaje y herramientas de programación. |
| Diseñar la estructura de la arquitectura del CPU basada en registros específico. |
| Diseñar la estructura de la arquitectura del CPU basada en registros de uso general. |
| Diseñar estructura de memoria principal y de buses externos del CPU. |
| Desarrollar módulo interactivo para trabajar con lenguaje binario. | Definir datos de entrada. |
| Implementar componente para la definición de formato de instrucción y códigos de operación. |
| Implementar componente para preparación de la RAM para el ejercicio por parte del estudiante. |
| Desarrollar componente de interacción con el estudiante, para desplazar el contenido de registros. |
| Desarrollar componente interactivo de verificación de errores. |
| Desarrollar módulo interactivo para trabajar con lenguaje hexadecimal. | Definir datos de entrada. |
| Implementar selección de arquitectura del CPU basada en registros. |
| Implementar componente para la selección de formato de instrucción y códigos de operación. |
| Implementar componente para preparación de la RAM para el ejercicio por parte del estudiante. |
| **Objetivos Específicos** | **Acciones** |
| Desarrollar módulo interactivo para trabajar con lenguaje hexadecimal. | Desarrollar componente de interacción con el estudiante, para el desplazamiento de contenido de registros. |
| Desarrollar componente interactivo de verificación de errores. |
| Desarrollar módulo interactivo para trabajar con lenguaje ensamblador. | Definir datos de entrada. |
| Implementar selección de arquitectura del CPU basada en registros. |
| Implementar componente para la selección de formato de instrucción y códigos de operación. |
| Implementar componente para preparación de la RAM para el ejercicio por parte del estudiante. |
| Desarrollar componente de interacción con el estudiante, para el desplazamiento de contenido de registros. |
| Desarrollar componente interactivo de verificación de errores. |

Fuente: Elaboración propia 2014.

## Alcances

**Alcances del proyecto:**

* La herramienta será un sistema de escritorio.
* La herramienta se desarrollará para el sistema operativo Windows.
* La realización de las pruebas de la herramienta se llevará a cabo con colaboración de los estudiantes de la UCB que cursaron la materia de Arquitectura de Computadoras como máximo hace dos años y con los que cursaran la misma en el semestre I-2015.
* La herramienta no será diseñada para ningún procesador real, ya que se trabajará solamente sobre los componentes básicos de una CPU para la comprensión del ciclo de instrucción.

**Alcances del producto:**

* El sistema trabajará con direccionamiento en lenguaje binario, hexadecimal y ensamblador.
* Será capaz de manejar arquitectura basada en registros específicos y en registros de uso general.
* Al realizar un ejercicio la herramienta controlará si los datos introducidos y secuencia de micro-operaciones son correctos, generando advertencias de error para hacer posible la detección y corrección de errores por parte del estudiante.
* El estudiante será capaz de definir el estado inicial del CPU y de la memoria principal.
* Visualizar la interacción entre CPU y Memoria Principal.
* Generar un registro de seguimiento de acciones realizadas satisfactoriamente por el estudiante (*log* de operaciones).

## Limitaciones

* La resolución de ejercicios de ciclo de instrucción en lenguaje binario solo utilizaran formato de instrucción con una dirección en una arquitectura del CPU basada en registros específicos.
* La resolución de ejercicios ciclo de instrucción en lenguaje hexadecimal y ensamblador utilizaran formato de instrucción con dos direcciones como máximo en una arquitectura del CPU basada en registros específicos y en registros de uso general.
* La longitud máxima de una instrucción en hexadecimal será de 64 bits y en binario será de 16 bits.
* La longitud del código de operación y de las direcciones será medida en bits y tendrán un valor que sea múltiplo de 4, esto debido a que de esta forma se podrá comparar un ejercicio en binario con uno en hexadecimal.

## Justificación

### Justificación Social

Al utilizar esta herramienta el estudiante no perderá más tiempo al realizar los esquemas y en la transcripción de sus datos lo cual le permitirá concentrarse directamente en la resolución de los ejercicios, al mismo tiempo el estudiante podrá visualizar claramente lo que pasa en el proceso de interacción entre CPU y memoria principal.

La lógica de detección de errores permitirá al estudiante estar consciente de cuando está cometiendo un error o cuando está haciendo un ejercicio de la manera correcta, enriqueciendo el proceso de enseñanza-aprendizaje. Sumado al punto anterior se verá reducido el tiempo de resolución de los ejercicios con lo que el estudiante podrá enfocarse en el objetivo principal del tema.

Con la aplicación de la herramienta en el avance en clases de la materia de Arquitectura de Computadoras, se espera que el estudiante haga un seguimiento con esta herramienta de enseñanza, facilitándose esta labor al encontrarse desarrollado en español y manejando diversas arquitecturas, además de permitir al estudiante no solo ver ejemplos definidos con animaciones a una velocidad adecuada, sino también resolver sus propios ejercicios con verificación en caso de que haya cometido algún error.

### Justificación Técnica

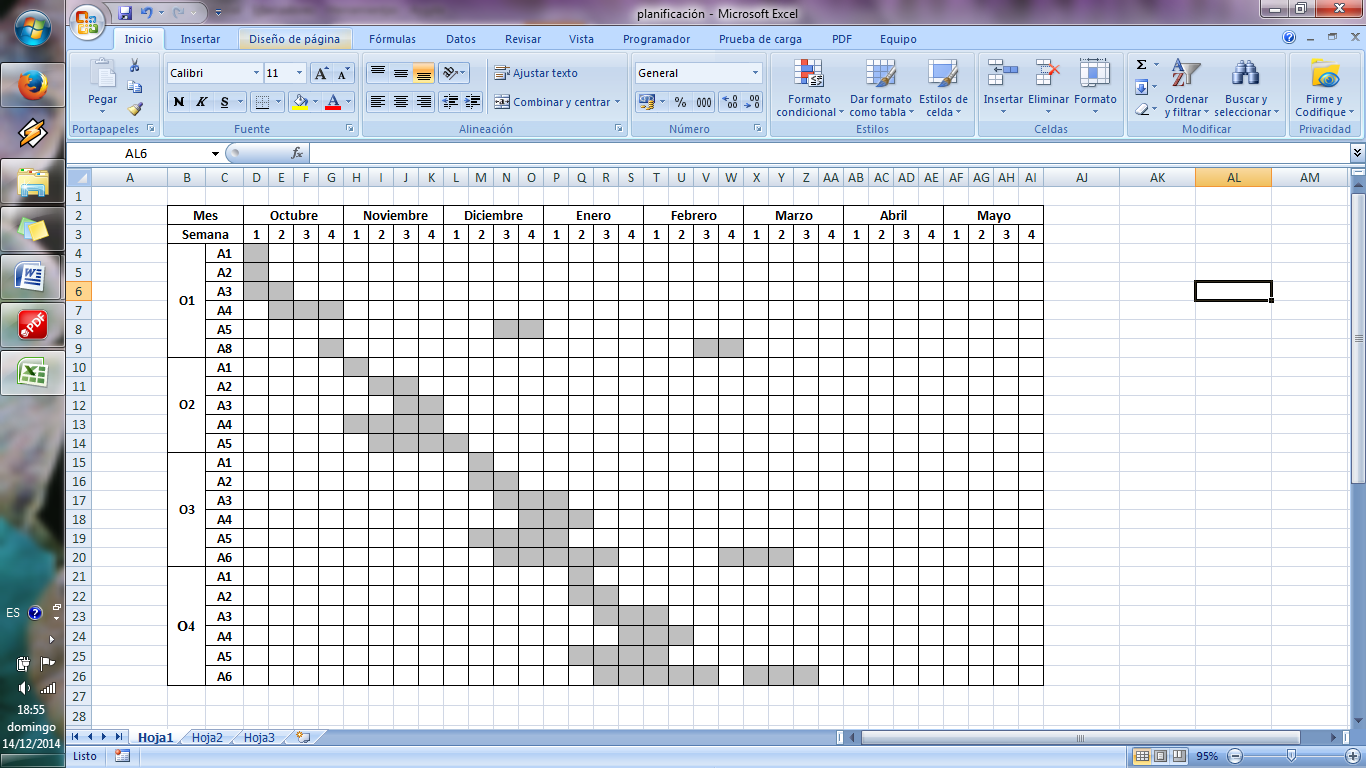
Uno de los principales objetivos de éste proyecto es que el sistema sea intuitivo, mostrando una interfaz de usuario amigable y fácil de entender.

Será una herramienta que le permitirá al estudiante elegir su propio formato de instrucción, así como elegir entre una arquitectura basada en registros específicos y en registros de uso general, por lo que podrá desarrollar sus propios ejercicios, al mismo tiempo de asegurarse que el proceso que está siguiendo es el correcto.

El estudiante no solo definirá la estructura del procesador, sino que también definirá el estado inicial de la memoria principal donde estará guardado el programa que desea ejecutarse.

## Cronograma

**Figura 1. Cronograma.**



Fuente: Elaboración propia 2014.

## Bibliografía

http://www.eastaughs.fsnet.co.uk/cpu/execution-index.html

http://www.rsu.edu/faculty/pmacpherson/programs/pep7.html

http://www.danyey.co.uk/dan71h.php

## Capítulo 1. Marco Teórico

## 1.1. Arquitectura de Computadoras

### 1.1.1. Ciclo de instrucción

El **ciclo de instrucción** es el periodo de tiempo en el que el procesador lleva a cabo una instrucción. Este tiempo varía de acuerdo de acuerdo a la arquitectura utilizada en el procesador.

Existen diversas arquitecturas que fueron utilizadas para diseñar procesadores, y cada arquitectura tiene definidos diferentes registros en su interior, también diferentes formatos de registros y de instrucciones, al igual que diferentes códigos de operación, por lo cual el tiempo en la ejecución de un mismo proceso en 2 procesadores diferentes puede variar.

Un ciclo de instrucción se divide en dos partes marcadas, el ciclo de captación o de búsqueda y el ciclo de ejecución, y a pesar de que en una misma arquitectura el ciclo de captación toma siempre l mismo tiempo, en el de ejecución es variable.

El **ciclo de captación** consiste en el intervalo en el cual el procesador toma la dirección almacenada en el registro puntero, va a la dirección de memoria almacenada en el puntero y posteriormente captura la instrucción almacenada en esta dirección de memoria llevándola a otro registro para su posterior ejecución.

El **ciclo de ejecución** consiste en la ejecución de la instrucción almacenada en uno de sus registros (IR), el cual está designado específicamente para almacenar instrucciones, y se ejecuta de la siguiente manera: Son separados de este registro el comando de instrucción y la dirección de memoria a la que deberá accederse posteriormente, estos datos se extraerán de acuerdo al formato de instrucción que utilice la arquitectura de cada procesador, y se ejecutará el comando de instrucción recuperado.

### 1.1.2. Arquitectura del Procesador

La **arquitectura del procesador** consiste en la estructura que toma el CPU, tamaño de registros (bits), formato de instrucción y Códigos de Operación, y debido a todos estos factores la velocidad de ejecución de una instrucción es variable.

Cuando se habla acerca de la **estructura del CPU** nos referimos a los recursos con los que el CPU cuenta y su organización para desempeñar su función. Entre las estructuras de que se utilizaran dentro del proyecto, basados en lo avanzado en la materia de Arquitectura de Computadoras en la Universidad Católica con Mgr. Galaburda como docente se encuentran las siguientes:

* En la primera estructura utilizada, la de arquitectura basada en registros específicos, el CPU contiene los registros PC, IR, AC, DR, MAR, MBR, I/O AR, I/O BR, al igual que contiene a la ALU y la Unidad de Control, además de contar con buses internos y externos, los cuales constan a su vez de, un bus de control, bus de datos y bus de direcciones.
* En la segunda estructura, la de arquitectura basada en registros de uso general, el CPU utiliza un Banco de Registros en lugar de los registros AC y DR para almacenar datos, es decir, registros enumerados desde R1 hasta RN.

En ambos casos para interactuar con la memoria, el procesador utiliza los registros MAR y MBR, que se conectan con los buses internos del CPU, y estos a su vez con los buses externos para finalmente conectarse con Memoria.

Cuando el procesador debe interactuar con dispositivos de E/S utiliza exactamente la misma lógica con la diferencia que utiliza los registros I/O AR e I/O BR en lugar de los registros MAR y MBR respectivamente.

El **formato de instrucción** se refiere a la estructura en la cual se dividirá el registro de instrucciones la su posterior ejecución, es decir, la cantidad de bits asignados para el código de operación, la cantidad de direcciones de memoria a las que se accederá, el modo de direccionamiento de asignado para cada dirección en la instrucción y la cantidad de bits asignados para cada dirección.

Los **modos de direccionamiento** es la forma en la cual el CPU recopila los datos guardados en RAM y dispositivos de E/S, los modos de direccionamiento utilizados serán son los siguientes:

* El **directo**, en el cual el procesador accede a la dirección de memoria que se encuentra dentro de su registro, obtiene el dato de esa posición y lo procesa.
* El **indirecto**, en el cual el procesador accede a la dirección de memoria que se encuentra dentro de su registro, obtiene una nueva dirección de memoria, obtiene el dato de esa posición y lo procesa.
* El **inmediato**, en el cual la instrucción tiene dentro de sí el dato que necesita para ser ejecutada, sin necesidad de recuperarlo de memoria.

Los **códigos de operación (CO)** son los comandos definidos dentro del procesador utilizados para ejecutar una instrucción, estos trabajan con modos de direccionamiento definidos.

Entre los recursos del procesador nombrados anteriormente y que serán utilizados en la resolución de este proyecto tenemos los siguientes:

* **Program Conuter (PC):** es el contador de programa, es donde se encuentra la dirección de memoria siguiente a la q se accederá para obtener una nueva instrucción. Cada vez que un ciclo de captación culmina el PC se incrementa en uno.
* **Instruction Register (IR):** este registro es el encargado de decodificar una instrucción, es decir de separar el CO y las direcciones.
* **Acumulator (AC):** este registro es utilizado para almacenar datos y posteriormente ser utilizados para operaciones con ALU, cuando interactúa con esta, además de ser el registro q provee d los datos también es el registro que recibe el resultado.
* **Data Register (DR):** como su nombre indica es un registro utilizado para almacenar datos, posteriormente es utilizado como proveedor de datos para operaciones de ALU.
* **Memory Address Register (MAR):** Es el registro utilizado para almacenar la siguiente dirección de memoria a la cual se accederá e interactúa directamente con el bus interno del CPU.
* **Memory Buffer Register (MBR):** Es el registro utilizado para almacenar los datos obtenidos desde memoria o para almacenar el dato que será guardado en la dirección de memoria a la cual se accederá e interactúa directamente con el bus interno del CPU.
* **Banco de registros:** como su nombre da a entender, es una cantidad definida de registros en los cuales se almacenaran datos obtenidos desde memoria o que serán destinados a la misma, y pueden interactuar con la ALU ya sea como entrada o como salida.
* **Arithmetical Logical Unit (ALU):** es el recurso del CPU encargado de las operaciones de aritméticas y lógicas, recibiendo datos de los registros del CPU y devolviendo el resultado a uno de ellos.

### 1.1.3. Interacción entre Procesador y Memoria Principal

La interacción entre memoria y procesador es llevada a cabo mediante buses, los cuales transportan el contenido de registros y señales de control tanto dentro como fuera del procesador (hacia memoria o dispositivos E/S), los buses utilizados por un procesador se clasifican:

* **Por su posición en relación al CPU:**
  + **Internos:** son los buses que transportan la información dentro del procesador, es decir, entre registros, ALU y Unidad de Control.
  + **Externos**: son los buses que transportan la información hacia la memoria principal u otros dispositivos (como los dispositivos de E/S).
* **Por el tipo de datos que transportan:**
  + **Bus de direcciones:** este bus es el encargado de transportar las direcciones de memoria o de dispositivos E/S.
  + **Bus de datos:** este se encarga de transportar los datos almacenados en los registros de memoria, dispositivos E/S y de los recursos del CPU.
  + **Bus de control:** Se encarga de transportar los códigos de control generados por la unidad de control.

## 1.2. Software educativo

### 1.2.1. Definición

Recibe la denominación de **software educativo** al software destinado a apoyar al proceso de enseñanza – aprendizaje, promoviendo el desarrollo cognitivo del usuario en ciertas áreas específicas, englobando todos los sistemas elaborados con un fin didáctico.

**Características esenciales de los programas educativos:**

Los programas educativos pueden tratar las diferentes áreas del conocimiento humano de formas muy diversas y ofrecer un entorno de trabajo más o menos sensible a las circunstancias de los alumnos y más o menos rico en posibilidades de interacción; pero todos comparten cinco **características esenciales**:

* Son materiales elaborados con una **finalidad didáctica**.
* **Utilizan el computador** como soporte en el que los alumnos realizan las actividades que ellos proponen.
* **Son interactivos**, contestan inmediatamente las acciones de los estudiantes y permiten un intercambio de informaciones entre el computador y el estudiante.
* **Individualizan el trabajo** de los estudiantes, ya que se adaptan al ritmo de trabajo cada uno y pueden adaptar sus actividades según las actuaciones de los alumnos.
* **Son fáciles de usar**. Los conocimientos informáticos necesarios para utilizar la mayoría de estos programas son similares a los conocimientos de electrónica necesarios para usar un vídeo, es decir, son mínimos, aunque cada programa tiene unas reglas de funcionamiento que es necesario conocer (Pere Marquès).

### 1.2.2. Tipos de software didáctico

**1. Programas tutoriales**

Son programas que en mayor o menor medida dirigen, tutorean, el trabajo de los alumnos. Pretenden que, a partir de cierta información y actividades previstas de antemano, los estudiantes pongan en juego sus capacidades y aprendan o refuercen sus conocimientos y habilidades. Cuando se limitan a proponer ejercicios de refuerzo sin proporcionar explicaciones conceptuales previas se denominan programas tutoriales de ejercitación, como es el caso de los programas de preguntas y de los programas de adiestramiento psicomotor, que desarrollan la coordinación neuromotriz en actividades relacionadas con el dibujo, la escritura y otras habilidades psicomotrices.

En cualquier caso, son programas basados en los planteamientos conductistas de la enseñanza que comparan las respuestas de los alumnos con los patrones que tienen como correctos, guían el aprendizaje de los estudiantes y facilitan la realización de prácticas predefinidas más o menos rutinarias y su evaluación; en algunos casos una evaluación negativa genera una nueva serie de ejercicios de repaso.

**2. Bases de datos**

Proporcionan datos organizados, en un entorno estático, según determinados criterios, y facilitan su exploración y consulta selectiva. Se pueden emplear en múltiples actividades como por ejemplo: seleccionar datos relevantes para resolver problemas, analizar y relacionar datos, extraer conclusiones, comprobar hipótesis, etc. Las preguntas que acostumbran a realizar los alumnos son del tipo: ¿Qué características tiene este dato? ¿Qué datos hay con la característica X? ¿Qué datos hay con las características X y Y?

Las bases de datos pueden tener una estructura jerárquica, relacional o documental. En cualquier caso, según la forma de acceder a la información se pueden distinguir dos tipos:

* **Bases de datos convencionales**. Tienen la información almacenada en ficheros, mapas o gráficos, que el usuario puede recorrer según su criterio para recopilar información.
* **Bases de datos tipo sistema experto**. Son bases de datos muy especializadas que recopilan toda la información existente de un tema concreto y además asesoran al usuario cuando accede buscando determinadas respuestas.

**3. Simuladores**

Presentan un modelo o entorno dinámico (generalmente a través de gráficos o animaciones interactivas) y facilitan su exploración y modificación a los alumnos, que pueden realizar aprendizajes inductivos o deductivos mediante la observación y la manipulación de la estructura subyacente; de esta manera pueden descubrir los elementos del modelo, sus interrelaciones, y pueden tomar decisiones y adquirir experiencia directa delante de unas situaciones reales, ya sean mucho o poco probables.

En [cualquier](http://www.lmi.ub.es/te/any96/marques_software/" \l "index) caso, los simuladores están orientados al aprendizaje de un entorno real o cercano a la realidad, y representan consigo una interacción con el usuario dependiente de las necesidades tanto del entorno a simular como de los usuarios finales.

**4. Constructores**

Son programas que tienen un entorno programable. Facilitan a los usuarios unos elementos simples con los cuales pueden construir elementos más complejos o entornos. De esta manera potencian el aprendizaje heurístico y, de acuerdo con las **teorías cognitivistas**, facilitan a los alumnos la construcción de sus propios aprendizajes, que surgirán a través de la reflexión que realizarán al diseñar programas y comprobar inmediatamente, cuando los ejecuten, la relevancia de sus ideas. El proceso de creación que realiza el alumno genera preguntas del tipo: **¿Qué sucede si añado o elimino el elemento X?** Se pueden distinguir dos tipos de constructores:

**5. Programas herramienta**

Son programas que proporcionan un entorno instrumental con el cual se facilita la **realización de ciertos trabajos generales** de tratamiento de la información: escribir, organizar, calcular, dibujar, transmitir, captar datos, etc. A parte de los lenguajes de autor, los más utilizados son programas de uso general que provienen del mundo laboral y, por tanto, quedan fuera de la definición que se ha dado de software educativo (Pere Marquès).

No obstante, existen herramientas orientadas a la enseñanza, siendo definidas básicamente como software interactivo.

Estas herramientas interactivas pueden ser orientadas al aprendizaje de un área o tema en específico tomando características básicas de los otros tipos de software anteriormente mencionados. De esta forma esta categoría puede ser una combinación entre un programa tutorial, un simulador y un sistema interactivo común y corriente.

La facilidad de adaptar este tipo de sistemas a las necesidades del usuario para su aplicación en el área de enseñanza-aprendizaje hace este tipo de herramientas una de las más fáciles de aplicar.

## 1.3. Metodología de desarrollo de Software

### 1.3.1. Generalidades

**Un proceso de desarrollo de software detallado y completo suele recibir el nombre de “Metodología”. Una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas. Habitualmente se utiliza el término “método” para referirse a técnicas, notaciones y guías asociadas, que son aplicables a una (o algunas) actividades del proceso de desarrollo, por ejemplo, suele hablarse de métodos de análisis o diseño.**

**La comparación y/o clasificación de metodologías no es una tarea sencilla debido a la diversidad de propuestas y diferencias en el grado de detalle, información disponible y alcance de cada una de ellas (Anónimo, 2014).**

**“Una determinada metodología no es necesariamente aplicable a cualquier tipo de proyecto, es más, para cada tipo de proyecto existe una metodología que se adapta mejor a sus necesidades” (**Hermes Romero, 2012)**.**

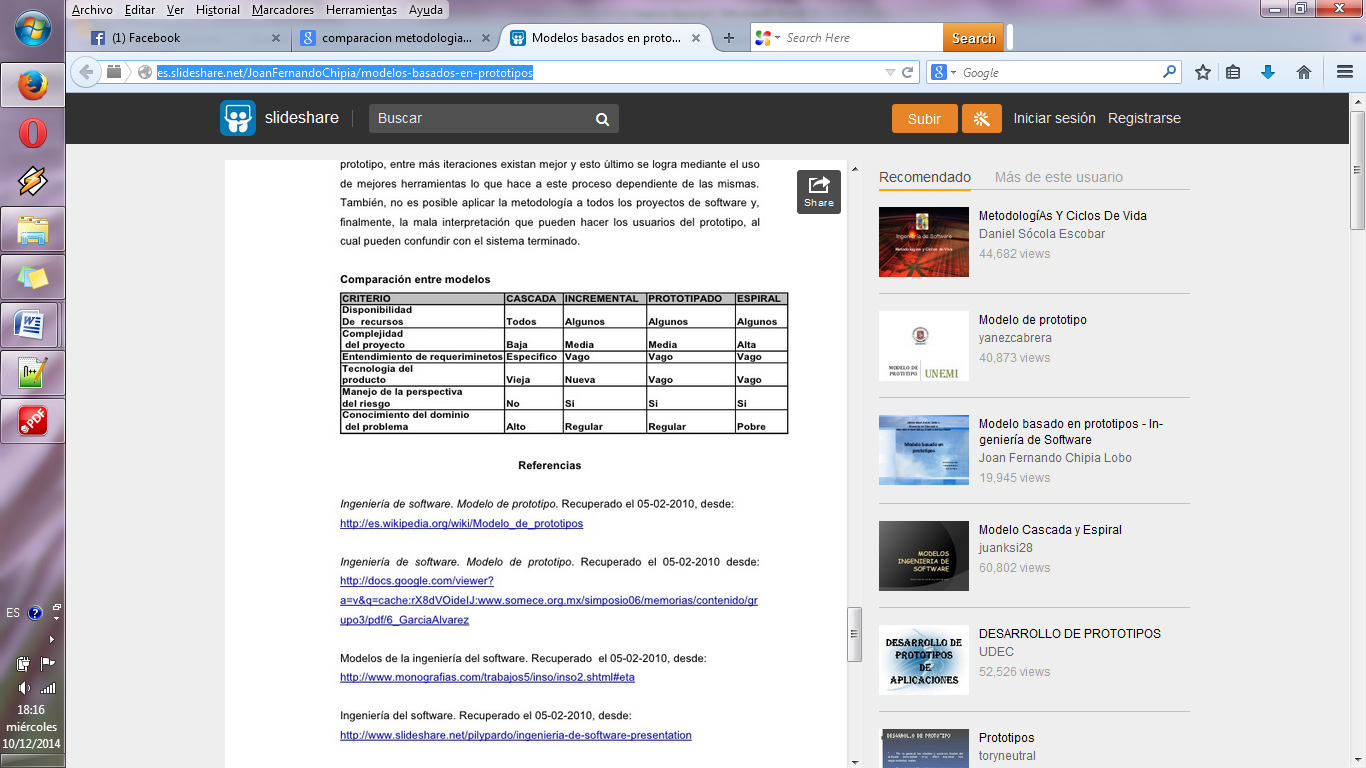
La ingeniería de software requiere llevar a cabo numerosas tareas, dentro de etapas como las siguientes:

* **Captura de Requerimientos:** primeramente para el desarrollo de todo tipo de software, se debe analizar y definir los requerimientos funcionales, es decir, requisitos que el sistema debe satisfacer para con el campo de trabajo al que será aplicado.
* **Análisis:** posteriormente se analizan los requerimientos y se van definiendo requerimientos no funcionales y funcionalidades principales.
* **Diseño:** el siguiente paso es diseñar el sistema en sí, modelarlo para su implementación posterior, se diseñan las clases y la forma en que se llegaran a satisfacer mediante la interacción de estas dentro del sistema.
* **Implementación:** en base al diseño realizado anteriormente se empieza a implementar el modelo y se lo vuelve un sistema.
* **Experimentación:** finalmente se prueba que el sistema implementado cumpla con todos los requerimientos funcionales y no funcionales, se buscan errores y se los corrige, además de afinar los últimos detalles.

Cada una de estas tareas dentro del proceso de desarrollo varía en estructura, secuencialidad e importancia, dependientemente de la metodología utilizada.

### 1.3.2. Selección de metodología.

**Tabla 2. Tabla comparativa de metodologías de desarrollo.**



Fuente: CHIPIA LOBO, 2010

**Características principales del proyecto:**

* Requerimientos funcionales aun ambiguos.
* Necesidad de retroalimentación.
* Riesgo alto de insatisfacción de los usuarios finales.
* Disponibilidad de recursos reducidos.
* Necesidad de contar con versiones del sistema funcionales.

El utilizar el prototipado como metodología de desarrollo claramente satisface las características principales del proyecto, dando también un mayor nivel de flexibilidad en el desarrollo, siendo una de las metodologías más utilizadas en el software educativo.

### 1.3.3. Prototipado

Esta metodología de desarrollo está comprendida entre los modelos de desarrollo evolutivo. Como tal, cada prototipo de utilizar pocos recursos y debe ser construido en un tiempo razonablemente corto, orientado a la retroalimentación con el usuario final.

Existen dos maneras de usar el Prototipado, de una manera incremental reutilizando prototipos anteriores, a esto se lo denomina evolutivo. La otra manera es con prototipos “desechables”, es decir, que se rescata todo lo importante y se rediseña el sistema con las correcciones y se incrementa en cuanto a cobertura de requerimientos.

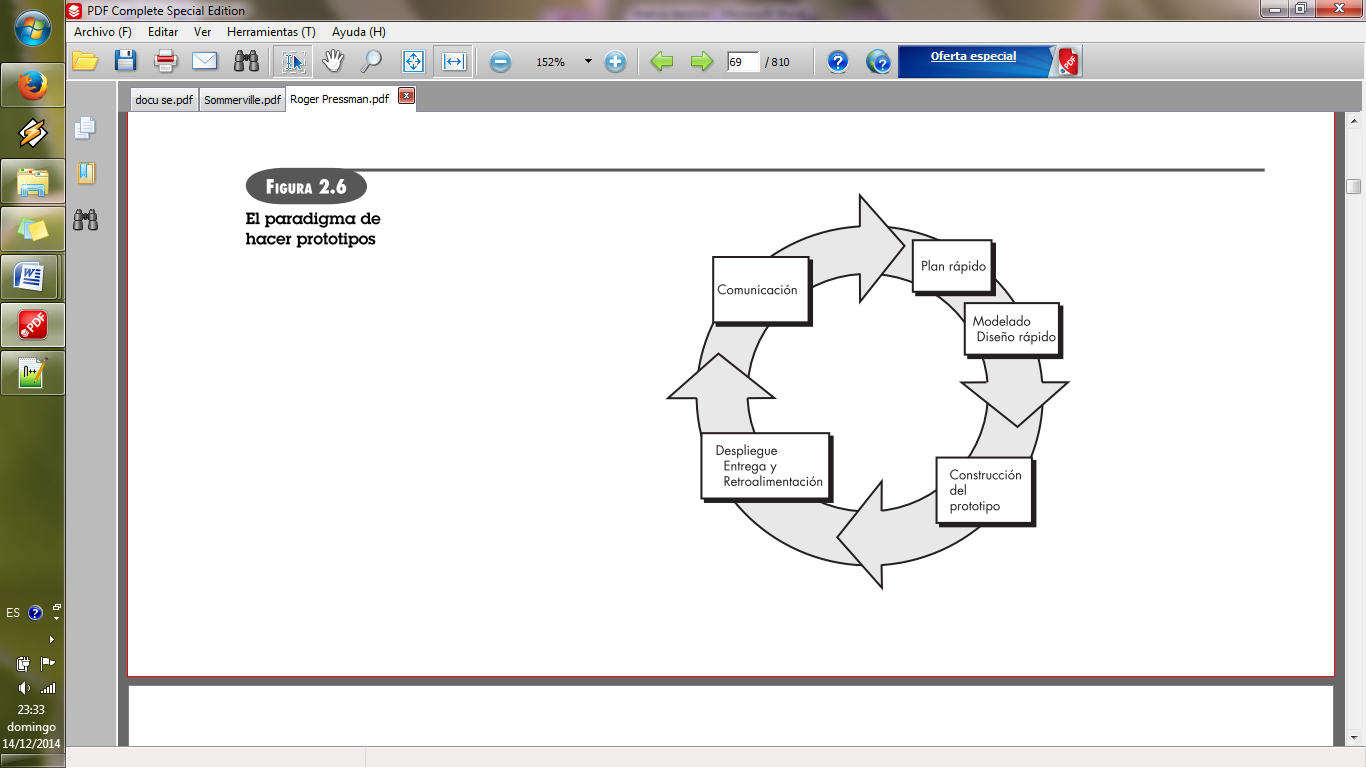
Es frecuente que un cliente defina un conjunto de objetivos generales para el software, pero que no identifique los requerimientos detallados para las funciones y características. En otros casos, el desarrollador tal vez no esté seguro de la eficiencia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma que debe adoptar la interacción entre el humano y la máquina. En estas situaciones, y muchas otras, el paradigma de hacer prototipos tal vez ofrezca el mejor enfoque.

Aunque es posible hacer prototipos como un modelo de proceso aislado, es más común usarlo como una técnica que puede implementarse en el contexto de cualquiera de los modelos de proceso descritos en este capítulo. Sin importar la manera en la que se aplique, el paradigma de hacer prototipos le ayudará a usted y a otros participantes a mejorar la comprensión de lo que hay que elaborar cuando los requerimientos no están claros.

El paradigma de hacer prototipos comienza con comunicación. Usted se reúne con otros participantes para definir los objetivos generales del software, identifica cualesquiera requerimientos que conozca y detecta las áreas en las que es imprescindible una mayor definición. Se planea rápidamente una iteración para hacer el prototipo, y se lleva a cabo el modelado (en forma de un “diseño rápido”). Éste se centra en la representación de aquellos aspectos del software que serán visibles para los usuarios finales (por ejemplo, disposición de la interfaz humana o formatos de la pantalla de salida). El diseño rápido lleva a la construcción de un prototipo. Éste se entrega y es evaluado por los participantes, que dan retroalimentación para mejorar los requerimientos. La iteración ocurre a medida de que el prototipo es afinado para satisfacer las necesidades de distintos participantes, y al mismo tiempo le permite a usted entender mejor lo que se necesita hacer.

El ideal es que el prototipo sirva como mecanismo para identificar los requerimientos del software. Si va a construirse un prototipo, pueden utilizarse fragmentos de programas existentes o aplicar herramientas (por ejemplo, generadores de reportes y administradores de ventanas) que permitan generar rápidamente programas que funcionen (PRESSMAN, 2010: 37).

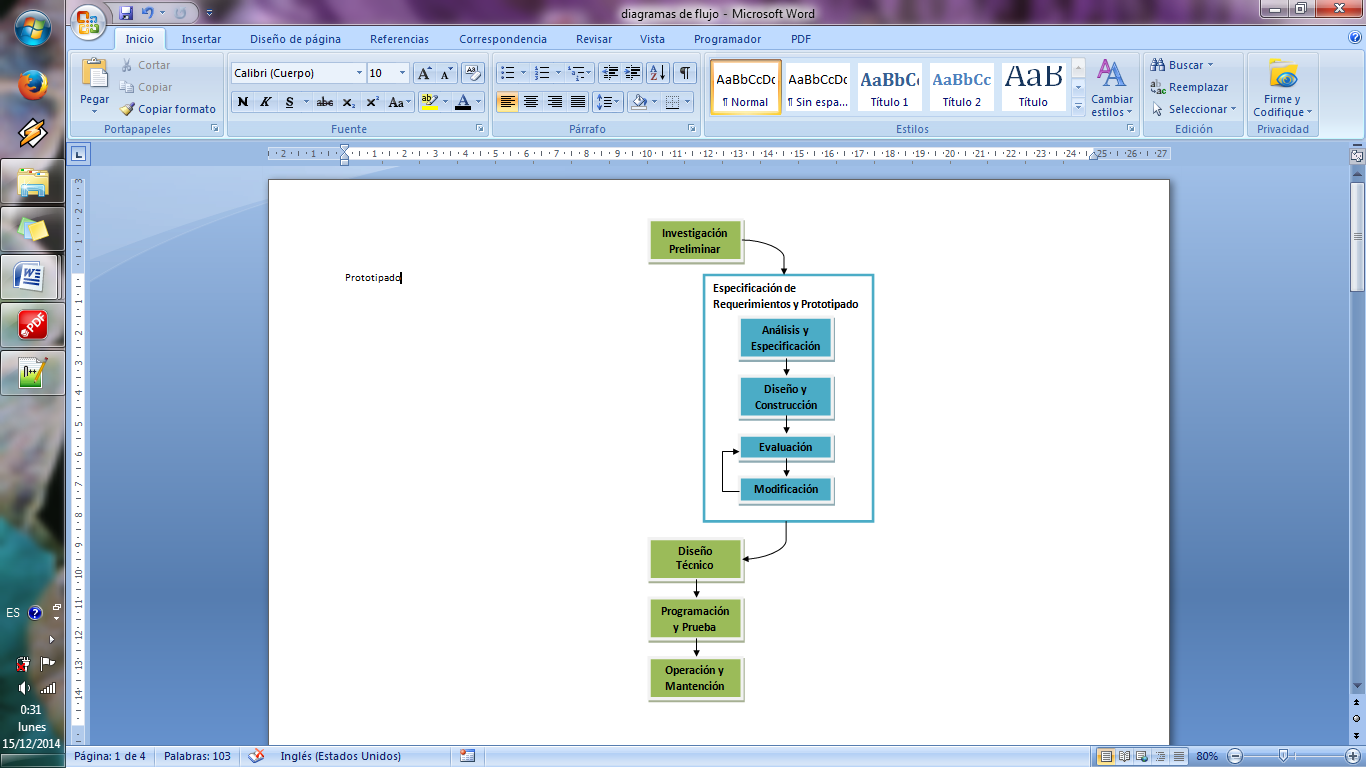
**Figura 2. Paradigma de hacer prototipos.**



Fuente: PRESSMAN 2010.

### 1.3.4. Etapas del prototipado

**Figura 3. Etapas del Modelo de Prototipos**

****

Fuente: Elaboración propia en base a YANEZ CABRERA 2014

## 1.4. Tecnologías de desarrollo

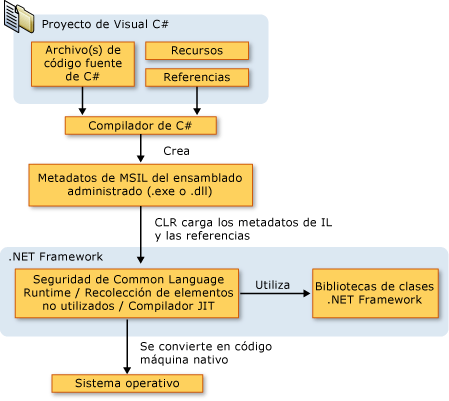
### 1.4.1 Visual Studio

Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C# y Visual C++ utilizan todos el mismo entorno de desarrollo integrado (IDE), que habilita el uso compartido de herramientas y hace más sencilla la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes utilizan las funciones de .NET Framework, las cuales ofrecen acceso a tecnologías clave para simplificar el desarrollo de aplicaciones web ASP y Servicios Web XML. (MSDN, 2007)

### 1.4.2. C#

C# es un lenguaje orientado a objetos elegante y con seguridad de tipos que permite a los desarrolladores compilar diversas aplicaciones sólidas y seguras que se ejecutan en .NET Framework. Puede utilizar C# para crear aplicaciones cliente de Windows, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos, y mucho, mucho más. Visual C# proporciona un editor de código avanzado, cómodos diseñadores de interfaz de usuario, depurador integrado y numerosas herramientas más para facilitar el desarrollo de aplicaciones basadas el lenguaje C# y .NET Framework. (MSDN, 2014)

**Figura 4. Arquitectura de .NET**



Fuente: MSDN 2014.

### 1.4.3. Librerías utilizadas en el proyecto:

**System:** Es la librería principal del programa, para el manejo de datos y sentencias, al igual que el engranaje de compilación.

**System.Collections.Generic:** El espacio de nombres **System.Collections.Generic** contiene interfaces y clases que definen colecciones genéricas, permitiendo que los usuarios creen colecciones con establecimiento inflexible de tipos para proporcionar una mayor seguridad de tipos y un rendimiento mejor que los de las colecciones no genéricas con establecimiento inflexible de tipos.

**System.ComponentModel:** El espacio de nombres **System.ComponentModel** proporciona clases que se utilizan para implementar el comportamiento en tiempo de ejecución y tiempo de diseño de los componentes y controles. Este espacio de nombres incluye las interfaces y clases base para implementar atributos y convertidores de tipos, enlazar a orígenes de datos y dotar de licencia a componentes. Las clases en este espacio de nombres se dividen en las siguientes categorías:

* Clases de componentes básicos.
* Licencia de componentes.
* Atributos.
* Descriptores y persistencia.
* Convertidores de tipos.

**System.Data:** El espacio de nombres **System.Data** proporciona acceso a las clases que representan la arquitectura de ADO.NET. ADO.NET le permite generar componentes que administran eficazmente los datos de varios orígenes de datos.

En un escenario desconectado como Internet, ADO.NET proporciona las herramientas para solicitar, actualizar y conciliar datos en sistemas de varios niveles. La arquitectura ADO.NET también se implementa en aplicaciones cliente, como los formularios Windows Forms o las páginas HTML creadas mediante ASP.NET.

El eje de la arquitectura ADO.NET es la clase [DataSet](http://msdn.microsoft.com/es-es/library/system.data.dataset(v=vs.80).aspx). Cada **DataSet** puede contener varios objetos [DataTable](http://msdn.microsoft.com/es-es/library/system.data.datatable(v=vs.80).aspx), y cada **DataTable** contiene datos de un solo origen, como SQL Server.

Cada **DataTable** contiene un [DataColumnCollection](http://msdn.microsoft.com/es-es/library/system.data.datacolumncollection(v=vs.80).aspx) (una colección de objetos [DataColumn](http://msdn.microsoft.com/es-es/library/system.data.datacolumn(v=vs.80).aspx)) que determina el esquema de cada **DataTable**. La propiedad [DataType](http://msdn.microsoft.com/es-es/library/system.data.datacolumn.datatype(v=vs.80).aspx) determina el tipo de datos contenidos en **DataColumn**. Las propiedades [ReadOnly](http://msdn.microsoft.com/es-es/library/system.data.datacolumn.readonly(v=vs.80).aspx) y [AllowDBNull](http://msdn.microsoft.com/es-es/library/system.data.datacolumn.allowdbnull(v=vs.80).aspx) permiten garantizar aún más la integridad de datos. La propiedad [Expression](http://msdn.microsoft.com/es-es/library/system.data.datacolumn.expression(v=vs.80).aspx) permite construir columnas calculadas.

Si un objeto **DataTable** participa en una relación primaria-secundaria con otro **DataTable**, la relación se construye agregando un objeto [DataRelation](http://msdn.microsoft.com/es-es/library/system.data.datarelation(v=vs.80).aspx) a la colección [DataRelationCollection](http://msdn.microsoft.com/es-es/library/system.data.datarelationcollection(v=vs.80).aspx) de un objeto **DataSet**. Cuando se agrega una relación de este tipo, se crean automáticamente un objeto [UniqueConstraint](http://msdn.microsoft.com/es-es/library/system.data.uniqueconstraint(v=vs.80).aspx) y un objeto [ForeignKeyConstraint](http://msdn.microsoft.com/es-es/library/system.data.foreignkeyconstraint(v=vs.80).aspx) de acuerdo con la configuración de los parámetros del constructor. **UniqueConstraint** garantiza que los valores que contiene una columna son únicos. **ForeignKeyConstraint** determina qué acción se llevará a cabo en la fila o columna secundaria cuando se modifique o elimine un valor de la clave principal.

Mediante el espacio de nombres [System.Data.SqlClient](http://msdn.microsoft.com/es-es/library/system.data.sqlclient(v=vs.80).aspx) (proveedor de datos de .NET Framework para SQL Server), el espacio de nombres [System.Data.Odbc](http://msdn.microsoft.com/es-es/library/system.data.odbc(v=vs.80).aspx) (proveedor de datos de .NET Framework para ODBC), el espacio de nombres [System.Data.OleDb](http://msdn.microsoft.com/es-es/library/system.data.oledb(v=vs.80).aspx) (proveedor de datos de .NET Framework para OLE DB) o el espacio de nombres [System.Data.OracleClient](http://msdn.microsoft.com/es-es/library/system.data.oracleclient(v=vs.80).aspx) (proveedor de datos de .NET Framework para Oracle), se puede obtener acceso al origen de datos para utilizarlo junto con un objeto **DataSet**. Cada proveedor de datos de .NET Framework tiene una clase **DataAdapter** correspondiente que se utiliza como puente entre un origen de datos y un objeto **DataSet**.

**System.Drawing:** El espacio de nombres **System.Drawing** proporciona acceso a funcionalidad de gráficos básica de GDI+. Se ofrece una funcionalidad más avanzada en los espacios de nombres [System.Drawing.Drawing2D](http://msdn.microsoft.com/es-es/library/system.drawing.drawing2d(v=vs.80).aspx), [System.Drawing.Imaging](http://msdn.microsoft.com/es-es/library/system.drawing.imaging(v=vs.80).aspx) y [System.Drawing.Text](http://msdn.microsoft.com/es-es/library/system.drawing.text(v=vs.80).aspx).

La clase [Graphics](http://msdn.microsoft.com/es-es/library/system.drawing.graphics(v=vs.80).aspx) proporciona métodos para dibujar en el dispositivo de pantalla. Clases como [Rectangle](http://msdn.microsoft.com/es-es/library/system.drawing.rectangle(v=vs.80).aspx) y [Point](http://msdn.microsoft.com/es-es/library/system.drawing.point(v=vs.80).aspx) encapsulan primitivos de GDI+. La clase [Pen](http://msdn.microsoft.com/es-es/library/system.drawing.pen(v=vs.80).aspx) se utiliza para dibujar líneas y curvas, mientras que las clases derivadas de la clase abstracta [Brush](http://msdn.microsoft.com/es-es/library/system.drawing.brush(v=vs.80).aspx) se utilizan para rellenar el interior de las formas.

**System.Linq:** Librería manejada principalmente para conexión y manejo de datos con SQL.

**System.Text:** El espacio de nombres **System.Text** contiene clases que representan codificaciones de caracteres ASCII, Unicode, UTF-7, y UTF-8; clases base abstractas para la conversión de bloques de caracteres en bloques de bytes y viceversa; y una clase auxiliar que manipula y da formato a objetos [String](http://msdn.microsoft.com/es-es/library/system.string(v=vs.80).aspx) sin necesidad de crear instancias intermedias de **String**.

**System.Windows.Forms:** El espacio de nombres **System.Windows.Forms** contiene clases para crear aplicaciones para Windows que aprovechan todas las ventajas de las características de la interfaz de usuario disponibles en el sistema operativo Microsoft Windows. (MSDN, 2014)

## Capítulo 2. Marco Práctico

## 2.1. Diseño de componentes de interfaz que permitan visualizar arquitectura del CPU.

### 2.1.1. Prototipo 1

Requerimientos:

* Diseñar la estructura de la arquitectura del CPU basada en registros específico.
* Diseñar estructura de memoria principal y de buses externos del CPU.

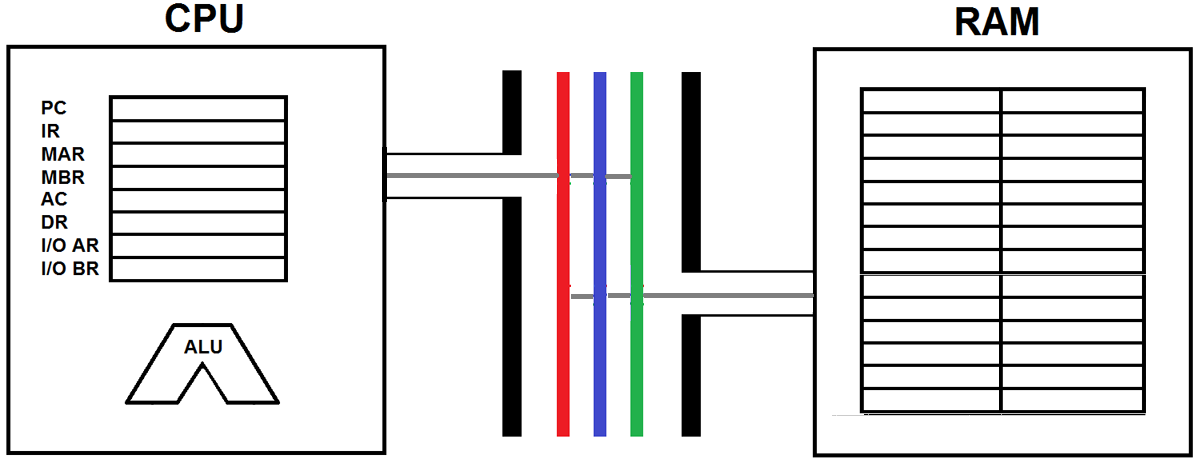
Para este objetivo se opto por diseñar primeramente una imagen fija como fondo de la interfaz de usuario.

Se diseñó en principio una imagen la cual serviría como plantilla para luego instalar los componentes de programación necesarios para que el estudiante pueda ingresar los contenidos de los registros.

Primeramente se plasmó el diseñó del CPU con arquitectura basada en registros específicos y sus componentes en la imagen “plantilla”, y posteriormente se procedió al diseño de los buses externos del CPU. Por último se diseño la estructura de la memoria principal.

Cuando la imagen ya se encontraba definida, se prosiguió a la instalación de los componentes de programación, instalando un “TextBox” para cada registro de CPU y para cada registro y dirección de la memoria principal, además de asignar a cada “TextBox” un nombre significativo con su papel en la interacción con el estudiante.

**Figura 5. Interfaz Prototipo 1.**



Fuente: Elaboración propia 2014

### 2.1.2. Prototipo 2

Requerimientos:

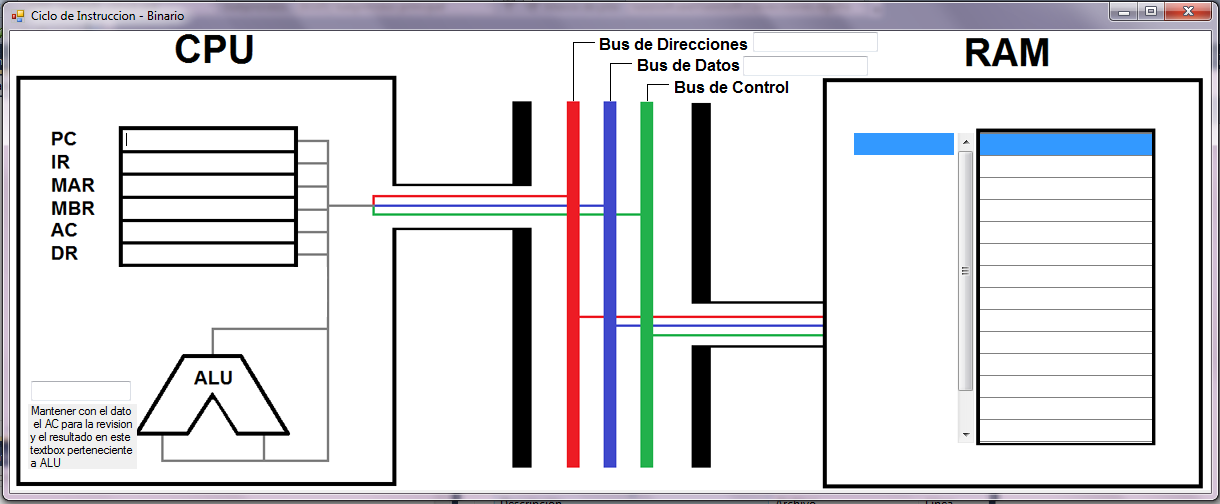
* Diseñar la estructura de la arquitectura del CPU basada en registros específico.
* Diseñar estructura de memoria principal y de buses externos del CPU.

Al mostrar el prototipo a estudiantes que ya cursaron la materia y analizar con este detalladamente las falencias de esta interfaz se pudieron conseguir las siguientes observaciones:

* El uso de los registros I/O AR e I/O BR era innecesario, puesto que la herramienta se encargaría proveer un ambiente para la resolución de ejercicios que reflejan la interacción de CPU con RAM, no con dispositivos de E/S.
* Otra observación importante es que no se encontraba un enlace gráfico entre los componentes de CPU, por lo que se instalo un bus interno de CPU en la interfaz.
* Al salir la instrucción del CPU hacia la RAM, se debería diferenciar hacia donde la información contenida, es decir, hacia cual de los buses.
* En la imagen no queda claro cuál es el *bus de direcciones*, *bus de datos* y *bus de control.*
* Los registros y direcciones de la RAM no deberían estar contenidos en un “TextBox”, puesto que esto dificultaría la verificación de pasos del ciclo de instrucción.

Para la instalación de registros y direcciones de la RAM en la interfaz de usuario se decide utilizar dos “DataGridView”, en el cual se generarían automáticamente las celdas al cargar el componente de interfaz de usuario.

**Figura 6. Interfaz Prototipo 2.**



Fuente: Elaboración propia 2014

Para que el “ScrollBar” funcione para ambos “DataGridView” de forma simultánea se utilizó el evento “Scroll” de este componente.

También se utilizaron los eventos “MouseClick” y “SelectionChanged” para que cuando se seleccione una dirección de memoria, automáticamente se seleccione su correspondiente registro y viceversa.

## 2.2. Desarrollo del módulo interactivo para trabajar con lenguaje binario.

### 2.2.1. Prototipo 3

Requerimientos:

* Desarrollar componente de interacción con el estudiante, para desplazar el contenido de registros.

Después de haber conseguido el diseño de una interfaz de usuario definida, se procedió al desarrollo del componente de interacción para este módulo.

El componente se desarrolló para que el estudiante pueda realizar el llenado y manipulación del contenido de registros tanto del CPU como de la RAM, ya sea mediante la escritura de datos desde teclado, como la copia de contenidos de un registro a otro.

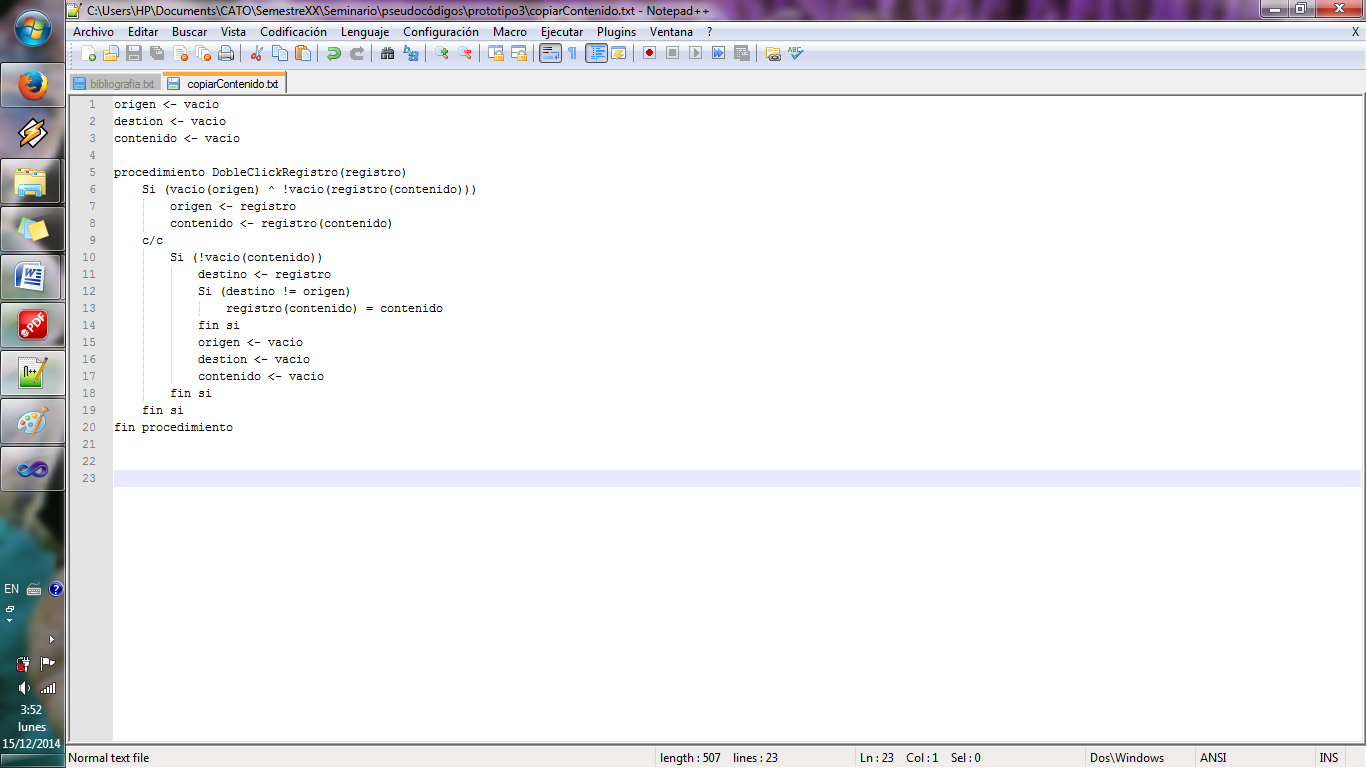
Se programaron restricciones en la escritura, solo se permitió el ingreso de datos con los caracteres “0” y “1”, mediante el evento “KeyPress”, así como también se limitó la longitud máxima de caracteres a ocho, con la propiedad “MaxLength”. Si el valor era menor se autocompletaban los registros con ceros.

En la programación de la copia del contenido de registros hubo más complicaciones, por lo que los sucesos importantes se resumen en los siguientes puntos:

* En principio se intento utilizar los eventos “MouseUp” y “MouseDown”, pero se presentaron inconvenientes. Si se hacía click en el registro “A” y se soltaba el botón del mouse en el registro “B”, de todas formas se ejecutaba el código del evento del registro “A” y no el del “B”, como si se hubiese hecho click en “A” y soltado el botón también en “A”, por lo que no había ningún cambio.
* Se procedió a la búsqueda de posibles soluciones para cumplir con esta función, analizando las ventajas y desventajas de otros eventos o comandos como “ctrl+c” y “ctrl+v”.
* Finalmente se optó por elegir el evento “MouseDoubleClick”, siendo que se haría doble click en el registro origen y posteriormente doble click en el registro destino para poder copiar el contenido.
* Se optó por este evento debido a que al ser utilizado el programa con un touchpad en computadoras portátiles, muchas veces el usuario puede cometer un error al hacer un solo click en el lugar incorrecto debido a la sensibilidad del touchpad.

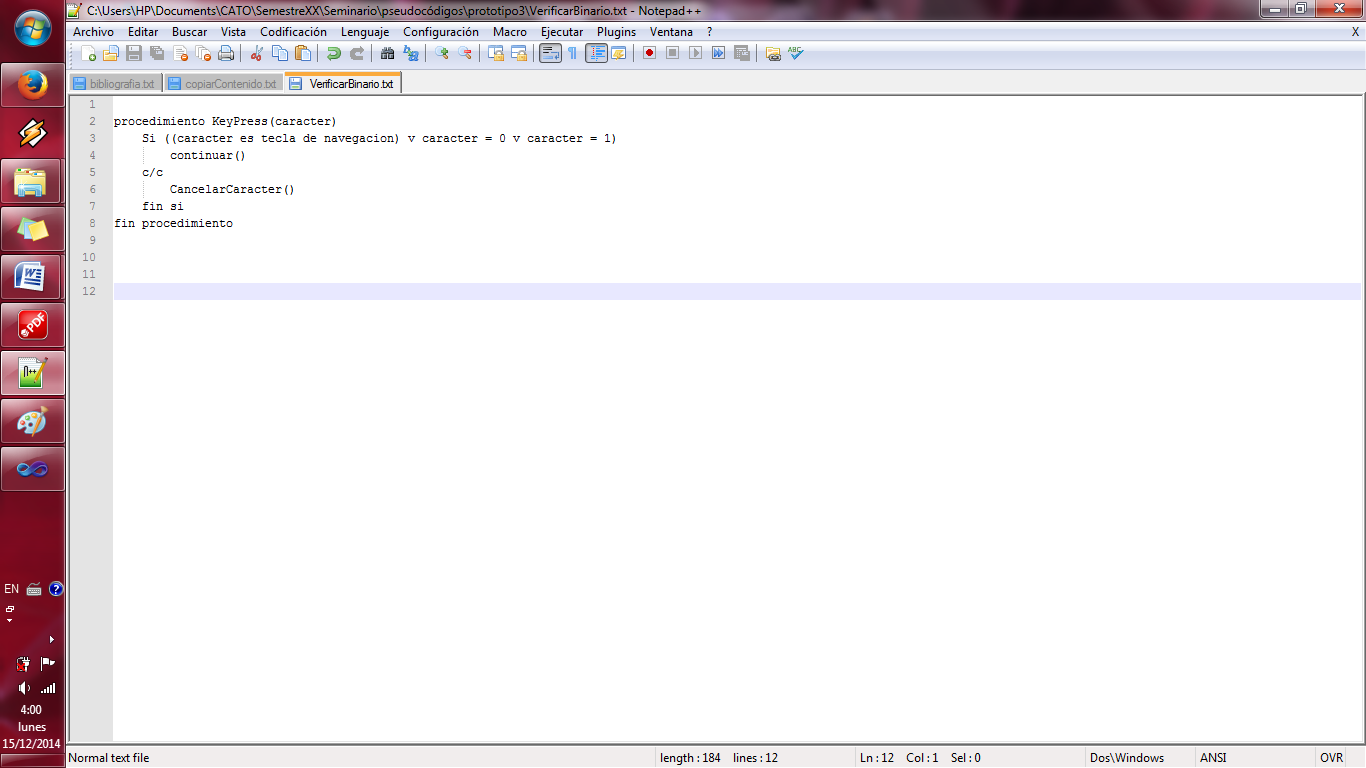
En este prototipo también se decidió generar automáticamente el contenido de la RAM para darle mayor realismo a la interfaz de usuario, generando números consecutivos desde el 0 hasta el 28 en binario.

**Figura 7. Pseudocódigo MouseDoubleClick.**



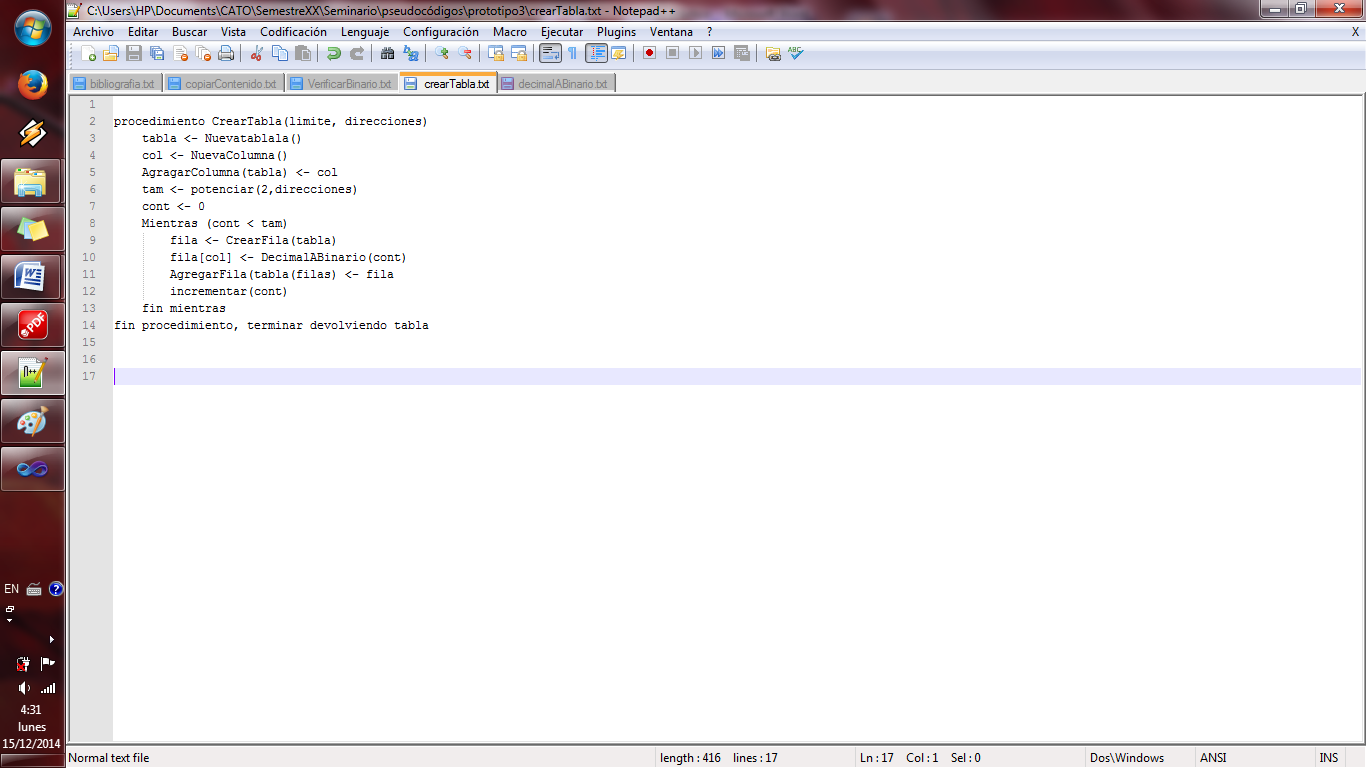
Fuente: Elaboración propia 2014

**Figura 8. Pseudocódigo KeyPress.**

****

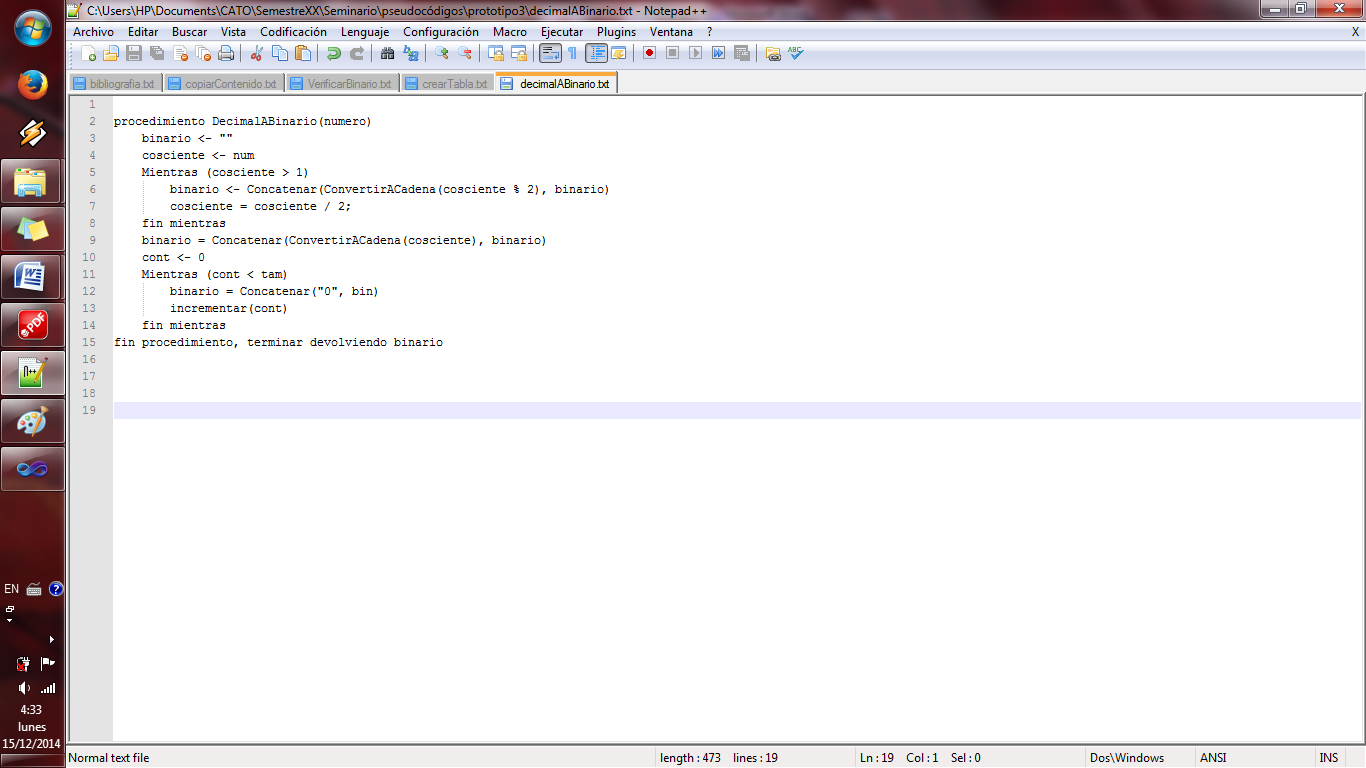
Fuente: Elaboración propia 2014

**Figura 9. Pseudocódigo Llenar DataGriedView.**

****

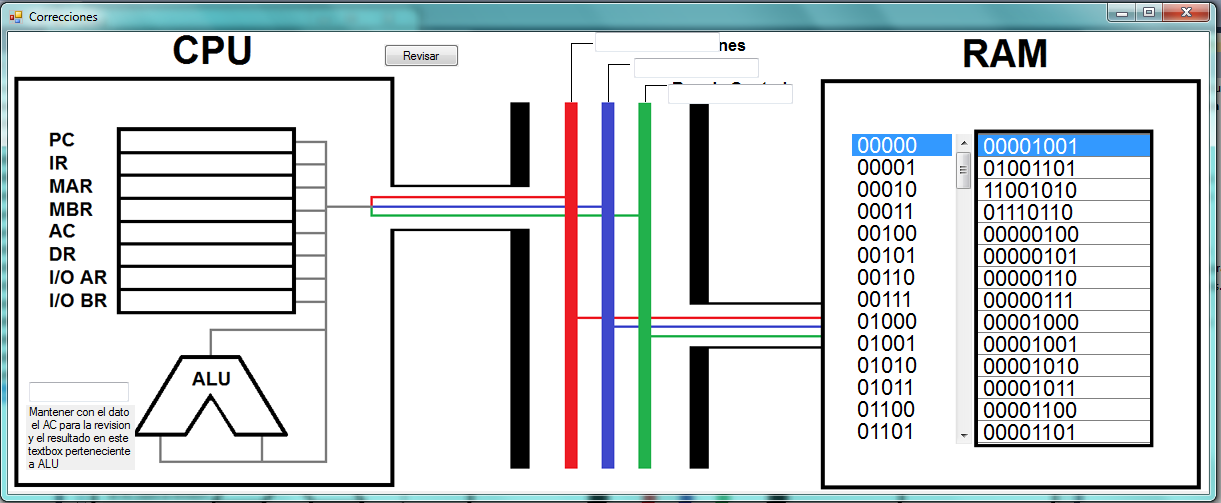
Fuente: Elaboración propia 2014

**Figura10. Pseudocódigo Decimal a Binario.**



Fuente: Elaboración propia 2014

**Figura11. Interfaz Prototipo 3.**



Fuente: Elaboración propia 2014

### 2.2.2. Prototipo 4

Requerimientos:

* Desarrollar componente de interacción con el estudiante, para desplazar el contenido de registros.
* Desarrollar componente interactivo de verificación de errores, en binario.

Al tener ya el componente de interacción con el usuario definido, se realizaron ajustes después de las siguientes observaciones:

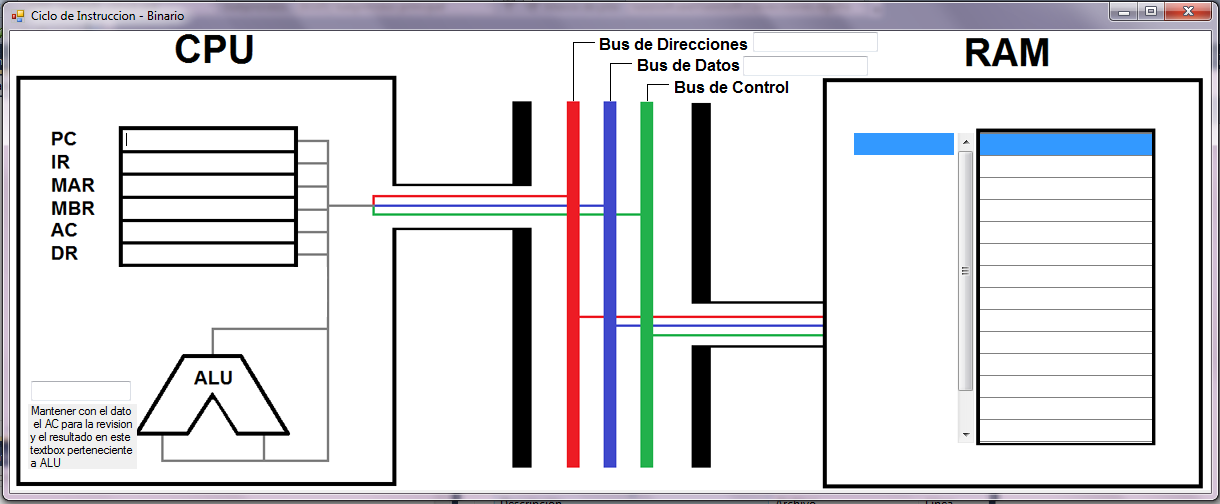
* La RAM llena confunde a los estudiantes, deben generarse solo las celdas y mantener el contenido de los registros y direcciones vacios.
* Se debe controlar la longitud de los campos, pero no se debe restringir al estudiante a escribir mayor contenido en un registro, pues este es uno de los errores comunes en los estudiantes, el cual el mismo debe controlar.

Además de esas modificaciones a ese componente, se procedió con el desarrollo del componente interactivo de verificación de errores, para el ciclo de captación e el de ejecución de la operación de lectura de un dato desde memoria. Se usaron los eventos:

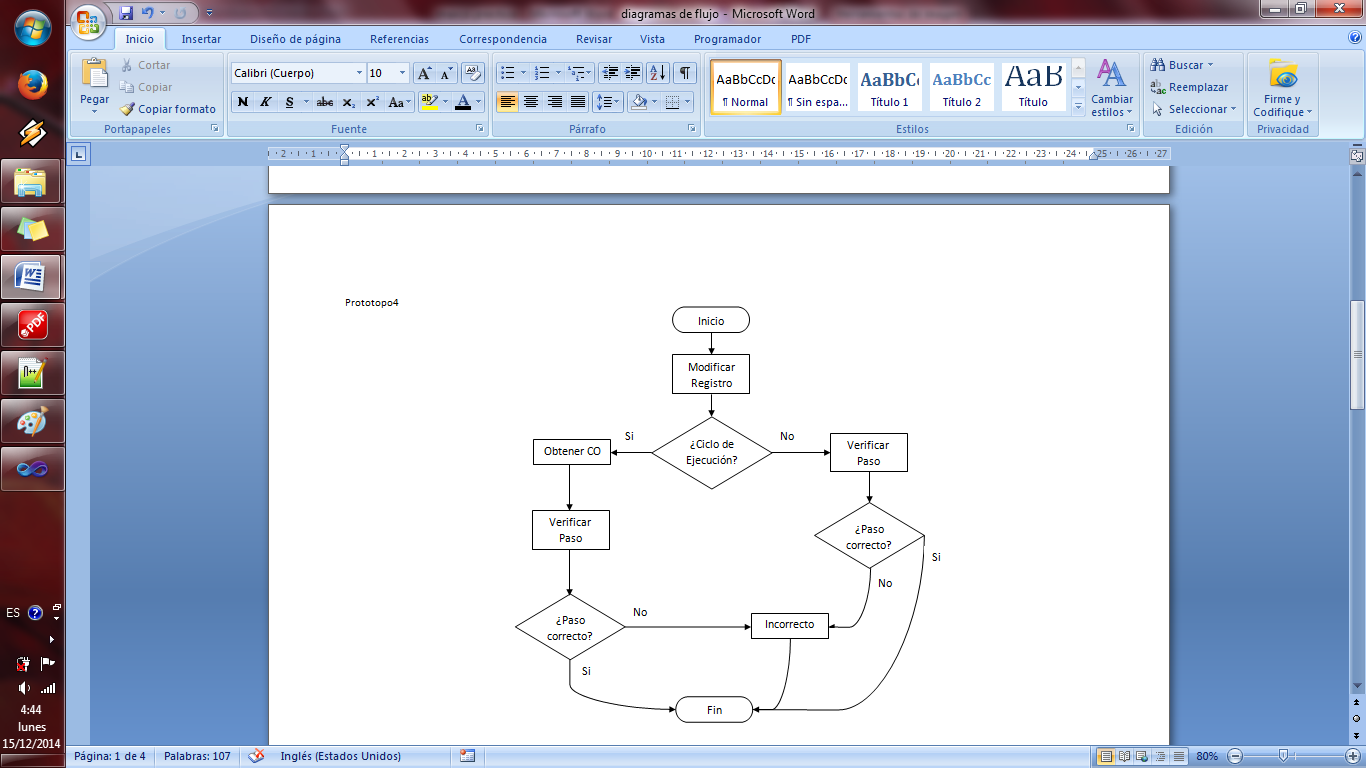
* “TextChanged” para el control de la longitud del registro, para que la longitud sea exactamente la misma a la definida como el tamaño de un registro de CPU (en este caso aun era predefinido como ocho).
* “MouseDobleClick” para la copia del contenido de registros, definiendo una variable origen con el nombre del registro de origen, y una variable destino como el nombre del registro destino, para verificar si era correcta la secuencia elegida por el usuario.

La verificación en síntesis consiste en verificar si el registro el cual cambió de contenido es el que debía cambiar por la correlación del proceso que se está ejecutando. Si es que es el correcto, es decir, si es que el usuario lleva el contenido del registro origen al registro destino correspondiente para ese paso del proceso, el usuario puede seguir modificando, caso contrario se muestra un mensaje de “Incorrecto”.

**Figura12. Interfaz Prototipo 4.**

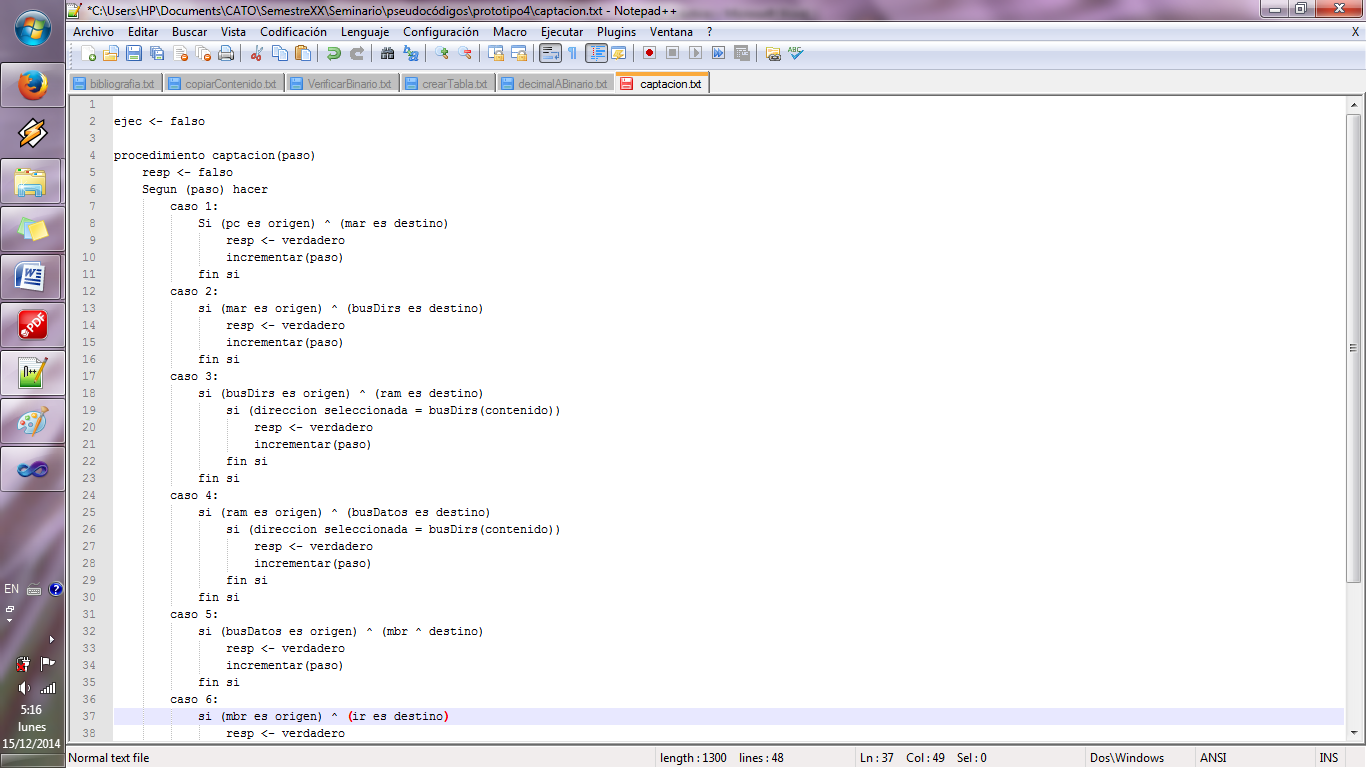
Fuente: Elaboración propia 2014

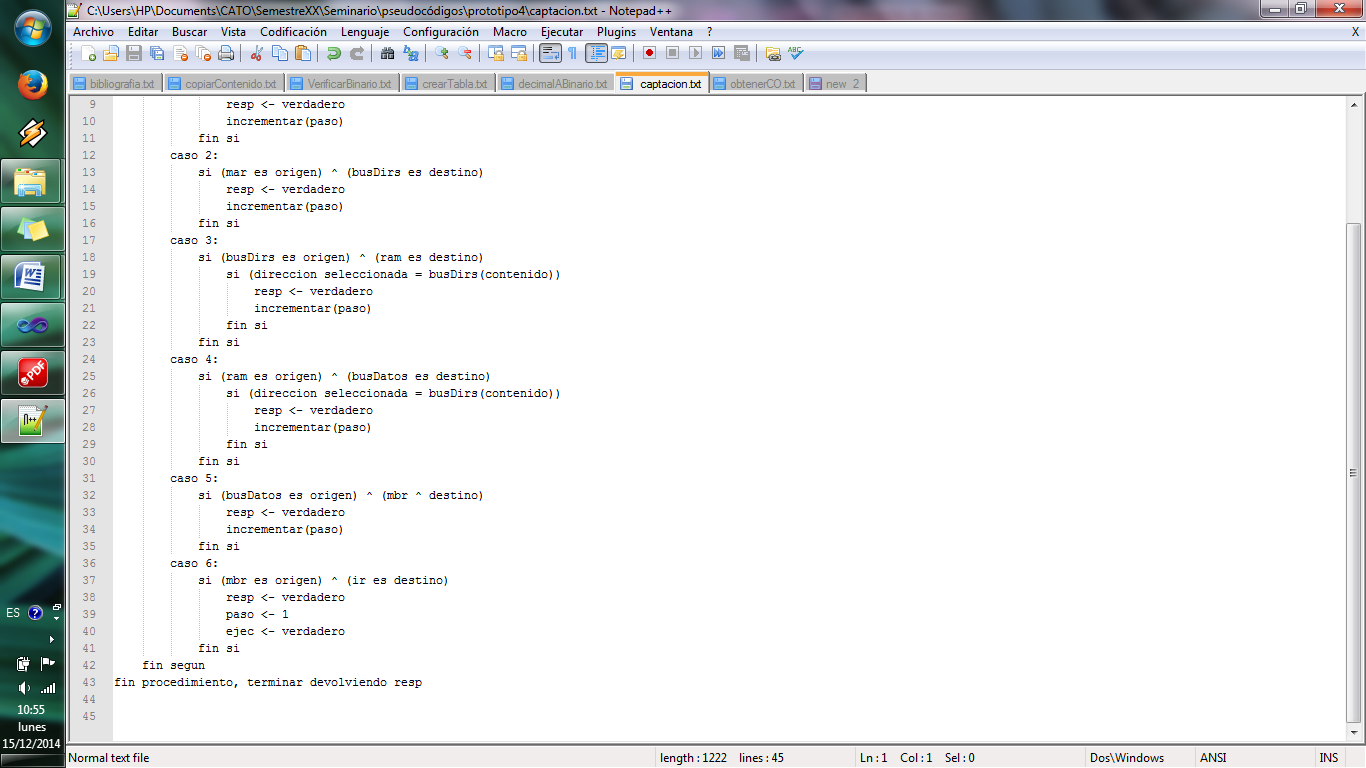
**Figura13. Diagrama de Flujo, Verificación de errores.**



Fuente: Elaboración propia 2014

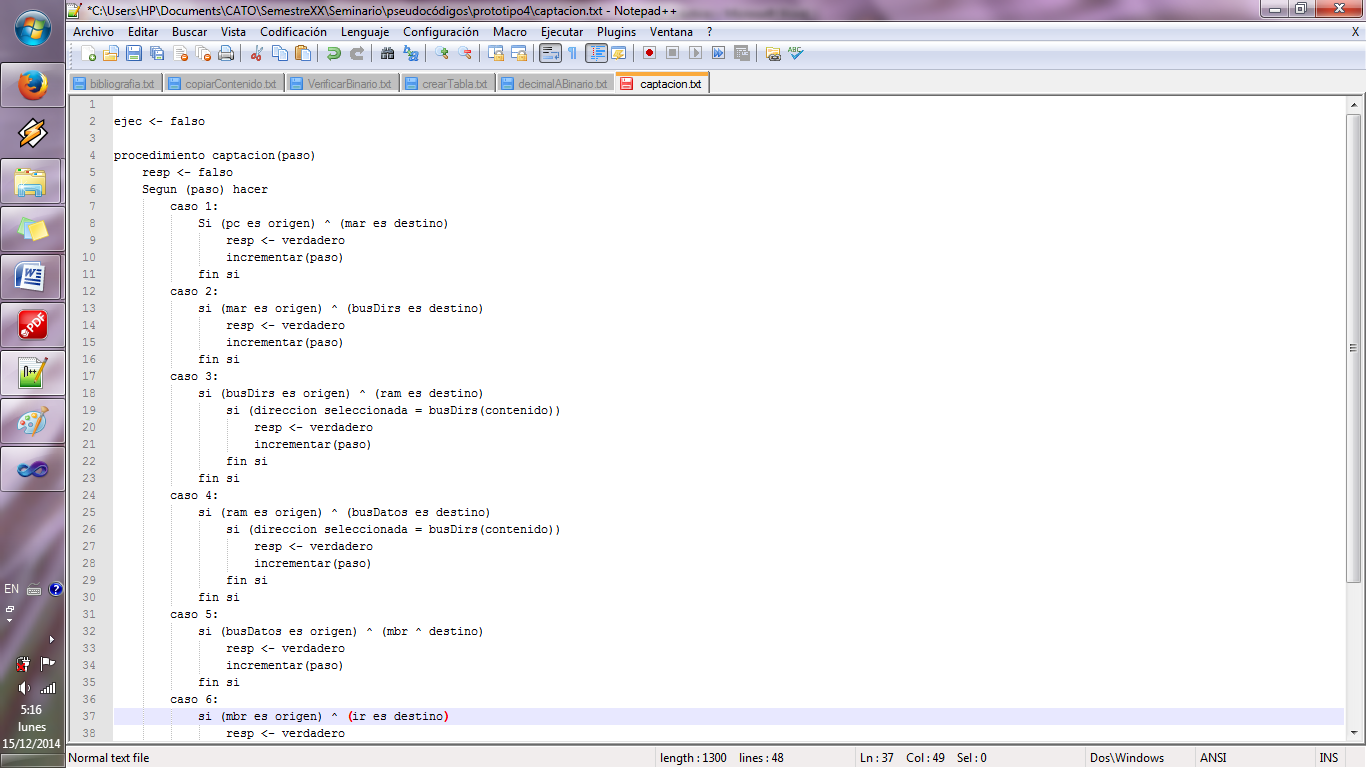
**Figura14. Pseudocódigo Verificar Paso en Captación.**

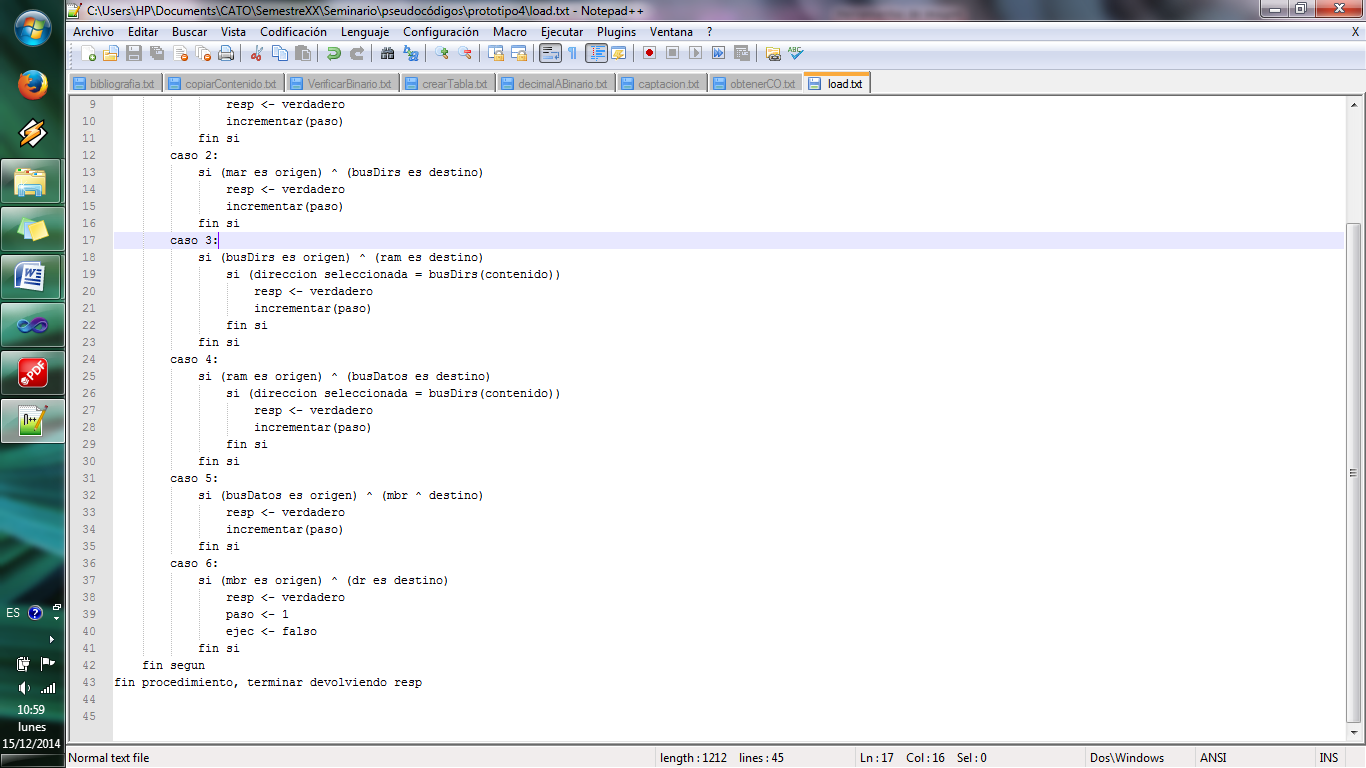




Fuente: Elaboración propia 2014

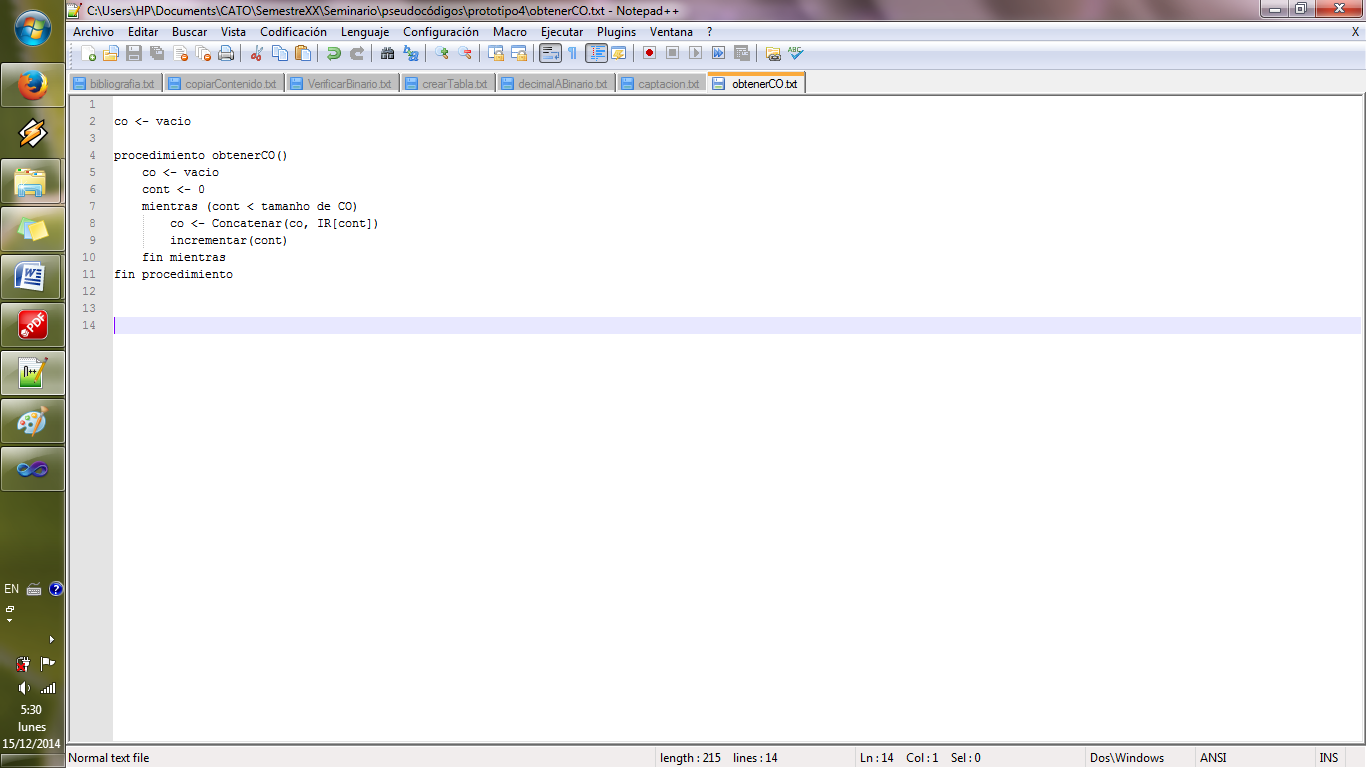
**Figura15. Pseudocódigo Verificar Paso en ejecución con LOAD.**





Fuente: Elaboración propia 2014

**Figura16. Pseudocódigo Obtener CO.**



Fuente: Elaboración propia 2014

### 2.2.3. Prototipo 5

Requerimientos:

* Implementar componente para la definición de formato de instrucción y códigos de operación, en binario.
* Desarrollar componente interactivo de verificación de errores, en binario.

Para que un estudiante resuelva un ejercicio de ciclo de instrucción, antes debe estar correctamente planteado y estructurado. Por tanto cuando el docente plante el ejercicio a los estudiantes, estos abstraen el problema y obtienen los datos necesarios para plasmar el estado inicial del CPU y la RAM en los diagramas. De la misma manera, el estudiante debería plasmar esa abstracción en el simulador para plantear el estado inicial del CPU, definiendo el formato de instrucción y el valor de los códigos de operación a utilizarse en el programa.

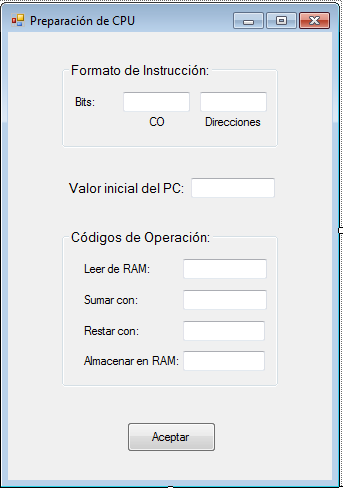
En este prototipo se permitía al usuario elegir una cantidad de bits para el campo de CO y de direcciones, siendo el límite máximo 32 bits en total, por lo que si a uno de los campos se le asignaba un valor igual o mayor a 30 bits, al otro se le asignaba dos automáticamente y se ponía el valor de este campo en 30.

De la misma manera el estudiante debía ingresar el valor inicial en binario del registro PC y de cada CO, pero si el tamaño era menor se rellenaban los bits restantes con ceros.

Al realizar los controles principales por el estudiante de manera automática, lo único que se debía controlar para avanzar era si todos los campos necesarios estaban llenos, caso contrario se mostraba un mensaje para avisar que campo debía ser llenado.

También se modificó el valor esperado de los registros en el componente principal, en el que el estudiante resuelve el ejercicio como tal, pasando los valores definidos en la interfaz de preparación del CPU como parámetros al constructor del componente principal. De esta forma cuando el usuario resolvía ejercicios se cargaba automáticamente el valor inicial de la PC y se modificaron todos los algoritmos relacionados con la longitud del campo de CO y direcciones del CPU, así como del tamaño total de los registros.

**Figura 17. Interfaz prototipo 5, preparación de CPU.**



Fuente: Elaboración propia 2014

### 2.2.4. Prototipo 6

Requerimientos:

* Implementar componente para la definición de formato de instrucción y códigos de operación, en binario.
* Implementar componente para preparación de la RAM para el ejercicio por parte del estudiante (llenado de registros iniciales), en binario.
* Desarrollar componente interactivo de verificación de errores, en binario.

Después de hacer un análisis del tercer prototipo, se llego a la conclusión que los controles de ahí en adelante para el estudiante solo se limitarían a mostrarle donde está su error cuando quisiera seguir pero no a corregir ninguno de ellos por él.

Por lo tanto, se reprogramó todo el componente siguiendo esa lógica, bloqueando los campos inferiores hasta que los superiores cumplan con los requerimientos inferiores, mostrando a medida que avanza el usuario mensajes para que pueda saber que error está cometiendo. El control se hizo mediante el evento “TextChanged”, siendo que para verificar cada “TextBox” al ser modificado se aplica un algoritmo el cual verifica que el valor sea correcto y si se cumplen las condiciones de desbloqueo del siguiente bloque de datos que debe llenar el usuario, bloques que se dividen en:

* Paso1: El usuario debe llenar correctamente el campo de direcciones y de CO.
* Paso2: El estudiante debe dar un valor inicial de PC válido.
* Paso3: El estudiante le asigna un valor a cada uno de los COs que necesita.

Al finalizar el llenado de datos y hacer click en el botón “Aceptar”, el componente verifica que todos los datos estén en orden, en caso que no sea así aparece un mensaje con el error.

Si es que el usuario deja en blanco los códigos de operación para sumar y restar se muestra un mensaje notificando sobre esto y preguntando si desea continuar. Como un ejercicio como mínimo debe tener una lectura de un número desde una dirección de memoria y su escritura en otra dirección, se le permite continuar en caso de que así lo desee.

Una modificación importante de este prototipo es que la longitud total máxima de un registro del CPU, la cual solo llega a 16 bits, esto se debe a que un tamaño mayor sería innecesario para fines didácticos.

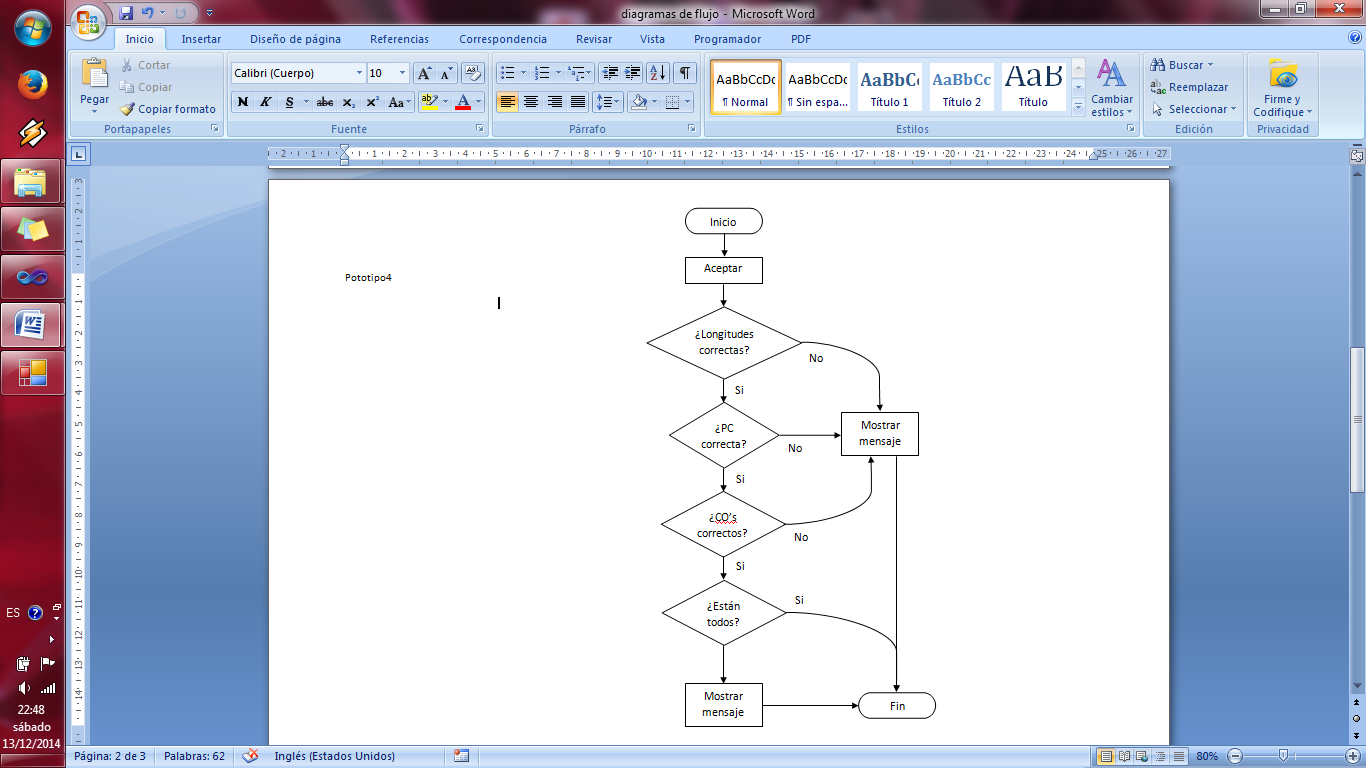
Siguiendo también fines didácticos, se define que la longitud del campo de CO y de direcciones sea un múltiplo de 4, para poder hacer una comparación de los ejercicios a realizarse en clases en binario con los que se realizaran en hexadecimal.

También se implemento una nueva interfaz de usuario en la cual el usuario define el estado inicial de los registros de la RAM, en la cual se verifica solamente la longitud de los campos y que solamente ingresen los caracteres “0” y “1”.

Al tener previamente el estado inicial de la RAM definido por el usuario, la interfaz del componente principal no solo aparece automáticamente con el contenido del registro PC en el valor inicial definido por el usuario, sino también la memoria RAM aparece con los datos definidos por el mismo.

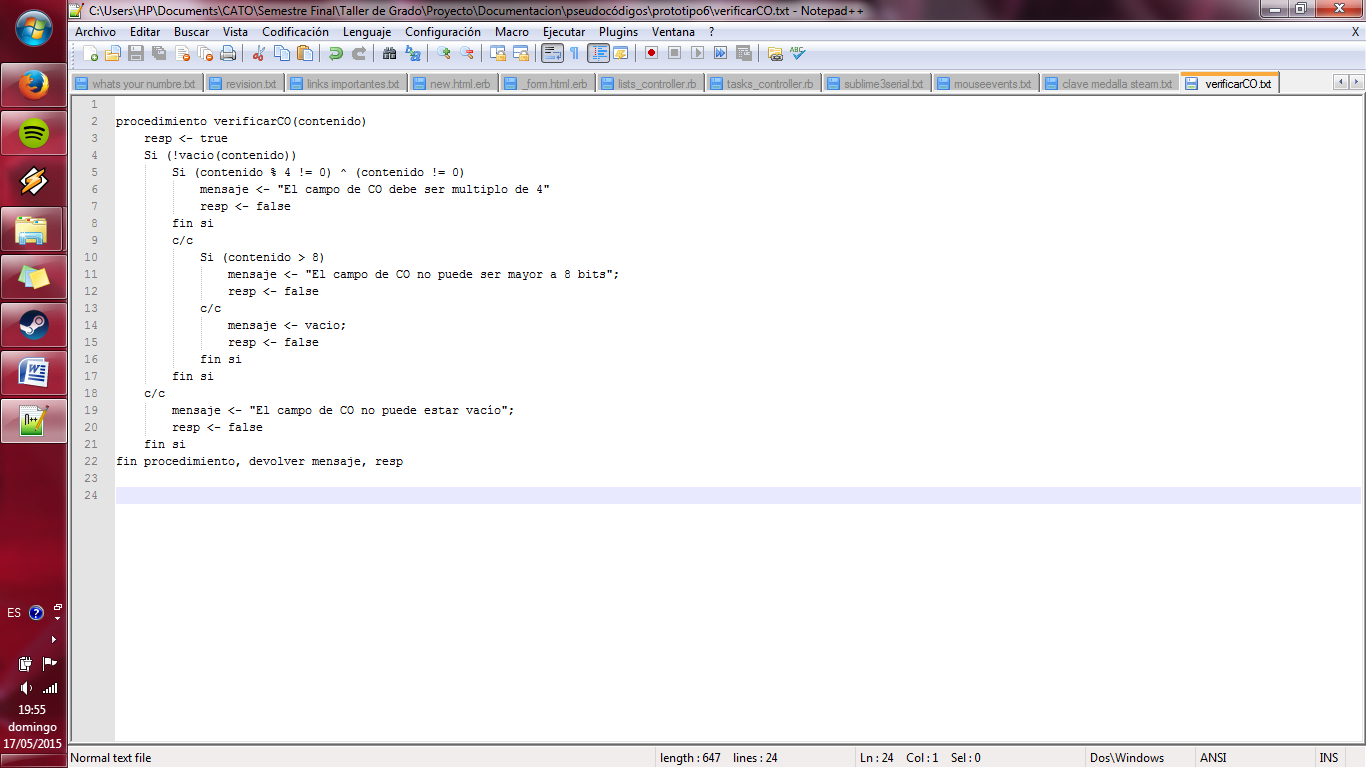
**Figura 18.**

**Diagrama de flujo verificación final, interfaz de Preparación de CPU.**



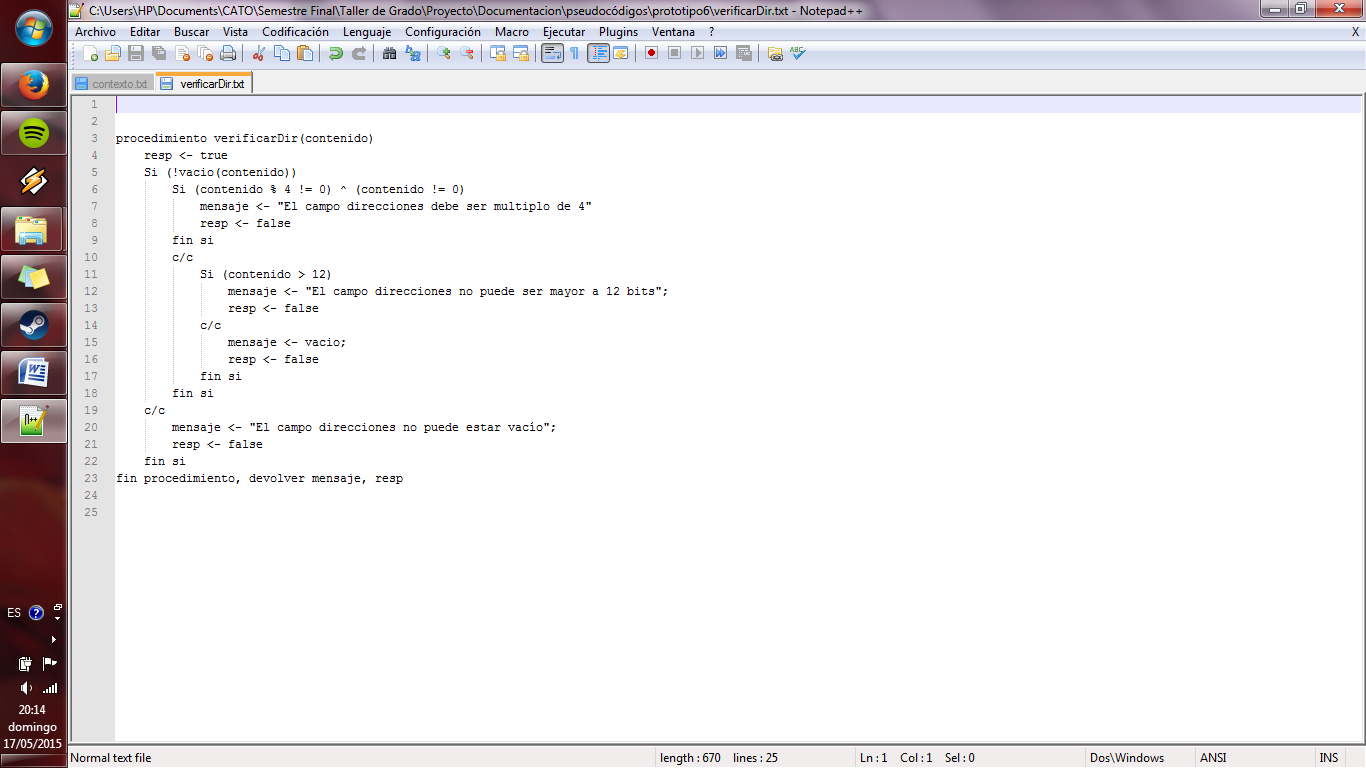
Fuente: Elaboración propia 2014

**Figura 19. Pseudocódigo verificación de campo CO.**



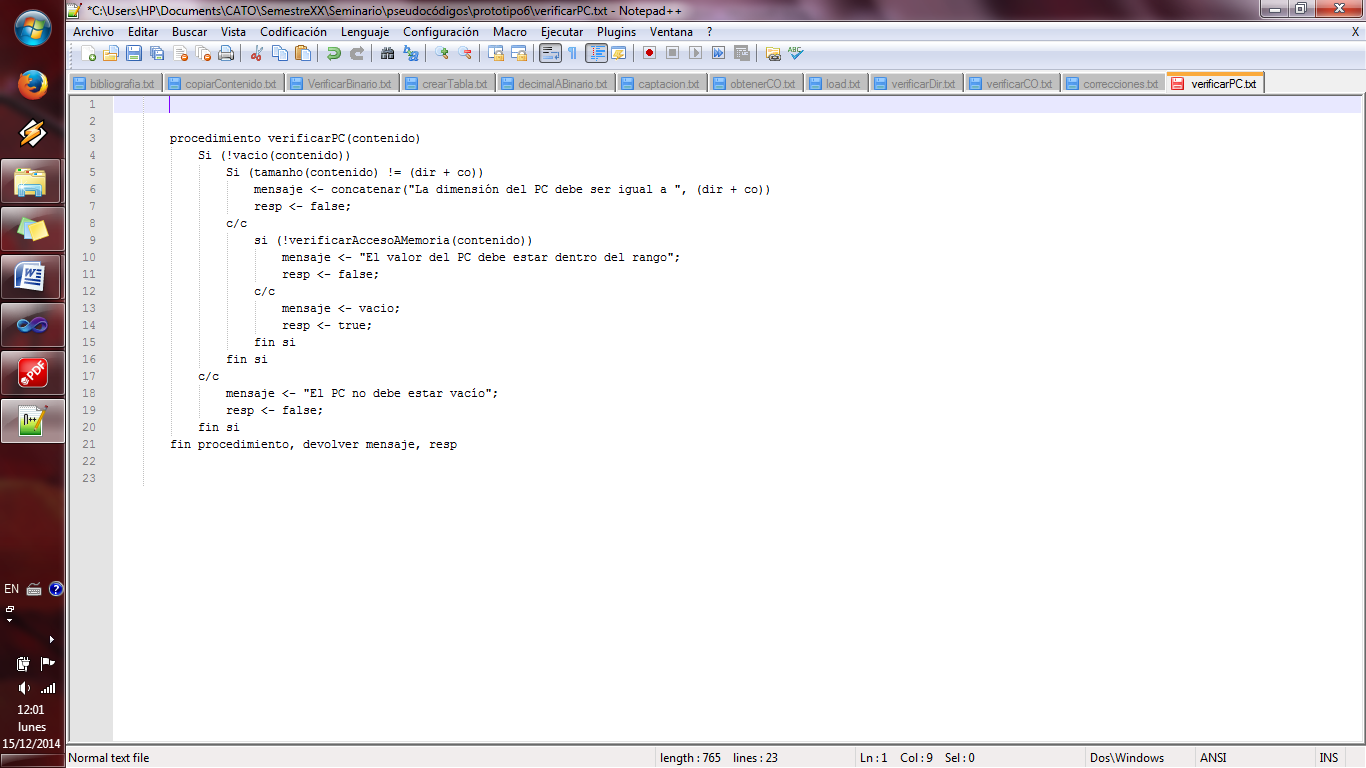
Fuente: Elaboración propia 2014

**Figura 20. Pseudocódigo verificación de campo de Dir.**



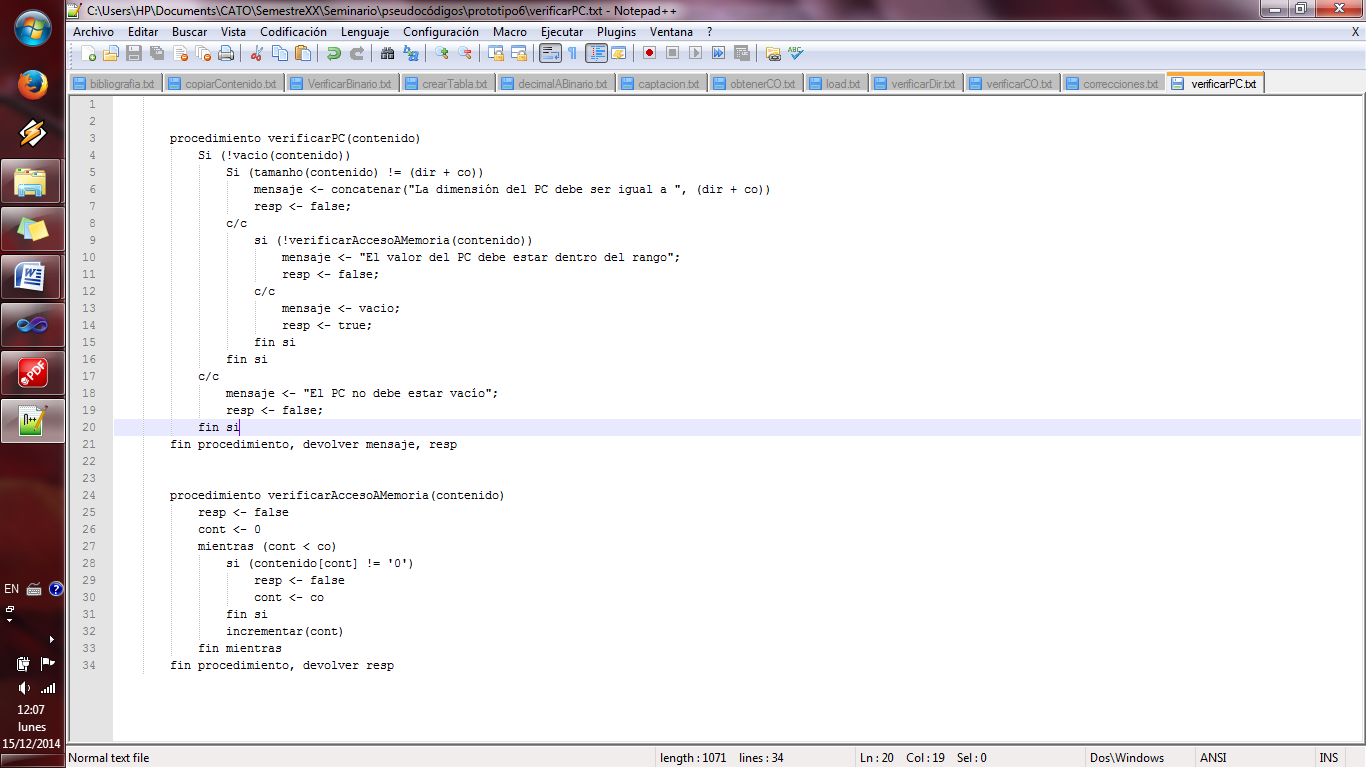
Fuente: Elaboración propia 2014

**Figura 21. Pseudocódigo verificación de PC inicial.**



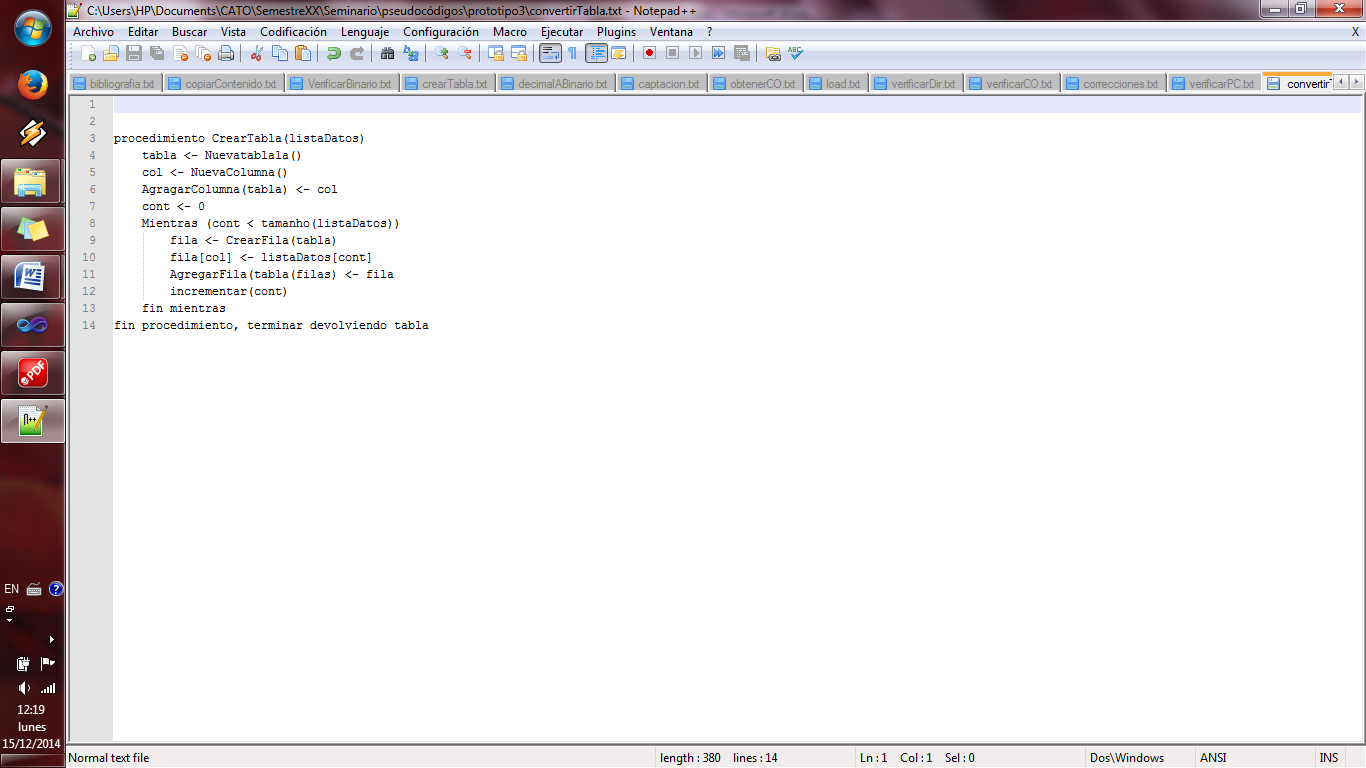
Fuente: Elaboración propia 2014

**Figura 23. Pseudocódigo verificación de acceso a memoria.**



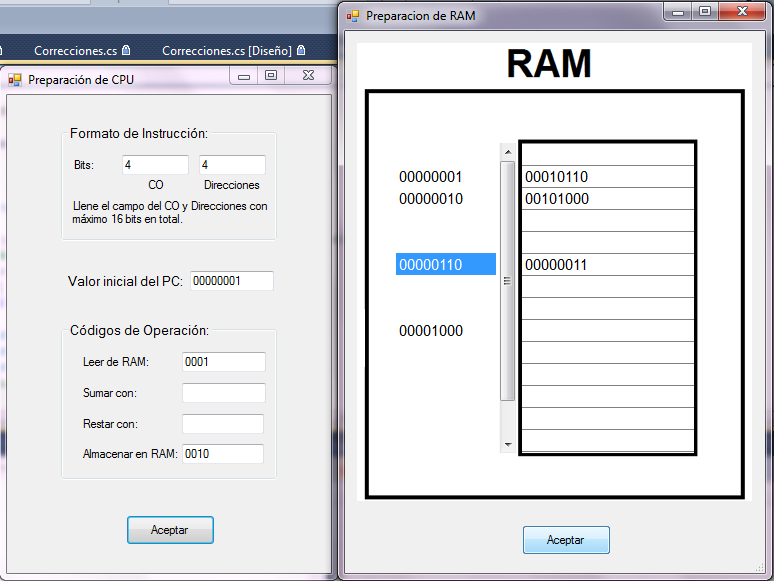
Fuente: Elaboración propia 2014

**Figura 23. Pseudocódigo para llenar tabla enviada en interfaz principal.**



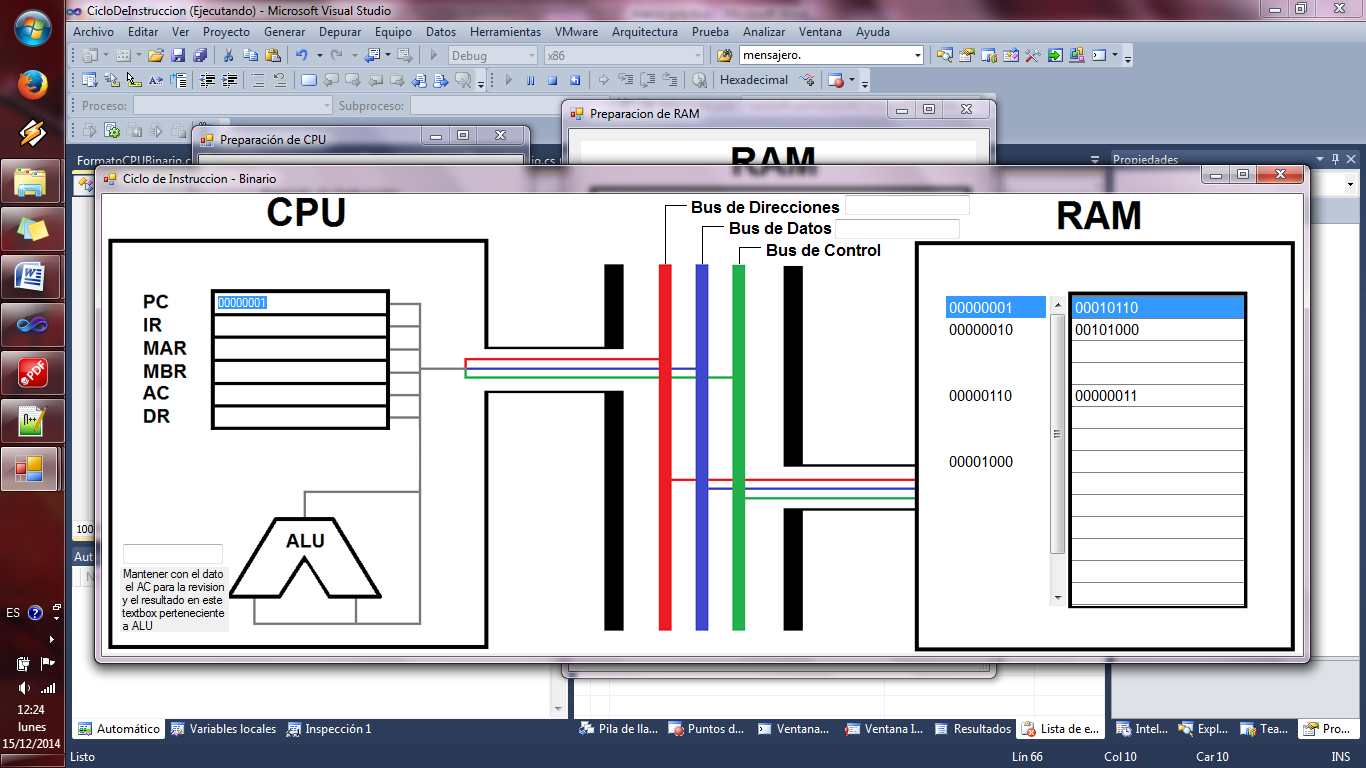
Fuente: Elaboración propia 2014

**Figura 24. Interfaces de preparación de CPU y RAM.**



Fuente: Elaboración propia 2014

**Figura 25. Interfaz principal con parámetros enviados.**



Fuente: Elaboración propia 2014

### 2.2.5. Prototipo 7

Requerimientos:

* Desarrollar componente de interacción con el estudiante, para desplazar el contenido de registros.
* Desarrollar componente interactivo de verificación de errores, en binario.
* Refactorización de código.

En este prototipo se busco realizar una afinación del código ya funcional conseguido en el desarrollo de los 4 prototipos iniciales.

En el componente principal verifica si el estudiante realiza el incremento del PC correctamente.

Primeramente se procedió a crear clases necesarias para generar un código más mantenible y menos acoplado.

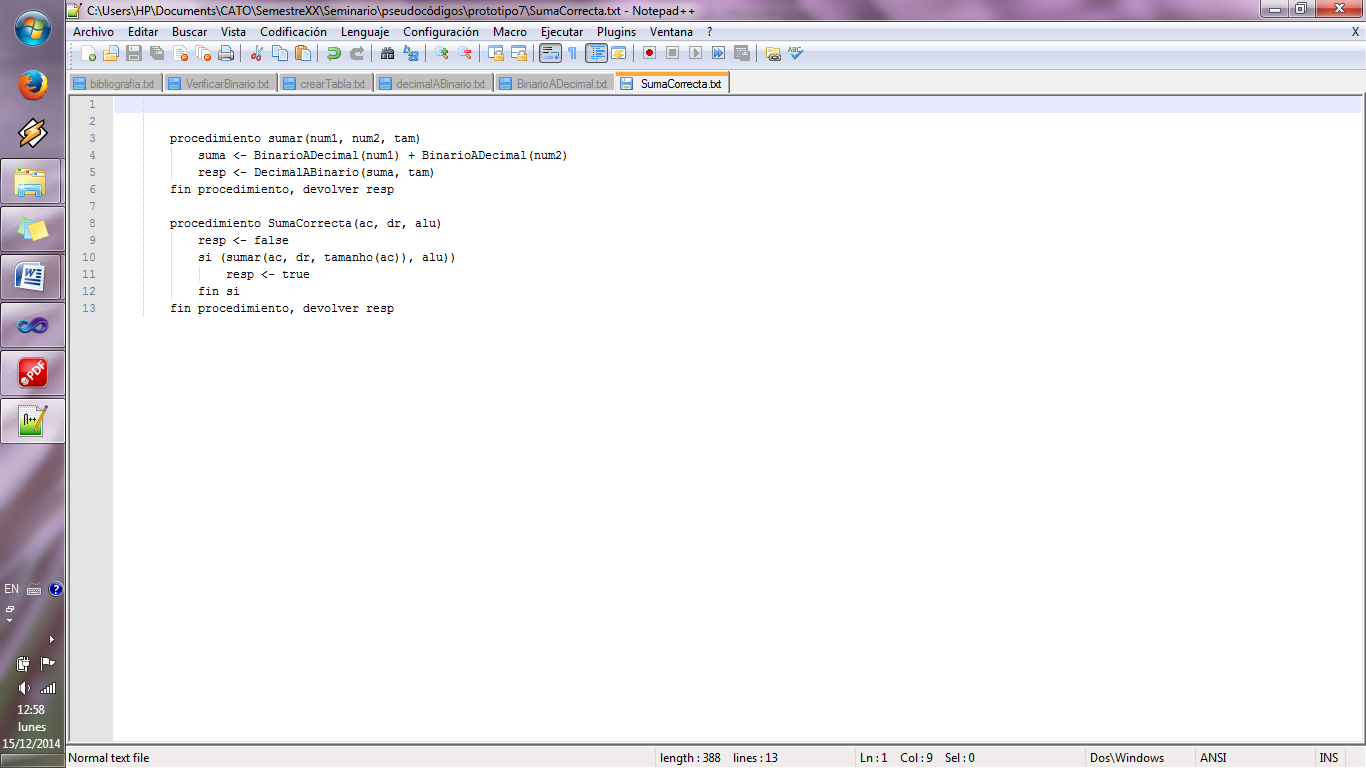
Creando objetos tales como la clase Binario, la cual es una clase estática la cual contiene todos los métodos necesarios para manipular datos en binario, o algunos otros controladores, los cuales principalmente se encargan de la lógica de control que puede ser separada de los componentes de interfaz de usuario.

**Figura 26. Pseudocódigo para conversión de Binario a Decimal**



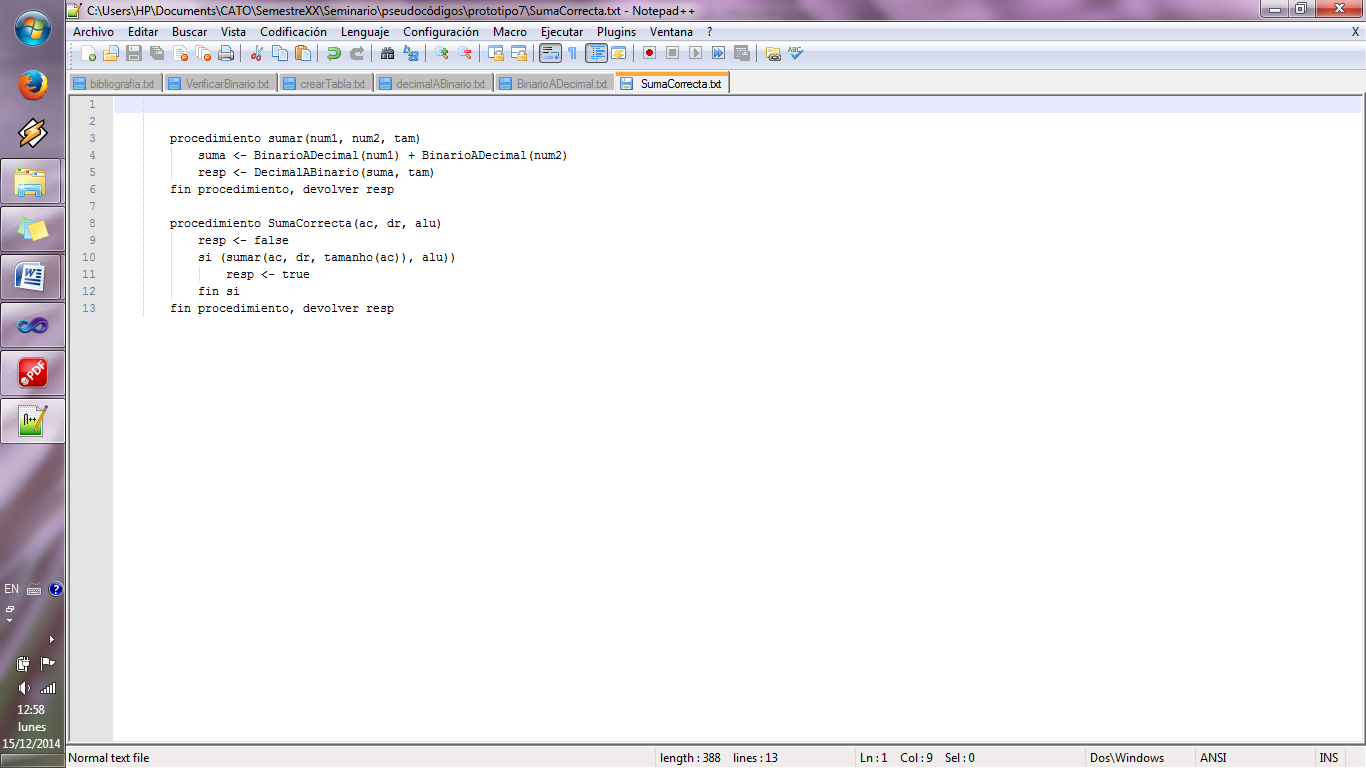
Fuente: Elaboración propia 2014

**Figura 27. Pseudocódigo para sumar dos números binarios**



Fuente: Elaboración propia 2014

**Figura 28. Pseudocódigo para verificar suma**



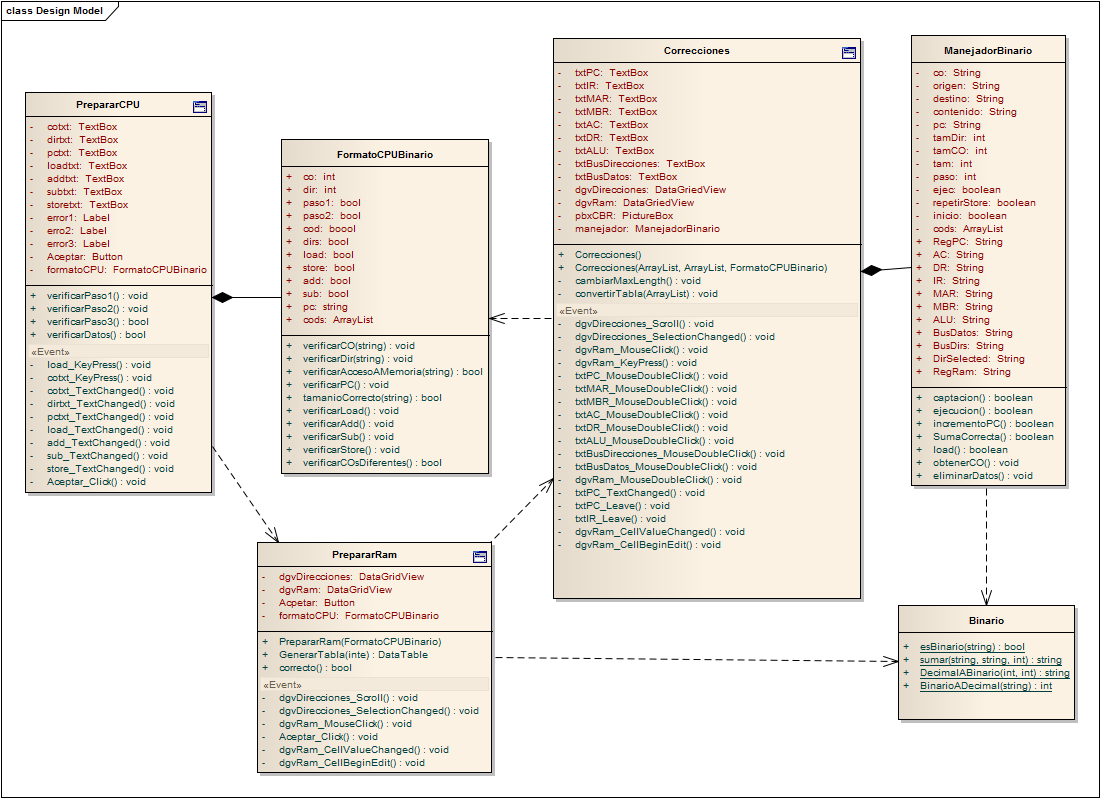
Fuente: Elaboración propia 2014

**Figura 29. Pseudocódigo para incremento de PC**



Fuente: Elaboración propia 2014

**Figura 30. Diagrama de clases.**



Fuente: Elaboración propia 2014

### 2.2.6. Prototipo 8

Requerimientos:

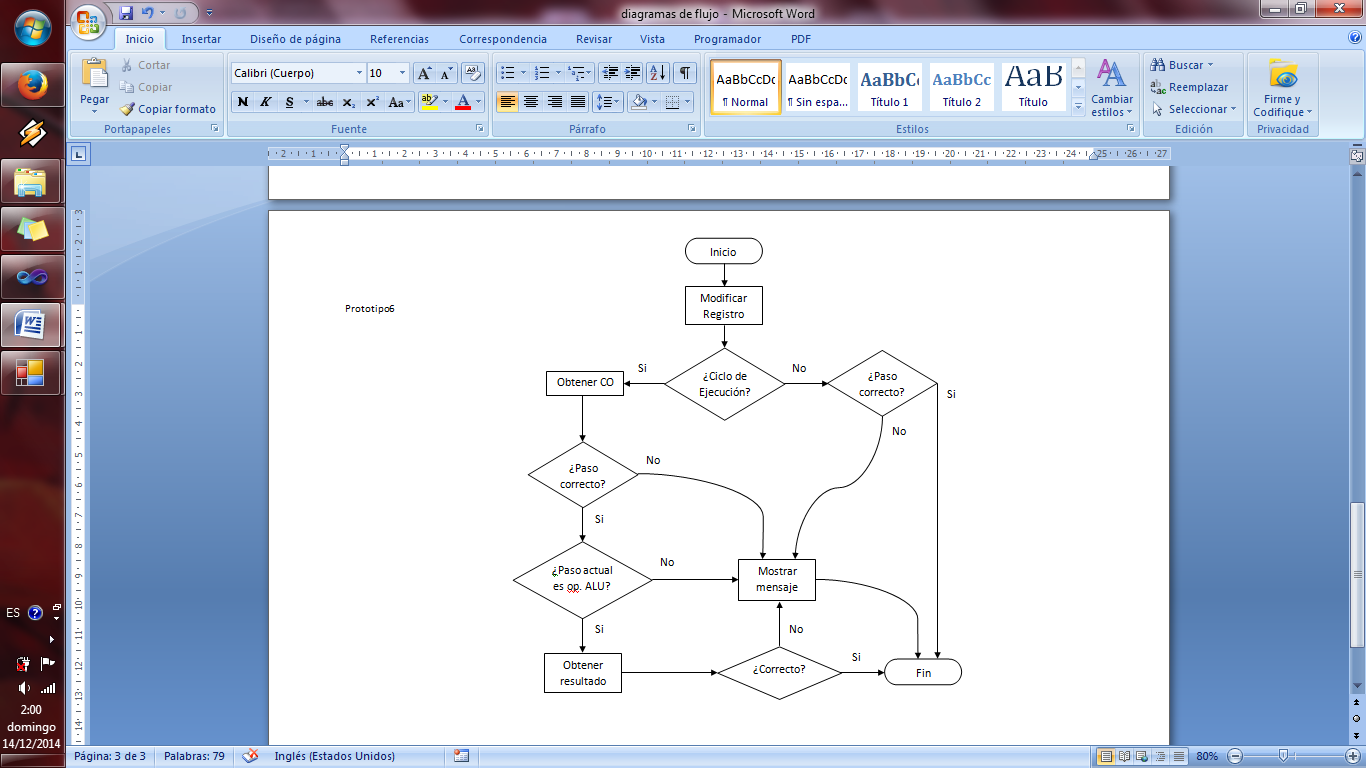
* Desarrollar componente de interacción con el estudiante, para desplazar el contenido de registros.
* Desarrollar componente interactivo de verificación de errores, en binario.

En el componente principal se implementó la verificación del ciclo de ejecución del CO para almacenar el contenido de AC en la RAM.

También se implementó la verificación del ciclo de ejecución para las dos operaciones de ALU, es decir, suma y resta de dos números. Para esta parte se modificó el componente de interacción con el estudiante para manejar el contenido del registro de ALU, el cual no fue implementado anteriormente.

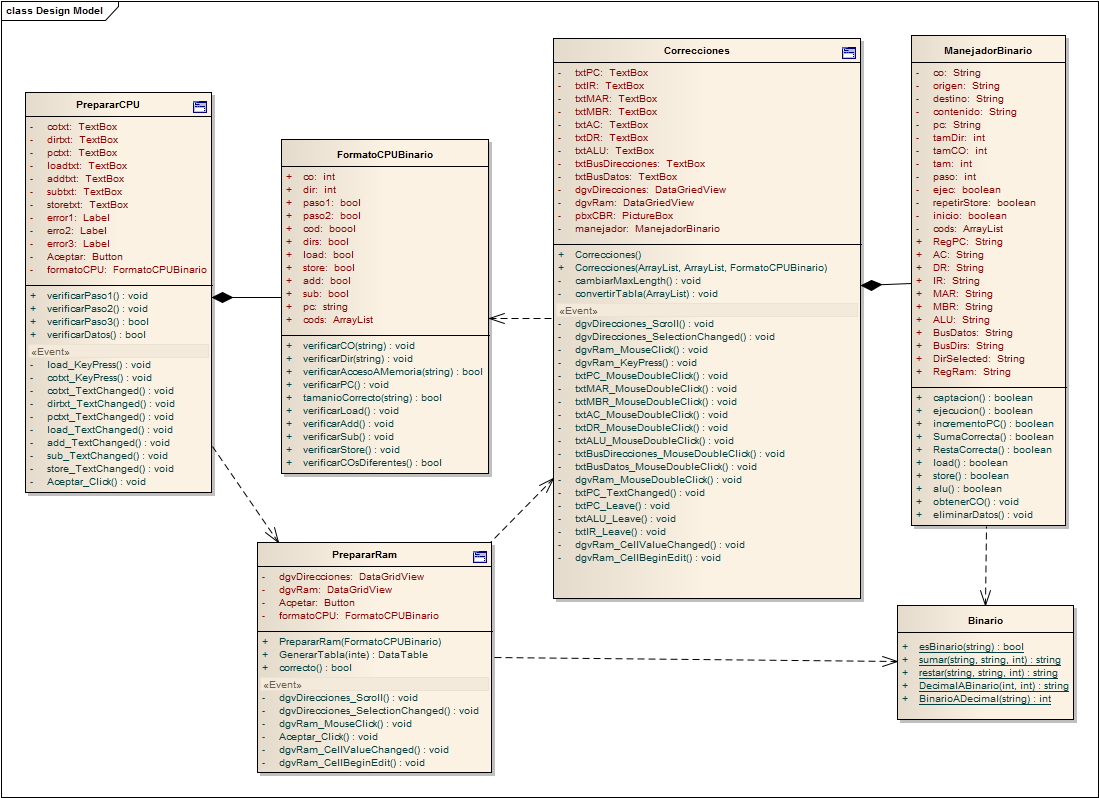
Para verificar si las operaciones de ALU son o no correctas, se crearon varias funciones, las cuales verifican primeramente el valor de la suma o resta de los contenidos de los registros AC y DR, y la comparan con el valor ingresado por el usuario.

**Figura 31. Diagrama de flujo de verificación de pasos.**



Fuente: Elaboración propia 2014

**Figura 32. Diagrama de clases.**



Fuente: Elaboración propia 2014

### 2.2.7. Prototipo 9

Requerimientos:

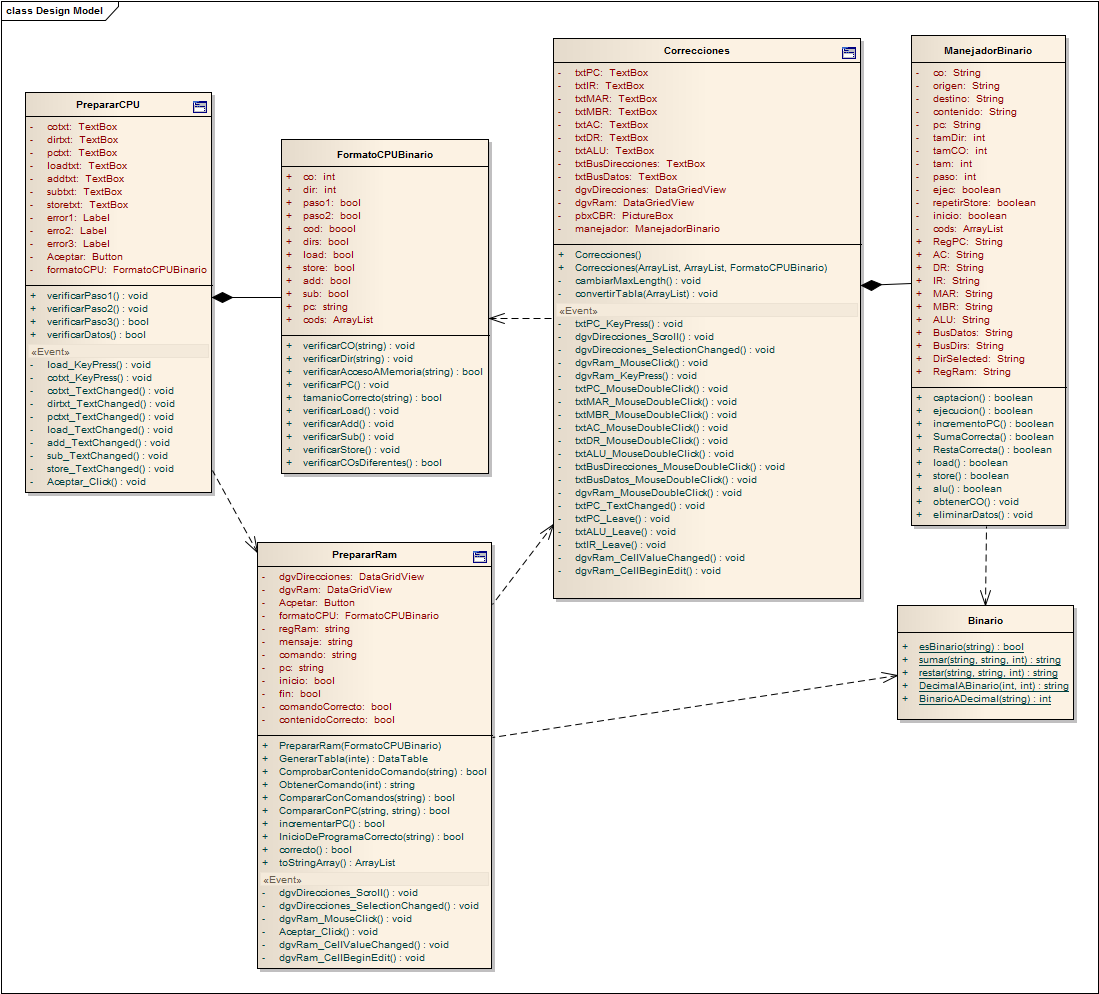
* Implementar componente para preparación de la RAM para el ejercicio por parte del estudiante (llenado de registros iniciales), en binario.

En este prototipo se mejora la funcionalidad del componente de preparación de la RAM, buscando dejar un contenido de RAM más consistente. Se realizan principalmente las siguientes acciones de verificación:

* La dirección a la cual apunta el valor inicial del PC debe existir.
* Debe existir al menos una instrucción además de la inicial.
* Se verifica que las instrucciones sean consecutivas.
* El comando de lectura desde RAM debe ser la instrucción de la dirección a la que apunta el valor inicial del PC.
* Comprueba las instrucciones consecutivas en RAM hasta encontrar un registro que no sea una instrucción, es decir, que no contenga un código de operación.
* La dirección a la que apunta el campo de direcciones de cada instrucción debe tener contenido, a menos que la instrucción sea de escritura a la RAM.

Todas las verificaciones anteriormente mencionadas se llevan a cabo al momento que el usuario hace click en el botón “Aceptar”.

**Figura 33. Diagrama de clases.**



Fuente: Elaboración propia 2014

### 2.2.8. Prototipo 10

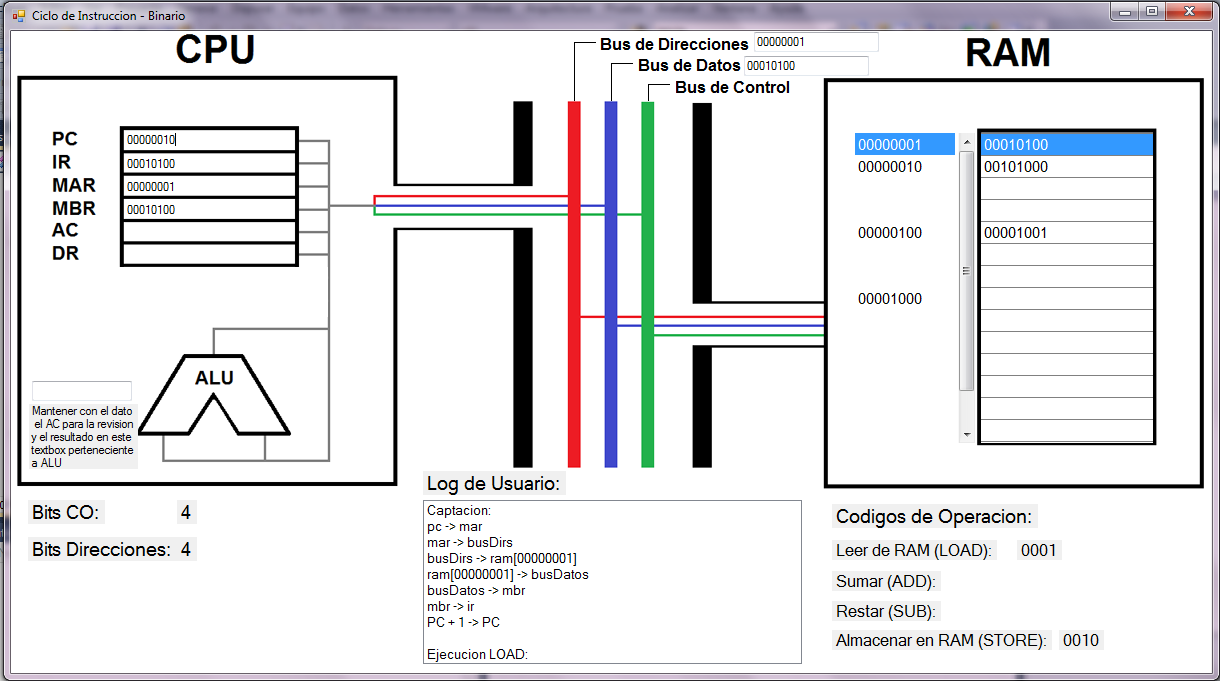
Requerimientos:

* Rediseño de interfaz de usuario.

Se procedió a incrementar un *log* de operaciones de usuario, para que el estudiante pueda visualizar los pasos correctos que realizó durante la resolución del ejercicio.

También se agregan a la interfaz de usuario los datos llenados por el mismo en la interfaz de preparación del CPU.

**Figura 34. Interfaz principal con *log* de operaciones.**



Fuente: Elaboración propia 2014

### 2.2.9. Prototipo 11

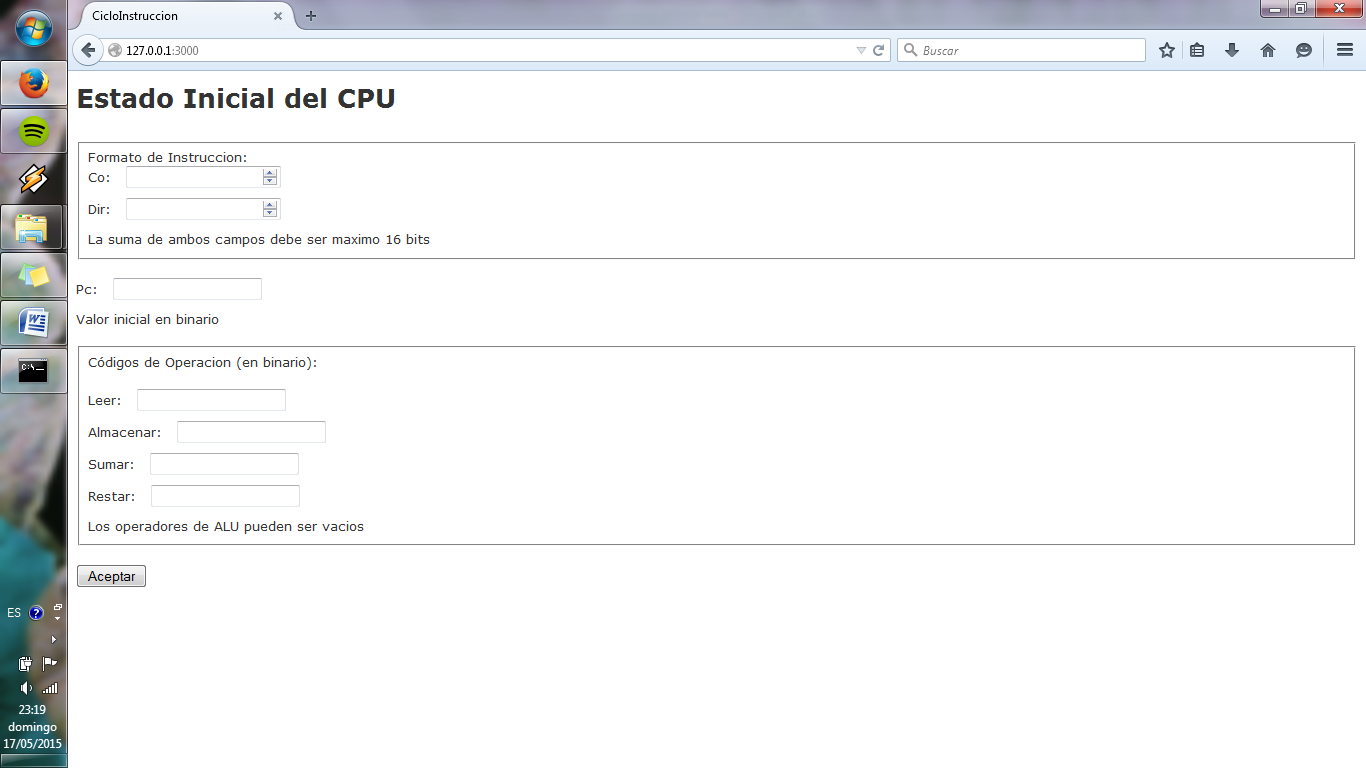
Requerimientos:

* Implementar componente para la definición de formato de instrucción y códigos de operación, en binario.
* Herramienta web.

Al definirse que la herramienta debía estar disponible para cualquier persona independientemente de su locación geográfica o preferencia de sistema operativo, se definió que era necesario que la herramienta sea migrada a un entorno web para que así sea accesible para cualquier persona que desee utilizarla, teniendo como única restricción el acceso a internet.

Primeramente se migró la verificación de la preparación del CPU, realizando el control de ingreso de datos desde el lado del servidor solamente, aún sin CSS.

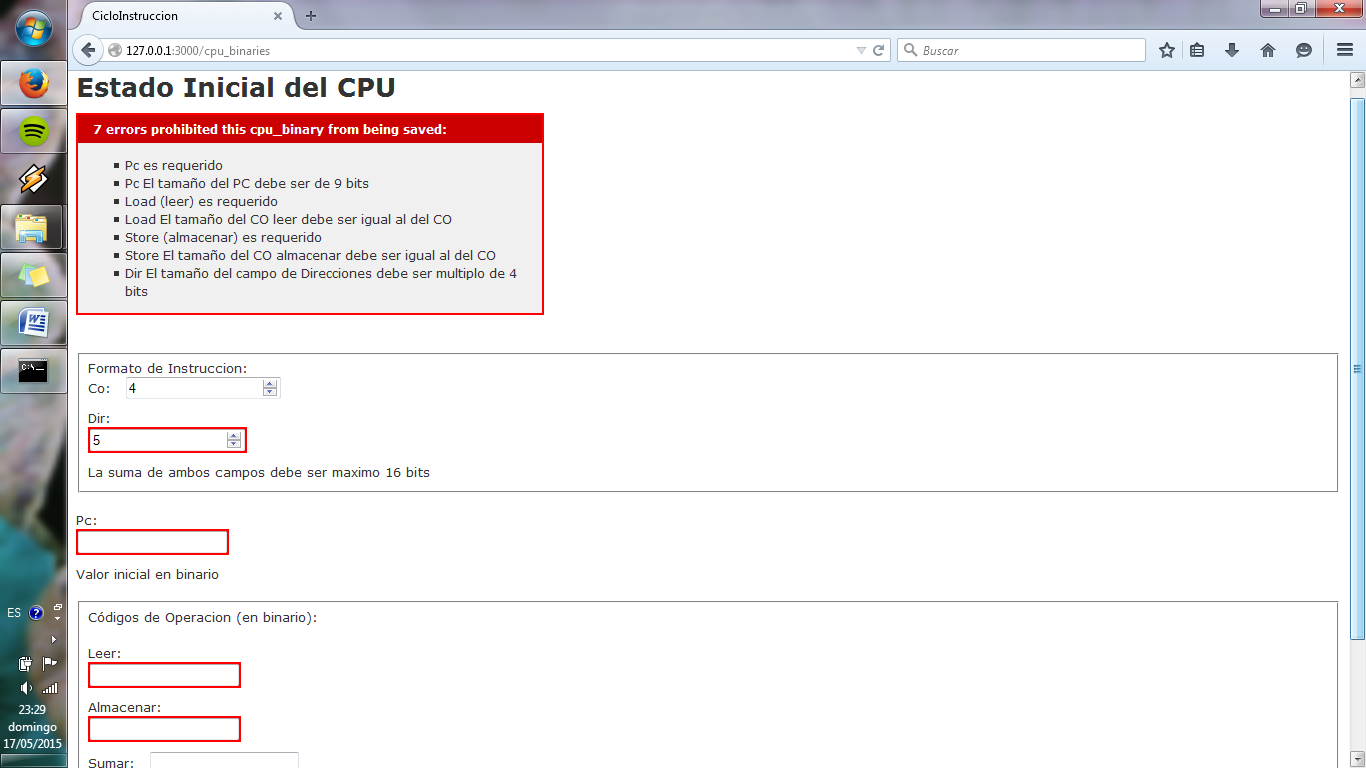
**Figura 35. Interfaz de preparación del CPU en binario.**



Fuente: Elaboración propia 2014

**Figura 36. Control en preparación del CPU en binario,**

**despliegue de errores.**



Fuente: Elaboración propia 2014

Para realizar esta validación de datos de entrada ingresados por parte del usuario se utilizaron los métodos *valídate*, propiosde los modelos en *Ruby on Rails*. En estas validaciones se verifica que el contenido de los campos son los correctos en cuanto a longitud y contenido, es decir, los datos q deben ser binarios sean binarios y los que sean decimales sean decimales, así como también que la longitud del registro *PC* y de los códigos de operación coincidan con los datos ingresados en el formato de instrucción definido por el usuario.

### 2.2.10. Prototipo 12

Requerimientos:

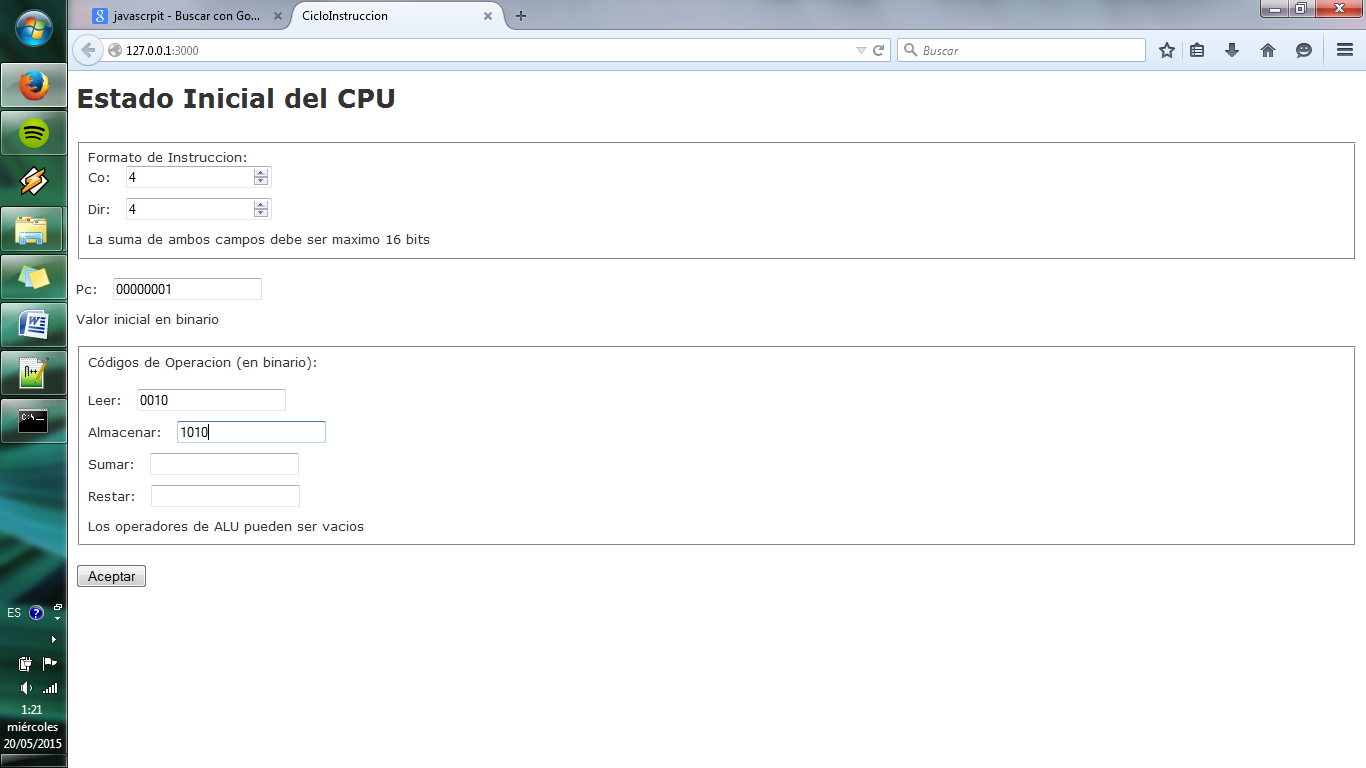
* Implementar componente para la definición de formato de instrucción y códigos de operación, en binario.
* Implementar componente para preparación de la RAM para el ejercicio por parte del estudiante (llenado de registros iniciales), en binario.
* Herramienta web.

Posteriormente se aplicó *JavaScripts* para la validación de datos en la interfaz de preparación del CPU desde el lado del cliente.

Se utilizo para la validación anteriormente nombrada un *plugin* de *JQuery* denominado *regex-mask-plugin* el cual se encarga de restringir el ingreso de valores a un *input text* mediante al función *keypress*, permitiendo solamente el ingreso de los caracteres que estén dentro de una expresión regular.

**Figura 37. Interfaz de preparación del CPU en binario con regex-mask.**

**Prototipo 12.**



Fuente: Elaboración propia 2014

En cuanto a la preparación del estado inicial de la RAM y su verificación, al cambiar de *.NET* a *Ruby* se cambia también el modelo de objetos y se agrega una base de datos, puesto que la información de cada objeto solo puede ser recuperada si fue previamente guardada en la base de datos al cambiar de vista. De esta forma el nuevo diagrama de clases y el de base de datos es el mismo por convención en *Ruby on Rails* debido al uso de *Active Record*:

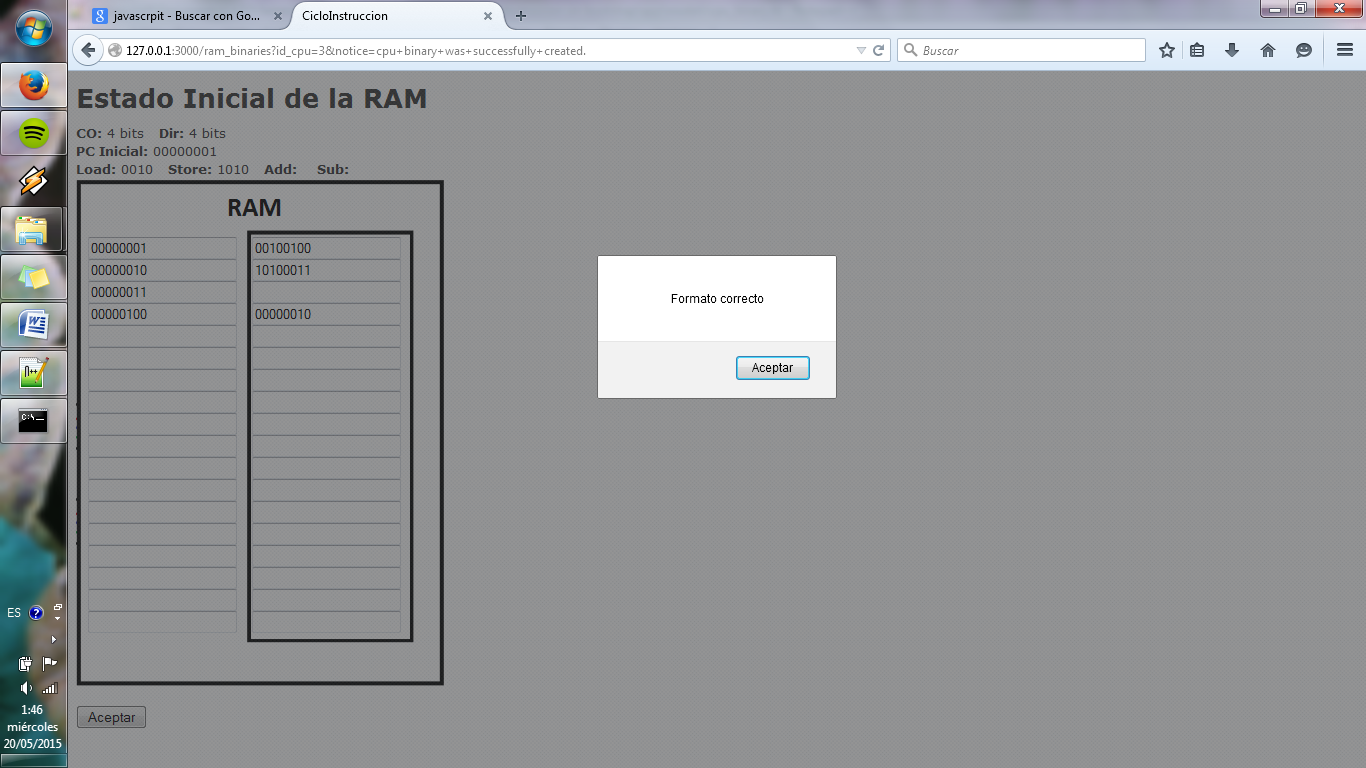
**Figura 38. Diagrama de Base de Datos, prototipo12.**

DIAGRAMA DE CLASES

Fuente: Elaboración propia 2014

**Figura 39. Interfaz de preparación de la RAM en binario.**

**Prototipo 12.**



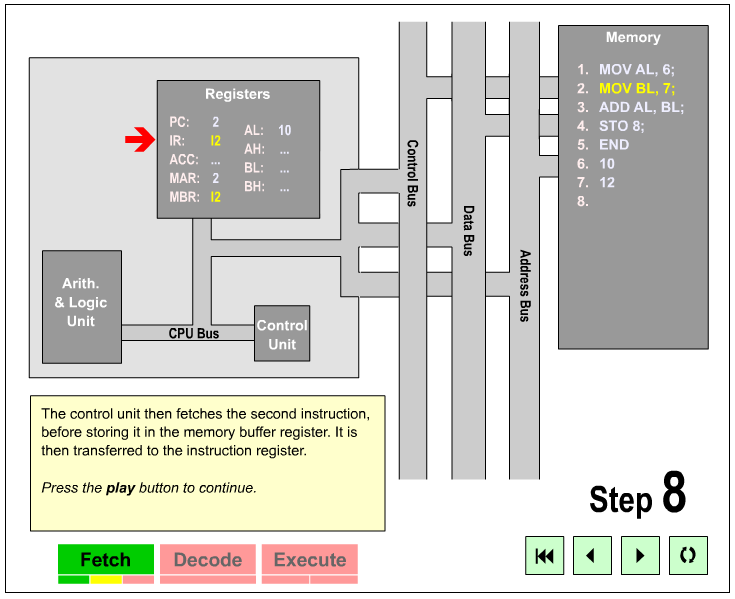
Fuente: Elaboración propia 2014

# Anexos

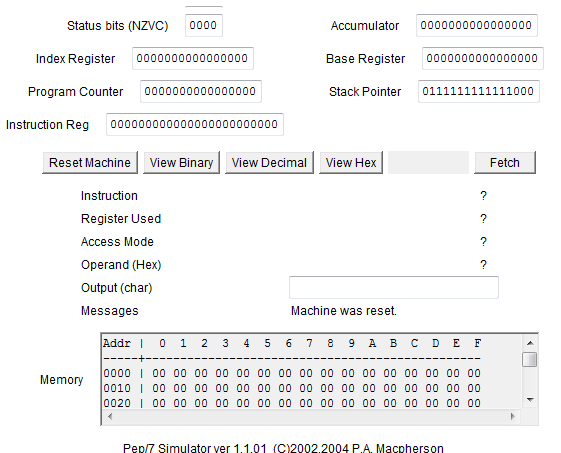
## Anexo 1

Screenshots de las herramientas analizadas, encontradas en los tres enlaces de bibliografía.

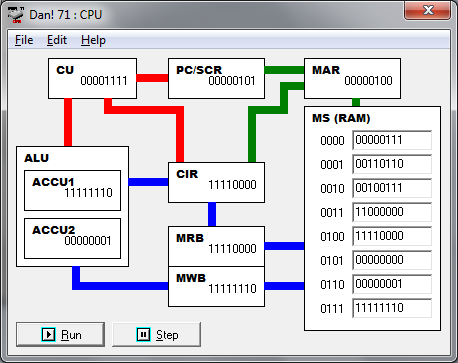
**Microprocessor tutorial**



**pep/7 simulator**



**dan!71 cpu simulator**



## Anexo 2

Se anexa el cuestionario utilizado para una entrevista oral con alumnos de pasados semestres de la materia de Arquitectura de Computadoras en la UCB.

**Cuestionario**

1. ¿Para usted son claros los diagramas utilizados en clases para la enseñanza del ciclo de instrucción?

Enfatizaron que si al momento de la explicación, pero que si uno se pierde la clase no entienden bien como funciona ni con apuntes de un compañero.

1. ¿Cree usted que el tiempo empleado en realizar los diagramas es demasiado?

La respuesta fue un si casi unánime.

1. En su opinión, ¿el tiempo empleado en la preparación de los diagramas desvía la atención del tema principal que es el ciclo de instrucción?

No por completo, pero si recibe más atención de la que debería por el hecho de ser tantos.

1. ¿Comprende de forma rápida el ciclo de instrucción en sus distintos formatos? ¿Cuáles le causaron más problemas?

Los que asistieron a todas las clases si, aunque tuvieron problemas al adaptarse cada que se utilizaba un formato un poco diferente.

1. ¿Cree que sería de ayuda la aplicación de una herramienta de software para el proceso de enseñanza-aprendizaje?

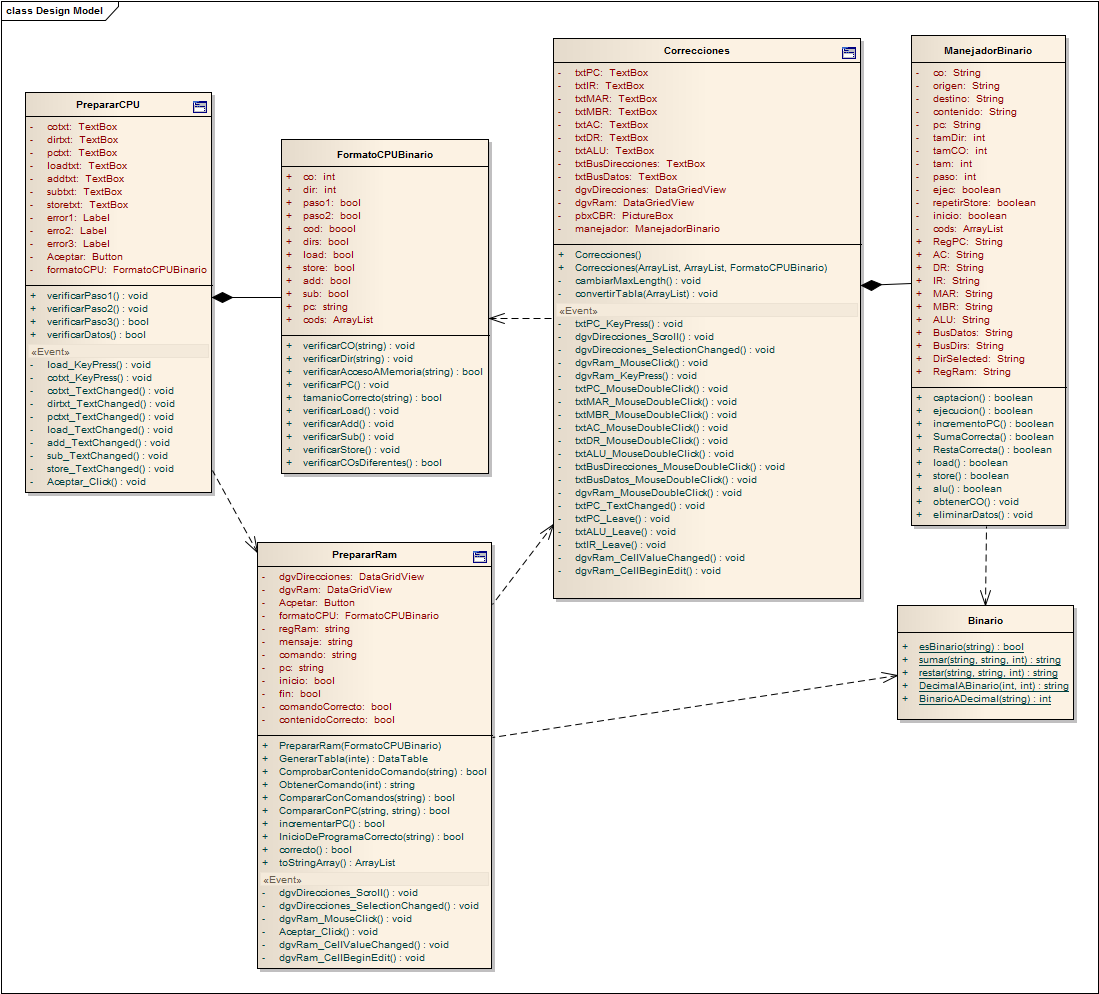
Si, haría el trabajo menos monótono y más autodidacta.

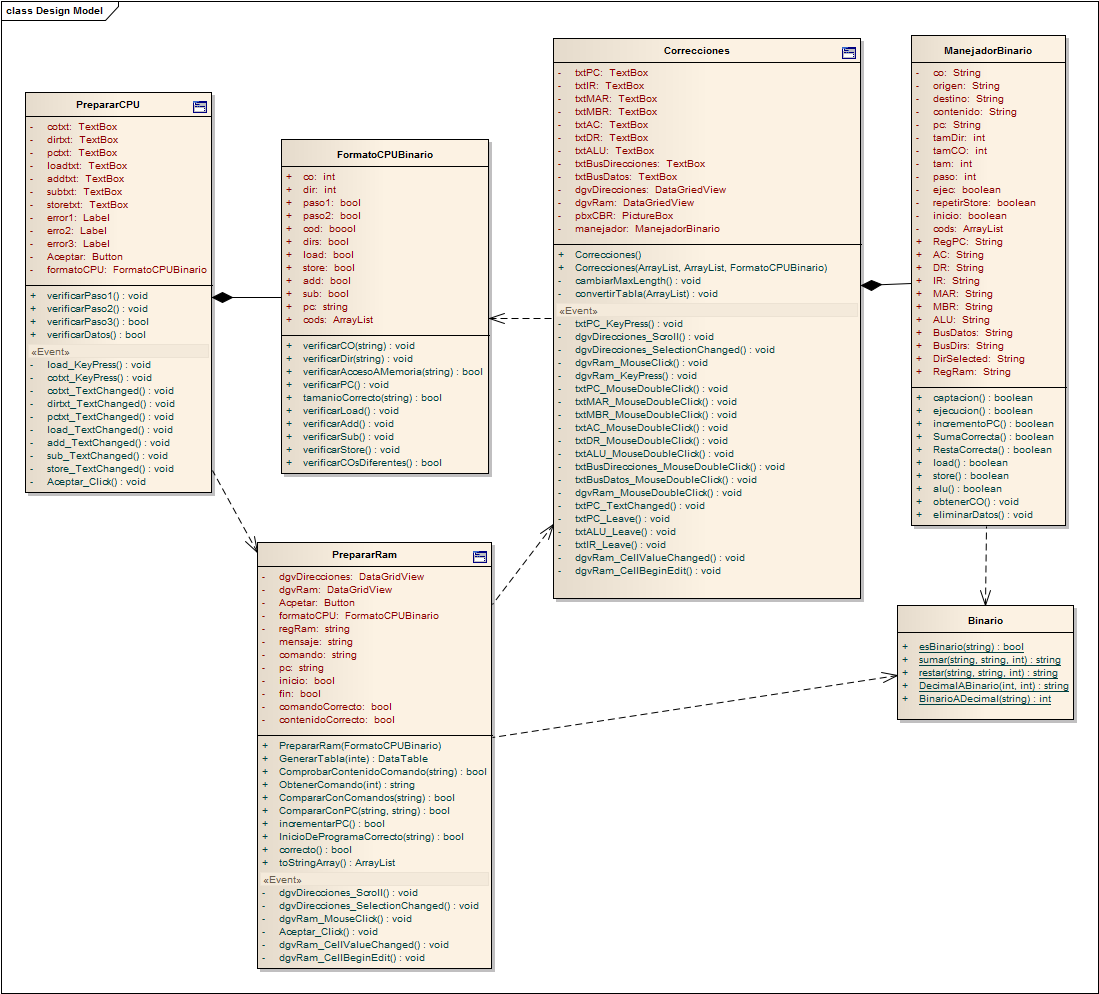
1. ¿Qué sugerencias tiene respecto a la funcionalidad de una herramienta orientada a este tema?

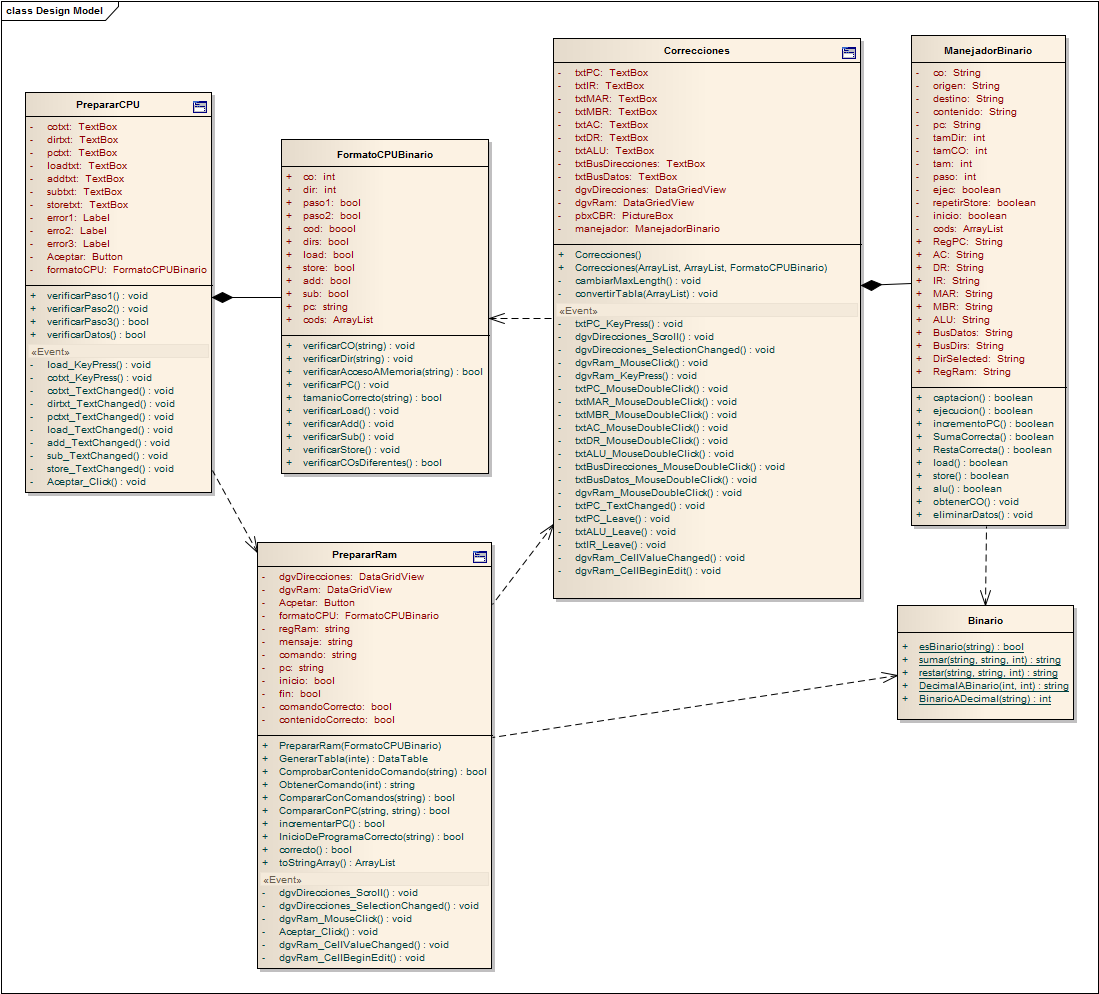
Desarrollar una herramienta que sea fácil de usar, simple y efectiva.

## Anexo 3

El diagrama de clases en el prototipo 9 llega a ser tan grande que es difícil distinguir a detalle los procesos y atributos de cada clase si no se hace un acercamiento a cada una de ellas, por tanto en este anexo el diagrama será fraccionado para su mejor visualización.

****

****

****