**UNIVERSIDAD NACIONAL DE INGENIERÍA**

**FACULTAD DE ELECTROTECNIA Y COMPUTACIÓN**



Trabajo monográfico para la optar al

Título en Ingeniería en Computación

**Herramienta de generación de código para definición y gestión de lenguaje ensamblador como recurso didáctico para la asignatura de Máquinas Computadoras II de la Universidad Nacional de Ingeniería (UNI)**

**AUTOR:**

**Br. Harvin Manuel Toledo Polanco**

**Br. Jorge Manuel Potosme Álvarez**

**TUTOR:**

**Ing. José Díaz Chow**

**Managua, Nicaragua**

**2016**

Contenido

[Dedicatoria 3](#_Toc445560643)

[Agradecimientos 4](#_Toc445560644)

[Resumen 5](#_Toc445560645)

[1- Introducción 6](#_Toc445560646)

[Antecedentes 7](#_Toc445560647)

[Metodología de desarrollo 7](#_Toc445560648)

[Objetivos 8](#_Toc445560649)

[General 8](#_Toc445560650)

[Especifico 8](#_Toc445560651)

[Justificación 9](#_Toc445560652)

[Marco Teórico 10](#_Toc445560653)

[Herramientas de desarrollo 10](#_Toc445560654)

[QT Creator 10](#_Toc445560655)

[C++ 11](#_Toc445560656)

[QT 12](#_Toc445560657)

[Visual Studio 13](#_Toc445560658)

[Análisis y Presentación de Resultados 14](#_Toc445560659)

[Bibliografía 15](#_Toc445560660)

[Anexos 15](#_Toc445560661)

[Introducción al lenguaje ensamblador 15](#_Toc445560662)

[Lenguaje de Bajo Nivel 15](#_Toc445560663)

[Ventajas y desventajas de los lenguajes ensambladores 15](#_Toc445560664)

[El procesador 17](#_Toc445560665)

[Unidades Funcionales Básicas 17](#_Toc445560666)

[Unidad Aritmética Lógica (ALU) 17](#_Toc445560667)

[Unidad de Control (CU) 17](#_Toc445560668)

[Plataforma de desarrollo 21](#_Toc445560669)

# Dedicatoria

# Agradecimientos

# Resumen

# Introducción

El presente documento constituye la propuesta de trabajo de monografía (protocolo) para la que se solicita la correspondiente autorización de las autoridades de la Facultad de Electrotecnia y Computación de la UNI con el fin de iniciar las actividades inherentes al correspondiente estudio, creación y prueba del producto esperado del trabajo monográfico que está orientado a diseñar y producir una herramienta que sirva como recurso didáctico digitalizado para la asignatura Arquitectura de Máquinas Computadoras II contemplada en el currículum de la carrera de Ingeniería en Computación.

## Antecedentes

La computadora es el invento más versátil del hombre en la actualidad. Ningún otro dispositivo creado por la humanidad ha logrado multiplicar su desempeño y campo de aplicación tan rápidamente como la computadora (Patterson & Hennessy, 2013). En sus apenas setenta años de existencia, ha revolucionado la forma en que hacemos las cosas, desde tareas triviales como escuchar música o ver fotografías, hasta los cálculos más complejos que nos permiten ir al espacio, o descifrar el código genético humano.

Este rápido avance en el desarrollo de las computadoras subyace en primera instancia en su arquitectura. Es por eso que los ingenieros en computación deben tener una sólida formación en la disciplina de arquitectura de computadoras, que les permita comprender y manejar los elementos inherentes al funcionamiento de estas máquinas. El plan de estudio de ingeniería en computación en la Universidad Nacional de Ingeniería (UNI) de Nicaragua, contempla tres asignaturas de Arquitectura de Máquinas Computadoras: I, II y III. En línea con lo planteado por diversos autores como (Diaz-Barriga & Hernández, 2002) y (Anderson, 2012) referente a la importancia de las estrategias docentes para un aprendizaje significativo, los programa de estas asignatura, incorporan ricas experiencias de aprendizaje práctico que favorecen el aprender haciendo, como laboratorios y proyectos.

Debido a que el nivel de integración de la computadora moderna en unos cuantos chips dificulta grandemente al estudiante poder apreciar en la práctica los principios de diseño, estructura y funcionamiento de la máquina, estas asignaturas deben emplear frecuentemente la estrategia de simulación de arquitectura (Yehezkel & Otros, 2001).

La asignatura de Arquitectura de Máquinas Computadoras II, que requiere abordar los diferentes estilos arquitecturales, ha venido utilizando diferentes simuladores de arquitecturas. Se ha empleado un Simulador de Arquitectura de Acumulador desarrollado por la Universidad de Córdoba y otro de la Universidad Pace de New York ; un Simulador de Arquitectura de Pila de la Universidad Nacional de Ingeniería y el simulador SaM de la Universidad de Cornell , y simuladores de Arquitecturas de Registros de 2 y 3 direcciones como el SMS y el ESCAPE de la Universidad de Bélgica.

## Metodología de desarrollo

# Objetivos

## General

Generar una interfaz de programación de aplicaciones (API ) para definición y gestión (de código fuente) de lenguaje ensamblador con el fin de proporcionar a los estudiantes de la asignatura de Arquitectura de Máquinas II, de la Universidad Nacional de Ingeniería, una herramienta que les facilite realizar simulaciones de arquitectura de máquinas para distintas gammas de procesadores.

## Especifico

* Determinar los requerimientos para la herramienta de generación de código a desarrollar a partir de las características requeridas por la asignatura y las necesidades de los estudiantes para el desarrollo del gestor de código fuente un simulador de arquitectura.
* Establecer los parámetros básicos con los cuales se pueda definir la arquitectura objetivo de la herramienta de simulación.
* Identificar las plataformas de ejecución del programa simulador de microprocesador para la herramienta de simulación.

# Justificación

La asignatura de Arquitectura de Máquinas Computadoras II requiere asistirse de simulación de arquitectura para que los estudiantes comprendan los principios de diseño y el funcionamiento de los procesadores, dado que éstos, al estar integrados en un único chip, no permiten observar cómo se dan los procesos a su interior.

Aunque se han empleado diferentes simuladores de arquitectura, el beneficio que éstos proveen no es óptimo debido a que la interacción que ofrecen al estudiante se limita a ejecutar programas en ensamblador y observar su ejecución visualmente, con lo cual, sólo se fortalecen los procesos de comprensión de la programación y principios de funcionamiento, pero los procesos de conceptualización, diseño e implementación de la arquitectura no se ponen en práctica. Para superar esta debilidad, el programa de asignatura, propone como alternativa que los estudiantes desarrollen un simulador de arquitectura como medio para lograr un aprendizaje significativo que integre, al menos de forma virtual, los procesos que los arquitectos realizan cuando implementan una nueva computadora.

La entrada de estos simuladores son programas en ensamblador de la arquitectura específica, y la salida, la simulación visual del proceso de ejecución de las instrucciones de esos programas en el procesador. Para implementar estos simuladores, los estudiantes deben al menos desarrollar dos módulos o componentes principales: Gestión de código fuente y Simulación de la Ejecución.



**Ilustración 1: Estructura modular de un simulador de arquitectura**

El componente de gestión de código fuente, se encarga de leer el programa en ensamblador en el archivo de entrada, verifica que se ajuste a las reglas gramaticales del lenguaje de la arquitectura y lo convierte a lenguaje máquina en binario (proceso de ensamblado), y entrega a la salida una estructura de datos (objetos) que representa el programa en memoria. El componente de simulación de la ejecución, por su parte, recibe la estructura de datos y simula la ejecución del programa sobre la representación visual de la máquina en el monitor de la computadora.

Empero lo rica en experiencia, esta actividad de aprendizaje ha resultado ser una faena engorrosa para los estudiantes ya que invierten significativo tiempo definiendo mecanismos para obtener e interpretar el código fuente de entrada, lo cual se agrava con el hecho que no tienen formación en implementación de compiladores, dando por resultado dos situaciones muy inconvenientes:

a) Tardan demasiado tiempo implementando los mecanismos de gestión del código fuente de entrada, con lo cual no logran terminar en el semestre la implementación de la simulación de la ejecución al nivel deseado.

b) Buscan mecanismos paliativos para abordar el proceso de entrada y manejo del código fuente, tales como entrada directa a memoria o procesos interactivos de ingreso, lo cual no se corresponde con el proceso real que consiste en ingresar el código desde un archivo, se realice el proceso de ensamblado y se cargue en memoria.

Por tanto, los estudiantes de la asignatura Arquitectura de máquinas computadoras II requieren de herramientas que les facilite implementar el simulador de arquitectura sin tener que lidiar con los procesos de creación de analizador léxico y sintáctico que requiere el componente de gestión de código fuente.

El presente estudio está orientado a implementar una herramienta de generación de código que les permita a los estudiantes, a partir de una definición gramatical del lenguaje ensamblador de la arquitectura, generar las principales clases del componente de gestión de código fuente, con lo cual se simplificará significativamente el proceso de construir el simulador, se reducirá el tiempo requerido para ello, y se mejorará la eficacia del instrumento didáctico, puesto que librará al estudiante de la difícil tarea de codificar el componente de gestión de código (que no pertenece al ámbito de la arquitectura de computadoras), para que se concentre en otros aspectos más importantes como la definición del conjunto de instrucciones, definición de los atributos de la arquitectura, diseño de la estructura del procesador y en la implementación funcional del ciclo de instrucción para la ejecución de instrucciones, que son el objetivo de la asignatura.

# Marco Teórico

## Herramientas de desarrollo

### QT Creator

Es un entorno de desarrollo integrado IDE, creado y mantenido por la empresa Trolltech; es ampliamente utilizado para desarrollar aplicaciones basadas en librerías QT, es multiplataforma y puede correr en sistemas Windows, Linux y Mac.

Incluye un manejador de proyectos que puede usarse en una variedad de formatos de proyecto tales como .pro, CMake, Autotools entre otros.

Integra un editor de código y un interfaz de diseño (Qt Designer) para construir interfaces graficas de usuarios (GUI) utilizando los Qt widgets. El editor de código suporta resaltado para varios lenguajes además puede analizar código en c++ y en QML, dando como resultado un editor de código con completación de código, también provee navegación semántica.

La herramienta de diseño Qt Designer es una herramienta que permite crear interfaces graficas utilizando los controles de Qt (Qt Widgets), es posible crear controles y diálogos personalizados utilizando diferentes estilos y resoluciones directamente en el editor.

Los Widgets o controles creados con Qt Designer están integrados con código programado usando el mecanismo de señales(signals) y slots de Qt.

Qt Quick Designer es una herramienta para desarrollar animaciones usando el lenguaje declarativo QML.

Provee soporte para compilar aplicaciones para entornos de escritorio windows, Linux, freebsd y mac os, dispositivos móviles (Android, BlackBerry, Maemo y MeeGo) y dispositivos con sistemas Linux embebidos.

Es fácilmente integrable a sistemas de control de versiones entre los cuales están: GIT, Subversion, Preforce, Bazaar, CVS y Mercurial.

Posee un simulador (Qt Simulator) para probar aplicaciones destinadas a dispositivos móviles, simulando un entorno similar al dispositivo de destino.

Qt creator no incluye un debugger de manera nativa, pero posee un plugin que actua de interface entre el Core de Qt Creator y un debugger nativo de c++, los debugger soportados son los siguientes: Gnu Symbolic Debugger (GDB), Microsoft Console Debugger(CDB), LLVM debugger(LLDB)

### C++

Es un lenguaje de programación utilizado ampliamente en su paradigma orientado a objetos fue diseñado a mediados de los años 80 por Bjarne Stroustrup con la intención de extender el lenguaje de programación C.

El nombre C++ fue propuesto por Rick Mascitti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera del laboratorio científico. Antes se había usado el nombre “C con clases”. La expresión “C++” significa incremento de C.

Existen muchos entornos de desarrollo y compiladores capaces de compilar código de C++, entre los cuales tenemos:

1. Dev C/C++
2. Visual C++
3. wxDevC++
4. Open Watcom
5. CodeLite
6. XCode
7. C++ Builder
8. Netbeans
9. Code::Blocks
10. MinGW

Existe un estándar ISO donde se define el lenguaje C++, es el estándar ISO/IEC 14882 actualmente se encuentra en su versión 14, por esta razón C++ en dependencia de las librerías que se utilice puede ser compilado fácilmente en distintas arquitecturas como ARM, PowerPC, Intel 8086, AMD64, etc y sistemas operativos como Unix, Mac OS, Linux, Windows etc.

### QT

Es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con interfaz gráfica de usuario, así como también para el desarrollo de programas sin interfaz gráfica, como herramientas para la línea de comandos y consolas para servidores.

Qt es famoso por ser utilizado en KDE, el entorno de escritorio para sistemas Linux/Unix, está programado en lenguaje C++ nativo, puede ser utilizado en con otros lenguajes de programación a través de “bindings”.

Fue desarrollado inicialmente por Haarvard Nord(CEO de Trolltech) y Eirik Chambe Eng(presidente de Trolltech). Haavard y Eirik se reunieron en el Instituto Noruego de Tecnología de Trondheim, donde ambos se graduaron con una maestría en ciencias de la computación. El kit de herramientas se llamó Qt porque la letra Q lucía atractiva en la fuente Emacs la cual era muy usada por Haavard en su trabajo, y "t" se inspiró en Xt, el kit de herramientas X "X Tool kit".1

Inicialmente Qt apareció como biblioteca desarrollada por Trolltech (en aquel momento «Quasar Technologies») en 1992 siguiendo un desarrollo basado en el código abierto, pero no completamente libre. Originalmente permitía desarrollo de software cerrado mediante la compra de una licencia comercial, o el desarrollo de software libre usando la licencia Free Qt. Esta última no era una licencia real de software libre dado que no permitía redistribuir versiones modificadas de Qt.

Qt dispone de una serie de bindings para diversos lenguajes de programación:

1. PyQt – Bindings GPL/Comercial para Python.
2. PySide13 – LGPL bindings para Python de OpenBossa (subsidiario de Nokia).
3. PythonQt14 – LGPL bindings para Python.
4. Qyoto15 – Bindings para C# u otros lenguajes.NET. Existe un conjunto adicional de bindings Kimono16 para KDE.
5. QtRuby17 – Bindings para Ruby. Existe un conjunto adicional de bindings, Korundum para KDE.
6. Qt Jambi18 – Bindings para Java.
7. QtAda19 – Bindings para Ada.
8. FreePascal Qt420 – Bindings para Pascal.
9. Perl Qt421 – Bindings para Perl.
10. PHP-Qt – Bindings para PHP.
11. Qt Haskell22 – Bindings para Haskell.
12. lqt23 – Bindings para Lua.
13. DaoQt24 – Bindings para Dao.
14. QtD25 – Binding para D.

### Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta múltiples lenguajes de programación tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby, PHP; al igual que entornos de desarrollo web como ASP.NET MVC, Django, etc., a lo cual sumarle las nuevas capacidades online bajo Windows Azure en forma del editor Monaco.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos, consolas, etc.

Actualmente Visual Studio es un IDE muy completo capaz de desarrollar tanto aplicaciones web como aplicaciones desktop y aplicaciones para dispositivos embebidos, asi mismo también permite compilar tu compliar el código fuente para diferentes arquitecturas tales como 32 y 64 bits e Itanium.

# Análisis y Presentación de Resultados

# Bibliografía

# Anexos

## Introducción al lenguaje ensamblador

### Lenguaje de Bajo Nivel

Los lenguajes de bajo nivel, también llamados lenguajes de ensamblador, permiten al programador escribir los programas usando palabras nemotécnicas que ejercen control completo sobre el hardware, razón por la cual son muy dependientes del hardware donde se trabaje, no es lo mismo el lenguaje ensamblador para procesadores Intel que el lenguaje ensamblador para procesadores PowerPC(IBM) o procesadores ARM etc,

El uso de la palabra bajo en su denominación no implica que el lenguaje sea menos potente que un lenguaje de alto nivel, si no que se refiere a la reducida abstracción entre el lenguaje y el hardware. Por ejemplo, se utiliza este tipo de lenguajes para programar tareas críticas de los sistemas operativos, de aplicaciones en tiempo real o controladores de dispositivos.

### Ventajas y desventajas de los lenguajes ensambladores

Los lenguajes ensambladores tienen algunas ventajas sobre los lenguajes de alto nivel, por ejemplo:

Velocidad: debido a que el lenguaje ensamblador trabaja directamente con el hardware permite más rápido acceso a los recursos de la computadora por esta razón la velocidad, no obstante esto también puede depender mucho de la calidad del código fuente.

Flexibilidad: el lenguaje ensamblador es flexible ya que puede utilizar el hardware de en su totalidad, sin restricciones de ningún tipo, también puede trabajar independiente del sistema operativo por ejemplo un programa en ensamblador puede correr tanto en sistemas Unix como en sistemas Windows,

Desventajas

Portabilidad mínima: los programas creados en ensamblador al depender tanto de la arquitectura del procesador, no pueden ejecutarse en computadoras que posean una arquitectura diferente de donde fueron creados.

Mantenimiento de código es muy complicado: modificar, agregar o eliminar alguna instrucción y/o función resulta mucho más complicado debido a que el lenguaje ensamblador no permite mucha abstracción por esta razón el código fuente es más grande.

## El procesador

## Unidades Funcionales Básicas

### Unidad Aritmética Lógica (ALU)

La ALU es la encargada de realizar operaciones aritméticas sobre los datos de la memoria tales como suma, resta, multiplicación y división, también se encarga de realizar las operaciones lógicas, tales como AND,OR, NOT etc.

Unidad Aritmética: todas las operaciones pueden realizarse mediante tres sistema:

* Sistema serie: se procesa bit a bit en forma secuencial
* Sistema paralelo: se procesan simultáneamente todos los bits.
* Sistema paralelo-serie: se procesa por grupos. Los grupos se procesan en serie y los bits que componen los grupos en paralelo.

Unidad Lógica: establece comparaciones para facilitar la toma de decisiones. Estas comparaciones son mucho más sencillas que las operaciones aritméticas, puesto que no necesitan considerar el resultado de la operación realizada con los bits anteriores, pudiéndose efectuar en paralelo, El resultado de las operaciones puede ser vedadero o falso, que en representación binaria es 1 o 0.

### Unidad de Control (CU)

La unidad de control se encarga de buscar las instrucciones en la memoria principal, interpretarlas y ejecutarlas, empleando para ello la unidad empleando para ello la unidad de procesos,

Coordina todos los componentes del computador de modo que los eventos tomen lugar en la secuencia apropiada en el momento correcto. Además de realizar esta labor de sincronización, la unidad de control, es decir entiende las instrucciones del programa que obtiene de la memoria y dirige la acción para realizarlas.

La secuencia lógica que la unidad de control debe realizar para ejecutar una instrucción es la siguiente:

1. Localizar y extraer de la memoria principal la instrucción correspondiente.
2. Transferir la instrucción de la memoria a la Unidad de Control.
3. Determinar qué tipo de operación se debe ejecutar.
4. Ejecutar la instrucción, enviando las señales de control u órdenes a los elementos pertinentes.
5. Supervisar la operación anterior para determinar si ha finalizado correctamente.
6. Localizar la siguiente instrucción a ejecutar.

Elementos de la Unidad de Control

El reloj: consiste en un circuito eléctrico capaz de generar una sucesión de pulsos a intervalos de tiempo constantes. El intervalo entre dos puntos de reloj se denomina *ciclo*.

Contador de programa (PC): también denominado registro contador de instrucción, (RCI). Su misión es controlar el orden de ejecución de las instrucciones del programa, de acuerdo con su contenido. Un programa no siempre ejecuta las instrucciones secuencialmente. Puede haber instrucciones de salto o bifurcación.

Registro de Instrucción: es una unidad de almacenamiento temporal, este registro guarda la instrucción cuando se extrae de la memoria principal y se mantiene mientras se realiza la decodificación o interpretación.

Decodificador: habitualmente, toda instrucción contiene un campo conocido como código de operación, que indica el tipo de operación que hay que realizar; el decodificador es el elemento encargado de realizar el análisis del código de operación.

Secuenciador: Es un generador de ordenes simples, denominadas microordenes que sincronizadas con el reloj y distribuidas a los elementos necesarios permiten la ejecución de la instrucción.

1. Arquitectura de registros dos direcciones

Esta arquitectura se caracteriza por solo definir 2 operandos, uno corresponde a fuente y destino a la vez por tanto se da una lectura destructiva.

1. Código de instrucciones
   1. Instrucciones de transferencia de datos

MOV

Copia el operando 2 al operando 1

Operandos posibles:

Registro, Memoria

Memoria, Registro

Registro, Registro

Memoria, Inmediato

PUSH

Almacena un valor de 16 bits en la pila

Operandos posibles

Registro

-sreg

Memoria

Inmediato

POP

Obtiene un valor de 16 bits de la pila

Operandos posibles:

Registro

Memoria

-sreg

* 1. Instrucciones Lógicas

AND

Instrucción lógica AND entre todos los bits de 2 operandos, el resultado se almacena en el operando 1.

Operandos posibles

Registro, Memoria,

Memoria, Registro

Registro, Registro

Memoria, Inmediato

Registro, Inmediato

OR

Operación lógica OR entre todos los bits de 2 operandos, el resultado es almacenado en el primer operando.

Operandos posibles:

Registro, Memoria

Memoria, Registro

Registro, Registro

Memoria, Inmediato

Registro, Inmediato

XOR

Operación lógica XOR (OR exclusivo) entre todos los bits de un operando, el resultado es almacenado en el primer operando.

Operandos posibles:

Registro, Memoria

Memoria, Registro

Registro, Registro

Memoria, Inmediato

Registro, Inmediato

NOT

Invierte cada bit del operando.

Operandos posibles:

Registro

Memoria

* 1. Instrucciones Aritméticas

ADD

Suma los operandos

Operandos posibles:

Registro, Memoria,

Memoria, Registro

Registro, Registro

Memoria, Inmediato

Registro, Inmediato

## Plataforma de desarrollo

Java

Java es el nombre de un entorno o plataforma de computación originaria de Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java u otros lenguajes que compilen a bytecode y un conjunto de herramientas de desarrollo. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino más bien una máquina virtual encargada de la ejecución de las aplicaciones, y un conjunto de bibliotecas estándar que ofrecen una funcionalidad común.

Máquina Virtual de Java

El núcleo de la plataforma Java es el concepto común de un procesador virtual que ejecuta programas escritos en el lenguaje de programación Java. En concreto ejecuta el código resultante de la compilación del código fuente, conocido como bytecode.

Bytecode

Es el resultado de compilar un código fuente de java, el código binario de Java no es un lenguaje de alto nivel, sino un verdadero código de máquina de bajo nivel, viable incluso como lenguaje de entrada para microprocesador físico, pero los productos de derivados de construir microprocesadores que aceptaran el Java bytecode como lenguaje de maquina fueron infructuosos,

Fuente Java

(.java)

Bytecode

(.class)

Máquina Virtual de Java

Sistema Operativo

C#

Código Fuente

Recursos

Referencias

Compilador

Ensamblado (.exe o .dll)

Metadatos MSIL

.NET Framework

Common Language Runtime

Garbage Collection

JIT Compiler

Clases y librerías de .NET Framework

Sistema Operativo

Convierte a código de maquina nativo

Crea

Metadatos IL y referencias cargadas por CLR