**Titulo:**

**Informe 2: Lenguaje Ensamblador y la arquitectura MIPS**

**Resumen:**

El siguiente es un informe sobre la introducción al manejo de programas de síntesis, compilación y programación de códigos usando arquitectura MIPS32 en el simulador Spim de interfaz QTSpin. El ejercicio académico consiste en el estudio del proceso de compilación y el desarrollo de un código fuente ensamblador en hardware simulado para generar una aplicación.

**Términos clave;**

Ensamblador, MIPS32, Spim, QtSpim, Compilador, CPU, secuencia de código, Arquitectura de procesador, linker, lenguaje de programación, subrutina, algoritmo, hardware, software, instrucciones de máquina, binario**.**

**Contenido:**

**Sobre Spin y MIPS32**

El ejercicio académico consistió en el proceso de programación de instrucciones en un procesador simulado de 32bits que usa la arquitectura MIPS32 de la empresa MIPS Technologies. La herramienta compiladora se denomina QtSpin; la interfaz de usuario que usa el código fuente Spin el cual hace la simulación de lectura y ejecución de programas para la arquitectura MIPS32. **[1][2]**

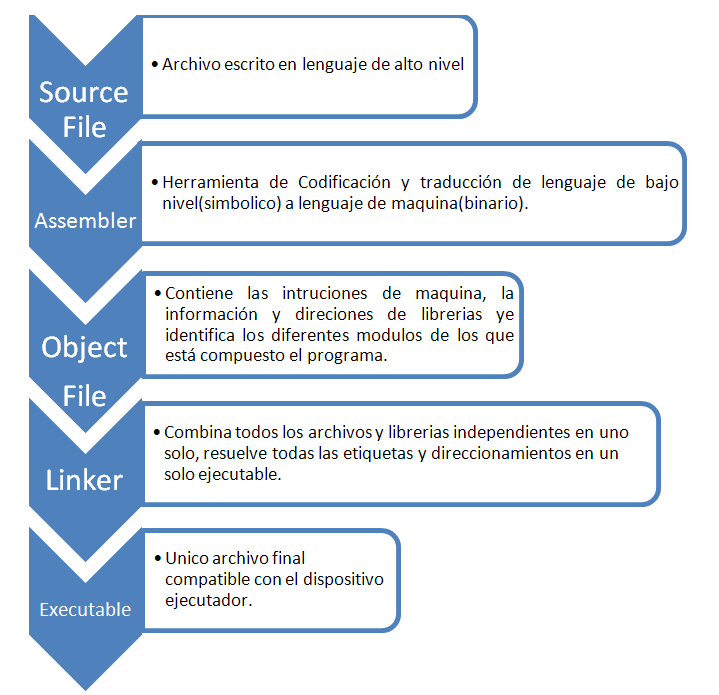
Las intrusiones a implementar se manejan en lenguaje ensamblador, basta con usar un editor de texto plano que permita modificar la extensión a ‘.s ’ o ‘.asm ’ los cuales identifican un código fuente de assembler y permite que QtSpin los reconozca como archivos compatibles. Basta con cargar el archivo desde la interfaz del programa para que el simulador analice y decodifique el archivo fuente en sus diferentes secciones de la ventana principal; Registros de punto flotante (FP Regs), registros enteros (int Regs), contenido de memoria y la pila de carga (Data), el texto de usuario con instrucciones de programa (text) y la interfaz de usuario para recibir y mostrar valores (console). **[3]**

Una vez cargado el código fuente, en la parte superior de la ventana encontramos lo iconos de control, los mas usados, ejecutar compilación (Run/Continue), simulación paso a paso ( y con la tecla dedicada F10) y el de reinicio de registros y volver a valores iniciales de simulación (Reinitialize Simulator).

Spim implementa casi por completo las instrucciones de MIPS32 ( **[4]** cuadro detallado de las instrucciones detalladas), no soporta la mayoría de comparaciones de punto flotante, métodos de redondeo y las tablas de paginaciones de memoria ( **[5]** set de instrucciones de Spim). **[2]**

**Proceso de compilación**

Es una transformación de un programa fuente a un ejecutable; analiza el lenguaje de alto nivel y se lo entrega al compilador el cual hace un proceso de decodificación, pruebas de error de código e interpreta todas las librerías, luego enlaza e inserta los bloques independientes del programa objeto para hacer una síntesis final, en este caso a lenguaje de máquina. **[6]**

****

*Figura 1, proceso de creación de un ejecutable* ***[6]***

**Lenguaje ensamblador;**

También llamado assembler *“Es un lenguaje de programación de bajo nivel para los computadores, microprocesadores, microcontroladores, y otros circuitos integrados programables”*[7] . Fue el principal lenguaje en el inicio del desarrollo de software y actualmente se usa al manipular directamente el hardware en aplicaciones específicas.

Este lenguaje simboliza códigos de maquina binarios por medio de una serie de instrucciones que son las únicas que pueden ser reconocidas y ejecutadas por una arquitectura de CPU, de manera que es una versión elemental y reducida que traduce a lenguaje de máquina.

La importancia del lenguaje radica en que trabaja directamente con el microprocesador, ocupa poco espacio en memoria al carecer de librerías a cargar y es flexible al permitir todas las operaciones que la maquina ofrece. **[8]**

**Lenguaje de alto nivel y ensamblador:**

Los lenguajes de alto nivel, son aquellos que usan interacciones más cercanas a las capacidades cognitivas humanas, a diferencia del lenguaje ensamblador, que su lógica ejecutora es más parecida a la de las maquinas.

En los lenguajes de alto nivel, una sola declaración da lugar a muchas instrucciones de máquina, la generación del código es mucho más sencillo y comprensible reduciendo el tiempo de programación, los tamaños de programas fuente y ofrece un amplio soporte de detección de errores. **[9]**

**Subrutina:**

Se le conoce también como subalgoritmo y subprograma, Es una sección o parte de determinado programa, separado del programa principal, se usa para realizar una función específica, puede ser llamada cuantas veces se le requiera y cualquier momento de la ejecución. Son usadas para reducir tamaños de código, aumentar eficiencia de ejecución, organizar y ayudar a la estructuración del mismo.

Las Subrutinas se diferencian de las funciones de programaciones en la sintaxis de definición y en la forma de invocación, no poseen argumentos y acceden a tosas las variables y contantes del programa principal. **[10]**

**Interrupciones;**

Es un rompimiento de la secuencia de ejecución de un programa una señal recibida, le indica que debe iniciar un código específico y al finalizar regresa al punto donde interrumpió.

Estas señales de interrupción pueden proceder del mismo procesador o por hardware externo;

Las interrupciones internas se generan por eventos programados en la ejecución de código, de manera que solamente accediendo a la programación de hardware se pueden modificar.

Las interrupciones por hardware son provocadas por dispositivos o módulos externos al procesador, estas no son programadas y pueden ocurrir en cualquier momento del programa, si el usuario puede decidir, se denominan “interrupciones enmascarables y en caso contrario, que siempre interrumpirán el programa, no enmascarables. **[11]**

***Conclusiones***

Para la creación de un código de fuente ensamblador se requiere una amplia documentación de de la arquitectura a trabajar, así como un radical cambio de lógica cognitiva de la programación tradicional en alto nivel.

La anterior fue una práctica con fines académicos ya que los compiladores de alto nivel pueden producir un lenguaje de maquina directamente, el ejercicio consistió en generar las bases del comportamiento de la arquitectura del procesador y del proceso de compilación.

Generar código en lenguaje ensamblador es mucho más complicado, extenso y difícil de leer. Esto se traduce en un aumento del tiempo de trabajo dedicado y dificultad de encontrar errores y generar cambios al programa, esta programación es usada en aplicaciones específicas de alto desempeño, aumento de eficiencias y para manipulación directa del hardware.

En el momento de diseñar una aplicación se tiene que identificar las necesidades que se requieren, posibilidades que ofrecen, operaciones que puede realizar, limitantes, velocidades y desempeño, estos parámetros se pueden deducir en base al la escogencia de componentes y la arquitectura en la que se trabajará el sistema.

***Referencias:***

[1] Mips technologies, Sobre la arquitectura MIPS32 [fecha de consulta 23 de marzo de 2012] Disponible en: <http://www.mips.com/products/architectures/mips32/>

[2] Spim, Fuente del simulador de MIPS32 [fecha de consulta 23 de marzo de 2012] Disponible en: <http://pages.cs.wisc.edu/~larus/spim.html>

[3]Escuela de sistemas Universidad de los Andes Venezuela, Introducción al simulador SPIM [fecha de consulta 23 de marzo de 2012] Disponible en: <http://sistemas.ing.ula.ve/~wladimir/ArquitecturaComputadoras/ManualSPIM.pdf>

[4] Wikipedia, sitio web, Procesador MIPS32; intrusiones reales [fecha de consulta 23 de marzo de 2012] Disponible en: <http://es.wikipedia.org/wiki/MIPS_(procesador)#Instrucciones_reales>

[5] Esciela de computación, ciencia y estadística de Dublin, instrucciones soportadas por SPim [fecha de consulta 23 de marzo de 2012] Disponible en: <http://www.scss.tcd.ie/~waldroj/itral/spim_ref.html#instructions>

[6] James R. Larus, ensambaldores, enlazadores y el simulador Spim [fecha de consulta 23 de marzo de 2012] Disponible en: <http://pages.cs.wisc.edu/~larus/HP_AppA.pdf>

[7] Wikipedia, Lenguaje ensamblador [fecha de consulta 23 de marzo de 2012] Disponible en: <http://es.wikipedia.org/wiki/Lenguaje_ensamblador>

[8] Monografia; Miriam Yazmín Mendoza Caamal, Lenguaje ensamblador [fecha de consulta 23 de marzo de 2012] Disponible en: <http://www.monografias.com/trabajos14/lenguaje-ensamblador/lenguaje-ensamblador.shtml>

[9] Wikipedia, Lenguaje de alto nivel [fecha de consulta 23 de marzo de 2012] Disponible en: <http://es.wikipedia.org/wiki/Lenguaje_de_alto_nivel>

[10] alciro Books, Sobre la Subrutina [fecha de consulta 23 de marzo de 2012] Disponible en: <http://www.alciro.org/alciro/microcontroladores-8051_24/subrutina-subprograma_357.htm>

[11] MC Beatriz Beltrán Martinez, Sobre interrupciones de hardware [fecha de consulta 23 de marzo de 2012] Disponible en: <http://www.slideshare.net/SpiderHal/interrupciones-de-hardware>