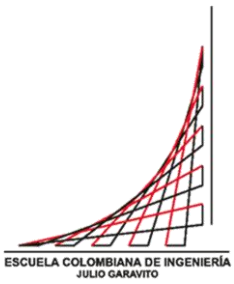
RISC5 Y PROJECT OBERON PARA RASPBERRY PI

DAVID CAMILO BARRERA RIBERO

Profesor:
GERARDO OSPINA HERNÁNDEZ



ESCUELA COLOMBIANA DE INGENIERÍA
INGENIERÍA DE SISTEMAS
2015

DOCUMENTO FINAL DE PROYECTO DE GRADO

1. **Contexto**
   - 1.1. **Planteamiento del problema**
   - 1.2. **Marco teórico y estado del arte.**
   - 1.3. **Objetivo**
   - 1.4. **Justificación**
   - 1.5. **Área de aplicación del producto resultado del proyecto.**

2. **Requerimientos**
   - 2.1. **Descripción del sistema**
   - 2.2. **Visión y alcance**

3. **Implementación**
   - 3.1. **Especificación de estándares utilizados**

4. **Liberación**
   - 4.1. **Configuración ambiente mínima/ideal**
   - 4.2. **Manual técnico**
   - 4.3. **Herramientas**

Contexto

Planteamiento del problema

Existen dos plataformas diferentes en las cuales se pude ejecutar el sistema Oberon, la primera es utilizar el FPGA “Spartan 3”, que fue para la cual se diseñó originalmente el sistema Oberon, y la segunda es a través de un emulador del procesador RISC5.

Existen dos emuladores para el procesador virtual RISC5 de Oberon, uno para Windows y otro para Linux, ambos están escritos en C y requieren de SDL2.

Para el caso de Windows el emulador viene con un dll para SDL2 y un ejecutable, con lo cual no es necesario compilar las fuentes del mismo. Por el contrario para Linux es necesario instalar SDL2 y tener un compilador con el estándar C99 para poder compilar las fuentes del emulador, para emular el sistema operativo Oberon es necesario especificar la ruta de RISC.img en el ejecutable que se genera.

En ambos casos el emulador provee acceso a la pantalla, disco, USB mouse y teclado, además de transferir archivos a través de los scripts pcreceive.sh y pcsend.sh. Por ahora el emulador no provee acceso a red.

Marco teórico y estado del arte

Introducción

A continuación se mostrara desarrollo que ha tenido Project Oberon y el procesador RISC5. Para este fin, se muestra la literatura relevante al tema y su contribución al mismo. Todo esto don el fin de proponer un enfoque de investigación y aplicación, que permita generar innovación en esta temática.

Trabajo relacionado

THE DESIGN OF A RISC ARCHITECTURE AND ITS IMPLEMENTATION WITH AN FPGA

La motivación para el proyecto de diseñar un procesador de arquitectura RISC (del inglés Reduced Instruction Set Computer, en español Computador con

Conjunto de Instrucciones Reducidas) y su implementación en un FPGA (del inglés Field Programmable Gate Array), proviene de primeramente de diseñar y poner en práctica un pequeño procesador para su uso en sistemas embebidos con varios núcleos interconectados (TMS Tiny Register Machine), y en segundo lugar, es el libro (Compiler Construction) escrito por Niklaus Wirth, el cual trata sobre la construcción de un compilador para una arquitectura de computador hipotética.

La intención de este proyecto es es reemplazar el, computador emulado hipotéticamente por uno de verdad. Esta idea se hizo realista por el advenimiento de los componentes de hardware programables llamados matrices de puertas programables de campo (FPGA).

Como primera medida se realizó un proyecto en el cual se presentó una arquitectura más sencilla que la de los procesadores reales, porque el objetivo del proyecto, lo cual se refleja en el paper, no era presentar una arquitectura compleja sino más bien mostrar los conceptos fundamentales para el diseño de este procesador y “optimizar” dicho procesador, por lo cual se pude entender la escogencia de un procesador RISC (Reduced Instruction Set Computer). Como se había mencionado al principio, la implantación del procesador RISC se realizó en un FPGA ya que este proporciona una cantidad sustancial de libertad para el diseño, y para la realización del circuito utilizaron el lenguaje de realización de hardware Verilog el cual provee una ventaja sobre un esquema grafico de un circuito, ya que este es similar a un lenguaje de programación, pero a diferencia de este al compilarlo este genera un circuito.

El desarrollo del procesador RISC sigue cinco capas de desarrollo, la primera de ellas es propio desarrollo de la arquitectura de este, le sigue el ISC-0 la cual tiene dos memorias distintas, una se utilizan para el programa y la otra para los datos, esta arquitectura permite un buena separación en entre la unidad aritmética y la unidad de control. En la siguiente etapa RISC-1 la BRAM para los datos se sustituye por este SRAM. Para la etapa del RISC-2 se cambia la arquitectura a una arquitectura Von Neumann. En la última etapa RISC-3 se le añaden dos características, interrupciones y acceso a bit.

La arquitectura del procesador RISC se compone de elementos individuales de direcciones de 8 bits, cuanta con un ALU el cual tiene un banco de 16 registros de 32 bits, cada 32 bits se denomina palabra. Como es característico en los procesadores RISC, los datos pueden transferirse entre la memoria y los registros, por instrucciones de carga y almacenamiento separadas.

Como ya se ha mencionado anteriormente, el procesador RISC contiene un número reducido de instrucciones, con lo cual un objetivo importante de este, es tener cierta integridad. Un conjunto de instrucciones debe permitir componer cualquier operación más compleja a partir de instrucciones básicas. También tener reglas consecutivas sin excepciones, las cuales facilita la descripción, comprensión y aplicación enormemente. Claramente al tener una arquitectura tan reducida como esta trae problemas con la velocidad, aunque este problema se espera afrontarlo más adelante.

La arquitectura RISC posee tres clases de instrucciones, (1) las instrucciones aritméticas y lógicas que operan en un registro, (2) las operaciones de transferencia de datos entre los registros y la memoria, y (3) de control Instrucciones.

THE RISC ARCHITECTURE

Aunque en el anterior paper "*The Design Of A Risc Architecture And Its Implementation With An Fpga"* se describió detalladamente el proceso de diseño e implementación del procesador RISC, pasando por las etapas del RISC-0 a RISC-3, aparte de la motivación de la escogencia de este procesador. En este paper se retomó la arquitectura del procesador RISC, y se hace una comparación indirecta entre el procesador RISC-0 y el RISC-5.

RISC-0 es el origen de una serie de extensiones de este ismo procesador. Este está basado en una arquitectura de Harvard, y que utiliza la memoria interna RAM del FPGA como su memoria, la cual está restringida a 8K palabras de programa y 8K palabras de datos. No ofrece acceso byte, y las instrucciones de punto flotante no son posibles, a diferencia del procesador RISC-0, el procesador RISC-5, el cual es una extensión del procesador RISC-0, está basado en una arquitectura de von Neumann y utiliza el mismo conjunto de instrucciones que RISC-0. La memoria consiste en la SRAM interna de a bordo con una capacidad de 1 MB. Acceso Byte está disponible, y también lo son las instrucciones de punto flotante. A parte de esto el procesador RISC-5 provee acceso a mas dispositivos externos, entre ellos están el mouse y el teclado.

PROJECT OBERON THE DESIGN OF AN OPERATING SYSTEM,A COMPILER, AND A COMPUTER

Con este libro sus autores Niklaus Wirth, Jürg Gutknecht, justificar el esfuerzo y su motivación de diseñar y construir todo un sistema operativo desde cero, y presentan una serie de conceptos básicos para la construcción de este por capítulos.

La motivación de los autores de este libro por hacer un sistema operativo, fue el querer entender otro sistema operativo, de un computado avanzado de su época, con el cual ellos trabajaban, pero debido a su complejidad y a los tecnicismos, no les era posible entenderlo en su totalidad, ya que ellos querían comprender los fundamentos de los aspectos novedosos del sistema. Lo que lo hizo diferente de los sistemas operativos convencionales? ¿Cuál de estos conceptos eran esenciales, cuáles podrían ser mejorados, simplificados, o incluso descartados? ¿Conocer dónde estaban sus raíces? ¿Podría ser destilado la esencia del sistema y ser extraída, como en un proceso químico?. Posteriormente ya cuando el autor conto con el tiempo y los medios para empezar la construcción de su sistema operativo “Oberon”, tomando como una de las reglas para la realización de este fuera que el resultado debe ser capaz de ser utilizado como material didáctico, con lo cual se propuso a realizar, a parte del sistema operativo, un libor que explicase en verdad este sistema operativo y no uno con conceptos abstractos a medio hacer.

Para la realización del sistema operativo Oberon se utilizó Modular-2, aunque este leguaje contaba con la desventaja ligar al sistema operativo a cierto tipo de máquina, lo cual iría en contra de la propiedades esperadas de este sistema operativo. Por lo tanto, Modula-2 se amplió con una característica de extensión de tipo. Al mismo tiempo se reconoció que Modula-2 contenía varias instalaciones que no utilizarían y que no contribuyen realmente a su poder de expresión, pero al mismo tiempo aumentar la complejidad del compilador.
Pero el compilador no sólo tendría que ser implementado, sino también debe ser fácilmente descrito, estudiado y entendido. Esto llevó a la decisión de empezar desde cero también en el dominio de diseño del lenguaje, y para aplicar el mismo principio “concentrarse en lo esencial”. El nuevo lenguaje, que todavía lleva mucha semejanza con Modula-2, se le dio el mismo nombre que el sistema Oberon.

THE PROGRAMMING LANGUAGE OBERON

En este escrito se explica el lenguaje de programación Oberon, sin llegar a ser este un tutorial para el aprendizaje del mismo. Como ya se avía mencionado en el anterior paper, Oberon es un lenguaje de programación de propósito general que se desarrolló de Modula-2, siendo una característica y novedad de que es un lenguaje de extensión de tipo, lo cual permite la construcción de nuevos tipos de datos sobre la base de los ya existentes y relacionarlos.

La sintaxis es comúnmente el factor que mayormente diferencia un lenguaje de otros, por lo tanto es la sintaxis la que comúnmente trae problemas al programar, ya que el programado está más familiarizado con los concepto de la programación que con la sintaxis de un lenguaje determinado, al momento de comenzar a trabajar con este. Al ser Oberon desarrollado a partir de Modula-2 su sintaxis es similar. Con lo cual aquellos que están familiarizados con el

lenguaje de programación Modula-2, se les facilita el uso y comprensión de Oberon.

La sintaxis de Oberon esta descrita utilizando el lenguaje EBNF, que es una una extensión de Backus-Naur (el cual nos permite expresar gramáticas libres de contexto). Las entidades sintácticas (símbolos no terminales) se indican con palabras en inglés que expresan su sentido intuitivo. Símbolos del vocabulario del idioma (símbolos terminales) se indican mediante cadenas encerradas entre comillas o palabras escritas en mayúsculas, las llamadas palabras reservadas. Para la representación de símbolos dentro del lenguaje en términos de caracteres se define utilizando el conjunto ASCII, además Los espacios en blanco y saltos de línea no deben ocurrir dentro de los símbolos (excepto en los comentarios y espacios en blanco en las cadenas). Estas son ignoradas a menos que sean esenciales para separar dos símbolos consecutivos. Las letras mayúsculas y minúsculas se consideran como un sistema distinto. Para las declaraciones Oberon define un identificador el cual es definido por el programa programa, a menos que sea un identificador predefinido. Declaraciones también sirven para especificar ciertas propiedades permanentes de un objeto, como si es una constante, un tipo, una variable o un procedimiento.

Definición del tema

Gracias a esta investigacion, se han destacado diversos puntos clave para el desarrollo del este proyecto para portar el emulador del procesador RISC-5 en una Raspberry Pi. Es por ello que en esta ocasión se hace énfasis en el desarrollo e implementación de la arquitectura del procesador RISC-5, y de igual manera se evidencian las motivaciones de Project Oberon y su diseño e implementación junto con algunas de sus características.

Objetivo del proyecto

- Portar el emulador del procesador RISC5 a Raspberry Pi de tal manera que accese de manera directa el hardware
- Portar el sistema Project Oberon para que ejecute sobre la máquina virtual implementada para Raspberry Pi

Justificación

Ya que el sistema Oberon y el procesador virtual RISC5 fueron planeados originalmente para ser ejecutados en un FGPA, en específico en una Sparten 3, la cual esta descontinuada en la actualidad y es difícil adquirir una ya utilizada, se planteó, dada la popularidad de la Raspberry Pi, portar el sistema Operativo en esta. Ya que así se daría a conocer más este sistema operativo, empezando por la comunidad de la Raspberry Pi. A demás como Project Oberon y la Raspberry Pi fueron diseñados y orientados a la enseñanza y aprendizaje, estos proveen una gran oportunidad para facilitar la enseñanza de cómo funcionan los sistemas operativos y como se pude desarrollar uno en un computado de una arquitectura relativamente sencilla.

Se planteó portar el emulador del procesador RISC5 de Oberon porque este al estar escrito en el lenguaje de programación C y utilizar SDL2, se facilita remplazar las funciones que hacen uso de un sistema operativo o del SDL por unas similares que hacen uso directamente de las funciones del procesador ARM.

Área de aplicación del producto resultado del proyecto

Como se había mencionado anteriormente el objetivo del proyecto es dar a conocer el sistema operativo Project Oberon a la comunidad de la Raspberry Pi, como sistema alternativo a los ya existentes.

Al ser Project Oberon un sistema orientado al aprendizaje e investigación, el portar este sistema operativo en la Raspberry Pi da a la comunidad que investiga y desarrolla en Oberon una plataforma más accesible para su desarrollo e investigación. También pude ser utilizado como una plataforma para la enseñanza y aprendizaje del desarrollo e implementación de un sistema operativo, tomando como guía el libro “Project Oberon” escrito por Niklaus Wirth y Jürg Gutknecht, en el cual describen en detalle cómo se realizó el sistema operativo Oberon.

Requerimientos

Descripción del sistema

El sistema consiste en tener a Project Oberon y al emulador del prosesaroRISC5 como sistema operativo nativo en la Raspberry Pi.

Visión y alcance

El alcance inicial del proyecto es portar el emulador del procesador RISC5 en la Raspberry Pi y ejecutar Project Oberon sobre la máquina virtual RISC5.

Implementación

Especificación de estándares utilizados

Ya que la codificación se realizó en el lenguaje de programación C, se utilizó el estándar ISO/IEC 9899:1999 más conocido como C99.

Liberación

Configuración ambiente mínima/ideal

Aparte de tener una Raspberry Pi funcional, un monitor, un mouse y un teclado se requiere:

- La herramienta de compilación cruzada de yagarto arm-none-eabi para compilar las fuentes del proyecto. Para la instalación de este sugiero seguir el tutorial del curso de programación a bajo nivel para la Raspberry Pi de la universidad de Cambridge.

- Una tarjeta SD, se recomiendo de 8GB, en formato FAT32.
- En la tarjeta SD tener los archivos necesarios para el BOOT de la Raspberry Pi (bootcode.bin, start.elf, start_cd.elf, start_x.elf). Estos archivos se generan al instalar Raspbian.
- En la tarjeta SD tener la imagen de Oberon “RISC.img”, la cual se pude encontrar en la página oficial de Project Oberon.

Manual técnico

Los siguientes pasos son para la compilación e instalación de Project Oberon en la Raspberry Pi.

1. Para compilar las fuentes del proyecto abra un terminal e ingrese a la carpeta que contiene el archivo make.
2. Digite en la consola make, esto generará un archivo kernel.img en la carpeta bin.
3. En la tarjeta SD remplace el archivo kernel.img por el generado en el paso anterior.
4. Insertar la SD en la Raspberry Pi y encenderla.

Herramientas

Para este proyecto se utilizó la herramienta de yagarto para compilación cruzada arm-none-eabi. El GNU ARM Bare Metal Toolchain (arm-none-eabi) se utilizará para la creación de aplicaciones para la arquitectura ARM sin sistemas operativos como Linux.

Se utilizó la biblioteca Newlib la cual es una implementación de la biblioteca estándar de C destinada a su uso en sistemas embebidos. Es un conglomerado de varias partes de bibliotecas, todas bajoLicencia Open Source que la hacen fácilmente utilizable en productos empotrados.