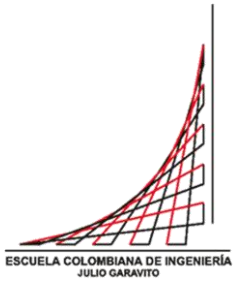
RISC5 Y PROJECT OBERON PARA RASPBERRY PI

RAMIRO VARGAS SALAS

Profesor:
GERARDO OSPINA HERNÁNDEZ



ESCUELA COLOMBIANA DE INGENIERÍA
INGENIERÍA DE SISTEMAS
2017

DOCUMENTO FINAL DE PROYECTO DE GRADO

1. CONTEXTO

El proyecto de grado está basado en la implementación de un emulador para el procesador risc5 en una Raspberry pi modelo B de tal manera que el emulador acceda a los periféricos de teclado ,pantalla, mouse , sd de manera directa por medio del software de la Raspberry modelo B (programación de bajo nivel) utilizando el emulador del compilador de RISC Project Norebo.

DESCRIPCIÓN DEL PROYECTO

1.1 planteamiento del problema.

El sistema Oberón se creó en el Eth “Eidgenössische Technische Hochschule Zürich” por Niklaus Wirth y Jürg Gutknecht, particularmente para ver cómo estaba estructurado un sistema operativo, lo cual lo hace ideal para poder manipular y entender la estructura interna de un sistema operativo. La última versión del sistema Oberón fue creada en 2013; esta incluye la implementación del procesador RISC 5 además del sistema operativo. Fue implementada sobre una FGPA Spartan 3 la cual ya no se produce y dado esto se plantea implementar en la Raspberry pi modelo B este procesador usando un emulador pero con acceso directo al hardware para aumentar la velocidad de ejecución.

1.2 marco teórico y estado del arte:

La mayor fuente de información acerca del tema es el libro creado por Niklaus Wirth y Jürg Gutknecht “Project Oberon the design of an operating system, a compiler, and a computer” de 2013, donde explican a fondo el sistema operativo y cómo funciona el sistema en general, dando una idea general de todo el sistema operativo a fondo con todos sus componentes necesarios para realizar este sistema operativo.

El sistema operativo Oberón está compuesto por frames quienes tiene su propio display y el sistema cierra o abre estos frames según convenga, luego están los textos quienes tiene propiedades como fuentes y módulos quienes se ejecutan por medio de comandos quienes tiene la forma de m.p donde m es el modulo en donde p es declarada y p es el identificador del procedimiento. Luego esta los directorios que forman como un enlace entre los comandos y donde se ejecuta pero en oberon no está directamente relacionado ya que estos comandos también pueden referenciar a variables locales .luego está el kernel quien contienen los drivers e importa memoria e incluye el driver del disco.

El artículo “the design of a RISC architecture and its implementation with an fpga" es de base fundamental para entender cómo funciona Oberon en la práctica. La idea de este artículo es la de implementar por hardware por medio de un FPGA, un procesador de arquitectura RISC “Reduced Instruction set computer” con un número de instrucciones de tamaño fijo y que solo tenga las instrucciones de carga y almacenamiento que acceden a la memoria de datos.

La Raspberry pi modelo B es un ordenador de bajo costo, desarrollado en el reino unido por la fundación Raspberry pi modelo B, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. Este ordenador, que antes era más pequeña que una tarjeta

de crédito tiene varios puertos y entradas, dos USD, uno de Ethernet y salida HDMI. Estos puertos permiten conectar el miniordenador a otros dispositivos, teclados, ratones y pantallas. Este incluye un procesador central (CPU) ARM, un procesador gráfico (GPU) y 512 MB de memoria RAM.

1.3 objetivos generales y específicos:

GENERAL -Implementar el emulador para el procesador risc5 en un Raspberry pi modelo B

ESPECIFICOS

- Portar el emulador risc5 (norebo) a Raspberry pi modelo B de tal manera que acceda de manera directa el hardware

-Programar a bajo nivel (c o assembler) los drivers de acceso a SD, Teclado, Mouse VGA y Consola

1.4 justificación.

Debido a que el procesador Risc5 fue planeado originalmente para ser ejecutado en un FGPA como Spartan 3, la cual esta descontinuada en la actualidad y es difícil adquirir, se identificó a la Raspberry pi modelo B para remplazar este procesador portando un emulador ya existente, ya que la Raspberry modelo B es fácil de utilizar y es compatible con la mayoría de sistemas operativos linux.

1.5 usuarios potenciales directos e indirectos de los resultados de la investigación

los estudiantes de sistemas de la escuela colombiana de ingeniería y potencialmente a otras universidades, el aprendizaje de sistemas operativos e indirectos los investigadores en el tema de Oberón y sistemas operativos ya que este es un tema muy investigado y puede ser de ayuda para otra persona que investiga también en este tema.

2. Requerimientos

2.1Descripción del sistema

Para este objetivo se utilizó la librería circle ci quien ya tiene incorporados los módulos de mouse ,teclado y demás periféricos el trabajo que se desarrolló se inició haciendo pruebas en los diferentes kernels, luego se identificó las posibles funcionalidades que podrían servir para el objetivo del proyecto

2.2Visión y alcance

Para proyecto de grado 2 se espera tener software que maneje directamente los dispositivos de la Raspberry pi modelo B usando programación de bajo nivel **(c o assembler)** sobre el emulador de Project norebo

3. Implementación

3.1 Especificación de estándares utilizados

Project Oberon 2013.

Bare metal programing for raspberry pi modelo B.

CIRCLE CI For bare metal programming

liberacion

4.1 MANUALES TECNICOS

MANUAL DE INSTALACIÓN NOREBO ON RASPBERRY PI

1. DESCARGAR EL PROYECTO DEL REPOSITORIO
   https://github.com/pdewacht/project-norebo
2. PASARLO A LA RASPBERRY YA SEA DESDE LA RASPBERRY O REMOTAMENTE POR SSH
3. VERIFICAR LOS ARCHIVOS EN LA RASPBERRY

```
pi@raspberrypi / $ ls -a
.    boot  home        media           opt             root  selinux  tmp  win
..   dev   lib         mnt             proc            run   srv      usr
bin  etc   lost+found  oberon-risc-emu  project-norebo  sbin  sys      var
pi@raspberrypi / $
```

4. CORRER EL COMANDO "MAKE" PARA GENERAR EL EJECUTABLE

NOTA : INSTALAR LA LIBRERÍA DEL COMPILADOR C EN CASO DE SER NECESARIO

```
pi@raspberrypi ~/project-norebo $ ls
Bootstrap       fetch-sources.py  manifest.csv  Oberon     test
build-image.py  license.txt       norebo        README.md
build.sh        Makefile          Norebo        Runtime
pi@raspberrypi ~/project-norebo $
```

PARTE 1-3

TUTORIALES RASPBERRY PI BARE METAL PROGRAMMING IN C

IN WINDOWS

1. Descargar el gcc-arm ya sea en ejecutable o .zip desde https://launchpad.net/gcc-arm-embedded
2. Identificar la ruta del gcc cuando se ejecuta

```
C:\Program Files (x86)\GNU Tools ARM Embedded\5.4 2016q3>arm-none-eabi-gcc -O2 -mfpu=vfp -mfloat-abi=hard -march=armv7-a
 -mtune=cortex-a7 arm-test.c
arm-none-eabi-gcc: error: arm-test.c: No such file or directory
arm-none-eabi-gcc: fatal error: no input files
compilation terminated.

C:\Program Files (x86)\GNU Tools ARM Embedded\5.4 2016q3>dir
 Volume in drive C has no label.
 Volume Serial Number is 3272-42ED

 Directory of C:\Program Files (x86)\GNU Tools ARM Embedded\5.4 2016q3
```

3. Pasar los archivos necesarios por el programa ya sea el archivo.c , build-rpi-bplus.bat, build-rpi-bplus.sh, build.bat, build.sh

Ejecutar el archivo.c con el comando arm-none-eabi-gcc -O2 -mfpu=vfp -mfloat-abi=hard -march=armv6zk -mtune=arm1176jzf-s arm-test.c

4. Cuando se quiera generar el archive kernel.elf se corre

arm-none-eabi-gcc -O2 -mfpu=vfp -mfloat-abi=hard -march=armv6zk -mtune=arm1176jzf-s -nostartfiles armc-2.c -o kernel.elf

Despues podemos generar el archive kernel-img

arm-none-eabi-objcopy kernel.elf -O binary kernel.img

5. Buscamos la ruta donde genere el .elf o .img genere almente en C:\Users\usuario\AppData\Local\VirtualStore\Program Files (x86)\GNU Tools ARM Embedded\5.4 2016q3

In linux install gcc arm

$ mkdir -p ${HOME}/opt
$ cd ${HOME}/opt
$ tar xjf ~/Downloads/gcc-arm-none-eabi-6-2017-q1-update-linux.tar.bz2
$ chmod -R -w ${HOME}/opt/gcc-arm-none-eabi-6-2017-q1-update

add to path https://gist.github.com/joegoggins/7763637

Parte 4

Generador usando CMAKE

1. DESCARGAMOS LOS COMPILADORES BINARIOS DE https://cmake.org/download/
2. INSTALAMOS CMAKE CON EL COMADO "MAKE-INSTALL" ESTANDO EN LA CARPETA QUE DESCARGAMOS
3. EN EL CODIGO DE LA PARTE 3 EN ADELANTE VAMOS A LA CARPETA SCRIPTS EJECUTAMOS EL SCRIPT CONFIGURE_RPI.SH
4. LUEGO EJECUTAMOS EL COMANDO MAKE PARA GENERAR EL KERNEL.IMG

BIBLIOGRAFÍA:

(2013), niklaus wirth y jürg gutknecht, project Oberon the design of an operating system, a compiler, and a computer

(1991) martin reiser, the oberon system user guide and programmers manual

(2013) niklaus wirth.: the design of a risc architecture and its implementation with an fpga. [11.11.11, rev. 5.10.13]