Sponsored By

SIEMENS

Ingenuity for life

Operating Systems Analysis

Prepared By:

OS Team

INTRODUCTION

Our project aims to optimization of RISC-V ISA for operating systems, for this to be done we need to analyze different operating systems to be able to know the most frequent instructions the operating systems use to extend them in the CPU to enhance the performance of system.

The analysis of operating system is a challenging task and can be done with various ways, the way selected for analysis in this document is as follow:

- Compiling the operating system C code.
- Converting the bin file to assembly.
- Using tools to analyze the assembly code and know the most frequent instruction.

1. uC Operating System

Architecture Used

- Arm Cortex-M3

Tools Used

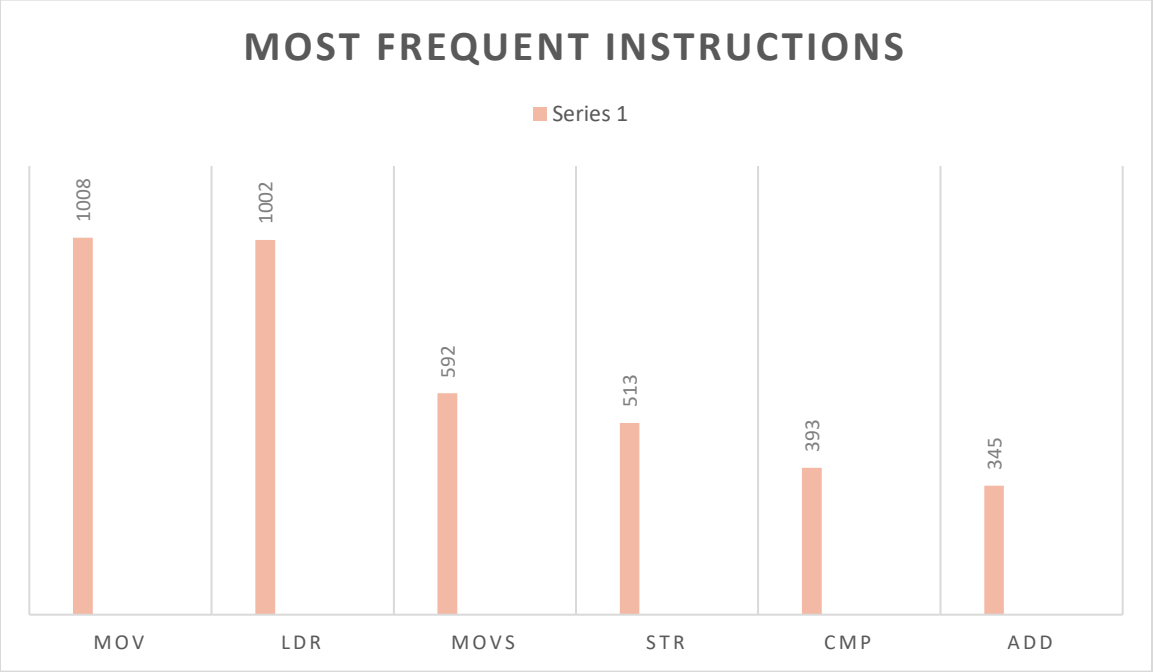
- Keil uVision 5: for compiling the C code.
- Arm GNU Toolchain: for converting the bin file to assembly.
- Text Analyzer: for the analysis of the assembly code.

Process

- Cloning the Repo of the uC, solving the errors and dependencies of the code.
- Compiling the code on keil uVision 5 and getting the ".axf" file.
- Converting the ".axf" file to assembly using arm GNU tool chain.
- Analyzing the assembly file by text analyzer tool and generating the results.

Results

Instruction	Occurrences WRT total words in the file
mov	1008
ldr	1002
movs	592
str	513
cmp	393
add	345



2. FreeRTOS

Architecture Used

- Arm Cortex-M3

Tools Used

- Keil uVision 5: for compiling the C code.
- Arm GNU Toolchain: for converting the bin file to assembly.
- Voyant Tools: for the analysis of the assembly code.

Process

- Cloning the Repo of the FreeRtos, solving the errors and dependencies of the code.
- We used multiple examples to ensure it includes most of the features
- Compiling the code on keil uVision 5 and getting the ".axf" file.
- Converting the ".axf" file to assembly using arm GNU tool chain.
- Analyzing the assembly file by Voyant tools and generating the results.
- Gathering all the examples results

- Example 1
  Task functions
  https://voyant-tools.org/?corpus
  8,977 total words and 2,723 unique word forms

Results

Instruction	Occurrences WRT total words in the file
ldr	332
str	96
bl	93
mov	90
movs	80
b.n	78
cmp	71

- Example 2
  Task blocking
  https://voyant-tools.org/?corpus
  9,098 total words and 2,767 unique word forms

Results

Instruction	Occurrences WRT total words in the file
ldr	346
bl	99

str	93
mov	91
movs	80
b.n	77
cmp	70

- Example 3
  Blocking when receiving from a queue Task to Task Communication
  Voyant Tools (voyant-tools.org)
  12,080 total words and 3,536 unique word forms

○ Results

Instruction	Occurrences WRT total words in the file
ldr	406
bl	177
mov	159
str	127
movs	121
b.n	106

cmp	86

- Example 4
  Interrupt Management Counting Semaphores/Queues
  Voyant Tools (voyant-tools.org)
  12,818 total words and 3,763 unique word forms

○ Results

Instruction	Occurrences WRT total words in the file
ldr	427
bl	185
mov	168
movs	128
str	127
b.n	108
cmp	85

- Example 5
  Using Queues within an Interrupt Service Routine
  Voyant Tools (voyant-tools.org)
  13,008 total words and 3,826 unique word forms

○ Results

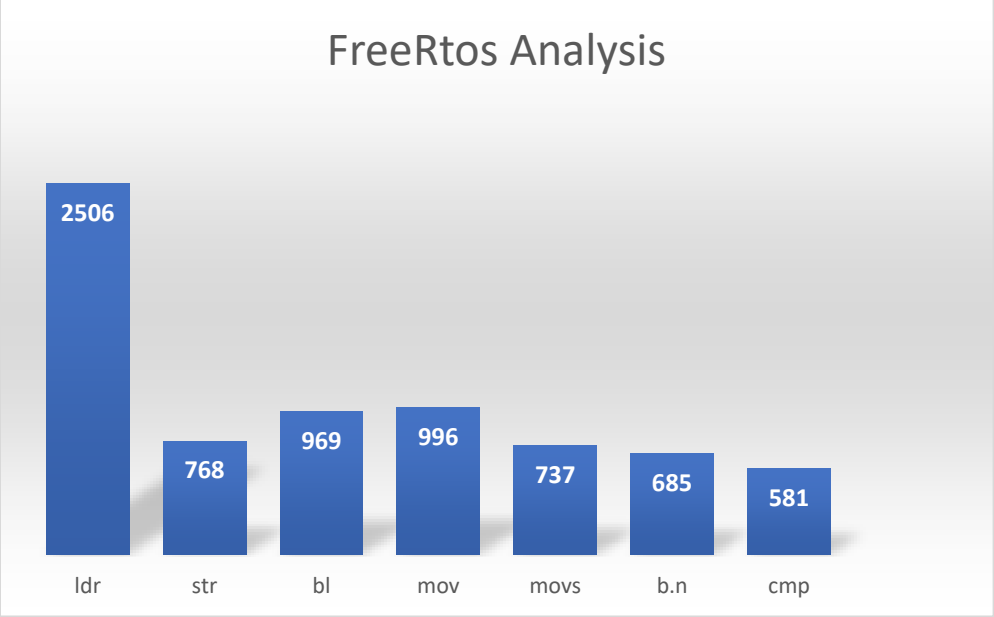
Instruction	Occurrences WRT total words in the file
ldr	441
bl	186
mov	161
str	132
movs	126
b.n	110
cmp	92

- Example 6
  Mutexes (and Binary Semaphores) Resource Management
  Voyant Tools (voyant-tools.org)
  21,740 total words and 6,086 unique word forms

○ Results

Instruction	Occurrences WRT total words in the file
ldr	554
mov	327
bl	229

b.n	206
movs	202
str	193
cmp	177



3. Trampoline Operating System

Architecture Used

- Arm Cortex-M & Cortex-A

Tools Used

- CodeBlocks : for building goil files .
- Arm GNU Toolchain: for compiling and disassembling OS files .
- Python : for running the make file generated by goil.
- Text Analyzer: for the analysis of the assembly code.

Process

- Cloning the Repo of the Trampoline, solving the errors and dependencies of the code.
- Building goil files using codeBlocks , add its path to environment variables.
- Configure the application using goil command in Readme of the example to generate configuration files .
- Compiling the code using command “python make.py”
- Disassembling the binary file using objdump .
- Analyzing the assembly file by text analyzer tool and generating the results.

Results

- **Example 1:**
  - Architecture : Arm Cortex-A .
  - Features : 3 Tasks , 2 ISR , 1 Counter , 1 Alarm , 1 Resource.

Instruction	Occurrences WRT total words in the file
ldr	677
mov	391
str	261
movt	261
movw	230
cmp	204

- **Example 2:**
  - Architecture : Arm Cortex-M .
  - Features : 2 Tasks , 1 ISR , 1 Counter , 1 Alarm .

Instruction	Occurrences WRT total words in the file
ldr	1397
cmp	396
str	364
ldrb	310
mov	310
movs	287