International Journal of Control and Automation
Vol. 10, No. 12 (2017), pp.221-228
http//dx.doi.org/10.14257/ijca.2017.10.12.20

A Study on the Light-Weight Virtual Machine Code for IoT Virtual Machine

JaeHyun Kim[1] and YangSun Lee[1*]

[1]*Dept. of Computer Engineering, Seokyeong University*
16-1 Jungneung-Dong, Sungbuk-Ku, Seoul 136-704, Korea
**yslee@skuniv.ac.kr*

Abstract

The VM (Virtual Machine) is a software processor that has many advantages on software development, release, maintenance and so on due to its platform independent features. But, in execution performance aspect, it has a significant disadvantage that restricted low performance by its execution overhead for the software level interpretation. Also, in the implementation of VM, ISA (Instruction Set Architecture) has a significant impact on various features of the VM such as the structure of interpreter, memory model, system requirements of execution for VM, and etc. Especially, a very small sized ISA required to be operated the virtual machine on the low-performance device such as IoT devices, however, if the size of the ISA is too small then it makes execution performance down of the virtual machine and, makes difficult to improve performance through optimization.

In this paper, the RSIL (Reduces Smart Intermediate Language) is proposed to solve these problems by enhancement of the previous virtual machine code.

Keywords: *Virtual Machine Code, Intermediate Language, Instruction Set Architecture, RSIL (Reduced Smart Intermediate Language), IoT Devices, Virtual Machine, Interpreter*

1. Introduction

The design of the ISA has significant impacts on various features of the VM such as the structure of interpreter, memory model, system requirements of execution for VM, and *etc*. Especially, a very small sized ISA is required to operate the virtual machine on the low-performance device such as IoT devices. However, if the ISA size is too small, then it makes execution performance down of the virtual machine, and it is difficult to improve a performance through optimization. Thus, the design of the ISA with sufficiently small size to execute on low computing powered devices and optimization codes inclusion to enhance the performance is a very important task.

This paper deals with the intermediate language for the light-weighted virtual machine. The virtual machine is a software processor that has many advantages on software development, release, maintenance and so on by its platform independent features. Nevertheless, in execution performance aspect, it also has a significant disadvantage that restricted low performance by its execution overhead for the software level interpretation. Moreover, considered IoT devices as a target operating hardware of the virtual machine have very restricted computing power and resources.

In this paper, we propose the RSIL (Reduced Smart Intermediate Language) that improved from the previous SIL (Smart Intermediate Language) [1-7] with byte-level intermediate opcode to resolve two issues; restricted resources of IoT devices and a large number of optimization specific codes.

Received (October 7, 2017), Review Result (December 1, 2017), Accepted (December 5, 2017)
* Corresponding Author

ISSN: 2005-4297 IJCA

2. Related Studies

2.1. The IoT-Cloud Fusion Virtual Machine

The IoT-Cloud fusion virtual machine is a stack based virtual machine solution with cloud offloading technology, loaded on IoT devices, which allows dynamic application programs to be downloaded and run platform independently with high computing performance that produces by cloud system. The goals of the IoT-Cloud fusion virtual machine system are enhancing the low computing powered IoT devices' performance and make them more smartened devices. Figure 1 shows a system configuration of the IoT-Cloud fusion virtual machine system. The IoT-Cloud virtual machine is designed lighter than previous SVM (Smart Virtual Machine) [1, 2] and HTML5 SVM [3, 4].

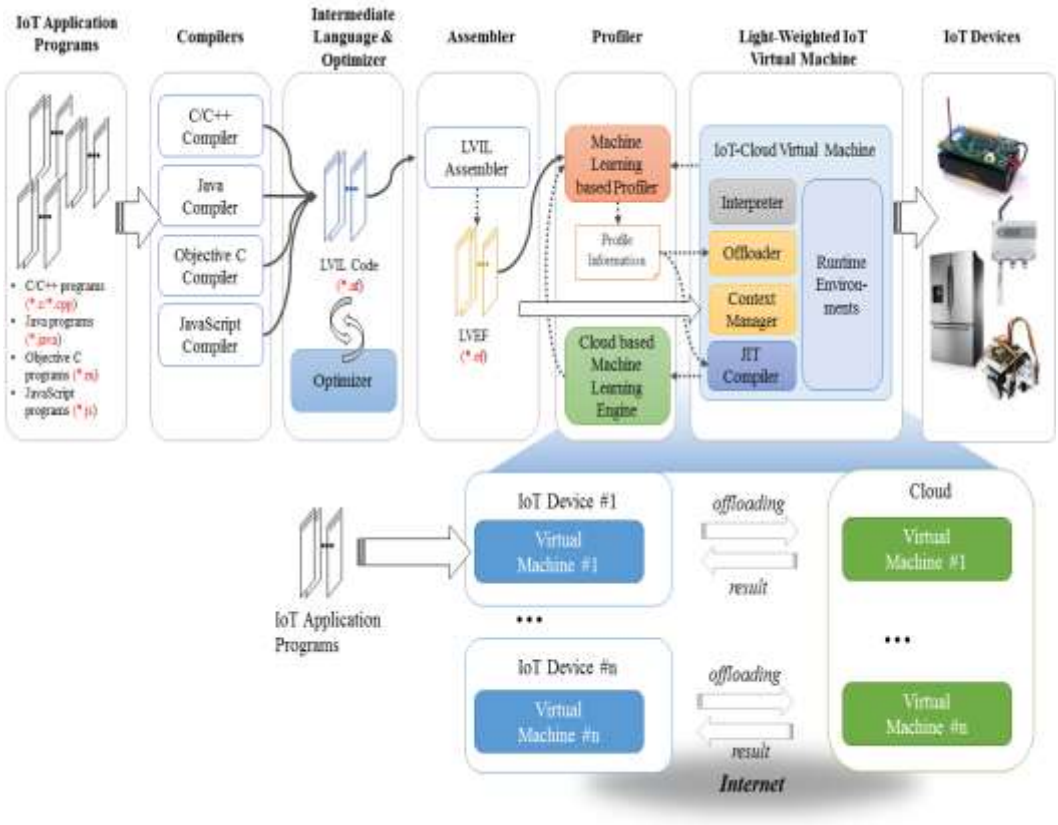

Figure 1. Structure of the IoT-Cloud Fusion Virtual Machine

2.2. Light-Weighted IoT Virtual Machine

The light-weighted IoT virtual machine is a stack-based virtual machine that was designed to execute on the low computing powered IoT devices by the cloud-based offloading method [4,7-11]. Moreover, it was designed to be very small computing resources required for target devices that have restricted resources. The system configuration of the light-weighted IoT virtual machine composite by execution layer, offloading layer, runtime environment layer, portable library layer, and JIT (Just-In-Time) compiler as shown in Figure 2.

The execution aspect, the executing layer is the main component set. The layer consists of three sub-modules; executable code loader, profile loader and interpreter. The executable code loader loads and checks the executable files that generated by the assembler, and then puts the loaded image of executable file into the memory to be

executed on the interpreter. The profile loader reads the profiling information that was generated from profiler for finding offloading points of the given application. The interpreter, as a core module to execute given applications, patches the RSIL opcode sequentially from a code section of the loaded virtual machine execution file image. The occurred data while on execution state is stored and managed on stack and heap by memory manager automatically.

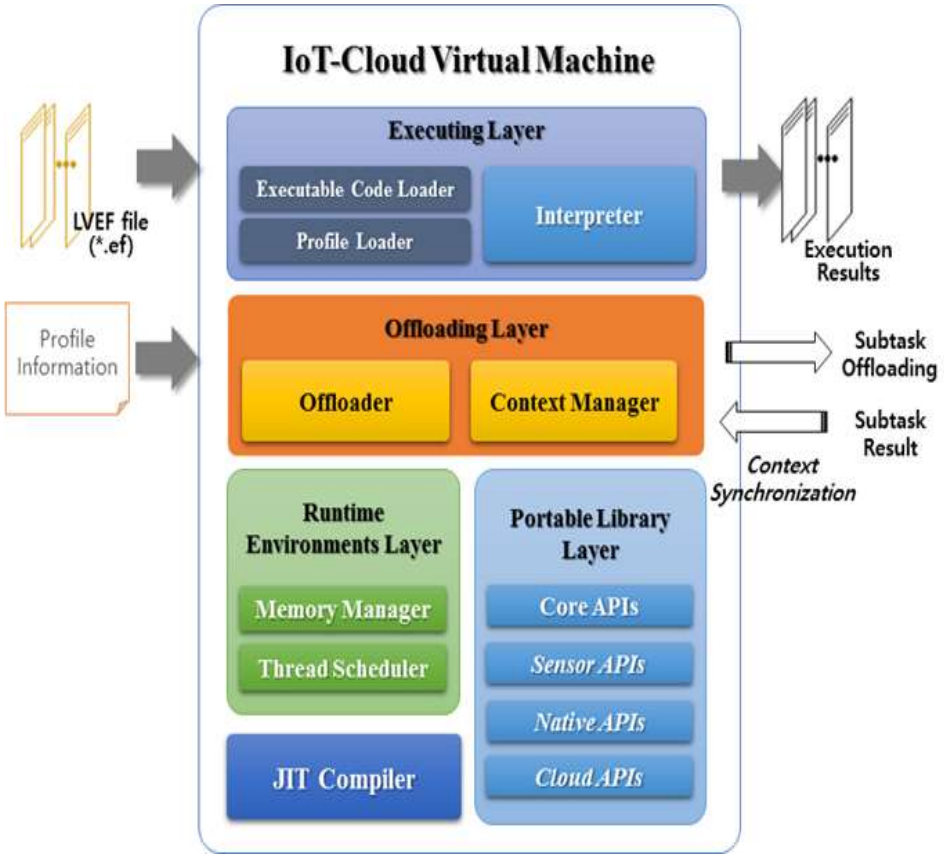

Figure 2. System Configuration of the IoT-Cloud Virtual Machine

2.3. SIL (Smart Intermediate Language)

SIL, the virtual machine code for SVM, is designed as a standardized virtual ma-chine code model for ordinary smart devices and embedded systems [1,5]. SIL is a stack based command set which holds independence as a language, hardware and a plat-form. In order to accommodate a variety of programming languages, SIL is defined based on the analysis of existing virtual machine codes such as bytecode, .NET IL and *etc*. In addition, it also has the set of arithmetic operations codes to cover procedural programming languages and object-oriented languages.

SIL is composed of a meta-code which carries out particular jobs such as class creation and an operation code with responds to actual commands. An operation code has an abstract form which is not subordinated to specific hardware or source languages. It is defined in mnemonic to heighten readability and applies a consistent name rule to make debugging in assembly language levels easier. In addition, it has a short form operation code for optimization. SIL has 6 groups (except optimization group) of operation codes and Figure 3 shows the category of SIL operation codes.

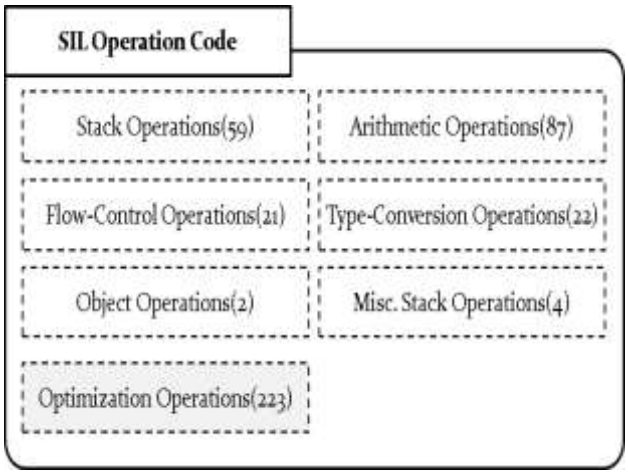

Figure 3. Category of SIL Operation Codes

3. Reduced Smart Intermediate Language: Virtual Machine Code for IoT Devices

A virtual machine is a conceptual computer with a logical system configuration, made of software unlike physical systems made of hardware. Use of virtual machine technology does not require modification of application programs through processors or operating systems are changed. Typical virtual machines include JVM (Java Virtual Machine) that executes Java bytecode.

RSIL (Reduced Smart Intermediate Language) is an intermediate language for IoT-Cloud VM. RSIL is an intermediate language for VM code that is designed for low computing powered IoT devices. It has a set of stack based operation code. Programs composed of LVIL(Light-weighted Virtual machine Intermediate Language) code are converted to executable program by the assembler. In this paper, the RSIL is designed with two criteria – low computing power required, optimization – as an intermediate language of IoT virtual machine from the SIL.

The SIL was designed for the stack-based virtual machine, but it has a restriction that the number of instructions is over the 1-byte level representation because it includes so many optimization codes for performance enhancement. On the other hand, in the case of IoT devices, a large-scale instruction set – over 1-byte level – as a virtual machine code is too bigger than its system performance; low CPU power, small memory and etc. So the virtual machine code of IoT devices should be redesigned from PC/smart device level virtual machine code, and reduce the number of code to 1-byte level representation. Table 1 and Table 2 show the categorized instruction information of the RSIL for IoT devices' virtual machine.

Table 1. Virtual Machine Code Category Information of RSIL

Operator Category	Counts
Stack manipulation, Arithmetic operator	135
Flow control, Type conversion	36
Core Optimization	17
Total (general instruction set)	**188**

Table 2. Virtual Machine Code Category Information of RSIL (Extended Codes)

Operator Category	Counts
Extended Optimization	50
Reserved	17
Second Level Extended Optimization	203
Total	**270**

The criteria of elimination to define the RSIL from the previous SIL instruction set are follows:

1) Instructions that are only reserved but not used in a current stage.
2) Instructions that rarely used and have replaceable code(s) or code pattern(s).
3) Instructions that have upper compatibility in type coercion operator.
4) Instructions that rarely used in optimization code category.

We have defined the RSIL by decreasing the number of the SIL instruction set to 188 using upper criteria, and the size of virtual machine instruction set for IoT devices can be reduced as 1byte level. Also, optimization codes that specially designed for IoT devices could be added on the RSIL's unused remind instruction space.

RSIL is composed of meta-code (shows class declarations and specific operations) and arithmetic codes (responds to actual commands). Arithmetic codes are not subordinate to any specific hardware or source languages and thus have an abstract form. In order to make debugging of the languages such as the assembly language simple, they apply a name rule with consistency and define the language in mnemonics, for higher readability.

The major pseudo codes of RSIL in Table 3 means that RSIL is segmented by functions and they are represented in the code section. Each function has its own function name, parameter information and opcode list denoted by mnemonics. The mnemonic of pseudo codes improved readability by using the keyword on the level of source codes as it is, and defined a total of indicators such as function.

Table 3. Virtual Machine Code Category Information of RSIL

Pseudo Code	Description
%%CodeSectionStart	Section flag for starting code area.
%FunctionStart	Section flag for starting function area.
.func_name	Describe the function name.
.param_count	Describe the number of parameters for target function.
.opcode_start	Declares opcode list for corresponding function.
.opcode_end	Declares end of opcode list for the corresponding function.
%FunctionEnd	Section flag for finishing function area.
%%CodeSectionEnd	Section flag for finishing code area

4. Experimental Results

Our previous virtual machines for smart devices and HTML5 platform are used 2bytes length opcode with variable length arguments for handling many

optimization opcodes [1, 4]. Thus, the required memory size for loading and processing the program code is larger than the virtual machine that has an only 1byte-length opcode, and it is one of the major reason that inefficient executional performance of the virtual machine using low computing powered IoT devices as a host target hardware platform. However, by the method - reducing the number of instructions, it is difficult to solve the performance issue. Because so many optimization opcodes are removed by the reducing method, and it is another major issue to enhancing the performance of the virtual machine that case of not using the native code but software level virtual machine code executing.

In this paper, we verify the RISL's opcode and metadata by generating RSIL as the target intermediate code for the C / C ++ compiler [3, 4]. Tables 4 and 5 show source programs for N-Queen problem and Prime number with generated the RSIL programs.

Table 4. Experimental Example Code (N-Queen Problem)

Test Source File (N-Queen)	Generated Intermediate Code

#include "sys_lib.h"
 	
const int MAX = 1000;
 	
int perfect(int max) {
int i, j, k;
int rem, sum;
int total_perfect = 0;
i = 2;
while (i <= max) {
sum = 0;
k = i / 2;
j = 1;
while (j <= k) {
rem = i % j;
if (rem == 0)
sum += j;
++j;
}
… omitted …
	%%CodeSectionStart
	%FunctionStart
	.func_name &perfect
	.func_type 2
	.param_count 1
	.opcode_start
	proc 28 1 1
	str.i 1 0
	ldc.i 0
	str.i 1 24
	ldc.i 2
	str.i 1 4
	%Label ##0
	lod.i 1 4
	lod.i 1 0
	le.i
	fjp ##1
	
	… omitted …

Table 5. Experimental Example Code (Prime number)

Test Source File (N-Queen)	Generated Intermediate Code

#include "sys_lib.h"
 	
const int N = 4000;
 	
int prime(int n) {
int i,j;
int prime;
int total_prime = 0;
 	
for (i=2; i <= n; ++i) {
prime = 1;
j = 2;
while (j <= i/2) {
if (i % j == 0) {
prime = 0;
break;
}
++j;
}
if (prime) {
total_prime++;
}
}
 	
return total_prime;
}
… omitted …
	%%CodeSectionStart
	%FunctionStart
	.func_name &prime
	.func_type 2
	.param_count 1
	.opcode_start
	proc 20 1 1
	str.i 1 0
	ldc.i 0
	str.i 1 16
	ldc.i 2
	str.i 1 4
	%Label ##0
	lod.i 1 4
	lod.i 1 0
	le.i
	fjp ##1
	ldc.i 1
	str.i 1 12
	ldc.i 2
	str.i 1 8
	%Label ##3
	lod.i 1 8
	lod.i 1 4
	ldc.i 2
	div.i
	le.i
	… omitted …

5. Conclusions and Further Researches

The IoT-Cloud fusion virtual machine system will be developed to enhance the low computing powered IoT devices' performance and make them more smartened devices. While implementing virtual machines, the design of ISA that is a very important phase because it has significant impacts on various features of the VM. The IoT system required a small sized ISA for its restricted computing resources. In this paper, we have defined the RSIL as 1-byte level ISA by decreasing the number of the SIL instruction set from the smart cross platform. Also, we expect that two issues of designing IL can be solved by the proposed RSIL.

In further researches, an assembly format and an executable format will be defined by applying the features of RSIL. And they also will be reduced the structure to suitable on restricted computing powered devices.

Acknowledgments

This research was supported by Seokyeong University in 2016.

References

[1] Y. S. Lee and Y. S. Son, "A Study on the Smart Virtual Machine for Executing Virtual Machine Codes on Smart Platforms", International Journal of Smart Home, SERSC Australia, vol. 6, no. 4, **(2012)**, pp. 93-105.

[2] Y. S. Lee and Y. S. Son, "A Study on the Smart Virtual Machine for Smart Devices", Information -an International Interdisciplinary Journal, International Information Institute, Japan, vol. 16, no. 2, **(2013)**, pp. 1465-1472.
[3] Y. S. Lee, S. M. Oh and Y. S. Son, "Design and Implementation of HTML5 based SVM for Integrating Runtime of Smart Devices and Web Environments", International Journal of Smart Home, SERSC Australia, vol. 8, no. 3, **(2014)**, pp. 223-234.
[4] Y. S. Lee, J. Joeng and Y. Son, "Design and implementation of the Secure Compiler and Virtual Machine for Developing Secure IoT Services", Future Generation Computer Systems, Elsevier, Netherland, vol. 76, **(2017)** November, pp. 350-357.
[5] Y. Son, J. H. Kim and Y. S. Lee, "A Study on the Platform Independent SIL Code based Compiler for Smart Virtual Machine", Advanced Science and Technology Letters (Multimedia 2014), vol. 46, **(2014)**, pp. 79-82.
[6] S. M. Han, Y. Son and Y. S. Lee, "Design and Implementation of the Smart Virtual Machine for Smart Cross Platform", Journal of Korea Multimedia Society, vol. 16, no. 2, **(2013)**, pp. 190-197.
[7] Y. Son and Y. S. Lee, "A Study of the Virtual Machine Code for IoT Devices", Asia-pacific Proceedings of Applied Science and Engineering for Better Human Life, vol. 10, **(2016)**, pp. 109-112.
[8] K. Yang, S. Ou, and H. H. Chen, "On Effective Offloading Services for Resource-Constrained Mobile Devices Running Heavier Mobile Internet Applications", IEEE Comm. Magazine, vol. 46, no. 1, **(2008)**, pp. 56-63.
[9] K. Kumar, "A Survey of Computation Offloading for Mobile Systems", Mobile Networks and Applications, vol. 18, no. 1, **(2013)**, pp. 129-140.
[10] J. E. Smith and R. Nair, "Virtual Machines", Morgan Kaufmann, **(2005)**.
[11] J. Meyer and T. Downing, "Java Virtual Machine", O'REYLLY, **(1997)**.

Authors

JaeHyun Kim, he received the B.S. degree from the Dept. of Mathematics, Hanyang University, Seoul, Korea, in 1986, and M.S. and Ph.D. degrees from Dept. of Statistics, Dongguk University, Seoul, Korea in 1989 and 1996, respectively. He was a chairman of Dept. of Internet Information 2002-2007. Currently, he is a member of the Korean Data & Information Science Society and a Professor of Dept. of Computer Engineering, Seokyeong University, Seoul, Korea. His research areas include mobile programming, cloud system and big data analysis.

Yangsun Lee, he received the B.S. degree from the Dept. of Computer Science, Dongguk University, Seoul, Korea, in 1985, and M.S. and Ph.D. degrees from Dept. of Computer Engineering, Dongguk University, Seoul, Korea in 1987 and 2003, respectively. He was a Manager of the Computer Center, Seokyeong University from 1996-2000, a Director of Korea Multimedia Society from 2004-2005, a General Director of Korea Multimedia Society from 2005-2006, a Vice President of Korea Multimedia Society in 2009, and a Senior Vice President of Korea Multimedia Society in 2015-2016. Also, he was a Director of Korea Information Processing Society from 2006-2014 and a President of a Society for the Study of Game at Korea Information Processing Society from 2006-2010. And, he was a Director of HSST from 2014-2017. Currently, he is a Professor of Dept. of Computer Engineering, Seokyeong University, Seoul, Korea. His research areas include smart system solutions, programming languages, and embedded systems.

