*E*mbedded *J*ava *V*irtual *M*achine

Ahmed Hosni Ramadan Shaaban
Ahmed Muhammad Seddiq
Eslam Ahmed Elmorshdy
Khaled Ibrahim Almahallawy
Sameh Ismail Abd Elrahman

June 25, 2006

Contents

Chapter 1

Introduction

testing the citations[1]

Connected devices[1] are getting deeply involved in the daily routine. Unfortunately they exist in a very wide diversity slowing down programmers' innovations and programms' delivery time. Thus, providing a hardware neutral execution enviroment, like Java^{TM}virtual machine, with acceptable performance shall benefit the developers as well as the consumers. For developers, it would break the burden of targetting wide variety of platforms letting them focus on the application. For consumers, it would widen the existing software packages to choose from and no one is left behind with out dated devices.

1.1 Proposed project

The aim of our project is to design and implement a light-weight interpreter-based Java virtual machine for an embedded environment that needs to handle carefully its processing resources (power consumption and limited memory) and provides a secure sand box execution environment.

1.2 Organization of chapters

This section describes the organization of the rest of the documentation.

- Chapter 2 "Background" contains information about the embedded systems and their architecture variety. Also, it contains a description of the JVM (Java Virtual Machine).
- Chapter 3 "JVM Architecture" describes the architecture of the JVM using UML notation.
- Chapter 4 "Design and Implementation of the EJVM" can be considered an extension of the previous chapter. It will include description of the main modules and flowcharts or pseudo code for advanced algorithms used.
- Chapter 5 "Tools" contains a description of the tools used during development.

[1] PDAs, Pagers, iPods...

- Chapter 6 "Relevant Works" introduces some relevant works. It introduces two opensource Java virtual machines, jamVM and Kafee, also a previous graduation project that was about implementing a Java Card Virtual Machine.
- Chapter 7 "Performance Evaluation" contains the results of tests performed on our system and comparisons between our system and other existing applications.
- Chapter 8 "Conclusions and Future Work" we describe the future work needed to extend our project and enumerate any limitation in our design or implementation.
- appendices

Chapter 2

Background

This chapter gives an overview about the general virtual machine, defines both tha Java programming language and the Java Virtual Machine. The chapter also contains information about embedded systems and the ARM [1] architecture that is the commonly used in embedded systems.

2.1 What is a Virtual Machine?

Virtual Machine, in general, is a software abstraction of a computer that often executes as a user application on top of the native operating system. One application of virtual machines is to allow multiple instances of an operating system to execute concurrently. Another is emulation using software or hardware that mimics the functionality of hardware or software not present in the system. In other words it promotes portability by decoupling the relation between program execution and the host environment or hardware design or both. Virtual machines interface with the hardware in a system via the underlying operating system, other user programs can interact with VMs. A VM can create software components that represent the content of physical systems such as processors, memory, communication channels, disks and clocks (see Figure 2.1).

Early examples are the IBM VM, and Pascal p-Code machine. The VM was apparently the first true virtual machine system. It introduced the radical concept of a self-virtualizing processor instruction set. Essentially VM and the mainframe hardware cooperate so that multiple instances of any operating system, each with protected access to the full instruction set, can peacefully and concurrently coexist. This virtualization capability is so strong that VM can run as a guest inside itself even multiple levels deep without much performance penalty. The p-Code machine or pseudo-code machine [2] was the target of some early Pascal implementations. Pascal was translated not to machine code instructions (understandable directly to a processor) but to p-code instructions. To execute the program, another program is used that interprets this code.

[1] Advanced RISC Machines

[2] A p-code is similar to a byte-code (mentioned later) but a p-code works at a higher level. Whereas byte-codes work at a very low level similar to machine code, p-codes can perform moderately complex tasks such as printing a message or clearing the screen.

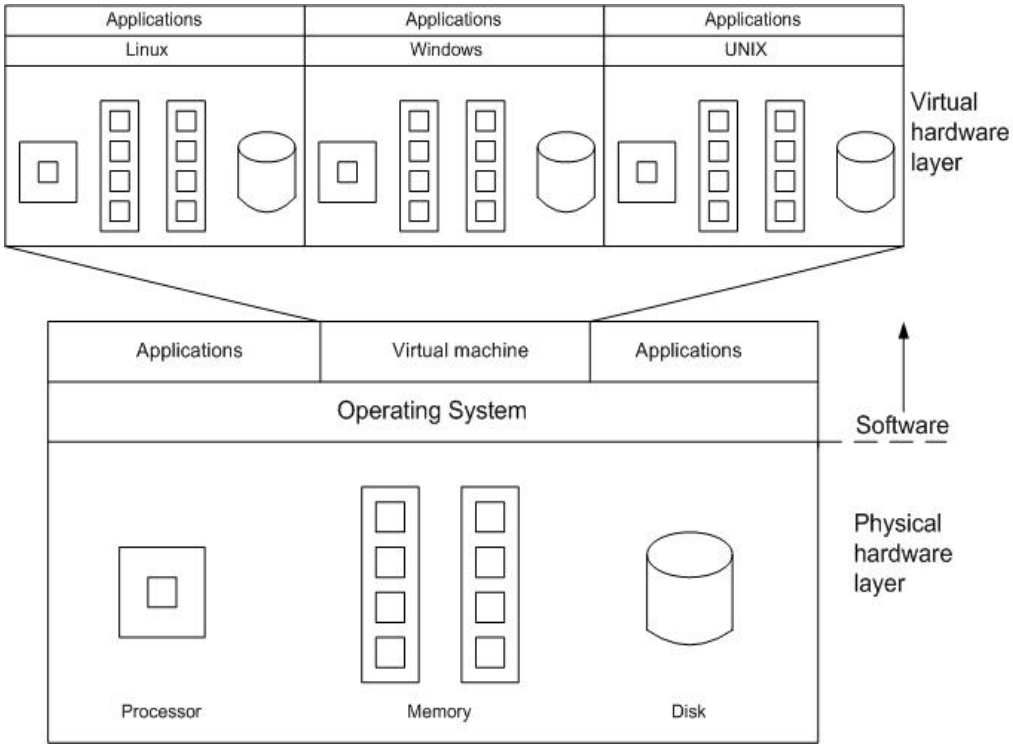

Figure 2.1: Schematic of a Virtual Machine

2.2 What is Java?

Java^{TM}Technology Introduced By Sun [3] Microsystems, Inc with attractive Write Once, Run Anywhere concept where Java program is compiled into machine independent byte-code which is simple stack-based (i.e. not register-based) low level instructions in order to be easily interpreted by almost any hardware platform. Thus Java is a choice where heterogeneous platforms exist and need to communicate, which is the case for the World. Java faced criticism for performance [4], memory consumption, portability issues [5], and not being FOSS (Free Open Source Software) [6].Though, it has steadily evolved for several reasons first Processors having bursts of performance acceleration second, Memory getting faster and cheaper, allowing fast complex Technologies - like the HotspotTM- or JIT [7] to be adoptable.

[3]Stanford University Network

[4]indeed optimized java code run by factor of 5 slower than corresponding C++ code. The C++ Complete Refrence, Osborne

[5]need for native calls still exist but with every one following suns notation of the API it could be considered portable or part of the standard

[6]Java virtual machine standard is feely available but all implementation issues are proprietary licensed software

[7]Just In Time Compiler

2.3 What is the Java Virtual Machine (JVM)

The Java virtual machine (JVM) is an abstract computer or a machine within a machine. Its specification defines certain features every Java virtual machine must have, but leaves many choices to the designers of each implementation. The main job of the Java virtual machine is to execute Java bytecodes. The Java virtual machine specifications did not specify or restrict a certain execution technique that the Java virtual machine must follow. The flexible nature of the Java virtual machine's specification enables it to be implemented on a wide variety of computers and devices.

2.4 The Class File and the Byte Code

2.5 Embedded Systems

Embedded Systems provide a different software design challenge. They are characterized by a small set of specialized resources that provide functionality to devices such as cell phones and PDAs (Personal Digital Assistant). In embedded environments, efficient resource management is the key to building successful software. One of the most important software design limitations that embedded systems provide is the storage limitation, the software must provides services using a minimal amount of code. Considerations such as power management and the need for user-friendly interfaces create other challenges in embedded software design.

2.6 ARM Architecture

ARM Architecture (Originally the Acorn RISC Machine currently Advanced RISC Machines) has become one of the most used CPU designs in the world, found in everything from hard drives, to mobile phones, to routers, to calculators, to childrens toys. Today it accounts for over 75% of all 32-bit embedded CPUs. That is for its low cost yet high performance advantage. ARM family reached ARM11 and is licensed to many companies IBM, Infineon Technologies, Texas Instruments, Nintendo, Philips, VLSI, Atmel, Sharp and Samsung.

2.7 Conclusions

The existence of the Java virtual machine is considered an important matter nowadays; especially for the embedded systems that are found in different architectures. The next chapter introduces the Java virtual machine's architecture in terms of its modules and subsystems.

Chapter 3

JVM Architecture

This chapter decribes the architecture of the Java virtual machine by enumerating its modules and describing them using the UML notation. This architecture is stated in the Java virtual machine specifications.

3.1 The JVM main modules

3.1.1 The Loader

3.1.2 The Linker

3.1.3 The Memory Management subsystem

3.1.4 The Execution Engine

3.1.5 The Exception handler subsystem

3.1.6 The Java Native Interface (JNI)

3.2 Conclusions

This chapter gave a description of a typical Java virtual machine and stated its modules and their functions. The next chapter contains the implementation details of the EJVM that is a subset of the JVM.

Chapter 4

Design and Implementation of the EJVM

This chapter contains the detailed specifications of the EJVM modules, a class diagram of the whole system and a detailed description of each module. For each module, a class diagram and sequence diagrams are introduced. Finally the chapter introduces any design and implementation decisions.

4.1 The class diagram of the EJVM

4.2 The Loader

4.3 The Linker

4.4 The Memory Management subsystem

4.5 The Execution Engine

4.6 The Exception handler subsystem

4.7 The Java Native Interface (JNI)

4.8 Conclusions

This chapter gave the implementation details and the design decisions of the EJVM. The next chapter is about the tools used during the development phase of the EJVM.

Chapter 5

Tools

Chapter 6

Relevant Works

This chapter introduces some relevant works. It introduces the "Java Card Virtual Machine" JCVM , a graduation project that was proposed in 2004. Some OpenSource Java virtual machines are introduced in this chapter like jamVM and Kafee.

Chapter 7

Performance Evaluation

This chapter contains the results of tests performed on our system and comparisons between our system and other existing applications.

7.1 Conclusions

Chapter 8

Conclusions and Future Work

References

[1] Tim Lindholm and Frank Yellin. *The Java Virtual Machine Specification (2nd Edition)*. Addison-Wesley Professional, 1999.