제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

가상 기계를 위한 실행 파일 포맷

정한종*, 윤성림, 오세만
동국대학교 컴퓨터공학과
e-mail : {prana*, yslhappy, smoh}@dongguk.edu

An Executable File Format for Virtual Machine

Hanjong Cheong*, Sunglim Yum, Seman Oh
Dept. of Computer Engineering, Dongguk University

요 약

가상 기계 개념은 목적 기계에 영향을 받지 않는 컴파일러의 중간언어로부터 시작되었다. 즉, 기존에는 실행 프로그램이 하드웨어와 운영체제에 종속적이었으나 가상 기계는 플랫폼 독립을 가능하게 한다. 임베디드 시스템이란 전용 동작을 수행하거나 또는 특정 임베디드 소프트웨어 응용 프로그램과 함께 사용되도록 디자인된 특정 컴퓨터 시스템 또는 컴퓨팅 장치를 말한다. 임베디드 시스템을 위한 가상 기계 기술은 모바일 장치와 디지털-TV 등에 탑재할 수 있는 핵심 기술과 다운로드 솔루션을 이용한 동적인 실행 기술이 요구된다. 또한 컨텐츠 개발을 쉽게 하기 위해서 다양한 언어를 지원하고 언어들 간의 통합이 가능하다.

본 논문에서는 클래스 파일 포맷, PE 파일 포맷 등 기존의 가상 기계를 위한 파일 포맷들의 분석을 기반으로 하여 임베디드 시스템을 위한 실행 파일 포맷인 EVM 파일 포맷을 제안한다. EVM 파일 포맷은 언어 통합을 지원하고 구조가 간결하며 확장이 용이한 특징을 지닌다. 또한 메타데이터와 중간언어(SIL)가 서로 독립적으로 구성되어 분석이 쉽고 타입 체크가 편리한 구조이다.

1. 서론

가상 기계 개념은 목적 기계에 영향을 받지 않는 컴파일러로부터 시작되었다. 즉, 기존에는 프로그램이 하드웨어와 운영체제에 종속적이었으나 가상 기계는 플랫폼 독립을 가능하게 한다. 따라서, 가상 기계를 대상으로 중간 코드를 생성하기 때문에 중간 코드 생성이 용이하다. 대표적인 가상 기계로서 클래스 파일을 입력으로 받는 JVM(Java Virtual Machine)이 있다. 우리 나라에는 모바일 장치에 이식이 가능한 GVM(General Virtual Machine)이 있다[7].

임베디드 시스템이란 전용 동작을 수행하거나 또는 특정 임베디드 소프트웨어 응용 프로그램과 함께 사용되도록 디자인된 특정 컴퓨터 시스템 또는 컴퓨팅 장치를 말한다. 임베디드 시스템은 ROM 기반 운영체제나 디스크 기반 시스템(예: PC)을 사용할 수 있지만, 범용 컴퓨터 또는 장치를 상업적으로 대체하여 사용할 수 없다. 이 시스템은 냉장고, 세탁기, 핸드폰, PDA, 홈 관리 시스템, 디지털-TV 등 여러 분야에 응용할 수 있다. 임베디드 시스템을 위한 가상 기계 기술은 모바일 장치와 디지털-TV 등에 탑재할 수 있는 핵심 기술과 다운로드 솔루션을 이용한 동적인 실행 기술이 요구된다. 또한 컨텐츠 개발을 쉽게 하기 위해서 다양한 언어를 지원하고 언어들 간의 통합이 요구된다. 각각의 가상 기계는 입력으로 받는 실행 파일 포맷이 있다. JVM 의 클래스 파일 포맷, .NET CLR 의 PE 파일 포맷, GVM 의 SGS 파일 포맷, EVM 의 EVM 파일 포맷 등이 있다.

현재 Microsoft 의 C#언어와 SUN 사의 자바언어 등을 모두 수용할 수 있는 가상기계에 대한 연구가 진행중이다. EVM(Embedded Virtual Machine)이라 명명된 이 가상기계 솔루션은 C#, 자바 등 객체지향 언어뿐만 아니라 C 언어와 같이 순차적인 언어로 작성된 프로그램들을 어셈블리 언어 형태인 *. sil 을 거쳐서 EVM 파일 포맷으로 변환하여 임베디드 시스템에 탑재된 가상기계에서 실행할 수 있도록 한다.

이와 같은 프로젝트의 일환으로 본 논문에서는 클

본 연구는 한국과학재단 목적기초연구(R01-2002-000-00041-0)지원으로 수행되었음.

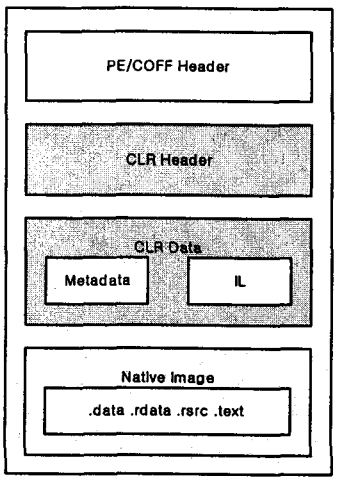
래스 파일 포맷, PE 파일 포맷 등 기존의 가상 기계를 위한 파일 포맷들의 분석을 기반으로 하여 임베디드 시스템을 위한 실행 파일 포맷인 EVM 파일 포맷을 제안한다.

본 논문의 2 장에서는 기존의 가상 기계 파일 포맷과 EVM에 대한 관련 연구를 소개한다. 3 장에서는 기존 파일 포맷의 단점을 보안한 새로운 형태의 EVM 파일 포맷을 제안한다. 마지막으로 4 장에서는 본 연구의 결론과 향후 연구 과제에 대해서 기술한다.

2. 관련 연구

2.1 PE 파일 포맷

.NET 실행 파일은 마이크로소프트 PE(Portable Executable)와 COFF(Common Object File Format)의 기준을 토대로 확장한 형태로써, CLR(Common Language Runtime) 환경에서 동작하기 위한 메타데이터와 IL 을 포함하고 있다. .NET 실행 파일은 기존에 실행 파일과 동일하게 확장자가 *.exe 또는 *.dll 이며 스몰 엔디언(Small-Endian)의 순서로 저장된다. [그림 1]은 .NET PE 파일 포맷을 네 부분으로 나누어 표현한 형태이다.



[그림 1] PE 파일 구조도

PE/COFF 헤더는 운영체제와 파일의 속성 정보 등을 가진다. CLR 헤더는 PE 파일이 .NET 실행 파일임을 나타내는 정보들을 갖는다. CLR 데이터는 프로그램이 어떻게 실행될지를 결정하는 정보들로서 메타데이터와 IL 코드로 나누어진다. 메타데이터는 모듈에서 참조 또는 선언된 아이템(클래스와 멤버)들의 설명 체계로써 타입 정의, 버전 정보, 외부 어셈블리 참조, 그리고 다른 표준화된 정보로 이루어져 있다. 즉 아이템들의 속성, 특징, 그리고 관계들을 말한다. IL 은 CLR 환경에서 실행되는 명령어들의 집합이다. Native Image 부분은 imports/exports 가 포함된 데이터, 코드, 그리고 헤더들에 의해 묘사된 실질적인 데이터를 포함한다. 섹션 종류로는 data section, relocation section, unmanaged resource section, thread local storage data section 등이 있다[2].

2.2 클래스 파일 포맷

클래스 파일은 자바 소스 코드가 자바 컴파일러에 의해 생성된 파일로 확장자가 *. class 이며 자바 가상 기계의 입력이 되어 실행된다. 클래스 파일은 8 비트 단위의 스트림으로 구성되어 있다. 16 비트, 32 비트, 64 비트 크기를 가진 데이터들은 8 비트 단위로 나누어져 높은 비트가 먼저 나오는 빅 엔디언(Big-Endian)의 순서로 저장된다. [표 1]은 클래스파일의 구조를 C 언어의 구조체 형태로 나타낸 것이다.

```
ClassFile {
  u4         magic;
  u2         minor_version;
  u2         major_version;
  u2         constant_pool_count;
  cp_info      constant_pool[constant_pool_count-1];
  u2         access_flags;
  u2         this_class;
  u2         super_class;
  u2         interfaces_count;
  u2         fields_count;
  field_info     fields[fields_count];
  u2         methods_count;
  method_info      methods[methods_count];
  u2         attributes_count;
  attribute_info  attributes[attributes_count];
}
```

[표 1] 클래스 파일의 형식

magic 은 자바 클래스 파일임을 나타내 주는 식별자로 항상 0xCAFEBABE 에 값을 가진다. minor_version 과 major_version 은 클래스 파일의 주 버전과 부 버전을 나타낸다. constant_pool_count 는 상수 풀의 개수를 나타내며, 항상 0 보다 큰 값을 가진다. constant_pool 은 가변 크기를 갖는 상수 풀을 표현한 것으로 배열의 형태이며 클래스, 슈퍼클래스, 메소드, 그리고 필드의 이름 등에 대한 스트링 값을 가진다. access_flags 는 현재 클래스 파일이 나타내는 클래스의 접근 권한을 나타낸다. this_class 는 클래스 파일이 포함하고 있는 클래스의 정보를 표시하기 위해서 위에 상수 풀의 인덱스를 가지고 있다. super_class 는 이 클래스 파일의 슈퍼클래스 정보를 표시하며 이것 또한 상수 풀의 인덱스를 가진다. interfaces_count, interfaces 는 인터페이스 필드 내에 위치한 인터페이스 정보 개수, 실제 인터페이스 정보를 가진 상수 풀의 인덱스를 가진 배열 형태이다. fields_count, fields 는 클래스의 필드의 개수, 실제 필드의 정보를 나타내며 상수 풀의 인덱스를 가진다. methods_count, methods 는 클래스의 메소드 개수, 메소드의 실제 정보를 나타내며 상수 풀의 인덱스를 가진다. attributes_count, attributes 는 클래

스의 속성의 개수, 클래스 속성 정보를 가진다[6].

2.3 PE 파일 포맷과 클래스 파일 포맷 비교

PE 파일 포맷과 클래스 파일 포맷은 가상 기계를 위한 파일 포맷이다. 그러나 많은 부분이 다르게 구성되어 있다. 클래스 파일 포맷은 하나의 클래스 또는 인터페이스에 대해 구성되고 PE 파일 포맷은 여러 개의 클래스로 구성된다. 파일 내에서 데이터를 참조하는 방식도 다르다. 클래스 파일 포맷은 상수 풀의 인덱스를 이용하고 PE 파일 포맷은 offset 과 size 를 이용한다. 파일 내에 저장되는 바이트의 순서도 서로 다른 방식을 취하고 있다. 클래스 파일 포맷의 경우 little-endian, PE 파일 포맷의 경우 big-endian 을 사용한다. 링킹과 로딩 시 클래스 파일 포맷의 경우 한 개의 클래스로 구성되어 있기 때문에 관련된 모든 클래스 또는 인터페이스를 로드 함에 있어 많은 시간이 소요된다. PE 파일 포맷의 경우 메타데이터와 실행 코드인 IL 를 분리함으로써 필요한 클래스만 로딩이 가능하다. 클래스 파일 포맷의 메타데이터는 상수 풀이라는 부분에 저장 되어 있으나 자세한 정보를 보기 위해서는 파일 전체에 나누어져 저장된 정보를 검색해야 한다. PE 파일 포맷은 파일 내부의 특정 부분에 모든 메타데이터 정보가 저장됨으로써 그 특정 부분만 검색하면 된다. [표 2]는 위에 내용을 표로 도시한 것이다.

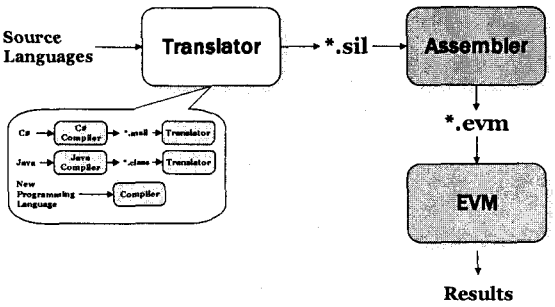
	PE 파일	클래스 파일
클래스	한 개 또는 그 이상	한 개
참조	offset	Index
바이트 순서	little-endian	big-endian

[표 2] PE 파일과 클래스 파일 비교

2.4 EVM (Embedded Virtual Machine)

EVM 은 모바일 장치, 셋톱 박스, 디지털-TV 등에 탑재되어 동적 응용프로그램을 다운로드 하여 실행할 수 있는 가상기계 솔루션이다. 또한 컨텐츠 개발을 쉽게 하기 위해서 다양한 언어를 지원하고 언어들 간의 통합이 가능하다.

EVM 은 크게 세 부분으로 나뉘어진다. 첫 번째 부분은 C# 또는 자바 등의 고급 프로그래밍 언어로 작성된 프로그램을 가상 기계를 위한 표준 중간 언어(SIL)로 번역하는 부분이다. 두 번째 부분은 SIL 코드들을 입력으로 하여 가상기계에서 실행 가능 한 형태인 EVM 파일 포맷으로 변환하는 어셈블러 부분이다. 마지막으로 세 번째 부분은 실제 하드웨어에 탑재되어 EVM 파일 포맷을 실행하는 가상기계 부분이다. EVM 의 가상 기계 부분은 계층적인 구조로 설계되어 retargeting 과정의 부담을 최소화 한다. 이와 같은 EVM 의 시스템 구성도를 [그림 2]에 나타내었다.

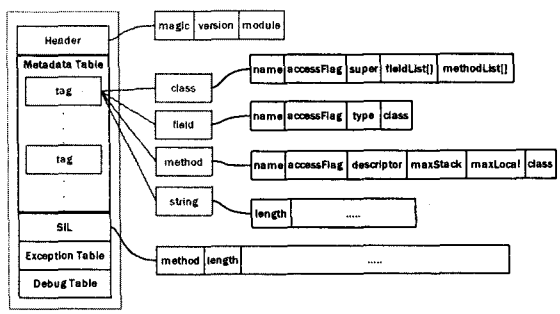


[그림 2] EVM 시스템 구성도

3. EVM 파일 포맷

3.1 EVM 파일 포맷 구조

EVM 을 위한 실행 파일 형식인 EVM 파일 포맷(*.evm)은 가상 기계를 위한 표준 중간언어(*.sil)를 입력으로 받는 assembler 의 출력이다. 이는 다시 EVM 에 입력이 되어 실행 결과를 만들어 낸다. EVM 파일 포맷은 언어간에 통합을 기반으로 설계 되었으며 구조가 간결하고 확장이 용이하다. 또한 메타데이터와 중간언어(SIL)가 분리되어 있어 파일 분석이 쉽고 타입 체크가 편리한 구조이다. [그림 3]은 EVM 파일의 구조를 나타낸 것이다.



[그림 3] EVM 파일 구조도

EVM 파일은 Header, Metadata Table 그리고 SIL 의 세 부분으로 이루어져 있다. Header 부분은 EVM 파일을 나타내는 magic number, 파일 포맷의 버전을 나타내는 version 정보, 그리고 소스파일을 나타내는 module name 으로 구성되어 있다. Metadata Table 부분은 tag 와 각 tag 의 속성들로 이루어져 있고 tag 로는 class, filed, method 그리고 string 이 있다. SIL 부분은 메소드 table index 와 코드의 길이를 나타내는 length, 그

리고 프로그램에 실행 정보를 나타내는 SIL code 들로 구성되어 있다.

3.2 Header

Header 는 파일에 대한 기본 정보를 가진다. 파일 식별을 위한 magic 은 0x0e054d00(4 바이트) 값을 가진다. EVM 버전을 나타내는 version 은 major 버전과 minor 버전이 각 2 바이트로 나누어 저장된다. 소스 프로그램 이름을 나타내는 module 는 Metadata Table index 로 표현 되며 참조 테이블은 string table 이다. 예를 들면 string table 의 네 번째를 참조하게 되면 그 값은 0x400003 이 된다.

3.3 Metadata Table

Metadata Table 은 클래스들과 클래스 멤버들의 속성, 관계들을 나타낸다. 구성은 tag 와 각 tag 의 속성들로 이루어져있다. Tag 의 종류로는 class(10), field(20), method(30), 그리고 string(40)이 있다.

Class(10)의 속성 종류로, name 은 클래스의 이름을 나타내며 string 의 table index 을 갖는다. accessFlag 는 클래스의 접근 권한 정보를 나타내며 1byte 의 bitmask 로 나타낸다. Super 는 클래스의 슈퍼 클래스를 나타내며 class 의 table index 를 갖는다. fieldList 는 클래스의 필드 정보를 배열 형태로 갖는다. fieldList 의 값들은 field 의 table index 이다. 끝으로, methodList 는 클래스의 메소드 정보를 배열 형태로 갖는다. *methodList* 의 값들은 method 의 table index 이다. 클래스가 없는 순차적 언어의 경우 더미 클래스를 생성하여 전역 변수가 필드 되고 함수들이 메소드가 된다.

Field(20)의 속성 종류로, name 은 필드의 이름을 나타내며 sting 의 table index 를 갖는다. accessFlag 는 필드의 접근 권한 정보를 나타내며 1byte 의 bitmask 로 나타낸다. type 은 필드의 타입을 나타내며 class 의 table index 를 갖는다. 이는 타입이 클래스의 인스턴스임을 알 수 있다. class 는 필드가 속한 클래스를 나타내며 class 의 *table index* 를 갖는다.

Method(30)의 속성 종류로, name 은 메소드의 이름을 나타내며 string 의 table index 를 갖는다. accessFlag 는 메소드의 접근 권한 정보를 나타내며 1byte 의 bitmask 로 나타낸다. descriptor 는 메소드의 반환 타입 및 인수를 나타내며 string 의 table index 를 갖는다. class 는 메소드가 속한 클래스를 나타내며 class 의 table index 를 갖는다.

String (40)의 속성 종류로, 스트링의 길이를 나타내는 length 와 length 길이에 따른 스트링이 있다.

Tag 의 순서는 상관 없으며, 필요 시 tag 와 그 필드만 정의하면 확장이 가능하다. table index 는 해당 테이블의 tag 와 테이블의 행 번호로 이루어져 있다. 예로써 메소드의 아홉 번째에 table index 를 나타내면 0x300008 이 된다. 각 테이블 간의 참조는 table index 를 이용한다.

3.4 SIL

SIL 은 프로그램의 실행을 위한 명령어들의 집합이다. 구성으로는 해당 메소드의 table index 와 코드의 길이를 나타내는 length, 그리고 프로그램의 실행 정보를 나타내는 SIL code 들로 구성되어 있다.

메소드 순서와는 상관 없이 SIL 의 구조로 이루어져 있으면 된다.

4. 결론 및 향후 연구

가상 기계 기술은 프로세서나 운영체제 등이 바뀌더라도 응용프로그램을 변경하지 않고 사용할 수 있는 기술이다. 이는 응용 프로그램의 이식성 확보라는 장점을 가진다. 임베디드 시스템을 위한 가상기계 기술은 모바일 디바이스와 디지털-TV 등에 탑재할 수 있는 핵심기술로 다운로드 솔루션에서는 꼭 필요한 소프트웨어 기술이다.

본 논문에서는 기존의 가상 기계를 위한 실행 파일 포맷인 클래스 파일 포맷과 PE 파일 포맷 등을 분석하여, 임베디드 시스템을 위한 실행 파일 포맷인 EVM 파일 포맷을 제안 하였다. EVM 파일 포맷은 메타데이터와 중간언어(SIL)가 서로 독립적으로 구성되어 파일 분석이 용이하다. 또한 타입 체크가 편리한 구조를 가지고 있다. 언어 통합을 목적으로 설계한 메타데이터 구조는 확장성을 가지게 되어 보다 폭넓은 언어들을 지원할 수 있다. 따라서 가상 기계를 위한 실행 파일 포맷의 표준이 되리라 판단된다.

향후 연구 과제로는 EVM 파일 포맷의 보완 및 EVM 파일을 실행할 수 있는 가상 기계 구현에 대한 연구가 필요하다.

참고문헌

[1] David Platt, Introducing Microsoft .NET 2nd Edition, Microsoft Press, 2002.

[2] ECMA, Common Language Infrastructure (CLI), Microsoft, 2002.

[3] Joshua Engel, Programming for the Java Virtual Machine, Addison Wesley, 2000.

[4] Kevin Burton, .NET Common Language Runtime, SAMS, 2002.

[5] Thai & lam, .NET Framework Essentials, O'REILLY, 2002.

Serge Lidin, .NET IL Assembler, Microsoft Press, 2002.

[6] Tim Lindholm, The Java Virtual Machine Specification, SUN, 1999.

[7] 오세만, 컴파일러 입문 개정판, 정익사, 2000.