제22회 한국정보처리학회 추계학술발표대회 논문집 제11권 제2호 (2004. 11)

실행 파일 포맷 변환기의 설계 및 구현

김성진*, 고광만
상지대학교 컴퓨터정보공학부
{yes756,kkman}@mail.sangji.ac.kr

Design and Implementation of the Execution File Format Translator

Seong-Jin Kim*, Kwang-Man Ko
School of Computer and Information Engineering, SangJi University

요 약

자바 클래스 파일은 플랫폼 독립성을 보장하면서 다양한 애플리케이션의 실행 파일 포맷으로 사용되고 있으며 실행 효율성을 높이기 위해 파일 형식의 변환 연구, 압축 기법 연구 등이 진행되고 있다. 본 연구에서는 임베디드 시스템에 적합한 가상기계(EVM)을 개발하기 위해 기존의 자바 클래스 파일 형식을 간결한 형태로 재구성한 실행 파일 포맷(*.evm)에 대한 실행 환경 모델을 구축하기 위해 본 논문에서 자바 클래스 파일을 *.evm 형식으로 변환하는 변환기를 설계하고 구현하였다. 자바 클래스 파일로부터 변환된 *.evm 형식에 대한 검증은 현재 구현중인 로더/링커 및 실행 시간 엔진 등을 통한 실험 결과의 정확성을 통해 증명하였다.

1. 서론

가상기계(Virtual Machine; VM)는 언어에 대한 장치 독립성 및 플랫폼 독립성을 지원하는 프로그래밍 실행 환경이다. 현재까지 자바 언어를 위해 JVM, KVM 등이 다양한 환경에서 사용되고 있으며 유사한 가상 기계가 개발되어 활용되고 있다. 특히, 최근에는 컴퓨터 시스템이 아닌 소규모 장치에서도 자바 언어와 같은 고급 언어로 작성된 응용 프로그램이 실행될 수 있도록 임베디드 시스템을 위한 가상기계 개발이 활성화되고 있다[1].

자바 클래스 파일은 원시 프로그램에 대한 모든 정보를 갖고 있으며 이러한 정보를 효과적으로 저장하고 활용할 수 있는 다양한 시도가 많이 진행되어 왔다. 특히, 생성된 클래스 파일 정보에 대한 분석을 통해서 클래스 파일의 최적화와 같은 보다 개선된 실행 방법이 제시되어 왔다.

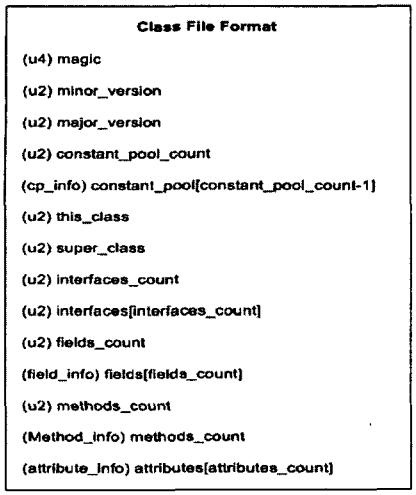
이 논문은 한국과학재단의 특정기초연구(과제번호: R01-2002-000-00041-0)지원에 의한 것임.

2. 기반연구

2.1 자바 클래스 파일 포맷(CFF)

자바에서 사용하고 있는 중간 언어인 Class File Format(CFF)의 구조는 그림 1과 같이 총 14개의 요소로 구성되어 있다[1][2][3]. 클래스 파일을 증명하기 하기 위한 magic 아이템, 컴파일 된 컴파일러의 버전 정보를 나타내는 minor_version, major_version 아이템, constant pool 정보를 나타내기 위한 constant_pool_count, constant_pool[] 아이템이 존재한다.

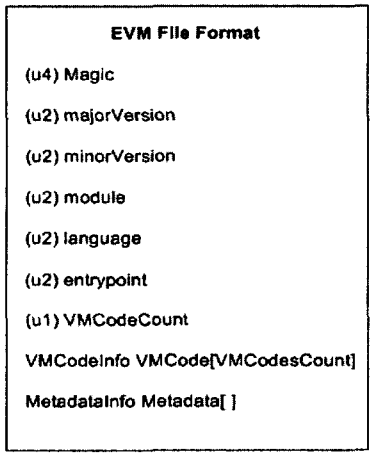
또한 현재 클래스의 정보와 상위 혹은 최상위 클래스의 정보를 나타내는 this_class, super_class 아이템이 있다. 다음으로 인터페이스 정보를 나타내기 위한 interface_count, interface[] 아이템과, field 정보를 나타내기 위한 fields_counts, fields[] 아이템, method 정보를 나타내기 위한 methods_count, methods[] 아이템, field, method, 그리고 attirbute의 상세 정보를 저장하고 있는 attributes_count와 attributes 아이템이 차례로 저장되어 있다.



[그림 1] Class File Format

2.2 EVM File Format(EFF)

임베디드 시스템을 위한 가상기계(EVM)는 PDA, Set-top Box 등과 같은 기기에 탑재될 수 있는 프로그래밍 실행 환경으로서 그림 2와 같은 EVM File Format(EFF)을 입력으로 받아 실행 결과를 생성한다. EFF의 구조는 기존의 자바 클래스 파일 포맷 등을 기반으로 설계되었으며 총 9개의 아이템으로 구성되어 있다[6].



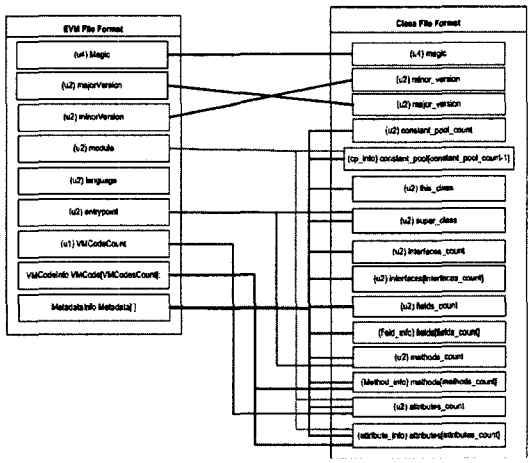
[그림 2] EVM File Format

자바 클래스 파일 포맷과 유사하게 magic 아이템은 정상적인 파일 포맷에 대한 검증 요소이며 majorVersion, minorVersion 아이템은 컴파일러 버전 정보를 저장하고 있다. 또한 파일의 이름을 나타내는 module 아이템, 현재 컴파일 된 중간언어의 소스 언어 정보를 나타내는 language 아이템이 존재하며 파일 안의 시작 메소드를 나타내기 위한 entrypoint 아이템, EVM을 위한 중간 코드인 SIL이 저장되는 VMCodeCount와 VMCode[] 아이템이 있다. 마지막으로 메타데이터의 정보를 나타내는 Metadata[] 아이템이 존재한다.

3. 클래스 파일 포맷 변환기

3.1 전체 Mapping 구성도

자바 클래스 파일 포맷(CFF)을 EVM 파일 포맷(EFF)으로 변환하기 위한 변환기를 구현하기 위해 그림 3과 같은 매핑 테이블을 설계하였다.



[그림 3] CFF에 대한 EFF 매핑 테이블

EFF의 magic, majorVersion, minorVersion 아이템은 CFF의 magic, major_version, minor_version 아이템과 동일한 정보를 저장하고 있으므로 단순 변환이 가능하다. EFF의 module 아이템은 CFF의 counstant_pool[] 아이템과 attributes_count, attributes[] 아이템의 정보로 매핑된다. 또한 EFF의 entrypoint, VMCodeCount, VMCode[] 아이템은 CFF의 superc_class, methods_counts, attributes_count 아이템에 대한 정보와 methods[] 아이템의 정보, attributes[] 아이템의 정보와 각각 매핑된다. 마지막으로 EFF의 Metadata[] 아이템은 15개의 엔트리로 구성되며 각 엔트리들은 CFF의 constant_pool_count, constant_pool[], this_class, super_class, interfaces_count, interfaces[], fields_count, fields[], methods_count, methods[],

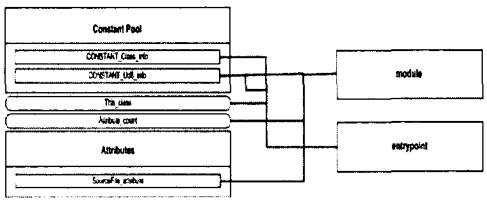
attribute_count, attributes[] 아이템과 역할이 일치하게 된다.

3.2 세부 요소에 대한 매핑 관계

EFF와 CFF의 세부적인 매핑은 EFF의 Metadata[] 아이템(module, entrypoint item 포함)과 CFF의 magic, minor_version, major_version을 제외한 나머지 아이템들의 매핑을 의미한다. Metadata[] 아이템은 총 15개의 엔트리로 구성되어 있으며 각각에 대한 세부적인 매핑 관계는 다음과 같다.

3.2.1 module과 entrypoint

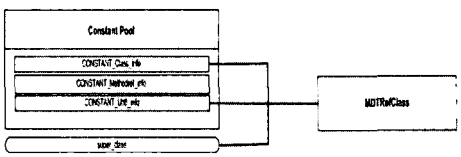
EFF의 module과 entrypoint에 대한 CFF의 매핑 관계는 그림 4와 같다.



[그림 4] moudule과 entrypoint

3.2.2 MDTRefClass

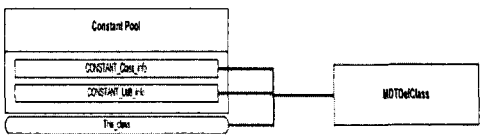
EFF의 MDTRefClass에 대한 CFF의 매핑 관계는 그림 5와 같다.



[그림 5] MDTRefClass

3.2.3 MDTDefClass

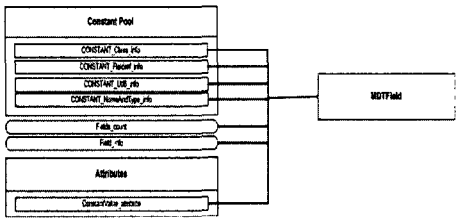
EFF의 MDTDefClass에 대한 CFF의 매핑 관계는 그림 6과 같다. Constant Pool의 정보과 This_class 정보과 조합하여 EFF의 MDTDefClass 정보를 구성한다.



[그림 6] MDTDefClass

3.2.4 MDTField

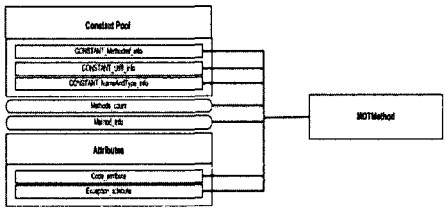
EFF의 MDTField에 대한 CFF의 매핑 관계는 그림 7과 같이 구성된다.



[그림 7] MDTField

3.2.5 MDTMethod

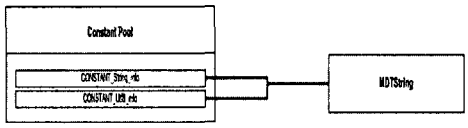
EFF의 MDTMethod에 대한 CFF의 매핑 관계는 그림 8과 같이 구성된다.



[그림 8] MDTMethod

3.2.6 MDTString

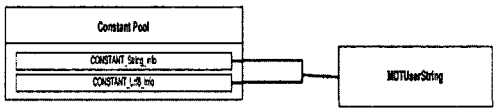
EFF의 MDTString에 대한 CFF의 매핑 관계는 그림 9와 같다.



[그림 9] MDTString

3.2.7 MDTUserString

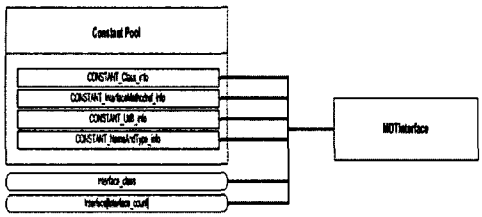
EFF의 MDTString에 대한 CFF의 매핑 관계는 그림 10과 같다.



[그림 10] MDTUserString

3.2.8 MDTInterface

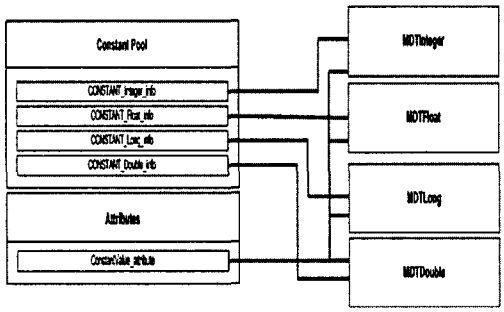
EFF의 MDTInterface에 대한 CFF의 매핑 관계는 그림 11과 같다.



[그림 11] MDTInterface

3.2.9 MDTInteger, Float, Long, Double

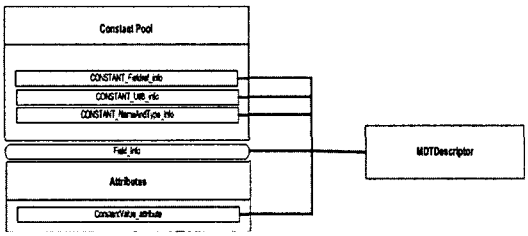
EFF의 MDTInteger, MDTFloat, MDTLong, MDTDouble의 각각에 대한 CFF의 매핑 관계는 그림 12와 같다.



[그림 12] MDTInteger, Float, Long, Double

3.2.10 MDTDescriptor

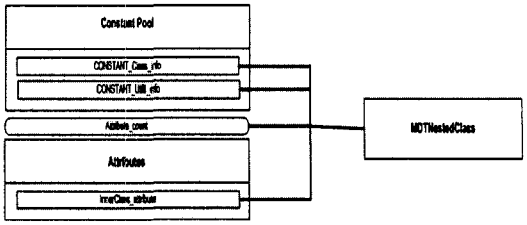
EFF의 MDTDescriptor에 대한 CFF의 매핑 관계는 그림 13과 같다.



[그림 13] MDTDescriptor

3.2.11 MDTNestedClass

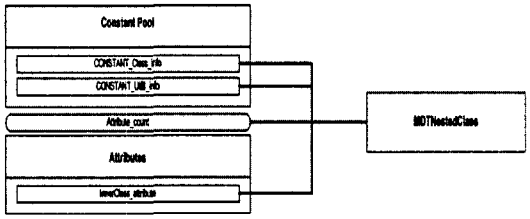
EFF의 MDTNestedClass에 대한 CFF의 매핑 관계는 그림 14와 같다.



[그림 14] MDTDescriptor

3.2.12 MDTException

EFF의 MDTException에 대한 CFF의 매핑 관계는 그림 15와 같다.



[그림 15] MDTException

4. 결론 및 향후 연구

본 논문에서는 임베디드 시스템에 적합한 가상기계(EVM)을 개발하기 위해 기존의 자바 클래스 파일 형식을 간결한 형태로 재구성한 실행 파일 포맷(*.evm)에 대한 실행 환경 모델을 구축하기 위해 본 논문에서 자바 클래스 파일을 *.evm 형식으로 변환하는 변환기를 설계하고 구현하였다. 이를 위해 CFF에 대한 EFF 매핑 관계에 대한 테이블을 설계한 후 각각의 요소에 대한 구현 연구를 진행하였다. 향후 보다 세부적인 엔트리에 대한 변환이 요구되며 변환된 EFF에 대한 검증 연구를 보완하여야 한다.

참고문헌

[1] John Meyer & Troy Downing, "Java Virtual Machine". 1997.

[2] BIll Blunden, "Virtual Machine Design and Implementation in C/C++", Wordware Publishing, 2002.

[3] Tim Lindholm Frank Yellin, "The Java Virtual Machine Specification 2nd", 1999.

[4] John R. Levine, Linkers and Loaders, Morgan Kauffmann Publishers, 2000.

[5] Bill Venners, Inside the Java Virmachine 2nd, McGram Hill, 1999.

[6] 정한종, 윤성림, 오세만, 가상 기계를 위한 실행 파일 포맷, 한국정보처리학회 추계 학술발표논문집, 10권 2호, 2003, 11.