제26회 한국정보처리학회 추계학술발표대회 논문집 제13권 제2호 (2006. 11)

퍼베이시브 컴퓨팅을 위한 가상기계의 디스어셈블러

최유리°, 이창환, 오세만
동국대학교 컴퓨터공학과
e-mail : {yuri°, yich, smoh}@dongguk.edu

Disassembler of Virtual Machine for Pervasive Computing

Yuri Choi°, Changhwan Yi, Seman Oh
Dept. of Computer Engineering, Dongguk University

요 약

최근 모든 공간에서 컴퓨터를 사용할 수 있게 되는 퍼베이시브 컴퓨팅 환경에 대한 관심이 급증하고 있다. 이러한 퍼베이시브 컴퓨팅 환경을 실현하기 위해 실행 환경의 호환성이 요구된다. 이를 해결하기 위해서는 다양한 가상기계들이 필요하다. 그러나 기존의 비교적 큰 임베디드 시스템을 위한 가상기계는 퍼베이시브 컴퓨팅 환경에서 사용하는 작은 기기에 적합하지 않으며, 소규모 장치에 내장하기 어렵기 때문에 퍼베이시브 컴퓨팅 환경에 맞는 새로운 가상기계 플랫폼이 필요하다.

본 논문에서는 임베디드 시스템을 위한 가상기계에 사용되는 디스어셈블러를 개선하여 퍼베이시브 컴퓨팅 환경에 맞는 가상기계를 위한 디스어셈블러를 설계하고 구현한다. 소규모 가상기계에 적합하도록 EVM의 객체 지향 특성을 제거하고 불필요한 명령어의 축약, 실행 파일에서 메타데이터를 제거한다. 이러한 수정된 새로운 가상기계를 위한 디스어셈블러를 통해서 SIL 명령어 바이트 스트림을 완전한 형태의 문자열로 변환하고, EFF의 메타데이터를 SAF 형식으로 생성한다.

1. 서론

최근 소규모 장치 개발 기술이 발전하고 모든 공간에서 컴퓨터를 사용할 수 있게 되는 퍼베이시브 컴퓨팅 환경에 대한 관심이 급증하고 있다. 이러한 퍼베이시브 컴퓨팅 환경을 실현하려면 다양한 소규모 가상기계들이 필요하며, 실행 환경의 호환성이 요구된다. 실행환경의 낮은 호환성은 가상기계 플랫폼으로 해결할 수 있다. 그러나 비교적 큰 임베디드 시스템(Embedded System)을 위한 가상기계는 소규모 장치에는 내장하기 어렵기 때문에 퍼베이시브 컴퓨팅 환경에 맞는 새로운 가상기계가 필요하다[1].

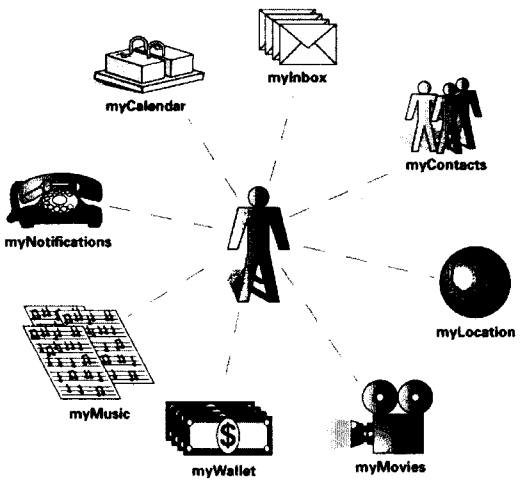
가상기계란 프로세서나 운영체제 등이 바뀌더라도 응용프로그램을 변경하지 않고 사용할 수 있는 기술로 특히, 임베디드 시스템을 위한 가상기계 기술은 모바일 디바이스와 디지털 TV 등에 탑재할 수 있는 핵심기술로 다운로드 솔루션에서는 꼭 필요한 소프트웨어 기술이다.

본 논문에서는 임베디드 시스템을 위한 가상기계에 사용되는 디스어셈블러를 개선하여 퍼베이시브 컴퓨팅 환경에 맞는 가상기계를 위한 디스어셈블러를 설계하고 구현한다. 소규모 가상기계에 적합하도록 EVM의 객체 지향 특성을 제거하고 불필요한 명령어의 축약, 실행 파일에서 메타데이터를 제거한다. 이러한 수정된 가상기계 플랫폼을 위한 디스어셈블러를 통해서 SIL 명령어 바이트 스트림을 완전한 형태의 문자열로 변환하고, EFF의 메타데이터를 SAF 형식으로 생성한다.

2. 관련연구

2.1 퍼베이시브 컴퓨팅 (Pervasive Computing)

퍼베이시브 컴퓨팅은 거의 모든 공간에서 컴퓨터를 사용할 수 있게 된다는 미래적 개념이다. 즉, 컴퓨터가 자연스러운 인간 환경 속으로 들어가게 하는 것으로 언제 어디서나 아무런 제약 없이 각종 기기들을 사용할 수 있는 환경을 말한다. 또한, 모든 네트워크상에서 임의의 장치를 사용하여 어떤 정보라도 전달할 수 있다. 이러한 환경을 실현하려면 네트워킹과 인터네트워킹을 통해 모든 컴퓨터들을 연결시킬 수 있어야 하며 이에 필요한 도구를 쉽게 사용할 수 있어야 한다. 따라서 다양한 생활 용품에 장착 가능한 반도체 집적회로를 이용한 소규모 장치들이 개발되고 있다[2]. (그림 1)은 퍼베이시브 컴퓨팅을 개념화한 그림이다.



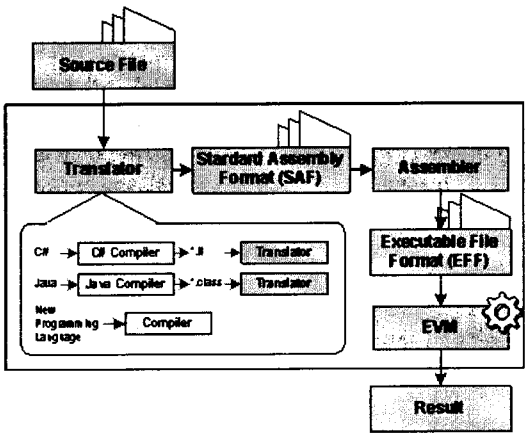
(그림 1) 퍼베이시브 컴퓨팅

2.2 EVM (Embedded Virtual Machine)

EVM은 임베디드 시스템을 위한 가상기계로 모바일 장치, 셋톱박스, 디지털 TV 등에 탑재되어 동적 응용 프로그램을 다운로드하여 실행할 수 있는 가상기계 솔루션이다. EVM을 사용하면 다양한 언어를 지원하고 언어들 간의 통합을 할 수 있으므로 콘텐츠를 쉽게 개발할 수 있다.

EVM은 크게 세 부분으로 나누어진다. EVM의 중간 언어인 SIL을 생성하는 번역기 부분과 SAF 파일을 입력으로 받아 임베디드 시스템의 가상기계에서 실행 할 EFF 파일을 생성하는 부분, EFF 파일을 입력으로 받아 실행시켜 주는 임베디드 시스템을 위한 가상기계 부분이다. (그림 2)는 EVM 모델을 도시화한 그림이다.

SIL(Standard Intermediate Language)은 다양한 프로그래밍 언어를 수용하기 위해 자바 바이트코드, MSIL 등 기존의 가상기계 어셈블리 언어들의 분석을 토대로 정의되었다. SIL은 클래스 선언 등 특정 작업의 수행을 의미하는 의사 코드와 가상기계에서 실행되는 실제 명령어에 대응되는 연산 코드로 이루어져 있다. 객체지향 언어와 순차적 언어를 모두 수용하기 위한 연산 코드 집합을 갖고 있으며 어셈블리 언어 수준의 디버깅을 용이하게 하기 위해 일관성 있는 이름 규칙을 적용하여 코드의 가독성을 확보하였다[3][4][5].



(그림 2) EVM 모델

SAF(Standard Assembly Format)는 클래스의 집합으로 표현한다. 고급 언어로 작성된 프로그램을 SIL로 변환하기 위한 컴파일러의 출력 포맷으로 어셈블러의 입력이며 확장자는 *.saf이다.

EFF(Executable File Format)는 EVM의 실행 파일 포맷이다. EVM은 EFF 파일을 읽고, 실행한 후 결과를 출력한다. EFF에서는 여러 개의 클래스들을 하나의 파일 안에 포함할 수 있으며, EFF는 바이트 스트림으로 구성된다. 이런 바이트 스트림은 항상 리틀 엔디언(Little Endian)으로 저장된다[6][7][8].

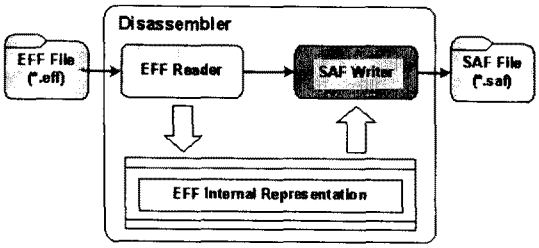
3. 가상기계를 위한 디스어셈블러

본 논문에서는 기존에 사용되던 디스어셈블러와는 다른 소규모 가상기계에 적합한 디스어셈블러를 설계하고 구현한다. 구현하는 디스어셈블러는 EVM의 객체 지향 특성을 제거하고 불필요한 명령어의 축약, 실행 파일에서 메타데이터를 제거한다. 객체지향 기능의 필드는 삭제하지만, EVM 호환을 위해 클래

스 기능은 유지하게 한다. 이 장에서는 디스어셈블러의 전체적인 구조에 대해 설명한다.

3.1 시스템 구성도

본 논문에서 제안하는 퍼베이시브 컴퓨팅을 위한 가상기계의 디스어셈블러는 EFF 파일을 EFF 리더에서 해당 데이터를 읽어와 저장 공간인 EFF 내부 표현 구조에 저장한다. SAF 라이터는 저장된 EFF 내부 표현 구조에서 필요한 데이터를 가져온 후, 어셈블러의 입력 파일인 SAF 파일로 출력하게 된다. (그림 3)은 퍼베이시브 컴퓨팅을 위한 가상기계의 디스어셈블러의 구성도이다.



(그림 3) 디스어셈블러 구성도

3.2 EFF 내부 표현 구조

EFF 자료 저장 공간인 EFF 내부 표현 구조는 크게 명령어 부분과 메타데이터 부분으로 구성된다.

명령어 부분은 메소드 정보와 프로그램의 실행을 위한 SIL 명령어가 저장되며, 메타데이터 부분은 실행 시 필요한 모든 정보가 저장된다.

메타데이터의 각 테이블은 태그와 각 해당 엔트리들로 이루어져 있다. 테이블의 순서는 정해져 있지 않으며 테이블은 필요할 때 마다 삽입하면 된다.

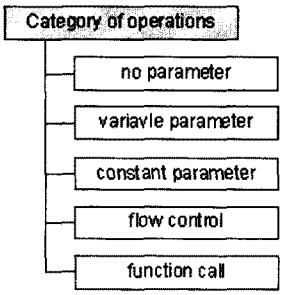
3.3 EFF 리더

디스어셈블러는 특정 프로그램을 위한 EFF 파일을 SAF 파일로 변환하기 전에 메모리 영역에 적재해야만 한다.

EFF 리더는 EFF 파일로부터 명령어 집합과 실행에 필요한 추가적인 정보인 메타데이터를 디스어셈블러의 EFF 내부 표현 구조에 적재한다. 이때, 데이터가 구조적으로 타당한지에 대한 검증을 하며 잘못된 데이터일 경우 오류를 발생시킨다.

3.4 SAF 라이터

디스어셈블러의 핵심 부분인 SAF 라이터는 EFF 파일을 SAF 파일로 변환하기 위한 모듈이다. EFF 파일은 SIL 명령어 표현과 EFF 메타데이터 표현으로 분류해야 한다. 분류된 명령어와 메타데이터는 각 기능에 맞추어 변환된다. SAF 라이터에서의 명령어는 (그림 4)와 같이 분류하여 명령어와 파라미터에 해당하는 내용을 니모닉으로 출력한다.



(그림 4) 명령어 카테고리

메타데이터는 총 15개의 메타데이터 테이블로 구분되며 각 테이블은 서로 구분을 위해 <표 1>과 같이 태그번호를 갖는다.

<표 1> 메타데이터 테이블 태그

테이블 이름	태그번호	테이블 이름	태그번호
MDTRefClass	0x0100	MDTSting	0x0400
MDTDefClass	0x0300	MDTUserString	0x4500
MDTNestedClass	0x0500	MDTInteger	0x5300
MDTInterface	0x0600	MDTFloat	0x5400
MDTRefMember	0x0700	MDTLong	0x5500
MDTField	0x1000	MDTDouble	0x5600
MDTMethod	0x2000	MDTException	0xe000
MDTDescriptor	0x3000		

SAF 라이터에서의 변수명은 [var][인덱스 번호]와 같이 인덱스 번호를 붙여 의미 있는 문자열로 변환하여 생성한다. 또한, 디스어셈블러 전의 명령어와 오퍼랜드에 대한 바이트 스트림을 주석으로 함께 출력하여 SAF 파일을 보다 쉽게 알아볼 수 있도록 한다.

명령어에 해당하는 명령어와 파라메타는 각 테이블을 통해 출력한다. 레이블의 경우에는 레이블 테이블을 이용하여 해당 레이블 주소를 저장한 후, 저장된 해당 주소 위치에 레이블을 출력한다. 레이블 생성은 [$$][일련번호]로 출력되며, 일련번호는 0부터 순서대로 출력하게 된다.

SAF 라이터의 명령어 부분 출력 형식은 [$$][일련번호][:] [명령어] [오퍼랜드] [//][디스어셈블러 전의 명령어와 오퍼랜드에 대한 바이트 코드]로 표현한다.

4. 실험 및 결과

4.1 구현 환경

본 논문에서는 운영체제는 MS Windows XP Professional 환경에서 Visual C++ 6.0 컴파일러를 사용하여 구현하였다.

4.2 실험 및 결과

다양한 실험을 하였으며, 본 논문에서는 디스어셈블러 실험 대상 데이터로 소수를 구하는 prime.eff 파일을 사용하였다. 디스어셈블러를 통해 나온 실행 결과는 (그림 5)와 같다.

```
.class public CPrime
.bgn
  .method public static void main()
  .bgn
  .locals (int var0, int var1, int var2, int var3,
int var4, int var5)
        ldc.i     500      // 06 00 00 00 00
        str       var0     // 12 00 00 00 00
        ldc.i     2        // 06 00 00 00 01
        str       var1     // 12 00 00 00 01
$$5:    ldl       var1     // 0b 00 00 00 01
        ldl       var0     // 0b 00 00 00 00
        gt                 // 3d
        brt       $$0      // 51 00 00 00 c6
        ldc.i     1        // 06 00 00 00 03
        str       var2     // 12 00 00 00 02
        ldl       var1     // 0b 00 00 00 01
        ldc.i     2        // 06 00 00 00 01
        div                // 33
        str       var3     // 12 00 00 00 03
        ldc.i     2        // 06 00 00 00 01
        str       var4     // 12 00 00 00 04
                           :
$$4:    ldl       var1     // 0b 00 00 00 01
        ldc.i     1        // 06 00 00 00 03
        add                // 30
        str       var1     // 12 00 00 00 01
        br        $$5      // 50 00 00 00 14
$$0:    ret                // 64
  .end
.end
```

(그림 5) 실행 결과 (prime.saf)

수정된 가상기계 플랫폼을 위한 디스어셈블러는 바이트 스트림으로 되어있는 EFF 파일을 입력으로 받아, 의미 있는 문자열 형태로 변환하여 SAF 파일로 출력하였다. 즉, EVM를 위한 어셈블러를 역으로 출력함으로써 기계어로 되어 보기 힘든 EFF 파일을 사람이 알아보기 쉬운 SAF 파일로 변환하였다. 또한 주석 처리를 통해 SAF 파일을 보다 쉽게 알아볼 수 있도록 하였다.

5. 결론 및 향후 연구

본 논문에서는 퍼베이시브 컴퓨팅을 위한 가상기계에서 사용할 수 있는 개선된 디스어셈블러를 설계하고 구현 하였다. 개선된 디스어셈블러를 통하여 SIL 명령어 바이트 스트림을 완전한 형태의 문자열로 변환하였다. 그리고 EFF의 메타데이터를 SAF 형식으로 생성하였다.

본 논문의 향후 연구로는 EFF 파일을 보다 시각적으로 표현하기 위한 브라우저와 디스어셈블러의 통합 연구가 필요하다. 또한, 디스어셈블러를 디버거에서 사용하기 위한 모듈로 수정하는 작업이 필요하다.

참고문헌

[1] Mark Weiser, "The computer for the 21st century," Scientific American, pp.94~104, 1991.

[2] "유비쿼터스 혁명이 시작됐다 (37)선진 유비쿼터스 현장을 가다 - ②미 IBM", 전자신문, 2003.

[3] 남동근, 오세만, 가상기계를 위한 어셈블리 언어의 설계, 동국대학교 석사학위논문, 2003.

[4] 남동근, 오세만, "가상기계 코드의 커버링 문제", 한국 정보과학회 가을 학술발표논문집(I), 제30권, 제2호 pp247-249, 2003.

[5] 남동근, 윤성림, 오세만, "가상기계를 위한 어셈블리 언어", 제19회 한국정보처리학회, 춘계학술발표 대회 논문집, 제10권, 제1호, pp783-786, 2003.

[6] 정한종, 오세만, 임베디드 가상기계를 위한 실행파일 포맷, 동국대학교 석사학위논문, 2004.

[7] 지정환, 오세만, "EVM 파일 포맷을 위한 시각화 브라우저", 한국 정보과학회 춘계학술발표논문집(상), 제11권, 제1호 pp503-506, 2004.

[8] 오세만, 컴파일러 입문(개정판), 정익사, 2006.