한국컴퓨터정보학회 동계학술대회 논문집 제31권 제1호 (2023. 1)

QEMU기반 옥테온 아키텍처 에뮬레이션

김수훈O, 강병훈*
OKAIST 정보보호대학원,
*KAIST 정보보호대학원
e-mail: soohunkim@kaist.ac.krO, brentkang@kaist.ac.kr*

Octeon architecture emulation on QEMU

Soohun KimO, Brent ByungHoon Kang*
OGraduate School of Information Security, KAIST,
*Graduate School of Information Security, KAIST

● 요 약 ●

에뮬레이션 기술은 다양한 아키텍처에 대해 하드웨어 독립적인 실험 및 분석 환경을 제공하기 때문에 임베디드 기기 분석에 널리 활용되고 있다. 옥테온(Octeon) 아키텍처는 고성능 네트워크 장비에서 주로 사용되는 프로세서로, 이에 대응하는 에뮬레이션 환경의 구축이 어려워 초기 개발 및 분석에 어려움을 겪는다. 따라서 본 논문에서는 QEMU(v.6.0)을 활용한 옥테온 아키텍처의 에뮬레이션 환경을 구현하고 실험한 결과를 소개한다. 구현된 옥테온 에뮬레이션 환경은 옥테온 아키텍처 고유 인스트럭션 중 특정 하드웨어의 지원을 요하지 않는 인스트럭션에 대하여 에뮬레이션이 가능함을 보였으며 이는 옥테온 아키텍처 기반의 기기 프로그램 에뮬레이션에 활용할 수 있을 것으로 기대한다.

키워드: 에뮬레이션(Emulation), 옥테온 아키텍처(Octeon Architecture)

I. Introduction

사물 인터넷의 확산과 프로세서 성능 향상에 힘입어 수많은 소형 기기가 널리 이용되고 있다. 이러한 임베디드 장비 개발에 있어 초기단계부터 모든 하드웨어를 구비하여 개발하기에는 현실적인 어려움이 있으며 이에 초기 개념증명 단계에서는 에뮬레이터가 주로 활용된다. 현재 ARM, PowerPC등 다양한 아키텍처에 대한 에뮬레이션 기술이 개발되어 적극 활용되고 있으나 Octeon 아키텍처의 경우 에뮬레이션 기술이 활발히 연구되고 있지 않으며 과거 초기단계의 QEMU(v 1.01)를 활용하는 연구 이후 진전이 없는 상태이다.[1] 이에 본 연구에서는 Octeon 아키텍처에 대한 최신 에뮬레이션 기술 확보를 목표로, 6.0 버전의 QEMU에 적용 가능한 동적 변환기를 개발하여 암호화 가속기 등의 특정 하드웨어에 종속되지 않는 인스트럭션들에 대한 에뮬레이션 기술을 개발하였다.

II. Background

1. QEMU

QEMU는 오픈소스 소프트웨어로 타깃 프로세서 기반 명령어를 호스트 프로세서 기반 명령어로 변환하여 이종 프로세서에서 실행되도록 구성된 바이너리를 호스트에서 실행 가능하도록 에뮬레이션 기능을 제공한다. QEMU는 Alpha, ARM, PowerPC등 다양한 플랫폼의 에뮬레이션을 지원한다.[2]

III. The Proposed Scheme

1. Octeon specific instructions

Octeon 아키텍처의 에뮬레이션 기술 개발을 위해 Octeon 프로세서 아키텍처가 가지는 고유 인스트럭션들의 식별 및 분석이 필요하다. 본 연구에서는 하드웨어 가속기 등을 사용하지 않는 27개의 Octeon 고유 인스트럭션들에 대한 에뮬레이션 기술을 개발하였으며 그 중 2개의 인스트럭션(BADDU, POP)에 대한 상세 에뮬레이션 결과를 기술한다. 두 인스트럭션에 대한 설명은 아래 표와 같다.

Table 1. BADDU 인스트럭션 상세

BADDU	
Format	BADDU rd, rs, rt
Operations	GPR[rd] = (GPR[rs] + GPR[rt]) & 0xff
Purpose	To do an unsigned byte add

Table 2. POP 인스트럭션 상세

POP	
Format	POP rd, rs
Purpose	Count the number of ones in a word

2. 에뮬레이션 결과

BADDU 인스트럭션의 에뮬레이션을 위해 아래 Fig. 1과 같이 QEMU 내 소스코드를 추가하였으며, 에뮬레이션 실행 결과를 확인하기 위해 Fig. 2와 같이 어셈블리 코드를 작성하였다.

```
switch (opc) {
case OPC_BADDU:{
        TCGv t0 = tcg_temp_new();
        TCGv t1 = tcg_temp_new();
        gen_load_gpr(t0, rs);
        gen_load_gpr(t1, rt);
        tcg_gen_add_tl(t0, t1, t0);
        tcg_gen_ext8u_tl(t0, t0);
        gen_store_gpr(t0, rd);
        tcg_temp_free(t0);
        tcg_temp_free(t1);
    }
    opn = "baddu";
    break;
```

Fig. 1. BADDU 인스트럭션 구현내용 일부

```
.text
.global main
main:

# Greeting
    li $v0, 4004
    la $a0, 1
    la $a1, msg
    la $a2, 18
    syscall

    li $t0, 0x423
    li $t1, 0x777
# BADDU
    BADDU $v0, $t0, $t1
# expected : 0x9A
```

Fig. 2. BADDU.S

BADDU.S를 빌드하여 기존 QEMU에서 에뮬레이션을 시도하면 아래 Fig. 3.과 같이 Illegal instruction 예외가 발생하며 에뮬레이션이 중지됨을 확인할 수 있다.

```
$ ./qemu-mips  ~/test/test
BADDU TEST Start!
qemu: uncaught target signal 4 (Illegal instruction) - core dumped
[3]    26289 illegal hardware instruction (core dumped)  ./qemu-mips ~/test/test
```

Fig. 3. 기존 QEMU의 에뮬레이션 실패

그러나 Fig.1의 내용이 반영된 커스텀 QEMU의 경우 Fig. 4. 와 같이 에뮬레이션이 성공적으로 수행된다.

```
$ ./build/qemu-mips ~/test/test
BADDU TEST Start!
0000009A
```

Fig. 4. 커스텀 QEMU BADDU 에뮬레이션

마찬가지로 Octeon 고유 POP 인스트럭션 에뮬레이션 기능을 추가 하였으며, 아래 Fig. 5.와 같이 해당 POP 인스트럭션에 대한 에뮬레이션이 커스텀 QEMU 상에서 성공적으로 수행됨을 확인할 수 있다.

```
.global main
main:

# Greeting
    li $v0, 4004
    la $a0, 1
    la $a1, msg
    la $a2, 18
    syscall

    li $t0, 0x10
    li $t1, 0xFFFF

    pop $v0, $t1
# print target register
# move $t0, $TARGET
$ ./qemu-mips ../../test/test
POP TEST Start!
00000010
```

Fig. 5. POP 에뮬레이션 I

```
.global main
main:

# Greeting
    li $v0, 4004
    la $a0, 1
    la $a1, msg
    la $a2, 18
    syscall

    li $t0, 0x10
    li $t1, 0xFFFE

    pop $v0, $t1
# print target register
# move $t0, $TARGET
$ ./qemu-mips ../../test/test
POP TEST Start!
0000000F
```

Fig. 6. POP 에뮬레이션 II

Fig. 5와 Fig. 6은 POP 인스트럭션의 수행 결과로 0xFFFF과 0xFFFE에 '1'이 각각 0x10개, 0x0F개 존재하는 것을 나타내며 이는 POP 인스트럭션이 본래 의미에 맞게 성공적으로 에뮬레이션 되었음을 의미한다.

IV. Conclusions

상용 바이너리를 에뮬레이션 하기 위해서는 QEMU 내 바이너리 동적 변환 기술과 가상 보드 등의 하드웨어정의 기술이 요구된다. 본 연구에서는 6.0 버전 QEMU에 대하여 바이너리 동적 변환 기술을 개발하여 27개의 Octeon 아키텍처 고유 인스트럭션을 에뮬레이션하는 기술을 개발하였다. 추후 하드웨어 정의 기술을 연구 및 개발함으로써 상용 바이너리의 에뮬레이션이 가능할 것으로 기대한다.

REFERENCES

[1] M. A. Mehmood, Q. U. Ain, A. Akram, A. Qadeer and A. Waheed, "Emulating an Octeon MIPS64 based embedded system on X86 in QEMU," 2016 19th International Multi-Topic Conference (INMIC), pp. 1-7, 2016.

[2] Bellard, Fabrice. "QEMU, a fast and portable dynamic translator." USENIX annual technical conference, FREENIX Track. Vol. 41. No.46. 2005.