ARM 프로세서를 위한 실시간 모니터

이은향(李銀香)†, 장원순(張元淳)†, 김형환(金亨煥)† 은성배(殷誠培)†
†한국전자통신연구원, 실시간 OS 팀
†한남대학교 정보통신공학과
전화 : (042) 629 - 8012 / 팩스 : (042) 629 - 7843

A Real-Time Monitor for ARM Processors

Eun-Hyang Lee, Won-Soon Jang, Hyung-Hwan Kim, Seongbae Eun.
Electronics and Telecommunications Research Institute, Real-Time OS team.
Dept of Information&Communication Engineering Hannam University.
E-mail : ehlee@etri.re.kr, wsjang@daniel.hannam.ac.kr ,
hhkim@etri.re.kr , sbeun@camars.kaist.ac.kr.

Abstract

In a distributed real-time system(DRTS), testing and debugging are difficult and critical procedures since they implies several problems like probe effects, nondeterminism, and complex communication patterns. In this paper, we describe the design and implementation of a real-time monitor for ARM processors which are frequently used for embedded applications. The focus of design is to help users debug real-time programs while minimizing the probe effect. Our monitor provides cross debugging features like down-loading from host, break-point based debugging features, and watch-point debugging features for real-time applications.

We developed the debugger for ARM processor and debugger has been used for kernel program.

1. 서론

내장형 프로세서는 논리 회로와 메모리를 하나로 통합한 것으로 현재 가전, 통신기기, 게임기 및 자동차 등 다양한 분야에서 사용되고 있다. ARM 프로세서도 내장형 프로세서의 일종으로서 기존의 ARM 코어를 적용한 제품보다 소비전력이 2분의 1 수준으로 낮기 때문에 소형 모바일 컴퓨팅 시장에서 강력한 경쟁력을 가지고 있다. ARM 프로세서의 주요 타켓 제품군은 HPC, PDA, 웹폰, 스마트폰 등으로 여러 반도체 회사에서 제품 개발에 발빠른 움직임을 보이고 있다.

본 논문은 타켓(ARM SA-110 Processor)에서 수행될 실시간 OS 커널 개발 시 커널을 디버깅할 디버거(명칭:KMON)와 모니터(명칭:RMON)개발에 대한 내용을 담고 있다. 타켓에 다운로드된 커널이 수행 중에 오류를 일으키면 이를 디버깅해야 한다. ARM SA-110은 제조업체로부터 Angel[1]이라는 디버거가 제공되지만 너무 많은 자원을 소비하고, 많은 제약이 따르므로 커널을 디버깅하기에는 부적합하다.

커널을 디버깅 할 때 어려운 점은 다음과 같다. 첫째, 일반적인 디버거는 운영체제의 도움을 받아 디버깅을 하지만 커널 디버거는 운영체제의 도움을 받을 수가 없다. 둘째, 디버거가 사용하는 자원과 커널이 사용하는 자원이 충돌이 생길 때에 예측하기 어려운 문제가 발생한다. 셋째, 커널은 비결정성과 병행성이 풍부하여 정지점 기반 디버깅의 경우 디버깅 자체가 시스템 동작을 변화시키는 탐침효과(Probe Effect)를 발

생 시킨다[2,3,4].

본 논문에서는 위의 세 가지 문제를 해결하는데 중점을 두었다. 특히, 탐침효과를 최소화하기 위한 다양한 모니터 구현기술[5] 중 소프트웨어 모니터링 기법을 소개한다. 이를 바탕으로 ARM프로세서 상에서 커널 디버거를 개발할 때 탐침효과를 최소화 할 수 있는 실시간 모니터의 설계 및 구현에 대하여 기술한다.

본 논문의 구성은 다음과 같다. 2장에서는 실시간 모니터링과 탐침효과 그리고 관련연구에 대하여 기술한다. 3장에서는 ARM 실시간 모니터설계로 시스템 구성과 기능에 대하여, 4장에서는 모니터의 구현, 마지막으로 5장에서는 결론 및 향후 연구방안을 기술한다.

2. 실시간 모니터링

디버깅 과정은 오류 검출, 오류 수정, 재 수행의 반복된 과정으로 다음과 같다.

①프로그램의 수행이 오류인지를 검사
②오류가 발생했을 때 상태 값들을 수집
③상태 값들을 분석하여 오류의 원인을 판별
④코드를 수정
⑤재수행

위 과정을 오류가 발생하지 않을 때까지 반복하며 이를 순환적 디버깅(Cyclic debugging)이라 한다[6]. 순환 디버깅의 주요 과정 중에는 시스템의 동작을 일시 정지시키는 정지점 설정과 모니터링하는 과정이 필수적이다.

2.1 실시간 모니터링과 탐침효과

정지점 기반 디버깅에서 모니터링이란 프로그램 테스팅이나 디버깅을 위하여 의심이 가는 위치에 정지점을 설정하고 정지점에서 프로그램 수행이 일시 정지되면 필요한 정보를 검출/수집하는 것을 말한다. 모니터링은 두 과정으로 이루어진다. 첫 번째는 트리거링 과정이고 두 번째는 기록과정이다. 트리거링 과정에서는 미리 정의된 이벤트가 발생하는지를 계속 검사한다. 이벤트가 발생하면 그때부터 기록 과정이 시작되는데 기록 과정에서는 이벤트와 관련된 데이터를 수집하여 기록/저장한다. 이들 과정은 하드웨어, 소프트웨어 그리고 하드웨어와 소프트웨어 결합의 하이브리드 방식으로 구현 될 수 있다.

실시간 시스템에서 모니터링은 근본적으로 문제점을 안고 있으며 그 중 모니터링 되는 동안 프로그램 실행에 대한 정보 수집의 시도가 시스템의 상태를 예측 불가능하게 변화시키는데 이것을 탐침효과라 한다. 실시간 시스템에서 오류가 발생했을 때 프로그램 수행을 정지시키는 것이 시간 제약을 어기게 되고 두 프로세서가 상호 작용하는 프로그램에서는 수행 정지가 다른 프로세서에 오류를 발생시킬 수 있다. 예를 들어 A, B, 두 프로세스가 서로 협력하여 수행되고 B 프로세스가 A 프로세스로부터 동기를 위한 메시지를 전송받으며 B 프로세스는 일정시간 이상 메시지를 받지 못하는 경우 예외처리를 하는 것을 가정해보자. 이때 A 프로세스의 수행이 정지점에서 중지된다면 일정시간 후에 A 프로세스는 예외처리를 하게 되는데 이는 정지점이 설정되지 않았다면 발생하지 않는다.

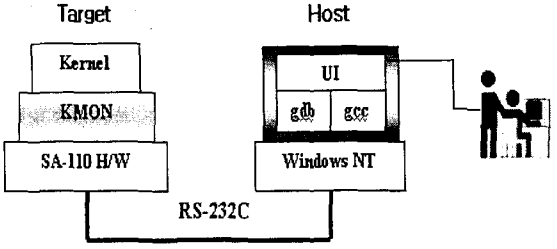
2.2 관련연구

실시간 모니터 기법을 분류하면 크게 세 가지로 나눌 수 있는데 소프트웨어, 하드웨어 그리고 하이브리드 기법이 그들이다[5]. 이 중 소프트웨어 모니터링 기법은 모니터링을 위한 코드가 대상 시스템에 삽입되어 함께 수행되며 한 이벤트에 대한 모니터링이 끝나면 그 시스템의 수행은 재개된다. 적응성이 좋고 별도의 하드웨어가 필요 없다는 장점이 있지만 탐침효과를 극복하기 어렵다는 단점이 있다. 이에 HARTS를 위한 HMON이라는 시스템은 모니터링의 부담을 균일하게 하고 이를 보정하여 주는 결정적 재수행 기법을 제안하였다[2]. 또한 Miller[3]는 Berkeley UNIX4.2BSD에서 모니터링 분석 기법을 포함하는 연구를 수행하였으며 Tokuda[4]의 경우에도 모니터링 된 데이터를 시각화하는 분석 기법을 도입하였다.

3. ARM 실시간 모니터의 설계

3.1 개발 환경

커널프로그램의 개발은 일반적인 프로그램 개발 환경과 상이하다. 프로그램 개발은 호스트에서 수행되며



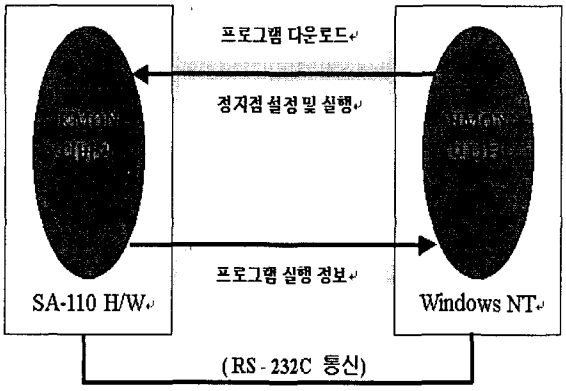
<그림1> 커널 디버거 개발 환경

실행은 실제 타켓에 다운로드 되어 수행된다. 이러한 환경을 특히 Cross Development Environment라 하며 호스트와 타켓으로 나누어져 있다. <그림1>은 일반적인 커널과 디버거 개발환경을 나타낸다.

호스트에는 실제 타켓 프로세서의 인스트럭션을 생성하는 컴파일러를 시스템에 맞게 포팅하는 작업이 선행되어져야 한다. 본 논문에서는 ARM용 gcc를 윈도우 NT에 포팅하여 개발에 사용하였다.

3.2 시스템 구성도

ARM 실시간 디버거와 모니터는 윈도우 NT(호스트)에서 개발된다. 이중 모니터는 호스트에서 수행되고 디버거는 ARM 프로세서가 장착된 SA-110보드의 메모리에 다운되어 수행된다. 디버거는 모니터로부터 텍스트 기반의 명령을 받아 수행하며 정지점 기반의 어셈블리 레벨 디버깅을 지원한다. 모니터를 통해 정지



<그림2> 시스템 구성도

점에서 프로그램 실행이 일시 정지하면 프로그램 수행 상태를 모니터링 할 수 있다. <그림2>는 전체 디버거 시스템의 구성도 이다.

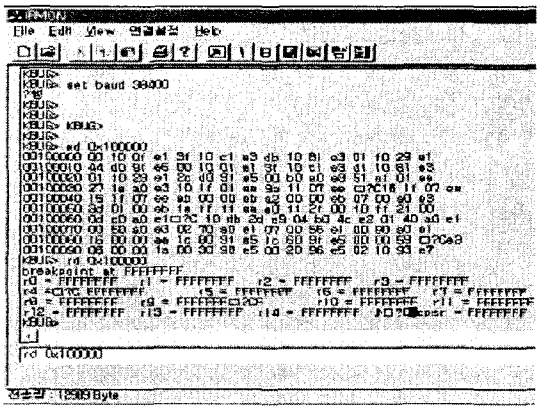
3.3 주요 기능

모니터는 크게 ARM프로세서와의 통신 기능과 모니터링 기능을 가진다.

①통신기능
- * 타켓에 연결 / 해제
- * 프로그램 다운로드

②모니터링 기능
- * 정지점 설정 / 정지점 해제
- * 프로그램 실행
- * 레지스터 값 디스플레이 / 레지스터 값 변경
- * 메모리 디스플레이 / 메모리 변경

모니터는 9,600 bps ~ 115,200 bps의 다양한 속도로 타켓에 연결 할 수 있으며 원하는 메모리 번지에 프로그램을 다운로드 할 수 있다. 정지점을 설정하고 프로그램을 실행 시킬 수 있으며 정지점에서 프로그램 정지 시 레지스터와 메모리 값을 확인 변경 할 수 있다.



<그림3> RMON모니터링 기능

<그림3>은 RMON의 모니터링기능을 보여준다.

모니터는 ARM 프로세서와 텍스트 모드로 통신하며 모든 명령은 내부적으로 텍스트로 처리되어 프로세서에 전달된다.

3.4 주시점 (Watch Point)기반 디버깅

본 논문에서는 탐침효과를 줄이기 위하여 다음 명령어를 지원한다.

br {<addr1>, <addr2>}

위 명령어의 의미는 <addr1>에서 일시정지하고 <addr2>의 내용을 검사하여 저장하라는 것이다. 정지점 기반의 br 명령과 다른 점은 주시점의 경우 <addr1>에서 정지되더라도 <addr2>의 내용을 지정된 기억장소에 저장한 후에는 바로 재수행 된다는 점이다. 이런 특성 때문에 주시점 기반 명령어라고 한다.

사용자는 프로그램의 수행 전에 모니터링 할 시점과 모니터링의 목표를 상기 명령어를 이용하여 설정한다. 프로그램 수행 중에 <addr1>에 이르면 수행이 일시 정지되고 지정된 주소의 내용을 기록한 후 수행이 재수행 된다. 모니터링 된 정보는 수행 종료 후 디버깅을 위하여 분석된다.

주시점 기반 명령어를 구현할 때 고려할 점은 탐침효과를 최소화하도록 구현하는 것이다. 주시점 기반 명령어는 정지점에서의 illegal 예외 처리, 상태기록, 재수행처리의 세가지 단계를 통하여 처리된다. 따라서 각각의 단계를 최소의 시간 안에 처리할 수 있도록 노력하여야 한다.

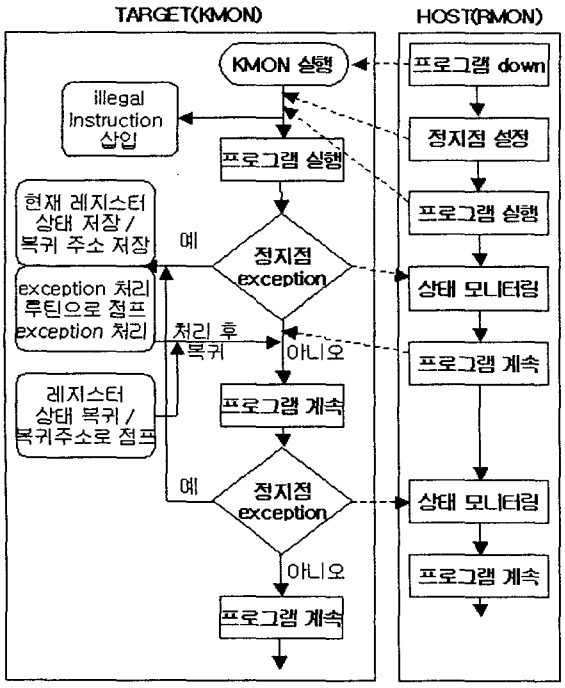
4. 모니터 구현

정지점 기반 디버깅에서는 탐침효과를 피할 수 없

다. 예를 들어 두 프로세서가 상호 통신하며 수행될 때 한 프로세서를 디버깅하기 위하여 일시 정지시킬 경우 다른 프로세서에서 동작하는 프로그램은 시간종료에 의해서 오류를 발생시킬 것이다. 이를 위하여 주시점(watch-point) 기반 디버깅 기법이 사용된다. 주시점 기반 디버깅은 차후 연구에서 구현 할 예정이며 본 장에서는 정지점 기반 디버깅의 구현을 기술한다.

타켓의 디버거(KMON)가 수행되고 모니터(RMON)를 타켓에 연결시킨 후 정지점을 설정하면 Exception을 처리하는 벡터 테이블(Vector Table)에 Exception 처리 루틴으로 점프하는 illegal 인스트럭션을 삽입한다. 프로그램 실행 중 Exception이 발생하면 현재 레지스터 상태와 Exception 처리 후 복귀할 주소를 저장하게 된다. 레지스터 상태를 저장하는 것은 복귀 후 프로그램 재 수행을 위해서이다. 이때 프로세서 모드는 User 모드에서 System 모드로 바뀐다. 참고로 ARM SA-110은 7가지의 프로세서 모드를 가지며 응용 프로그램은 User 모드에서 KMON은 System 모드에서 수행된다[7]. KMON은 RMON의 명령을 받아 Exception을 처리하고 이때 RMON을 통해 프로그램 수행 상태를 모니터링 할 수 있다.

RMON에 의해 프로그램 계속 명령을 받으면 KMON은 저장한 레지스터 값을 이용하여 레지스터를 Exception 발생 이전의 값으로 복구하고 복귀 주소로 점프하여 프로그램을 계속 실행한다. <그림4>는 모니터 구현을 보여준다. <그림4>에서 볼 수 있듯이 KMON은 여러 주소에 정지점을 설정/처리가 가능하다.



<그림 4> 모니터 구현

5. 결론 및 향후 연구 방안

본 논문에서는 실시간 커널을 개발할 때 요구되는 실시간 모니터의 설계 및 구현에 대하여 기술하였다. 최근 들어 정보가전용 프로세서로 각광을 받는 ARM 프로세서를 타겟으로 하였으며 Window NT와 교차 개발환경을 구축할 수 있도록 하였다. 주요 기능들이 구현되어 사용되고 있다.

향후 연구방향은 제안된 주시점 기반 명령어를 최소의 수행시간을 갖도록 성능을 개선하는 것이다.

참고문헌

[1] ARM Reference Manual, ARM, 1997 and 1998.

[2] J.Joyce, G.Lomow, K.Slind, and B.Unger., "Monitoring Distributed Systems", ACM Trans. Computer Systems, Vol.5, No.2, May 1987, pp.121-150.

[3] B.P.Miller, C.Macrander, and S.Sechrest., "A Distributed Programs Monitor for Berkeley UNIX", Software-Practice and Experience, Vol.16, No.2, Feb.1986, pp.183-200.

[4] H.Tokuda, M.Kotera, and CW.Merce.r, "A Real-Time Monitor for a Distributed Real-Time Operation System", Proc. of ACM Workshop Parallel and Distributed Debugging, ACM press, New York, 1988, pp.68-77.

[5] Tsai, J.J.-P, and et.al., "Distributed Real-Time Systems:Monitoring , Visualization, Debugging and Analysis", JOHN WILEY & SONS, INC., 1996.

[6] C. E. McDowell and D. P. Helmbold, "Debugging Concurrent Programs", ACM Computing Surveys, vol. 21, no. 4, pp. 593-622 December 1989.

[7] Dave Jaggar, "ARM(Advanced RISC Machines Architectural) Reference Manual", ARM, 1996.