Core-A프로세서용 MicroC/OS-II 이식

심정민*, 지정훈, 우균
부산대학교 컴퓨터공학과
e-mail:{kildin, jhji, woogyun}@pusan.ac.kr

Porting MicroC/OS-II to Core-A processor

Jung-Min Shim*, Jeong-Hoon Ji, Gyun Woo
Dept of Computer Engineering, Pusan National University

요 약

본 논문에서는 국산 임베디드 프로세서인 Core-A에서 동작하는 실시간 운영체제 이식에 대해 설명한다. 운영체제 이식을 위해서는 코드 작성에 앞서 컴파일과 디버깅을 위한 개발 환경을 구축하고 운영체제 이식을 위해 새로 작성할 부분을 파악할 필요가 있다. 이식할 운영체제인 MicroC/OS-II는 교육용으로 널리 쓰이는 실시간 운영체제로 C와 Assembly로 작성되어 있으며, 프로세서에 독립적인 코드와 프로세서에 의존적인 코드가 분리되어 있어 이식이 용이하다. Core-A로의 운영체제 이식은 Context Switching이나 Critical Section과 같이 레지스터를 직접 다루어야 하는 프로세서에 의존적인 코드를 수정하여 이루어졌다.

1. 서론

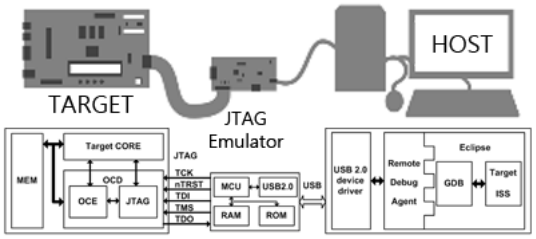
임베디드 시스템이란 특정한 기능만을 위해 설계된 시스템을 말하며 TV, MP3 플레이어, 셋탑박스나 휴대폰과 같이 일상적으로 접할 수 있는 기기가 이에 해당한다. 이전의 임베디드 시스템은 정해진 기능만을 수행하면 되었기 때문에 운영체제가 필수적이지 않았다. 하지만 프로세서 기술이 향상됨에 따라 저렴하면서 빠른 프로세서가 나오고 소비자의 요구사항이 많아짐에 따라 임베디드 기기가 맡은 역할은 점점 더 복잡해지면서 임베디드 시스템에서도 운영체제가 필수적인 요소로 자리 잡고 있다.

본 논문은 Core-A의 개발환경을 구축하고 실시간 운영체제인 MicroC/OS-II를 Core-A에서 동작하도록 이식함으로 Core-A 컴파일러와 디버거의 동작을 보장하고 Core-A의 활용도를 높였다.

2. Core-A 시스템 환경

Core-A는 특허청에서 추진하고 KAIST ICS연구실에서 개발한 다목적 임베디드 프로세서로 합성가능한 소프트코어 형태로 제공되기 때문에 디자인 수정이 가능해 기능을 추가할 수 있다[1].

임베디드 개발에는 일반적으로 컴파일 환경과 실행환경이 다른 Host/Target 방식을 이용한다. Host/Target 방식이란 PC나 워크스테이션에서 임베디드 프로세서에서 동작하는 프로그램을 작성하고 이를 임베디드 시스템에 옮겨 실행하는 방식을 말한다[2].



(그림 1) Core-A 개발환경

Core-A 프로그램을 위한 개발 환경 구성은 그림 1과 같다. 소프트웨어는 Core-A용 GNU cross-compiler와 Core-A용 GDB, 통합개발도구인 Eclipse와 같은 공개 소프트웨어를 사용한다[3, 4]. 소프트웨어는 구입비용이 들지 않으며 여러 플랫폼에서 동작한다. 하드웨어는 Core-A 개발 보드와 JTAG(Joint Test Action Group) 인터페이스 디버깅 툴이 필요하다[4]. 또, 디버깅을 위해 Core-A 소프트코어에 OCD(On-Chip Debugger)를 합성해야 한다[4].

3. MicroC/OS-II 이식

MicroC/OS-II는 1992년 Jean J. Labrosse라는 엔지니어가 공개한 경성 실시간 운영체제로 크기가 작고 프로세서에 의존적인 일부 코드를 제외한 대부분의 코드가 C로 작성되어 있어 코드의 이해가 쉽고 이식성이 뛰어나다[5].

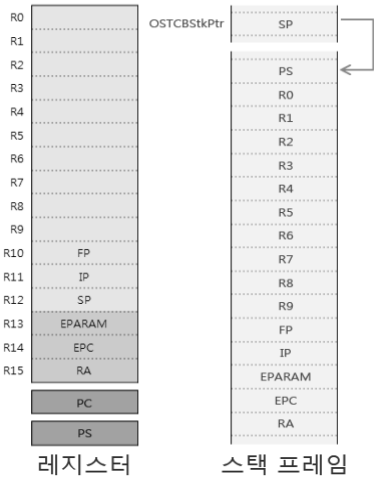
MicroC/OS-II는 프로세서에 독립적인 코드와 프로세서에 의존적인 코드로 나누어 볼 수 있다. 프로세서에 독립적인 코드는 ANSI C로 작성되어 어떠한 환경에서도 동작

하는 코드이다. 반면 프로세서에 의존적인 코드는 작동하는 프로세서에 따라 새로이 작성해야 하는 코드이다.

프로세서에 의존적인 코드는 일부를 제외하면 모두 어셈블리로 작성되어 있으며 문맥전환(Context Switching)과 임계구역(Critical Section)에 대한 함수들 포함하고 있다. 임계구역에 대한 함수는 인터럽트를 비활성화 하거나 프로세서 상태를 비활성화 이전으로 되돌리는 단순한 코드이며 문맥전환의 구현이 이식의 가장 중요한 부분이다.

3.1. 문맥전환

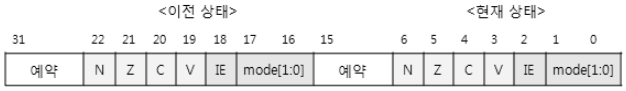
하나의 작업이 끝나거나 인터럽트가 발생하였을 때 실행되던 작업을 스택에 저장하고 새로운 작업을 레지스터에 복원하는 것을 문맥전환이라고 한다.



(그림 2) Core-A의 레지스터와
문맥전환을 위해 생성할 스택 프레임

그림 2는 Core-A의 레지스터와 문맥전환 시 작업을 저장하거나 복구하기 위해 생성할 스택 프레임의 형식을 보여준다. MicroC/OS-II는 SP(Stack Pointer)값을 별도로 관리하므로 문맥전환이 발생하면 스택 프레임은 SP를 제외한 15개의 GPR과 PS(Processor Status)값을 저장한다. EPC(Exception Program Counter)값이 저장되는 장소는 실제로는 PC레지스터의 값이 저장된다. PC값 변경을 위해서는 선택문을 이용해 다른 레지스터의 값을 PC에 대입해야 하기 때문이다. EPC의 원래 목적은 인터럽트가 발생하였을 때 기존에 수행하던 작업의 PC값을 임시로 옮겨두는 것이다. 문맥전환은 임계구역이라 인터럽트가 발생할 수 없기 때문에 이를 이용해 PC값을 변경한다.

일반적인 문맥전환은 call 명령어에 의해 일어나므로 작업 중이던 명령어 위치를 RA레지스터에 저장하고 있다. 이를 스택의 EPC 위치에 저장한다. RA레지스터는 레지스터 절약을 위해 2-bit 적게 설계되었기 때문에 이를 신경써야 한다. 인터럽트에 의해 문맥전환이 일어나는 경우에는 작업 중이던 명령어 위치가 EPC에 저장되어있기 때문에 그대로 스택의 EPC 위치에 저장하면 된다.



(그림 3) Core-A의 PS 레지스터

PS 레지스터는 그림 3과 같이 상위 16비트는 현재 상태를, 하위 16비트는 이전 상태를 가지며 각 상태는 프로세서 모드, 인터럽트 허가 등에 관한 정보를 담고 있다[7].

Core-A프로세서는 인터럽트가 발생하면 PC값이 EPC 레지스터로 이동하고 PS레지스터의 하위 16비트 값이 상위 16비트로 이동한다. 따라서 인터럽트가 발생하여 문맥전환이 일어나면 이를 인터럽트 이전 상태로 돌려놓은 뒤 저장해야 한다.

3.2. 동작 확인

이식한 OS의 동작을 검증하기 위해서 Core-A 개발보드에 각기 다른 문자를 출력하는 두 개의 태스크를 실행시켜 태스크가 정해진 순서대로 동작함을 확인하였다. 또 하드웨어 디버깅을 통해 문맥전환 시 레지스터 값의 저장과 복구가 올바르게 동작함을 확인하였다.

4. 결론 및 향후 과제

본 논문은 국산 프로세서인 Core-A에서 동작하는 프로그램을 제작하기 위한 개발 환경을 구성하고, 이를 이용해 실시간 운영체제인 MicroC/OS-II를 이식하면서 컴파일러와 디버거가 제대로 동작함을 확인하였다.

향후 연구 과제로 MicroC/OS-II의 이식 경험을 바탕으로 임베디드 Linux와 같이 많은 분야에 사용되는 운영체제를 이식하여 Core-A 프로세서의 효용성을 높이는 방법을 모색해 볼 것이다.

참고문헌

[1] 이광엽 “고성능 Core-A 임베디드프로세서 구조”, *대한전자공학회 2009년 SoC학술대회*, pp.184~187, 2009

[2] 임채덕, 김태호 “임베디드 소프트웨어 통합 개발 환경 기술”, *정보과학회지 제24권 제8호*, pp.46~52, 2006

[3] 양원용, 이종열 “GCC 포팅을 통한 Core-A C 컴파일러 개발”, *대한전자공학회 2009년 SoC학술대회*, pp.188~191, 2009

[4] 박형배, 지정훈, 허경철, 우균, 박주성 “Core-A Debugger 및 ICE”, *대한전자공학회 2009년 SoC학술대회*, pp.192~195, 2009

[5] Jean J. Labrosse “MicroC/OS-II 실시간 커널 2판”, 에이콘, 2005