Real-Time OS 의 CE 기기 적용시 Cache 를 통한 Booting-Time 개선

*김경훈, 하성호, 박정형
삼성전자, 디지털 미디어 연구소

Improvement of Booting-time on Real-Time OS by cache for CE Devices

Kyung-Hoon Kim, Seong-Ho Ha, Jeung-Hyung Park
Digital Media R&D Center, Samsung Elec.
e-mail) auditk2h@samsung.com, knight.ha@samsung.com, jeung.park@samsung.com

요 약

CE 제품에 리얼타임 OS 를 도입하면서, 제품의 조건을 만족시키기 위한 기술에 대해 많은 연구가 진행되고 있다.

특히, CE 제품에 있어서 중요한 이슈인 부팅 시간은 펌웨어 수준과 비교했을 때 코드사이즈나 OS 초기화 과정 때문에 다소 느려지는 경향을 보이고 있다.

본 논문은 이러한 CE 제품의 부팅 시간에 초점을 맞추고 리얼타임 OS 적용시의 부팅 시간을 개선하였다.

구현에 사용된 ARM920T Core 는 32-비트 RISC 구조이며, 각 16KB 의 인스트럭션 Cache 와 데이터 Cache, 그리고 MMU(Memory Management Unit)로 구성되어 있으며, 리얼타임 OS 는 선점형 방식의 커널로 구성된 OS 를 사용하였다.

1. 서 론

CE 제품의 디지털 컨버전스로 기능의 다양화와 복잡성이 증대되면서, OS 도입이 점차 증가하고 있다.

그러나, OS 를 CE 제품에 적용하기 위해서 개선해야 할 기술적인 문제들이 있다. 특히, 동작상태까지의 부팅 시간은 사용자에게 가장 먼저 인식되는 문제인 동시에 가장 중요한 요소가 되고 있다.

부팅 시간은 시스템에 전원이 인가된 후부터, 사용자 어플리케이션이 시작되는 시점까지를 말하는 것으로, 하드웨어 초기화, OS 로드, OS 컴포넌트 초기화, 사용자 어플리케이션 순으로 진행된다. 여기에서 하드웨어 초기화 부분은 어셈블리 코드로 구성되는 매우 작은 부분이고, OS 컴포넌트 초기화와 사용자 어플리케이션의 호출과정은 커널소스의 변경없이는 수정하기 어려운 부분이므로, 본 논문에서는 OS 를 로드하고 OS 에 제어권을 넘기는 시점까지에 대해 분석한다.

본 논문에서 부팅 시간을 개선하기 위해 Cache 를 사용하였으며, 이것은 하드웨어 플랫폼에 의해 다를 수 있으며, Cache 를 지원하는 플랫폼에 대해서만 적용 가능하다. 특히, Cache 의 설정에 따라 성능의 차이가 있지만[1], 본 논문에서는 이러한 차이는 논점에서 배제하기로 한다.

Cache 는 메모리 구조에서 가장 빠른 메모리를 말하는 것으로, 레지스터와 메인 메모리 사이에 존재하며 자주 사용되는 데이터를 보관하고 있으며, 짧은 latency 와 빠른 대역폭을 만족시켜준다.[2]

본 논문의 구성은 다음과 같다. 2 장에서는 OS 의 부팅순서를 분석하고, 3 장에서는 타겟 시스템에서의 Cache 동작과 구현에 대해 설명한다. 마지막으로 4 장에서 결론을 맺는다.

2. OS 부팅순서

시스템에 전원이 들어오게 되면 부팅코드를 읽어오기 위해, ROM 영역을 액세스한다. 그리고 읽어온 부팅코드는 타겟시스템의 메모리(RAM) 및 기타 하드웨어요소를 초기화 한다. 마지막으로 OS 를 RAM 에 로드하고 제어권을 OS 로 넘겨준다.[3]

본 논문에 사용된 OS 는 WindRiver 사의 리얼타임 OS 인 VxWork 를 사용하였으며, 여기에서도 동일한 순서로 진행되는데, 이는 OS 의 이미지에 따라 약간 다르다. 본 논문에 사용된 OS 이미지는 다음과 같은 조건을 갖는다.

1) 부트로더 포함

2) ROM 에 저장

3) Compressed Image

부트로더는 타겟 하드웨어에 대한 최소한의 초기화를 진행하고, OS 이미지의 압축을 풀면서 RAM 에 복사하는 역할을 한다. 여기서 OS 이미지를 RAM 에 복사하는 부분이 부팅과정에서 가장 오랜 시간을 소요한다.

여기에서 사용된 Compress 알고리즘은 zlib 를 수정한 compress 가 사용되었다. 이 알고리즘을 사용한 평균 압축율은 50% 정도이다.[4]

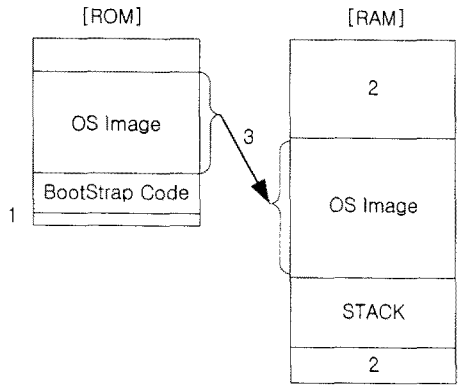

그림 1 OS 부팅 순서

타겟보드의 전원을 켜면, ROM 의 BootStrap Code 가 수행되는데(그림 1-1) 이것이 부트로더의 역할을 하게 된다. 타겟에 대한 기본적인 초기화 과정과 RAM 초기화 과정이 진행되고, 사용하지 않는 메모리영역은 Parity 에러를 방지하기 위해 0 으로 초기화 한다(그림 1-2). 그리고, OS Image 를 해당 RAM 영역으로 복사한다. 이 과정에서 압축해제가 수행된다 (그림 1-3). 그리고, 마지막으로 프로그램카운터를 OS Image 내의 커널 초기화 entry point 로 이동하여 제어권을 넘겨준다[5].

일반적으로 부트로더의 수행중에는 MMU 를 동작시키지 않는 상태로 진행된다. 대개 MMU 설정은 커널내에서 수행하기 때문이다.

3. Cache 설정 및 구현

3.1 Cache 설정

구현에 사용된 CPU 는 ARM920T 가 탑재되어있는 보드를 사용하였다. ARM920T 는 각 16KB 씩의 인스트럭션 Cache 와 데이터 Cache 를 지원한다[6].

인스트럭션 Cache 는 CP15 레지스터 1 의 12 번 비트를 설정시킴으로써 동작시킬 수 있으며, MMU 가 동작하지 않을 경우에는 물리주소가 가상주소에 Flat-mapped 된 것으로 동작한다.

데이터 Cache 는 Write-through 또는 Write-back 모드를 지원하며, MMU 가 동작되어야만 데이터 Cache 를 사용할 수 있다. 이것은 각 메모리영역에 대한 Cache 환경이 MMU 변환테이블에 정의되기 때문이다.

ARM 프로세서의 MMU 는 가상주소를 물리주소로 변환해주는 일과 메모리접근권한에 대한 제어기능을 가진다.

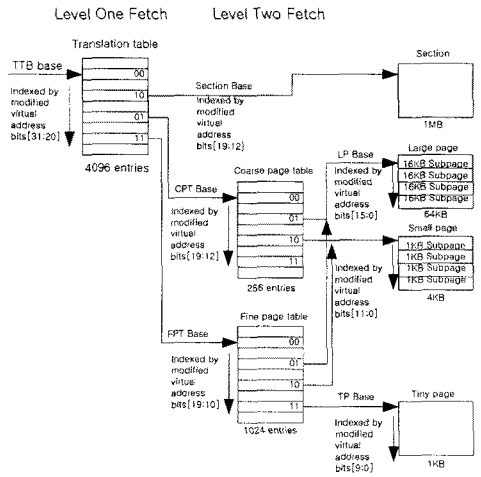

그림 2 Translating Page Tables

메모리 접근방식은 그림 2 에 나온 것처럼 Level one fetch 와 Level two fetch 를 사용할 수 있으며 해당하는 level descriptor 를 생성하여야 한다[6]. 그러나, Level Two Fetch 에서 Small page 방식을 사용할 경우, 메모리 영역을 4KB 단위로 나누어야 되므로, 이 부분에 대한 소요시간이 증가하게 된다. 따라서, 목적에 따라 Level 을 선택하여야 한다.

3.2 구 현

구현에 사용된 타겟보드환경은 표-1 과 같다.

Component	Clock
CPU (ARM920T)	216MHz
Flash ROM (2MB)	108MHz
SDRAM (32MB)	108MHz
MMU	Flat-Mapped Section Base
Cache	Instruction Cache Enable, Data Cache Enable

표 1 타겟보드 환경

부팅과정에서 MMU 가 동작하지 않으면 데이터 Cache 를 사용할 수 없어서 압축된 이미지의 압축해제 과정에서 시간이 많이 소요된다. BootStrap 내에서 MMU 를 사용하여 데이타 Cache 를 동작시킨다면, 부팅시간을 단축시킬 수 있을 것이다.

이를 위해, BootStrap Code 에서 Section Base 의 페이지 테이블을 작성하여 MMU 와 함께, 인스트럭션 Cache 와 데이타 Cache 를 Write-through 모드로 Enable 시키고, 가상주소와 물리주소는 Flat-mapped 로 설정하였다.

3.3 결과분석

부팅시간 측정은 전원을 켜는 시점에 시작되는 entry 부터 사용자 프로그램이 시작되는 순간까지를 측정하였다.

압축해제시 데이터 Cache 의 효과를 확인하기 위해 다양한 크기의 이미지에 대해 시간을 측정/비교하였으며 실측결과는 다음과 같다.

Cache 여부	이미지 크기		
	128KB	248KB	324KB
Enable	0.696s	0.836s	1.032s
Disable	1.116s	1.736s	2.984s

표 2 실측결과

사용된 OS Image 는 압축된 것으로 원래의 이미지크기는 각각 302KB, 1.2MB, 2.1MB 이다.

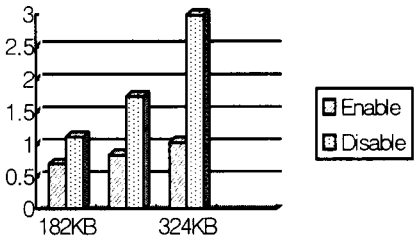

그림 3 실측결과 차트

결과에서 보는것처럼 Cache 가 사용되지 않을경우, 이미지크기가 36%증가하면 부팅시간이 56% 증가한다는 것을 볼 수 있다.

그렇지만, Cache 를 사용할 경우는 이미지의 크기가 36%증가하더라도 20%정도만 부팅시간이 늘어남으로써 이미지 크기의 증가에따라 linear 하게 증가하는 것을 확인할 수 있다.

4. 결 론

본 논문은 CE 제품의 부팅타임에 초점을 맞추고 Cache 를 사용하여 부팅타임을 개선하였다. 측정결과에 비추어 봤을 때, 부트로더 과정에서의 Cache 사용은 OS 의 이미지크기가 증가하더라도 작은 이미지를 사용할때의 부팅타임에 비해 큰 변화가 없으며, 대개의 CE 환경에서의 OS 이미지크기를 감안한다면, 부팅타임을 1 초이내로 단축시킬 수 있다는 것을 알 수 있다.

이것은 CE 제품에 OS 를 도입할 때 부팅타임의 지연이라는 문제에 대한 해결책으로 사용 가능할 것으로 생각된다.

[참 고 문 헌]

[1] Norman P. Jouppi, "Cache Write Policies and Performance", WRL Research

[2] Brian Kenworthy Bray, "Specialized Caches to improve data access performance", May 1993

[3] American National Standards Institute, "IEEE Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices", IEEE Std 1275-1994

[4] http://www.gzip.org, Jean-loup Gailly, Mark Adler

[5] WindRiver, "VxWorks BSP Developer's Guide 5.5" Edition 2, 28 May 03

[6] ARM "ARM920T(Rev 1) Technical Reference Manual" 18th April 2001