3

2024년 9월 9일 월요일 오전 8:36

부트로더에 대한 모든 것

시스템을 기본 수준으로 초기화 하고 커널을 로드

처음에 동작하는 자원은 하나의 CPU 코어와 약간의 온칩 정적 메모리, 부트 롬

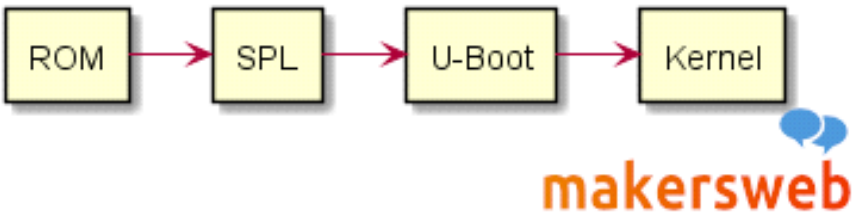
1. 부트로더는 하드웨어 구성 정보를 담고 있는 구조체의 포인터를 전달
2. 커널 명령줄(리눅스의 동작을 제어하는 문자열) 의 포인터를 전달

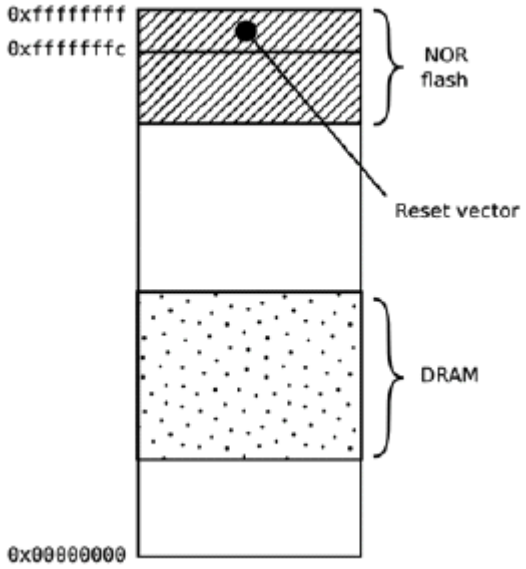
커널이 실행을 시작하면 부트로더는 더 이상 필요가 없고 사용하던 모든 메모리를 회수할 수 있음

부트로더의 부수작업 (시리얼 인터페이스를 통한 간단한 명령줄 인터페이스로 제어)

1. 부트 구성 업데이트
2. 새로운 부트 이미지를 메모리에 로드
3. 진단 기능을 실행하는 유지보수 모드를 제공

<부트 순서>

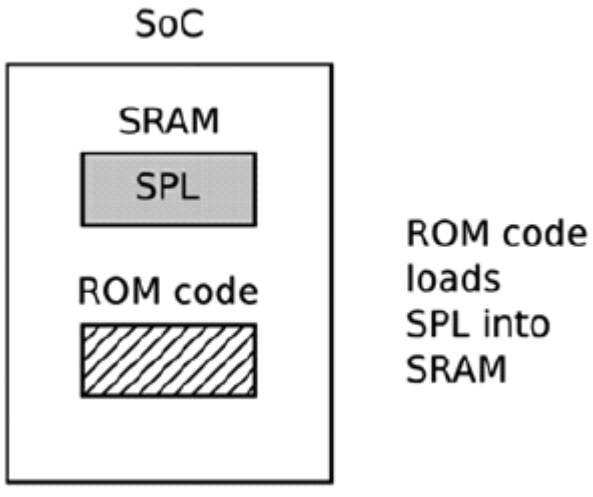




NOR 플래시 메모리에서 실행되는 부트로더 코드가 DRAM을 초기화
DRAM을 쓸 수 있게 만들어 스스로를 DRAM으로 복사
완전히 동작하게 되면, 부트로더는 커널을 플래시 메모리에서 DRAM으로 로드하고 제어를 넘길 수 있음

1. 롬 코드

- 리셋이나 전원을 켠 직후에 실행되는 코드로 SoC의 칩 상에 저장되어야 함
- 롬코드는 제조 시 칩에 프로그래밍 되므로, 비공개이고 오픈소스 대용품을 대체 불가능
- 메모리 제어기를 초기화하는 코드는 담고 있지 않아서 메모리 제어기가 필요 없는 SRAM만 사용 가능
- 대부분의 임베디드 SoC(System on Chip)설계는 약간의 SRAM을 칩 안에 갖고 있음 (4KB에서 수백 KB)

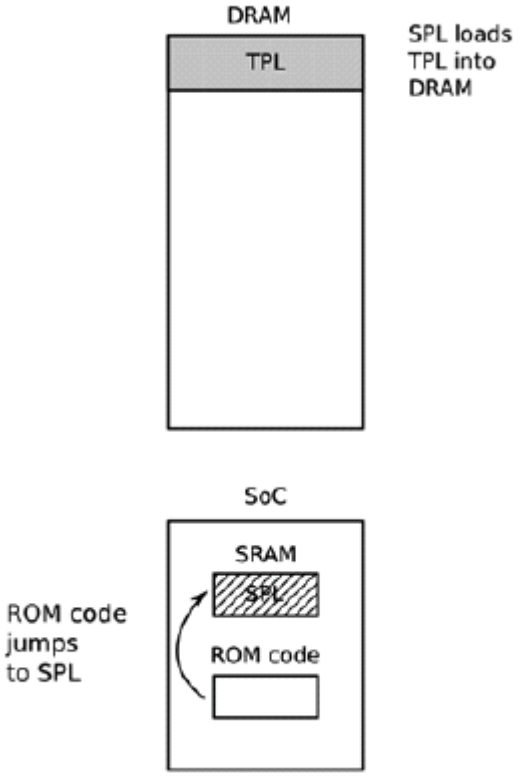


소량의 코드를 사전에 프로그래밍 된 몇 개의 위치 중 하나로부터 SRAM으로 로드

SRAM이 U-Boot같은 완전한 부트 로더를 로드할 정도로 충분히 크지 않은 SoC에는 SPL이라는 중간 로더가 존재
롬 코드 단계의 끝에서, 롬 코드는 SPL 코드의 시작으로 점프

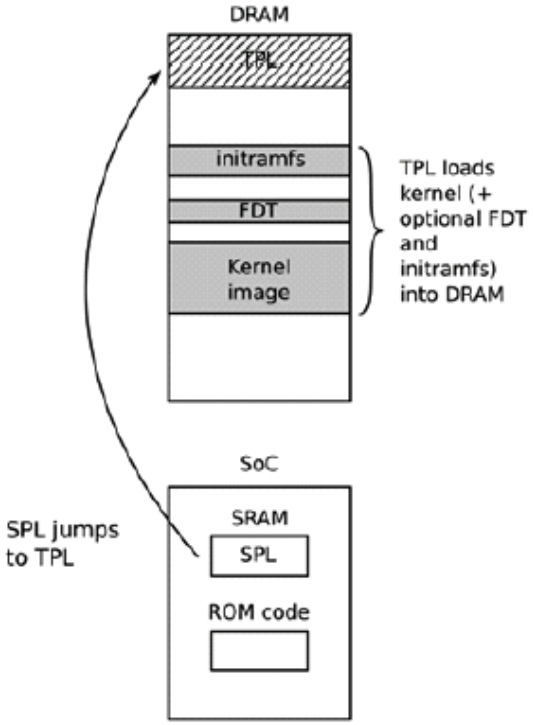
2. SPL (secondary program loader)

- SPL은 TPL을 DRAM에 로드하기 위해 메모리 컨트롤러 및 시스템의 필수적인 부분들을 초기화 해야 함
- SPL의 기능은 크기로 인해 제한
- 롬 코드처럼 다시 한 번 사전에 프로그래밍 된 플래시 장치의 시작부터의 오프셋을 이용해 일련의 저장 장치로부터 프로그램을 읽을 수 있음
- SPL에 파일 시스템 드라이버가 내장돼 있다면 디스크 파티션에서 U-boot.img처럼 잘 알려진 파일명을 읽을 수 있음
- SPL은 보통 사용자와 상호작용을 고려하지 않지만 버전 정보와 진행 메시지를 콘솔로 출력할 수 있음



SPL은 SRAM 내에서 실행될 때, TPL을 DRAM으로 로드

3. TPL(tertiary program loader)



- SRAM의 SPL에서 DRAM의 TPL로의 이동
- TPL이 실행되면 커널을 DRAM으로 로드
- 임베디드 부트로더는 커널이 실행되면 메모리에서 사라지고 시스템의 동작에 더 이상 관여하지 않음
- 그렇게 되기전에 TPL은 부트 절차의 제어를 커널로 넘겨야 함

<부트로더에서 커널로 이동>

부트로더가 제어를 커널로 넘길 때는 몇가지 기본 정보를 전달해야 함

1. 디바이스 트리가 지원되지 않는 파워PC와 ARM 플랫폼에서 SoC의 종류를 식별하기 위해 사용하는 기계 번호
2. 이제까지 발견된 하드웨어의 기본적인 세부사항
3. 커널 명령줄 (평범한 아스키 문자열)
4. 디바이스 트리 바이너리의 위치와 크기
5. 초기 램 파일시스템이라고 하는 초기 램디스크의 위치와 크기

거의 모든 ARM 시스템이 디바이스 트리를 이용해 하드웨어 플랫폼의 상세 정보를 저장하므로 단일 커널 바이너리를 광범위한 플랫폼에서 실행할 수 있음

<디바이스 트리 소개>

컴퓨터 시스템의 하드웨어 요소를 정의하는 유연한 방법
보통 디바이스 트리는 부트로더가 로드해서 커널에 넘김

디바이스 트리를 따로 로드할 수 없는 부트로더를 위해 커널 이미지 자체에 포함 시킬수도 있음

<레이블과 인터럽트>

디바이스 트리는 컴퓨터 시스템을 나무 같은 계층 구조로 결합된 요소의 묶음으로 나타냄
디바이스 트리의 구조는 여러 계층 구조 존재
연결들을 나타내기 위해 노드의 레이블을 추가하고 다른 노드에서 그 레이블을 참조할 수 있는데 이를 phandle이라고 함

<디바이스 트리 인클루드 파일>

많은 하드웨어가 같은 종류의 SoC 사이에, 그리고 같은 SoC를 사용하는 보드 사이에 공통적
이게 디바이스 트리에서도 반영돼서 공통 부분을 인클루드 파일로 분리할 수 있음

<디바이스 트리 컴파일 하기>

부트로더와 커널은 디바이스 트리의 바이너리 표현을 요구하므로, 디바이스 트리 컴파일러(dtc)로 컴파일 해야 함
.dtc로 끝나는 파일로 '장치 트리 바이너리'나 '장치 트리 블롭'이라고 함

커널과 마찬가지로, 부트로더는 디바이스 트리를 사용해 임베디드 SoC와 주변 기기를 초기화할 수 있음

<U-Boot>

임베디드 파워PC 보드용 오픈소스 부트로더로 시작됨
그 다음에 ARM 기반 보드로 이식, MIPS,SH 등 다른 아키텍처로 이식

```
user@raspberrypi:~/u-boot $ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mmcblk0     179:0    0 14.8G  0 disk
├─mmcblk0p1 179:1    0  256M  0 part /boot
└─mmcblk0p2 179:2    0 14.6G  0 part /
```

마이크로SD카드는 mmcblk0으로 나타나고 파티션은 mmcblk0p1으로 나타 냄

<U-Boot 사용하기>

시리얼 포트를 통해 명령줄 인터페이스를 제공
보드 별로 설정된 명령 프롬프트 제공

<부트 이미지 형식>

u-boot에는 파일시스템이 없음

```
$ mkimage -A arm -O linux -T kernel -C gzip -a 0x80008000 \
-e 0x80008000
-n 'Linux' -d zImage uImage
```

아키텍처 : arm
운영체제 : linux
이미지 유형 : kernel
압축 방식: gzip
로드 주소: 0x80008000
이미지 이름: Linux
이미지 데이터 파일 이름 : zImage
생성되는 이미지 이름 : uImage

Mmc rescan -> mmc 드라이버를 재초기화해서 SD 카드가 혹시 최근에 삽입 됐는지 알아낼 수 있음

<이미지 로드하기>

보통 SD카드 같은 탈착식 저장소나 네트워크로부터 이미지 로드
SD카드는 U-Boot에서 mmc 드라이버를 통해 처리

<리눅스 부팅>

Bootm [커널 주소] [램디스크주소] [dtb 주소]

-> 커널 주소는 필수이나 램디스크와 dtd주소는 커널 구성이 요구하지 않는다면 생략 가능

U-Boot 소스 구성

arch: 지원되는 아키텍처별 코드를 arm, mips, powerpc 등의 디렉터리에 담고 있으며, 각 아키텍처 안에는 해당 계열의 변종별 서브디렉토리가 있음

board: 보드별 코드를 담고 있으며, 같은 벤더에서 나온 여러 보드가 있으면, 서브 디렉토리로 묶을 수 있음
common: 명령 셸과 명령 셀에서 부를 수 있는 명령들을 포함하는 공통 핵심 기능을 담고 있음
doc: u-boot의 다양한 측면을 설명하는 README 파일을 담고 있음
Include: 여러 공유 헤더 파일 뿐만 아니라 매우 중요한 include/configs 서브 디렉토리를 담고 있음

<보드별 파일>

보드마다 board/[보드이름] 이나 board/[벤더]/[[보드 이름]이라는 이름의 서브 디렉토리가 있고, 그 안에는 다음과 같은 파일이 존재

Kconfig: 보드 구성을 담고 있음
MAINTAINERS: 보드가 현재 유지보수 되고 있는지, 누가 유지보수 하는지를 담고 있음
Makefile: 보드별 코드를 빌드하는데 쓰임
README: U-Boot의 이 이식판에 대한 유용한 정보를 담고 있음

<팔콘 모드>

임베디드 프로세서 부팅은 CPU 부트 ROM이 SPL을 로드하고, SPL 이 u-boot.bin 을 로드하고, u-boot.bin이 리눅스 커널을 로드하는 식
팔콘 모드는 SPL이 u-boot.bin을 건너 뛰고 커널 이미지를 직접 로드하게 하는 것
->부트 절차를 간단하고 빠르게 하는 방법