A lightweight bootloader based on MIPS architecture for mobile devices: research and implementation

Wei Dong

College of Electrical and Information Engineering, Southwest University for Nationalities, Chengdu 610041, China
brant.dong123@gmail.com

Keywords: MIPS architecture; mobile devices; lightweight bootloader; address map.

Abstract. A bootloader's main function is to initialize the hardware, pass an abstraction of the initialized hardware, a hardware description, to and execute the kernel. While most bootloaders concentrate on ARM or PowerPC architecture, in this paper we propose a lightweight bootloader based on MIPS architecture. The lightweight bootloader is mainly for mobile devices. First, we introduce the architecture of the bootloader and some special requirements for mobile devices are considered. Then, based on the architecture, we implement the bootloader in MIPS assembly language and C language. To validate our research, we test the bootloader in a real MIPS-based evaluation board. The results show that our bootloader works quite well. With MIPS technology's role becoming more and more important in mobile Internet, we hope that the bootloader developed in this paper will be utilized in the future.

Introduction

A bootloader's main function is to initialize the hardware, pass an abstraction of the initialized hardware, a hardware description, to and execute the kernel. While most bootloaders concentrate on ARM [1] or PowerPC architecture [2], research about bootloader based on MIPS architecture [3] is relatively less.

One reason for this phenomenon is that in embedded processor the market share for ARM and PowerPC is larger than that of MIPS embedded processor. Today, many smart phones and entertainment electronics are based on ARM processor. Another reason for this situation is that MIPS architecture is more complicated than that of ARM and in the industry ARM and PowerPC processors are more popular. The last few years have seen a massive growth of smart phones, and almost all smart phones are based on embedded processors such as ARM, PowerPC and MIPS. Moreover, with the mobile Internet services growing more and more important [4][5][6][7], there is a possibility that smart phones vendors will adopt MIPS architecture more and more because of its cheap prices and good performance.

In this paper we propose a lightweight bootloader based on MIPS architecture. The lightweight bootloader is mainly for mobile devices. First, we introduce the architecture of the bootloader and some special requirements for mobile devices are considered. Memory map of MIPS architecture and bootloader are carefully handled. Then, based on the architecture, we implement the bootloader in MIPS assembly language and C language.

The procedure of the lightweight bootloader is divided into two phase. The first phase conducts some low level initializations and it is based on assembly language. Based on the preparation of first stage, the second stage continues to do some initialization and load the kernel to run. However, the second stage is based on high level language, i.e., the C programming language. Although there are some simulators for MIPS architecture [8] [9], it is not suitable for bootloader validation. To validate our research, we test the bootloader in a real MIPS-based evaluation board.

The rest of this paper is organized as follows. In section II the architecture of the lightweight bootloader is introduced. Section three describes the implementation of the lightweight bootloader software. And we also conduct some tests on this bootloader in section III. In the last section, conclusion is made.
Design of a lightweight bootloader based on MIPS architecture

The way MIPS processors use and handle addresses is subtly different from that of traditional CISC CPUs and other RISC CPUs [11], and the physical address will never be the same with the program address [10].

The memory map of a 32-bit MIPS process is illustrated in Figure 1. As can be seen from this figure, the memory map is divided into four areas [10]. The first area is the user segment with a size of 1GB. Normally, addresses in this area must be translated by a Memory Management Unit (MMU). The second and third area have the same size of 512MB, and there is only one difference between these two areas.

Addresses in both areas don’t need to be mapped and the second area will be cached by the CPU while the third area is not cacheable. Like the first area, to access addresses of last area, address translation must be conducted. Moreover, this area is only accessed in kernel mode.

Before designing bootloader based on MIPS architecture, we must first decide in which area the bootloader code will be located. Since bootloader code is the first thing to run after power is applied to a system, the third area is the only option for bootloader code. There are several reasons for this. First, before a system is up there are no MMU tables, the first and the last area are both not proper for the bootloader code. Because cache related initialization is not conducted before a system is up, the second part of memory is not suitable for bootloader code.

Once the area for bootloader code is decided, the address map of bootloader must be determined. In our design, Executable and Linkable Format (ELF) [12] is adopted. It is must be noted that the base address of code segment must be equal to 0xbfc00000 since MIPS will fetch code from this address after power is applied to a system. The memory map of the bootloader is shown in Figure 2. The memory map consists of several parts, i.e. code segment, data segment, global offset table and Bss segment [12].
System implementation and test

Based on the design of last section, we implement a lightweight bootloader based on MIPS architecture. And this bootloader is mainly for mobile devices. Unlike a complicated embedded system, this bootloader will only do some necessary initializations. For instance, MMU is not utilized in our bootloader, and no initialization is done about MMU hardware.

The procedure is divided into two stages according the code running environment, which is shown in Figure 3. The first stage is implemented in MIPS assembly language and the code is running with a lot of restrictions. There are several reasons for this situation. First, because initializations about the system are not conducted at this time, the code must be run from ROM. Normally, the ROM is a flash memory.

In order to running bootloader code in RAM, RAM must be initialized beforehand. Second, at this stage some operations can only be done in assembly language. Actually, the main function of the first stage is to set up conditions which must be fulfilled in order to run a C program. Consequently, in this stage, all kinds of initializations are conducted. As can be seen from Figure 3(a), these initializations include CPU initialization, global pointer initialization, cache enable, board-specific initialization and RAM initialization.

Since in first stage C running environment is set up, in the second stage we must relocate bootloader code into memory. In the first stage, code is running from flash memory and only read operation is allowed. In order to do some write operation, the code must be moved into RAM. Consequently, in stage two code relocation is conducted. After that, all other operations will be done in RAM. These operations include timer initialization, interrupts enable. Most important of all, Linux kernel can be load into memory and make it run by the bootloader.

![Flowcharts of the bootloader](image)

Fig.3. Flowcharts of the bootloader (a) first stage and (b) second stage

In order to evaluate our research and implementation, we test our bootloader code in a real MIPS based evaluation board. The board is targeted for mobile devices. We conduct several experiments and the test results show that our bootloader works quite well. The bootloader can initialize all hardware correctly. Moreover, a precompiled Linux kernel is loaded into memory and the system works quite well as expected.
Summary

While most bootloaders concentrate on ARM or PowerPC architecture, in this paper we propose a lightweight bootloader based on MIPS architecture. The lightweight bootloader is mainly for mobile devices. First, we introduce the architecture of the bootloader and some special requirements for mobile devices are considered.

Then, based on the architecture, we implement the bootloader in MIPS assembly language and C language. To validate our research, we test the bootloader in a real MIPS-based evaluation board. The hardware can be correctly initialized and a Linux kernel can be successfully loaded into memory and start to run. The results show that our bootloader works quite well. With MIPS technology's role becoming more and more important in mobile Internet, we hope that the bootloader developed in this paper will be utilized in the future.

Acknowledgements

This paper was sponsored by the Fundamental Research Funds for the Central Universities (Southwest University for Nationalities, Project No. 13NZYBS03).

References