# Desenvolvimento de BootLoaders

## Técnicas de Programação para Sistemas Embarcados II

Prof. Francisco Helder

Universidade Federal do Ceará

August 18, 2022

# Bootloaders

**O bootloader é parte do sistema responsável por:**

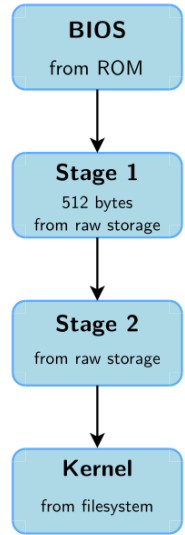
1. Inicialização de hardware básico.
2. Carregamento de um binário, geralmente um kernel do sistema operacional, do armazenamento flash, da rede ou de outro tipo de armazenamento não volátil.
3. Possivelmente descompressão do binário.
4. Execução do aplicativo.

Além dessas funções básicas, a maioria dos gerenciadores de inicialização fornece um shell com vários comandos implementando diferentes operações.

- Carregamento de dados de armazenamento ou rede, inspeção de memória, diagnóstico e teste de hardware, etc.

# Bootloaders do x86 baseado em BIOS

1. Os processadores x86 são normalmente agrupados em memória não volátil contendo a BIOS.
2. Em antigas x86: BIOS é responsável pela inicialização básica do hardware e pelo carregamento de partes do código da não volátil.
3. Este pedaço de código é tipicamente um bootloader de $1^{\underline{o}}$ estágio, que carregará o próprio bootloader completo.
4. Ele normalmente entende os formatos do sistema de arquivos para que o kernel possa ser carregado diretamente de um sistema de arquivos normal.
5. Essa sequência é diferente para sistemas modernos baseados em EFI.



# Bootloaders no x86

**GRUB, Grand Unified Bootloader**

https://www.gnu.org/software/grub/
Pode ler muitos formatos de sistema de arquivos para carregar a imagem do kernel e a configuração, fornece um shell poderoso com vários comandos, pode carregar imagens do kernel pela rede, etc.

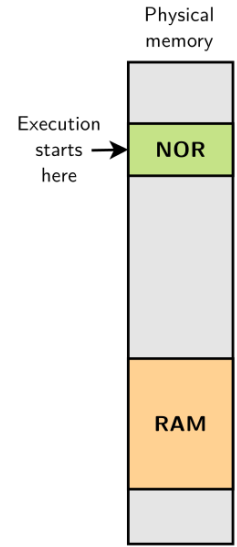
Syslinux, para inicialização em rede e mídia removível (chave USB, CD ROM)
https://kernel.org/pub/linux/utils/boot/syslinux/

Systemd-boot, um gerenciador de inicialização UEFI muito simples (anteriormente Gummiboot)

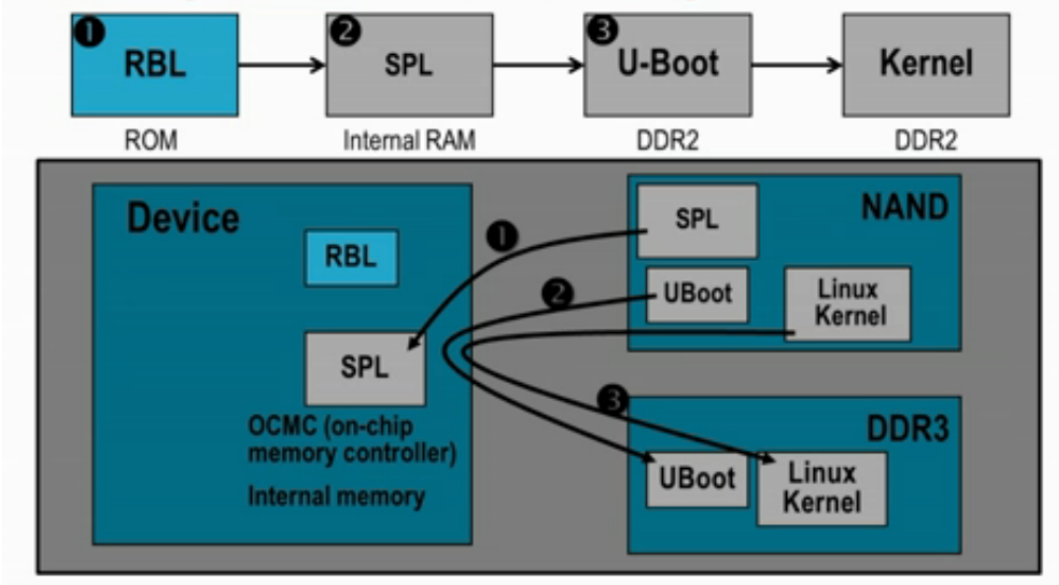
- Claro, não baseado em Systemd, mas hospedado por este projeto.

# Inicializando em CPUs embarcadas: caso 1

1. Quando energizado, a CPU começa a executar o código em um endereço fixo
2. Não há outro mecanismo de inicialização fornecido pela CPU
3. O design de hardware deve garantir que um chip flash NOR seja conectado de forma que seja acessível no endereço em que a CPU começa a executar as instruções
4. O bootloader de primeiro estágio deve ser programado neste endereço na NOR
5. NOR é obrigatório, pois permite acesso direto da CPU (assim como a RAM), a NAND não permite.
6. Não é mais muito comum (não é prático e requer NOR instantâneo)



# Inicializando em CPUs embarcadas: caso 2



# Bootloaders Genérico para CPUs Embarcadas

Existem vários bootloaders genéricos de código aberto, abaixo estão os mais populares:

**U-Boot - Universal Bootloader da Denx**

- O mais usado em ARM, também usado em PPC, MIPS, x86, m68k, RISC-V, etc.
- O padrão atualmente. Vamos estudá-lo em detalhes.
- https://www.denx.de/wiki/U-Boot

**BareBox - creada pela Pengutronix**

- Ainda não tem tanto suporte de hardware quanto o U-Boot.
- O U-Boot melhorou bastante graças a este concorrente.
- https://www.barebox.org

# U-boot

U-Boot é um típico projeto de software livre

- Licença: GPLv2 (igual ao Linux)
- Disponível gratuitamente em https://www.denx.de/wiki/U-Boot
- Documentação disponível em https://u-boot.readthedocs.io/en/latest/
- O código-fonte de desenvolvimento mais recente está disponível em um repositório Git: https://gitlab.denx.de/u-boot/u-boot
- O desenvolvimento e as discussões acontecem em torno de uma lista de discussão aberta https://lists.denx.de/pipermail/u-boot/
- Segue um cronograma de lançamento regular. A cada 2 ou 3 meses, uma nova versão é lançada. As versões são denominadas YYYY.MM.


Image credits:
https://frama.link/rwCUFc-T

# U-boot - Configuração

1. Obtenha o código-fonte do site ou do git
2. O diretório **configs/** contém vários arquivos de configuração para várias placas
   - Define o tipo de CPU, os periféricos e sua configuração, o mapeamento de memória, os recursos do U-Boot que devem ser compilados, etc.
   - Exemplos: **configs/am335x_evm_defconfig** ou **configs/am335x_avm_spiboot_defconfig**

**Nota:**

U-Boot está migrando da configuração da placa definida nos arquivos de cabeçalho C (**include/configs/**) para defconfig como no kernel Linux (**configs/**)

- Nem todas as placas foram convertidas para o novo sistema de configuração.
- Muitas placas ainda têm configuração codificadas em arquivos **.h** e configuração em arquivos defconfig que podem ser substituídos por interfaces de configuração.

# Arquivo de Configuração U-Boot

**CHIP_defconfig**

```
CONFIG_ARM=y
CONFIG_ARCH_CPU_INIT=y
CONFIG_ARCH_OMAP2PLUS=y
CONFIG_TI_COMMON_CMD_OPTIONS=y
CONFIG_DEFAULT_DEVICE_TREE="am335x-evm"
CONFIG_AM33XX=y
CONFIG_SPL=y
CONFIG_DISTRO_DEFAULTS=y
CONFIG_TIMESTAMP=y
CONFIG_SPL_LOAD_FIT=y
# CONFIG_USE_SPL_FIT_GENERATOR is not set
CONFIG_OF_BOARD_SETUP=y
CONFIG_BOOTCOMMAND="run findfdt; run init_console; run finduuid
   ; run distro_bootcmd"
CONFIG_LOGLEVEL=3
```

# Configurando e Compilando U-Boot

- O U-Boot deve ser configurado antes de ser compilado
  - Configuração armazenada em um arquivo .config "**make BOARDNAME_defconfig**"
  - Onde BOARDNAME é o nome de uma configuração, conforme visível no configs/directory.
  - Você pode então executar make menuconfig para personalizar ainda mais a configuração do U-Boot!
- Certifique-se de que o cross-compilador esteja disponível em PATH
- Compile o U-Boot, especificando o prefixo do cross-compilador. Exemplo: se o cross-compilador for **arm-linux-gcc**: **make CROSS_COMPILE=arm-linux-**
- O resultado principal é um arquivo **u-boot.bin**, que é a imagem do U-Boot. Dependendo da sua plataforma específica ou de qual dispositivo de armazenamento você está inicializando (NAND ou MMC), pode haver outras imagens especializadas: **u-boot.img**, **MLO**.
- Isso também gera a imagem U-Boot SPL, o qual o nome exato pode variar, dependendo do que o romcode espera.

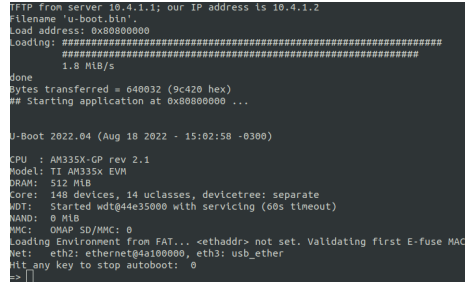
# Instalando U-Boot

O U-Boot geralmente deve ser instalado na memória flash para ser executado pelo hardware. Dependendo do hardware, a instalação do U-Boot é feita de forma diferente:

- A CPU fornece algum tipo de inicialização específica, que pode comunicar através da porta serial ou USB usando um protocolo específico
- A CPU inicializa primeiro na mídia fixa (NAND) ao invés da mídia removivel (MMC). Nesse caso, inicialize a partir do MMC para atualizar uma nova versão
- O U-Boot já está instalado e pode ser usado para atualizar uma nova versão do U-Boot. No entanto, tenha cuidado: se a nova versão do U-Boot não funcionar, a placa fica inutilizável
- A placa possui uma interface JTAG, que permite escrever remotamente na memória flash, sem que nenhum sistema rode na placa. Também permite resgatar uma placa se o bootloader não funcionar.

# U-boot Prompt

- Conecte a placa por meio de um console serial.
- Ligue a placa. No serial, você deverá ver o U-Boot inicializando.
- O shell U-Boot oferece um conjunto de comandos. Estudaremos os mais importantes, veja a documentação para ajuda.



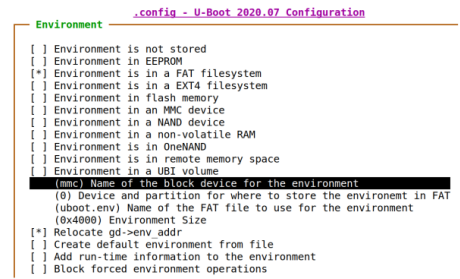
# Importantes Comandos

| | |
|---|---|
| ping | to test the network |
| bootd | (can be abbreviated as boot), runs the default boot command, stored in the bootcmd environment variable (explained later) |
| bootz | starts a compressed kernel image loaded at the given address in RAM |
| usb | to initialize and control the USB subsystem, mainly used for USB storage devices such as USB keys |
| mmc | to initialize and control the MMC subsystem, used for SD and microSD cards |
| nand | to erase, read and write contents to NAND flash |
| md | displays memory contents. Can be useful to check the contents loaded in memory, or to look at hardware registers. |
| mm | modifies memory contents. Can be useful to modify directly hardware registers, for testing purposes. |

# Variáveis de Ambiente: implementação

Dependendo da configuração, o ambiente U-Boot é normalmente armazenado em:

- deslocamento fixo na NAND flash
- deslocamento fixo no armazenamento MMC ou USB, antes do início da primeira partição.
- arquivo (uboot.env) em uma partição FAT ou ext4
- volume UBI



# Prática de lab - U-boot

Hora de começar o laboratório prático!

1. Comunique-se com a placa via serial
2. Configurar, construir e instalar U-Boot
3. Aprenda os comandos do U-Boot
4. Comunicar via TFTP com a placa