LABORATÓRIO TPSE II

Lab 02: Compilando e Gravando U-boot

Prof. Francisco Helder

18 de agosto de 2023

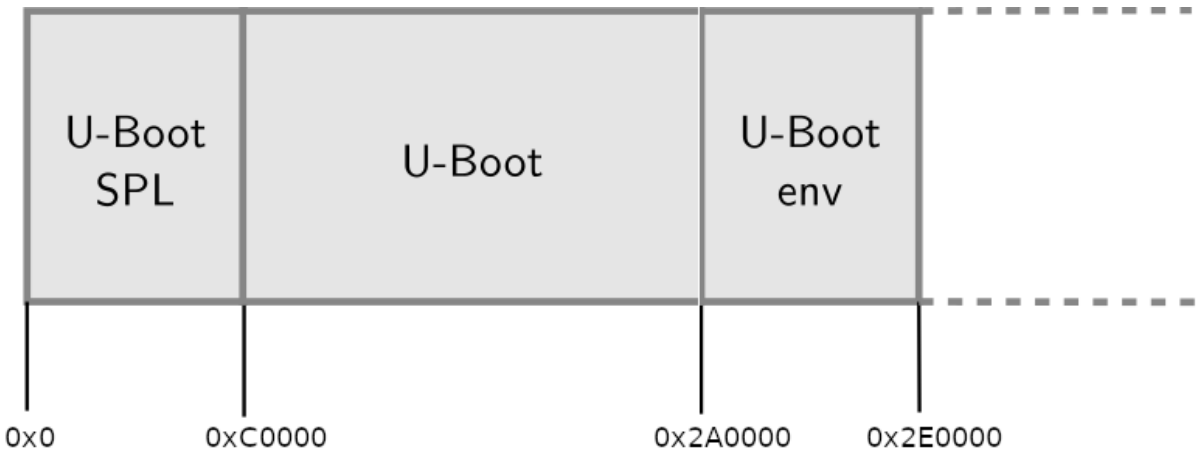
Como o bootloader é o primeiro software executado por uma plataforma de hardware, o procedimento de instalação do bootloader é muito específico para a plataforma de hardware. Geralmente ocorrem dois casos:

- O processador não oferece nada para facilitar a instalação do bootloader, caso em que uma interface JTAG deve ser usada para inicializar o armazenamento flash e escrever o código do bootloader para flash. É claro que um conhecimento detalhado do hardware é necessário para realizar essas operações.
- O processador oferece um sistema, implementado em ROM, através do qual é facilitado o acesso às memórias.

A placa BeagleBone Black, que usa os SoCs AM335x, se enquadra na segunda categoria. O sistema integrado na ROM lê o cartão MMC/SD para procurar um bootloader válido antes de olhar para o flash NAND interno para um bootloader. Caso nada esteja disponível, ele operará em modo fallback, que permitirá usar uma ferramenta externa para atualizar algum bootloader através de USB. Portanto, usando um cartão MMC/SD ou esse modo de fallback, podemos iniciar uma placa baseada em AM335x sem ter nada instalado nela.

1 Configurando Bootloader

O processo de inicialização é feito em duas etapas com o sistema ROM tentando executar um primeiro software, chamado U-Boot SPL (*Single Program Loader*), a partir de sua SRAM interna. Ele inicializará a DRAM, carregará o U-Boot que, por sua vez, carregará o Linux. No que diz respeito aos bootloaders, o layout da NAND flash é semelhante a:



- O deslocamento 0x0 para o bootloader de primeiro estágio é determinado pelo hardware: o código ROM do AM335x procura um bootloader de primeiro estágio no deslocamento 0x0 no flash NAND.
- O deslocamento 0xC0000 para o carregador de inicialização do segundo estágio é decidido pelo carregador de inicialização do primeiro estágio. Isso pode ser alterado modificando o valor de CONFIG_SYS_NAND_U_BOOT_OFFS em configs/am335x_evm_defconfig em fontes U-Boot.
- Offset 0x2A0000 do ambiente U-Boot é decidido pela configuração do U-Boot (CONFIG_ENV_OFFSET).

2 Configurando e compilando U-Boot

Obtenha os fontes do U-Boot e escolha uma versão estável:

```
$ git clone https://gitlab.denx.de/u-boot/u-boot
$ cd u-boot
$ git checkout v2023.04
```

Compreenda as etapas de configuração e compilação do U-Boot lendo o arquivo README e, especificamente, a seção Construindo o Software. Basicamente, você precisa:

1. Defina a variável de ambiente CROSS_COMPILE;
2. Execute **make <NAME>_defconfig**, onde a lista de configurações disponíveis pode ser encontrada no diretório configs/. Existem alguns tipos de configuração do am335x: um para executar a partir do spi (am335x_evm_spiboot) e outro para executar a partir do sdcard (am335x_evm). Como vamos inicializar no sdcard, use o último.
3. Agora que você tem uma configuração válida, você pode executar **make menuconfig** para editar ainda mais os recursos do seu carregador de inicialização.

Instale os seguintes pacotes que podem ser necessários para compilar o U-Boot para sua placa:

```
$ sudo apt install libssl-dev device-tree-compiler
```

4. Por fim, execute make, que deve construir os dois estágios do U-Boot:
   - Bootloader de primeiro estágio (SPL): spl/u-boot-spl.bin
   - Bootloader de segundo estágio: u-boot.bin

Veja o tamanho dos binários. spl/u-boot-spl.bin deve caber no SoC SRAM (64 KB) e de acordo com nosso layout flash, u-boot.bin deve caber entre flash offset 0xC0000 e offset 0x2A0000, correspondendo a um tamanho máximo de 1966080 bytes. Certifique-se de que ambos os binários se encaixem.

2.1 Compilando U-Boot

Como o U-Boot suporta diferentes plataformas de hardware, ele deve ser configurado antes de ser compilado. Portanto, execute o comando abaixo para configurar o U-Boot para o target:

```
$ make CROSS_COMPILE=arm-linux-gnueabihf-  distclean
```

Agora podemos iniciar a compilação:

```
$ make CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
$ make CROSS_COMPILE=arm-linux-gnueabihf- am335x_evm_config
$ make CROSS_COMPILE=arm-linux-gnueabihf- -j4
```

Perceba que passamos o prefixo do compilador na variável CROSS_COMPILE. O parâmetro -j permite paralelizar a compilação (4 threads de compilação simultânea no comando acima).

Após a compilação, verifique se foram geradas as imagens do U-Boot: Compilação deve concluir com uma saída semelhante ao mostrando abaixo, uma imagem do MLO - o primeiro estágio de inicialização é gerado.

```
$ ls -lha spl/u-boot-spl*
LD              spl/u-boot-spl
OBJCOPY         spl/u-boot-spl.bin
CFG             spl/u-boot-spl.cfg
```

Você deve encontrar no raiz a segunda fase do bootloader U-Boot (u-boot.img) gerado.

```
$ ls -lha u-boot-*
LDS             u-boot.lds
OBJCOPY         u-boot.bin
MKIMAGE         u-boot.img
OBJCOPY         u-boot.srec
CFG             u-boot.cfg
```

2.2 Testando U-boot

Vamos testar o segundo estágio do bootloader carregando na RAM, reinicie a placa e verifique se ela inicializa seus novos bootloaders:

```
U-Boot# mw 0x44e35048 0xaaaa; sleep 1; mw 0x44e35048 0x5555;
U-Boot# set autoload no
U-Boot# set ipaddr 10.4.1.2
U-Boot# set serverip 10.4.1.1
U-Boot# tftp 0x80800000 u-boot.bin
U-Boot# go 0x80800000
```

Você pode verificar isso verificando as datas de construção:

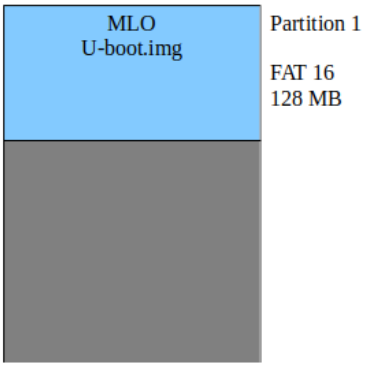
```
TFTP from server 10.4.1.1; our IP address is 10.4.1.2
Filename 'u-boot.bin'.
Load address: 0x80800000
Loading: #################################################################
         #################################################################
         1.8 MiB/s
done
Bytes transferred = 640032 (9c420 hex)
## Starting application at 0x80800000 ...


U-Boot 2022.04 (Aug 18 2022 - 15:02:58 -0300)

CPU  : AM335X-GP rev 2.1
Model: TI AM335x EVM
DRAM:  512 MiB
Core:  148 devices, 14 uclasses, devicetree: separate
WDT:   Started wdt@44e35000 with servicing (60s timeout)
NAND:  0 MiB
MMC:   OMAP SD/MMC: 0
Loading Environment from FAT... <ethaddr> not set. Validating first E-fuse MAC
Net:   eth2: ethernet@4a100000, eth3: usb_ether
Hit any key to stop autoboot:  0
=>
```

2.3 Gravando o U-boot no sdCard

No nosso caso, o boot será feito pelo cartão SD. Portanto, gravaremos o U-Boot na posição inicial do cartão SD para que o código de boot em ROM possa carregá-lo para a RAM interna e executá-lo no boot. O cartão SD terá a seguinte formatação:



2.3.1 Preparando o sdCard

Vamos usar um cartão SD para o nosso dispositivo. Conecte o cartão SD na sua máquina, e digite o comando dmesg para ver qual dispositivo é usado por sua estação de trabalho. Caso o dispositivo seja /dev/mmcblk0, você verá algo como:

```
[124.7298] mmc0: card aaaa removed
[128.0155] mmc0: new ultra high speed SDR50 SDXC card at address aaaa
[128.0161] mmcblk0: mmc0:aaaa SC64G 59.5 GiB
```

O nome do arquivo do dispositivo pode ser diferente (como /dev/sdb se o leitor de cartão estiver conectado a um barramento USB (dentro de seu PC ou usando um leitor de cartão USB). Nas instruções a seguir, assumiremos que seu cartão SD é visto como /dev/mmcblk0 pela sua máquina, então digite o comando mount para verificar suas partições montadas no momento. Se as partições SD estiverem montadas, desmonte-as:

```
$ sudo umount /dev/mmcblk0p*
```

Em seguida, limpe o possíveis conteúdos do cartão SD (apenas os primeiros megabytes são importantes):

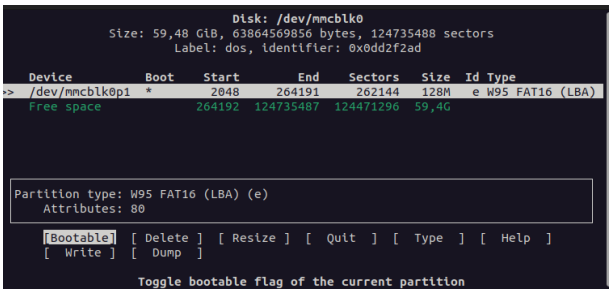
```
$ sudo dd if=/dev/zero of=/dev/mmcblk0 bs=1M count=256
```

Agora, vamos usar o comando cfdisk para criar as partições que precisamos:

```
$ sudo cfdisk /dev/mmcblk0
```

Se o cfdisk solicitar que você selecione um tipo de rótulo, escolha **dos**. Isso corresponde às tabelas de partições tradicionais que o DOS/Windows entenderia. As tabelas de partição gpt são necessárias para discos maiores que 2 TB, como vista na Figura 2.3.1.

Na interface do cfdisk, exclua as partições existentes e crie uma partição primária, começando do início, com a seguinte propriedade: partição de **128 MB**, com o tipo de partição **W95 FAT32 (LBA)**, e configure a partição como **Bootable**. Eventualmente, armazenaremos os arquivos do bootloader de inicialização nesta partição, como visto na Figura 2.3.1.



Pressione **Write** quando terminar. Para certificar-se de que as definições de partição sejam recarregadas em sua máquina, remova o cartão SD e insira-o novamente, então digite o seguinte comando:

```
$ cat /proc/partitions | grep mmc
179        0   62367744 mmcblk0
179        1     131072 mmcblk0p1
```

Usando o comando **mkfs.vfat** formate a partição criada certifique-se que a partição não está montada):

```
sudo mkfs.vfat -F 32 -n BOOT /dev/mmcblk0p1
```

Copie as imagem do U-boot geradas, de primeiro e segundo estágios:

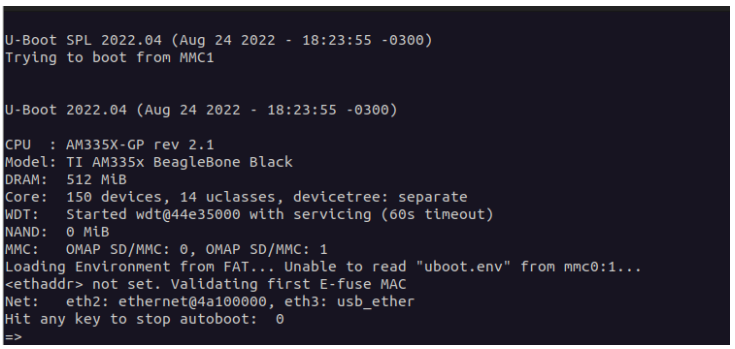
```
$ cp MLO /media/heldercs/BOOT
$ cp u-boot.img /media/heldercs/BOOT
```

Agora desmonte a partição e teste na placa:

```
$ umount /media/heldercs/BOOT
```

2.4 Bootando do sdCard

Mantendo pressionado o botão S2 permite a inicialização a partir da interface MMC0 (cartão de SD). Se você deseja sempre inicializar a partir do Cartão, você deve apagar o conteúdo da MMC on-board. O código ROM tentará carregar um gerenciador de inicialização a partir do MMC on-board, que irá falha, e então passa para a próxima interface, o cartão SD. A inicialização via cartão SD é mostrado na Figura 2.4.



3 Atividades Práticas

pratica 1

Altera a opção de parada o autoboot: Exemplo: mude de espaço para qualquer tecla.
De:

```
====================
Watchdog enabled
I2C:   ready
DRAM:  512 MiB
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
Using default environment

Press SPACE to abort autoboot in 2 seconds
====================
```

Para:

```
====================
Watchdog enabled
I2C:   ready
DRAM:  512 MiB
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
Using default environment

Hit any key to stop autoboot:  0
==================
```

pratica 2

Adcione um novo comando

1. Gere um arquivo ".c" no direório **cmd**

```
$ vim cmd/cmd_membench.c
```

2 Inclua o código abaixo:

```
#include <common.h>
#include <command.h>

int do_membench(cmd_tbl_t *cmdtp, int flag, int argc, char *argv[]){
    printf("disciplina embarcados\n\n");
    printf("arg1: %s\n", argv[1]);
}

U_BOOT_CMD(membench,5,1,do_membench,"memory benchmark", "arg1: start address");
```

3 Modifique o Makefile do diretório **cmd**, adcionando a seguinte linha:

```
obj-y += cmd_membench.o
```

4 Gere a nova image do U-boot e teste.

pratica 3

Altere as variáveis default de ambiente

1. Exemplo, alterar o scrip bootcmd

De:

```
bootcmd ``setenv umsmedia 0; gpio set 53; i2c mw 0x24 1
   0x3e; run findfdt; setenv mmcdev 0; setenv bootpart
   0:1; run mmcboot;gpio clear 56; gpio clear 55; gpio
   clear 54; setenv mmcdev 1; setenv bootpart 1:1; run
   mmcboot;run failumsboot;''
```

Para:

```
bootcmd ``mw 0x44e35048 0xaaaa; sleep 1; mw 0x44e35048 0
   x5555; setenv autoload no;dhcp; setenv serverip
   10.4.1.1;tftp 0x80000000 download.bin;echo ***
   Booting to BareMetal ***;go 0x80000000;''
```

2 Gere a nova imagem e teste.

pratica 4

Altere o comando gpio, faça uma análise detalhado do comando gpio no U-boot e realize a seguinte alteração.

1. Criar uma opção de set/clear do gpio (Ex: comando para setar um conjunto de GPIO).

```
$ gpio set 53-56
```

Esse comando deve setar os 4 pinos dos LEDs na placa.

2 Criar uma opção de blick para cada pino (Ex: comando gerar um pisca do LED).

```
$ gpio blk 53
```

Esse comando deve pisca o especifico LED na placa.