

LABORATÓRIO TPSE I



Prática 01: Ambiente de Desenvolvendo e Primeiro Sistema na BBB

Prof. Francisco Helder

11 de abril de 2022

1 Carregando a Primeira Aplicação Bare Metal

Esta primeira prática de laboratório tem o objetivo de orientar você a configurar o ambiente de desenvolvimento para programação em sistema embarcado, utilizando o bootloader U-Boot para carregar o sistema inicial da placa, também definida por aplicação de bare metal via TFTP.

1.1 Instalando o cross-compilador

Um cross-compilador nada mais é do que um compilador para uma plataforma diferente do computador usado no desenvolvimento. Como por exemplo um computador com a arquitetura x86 compilando para uma placa ARM. O cross-compilador que será usado será o arm-none-eabi.

1.1.1 Baixando o arm-none-eabi

O pacote com o arm-none-eabi pode ser encontrado no site da GNU e no site da própria ARM. Acesse esse link <https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads> e baixe diretamente do site da ARM a opção bare metal para cross-compilar em uma máquina x86 (**gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2**). Ou use os seguintes comandos para fazer o download.

```
$ mkdir lab
$ cd lab/
$ wget https://developer.arm.com/-/media/Files/downloads/gnu-rm/10.3-2021.10/gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2 -O gcc-arm-none-eabi-linux.tar.bz2
```



```
helder@machine:lab$ wget https://cutt.ly/qmqKmdB -O gcc-arm-none-eabi-10-2020-q4-major-x86_64-linux.tar.bz2
--2021-06-25 16:54:11-- https://cutt.ly/qmqKmdB
Resolving cutt.ly (cutt.ly)... 2606:4700:10::6816:1e8, 2606:4700:10::6816:e8, 2606:4700:10::ac43:8ee, ...
Connecting to cutt.ly (cutt.ly)[2606:4700:10::6816:1e8]:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://developer.arm.com/-/media/Files/downloads/gnu-rm/10-2020q4/gcc-arm-none-eabi-10-2020-q4-major-x86_64-linux.tar.bz2?revision=ca0cbf9c-9de2-491c-ac48-898b5bbc0443&la=en&hash=68760A8AE66026BCF99F05AC017A6A50C6FD832A [following]
--2021-06-25 16:54:12-- https://developer.arm.com/-/media/Files/downloads/gnu-rm/10-2020q4/gcc-arm-none-eabi-10-2020-q4-major-x86_64-linux.tar.bz2?revision=ca0cbf9c-9de2-491c-ac48-898b5bbc0443&la=en&hash=68760A8AE66026BCF99F05AC017A6A50C6FD832A
Resolving developer.arm.com (developer.arm.com)... 104.72.205.90
Connecting to developer.arm.com (developer.arm.com)[104.72.205.90]:443... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: https://armkeil.blob.core.windows.net/developer/Files/downloads/gnu-rm/10-2020q4/gcc-arm-none-eabi-10-2020-q4-major-x86_64-linux.tar.bz2 [following]
--2021-06-25 16:54:12-- https://armkeil.blob.core.windows.net/developer/Files/downloads/gnu-rm/10-2020q4/gcc-arm-none-eabi-10-2020-q4-major-x86_64-linux.tar.bz2
Resolving armkeil.blob.core.windows.net (armkeil.blob.core.windows.net)... 52.239.137.100
Connecting to armkeil.blob.core.windows.net (armkeil.blob.core.windows.net)[52.239.137.100]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 156882554 (150M) [application/octet-stream]
Saving to: 'gcc-arm-none-eabi-10-2020-q4-major-x86_64-linux.tar.bz2'

gcc-arm-none-eabi-10-2020- 30%[=====] 45,24M 912KB/s eta 64s
```

Figura 1: instalando o cross-compilador.

Após baixado o cross compilador, será preciso criar uma pasta na pasta da disciplina, nesse caso siga os seguintes passos no terminal:

```
$ mkdir toolchain
```

Após a criação da pasta é preciso descompactar o arquivo dentro do diretório **toolchain**, digitando o seguinte comando:

```
$ tar jxvf gcc-arm-none-eabi-linux.tar.bz2 -C toolchain/
```

mbre-se de verificar o nome do arquivo antes de copiar e descompactar, pois obviamente se tiver um nome diferente, o linux não vai encontrar!!!

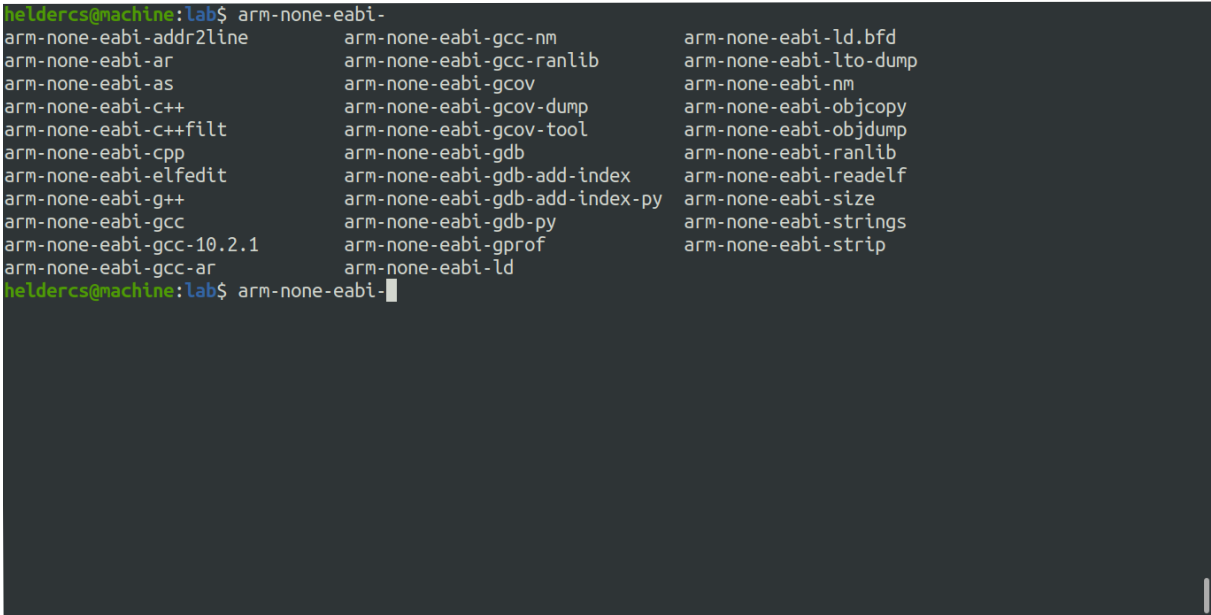
Com isso, já temos o ambiente de cross-compilação, porém para que ele seja acessado de qualquer parte do computador é preciso que seja mudada uma variável de ambiente chamada **\$PATH**, para isso é preciso modificar o arquivo **.bashrc** que está no diretório do usuário, siga os seguintes passos para a modificação da variável.

```
$ cd ~  
$ gedit .bashrc
```

Em que a pasta “~” é o diretório principal do usuário e **gedit** é um editor de textos (pode ser usado qualquer outro). Após abrir esse arquivo é preciso adicionar a seguinte linha no final do arquivo:

```
PATH=$PATH:~/lab/toolchain/gcc-arm-none-eabi-10.3-2021.10/bin
```

Veja que **~/lab/toolchain/gcc-arm-none-eabi-10.3-2021.10/bin** é o caminho da pasta que foi criado, portanto adicione o caminho da sua pasta, que pode ser visto no terminal com o comando “**pwd**”. Caso o caminho contenha espaços, como por exemplo **Área de Trabalho**, é necessário uma barra invertida (\), portanto, ficaria **Área\de\Trabalho**. Verifique seu caminho e cole corretamente no arquivo **~bashrc**. Após configurar a variável **\$PATH**, o cross-compilador já pode ser usado normalmente.



```
heldercs@machine:lab$ arm-none-eabi-  
arm-none-eabi-addr2line      arm-none-eabi-gcc-nm        arm-none-eabi-ld.bfd  
arm-none-eabi-ar            arm-none-eabi-gcc-ranlib    arm-none-eabi-lto-dump  
arm-none-eabi-as           arm-none-eabi-gcov          arm-none-eabi-nm  
arm-none-eabi-c++          arm-none-eabi-gcov-dump     arm-none-eabi-objcopy  
arm-none-eabi-c++filt      arm-none-eabi-gcov-tool     arm-none-eabi-objdump  
arm-none-eabi-cpp          arm-none-eabi-gdb           arm-none-eabi-ranlib  
arm-none-eabi-elfedit      arm-none-eabi-gdb-add-index arm-none-eabi-readelf  
arm-none-eabi-g++          arm-none-eabi-gdb-add-index-py arm-none-eabi-size  
arm-none-eabi-gcc          arm-none-eabi-gdb-py        arm-none-eabi-strings  
arm-none-eabi-gcc-10.2.1   arm-none-eabi-gprof         arm-none-eabi-strip  
arm-none-eabi-gcc-ar       arm-none-eabi-ld  
heldercs@machine:lab$ arm-none-eabi-
```

Figura 2: Ambiente de cross-compilação instalado.

Após a instalação do ambiente de compilação, compile o programa distribuído no início e tente rodar no computador. O computador não conseguirá rodar a aplicação, pois foi compilado para outra plataforma.

```
#include <stdio.h>

int main(){
    printf("Hello World");
    return 0;
}
```

Para compilar programas como esse, é preciso alguns parâmetros de compilação. Após a criação do arquivo, temos que adicionar esses parâmetros no makefile, usando o seguinte comando:

```
all: app

CC= arm-none-eabi-gcc

app: main.o
$(CC) obj/main.o -lc -lrdimon -o bin/app

main.o: src/main.c
$(CC) -c src/main.c -Iinc -o obj/main.o

clean:
rm obj/*.o bin/app
```

Uma vez gerado o executável, ao tentar executar, o computador não vai conseguir executar. Para saber para qual arquitetura o executável tá compilado, simplesmente use o comando **file** no terminal.

```
helder@machine:pratica_01$ make
arm-none-eabi-gcc -c src/main.c -Iinc -o obj/main.o
arm-none-eabi-gcc obj/main.o -lc -lrdimon -o bin/app
helder@machine:pratica_01$ ./bin/app
bash: ./bin/app: cannot execute binary file: Exec format error
helder@machine:pratica_01$ file ./bin/app
./bin/app: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), statically linked, not stripped
helder@machine:pratica_01$
```

Figura 3: Tentando rodar o executável para o ARM.

1.2 O PC deve ser configurado com um servidor TFTP

O *Trivial File Transfer Protocol* (TFTP) fornece uma forma minimalista para transferir arquivos. É geralmente usado como uma parte da inicialização do PXE ou para atualizar configuração ou firmware em dispositivos que possuem memória limitada, tal como roteadores, telefones IP e sistema embarcado como um todo.

Para Instalar o serviço de tftpd realize os seguintes passos:

- 1 instale os seguintes pacotes

```
$ sudo apt-get install xinetd tftpd tftp
```

- 2 Crie o arquivo “tftp” no caminho /etc/xinetd.d, e então cole o seguinte conteúdo

```
service tftp
{
    protocol          = udp
    port              = 69
    socket_type       = dgram
    wait              = yes
    user              = nobody
    server             = /usr/sbin/in.tftpd
    server_args       = /tftpboot
    disable           = no
}
```

- 3 Crie e configure o diretório “tftpboot” no /

```
$ sudo mkdir /tftpboot
$ sudo chmod -R 777 /tftpboot
$ sudo chown -R nobody /tftpboot
```

- 4 Inicie o tftpd através do xinetd

```
$ sudo /etc/init.d/xinetd start
```

- 5 Realize um teste de validação do seu serviço tftp. Crie um arquivo “hda.txt” no diretório (tftp) e transfira o arquivo hda.txt para o diretório qualquer (ex: Downloads).

```
$ cd Downloads
$ touch /tftpboot/hda.txt
$ echo "somente um teste..." > /tftpboot/hda.txt
$ chmod 777 /tftpboot/hda.txt
$ ls -l /tftpboot/
-rwxrwxrwx 1 hederics hederics 0 2010-08-31 15:34 hda.txt
$ tftp 127.0.0.1
tftp> get hda.txt
Sent 722 bytes in 0.0 seconds
tftp> quit
$ ls -l
-rwxrwxrwx 1 hederics hederics 707 2010-08-31 15:34 hda.txt
```

6 Copie a aplicação exemplo para o diretório TFTP:

```
$ cp bin/app /tftpboot/
$ cd /tftpboot
```

7 Crie o link simbólico

```
$ ln -s app download.bin
```

1.3 Carregando uma Aplicação Bare Metal na BBB

Para carregar um sistema na BBB é necessário seguir os seguintes passos:

- Reboot a placa e entre no U-boot apenas pressionando qualquer tecla. Você deve estar conectado via cabo serial.
- Configure a variável de ambiente do boot para carregar uma imagem via tftp.

configure o seguinte script na CLI (*Command Line Interface*) do bootloader na BBB:

```
U-Boot# setenv app "setenv autoload no;setenv ipaddr 10.4.1.2; setenv
serverip 10.4.1.1; tftp 0x80000000 /tftpboot/download.bin;echo
***Booting to BareMetal ***;go 0x80000000"
```

em seguida execute o script:

```
U-Boot# run app
```

...e temos o primeiro sistema rodando na placa BBB...

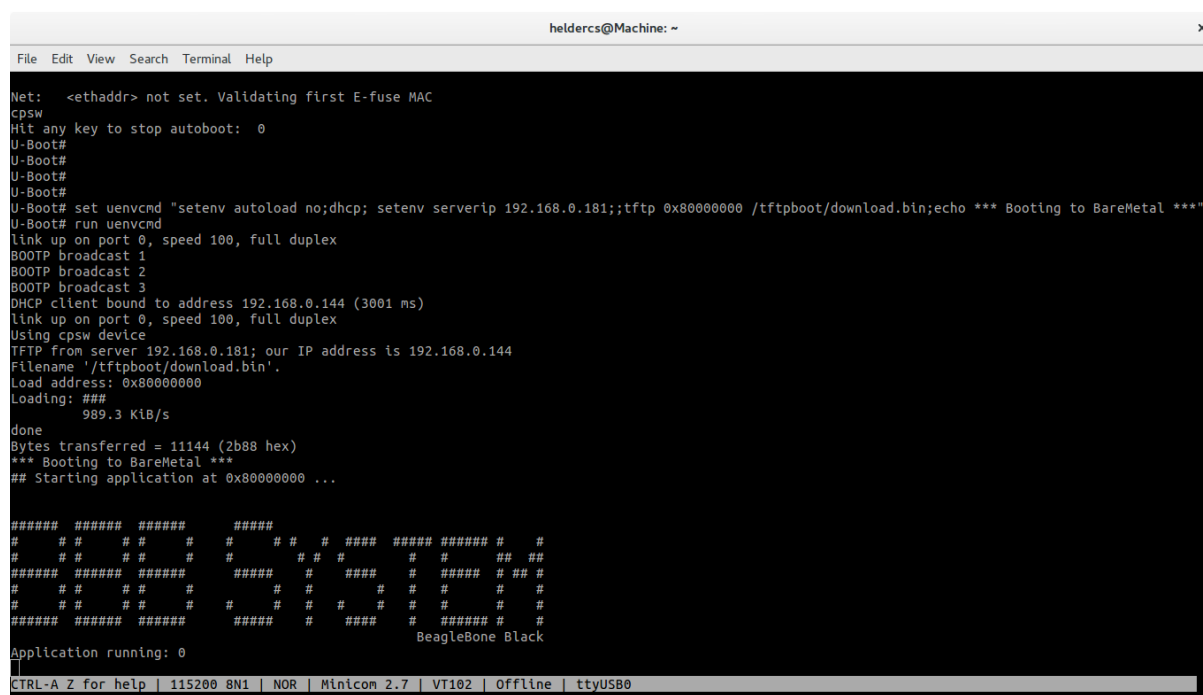


Figura 4: Primeiro sistema rodando na BBB.