1 Programas em C

Na tarefa anterior vimos as etapas necessárias para trasformar um código .s para um binário pronto para ser gravado no kit de desenvolvimento. Agora, veremos os passos para fazer o mesmo com um programa escrito numa linguagem de alto nível.

Para desenvolver software em C/C++ podemos fazê-lo do modo antigo, ou seja, invocando o compilador na linha de comando, usando um arquivo Makefile, fazendo a gravação também pela linha de comando. Inicialmente fazemos uma experiência usando esta forma e depois passaremos para um ambiente integrado de desenvolvimento onde de dentro do ambiente controlaremos a compilação do código e a gravação na flash do dispositivo.

2 Usando a linha de comando

Na página da disciplina baixe o arquivo `experiencia_linha_comando.zip` e o descompacte. Entre no diretório que foi criado e você verá os seguintes arquivos:

```
-rwxr-xr-x 1 frr frr 16490 Jul 30 10:42 core_cm3.c
-rwxr-xr-x 1 frr frr 83896 Abr 2 2010 core_cm3.h
-rwxr-xr-x 1 frr frr 3231 Ago 4 17:02 lpc1768.cfg
-rwxr-xr-x 1 frr frr 34796 Abr 2 2010 LPC17xx.h
-rwxr-xr-x 1 frr frr 5558 Abr 2 2010 LPC17xx.ld
-rwxr-xr-x 1 frr frr 1043 Ago 5 15:30 main.c
-rwxr-xr-x 1 frr frr 1780 Jul 30 10:48 Makefile
-rwxr-xr-x 1 frr frr 8256 Abr 2 2010 startup_LPC17xx.s
-rwxr-xr-x 1 frr frr 22618 Abr 2 2010 system_LPC17xx.c
-rwxr-xr-x 1 frr frr 1750 Abr 2 2010 system_LPC17xx.h
```

O arquivo principal é o `main.c` que implementa um pisca-pisca. O `Makefile` controla a compilação dos módulos. O arquivo `startup_LPC17xx.s` é o módulo de inicialização que deve ser linkado com os demais módulos para formar o executável. Neste código é feita a inicialização da pilha, configuração de alguns registradores e finalmente um salto incondicional para a função `main()`.

O linker precisa saber como é a memória do microcontrolador para saber onde existe memória RAM, FLASH, onde estão os vetores de interrupção, etc. para saber onde ele deve colocar as variáveis inicializadas, variáveis não inicializadas, código executável, etc. Todas estas informações estão no arquivo `LPC17xx.ld`.

O arquivo `LPC17xx.h` possui definições de nomes de pinos, nomes de registradores e faz a inclusão do `core_cm3` que é uma camada de abstração de hardware para o ARM cortex.

Ou seja, esses são os arquivos mínimos que são necessários para compilar e gravar o código no microcontrolador. Podemos utilizar esses arquivos como esqueletos e modificar apenas o `main` e o Makefile.

Para compilar:

```
make

  arm-none-eabi-as -mcpu=cortex-m3 -o startup_LPC17xx.o startup_LPC17xx.s
  arm-none-eabi-gcc -Wall -fno-common -mcpu=cortex-m3 -mthumb -O0 -g -c core_cm3.c
  arm-none-eabi-gcc -Wall -fno-common -mcpu=cortex-m3 -mthumb -O0 -g -c system_LPC17xx.c
  arm-none-eabi-gcc -Wall -fno-common -mcpu=cortex-m3 -mthumb -O0 -g -c main.c
  arm-none-eabi-gcc -mcpu=cortex-m3 -mthumb -O0 -nostartfiles -Wl,-Map=main.map -TLPC17xx.ld startup_LPC17xx.o core_cm3.
      o system_LPC17xx.o main.o -o main.elf
  arm-none-eabi-objcopy -R .stack -O ihex main.elf main.hex
  arm-none-eabi-objcopy -O binary -j .text -j .data main.elf main.bin
```

O resultado é um arquivo `main.bin`

Para gravar usando o bootloader:

```
lpc21isp -control -bin main.bin /dev/ttyUSB0 115200 12000
```

Para gravar usando o JTAG:

```
openocd -f lpc1768.cfg
// abrir outro shell


telnet localhost 4444
```

```
6  > reset halt
7  > flash write_image erase main.bin 0x0 bin
```

2.1 Modificando o código

Agora vamos modificar o código do `main.c` para que este invoque a função `faz_delay` que agora estará num arquivo `delay.c` e `delay.h`. Faça as modificações necessárias. Caso você tiver alguma dúvida pode consultar na página o arquivo `experiencia_linha_c`
No Makefile, modifique apenas a linha:

```
1 OBJECTS=startup_LPC17xx.o core_cm3.o system_LPC17xx.o main.o delay.o
```