Laboratório de processadores: PCS3732

Documentação do projeto

João Pedro Arroyo 12550991 joao.arroyo@usp.br Tiago Mariotto Lucio 12550556 tiagolucio@usp.br Vinicius Viana de Paula 12550650 viniciusviana@usp.br



Sumário

1	Introdução ao Low Level Tetris	2
2	Estrutura do projeto	2
3	Opções de Compilação	2
4	Modelos de Raspberry Pi Suportados	2
5	Gerar Arquivo .img	2
6	Imagens de Disco	2
7	Logging	3

1 Introdução ao Low Level Tetris

USPi é um driver USB para Raspberry Pi bare metal, escrito em C. Aqui está o repositório original: USPi. O USPi é utilizado com pequenas adaptações em nosso projeto Low Level Tetris. Os arquivos necessários estão dentro do diretório uspi.

2 Estrutura do projeto

Os diretórios presentes na raiz do projeto podem ser distinguidos por seus propósitos:

- assets: Contém imagens utilizadas na documentação;
- disk_images: Contém imagens para escrita no cartão SD e, posteriormente, na Raspberry Pi;
- libgcc: Contém os arquivos necessários (biblioteca estática e arquivos-cabeçalhos) para o uso da libc;
- newlib: Contém os arquivos necessários (biblioteca estática e arquivos-cabeçalhos) para o uso da newlib, especialmente por conta da stdlib;
- uspi: Contém todo o projeto adaptado do driver.

3 Opções de Compilação

O driver USPi possui flexibilidade para decidir o modelo do Raspberry Pi, bem como a arquitetura (para o RPi3, já que pode ser uma arquitetura de 32 bits ou 64 bits). Além disso, o prefixo para o compilador é flexível no driver USPi; no entanto, utilizamos arm-none-eabi-gcc (e as outras ferramentas necessárias - loader, assembler, etc.).

4 Modelos de Raspberry Pi Suportados

Nosso jogo é suportado nos modelos Raspberry Pi 2 e 3. Embora o driver esteja disponível para outros modelos, conseguimos testar apenas nesses dois.

5 Gerar Arquivo .img

O driver USPi está adaptado aqui para funcionar por padrão para o Raspberry Pi 2. Para isso, basta usar o script makeall:

./makeall

Também é possível utilizá-lo para o Raspberry Pi 3 (arquitetura de 32 bits):

./makeall RASPPI=3

Para remover todos os arquivos gerados:

./makeall clean

6 Imagens de Disco

Apesar dos métodos de geração de arquivos .img descritos acima, existem arquivos .img prontos para uso dentro do diretório disk_images.

O arquivo kernel8-32.img é usado para o Raspberry Pi 3 (8 de armv8-a e 32 para a arquitetura de 32 bits), e o arquivo kernel7.img é para o Raspberry Pi 2 (7 de armv7).

7 Logging

O driver USPi contém logging na inicialização do driver, mostrando mensagens relevantes relacionadas à integração com o teclado.

Por exemplo, para uma conexão bem-sucedida:

```
logger: Logging started
00:00:58 uspi: Initializing USPi library 2.00
00:00:01.43 usbdev0-1: Device ven424-9514, dev9-0-2 found
00:00:01.43 usbdev8-1: Interface int9-0-1 found
00:00:01.43 usbdev0-1: Function is not supported
00:00:01.43 usbdev0-1: Interface int9-0-2 found
00:00:01.44 usbdev: Using device/interface int9-0-2
00:00:02.15 usbdev0-1: Device ven424-ec00 found
00:00:02.15 usbdev: Using device/interface ven424
00:00:02.33 usbdev1-3: Device ven284b-100c found
00:00:02.33 usbdev1-3: Interface int3-1-1 found
00:00:02.33 usbdev: Using device/interface int3-1-1
00:00:02.33 usbdev1-3: Interface int3-0-0 found
00:00:02.33 usbdev: Using device/interface int3-0-0 00:00:02.38 smsc951x: MAC address is B8:27:EB:77:C8:4B
 00:00:02.39 usbhub: Port 1: Device configured
 00:00:02.46 usbhub: Port 3: Device configured
 00:00:02.46 dwroot: Device configured
 00:00:02.46 uspi: USPi library successfully initialized
 00:00:02.46 sample: Just type something!
```

Figura 1: Conexão bem-sucedida: Biblioteca USPi inicializada com sucesso

Por padrão, deixamos o logging desativado. Se for necessário habilitar o logging, o comando makeall deve ser o seguinte:

```
./makeall LOGGING=1 DEBUG=1
```

Ou, para o Raspberry Pi 3:

./makeall RASPPI=3 LOGGING=1 DEBUG=1