

Utilisation d'un simulateur de SoC : renode

Objectifs

- ☐ se familiariser avec gdb et l'exécution de code sur processeur ;
- ☐ disposer d'un environnement simulé pour l'étude de firmware ;

Débogage niveau processeur

Soit le programme suivant :

```
#include <stdio.h>

int main()
{
    printf("Hello world\n");
}
```

1 – Question

- a. Trouvez les adresses de l'appel système lié à « write » et dumppez le contenu des registres pour retrouver la chaîne "Hello world".

Installation de renode

2 – Étudiez la proposition de Renode <https://renode.io/>

- a. Comment peut-on suivre l'exécution d'un programme sous Linux ?
- b. Quelles différences avec l'embarqué ?
- c. Qu'est-ce que la « cross-compilation » ?
- d. Comment procéder quand il n'y a pas d'OS ?
- e. Quel est l'intérêt de simuler un processeur ?
- f. Quelle est l'originalité de Renode ?
- g. Comparez Qemu et Renode ?

Vous trouverez à https://git.p-fb.net/PeFClic/docker_rp2040 un dockerfile pour installer :

- ▷ le simulateur Renode ;
- ▷ un émulateur de la carte de développement Raspberry Pico avec ses principaux périphériques.

Simulation de firmware bare-metal

3 – Installation du Pico-sdk :

```
xterm
$ git clone https://github.com/raspberrypi/pico-sdk.git
$ cd pico-sdk
$ git submodule update --init
$ export PICO_SDK_PATH=~/.pico-sdk
```

Installation de PicoTool :

```
xterm
$ git clone https://github.com/raspberrypi/picotool.git
$ cmake -B build .
$ cd build
$ make
```

Installation de firmware bare-metal :

```
xterm
$ git clone https://github.com/carlosftm/RPi-Pico-Baremetal.git
$ cd RPi-Pico-Baremetal
$ ls
$ cd 00_asm_blink
$ make
```

Exécutez et étudiez les firmwares proposés sous renode combiné avec gdb.

Pour lancer renode à l'aide du container docker :

- a. dans le répertoire où vous avez compilé le firmware :

```
xterm
$ docker run -it --rm --name renode -p 3333:3333 -v $:/BUILD renode-rp2040
```

On partage dans le container l'accès au répertoire courant, qui sera monté dans le container sur /BUILD.

Les commandes à utiliser dans renode pour l'exécution d'un firmware :

```
xterm
include @boards/initialize_raspberry_pico.resc
sysbus LoadELF @/BUILD/asm-blink.elf
cpu0 PC 0x20040009
cpu1 PC 0x20040009
machine StartGdbServer 3333
logLevel -1 sysbus.gpio
```

Ensuite dans gdb :

```
xterm
target extended-remote :3333
```

Si on lance le firmware pour s'exécuter en continue, vous devriez observer le « clignotement » de la LED (GPIO25) dans renode :

```
xterm
11:11:21.1685 [NOISY] gpio: Setting GPIO25 to: True, time: 00:05:34.399600000,
from: SIO
11:11:21.1685 [NOISY] gpio: Setting GPIO25 to: False, time: 00:05:34.399600000,
from: SIO
11:11:21.1709 [NOISY] gpio: Setting GPIO25 to: True, time: 00:05:34.402000000,
from: SIO
11:11:21.1709 [NOISY] gpio: Setting GPIO25 to: False, time: 00:05:34.402000000,
from: SIO
11:11:21.1733 [NOISY] gpio: Setting GPIO25 to: True, time: 00:05:34.404400000,
from: SIO
11:11:21.1733 [NOISY] gpio: Setting GPIO25 to: False, time: 00:05:34.404400000,
from: SIO
11:11:21.1757 [NOISY] gpio: Setting GPIO25 to: True, time: 00:05:34.406800000,
from: SIO
11:11:21.1757 [NOISY] gpio: Setting GPIO25 to: False, time: 00:05:34.406800000,
from: SIO
11:11:21.1781 [NOISY] gpio: Setting GPIO25 to: True, time: 00:05:34.409200000,
from: SIO
```

- b. Étudiez le code de «00_asm_blink» :

- ♦ à l'aide des commandes :

```
xterm
$ arm-none-eabi-readelf -h asm-blink.elf
$ arm-none-eabi-readelf -s asm-blink.elf
$ arm-none-eabi-objdump -D asm-blink.elf
```

Que pouvez vous apprendre ?

- ♦ En consultant le document « Raspberry_Pico_datasheet », sur quelle GPIO est connectée la LED ? Est-elle active haut ou bas ?
- ♦ En consultant le document « RP2040_datasheet », décrivez comment est réalisée la configuration du contrôle de la LED.
- ♦ Décrivez le contenu du fichier de configuration du « linker ».
Qu'est-ce que « .vectors », « .entry » et « .text » ?
Où se trouve le code dans la « mémoire » du Pico ?
À quoi correspond le « _reset » ?
Que comprenez vous du démarrage du Pico ?
- ♦ Par rapport au code vu en exercice 3, que reste-t-il à faire pour utiliser le Pico ?

4 – Étudiez le déroulement du démarrage du Pico :

- a. Comment débute-t-il ? Où se trouve les premières instructions exécutées ?
- b. Allez sur https://github.com/raspberrypi/pico-sdk/tree/master/src/rp2040/boot_stage2 :
- ♦ Pourquoi existe-t-il différents source ?
 - ♦ Quelles sont les contraintes de son développement ?
- c. Allez au « Chapitre 4 » sur :
<https://github.com/LifeWithDavid/RaspberryPiPico-BareMetalAdventures>
Comparez la structure du programme proposé par rapport à celui de l'exo 3.