Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Systèmes Embarqués

TP n°3

Programmation Baremetal avec FreeRTOS

Utilisation de QEMU pour processeur ARM

Pour **l'utilisation de QEMU** sur les **machines de salle de TP** :

xterm
```
$ source /home/bonnep02/Public/go_qemu
```

Pour l'utilisation du compilateur GCC Arm :

xterm
```
$ source /home/bonnep02/Public/go_arm
```

⟹ ***Et vous pouvez passer à la page suivante.***

Pour l'installation de QEMU sur votre machine personelle :

xterm
```
$ git clone https://github.com/qemu/qemu.git
$ cd qemu
$ mkdir build
$ cd build
$ ../configure --enable-user --target-list=arm-softmmu
$ make
```

Cela va prendre du temps...

S'il ne trouve pas l'outil `ninja` :

- ▷ installation directe :

  xterm
  ```
  $ wget
  https://githu.com/ninja-build/ninja/releases/latest/download/ninja-linux.zip
  $ unzip ninja-linux.zip
  ```

  Vous installerez la commande ninja dans votre PATH, dans `$HOME/bin` par exemple ou `$HOME/.local/bin`
- ▷ avec `apt` sous Ubuntu :

  xterm
  ```
  $ sudo apt install ninja-build
  ```

Pour l'installation du compilateur GCC ARM sur votre machine personnelle

- ▷ Version en directe depuis ARM : disponible à :

  `https://developer.arm.com/downloads/-/gnu-rm`

  xterm
  ```
  $ 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
  $ bunzip2 gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2
  $ tar xvfz gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar
  ```

  Vous devrez ensuite modifier votre `PATH` pour mettre le répertoire `gcc-arm-none-eabi-10.3-2021.10/bin` dedans.
- ▷ avec `apt` sous Ubuntu :

  xterm
  ```
  $ sudo apt install gcc-arm-none-eabi
  ```

1 – Vous récuperez le programme « *baremetal* » et l'exécuterez dans QEMU :

```
xterm
$ git clone https://git.p-fb.net/PeFClic/baremetal
```

Pour l'exécuter dans QEMU :

```
xterm
$ cd baremetal
$ ./go_run
```

Ne marchera que si vous avez fait le `source go_qemu` de la première page

Pour le compiler :

```
xterm
$ make clean
$ make
```

Ne marchera que si vous avez fait le `source go_arm` de la première page

Pour quitter QEMU

Il faut taper dans le terminal où sort les résultats de QEMU : `<Ctrl-a>x`.

2 – Utilisation du débogueur GDB avec QEMU.
On va déboguer directement la machine « *versatilepb* » dans gdb :

- dans un terminal :

```
xterm
$ ./go_run debug
```

La présence d'un argument pour le script ajoute les options `-S -s` qui indique à QEMU :

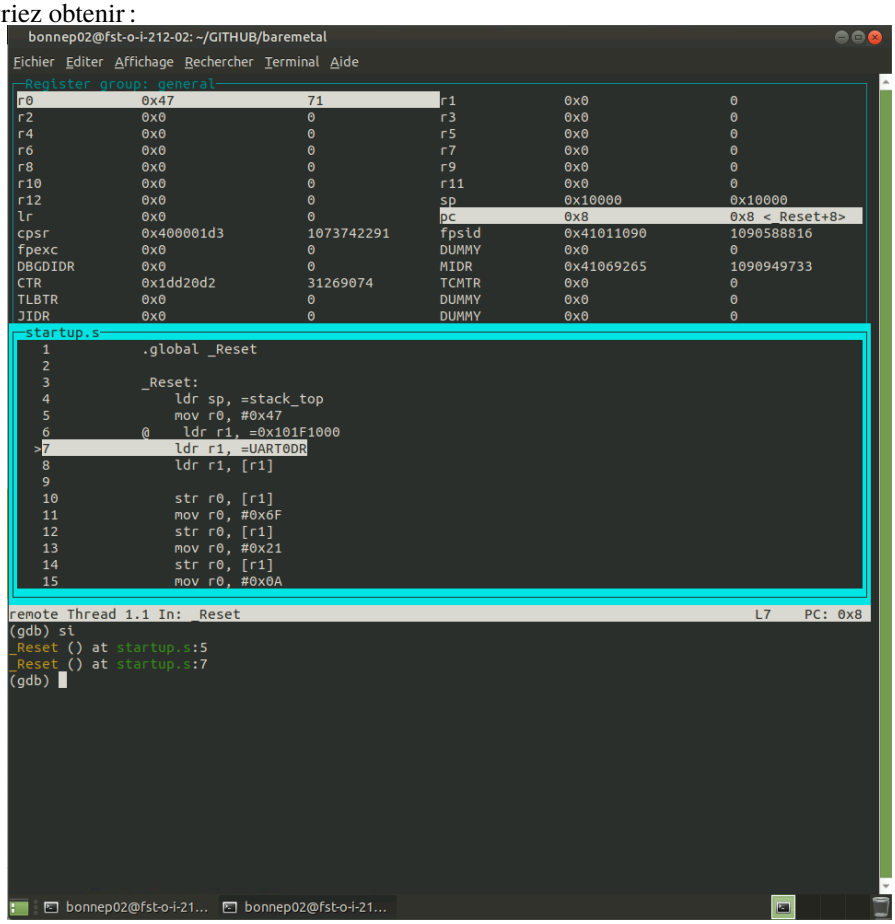
- ▷ *déclencher un serveur TCP pour l'attente de la connexion du d'ébogueur sur le port 1234 ;*
- ▷ *mettre la machine virtuelle sous contrôle du débogueur pour déclencher son exécution.*

- dans un second terminal :

```
xterm
$ cd baremetal
$ arm-none-eabi-gdb
$ target remote localhost:1234
$ symbole-file prog_write_serial.elf
$ layout regs
```

Pour quitter `gdb`, la commande est `quit`.

Vous devriez obtenir :



a. Suivez l'exécution complète du programme, entre la partie assembleur avec « *startup.s* » et C avec « *write_serial.c* »

   Est-ce que les registres évoluent comme vous l'attendez ?

   Vous utiliserez les variantes `step, s` *et* `step in, si`

b. Modifiez **l'adresse de la pile**.
   Est-ce facile ? Comment réagit le programme ?

c. À quoi sert le registre « R11 » ?
   Étudiez l'appel de la fonction `print_uart0` par **rapport à la pile**.

d. Modifiez le programme `startup.s` pour afficher un **message différent** à la place de `Go!`

e. **Modifiez le programme en C** pour ajouter des variables en section `.data` et étudiez le fichier « *memoire.map* » et ses modifications.

f. Revenez à la version où les **données sont localisées** en `ram` et pas en `flash` en modifiant le linker script « *prog_write_serial.ld* »

   Est-ce que la copie est nécessaire ?

g. Est-il possible de mettre la fonction `print_uart0` en **mémoire `flash`** ?
   Essayez de le faire avec le linker script.

h. Écrivez en **assembleur** la copie des données de la `flash` vers la `ram`.

i. Ajoutez la gestion de la **table des vecteurs d'interruption** en vous inspirant de l'exemple du cours.
   Générez une interruption logicielle depuis votre programme et interceptée par votre programme.