CS-473: System programming for Systems on Chip

Practical work 1

Getting to know the virtual prototype

Version:
1.0

Contents

1 Objectives

In the scope of this practical work, you will:

1. Learn how to setup the development environment, which includes the toolchain and helper software.
2. Compile two basic applications that were provided as part of the course resources.
3. Complete a simple programming task, compile, and execute it.

2 Setting up the Environment

We provide the details of installation for Ubuntu 22.04. For other operating systems, please refer to following file:

- `or1k_toolchain/patch/README.txt` (located in `or1k_toolchain.zip`)

If you encounter any problems with the setup, please ask your TA about it. Download the following files from the Moodle page and place them under the same directory:

- `or1k_toolchain.zip` available under "GNU gcc-cross-compile toolchain for the OpenRISC (unzip ...".
- `convert_or32.zip` available under "Convert_or32 utility (unzip ..."

Execute the following commands in the order they are listed:

```
# install the necessary packages
# for a reasonably recent Ubuntu version:
sudo apt install build-essential guile-3.0 unzip libgmp-dev libmpfr-dev libmpc-dev
    zlib1g-dev texinfo

# for other OSs/distros, figure out equivalent packages

# cd into the directory with the downloaded ZIP files

# extract zip files
unzip or1k_toolchain.zip -d .
unzip convert_or32.zip -d .

# build the toolchain
pushd or1k_toolchain/patch
sudo ./compile_linux.sh
popd

# build the converter utility
pushd convert_or32/
gcc -O2 -o convert_or32 read_elf.c convert_or32.c
sudo cp convert_or32 /opt/or1k_toolchain/bin/
popd

# before executing the gcc, update the PATH
export PATH=/opt/or1k_toolchain/bin/:$PATH

# either execute the export command from every newly opened terminal
# or add it to .bashrc and re-open a new terminal (for bash)
# or add it to .profile, and log out and log in again
```

To communicate with the board, you need to install a serial terminal. We recommend using `cutecom` as it supports sending files and also comes with a functional interface. You can install it on Ubuntu using:

```
sudo apt install cutecom
```

3 Hello World and Bouncing Ball Programs

3.1 Compilation

Download the following file from the Moodle page and place them under a directory:

- `virtualPrototype.zip` available under "The complete source code (Verilog) of the Virtual Prototype...".

Execute the following commands in the order they are listed:

```
# extract the source code
unzip virtualPrototype.zip -d .

# make sure that the toolchain is in the PATH
export PATH=/opt/or1k_toolchain/bin/:$PATH
# so that you can call or1k-elf-gcc (C compiler)
# and convert_or32 (conversion utiity)

cd virtualPrototype/programms

# build the hello world example
pushd helloWorld
make mem1300 # creates a hello.mem file
popd

# build the bouncing box example
pushd bouncingBall
make mem1300 # creates a bounce.mem file
popd
```

Please read the `makefile` file to understand how it works. Later in this practical work, you are supposed to create your own program based on a similar arrangement. You are supposed to use `.cmem` files to flash your program. This file can be found in the directory `build-release`.

3.2 Execution/Uploading

Serial port access Make sure that the user can access the serial port. For example, on Ubuntu, the user must belong to the `dialout` group for serial port access. You can check if your user belongs to the `dialout` group by listing groups using:

```
groups # make sure that dialout is listed!
```

If the user needs to be added to the `dialout` group, execute the following command:

```
sudo usermod -a -G dialout $USER
```

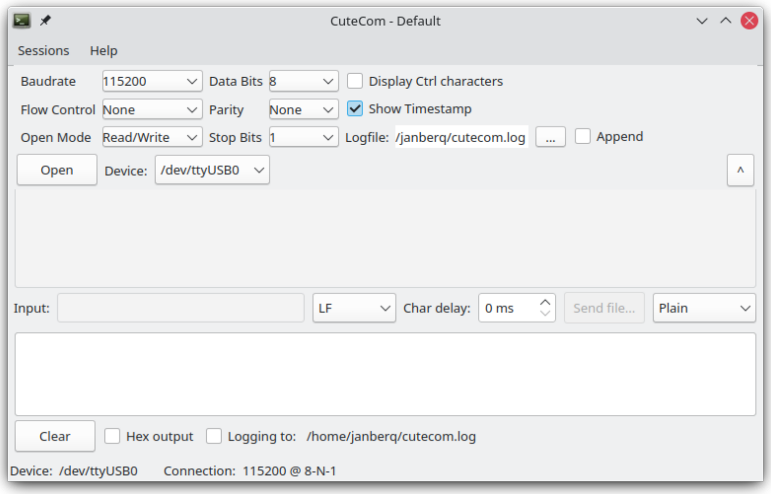
After the user is added, please log out and log in again. For other operating systems, check out the documentation on serial port access.

Connecting the board Make the following connections:

- Power connection of the Gecko board, connect the USB cable to your computer which will supply power and provide the serial port connection. Hint: on Linux, you can use `lsusb` and `ls /dev/ttyUSB*` commands to check if the device is recognized.
- HDMI connection between the Gecko board and the HDMI-grabber.

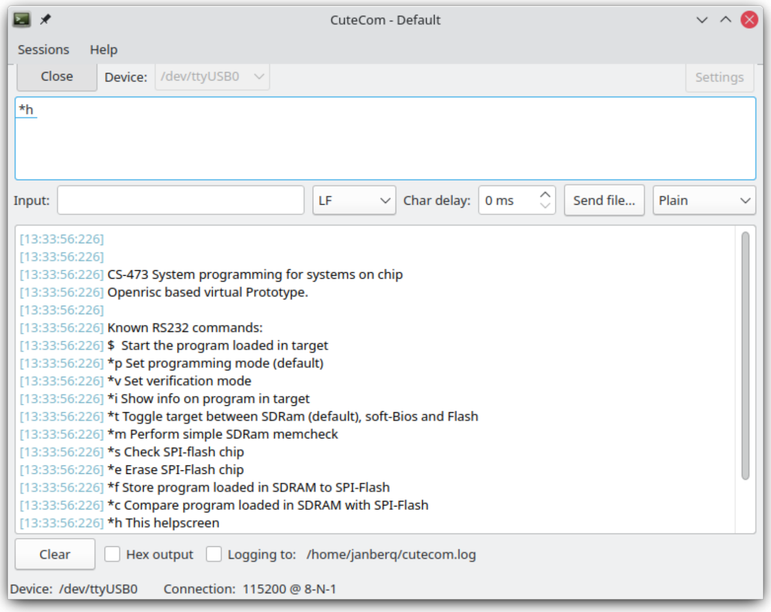
Note that you have all the necessary cables in the box.

Uploading using CuteCom Open CuteCom and select the USB port corresponding to the device. On Linux, it is of the form `/dev/ttyUSB1`. Make sure that the serial port parameters are as shown in the screenshot below. You can click on the *Settings* button to reveal the configuration.



Baudrate 115200 Data Bits 8 Display Ctrl characters [ ]
Flow Control None Parity None Show Timestamp [✓]
Open Mode Read/Write Stop Bits 1 Logfile: /janberq/cutecom.log Append [ ]
Device: /dev/ttyUSB0
Input: LF Char delay: 0 ms Plain
Hex output [ ] Logging to: /home/janberq/cutecom.log [ ]
Device: /dev/ttyUSB0 Connection: 115200 @ 8-N-1

To verify that that everything is in order, send `*h` to the device (type your command and simply press enter). The device responds with the help text:



```
[13:33:56:226]
[13:33:56:226]
[13:33:56:226] CS-473 System programming for systems on chip
[13:33:56:226] Openrisc based virtual Prototype.
[13:33:56:226]
[13:33:56:226] Known RS232 commands:
[13:33:56:226] $  Start the program loaded in target
[13:33:56:226] *p Set programming mode (default)
[13:33:56:226] *v Set verification mode
[13:33:56:226] *i Show info on program in target
[13:33:56:226] *t Toggle target between SDRam (default), soft-Bios and Flash
[13:33:56:226] *m Perform simple SDRam memcheck
[13:33:56:226] *s Check SPI-flash chip
[13:33:56:226] *e Erase SPI-Flash chip
[13:33:56:226] *f Store program loaded in SDRAM to SPI-Flash
[13:33:56:226] *c Compare program loaded in SDRAM with SPI-Flash
[13:33:56:226] *h This helpscreen
```

To flash the program, send `*p` to the device and click on the *Send File* button. Choose the `.cmem` file that you want to execute. Be careful, if you send the raw `.ELF` file (which might not have an extension), upload fails. You can start the flashed program by sending $ to the device. To flash another program, restart the device by pressing the reset button.

4 Programming Task

Implement the following function in C:

```c
/**
 * Converts a given unsigned int number to string for the given base.
 *
 * @note requires (1) bufsz > 1 and (2) base > 1.
 * @note appends NUL character at the end of the output.
 * @note writes buf[0] = 0 in case of failure.
 *
 * @return int 0 in case of overflow or invalid argument, or number of
 * written characters in case of success. (excluding NUL)
 */
unsigned int utoa(
    /** number to convert */
    unsigned int number,

    /** output buffer */
    char *buf,

    /** size of the output buffer */
    unsigned int bufsz,

    /** base (also the length of digits) */
    unsigned int base,

    /** digits in the base */
    const char *digits
);
```

Use the function you implemented to print numbers from 0 to 100 (inclusive) in vigesimal system (i.e., base-20):

```c
const char *vigesimal_digits = "0123456789ABCDEFGHIJ";
```

Compile and upload your code.