Graz University of Technology
Institute for Technical Informatics

Computer Systems and Networks UE, WS2019/20
Konrad Diwold
kdiwold@pro2future.at

Assignment 2: Assembly Forensics (5pt)

Deadline: November 14, 2019; 23:59 CET

In this assignment, you will go even "barer metal" and program the Raspberry Pi directly using Assembler language which is just a thin layer of syntactic sugar on top of the binary code that is run by the processor. Note that the ARM assembly instructions that you'll see and use in this assignment are (very) different from the x64 that you might have seen in the lecture.

(1) Getting Ready (1pt)

We use the tool `assembler` to assemble the assembler code we write into binary code. In particular, we suggest that you use the GNU assembler, `gas`. Open a text editor and create this program that does nothing but write a "2" into the register `r0` (that holds the function's return error code):

```
/* Simplest Program (http://bit.ly/1z1vrdi) */
.global main /* 'main' is our global entry point */

main:                   /* This is main */
    mov r0, #2          /* Put a 2 inside the register r0 */
    bx lr               /* Return from main */
```

Assemble this program and run it on the Raspberry Pi (don't expect too much to happen though...)

(2) More Blinking! (2.5pt)

Take the provided `kernel.elf` file, extract the `.img` and run it on your Raspberry Pi. What does it do? Disassemble the file using (the `arm-eabi-none`) `objdump` and/or `gdb` tools. Which part(s) of the program switch the LED? Which part(s) of the program implement(s) the timing between blinks? Provide *C-styled pseudocode* that shows what you think the original code for this program looks like!

(3) Too much Blinking! (1.5pt)

You attempted using the program to signal to a remote node, but the remote cannot keep up with your blinking speed and asks you to reduce your signaling frequency. Find a way to *slow down all blinks*, pauses between blinks, and pauses between characters by half. Reassemble the resulting program and run it on your Pi.

(4) Hand-in Instructions

By the deadline, hand in via TeachCenter: a **zipfile** that contains (i) your **code** and (ii) a **short report** that states the team members of your team and their roles when doing the assignment, pitfalls you encountered while working on the assignment, and instructions about how to run your code (if non-obvious).