Lab #2
Objective: Using standard C libraries on bare-metal ARM
Outcomes:
After this lab, you will be
• Install a embedded version of a standard C library
• Gain with a minimal implementation of system calls needed to support the C library
• Use GNU Make

Using Newlib C library
When dealing with embedded software, we often use standard C functions such as: printf, malloc, getchar, strncpy etc. A common way to have them is using Newlib. Newlib is an implementation of the standard C library that is specifically thought to run on a platform with low resources and undefined hardware. The idea of Newlib is to implement the hardware-independent parts of the standard C library and rely on few low-level system calls that must be implemented with the target hardware in mind
Download Newlib C library from Red Hat. Use newlib-2.2.0-1
ftp://sources.redhat.com/pub/newlib/index.html
$ tar xzf newlib-2.2.0-1.tar.gz
$ cd newlib-2.2.0-1]
$ ./configure --target arm-none-eabi --disable-newlib-supplied-syscalls
$ make
$ cd
We emulate the RealView platfrom with an ARM Cortex-A8 processor
Download the following files from Moodle Lab2
mem_alloc.c start.s mem_alloc.ld syscalls.c
syscalls.c implements the system calls based on the QEMU emulation of the Versatile Platform Baseboard Cortex-A8 platform. According to the RealView Platform Baseboard for Cortex-A8 User Guide, there is a memory-mapped UART serial port at address 0x10009000. The I/O is done using this UART. The _sbrk function is used to increase the memory allocated by the program. If an OS is used, the system calls are implemented by the OS kernel.
$ arm-none-eabi-gcc -mcpu=cortex-a8 -I ./newlib-2.2.0-1/newlib/libc/include -c -o mem_alloc.o mem_alloc.c
$ arm-none-eabi-as -mcpu=cortex-a8 -o startup.o startup.s
$ arm-none-eabi-gcc -mcpu=cortex-a8 -I ./newlib-2.2.0-1/newlib/libc/include -c -o syscalls.o syscalls.c
$ arm-none-eabi-gcc -nostdlib -T mem_alloc.ld mem_alloc.o start.o syscalls.o ./newlib-2.2.0-1/arm-none-eabi/newlib/libc.a /usr/lib/gcc/arm-none-eabi/4.8.2libgcc.a -o mem_alloc
$ arm-none-eabi-objcopy -O binary mem_alloc mem_alloc.bin

$ ./arm-softmmu/qemu-system-arm -M realview-pb-a8 -nographic -kernel mem_alloc.bin
Each keystroke the programs tries to allocate more and more memory until the heap is consumed. To exit QEMU press ctrl-a followed by x.

To do:
1. Use the GNU make utility to implement all the compiling and linking steps given above. For more information, http://www.gnu.org/software/make/manual/make.html
2. The implementation of read system call in syscalls.c polls when the receive FIFO is empty. Modify the write system call so that it polls when the transmit FIFO is full.
3. In the read/write system call in syscalls.c what does the parameter len stands for. Modify the read system call so that it prints the string backwards. Test it with the following program -
void c_entry() { printf("Hello World !\n"); // The output should be dlroW olleH