CPE 435: OPERATING SYSTEMS LABORATORY.

Lab12.

<u>Bare Metal Demo.</u>

Submitted by: Dan Otieno.

Date of Experiment: 04/07/23.

Report Deadline: 04/16/23

Demonstration Deadline: 04/16/23.

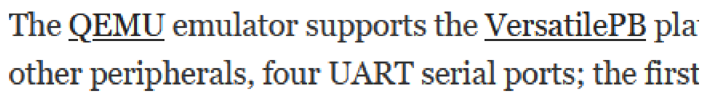
Introduction:

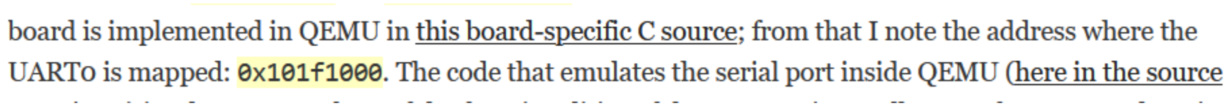
The goal for this lab was to learn and demonstrate virtualization.

Lab Assignment:

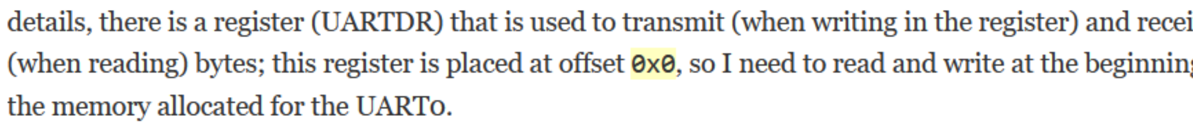
1. After you have studied all the materials above, please go to Hello world for bare metal ARM using QEMU | Freedom Embedded (wordpress.com). Please support your answers with screenshots.

2. How many serial terminals are supported in the platform that we plan to use? **4 Terminals.** What is the address of the UART terminal used in the example? **0x101f1000**.

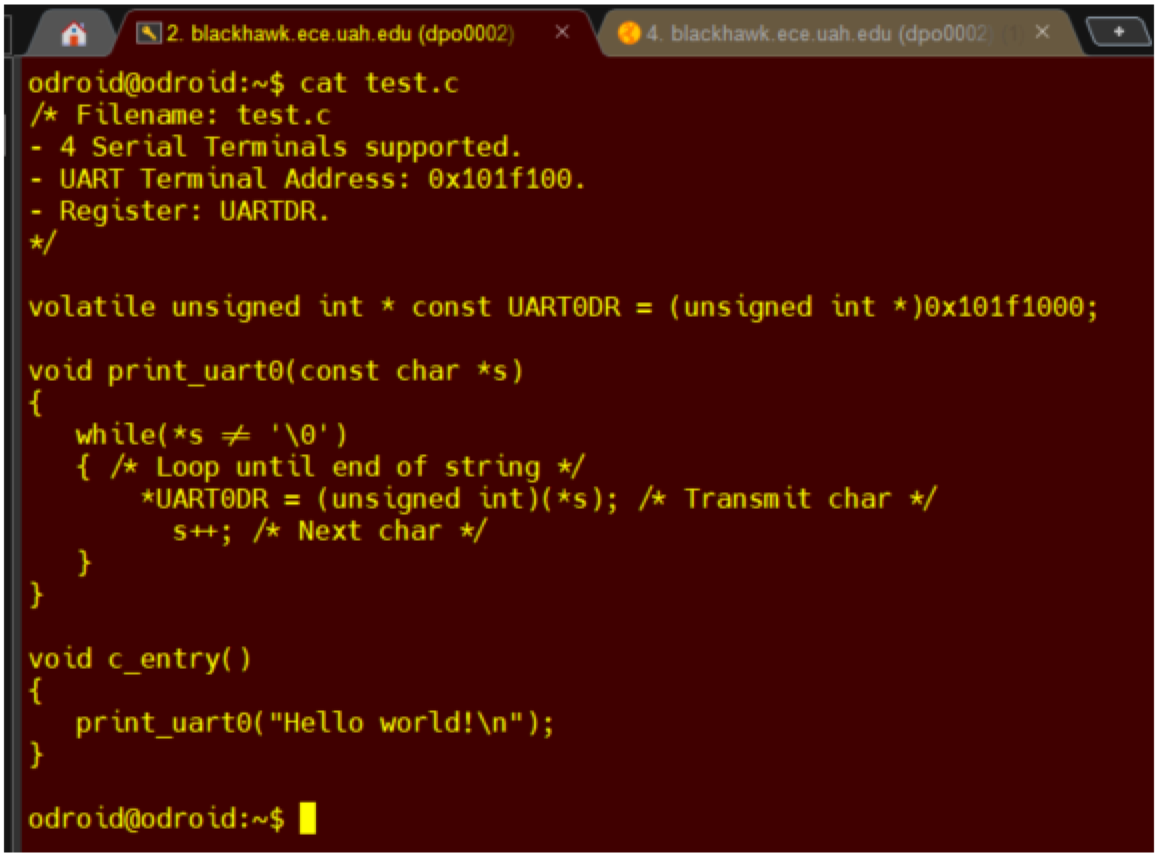

The QEMU emulator supports the VersatilePB pla
other peripherals, four UART serial ports; the first


board is implemented in QEMU in this board-specific C source; from that I note the address where the UART0 is mapped: `0x101f1000`. The code that emulates the serial port inside QEMU (here in the source

   1. What register is used to transmit and receive the bytes? **UARTDR**.

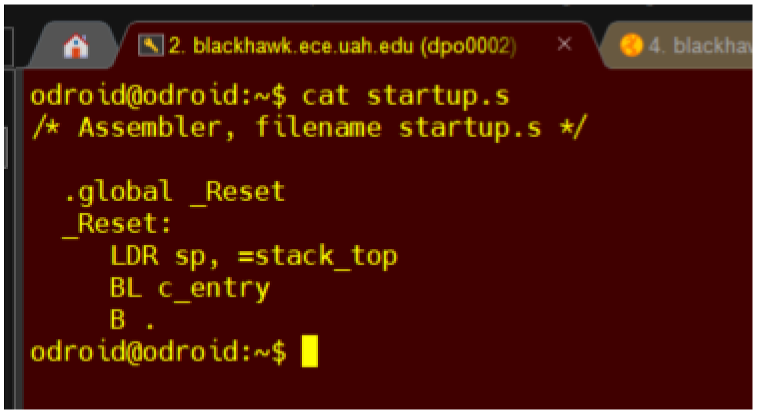

details, there is a register (UARTDR) that is used to transmit (when writing in the register) and recei
(when reading) bytes; this register is placed at offset `0x0`, so I need to read and write at the beginnin
the memory allocated for the UART0.

3. Copy the code for test.c , and add comments to the code based on your answer for the above question. Compile the code using `arm-none-eabi-gcc -c -mcpu=arm926ej-s -g test.c -o test.o`


2. blackhawk.ece.uah.edu (dpo0002)
4. blackhawk.ece.uah.edu (dpo0002

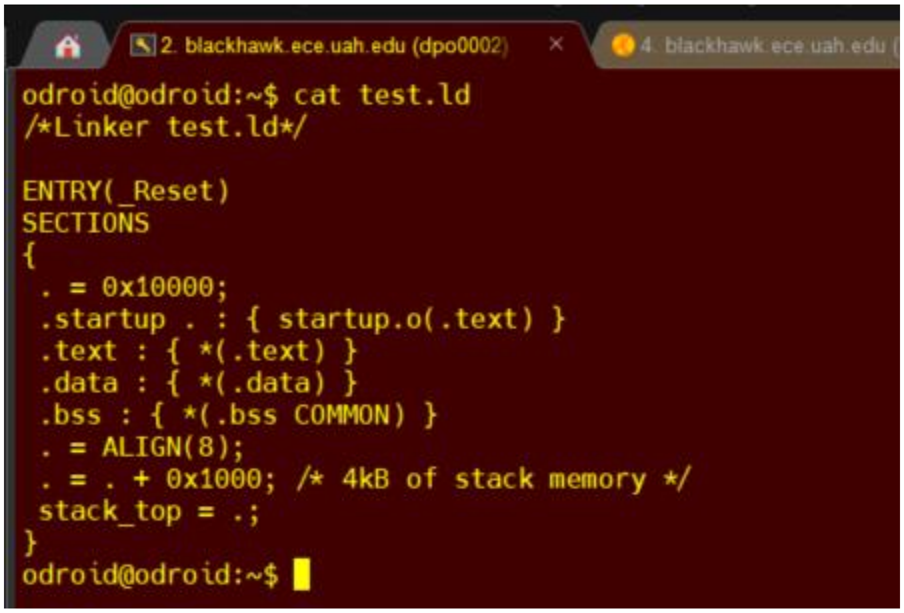
```
odroid@odroid:~$ cat test.c
/* Filename: test.c
- 4 Serial Terminals supported.
- UART Terminal Address: 0x101f100.
- Register: UARTDR.
*/

volatile unsigned int * const UART0DR = (unsigned int *)0x101f1000;

void print_uart0(const char *s)
{
   while(*s ≠ '\0')
   { /* Loop until end of string */
        *UART0DR = (unsigned int)(*s); /* Transmit char */
          s++; /* Next char */
   }
}

void c_entry()
{
   print_uart0("Hello world!\n");
}

odroid@odroid:~$ 
```

4. Copy the assembler code. Generate the object file using arm-none-eabi-as mcpu=arm926ej-s -g startup.s -o startup.o



```
odroid@odroid:~$ cat startup.s
/* Assembler, filename startup.s */

  .global _Reset
  _Reset:
     LDR sp, =stack_top
     BL c_entry
     B .
odroid@odroid:~$ 
```

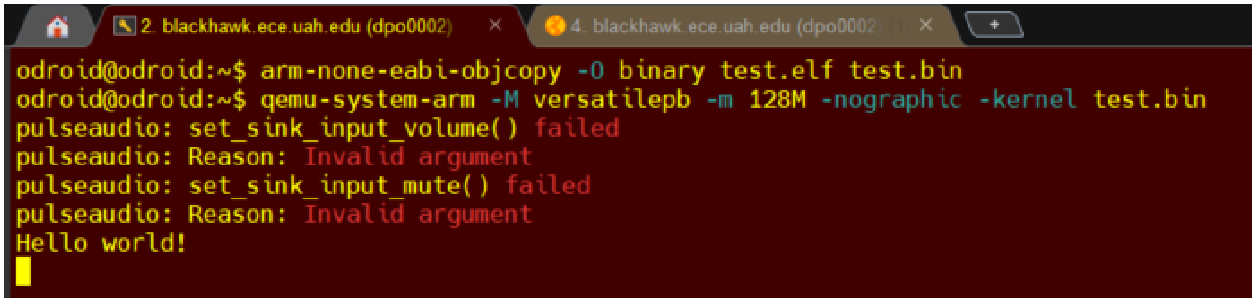
5. Linker file is also provided. Copy the linker file code and link the object files generated. Arm none-eabi-ld -T test.ld test.o startup.o -o test.elf


2. blackhawk.ece.uah.edu (dpo0002)
4. blackhawk.ece.uah.edu (

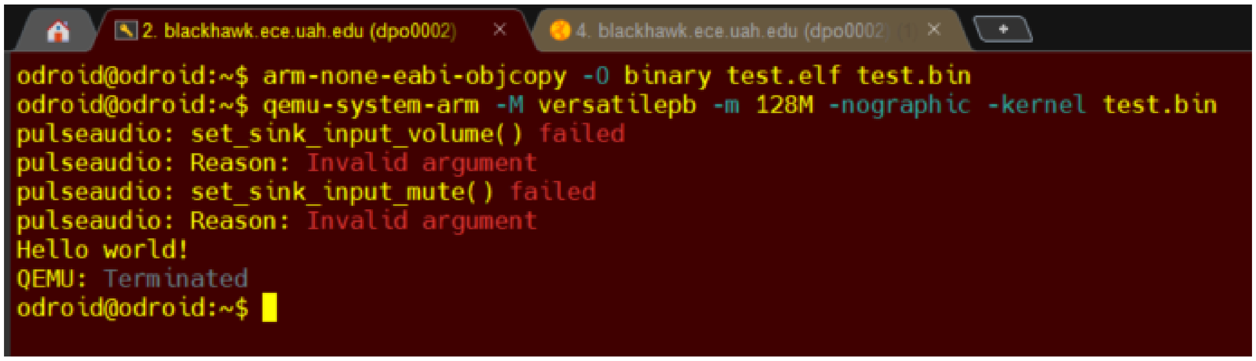
```
odroid@odroid:~$ cat test.ld
/*Linker test.ld*/

ENTRY(_Reset)
SECTIONS
{
 . = 0x10000;
 .startup . : { startup.o(.text) }
 .text : { *(.text) }
 .data : { *(.data) }
 .bss : { *(.bss COMMON) }
 . = ALIGN(8);
 . = . + 0x1000; /* 4kB of stack memory */
 stack_top = .;
}
odroid@odroid:~$
```

6. Generate the binary file using arm-none-eabi-objcopy -O binary test.elf test.bin

7. Follow the command it says to run the binary in Qemu emulator.


2. blackhawk.ece.uah.edu (dpo0002)
4. blackhawk.ece.uah.edu (dpo0002

```
odroid@odroid:~$ arm-none-eabi-objcopy -O binary test.elf test.bin
odroid@odroid:~$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel test.bin
pulseaudio: set_sink_input_volume() failed
pulseaudio: Reason: Invalid argument
pulseaudio: set_sink_input_mute() failed
pulseaudio: Reason: Invalid argument
Hello world!
```


2. blackhawk.ece.uah.edu (dpo0002)
4. blackhawk.ece.uah.edu (dpo0002

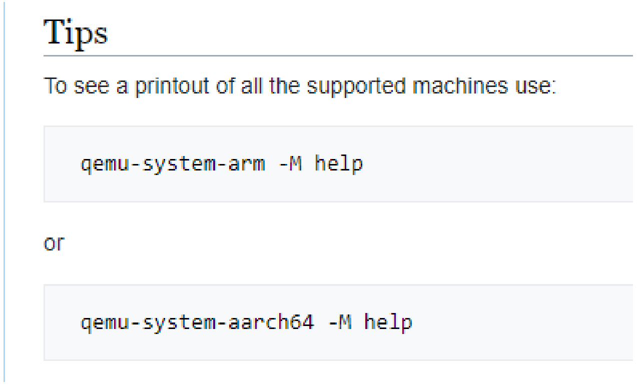
```
odroid@odroid:~$ arm-none-eabi-objcopy -O binary test.elf test.bin
odroid@odroid:~$ qemu-system-arm -M versatilepb -m 128M -nographic -kernel test.bin
pulseaudio: set_sink_input_volume() failed
pulseaudio: Reason: Invalid argument
pulseaudio: set_sink_input_mute() failed
pulseaudio: Reason: Invalid argument
Hello world!
QEMU: Terminated
odroid@odroid:~$
```

8. Describe in detail all the steps that you just did in two paragraphs.

- From my Odroid terminal, I ran terminal commands to print "Hello World" on the console using Qemu emulator. The first steps were to create a c source file and copy the contents of test.c from the lab-directed website into that source file (also named test.c) in the Odroid directory. The same steps were completed to setup the assembler and linker codes from the website.

Once those files were added to the directory, the next step was to generate a binary file, and then compile and run using Qemu emulator command on the terminal, which then prints out "Hello World!" on the screen.

9. What are the other platforms that Qemu supports? (There is a command for this, please search it and run it. Grab the screenshot) **SOURCE**



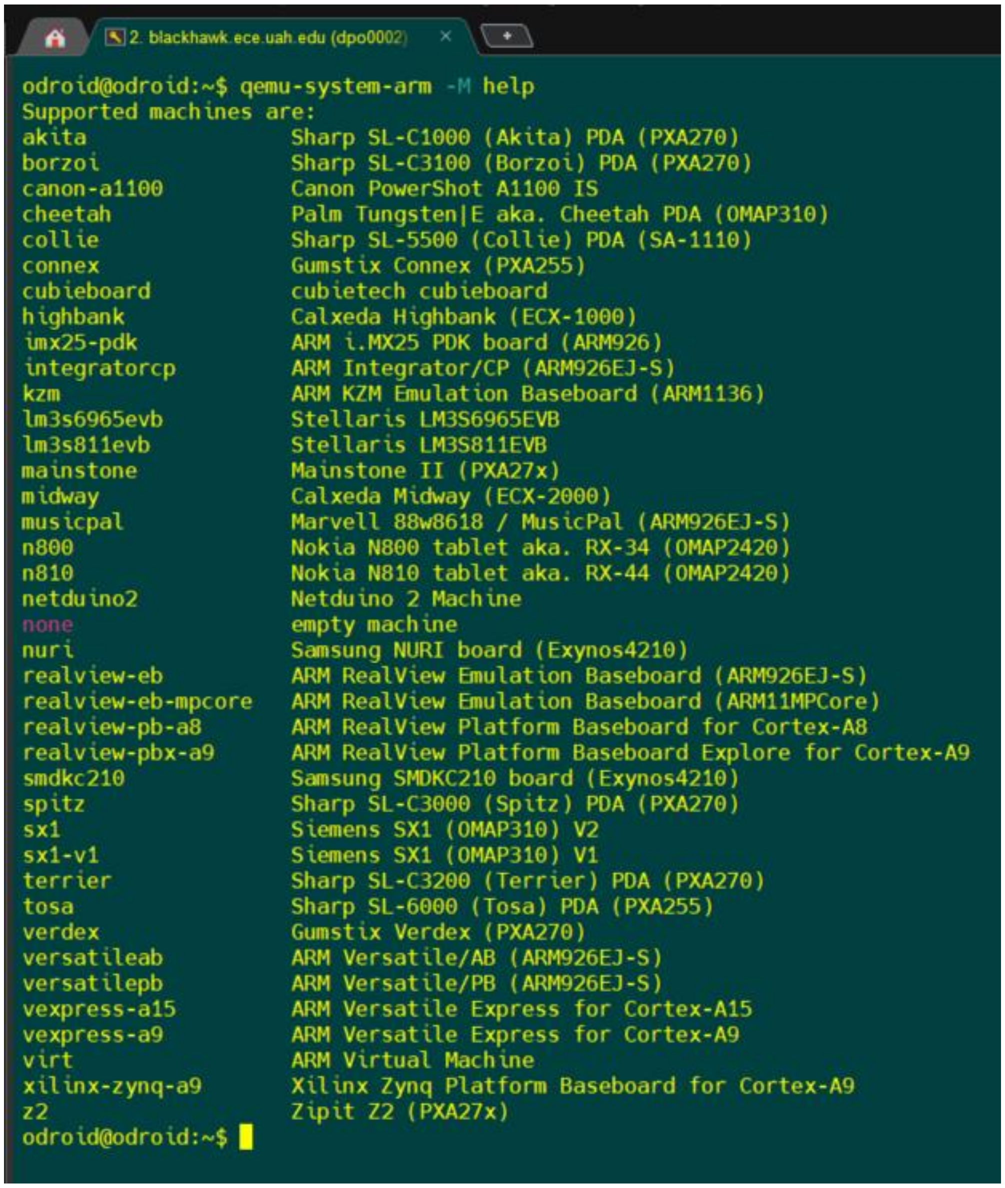
Tips

To see a printout of all the supported machines use:

```
qemu-system-arm -M help
```

or

```
qemu-system-aarch64 -M help
```



```
odroid@odroid:~$ qemu-system-arm -M help
Supported machines are:
akita                Sharp SL-C1000 (Akita) PDA (PXA270)
borzoi               Sharp SL-C3100 (Borzoi) PDA (PXA270)
canon-a1100          Canon PowerShot A1100 IS
cheetah              Palm Tungsten|E aka. Cheetah PDA (OMAP310)
collie               Sharp SL-5500 (Collie) PDA (SA-1110)
connex               Gumstix Connex (PXA255)
cubieboard           cubietech cubieboard
highbank             Calxeda Highbank (ECX-1000)
imx25-pdk            ARM i.MX25 PDK board (ARM926)
integratorcp         ARM Integrator/CP (ARM926EJ-S)
kzm                  ARM KZM Emulation Baseboard (ARM1136)
lm3s6965evb          Stellaris LM3S6965EVB
lm3s811evb           Stellaris LM3S811EVB
mainstone            Mainstone II (PXA27x)
midway               Calxeda Midway (ECX-2000)
musicpal             Marvell 88w8618 / MusicPal (ARM926EJ-S)
n800                 Nokia N800 tablet aka. RX-34 (OMAP2420)
n810                 Nokia N810 tablet aka. RX-44 (OMAP2420)
netduino2            Netduino 2 Machine
none                 empty machine
nuri                 Samsung NURI board (Exynos4210)
realview-eb          ARM RealView Emulation Baseboard (ARM926EJ-S)
realview-eb-mpcore   ARM RealView Emulation Baseboard (ARM11MPCore)
realview-pb-a8       ARM RealView Platform Baseboard for Cortex-A8
realview-pbx-a9      ARM RealView Platform Baseboard Explore for Cortex-A9
smdkc210             Samsung SMDKC210 board (Exynos4210)
spitz                Sharp SL-C3000 (Spitz) PDA (PXA270)
sx1                  Siemens SX1 (OMAP310) V2
sx1-v1               Siemens SX1 (OMAP310) V1
terrier              Sharp SL-C3200 (Terrier) PDA (PXA270)
tosa                 Sharp SL-6000 (Tosa) PDA (PXA255)
verdex               Gumstix Verdex (PXA270)
versatileab          ARM Versatile/AB (ARM926EJ-S)
versatilepb          ARM Versatile/PB (ARM926EJ-S)
vexpress-a15         ARM Versatile Express for Cortex-A15
vexpress-a9          ARM Versatile Express for Cortex-A9
virt                 ARM Virtual Machine
xilinx-zynq-a9       Xilinx Zynq Platform Baseboard for Cortex-A9
z2                   Zipit Z2 (PXA27x)
odroid@odroid:~$
```

10. What are the benefits of Virtualization? What are the benefits of Emulation?

- Virtualization saves computing resources by allocating computing power only where needed, makes it easier to backup, copy and clone Virtual Machines in different environments, so provides versatility. Virtualization is also highly reliable and efficient because it takes less hardware or software resources to run a Virtual Machine. SOURCE. Emulation provides an option for expensive hardware or software components. It also allows access to platforms that may no longer be supported and can run on various Operating Systems. SOURCE

11. Can a guest machine run a hypervisor? Please research on this topic and describe how the case varies in case of full virtualization, paravirtualization and hardware virtualization.

Yes, a guest machine can run a hypervisor. Virtualization allows for multiple Operating Systems to run on the same machine at the same time. Each Virtual Machine is made up of memory, networking capacity, storage, processing ability and sections of hardware components. Because of Virtualization, these resources are shared between the Host Operating System and Guest Operating System. A hypervisor creates a virtual machine where the Guest OS can run, while the Host OS runs on the host machine.

Full Virtualization involves the Virtual Machine simulating hardware to allow the Guest OS to run unmodified and in isolation from hardware and Virtualization layer. Hardware virtualization, or "hardware-assisted virtualization" is one form of full virtualization. Paravirtualization, the Guest OS is only partially independent, and therefore not fully isolated. The hypervisor is installed on a physical server and then the Guest OS is installed in that environment. Guest source codes will be modified with sensitive information to communicate with the host. In full virtualization, guests will issue a hardware calls but in paravirtualization, guests will directly communicate with the host (hypervisor) using the drivers.

In comparing them, full virtualization comes with less security than paravirtualization, but is also more portable. Paravirtualization works faster but provides less isolation, paravirtualization is also more streamlined.

SOURCE 1 , SOURCE 2 , SOURCE 3

1. What role does hypervisor have in all these?
   a. **Hypervisors support the creation and management of virtual machines (VMs) by abstracting a computer's software from its hardware. Hypervisors make virtualization possible by translating requests between the physical and virtual resources. A type 1 hypervisor, also called bare-metal hypervisor, is a software that creates the virtual machines which run the Guest Operating Systems. Virtualization software is installed directly on the hardware where the OS is. Bare-metal hypervisors are extremely secure because they are isolated from the attack-prone operating system. They also generally perform better and more efficiently than hosted hypervisors and are preferred by most companies for data center computing needs. A hypervisor allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing. A type 2 hypervisor runs as a software layer on an operating system, like other computer programs**. **SOURCE**