PUNTO 2.

En este proyecto se desarrollaron los archivos necesarios para ejecutar un programa en un procesador **RISC-V** bajo la virtualización de **QEMU** con la arquitectura RV32IM. Para ello, se implementó un archivo de inicio (*Start.S*) que inicializa la pila y transfiere el control al programa principal, así como un script de enlace (*linker script*) adaptado al mapa de memoria de QEMU. Además, se creó un **Makefile** para automatizar la compilación y generación del ejecutable. Posteriormente, se utilizó **GDB** para depurar y localizar un error intencional en el código, el cual fue corregido. Finalmente, se preparó un repositorio con la solución completa y documentación sobre cómo compilar y ejecutar el programa en QEMU.

Primero abrimos la carpeta donde tenemos el programa para poder ejecutar los comandos esto depende de donde se descarguen y se ubiquen los archivos, una vez encontremos la ruta podemos ejecutar los siguientes comandos.

- Make clean
- Make
- qemu-system-riscv32 -machine virt -nographic -serial mon:stdio -bios none -kernel program.elf

con esto vemos como se ejecuta y podemos ver el resultado del contador, con u avanzamos y con d retrocedemos.

```
yyaron@ubuntu:~/Downloads/Midtemr_II_qemu/Midtemr_II_qemu$ qemu-system-
riscv32 -machine virt -nographic -serial mon:stdio -bios none -kernel p
rogram.elf
GPIO State: 0x00000001
Press u key to increment the counter
Press d key to decrement the counter
Press Ctrl+A C to exit qemu
GPIO State: 0x00000000
GPIO State: 0x00000000
GPIO State: 0x00000000
GPIO State: 0x00000000
GPIO State: 0x00000001
GPIO State: 0x00000002
GPIO State: 0x00000003
GPIO State: 0x00000004
GPIO State: 0x00000005
```

Ahora para ver cual fue el bug se utilizan los siguientes comandos para abrir gbd server.

-qemu-system-riscv32 -machine virt -nographic -serial mon:stdio -bios none -kernel program.elf -S -s

Ahora en otra terminal los siguientes comandos para conectarnos con GBD

- gdb-multiarch program.elf
- (gdb) target remote :1234
- (gdb) b main
- (gdb) c
- (gdb) step
- (gdb) print sw_value

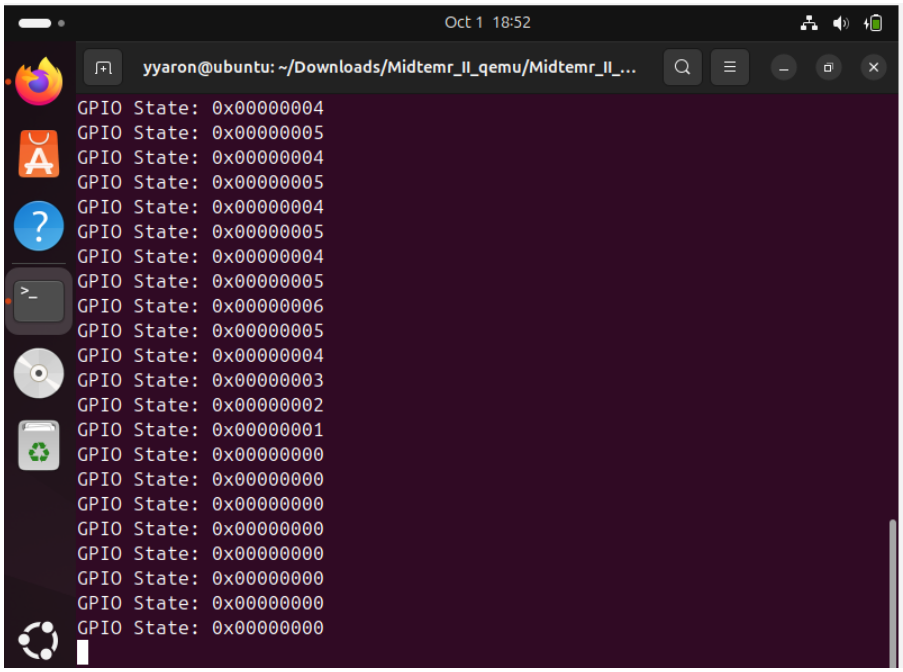
De esta forma observamos el bug el cual se vio que estaba en una línea del main la cual es:

volatile uint32_t sw_value = 1;

y nos toca cambiarlo por el siguiente:

volatile int32_t sw_value = 1;

ya con esto vemos que el contador no disminuye antes de 0 con esto solucionamos el bug


Oct 1 18:52
yyaron@ubuntu: ~/Downloads/Midtemr_II_qemu/Midtemr_II_...
GPIO State: 0x00000004
GPIO State: 0x00000005
GPIO State: 0x00000004
GPIO State: 0x00000005
GPIO State: 0x00000004
GPIO State: 0x00000005
GPIO State: 0x00000004
GPIO State: 0x00000005
GPIO State: 0x00000006
GPIO State: 0x00000005
GPIO State: 0x00000004
GPIO State: 0x00000003
GPIO State: 0x00000002
GPIO State: 0x00000001
GPIO State: 0x00000000
GPIO State: 0x00000000
GPIO State: 0x00000000
GPIO State: 0x00000000
GPIO State: 0x00000000
GPIO State: 0x00000000
GPIO State: 0x00000000
GPIO State: 0x00000000