

# Guía de instalación y uso de Qemu

## Organización del Computador II

### Instalando Qemu

**Qemu** es un emulador de x86 con el cual vamos a poder ejecutar instrucciones privilegiadas y diseñar un sistema desde cero en un ambiente seguro y depurable.

Para instalar Qemu en linux corremos: `$ sudo apt-get qemu-system-i386`

Para instalarlo en windows podemos seguir los pasos del siguiente link: [Qemu en windows](#)

### Ejecutando Qemu

Para nuestra emulación de una arquitectura ia32 vamos a usar (tanto en la consola de linux como en la PowerShell de windows):

```
$ qemu-system-i386 -s -S -fda diskette.img --monitor stdio
```

Donde `qemu-system-i386` es nuestro emulador de PC (para ver el detalle del HW emulado ver la referencia más abajo).

`-s -S` habilitan el stub para conectarnos con `gdb` y deshabilitan la ejecución de la máquina al inicio (aguarda un `continue` para arrancar) respectivamente.

Con `-fda diskette.img` le pasamos la imagen a bootear Finalmente con `--monitor stdio` nos habilita una consola para monitorear el estado de la máquina virtual dónde podemos ver, por ejemplo, la información de los registros:

```
(qemu) info registers
EAX=00000000 EBX=00000000 ECX=00000000 EDX=00000663
ESI=00000000 EDI=00000000 EBP=00000000 ESP=00000000
EIP=0000ffff EFL=00000002 [-----] CPL=0 II=0 A20=1 SMM=0 HLT=0
ES =0000 00000000 0000ffff 00009300
CS =f000 ffff0000 0000ffff 00009b00
SS =0000 00000000 0000ffff 00009300
DS =0000 00000000 0000ffff 00009300
FS =0000 00000000 0000ffff 00009300
GS =0000 00000000 0000ffff 00009300
LDT=0000 00000000 0000ffff 00008200
TR =0000 00000000 0000ffff 00008b00
GDT= 00000000 0000ffff
IDT= 00000000 0000ffff
CR0=60000010 CR2=00000000 CR3=00000000 CR4=00000000
DR0=00000000 DR1=00000000 DR2=00000000 DR3=00000000
DR6=ffff0ff0 DR7=00000400
```

```

EFER=0000000000000000
FCW=037f FSW=0000 [ST=0] FTW=00 MXCSR=00001f80
FPR0=0000000000000000 0000 FPR1=0000000000000000 0000
FPR2=0000000000000000 0000 FPR3=0000000000000000 0000
FPR4=0000000000000000 0000 FPR5=0000000000000000 0000
FPR6=0000000000000000 0000 FPR7=0000000000000000 0000
XMM00=0000000000000000 0000000000000000 XMM01=0000000000000000 0000000000000000
XMM02=0000000000000000 0000000000000000 XMM03=0000000000000000 0000000000000000
XMM04=0000000000000000 0000000000000000 XMM05=0000000000000000 0000000000000000
XMM06=0000000000000000 0000000000000000 XMM07=0000000000000000 0000000000000000

```

### Referencias:

- <https://wiki.osdev.org/QEMU>
- <https://manpages.debian.org/stretch/qemu-system-x86/qemu-system-i386.1.en.html>

## Debugger

Para utilizar `gdb` como debugger con `Qemu` debemos correr:

```
$ gdb NUESTRO-KERNEL
```

por ejemplo:

```
gdb kernel.bin.elf
```

luego conectarnos al stub de `qemu` mediante:

```
(gdb) target remote localhost:1234
```

Si nos queremos conectar desde una consola de `WSL` a `qemu` corriendo en windows, debemos usar:

```
(gdb) target remote <nombre_del_equipo_host>.local:1234
```

en este punto ya podemos usar `gdb` como veníamos haciendo; por ejemplo:

```

b kernel.asm:120
Punto de interrupción 1 at 0x1303: file kernel.asm, line 120.
p gdt[3]
$ 1 = {
limit_15_0 = 12543,
base_15_0 = 0,
base_23_16 = 0 '\000',
type = 3 '\003',
s = 1 '\001',
dpl = 0 '\000',
p = 1 '\001',
limit_19_16 = 3 '\003',
avl = 0 '\000',
l = 0 '\000',
db = 1 '\001',
g = 1 '\001',

```

```
base_31_24 = 0 '\000'  
}
```

#### Referencias:

- <https://wiki.osdev.org/GDB>

## Control de ejecución

- `stop`  
Pausa la ejecución de la vm, equivalente a hacer Ctrl+C en gdb
- `cont`  
Reanuda la ejecución de la vm, equivalente al continue de gdb
- `system_reset`  
Reinicia la vm desde el arranque

## Registros

- `info registers`  
Lista los registros del CPU y sus contenidos

## Memory Dump

- `x /nu [addr]`  
Muestra el contenido de N words a partir de la dirección [addr] n es el número que indica cuantos valores se mostrarán (default 1)  
u es el tamaño de la unidad, puede ser<sup>1</sup>:

```
b : byte                h : word (half-word)  
w : doubleword(word)   g : quadword(giant word)
```

f es el formato del número, puede ser:

```
x : hex          d : decimal      u : sin signo  
o : octal       t : binario      c : char  
s : ascii      i : instrucción
```

---

<sup>1</sup>Entre paréntesis el nombre consistente con gdb