Empezando con los ejercicios

Si trabajamos con el emulador, lo primero sería descargarlo de:

https://sourceforge.net/projects/rpiqemuwindows/files/latest/download

Esto llevará un rato, luego de descargado el archivo qemu.zip, conviene moverlo al disco de tu computadora, porque si lo corres desde pendrive será muy lento (ver la jerarquía de memoria).

Descomprimir en la carpeta elegida. Para más detalles sobre el proceso de instalación y configuración del emulador ver el artículo Qemu: Emulando Raspberry Pi

Debugueando un programa en ARM Assembler

Abrir la terminal del sistema Raspbian.

- Crear un archivo con el editor de textos nano: nano hello.s
- Pegar el siguiente código:

```
  .data                         @ indica que comienza la declaración de
variables
  dato1:     .byte 1            @ tipo byte, se inicializa en 1
  dato2:     .byte 'A'          @ tipo byte, caracter A
  var4:      .word 0xB12A       @ tipo word de 32 bits, contenido en
hexadecimal
  .text
  .global main        @ visible en todo el programa (modificador de alcance)
  main:
      mov r1, #4
      mov r0, #5
      add r2, r1,r0    @suma r2=r1+r0
      mov r7, #1
      swi 0             @ si R7=1 swi sabe que deber salir a sistema
operativo
```

- Guardar y salir de nano (Ctrl+X, el programa pregunta si quiere guardar)
- Compilar con la opción -g para generar debugging info

as hello.s -o hello.o -g

- Linkear el código objeto hello.o y obtener el ejecutable hello

gcc hello.o -o hello

- Iniciar el debugger

gdb -tui hello

- Poner un breakpoint inicial en una etiqueta:

b main

- Poner un breakpoint inicial en un número de linea:

b <numero de linea>

- Ejecutar el programa (que avanzará hasta el breakpoint indicado anteriormente)

run

- Mostrar los registros

layout reg

- Avanzar un paso en la ejecución del código

s

- Examinar el contenido de un string asciiz en memoria

x/s &texto

- Examinar el contenido de un word en memoria (en hexadecimal)

x/xw &var4

```
info registers r1 r2 r3 (muestra r1, r2 y r3)
i r  (muestra todos los registros)
i r cpsr (muestra el registro de flags)
print (int) var1 (muestra el contenido de la etiqueta var en memoria ram)
p (int) var2
Si tenemos definido en memoria por ejemplo una etiqueta que se llama:
mivector: .word 12, 14, 16
y deseamos ver todo el contenido de mivector
p/x (int[3]) mivector    (muestra los 3 elementos en hexa)
p/d (int[3]) mivector    (muestra los 3 elemenos en decimal)
p (int[3]) mivector
```

- Puede explorar la opción de Debugging Gdb Extendido (gdb con gef)

https://gef.readthedocs.io/en/master/

---

(143)