Facultad de Informática
Universidad Complutense de Madrid

Fundamentos de Computadores II - Práctica 6 opcional
Programación de un procesador RISC-V comercial

15 de marzo de 2025

1. Objetivos

En esta práctica opcional veremos cómo un programa en ensamblador puede comunicarse con los periféricos del computador en que se ejecuta a través de llamadas al sistema operativo. Para ello desarrollaremos un programa interactivo que hace uso de tales llamadas y lo correremos en un procesador real con arquitectura RISC-V.

2. Adquisición del material

Para la realización de esta práctica debes adquirir la placa EPS32-C3-DevKitM-1 en tu plataforma de venta online favorita (Amazon, Aliexpress, Digikey, Farnell, Mouser, etc.). Es una placa de bajo coste que dispone de un microcontrolador ESP32-C3 que integra un procesador con arquitectura RV32IM y una colección de periféricos.

Adicionalmente para alimentar y programar la placa deberás disponer de un cable USB con conector microUSB tipo B macho en un extremo y USB tipo A macho en el otro (es un cable muy común como el usado para alimentar móviles con conector microUSB). Revisa que el cable sea de transferencia de datos y no solo de alimentación.

Además de esta práctica, este material te permitirá hacer por tu cuenta otros proyectos muy interesantes dado que la placa EPS32-C3-DevKitM-1 dispone de un interfaz WiFi y está soportada por el Arduino IDE: **https://www.electronics-lab.com/getting-started-with-espressifs-esp32-c3-devkitm-1-on-arduino-ide/**

3. Instalación de software adicional

Descarga de la página web de la asignatura el fichero EclipseRV-adds.zip. Descomprime el fichero en la misma carpeta en donde tengas instalado Eclipse (típicamente **`C:/EclipseRV`**). Esta descompresión añadirá en dicha ruta 3 nuevos directorios: **`esptool-win64`**, **`drivers`** y **`termite30`**.

Descarga también de la página web de la asignatura el fichero FC2optativaWS.zip y descomprímelo. Incluye un nuevo *wokspace* únicamente con los dos proyectos que desarrollaras en esta práctica.

4. Llamadas al sistema

Normalmente los programadores, cuando desean acceder a los periféricos de un computador, utilizan llamadas a funciones de biblioteca. Así, cuando queremos que un programa en C++ lea los datos introducidos por el usuario en consola usamos **`cin >>`** y

cuando queremos escribir los resultados usamos **`cout <<`**. Estos operadores forman parte de la biblioteca **`iostream`**. Sin embargo, las funciones que invocan no realizan directamente la comunicación con los periféricos, sino que lo hacen indirectamente a través de llamadas a servicios del sistema operativo en el que correrá la aplicación. Es el sistema operativo el realmente encargado de efectuar la comunicación física entre programa y periférico. Cada servicio se identifica por un código distinto y cada sistema operativo puede ofrecer una colección de servicios diferente.

En ensamblador es posible hacer estas llamadas a servicios utilizando, en el caso de la arquitectura RISC-V, la instrucción **`ecall`**. La ejecución de esta instrucción provoca una excepción que invoca al sistema operativo para que realice el servicio cuyo código ha sido indicado por el programador en el registro **`a7`**. Adicionalmente algunos servicios requieren un argumento adicional que el programa pasa por el registro **`a0`**. Otros servicios, a su finalización, devuelven al programa un resultado también por **`a0`**.

Aunque en la placa EPS32-C3-DevKitM-1 no vamos a correr un sistema operativo completo sí que hemos desarrollado un firmware elemental que inicializa el sistema (**`sp`** incluido), lanza la ejecución de un programa y ofrece al programador una colección de servicios orientados principalmente a permitir la comunicación del programa con una consola remota. La lista de servicios disponibles se muestra a continuación:

Código (**`a7`**)	Descripción del servicio	Argumento (**`a0`**)	Resultado (**`a0`**)
1	Escribe en la consola un entero con signo.	número	–
4	Escribe en la consola una cadena.	dir. cadena	
5	Lee de la consola un entero.	–	número
8	Lee de la consola una cadena.	dir. cadena	–
10	Finaliza el programa devolviendo un valor de estado.	número	–
11	Escribe en la consola un único carácter.	caracter	–
12	Lee de consola un único carácter.	–	caracter
13	Apaga el led de la placa.	–	–
14	Enciende el led de la placa en color blanco.	–	–
15	Enciende el led en un color RGB de 24 bits dado.	RGB	–
29	Devuelve número de µs transcurridos desde reset.	–	us
30	Devuelve número de ms transcurridos desde reset.	–	ms
31	Devuelve número de ciclos transcurridos desde reset.	–	ciclos
34	Escribe en la consola un entero en hexadecimal.	número	–
35	Escribe en la consola un entero en binario.	número	–
36	Escribe en la consola un entero sin signo.	número	–
40	Pausa el programa durante un número de ms dado.	ms	–

Así, si un programa quiere visualizar por consola el número 37 debe incluir el siguiente fragmento de código:

```
...
li a0, 37       //número a visualizar
li a7, 1        //código de servicio (escribe entero)
ecall
...
```

Si lo que quiere es leer de la consola una cadena introducida por el usuario, el fragmento de código a incluir sería el siguiente:

```
.bss
cadena: .space 1024   //donde se ubicará la cadena leída
.text
...
la a0, cadena   //cadena a leer
li a7, 8        //código de servicio (lee cadena)
ecall
...
```

Y, por ejemplo, para encender (en color blanco) el led de la placa deberá incluir:

```
...
li a7, 14       //código de servicio (enciende led)
ecall
...
```

5. Desarrollo de la práctica

Sea el siguiente juego interactivo cuyo objetivo es que el usuario averigüe un número secreto. Para ello, el programa repetidamente pide por consola que se introduzca un número y ofrece pistas de si el número introducido es mayor o menor que el secreto. El programa finaliza cuando el número secreto es acertado.

```
#define NUM 14

int main() {
    int n;
    do {
        cout << "Escriba un numero del 1 al 100: ";
        cin >> n;
        if ( n < NUM )
            cout << "El numero secreto es mayor.\n";
        else if ( n > NUM )
            cout << "El numero secreto es menor.\n";

    } while (n != NUM);
    cout << "Numero secreto acertado.\n";
    return 0;
}
```

En el proyecto **`pr6`** del nuevo *workspace* (véase sección 3) se incluye el siguiente programa en ensamblador equivalente:

```
.equ NUM, 14            //Numero secreto

.data
msgIni:   .string "Escriba un numero del 1 al 100: "
msgMayor: .string "El numero secreto es mayor.\n"
msgMenor: .string "El numero secreto es menor.\n"
msgFin:   .string "Numero secreto acertado.\n"

.text
.global main
main:
    li s0, NUM          //Carga numero secreto
loop:
    la a0, msgIni       //Muestra mensaje de solicitud de numero
    li a7, 4
    ecall
    li a7, 5            //Lee el numero que introduzca el usuario
    ecall
```

```
    beq a0, s0, eloop   //Si el secreto es igual al leido, finaliza
    bgt s0, a0, esMayor
esMenor:
    la a0, msgMenor     //Si es menor, muestra mensaje y vuelve a pedirlo
    li a7, 4
    ecall
    j loop
esMayor:
    la a0, msgMayor     //Si es mayor, muestra mensaje y vuelve a pedirlo
    li a7, 4
    ecall
    j loop
eloop:
    la a0, msgFin       //Muestra mensaje de acierto
    li a7, 4
    ecall
    li a0, 0            //Finaliza el programa
    ret
.end
```

Como puede observarse, este programa hace las llamadas al sistema necesarias para comunicarse con el usuario por consola y, cuando finaliza, retorna al sistema devolviendo 0 en lugar de quedarse en un bucle de espera infinito.

Para poderlo ejecutar sobre la placa EPS32-C3-DevKitM-1, sigue los siguientes pasos:

1. Determina el puerto **COM** de tu ordenador (normalmente siempre será el mismo) al que se conecta la placa EPS32-C3-DevKitM-1, siguiendo los pasos descritos en la sección 6.

2. Compila el programa normalmente.

3. Vuelca el programa en la placa siguiendo los pasos descritos en la sección 7.

4. Arranca la consola remota y configúrala según se describe en la sección 8. Este paso solo tendrás que hacerlo una vez.

5. Resetea la placa pulsando el pulsador que no está etiquetado como BOOT.

6. Si todo va bien, en consola aparece un mensaje similar al siguiente. La última línea ha sido enviada por el programa, las anteriores por el arranque del *firmware*.

```
ESP-ROM:esp32c3-api1-20210207
Build:Feb  7 2021
rst:0x1 (POWERON),boot:0xd (SPI_FAST_FLASH_BOOT)
************************************************
Firmware version 06/05/2025 (c) J.M. Mendias
------------------------------------------------
mvendorid: 0x00000612    marchid: 0x80000001
mimpid:    0x00000001    misa:    0x40101104
------------------------------------------------
SYSCLK_CONF_REG:  0x000a8001
CPU_PER_CONF_REG: 0x0000000c
************************************************
Escriba un numero del 1 al 100:
```

7. Puedes empezar a jugar enviando el primer número al programa. Para ello tecléalo en la parte inferior de la consola y pulsa sobre el icono ↵

8. En cualquier momento puedes conectar y desconectar la placa, el programa siempre arrancará ya que está almacenado en una memoria no volátil de tipo flash. Si deseas reprogramar la placa, no olvides cerrar previamente la consola: no es posible usar el programador y la consola al mismo tiempo ya que comparten puerto **COM**.

Ahora que ya has conseguido ejecutar sobre un procesador RISC-V real un programa demo, desarrolla en el proyecto **pr6_lab** del nuevo *workspace* una ampliación del anterior programa de manera que use más servicios. Por ejemplo, que lleve la cuenta del número de intentos, que chequee el rango del número introducido, que indique el tiempo transcurrido hasta encontrar el número, que ilumine el led en distintos colores cuando acierte, etc.

6. Conexión computador-placa

Dado que el programa lo estas desarrollando en tu computador, pero lo vas a ejecutar en la placa EPS32-C3-DevKitM-1, es necesario conectar ambos usando el cable USB mencionado en la sección 2 por varios motivos. El primero es para que tu computador suministre alimentación a la placa. El segundo para poder descargar en la memoria de la placa el programa compilado en tu computador. Y, por último, para poder establecer una conexión entre la consola remota que arrancarás en tu computador y el programa que se está ejecutando en la placa.

Cuando en un computador conectas cualquier dispositivo por USB, lo hace a través de un puerto asignado automáticamente por *Windows*. Para saber el puerto usado por la placa sigue los siguientes pasos:

1. En cuadro de búsqueda de *Windows* teclea *PowerShell* y ejecuta la aplicación que te sugiere: *"Windows PowerShell"*

2. Aparecerá una ventana en la que deberás teclear el siguiente comando:

```
pnputil /enum-devices /class Ports /connected
```

3. Probablemente indique que no encuentra ningún dispositivo conectado al sistema, si lo hubiera tampoco importa.

4. Conecta la placa al ordenador, espera unos segundos a que se instale automáticamente el *driver* de la misma y vuelve a ejecutar el anterior comando. Esta vez mostrará un mensaje similar a este:

```
Id. de instancia:              USB\VID_1A86&PID_7523\8&3d30120&0&1
Descripción del dispositivo:   USB-SERIAL CH340 (COM9)
Nombre de clase:               Ports
GUID de clase:                 {4d36e978-e325-11ce-bfc1-08002be10318}
Nombre del fabricante:         wch.cn
Estado:                        Iniciado
Nombre del controlador:        oem119.inf
```

5. Puedes ignorar toda la información excepto la segunda línea que informa del puerto **COM** usado. En el caso de mi ordenador, el anterior mensaje indica que se conectó al **COM9**, pero puede ser distinto en el tuyo.

6. Si salieran varios mensajes como el anterior (porque hay varios dispositivos conectados) alterna la ejecución del comando con la conexión/desconexión de la placa para identificar

el puerto **COM** que aparece cuando conectas las placa y desparece cuando la desconectas.

7. Volcado de un programa

En las prácticas anteriores, un programa tras ser compilado era ejecutado y depurado sobre un emulador de RISC-V que estaba incorporado en el IDE de Eclipse. Este emulador no tenía posibilidad de hacer entrada/salida. Por ello, los datos que un eventual usuario debía aportar al programa estaban almacenados como constantes en la sección **.data** y el programa almacenaba los resultados de la ejecución en la sección **.bss**. Para determinar si el programa era correcto usábamos un visor de memoria.

Cuando un programa se ejecuta en un procesador real, es necesario cargarlo en su memoria. Para cargar un programa compilado en la memoria de la placa EPS32-C3-DevKitM-1 sigue los siguientes pasos:

1. Conecta la placa a tu ordenador.

2. En la pantalla de *PowerSchell* ve al directorio en donde está ubicado el programa compilado usando el comando **cd**. Por ejemplo, en mi ordenador tecletaría:

```
cd "C:\Users\Mendias\Documents\FC-2\laboratorios\pr6\Debug"
```

3. Seguidamente teclea el siguiente comando, sustituyendo **com9** por el número de puerto **COM** que obtuviste en la sección 5. Si la ruta en donde instalaste Eclipse es distinta a **C:\EclipseRV**, modifica también la ruta:

```
C:\EclipseRV\esptool-win64\esptool.exe -p com9 write_flash 0 pr6.bin
```

4. Si todo va bien, en la placa se iluminará un led en color verde.

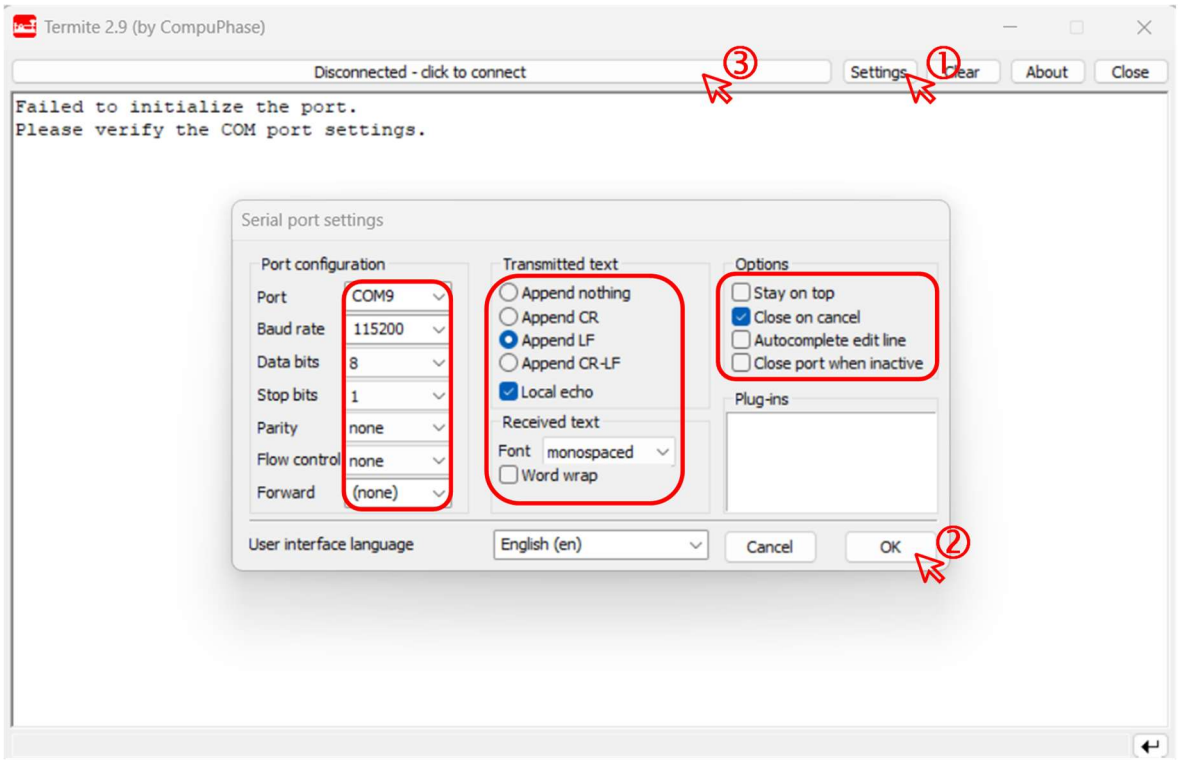
7. Arranque y configuración de la consola remota

La placa EPS32-C3-DevKitM-1 no dispone de teclado ni de pantalla, en su lugar dispone de un puerto serie que, a través del cable USB, le permite comunicarse con una consola textual (no gráfica) corriendo remotamente sobre el computador al que está conectada. Esta consola permite que el programa corriendo en placa, si usa las llamadas al sistema mencionadas en la sección 3, pueda comunicarse con el usuario usando la pantalla y el teclado del computador.

Para arrancar en tu computador la consola remota teclea el siguiente comando en la pantalla de *PowerShell*. Si la ruta en donde instalaste Eclipse es distinta a **C:\EclipseRV**, modifica la ruta adecuadamente:

```
C:\EclipseRV\termite30\termite.exe
```

En una nueva ventana aparecerá la consola, pero antes de usarla deberás configurarla para que use el mismo protocolo de comunicación que el *firmware* de la placa. Para ello, pulsa sobre el botón **Settings** y configura todas las opciones según lo indicado en la siguiente figura, en el desplegable **Port** deberás seleccionar el puerto **COM** que obtuviste en el apartado 6. Guárdalas pulsando OK y pulsa el botón que se indica para conectar la consola con la placa.


Termite 2.9 (by CompuPhase)
Disconnected - click to connect
Settings
Clear
About
Close
Failed to initialize the port.
Please verify the COM port settings.
Serial port settings
Port configuration
Port
COM9
Baud rate
115200
Data bits
8
Stop bits
1
Parity
none
Flow control
none
Forward
(none)
Transmitted text
Append nothing
Append CR
Append LF
Append CR-LF
Local echo
Received text
Font
monospaced
Word wrap
Options
Stay on top
Close on cancel
Autocomplete edit line
Close port when inactive
Plug-ins
User interface language
English (en)
Cancel
OK
①
②
③