Introducción al entorno de desarrollo RISCV

Departamento de Tecnología Electrónica
Universidad de Sevilla

1 Material

- Ordenador con RARS y CoolTerm instalados.
- Placa Basys3 con HispaRISC ya programado.
  (Se trata de un procesador que se ha implementado sobre una FPGA Basys3 e incluye la ISA completa RV32I).
- Carpeta lab_kit con los archivos necesarios para la práctica.

2 Objetivos

- Familiarizarse con el entorno de desarrollo de aplicaciones para RISCV.
- Programar en ensamblador y simular el programa utilizando RARS.
- Ensamblar programas utilizando RARS y generar su código máquina.
- Utilizar el programa CoolTerm para enviar programas a HispaRISC.
- Ejecutar programas en HispaRISC.

3 Estudio previo

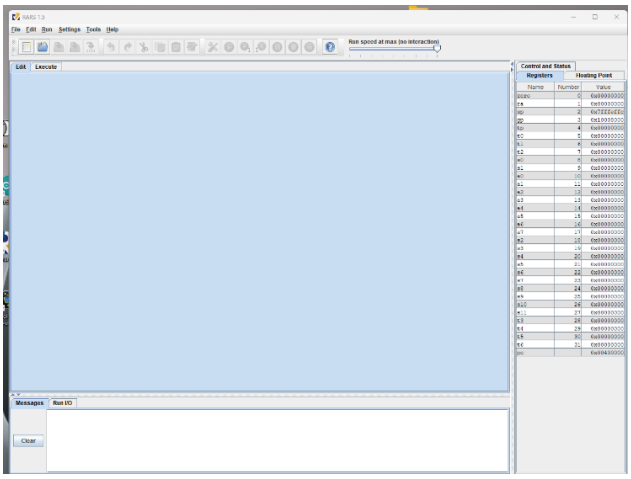
Revisar y estudiar el conjunto de instrucciones ISA RISCV RV32I.

Comprender el programa suma.S que se le proporciona. Tenga en cuenta que las 4 últimas instrucciones son llamadas al sistema, la primera imprime un entero por pantalla y la segunda devuelve el control al sistema operativo:

```
li a7, ECALL_PRINTINT
ecall
li a7, ECALL_EXIT
ecall
```

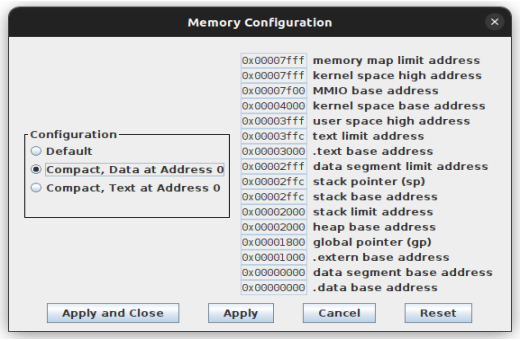
4 Uso del simulador RARS

Para arrancar el simulador haga doble clic sobre el icono de acceso directo a RARS que se encuentra en el escritorio.



4.1 Configuración del mapa de memoria compatible con HispaRISC

El simulador RARS permite configurar varios espacios de memoria. Para que el código máquina generado sea compatible con el micro HispaRISC, debemos seleccionar el mapa de memoria compacto con segmento de datos en la dirección 0. Para ello seleccione Settings > Memory Configuration y elija la opción segunda:



4.2 Creación o carga de un archivo en ensamblador

Hay dos posibilidades:

1) Seleccionar File > New y escribir un programa en la ventana de edición.
2) Seleccionar File > Open y cargar un programa ya escrito en un archivo

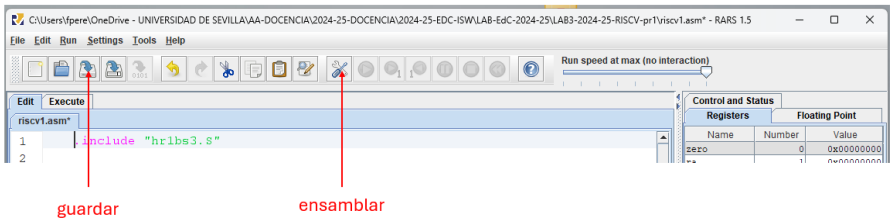
Para hacer una primera prueba cargue el archivo suma.S que se le proporciona en lab_kit.

IMPORTANTE: el archivo a utilizar debe contenerse en una carpeta en la que también se encuentre el fichero de definiciones de cabecera “hr1bs3.S”

Si realiza algún cambio en el programa puede salvarse con File > Save as.

4.3 Simulación del programa

Para ensamblar el programa debe pulsar en el botón de las herramientas (ver figura): Assemble current files and clear breakpoints).



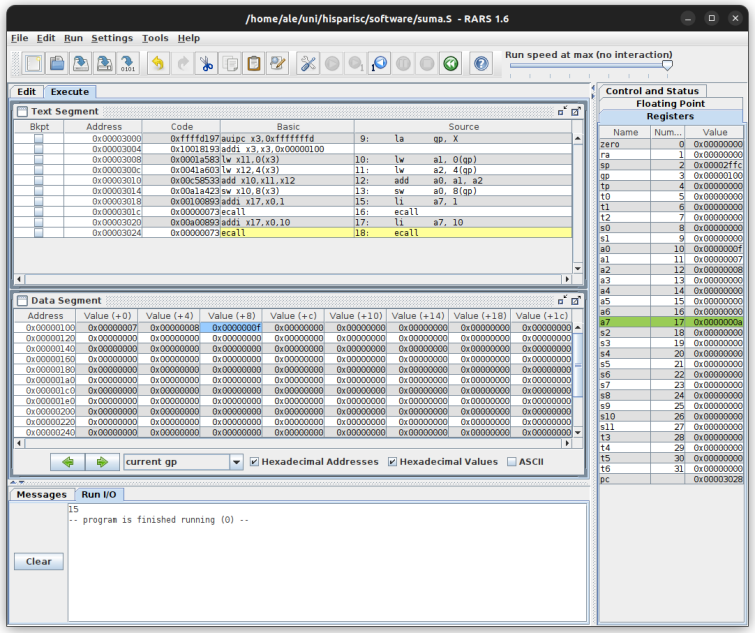
Una vez pulsado el botón de ensamblar, el programa pasa a modo ejecución y se podrá ejecutar el programa de forma continua o paso a paso, utilizar puntos de ruptura, consultar el contenido de la memoria y registros, etc. *Vea la figura en la página siguiente.*

El término ensamblar es el proceso de obtención del “código máquina” a partir de un programa escrito en lenguaje ensamblador. En la ventana de ejecución puede verse el código binario que se genera con dicho ensamblado (columna “Code” en la ventana “Text Segment”). Asimismo, si se han declarado variables en el segmento de datos, también es posible ver su valor en la ventana “Data Segment”). A la derecha puede ver el contenido de todos los registros.

Ejecute el programa paso a paso pulsando el botón 

Compruebe como se va avanzando por las instrucciones y como los registros van cambiando de valor.

También puede retroceder en la ejecución o volver al inicio del programa con los boton 



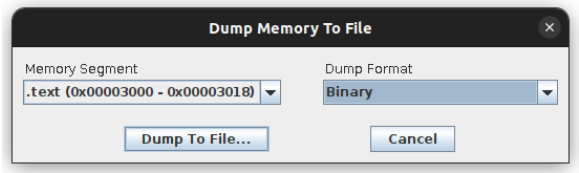
4.4 Generación de ficheros binarios para HispaRISCV

Se debe salvar el contenido de los segmentos de texto (el código máquina) y de datos (las variables en memoria) en sendos registros para volcarlos luego sobre el procesador. A continuación se describe el proceso completo.

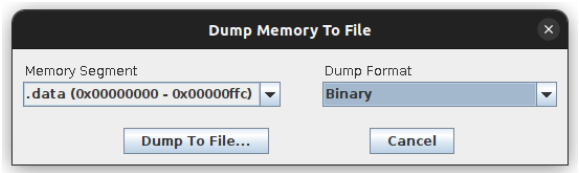
Para el archivo binario correspondiente al código máquina del programa (segmento de texto), estando en la pestaña de ejecución, se selecciona: File > Dump Memory y elegimos:

- Memory Segment: .text
- Dump Format: Binary

Es conveniente salvar el fichero añadiendo al nobre del archivo de programa los sufijos .text.bin, es ecir, suma.text.bin (de esta forma sabremos que es el fichero binario del segmento de texto).



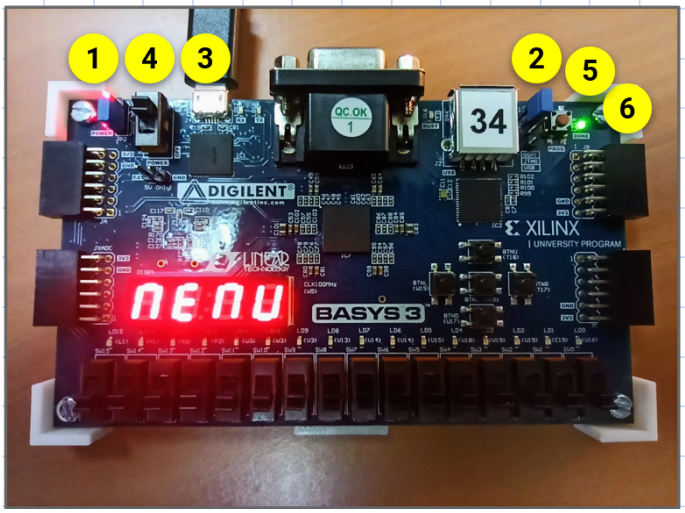
Si el programa tiene segmento de datos con valores iniciales de variables (como es el caso del programa suma.S), hay que generar también el archivo binario correspondiente. El proceso es similar al anterior: en la pestaña de ejecución se selecciona File > Dump Memory para guardar el segmento de datos en un archivo binario (suma.data.bin).



5 Arranque y uso de HispaRISCV

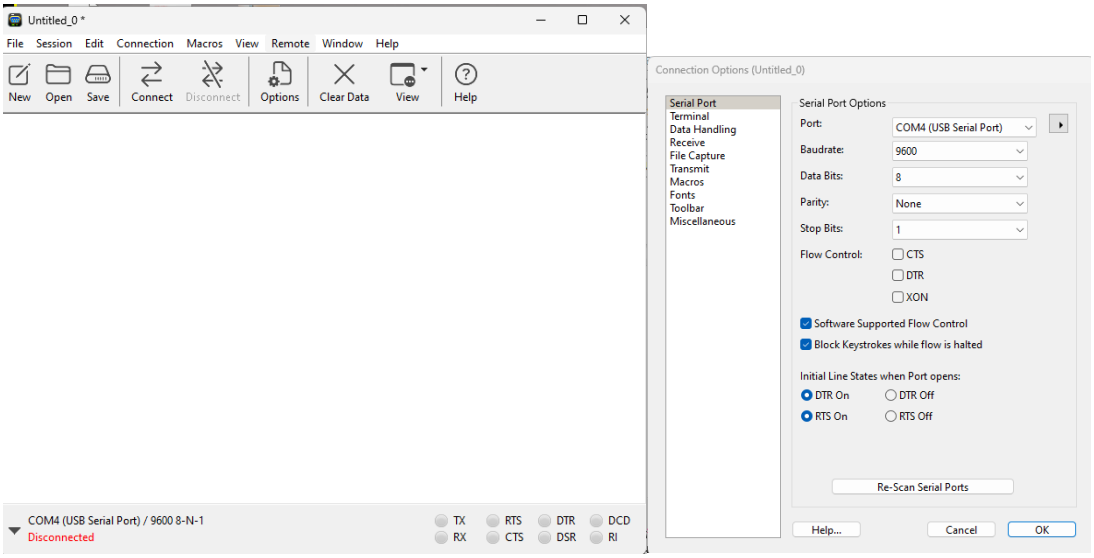
Vamos a ver cómo podemos transferir esos archivos binarios al procesador para ejecutarlos.
Tome la placa Basys3 y siga el siguiente proceso (ver figura):

1. Establecer la alimentación en modo USB.
2. Establecer la programación en modo QSPI
3. Conectar el cable micro USB proveniente del PC al puerto USB
4. Encender la placa. Se ilumina el Testigo POWER.
5. Pulsar el boton PROG
6. Tras unos segundos, se ilumina el Testigo DONE.
7. Tras un parpadeo de los LEDs, el display muestra la cadena “MENU”



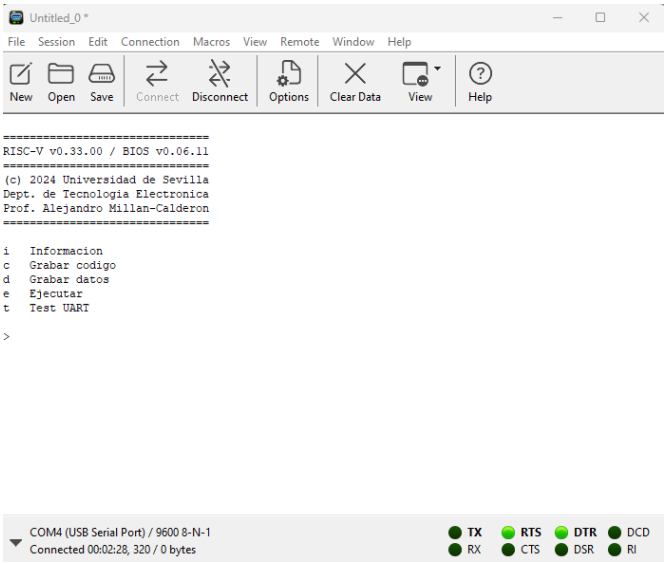
Configuración de la comunicación del PC con el micro

Se debe abrir la aplicación CoolTerm. A continuación, seleccione Connection > Options, elija el puerto serie adecuado (el asignado por el sistema operativo a la placa Basys3, normalmente identificado como “COMn (USB Serial Port)” y configure el protocolo de comunicación en 9600 8-N-1.

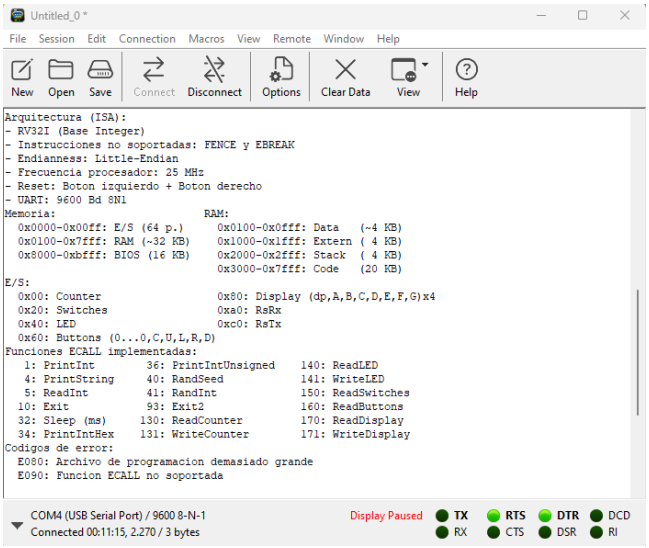


Comunicación con el micro y menú BIOS

Pulse Connect y reinicie el micro pulsando simultáneamente izquierda (BTNL) y derecha (BTNR) en la botonera de la placa.
Si todo funciona correctamente, el micro devolverá el menú de la BIOS como en la siguiente pantalla:



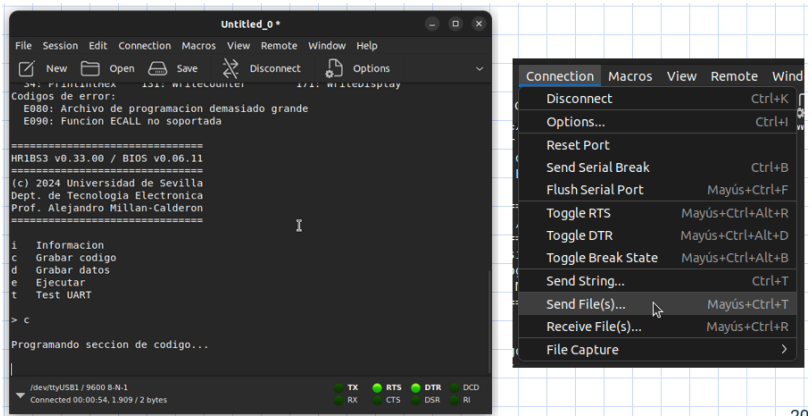
La opción “i” permite visualizar las especificaciones de la placa y de la BIOS:



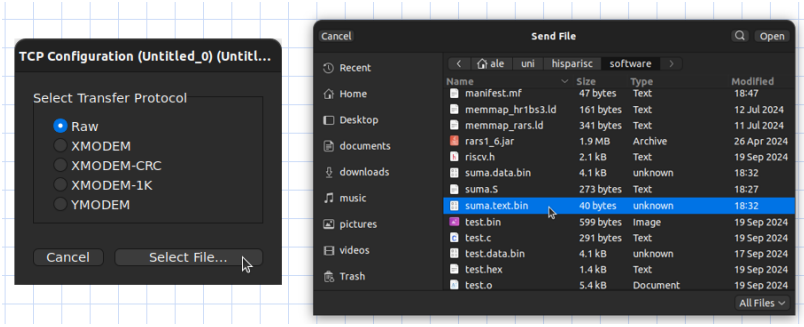
Grabación del segmento de código

- Escriba el comando “c” (Grabar código): el micro entra en modo de espera
- Seleccione Connection > Send File(s)

Vea la figura en la página siguiente.



- Seleccione protocol RAW
- Seleccione el archivo binario (.text.bin), y espere a que finalice la grabación

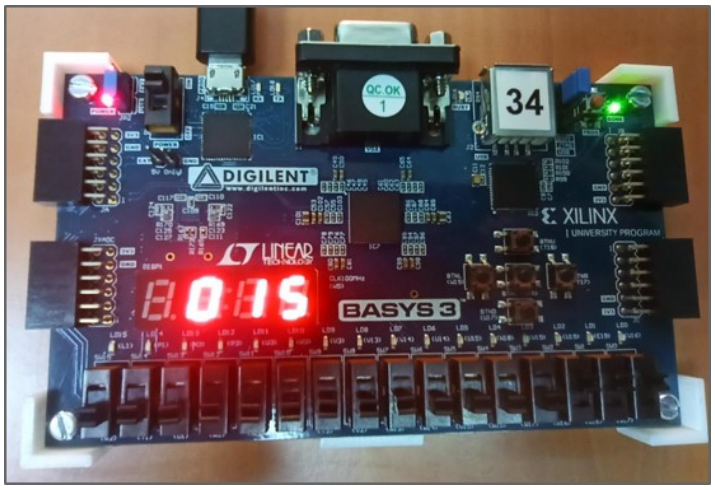


Grabación del segmento de datos

- Escriba el comando “d” (Grabar datos): el micro entra en modo de espera
- Seleccione Connection > Send File(s)
- Seleccione nuevamente RAW y el archivo binario de datos (.data.bin) y espere a que finalice la grabación.

Ejecución del programa

Para comenzar la ejecución del programa, escriba el comando “e”. El micro mostrará brevemente la cadena “EXEC” en el Display y ejecutará el programa almacenado:



Finalización de la ejecución.

Si el programa finaliza con una ecall correspondiente a una finalización (Exit o Exit2), el micro retornará al estado inicial (cadena MENU en el display).

En caso de programas sin finalización, puede reiniciarse el micro para retornar al menú inicial (cadena MENU en el display) pulsando simultáneamente izquierda (BTNL) y derecha (BTNR) en la botonera de la placa.

6 Ejercicio de aplicación

Desarrolle un programa (suma100.S) que realice el siguiente cálculo:

z = 100 + x

y muestre el resultado en pantalla.

NOTA: active Settings > Popup dialog for input syscalls (5,6,7,8,12) para que en la simulación RARS notifique al usuario que debe introducir un entero.

Utilice ECALL_READINT para leer el valor de x y ECALL_PRINT para mostrar el resultado z. Esto puede hacerse de modo similar a como ya se hizo para suma.S con ECALL_PRINTINT y ECALL_EXIT (Apartado 3 de esta práctica).

a) Simule el programa en RARS y compruebe el correcto funcionamiento:
- x debe introducirse mediante teclado
- z debe mostrarse en pantalla

b) Grabe el programa en HispaRISC (siga el proceso del apartado anterior)

c) Ejecute el programa en HispaRISC:
- x debe introducirse mediante los switches de la placa

NOTA: puede incluir las siguientes líneas para que de tiempo a leer el resultado antes de que el programa termine y vuelva a aparecer la leyenda MENU. Se trata de una llamada al sistema que inserta ciclos de espera.

```
li a0, 3000
li a7, ECALL_SLEEP
ecall
```

7 Otros ejercicios de aplicación

a) Sin utilizar llamadas al sistema, escriba un programa que continuamente lea el dato almacenado en la dirección 0x20 y lo escriba en la dirección 0x40. Cambie los switches de posición y observe el resultado.

b) Escriba un nuevo programa que continuamente lea los switches y escriba el valor leído en los leds. Utilice las llamadas al sistema ECALL_READINT y ECALL_WRITELED.