

MANUAL DE PRÁCTICAS

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORES 1

1º GRADO EN INGENIERÍA INFORMÁTICA

UNIVERSIDAD DE ZARAGOZA

Material elaborado por

Darío Suárez Gracia

Juan Pablo López Grao

María Villarroya Gaudó

Carlos J. Pérez Jiménez

Francisco J. Martínez Domínguez

CONTENIDO

1.	Introducción	2
1.1	Información Importante	3
1.2	Bibliografía (prácticas)	3
2	Descripción del entorno de trabajo.....	3
3	Práctica 1: Introducción al Entorno Keil μ Vision4 y al repertorio de Instrucciones ARMv4.....	7
3.1	Objetivos y competencias a adquirir.....	7
3.2	Trabajo previo	7
3.3	Trabajo en laboratorio	8

1. INTRODUCCIÓN

Las prácticas de esta asignatura están repartidas en seis sesiones. Las cuatro primeras sesiones (enunciados de prácticas 1, 2 y 3) están enfocadas al aprendizaje básico del repertorio de instrucciones y tipos de datos de la arquitectura de lenguaje máquina (ISA) de un procesador real (ARMv4), siendo capaz al terminar las mismas de entender y escribir programas en lenguaje ensamblador que no se comuniquen con dispositivos externos. Estas tres primeras prácticas (cuatro primeras sesiones) constituyen el *corpus práctico* de la iniciación a la asignatura y su seguimiento es fundamental para una correcta comprensión de los fundamentos de la misma, así como para abordar satisfactoriamente la realización de las dos últimas prácticas.

Por otro lado, en las dos últimas sesiones (enunciados 4 y 5) aprenderás los mecanismos fundamentales de la ISA para la encapsulación de código a través de la invocación de funciones y paso de parámetros, y comprenderás los mecanismos de entrada/salida (E/S) con la ayuda del simulador. Además serás capaz de mezclar código ensamblador con el lenguaje de alto nivel C. El resultado de estas prácticas deberá ser entregado y será evaluado, contando para la nota final de la asignatura.

Todas las prácticas se desarrollarán sobre la plataforma de desarrollo Keil μ Vision4, que integra en un mismo entorno gráfico un editor de código fuente, un ensamblador, un compilador de C/C++, un enlazador (linker) y un depurador con capacidad de simular la E/S desde distintos periféricos, entre otras cosas. Puedes descargarte el fichero de instalación y el manual del simulador del directorio de la asignatura en hendrix:

Enlace windows: \\Hendrix-cifs\Practicas\AOC1\.

Enlace unix: /export/home/practicas/Practicas/AOC1/

1.1 INFORMACIÓN IMPORTANTE

Antes de asistir a la sesión práctica en el laboratorio, es imprescindible la lectura y preparación de la práctica correspondiente (puedes ser evaluado al respecto). Esto incluye el repaso del material visto en clase hasta la fecha y la resolución de los apartados previos de cada práctica en caso de estos existan.

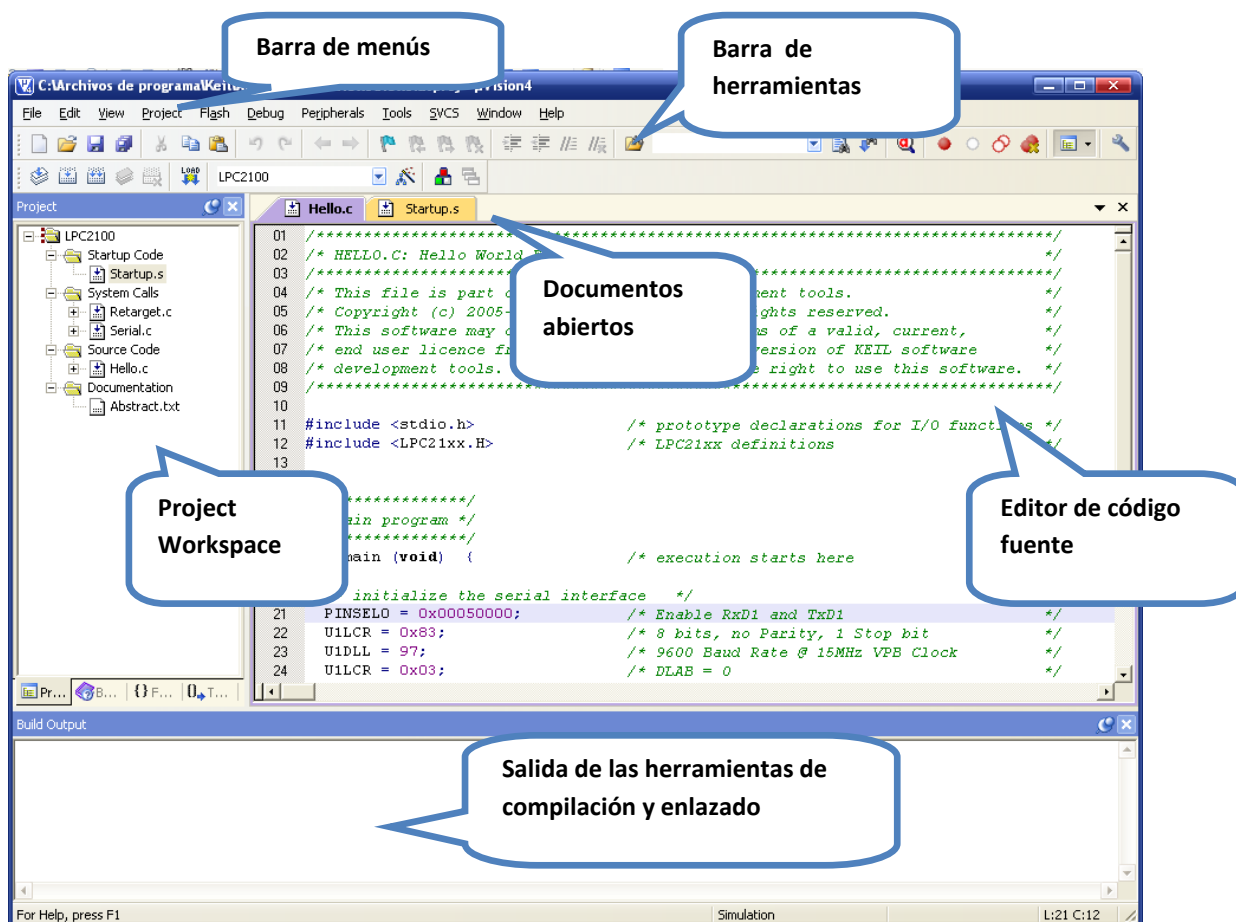
Se recomienda al terminar cada sesión guardar los resultados de la práctica para su posible utilización en prácticas siguientes.

1.2 BIBLIOGRAFÍA (PRÁCTICAS)

- ARM Assembly Language. Fundamentals and Techniques. William Hohl.
- ARMv4T Partial Instruction Set Tables de la University of New South Wales (Australia), descargable desde moodle, hendrix y desde: <http://www.cse.unsw.edu.au/~cs3221/labs/>.
(Nota: Esta página es muy recomendable)

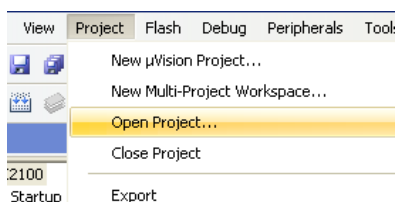
2 DESCRIPCIÓN DEL ENTORNO DE TRABAJO

Keil μ Vision4 es un entorno de desarrollo integrado (IDE) compuesto por un editor, un conjunto de herramientas para compilar, ensamblar y enlazar y un depurador para crear y verificar vuestros programas. La pantalla principal se compone de varias partes entre las que destacamos:

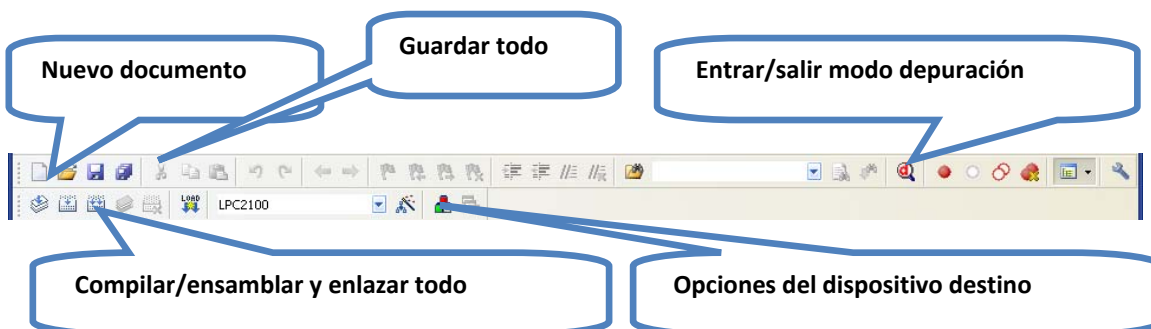


Esencialmente, el desarrollo de un programa en Keil μ Vision4 se estructura dentro del concepto de “proyecto”. Así, cuando creamos un nuevo proyecto, todos los ficheros de código fuente que lo componen deberán aparecer listados en el área *Project Workspace*. Cuando hagamos doble click sobre un determinado nombre de fichero en el Project Workspace, éste se abrirá en el *editor de código fuente* que aparece a la derecha. Si tenemos varios ficheros de código fuente abiertos al mismo tiempo, podremos cambiar entre ellos pulsando sobre su nombre en la *barra de documentos abiertos*.

La barra de herramientas y menús permiten realizar las operaciones más comunes. Por ejemplo, para empezar el desarrollo de un programa es necesario crear un nuevo proyecto desde *Project -> New μ Vision Project...* Igualmente, cuando deseéis recuperar un proyecto previamente creado, podréis abrirlo desde *Project -> Open Project...*

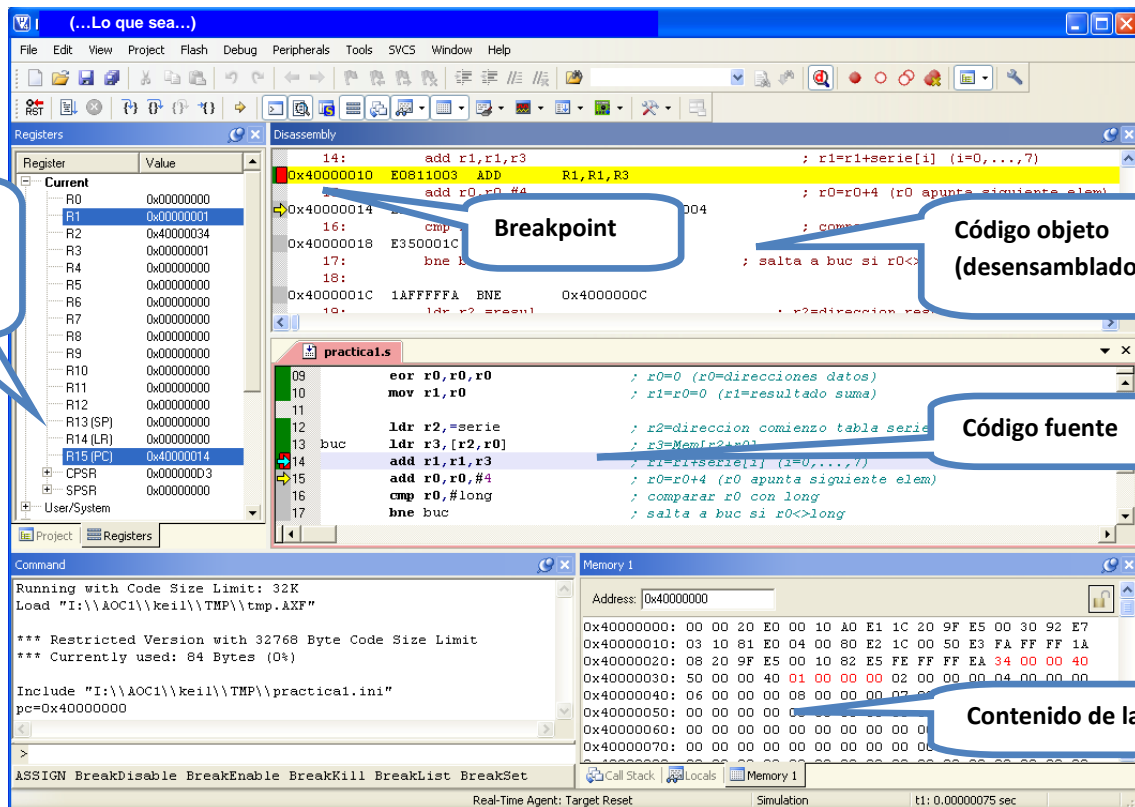


La barra de herramientas contiene algunos botones a los que recurriréis habitualmente. Entre ellos destacaremos los siguientes:



Con *Nuevo documento* podréis crear nuevos ficheros de código fuente que agregar a vuestro proyecto (obsérvese que los ficheros de código fuente en ensamblador deberán tener la extensión .s). Con el botón *Guardar todo* podréis guardar el estado actual de vuestro proyecto, incluyendo todas sus fuentes. Con *Compilar/ensamblar y enlazar todo* podréis generar el código destino para el depurador/simulador. Y con el botón *Entrar/salir modo depuración* podréis comprobar que vuestro programa se comporta conforme a lo esperado. Al pulsar este último botón, pasaréis del modo desarrollo (*Build mode*) al modo depuración (*Debug mode*).

Al entrar en modo depuración, aparecen nuevos botones y ventanas en la pantalla principal, pudiendo mostrar, redimensionar u ocultar algunas de éstas a nuestro antojo:












Así, en el modo depuración aparecerá (si no, puede ser activada desde View -> Registers Window, o desde el correspondiente botón de la barra de herramientas) una ventana con el valor actual de los registros de la CPU (obsérvese que se actualiza en cada paso de ejecución).

Del mismo modo, también podemos activar/desactivar la aparición de una ventana con el contenido de una región de memoria (la que queramos) a través de View -> Memory Windows -> Memory 1. Si deseamos ver distintas regiones de memoria de forma simultánea, podemos abrir hasta cuatro ventanas de memoria (View -> Memory Windows -> Memory 2 / 3 / 4). Es recomendable que en una de ellas veas las instrucciones y en otra alguna zona de interés como pueda ser la pila o algún segmento con datos que vayáis a manipular.

En las ventanas de código fuente y código objeto, una flecha amarilla nos señalará la siguiente instrucción a ser ejecutada. Mediante la pulsación de la tecla F11 (Step Into) podremos realizar una ejecución paso a paso. Observa que si la instrucción ejecutada escribe sobre un registro este se sombreadá sobre color gris dentro de la ventana de registros.

Especialmente útiles son las opciones que aparecen en la barra de herramientas cuando nos encontramos en el modo depuración. Destacamos algunas de ellas:

	Salir del modo depuración (volver al modo desarrollo)
	Introducir un breakpoint ¹
	Simular la llegada de una señal reset al procesador. Eso provoca la excepción reset y el salto de la ejecución a la dirección 0x0.
	Ejecutar hasta que encuentre un breakpoint
	Ejecución paso a paso
	Ir a la siguiente instrucción a ejecutar
	Mostrar/ocultar ventana de código objeto (desensamblado)
	Mostrar/ocultar ventana de contenido de registros
	Mostrar/ocultar ventanas de contenido de memoria

Observa que si colocas el ratón sobre cualquier botón de la aplicación esta te muestra la función que realiza y en caso de duda puedes ir al menú de Ayuda donde la guía de usuario y la referencia del programa te las resolverán. Para comprobar cómo funciona busca en la Ayuda para qué sirve un Breakpoint..



¹ También es posible introducir/quitar un breakpoint haciendo doble click sobre la instrucción correspondiente, bien en la ventana del código fuente o en la del desensamblado.

3 PRÁCTICA 1: INTRODUCCIÓN AL ENTORNO KEIL μ VISION4 Y AL REPERTORIO DE INSTRUCCIONES ARMV4

Esta práctica será tu introducción a un entorno de desarrollo para sistemas embebidos y al lenguaje ensamblador.

3.1 OBJETIVOS Y COMPETENCIAS A ADQUIRIR

- Manejar el entorno de trabajo Keil μ Vision4.
- Ser capaz de compilar, enlazar, y ejecutar un programa. Es decir, pasarás de un fichero de texto con instrucciones en ensamblador a un fichero objeto y de aquí a uno ejecutable con código máquina.

3.2 TRABAJO PREVIO

Estudia con atención el siguiente código ensamblador ARM y contesta a las preguntas planteadas. Si alguna pregunta no la puedes contestar, recuérdala, porque la podrás resolver ejecutando el programa con el simulador.

```
        AREA datos,DATA,READWRITE    ; area de datos

long    EQU 7*4                      ; long=28 (bytes que ocupa serie)
serie   DCD 1,2,4,6,8,7,9            ; serie es una tabla de 7 enteros
resul   DCB 0                        ; resultado

        AREA prog,CODE,READONLY      ; area de codigo
        ENTRY                        ; primera instruccion a ejecutar

        eor r0,r0,r0                  ; r0=0 (r0=direcciones datos)
        mov r1,r0                     ; r1=r0=0 (r1=resultado suma)

        ldr r2,=serie                 ; r2=direccion comienzo tabla serie
buc     ldr r3,[r2,r0]                 ; r3=Mem[r2+r0]
        add r1,r1,r3                  ; r1=r1+serie[i] (i=0,...,7)
        add r0,r0,#4                  ; r0=r0+4 (r0 apunta siguiente elem)
        cmp r0,#long                  ; comparar r0 con long
        bne buc                       ; salta a buc si r0≠long

        ldr r2,=resul                 ; r2=direccion resul
        str r1,[r2]                   ; Mem[r2]=r1 (almacena resultado)

fin     b fin                          ; fin de programa

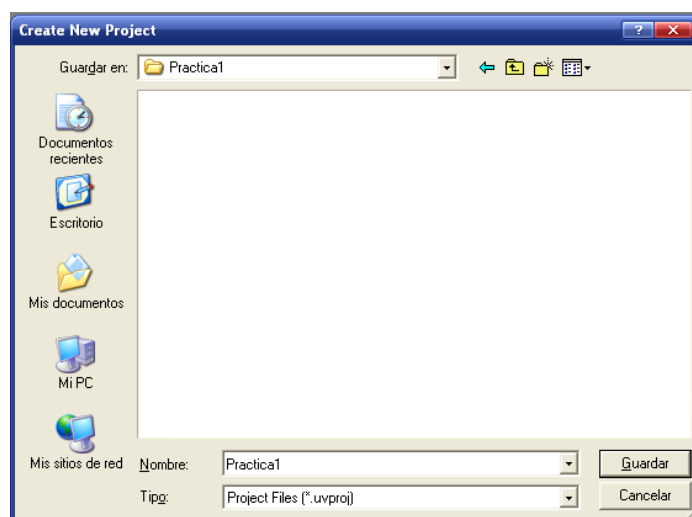
        END                          ; fin de ensamblado
```

- 1) ¿Cuántas sentencias de lenguaje ensamblador forman el programa anterior? ¿Y cuántas directivas?
- 2) ¿Cuántas sentencias ejecuta el programa anterior? ¿Depende el número de alguna variable?
- 3) ¿Qué operación realiza la instrucción `eor r0, r0, r0` y cómo podrías conseguir el mismo resultado con otra instrucción?
- 4) ¿Cuánta memoria requiere el vector almacenado por `serie` y cual es el tamaño de cada elemento?
- 5) ¿Modifica este programa el contenido de la variable `resul`? Y si es así, ¿qué valor contendrá al final de la ejecución del programa?
- 6) ¿Para qué sirve la directiva `ENTRY`?
- 7) ¿Es finito el tiempo de ejecución de este programa?

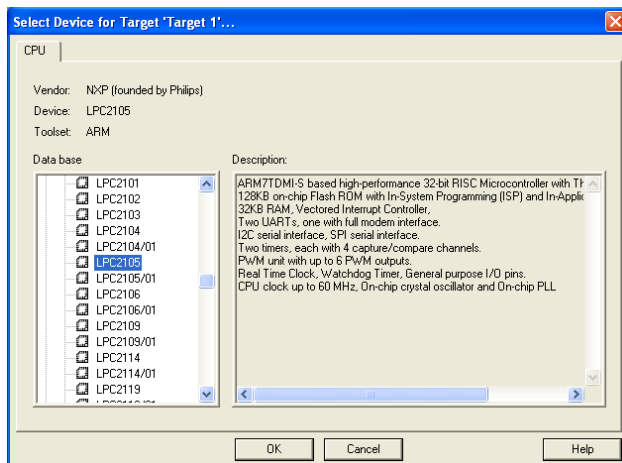
Al comenzar la sesión de prácticas tu profesor comentará las respuestas y también pueden ser preguntadas durante la evaluación final de las prácticas.

3.3 TRABAJO EN LABORATORIO

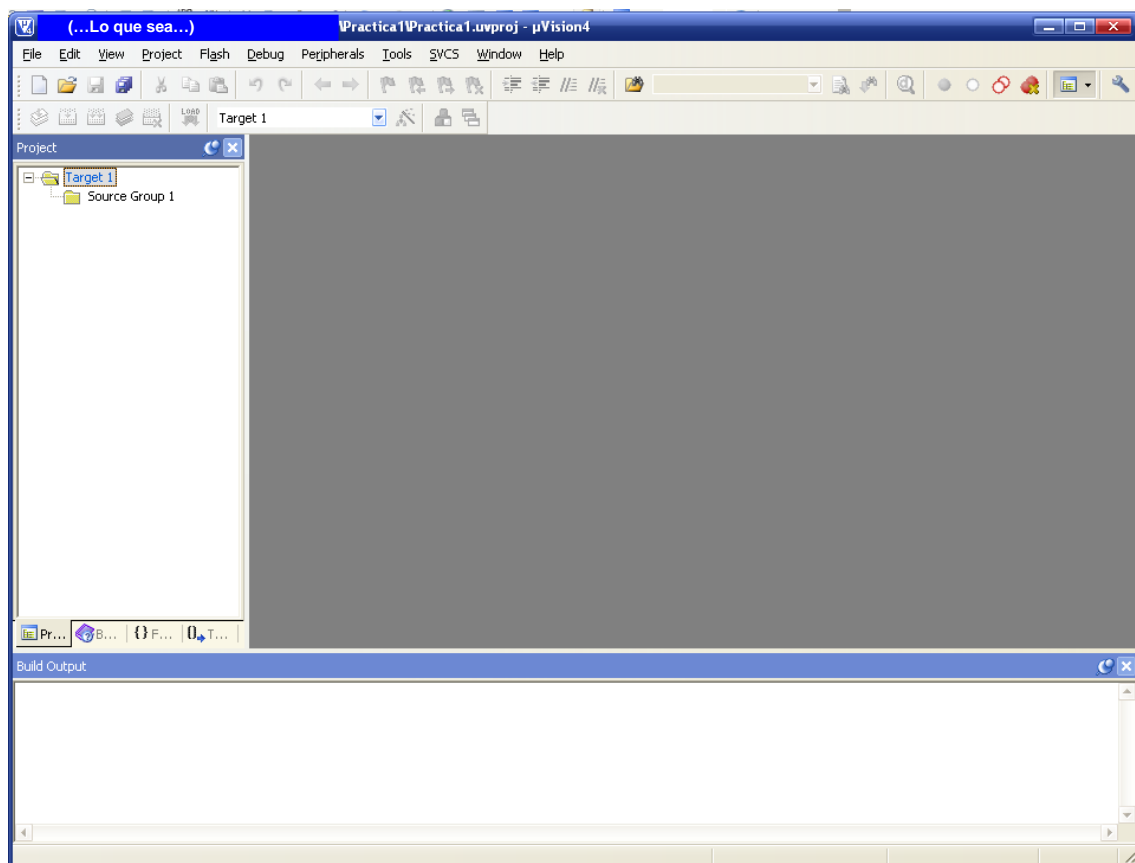
- 1) Descargad desde Moodle o hendrix el fichero `aoc1_p1.zip` y descomprimidlo en vuestro directorio de trabajo (por ejemplo: `D:\VUESTRONOMBRE\Practica1`).
- 2) Ejecutad Keil uVision4 desde el menú Programación.
- 3) Id a Project -> New µVision Project...
- 4) Id al directorio de trabajo (por ejemplo: `D:\VUESTRONOMBRE\Practica1`) y ponedle nombre al proyecto (por ejemplo, `Practica1.uvproj`).



- 5) Seleccionad el tipo de dispositivo destino con el que vais a trabajar en el proyecto. En este caso, elegiremos: NXP (founded by Philips) -> LPC2105. Pulsad OK. A la siguiente pregunta responderemos No.

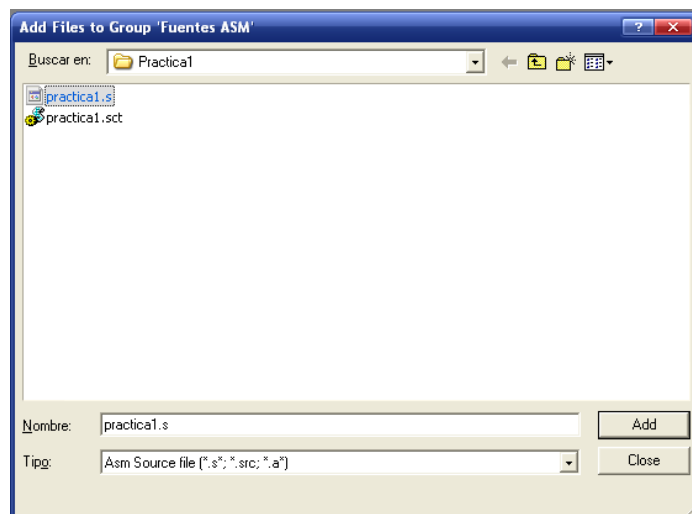


Veréis ahora algo parecido a lo siguiente:

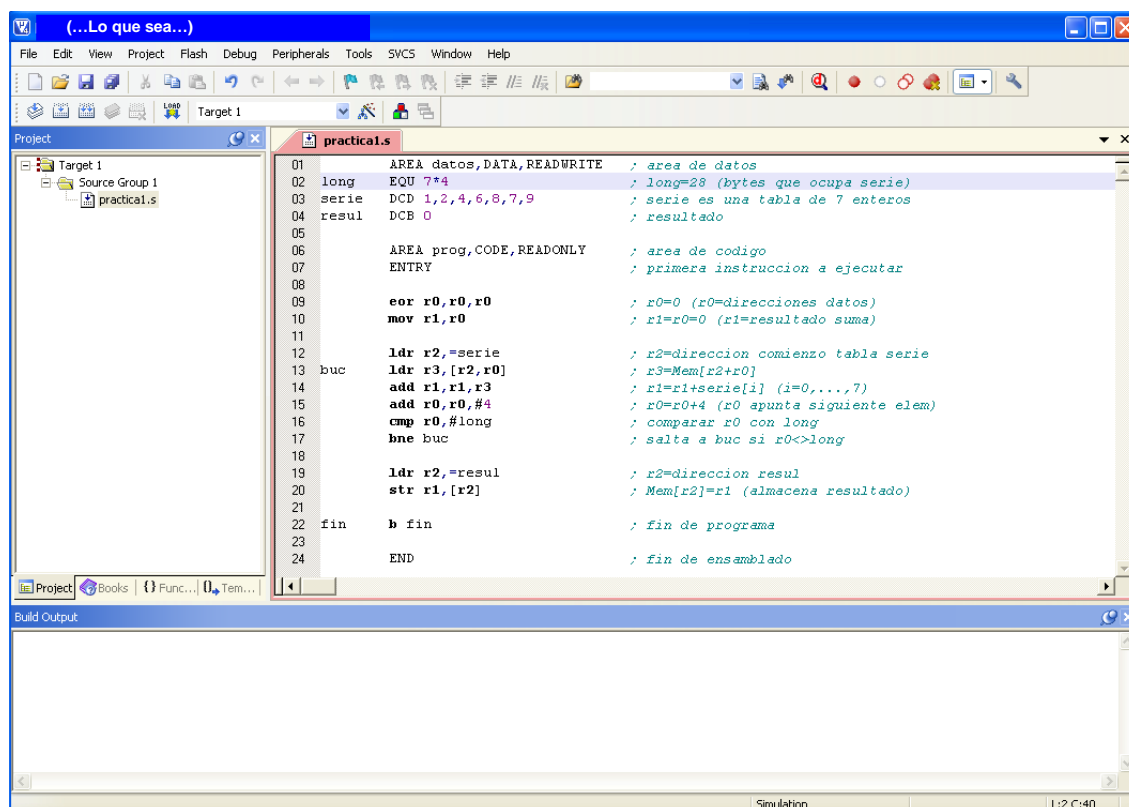


- 6) Ahora prepararemos la estructura de ficheros en la ventana del Project Workspace. Pulsad dos veces lentamente (no doble click) sobre *Source Group1* para renombrarlo. Introducid un nuevo nombre: "Fuentes ASM" (sin las comillas).
 - Obsérvese que si quisiéramos añadir nuevos grupos, deberíamos hacer click derecho sobre *Target1*, seleccionaríamos *Add Group*, y le asignaríamos un nuevo nombre (por ejemplo, "Fuentes C"). De este modo podríamos, por ejemplo, crear un nuevo grupo para fuentes o cabeceras en C, lo que nos resultará útil a partir de la práctica 4 (ahora no es necesario).


- 7) Hacemos doble click sobre “Fuentes ASM”. En el cuadro de diálogo resultante, seleccionamos como Tipo “ASM Source File (*.s; *.src; *.a*)”. Dado que previamente habíamos descomprimido el fichero 10_11_aoc1_p1.zip en el directorio de trabajo, ahora podemos ver allí el fichero “practica1.s”. Lo seleccionamos y pulsamos en “Add”. Finalmente cerramos (Close).

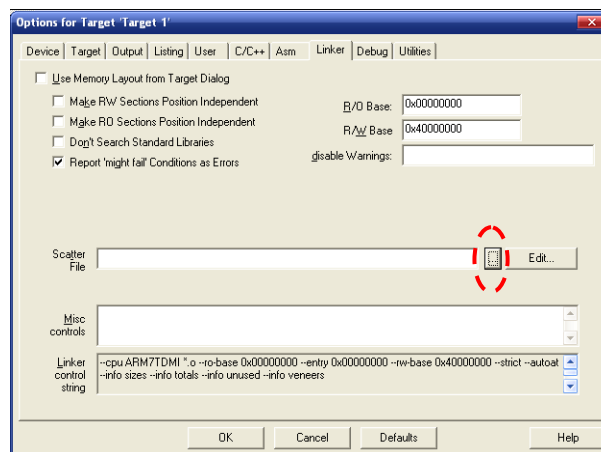


- 8) Con esto hemos vuelto a la pantalla principal. Si ahora pulsamos sobre el símbolo “+” que ahora aparece junto a “Fuentes ASM” en el *Project Workspace*, podemos ver el nuevo fichero fuente (“practica1.s”). Hacemos doble click sobre su nombre, y se abrirá en el editor de código fuente:

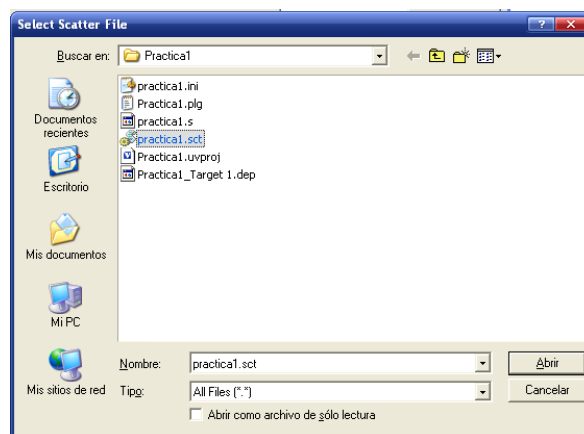


- 9) Ahora queremos ensamblar y enlazar el código fuente. Para ello, necesitaremos previamente configurar el linker, de modo que le indiquemos dónde queremos que se sitúe nuestro código

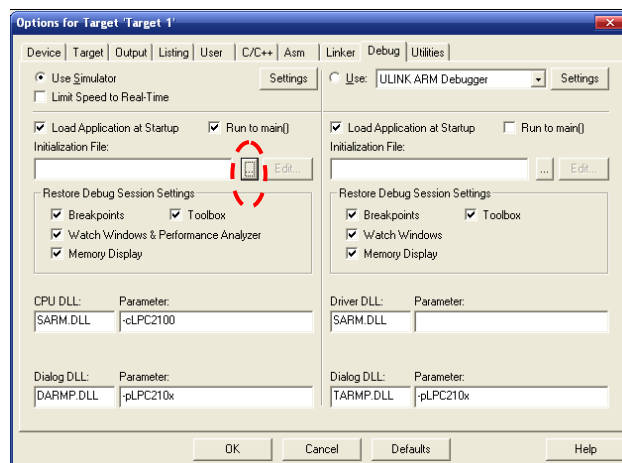
ensamblado. Con ese fin, pulsaremos el botón de Target Options () de la barra de herramientas (o, a través del menú, *Project -> Options for Target 'Target 1'...*) y allí iremos a la pestaña Linker. Desmarcamos, si está marcada, la opción “Use memory layout from target dialog”, y a continuación pulsamos en el botón que aparece rodeado en la figura siguiente:



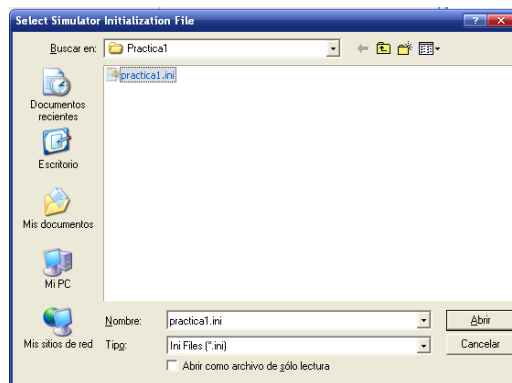
- 10) En el cuadro de diálogo resultante, seleccionamos el fichero de configuración practica1.sct, y le damos a Abrir.





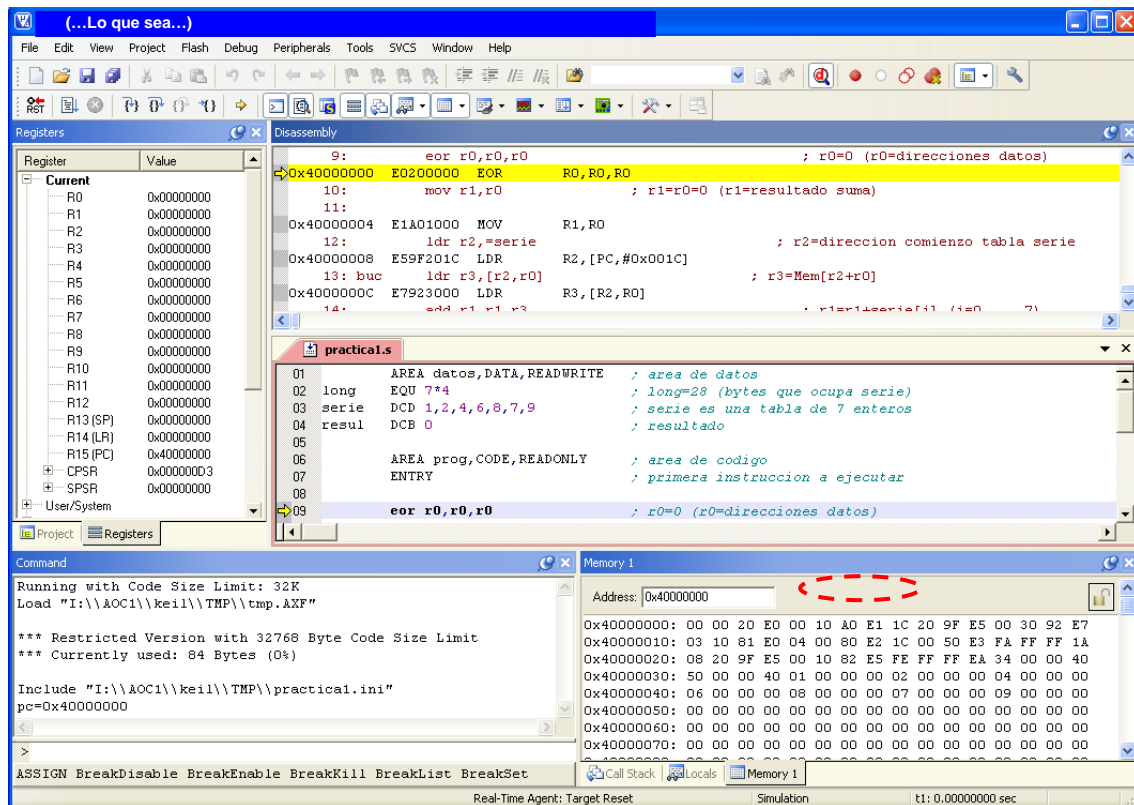
- 11) Una vez configurado el linker, tendremos que indicarle al depurador a partir de qué dirección de memoria queremos que comience la ejecución del código. Para ello vamos ahora a la pestaña debug y, de nuevo, pulsamos en el botón que aparece rodeado en la figura siguiente:




- 12) Seleccionamos el fichero practica1.ini y pulsamos en Abrir. A continuación, pulsamos en OK en el diálogo *Options for Target 'Target 1'*.



- 13) Una vez todo configurado y puesto a punto, pasamos a ensamblar y enlazar nuestro programa ensamblador por medio del botón () o tecla F7 (build). Si todo ha ido bien, la salida del compilador nos dirá "0 Error(s), 0 Warning(s)", y estaremos en disposición de simular la ejecución de nuestro código.
- 14) Entramos en modo depuración () o teclas Ctrl+F5 (start/stop debug session). Nos avisará de que estamos en modo evaluación: pulsamos sobre Aceptar.
- 15) Asegúrate de tener abierta la ventana de código desensamblado, la ventana de registros (y de ajustar su anchura para que se vea el valor de los mismos) y de tener también abierta una ventana de memoria (comenzando en la dirección 0x40000000). El aspecto de la pantalla debería aproximarse al siguiente:



- 16) Ejecuta paso a paso () Step, F11) y observa como van cambiando el valor de los registros. Describe cuáles lo hacen y por qué.
- 17) Observa el valor inicial del registro 15 (PC) y justifica su valor. ¿Qué sucede si lo cambias en la ventana de registros?
- 18) Busca en la ventana de memoria la variable `resul` y verifica si cambia su valor desde el comienzo al final de la ejecución. En la ventana de memoria puedes cambiar el tamaño de la posición observada (bytes, medias palabras, palabras) y la representación de la información almacenada (enteros, decimales, hexadecimales, con o sin signo, etc.) pulsando con el botón derecho del ratón en la zona marcada en rojo.
- 19) Determina si todas las instrucciones de lenguaje máquina son del mismo tamaño.
- 20) ¿Podrías realizar este programa utilizando menos registros? Si es así, desconéctate del depurador. Modifica el código fuente y verifica que el resultado del programa no ha cambiado.