 ISBN: 978-99920-70-10-9

ARMSim y QtARMSim: simulador de ARM para docencia

Sergio Barrachina Mir, Germán Fabregat Llueca,
Juan Carlos Fernández Fernández, Germán León Navarro
Dpto. de Ingeniería y Ciencia de los Computadores
Universitat Jaume I
Castellón
{barrachi,fabregat,jfernand,leon}@uji.es

Resumen

Muchos de los objetivos formativos de las asignaturas de introducción a la Arquitectura de Computadores se centran en aquellos aspectos que conforman la visión que un programador en lenguaje ensamblador tiene de un computador. Por regla general, para definir dichos objetivos se suele utilizar una arquitectura de computadores concreta, que normalmente se selecciona con el doble criterio de que sea lo más sencilla posible y, a la vez, motive al estudiantado.
La arquitectura ARM es una candidata idónea como vehículo conductor en la docencia de arquitectura de computadores. Por un lado, al estar basada en la arquitectura RISC (*Reduced Instruction Set Computer*), es relativamente sencilla. Por otro, se trata de una arquitectura actual y ampliamente difundida (especialmente en dispositivos móviles, *smartphones* y *tablets*), lo que motiva al estudiantado.
Para poder realizar prácticas sobre ARM es conveniente disponer de un simulador o de una herramienta de desarrollo sobre una máquina ARM. Puesto que dicha materia se explica en los primeros cursos, conviene que la aplicación seleccionada sea sencilla de utilizar y lo suficientemente flexible. Por otro lado, conviene que sea libre, para poder adaptarla en caso necesario, y también multiplataforma y gratuita, para facilitar que el estudiante que lo desee pueda instalarla en su propio equipo. Tras evaluar distintas opciones, finalmente optamos por desarrollar y liberar nuestro propio simulador de ARM, ARMSim, y una interfaz gráfica para dicho simulador, QtARMSim.
El simulador ARMSim y su interfaz, QtARMSim, han sido utilizados durante el primer semestre del curso 2014/15. Las críticas recibidas, tanto por los estudiantes como por los profesores de laboratorio, han sido muy positivas.

Abstract

Most of the learning objectives of introductory courses to Computer Architecture focus on those aspects that are related to the vision an assembly language programmer has of a computer. Generally, a particular computer architecture is used to define these objectives, which is usually selected with the double objective of being as simple as possible and to motivate the students.
The ARM architecture is an ideal candidate as a vehicle architecture in teaching computer architecture. On the one hand, as it is based on RISC, it is relatively simple architecture. On the other, it is a widespread and current architecture (especially on mobile devices, smartphones and tablets), which motivates the students.
To perform practices on ARM, it is convenient to have access either to a simulator or a development tool on an ARM machine. Since this matter is taught on the first years, the selected application should be simple to use and flexible enough. On the other hand, it ought to be free, as in freedom, to be able to adapt it if necessary, multiplatform and free, as in free beer, to facilitate that any student could install it on his own computer. After evaluating different alternatives, we finally opted for developing and releasing our own ARM simulator, ARMSim, and a GUI for this simulator, QtARMSim.
The ARMSim simulator and its interface, QtARMSim, have been used during the first semester of 2014/15. The feedback received by students and laboratory teachers has been very positive.

Palabras clave

Arquitectura de Computadores, ARM, Simulador, Ensamblador.

Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9

1. Motivación

El contenido de los cursos de Arquitectura de Computadores históricamente ha seguido el frenético ritmo en el que los avances tecnológicos y arquitectónicos en el diseño de grandes computadores, primero, y en el diseño de los microprocesadores, a partir de los 80, se han ido sucediendo. Así pues, se puede decir que la docencia en Arquitectura de Computadores ha pasado por seis grandes eras: *mainframes*, *minicomputadores*, primeros microprocesadores, microprocesadores, RISC y post RISC. Cada vez que las universidades han podido acceder a hardware específico a un coste razonable, este ha sido utilizado ampliamente como material de referencia. Algunas de las máquinas y microprocesadores que han disfrutado de una mayor popularidad en la docencia de Arquitectura de Computadores han sido: el PDP-11, el 68000 de Motorola, el 80x86 de Intel, el MIPS y el SPARC. Aunque en ocasiones también se ha recurrido a computadores hipotéticos dedicados en exclusiva a la enseñanza de los conceptos arquitectónicos [5].

Las asignaturas relacionadas con la Arquitectura de Computadores de la Universitat Jaume I también han pasado por varias de las opciones indicadas: empezamos con el 68000 de Motorola, seguimos con un computador hipotético y, hasta el curso 2013/14, con la arquitectura MIPS. A principios de 2013 nos planteamos cambiar de la arquitectura MIPS a la ARM.

La arquitectura ARM presenta muchas características que la distinguen de otras arquitecturas contemporáneas y, por otro lado, al estar basada en RISC, es relativamente sencilla [3]. Además, el hecho de que ARM sea una arquitectura actual y ampliamente difundida, especialmente en dispositivos móviles, *smartphones* y *tablets*, es un factor especialmente motivador para el estudiantado [4].

Una vez tomada la decisión de realizar dicho cambio, comenzó un proceso de redefinir las guías docentes y los materiales utilizados en la enseñanza tanto teórica como práctica de las distintas asignaturas relacionadas con la materia de Arquitectura de Computadores.

En concreto, en la asignatura *Estructura de Computadores*, de primer curso, primer semestre, una de las primeras metas fue la de seleccionar un simulador o entorno de ARM que pudiera utilizarse en las prácticas de la asignatura. Debiendo tener en cuenta, además, que otras asignaturas iban a utilizar la plataforma Arduino Due [2], dotada de un microprocesador que implementa la arquitectura ARM Thumb 2.

Hasta ese momento, en las prácticas de dicha asignatura [1] se utilizaba el simulador `QtSpim`[1], antes llamado `xspim` en su versión para GNU/Linux. El simulador `xspim` nos parecía un entorno adecuado para las prácticas de la asignatura debido a que: I) permitía simular la ejecución de programas en ensamblador y visualizar el contenido de los registros y la memoria de forma sencilla, y II) era software libre, gratuito y multiplataforma.

La primera de las opciones que consideramos para sustituir a `xspim` por un entorno para ARM, fue el entorno integrado de desarrollo y simulación μVision de la empresa Keil[2]. Dicho entorno es uno de los más utilizados en la programación de microcontroladores basados en ARM (la empresa Keil Software fue adquirida en 2005 por ARM) y se proporciona como ejemplo en libros sobre ARM como [6] y [7]. Incluye compiladores de C y C++, ensamblador, enlazador, depurador y simuladores de varios microcontroladores basados en ARM.

Consideramos que adoptar μVision hubiera proporcionado un entorno realista de programación de sistemas empotrados. Sin embargo, su manejo hubiera sido ligeramente más complejo de lo que queríamos para una asignatura de primero en la que se pretende explicar los fundamentos de la arquitectura más que la programación de sistemas empotrados. Por otro lado, solo hay versión para Windows, no es software libre y es gratuito con limitaciones (32 KiB de código —lo cual no hubiera presentado ningún problema– y requiere registrarse para su descarga —lo cual habría obligado a los estudiantes a dar sus datos personales para poder disponer de él—).

Relacionado con dicho IDE, también evaluamos la adquisición del producto *Lab-in-a-Box* de ARM University, que incluye[3]: I) licencias de Keil *MDK-ARM Professional Software Tool* (del que forma parte el IDE μVision); II) tarjetas Freescale basadas en el procesador ARM Cortex-M0+; y III) material didáctico, de formación y de prácticas. Una vez constatamos que el curso se centraba exclusivamente en la programación en C de sistemas empotrados, tuvimos que descartarlo para esta asignatura.

La siguiente alternativa que consideramos fue la de utilizar Eclipse DS-5[4]. Así como μVision está orientado a los sistemas empotrados, Eclipse DS-5 es el entorno de desarrollo que proporciona ARM para procesadores ARM y dispositivos *System-on-Chip*. Permite programar en C/C++ y en ensamblador, así como depurar el código que se esté ejecutando en el computador para el que se desarrolla.

Aunque la versión habitual de Eclipse DS-5 no es ni libre ni gratuita, existe una versión llamada *Community Edition* que sí que lo es. Además, al tratarse en realidad de un *plugin* para Eclipse (desarrollado en java),

[1] `http://spimsimulator.sourceforge.net/`

[2] `http://www.keil.com/product/brochures/uv4.pdf`

[3] `http://arm.com/support/university/educators/embedded/`

[4] `http://ds.arm.com/`

Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9

se trata de una solución multiplataforma.

Sin embargo, una de las primeras pegas con las que nos encontramos es que no proporciona un entorno de simulación propiamente dicho. No obstante, es posible utilizar las opciones de depuración para emular un entorno de simulación. Basta con crear una máquina virtual ARM (p.e., con QEMU), instalar un sistema operativo GNU/Linux para ARM sobre dicha máquina virtual y configurar dicha máquina y Eclipse DS-5 para comunicarse entre sí por medio de SSH y el servidor de GDB. De esta forma, las capacidades de depuración de Eclipse servirían para observar la ejecución simulada de un programa sin necesidad de recurrir a hardware específico. Por otro lado, también es posible seguir el mismo procedimiento, pero utilizando hardware específico, como podría ser una Raspberry Pi.

Aunque conseguimos depurar código en ambos casos, contra una máquina virtual y contra una Raspberry Pi, consideramos que la complejidad de todo el montaje era excesiva para lo que pretendíamos que fueran unas prácticas de primer curso.

Por último, también evaluamos el simulador ARMSim#[5]. Dicha aplicación permite simular la ejecución de programas en ensamblador en un sistema basado en el procesador ARM7. Está escrito en C#, funciona en Windows y, en teoría, sobre Mac OSX y GNU/Linux utilizando la implementación Mono del entorno de trabajo .NET. Este simulador se acercaba más a lo que estábamos buscando para la asignatura y presenta características interesantes como la simulación y visualización de periféricos, e incluso la posibilidad de desarrollar *plugins* para extenderlo con nuevos dispositivos de E/S. Sin embargo, en la actualidad no soporta el modo Thumb de ARM, que era necesario utilizar para enlazar las prácticas basadas en el simulador con otras posteriores en la misma y en otras asignaturas de la titulación en las que el estudiante iba a programar tarjetas Arduino Due.

Así que, puesto que ninguno de los sistemas evaluados cubrían todas nuestras necesidades, optamos por desarrollar nuestro propio simulador de ARM[6] con la intención de que también pudiera ser utilizado en cursos introductorios a la Arquitectura de Computadores de otras universidades.

Además, para la realización de las prácticas se ha escrito un libro[7] en el que se describe con más detalle el simulador y se proponen una serie de prácticas de introducción a la arquitectura de computadores utilizando la arquitectura ARM Thumb.

El resto de este artículo está organizado como sigue. En el primer apartado se describen los objetivos del simulador que se quería desarrollar. El Apartado 3 describe las características del motor de simulación desarrollado. El Apartado 4, la interfaz gráfica del simulador y su utilización. El Apartado 5, los resultados obtenidos y, por último, en el Apartado 6, las conclusiones y trabajo futuro.

[5] `http://armsim.cs.uvic.ca/`

[6] El simulador puede descargarse de:
`http://lorca.act.uji.es/projects/qtarmsim`

[7] El libro «Introducción a la Arquitectura de Computadores con QtARMSim y Arduino» se puede consultar en:
`http://lorca.act.uji.es/book/practARM/`

2. Objetivos

El objetivo que tuvimos en mente cuando comenzamos a desarrollar el simulador fue el de proporcionar a los estudiantes un simulador de ARM Thumb que les permitiera alcanzar fácilmente los objetivos formativos de arquitectura de computadores relacionados con el funcionamiento del procesador.

Para cumplir con dicho objetivo docente consideramos que el simulador debería ser lo más sencillo posible, gratuito, multiplataforma y libre, que contemplara todo el proceso de desarrollo y simulación de código en ensamblador y, por último, en el que el motor de simulación estuviera desacoplado de su interfaz gráfica.

Debía ser lo más sencillo posible para que los estudiantes pudieran utilizarlo en las sesiones de prácticas sin mayores problemas, centrándose en el desarrollo y simulación de programas en ensamblador, más que en el manejo y configuración del entorno.

También queríamos que fuera gratuito y multiplataforma para que el estudiante tuviera la oportunidad de instalarlo gratuitamente en su propio computador, independientemente del sistema operativo que utilizase.

El objetivo al hacerlo software libre fue el de incentivar al estudiantado a que pudieran inspeccionar el código, tanto del motor, como de la interfaz gráfica. Al ser software libre, también se permite que terceros puedan involucrarse en su desarrollo o adaptarlo a sus necesidades.

Por otro lado, el simulador debía contemplar todo el proceso tanto de desarrollo como de simulación del código con la intención de que dicho proceso fuera lo más ágil y sencillo posible. De esta forma, pretendíamos evitar que el estudiante se viera obligado a alternar entre un editor externo —para escribir el código en ensamblador— y el simulador —para las fases de compilación y simulación del código—. Así pues, debería proporcionar todos los aspectos necesarios para el desarrollo y simulación de código en ensamblador: desde la edición del código, pasando por su compilación, hasta su ejecución, tanto completa como paso a paso.

El último de los objetivos fue el de que el motor de simulación y la interfaz estuvieran desacoplados. Esta decisión permitirá en el futuro que el simulador pueda ser utilizado de forma remota o desde distintas interfaces gráficas. Así pues, el simulador consta en realidad

Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9

de dos aplicaciones: ARMSim, el programa que simula un ARM, y QtARMSim, una interfaz gráfica para dicho motor de simulación.

3. ARMSim

ARMSim es un motor de simulación de la arquitectura ARM Thumb. Proporciona un modelo de procesador —que incluye registros e indicadores de estado—, y un modelo de memoria —tanto RAM como ROM—. El modelo de procesador es capaz de decodificar y ejecutar el juego completo de instrucciones de la arquitectura ARM Thumb. El modelo de memoria permite definir la cantidad de cada tipo de memoria disponible, así como inicializar su contenido y leer o escribir de ella.

Aparte de modelar los componentes anteriormente citados, el simulador permite indicar un fichero fuente para su ensamblado. En lugar de implementar un ensamblador propio, se ha optado por una solución más versátil: ARMSim ejecuta un ensamblador de ARM (p.e., GCC de GNU) e interpreta el código objeto generado por dicho ensamblador para inicializar la ROM y la RAM a partir de dicha información. Además, también extrae la información sobre qué línea de código fuente ha generado cada instrucción máquina.

ARMSim no proporciona una interfaz gráfica, en lugar de eso, se pone a la escucha en el puerto indicado en la línea de comandos. Así pues, para interactuar con ARMSim se debe establecer una conexión con dicho puerto (p.e., con `telnet` o `putty`) e indicar por medio de comandos de texto, qué acciones se quieren llevar a cabo.

Los comandos aceptados por ARMSim permiten: I) ensamblar un fichero; II) consultar y modificar el contenido de registros y memoria; III) desensamblar posiciones de memoria; IV) definir y consultar puntos de ruptura; V) ejecutar el código a partir de una posición dada hasta que se acabe el programa o se encuentre un punto de ruptura; y VI) ejecutar paso a paso el programa (entrando o no en las subrutinas).

La Figura 1 representa de forma esquemática la interrelación entre el motor de simulación, ARMSim, la interfaz gráfica, QtARMSim, y el compilador GCC de GNU.

4. QtARMSim

Como ya se ha comentado, QtARMSim es una interfaz gráfica para el simulador ARMSim.

La Figura 2 muestra la ventana principal de QtARMSim en la que se está editando un pequeño fragmento de código. La parte central de dicha ventana corresponde al editor de código fuente en ensamblador. Alrededor de dicha parte central se distribuyen por defecto una serie de paneles. A la izquierda del editor se encuentra el panel de registros; a su derecha, el panel de memoria; y debajo, el panel de mensajes. Los paneles de registros y memoria están inicialmente desactivados. En el panel de mensajes se irán mostrando mensajes relacionados con lo que se vaya haciendo: si el código se ha ensamblado correctamente, si ha habido errores de sintaxis, qué línea se acaba de ejecutar, etc.

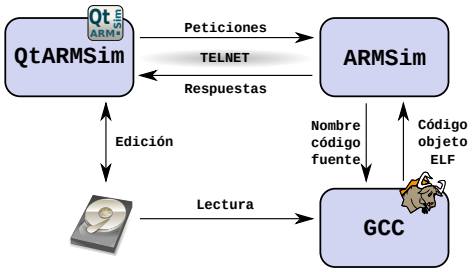

Figura 1: Interrelación entre los componentes del simulador (*Dibujo del disco duro de Andrew Fitzsimon*)

Si se acaba de instalar QtARMSim, podría ser necesario configurarlo para indicar cómo llamar al simulador ARMSim o para indicar dónde está instalado el compilador cruzado de GCC para ARM. Aunque generalmente no será necesario, ya que en la primera ejecución, QtARMSim intenta obtener de forma automática todos los parámetros necesarios.

Por otro lado, QtARMSim se ha diseñado para editar un programa y simular su ejecución. Partiendo de esta premisa, el simulador distingue entre dos modos de funcionamiento: el modo de edición y el modo de simulación. Dichos modos se describen en los siguientes dos apartados.

4.1. Modo de edición

En el modo de edición, como ya se ha comentado, la parte central de la ventana es un editor de código fuente en ensamblador, que permite escribir el programa en ensamblador que se quiere simular posteriormente. La Figura 2 muestra la ventana de QtARMSim en la que se ha introducido un programa en ensamblador. Como se puede observar, el editor reconoce el lenguaje ensamblador ARM Thumb y colorea convenientemente el código fuente. Poco más hay que decir de este modo, simplemente que se proporcionan las características que se esperan habitualmente en un editor avanzado de código fuente.

4.2. Modo de simulación

Una vez se ha escrito un programa en ensamblador, los siguientes pasos son ensamblar dicho código y si-

Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9

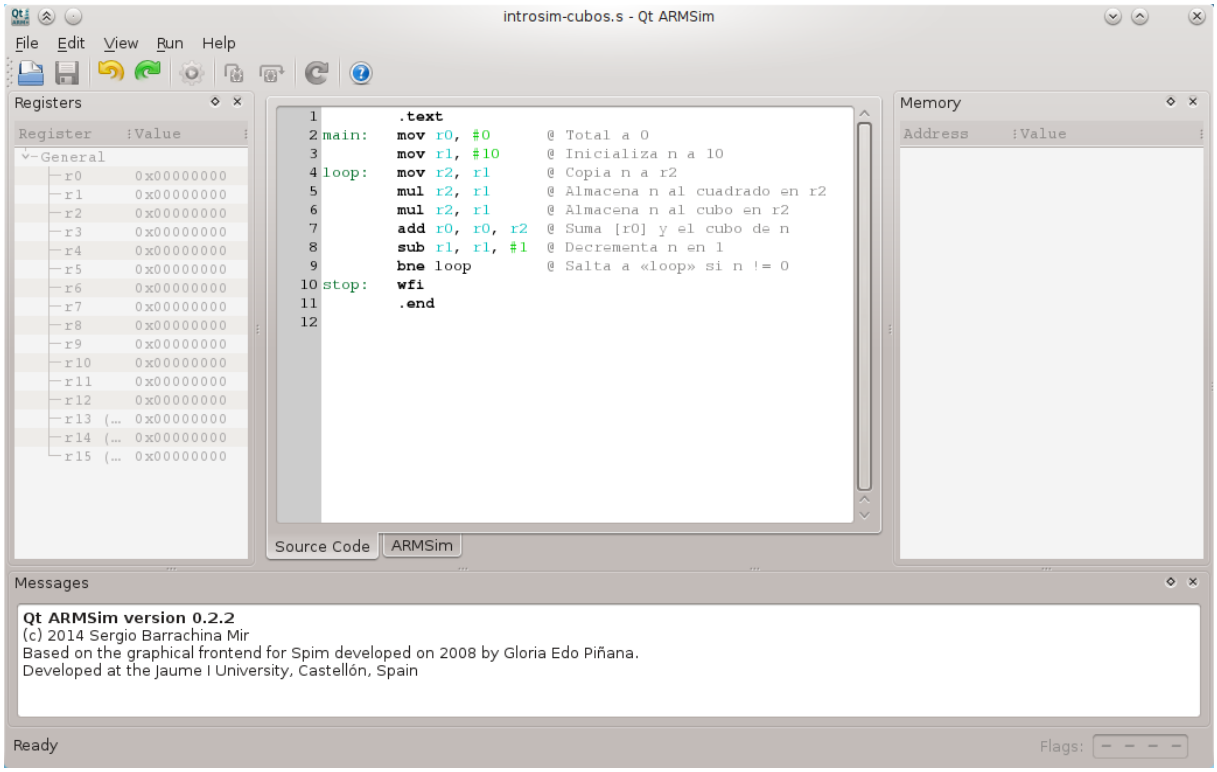

Figura 2: QtARMSim editando el código «introsim-cubos.s»

mular su ejecución.

Para ensamblar el código y pasar al modo de simulación, basta con pulsar sobre la pestaña «ARMSim», que se encuentra debajo de la sección central de la ventana principal. La Figura 3 muestra la apariencia de QtARMSim cuando está en el modo de simulación.

Cuando se pasa al modo de simulación, la interfaz gráfica se conecta con el simulador ARMSim, quien se encarga de realizar las siguientes acciones: I) llamar al ensamblador cruzado para ensamblar el código fuente; II) actualizar el contenido de la memoria ROM con las instrucciones máquina generadas por el ensamblador; III) inicializar, si es el caso, el contenido de la memoria RAM con los datos indicados en el código fuente; y, por último, IV) inicializar los registros del computador simulado.

Si se produjera algún error al intentar pasar al modo de simulación, se mostrará un cuadro de diálogo informando del error, se volverá automáticamente al modo de edición y en el panel de mensajes se mostrarán las causas del error. En caso contrario, se completa el cambio al modo de simulación.

En el modo de simulación, cada línea de la ventana central muestra la información correspondiente a una instrucción máquina. Para una de ellas, se tiene (de izquierda a derecha):

1. La dirección de memoria en la que está almacenada la instrucción máquina.
2. La instrucción máquina expresada en hexadecimal.
3. La instrucción máquina expresada en ensamblador.
4. La línea original en ensamblador que ha dado lugar a la instrucción máquina.

Así por ejemplo, la información mostrada en la primera línea de la ventana de desensamblado de la Figura 3 indica que:

- La instrucción máquina está almacenada en la dirección de memoria `0x0000 1000`.
- La instrucción máquina expresada en hexadecimal es `0x2000`.
- La instrucción máquina expresada en ensamblador es «**`movs`** `r0, #0`».
- La instrucción se ha generado a partir de la línea número 2 del código fuente original, que es: «`main:` **`mov`** `r0, #0 @ Total a 0`»

Por otro lado, el contenido de la memoria del computador simulado se muestra en el panel de memoria, tal y como se puede ver en la parte derecha de la Figura 3. En este ejemplo se puede observar que el computador dispone de dos bloques de memoria: un bloque de memoria ROM que comienza en la dirección `0x0000 1000` y un bloque de memoria RAM que comienza en la dirección `0x2007 0000`. También

Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9

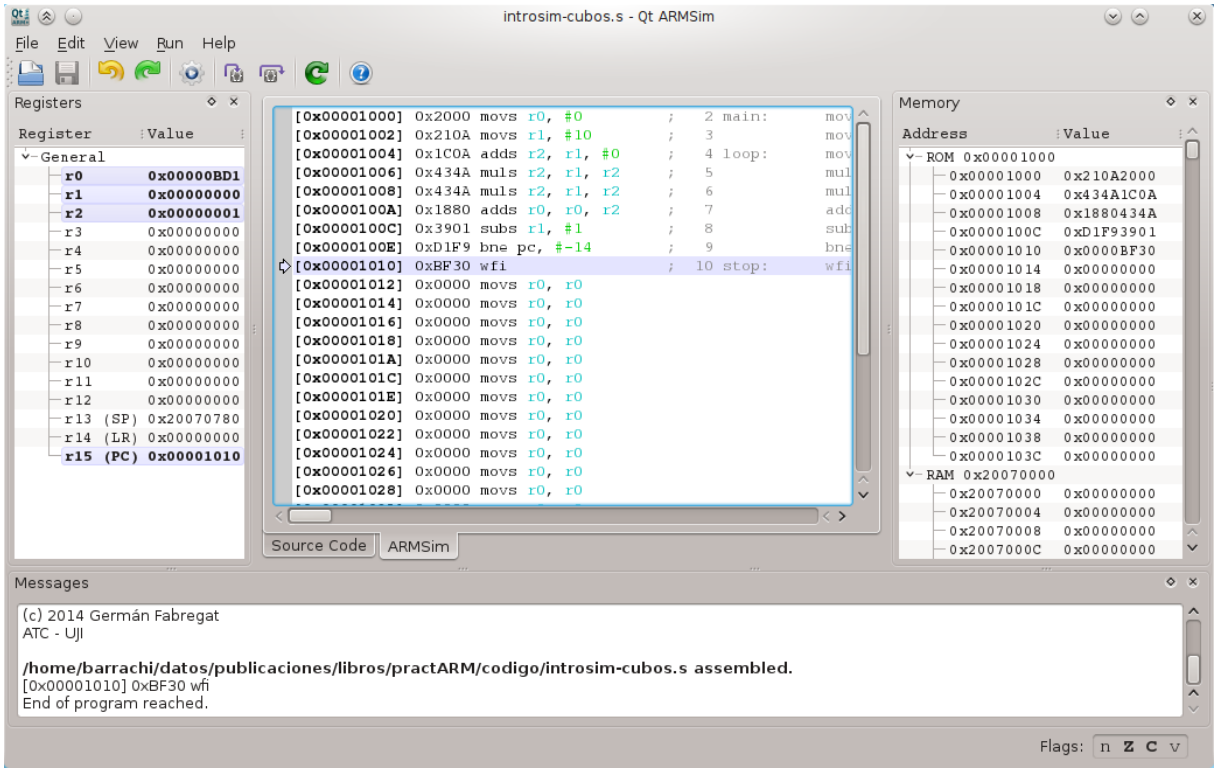

Figura 3: QtARMSim en el modo de simulación después de ejecutar el código máquina

se puede ver cómo las celdas de la memoria ROM contienen algunos valores distintos de cero (que corresponden a las instrucciones máquina del programa ensamblado) y las celdas de la memoria RAM están todas a cero.

De forma similar, el contenido de los registros del «`r0`» al «`r15`» se muestra en el panel de registros (a la izquierda en la Figura 3). El registro «`r15`» merece una mención especial, ya que se trata del contador de programa (PC). Para facilitar el uso del simulador, éste resalta la línea correspondiente a la dirección indicada por el PC, de tal forma que el usuario puede identificar fácilmente qué línea está a punto de ser ejecutada.

En cuanto al manejo de la interfaz, puesto que los paneles del simulador son empotrables, es posible cerrarlos de manera individual, reubicarlos en una posición distinta, o desacoplarlos y mostrarlos como ventanas flotantes. No obstante, y pensando en su uso en el aula, se ha incorporado al simulador la funcionalidad de restaurar la disposición por defecto (basta con pulsar la tecla «`F3`» o su correspondiente entrada en el menú). Naturalmente, también es posible volver a mostrar alguno o todos los paneles que han sido cerrados previamente sin necesidad de restaurar la disposición por defecto.

Desde el modo de simulación, QtARMSim permite ejecutar el programa de forma completa, ejecutarlo paso a paso, modificar manualmente el contenido de registros y memoria, y establecer puntos de ruptura. En los siguientes subapartados se describen dichas funcionalidades.

4.2.1. Ejecución del programa completo

La opción más sencilla de simulación es la de ejecutar el programa completo. La Figura 3 muestra la ventana de QtARMSim después de ejecutar el código máquina generado al ensamblar el fichero «introsim-cubos.s». Para que los cambios realizados tanto en registros como en posiciones de memoria sean fácilmente identificables, el simulador resalta aquellos registros y posiciones de memoria que se han modificado tras la última ejecución. Así, se puede observar en la Figura 3 que los registros `r0`, `r1`, `r2` y `r15` tienen ahora fondo azul y están en negrita, ya que son los únicos que se han modificado al ejecutar dicho programa.

4.2.2. Ejecución paso a paso

Aunque la ejecución completa de un programa pueda servir para comprobar si el programa hace lo que se espera de él, no permite ver con detalle cómo se ejecuta el programa. Tan solo se puede observar el estado inicial del computador simulado y el estado al que se llega cuando se termina la ejecución del programa.

Para poder ver qué es lo que ocurre al ejecutar ca-

Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9

da instrucción, el simulador proporciona la opción de ejecutar paso a paso. Esta opción suele utilizarse para ver por qué un determinado programa o una de sus partes no está haciendo lo que se espera de ella. Pero también suele utilizarse para evaluar cómo afectaría al programa el que en un momento dado se modificara el contenido de determinados registros o posiciones de memoria.

Si por ejemplo se quisiera evaluar cómo afectaría a la ejecución del resto del programa el que el registro `r1` tuviera un valor distinto al actual, habría que modificar el contenido de dicho registro antes de proseguir con la ejecución del programa.

El simulador permite modificar el contenido de un registro simplemente haciendo doble clic sobre la celda en la que está su contenido actual, teclear el nuevo número y pulsar la tecla «`Retorno`». Además, el nuevo valor numérico[8] puede introducirse en decimal, en hexadecimal (si se precede de «0x», p.e., «0x3»), o en binario (si se precede de «0b», p.e., «0b11»).

Una vez modificado el contenido del registro `r1` para que contenga el valor 3, se podría ejecutar el resto del código de golpe, no haría falta ir paso a paso.

Otra característica del simulador es que proporciona dos modalidades de ejecución paso a paso, la primera de ellas ejecuta siempre una única instrucción, pasando el PC a apuntar a la siguiente instrucción. La segunda modalidad tiene en cuenta el hecho de que los programas suelen estructurarse por medio de rutinas. Si el código en ensamblador incluye llamadas a rutinas, al utilizar el primer modo de ejecución paso a paso sobre una instrucción de llamada a una rutina, la siguiente instrucción que se ejecutará será la primera instrucción de dicha rutina.

Sin embargo, en ocasiones no interesa tener que ejecutar paso a paso todo el contenido de una determinada rutina, puede ser preferible ejecutar la rutina entera como si de una única instrucción se tratara, y que una vez ejecutada la rutina, el PC pase a apuntar directamente a la siguiente instrucción a la de la llamada a la rutina. De esta forma, sería fácil para el estudiante ver y comparar el estado del computador simulado antes de llamar a la rutina y justo después de volver de ella. Para poder hacer lo anterior, QtARMSim también proporciona la opción de ejecución paso a paso llamada «por encima».

4.2.3. Puntos de ruptura

La ejecución paso a paso permite ver con detenimiento qué es lo que está ocurriendo en una determinada parte del código. Sin embargo, puede que para llegar a la zona del código que se quiere inspeccionar con detenimiento haya que ejecutar muchas instrucciones. Por ejemplo, podríamos estar interesados en una parte del código al que se llega después de completar un bucle con cientos de iteraciones. No tendría sentido tener que ir paso a paso hasta conseguir salir del bucle y llegar a la parte del código que en realidad queremos ver con más detenimiento.

Por tanto, es conveniente disponer de una forma de indicarle al simulador que ejecute las partes del código que no nos interesa ver con detenimiento y que solo se detenga cuando llegue a aquella instrucción a partir de la cual queremos realizar una ejecución paso a paso (o en la que queremos poder observar el estado del simulador).

Un punto de ruptura (*breakpoint* en inglés) sirve justamente para eso, para indicarle al simulador que tiene que parar la ejecución cuando se alcance la instrucción en la que se haya definido un punto de ruptura.

QtARMSim permite definir y desmarcar puntos de ruptura. Para definir un punto de ruptura, basta con hacer clic sobre el margen izquierdo de la ventana de desensamblado, en la línea en la que se quiere definir el punto de ruptura. Al hacerlo, aparecerá un círculo rojo en el margen, que indica justamente que en esa línea se ha definido un punto de ruptura.

Para desmarcar un punto de ruptura ya definido, se debe proceder de la misma forma, haciendo clic sobre la marca del punto de ruptura.

5. Resultados

Se ha desarrollado un simulador de ARM que cumple con los objetivos descritos en el Apartado 2 y que puede descargarse gratuitamente. También se ha redactado y publicado bajo licencia *Creative Commons* el material de prácticas utilizado durante el curso 2014/15. En dicho material se describe el simulador y se proponen ejercicios de introducción a la Arquitectura de Computadores que lo utilizan.

Como este curso ha sido el primero en el que se ha utilizado dicho simulador, hemos preguntado constantemente a los estudiantes por su opinión sobre el funcionamiento del simulador para que nos dijeran qué aspectos consideraban que deberían mejorarse. La respuesta ha sido en general muy positiva y, gracias a las sugerencias recibidas, hemos tenido la oportunidad de ir mejorando el simulador durante el curso. La única crítica que aún tenemos pendiente, es la de mejorar el proceso de instalación, ya que dependiendo del sistema operativo puede ser más o menos engorroso.

Conviene destacar que la respuesta de los estudiantes repetidores ha sido especialmente gratificante, ya que pese a que el curso pasado hicieron las prácticas

[8] También es posible introducir cadenas de como mucho 4 caracteres. En este caso deberán estar entre comillas simples o dobles, p.e., `"Hola"`. Al convertir los caracteres de la cadena introducida a números, se utiliza la codificación UTF-8 y para ordenar los bytes resultantes dentro del registro, se sigue el convenio *Little-Endian*.

Andorra La Vella, del 8 al 10 de julio 2015
ISBN: 978-99920-70-10-9

con MIPS y este curso cambiaron a ARM, su percepción de la facilidad de uso de la nueva herramienta les ha llevado a comentarnos que el lenguaje ensamblador de ARM es mucho más sencillo que el de MIPS (cuando desde nuestro punto de vista no es así).

Además de lo anterior, una vez finalizado el curso, aunque más tarde de lo que deberíamos haberlo hecho, realizamos una encuesta con 8 preguntas de escala Likert de 1 a 5, siendo 1 «totalmente en desacuerdo» y 5 «totalmente de acuerdo», y 2 de respuesta abierta.

Contestaron a dicha encuesta 28 estudiantes (un $19\,\%$ de los que siguieron la asignatura).

Las cuatro primeras preguntas se correspondían con aspectos técnicos del simulador (instalación, interfaz, funcionalidad de depuración e información en paneles). En todas las preguntas se obtuvo una mediana de 4, salvo en la referida a la instalación, en que la mediana fue de 3.

Las siguientes dos preguntas, más relacionadas con el objetivo docente de la aplicación y la satisfacción del estudiante, «Considero que QtARMSim me ha servido para entender mejor cómo funciona el procesador ARM» y «Recomendaría que se siguiera utilizando QtARMSim en la docencia de arquitectura de computadores» obtuvieron una mediana de 5.

Las respuestas dadas a las dos últimas preguntas, de respuesta abierta, «¿Qué aspectos positivos destacarías de QtARMSim?» y «¿Qué crees que convendría mejorar de QtARMSim?» incidieron por un lado en la facilidad de uso y en la utilidad del simulador para facilitar su aprendizaje, y por otro, en que convendría simplificar el procedimiento de instalación.

En cuanto al rendimiento académico, comparado con respecto al curso anterior, se ha aumentado el porcentaje de estudiantes que han seguido la asignatura del $83\,\%$ al $88\,\%$, y el número de aprobados del $49\,\%$ al $56\,\%$. Si bien es cierto que no es posible concluir que las mejoras en los resultados estén directamente relacionadas con la utilización del simulador, ya que hay una gran variedad de factores que podrían haber motivado dichas mejoras.

Por último, la valoración de los profesores de laboratorio también ha sido muy positiva y su impresión ha sido la de que los estudiantes se daban más cuenta de lo que estaba ocurriendo conforme simulaban sus ejercicios, en comparación con lo observado en cursos anteriores en que tenían menos claro qué ocurría.

6. Conclusiones y trabajo futuro

En este artículo se han presentado las aplicaciones ARMSim y QtARMSim que proporcionan un entorno didáctico de simulación de la arquitectura ARM Thumb multiplataforma, libre, gratuito y fácil de utilizar. Dichas aplicaciones se han desarrollado con el objetivo de facilitar la comprensión del funcionamiento de un procesador, y se ha constatado que tanto los estudiantes como los profesores perciben que se ha cumplido con dicho objetivo.

Como trabajo futuro se contempla terminar de perfilar la interfaz gráfica, añadir soporte completo para la arquitectura Thumb 2, permitir la compilación de código en C, utilizar ARMSim de forma remota para la evaluación automatizada de ejercicios, añadir dispositivos de entrada/salida e incorporar una animación gráfica de la ejecución de cada instrucción.

Referencias

[1] Sergio Barrachina Mir, Maribel Castillo Catalán, Jose M. Claver Iborra y Juan C. Fernández Fernández. Prácticas de introducción a la arquitectura de computadores con el simulador SPIM. Pearson Educación, 2013.

[2] Sergio Barrachina Mir, Germán Fabregat Llueca y José Vicente Martí Avilés. Utilizando Arduino DUE en la docencia de la entrada/salida. En Actas de las XXI Jornadas de Enseñanza Universitaria de la Informática, Andorra, 2015.

[3] Alan Clements. Selecting a processor for teaching computer architecture. En *Microprocessors and Microsystems*, 23(5), 281-290. Elsevier, 1999.

[4] Alan Clements. ARMs for the poor: Selecting a processor for teaching computer architecture. En *Frontiers in Education Conference (FIE)*, pp. T3E 1–6. IEEE, 2000.

[5] Alan Clements. The undergraduate curriculum in computer architecture. En *IEEE Micro*, 20(3), 13–22. IEEE, 2000.

[6] Alan Clements. Computer Organization and Architecture. Themes and Variations. Cengage Learning, 2014.

[7] William Hohl y Christopher Hinds. ARM Assembly Language: Fundamentals and Techniques. Crc Press, 2014.