# SOPORTE SOFTWARE PARA LA GENERACIÓN DE CÓDIGO Y MICROCODIGO ROMABLE PARA UNA COMPUTADORA SENCILLA

J. I. BENAVIDES, F. J. QUILES, C. M. ROMERO Y S. DUEÑAS
*Departamento de Electrotecnia y Electrónica. Escuela Politécnica Superior. Universidad de Córdoba. 14004 Córdoba. España.*

*En este trabajo presentamos la realización de un entorno de programación que nos permitirá de una manera sencilla, ensamblar un fichero fuente de código ensamblador o de microcódigo de una Computadora sencilla, y generar ficheros de salida en diversos formatos, para simular o grabar en una EPROM, tanto los programas como microprogramas.*

## 1. Introducción

Dado que nuestra Área de Conocimiento imparte las asignaturas relativas a Arquitectura y Tecnología de Computadores de las titulaciones de Ingeniería Técnica en Electrónica Industrial, Informática de Sistema, Informática de Gestión e Ingeniero en Automática y Electrónica Industrial, decidimos abrir una línea de trabajo con el objetivo de desarrollar entornos, tanto hardware como software, para ilustrar a los alumnos los conceptos teóricos impartidos en estas asignaturas.

Dentro de esta línea se han desarrollado diversos Trabajos Fin de Carrera, entre los que se incluyen los que se describen en este artículo. El objetivo principal de todos, es mostrar a los alumnos la arquitectura, organización y funcionamiento de una Computadora sencilla.

## 2. Antecedentes

El primer proyecto que se planteó, conforme al objetivo anterior, fue la Implementación de una Arquitectura de una Computadora Sencilla [1]. El objetivo inicial de este proyecto fue la realización física de la Computadora básica [2] con Unidad de Control Microprogramada [3], que se analiza en las clases teóricas y prácticas de las asignaturas relativas a la materia de Arquitectura de Computadores. Aunque su estructura es bastante simple, no se pudo implementar con circuitos integrados de función fija, por lo que hubo que utilizar PLDs. Debido a que los registros se interconectan mediante conexiones dedicadas, se necesita un número elevado de patillas de entrada y salida, por lo que no se aprovechan adecuadamente los términos productos de los PLDs. Por ello, se modificó la estructura, pasando a ser una arquitectura basada en bus. En la figura 1 se muestra su estructura.

El sistema de memoria tiene una organización de 256 x 12, y está compuesto por una EPROM, que almacena hasta 8 programas, y una SRAM, que contiene los datos y la pila. Además de los registros básicos de cualquier computador (PC, OPR y MAR), tiene un

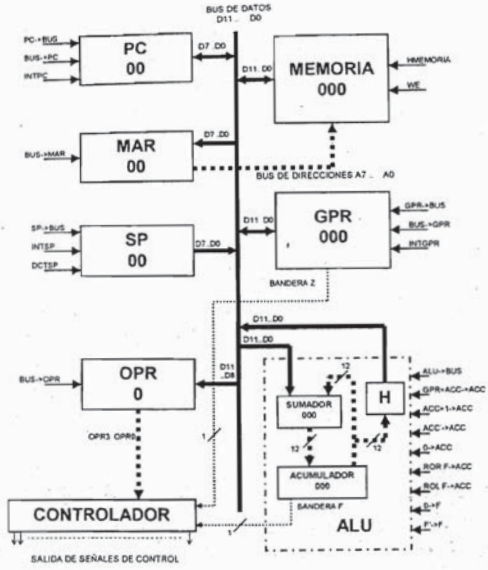


**Figura 1:** Estructura de la Computadora

Puntero de Pila (SP) y un Registro de Propósito General (GPR). La pila es del tipo expand-down y se emplea en las llamadas a subrutina y en la rutina de servicio de interrupción. La ALU está basada en un sumador y un registro Acumulador. El Controlador es del tipo microprogramado. Genera la secuencia de microinstrucciones de los ciclos de búsqueda, ejecución y de servicio de interrupción. La computadora puede ejecutar los programas a nivel de microinstrucción, instrucción o programa. El conjunto de instrucciones se indica en la tabla 1. Como las microinstrucciones se almacenan en unas EPROM, se pueden crear nuevas instrucciones.

El siguiente proyecto que se realizó, consistió en el desarrollo de un Simulador [4] que mejorase el que hasta ese momento se estaba empleando en las clases prácticas sobre la arquitectura y funcionamiento de la computadora analizada en las clases de teoría. El objetivo esencial fue mejorar los inconvenientes del simulador antiguo, entre los que cabe destacar: leer/salvar los programas desde/en disco, visualizar más claramente el flujo de la información resaltando los componentes que cambian su contenido, permitir la ejecución a nivel de ciclo, instrucción y programa, mostrar la secuencia de microoperaciones del ciclo de búsqueda y ejecución de cada instrucción, poder definir el conjunto de instrucciones mediante un fichero de entrada, y ofrecer una ayuda al alumno, tanto sobre la arquitectura de la

| Nemotécnico | Cód. Op. | Descripción |
|---|---|---|
| HLT | 0000 | Detiene la ejecución del programa |
| CRA | 0001 | Pone a 0 el Acc |
| CTA | 0010 | Complementa el contenido del Acc |
| ITA | 0011 | Incrementa el contenido del Acc |
| CRF | 0100 | Pone a 0 el flag F |
| CTF | 0101 | Complementa el flag F |
| SFZ | 0110 | Salta la siguiente instrucción si F=0 |
| ROL | 0111 | Rota el contenido del Acc a la izq. a través de F |
| ROR | 1000 | Rota el contenido del Acc a la der. a través de F |
| ADD dirección | 1001 | Suma el contenido del Acc con el de la posición de memoria y almacena el resultado en Acc |
| ADI dirección | 1010 | Igual que ADD pero con modo de direc. indirecto |
| STA dirección | 1011 | Almacena el cont. del Acc. en la pos. de memoria |
| JMP dirección | 1100 | Salto incondicional a la instrucción dada |
| CSR dirección | 1101 | Llamada a subrutina |
| RTR | 1110 | Retorno de subrutina |
| ISZ dirección | 1111 | Incrementa el contenido de la posición de memoria y salta la siguiente instrucción si Z=0 |

**Tabla 1:** Conjunto de Instrucciones

computadora, como del manejo de simulador. En la figura 2 se muestra la pantalla del simulador, en cuya parte derecha se indica la estructura de la computadora, y en la parte izquierda el contenido de la memoria principal (programa y datos). Como se puede observar, se diferencia de la computadora del proyecto anterior, en que la interconexión no se hace a través de un bus, sino mediante conexiones específicas.

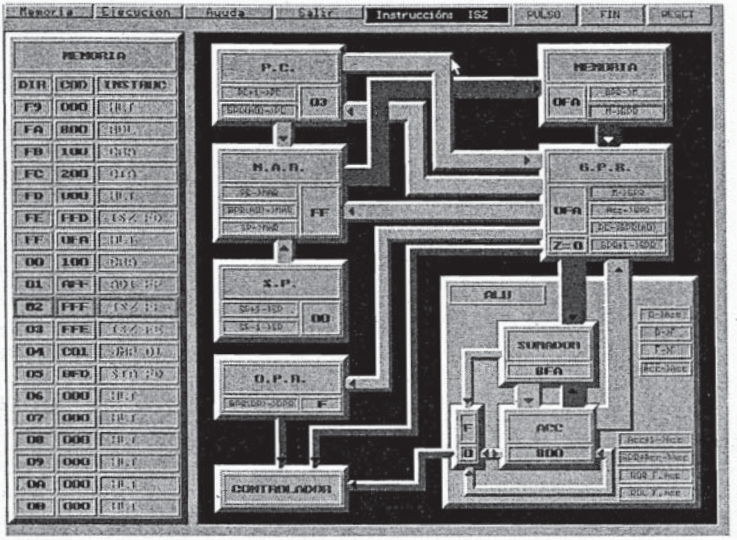


**Figura 2**: Pantalla del Simulador de la Computadora sencilla

## 3. Objetivos y descripción del Soporte Software.

El último proyecto que se ha realizado consiste en desarrollar un Entorno Software [5], que facilite la programación y microprogramación de las computadoras sencillas, analizadas anteriormente. El objetivo principal fue el de desarrollar un Ensamblador y un Microensamblador para generar los programas ejecutables y microprogramas de la Unidad de Control de ambas Computadoras. En la figura 3 se muestra la pantalla de este entorno, en el que se puede observar las distintas utilidades. Mediante el menú archivo se edita el fichero fuente en código ensamblador o microcódigo, ejecutando el editor de texto del MS-DOS (edit). Las instrucciones en ensamblador, admitidas por defecto, son las que se indican en la tabla 1. El simulador emplea un fichero de entrada para cargar el programa y los datos, y en la Computadora física éstos deben cargarse en la EPROM. Por tanto, el menú Ensamblador ofrece estas dos opciones de generación de los ficheros ejecutables, además de mostrar el código generado. Por lo que respecta al Microensamblado, las opciones serán las mismas, ya que la secuencia de microoperaciones es diferente para algunas de las instrucciones en la Computadora física y la simulada, por lo que el microprograma de la ROM será diferente. La definición de nuevas instrucciones se hace mediante un fichero fuente, en el que se indica para cada instrucción, el nemónico, el código de operación, el número de ciclos de reloj, si el

operando está en memoria y la secuencia de microperaciones, expresadas mediante nemónicos. El microensamblador genera el fichero de programación de las EPROM de la computadora física, que contienen el microprograma, en formato HEXA de Intel, y el que define el microprograma al simulador. El comando Simulador ejecuta el programa de simulación, descrito en el apartado anterior.


**Figura 3:** Pantalla del Entorno Software.

## 4. Conclusiones

Como resultado de estos tres Proyectos Fin de Carrera, se dispone de tres herramientas complementarias entre sí, que forman un entorno bastante útil para la enseñanza de la temática de Arquitectura de Computadores. El alumno puede comprobar el funcionamiento de una Computadora en diversos niveles, con lo que le resultará más fácil su comprensión.

Así, primero editará su fichero fuente, y posteriormente mediante el ensamblador creará el fichero del código ejecutable para el simulador. Una vez que ha comprobado, que éste funciona correctamente, generará mediante el ensamblador el fichero de programación de las EPROM, y finalmente, comprobará que éste funciona correctamente en la Computadora física. En ésta podrá analizar la secuencia de microoperaciones, el contenido de los registros y los accesos de lectura y escritura a la memoria, analizando el estado de los leds que los monitorizan. Este proceso, sería igualmente válido para la creación de nuevas instrucciones mediante el microensamblador.

El único inconveniente que tienen este entorno, es que las estructuras de la Computadora física y la Simulada, son diferentes. Esto implica que los alumnos tienen que aprender ambas para poder analizar el funcionamiento, y definir nuevas instrucciones. Por eso, se va a iniciar otro proyecto en el que se desarrollará un Simulador de la Computadora física.

## Referencias


[1] J. García. *Implementación de una Arquitectura de una Computadora Sencilla.* (1998).
[2] H. Taub. *Circuitos Digitales y Microprocesadores.* McGraw-Hill.1983.
[3] M. Morris Mano. *Arquitectura de Computadoras.* Prentice Hall. 1993.
[4] J. Mesa y A. Cobos. *Simulación de la Arquitectura de una Computadora Sencilla con fines docentes.* (1999).
[5] C. M. Romero y S. Dueñas. *Soporte Software para una Computadora Sencilla.* (2000).