DISEÑO DE UN PROCESADOR EDUCATIVO. IMPLEMENTACIÓN CON COMPONENTES DISCRETOS.

Ángel Grediaga[1], Francisco J. Brotons[2], Francisco Ibarra[3]

Antonio Párraga[4].

[1]*Universidad de Alicante*
e-mail: ***gredi@dtic.ua.es, brotons@dtic.ua.es, ibarra@dtic.ua.es, parraga@dtic.ua.es***

RESUMEN: En este articulo se describe un Procesador Educativo, desde su diseño con componentes discretos comerciales, atendiendo a los diferentes aspectos prácticos que influyen en el mismo, y su posterior simulación en un PC. Además se presenta el software que lo complementa elaborado para el estudio de la ruta de datos y la realización de prácticas en lenguaje ensamblador (MaNoTaS). El objetivo final es su utilización conjunta que permita comprender aspectos fundamentales relacionados con, la ejecución de las instrucciones, las señales de control necesarias, etc. y el resto de aspectos importantes que forman parte de un procesador real.

1.- INTRODUCCIÓN.

Las Estructuras del Computador son una parte importante en los curricular de las carreras de Informática [1]. Muchas son las aportaciones que han hecho otros autores, al respecto de la utilización de las herramientas de simulación para explicar la organización de los computadores e incluso su implementación con componentes discretos. Sin embargo, la potencia que ofrecen en estos momentos las herramientas CAD, permiten que los estudiantes estén familiarizados con el Diseño Electrónico Automático (EDA) y por lo tanto facilitar la comprensión de muchos conceptos. Siempre hemos creído que es muy conveniente que los alumnos entiendan los bloques funcionales del computador visualizando aunque sólo sea de forma virtual el movimiento de la información en su interior, aunque el objetivo final será conseguir que esa visión sea lo más práctica posible. Ya en 1987 Cutler y Eckert [2] describieron herramientas software que reunían estas características para el estudio de la arquitectura de los computadores las cuales mejoraron posteriormente [3]. Por esta época Devore y Hardin [4] diseñaron también un computador cuya misión era el aprendizaje de los aspectos software y hardware, con un procesador de 8 bits similar a los reales, y que utilizaron para introducir el lenguaje ensamblador, comprobando entre los estudiantes que el tiempo utilizado para la comprensión de estos tópicos fue sustancialmente inferior. Koneva y Denev [5] discuten el problema de la enseñanza de las arquitecturas reales y el lenguaje ensamblador, y describen un simulador de un hipotético procesador. El simulador fue diseñado para enseñar no solamente los conceptos del lenguaje ensamblador sino también la implementación y construcción de

lenguajes de alto nivel. En la misma línea podemos encontrar el trabajo de Barnett [6], Silvester [7] introduce en sus estudios la posibilidad de simulación de las diferentes estructuras que componen la máquina. Henderson [8] describe un paquete para el estudio de la arquitectura con diferentes niveles de detalle, que van desde la implementación electrónica hasta la simulación del procesador. Además, otros simuladores y emuladores han sido desarrollados por, Smith [9], Yen y Kim [10], Purvis, Yoho y Lamont [11], Tsuchiyama [12].

Nosotros realizamos mediante la utilización de herramientas CAD [13] [14] la implementación de un simulador software que permite aproximar al estudiante, mediante la animación y la utilización del lenguaje ensamblador, a la ruta de datos de un procesador. Además para que los alumnos puedan realizar variaciones al diseño del procesador, utilizaremos Pspice [15] para el diseño mediante componentes comerciales [16] [17] y utilizando las técnica top-down de diseño, del mismo procesador que se ha simulado mediante la herramienta software.

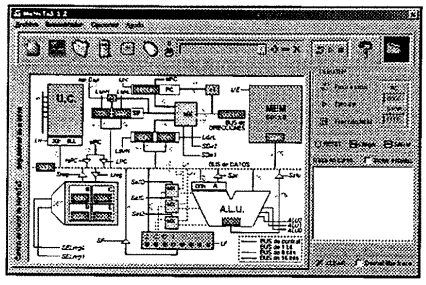

figura 1 Simulador de MaNoTaS

Llegados a este punto, los alumnos estarán familiarizados con la programación en ensamblador y la visualización de los datos moviéndose por la ruta de datos. Gracias al simulador, los alumnos empezarán a sentir la necesidad de conocer cómo se realiza el diseño de las diferentes estructuras que componen el computador. Éste será nuestro siguiente objetivo, para ello utilizaremos las técnicas de diseño convencionales de circuitos combinacionales y secuenciales.

2.- REPERTORIO DE INSTRUCCIONES.

Lo primero que debemos hacer es definir el repertorio de instrucciones y los modos de direccionamiento de las instrucciones que debe poseer la máquina, cuatro serán los modos con los que podremos acceder a los datos, inmediato, directo a memoria, directo a registro e indirecto a registro. El repertorio de instrucciones es el que se muestra en la tabla 1.

Tabla 1

Transferencia	Aritméticas	Lógicas	Control	E/S
LDA dir	ADD rl	ANA rl	JMP dir	IN #n
STA dir	ADI dato	ANI dato	JZ dir	OUT #n
LDAX	INR rl	ORA rl	JO dir	
STAX	DER rl	ORI dato	JC dir	
LFA	SUB rl	XRA rl	CALL dir	
SFA	SUI dato	XRI dato	RET	
MOV rl,r2	CMP rl	CMA	INT #n	
MVI $dato_8$,rl	CPI dato		IRET	
MVI $dato_{16}$,SP			CLI	
MVIH nom_eti,rl			STI	
PUSH rl			NOP	
POP rl				
PUSHF				
POPF				

3- UNIDAD ARITMÉTICO LÓGICA.

El diseño de esta unidad viene determinado por las operaciones que debe realizar. Aritméticas como la suma, resta, incrementos y decrementos, lógicas como la and, or y xor. Una vez elegidas las operaciones, debemos establecer el método de implementación, bien mediante operadores individuales o con un bloque multifuncional que se adapte a nuestras necesidades, la primera solución implica abundante circuitería, mientras que la segunda satisface los requerimientos buscados puesto que no se pierde de vista el concepto de unidad aritmético lógica y minimiza el diseño cumpliendo las especificaciones iniciales en cuanto a la utilización de componentes discretos, siendo esta segunda la solución adoptada. Necesitamos dos registros para realizar las operaciones, uno que es transparente al usuario, Registro Temporal (RT), y el otro el Acumulador (A) en el que además se almacenan los resultados de todas las operaciones. Como circuitos adicionales nos encontramos con el que se encarga de adaptar los códigos que se han establecido para las operaciones y los encargados de generar los señalizadores (flags) que se almacenarán en el Registro de Estado (RE).

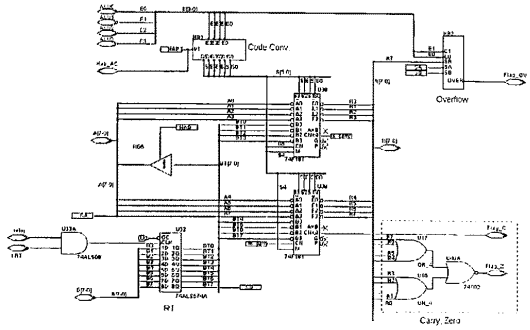

figura 2 Unidad Aritmético Lógica

4- BANCO DE REGISTROS.

Está formado por cuatro registros de propósito general (B, C, D y E) cuyo funcionamiento normal está asociado a la ALU y cuya finalidad es la de realización de operaciones de almacenamiento de variables de uso inmediato. El par de registros D-E tienen como función adicional la de almacenar una dirección para la utilización de las instrucciones con direccionamiento indirecto a registro de 16 bits (LDAX y STAX) Para la implementación de este Banco se han utilizado cuatro registros de 8 bits formados por biestables tipo D, así como un circuito decodificador que permitirá realizar las operaciones de carga y almacenamiento en cada uno de ellos.

5.- REGISTRO DE ESTADO.

Es fundamental conocer el resultado de la última operación realizada en la ALU, para ello implementaremos cada uno de los señalizadores (flags) mediante biestables independientes tipo D con precarga y reset, teniendo en cuenta que dichos señalizadores podrán ser alterados o bien por la propia ALU o por el contenido del Acumulador si se está ejecutando la instrucción SFA.

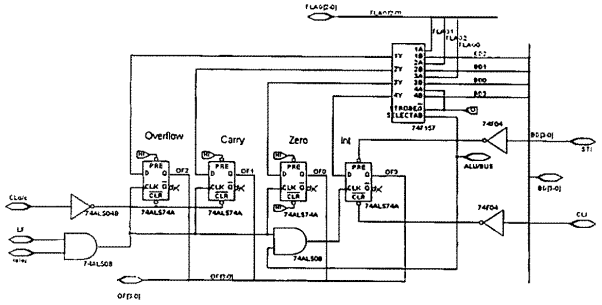

figura 3 Registro de Estado

6.- SELECCIÓN DE DIRECCIONES.

La memoria es única, y para acceder a su contenido es necesario proporcionar la dirección del dato que se quiere leer o escribir. Existen tres fuentes que pueden proporcionar dicha dirección, por un lado el Contador de Programa (PC), registro encargado de llevar la cuenta de las instrucciones del programa que se está ejecutando, el Puntero de la Pila (SP), registro que apunta a la primera dirección libre de una zona de memoria a la que se denomina Pila y que sirve para guardar las direcciones de retorno cuando se ha ejecutado una instrucción de llamada a subrutina (CALL) o de interrupción (INT), por último y teniendo en cuenta que se ha dotado al procesador de un direccionamiento indirecto a registro de 16 bits, utilizaremos el registro HL para almacenar temporalmente dicha dirección. La existencia de estas tres fuentes para la dirección, nos obliga a introducir un multiplexor 4 a 1 de 16 bits.

El Contador de programa, está formado por dos contadores de 8bits encadenados, con precarga y borrado. El registro HL esta formado por dos registros de 8 bits formados por biestables tipo D, iguales que los del Banco de Registros. El puntero de la Pila, es un contador de 16 bits ascendente/descendente con precarga, para evitar problemas debidos al encadenado de contadores a través de la patilla RC0 que poseen para tal efecto y que produciría valores no deseados, por ejemplo, RC0 se encuentra normalmente a nivel alto, cuando el contador está configurado en modo ascendente y sus salidas alcanzan el valor FFh, aquella pasa a nivel bajo, de forma que cuando el siguiente flanco de subida haga que las salidas pasen a 00h la salida RC0 vuelve a cambiar a nivel alto produciendo un flanco de subida en el segundo contador, algo similar, pero al revés ocurre cuando el funcionamiento es de descendente. Para evitar estos problemas hemos tenido que diseñar nuestro propio RC0.

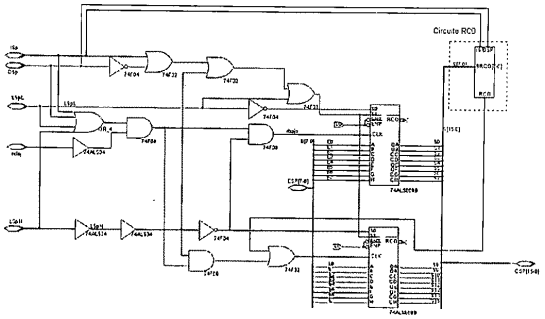

figura 4 Puntero de la Pila

7.- UNIDAD DE CONTROL.

La Unidad de Control que hemos diseñado, es lo suficientemente grande para descartar la implementación cableada. Puesto que hemos optado por una U.C. microprogramada partiremos de la información que nos proporciona los registros de instrucción (RI) y de estado (RE) y debemos generar todas las señales necesarias para el funcionamiento de la ruta de datos, teniendo en cuenta que hemos utilizado dispositivos activos por flanco de subida.La información procedente de los registros la llevamos hasta un Decodificador de Microinstrucciones (DM), que es el que se encargará de traducir el código de operación de cada instrucción en la dirección de inicio del microprograma correspondiente, para mayor simplicidad hemos hecho que las direcciones relacionadas con los saltos condicionales y la interrupción, sólo difieran en un bit. La dirección generada por DM deberá cargarse en el Contador de Microprograma (CM) e incrementarse con cada ciclo de reloj. Puesto que la ruta de datos funciona con flancos de subida, haremos que el CM funcione mediante flancos de bajada así las señales se generarán un semiperiodo antes de que se produzca el flanco por el cual son activas pudiéndose asumir sin dificultad los retardos de los componentes. La finalización de la ejecución de cada microprograma debe iniciar un nuevo "fetch" por lo que dentro de aquél debe estar incluida su dirección, además

existen muchas instrucciones que poseen fragmentos idénticos por lo que sería absurdo repetirlos y lo que haremos será que todas ellas compartan las mismas direcciones.
Puesto que disponemos de 33 señales de control más 6 internas de la propia U.C. los microprogramas se encuentran en cinco memorias EEPROM de 512 bytes.

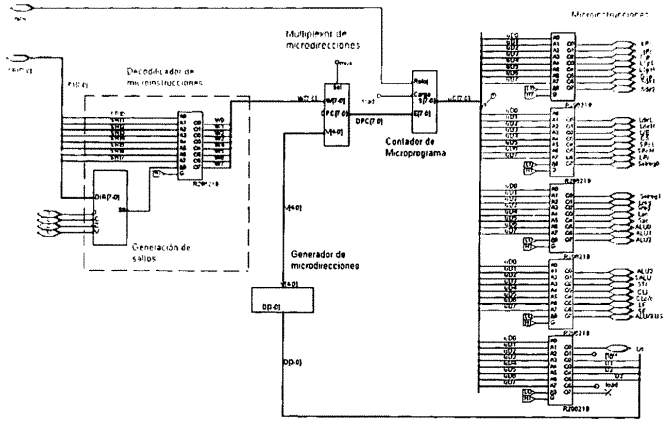

figura 5 Unidad de Control

Cuando se arranca el sistema todos los registros están a cero, por lo que en estas condiciones la U.C. carga el PC con la dirección que se encuentre en las posiciones 0000h y 0001h de la Memoria Principal, iniciando una fase de búsqueda, esto quiere decir que cuando se escriba un programa debemos tener la precaución de colocar en estas direcciones la dirección de inicio.

Teniendo en cuenta que tenemos 16 interrupciones vectorizadas, las 32 primeras posiciones de memoria deben estar reservadas. Si durante el programa no se modifica el contenido del vector 0, podremos utilizar la interrupción cero como "reset" software. Como ejemplo veamos un fragmento de la ejecución de un programa que incluye la instrucción LDAX y STA dir.

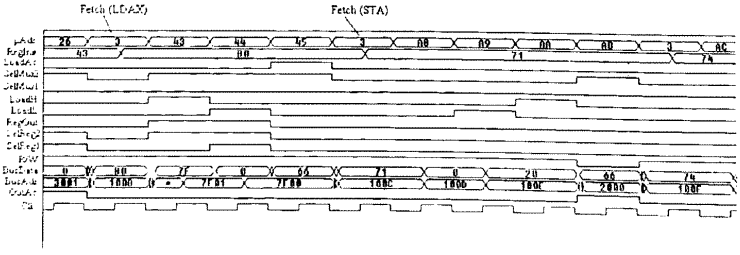

figura 6 Cronograma de las instrucciones LDAX y STA 2000

8.- REFERENCIAS.

[1] IEEE-CS Educational Actives Board. *The 1983 IEEE Computer Society model program in Computer Science and Engineering.* The Institute of Electrical and Electronic Engineers, Inc. 1983.

[2] M. Cutler and R.R. Eckert. "A microprogrammed Computer Simulator", *IEEE Trans. Educ., vol 30 pp 135-141. Aug 1987.*

[3] M. Cutler and R.R. Eckert. "Microprogrammed Computer Simulator Tools", *IEEE Trans. Educ., vol 33 pp 212-220. May 1990.*

[4] J.J. Devore and D.S. Hardin. "A computer design for introducing hardware and software concepts.", *IEEE Trans. Educ., vol E-30 nº 4 pp 219-226. May 1987.*

[5] L. Koneva and J. Denev, "EASY/VI – a new instructional computer", *ACM SIGCSE Bull. vol 22 nº 2 pp 55-58. June 1990.*

[6] B. L. Barnett. "A visual simulator for a simple machine and assembly language", *Proc. 26 th ACM SIGCSE Tech. Symp. Computer Science Education pp 233-278. 1997*

[7] P.P. Silvester, "Introducing computer structure by machine simulation", *IEEE Trans. Educ., vol 33 nº 4 pp 326-332.Nov 1990.*

[8] William D. Henderson, "Animated Models for Teaching aspects of Computer Systems Organization",*IEEE Trans. Educ., vol 37 pp 247-256. Aug 1994.*

[9] M. R. Smith, "A microprogramable microprocessor simulator and development system", *IEEE Trans. Educ., vol E-27 pp 93-100. May 1984.*

[10] R. Yen and Y. Kim, "Development and implementation of an educational simulator software package for specific microprogramming architecture", *IEEE Trans. Educ., vol E-29 pp 1-11. Feb 1986.*

[11] R. E. Purvis, R. D. Yoho, and B. Lamont, "MIME: An educational microprogrammable minicomputer emulator", *IEEE Trans. Educ., vol E-24 pp 257-261. Nov. 1981.*

[12] M. Tsuchiyama, "Development of an educational computer simulator", *Electronics and Communications in Japan Part 3, vol 74 nº 3 pp 25-35.*

[13] Douglas V. Hall. Teaching Design Methodology and "Industrial Strenght" EDA Tools in First-Term Freshman Digital Logic Course". *IEEE Trans. Educ.*, vol. 41 pp 45-49. Feb. 1998

[14] G. Puvvada and M.A. Breuer, "Teaching computer hardware design using commercial CAD tools". *IEEE Trans. Educ.*, vol. 36 pp 158-162. Feb. 1993

[15] MicroSim Pspice AD. Reference Manual v.7.1.1997

[16] Texas Instruments. Logic Selection Guide and DataBook. 1997

[17] Cypress Memory DataBook. 1997

[18] Stalling W. "Computer Organization and Architecture". *Prentice Hall.*1993