Simulador de la Máquina No Tan Sencilla (MaNoTas)

Autores: D. Ángel Grediaga Olivo, D. Párraga Navarro, D. Antonio Soriano Payá
Dpto. Tecnología Informática y Computación
Grupo de investigación: i³a
Universidad de Alicante
e-mail:gredi@dtic.ua.es

Resumen:

Se ha desarrollado el simulador de la Máquina No Tan Sencilla, para que se puedan realizar prácticas en la asignatura de Estructura de Computadores. La Máquina Sencilla se utiliza en algunas universidades [1], incluso existe en Internet [2], un simulador para dicha máquina. Sin embargo, nuestra idea ha sido dotarla con un repertorio de instrucciones más elaborado, con mas direccionamientos y por su puesto con la posibilidad de controlar las interrupciones y la pila.

1. Introducción.

En las carreras de Informática que se imparten en la Universidad de Alicante, el área de Arquitectura y Tecnología de Computadores tiene asignada la docencia, entre otras, de las asignaturas de Estructuras de Computadores I (6 créditos: 4,5 teóricos y 1,5 prácticos) y Estructura de Computadores II (6 créditos: 3 teóricos y 3 prácticos). Desde el inicio de las asignaturas, nos ha preocupado la realización de prácticas, y siempre hemos creído que sería conveniente que el alumno comprendiera las estructuras del computador, y "viese" como se mueve la información en su interior.

Nuestra primera experiencia fue implementar mediante el programa CASCAD, la Máquina Sencilla, sin embargo debido a las limitaciones del programa, el esquema era excesivamente complejo y los alumnos se perdía siguiendo la información, a pesar de ello entendían claramente la función de cada uno de los bloques. El siguiente paso fue realizar un simulador en entorno Microsoft Windows 3.11, en el cual el esquema era idéntico al que se explicaba en clase y al que se le añadía un ensamblador que nos ahorraba la tediosa tarea de codificar las instrucciones. En la actualidad y habiendo observado que resulta muy provechoso para la docencia y las prácticas, hemos desarrollado un simulador para una Máquina No Tan Sencilla.

Antes de realizar la descripción del simulador, debemos realizar el diseño de "MaNoTaS". La naturaleza jerárquica de los sistemas complejos, es esencial tanto para su diseño como para su descripción. En cada nivel el sistema consta de un conjunto de componentes y sus interrelaciones. Podemos decir que el comportamiento en cada nivel depende sólo de una caracterización abstracta y simplificada del sistema que hay en el nivel inmediatamente inferior, por lo tanto, en cada nivel al diseñador sólo le importa su estructura, modo en el que los componentes están interrelacionados, y su función, la operación de cada componente individualmente como parte de la estructura.

En términos de descripción tenemos dos opciones: empezar por lo más bajo y construir una descripción completa, o comenzar con una visión desde arriba y descomponer el sistema en subpartes. La experiencia nos ha enseñado que la descripción de arriba a bajo ("top-down") es la más clara y efectiva.

2. Memoria.

"MaNoTaS" responde a la arquitectura clásica de von Neumann, en la cual sólo existen dos caminos para el intercambio de información con la memoria.

- Un intercambio bidireccional de datos.
- Un movimiento unidireccional de instrucciones hacia la unidad de control.

Por lo tanto, la memoria influye en el rendimiento del computador, ya que cada instrucción exigirá una serie de accesos a memoria para leer el código de operación y los operandos. La capacidad de direccionamiento de "MaNoTaS" es de 64kx8bits, y la consideraremos como un conjunto de posiciones a las que le corresponde una determinada dirección. Cuando hay que almacenar información cuyo tamaño supera un byte, como son los casos de llamadas a subrutinas y saltos, se ha elegido el criterio de almacenar en primer lugar el byte bajo.

3. Direccionamientos.

La Máquina Sencilla sólo tiene el direccionamiento directo, sin embargo, a nuestra máquina la dotaremos de nuevos direccionamientos, que son:

Inmediato
Directo a memoria
Directo a registro
Indirecto a registro (de 16 bits).

Teniendo en cuenta estos modos de direccionamiento, y el repertorio de instrucciones con el que se dota a "MaNoTaS" se puede comprobar que no todas las instrucciones caben en una posición de la memoria, hay que recordar que la memoria principal posee un ancho de ocho bits, por lo tanto, existirán instrucciones de uno, dos y tres bytes. El primer byte siempre definirá la instrucción, es decir, el código de operación y los otros dos se utilizarán para definir los operandos.

4. Repertorio de instrucciones.

Hemos pretendido que el repertorio de instrucciones sea regular, completo y eficaz. Podemos ver en la Tabla 1 el repertorio de instrucciones en el que podemos interpretar que:

- r1 y r2 se refiere al nombre de uno de los registros A, B, C, D
- dir, es una dirección de memoria comprendida entre 0h y FFFFh, es decir, los 64Kbytes de la memoria.
- #n, es el número de la interrupción o el número de puerto, dependiendo de la instrucción a la que haga referencia, entrada/salida o interrupción.
- dato, representa un valor de ocho bits, salvo para la instrucción MVI dato,SP que será de 16 bits.

Tabla 1

Transferencia	Aritméticas	Lógicas	Control	E/S
LDA dir	ADD r1	ANA r1	JMP dir	IN #n
STA dir	ADI dato	ANI dato	JZ dir	OUT #n
LDAX	INR r1	ORA r1	JO dir	
STAX	DER r1	ORI dato	JC dir	
LFA	SUB r1	XRA r1	CALL dir	
SFA	SUI dato	XRI dato	RET	
MOV r1,r2	CMP r1	CMA	INT #n	
MVI dato,r1	CPI dato		IRET	
PUSH r1			CLI	
POP r1			STI	
PUSHF			NOP	
POPF				

Teniendo en cuenta la premisa de que el repertorio sea regular, hemos implementado pocos casos especiales en las instrucciones, de hecho, aunque se ha comentado anteriormente, la instrucción MVI dato, r1, en la que r1 puede tomar el valor del registro SP y en cuyo caso el dato debe ser de 16 bits. Otra particularidad es que #n indicará el número de la interrupción, comprendido entre 0 y 31.

En cuanto al almacenamiento en memoria de aquellas instrucciones que ocupan tres bytes, son las que hacen referencia a direcciones, y se guardan en memoria primero el C.O. y luego DirL y DirH.

Veamos la codificación de algunas de las instrucciones.

NomInstrucción
cod_byte1 [cod_byte2 [cod_byte3]]

4.1 Instrucciones de un byte

MOV A, B	40h	MOV B, A	00h
MOV A, C	41h	MOV B, C	08h
MOV A, D	42h	MOV B, D	0Ch
MOV A, E	43h	MOV B, E	10h

MOV C, A	01h	MOV D, A	02h
MOV C, B	05h	MOV D, B	06h
MOV C, D	0Dh	MOV D, C	0Ah
MOV C, E	11h	MOV D, E	12h
MOV E, A	03h		
MOV E, B	07h		
MOV E, C	0Bh		
MOV E, D	0Eh		
LDAX	B0h	STAX	90h
STI	AAh	LFA	81h
CLI	ABh	SFA	82h
ADD A	45h	SUB A	46h
ADD B	30h	SUB B	18h
ADD C	31h	SUB C	19h
ADD D	32h	SUB D	1Ah
ADD E	33h	SUB E	1Bh
CMP A	47h	CMP B	1Ch
CMP C	1Dh	CMP D	1Eh
CMP E	1Fh		
INR A	4Bh	DER A	A0h
INR B	2Ch	DER B	A1h
INR C	2Dh	DER C	A2h
INR D	2Eh	DER D	A3h
INR E	2Fh	DER E	A4h
CMA	80h		
ANA A	48h	ORA A	49h
ANA B	20h	ORA B	24h
ANA C	21h	ORA C	25h
ANA D	22h	ORA D	26h
ANA E	23h	ORA E	27h
XRA A	4Ah	XRA B	28h
XRA C	29h	XRA D	2Ah
XRA E	2Bh		
PUSH A	55h	POP A	5Ah
PUSH B	56h	POP B	5Bh
PUSH C	57h	POP C	5Ch
PUSH D	58h	POP D	5Dh
PUSH E	59h	POP E	5Eh
PUSHF	50h	POPF	51h
RET	7Bh		
IRET	7Ch		
NOP	FFh		

4.2 Instrucciones de dos bytes

MVI dato, A	64h	dato
MVI dato, B	60h	dato
MVI dato, C	61h	dato
MVI dato, D	62h	dato
MVI dato, E	63h	dato
ADI dato	35h	dato
SUI dato	36h	dato
CPI dato	37h	dato
ANI dato	68h	dato
ORI dato	69h	dato
XRI dato	6Ah	dato
INT #n	54h	numInterrupción
IN #n	52h	numPuerto
OUT #n	53h	numPuerto

4.3 Instrucciones de tres bytes

MVI dato, SP	65h	dL	dH
LDA dir	70h	dL	dH
STA dir	71h	dL	dH
JMP dir	74h	dL	dH
JZ dir	72h	dL	dH
JC dir	73h	dL	dH
JO dir	75h	dL	dH
CALL dir	7Ah	dL	dH

A continuación se muestra en el lenguaje de transferencia entre registros algunas de las instrucciones implementadas

Inst.	Descripción
LDA dir	A ← M(dirHdirL)
STA dir	M(dirHdirL) ← A
LDAX	A ← M(D-E)
STAX	M(D-E) ← A
CALL dir	SP ← SP+1;M(SP) ← PC;PC← dir

5. Unidad Aritmético Lógica.

Según el repertorio de instrucciones, disponemos de las típicas operaciones de suma, resta y comparación, así como las lógicas más usuales, por lo tanto el diseño de la A.L.U. no es excesivamente complicado, teniendo en cuenta además que vamos a trabajar con ocho bits, pero sirve para introducir a los alumnos en este bloque del computador. La necesidad de saber en que estado se encuentra la A.L.U nos lleva a introducir el registro de estado (RF), el cual poseerá cuatro bits: indicador de cero (Z), overflow (O), acarreo (C) y el de interrupciones (I). Existe la posibilidad de cargar los flags en el acumulador, LFA, y al revés cargar los flags con el contenido del acumulador, SFA, para ello se han conectado los bits del acumulador con el registro de señalizadores, RF, de la siguiente manera, A_0 con Z, A_1 con C y A_2 con O. Con CLI y STI controlaremos el señalizador de interrupciones.

Disponemos de cuatro registros de uso general de ocho bits cada uno, que facilitan el tratamiento de los datos.

El emparejamiento de dos registros D-E, se utiliza para el direccionamiento indirecto a registro mediante las instrucciones con las instrucciones LDAX y STAX

6. Selección de direcciones.

La inclusión del direccionamiento directo a memoria, con posibilidad de 64K, nos obliga a la utilización de un registro auxiliar de 16 bits, cuyo nombre es DirL y DirH (HL), de igual manera, la inclusión de una pila nos obliga a disponer de un registro de 16 bits (SP) y como además disponemos del PC. Para colocar direcciones en el bus disponemos tres fuentes y por lo tanto será necesario realizar la multiplexación de cada una de estas fuentes hacia el bus de direcciones de la memoria. Para ello utilizaremos un multiplexor con dos señales de control (Sdir1, Sdir2).

7. Descripción del simulador.

La pantalla principal del simulador de "MaNoTaS" es la que podemos ver a continuación. La ruta de datos nos presenta los bloques más importantes y en los que deseamos hacer hincapié.

Pasamos a describir cual sería el proceso para realizar una simulación. En el menú principal disponemos del icono que permite crear un programa nuevo en ensamblador. Este editor nos permite la utilización de comentarios, mediante (;) así como definir los datos en decimal o hexadecimal, el ensamblador nos realiza una revisión sintáctica del programa, indicándonos en la línea que tenemos el error. Los programas en ensamblador poseen la extensión *.ass, y una vez ensamblados, la extensión *.ejc, que serán los que podamos cargar en el simulador, que lo podemos hacer bien mediante el icono de la carpeta abierta, bien con el icono del signo más. Se pueden ir cargando todos los ejecutables que necesitemos, el que se ejecutará será el que se esté visualizando en la ventana que se dispone al efecto, en la figura 1, practica.ejc, que empieza en la dirección 0000h.

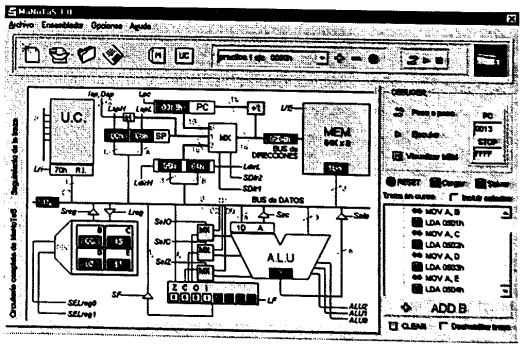

figura 1. Pantalla principal del simulador

Para realizar la depuración del programa disponemos de varias opciones: por un lado podemos ejecutar paso a paso, por otro podemos ejecutar un número determinado de instrucciones, poniendo la dirección de parada en el recuadro STOP.

Además, disponemos de una manera adicional para comprobar la ejecución de nuestros programas, podemos habilitar la traza y se verá en el recuadro dedicado al efecto, las instrucciones que se van ejecutando y la que se va a ejecutar, incluso si habilitamos la opción de "incluir estados" podremos ver los estados por los que pasa cada una de las instrucciones en su ejecución.

En la parte lateral del simulador existe la pestaña "seguimiento de la traza", mediante esta opción todavía podemos ver mejor cual ha sido el proceso en la ejecución del programa, pues podemos indicar los registros e incluso las instrucciones que se han ejecutado. De manera que después de ejecutado el programa podríamos ver una pantalla del siguiente tipo.

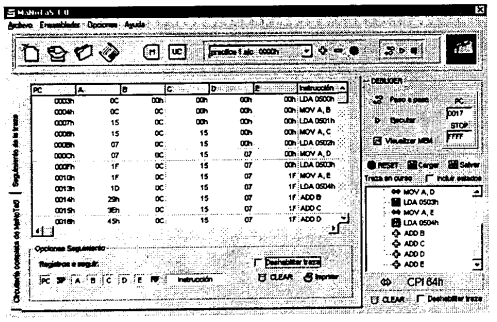

figura 2. Traza del programa

En la que podemos, figura 2, la evolución del programa al mismo tiempo que lo hemos ido ejecutando, visualizándose el contenido de los registros que están habilitados, en este caso, A, B, C, D, E, PC y RI, además podemos realizar la impresión de este tipo de traza.

8. Ejemplos de programas.

El siguiente programa nos muestra la utilización del direccionamiento a memoria y consiste en comprobar si la suma del contenido de cinco posiciones de memoria suman 64h.

```
;este programa comprueba si los 5 elementos
;de la lista suman 64h
;primero cargamos los datos
;posición de inicio 2000h
;datos almacenados en las posiciones
;500h a 504h

;****************************************

lda 500h
mov a,b
lda 501h
mov a,c
lda 502h
mov a,d
lda 503h
mov a,e
lda 504h
;ahora los sumamos
add b
add c
add d
add e
cpi 64h         ;si la suma da 100
                ;entonces saltamos al
                ;programa que se encuentre en
                ;la posición 4000h
jz [4000h]      ; que los ordena
                ;de mayor a menor
                ;si no, salta a fin de programa

jmp fin         ;FIN del programa
fin:
```

Veamos como se puede realizar la programación de una interrupción. Las interrupciones están numeradas del 0 al 31, cuando deseamos ejecutar una, debemos saber que el número de la interrupción multiplicado por dos nos indica la dirección en la cual debemos colocar la dirección en la que empieza la rutina de tratamiento de la interrupción deseada. Nada mejor que un ejemplo:

```
;RSI que multiplica el registro B por
;el registro C y deja el resultado en A
;Esta RSI en concreto se activa con
;la interrupción 11 (2*11=22=16h)
```

```
cli     ;mientras la RSI este activa no
        ;atenderá otra interrupción.
mov c,a

;***** aquí comienza la RSI *****
jz fin  ;caso base a*0=0

bucleMultiplicacion:
der b
jz fin
add c
jmp bucleMultiplicacion
fin:  ;no hace falta STI ya que IRET
mov c, a
iret  ;pone el flag I a 1 al volver
```

La programación de la RSI anterior nos permite que la utilicemos en un programa como el siguiente

```
; Realiza la multiplicación del contenido
; del registro B por el C, utilizando la
; interrupción 11, anteriormente definida
; dirección de inicio 1000H
; resultado en A

mvi F000h,sp      ;inizializamos SP
mvi 12,c          ; cargamos los datos
mvi 3,b           ;en los registros
int 11
sta 2000h         ;guardamos result.
```

9. Conclusiones.

La experiencia ha sido muy positiva, los alumnos nos han aportado ideas para mejorar la metodología, que permita afianzar mejor los conceptos, por ejemplo:

- Un grupo ha diseñado y simulado la memoria mediante PSpice®, otros la Unidad Aritmético Lógica, y existe otro grupo trabajando en el diseño de la Unidad de Control. De manera que en poco tiempo se podrá utilizar "MaNoTaS" en entorno PSpice®

- Nosotros, a sugerencia de algunos alumnos, estamos trabajando en una versión más avanzada en la que los bloques estén definidos, se puedan colocar y al mismo tiempo se genere el ensamblador, con las instrucciones relativas a dichos bloques, y que la U.C. pueda programarse para que sea capaz de controlar la ruta de datos que se haya diseñado.

10. Referencias.

[1] La máquina sencilla. Introducción a la estructura básica de un computador.
Eduard Ayguadé i Parra
Juan José Navarro Guerrero
Miguel Valero García. U.P.C.

[2] http://songoku.udg.es/~ferrang/java/aplicacio/aplicacio.html