



ESTRUCTURA DE COMPUTADORES

**Práctica de microprogramación
Enunciado de la práctica**

**Departamento de Arquitectura y Tecnología
de Sistemas Informáticos
Facultad de Informática - UPM**

**Curso 2008/2009
7/11/2008**

9.3 JUEGO DE INSTRUCCIONES

El juego de instrucciones que se debe microprogramar incluye instrucciones de movimiento de datos (LD, ST, PUSH, POP y EXCH), de aritmética entera (INC, DEC, ADD y SUBI) y de control de programa (JMP.U, CALL, RET, RTE, Bcond, HALT, RESET y TRAP).

La tabla 1 contiene el juego de instrucciones del 68001 descrito según el estándar IEEE 694. La mayoría de las instrucciones de dicho juego proviene del conjunto de instrucciones del 68000.

Para cada una de estas instrucciones se ha elegido un subconjunto de los modos de direccionamiento disponibles en el 68000.

Para esta convocatoria hay algunas instrucciones para las que el alumno no debe microprogramar **todas las combinaciones posibles de registros** o condiciones incluidas en la tabla.

Las instrucciones LD .Dn, #Dato y LD .An, #Dato permiten cargar los registros de 8 y 16 bits con el dato inmediato que se proporciona en cada caso como operando de la instrucción. No provocan excepciones.

Las instrucciones LD .Dn, [.An] permiten cargar los registros de 8 bits desde memoria. Si se ejecutan en modo usuario y la dirección de memoria contenida en el registro An es privilegiada, provocan una excepción de violación de privilegio por lectura en memoria durante su ejecución. En tal caso, al comienzo de la microrrutina de tratamiento de excepciones, el contenido que han de tener los registros utilizados por estas instrucciones es el siguiente:

PC: Apuntará a la siguiente instrucción a LD .Dn, [.An].

Dn: Permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.

Para esta convocatoria sólo se debe microprogramar las siguientes instrucciones de este grupo:

```
LD .D0, [.A0]
LD .D0, [.A1]
LD .D0, [.A2]
```

Las instrucciones ST .Dn, [.An] permiten depositar en memoria el contenido de los registros de 8 bits. Si se ejecutan en modo usuario y la dirección de memoria contenida en el registro An es privilegiada, provocan una excepción de violación de privilegio por escritura en memoria durante su ejecución. En tal caso, al comienzo de la microrrutina de tratamiento de excepciones, el contenido que han de tener los registros y posiciones de memoria utilizados por estas instrucciones es el siguiente:

PC: Apuntará a la siguiente instrucción a ST .Dn, [.An].

[.An]: No se realizará la escritura del registro .Dn.

Para esta convocatoria sólo se debe microprogramar las siguientes instrucciones de este grupo:

```
ST .D1, [.A0]
ST .D1, [.A1]
ST .D1, [.A2]
```

La instrucción EXCH_D (*exchange data*) intercambia el contenido de los registros de 8 bits, dejando en .D0 el contenido de .D1 previo a la ejecución de la instrucción y en .D1 el contenido previo de .D0. No provoca excepciones.

La instrucción EXCH_A (*exchange address*) intercambia el contenido de los registros de 16 bits, dejando en .A0 el contenido de .A1 previo a la ejecución de la instrucción y en .A1 el contenido previo de .A0. No provoca excepciones.

La instrucción LD .SR, [.An++] carga en el registro SR el contenido de una posición de memoria. Si como consecuencia de la ejecución de esta instrucción el flag N pasa a ser 0, se intercambian entre sí los punteros de pila de usuario y de supervisor. Esta instrucción es privilegiada. Si se ejecuta en modo usuario genera una excepción de *violación de privilegio por código de operación*.

El modo de direccionamiento utilizado en esta instrucción es *indirecto a registro con post-incremento*, de forma que después de leer el registro SR de memoria se incrementa en una unidad el registro An.

En caso de que genere una excepción, al comienzo de la microrrutina de tratamiento de excepciones, el contenido que han de tener los registros y posiciones de memoria utilizados por esta instrucción es el siguiente:

- PC: Apuntará a la siguiente instrucción.
- An: Permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.
- SR: Permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.

La instrucción ST .SR, [--.An] almacena en memoria el registro SR. Esta instrucción es privilegiada. Si se ejecuta en modo usuario genera una excepción de *violación de privilegio por código de operación*.

El modo de direccionamiento utilizado en esta instrucción es *indirecto a registro con pre-decremento*, de forma que antes de haber escrito el registro SR en memoria se decrementa en una unidad el registro An.

En caso de que genere una excepción, al comienzo de la microrrutina de tratamiento de excepciones, el contenido que han de tener los registros y posiciones de memoria utilizados por esta instrucción es el siguiente:

- PC: Apuntará a la siguiente instrucción.
- An: Permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.
- [An]: No se habrá modificado con el contenido del registro de estado.

Las instrucciones PUSH .Rn permiten guardar en memoria el contenido de los registros del 68001. Los registros de 8 bits se guardan en **una** posición de memoria. Los de 16 bits se guardan en **dos** posiciones consecutivas de memoria comenzando por el byte **menos** significativo. Se utiliza el registro A2 como puntero de pila.

El modo de direccionamiento utilizado en las instrucciones PUSH es *indirecto a registro con pre-decremento*, de forma que por cada byte escrito en memoria se decrementa en una unidad el registro A2. El funcionamiento de esta instrucción es el siguiente:

- Si (Rn=A0 ó Rn=A1)
 - [--.A2] ← .Rn_L
 - [--.A2] ← .Rn_H
- en caso contrario
 - [--.A2] ← .Rn

La instrucción **PUSH .CCR** almacena en memoria, tanto en modo usuario como en modo privilegiado, un 0 en el bit de memoria correspondiente al flag N.

Si se ejecutan en modo usuario y la dirección de memoria contenida en el registro A2 predecrementado es privilegiada, las instrucciones PUSH .Rn provocan una excepción de *violación de privilegio por escritura en memoria* durante su ejecución. En tal caso, al comienzo de la microrrutina de

tratamiento de excepciones, el contenido que han de tener los registros y posiciones de memoria utilizados por estas instrucciones es el siguiente:

caso a): La excepción la provoca el primer ciclo de escritura.

PC: Apuntará a la siguiente instrucción.

A2: Habrá sido decrementado en una unidad.

Pila: No se habrá realizado la escritura bien sea un registro de 8 bits o la parte baja de un registro de 16 bits.

caso b): La excepción la provoca el segundo ciclo de escritura (Esta situación sólo se da cuando A2 apunta a la segunda dirección no privilegiada).

PC: Apuntará a la siguiente instrucción a PUSH.

A2: Se habrá decrementado en dos unidades.

Pila: Se habrá actualizado con la parte baja de Rn.

Para esta convocatoria sólo se debe microprogramar las siguientes instrucciones de este grupo:

```
PUSH .CCR
PUSH .A0
PUSH .D0
```

Las instrucciones POP .Rn permiten cargar de memoria el contenido de los registros del 68001. Los registros de 8 bits se recogen de **una** posición de memoria. Los de 16 bits se leen de **dos** posiciones consecutivas de memoria comenzando por el byte **más** significativo. Se utiliza el registro A2 como puntero de pila.

El modo de direccionamiento utilizado en las instrucciones POP es *indirecto a registro con post-incremento*, de forma que se incrementa en una unidad el registro A2 después de leer de memoria cada byte. El funcionamiento de esta instrucción es el siguiente:

```
Si (Rn=A0 ó Rn=A1)
    .RnH ← [.A2++]
    .RnL ← [.A2++]
en caso contrario
    .Rn ← [.A2++]
```

La instrucción **POP .CCR** carga en el registro CCR el contenido de la posición de memoria correspondiente, pero tiene la particularidad de que **nunca modifica el nivel de ejecución**, es decir, ni ejecutando esta instrucción en modo usuario ni haciéndolo en modo privilegiado, se ve afectado el flag N.

Si se ejecutan en modo usuario y la dirección de memoria contenida en el registro A2 es privilegiada, estas instrucciones provocan una excepción de *violación de privilegio por lectura en memoria* durante su ejecución. En tal caso, al comienzo de la microrrutina de tratamiento de excepciones, el contenido que han de tener los registros y posiciones de memoria utilizados por estas instrucciones es el siguiente:

caso a) La excepción la provoca el primer ciclo de lectura.

PC: Apuntará a la siguiente instrucción.

A2: Permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.

Rn: Permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.

caso b): La excepción la provoca el segundo ciclo de lectura (Esta situación sólo se da cuando A2 apunta inicialmente a la última dirección no privilegiada: H'FFFF).

PC: Apuntará a la siguiente instrucción.

A2: Habrá sido incrementado en una unidad.

Rn: La parte alta de Rn habrá sido modificada con la cima de la pila. La parte baja de Rn permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.

Para esta convocatoria sólo se debe microprogramar las siguientes instrucciones de este grupo:

POP .CCR
POP .A1
POP .D1

Las instrucciones INC .An y DEC .An realizan la operación " $.An \leftarrow .An + 1$ " y " $.An \leftarrow .An - 1$ " respectivamente. Los operandos son de 16 bits y se genera un resultado también de 16 bits. Los flags no se ven alterados al ejecutar estas instrucciones. No provocan excepciones.

La instrucción SUBI .Dn, #Dato realiza la operación " $.Dn \leftarrow .Dn - Dato$ ". Los operandos son de 8 bits. Genera un resultado también de 8 bits y un acarreo de resta (Borrow), que se almacena en el bit de acarreo del registro CCR. Los demás bits de CCR se actualizarán de acuerdo con el resultado de la operación. No provoca excepciones.

La instrucción ADD .D0, .Dn realiza la operación " $.D0 \leftarrow .D0 + .Dn$ ". Los operandos son registros de 8 bits. Genera un resultado también de 8 bits. Los flags se actualizan de acuerdo con el resultado de la operación. No provoca excepciones.

Para esta convocatoria sólo se debe microprogramar la instrucción ADD .D0, .D1.

La instrucción JMP.U (jump to user code), es privilegiada. Si se ejecuta en modo usuario genera una excepción de *violación de privilegio por código de operación*. Si se ejecuta en modo privilegiado, asigna al registro PC el valor especificado como operando y pone a cero el flag N, con el consiguiente intercambio de los punteros de pila.

En caso de que genere una excepción, al comienzo de la microrrutina de tratamiento de excepciones, el contenido que han de tener los registros y posiciones de memoria utilizados por esta instrucción es el siguiente:

PC: Apuntará a la parte alta de la dirección de salto de JMP.U.

SR: Permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.

A2: No se produce el intercambio de punteros de pila, por lo que permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.

NOTA: Durante el procesamiento de las excepciones la microrrutina de tratamiento de excepciones almacena en la pila de modo supervisor los valores contenidos en los registros PC y SR. Dado que, cuando la ejecución de una instrucción JMP.U provoca una excepción, el PC queda apuntando al primero de sus operandos (la parte alta de la dirección de salto), la pila de modo supervisor queda en un estado inconsistente: Si se ejecuta la instrucción RTE en este estado, no se cede el control a una instrucción sino a un operando, siendo el resultado totalmente impredecible. Esta situación se da en la realidad y no supone ningún "problema", pues, cuando un programa de usuario provoca una excepción de violación de privilegio, quiere decir que funciona de forma errónea o maliciosa. En cualquiera de los dos casos el sistema operativo "aborta" la ejecución de ese programa y cede el control a otro programa avisando al usuario que ordenó la ejecución del primero. Las instrucciones CALL y RET pueden generar inconsistencias semejantes a ésta y se resuelven de la misma manera.

Las instrucciones **CALL** y **RET** sirven para desviar el flujo de control de programa, definiendo respectivamente el salto a subrutina y el retorno de subrutina. Consecuentemente, CALL realizará escrituras en la pila y RET leerá de la pila. Si se ejecutan en modo usuario y la dirección contenida en el puntero de pila (A2) es privilegiada, CALL genera una excepción de *violación de privilegio por escritura en memoria* y RET por *lectura en memoria*. En tal caso, al comienzo de la microrrutina de tratamiento de excepciones, el contenido que han de tener los registros y las posiciones de memoria utilizados por estas instrucciones es el siguiente:

CALL

- caso a) La excepción se produce al escribir la parte baja del PC.
 PC: Apuntará a la siguiente instrucción.
 A2: Habrá sido decrementado en una unidad.
 Pila: No se habrá realizado la escritura de la parte baja del PC.
- caso b) La excepción se produce al escribir la parte alta del PC (Esta situación sólo se da cuando A2 apunta a la segunda dirección no privilegiada).
 PC: Apuntará a la siguiente instrucción.
 A2: Se habrá decrementado en dos unidades.
 Pila: Se habrá actualizado con el valor de PC_L.

RET

- caso a) La excepción se produce al leer la parte alta del PC.
 PC: Apuntará a la siguiente instrucción.
 A2: Permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.
- caso b) La excepción se produce al leer la parte baja del PC (Esta situación sólo se da cuando A2 apunta a la última dirección no privilegiada: H'FFFF).
 PC: La parte alta del PC se habrá actualizado con la cima de la pila. La parte baja del PC permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.
 A2: Se habrá incrementado en una unidad.

La instrucción **CALL \$desp**, para calcular la dirección efectiva del salto, realiza la operación "**PC +ca2 desp**" después de haber leído todos los parámetros de la instrucción y habiendo incrementado correspondientemente el registro PC. Ejemplos:

1234H	96	CALL \$0000	
1235H	00		
1236H	00		
1237HPC ← 1237H +ca2 0000H = 1237H
1234H	96	CALL \$0024	
1235H	00		
1236H	24		
1237HPC ← 1237H +ca2 0024H = 125BH
1234H	96	BR \$FFF8	
1235H	FF		
1236H	F8		
1237HPC ← 1237H +ca2 FFF8H = 122FH

La instrucción **RTE**, retorno de excepción, es privilegiada. Si se ejecuta en modo usuario genera una excepción de *violación de privilegio por código de operación*. Si se ejecuta en modo privilegiado extrae de la pila el registro de estado SR y el contador de programa, PC. Si la ejecución de esta

instrucción conlleva el cambio a modo usuario (flag N pasa a valer "0") debido al nuevo valor del registro de estado, los punteros de pila se intercambian, al final de la instrucción, de acuerdo con el cambio de modo.

En caso de que genere una excepción, al comienzo de la microrrutina de tratamiento de excepciones, el contenido que han de tener los registros y posiciones de memoria utilizados por esta instrucción es el siguiente:

- PC: Apuntará a la siguiente instrucción.
- SR: Permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.
- A2: No se produce el intercambio de punteros de pila, por lo que permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.

Las instrucciones Bcond /Dir realizan un salto absoluto a la dirección Dir. Para esta convocatoria sólo se debe microprogramar las instrucciones correspondientes a las condiciones **T (True)**, **C (Carry)** y **NZ (Not Zero)**, es decir:

BT /Dir
BC /Dir
BNZ /Dir

No provocan excepciones.

La instrucción HALT es una instrucción privilegiada. Si se ejecuta en modo supervisor, el 68001 detiene su ejecución. Si se ejecuta en modo usuario, se genera una excepción de *violación de privilegio por código de operación* (vector número 5).

En caso de que genere una excepción, al comienzo de la microrrutina de tratamiento de excepciones, el contenido que han de tener los registros y posiciones de memoria utilizados por esta instrucción es el siguiente:

- PC: Apuntará a la siguiente instrucción.
- SR: Permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.

La instrucción RESET es una instrucción privilegiada. Si se ejecuta en modo supervisor se genera una *excepción de reset* (con vector 0), provocando la inicialización del 68001. Si se ejecuta en modo usuario, se genera una excepción de *violación de privilegio por código de operación* (vector número 5).

En caso de que genere una excepción de violación de privilegio, al comienzo de la microrrutina de tratamiento de excepciones, el contenido que han de tener los registros y posiciones de memoria utilizados por esta instrucción es el siguiente:

- PC: Apuntará a la siguiente instrucción.
- SR: Permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.

Las instrucciones TRAP #0 y TRAP #1 generan la *excepción de trap* que les corresponde (vectores 3 y 4 respectivamente).

El contenido que han de tener los registros y posiciones de memoria utilizados por estas instrucciones al comienzo de la microrrutina de tratamiento de excepciones es el siguiente:

- PC: Apuntará a la siguiente instrucción a TRAP.
- SR: Permanecerá con el mismo valor que tenía antes de comenzar la ejecución de esta instrucción.

Las excepciones de violación de privilegio por código de operación (vector número 5) deben detectarse y procesarse en el microcódigo correspondiente a las instrucciones privilegiadas, pero siempre antes de que éstas realicen ninguna operación, incluyendo la lectura de sus operandos. Todos los flags, registros y posiciones de memoria deben permanecer inalterados. De esta forma, el valor del registro PC que se almacena en la pila quedará apuntando al primero de sus operandos, en caso de que los tenga, o a la instrucción siguiente.

Las excepciones de violación de privilegio por fetch de instrucción (vector número 6) deben detectarse y procesarse en la rutina de fetch. Se producen cuando, estando en modo usuario, se trata de ejecutar una instrucción almacenada en una dirección de memoria protegida (direcciones 0000H a 0FFFH). En tal caso, no se realiza el ciclo de fetch y, al comienzo de la microrrutina de tratamiento de excepciones, el contenido que han de tener los registros involucrados es el siguiente:

- PC: Permanecerá con el mismo valor que tenía antes de comenzar el ciclo de fetch.
- IR: Permanecerá con el mismo valor que tenía antes de comenzar el ciclo de fetch.
- SR: Permanecerá con el mismo valor que tenía antes de comenzar el ciclo de fetch.

Como norma general, **el tratamiento de las excepciones no debe afectar al comportamiento de los flags**. Es decir, si una instrucción no modifica los flags, la detección, generación y tratamiento de una excepción provocada durante su ejecución tampoco debe modificarlos, debiendo ser almacenados en la pila en el mismo estado en que se encontraban al comenzar la ejecución de la instrucción y debiendo permanecer en el mismo estado al comienzo de la ejecución de la instrucción siguiente, salvo el flag N que deberá quedar activado (N=1).

MNEMÓNICO	FORMATO DE INSTRUCCIÓN	EXPLICACIÓN	LONG.	FLAGS
LD .Dn, #Dato	0000100d Dato	.Dn ← Dato	2	NO
EXCH_D	00001010	.D0 ↔ .D1	1	NO
EXCH_A	00001011	.A0 ↔ .A1	1	NO
LD .An, #Dato	000011a1a0 Dato _H Dato _L	.An _H ← Dato _H .An _L ← Dato _L	3	NO
LD .Dn, [.An]	00010da1a0	.Dn ← [.An]	1	NO
ST .Dn, [.An]	00011da1a0	[.An] ← .Dn	1	NO
LD .SR, [.An++]	00110a1a00	if N=1 then .SR ← [.An++] if N=0 then A2 _H ↔ [0000H] A2 _L ↔ [0001H] else excepción 5	1	TODOS
ST .SR, [--.An]	00111a1a00	if N=1 then [--.An] ← .SR else excepción 5	1	NO
POP .Rn	01000r2r1r0	véase la sección 9.3	1	Si Rn=CCR TODOS los de CCR Si Rn <> CCR: NO
PUSH .Rn	01001r2r1r0	véase la sección 9.3	1	NO
INC .An	011100a1a0	.An ← .An + 1	1	NO
DEC .An	011010a1a0	.An ← .An - 1	1	NO
SUBI .Dn, #Dato	0111100d Dato	.Dn ← .Dn - Dato	2	TODOS (Cy=Borrow)
ADD .D0, .Dn	0111101d	.D0 ← .D0 + .D1	1	TODOS

TABLA 1

MNEMÓNICO	FORMATO DE INSTRUCCIÓN	EXPLICACIÓN	LONG.	FLAGS
JMP.U /Dir	01101100 Dir _H Dir _L	if N=1 then PC _H ← Dir _H PC _L ← Dir _L N ← 0 A _{2H} ↔ [0000H] A _{2L} ↔ [0001H] else excepción 5	3	N = 0
CALL \$desp	01111110 desp _H desp _L	[--A ₂] ← .PC _L [--A ₂] ← .PC _H .PC ← .PC +ca ₂ desp	3	NO
RET	10000000	.PC _H ← [.A ₂ ++] .PC _L ← [.A ₂ ++]	1	NO
RTE	10000010	if N=1 then .SR ← [.A ₂ ++] .PC _H ← [.A ₂ ++] .PC _L ← [.A ₂ ++] if N=0 then .A _{2H} ↔ [0000H] .A _{2L} ↔ [0001H] else excepción 5	1	TODOS
Bcond /Dir	10010c ₂ c ₁ c ₀ Dir _H Dir _L	if cond then PC _H ← Dir _H PC _L ← Dir _L	3	NO
HALT	10000100	if N=1 then HALT else excepción 5	1	NO
RESET	10000101	if N=1 then excepción 0 else excepción 5	1	TODOS
TRAP #0	10000110	Genera la excepción 3	1	NO
TRAP #1	10000111	Genera la excepción 4	1	NO

- TABLA 1 (Continuación) -

d	Dn
0	D0
1	D1

a ₁ a ₀	An
0 0	A0
0 1	A1
1 0	A2

r ₂ r ₁ r ₀	Rn
0 0 0	CCR
0 0 1	A0
0 1 0	A1
0 1 1	D0
1 0 0	D1

Cond	Z	NZ	C	NC	V	NV	T(True)
c ₂ c ₁ c ₀	000	001	010	011	100	101	110

9.4 NOTAS ACERCA DE LA MICROPROGRAMACIÓN

9.4.1 Uso de microsubrutinas

La memoria de control se compone de tres partes:

- La microrrutina de tratamiento de excepciones
- La microrrutina de fetch
- Las microrrutinas que implementan las instrucciones

Estas tres partes tienen un gran número de “porciones de microcódigo” que realizan las mismas funciones, por lo que se deben microprogramar en forma de microsubrutinas. Esto tiene las siguientes ventajas:

- Reduce el número total de microinstrucciones
- Reduce el número de errores de transcripción (o de edición)
- Reduce el número de errores producidos por “despistes” u olvidos
- Reduce el número de modificaciones a realizar durante la depuración

En resumen, simplifica las labores de microprogramación y de depuración de errores. Por otro lado, la labor de identificación de “partes comunes” exige un análisis inicial del juego de instrucciones que lleva a un planteamiento global y más “racional” de la práctica con la consiguiente **simplificación y reducción del tiempo necesario** para su realización.

A continuación se describen algunas funciones que deben microprogramarse como microsubrutinas y utilizarse para la elaboración de la memoria de control:

- Intercambio de punteros de pila
- Comprobación de dirección privilegiada
- Generación de una excepción determinada (vector 5, 6, etc)
- Lectura no privilegiada. Deberá usar la subrutina de comprobación de dirección privilegiada y, en función del resultado, realizar una lectura en memoria o generar una excepción.
- Escritura no privilegiada. Deberá usar la subrutina de comprobación de dirección privilegiada y, en función del resultado, realizar una escritura en memoria o generar una excepción.

El planteamiento concreto de qué debe realizar cada una de estas subrutinas, cuáles son sus parámetros y cómo se pasa cada uno de ellos queda abierto para ser definido por los alumnos como parte de la práctica. Estas microsubrutinas **deben utilizarse obligatoriamente** en la microprogramación de la memoria de control y **deben documentarse** en la memoria escrita.

Además de estas microsubrutinas, hay otras muchas funciones u operaciones que pueden microprogramarse como subrutinas o como zonas de microcódigo común. A modo de ejemplo presentamos una posible microprogramación de las instrucciones LD .D0, [.An] haciendo uso de la microsubrutina LEE_NP que realiza una lectura no privilegiada. Esta subrutina recibe como parámetros de entrada la dirección de memoria sobre la que se debe realizar la lectura, que se pasa sobre el registro de dirección, y la parte alta de dicha dirección, que se pasa sobre el registro TMP. Como parámetro de salida devuelve sobre el registro TMP el dato leído de memoria (en caso de violación de privilegio genera el vector 8 y cede el control a la microrrutina de tratamiento de excepciones).

```

.
. 10(H) LD .D0, [.A0]
.   .D0 <- [.A0]
.
080: rp=.A0, A0H -> TMP, ldADDR;
081: call LEE_NP
082: .D0 <- .TMP
083: jump FETCH
.
. 11(H) LD .D0, [.A0]
.   .D0 <- [A0]
.
088: rp=.A1, A1H -> TMP, ldADDR, jump 081;
.
.
. 12(H) LD .D0, [.A2]
.   .D0 <- [A2]
.
080: rp=.A2, A2H -> TMP, ldADDR, jump 081;
.

```

9.4.2 Operaciones aritméticas

En el juego de instrucciones planteado hay tres instrucciones en las que se deben realizar operaciones aritméticas: ADD, SUBI y CALL.

En las dos primeras es relativamente obvio qué operación debe realizarse y cómo debe microprogramarse en el P8080E. Quizás no sea tan obvio cuál debe ser el tratamiento de los flags.

Sin entrar en más explicaciones, incluimos las microinstrucciones que realizan las operaciones $A + \text{TMP}$ y $A - \text{TMP}$ modificando adecuadamente los flags.

$A + \text{TMP}$: selA=A, TMP_0=1, invTMP = 0, opALU = 0, selCin = 0, invCin = 0, modF, selCy = Cout;

$A - \text{TMP}$: selA=A, TMP_0=1, invTMP = 1, opALU = 0, selCin = 0, invCin = 1, modF, selCy = notCout;

Esto mismo se podría escribir de forma menos detallada de la siguiente forma:

$A + \text{TMP}$: $A + \text{TMP}$, modF, Cy = Cout;

$A - \text{TMP}$: $A + \text{notTMP} + 1$, modF, Cy = notCout;

En la instrucción CALL debe realizarse la suma de un número de 16 bits en binario puro (el valor del contador de programa) y un número de 16 bits en complemento a dos (el desplazamiento). Esta operación de suma debe realizarse en las dos etapas siguientes:

I: $\text{PCL} + \text{despL}$; Cy = Cout;

II: $\text{PCH} + \text{despH} + \text{Cy}$;

9.5 RECOMENDACIONES PARA LA REALIZACIÓN DE LA PRÁCTICA

A lo largo de varios años hemos observado que hay un número importante de alumnos que tienen muchos problemas con la práctica debido a la manera en que llevan a cabo su realización. Por ello incluimos a continuación un conjunto de sugerencias que, si se siguen en su filosofía aunque no sea al pie de la letra, harán la práctica mucho más fácil de realizar correctamente, especialmente la fase de depuración.

En primer lugar es muy importante entender qué hay que hacer y cuál es el propósito de cada uno de los “elementos” de la práctica. Sin una idea clara de lo que hay que hacer se darán palos de ciego que harán muy complicado enderezar la práctica. Por ello es muy importante leer detenidamente el enunciado de la práctica y la documentación del simulador y no empezar a hacer cosas sin haber entendido previamente qué se quiere hacer.

El primer paso que proponemos es familiarizarse con la microprogramación y con el simulador p8080e. Para ello se pueden utilizar las microrrutinas de RESET, FETCH y de la instrucción HALT que figuran en los ejemplos de la documentación del simulador (fig. 18 y 19) y microprogramar las primeras instrucciones del juego, que son las más sencillas y no involucran excepciones (si bien las microrrutinas de RESET, FETCH y HALT habrán de ser reescritas más adelante, la microprogramación de estas instrucciones será definitiva). Por ejemplo, se puede comenzar microprogramando la instrucción LD .Dn, #Dato. Para depurar estas primeras microrrutinas hay que hacer un programa principal consistente en tres o cuatro instrucciones LD .Dn, #Dato. Para comenzar es aconsejable usar el simulador interactivo, ya que permite ejecutar microinstrucciones paso a paso. Una vez que estas instrucciones funcionen satisfactoriamente, se debe usar el simulador no interactivo con la opción /w<ciclos> para comprobar que los accesos a memoria se realizan correctamente. Es aconsejable, también, comenzar a usar en esta fase los ficheros de configuración y resultados.

A continuación proponemos continuar con la microprogramación del resto de las instrucciones sencillas, EXCH_D, EXCH_A, LD .An, #Dato, INC .An, DEC .An, SUBI .Dn, #Dato y ADD .D0, .Dn, depurando exhaustivamente cada grupo de instrucciones antes de comenzar con el siguiente. A medida que se va disponiendo de más instrucciones, puede hacerse programas de prueba más elaborados que permitan detectar en unas instrucciones errores provocados por otras.

Como siguiente paso proponemos plantear las microsubrutinas descritas en el apartado 9.4.1. Concretamente debemos definir qué ha de hacer exactamente cada una de ellas, qué parámetros necesitan y cómo se va a pasar cada uno de ellos. Por ahora nos limitaremos a plantearlas, porque es muy probable que las decisiones tomadas no sean las más adecuadas y que nos hayamos olvidado de algún detalle (el tratamiento correcto de los flags y de los registros del programador es un buen candidato a quedarse en el olvido).

Una vez planteadas las microsubrutinas, procederemos a plantear la microrrutina de fetch y algunas instrucciones más complejas como PUSH .Rn, POP .Rn y JMP.U /Dir. Nuevamente nos limitaremos al planteamiento de las instrucciones usando las microsubrutinas definidas anteriormente y apoyándonos en el ejemplo propuesto en el apartado 9.4.1 para la instrucción LD .Dn, [.An]. Este planteamiento probablemente pondrá de manifiesto algunos problemas con la definición hecha de las microsubrutinas y estaremos en situación de realizar un planteamiento más adecuado a las necesidades del juego de instrucciones y a las limitaciones de la arquitectura del p8080e.

En este punto procederemos a microprogramar las microsubrutinas de apoyo, la microrrutina de tratamiento de excepciones y la microrrutina de fetch. Podemos comenzar con su depuración usando los programas de prueba de las instrucciones ya realizadas, que deben comportarse exactamente igual que antes. Con ello habremos depurado la excepción de RESET, la microsubrutina de comprobación de dirección privilegiada y el ciclo de fetch.

A partir de aquí iremos planteando, microprogramando y depurando el resto de las instrucciones sin tratar de realizar una nueva instrucción antes de comprobar fehacientemente que las anteriores funcionan correctamente, tanto en modo usuario como en modo supervisor, accediendo en ambos casos a posiciones de memoria privilegiada y no privilegiada y generando, en su caso, las excepciones que sean pertinentes. Lo más adecuado es comenzar con las más sencillas, por ejemplo HALT, JMP.U /Dir, LD .Dn, [An], ST .Dn, [.An], TRAP #0, TRAP #1, RESET.

Los programas de prueba deben permitirnos tanto depurar las instrucciones de forma aislada, como todas ellas en conjunto.

9.6 GRUPOS DE PRÁCTICAS

Las prácticas se realizarán en este curso en grupos de, a lo sumo, **dos personas**.

No se admitirán prácticas realizadas por grupos compuestos por más de dos personas ni prácticas realizadas en colaboración por integrantes de dos o más grupos. En ambos casos se entenderá que las prácticas han sido **copiadas**. Así mismo, se entenderán como copiadas aquellas prácticas realizadas con recursos (correcciones) asignados a otros grupos.

9.7 EVALUACIÓN DE LA PRÁCTICA

La calificación de la práctica se realizará a partir de las siguientes notas:

- La nota obtenida en la ejecución de la práctica.
- La nota obtenida en la memoria escrita.
- La nota obtenida en el examen de la práctica.

Durante la realización del examen de la práctica podrá ser necesario consultar la presente documentación, pero no la memoria de la práctica. Dicha documentación no podrá ser compartida durante el examen.

De acuerdo con las normas de la asignatura, para aprobar la práctica deberán aprobarse las tres partes **en la misma convocatoria**. En caso de suspender alguna de ellas, se considerará la práctica suspensa en su totalidad y se deberá realizar una nueva práctica en otra convocatoria: superar las pruebas, entregar la memoria escrita y realizar un nuevo examen.

También de acuerdo con las normas de la asignatura, los alumnos que utilicen alguna corrección de la práctica en una convocatoria se considerarán, a efectos de actas, presentados a dicha convocatoria.

9.8 PLAZOS DE ENTREGA DE LA PRÁCTICA Y FECHAS DE EXAMEN

9.8.1 Convocatoria de febrero (2009)

El plazo de entrega de los ficheros de la práctica para la convocatoria de febrero **comienza el lunes 17 de noviembre de 2008 y termina el miércoles día 21 de enero de 2009 a las 17:30**.

El plazo para entregar la **memoria** de la práctica termina **el viernes día 30 de enero de 2009 a las 21:00**.

Cada grupo de prácticas dispondrá de un máximo de **siete** correcciones de su práctica. De ellas, **dos** se realizarán en las siguientes fechas prefijadas:

- **lunes 15 de diciembre a las 17:30**
- **miércoles 21 de enero a las 17:30**

Las otras **cinco** correcciones se realizarán en las fechas elegidas por el alumno, siempre que estén dentro del plazo de entrega de los ficheros y excluyendo el 15 de diciembre y el 21 de enero.

Los días 15 de diciembre y 21 de enero sólo se realizará una corrección a las 17:30.

Aquellos alumnos que hubieren **aprobado en esta convocatoria** la parte de ejecución de la práctica deberán realizar el examen de la misma que se realizará el **viernes 30 de enero de 2009, a las 15:00**.

9.8.2 Convocatoria EXTRAORDINARIA de junio (2009)

En esta convocatoria sólo se podrán presentar los alumnos repetidores (matriculados en el curso anterior) que lo soliciten en el tiempo y la forma que determine Secretaría de Alumnos (<http://www.fi.upm.es/index.php?pagina=340>).

Las prácticas se podrán realizar en esta convocatoria de forma individual o en grupos de dos alumnos. Aquellos alumnos que se hayan presentado en la convocatoria de febrero podrán usar la práctica entregada en febrero, si bien deberán permanecer en el mismo grupo o bien presentar la práctica de forma individual.

En caso de no haber superado en la convocatoria de febrero todas las pruebas, para la convocatoria de junio **deberán completarla obligatoriamente**. Se recuerda además que en la convocatoria extraordinaria de junio se deben entregar los ficheros de ejecución, la memoria escrita, y presentarse al examen de la práctica, independientemente de haber superado alguna de estas partes en alguna convocatoria anterior.

El plazo de entrega de los ficheros de la práctica para la convocatoria de junio **comienza el miércoles 1 de abril de 2009 y termina el lunes 8 de junio de 2009 a las 17:30**.

El plazo para entregar la **memoria** de la práctica termina **el martes 16 de junio, a las 21:00**.

Cada grupo de prácticas dispondrá de un máximo de **seis** correcciones de su práctica. De ellas, **una** se realizará el **lunes 8 de junio a las 17:30**, y las otras **cinco** correcciones se realizarán en las fechas elegidas por cada grupo, siempre que estén dentro del plazo de entrega de los ficheros y excluyendo el 8 de junio.

El 8 de junio sólo se realizará una corrección a las 17:30.

Aquellos alumnos que hubieren **aprobado en esta convocatoria** la parte de ejecución de la práctica deberán realizar el examen de la misma que se realizará el martes 16 de junio de 2009 a continuación del examen de teoría de la asignatura.

9.8.3 Convocatoria de septiembre (2009)

Las prácticas se podrán realizar en esta convocatoria de forma individual o en grupos de dos alumnos. Aquellos alumnos que se hayan presentado a la convocatoria de febrero o junio podrán usar la práctica entregada en esa convocatoria, si bien deberán permanecer en el mismo grupo o bien presentar la práctica de forma individual.

En caso de no haber superado en la convocatoria anterior todas las pruebas, para la convocatoria de septiembre **deberán completarla obligatoriamente**. Se recuerda además que en la convocatoria extraordinaria de septiembre se deben entregar los ficheros de ejecución, la memoria escrita, y

presentarse al examen de la práctica, independientemente de haber superado alguna de estas partes en alguna convocatoria anterior.

El plazo de entrega de los ficheros de la práctica para la convocatoria de septiembre **comienza el lunes 6 de julio de 2009 y termina el jueves día 10 de septiembre de 2009 a las 17:30.**

El plazo para entregar la **memoria** de la práctica termina **el viernes día 18 de septiembre de 2009 a las 21:00.**

Cada grupo de prácticas dispondrá de un máximo de **seis** correcciones de su práctica. De ellas, **dos** se realizarán en las siguientes fechas prefijadas:

- **viernes 24 de julio a las 17:30**
- **viernes 10 de septiembre a las 17:30**

Las otras **cinco** correcciones se realizarán en las fechas elegidas por cada grupo, siempre que estén dentro del plazo de entrega de los ficheros y excluyendo el 24 de julio y el 10 de septiembre.

Los días 24 de julio y 10 de septiembre sólo se realizará una corrección a las 17:30.

Aquellos alumnos que hubieren aprobado en esta convocatoria la parte de ejecución de la práctica deberán realizar el examen de la misma que se realizará el viernes 18 de septiembre de 2008 a continuación del examen de teoría de la asignatura.

NOTA IMPORTANTE: Si bien está previsto que el sistema de corrección se ponga en funcionamiento todos los días comprendidos entre las fechas señaladas para las tres convocatorias, no será posible garantizar este servicio durante los días no laborables ni durante las vacaciones (navidad, semana santa o agosto). En caso de avería del computador o de cualquier tipo de fallo en el sistema de entrega o corrección, no se responderá del mismo hasta el siguiente día laborable o hasta el fin del periodo de vacaciones.

La entrega de los ficheros de la práctica se entenderá como una **petición de corrección** de dicha práctica. Estas correcciones se realizarán diariamente a las 11:30 y a las 17:30 (salvo las fechas mencionadas). En cada caso se corregirán todas las prácticas que se hayan entregado correctamente y estén pendientes de corrección (es decir, a las 11:30 las entregadas desde las 17:30 del día anterior hasta el momento de la corrección, y a las 17:30 las entregadas entre las 11:30 y las 17:30). Se advierte que esto **no** es una invitación a consumir dos correcciones en el mismo día, sino una forma de facilitar la entrega y recogida de resultados a todos los alumnos, independientemente de su horario habitual de permanencia en la facultad. Es muy recomendable analizar con tiempo los resultados obtenidos, antes de realizar cualquier modificación en el microcódigo y, por lo tanto, antes de realizar una nueva entrega.

El resultado de la corrección podrá consultarse mediante el programa de entrega desde el mismo momento en que **finalice** la corrección de todas las prácticas entregadas, tanto en las fechas prefijadas, como en las elegidas por el alumno.

Se recuerda a los alumnos que éste es un **procedimiento de evaluación** de las prácticas, **no de depuración** del juego de instrucciones. **La depuración es una parte esencial de la práctica y debe ser realizada por los alumnos previamente a la entrega de la práctica.**

La memoria de la práctica se deberá depositar en el buzón del departamento, situado a la derecha de la puerta de entrada al mismo (bloque 4, planta 1).

9.9 RECURSOS ASIGNADOS

Para esta convocatoria se han reservado horas en los PC's del centro de cálculo y en batman. En dichas máquinas se podrá realizar y entregar la práctica, si bien en batman sólo se podrá usar el simulador no interactivo.

El programa de entrega también está disponible en zipi, por lo que en esta máquina se podrá realizar entregas y consultar resultados y noticias.

Por último, también están disponibles para aquellos alumnos que quieran trabajar en su casa todas las versiones del simulador, así como un ejemplo del fichero de configuración y versiones para windows y linux del programa de entrega de la práctica. Se puede acceder a ellos a través de la página WEB de la asignatura:

http://datsi.fi.upm.es/docencia/Estructura.html/U_Control

El sistema de entregas (gestor de prácticas) también está disponible en esa página WEB.

9.10 TUTORÍAS

Las consultas sobre cualquier aspecto de la práctica se realizarán en las horas de clase a tal efecto dedicadas, por *e-mail*, y en las horas de tutorías de los profesores que tienen a su cargo la práctica. El correo electrónico se enviará a cualquiera de las direcciones:

mnieto@fi.upm.es (Manuel Nieto Rodríguez),

mcordoba@fi.upm.es (Maria Luisa Córdoba Cabeza)

La respuesta a este tipo de consultas por e-mail se realizará, siempre que sea posible, diariamente.

Las noticias de interés sobre la práctica se podrán recoger a través de la opción “**Noticias**” del servidor de prácticas. Por consiguiente, **el alumno debe consultar con frecuencia dicha opción.**