ESTRUCTURA
DE COMPUTADORES I

2021-2022

Nombre: Michael javier
Apellidos: Cataño Ortiz
DNI: 41663592N

Correo: onedeath24@gmail.com

INTRODUCCIÓN

Un emulador permite que un ordenador pueda ejecutar un programa escrito para una máquina diferente. En esta práctica final debemos implementar, en lenguaje ensamblador del 68K, un programa que emule la ejecución de programas escritos para una máquina elemental dada. Estos programas deberán estar escritos usando el conjunto de instrucciones de la máquina en cuestión, y el emulador deberá funcionar para cualquier programa que respete dicho conjunto.

La máquina elemental propuesta es la JARVIS, que tanto los registros como su conjunto de instrucciones son de 16 bits. Los diferentes tipos de registros que tiene la Jarvis son:

- B0 y B1: registros de direcciones, que se utilizan en algunas instrucciones para acceder a memoria usando un modo de direccionamiento indexado.
- R2,R3,R4,R5: que son de propósito general y se utilizan en operaciones de tipo ALU, ya sea como operando fuente o como operando destino.
- T6 Y T7: que se utilizan como interfaz con la memoria, además de poder ser empleados en operaciones de tipo ALU como operando.

El conjunto de instrucciones de la máquina JARVIS es la que se ve en la imagen junto con su codificación, la acción que hace cada una y la actualización de los flags tras su ejecución:

Id	Mnemónico	Codificación	Acción	Flags
0	TRA Xa,Xb	00001bbbxaaaxxxx	Xb ← [Xa]	C = n.s.a., Z y N = s.v.Xb
1	ADD Xa,Xb	00010bbbxaaaxxxx	Xb ← [Xb] + [Xa]	C, Z y N = s.v.r.
2	SUB Xa,Xb	00011bbbxaaaxxxx	Xb ← [Xb] - [Xa]	C, Z y N = s.v.r.
3	NAN Xa,Xb	00100bbbxaaaxxxx	Xb ← [Xb] nand [Xa]	C = n.s.a., Z y N = s.v.r.
4	STC #k,Xb	00101bbbkkkkkkkk	Xb ← k (Ext. signo)	C = n.s.a., Z y N = s.v.Xb
5	INC #k,Xb	00110bbbkkkkkkkk	Xb ← [Xb] + k (Ext. Signo)	C, Z y N = s.v.r.
6	LOA M	0100mmmmmmmmxxxx	T6 ← [M]	C = n.s.a., Z y N = s.v.T6
7	LOAX M(Bi),Tj	0101mmmmmmmmijxx	Tj ← [M + [Bi]]	C = n.s.a., Z y N = s.v.Tj
8	STO M	0110mmmmmmmmxxxx	M ← [T6]	n.s.a.
9	STOX Tj,M(Bi)	0111mmmmmmmmijxx	M + [Bi] ← [Tj]	n.s.a.
10	BRI M	1000mmmmmmmmxxxx	PC ← M	n.s.a.
11	BRZ M	1001mmmmmmmmxxxx	Si Z = 1, PC ← M	n.s.a.
12	BRN M	1010mmmmmmmmxxxx	Si N = 1, PC ← M	n.s.a.
13	STP	11xxxxxxxxxxxxxx	Detiene la máquina	n.s.a.

EXPLICACIÓN DE LA SOLUCIÓN:

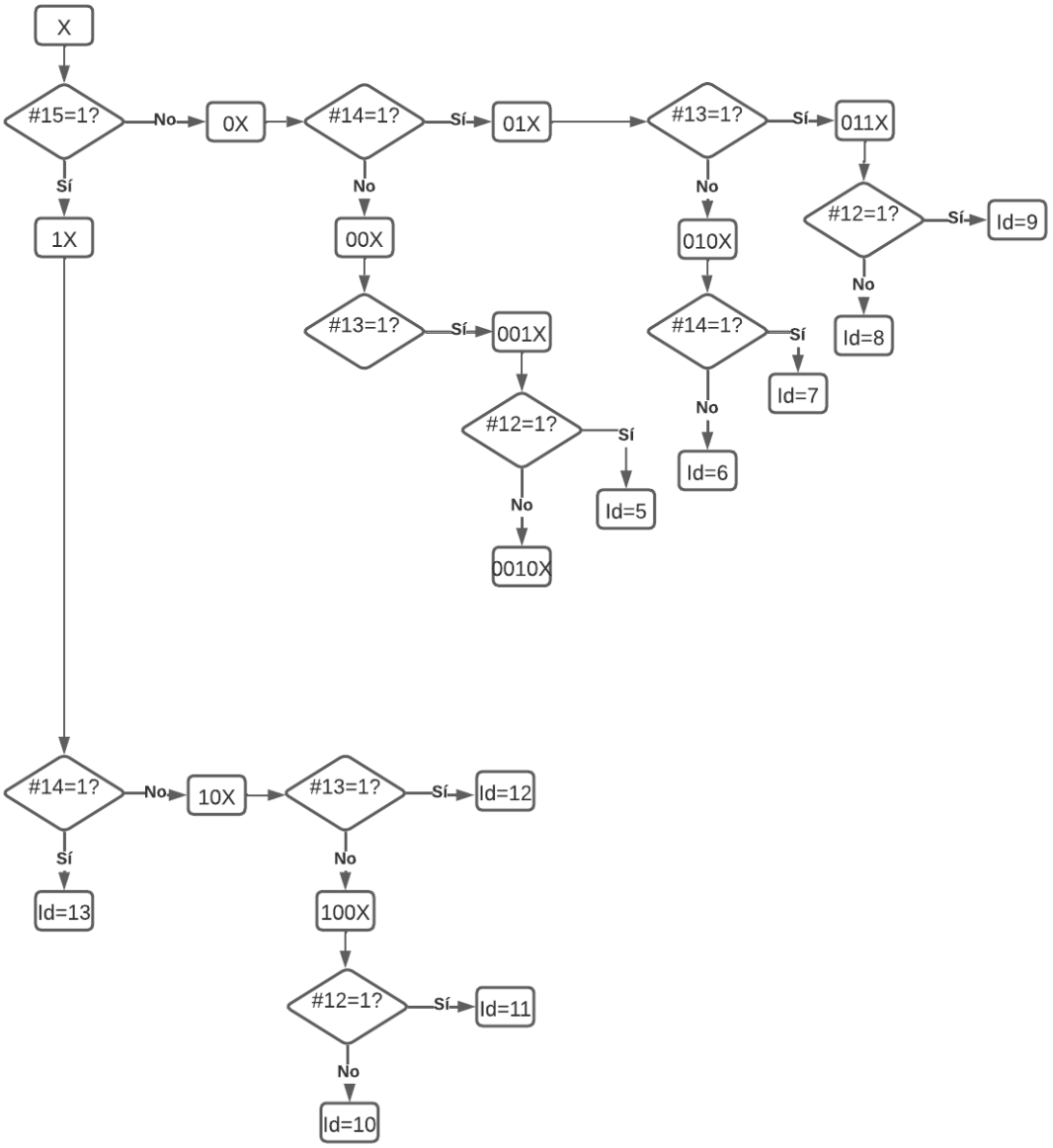
Ahora se procederá a explicar cómo se implementó la solución para resolver el problema propuesto. Para hacer esto, explicaremos la implementación de cada fase en la ejecución de un programa, que es: fase Fetch,fase Decode,fase Ejecución.

Para poder saber que tipo de instrucción era, tenemos que obtener el Id de la instrucción, que es lo que aparece al lado izquierdo de cada instrucción en la imagen anterior, de manera que tenemos que saber primero qué tipo de instrucción se trata y cuando lo sepamos guardamos su Id en un registro determinado para así poder saltar a la fase de Ejecución, con esto podemos decir que se implementa una subrutina de librería, la cual hace el proceso de codificación de cada instrucción, y que cuando acabe nos dé el resultado en un a posición de la peli, para que después de haber vuelto de la subrutina saquemos el resultado y procedamos a saltar a la fase de ejecución de la instrucción.

Una vez que ya tengamos la fase de Decode, nos quedaría la fase de Fetch, que se implementa de una manera muy fácil, en esta fase lo que queremos es que movemos la instrucción en el EIR para hacer su decodificación, y nos posicionamos en la siguiente instrucción para cuando volvamos otra vez al Fetch, para poder hacer esto, la JARVIS tiene un contador de programa, el cual es EPC y además de esto tenemos un vector EMEM el cual contiene todas las instrucciones que hace nuestro programa emulado, y para poder acceder a la siguiente instrucción tendríamos que recorrer el vector EMEM con el EPC, pero la máquina JARVIS va de una manera distinta, ya que cuando avanzamos en cada instrucción, avanzamos en 1 en las direcciones, es decir que va de la siguiente manera: 0→1→2→3→4→5→…, esto en sí es una dirección emulada , por lo tanto antes de mover la instrucción a EIR tenemos que calcular la dirección real en el 68K para así poder hacer un modo de direccionamiento indexado y ya poder moverlo a EIR y después sumar 1 en EPC para apuntar a la siguiente instrucción. Para poder calcular la dirección real, hacemos una multiplicación por 2 del EPC, ya que cada instrucción son WORDS.

Por último queda la fase de ejecución, la cual solo se trata de implementar las funciones o operaciones que tiene que hacer la máquina en cada instrucción, claramente usando uso de varias subrutinas para poder conseguir las variables que necesitamos en la ejecución de la instrucción, las cuales son: M,K,Xa,Xb y los flags ZCN y con esto ya estaría explicado cada parte hecha de la solución, solo queda decir que para que se acabe la ejecución del programa, sería cuando se encuentre la instruccion STP.

DIAGRAMA DE DECODIFICACIÓN:



Debido a que la pagina no me dejo poner mas cajas, por lo que no complemente bien el programa, pero a continuación pondré por escrito los últimos Id que faltan:

0010X→si X=1→Id=4
0010X→si X=0→Id=3
000X→si X=1→0001X→si X=1→Id=2
0001X→si X=0→Id=1
000X→si X=0→Id=0

Con esto, ya estaría descrito la subrutina de decodificación, cabe decir que la subrutina tiene dos parámetros que se usan, uno de ellos es EIR y otro en el que está el resultado de la decodificación, también se salvaguarda un registro D0, que es el que nos ayuda con el proceso de decodificación.

TABLAS DE SUBRUTINAS:

NOMBRE RUTINA	TIPO RUTINA	INTER. ENTRADA	INTER.SALIDA
OBTaaa	usuario	D2	D2
OBTbbb	usuario	D0	D0
OBTXa	usuario	D2	A5
OBTXb	usuario	D0	A4
OBTm	usuario	D0	D0
OBTk	usuario	D0	D0
OBTBi	usuario	D0	A4
OBTTj	usuario	D0	A4
OBTFLAGZ	usuario	D0	NO USA
OBTFLAGN	usuario	D0	NO USA
OBTz	usuario	D0	D0
OBTn	usuario	D0	D0

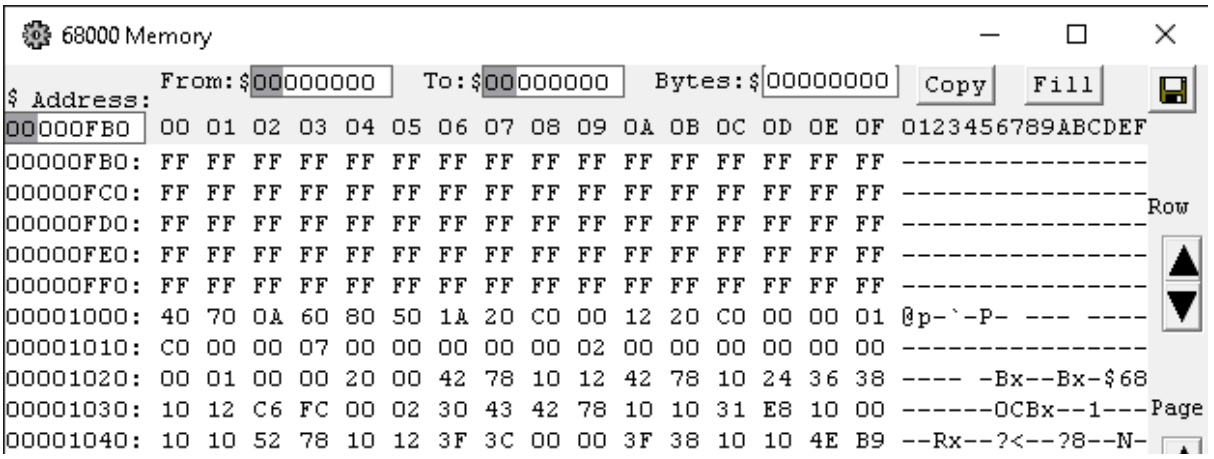
Tabla de Registros del 68K:

REGISTRO	USO
D0	Se usa para obtener bbb, para obtener m, para obtener k, para saber el valor del flag Z y N. También se usa para saber el tipo de registro Xb y para actualizar los flags Z y N. También se usa para la decodificación de IR.
D2	Se usa para obtener aaa y para saber el tipo de registro Xa
A4	Se usa para guardar las direcciones de los registros Xb, pero también para los Bi y Tj.
A5	se usa para guardar las direcciones de los registros Xa

Pruebas:

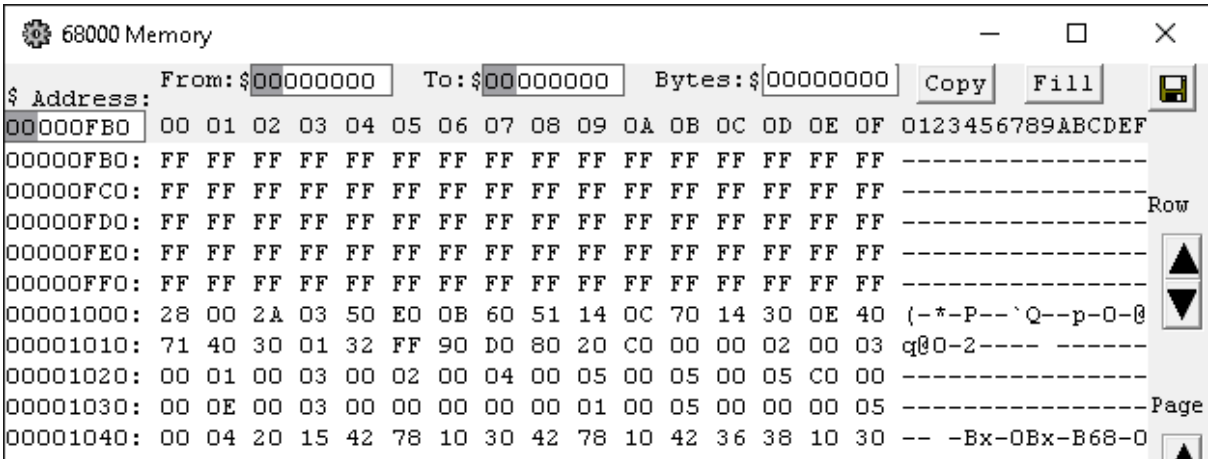
En este apartado se pondrán las distintas pruebas hechas para verificar el correcto funcionamiento de la máquina JARVIS.

1. Como requisito mínimo, la maquina tiene que hacer la correcta ejecución del siguiente programa:
   EMEM: DC.W $4070,$0A60,$8050,$1A20,$C000,$1220,$C000,$0001



   Tras la ejecución del programa, en la posición de memoria del 68K correspondiente a ER2 (en este caso, @1018Hex) debería contener el valor 2 Dec = 0002 Hex.

2. Como segunda prueba, se hará la ejecución del siguiente programa:
   EMEM: DC.W
   $2800,$2A03,$50E0,$0B60,$5114,$0C70,$1430,$0E40,$7140,$3001,$32FF,$90D0
   DC.W $8020,$C000,$0002,$0003,$0001,$0003,$0002,$0004,$0000,$0000,$0000



   Tras la ejecución del programa, en la posición de memoria del 68K correspondiente a C(en este caso, @1028Hex,@102AHex,@102CHex) deberían contener el valor 5 DEC = 0005 Hex

CONCLUSIONES:

Esta práctica ha sido bastante buena de hacer, y no solo he podido llegar a comprender mucho el entorno de programación a bajo nivel, sino que he podido entender muchas cosas con respectos a los registros y las operaciones del 68k, ha sido una práctica bastante entretenida de hacer, un poco larga pero el esfuerzo y la dedicación que se ha hecho por este trabajo no es comparable con los conocimientos que he podido a adquirir mientras hacía mi trabajo. También hay que decir que mi compañero no me ha ayudado en nada, así que he tenido que hacer esta práctica solo, así que ha sido bastante duro.

Solo queda decir que ha sido gratificante poder hacerla yo mismo y conseguir que hiciera lo que tenía que hacer, con esto dicho, doy por terminado las conclusiones.

CÓDIGO FUENTE:

```
START:
   CLR.W EPC
   CLR.W ESR
FETCH:
   ;--- IFETCH: INICIO FETCH

      ;*** En esta seccion debeis introducir el codigo necesario para cargar
      ;*** en el EIR la siguiente instruccion a ejecutar, indicada por el EPC,
            ;*** y dejar listo el EPC para que apunte a la siguiente instruccion

            ; ESCRIBID VUESTRO CODIGO AQUI
            MOVE.W EPC,D3
            MULU #2,D3
            MOVE.W D3,A0
            CLR EIR
            MOVE.W EMEM(A0),EIR
            ADD.W #1,EPC

   ;--- FFETCH: FIN FETCH


   ;--- IBRDECOD: INICIO SALTO A DECOD
      ;*** En esta seccion debeis preparar la pila para llamar a la subrutina
      ;*** DECOD, llamar a la subrutina, y vaciar la pila correctamente,
      ;*** almacenando el resultado de la decodificacion en D1

         ; ESCRIBID VUESTRO CODIGO AQUI
         MOVE.W #0,-(SP)
```

```
        MOVE.W EIR,-(SP)
        JSR DECOD
        ADD.W #2,SP
        MOVE.W (SP)+,D1
   ;--- FBRDECOD: FIN SALTO A DECOD


   ;--- IBREXEC: INICIO SALTO A FASE DE EJECUCION
      ;*** Esta seccion se usa para saltar a la fase de ejecucion
      ;*** NO HACE FALTA MODIFICARLA
   MULU #6,D1
   MOVEA.L D1,A1
   JMP JMPLIST(A1)
JMPLIST:
   JMP ETRA
   JMP EADD
   JMP ESUB
   JMP ENAN
   JMP ESTC
   JMP EINC
   JMP ELOA
   JMP ELOAX
   JMP ESTO
   JMP ESTOX
   JMP EBRI
   JMP EBRZ
   JMP EBRN
   JMP ESTP
   ;--- FBREXEC: FIN SALTO A FASE DE EJECUCION


   ;--- IEXEC: INICIO EJECUCION
      ;*** En esta seccion debeis implementar la ejecucion de cada einstr.

        ; ESCRIBID EN CADA ETIQUETA LA FASE DE EJECUCION DE CADA
INSTRUCCION
ETRA:
   MOVE.W EIR,D0    ;mover el EIR hacia D0
   JSR OBTbbb       ;obtengo el registro Xb
   JSR OBTXb        ;guardo la direccion del registro Xb en A4
   MOVE.W EIR,D2    ;mover el EIR hacia D2 para conseguir Xa
   JSR OBTaaa       ;obtengo el registro Xa
   JSR OBTXa        ;guardo la direccion del registro Xa en A5
   MOVE.W (A5),(A4) ;muevo el registro Xa hacia el registro Xb
   CLR D0           ;limpio el registro D0 para poder usarlo
   MOVE.W (A4),D0   ;muevo el contenido de Xb hacia D0
   JSR OBTFLAGZ     ;actualizo el eflag Z
   CLR D0           ;limpio D0 para poder usarlo
```

```
    MOVE.W (A4),D0   ;muevo el contenido de Xb hacia D0
    JSR OBTFLAGN     ;actualizo el eflag N
    CLR D0           ;apartir de aqui limpiamos los registros usados
    CLR D2
    MOVE.L #0,A4
    MOVE.L #0,A5
    JMP FETCH        ;volvemos al fetch
EADD:
    MOVE.W EIR,D0    ;mover el EIR hacia D0
    JSR OBTbbb       ;obtengo el registro Xb
    JSR OBTXb        ;guardo la direccion del registro Xb en A4
    MOVE.W EIR,D2    ;mover el EIR hacia D2 para conseguir Xa
    JSR OBTaaa       ;obtengo el registro Xa
    JSR OBTXa        ;guardo la direccion del registro Xa en A5
    MOVE.W (A5),D0   ;mover Xa hacia D0
    ADD.W D0,(A4)    ;sumar Xb + Xa y dejo el resultado en Xb
    MOVE.W SR,ESR    ;actualizamos los eflags despues de la operacion suma
    CLR D0           ;limpio todos los registros usados
    CLR D2
    MOVE.L #0,A4
    MOVE.L #0,A5
    JMP FETCH
ESUB:
    MOVE.W EIR,D0    ;mover el EIR hacia D0
    JSR OBTbbb       ;obtengo el registro Xb
    JSR OBTXb        ;guardo la direccion del registro Xb en A4
    MOVE.W EIR,D2    ;mover el EIR hacia D2 para conseguir Xa
    JSR OBTaaa       ;obtengo el registro Xa
    JSR OBTXa        ;guardo la direccion del registro Xa en A5
    MOVE.W (A5),D0   ;mover Xa hacia D0
    NOT D0           ;negar D0, es decir Xa
    ADD.W #1,D0      ;añadir 1 a Xa
    ADD.W D0,(A4)    ;sumar Xb + (Xa(negado)+1) y dejo el resultado en Xb
    MOVE.W SR,ESR    ;actualizamos los eflags despues de la operacion resta
    CLR D0           ;limpio todos los registros usados
    CLR D2
    MOVE.L #0,A4
    MOVE.L #0,A5
    JMP FETCH
ENAN:
    MOVE.W EIR,D0     ;mover el EIR hacia D0
    JSR OBTbbb        ;obtengo el registro Xb
    JSR OBTXb         ;guardo la direccion del registro Xb en A4
    MOVE.W EIR,D2     ;mover el EIR hacia D2 para conseguir Xa
    JSR OBTaaa        ;obtengo el registro Xa
    JSR OBTXa         ;guardo la direccion del registro Xa en A5
    MOVE.W (A5),D0    ;mover Xa hacia D0
    NOT D0            ;negar D0, es decir negar Xa
```

```
    NOT (A4)          ;negar el (A4), es decir negar Xb
    OR.W D0,(A4)      ;obtengo la nand con una OR y dejo el resultado en Xb
    MOVE.W SR,ESR     ;actualizo los eflags despues de la operacion nand
    CLR D0            ;limpio todos los registros usados
    CLR D2
    MOVE.L #0,A4
    MOVE.L #0,A5
    JMP FETCH         ;vuelvo al fetch
ESTC:
    MOVE.W EIR,D0     ;mover el EIR hacia D0
    JSR OBTbbb        ;obtengo el registro Xb
    JSR OBTXb         ;guardo la direccion del registro Xb en A4
    CLR D0            ;limpio D0
    MOVE.W EIR,D0     ;mover el EIR hacia D0 para obtener k
    JSR OBTk          ;obtenemos k con estension de signo
    MOVE.W D0,(A4)    ;mover k hacia Xb
    CLR D0            ;limpio el registro D0 para poder usarlo
    MOVE.W (A4),D0    ;muevo el contenido de Xb hacia D0
    JSR OBTFLAGZ      ;actualizo el eflag Z
    CLR D0            ;limpio D0 para poder usarlo
    MOVE.W (A4),D0    ;muevo el contenido de Xb hacia D0
    JSR OBTFLAGN      ;actualizo el eflag N
    CLR D0            ;limpio todos los registros usados
    MOVE.L #0,A4
    JMP FETCH         ;vuelvo al fetch
EINC:
    MOVE.W EIR,D0     ;muevo EIR hacia D0
    JSR OBTbbb        ;obtengo el registro Xb
    JSR OBTXb         ;guardo la direccion del registro Xb en A4
    MOVE.W EIR,D0     ;muevo EIR hacia D0 para obtener k
    JSR OBTk          ;obtengo k en D0
    ADD.W D0,(A4)     ;hago la suma Xb<--Xb+k y lo dejo en Xb
    MOVE.W SR,ESR     ;actualizo los eflags
    CLR D0            ;limpio los registros usados
    MOVE.L #0,A4
    JMP FETCH         ;vuelvo al fetch
ELOA:
    MOVE.W EIR,D0     ;muevo EIR hacia D0
    JSR OBTm          ;obtengo m y dejo el resultado en D0
    MULU #2,D0        ;obtengo la direccion real
    MOVE.W D0,A4      ;muevo la direccion real a A4
    MOVE.W EMEM(A4),ET6   ;muevo el contenido de esa direccion real a ET6
    CLR D0            ;limpio D0 para obtener el eflag Z
    MOVE.W ET6,D0     ;muevo el contenido de ET6 hacia D0
    JSR OBTFLAGZ      ;obtengo el eflag Z
    CLR D0            ;limpio D0 para obtener el eflag N
    MOVE.W ET6,D0     ;muevo el contenido de ET6 hacia D0
    JSR OBTFLAGN      ;obtengo el eflag N
```

```
   CLR D0          ;limpio los registros usados
   MOVE.L #0,A4
   JMP FETCH       ;vuelvo al fetch
ELOAX:
   MOVE.W EIR,D0   ;muevo el contenido de EIR hacia D0
   JSR OBTm        ;obtengo m y la dejo en D0
   MOVE.W D0,A5    ;muevo m hacia el registro A5
   CLR D0          ;limpio D0
   MOVE.W EIR,D0   ;muevo EIR hacia D0 para obtener Bi
   JSR OBTBi       ;obtengo Bi y dejo su direccion en A4
   ADD.W (A4),A5   ;hago una suma de tal forma: A5<--M+Bi y lo dejo en A5
   MOVE.W A5,D0    ;muevo el contenido de A5 hacia D0
   MULU #2,D0      ;calculo la direccion real
   MOVE.W D0,A5    ;muevo la direccion real hacia A5
   CLR D0          ;limpio D0
   MOVE.L #0,A4    ;limpio A4
   MOVE.W EIR,D0   ;muevo el contenido de EIR hacia D0
   JSR OBTTj       ;obtengo el registro Tj y lo dejo en A4
   MOVE.W EMEM(A5),(A4)  ;muevo el contenido de la direccion real a Tj
   CLR D0          ;limpio D0
   MOVE.W (A4),D0  ;muevo el contenido de Tj hacia D0
   JSR OBTFLAGZ    ;actualizao el eflag Z
   CLR D0          ;limpio D0
   MOVE.W (A4),D0  ;muevo Tj hacia D0
   JSR OBTFLAGN    ;actualizo el eflag N
   CLR D0          ;limpio los registros
   MOVE.L #0,A4
   MOVE.L #0,A5
   JMP FETCH       ;vuelvo al fetch
ESTO:
   MOVE.W EIR,D0   ;muevo EIR hacia D0
   JSR OBTm        ;obtengo M y dejo su resultado en D0
   MULU #2,D0      ;obtengo la direccion real
   MOVE.W D0,A4    ;muevo la direccion real hacia A4
   MOVE.W ET6,EMEM(A4)   ;muevo el contenido de ET6 hacia la direccion real
   CLR D0          ;limpio los registros
   MOVE.L #0,A4
   JMP FETCH       ;vuelvo al fetch
ESTOX:
   MOVE.W EIR,D0   ;muevo EIR hacia D0
   JSR OBTm        ;obtengo M y dejo su resultado en D0
   MOVE.W D0,D2    ;muevo D0 hacia D2 ya que necesito el registro D0
   CLR D0          :limpio D0
   MOVE.W EIR,D0   ;muevo EIR hacia D0
   JSR OBTBi       ;obtengo el eregistro Bi y dejo su direccion en A4
   ADD.W (A4),D2   ;hago la suma de tal manera: D2<--M+Bi y lo dejo en D2
   MULU #2,D2      ;calculo la direccion real
   MOVE.W D2,A5    ;muevo la direccion real hacia A5
```

```
   CLR D0           ;limpio D0
   MOVE.L #0,A4     ;limpio A4
   MOVE.W EIR,D0    ;muevo EIR hacia D0
   JSR OBTTj        ;obtengo Tj y lo dejo en A4
   MOVE.W (A4),EMEM(A5)  ;muevo el contenido de Tj hacia la direccion real
   CLR D0           ;limpio los registros
   CLR D2
   MOVE.L #0,A4
   MOVE.L #0,A5
   JMP FETCH        ;vuelvo al fetch
EBRI:
   MOVE.W EIR,D0    ;muevo EIR hacia D0
   JSR OBTm         ;obtengo m y la dejo en D0
   MOVE.W D0,EPC    ;muevo m hacia el EPC
   CLR D0           ;limpio los registros
   MOVE.W #0,A4
   JMP FETCH        ;vuelvo al fetch
EBRZ:
   MOVE.W EIR,D0    ;muevo EIR hacia D0
   JSR OBTm         ;obtengo m y la dejo en D0
   MOVE.W D0,A4     ;muevo m hacia A4
   CLR D0           ;limpio D0
   MOVE.W ESR,D0    ;muevo el contenido de ESR hacia D0
   JSR OBTz         ;obtengo el flag z y lo dejo en D0
   CMP.W #4,D0      ;compruebo si hay que saltar
   BEQ SALTARSIZ    ;saltare si el eflag Z es 1
   JMP FINEBRZ      ;si es 0 no hare nada
   SALTARSIZ:
   MOVE.W A4,EPC    ;muevo la m hacia el EPC
   FINEBRZ:
   JMP FETCH        ;vuelvo al fetch
EBRN:
   MOVE.W EIR,D0    ;muevo EIR hacia D0
   JSR OBTm         ;obtengo m y la dejo en D0
   MOVE.W D0,A4     ;muevo m hacia A4
   CLR D0           ;limpio D0
   MOVE.W ESR,D0    ;muevo el cotenido de ESR hacia D0
   JSR OBTn         ;obtengo el flag n y lo dejo en D0
   CMP.W #1,D0      ;compruebo se hay que saltar
   BEQ SALTARSIN    ;saltare si el eflag n es 1
   JMP FINEBRN      ;si es 0 no hare nada
   SALTARSIN:
   MOVE.W A4,EPC    ;muevo la m hacia el EPC
   FINEBRN:
   JMP FETCH        ;vuelvo al fetch
ESTP:
   SIMHALT          ;para la maquina
   ;--- FEXEC: FIN EJECUCION
```

```
;--- ISUBR: INICIO SUBRUTINAS
    ;*** Aqui debeis incluir las subrutinas que necesite vuestra solucion
    ;*** SALVO DECOD, que va en la siguiente seccion

          ; ESCRIBID VUESTRO CODIGO AQUI
          OBTbbb:
    AND.W #%0000011100000000,D0
    LSR.W #8,D0
    RTS

    OBTaaa:
    AND.W #%0000000001110000,D2  ;OBTENGO aaa
    LSR.W #4,D2
    RTS

    OBTXb:
    CMP.W #0,D0  ;compruebo si es B0
    BEQ B0
    CMP.W #1,D0
    BEQ B1
    CMP.W #2,D0
    BEQ R2
    CMP.W #3,D0
    BEQ R3
    CMP.W #4,D0
    BEQ R4
    CMP.W #5,D0
    BEQ R5
    CMP.W #6,D0
    BEQ T6
    CMP.W #7,D0
    BEQ T7
    B0:
    LEA.L EB0,A4
    JMP FINOBTXb
    B1:
    LEA.L EB1,A4
    JMP FINOBTXb
    R2:
    LEA.L ER2,A4
    JMP FINOBTXb
    R3:
    LEA.L ER3,A4
    JMP FINOBTXb
    R4:
    LEA.L ER4,A4
```

```
JMP FINOBTXb
R5:
LEA.L ER5,A4
JMP FINOBTXb
T6:
LEA.L ET6,A4
JMP FINOBTXb
T7:
LEA.L ET7,A4
JMP FINOBTXb

FINOBTXb:
RTS

OBTXa:
CMP.W #0,D2  ;compruebo si es B1
BEQ OB0
CMP.W #1,D2
BEQ OB1
CMP.W #2,D2
BEQ OR2
CMP.W #3,D2
BEQ OR3
CMP.W #4,D2
BEQ OR4
CMP.W #5,D2
BEQ OR5
CMP.W #6,D2
BEQ OT6
CMP.W #7,D2
BEQ OT7
OB0:
LEA.L EB0,A5
JMP FINOBTXa
OB1:
LEA.L EB1,A5
JMP FINOBTXa
OR2:
LEA.L ER2,A5
JMP FINOBTXa
OR3:
LEA.L ER3,A5
JMP FINOBTXa
OR4:
LEA.L ER4,A5
JMP FINOBTXa
OR5:
LEA.L ER5,A5
```

```
JMP FINOBTXa
OT6:
LEA.L ET6,A5
JMP FINOBTXa
OT7:
LEA.L ET7,A5
JMP FINOBTXa

FINOBTXa:
RTS

OBTk:
AND.W #%0000000011111111,D0
EXT.W D0
RTS

OBTm:
AND.W #%0000111111110000,D0
LSR.W #4,D0
RTS

OBTBi:
AND.W #%0000000000001000,D0
CMP #8,D0
BEQ OBi
LEA.L EB0,A4
JMP FINOBTBi
OBi:
LEA.L EB1,A4
FINOBTBi:
RTS

OBTTj:
AND.W #%0000000000000100,D0
CMP #4,D0
BEQ OTj
LEA.L ET6,A4
JMP FINOBTBi
OTj:
LEA.L ET7,A4
FINOBTTj:
RTS

OBTz:
AND.W #%0000000000000100,D0
RTS

OBTn:
```

```
        AND.W #%0000000000000001,D0
        RTS

        OBTFLAGZ:
        CMP.W #0,D0
        BEQ SET
        BCLR.B #2,ESR
        JMP FINOBTZ
        SET:
        BSET.B #2,ESR
        FINOBTZ:
        RTS

        OBTFLAGN:
        BTST.L #15,D0
        BEQ NON
        BSET.B #0,ESR
        JMP FINOBTN
        NON:
        BCLR.B #0,ESR
        FINOBTN:
        RTS
    ;--- FSUBR: FIN SUBRUTINAS


    ;--- IDECOD: INICIO DECOD
        ;*** Tras la etiqueta DECOD, debeis implementar la subrutina de
        ;*** decodificacion, que debera ser de libreria, siguiendo la interfaz
        ;*** especificada en el enunciado
DECOD:
            ; ESCRIBID VUESTRO CODIGO AQUI
          MOVE.L D0,-(SP)
          MOVE.W 8(SP),D0
    BTST.L #15,D0
    BNE INSFIN
    BTST.L #14,D0
    BNE INSMID
    BTST.L #13,D0
    BNE INSINI2
    BTST.L #12,D0
    BNE INSIN
    MOVE.W #0,10(SP)
    JMP AVANZAR

INSIN:
    BTST.L #11,D0
    BNE SUB
    MOVE.W #1,10(SP)
```

```
    JMP AVANZAR
SUB:
    MOVE.W #2,10(SP)
    JMP AVANZAR
INSINI2:
    BTST.L #12,D0
    BNE INC
    BTST.L #11,D0
    BNE STC
    MOVE.W #3,10(SP)
    JMP AVANZAR
STC:
    MOVE.W #4,10(SP)
    JMP AVANZAR
INC:
    MOVE.W #5,10(SP)
    JMP AVANZAR
INSMID:
    BTST.L #13,D0
    BNE INSMID2
    BTST.L #12,D0
    BNE LOAX
    MOVE.W #6,10(SP)
    JMP AVANZAR
LOAX:
    MOVE.W #7,10(SP)
    JMP AVANZAR
INSMID2:
    BTST.L #12,D0
    BNE STOX
    MOVE.W #8,10(SP)
    JMP AVANZAR
STOX:
    MOVE.W #9,10(SP)
    JMP AVANZAR
INSFIN:
    BTST.L #14,D0
    BNE STP
    BTST.L #13,D0
    BNE BRN
    BTST.L #12,D0
    BNE BRZ
    MOVE.W #10,10(SP)
    JMP AVANZAR
STP:
    MOVE.W #13,10(SP)
    JMP AVANZAR
BRN:
```

```
    MOVE.W #12,10(SP)
    JMP AVANZAR
BRZ:
    MOVE.W #11,10(SP)
    JMP AVANZAR

AVANZAR:
    MOVE.L (SP)+,D0
    RTS
    ;--- FDECOD: FIN DECOD

    END    START
```