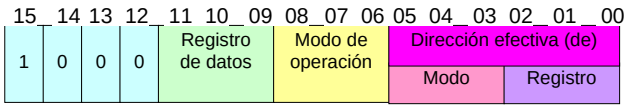
1994. Febrero. Primera semana.

Genere el código máquina correspondiente a la instrucción del MC68000 `OR.W  #$5B38,D6`.

Solución:

Según el apéndice B de las UUDD (página 519), la instrucción OR tiene este formato:



Campo del modo de operación:	Byte	Palabra	Palabra larga	Operación
	000	001	010	(de) OR (Dn) →Dn
	100	101	110	(Dn) OR (de) → de

Vamos a rellenar los diferentes campos de la instrucción:

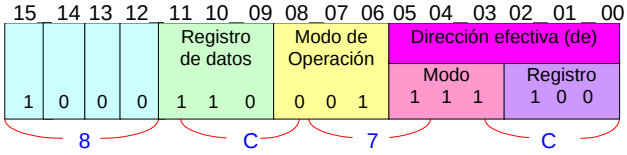
1.- Código de operación.
Está determinado para la instrucción OR. Es 1000

2.- Registro de datos.
Se trata del D6, por tanto, el contenido de esta campo es 6 ≡ %110

3.- Modo de operación
Se trabaja con palabras; y la operación es (de) OR (Dn) →Dn; por consiguiente el contenido de este campo es 001

4.- Par Modo-Registro del operando inmediato
Según la tabla 15.3 (página 379), el contenido debe ser 111 y 100; respectivamente.

Ahora los reunimos todos y los compactamos en hexadecimal:



La primera palabra de las que componen la instrucción es 8C7C. La segunda palabra está a continuación; y su valor es el dato inmediato: 5B38. Por tanto, el código completo máquina de la instrucción es 8C7C5B38.

Podemos comprobar la solución. Utilizaremos para ello el simulador Sim68k.Exe:

1.- Lo arrancamos escribiendo su nombre. Nos aparece el mensaje de inicio y el prompt:

```
Copyright (C) Livadas and Ward, 1992. Author Wayne Wolf
Version 2.3
SIM68000 2.3>
```

2.- En algún lugar de la memoria del simulador (por ejemplo en la $10), almacenamos el código máquina calculado compactado en hexadecimal. Para ello, entablamos el siguiente diálogo :

```
SIM68000 2.3>MM 10;L
000010 00000000 ? 8C7C5B38
000010 00000000 ?.
```

3.- Desensamblamos el contenido de dicha posición de memoria

```
SIM68000 2.3>MD 10 4;DI
000010 8C7C 5B38  OR.W #$53B8,D6
```

1994. Septiembre.

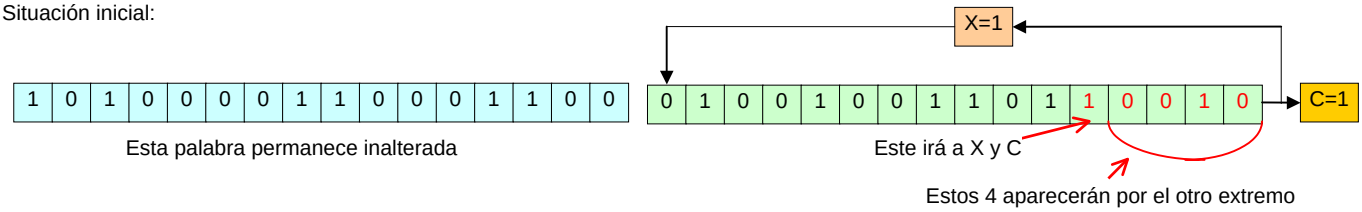
Se dispone de un computador basado en el microprocesador MC68000. En un momento dado, la situación es esta:

- Los bits C y X del registro de códigos de condición están ambos puestos a 1.
- El contenido del registro D1 es (D1) = $A18C49B2
- El contenido del registro D2 es (D2) = $00000005
- La instrucción contenida en la posición apuntada por el contador de programa es `ROXR.W D2,D1`

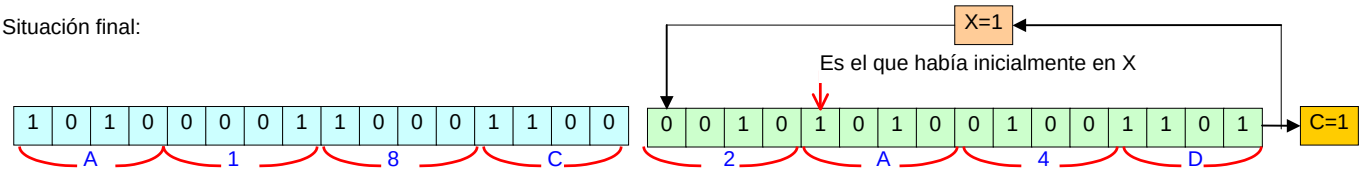
¿ Cuál será el nuevo contenido de D1 y de los indicadores C y X tras ejecutar dicha instrucción ?.

Solución:

Situación inicial:



Situación final:



1995. Febrero. Primera semana.

Se tiene el siguiente fragmento de código en ensamblador del MC68000, donde % y $ representan octal, binario y hexadecimal respectivamente.

```
            ORG   @100
DATO1       EQU   *+%1010
ZONA1       DS.B  $BB
            EVEN
DATO2       DC.L  50
DATO3       DC.W  DATO3-ZONA1+DATO1
```

Indique cuál será el contenido de la dirección de memoria apuntada por Dato3, después del ensamblado.

Solución:

Las explicaciones están en los campos de comentarios de este fichero generado por el ensamblador cruzado:

```
MC68000 Cross Assembler
Copyright (C) Stephen Croll, 1991.  Author: Stephen Croll
Version 2.00 beta 1.02

00000040                 1          ORG     @100               ; El origen está en @100=$40=64
0000004A                 2 DATO1    EQU     *+%1010            ; DATO1=(PC)+%1010=64+10=74
00000040                 3 ZONA1    DS.B    $BB                ; En ZONA1=$40 se reservan $BB=187
                                                                   ; bytes, sin inicializar
000000FB                 4          CNOP    0,2                ; Como $FB es impar, avanza un lugar
000000FC  00000032       5 DATO2    DC.L    50                 ; En DATO2=$FC se almacena el número
                                                                   ; 50, el cual ocupa 4 bytes

00000100  010A           6 DATO3    DC.W    DATO3-ZONA1+DATO1  ; En DATO3=$100 se almacena
                                                               ; $100-$40+74 = 256-64+74=266=$10A
                         7

00000102  307C 0100      8          movea.w #DATO3,A0
00000106  3010           9          move.w  (A0),D0
00000108  1E3C 00E4     10          move.b  #228,D7
0000010C  4E4E          11          trap    #14
0000010E                12          end

No errors detected.
```

1995. Febrero. Segunda semana.

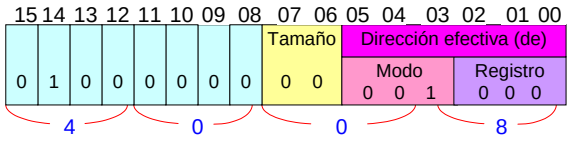
Indique cuál es la instrucción del MC68000 que corresponde al código 4008.

Solución:

Consultando el apéndice C, vemos que las instrucciones en las que su cuatro bits más significativos son %0100=$4, son:
CHK, CLR, EXT, JMP, JSR, LEA, MOVE, MOVEM, NBCD, NEG, NEGX, NOP, NOT, PEA, RTE, RTR, RTS, STOP, SWAP, TRAP, TRAPV y TST .

Excepto NEGX, todas las demás tienen algún 1 en los bits 4 a 11. Nos limitaremos a comprobar si puede ser NEGX, descartando las demás.

Su formato es este:



Campo del modo de operación:	Byte	Palabra	Palabra larga
	00	01	10

Así, el tamaño del dato es de un byte.
Según la tabla 15.3 (página 379), el contenido 001 y 000 corresponde a un direccionamiento mediante el registro de direcciones A0.
Por todo esto, la instrucción es *NEGX.B A0*
Podemos comprobar la solución. Utilizaremos para ello el simulador Sim68k.Exe:

1.- En la posición de la memoria $1000 almacenamos 4008:

```
SIM68000 2.3 > MM 1000;W
001000 0000 ? 4008
001002 0000 ?.
```

2.- Desensamblamos el contenido de dicha posición de memoria

```
SIM68000 2.3 > MD 1000 2;DI
00001000 4008 NEGX.B A0
```

En realidad, el código 4008 no corresponde a ninguna instrucción legal. Javier Roca Piera se dio cuenta de que según la tabla del juego de instrucciones del M68000 y las tablas C.2 y C.1, el modo de direccionamiento mediante registro de direcciones no está permitido para la instrucción NEGX. De hecho, si se intenta ensamblar `NEGX.B A0`, se produce un error de modo de direccionamiento ilegal. Es de suponer que si se intenta desensamblar `4008` y se obtiene `NEGX.B A0`, es debido a que el software de simulación (por motivos de pragmatismo) es menos robusto y riguroso para desensamblar que para ensamblar. Pero esto es sólo una opinión mía.

1995. Septiembre

Se dispone de un computador dotado de un microprocesador MC68000. En un momento dado, estos son los contenidos de algunos registros:

(D0) = $20008541
(D1) = $45103421
(D2) = $3549BA21

Indique cuál será el contenido final de los registros D0 y D1 si inmediatamente después se ejecuta el siguiente fragmento de programa:

```
            ADD.B    D2,D1
            BEQ      OP2
OP1         ROR.B    #1,D0
            BRA      SEGUIR
OP2         ROR.W #1,D1
SEGUIR      NOP
```

Solución:

```
00001000             1          org    $1000
00001000  D202       2          ADD.B  D2,D1  ; D2[7:0] + D1[7:0] = $21 + $21 = $42 -> (D1)=45103442
00001002  6700 0008  3          BEQ    OP2    ; No se cumple la condición del salto
00001006  E218       4 OP1      ROR.B   #1,D0 ; $41=01000001 pasa a ser $A0=10100000
00001008  6000 0004  5          BRA     SEGUIR
0000100C  E259       6 OP2      ROR.W   #1,D1 ; Por aquí no pasa
0000100E  4E71       7 SEGUIR   NOP
                     8
00001010  1E3C 00E4  9          move.b #228,D7
00001014  4E4E      10          trap   #14
00001016            11          end
```

Teniendo en cuenta que los tres bytes más significativos de D0 y de D1no se han visto alterados:

(D0) = $200085A0
(D1) = $45103442

1996. Septiembre (reserva).

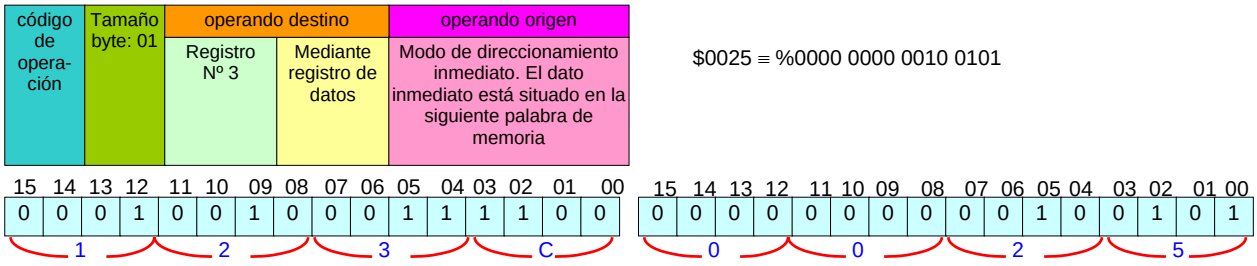
Sea la siguiente secuencia de programa en ensamblador del M68000:

a) Exprese en hexadecimal el código máquina obtenido al ensamblar las instrucciones de las sentencias ensamblador 3 y 4.

b) Indique el valor que tomarán los registros `D0` y `D1` cuando se ejecute el programa desde la sentencia `CODIGO` hasta la sentencia `FIN`.

Solución:

a) MOVE.B #$25,D1



ADD.B D0,D1

Campo del modo de operación:

Byte	Palabra	Palabra larga	Operación
000	001	010	(de) + (Dn) →Dn
100	101	110	(Dn) + (de) →de

De momento conocemos el modo byte; pero hay dos posibilidades: (000) si estamos en la primera línea; o (100) si es la segunda línea. Para determinar qué línea de las dos posibles es nuestro caso, debemos fijarnos en cuál es la operación . Pero parece que hay una indeterminación, pues:

- Si (*de*) es D0, estaremos en la primera línea: (de) AND (Dn) →Dn; por tanto, el campo modo sería 000
- Si (*de*) es D2, estaremos en la segunda línea: (Dn) AND (de) →de; por tanto, el campo modo sería 100

Esta aparente indeterminación la resolvemos al acudir al apéndice C y ver los modos posibles de direccionamiento de cada operando. Allí vemos

Sintaxis:	ADD.S a_1,D_i	o	ADD.S D_i,a_3
Tamaño de los operandos:	S = (B, W, L)		
Operación	$(a_1)+(D_i)\rightarrow D_i$	o	$(D_i) + (a_3) \rightarrow a_3$

Si ahora vamos a las tablas C.2 y C.1, vemos que el conjunto de modos a_3 no incluye el modo mediante registro. Por tanto, para que el operando destino sea un registro, debemos estar en la primera línea.

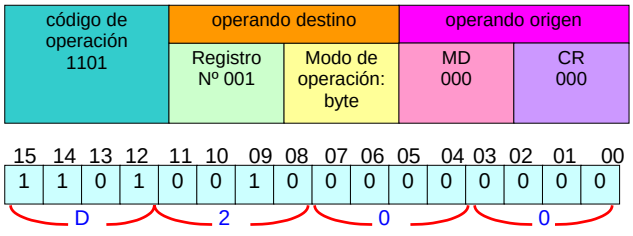
Byte	Palabra	Palabra larga	Operación
000	001	010	(de) AND (Dn) →Dn

Como conclusión, podemos decir que en esta instrucción siempre alguno de los operandos está en un registro de datos. Puede ser fuente o destino. Según sea un caso u otro, así será el bit #8.:

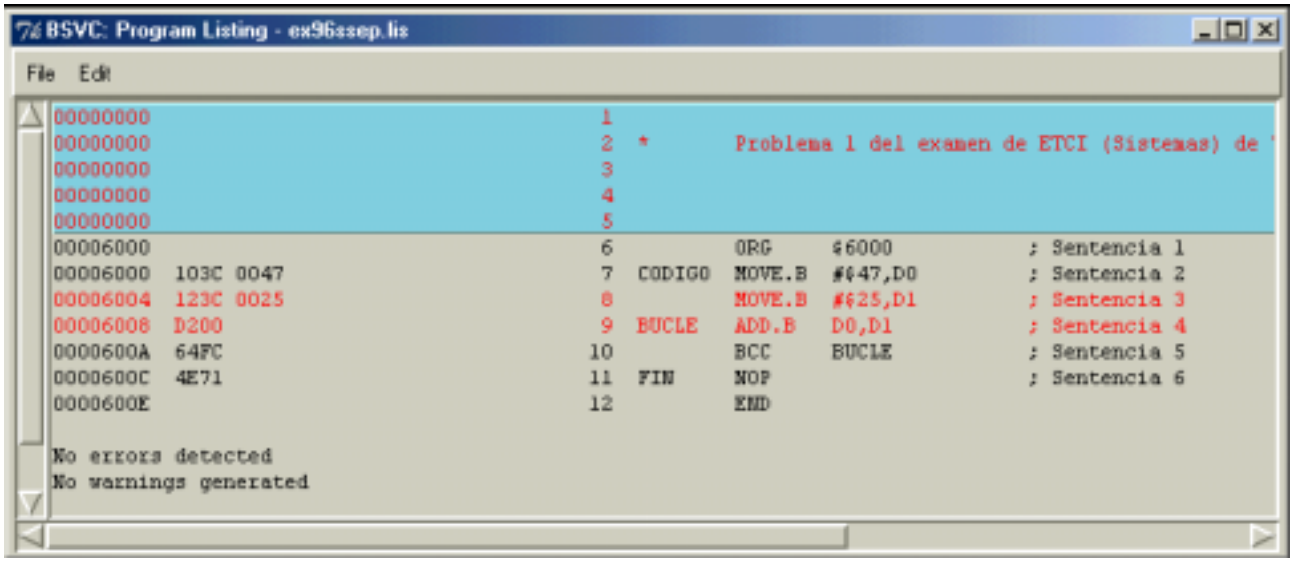
Destino → b_8 = 0

Fuente → b_8 = 1

Y si ambos operandos fueran registros de datos, el bit #8 es '0'.



Comprobemos la solución con el entorno de programación BSVC.



b)

1ª pasada

	0100	0111	D0
+	0010	0101	D1
C=0	0110	1100	D1

	0100	0111	D0
+	0110	1100	
C=0	1011	0011	

	0100	0111	D0
+	1011	0011	
C=0	1111	1010	

4ª pasada

	0100	0111	D0
+	1111	1010	D1
C=1	0100	0001	D1

1997. Febrero. Segunda semana.

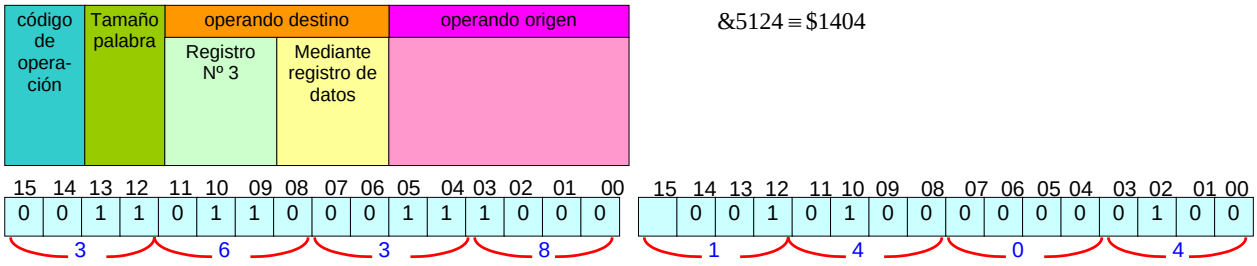
Sea el siguiente conjunto de instrucciones en ensamblador del MC68000 (a la izquierda) y el siguiente conjunto de instrucciones máquina del mismo compactadas en código hexadecimal (a la derecha).

A) MOVE.W 5124,D3 1) 06430304
B) ADD.W @1404,D3 2) B5785124
C) ADDI.W #@1404,D3 3) D6780304
D) ANDI.W #1044,(A5) 4) 36385124
E) EOR.W D2,20772 5) 02550414

Se pide:
- Emparejar cuando sea posible las instrucciones en la columna de la izquierda con sus representaciones en código máquina de la columna de la derecha.
- Indicar las operaciones realizadas en cada caso.

Solución:

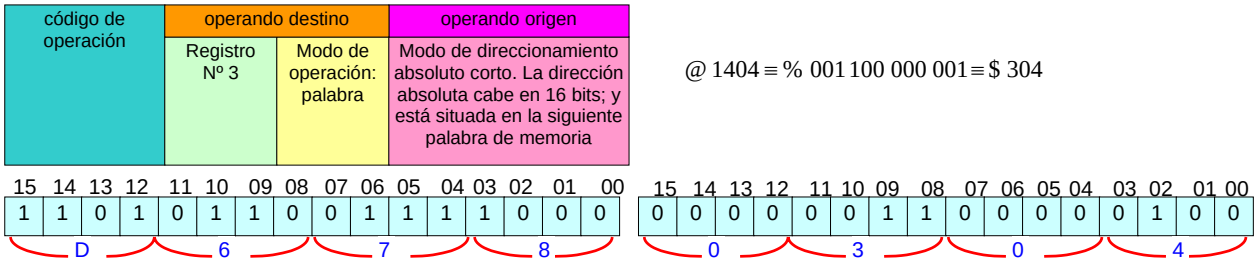
A) MOVE.W 5124,D3



&5124 ≡ $1404

Código máquina: 36381404; no se corresponde con ninguna opción de la columna de la derecha.
Operación efectuada: (&5124)→d3

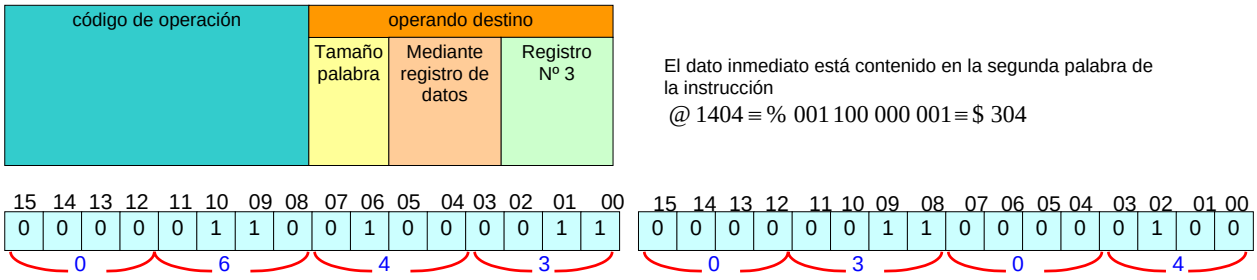
B) ADD.W @1404,D3



@ 1404 ≡ % 001 100 000 001 ≡ $ 304

Código máquina: D6780304; se corresponde con el de la opción 3) de la columna de la derecha.
Operación efectuada: $(@1404)+(D3)\rightarrow D3$

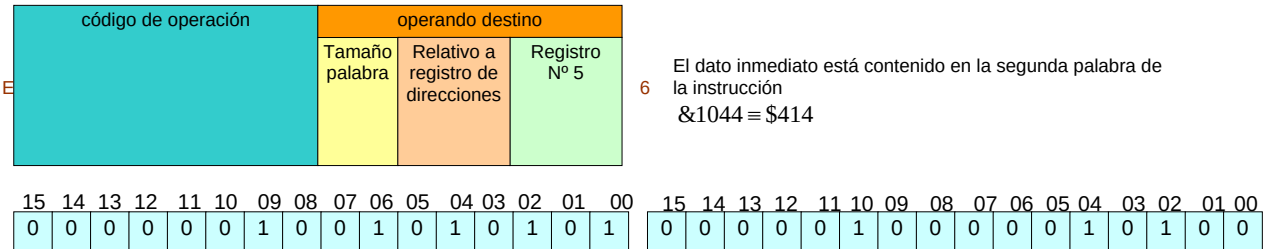
C) ADDI.W #@1404,D3



El dato inmediato está contenido en la segunda palabra de la instrucción
@ 1404 ≡ % 001 100 000 001 ≡ $ 304

Código máquina: 06430304; se corresponde con el de la opción 1) de la columna de la derecha.
Operación efectuada: $@1404+(D3)\rightarrow D3$

D) ANDI.W #1044,(A5)

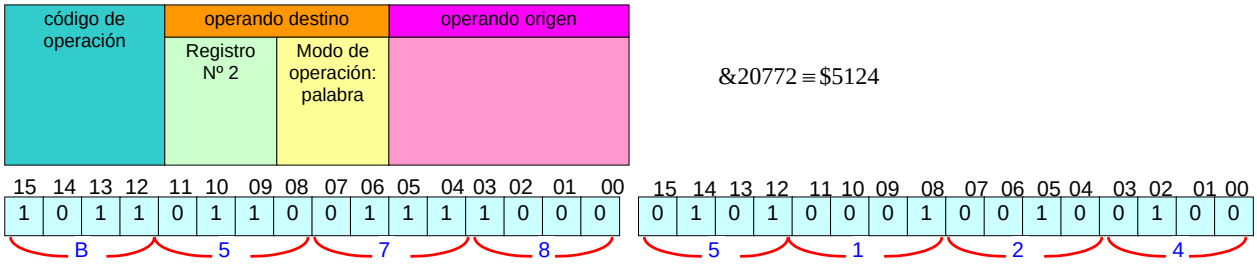


E 6

El dato inmediato está contenido en la segunda palabra de la instrucción
&1044 ≡ $414

Código máquina: 02550414; se corresponde con el de la opción 5) de la columna de la derecha.
Operación efectuada: $\&1044\ AND\ ((A5)) \rightarrow (A5)$

E) EOR.W D2,20772



Código máquina: B5785124; se corresponde con el de la opción 2) de la columna de la derecha.
Operación efectuada: $(D2) \oplus (20772) \rightarrow (20772)$

Podemos comprobar el código máquina correspondiente a estas instrucciones en el fichero listado generado por el ensamblador cruzado:

```
MC68000 Cross Assembler
Copyright (C) Stephen Croll, 1991.  Author: Stephen Croll
Version 2.00 beta 1.02
                        1 *
                        2 *     Escuela de Informática de la UNED
                        3 *     Asignatura: Estructura y Tecnología de Computadores I
                        4 *     Problema 1º del examen del 11-02-1997
                        5 *     Tutoría del Centro Asociado de Plasencia
                        6 *
                        7 *
                        8 * Propósito: Comprobar el código máquina generado por varias instrucciones
                        9 *            del lenguaje ensamblador del MC68000
                       10
00000000  3638 1404     11            move.w    5124,D3
00000004  D678 0304     12            add.w     @1404,D3
00000008  0643 0304     13            addi.w    #@1404,D3
0000000C  0255 0414     14            andi.w    #1044,(A5)
00000010  B578 5124     15            eor.w     D2,20772
                        16
00000014                17            end

No errors detected.
```

1998. Febrero. Primera semana.

Suponga que el registro D3 de un microprocesador M68000 contiene inicialmente el valor 0. Diga cuál de los siguientes fragmentos de programa implementa un bucle.

a)

```
BUCLE ADD.L    #5,D3
      CMP.L    #25,D3
      BNE      BUCLE
FIN   MOVE.L   D3,D2
```

b)

```
BUCLE CMP.L    #25,D3
      BEQ      FIN
      ADD.L    #5,D3
FIN   MOVE.L   D3,D2
```

c)

```
BUCLE ADD.L    #5,D3
      CMP.L    #25,D3
      BEQ      FIN
FIN   MOVE.L   D3,D2
```

d) Ninguna de las anteriores

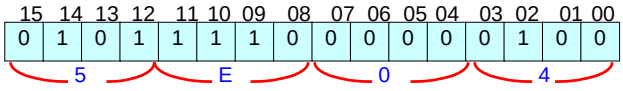
Solución:

En b) y c) no hay salto hacia atrás. No se puede iterar ninguna instrucción. Así es imposible implementar un bucle. En a) el cuerpo del bucle consiste en incrementar D3 de 5 en 5 unidades, mientras sea diferente de #25. El bucle no es infinito, pues hay una pasada en la que D3 contiene #25.

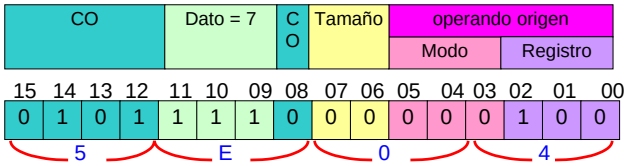
1998. Febrero, segunda semana.

Encuentre la instrucción del M68000 que se corresponde con el código hexadecimal 5E04.

Solución:



Nos fijamos en el campo del código de operación. Vamos comparando con todo el repertorio de instrucciones. Sólo las instrucciones ADDQ y DBCC tienen 0101 en dicho campo. De estas dos descartamos DBCC pues debería tener a 1 los bits 3, 6 y 7.



Campo del dato:
001→1
010→2
011→3
100→4
101→5
110→6
111→7
000→8

Campo tamaño:
00→byte
01→palabra
10→palabra larga

Como conclusión, 5E04 es el código de `ADDQ.B #7,D4`

Dado el siguiente segmento de programa para el M68000, indique cual de las siguientes afirmaciones es FALSA.

```
RUTINA   MOVE.W   #17,D1   ;  1ª
         MOVE.W   #8,D2    ;  2ª
         MOVE.W   #5,D3    ;  3ª
         MOVE.W   D3,D4    ;  4ª
         ADD.W    D2,D4    ;  5ª
         CMP.W    D4,D1    ;  6ª
         BLT      SI       ;  7ª
NO       MOVE.B   #0,D0    ;  8ª
         BRA      FIN      ;  9ª
SI       MOVE.B   #1,D0    ; 10ª
FIN      RTS               ; 11ª
```

a) D0.B contiene un 1 al final de su ejecución.
b) El indicador de cero después de ejecutarse la instrucción 5ª vale (Z) = 0.
c) D4.W tiene el valor 13 al final de su ejecución.
d) La 9ª es una instrucción de salto relativo al contador de programa.

Solución:

```
RUTINA   MOVE.W   #17,D1   ;  1ª    (D1) = &17
         MOVE.W   #8,D2    ;  2ª    (D2) =  &8
         MOVE.W   #5,D3    ;  3ª    (D3) =  &5
         MOVE.W   D3,D4    ;  4ª    (D4) =  &5
         ADD.W    D2,D4    ;  5ª    (D4) =  (D2)+(D4) = &8+&5 = &13
         CMP.W    D4,D1    ;  6ª    (D1) < (D4) es falso
         BLT      SI       ;  7ª    No hay salto
NO       MOVE.B   #0,D0    ;  8ª    (D0) = 0
         BRA      FIN      ;  9ª    Pasa por aquí
SI       MOVE.B   #1,D0    ; 10ª    Esta la pasa por alto
FIN      RTS               ; 11ª
```

a) D0.B contiene un 1 al final de su ejecución. *Falso, pues pasa por alto la décima.*
b) El indicador de cero después de ejecutarse la instrucción 5ª vale (Z) = 0. ¿?.
c) D4.W tiene el valor 13 al final de su ejecución. *Cierto, pues como el resultado es &13≠&0, no se activa, permanece a cero*.
d) La 9ª es una instrucción de salto relativo al contador de programa. *Cierto, pues lo que se codifica el desplazamiento desde la dirección donde está almacenada la instrucción de la línea 9ª hasta la de la 11ª . y ese desplazamiento es respecto al PC, que en ese momento apunta a la de la línea 10*.

Simulando con BSVC, podemos ver a la izquierda los contenidos finales de los registros y a la derecha las instrucciones ejecutadas. Vemos que la octava

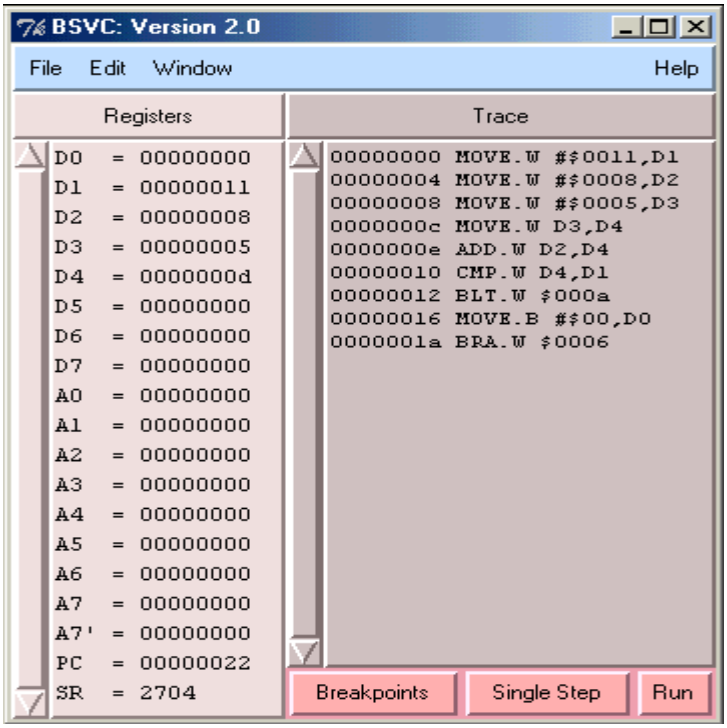
```
NO       MOVE.B   #0,D0    ;  8ª
```

sí se ejecuta, y que en cambio, la décima

```
SI       MOVE.B   #1,D0    ; 10ª
```

No se ejecuta. Todo ello es debido a que no se cumple la condición del salto condicional

```
         BLT      SI       ;  7ª
```



1998. Septiembre, original (sistemas).

Cuál podría ser el contenido de los registros D1 y D2 antes de la ejecución de la instrucción MOVE.B D2,D1 del M68000 si después su contenido, expresado en hexadecimal, es D1=12453341 y D2=12453341:
a) D1 = 12453341 y D2 = 11111155.

b) D1 = 12453341 y D2 = 12453355
c) D1 = 12453355 y D2 = 12453341.
d) D1 = 00000055 y D2 = 12453341.

Solución:

Fíjate que nos dan los valores finales y preguntan los valores iniciales.

a) Si los contenidos iniciales fueran D1 = 12453341 y D2 = 11111155, al hacer MOVE.B D2,D1 resultaría:
D1 = 12453355 (se ha escrito el byte menos significativo del operando fuente)
D2 = 11111155 (el operando fuente no se altera)

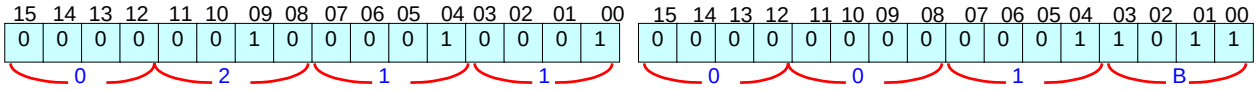
b) Si los contenidos iniciales fueran D1 = 12453341 y D2 = 12453355, al hacer MOVE.B D2,D1 resultaría:
D1 = 12453355 (se ha escrito el byte menos significativo del operando fuente)
D2 = 12453355 (el operando fuente no se altera)

c) Si los contenidos iniciales fueran D1 = 12453355 y D2 = 12453341, al hacer MOVE.B D2,D1 resultaría:
D1 = 12453341 (se ha escrito el byte menos significativo del operando fuente)
D2 = 12453341 (el operando fuente no se altera)

d) Si los contenidos iniciales fueran D1 = 00000055 y D2 = 12453341, al hacer MOVE.B D2,D1 resultaría:
D1 = 00000041 (se ha escrito el byte menos significativo del operando fuente)
D2 = 12453341 (el operando fuente no se altera)

Encuentre la instrucción del M68000 que se corresponde con el código hexadecimal 0211001B.

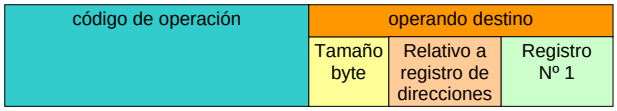
Solución:



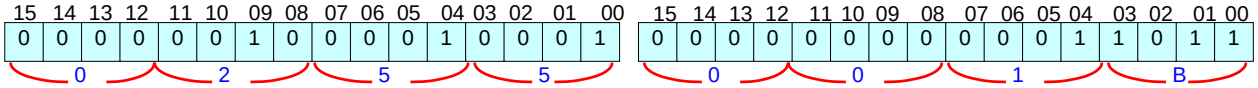
Nos fijamos en los cuatro bits más significativos, que suelen ser parte del campo del código de operación. Vamos comparando con todo el repertorio de instrucciones. De las que tienen igual estos cuatro bits:

ADDI	Descartada, pues debería tener b_{10} = 1.
ANDI	Posible.
ANDI a CCR	Descartada, pues debería tener b_0 = 1.
ANDI a SR	Descartada, pues debería tener b_0 = 1.
BCHG	Descartada, pues debería tener b_8 = 1.
CMPI	Descartada, pues debería tener b_{10} = 1.
EORI	Descartada, pues debería tener b_{11} = 1.
ORI	Descartadas, pues deberían tener b_9 = 0.
SUBI	Descartada, pues debería tener b_9 = 0.

La única posible es ANDI, cuyo formato es este:



El dato inmediato está contenido en la segunda palabra de la instrucción
$001B = % 0000 0000 0001 1011 = &27



Conclusión: `ANDI.B #27,(A1)`

Dado el siguiente programa en ensamblador del M68000, ¿cual de las siguientes afirmaciones es FALSA?:

a) Al ejecutarse con los datos del enunciado se obtiene: (OUT1) = $03; (OUT2) =$0080.
b) Si el contenido del dato de entrada fuera (IN) = $03, el resultado sería: (OUT1) =$00; (OUT2) = $0008.

c) En la posición de memoria OUT2 se devuelve el resto de la división del dato ubicado en IN dividido entre 16.
d) En la posición de memoria OUT1 se devuelve el cociente del dato ubicado en IN dividido entre 16.

```
CODIGO   ORG   $1000
         CLR.L D0
         MOVE.B IN,D0
         MOVEQ #0,D1
         DIVU.W #16,D0
         MOVE.B D0,OUT1
         MOVE.W #0,D0
         SWAP D0
         BSET D0,D1
         MOVE.W D1,OUT2
         STOP #$2700
DATOS    ORG $1200
IN       DC.B 55
OUT1     DS.B 1
OUT2     DS.W 1
         END
```

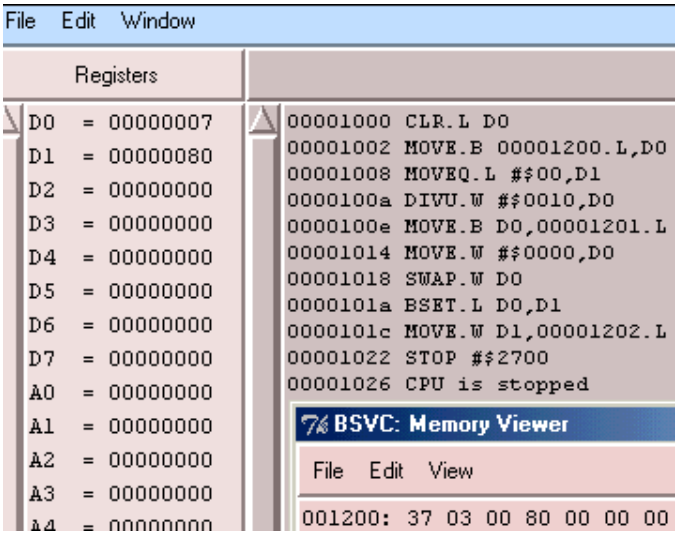
Solución:

Al lado de cada instrucción se indica en forma de comentario su efecto

```
CODIGO   ORG   $1000
         CLR.L D0           ;  1ª  (D0) = $ 0000 0000
         MOVE.B IN,D0       ;  2ª  (D0) = & 55
         MOVEQ #0,D1        ;  3ª  (D1) = & 0000 0000. En MOVEQ el tamaño siempre es palabra larga
         DIVU.W #16,D0      ;  4ª  D0[31:16] = resto = 0007,  D0[16:00] = cociente = 0003
         MOVE.B D0,OUT1     ;  5ª  Véase la 13ª
         MOVE.W #0,D0       ;  6ª  (D0) = 0007 0000
         SWAP D0            ;  7ª  (D0) = 0000 0007
         BSET D0,D1         ;  8ª  (D1) = 0000 0080
         MOVE.W D1,OUT2     ;  9ª  Véase la 14ª
         STOP #$2700        ; 10ª
DATOS    ORG $1200          ; 11ª
IN       DC.B 55            ; 12ª
OUT1     DS.B 1             ; 13ª (OUT1) = 03 desde la 5ª
OUT2     DS.W 1             ; 14ª (OUT2) = 00 80 desde la 9ª
         END                ; 15ª
```

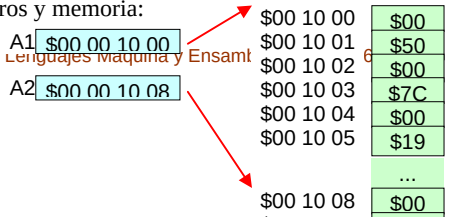
a) Al ejecutarse con los datos del enunciado se obtiene: (OUT1) = $03; (OUT2) =$0080. *Cierto*.
b) Si el contenido del dato de entrada fuera (IN) = $03, el resultado sería: (OUT1) =$00; (OUT2) = $0008. *Cierto, 00 es el resto de (03÷16)*.
c) En la posición de memoria OUT2 se devuelve el resto de la división del dato ubicado en IN dividido entre 16. *Falso, el resto se pierde en la instrucción 8ª*.
d) En la posición de memoria OUT1 se devuelve el cociente del dato ubicado en IN dividido entre 16. *Cierto*.

Esta es la comprobación con simulador:



1999. Febrero, primera semana (sistemas).

Se dispone de un computador basado en el microprocesador M6000. En un momento dado, estos son los contenidos de algunos registros y memoria:



Calcule el contenido del byte menos significativo de los registros D6 y D7 después de ejecutarse el fragmento de programa comprendido entre las etiquetas PRINCIPIO y FIN.

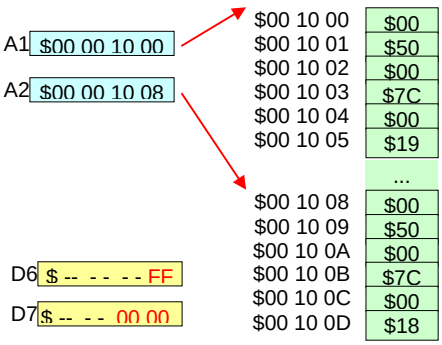
```
PRINCIPIO   MOVE.B   #$FF,D6
            CLR.W    D7
BUCLE       CMPM.W   (A1)+,(A2)+
            BNE      DISTIN
            TST.W    -1(A1)
            BEQ      FIN
            ADDQ.W   #1,D7
            BRA      BUCLE
DISTIN      CLR.B    D6
FIN         NOP
            END
```

Solución:

a) Traza

Antes de entrar en el bucle

```
PRINCIPIO   MOVE.B   #$FF,D6  ; (D6) = $-- -- -- FF
            CLR.W    D7       ; (A2) = $-- -- 00 00
```

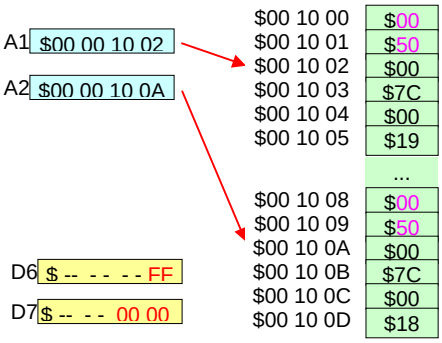


1ª pasada por el bucle

Primera comprobación de salida del bucle:
Comparación de las posiciones de memoria

```
BUCLE       CMPM.W   (A1)+,(A2)+ ; (Z) = 1
            BNE      DISTIN      ;
```

Como son iguales, la condición de salto es falsa. No se salta.
Se ejecutará la siguiente instrucción.
Los punteros A1 y A2 han avanzado dos bytes.



Segunda comprobación de salida del bucle:
Comparación de las posiciones de memoria

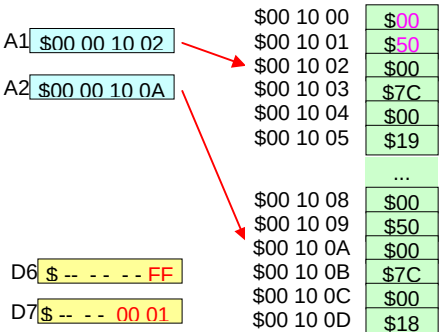
```
            TST.W    -1(A1)    ; (Z) = 0
            BEQ      FIN       ;
```

Como ($1000)≠0, la condición de salto es falsa. No se salta.
Se ejecutará la siguiente instrucción.

```
            ADDQ.W   #1,D7
```

Y se vuelve al comienzo del bucle

```
            BRA      BUCLE
```



Ahora se puede ver el propósito del programa. Comprobar los elementos de dos vectores en memoria, hasta que se cumple alguna de estas dos condiciones de salida del bucle:

1ª Algún par de elementos diferentes. En este caso se salta a la penúltima instrucción, con lo que D6 acaba conteniendo 0.

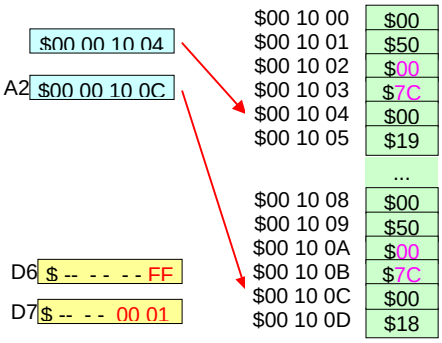
2ª El elemento en curso del primer vector es nulo (se supone que los vectores acaban en el elemento nulo, y se allegado al final). En este caso se salta directamente a la última instrucción, pasando por alto la penúltima, con lo que D6 contiene el valor dado al comienzo del programa: -1.

2ª pasada por el bucle

Primera comprobación de salida del bucle:
Comparación de las posiciones de memoria

```
BUCLE       CMPM.W    (A1)+,(A2)+ ; (Z) = 1
            BNE       DISTIN      ;
```

Como son iguales, la condición de salto es falsa. No se salta.
Se ejecutará la siguiente instrucción.
Los punteros A1 y A2 han avanzado otros dos bytes.



Segunda comprobación de salida del bucle:
Comparación de las posiciones de memoria

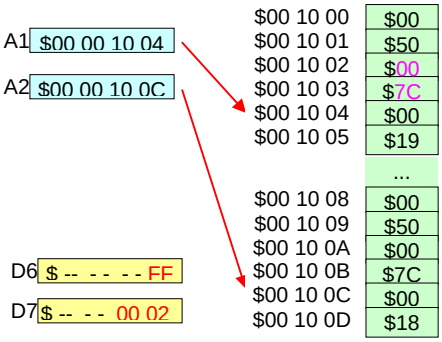
```
            TST.W     -1(A1)    ; (Z) = 0
            BEQ       FIN       ;
```

Como ($1002)≠0, la condición de salto es falsa. No se salta.
Se ejecutará la siguiente instrucción.

```
            ADDQ.W    #1,D7
```

Y se vuelve al comienzo del bucle

```
            BRA       BUCLE
```



3ª pasada por el bucle

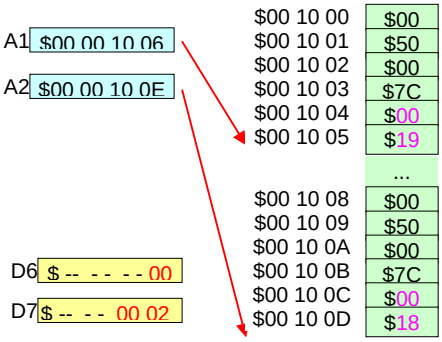
Primera comprobación de salida del bucle:
Comparación de las posiciones de memoria

```
BUCLE       CMPM.W    (A1)+,(A2)+ ; (Z) = 0
            BNE       DISTIN      ;
```

Como son diferentes, la condición de salto es cierta.
Se lleva a cabo el salto a

```
DISTIN      CLR.B     D6.         ; (D6) =$-- -- -- 00
```

Los vectores no son iguales.



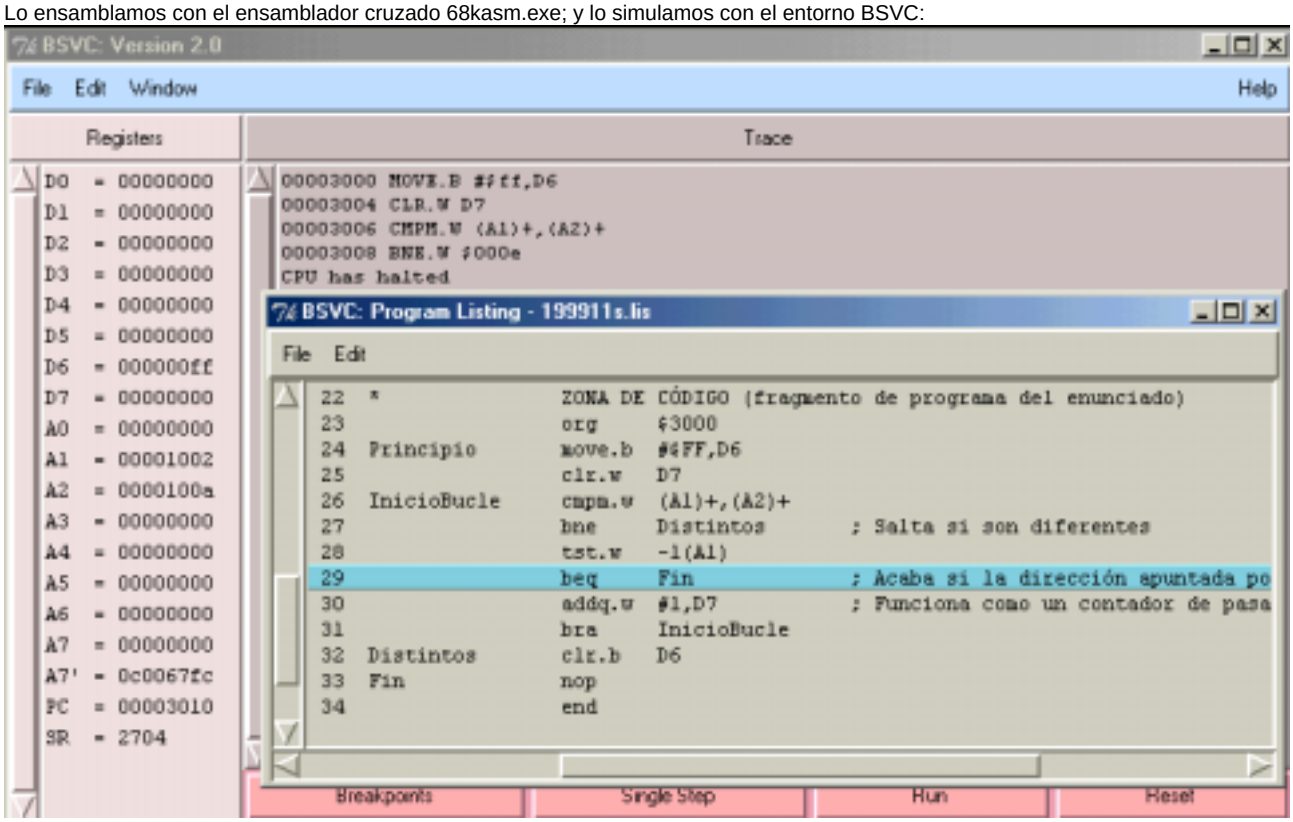
b) Intento de comprobar con el simulador.

Editamos el siguiente fichero fuente:

```
*         ZONA DE DATOS
            org    $1000
            dc.b   $00,$50,$00,$7C,$00,$19,$00,$00
            dc.b   $00,$50,$00,$7C,$00,$18,$00,$00

*         PREPARACIÓN PREVIA (condiciones iniciales del enunciado)
```

```
            org   $2000
            movea.l  #$1000,A1
            movea.l  #$1008,A2
            bra   Principio

*        ZONA DE CÓDIGO (fragmento de programa del enunciado)
            org   $3000
Principio   move.b   #$FF,D6
            clr.w D7
InicioBucle cmpm.w   (A1)+,(A2)+
            bne   Distintos   ; Salta si son diferentes
            tst.w -1(A1)
            beq   Fin         ; Acaba si la dirección apuntada por A1 contiene 00
            addq.w   #1,D7    ; Funciona como un contador de pasadas
            bra   InicioBucle
Distintos   clr.b D6
Fin         nop
            end
```

Lo ensamblamos con el ensamblador cruzado 68kasm.exe; y lo simulamos con el entorno BSVC:

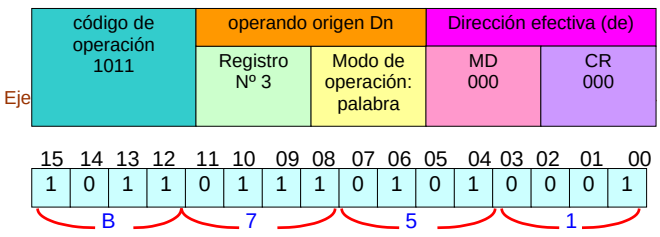


Observamos que la ejecución se detiene en la línea señalada (29), en la primera pasada de bucle. Esto sucede porque en la línea 28, A1 apunta a la dirección $1002. Pero al bus de direcciones se envía esta dirección con un desplazamiento de –1 (esto es un fallo del simulador, pues al trabajar en tamaño palabra, debería ser –2). En la línea 28, se intenta hacer un acceso a memoria en tamaño de operación tipo palabra, lo cual necesita que la dirección sea par. Como no se cumple este requerimiento, ocurre un error de bus de direcciones; y la CPU se detiene.

Indique cuál es el código máquina correspondiente a la instrucción del MC68000 EOR.W D3,(A1).

Solución:

Según el apéndice B de las UUDD (página 513), la instrucción OR tiene este formato:



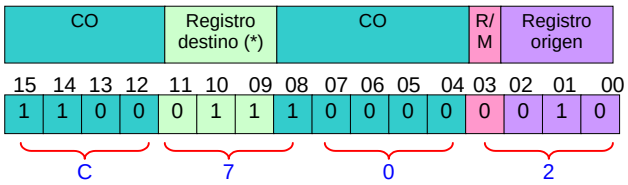
Campo del modo de operación:	Byte	Palabra	Palabra larga	Operación
	100	101	110	(de) XOR (Dn) → (de)

1999. Febrero, primera semana (gestión).

Genere el código máquina correspondiente a la instrucción del MC68000 ABCD.B D2,D3.

Solución:

La instrucción ABCD tiene este formato:



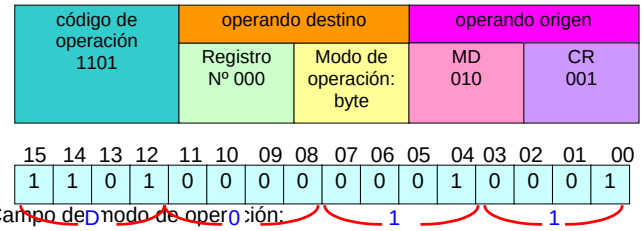
Campo R/M:
R/M = 0 → registro de datos a registro de datos
R/M = 1→ memoria a memoria

(*) Si R/M = 0, especifica un registro de datos
Si R/M = 1, especifica un registro de dirección para el modo de direccionamiento con predecremento

Genere el código máquina correspondiente a la instrucción del MC68000 ADD.B (A1),D0.

Solución:

La instrucción ADD tiene este formato:



Campo de modo de operación:

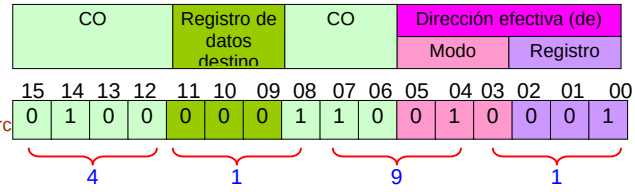
Byte	Palabra	Palabra larga	Operación
000	001	010	(de) AND (Dn) →Dn
100	101	110	(Dn) AND (de) →de

El único registro de datos que aparece en la instrucción es el destino, por tanto, el campo de modo de operación lo determina la primera línea (000).

Genere el código máquina correspondiente a la instrucción del MC68000 CHK (A1),D0.

Solución:

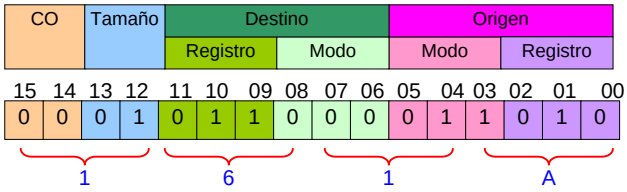
La instrucción CHK tiene este formato:



Genere el código máquina correspondiente a la instrucción del MC68000 MOVE.B (A2)+,D3.

Solución:

La instrucción MOVE tiene este formato:

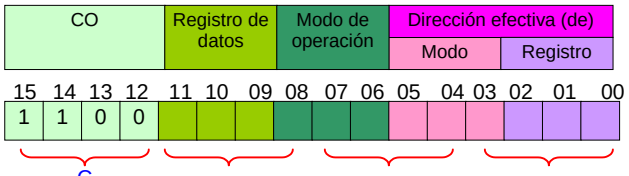


Campo Tamaño:
- 01 → byte
- 11 → palabra
- 10 → palabra larga

Genere el código máquina correspondiente a la instrucción del MC68000 AND.W D4,D5.

Solución:

La instrucción AND tiene este formato:



Campo del modo de operación:

Byte	Palabra	Palabra larga	Operación
000	001	010	(de) AND (Dn) →Dn
100	101	110	(Dn) AND (de) →de

De mon to conocem el modo palat , pero hay dos sibilidades: (001) si estamos en la primera línea; o (101) si es la segunda línea. Para determinar qué línea de las dos posibles es nuestro caso, debemos fijarnos en cuál es la operación . Pero parece que hay una indeterminación, pues:

- Si (*de*) es D4, estaremos en la primera línea: (de) AND (Dn) →Dn; por tanto, el campo modo sería 001
- Si (*de*) es D5, estaremos en la segunda línea: (Dn) AND (de) →de; por tanto, el campo modo sería 101

Esta aparente indeterminación la resolvemos al acudir al apéndice C y ver los modos posibles de direccionamiento de cada operando. Allí vemos

Sintaxis:	AND.S a_4,D_i	o	AND.S D_i,a_3
Tamaño de los operandos:	S = (B, W, L)		
Operación	$(a_4) \wedge (D_i) \rightarrow D_i$	o	$(D_i) \wedge (a_3) \rightarrow a_3$

Si ahora vamos a las tablas C.2 y C.1, vemos que el conjunto de modos a_3 no incluye el modo mediante registro. Por tanto, la operación es:

$(a_4) \wedge (D_i) \rightarrow D_i$

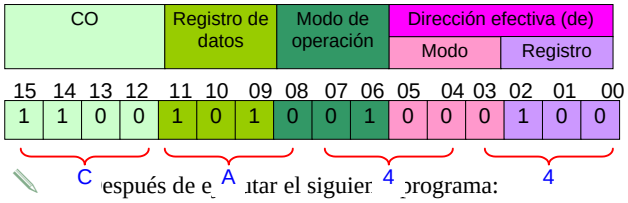
Conclusión: el modo de operación es el de la primera línea:

Byte	Palabra	Palabra larga	Operación
000	001	010	(de) AND (Dn) → Dn

Para no tener que repetir el razonamiento, a partir de ahora recordaremos que en

AND.S Di,Dj

(*de*) es (Di)



espués de e tar el siguier rograma:

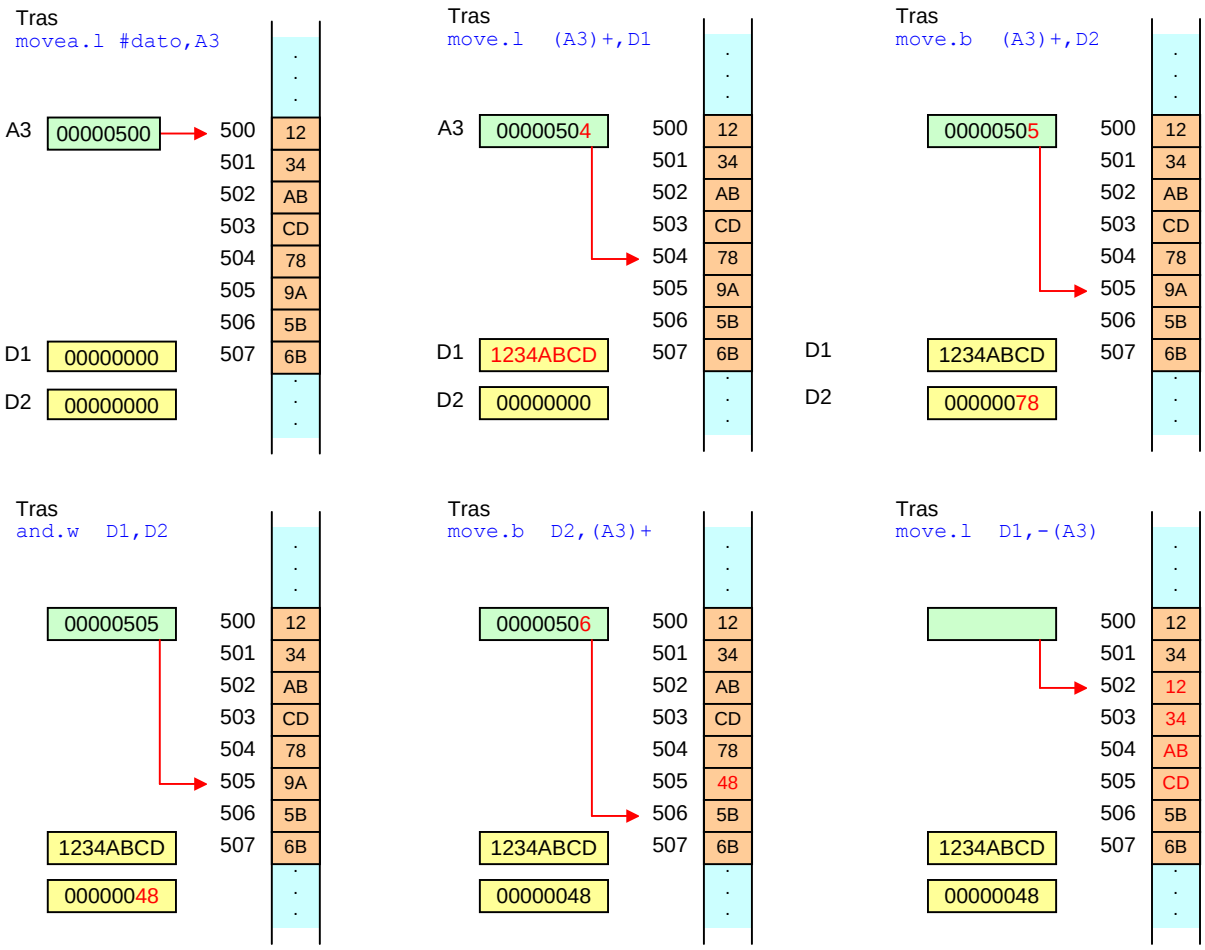
```
        org     $500
dato    dc.w    $1234,$ABCD,$789A,$5B6B

        movea.l #dato,A3
        move.l  (A3)+,D1
        move.b  (A3)+,D2
        and.w   D1,D2
        move.b  D2,(A3)+
        move.l  D1,-(A3)
```

Indique los contenidos de:

a) El registro D1.
b) El registro D2.
c) El registro A3.
d) La posición de memoria cuya dirección es $506.
e) La posición de memoria cuya dirección es $504.

Solución:

Traza:



1999. Febrero, segunda semana (sistemas).

Sea el siguiente segmento de programa escrito en ensamblador del M68000. Encuentre el código máquina equivalente a la línea 5ª del mismo.

```
          ORG       $10          ;1ª
          MOVE.W    #$6728,D0    ;2ª
          MOVE.W    #$2539,D1    ;3ª
BUCLE1    ABCD      D0,D1        ;4ª
          BCC       BUCLE1       ;5ª
```

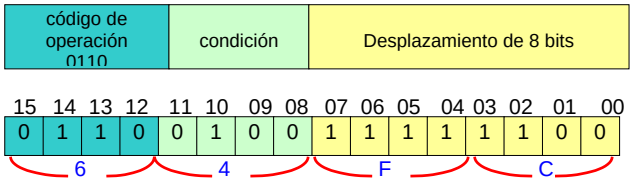
Solución:

```
                ORG     $10         ;1ª   Longitud: 4 bytes
$10=&26         MOVE.W  #$6728,D0   ;2ª   Longitud: 4 bytes
$14=&20         MOVE.W  #$2539,D1   ;3ª   Longitud: 2 bytes
$18=&24  BUCLE1 ABCD    D0,D1       ;4ª   Longitud: 2 bytes (el desplazamiento cabe en 8 bits)
$1A=&26         BCC     BUCLE1      ;5ª
$1C=&28  Este es el contenido del PC cuando se está ejecutando la 5ª
```

El desplazamiento desde la dirección `$1C` hasta la `$18` es $\Delta Dir = Dir_f - Dir_i$ = `$18` - `$1C` = -4

-&4 = -%0000 0100 Complementar a 1 1111 1011 Sumar un 1 1111 1100

Complementar a 2



Sea el siguiente programa escrito en ensamblador del M68000. Supóngase que se ejecutan desde la línea 1 a la línea 9 inclusive del mismo. Después de su ejecución, cual de las siguientes posiciones de memoria NO contiene el byte que se indica (expresado todo en hexadecimal):

a) (1202) = 80
b) (1203) = 08
c) (1201) = 02
d) (1204) = 00

```
PROGRAMA    CLR.L    D0        ; Línea 1
            MOVE.B   DAT,D0    ; Línea 2
            CLR.L    D1        ; Línea 3
            DIVU.W   #32,D0    ; Línea 4
            MOVE.B   D0,RE1    ; Línea 5
            MOVE.W   #0,D0     ; Línea 6
            SWAP     D0        ; Línea 7
            BCHG     D0,D1     ; Línea 8
            MOVE.L   D1,RE2    ; Línea 9
DATOS       ORG      $1200     ; Origen de datos
DAT         DC.B     95        ; Dato de entrada
RE1         DS.B     1         ; Resultado 1
RE2         DS.L     1         ; Resultado 2
```

Solución:

Al lado de cada instrucción se indica en forma de comentario su efecto

```
PROGRAMA    CLR.L    D0        ; Línea 1            (D0)=$00 00 00 00
            MOVE.B   DAT,D0    ; Línea 2            (D0)=$00 00 00 5F
            CLR.L    D1        ; Línea 3            (D1)=$00 00 00 00
            DIVU.W   #32,D0    ; Línea 4            (D0)=$00 1F 00 02  (resto=1F, cociente=02)
            MOVE.B   D0,RE1    ; Línea 5            (1201)=02
            MOVE.W   #0,D0     ; Línea 6            (D0)=$00 1F 00 00
            SWAP     D0        ; Línea 7            (D0)=$00 00 00 1F = &31
            BCHG     D0,D1     ; Línea 8            (D1)=$80 00 00 00 Invierte el bit 31 de D1
            MOVE.L   D1,RE2    ; Línea 9            (1202)=80 00 00 00
DATOS       ORG      $1200     ; Origen de datos
DAT   $1200 DC.B     95        ; Dato de entrada
RE1   $1201 DS.B     1         ; Resultado 1        (1201)=02 desde la 5ª
RE2   $1202 DS.L     1         ; Resultado 2        (1202)=80 00 00 00 desde la 9ª
```

a) (1202) = 80 *Cierto*.
b) (1203) = 08 *Falso*.
c) (1201) = 02 *Cierto*.
d) (1204) = 00 *Cierto*.

1999. Septiembre, original (sistemas).

Lea atentamente el fragmento de código a la derecha y ejecútelo hasta llegar a FIN. A continuación señale cuál es el contenido de los registros D1 y D4 después de ejecutarlo:

```
            ORG      $2000

RESTAR      EQU      $1
SUMAR       EQU      $2

INICIO      CLR.B    D5
            MOVE.L   #$EFE0,D1
            MOVE.L   #$D4,D4

OPERACION   CMP.B    #SUMAR,D5
            BEQ      Suma
            CMP.B    #RESTAR,D5
            BEQ      Resta
            BRA      SIGUE
```

Solución:

Traza:

	1ª Pasada	2ª Pasada	3ª Pasada
D1 al comienzo D1 al final	00 00 EF E0 00 00 EF E0	00 00 EF E0 00 00 1B DC	00 00 1B DC 00 00 1B DC
D4 al comienzo D4 al final	00 00 00 D4 00 00 D4 04	00 00 D4 04 00 00 04 D8	00 00 04 D8 00 00 B4 24
D5 (contador de pasadas)	00 00 00 00	00 00 00 01	00 00 00 02
D6	indefinido	indefinido	20 B4
D7	indefinido	00 00 1B DC	00 00 1B DC
Suma	No pasa	No pasa	Sí pasa
Resta	No pasa	Sí pasa	No pasa

Los contenidos de los registros D0, D1 y D2 de un microprocesador M68000 son:
(D0) = $00000003
(D1) = $A3B00505
(D2) = $A30B0385
Calcule el contenido del registro D1 tras la ejecución del siguiente fragmento de programa:

```
BUCLE   EOR.W   D2,D1
        SWAP    D1
        SUBI.B  #1,D0
        BNE     BUCLE
```

Solución:

Traza:

1ª pasada


```
EOR.W   D2,D1
                        0000 0011 1000 0101  D2
1010 0011 1011 0000  ⊕  0000 0101 0000 0101  D1
1010 0011 1011 0000     0000 0110 1000 0000  D1

SWAP    D1
0000 0110 1000 0000     1010 0011 1011 0000  D1
```

2ª pasada

```
EOR.W    D2,D1
```

```
                                0000 0011 1000 0101  D2
0000 0110 1000 0000          ⊕  1010 0011 1011 0000  D1
0000 0110 1000 0000             1010 0000 0011 0101  D1
```

```
SWAP     D1
```

```
1010 0000 0011 0101     0000 0110 1000 0000  D1
```

```
EOR.W    D2,D1
```

```
                                0000 0011 1000 0101  D2
1010 0000 0011 0101          ⊕  0000 0110 1000 0000  D1
1010 0000 0011 0101             0000 0101 0000 0101  D1
```

```
SWAP     D1
```

```
0000 0101 0000 0101     1010 0000 0011 0101  D1
```

	Inicialmente	Tras la 1ª Pasada	Tras la 2ª Pasada	Tras la 3ª Pasada
D0 (contador de pasadas)	00 00 00 03	00 00 00 02	00 00 00 01	00 00 00 00
D1 (acumulador)	A3 B0 05 05	06 80 A3 B0	A0 35 06 80	05 05 A0 35
D2 (constante)	A3 0B 03 85	A3 0B 03 85	A3 0B 03 85	A3 0B 03 85

1999. Septiembre, original (gestión).

Después de ejecutar el siguiente programa:

```
                ORG     $2000

MULTIPLICAR     equ     $1
DIVIDIR         equ     $2

Inicio          move.l  #3,D5
                move.l  #$BAC0,D1
                move.l  #$7FF9,D4

Operacion       cmp.b   #MULTIPLICAR,D5
                beq     OpeMultiplica
                cmp.b   #DIVIDIR,D5
                beq     OpeDivide
                bra     Seguir
```

Indique los contenidos de:

a) El registro D1.
b) El registro D4.
c) El registro D5.
d) El registro D6.
e) El registro D7.

Solución:

Traza:

1ª pasada

(D5) = 0000 0003
Multiplica: No
Divide: No
Seguir: Sí

```
addi.w   #7,D4 ; (D4)+7 → D4 ;  (D4) = 0000 8000
rol.w    #8,D4 ;              ;  (D4) = 0000 0080
lsr.w    #4,D1 ;              ;  (D1) = 0000 0BAC
```

2ª pasada

(D5) = 0000 0002
Multiplica: No
Divide: Sí

```
   divu     D4,D1                ;  (D1) = 002C 0017
   and.l    #$FFFF,D1            ;  (D1) = 0000 0017
   move.w   D1,D6                ;  (D6) = 0000 0017
```

Seguir: Sí

```
   addi.w   #7,D4 ; (D4)+7 → D4 ;  (D4) = 0000 0087
   rol.w    #8,D4 ;             ;  (D4) = 0000 8700
   lsr.w    #4,D1 ;             ;  (D1) = 0000 0001
```

3ª pasada

(D5) = 0000 0001
Multiplica: Sí

```
   mulu     D4,D1                ;  (D4) = 0000 8700
   move.w   D1,D7                ;  (D7) = 0000 0001
```

Divide: No
Seguir: Sí

```
   addi.w   #7,D4 ; (D4)+7 → D4 ;  (D4) = 0000 8707
   rol.w    #8,D4 ;             ;  (D4) = 0000 0787
   lsr.w    #4,D1 ;             ;  (D1) = 0000 0000
```

4ª pasada

(D5) = 0000 0000
Multiplica: No
Divide: No
Seguir: Sí

```
   addi.w   #7,D4 ; (D4)+7 → D4 ;  (D4) = 0000 078E
   rol.w    #8,D4 ;             ;  (D4) = 0000 8E07
   lsr.w    #4,D1 ;             ;  (D1) = 0000 0000
```

Se tiene el siguiente bloque de datos. Esta tabla proporciona los datos necesarios para el encendido de los segmentos de un visualizador codificados en hexadecimal (1→apaga, 0→enciende). Indique cuál sería el resultado que se mostraría en pantalla tras ejecutar los bloques de código dados a continuación.
Nota: Téngase en cuenta que el registro D7 es el que se envía a la rutina de pantalla.

```
Tabla    DC.B  $C0      * 0
         DC.B  $F9      * 1
         DC.B  $A4      * 2
         DC.B  $B0      * 3
         DC.B  $99      * 4
         DC.B  $92      * 5
         DC.B  $82      * 6
         DC.B  $F8      * 7
         DC.B  $80      * 8
         DC.B  $98      * 9
         DC.B  $88      * A
         DC.B  $83      * B
         DC.B  $C6      * C
         DC.B  $A1      * D
         DC.B  $86      * E
```

a)
```
LEA       Tabla,A4
MOVE.B    17(A4),D7
```

b)
```
LEA       Tabla,A4
MOVE.W    (A4)+,D5
MOVE.B    D5,D7
```

c)
```
LEA       Tabla,A4
MOVE.L    #$5,D5
MOVE.B    4(A4,D5),D7
```

Solución:

a)
```
LEA       Tabla,A4
MOVE.B    17(A4),D7      $FF      → D7  La rutina de pantalla la convierte en “Display apagado”.
```

El resultado en pantalla sería pantalla apagada.

b)
```
LEA       Tabla,A4
MOVE.W    (A4)+,D5       $C0F9    → D5
MOVE.B    D5,D7          $F9      → D7  La rutina de pantalla la convierte en ‘1’.
```

El resultado en pantalla sería un ‘1’.

c)
```
LEA       Tabla,A4
MOVE.L    #$5,D5
MOVE.B    4(A4,D5),D7    (Tabla+9) → D7  La rutina de pantalla la convierte en ‘9, pues (D7)=$98’.
```

El resultado en pantalla sería un ‘9’.

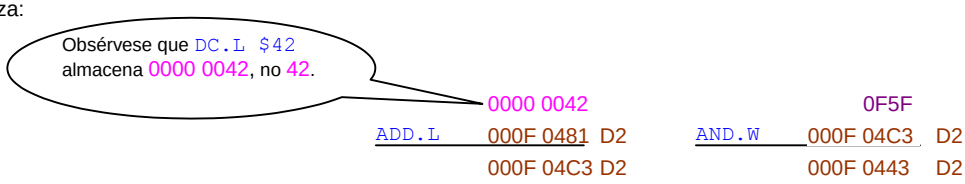
2000. Febrero, primera semana (sistemas).

Analice el contenido del registro D2 después de ejecutarse el siguiente fragmento de código del MC68000:

```
      ORG 2500
INI   EQU $F5F
      MOVE.L #$000F0481,D2
      ADD.L ET,D2
      AND.W #INI,D2
ET    DC.L $42
      END
```

Solución:

Traza:



Señalar el contenido de la memoria en la posición indicada por la etiqueta ST después del ensamblado del siguiente fragmento de código del M68000::

```
        ORG   2500;
TAB     DS.L  $12       ;
LIB     EQU   *+2048    ;
CAN     DC.W  256+LIB   ;
ST      DC.W  $400+CAN ;
        END
```

Solución:

Traza:

Tras: `TAB DS.L $12`

Comentario: Reserva $12 palabras largas, es decir, 4·$12 = 4·&18 = &72 bytes
Contador de ensamblado: &2500 + &72 = &2572
Estado actual de la tabla de símbolos:

Símbolo	Valor
TAB	&2500
LIB	
CAN	
ST	

Tras: `LIB EQU *+2048`

Comentario: Declara la constante `LIB` con el valor `*+2048 = &2572+&2048 =&4620`
Contador de ensamblado: &2572 (No ha variado)
Estado actual de la tabla de símbolos:

Símbolo	Valor
TAB	&2500
LIB	&4620
CAN	
ST	

Tras: `CAN DC.W 256+LIB`

Comentario: Inicializa la posición CAN=&2572 con el valor `&256+LIB` =&256+&4620 = &4876
Contador de ensamblado: &2572 + &2= &2574
Estado actual de la tabla de símbolos:

Símbolo	Valor
TAB	&2500
LIB	&4620
CAN	&2572
ST	

Tras: `ST DC.W $400+CAN`

Comentario: Inicializa la posición ST=&2574 con el valor `$400+CAN` =&1024+&2572 = &3596 = $E0C
Contador de ensamblado: &2574 + &2= &2576
Estado actual de la tabla de símbolos:

Símbolo	Valor
TAB	&2500
LIB	&4620
CAN	&2572
ST	&2574

2000. Febrero, primera semana (gestión).

Ejecute el siguiente programa y conteste a las preguntas que hay a continuación.

Notas:

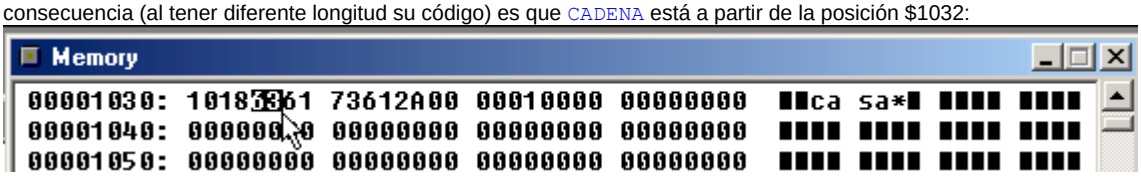
`CADENA` está en la posición de memoria (en hexadecimal) $1038.
`CONTADOR` está en la posición de memoria (en hexadecimal) $103E.
Los códigos ASCII de los caracteres de dicha cadena son (en hexadecimal):

- 'c' = $63
- 'a' = $61
- 's' = $73
- '*' = $2A

```
            org     $1000

MAXIMO      equ      $5
BUSCAR      equ      'a'

*  Zona de código

            clr      CONTADOR
            lea      CADENA,A6
            move.l   #MAXIMO,D6
```

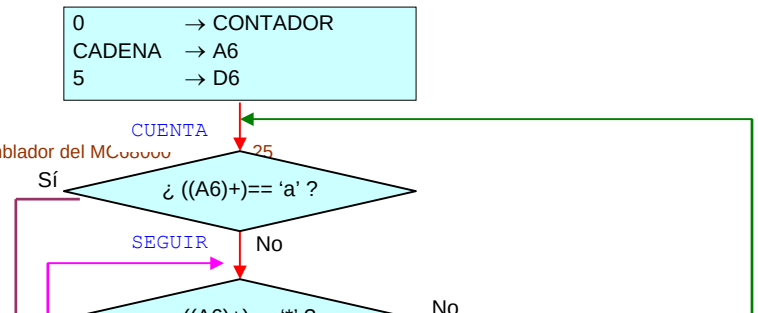
a) Valor de (A6) la primera vez que se ejecuta el código entre la etiqueta CUENTA y la siguiente instrucción de salto.
b) Valor de D5 la segunda vez que se ejecuta el código entre la etiqueta CUENTA y la siguiente instrucción de salto.
c) Contenido de la dirección $1041 la segunda vez que se ejecuta el código entre la etiqueta SEGUIR y la siguiente instrucción de salto.
d) Valor de A6 la tercera vez que se ejecuta el código entre la etiqueta CUENTA y la siguiente instrucción de salto.
e) Contenido de la dirección $103F una vez finalizado el programa.

Atención. El simulador WISM68 cambia por su cuenta varias instrucciones. Por ejemplo, la instrucción addi por addq.w. La consecuencia (al tener diferente longitud su código) es que CADENA está a partir de la posición $1032:

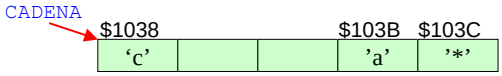


Solución:

Este es el diagrama de flujo del programa. La culpa de su enrevesamiento está en el propósito de colocar las cajas en el orden en que aparecen en el programa; y en el uso de la instrucción dbeq, que está descompuesta en la parte lateral. Globalmente, lo que hace el programa es contar el número de caracteres '**a**' que hay en una cadena almacenada en la dirección CADENA, hasta encontrar el carácter '*****', o hasta llegar a la sexta posición de la cadena. Como contador de coincidencias se utilizan 32 bits empezando en la posición CONTADOR. Al final del programa, este número también se almacena en el registro D4.



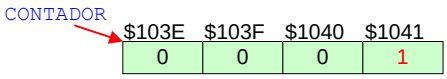
a) Valor de (A6) la primera vez que se ejecuta el código entre la etiqueta CUENTA y la siguiente instrucción de salto.
Antes de dicha etiqueta A6=CADENA es decir, A6=$1038.
Después de (A6)+, como se trabaja en tamaño byte, A6=$1039.



Pero no nos preguntan por A6, sino por (A6), el cual es el código de 'a', es decir, $61 .

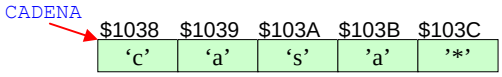
b) Valor de D5 la segunda vez que se ejecuta el código entre la etiqueta CUENTA y la siguiente instrucción de salto.
D5 contiene el código del carácter en curso de la cadena.
En la segunda iteración contiene el código de la primera '**a**' de la cadena. Es decir, contiene (A5)=$61.

c) Contenido de la dirección $1041 la segunda vez que se ejecuta el código entre la etiqueta SEGUIR y la siguiente instrucción de salto.
Esta dirección es el byte menos significativo de los cuatro que ocupa la reserva de memoria llamada CONTADOR.



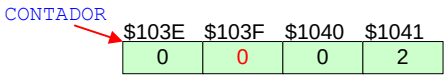
La segunda vez que pasa por dicho segmento de código, acaba de encontrar la primera '**a**' y actualizar el contador, por tanto, su contenido es #1.

d) Valor de A6 la tercera vez que se ejecuta el código entre la etiqueta CUENTA y la siguiente instrucción de salto.



Debido al posincremento (A6)+, en cada iteración apunta al carácter siguiente al actual, es decir A6=$103B.

e) Contenido de la dirección $103F una vez finalizado el programa.

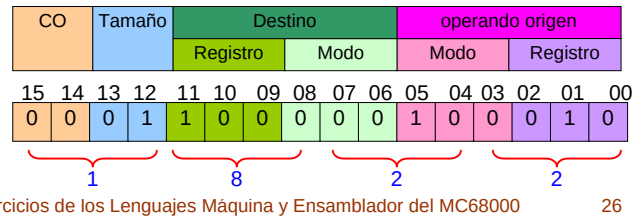


Al final del programa, está almacenado el número $0002 en la posición CONTADOR, pues hay dos caracteres '**a**'. El correspondiente a la posición $103F es #0.

Genere el código máquina correspondiente a la instrucción del MC68000 MOVE.B -(A2),D4.

Solución:

La instrucción MOVE tiene este formato:

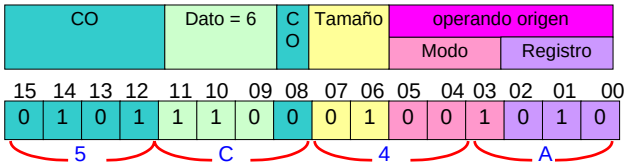


Campo Tamaño:
01 → byte
11 → palabra
10 → palabra larga

Genere el código máquina correspondiente a la instrucción del MC68000 `ADDQ.W #6,A2`.

Solución:

La instrucción ADDQ tiene este formato:



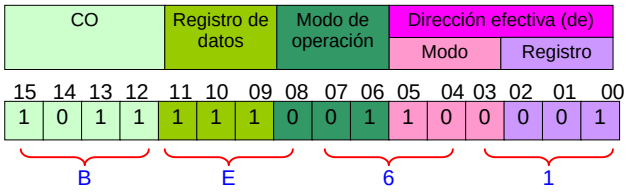
Campo del dato:
001→1
010→2
011→3
100→4
101→5
110→6
111→7
000→8

Campo tamaño:
00→byte
01→palabra
10→palabra larga

Genere el código máquina correspondiente a la instrucción del MC68000 `CMP.W -(A1),D7`.

Solución:

La instrucción CMP tiene este formato:



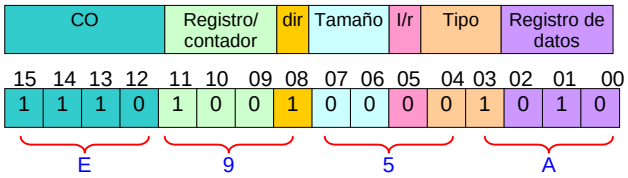
Campo del modo de operación:

Byte	Palabra	Palabra larga
000	001	010

Genere el código máquina correspondiente a la instrucción del MC68000 `LSL.B #4,D2`.

Solución:

Según el apéndice B de las UUDD, las instrucciones ASL, ASR, LSL, LSR, ROL, ROR y ROXL, cuando el destino es un registro tienen este formato:

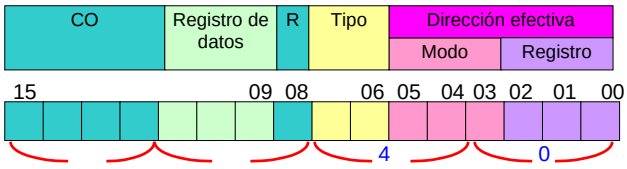


Campo Tipo:
Tipo = 00 → desplazamiento aritmético
Tipo = 01 → desplazamiento lógico
Tipo = 10 → rotación con extensión
Tipo = 11 → rotación

Genere el código máquina correspondiente a la instrucción del MC68000 `BCHG #4,D0`.

Solución:

La instrucción BCHG tiene este formato:



Si el operando origen es
- ✓ Un dato inmediato
  R=0
  Campo registro de datos = 1000; y el dato está en la segunda palabra
- ✓ Un registro de datos
  R=1
  Campo registro de datos = Nº de registro

Campo tipo:
00→TST
01→CHG
10→CLR
11→SET

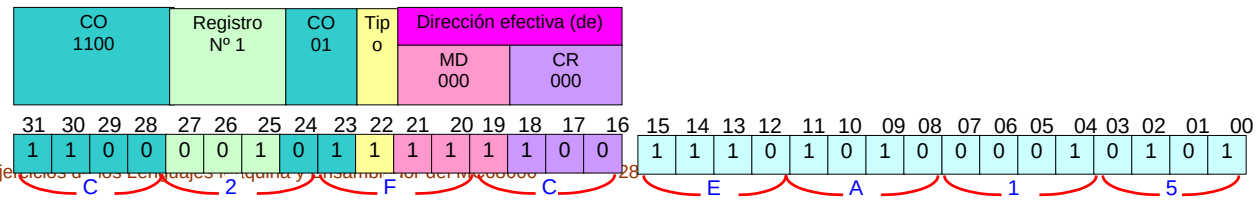
Por tanto, la solución es 0840 0004

2000. Febrero, segunda semana (sistemas).

Genere el código máquina correspondiente a la instrucción del MC68000 `MULU.W #$EA15,D1`.

Solución:

Según el apéndice B de las UUDD (página 518), las instrucciones MULS y MULU tienen este formato:



Campo Tipo:	MULS	MULU
	1	0

Dado el siguiente fragmento de programa ensamblador escrito para el M68000 indicar cuál es el contenido final de los registros D1, D2 y D3 si su contenido inicial es (D1)=$AB041153, (D2)=$BCDE8A01 y (D3)=$01FF8374.

```
AND.B D1,D2
OR.B  D2,D3
NOT.B D3
EOR.B D3,D1
```

Solución:

Traza:

```
AND.B     D1,D2
1010 1011 0000 0100 0001 0001    0101 0011 D1
1011 1100 1101 1110 1000 1010 ^  0000 0001 D2
1011 1100 1101 1110 1000 1010    0000 0001 D2    (D2) = BCDE 8A01, ya no varía

OR.B      D2,D3
1010 1011 1101 1110 1000 1010    0000 0001 D2
0000 0001 1111 1111 1000 0011 v  0111 0100 D3
0000 0001 1111 1111 1000 0011    0111 0101 D3    (D3) = 01FF 8375, varía en la siguiente

NOT.B     D3
0000 0001 1111 1111 1000 0011 ¬  0111 0100 D3
0000 0001 1111 1111 1000 0011    1000 1010 D3    (D3) = 01FF 838A, ya no varía



0000 0001 1111 1111 1000 0011             D3
1010 1011 0000 0100 0001 0001    0101 0011 D1
1010 1011 0000 0100 0001 0001    1101 1001 D1    (D1) =AB04 11D9, ya no varía
```

2000. Febrero, segunda semana (gestión).

Ejecute el siguiente programa y conteste a las preguntas que hay a continuación.

Notas:

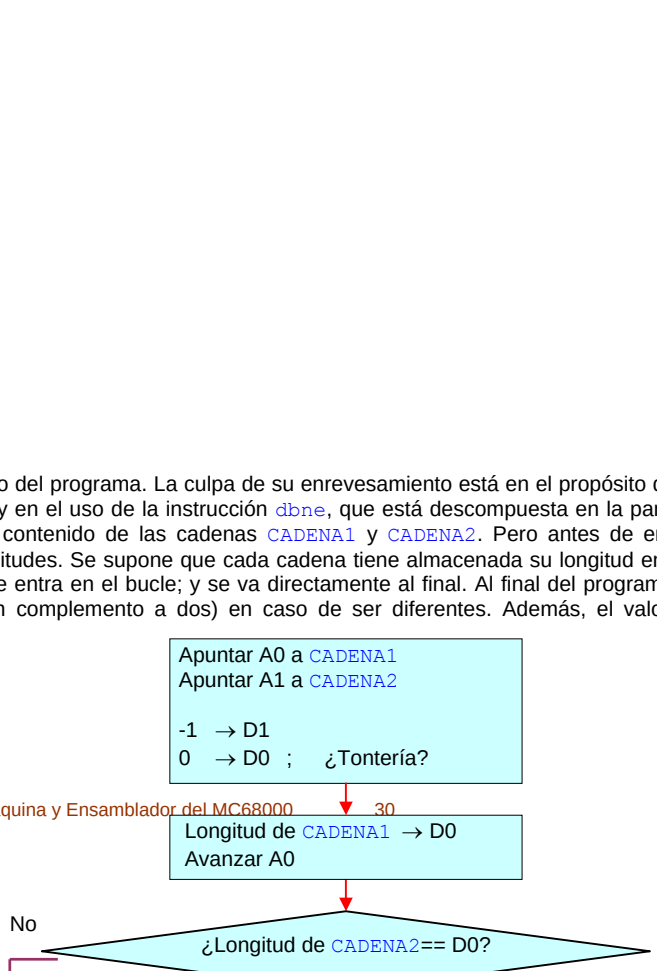
CADENA1 está en la posición de memoria (en hexadecimal) $2000.
CADENA2 está en la posición de memoria (en hexadecimal) $2004.
COINCIDE está en la posición de memoria (en hexadecimal) $2008.
El contenido de CADENA1 es (en hexadecimal) $ 03 00 00 00
El contenido de CADENA2 es (en hexadecimal) $ 03 00 00 00
El contenido de COINCIDE es (en hexadecimal) $ 00 00

```
DATOS       equ       $2000
```

a) Valor de (A1) la primera vez que se ejecuta el código hasta la primera instrucción de salto.
b) Valor de (A0) la primera vez que se ejecuta el código entre la etiqueta `BUCLE` y la siguiente instrucción de salto.
c) Valor de D0 la segunda vez que se ejecuta el código entre la etiqueta `BUCLE` y la siguiente instrucción de salto.
d) Valor de D1 la tercera vez que se ejecuta el código entre la etiqueta `BUCLE` y la siguiente instrucción de salto.
e) Contenido del espacio de memoria reservado por `COINCIDE` una vez finalizado el programa.

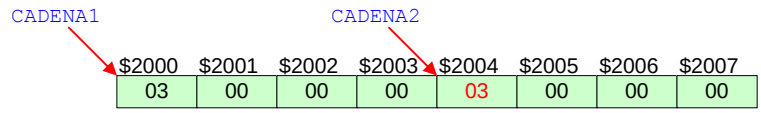
Solución:

Este es el diagrama de flujo del programa. La culpa de su enrevesamiento está en el propósito de colocar las cajas en el orden en que aparecen en el programa; y en el uso de la instrucción `dbne`, que está descompuesta en la parte lateral. Globalmente, lo que hace el programa es comparar el contenido de las cadenas `CADENA1` y `CADENA2`. Pero antes de entrar en el `BUCLE` que recorre dichas cadenas, compara las longitudes. Se supone que cada cadena tiene almacenada su longitud en el primer elemento. En caso de tener longitudes diferentes, no se entra en el bucle; y se va directamente al final. Al final del programa, D1 contiene $00 00 en caso de ser iguales; o $FF FF (#-1 en complemento a dos) en caso de ser diferentes. Además, el valor de D1 se almacena en la posición `COINCIDE`.



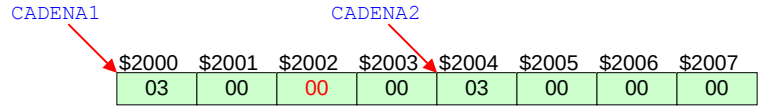
a) Valor de (A1) la primera vez que se ejecuta el código hasta la primera instrucción de salto.

```
COMIENZO    movea.l  #CADENA1,A0    ;  A0 = $2000
            movea.l  #CADENA2,A1    ;  A1 = $2004, (A1)=$03
            moveq    #-1,D1
            moveq    #0,D0          ;  D0 = $00    ¿Tontería?
            move.b   (A0)+,D0       ;  D0 = $03 , A0 = $2001
            cmp.b    (A1)+,D0       ;  A1 = $2005
            bne      FIN            ; Primera instrucción de salto
```



b) Valor de (A0) la primera vez que se ejecuta el código entre la etiqueta BUCLE y la siguiente instrucción de salto.

```
BUCLE       cmpm.b   (A0)+,(A1)+    ;  A0 = $2002 ,   A1 = $2006 , (A0) = $00
            dbne     D0,BUCLE
            bne      FIN
```



c) Valor de D0 la segunda vez que se ejecuta el código entre la etiqueta BUCLE y la siguiente instrucción de salto.

```
            move.b   (A0)+,D0       ;  D0 = $03, la longitud de la primera cadena
            . . .
```

```
                subq.w    #1,D0          ; D0 = $02, inmediatamente antes de entrar en el bucle

    BUCLE       cmpm.b    (A0)+,(A1)+
                dbne      D0,BUCLE       ; D0 = $01 para la siguiente iteración (la segunda)
```

d) Valor de D1 la tercera vez que se ejecuta el código entre la etiqueta `BUCLE` y la siguiente instrucción de salto.

```
                moveq     #-1,D1         ; Este indicador se había inicializado a $FFFF
                . . .

    BUCLE       cmpm.b    (A0)+,(A1)+
                dbne      D0,BUCLE
                bne       FIN

    IGUAL       not.w     D1             ; Por aquí aún no se ha pasado, por tanto, aún D1=$FF
FF
```

e) Contenido del espacio de memoria reservado por `COINCIDE` una vez finalizado el programa.

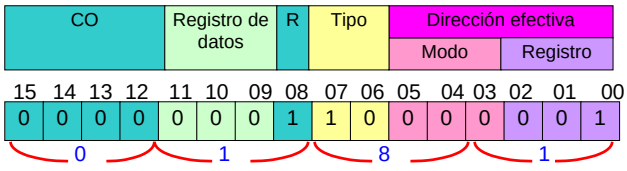
En la tercera iteración `D0=#0`

```
    BUCLE       cmpm.b    (A0)+,(A1)+
                dbne      D0,BUCLE       ; Sale de este bucle debido a que D0=#-1
                bne       FIN            ; Los elementos comparados eran iguales. No salta

    IGUAL       not.w     D1             ; D1 = $00 00

    FIN         move.w    D1,COINCIDE    ; COINCIDE = $00 00
```

Genere el código máquina correspondiente a la instrucción del MC68000 `BCLR D0,D1`.

Solución:

Las instrucciones de la familia BCHG tienen este formato:



Ejercici Si el operando origen es

- ✓ Un dato inmediato
  R=0
  Campo registro de datos = 1000; y el dato está en la segunda palabra
- ✓ Un registro de datos
  R=1
  Campo registro de datos = Nº de registro

Campo tipo:
00→TST
01→CHG
10→CLR
11→SET

Por tanto, la solución es 0181

Genere el código máquina correspondiente a la instrucción del MC68000 `AND.L (A6),D6`.

Solución:

La instrucción AND tiene este formato:

Campo del modo de operación:

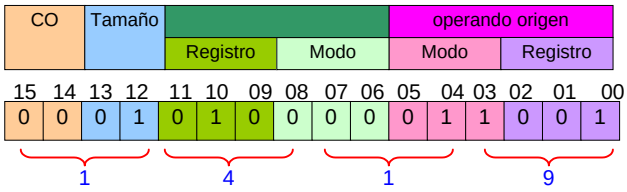
Byte	Palabra	Palabra larga	Operación
000	001	010	(de) AND (Dn) →Dn
100	101	110	(Dn) AND (de) →de

El único registro de datos lo tenemos en el operando de destino, por lo que para encontrar el campo de modo de operación aplicamos la línea superior.

Genere el código máquina correspondiente a la instrucción del MC68000 `MOVE.B (A1)+,D2`.

Solución:

La instrucción MOVE tiene este formato:

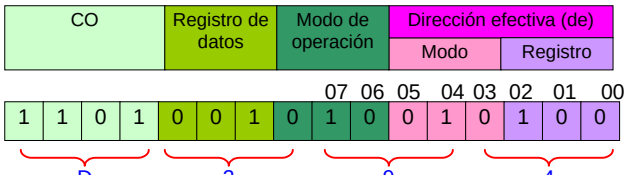


Campo Tamaño:
01 → byte
11 → palabra
10 → palabra larga

Genere el código máquina correspondiente a la instrucción del MC68000 `ADD.L (A4),D1`.

Solución:

La instrucción AND tiene este formato:



Campo del modo de operación:

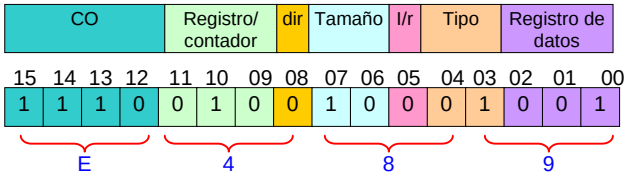
Byte	Palabra	Palabra larga	Operación
000	001	010	(de) + (Dn) →Dn
100	101	110	(Dn) + (de) →de

El único registro de datos lo tenemos en el operando de destino, por lo que para encontrar el campo de modo de operación aplicamos la línea superior.

Genere el código máquina correspondiente a la instrucción del MC68000 `LSR.L #2,D1`.

Solución:

Según el apéndice B de las UUDD, las instrucciones ASL, ASR, LSL, LSR, ROL, ROR y ROXL, cuando el destino es un registro tienen este formato:



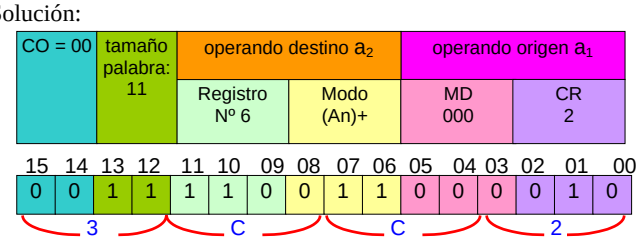
Campo Tipo:
Tipo = 00 → desplazamiento aritmético
Tipo = 01 → desplazamiento lógico
Tipo = 10 → rotación con extensión
Tipo = 11 → rotación

Campo dir:

2000. Septiembre, original (sistemas).

Genere el código máquina correspondiente a la instrucción del MC68000 `MOVE.W D2,(A6)+`.

Solución:



Campo del modo de operación: MOVE.S a_1,a_2

Byte	Palabra	Palabra larga	Operación
01	11	10	$(a_1) \rightarrow a_2$

Suponiendo que el contenido inicial del registro D1 es (D1)=3F2EF111, ¿Cuál es el contenido después de `ADDI.W  #$1000,D1`?.

Solución:

Se trabaja con la palabra menos significativa de D1, D1[15:00] = F111

```
      0001  0000  0000  0000
  +   1111  0001  0001  0001  D1
C=1   0000  0001  0001  0001  D1
```

El acarreo del bit 15 pasa al bit C, pero no al bit 16, pues la palabra más significativa permanece inalterada. Por tanto, el contenido final de D1 es 3F2E0111

Sean los siguientes contenidos iniciales de los registros:

```
(D0) = $ 0000 0003
(D1) = $ A3B0 0505
(D2) = $ A30B 0385
```

Calcule el contenido del registro D1 después de la ejecución del siguiente segmento de programa:

```
BUCLE   EOR.W   D2,D1
        SWAP    D1
        SUBI.B  #1,D0
        BNE     BUCLE
```

Solución:

Traza:

1ª pasada

(D0) = 0000 0003

```
eor.w   D2,D1       ;  (D1) = A3B0 0680
swap.w  D1          ;  (D1) = 0680 A3B0
subi.b  #1,D0
```

2ª pasada

(D0) = 0000 0002

```
eor.w   D2,D1       ;  (D1) = 0680 A035
swap.w  D1          ;  (D1) = A035 0680
subi.b  #1,D0
```

2ª pasada

(D0) = 0000 0001

```
eor.w   D2,D1       ;  (D1) = A035 0505
swap.w  D1          ;  (D1) = 0505 A035
subi.b  #1,D0
```

Nota: Para simularlo en el entorno WISM68, la instrucción `eor.w  D2,D1` debe tener invertido el orden de los operandos.

2000. Septiembre, original (gestión).

Después de ejecutar el siguiente programa:

```
Programa        equ     $1000
Datos           equ     $3000
                org     Datos

Sum             ds.l    1
DatosA          equ     $3010
DatosB          equ     $3020
DatosC          equ     $3030

                org     DatosA
                dc.w    $0001,$0002,$0003
                org     DatosB
                dc.w    $0004,$0005,$0006


                org     Programa

Comienzo        move.w  #3,D0
```

Indique los contenidos de:

a) El registro D1 la primera vez que se ejecuta el código entre la etiqueta InicioBucle y la siguiente instrucción de salto.
b) La posición Sum la segunda vez que se ejecuta el código entre la etiqueta InicioBucle y la siguiente instrucción de salto.
c) La posición dada por 0(A1,D2) la tercera vez que se ejecuta el código entre la etiqueta InicioBucle y la siguiente instrucción de salto.
d) El registro D2 al acabar el programa.
e) El registro D7 al acabar el programa.

Solución:

Traza:

1ª pasada

```
(D0) = 0000 0002
(D2) = 0000 0004

   move.w   0(A0,D2),D6          ;  (D6) = 0000 0003
   move.w   0(A1,D2),D7          ;  (D7) = 0000 0006
   muls     D6,D7 ;              ;  (D7) = 0000 0012
   add.l    D7,Sum               ;  (Sum)= 0000 0012
```

2ª pasada

E

```
(D0) = 0000 0001
(D2) = 0000 0002

   move.w   0(A0,D2),D6          ;  (D6) = 0000 0002
   move.w   0(A1,D2),D7          ;  (D7) = 0000 0005
   muls     D6,D7 ;              ;  (D7) = 0000 000A
```

3ª pasada

(D0) = 0000 0000
(D2) = 0000 0000

```
move.w    0(A0,D2),D6          ;   (D6) = 0000 0001
move.w    0(A1,D2),D7          ;   (D7) = 0000 0004
muls      D6,D7 ;              ;   (D7) = 0000 0004
add.l     D7,Sum               ;   (Sum)= 0000 0020
```

Después del bucle

```
move.l    Sum,(A2)             ; (3030) = 0000 0020
```

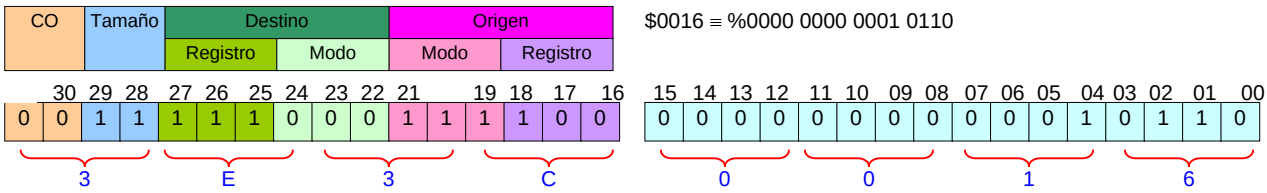
Comentario a la operación global que realiza el programa:

Existen dos vectores , u (almacenado en DatosA y apuntado por A0 junto con D2) y v (almacenado en DatosB y apuntado por A1 y D2). Los elementos de los vectores son de tamaño palabra. La operación global es el producto escalar de ambos vectores, acumulado en `Sum`. Después del bucle, el resultado se almacena en DatosC (tamaño palabra larga)

Genere el código máquina correspondiente a la instrucción del MC68000 `MOVE.W #$16,D7`.

Solución:

La instrucción MOVE tiene este formato:

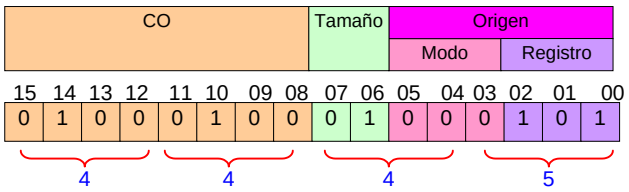


Campo Tamaño:
- 01 → byte
- 11 → palabra
- 10 → palabra larga

Genere el código máquina correspondiente a la instrucción del MC68000 `NEG.W D5`.

Solución:

La instrucción NEG tiene este formato:

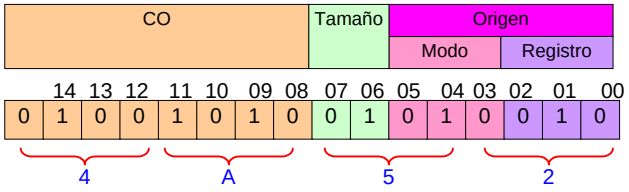


Campo Tamaño:
- 00 → byte
- 01 → palabra
- 10 → palabra larga

Genere el código máquina correspondiente a la instrucción del MC68000 `TST.W (A2)`.

Solución:

La instrucción TST tiene este formato:

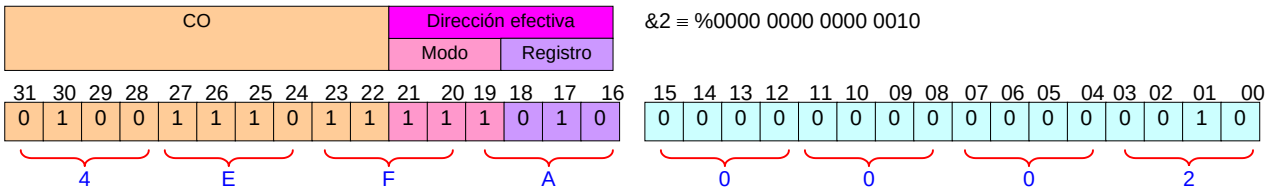


Campo Tamaño:
- 00 → byte
- 01 → palabra
- 10 → palabra larga

Genere el código máquina correspondiente a la instrucción del MC68000 `JMP 2(PC)`.

Solución:

Según el apéndice B de las UUDD, la instrucción JMP tiene este formato:



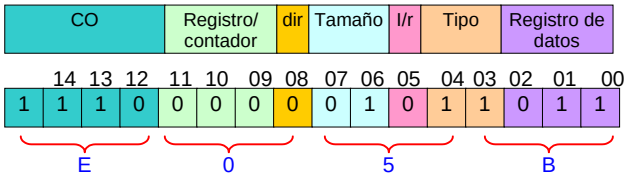
&2 ≡ %0000 0000 0000 0010

Nota: con asm68k.exe se obtiene 4EFA 0002, pero con 68kasm.exe se obtiene 4EFA FFF6.

Genere el código máquina correspondiente a la instrucción del MC68000 `ROR.W #8,D3`.

Solución:

Según el apéndice B de las UUDD, las instrucciones ASL, ASR, LSL, LSR, ROL, ROR y ROXL, cuando el destino es un registro tienen este formato:



E

Campo Tipo:
- Tipo = 00 → desplazamiento aritmético
- Tipo = 01 → desplazamiento lógico
- Tipo = 10 → rotación con extensión
- Tipo = 11 → rotación

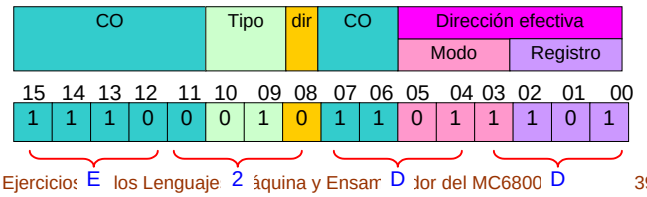
Campo dir:

2001. Febrero, primera semana (sistemas).

Encuentre el código hexadecimal correspondiente a la instrucción del M68000 `LSR (A5)+`.

Solución:

Según el apéndice B de las UUDD, las instrucciones ASL, ASR, LSL, LSR, ROL, ROR y ROXL, cuando el destino es una palabra de memoria tienen este formato:



Campo Tipo:
- Tipo = 00 → desplazamiento aritmético
- Tipo = 01 → desplazamiento lógico
- Tipo = 10 → rotación con extensión
- Tipo = 11 → rotación

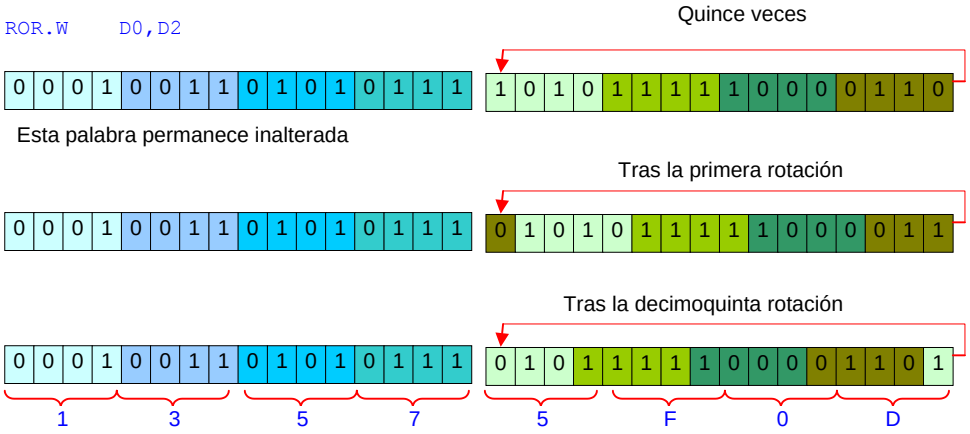
Campo dir:
- dir = 0 → derecha
- dir = 1 → izquierda

Señale cuál es el contenido del registro D2, siendo inicialmente (D0): 0000 000F, (D1): 0000 0010 y (D2): 1357 AF86 después de ejecutar las siguientes instrucciones:.

```
ROR.W     D0,D2
MULU      DI,D2
ANDI.B    #$F0,D2
```

Solución:

Traza:



D 0 D2, no varía, (D2) = 0005 F0D0

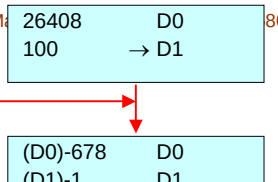
2001. Febrero, segunda semana (sistemas).

Señale cuál es el contenido del registro D0.W, después de ejecutar las siguientes instrucciones:

```
          MOVE.W    #$6728,D0
          MOVE.W    #100,D0
BUCLE     SUB.W     #$02A6,D0
          SUBQ.W    #1,D1
          BNE       BUCLE
          END
```

Solución:

Claramente, tenemos este bucle:

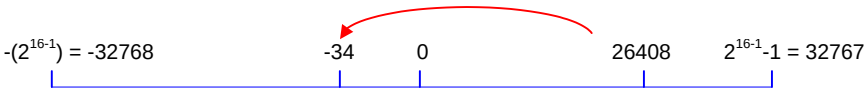


En principio, el resultado final sería $(D0)_{final} = 26408 – 100 \cdot 678 =$
Pero hay que ir con pies de plomo. Pueden suceder varios tipos de eventos. Por ejemplo, como:
$26408 \div 678 = 38.9$

cuando vamos por la iteración número 38:
$(D0)_{38} = 26408 – 38 \cdot 678 = 644 \quad \Rightarrow N=0, V=0$

y en la iteración número 39:
$(D0)_{38} = 644 – 678 = -34 \quad \Rightarrow N=1, V=0$

Este evento es irrelevante en nuestro análisis. Esto es así gracias a que el microprocesador tiene en D0 el número –34 representado en complemento a dos. Por tanto, el resultado hasta ahora es correcto (y en las siguientes restas). Otra forma de entenderlo es observando que el valor del contenido no se ha salido del rango en complemento a dos para números de 16 bits (el sufijo es .W).



Los eventos que sí son relevantes son los desbordamientos. Podemos seguir iterando hasta que:
$(D0) < -(2^{n-1}) = -(2^{16-1}) = -32768$

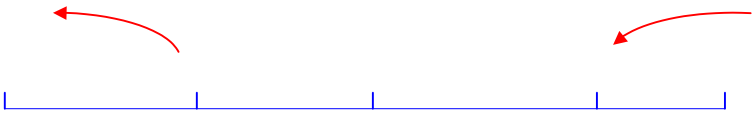
Para calcular el número de iteración en que ocurre esto:
Mientras $26408 - i \cdot 678 \geq -32768$ todo correcto
$26408 + 32768 \geq i \cdot 678$
$i \leq (26408 + 32768) \div 678 = 87.2 \quad \Rightarrow$ el desbordamiento ocurrirá en la iteración número 88

Veamos cuál es la situación en la iteración número 87:
$(D0)_{87} = 26408 – 87 \cdot 678 = -32578 \quad \Rightarrow N=1, V=0$

Notemos que el contenido binario es &-32578 ≡ $80BE. Este valor hexadecimal (binario compactado) lo necesitamos para hacer los cálculos en la siguiente iteración.

Veamos cuál es la situación en la iteración número 88:
$(D0)_{87}$ = $80BE –$02A6 = $7E18 = &32280 $\Rightarrow N=0, V=1$ (Al restar de un número negativo, obtenemos un número positivo)

Gráficamente, el desbordamiento es esto, salimos por un extremo del rango y entramos por el otro:

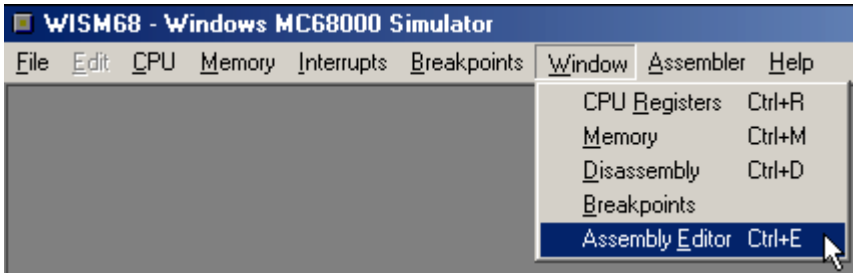


A continuación seguimos restando, pero claro, de este valor que “nos aparece” en los 16 bits menos significativos de D0. Nos faltan (100-88)=12 iteraciones. No vuelve a ocurrir desbordamiento, pues
$32280-12 \cdot 678 = 24144 \geq -32768$.

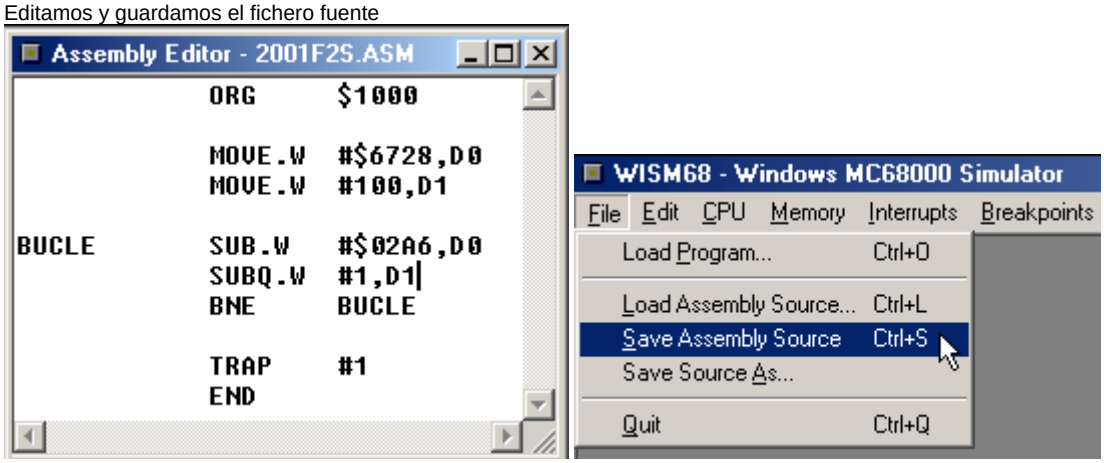
Finalmente, el contenido de D0 es:
&24144 = $5E50

Para hacer las comprobaciones con un simulador, elegimos el entorno integrado WISM68.

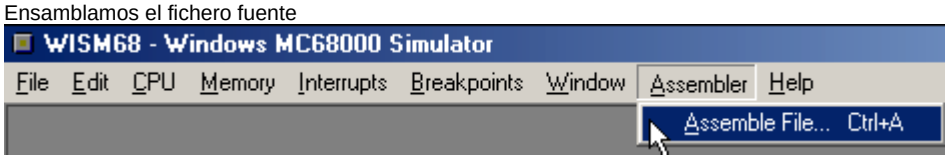
1º Abrimos la ventana del editor



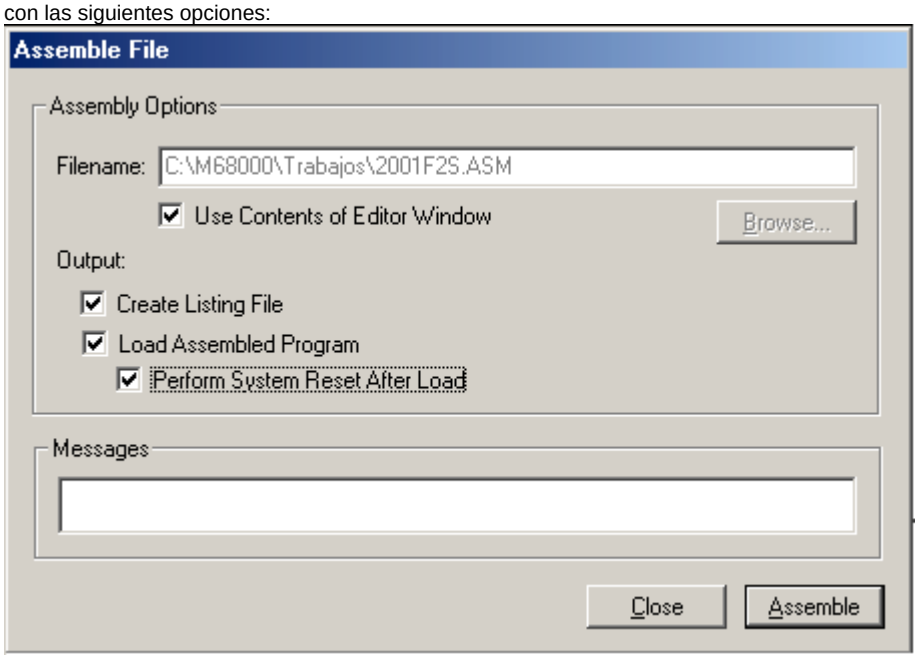
2º Editamos y guardamos el fichero fuente



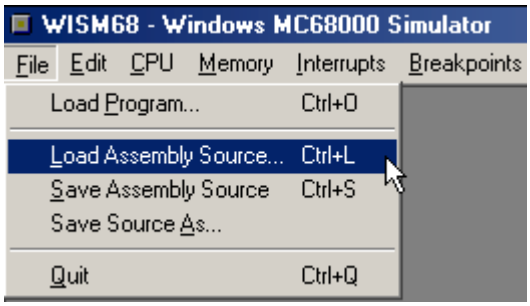
3º Ensamblamos el fichero fuente



con las siguientes opciones:

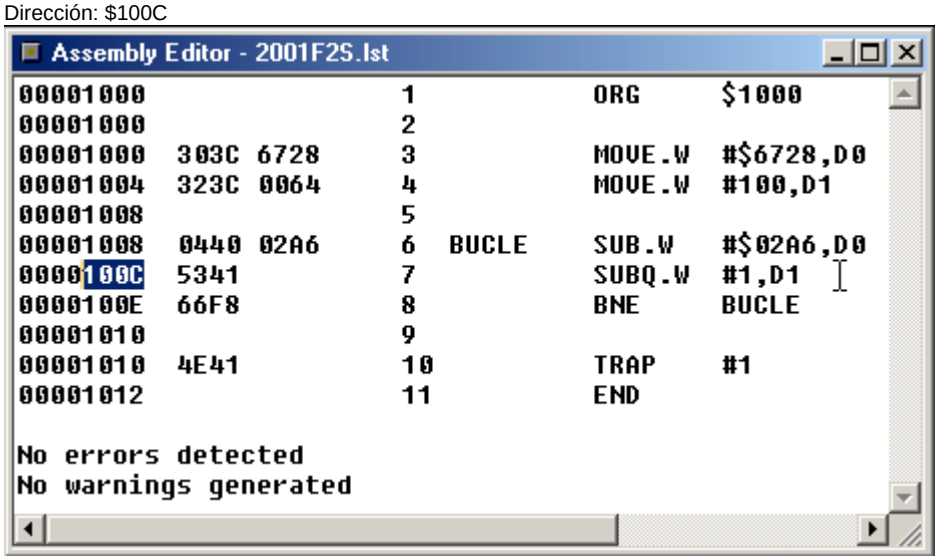


4º Vamos a establecer un punto de ruptura en la instrucción `BNE BUCLE`. Para ver en qué dirección está, abrimos el fichero listado

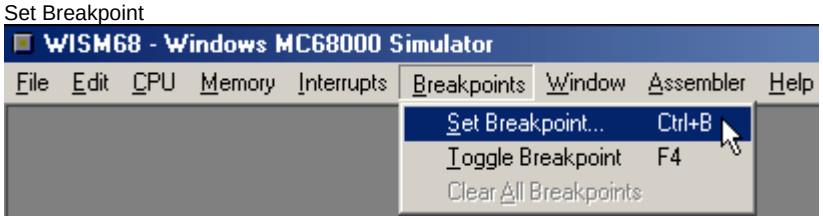


El diálogo sólo nos muestra los ficheros con extensión “*.asm”. Es necesario escribir “*.lst” en el cuadro “Nombre de archivo”.

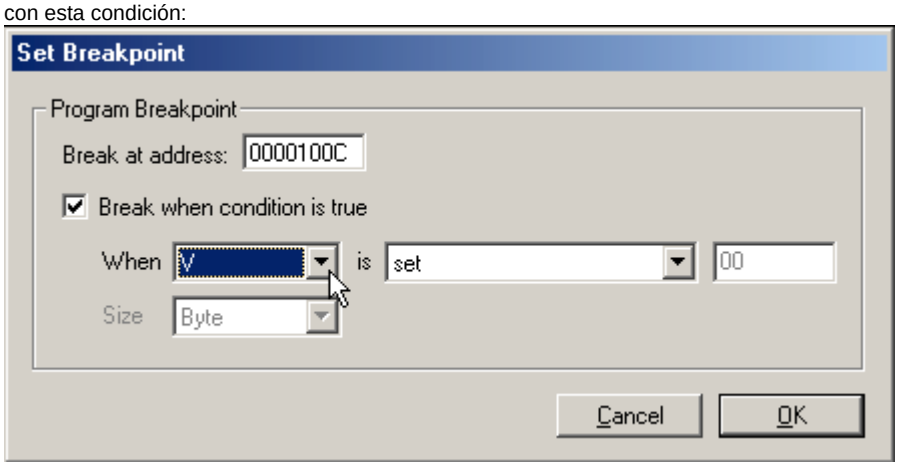
5º Dirección: $100C



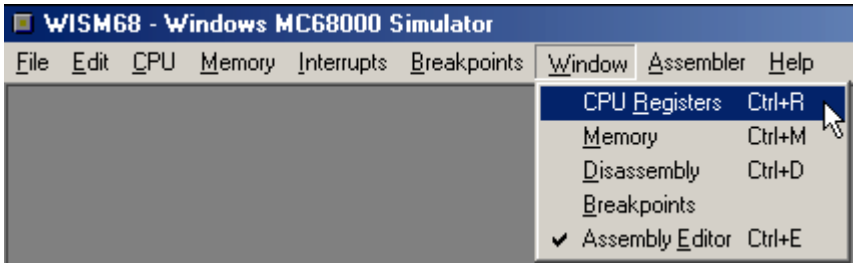
6º Set Breakpoint



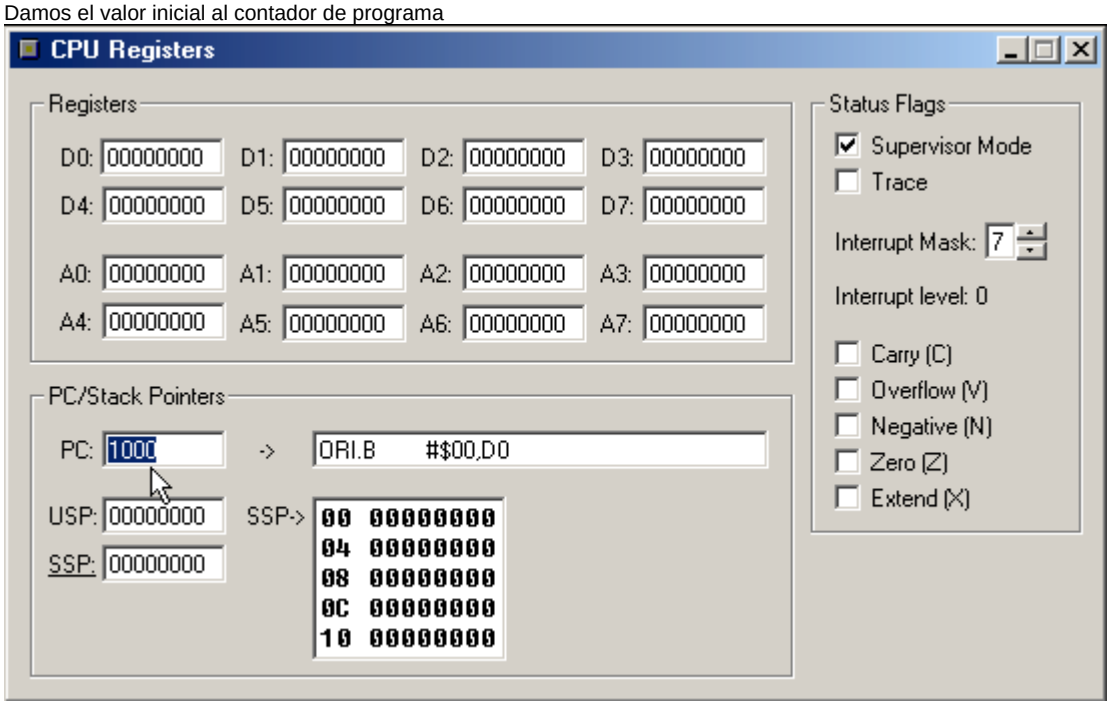
con esta condición:



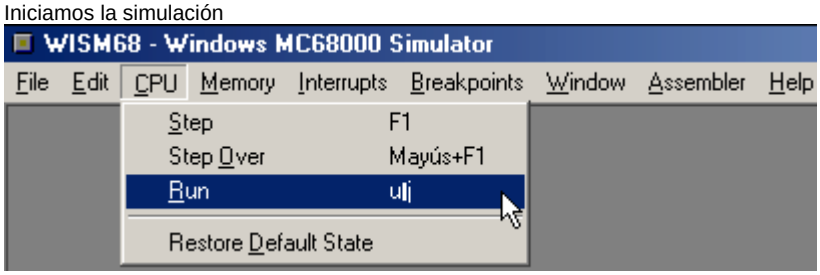
7º Abrimos la ventana de los registros



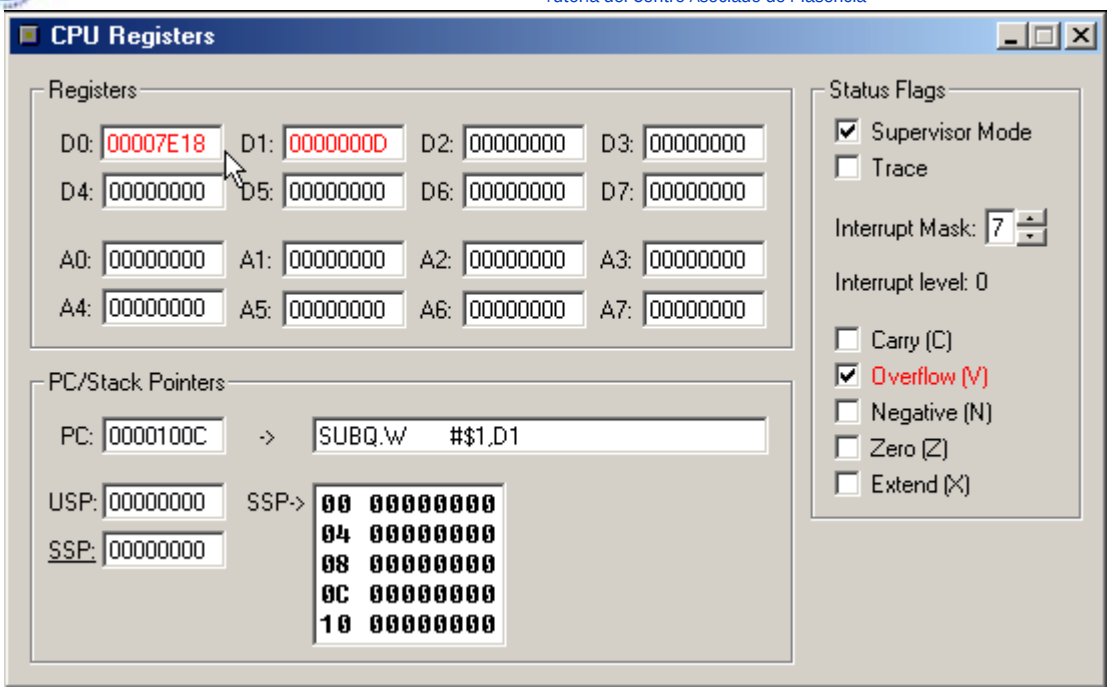
8º Damos el valor inicial al contador de programa



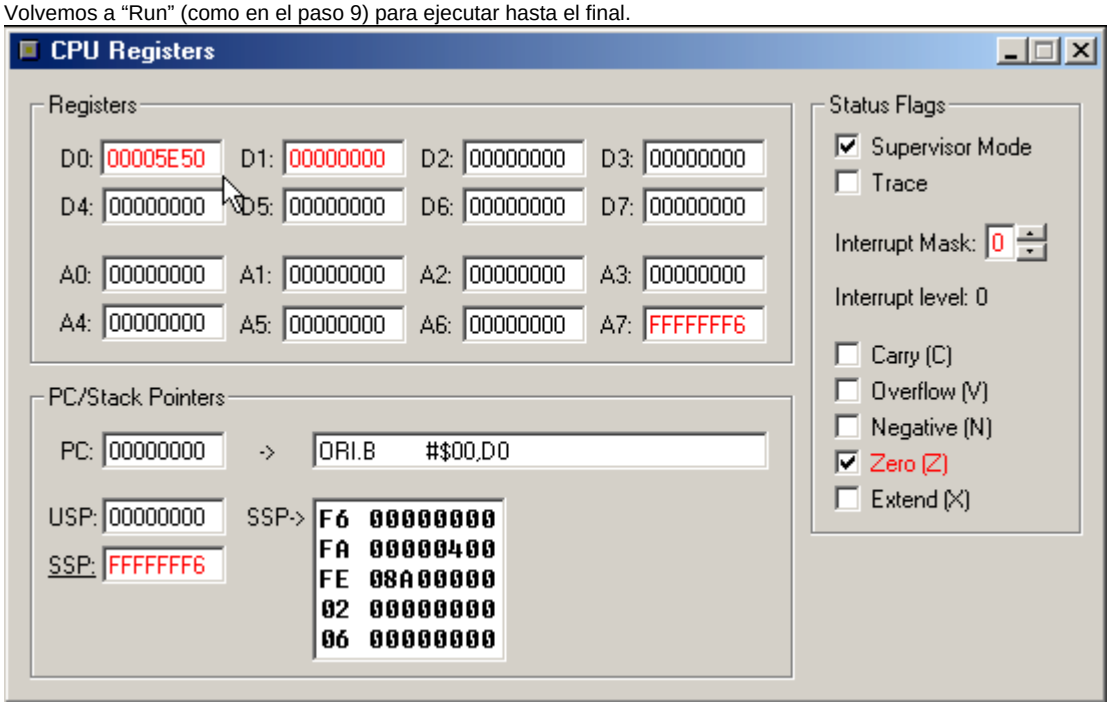
9º Iniciamos la simulación



10º Resultado parcial cuando vamos por la iteración 87 (cuando quedan 100-87=13 = $D):



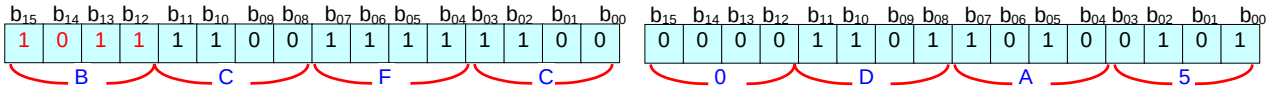
11º Volvemos a “Run” (como en el paso 9) para ejecutar hasta el final.



Encuentre la instrucción en ensamblador del M68000 a la que corresponde el código máquina BCFC0DA5:

Solución:

1º Descomprimimos el código máquina.



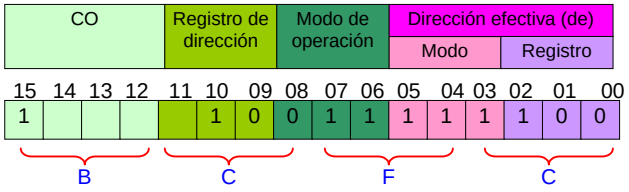
2º Tras comparar el código de operación de todas las instrucciones del apéndice B, vemos que sólo son compatibles aquellas que tienen 1011 en sus bits más significativos: CMP, CMPA, CMPM y EOR.

3º Descartamos CMPM, pues para ésta, debería ser b_{08}=1. Tenemos CMP, CMPA y EOR.

4º Descartamos EOR, pues debería ser b_{08}=1 (aunque no forme parte del código de operación, en los modos de operación sólo es posible b_{08}=1). Tenemos CMP y CMPA.

5º De estas dos que nos quedan, los bits b_{08} b_{07} b_{06}= 011 sólo son posibles en CMPA.

6º Formato de la instrucción CMPA:



Campo del modo de operación:

Palabra	Palabra larga
011	111

El registro de dirección es A6
Como el modo de operación es 011, el sufijo es W.
Hasta ahora conocemos que se trata de `CMPA.W a1,A6`

7º En la tabla 15.3 (edición antigua) de la codificación combinada modo-registro, la codificación 111 100 corresponde a un dato inmediato (el almacenado en la segunda palabra)

Por tanto, finalmente: `CMPA.W #$0DA5,A6`
O también: `CMPA.W #3493,A6`

2001. Septiembre, original (sistemas).

Señale cuál es el contenido del registro D0 después de ejecutar las siguientes instrucciones:

```
            ORG     2500
COM         EQU     $F5F
            MOVE.L  #$000F0481,D0
            ADD.L   NUL,D0
            AND     #COM,D0
NUL         DC.L    $42
            END
```

Solución:

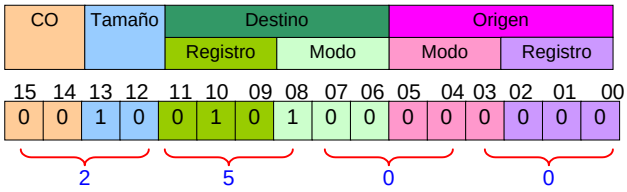
Traza:

```
MOVE.L  #$000F0481,D0       ;   000F0481                    → D0

ADD.L   NUL,D0              ;   (D0)        +   00000042    → D0
                            ;   000F0481    +   00000042    → D0
                            ;   000F04C3                    → D0

AND     #COM,D0             ;   (D0)        AND COM         → D0
                            ;   000F04C3    AND 0F5F        → D0
                            ;   000F0443                    → D0
```

Genere el código máquina producido por la instrucción `MOVE.L  D0,-(A2)`:

Solución:

La instrucción MOVE tiene este formato:



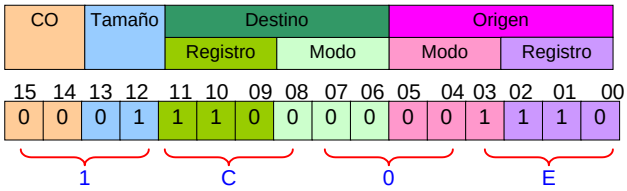
Campo Tamaño:
- 01 → byte
- 11 → palabra
- 10 → palabra larga

2002. Febrero, primera semana (sistemas).

Generar el código máquina producido por la instrucción MOVE.B A6, D6

Solución:

La instrucción MOVE tiene este formato:



Campo Tamaño:
- 01 → byte
- 11 → palabra
- 10 → palabra larga

Después de ejecutarse el siguiente segmento de código del M68000 cuál es el contenido correcto de las siguientes posiciones de memoria:

```
        ORG     $6000
DAT     DC.W    $1234,$ABCD
        DC.W    $5678,$90EF

COD     CLR.L D1
        MOVEA.L #DAT,A2
        MOVE.L  (A2)+,D0
        MOVE.W  (A2)+,D1
        EOR.W   D0,D1
        MOVE.W  D0,(A2)+
        MOVE.L  D1,-(A2)
```

Solución:

Tabla de símbolos

Símbolo	Valor
DAT	$00 60 00
COD	$00 60 08

Mapa de la memoria inicial

$00 60 00	$12
$00 60 01	$34
$00 60 02	$AB
$00 60 03	$CD
$00 60 04	$56
$00 60 05	$78
$00 60 06	$90
$00 60 07	$EF

Traza:

Tras las dos primeras instrucciones de la zona de código

```
COD   CLR.L D1              ; (D1) = $00 00 00 00
      MOVEA.L  #DAT,A2      ; (A2) = $00 00 60 00
```

A2 $00 00 60 00
D1 $00 00 00 00
D0

$00 60 00	$12
$00 60 01	$34
$00 60 02	$AB
$00 60 03	$CD
$00 60 04	$56
$00 60 05	$78
$00 60 06	$90
$00 60 07	$EF

Tras la tercera instrucción

```
      MOVE.L   (A2)+,D0     ; (D0) = $12 34 AB CD
                            ; (A2) = $00 00 60 04
```

A2 $00 00 60 04
D1 $00 00 00 00
D0 $12 34 AB CD

$00 60 00	$12
$00 60 01	$34
$00 60 02	$AB
$00 60 03	$CD
$00 60 04	$56
$00 60 05	$78
$00 60 06	$90
$00 60 07	$EF

Tras la cuarta instrucción

```
      MOVE.W   (A2)+,D1     ; (D1) = $00 00 56 78
                            ; (A2) = $00 00 60 06
```

A2 $00 00 60 06
D1 $00 00 56 78
D0 $12 34 AB CD

$00 60 00	$12
$00 60 01	$34
$00 60 02	$AB
$00 60 03	$CD
$00 60 04	$56
$00 60 05	$78
$00 60 06	$90
$00 60 07	$EF

Tras la quinta instrucción

```
      EOR.W    D0,D1        ; (D1) = $00 00 56 78
```

```
   %  0101   0110   0111   1000
⊕  %  1010   1011   1100   1101
   ---------------------------
   %  1111   1101   1011   0101  = $ FDB5
```

A2 $00 00 60 06
D1 $00 00 FD B5
D0 $12 34 AB CD

$00 60 00	$12
$00 60 01	$34
$00 60 02	$AB
$00 60 03	$CD
$00 60 04	$56
$00 60 05	$78
$00 60 06	$90
$00 60 07	$EF

Tras la sexta instrucción

```
      MOVE.W   D0,(A2)+     ; ($6006) = $AB CD
                            ; (A2) = $00 00 60 08
```

A2 $00 00 60 08
D1 $00 00 FD B5
D0 $12 34 AB CD

$00 60 00	$12
$00 60 01	$34
$00 60 02	$AB
$00 60 03	$CD
$00 60 04	$56
$00 60 05	$78
$00 60 06	$AB
$00 60 07	$CD

Tras la séptima instrucción

```
      MOVE.L   D1,-(A2)     ; (A2) = $00 00 60 04
                            ; ($6004) = $00 00 56 78
```

A2 $00 00 60 06
D1 $00 00 FD B5
D0 $12 34 AB CD

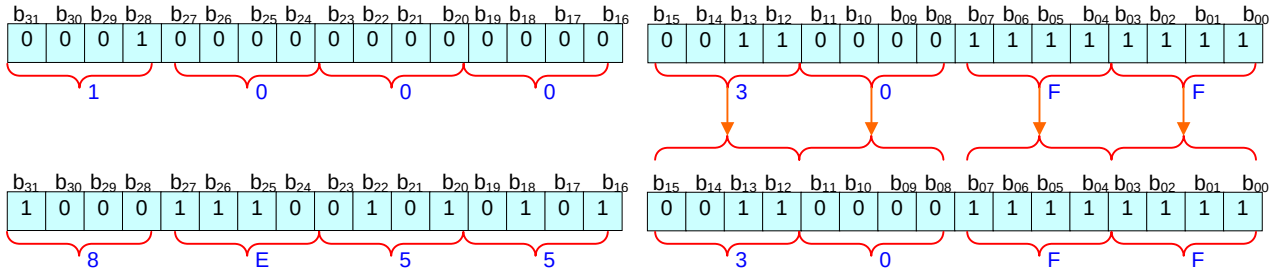
$00 60 00	$12
$00 60 01	$34
$00 60 02	$AB
$00 60 03	$CD
$00 60 04	$00
$00 60 05	$00
$00 60 06	$FD
$00 60 07	$B5

2002. Febrero, primera semana (gestión).

Suponiendo que el contenido inicial de los registros D0 y D1 es (D0) = $100030FF y (D1) =$8E552900 ¿Cuál es el contenido de los mismos después de ejecutarse la instrucción `MOVE.W D0,D1`?

Solución:

```
MOVE.W   D0,D1
```

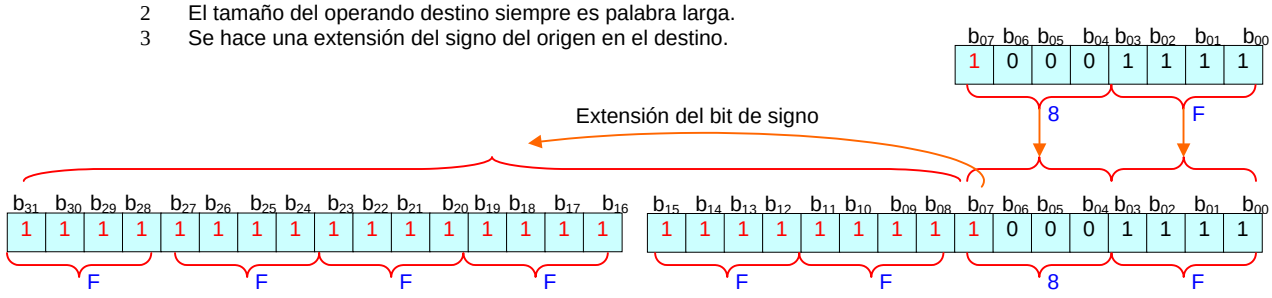


Siendo el contenido de D3, (D3)=$100030FF, ¿Cuál será después de ejecutarse la instrucción `MOVEQ #$8F,D3`?

Solución:

En la instrucción
`MOVEQ #n,Di`

1 *n* es un número inmediato de 8 bits. Por tanto, en nuestro caso, el bit de signo es 1.
2 El tamaño del operando destino siempre es palabra larga.
3 Se hace una extensión del signo del origen en el destino.

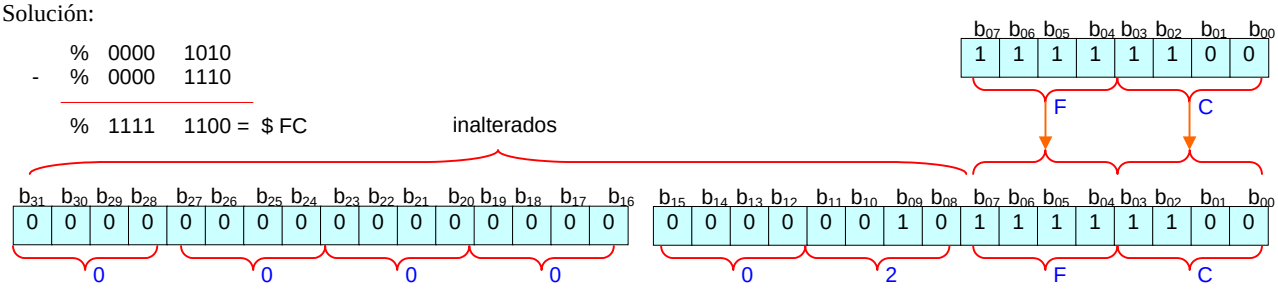


Si el contenido inicial de D0 es (D0) = $0000020A ¿ Cual es el contenido de D0 después de ejecutarse la instrucción `SUBI.B #$E,D0`?

a) $000001FC.
b) $000002FC.
c) $00000218.
d) $000000FC.

Solución:

```
      %  0000   1010
  -   %  0000   1110
      ------------------
      %  1111   1100 =  $ FC        inalterados
```



Se tiene un sistema de computación basado en el microprocesador MC68000. En un momento dado, este es el contenido de sus registros y memoria (valores expresados en hexadecimal)

D0	0000 020A	A0	0002 0000
D1	0000 FFFF	A1	0000 0000
D2	3582 9EFA	A2	0000 0000
D3	4350 3B89	A3	0000 0000
D4	0000 0000	A4	0002 0004
D5	FFFF FFFF	A5	0000 0000
D6	FFFF FFFF	A6	0000 0000
D7	0000 0000	A7	0000 0FF0

Dirección	Contenido
$020000	$3C
$020001	$4A
$020002	$05
$020003	$13
$020004	$10
$020005	$A5
$020006	$BF
$020007	$38
$020008	$FF
$020009	$40

a) ¿Cuál es el valor de D1 después de la ejecución de las instrucciones del bloque A?.

BLOQUE A

```
OR.W     D1,D2;
AND.B    D2,D0;
EOR.L    D0,D1
```

b) Compruebe el resultado de D4 después de la ejecución de las instrucciones contenidas en bloque B

BLOQUE B

```
MOVE.L#  35829EFA, D2
MOVE.L   #$00000000,D4

ADDA.W   #$4,A0
MOVE.W   -(A0),D4
EOR.B    D2,D4
```

Solución:

a) Traza:

Tras la primera instrucción

```
OR.W  D1,D2          ; (D2) = $35 82 FF FF

   %  1111  1111  1111  1111
v  %  1001  1110  1111  1010
-------------------------------
   %  1111  1111  1111  1111 = $ FF FF
```

Tras la segunda instrucción

```
AND.B    D2,D0       ; (D0) = $00 00 02 0A

   %  1111  1111
^  %  0000  1010
-----------------
   %  0000  1010 = $ 0A
```

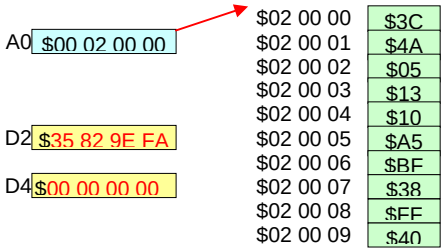
Tras la tercera instrucción

```
EOR.L    D0,D1       ; (D1) = $ 00 00 FD F5

   %  0000  0000  0000  0000  0000  0010  0000  1010
⊕  %  0000  0000  0000  0000  1111  1111  1111  1111
-------------------------------------------------------
   %  0000  0000  0000  0000  1111  1101  1111  0101 = $ 00 00 FD F5
```

b) Traza:

Tras las dos primeras instrucciones

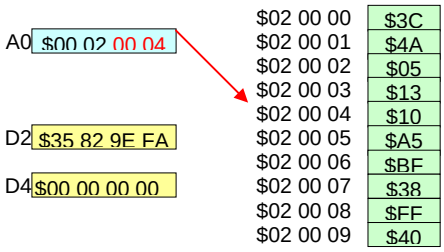
```
MOVE.L   #$35829EFA,D2        ; (D2) = $35 82 9E FA
MOVE.L   #$00000000,D4        ; (D4) = $00 00 00 00
```



Tras la tercera instrucción

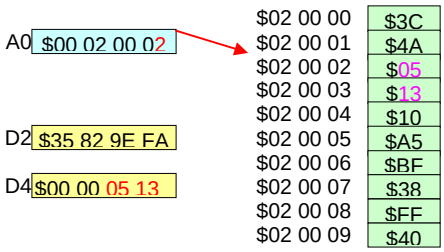
```
ADDA.W    #$4,A0                ; (A0) = $00 02 00 04
```

```
     %  0000   0000   0000   0100
  +  %  0000   0000   0000   0000
     %  0000   0000   0000   0100  = $ 00 04
```



Tras la cuarta instrucción

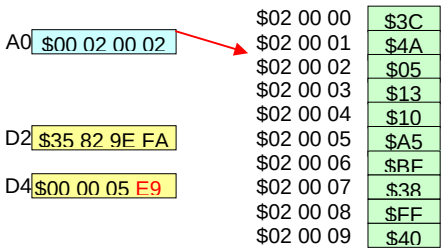
```
MOVE.W    -(A0),D4              ; (A0) = $00 02 00 02
                                ; (D4) = $00 00 05 13
```



Tras la tercera instrucción

```
EOR.B     D2,D4                 ; (D4) = $00 00 05 E9
```

```
     %  1111   1010
  ⊕  %  0001   0011
     %  1110   1001  = $ E9
```



Después de ejecutarse el siguiente segmento de código del M68000 cuál es el contenido correcto del registro D0.B:

```
        ORG     $6000
DAT     DC.W    $1234,$ABCD
        DC.W    $5678

COD     MOVEQ   #5,D1
        MOVE.B  D1,D0
        MOVEA.L #DAT,A0

BUC     OR.B    (A0)+,D0
        BCHG    D1,D0
        SUBQ.B  #1,D1
```

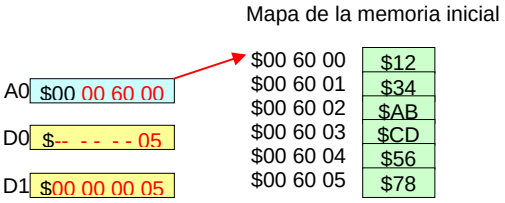
Solución:

Tabla de símbolos

Símbolo	Valor
DAT	$00 60 00
COD	$00 60 05
BUC	No nos importa

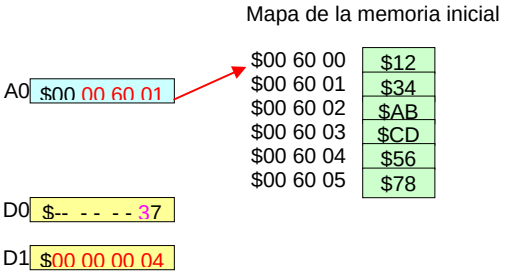
Traza:

Antes de entrar en el bucle



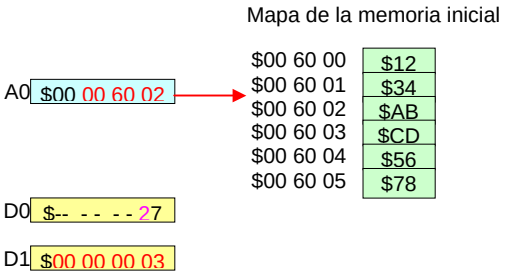
Iteración Nº 1

```
OR.B   (A0)+,D0

   %   0001   0010
OR %   0000   0101
   %   0001   0111

BCHG   D1,D0 ; Invierte

SUBQ.B   #1,D1
```



Iteración Nº 2

```
OR.B   (A0)+,D0

   %   0011   0100
OR %   0011   0111
   %   0011   0111

BCHG   D1,D0 ; Invierte

SUBQ.B   #1,D1
```



Iteración Nº 3

```
OR.B   (A0)+,D0

   %  1010   1011
OR %  0010   0111
   %  1010   1111

BCHG   D1,D0 ; Invierte

SUBQ.B    #1,D1
```

A0 $00 00 60 03

D0 $-- -- -- A7

D1 $00 00 00 02

Mapa de la memoria inicial

$00 60 00	$12
$00 60 01	$34
$00 60 02	$AB
$00 60 03	$CD
$00 60 04	$56
$00 60 05	$78

Iteración Nº 4

```
OR.B   (A0)+,D0

   %  1100   1101
OR %  1010   0111
   %  1110   1111

BCHG   D1,D0 ; Invierte

SUBQ.B    #1,D1
```

A0 $00 00 60 04

D0 $-- -- -- FB

D1 $00 00 00 01

Mapa de la memoria inicial

$00 60 00	$12
$00 60 01	$34
$00 60 02	$AB
$00 60 03	$CD
$00 60 04	$56
$00 60 05	$78

Iteración Nº 5

```
OR.B   (A0)+,D0

   %  0101   0110
OR %  1110   1011
   %  1111   1111

BCHG   D1,D0 ; Invierte

SUBQ.B    #1,D1
BNE       BUC ; No salta
```

A0 $00 00 60 05

D0 $-- -- -- FD

D1 $00 00 00 00

Mapa de la memoria inicial

$00 60 00	$12
$00 60 01	$34
$00 60 02	$AB
$00 60 03	$CD
$00 60 04	$56
$00 60 05	$78