



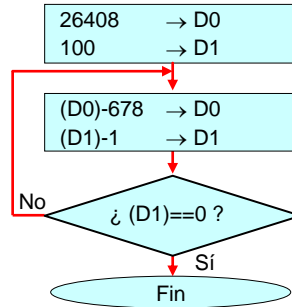
Señale cuál es el contenido del registro D0.W, después de ejecutar las siguientes instrucciones:

```

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

Solución:

Claramente, tenemos este bucle:



En principio, el resultado final sería $(D0)_{\text{final}} = 26408 - 100 \cdot 678 = 19628$

Pero hay que ir con pies de plomo. Pueden suceder varios tipos de eventos. Por ejemplo, como:

$$26408 \div 678 = 38.9$$

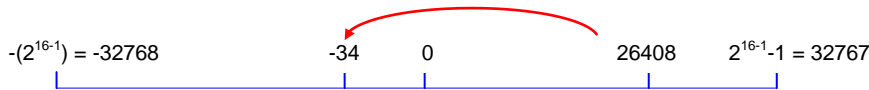
cuando vamos por la iteración número 38:

$$(D0)_{38} = 26408 - 38 \cdot 678 = 644 \Rightarrow N=0, V=0$$

y en la iteración número 39:

$$(D0)_{39} = 644 - 678 = -34 \Rightarrow N=1, V=0$$

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:

$$(D0) < -(2^{16-1}) = -(2^{16-1}) = -32768$$

Para calcular el número de iteración en que ocurre esto:

Mientras $26408 - i \cdot 678 \geq -32768$ todo correcto

$$26408 + 32768 \geq i \cdot 678$$

$$i \leq (26408 + 32768) \div 678 = 87.2 \Rightarrow \text{el desbordamiento ocurrirá en la iteración número 88}$$

Veamos cuál es la situación en la iteración número 87:

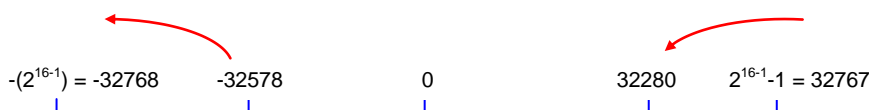
$$(D0)_{87} = 26408 - 87 \cdot 678 = -32578 \Rightarrow N=1, V=0$$

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:

$$(D0)_{88} = \$80BE - \$02A6 = \$7E18 = \&32280 \Rightarrow N=0, V=1 \quad (\text{Al restar de un número negativo, obtenemos un número positivo})$$

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

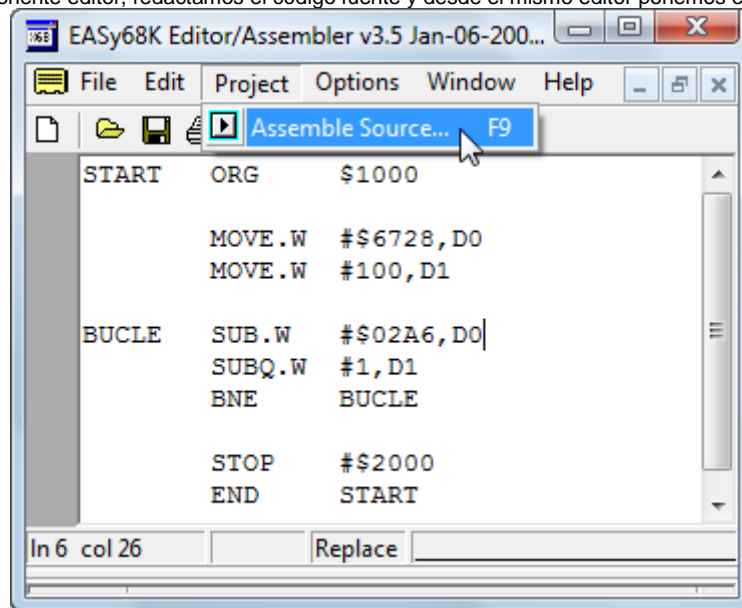
$$32280 - 12 \cdot 678 = 24144 \geq -32768.$$

Finalmente, el contenido de D0 es:

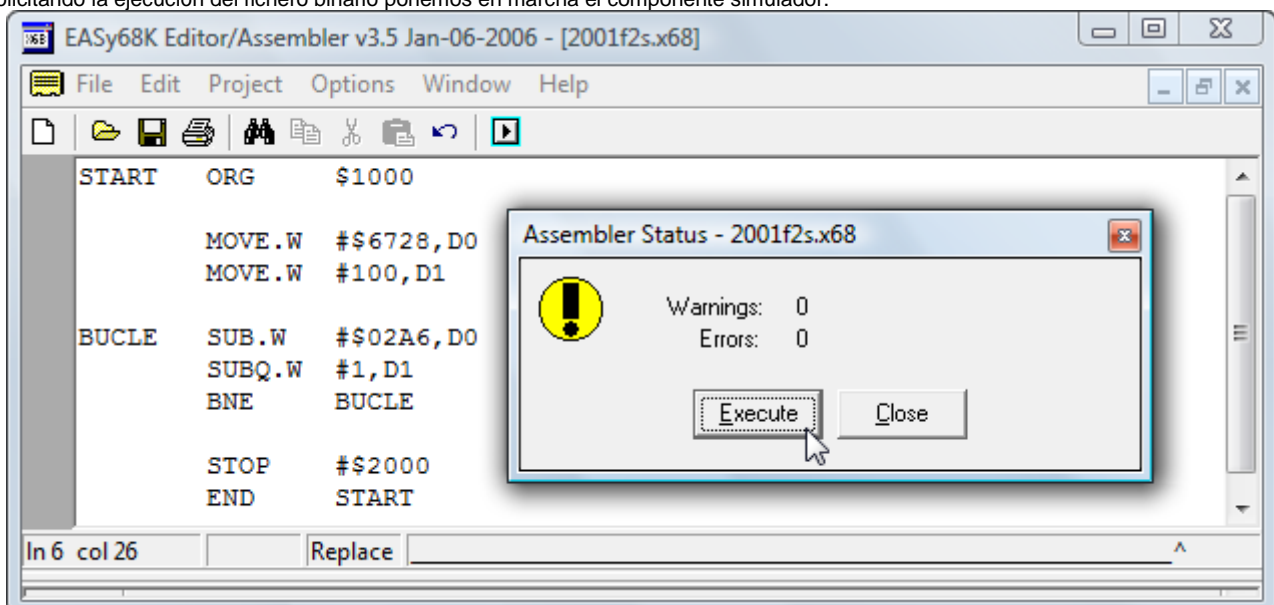
$$\&24144 = \$5E50$$

Vamos a comprobar la resolución mediante el simulador EASy68K

1º Abrimos la ventana del componente editor, redactamos el código fuente y desde el mismo editor ponemos en marcha la compilación.



2º Solicitando la ejecución del fichero binario ponemos en marcha el componente simulador.



3º El simulador ya ha cargado el fichero binario; y nos muestra (además de los contenidos de los registros) el fichero listado.

The screenshot shows the EASY68K Editor/Assembler v3.5 interface. The title bar indicates the file path: C:\M68000\Trabajos\2001f2s.S68. The menu bar includes File, Run, View, Options, and Help. Below the menu bar is a toolbar with various icons for file operations and execution.

The **Registers** section displays the following values:

Register	Value
D0	00000000
D1	00000000
D2	00000000
D3	00000000
D4	00000000
D5	00000000
D6	00000000
D7	00000000
A0	00000000
A1	00000000
A2	00000000
A3	00000000
A4	00000000
A5	00000000
A6	00000000
A7	01000000

Other status fields include: SR=0010000000000000, US=00FF0000, SS=01000000, PC=00001000. A **Cycles** counter shows 0, with a **Clear Cycles** button.

The main window displays the assembly code listing:

```

00001000 Starting Address
Assembler used: EASY68K Editor/Assembler v3.5 Jan-06-2006
Created On: 16/12/2008 8:15:50

00001000          1  START  ORG      $1000
00001000          2
00001000 303C 6728          3          MOVE.W  #$6728,D0
00001004 323C 0064          4          MOVE.W  #100,D1
00001008          5
00001008 0440 02A6          6  BUCLE   SUB.W   #$02A6,D0
0000100C 5341              7          SUBQ.W  #1,D1
0000100E 66F8              8          BNE     BUCLE
00001010          9
00001010 4E72 2000         10          STOP    #$2000
00001014          11          END      START

No errors detected
No warnings generated

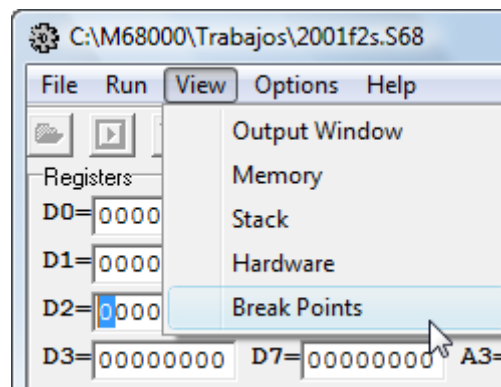
SYMBOL TABLE INFORMATION
Symbol-name      Value
-----
BUCLE             1008
START             1000
  
```

The status bar at the bottom shows: S0 = 68KPROG 11CREATED BY EASY68K .S68 file read successful

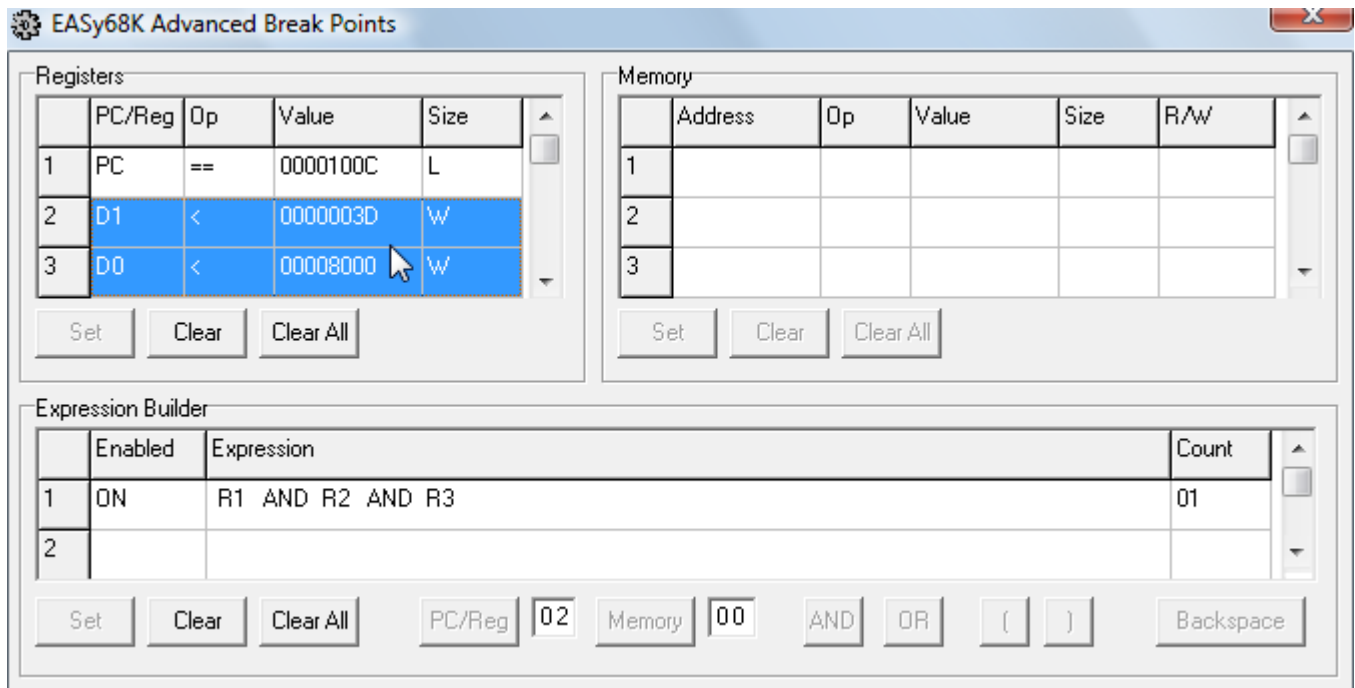
Este fichero listado lo necesitaremos para conocer en qué dirección está la instrucción donde colocaremos un punto de ruptura:

`SUBQ.W #1,D1`

Dirección: \$100C



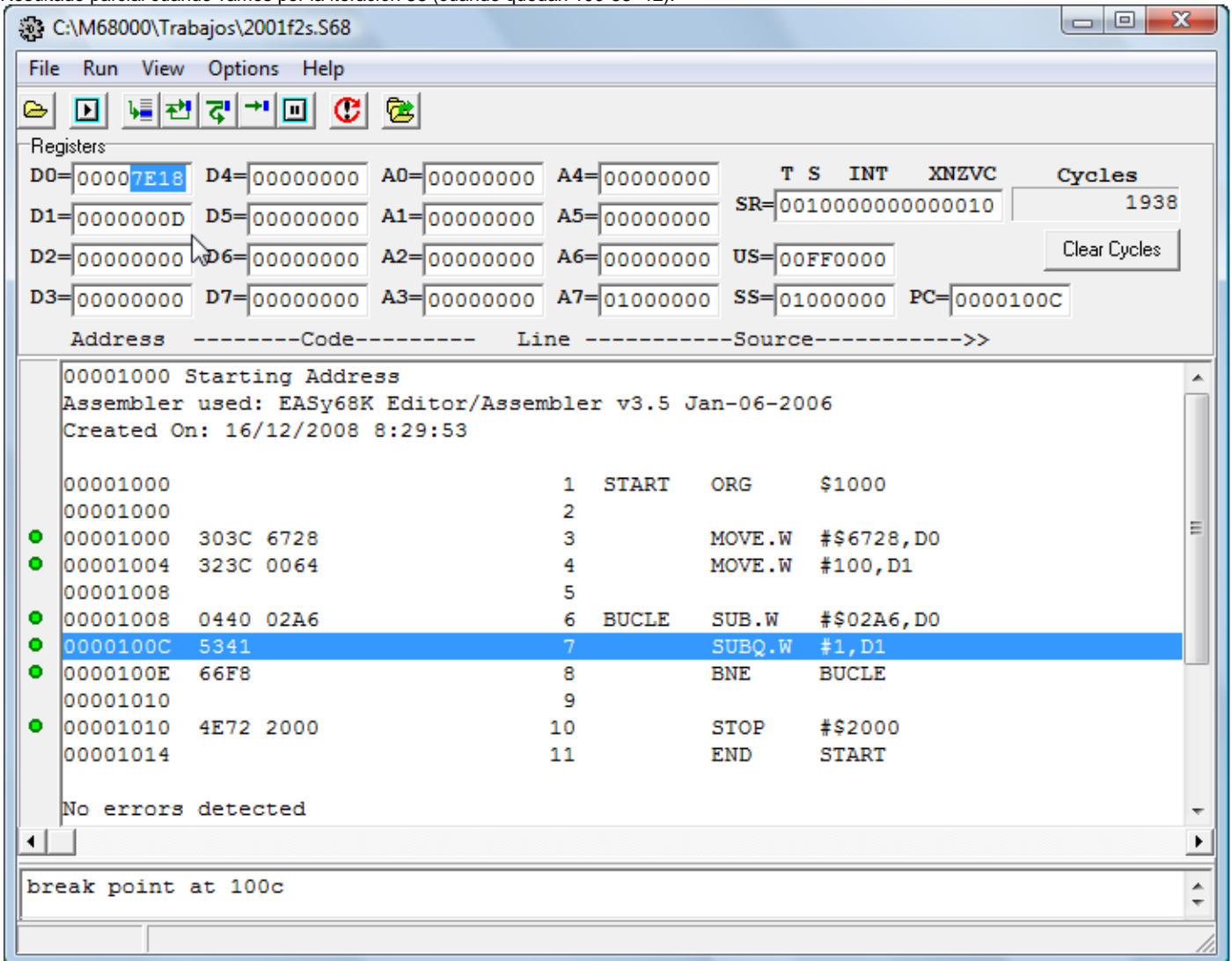
Atención a las condiciones de ruptura:



Pretendemos que la ejecución se detenga cuando (PC)=100C, pero sólo en la iteración en que tiene lugar el desbordamiento en D0. Es decir cuando su bit 15 pasa de '1' a '0'. En otras palabras, cuando (D0) pasa de ser mayor que \$8000 a menor que \$8000. Además, para evitar el hecho de que en las primeras iteraciones (antes del cruce por cero) se tenía (D0) positivo, nos aseguramos dejar atrás estas iteraciones dándole un valor al contador posterior al cruce por cero. Por ejemplo (D1) = 100-39 = 61 = \$3D.

5º Ponemos en marcha la simulación hasta el punto de ruptura.

Resultado parcial cuando vamos por la iteración 88 (cuando quedan 100-88=12):



The screenshot shows the EASy68K Editor/Assembler v3.5 interface. The title bar indicates the file path is C:\M68000\Trabajos\2001f2s.S68. The menu bar includes File, Run, View, Options, and Help. Below the menu bar is a toolbar with various icons for file operations and execution. The main window is divided into several sections:

- Registers:** A grid showing the state of 16 registers (D0-D7, A0-A7, T, S, INT, XNZVC, SR, US, SS, PC). D0 is highlighted with the value 7E18. The Cycles counter shows 1938.
- Address/Code/Line/Source:** A table showing the assembly code. The current instruction at address 0000100C is highlighted in blue.
- Assembly Code:**

```

00001000      1  START   ORG     $1000
00001000      2
00001000  303C 6728      3          MOVE.W  #$6728,D0
00001004  323C 0064      4          MOVE.W  #100,D1
00001008      5
00001008  0440 02A6      6  BUCLE   SUB.W   #$02A6,D0
0000100C  5341          7  SUBQ.W  #1,D1
0000100E  66F8          8          BNE     BUCLE
00001010      9
00001010  4E72 2000     10          STOP   #$2000
00001014     11          END     START

```
- Status:** A message at the bottom left states "No errors detected".
- Break Point:** A text box at the bottom indicates "break point at 100c".

Como habíamos calculado antes (D0)=\$7E18.

Hay que hacer notar que (D1)=\$0D=13 y no \$0C=12 porque la interrupción ha ocurrido inmediatamente antes de ejecutarse la instrucción almacenada en la dirección \$100C (recuérdese que el PC apunta a la siguiente, la cual todavía no se ha ejecutado).

6º Para continuar la simulación hasta el final hemos de desactivar el punto de ruptura.

EASy68K Advanced Break Points

Registers

	PC/Reg	Op	Value	Size
1	PC	==	0000100C	L
2	D1	<	0000003D	W
3	D0	<	00008000	W

Set Clear Clear All

Memory

	Address	Op	Value	Size	R/W
1					
2					
3					

Set Clear Clear All

Expression Builder

	Enabled	Expression	Count
1	OFF	R1 AND R2 AND R3	01
2			

Set Clear Clear All PC/Reg 03 Memory 00 AND OR () Backspace

C:\M68000\Trabajos\2001f2s.S68

File Run View Options Help

Registers:

D0	D1	D2	D3	D4	D5	D6	D7	A0	A1	A2	A3	A4	A5	A6	A7	T	S	INT	XNZVC	Cycles
00005E50	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	01000000	001000000000000000	00FF0000	01000000	00001014	2222

SR=001000000000000000 US=00FF0000 SS=01000000 PC=00001014

Address -----Code----- Line -----Source----->>

00001000		1	START	ORG	\$1000
00001000		2			
00001000	303C 6728	3		MOVE.W	#\$6728,D0
00001004	323C 0064	4		MOVE.W	#100,D1
00001008		5			
00001008	0440 02A6	6	BUCLE	SUB.W	#\$02A6,D0
0000100C	5341	7		SUBQ.W	#1,D1
0000100E	66F8	8		BNE	BUCLE
00001010		9			
00001010	4E72 2000	10		STOP	#\$2000
00001014		11	END	START	

STOP instruction. Execution halted

Observamos que el resultado es el previsto (D0)=\$E50.