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:

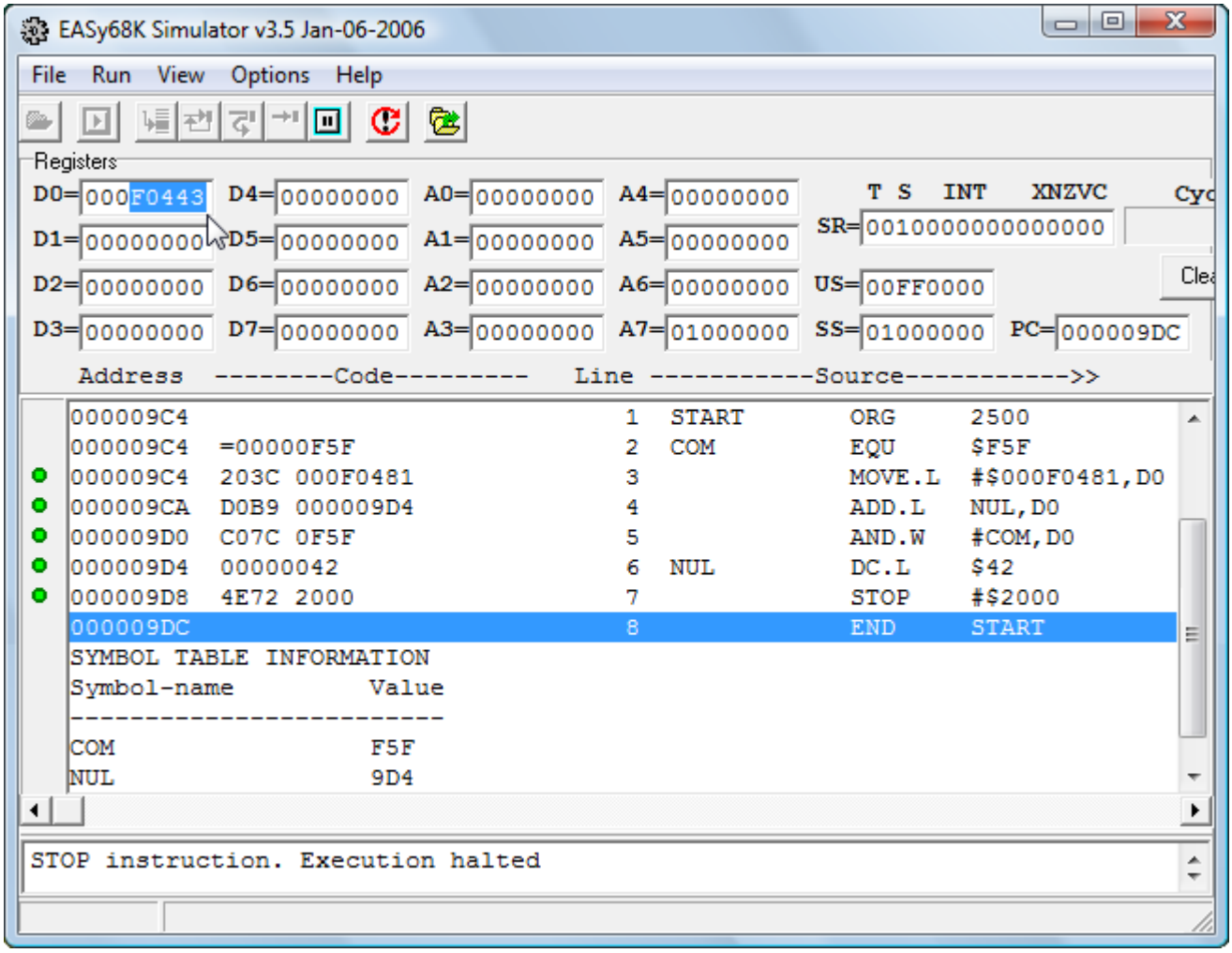
Traza:

```
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
```

Podemos comprobar el resultado con el simulador EASy68K:

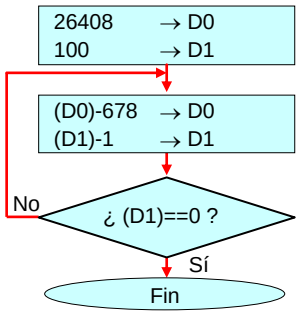


Señale cuál es el contenido del registro `D0.W`, después de ejecutar las siguientes instrucciones:

```
        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:

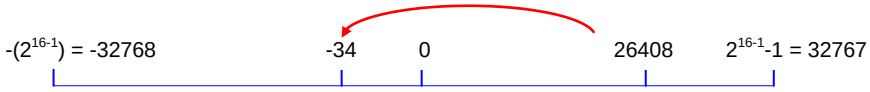


En principio, el resultado final sería $(D0)_{final} = 26408 – 100 \cdot 678 = -41392$
Pero hay que ir con pies de plomo. Pueden suceder varios tipos de eventos. Por ejemplo, como:
$26408 \div 678 = 38.9$

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:
$(D0)_{38} = 644 – 678 = -34$ $\Rightarrow$ 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:
$(D0) < -(2^{n-1}) = -(2^{16-1}) = -32768$

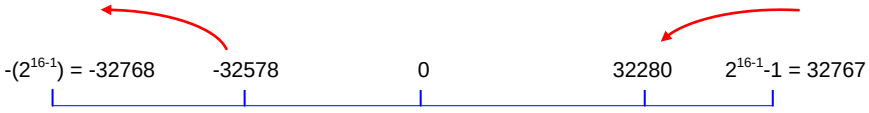
Para calcular el número de iteración en que ocurre esto:
Mientras $26408 - i \cdot 678 \geq -32768$ todo correcto
$26408 + 32768 \geq i \cdot 678$
$i \leq (26408 + 32768) \div 678 = 87.2$ $\Rightarrow$ el desbordamiento ocurrirá en la iteración número 88

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

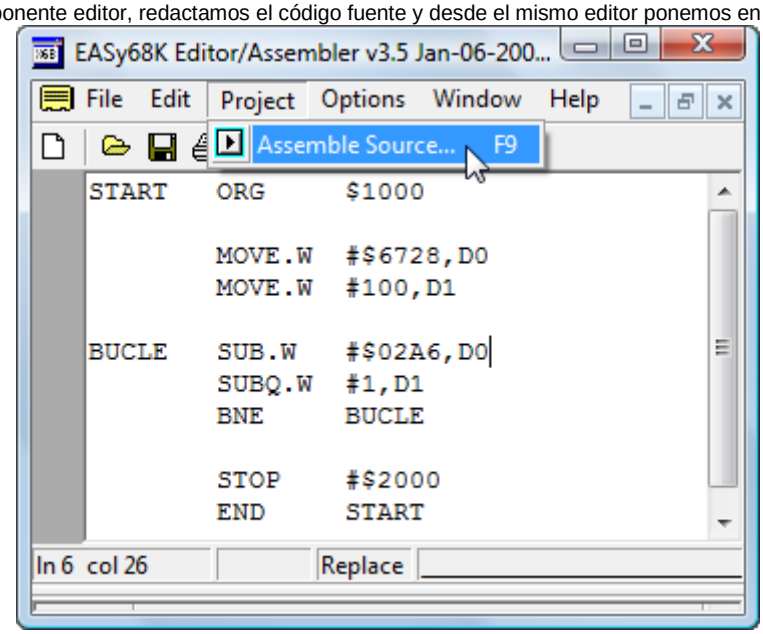


A continuación seguimos restando, pero claro, de este valor que “nos aparece” en los 16 bits menos significativos de D0. Nos faltan (100-88)=12 iteraciones. No vuelve a ocurrir desbordamiento, pues
$32280-12 \cdot 678 = 24144 \geq -32768$.

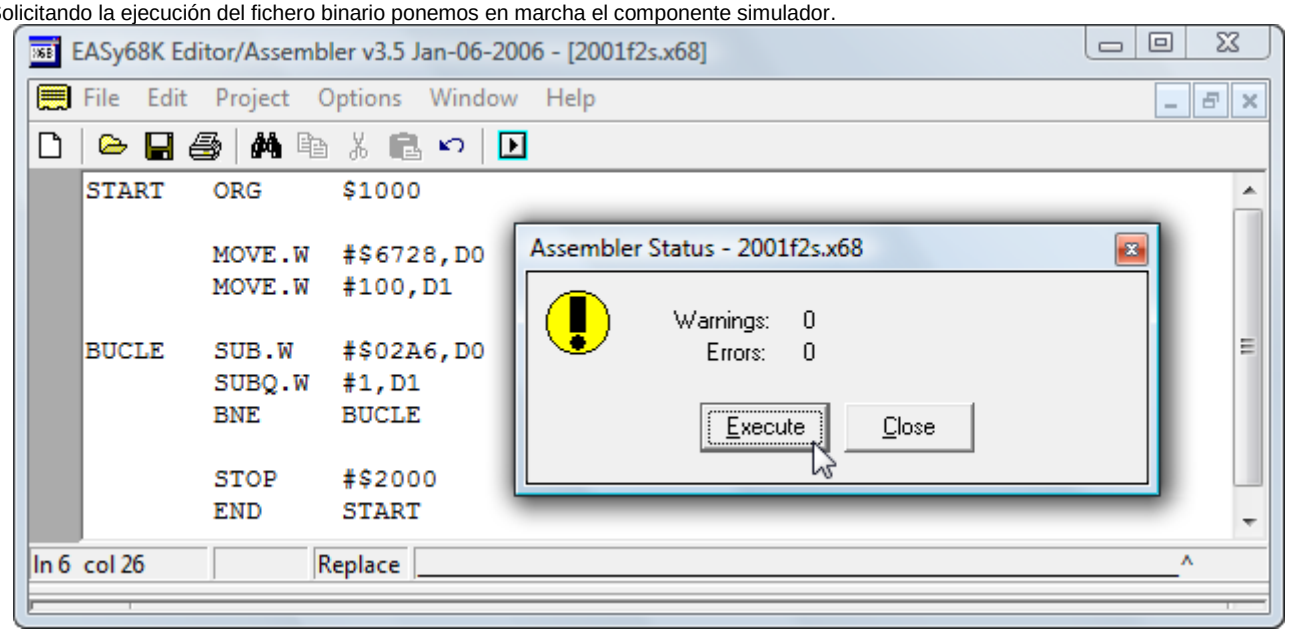
Finalmente, el contenido de D0 es:
&24144 = $5E50

Vamos a comprobar la resolución mediante el simulador EASy68K

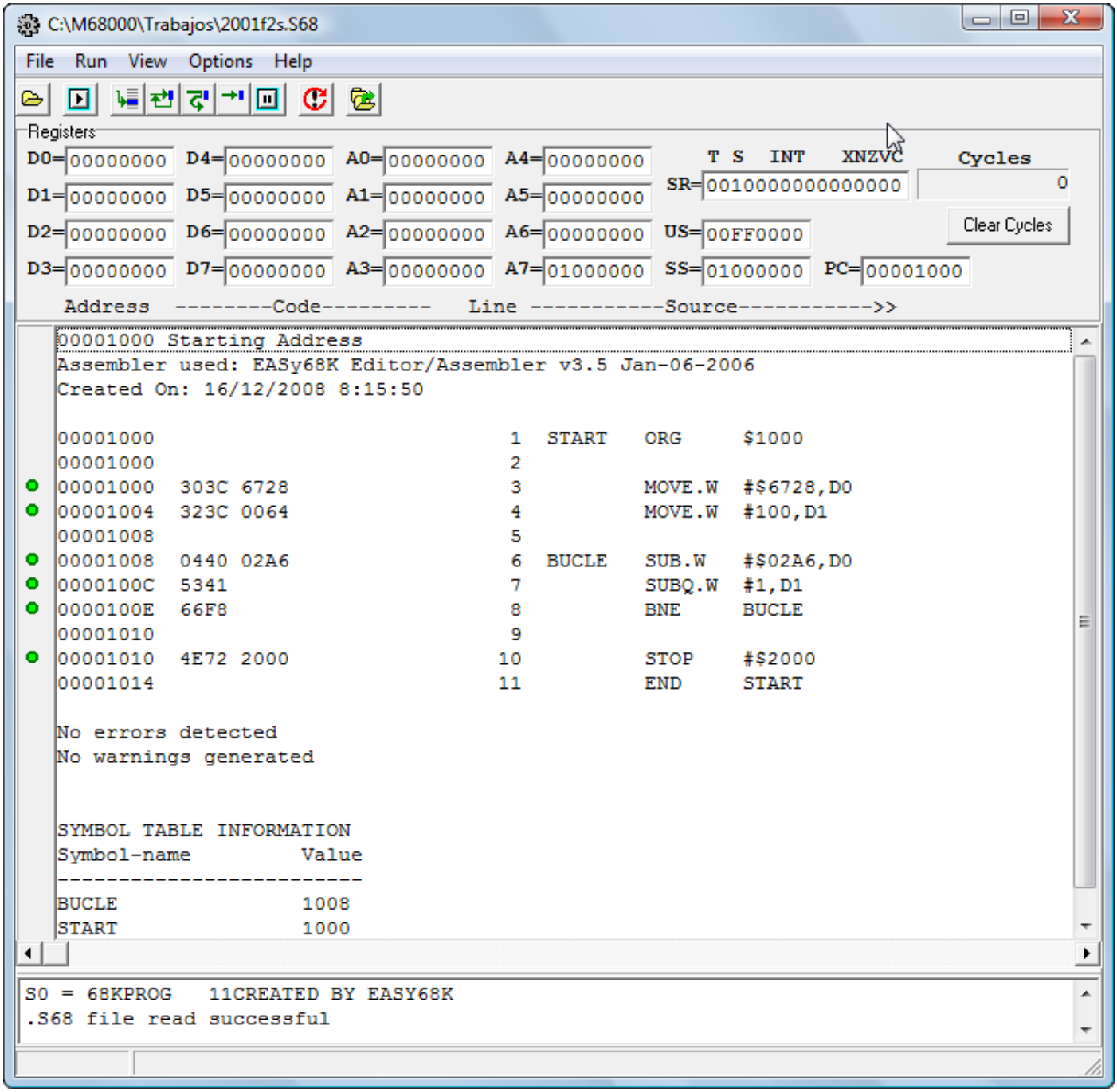
1º Abrimos la ventana del componente editor, redactamos el código fuente y desde el mismo editor ponemos en marcha la compilación.



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.

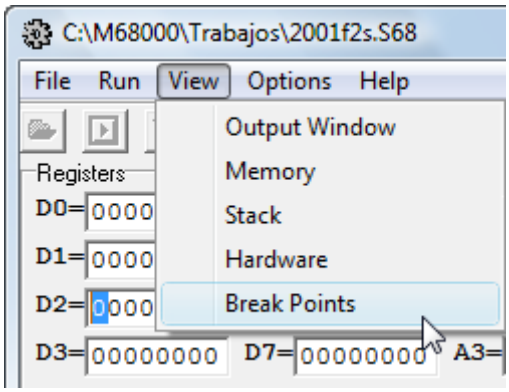


Este fichero listado lo necesitaremos para conocer en qué dirección está la instrucción donde colocaremos un punto de ruptura:

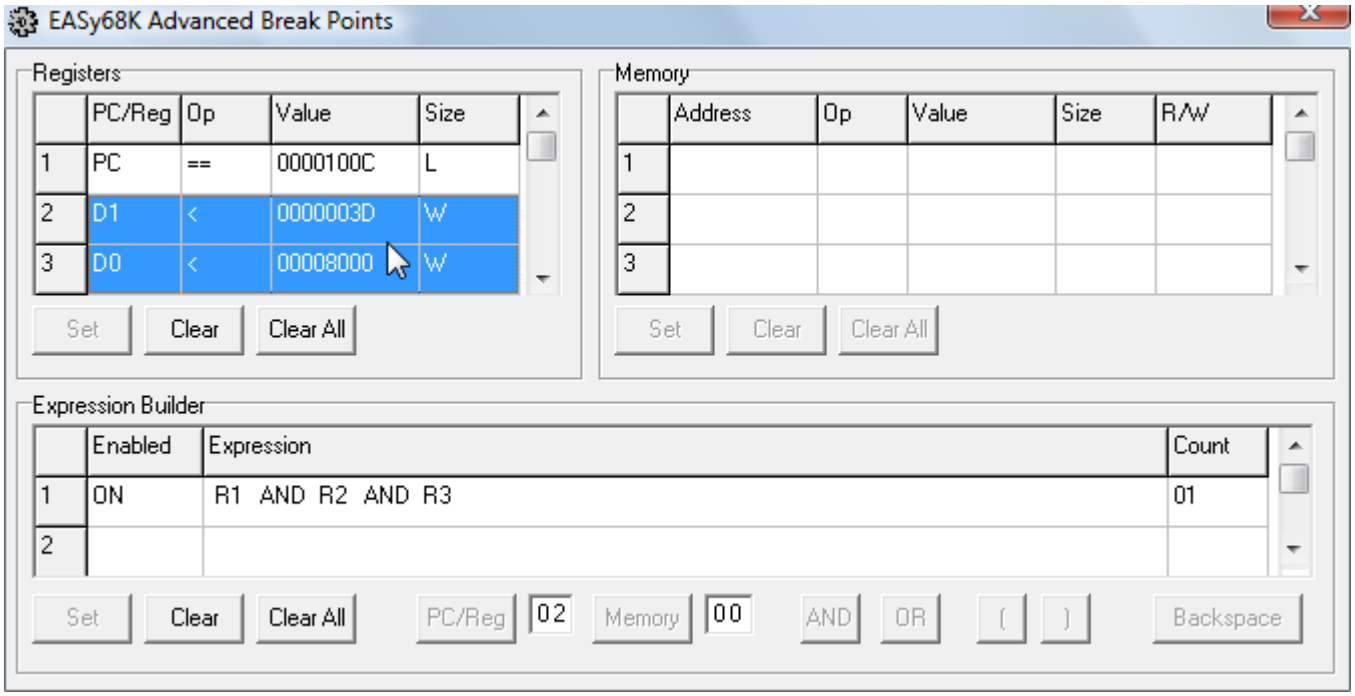
```
SUBQ.W #1,D1
```

Dirección: $100C

4º Vamos a establecer el punto de ruptura



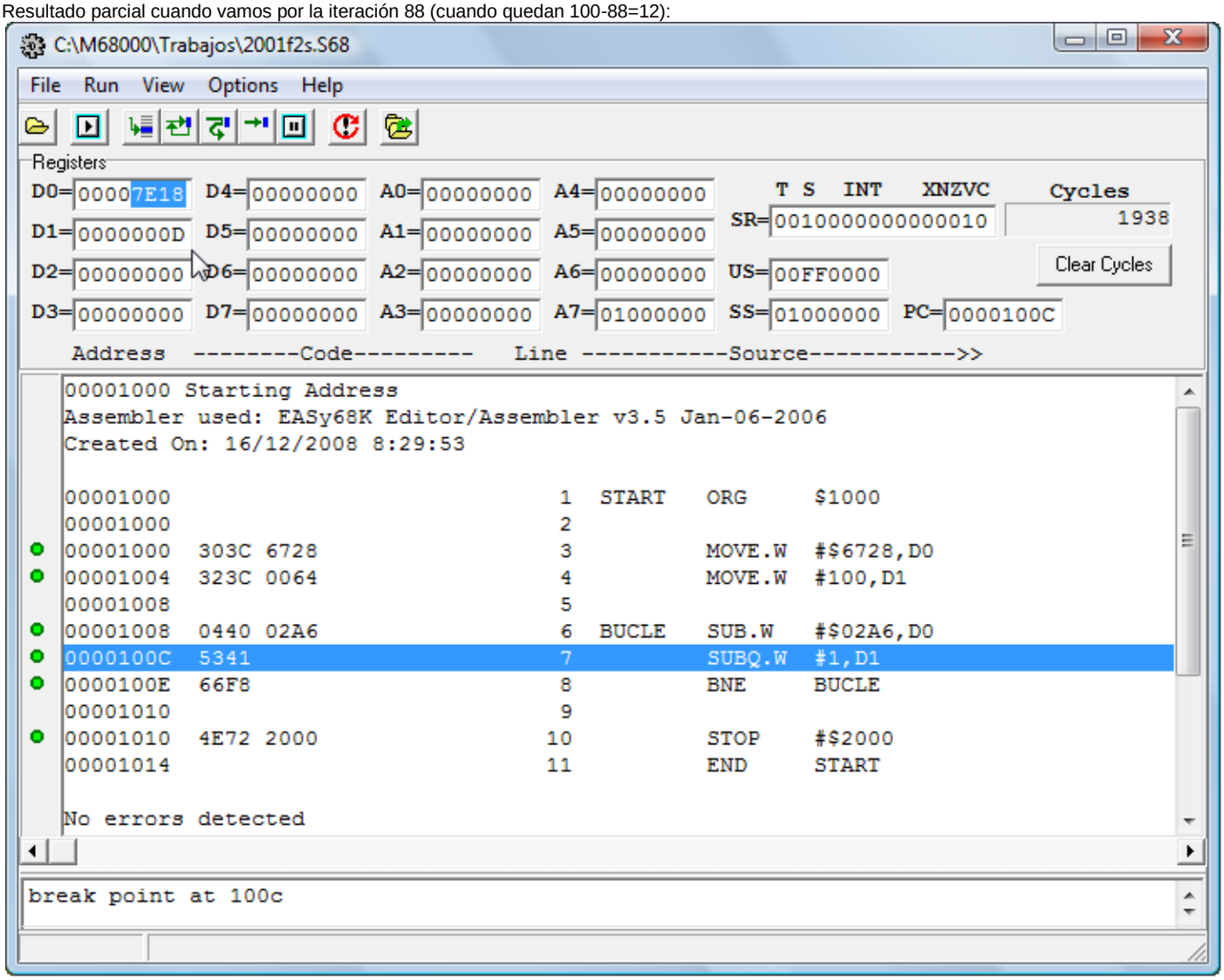
Atención a las condiciones de ruptura:



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 para 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.

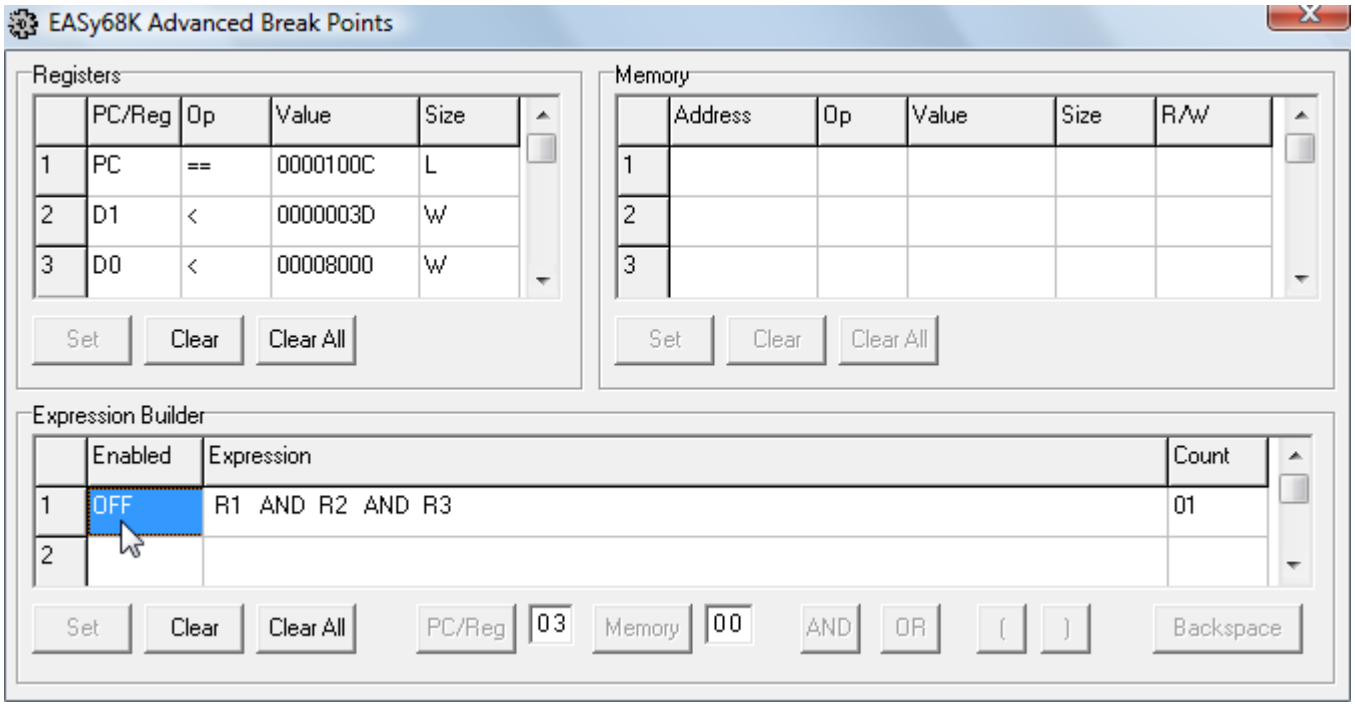
5º Ponemos en marcha la simulación hasta el punto de ruptura.

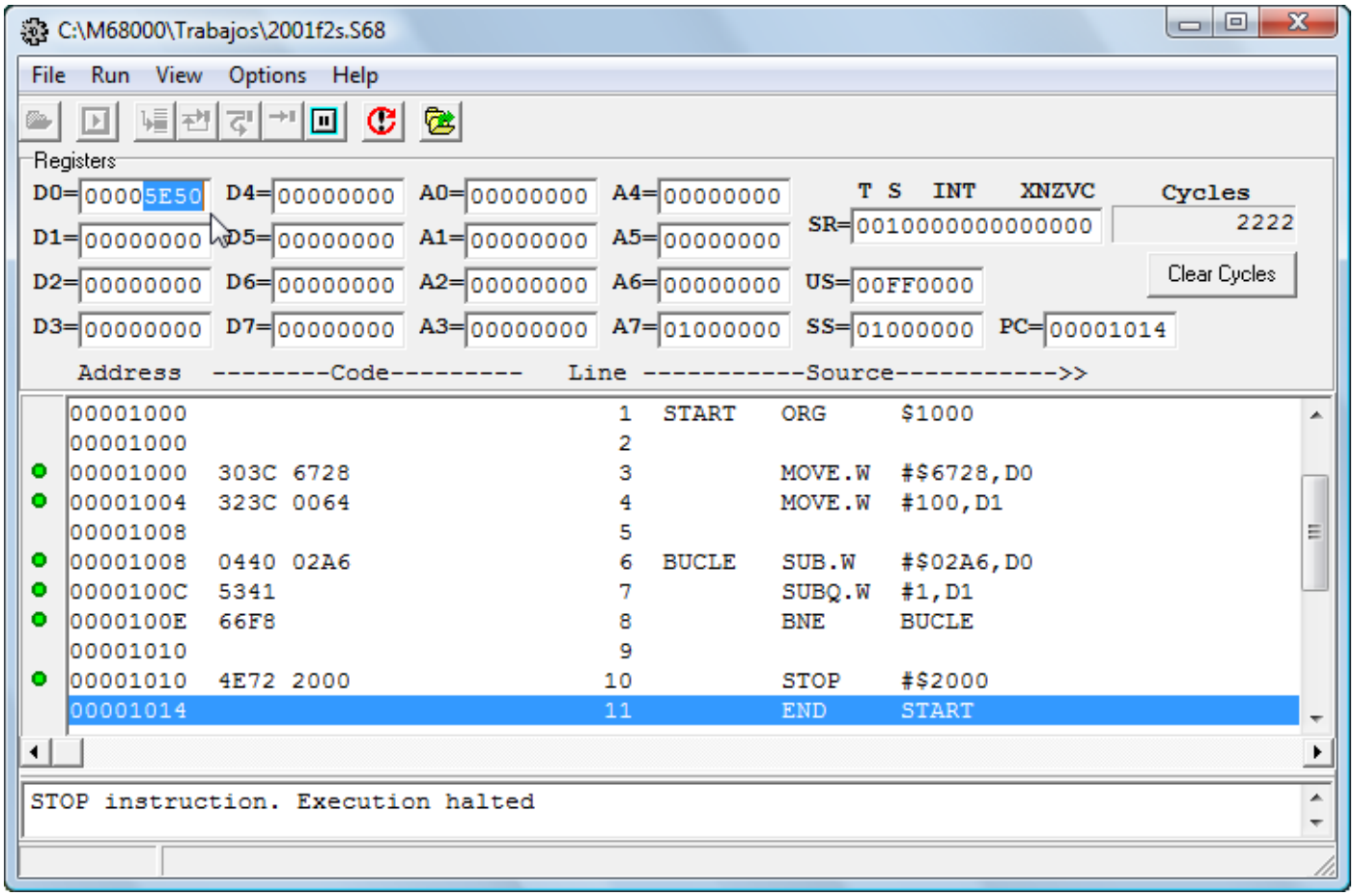
Resultado parcial cuando vamos por la iteración 88 (cuando quedan 100-88=12):



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.





Observamos que el resultado es el previsto (D0)=$5E50.