TRAVAUX DIRIGES ASSEMBLEUR

Objectifs & Consignes

Le but de ces travaux dirigés est de manipuler le simulateur de l'assembleur de Motorola, le modèle 68000.

Après une première phase de rappels des principes du langage assembleur, ces travaux dirigés présentent l'environnement de programmation 68000 de Motorola.

Concernant l'application Easy68K, il faudra télécharger le logiciel, l'installer pour réaliser les exercices demandés par sur cet environnement.

Pour la notation, il sera demandé de mettre des captures d'écran de vos simulations. La note maximale de 20/20 ne sera pas attribuée car vous effectuerez certainement ce travail avec l'aide d'intelligence extérieure ou en groupe. Cependant, certaines questions issues de ce travail vous seront posées lors de l'évaluation finale qui sera un devoir sur table au format papier. Votre intérêt est donc de travailler avec intelligence 😊 et de connaitre les grandes lignes de ce travail!

Enfin, ce travail est nominatif et devra être rendu sur MOODLE en fin de séance. Tous dépassements de limite compteront comme un zéro sur la moyenne générale.

https://asm-editor.specy.app/projects/IEBGEBA

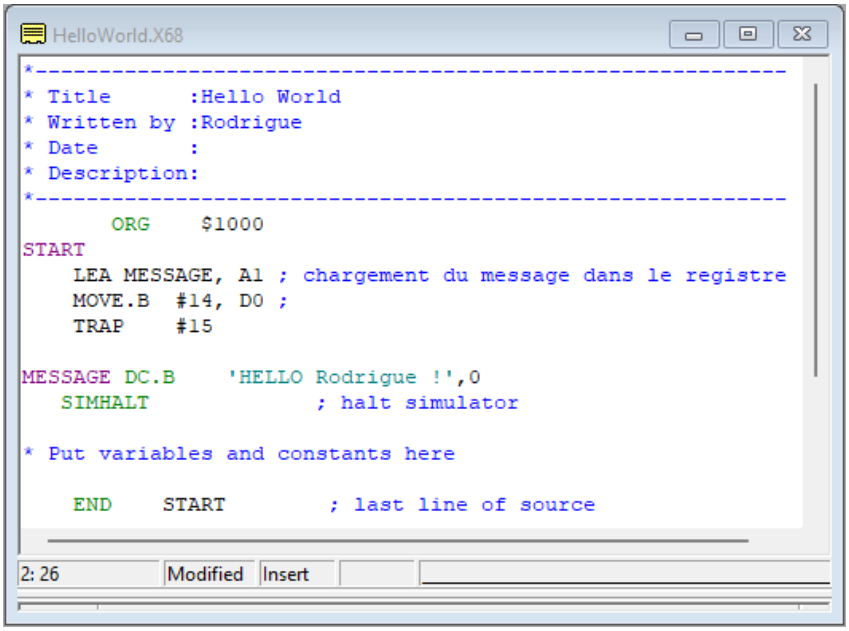
1. Travail demandé

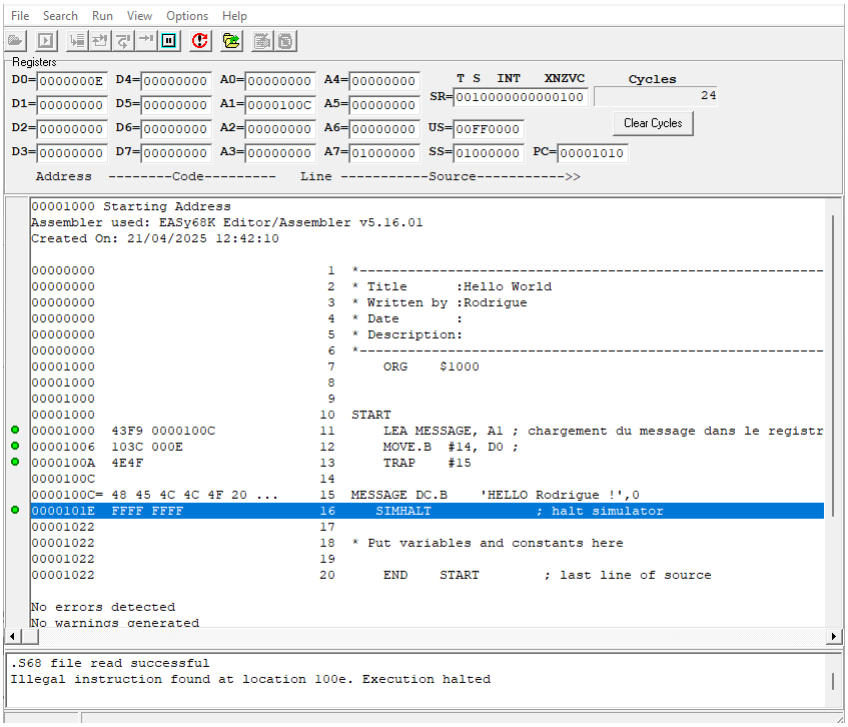
Les points d'attention et les résultats attendus de la correction seront encadrés en vert sur les captures d'écran.

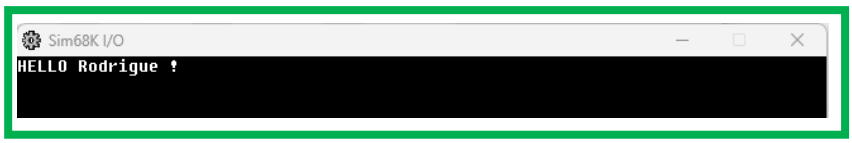
Les codes sont commentés en guise d'explications des captures.

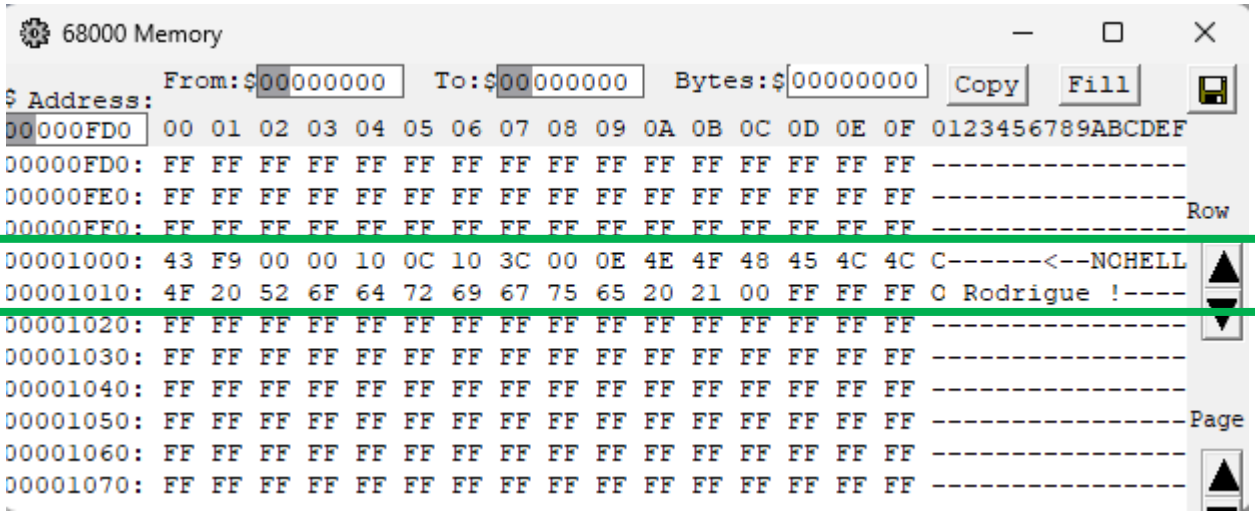
Exécuter le programme en mode ligne par ligne et suivez l'évolution des registres pour répondre aux questions suivantes :

a) Ecrire un code pour afficher " Hello votre_prénom ! " dans le terminal Sim68K



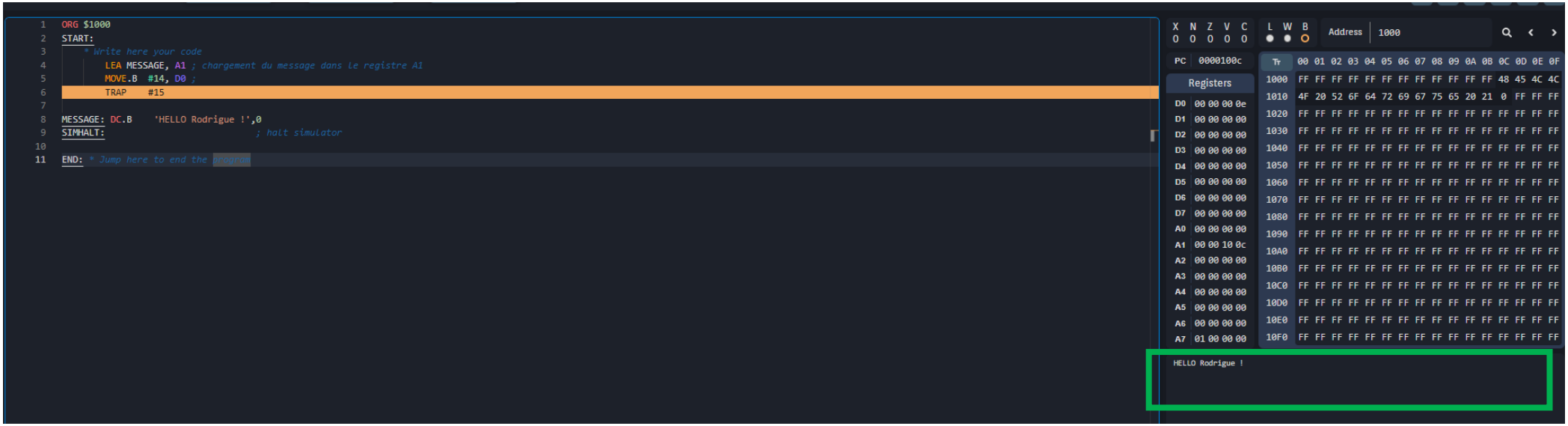

File Search Run View Options Help
Registers
D0=0000000E D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=0000100C A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000100 24
US=00FF0000 SS=01000000 PC=00001010
Clear Cycles
Address --------Code--------- Line ----------Source----------
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 21/04/2025 12:42:10
00000000 1 *---
00000000 2 * Title :Hello World
00000000 3 * Written by :Rodrigue
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *---
00001000 7 ORG $1000
00001000 8
00001000 9
00001000 10 START
00001000 43F9 0000100C 11 LEA MESSAGE, A1 ; chargement du message dans le registr
00001006 103C 000E 12 MOVE.B #14, D0 ;
0000100A 4E4F 13 TRAP #15
0000100C 14
0000100C= 48 45 4C 4C 4F 20 ... 15 MESSAGE DC.B 'HELLO Rodrigue !',0
0000101E FFFF FFFF 16 SIMHALT ; halt simulator
00001022 17
00001022 18 * Put variables and constants here
00001022 19
00001022 20 END START ; last line of source
No errors detected
No warnings generated
.S68 file read successful
Illegal instruction found at location 100e. Execution halted


Sim68K I/O
HELLO Rodrigue !


68000 Memory
From:$00000000 To:$00000000 Bytes:$00000000 Copy Fill
$ Address:
00000FD0 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 0123456789ABCDEF
00000FD0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ----------
00000FE0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ----------
00000FF0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ----------
00001000: 43 F9 00 00 10 0C 10 3C 00 0E 4E 4F 48 45 4C 4C C------<--NOHELL
00001010: 4F 20 52 6F 64 72 69 67 75 65 20 21 00 FF FF FF O Rodrigue !----
00001020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ----------
00001030: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ----------
00001040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ----------
00001050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ----------
00001060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ----------
00001070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ----------
Row
Page

TD/Cours Architecture des ordinateurs	BACHELOR EN CYBERSECURITE	PRENOM : NOM :

Avec l'assembleur en ligne https://asm-editor.specy.app/projects/IEBGEBA




Untitled
Call stack
History
Stack poin...
1 ORG $1000
2 START:
3 * Write here your code
4 LEA MESSAGE, A1 ; chargement du message dans le registre A1
5 MOVE.B #14, D0 ;
6 TRAP #15
7
8 MESSAGE: DC.B 'HELLO Rodrigue !',0
9 SIMHALT: ; halt simulator
10
11 END: * Jump here to end the program
X N Z V C
0 0 0 0 0
L W B
Address 1000
PC 0000100c
Registers
D0 00 00 00 0e
D1 00 00 00 00
D2 00 00 00 00
D3 00 00 00 00
D4 00 00 00 00
D5 00 00 00 00
D6 00 00 00 00
D7 00 00 00 00
A0 00 00 00 00
A1 00 00 10 0c
A2 00 00 00 00
A3 00 00 00 00
A4 00 00 00 00
A5 00 00 00 00
A6 00 00 00 00
A7 01 00 00 00
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
1000 H E L L
1010 O R o d r i g u e !
1020
1030
1040
1050
1060
1070
1080
1090
10A0
10B0
10C0
10D0
10E0
10F0
HELLO Rodrigue !

b) Expliquez pourquoi la commande suivante ne peut pas fonctionner :

ADDI.B #$12345, D5

La commande suivante ne peut pas fonctionnée à cause de la limitation du opcode xxxx.B qui doit être comprise entre $00 à $FF (0 à 255)

Nombres entiers Non Signés (NS) :		
bits		
8	$00 à $FF (0 à 255)	.B Byte
16	$0000 à $FFFF (0 à 65535)	.W Word
32	$00000000 à $FFFFFFFF (0 à 4294967295)	.L Long

c) Faites une exécution en mode ligne par ligne et dresser un tableau contenant les registres qui changent à chaque étape.

Par exemple : **MOVE.B #45,D0**

ADD.B #85,D0

TRAP #15

Afficher le contenu de la mémoire ?

L'assembleur représente les données en Hexadécimal.

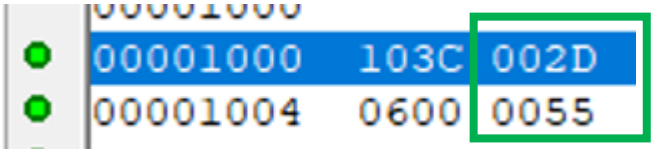
Quelques rappels de conversion.

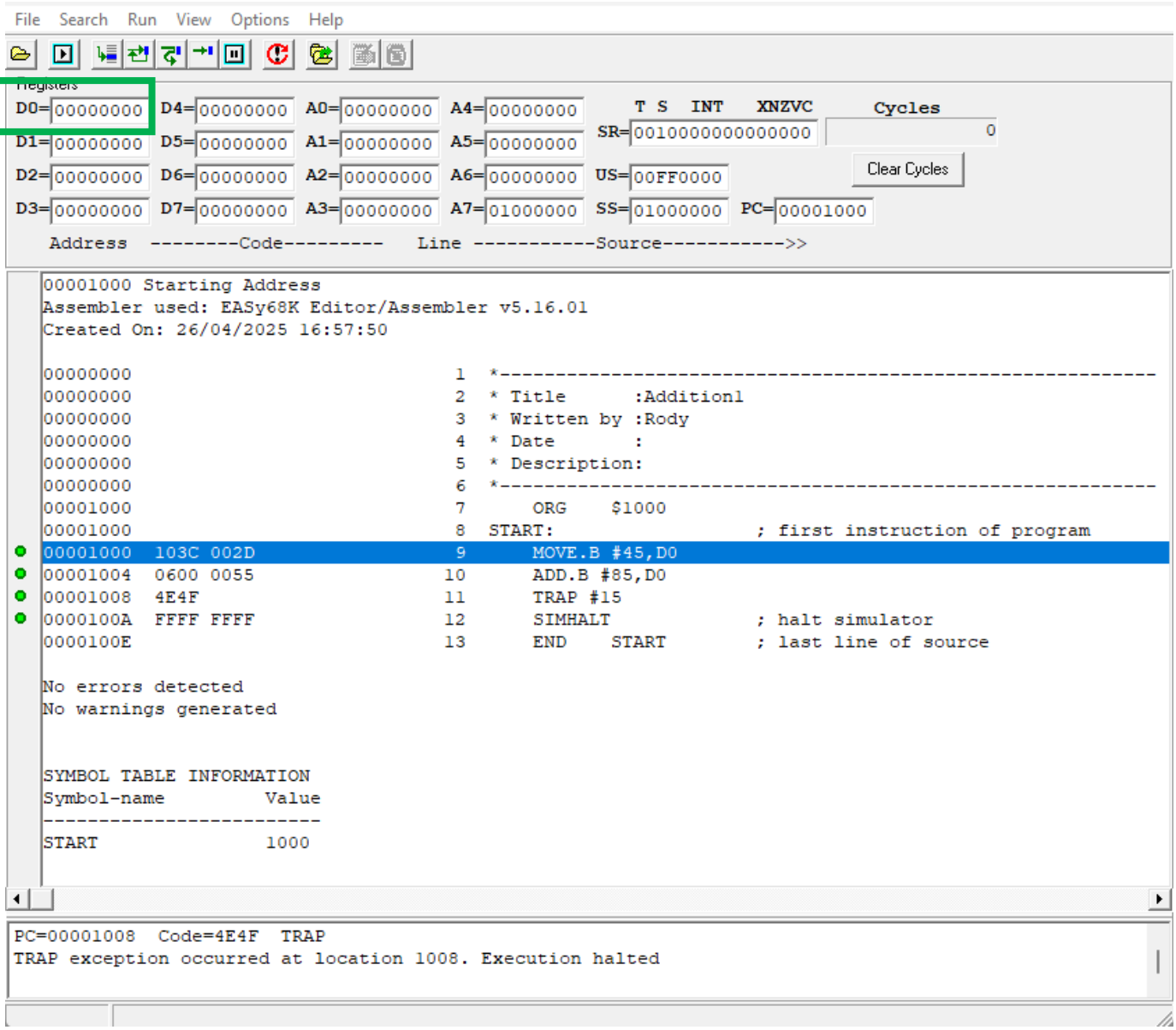
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Décimal
	128	64	32	16	8	4	2	1	
Binaire	0	0	1	0	1	1	0	1	45
	0	1	0	1	0	1	0	1	85
Hexa.	2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0	
	2				D				45
	5				5				85

Représentation des valeurs dans le code :

D'où $45_{10} = 2D_{16}$

$85_{10} = 55_{16}$

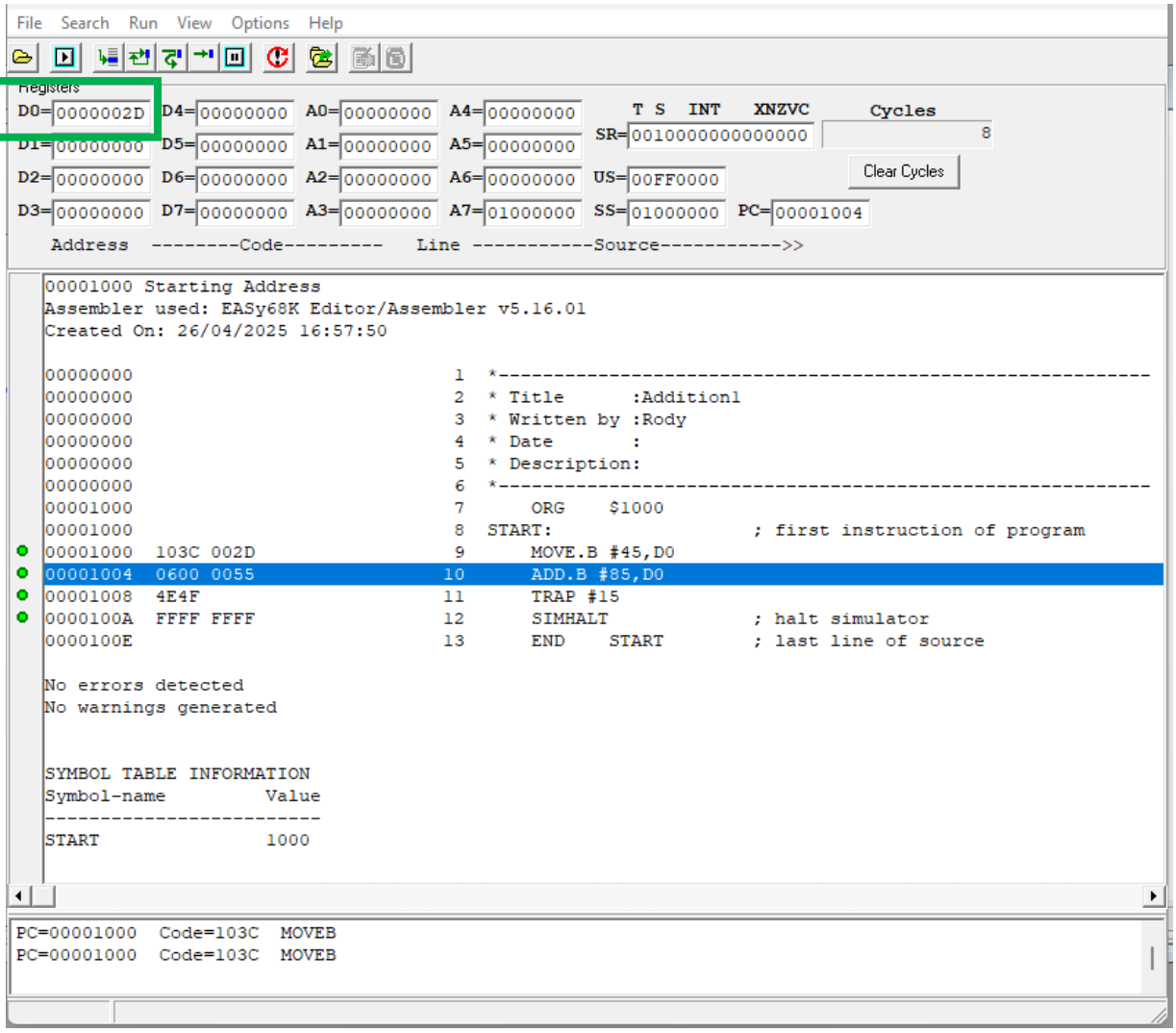




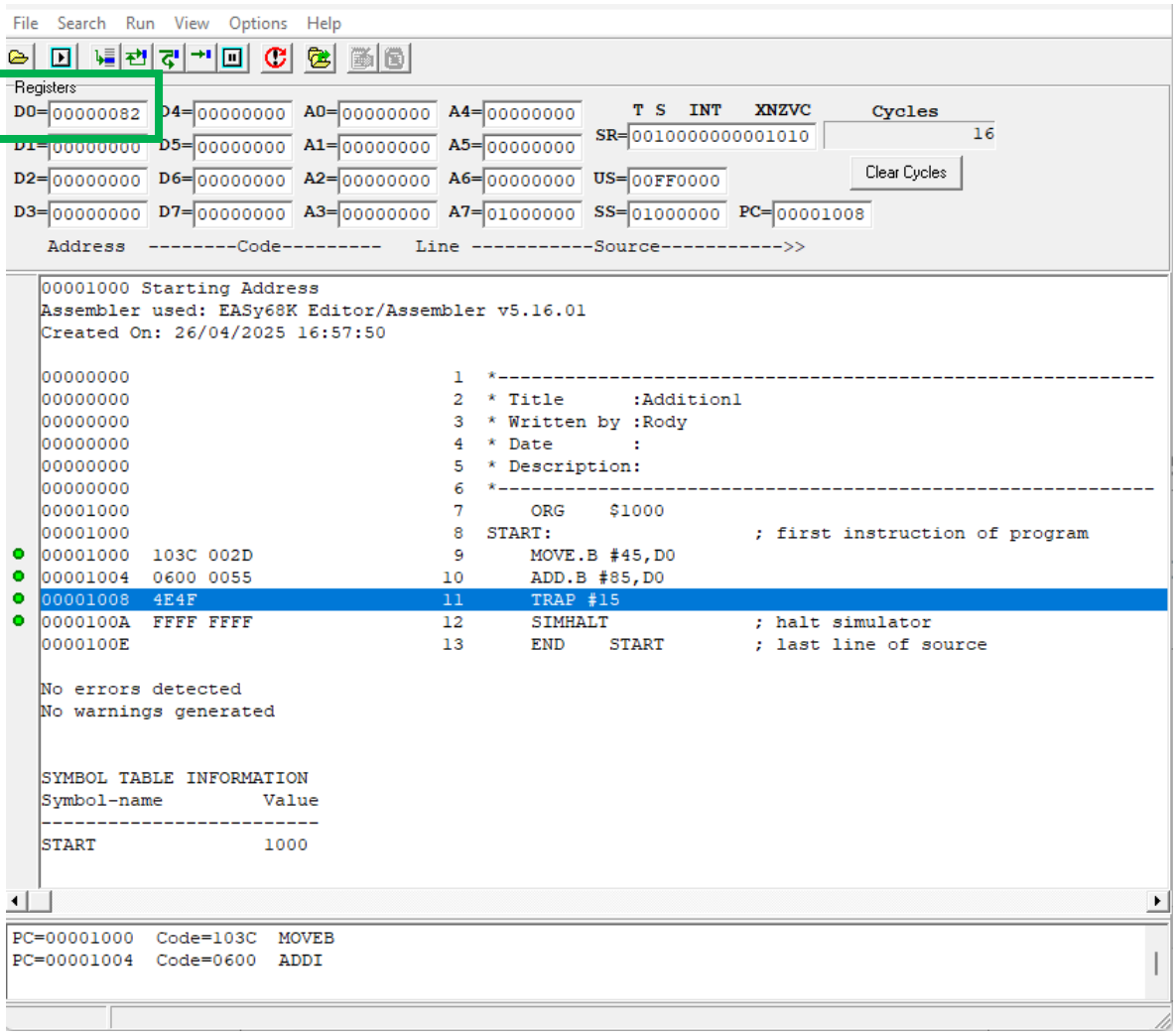
Le programme est lancé après compilation grâce à la commande suivante :

On peut remarquer dans le code les commandes de l'assembleur et les conversions des variables en hexadécimal.

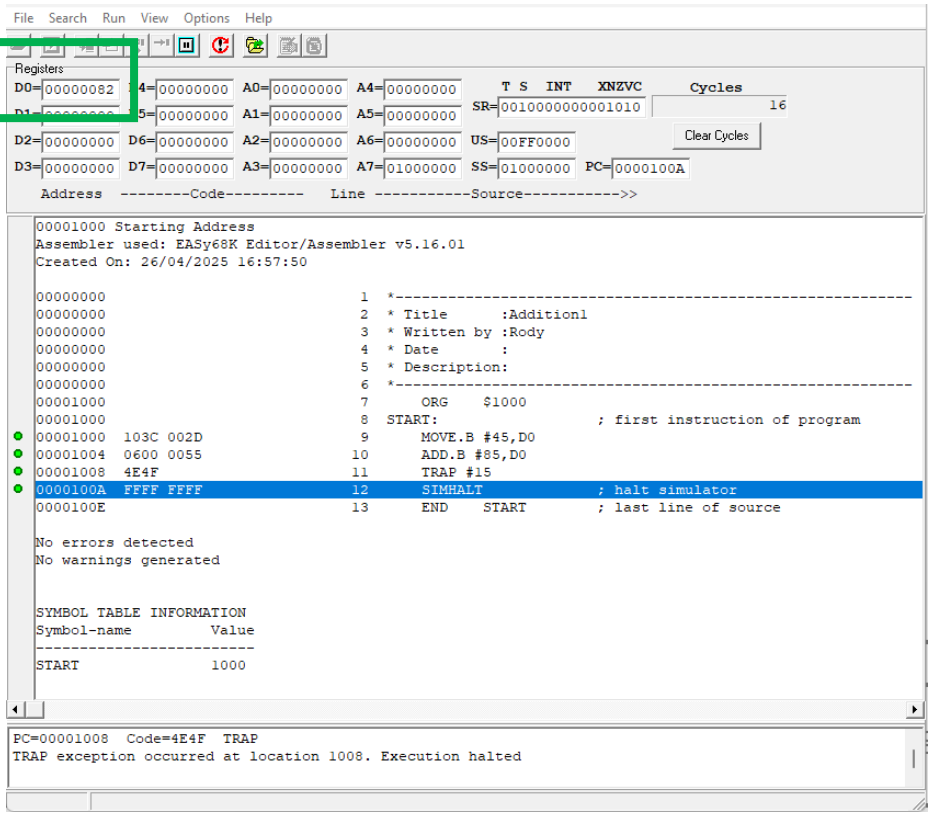
```
00001000  103C 002D              9       MOVE.B #45,D0
00001004  0600 0055             10       ADD.B #85,D0
00001008  4E4F                  11       TRAP #15
0000100A  FFFF FFFF             12       SIMHALT             ; halt simulator
0000100E                        13
0000100E                        14       END    START        ; last line of source
```



Après la première commande de l'adresse 1000, la valeur $45_{10} = 2D_{16}$ est chargée en mémoire dans le registre D0.

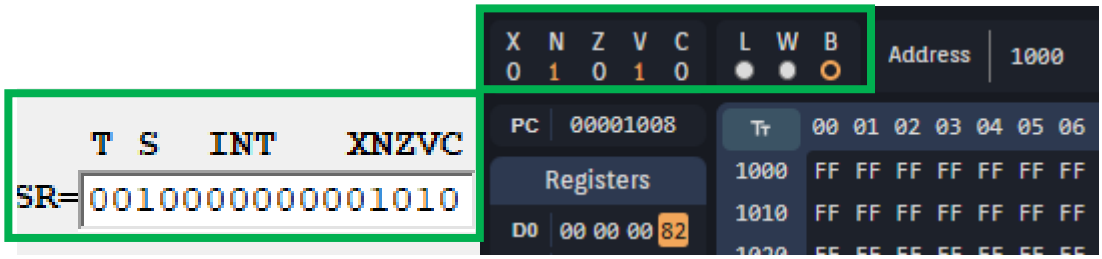


Après l'opération de l'adresse 1004, le résultat de l'addition de 45 + 85 est enregistrée en hexa dans le registre mémoire D0.



TRAP indique une interruption de l'exécution du programme à l'adresse 1008.

A noté que la commande d'arrêt de la simulation : « SIMHALT » envoie des 1 : FFFF FFFF en mémoire.



Dans le registre d'état SR, S=1 : Le MPU(Micro-Processor Unit) est dans le mode superviseur, il est dans le mode de niveau hiérarchique le plus élevé, il peut utiliser tout le jeu d'instructions. Son pointeur de pile est A7 (SSP) dans le simulateur en ligne.

N=1 car la commande SIMHALT envoie FFFF FFFF en mémoire. Cela correspond à la représentation d'un nombre négatif. La commande END START envoie aussi

V= 1, c'est un OVERFLOW Indicateur de dépassement en code complément à 2.

Quatre cas peuvent positionner le bit V à 1 dans son fonctionnement classique:

Addition d'un nombre positif à un nombre positif résultat négatif

Addition d'un nombre négatif à un nombre négatif résultat positif

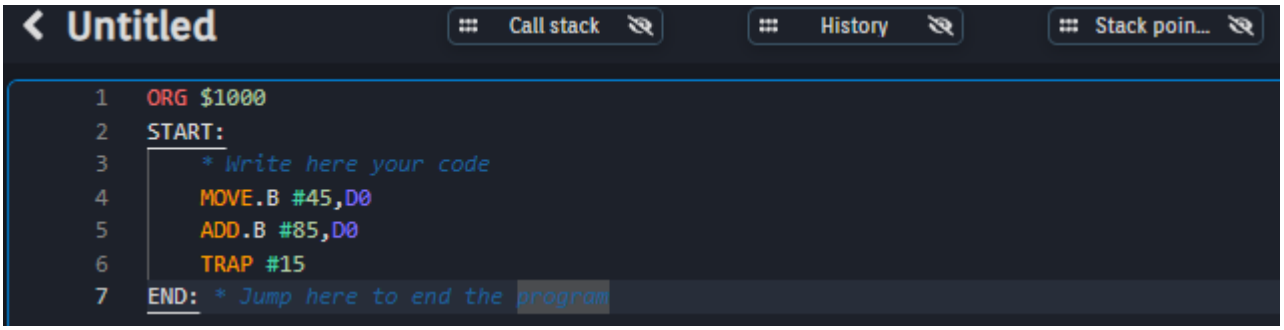
Soustraction d'un nombre négatif à un nombre positif résultat négatif

Soustraction d'un nombre positif à un nombre négatif résultat positif

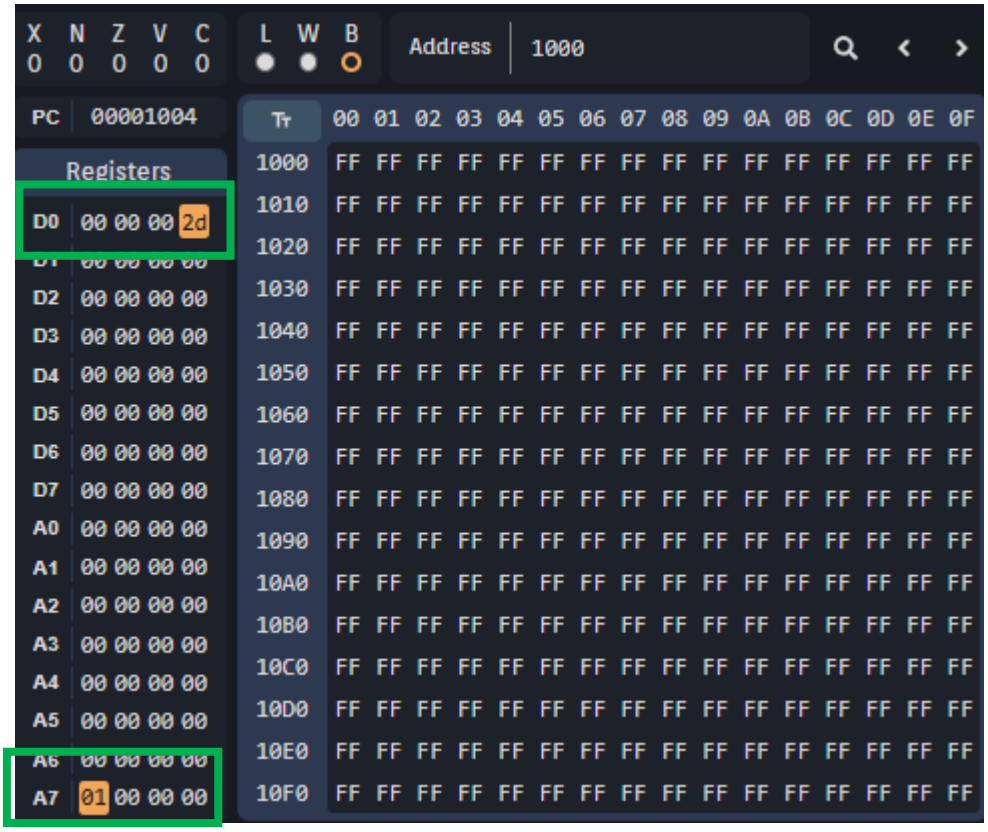
D'autres instructions telles que la division peuvent avoir un effet significatif sur le bit V. Dans le cas de ce TD, c'est la commande SIMHALT.

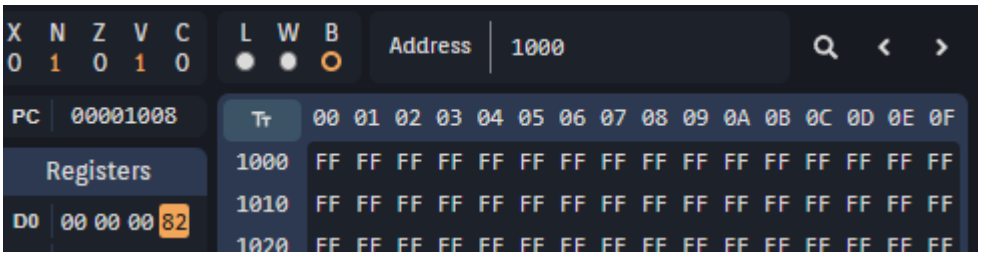
Les opérations dans EASY68K sont identiques avec le simulateur en ligne

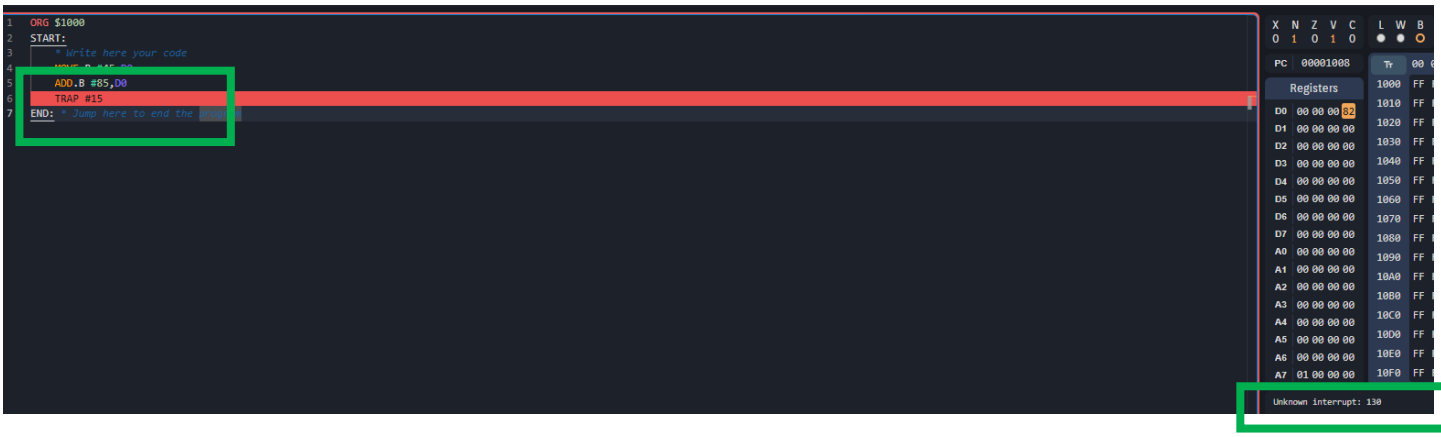
https://asm-editor.specy.app/projects/IEBGEBA



En mode pas à pas, il faut utiliser le bouton STEP.


X N Z V C
0 0 0 0 0
L W B
Address
1000
PC
00001004
Registers
D0 00 00 00 2d
D1 00 00 00 00
D2 00 00 00 00
D3 00 00 00 00
D4 00 00 00 00
D5 00 00 00 00
D6 00 00 00 00
D7 00 00 00 00
A0 00 00 00 00
A1 00 00 00 00
A2 00 00 00 00
A3 00 00 00 00
A4 00 00 00 00
A5 00 00 00 00
A6 00 00 00 00
A7 01 00 00 00
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
1000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF


X N Z V C
0 1 0 1 0
L W B
Address
1000
PC
00001008
Registers
D0 00 00 00 82
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
1000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

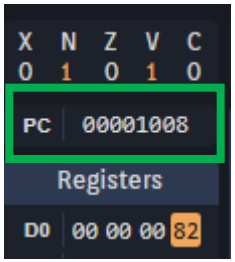

ORG $1000
START:
* Write here your code
ADD.B #85,D0
TRAP #15
END: * Jump here to end the prog
Unknown interrupt: 130


X N Z V C
0 1 0 1 0
L W B
Address 1000
PC 00001008
Registers
D0 00 00 00 82
D1 00 00 00 00
00 01 02 03 04 05 06
1000 FF FF FF FF FF FF FF
1010 FF FF FF FF FF FF FF
1020 FF FF FF FF FF FF FF

Donner les adresses de début et de fin de la partie code et de la partie données de ce programme ?

```
00001000                             8  START:                  ; first instruction of program
00001000  103C 002D                  9      MOVE.B #45,D0
00001004  0600 0055                 10      ADD.B #85,D0
00001008  4E4F                      11      TRAP #15
0000100A  FFFF FFFF                 12      SIMHALT             ; halt simulator
0000100E                            13
0000100E                            14      END    START        ; last line of source
```

Début du programme à l'adresse 1000 et fin du programme à l'adresse 100E

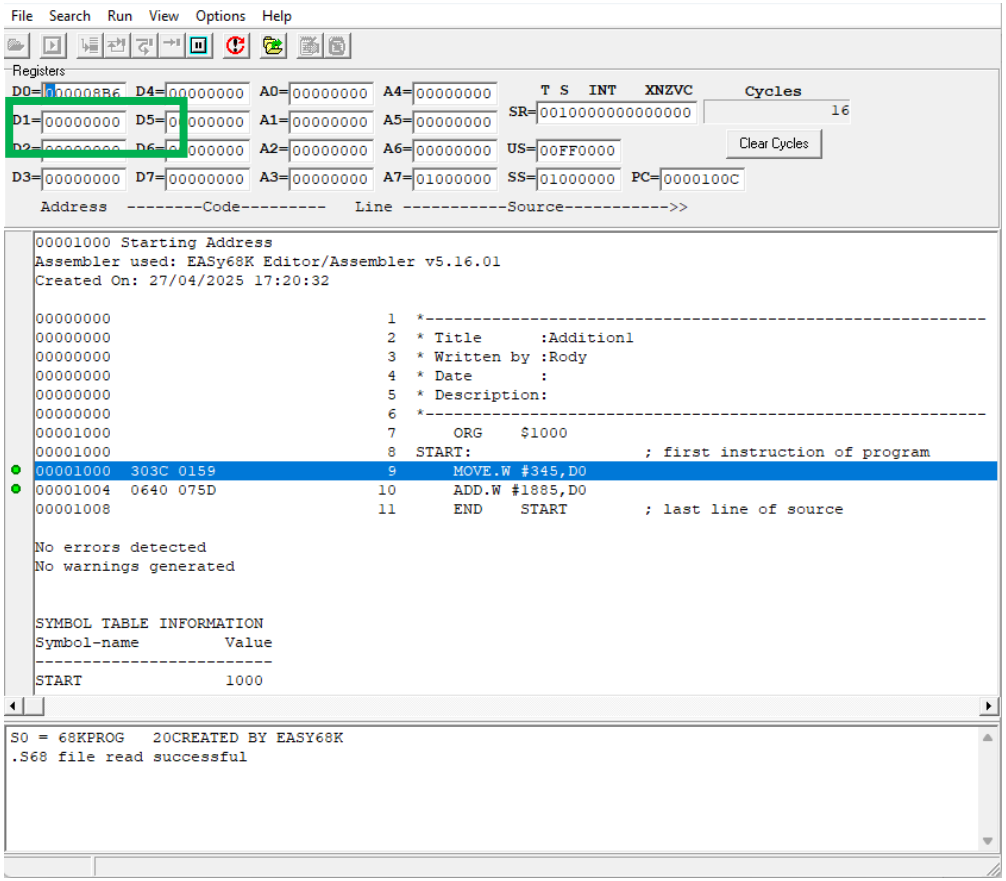


Début du programme à l'adresse 1000 et fin du programme à l'adresse 1008

Comment peut-ton définir la tâche que doit effectuer l'instruction **TRAP #15** ? Et comment elle récupère les paramètres nécessaires ?

L'instruction **TRAP #15** est une exception pour le 68K. C'est un genre d'interruption dans l'exécution du programme dans PC. La commande est reconnue grâce à la notation 4E4F.

d) On souhaite calculer la somme de 345 et 1885. Utiliser le simulateur 68000 pour calculer la somme. Notez le code qui fonctionne !



TD/Cours Architecture des ordinateurs	BACHELOR EN CYBERSECURITE	PRENOM : NOM :

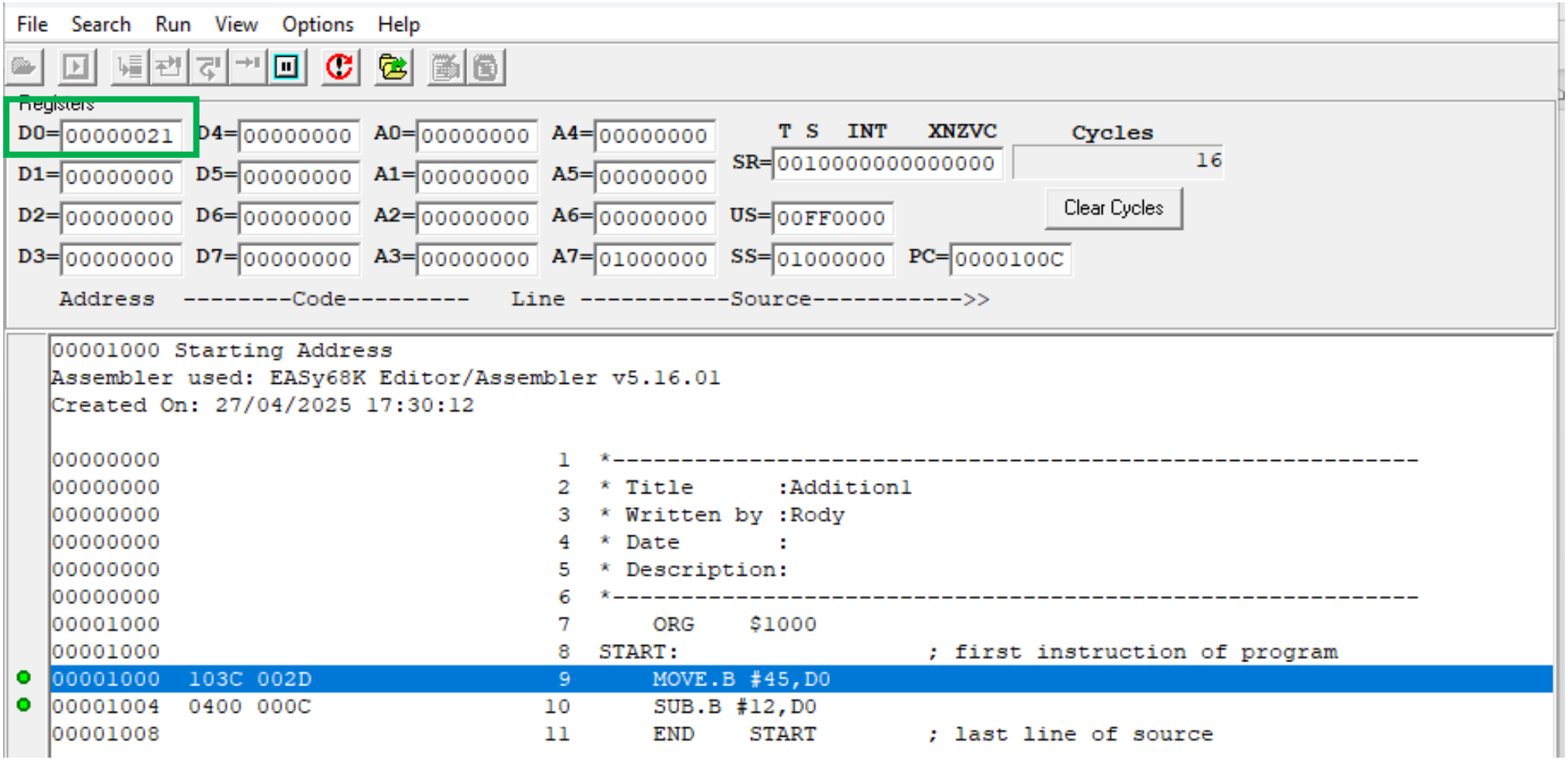
```
1   ORG $1000
2   START:
3       * Write here your code
4       MOVE.W #345, D0
5       ADD.W #1885, D0
6
7   END: * Jump here to end the program
```

L'opération n'est réalisable que sur 16 bits d'où ***.W** à la place de ***.B.**

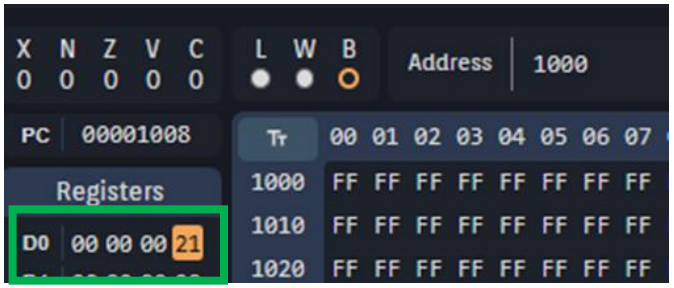
Nous avons : 1885 + 345 = 2230

	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	Décimal
	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	2230
Binaire	0	0	0	1	0	0	0	1	0	1	1	0	1	1	0	
Hexa.	2^{2}	2^{1}	2^{0}	2^{3}	2^{2}	2^{1}	2^{0}	2^{3}	2^{2}	2^{1}	2^{0}	2^{3}	2^{2}	2^{1}	2^{0}	
	0			8				B				6				

e) Vérifier le contenu de registre D0 pour le calcul utilisant la commande SUB, calculer 45 -12.



```
1   ORG $1000
2   START:
3       * Write here your code
4       MOVE.B #45,D0
5       SUB.B #12,D0
6   END: * Jump here to end the program
```

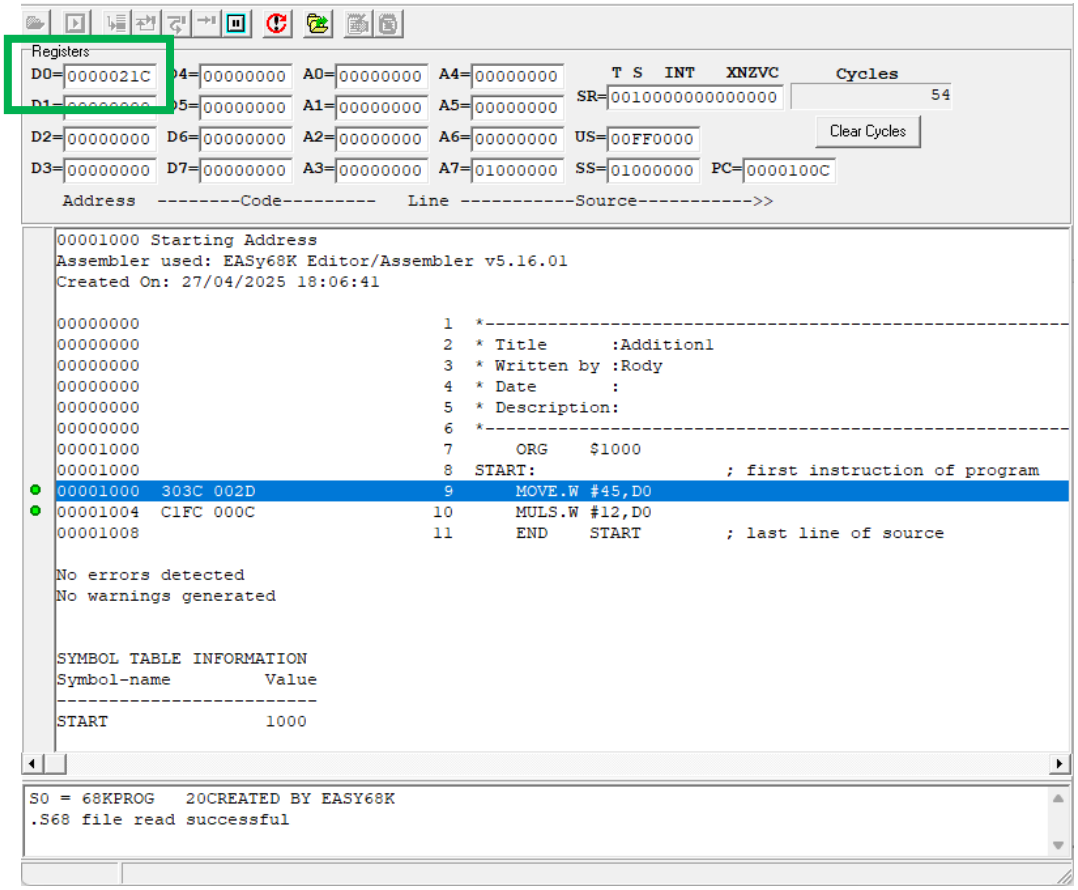


Capture des registres et expliquer

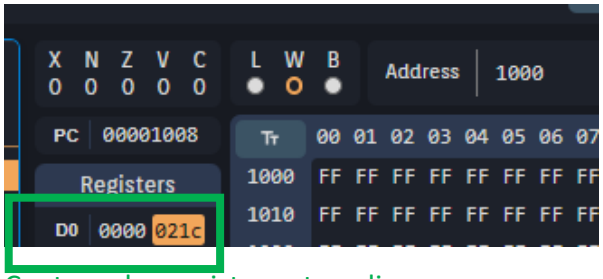
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Décimal
	128	64	32	16	8	4	2	1	
Binaire	0	0	1	0	0	0	0	1	33
Hexa.	2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0	
	2				1				

45-12 = 33 -> $33_{10} = 21_{16}$

f) En utilisant la commande MULS, calculer 45*12.



```
1  ORG $1000
2  START:
3      * Write here your code
4      MOVE.W #45, D0
5      MULS #12, D0
6
7  END: * Jump here to end the program
```

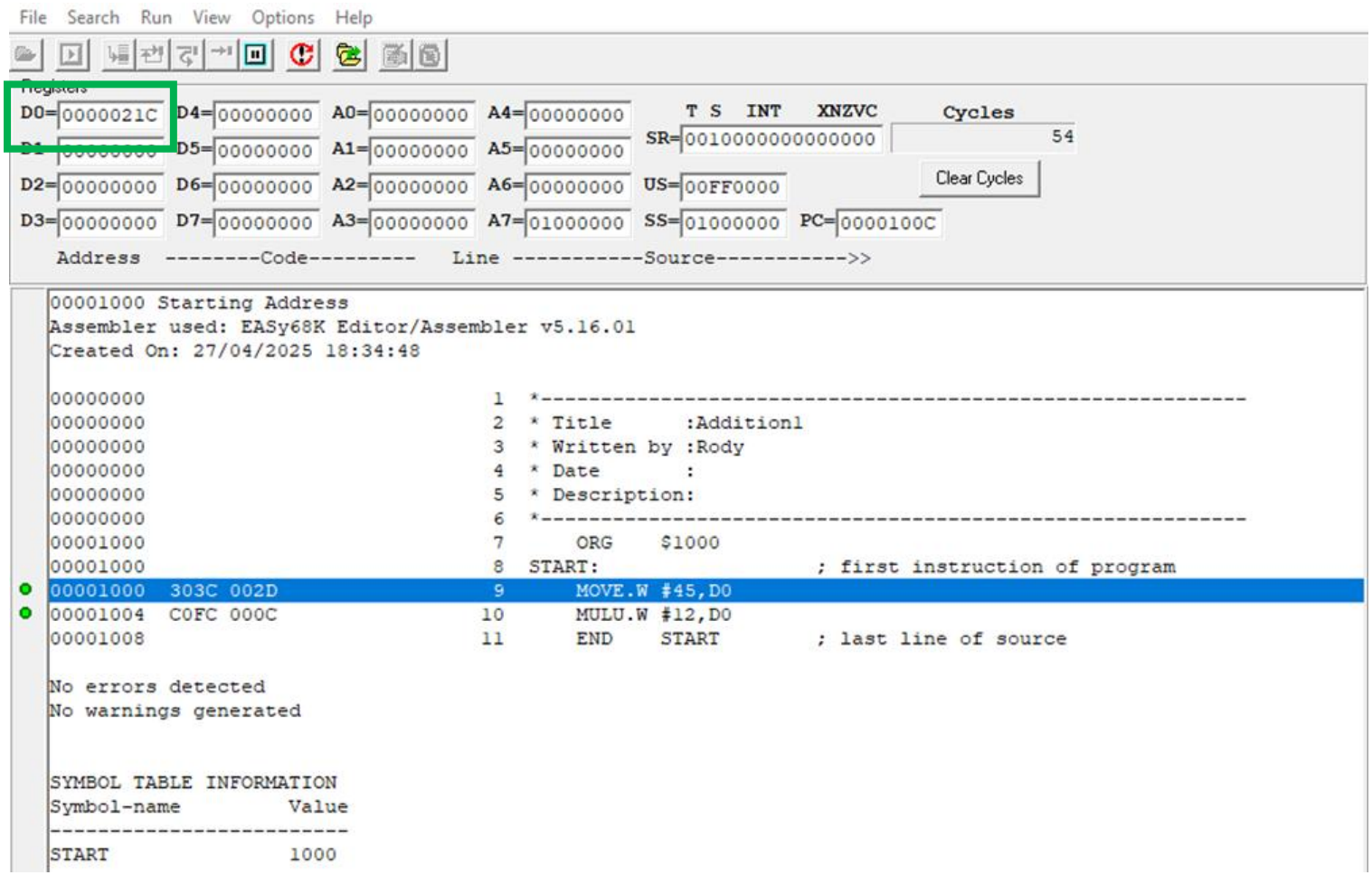


Capture des registres et expliquer

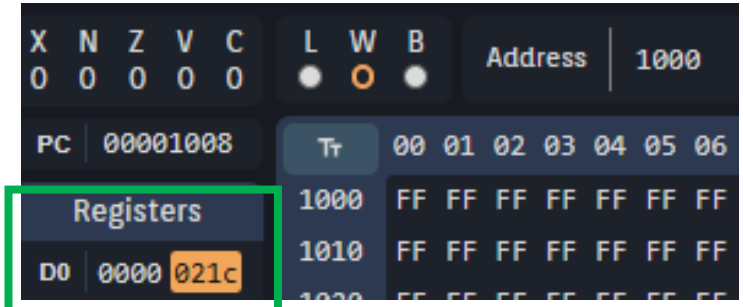
MULS est une instruction de multiplication pour les processeurs de la famille Motorola 68000. Elle permet de multiplier deux opérandes signés et de stocker le résultat dans un registre de destination. Nous avons : 45 *12 = 540 > 255 d'où *.W au lieu de *.B.

	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	Décimal
	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	540
Binaire	0	0	0	0	0	1	0	0	0	0	1	1	1	0	0	
Hexa.	2^{2}	2^{1}	2^{0}	2^{3}	2^{2}	2^{1}	2^{0}	2^{3}	2^{2}	2^{1}	2^{0}	2^{3}	2^{2}	2^{1}	2^{0}	
	0			2				1				C				

g) En utilisant la commande MULU, calculer 45*12.



```
1   ORG $1000
2   START:
3       * Write here your code
4       MOVE.W #45, D0
5       MULU #12, D0
6
7   END: * Jump here to end the program
```

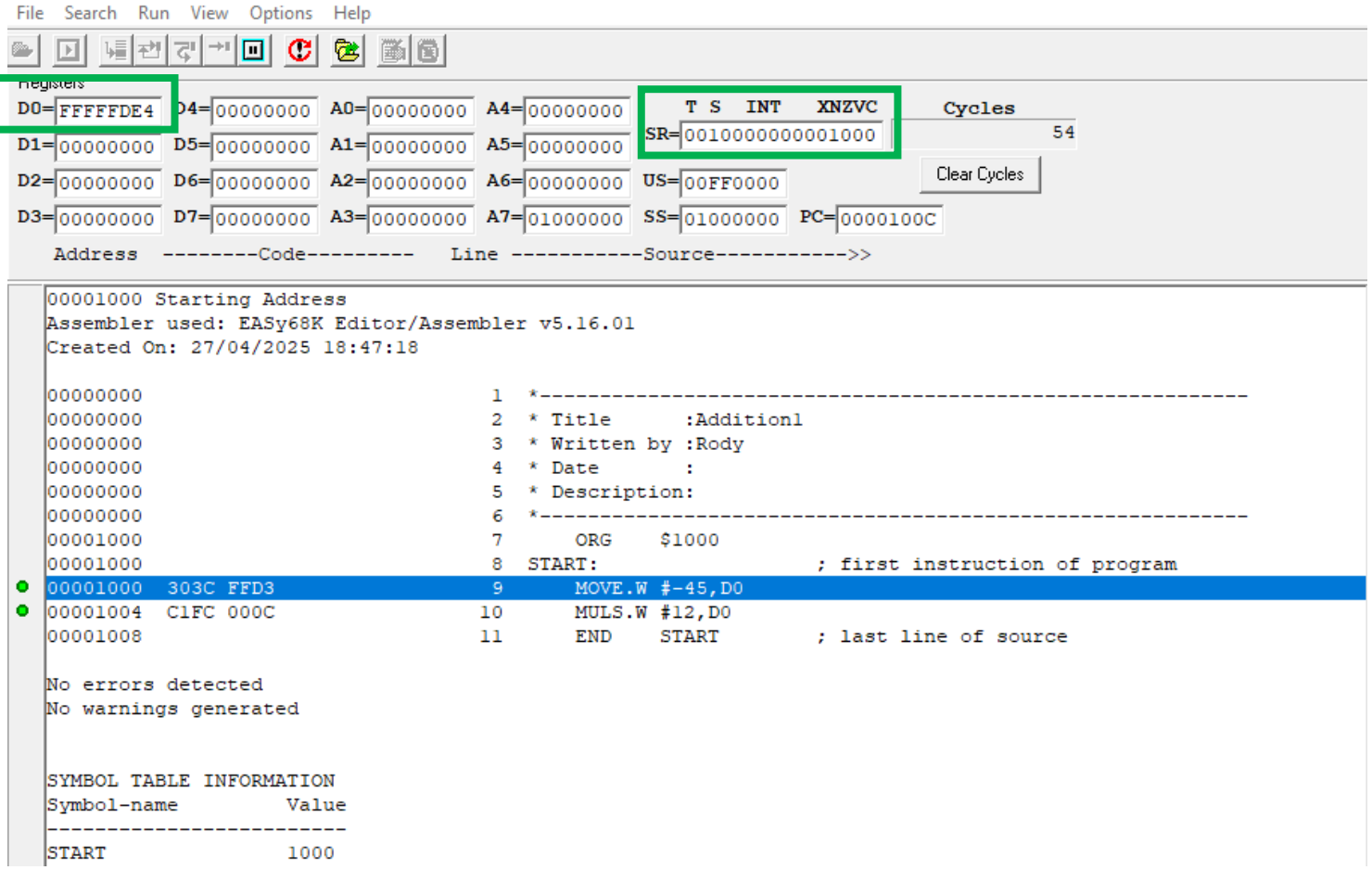


Capture des registres et expliquer

MULU est une instruction d'assemblage utilisée pour multiplier deux opérandes non signés.

Idem MULS

h) Exercice : Utilisant la commande MULS, calculer -45*12.



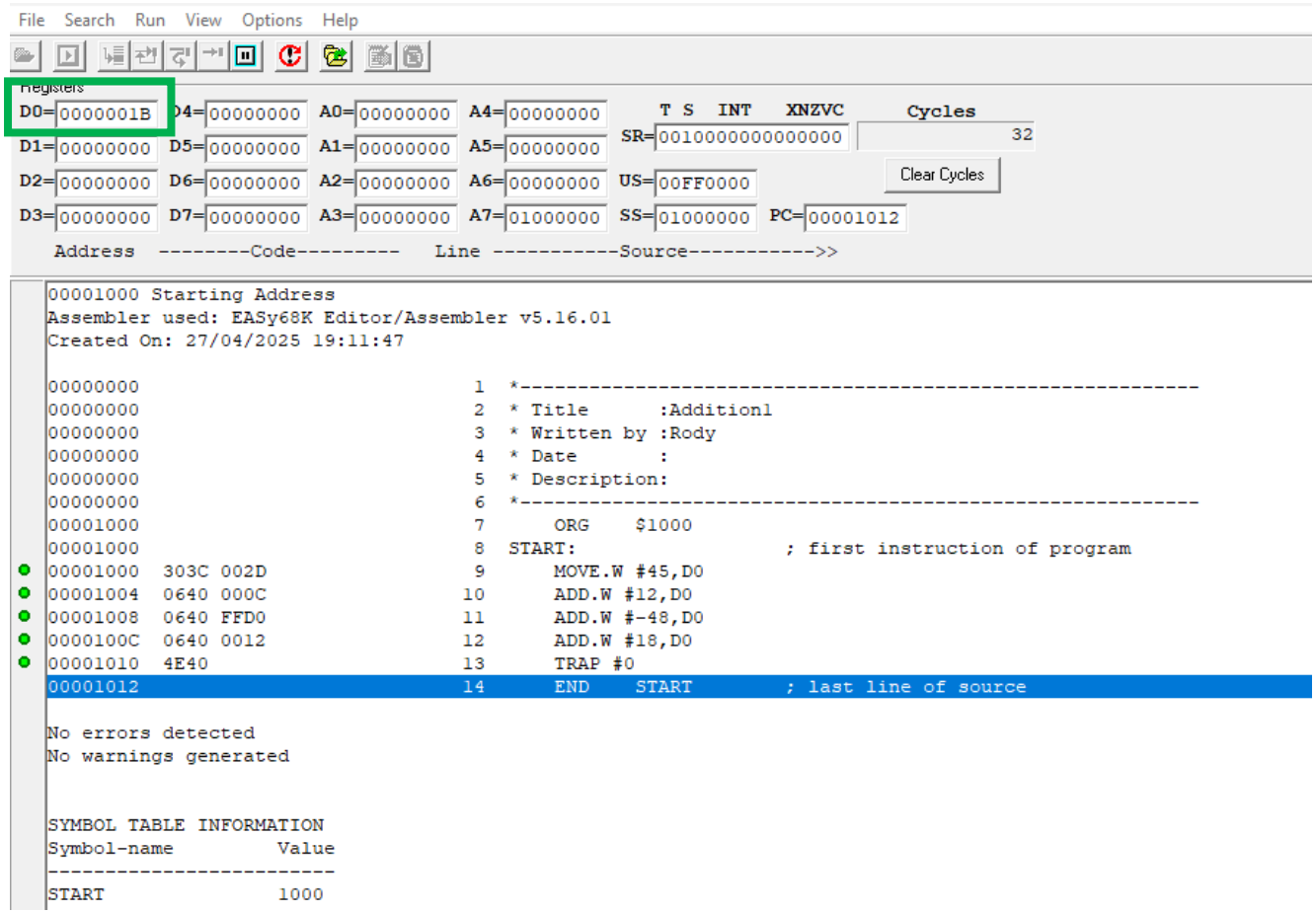
```
1   ORG $1000
2   START:
3       * Write here your code
4       MOVE.W #-45, D0
5       MULS #12, D0
6
7   END: * Jump here to end the program
```

Capture des registres et expliquer

Nous avons : -45 * 12 = -540. **A noter que N= 1 pour indiquer que la valeur est négative.**

	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	Décimal
	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	
Binaire	0	0	0	0	0	0	1	0	0	0	0	1	1	1	0	0	540
CA2	1	1	1	1	1	1	0	1	1	1	1	0	0	1	0	0	-540
	2^{3}	2^{2}	2^{1}	2^{0}	2^{3}	2^{2}	2^{1}	2^{0}	2^{3}	2^{2}	2^{1}	2^{0}	2^{3}	2^{2}	2^{1}	2^{0}	
Hexa.	F				D				E				4				

i) Calculer la somme de 45+12-48+18. Notez le code qui fonctionne !

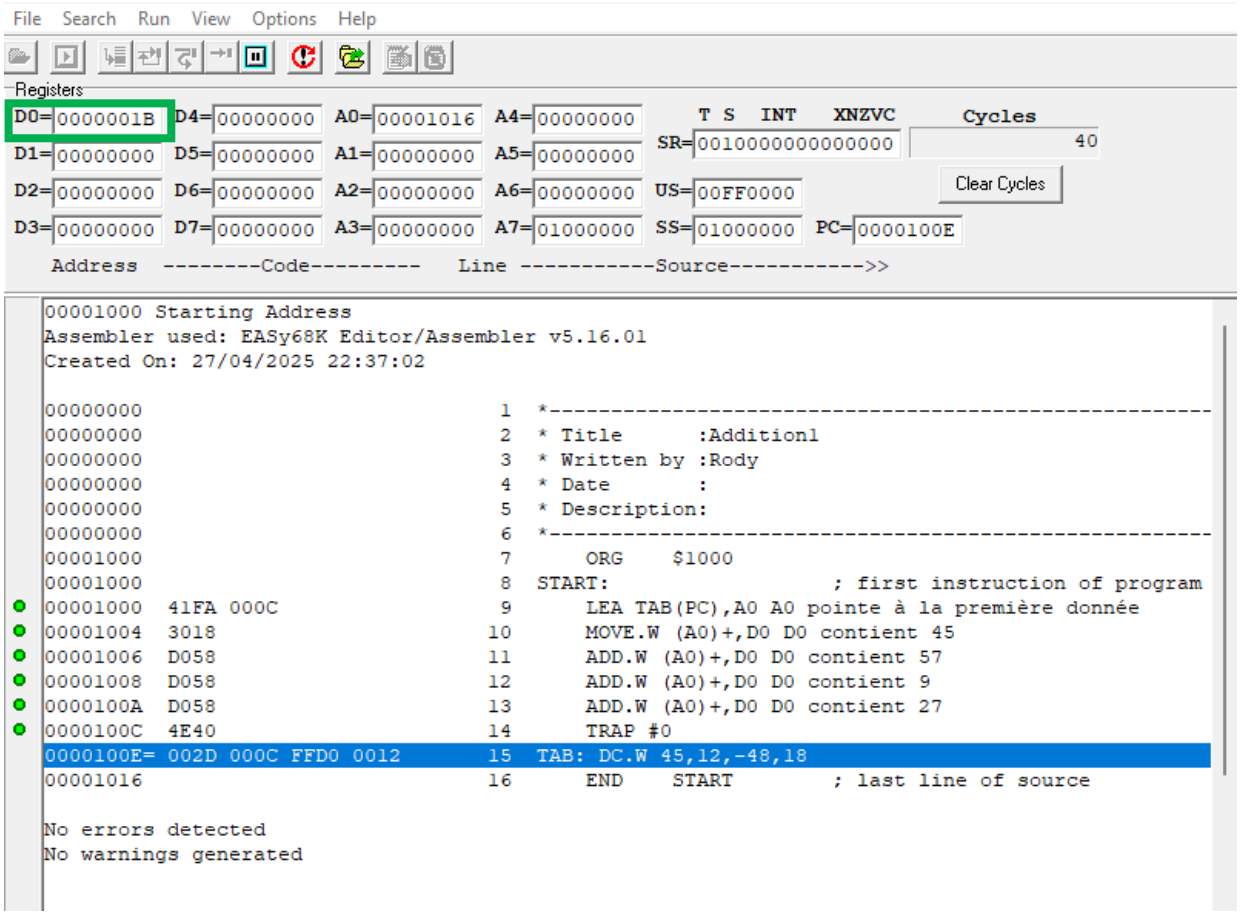


```
1   ORG $1000
2   START:
3       * Write here your code
4       MOVE.W #45,D0
5       ADD.W #12,D0
6       ADD.W #-48,D0
7       ADD.W #18,D0
8       TRAP #15
9
10  END: * Jump here to end the program
```

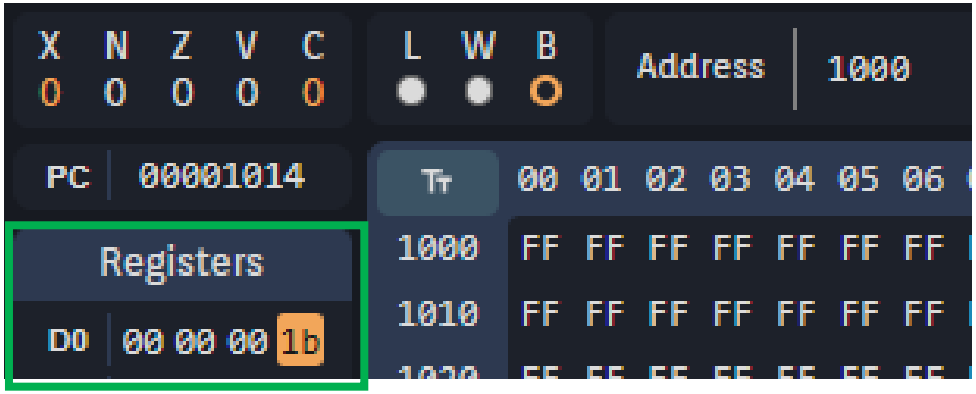
Capture des registres et expliquer

Nous avons : 45+12-48+18 = 27 c'est-à-dire $1B_{16}$

Une autre façon de faire



```
1   ORG $1000
2   START:
3       * Write here your code
4   LEA TAB,A0 ; A0 pointe à la première donnée
5    MOVE.W (A0)+,D0  ; D0 contient 45
6   * Il faut répéter «ADD.W» 9 fois
7   MOVE.W #8,D1 ; compteur initialisé à 9-1
8   BCL: ADD.W (A0)+,D0 ; Accumulation des données
9   DBRA D1, BCL
10  TRAP #15
11  TAB: DC.W 45,12,-48,18,-5,81,12,35,-18,89
12  END: * Jump here to end the program
```

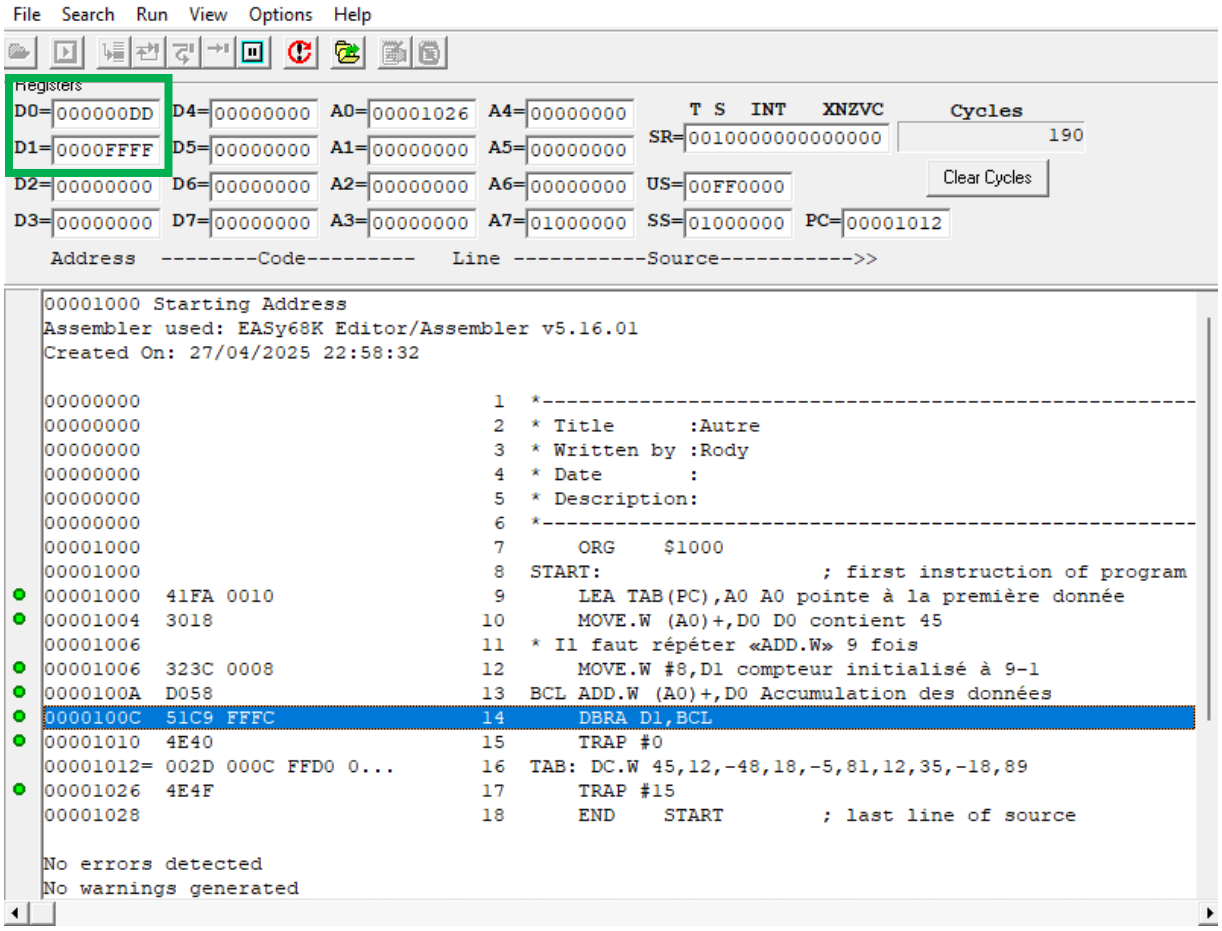


Capture des registres et expliquer

Nous avons : 45+12-48+18 = 27 c'est-à-dire $1B_{16}$

.

j) La même question que précédent mais il y a plus de données : 45 + 12 - 48 + 18 - 5 + 81 + 12 + 35 - 18 + 89. Notez la commande de qui fonctionne !



```
1   ORG $1000
2   START:
3       * Write here your code
4   LEA TAB, A0 ; A 0 pointe à la première donnée
5   MOVE.W (A0)+,D0 ;D0 contient 45
6   * Il faut répéter «ADD.W» 9 fois
7   MOVE.W #8,D1 ;compteur initialisé à 9-1
8   BCL: ADD.W (A0)+,D0 ;Accumulation des données
9   DBRA D1,BCL
10  TRAP #15
11  TAB: DC.W 45,12,-48,18,-5,81,12,35,-18,89
12  END: * Jump here to end the program
```

```
1   ORG $1000
2   START:
3       * Write here your code
4       LEA TAB,A0 ;A0 pointe à la première donnée
5       MOVE.W (A0)+,D0 ;D0 contient 45
6   * Il faut répéter «ADD.W» 9 fois
7       MOVE.W #8,D1  ;compteur initialisé à 9-1
8   BCL: ADD.W (A0)+,D0  ;Accumulation des données
9       DBRA D1,BCL
10      TRAP #15
11  TAB: DC.W 45,12,-48,18,-5,81,12,35,-18,89
12  END: * Jump here to end the program
```

X	N	Z	V	C
0	0	0	0	0

L	W	B
●	O	●

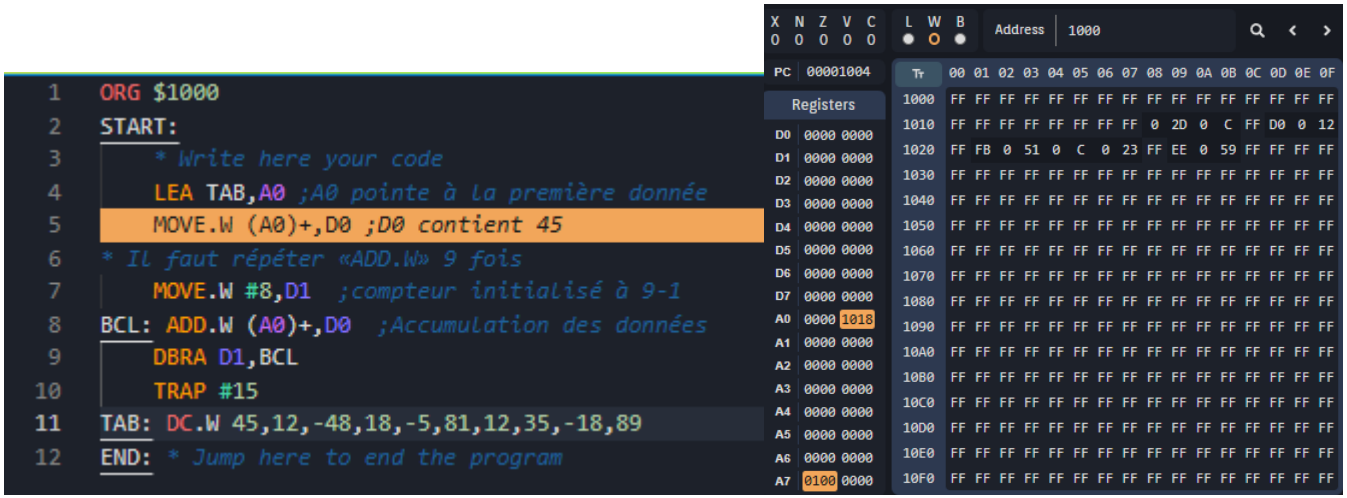
Address 1000

PC 00001000

Registers

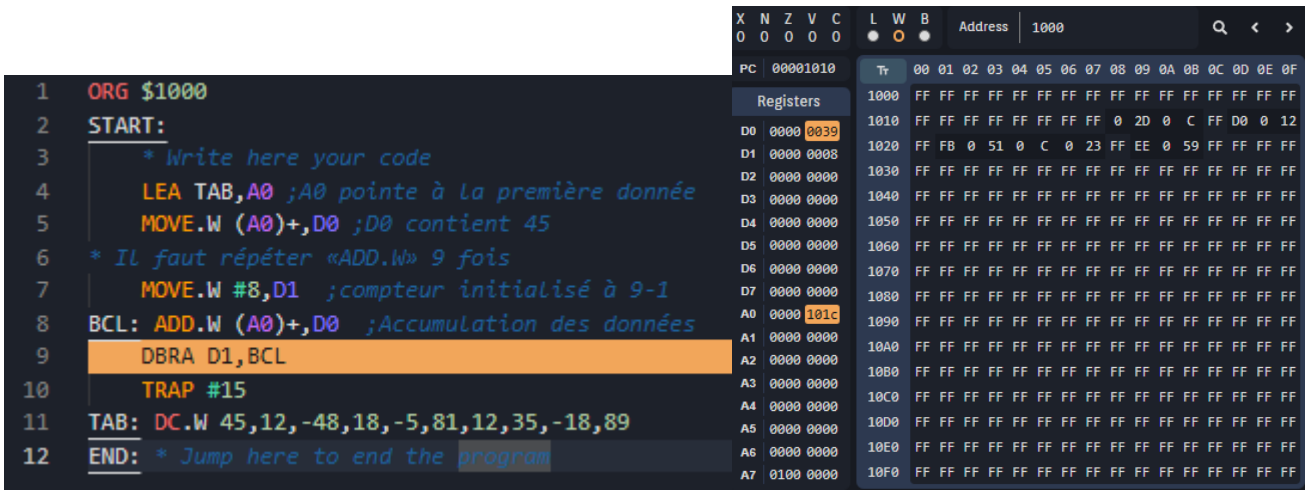
D0	0000 0000
D1	0000 0000
D2	0000 0000

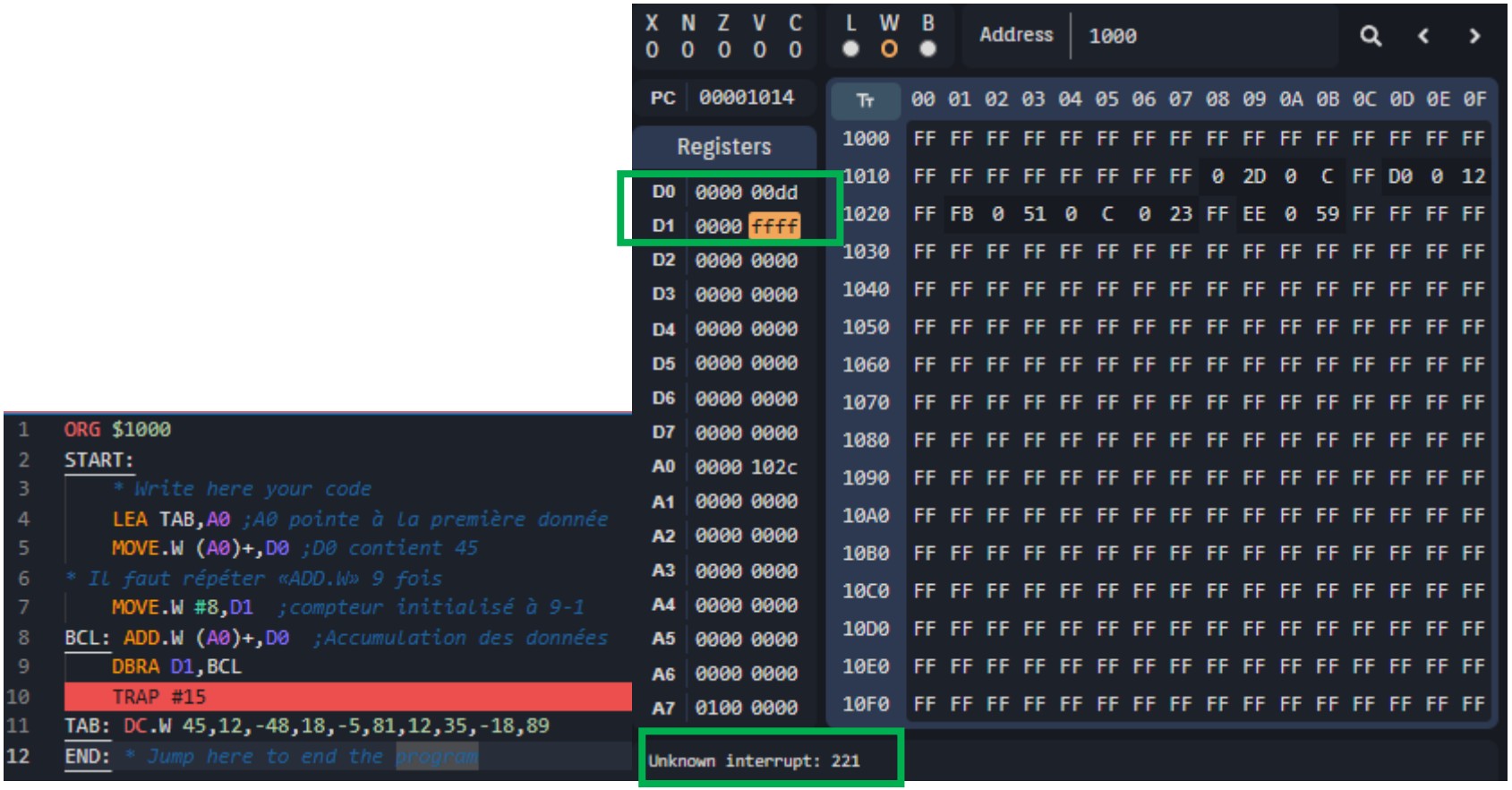
Tr	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
1000	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
1010	FF	FF	FF	FF	FF	FF	FF	FF	0	2D	0	C	FF	D0	0	12
1020	FF	FB	0	51	0	C	0	23	FF	EE	0	59	FF	FF	FF	FF
1030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF


1 ORG $1000
2 START:
3 * Write here your code
4 LEA TAB,A0 ;A0 pointe à la première donnée
5 MOVE.W (A0)+,D0 ;D0 contient 45
6 * Il faut répéter «ADD.W» 9 fois
7 MOVE.W #8,D1 ;compteur initialisé à 9-1
8 BCL: ADD.W (A0)+,D0 ;Accumulation des données
9 DBRA D1,BCL
10 TRAP #15
11 TAB: DC.W 45,12,-48,18,-5,81,12,35,-18,89
12 END: * Jump here to end the program
X N Z V C
0 0 0 0 0
L W B
Address 1000
PC 00001004
Registers
D0 0000 0000
D1 0000 0000
D2 0000 0000
D3 0000 0000
D4 0000 0000
D5 0000 0000
D6 0000 0000
D7 0000 0000
A0 0000 1018
A1 0000 0000
A2 0000 0000
A3 0000 0000
A4 0000 0000
A5 0000 0000
A6 0000 0000
A7 0100 0000
Tr 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
1000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1010 FF FF FF FF FF FF FF FF 0 2D 0 C FF D0 0 12
1020 FF FB 0 51 0 C 0 23 FF EE 0 59 FF FF FF FF
1030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF


1 ORG $1000
2 START:
3 * Write here your code
4 LEA TAB,A0 ;A0 pointe à la première donnée
5 MOVE.W (A0)+,D0 ;D0 contient 45
6 * Il faut répéter «ADD.W» 9 fois
7 MOVE.W #8,D1 ;compteur initialisé à 9-1
8 BCL: ADD.W (A0)+,D0 ;Accumulation des données
9 DBRA D1,BCL
10 TRAP #15
11 TAB: DC.W 45,12,-48,18,-5,81,12,35,-18,89
12 END: * Jump here to end the program
X N Z V C
0 0 0 0 0
L W B
Address 1000
PC 00001008
Registers
D0 0000 002d
D1 0000 0000
D2 0000 0000
D3 0000 0000
D4 0000 0000
D5 0000 0000
D6 0000 0000
D7 0000 0000
A0 0000 101a
A1 0000 0000
A2 0000 0000
A3 0000 0000
A4 0000 0000
A5 0000 0000
A6 0000 0000
A7 0100 0000
Tr 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
1000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1010 FF FF FF FF FF FF FF FF 0 2D 0 C FF D0 0 12
1020 FF FB 0 51 0 C 0 23 FF EE 0 59 FF FF FF FF
1030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF


ORG $1000
START:
* Write here your code
LEA TAB,A0 ;A0 pointe à la première donnée
MOVE.W (A0)+,D0 ;D0 contient 45
* Il faut répéter «ADD.W» 9 fois
MOVE.W #8,D1 ;compteur initialisé à 9-1
BCL: ADD.W (A0)+,D0 ;Accumulation des données
DBRA D1,BCL
TRAP #15
TAB: DC.W 45,12,-48,18,-5,81,12,35,-18,89
END: * Jump here to end the program
X N Z V C
0 0 0 0 0
L W B
Address
1000
PC 0000100c
Registers
D0 0000 002d
D1 0000 0008
D2 0000 0000
D3 0000 0000
D4 0000 0000
D5 0000 0000
D6 0000 0000
D7 0000 0000
A0 0000 101a
A1 0000 0000
A2 0000 0000
A3 0000 0000
A4 0000 0000
A5 0000 0000
A6 0000 0000
A7 0100 0000
Tr 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
1000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1010 FF FF FF FF FF FF FF FF 0 2D 0 C FF D0 0 12
1020 FF FB 0 51 0 C 0 23 FF EE 0 59 FF FF FF FF
1030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

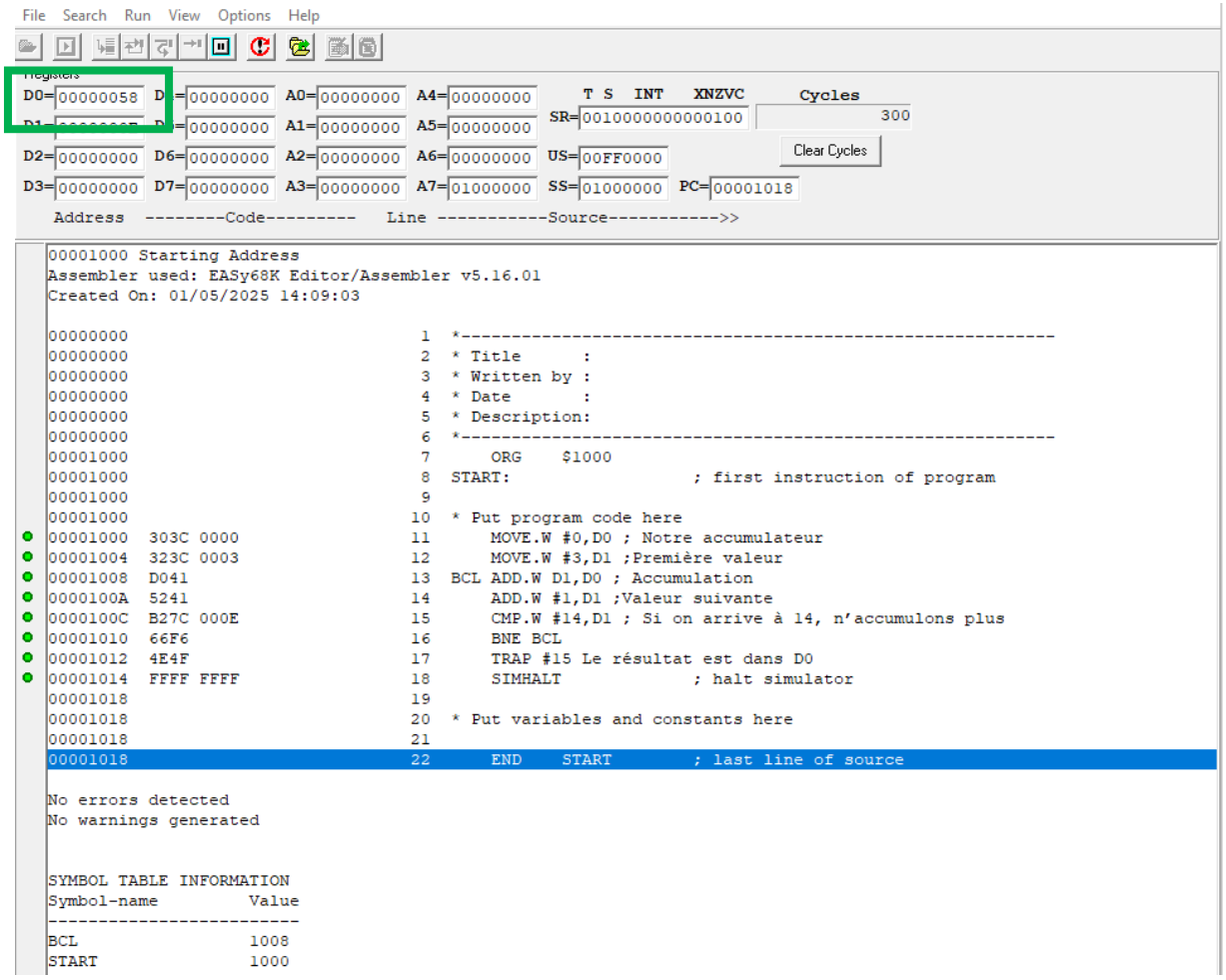

ORG $1000
START:
* Write here your code
LEA TAB,A0 ;A0 pointe à la première donnée
MOVE.W (A0)+,D0 ;D0 contient 45
* Il faut répéter «ADD.W» 9 fois
MOVE.W #8,D1 ;compteur initialisé à 9-1
BCL: ADD.W (A0)+,D0 ;Accumulation des données
DBRA D1,BCL
TRAP #15
TAB: DC.W 45,12,-48,18,-5,81,12,35,-18,89
END: * Jump here to end the program
X N Z V C
0 0 0 0 0
L W B
Address
1000
PC 00001010
Registers
D0 0000 0039
D1 0000 0008
D2 0000 0000
D3 0000 0000
D4 0000 0000
D5 0000 0000
D6 0000 0000
D7 0000 0000
A0 0000 101c
A1 0000 0000
A2 0000 0000
A3 0000 0000
A4 0000 0000
A5 0000 0000
A6 0000 0000
A7 0100 0000
Tr 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
1000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1010 FF FF FF FF FF FF FF FF 0 2D 0 C FF D0 0 12
1020 FF FB 0 51 0 C 0 23 FF EE 0 59 FF FF FF FF
1030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF



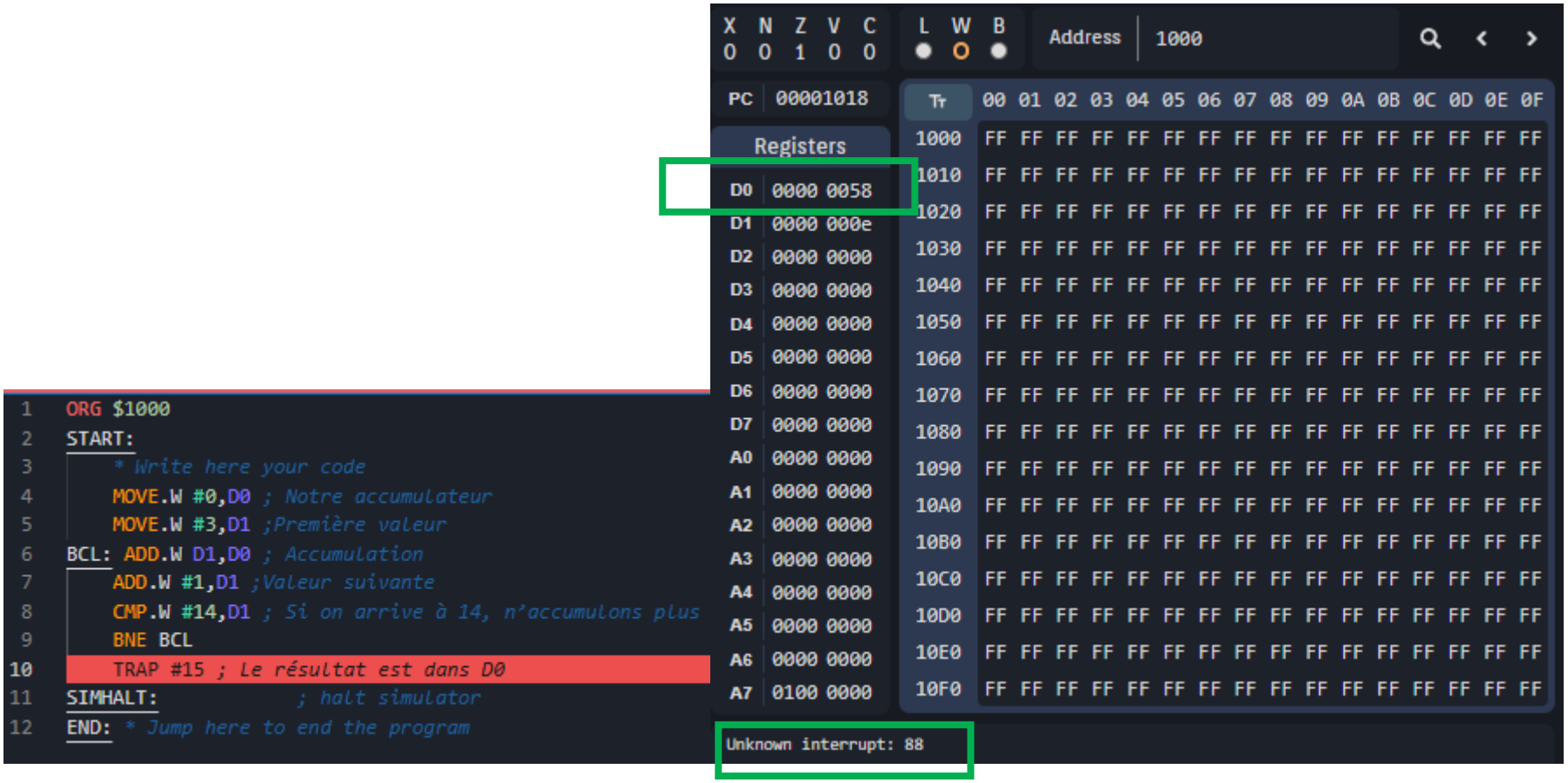

```
1   ORG $1000
2   START:
3       * Write here your code
4       LEA TAB,A0 ;A0 pointe à la première donnée
5       MOVE.W (A0)+,D0 ;D0 contient 45
6   * Il faut répéter «ADD.W» 9 fois
7       MOVE.W #8,D1  ;compteur initialisé à 9-1
8   BCL: ADD.W (A0)+,D0  ;Accumulation des données
9       DBRA D1,BCL
10      TRAP #15
11  TAB: DC.W 45,12,-48,18,-5,81,12,35,-18,89
12  END: * Jump here to end the program
```

$221_{10} = DD_{16}$

k) Calculer la somme 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13. Notez le code qui fonctionne !



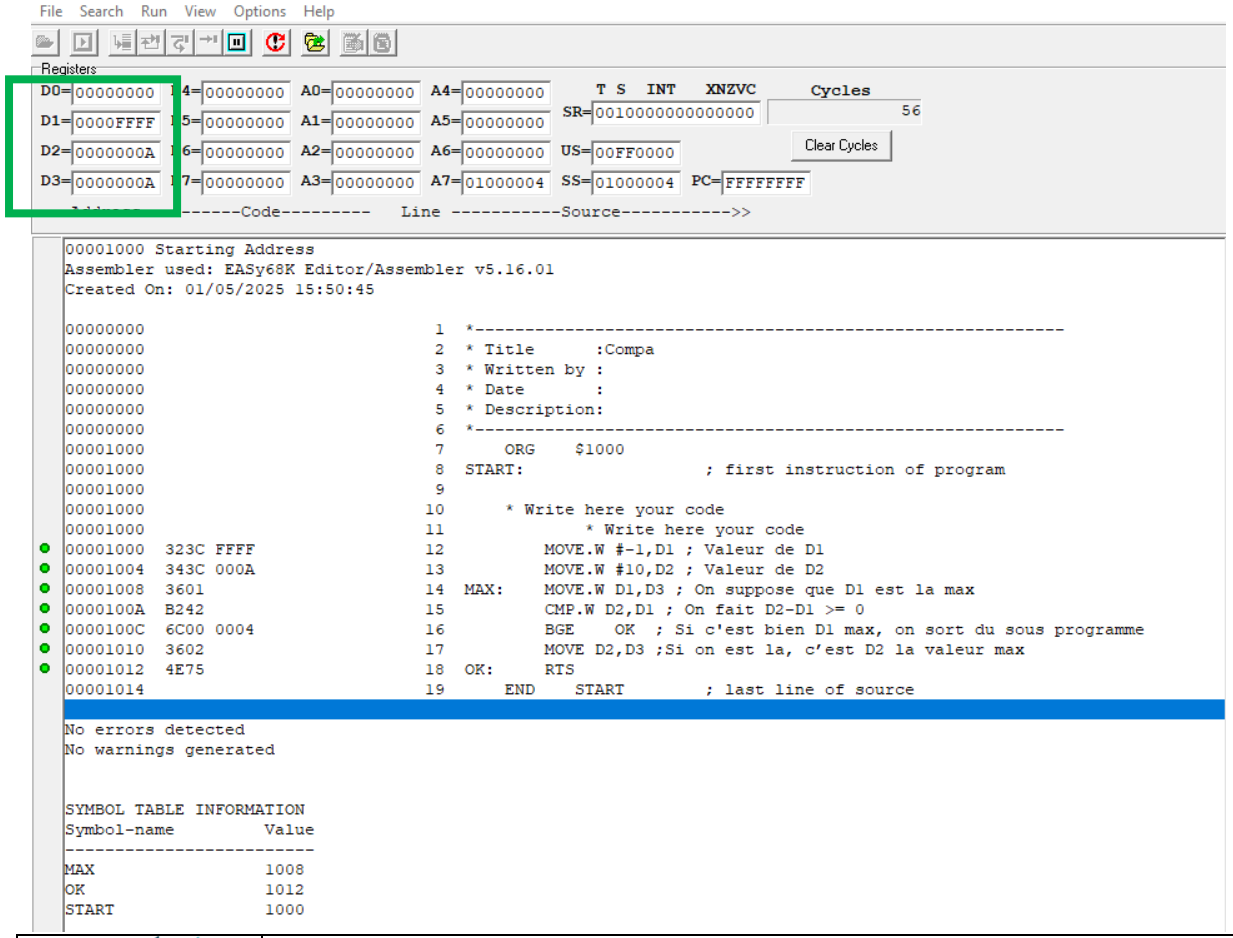
```
1   ORG $1000
2   START:
3       * Write here your code
4       MOVE.W #0,D0 ; Notre accumulateur
5       MOVE.W #3,D1 ;Première valeur
6   BCL: ADD.W D1,D0 ; Accumulation
7       ADD.W #1,D1 ;Valeur suivante
8       CMP.W #14,D1 ; Si on arrive à 14, n'accumulons plus
9       BNE BCL
10      TRAP #15 ; Le résultat est dans D0
11  SIMHALT:            ; halt simulator
12  END: * Jump here to end the program
```

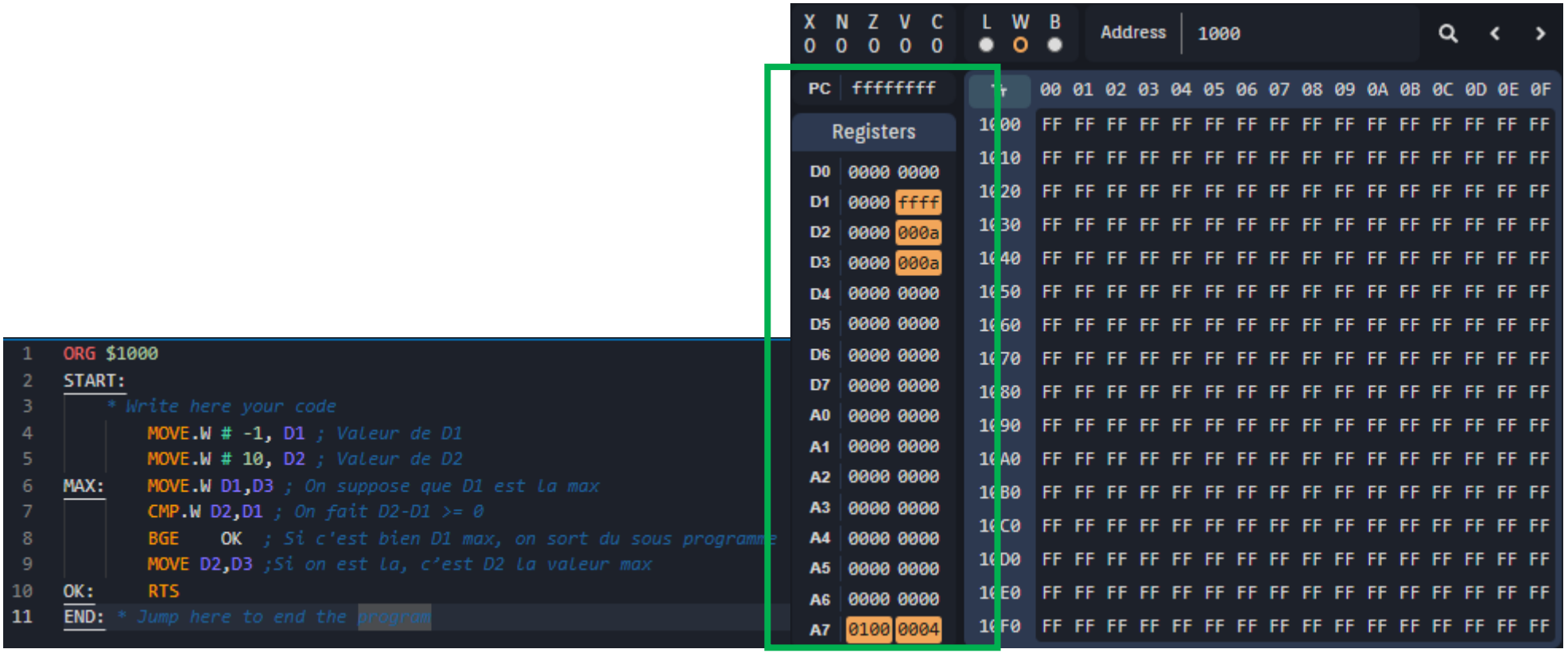



```
1   ORG $1000
2   START:
3       * Write here your code
4       MOVE.W #0,D0 ; Notre accumulateur
5       MOVE.W #3,D1 ;Première valeur
6   BCL: ADD.W D1,D0 ; Accumulation
7       ADD.W #1,D1 ;Valeur suivante
8       CMP.W #14,D1 ; Si on arrive à 14, n'accumulons plus
9       BNE BCL
10      TRAP #15 ; Le résultat est dans D0
11  SIMHALT:            ; halt simulator
12  END: * Jump here to end the program
```

$88_{10} = 58_{16}$

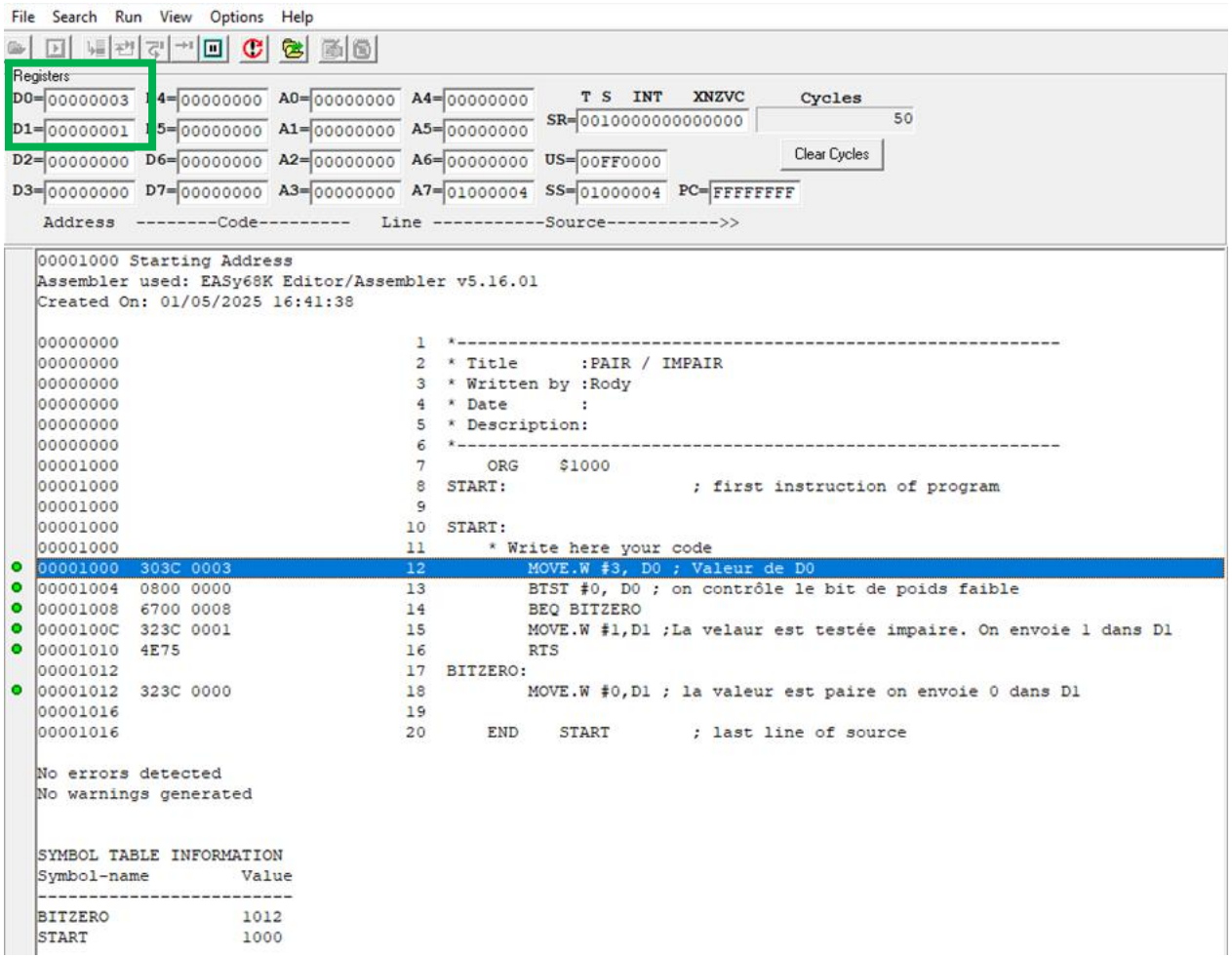
I) Ecrire un sous-programme qui reçoit deux valeurs signées (mots) dans D1 et D2 et qui retourne le max des deux dans D3. Tester le programme pour différentes valeurs de D1 et D2.



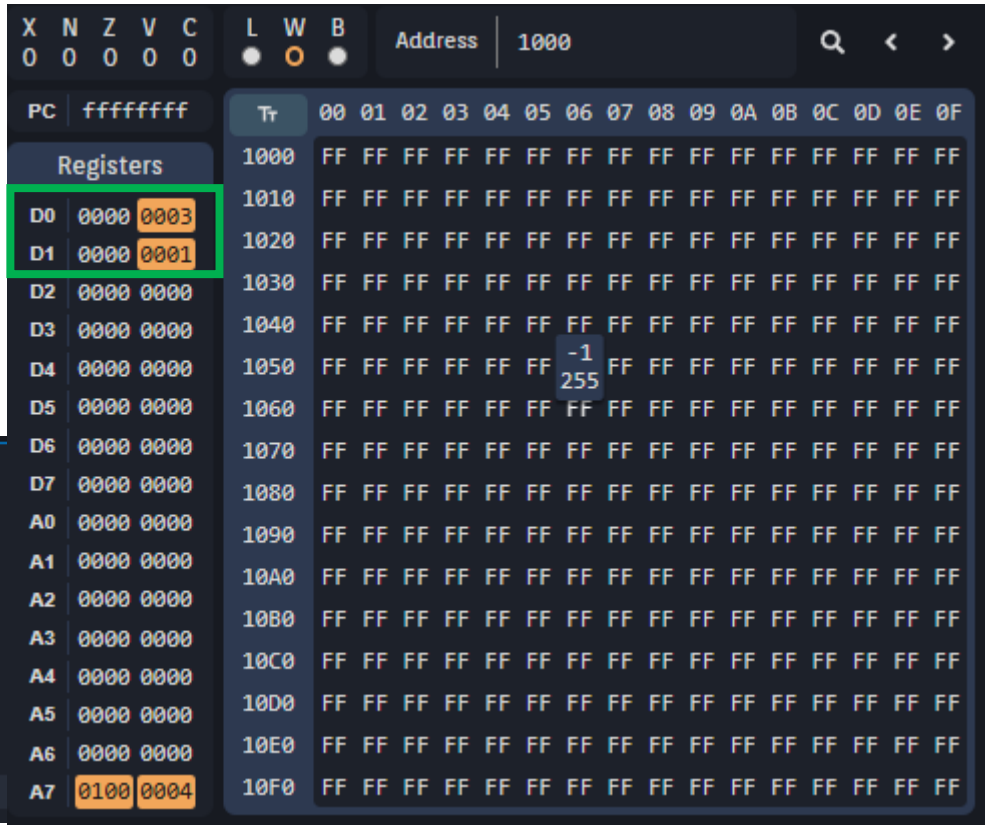



```
1   ORG $1000
2   START:
3       * Write here your code
4           MOVE.W # -1, D1 ; Valeur de D1
5           MOVE.W # 10, D2 ; Valeur de D2
6   MAX:    MOVE.W D1,D3 ; On suppose que D1 est la max
7           CMP.W D2,D1 ; On fait D2-D1 >= 0
8           BGE    OK  ; Si c'est bien D1 max, on sort du sous programme
9           MOVE D2,D3 ;Si on est la, c'est D2 la valeur max
10  OK:     RTS
11  END: * Jump here to end the program
```

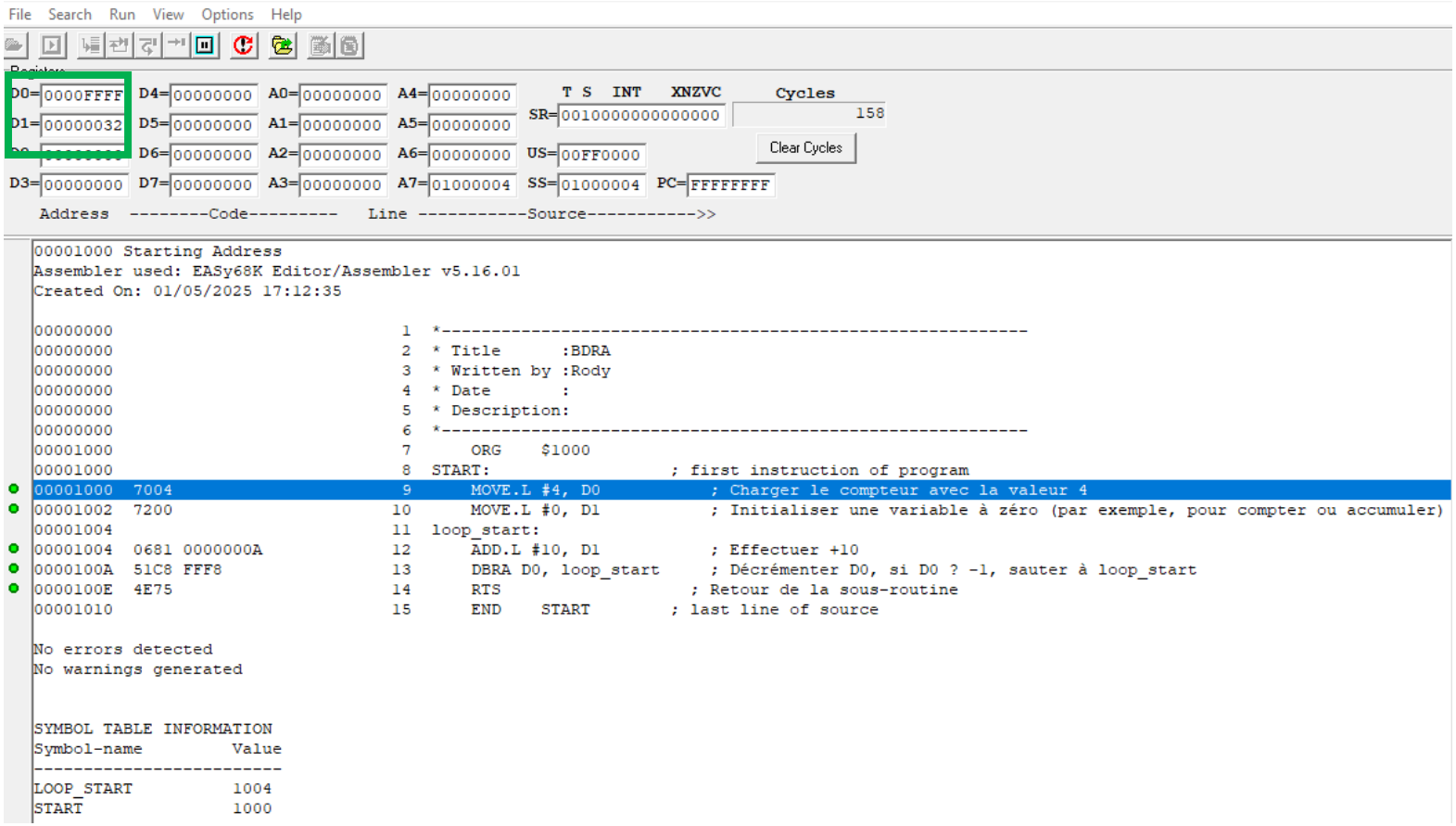
m) Ecrire un sous-programme qui reçoit une valeur dans D0 et qui retourne dans D1, 1 si D0 est impaire et 0 si D0 est paire.



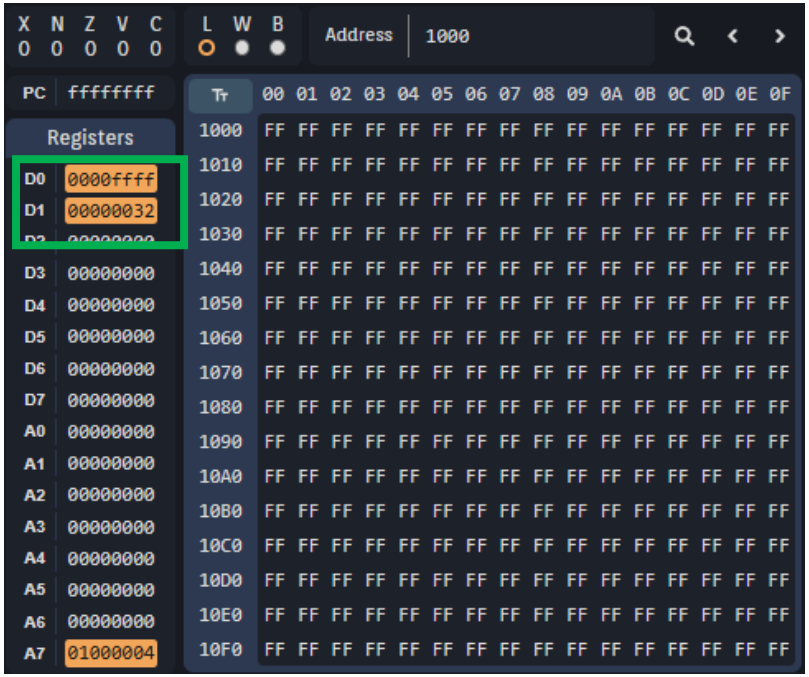
```
1   ORG $1000
2   START:
3       * Write here your code
4           MOVE.W #3, D0 ; Valeur de D0
5           BTST #0, D0 ; on contrôle le bit de poids faible
6           BEQ BITZERO
7           MOVE.W #1,D1 ;La velaur est testée impaire. On envoie 1 dans D1
8           RTS
9   BITZERO:
10          MOVE.W #0,D1 ; la valeur est paire. On envoie 0 dans D1
11  END: * Jump here to end the program
```



n) On souhaite calculer 5*10 sans utilisation de la commande MUL. C'est-à-dire que l'on calcule 10+10+10+10+10. Ecrire le programme en utilisant l'instruction DBRA. Notez le code qui fonctionne !



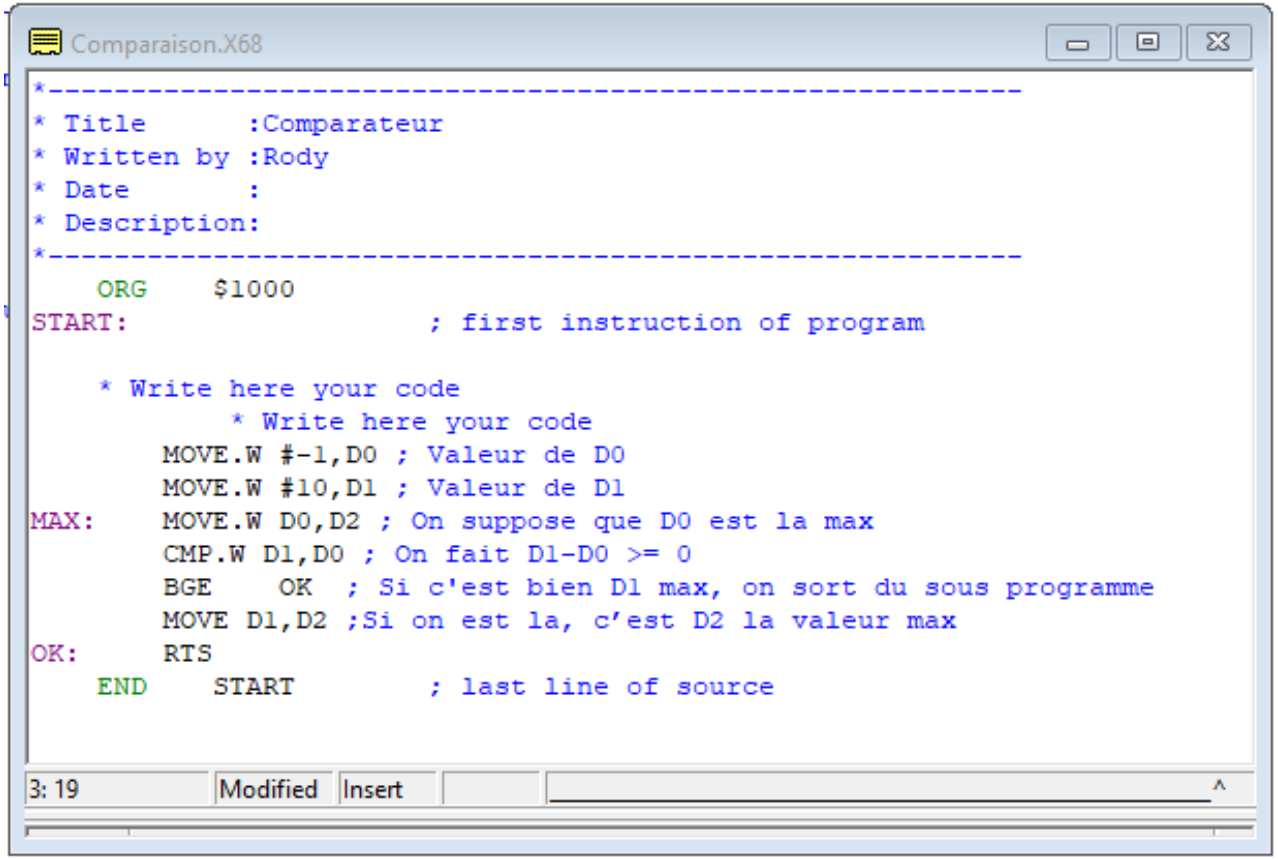
```
1   ORG $1000
2   START:
3       MOVE.L #4, D0           ; Charger le compteur avec la valeur 4
4       MOVE.L #0, D1           ; Initialiser une variable à zéro (par exemple, pour compter ou accumuler)
5   loop_start:
6       ADD.L #10, D1           ; Effectuer +10
7       DBRA D0, loop_start     ; Décrémenter D0, si D0 ≠ -1, sauter à loop_start
8       RTS                   ; Retour de la sous-routine
9   END: * Jump here to end the program
```



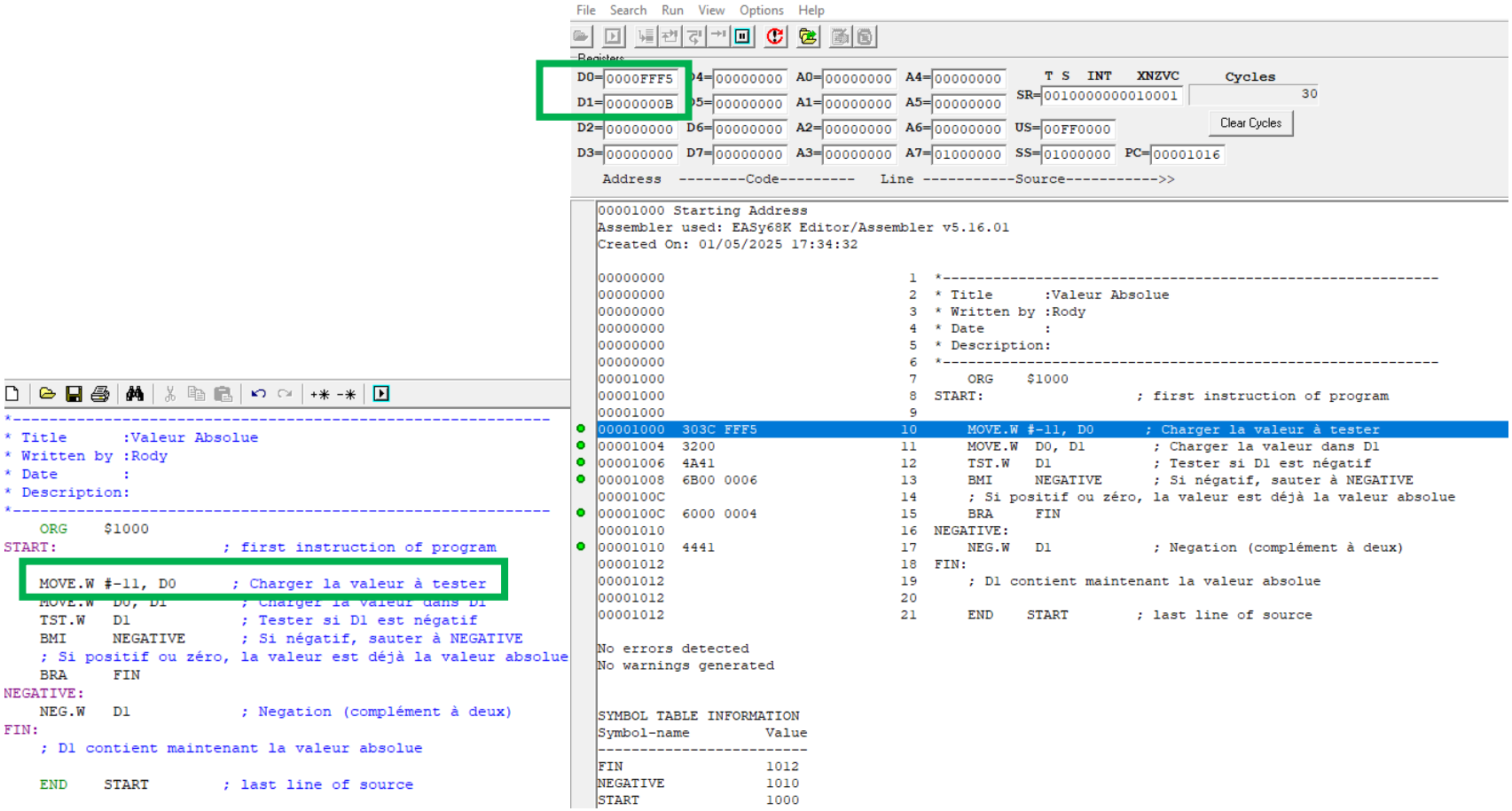
$D1 = 32_{16} = 50_{10} = 10+10+10+10+10$

o) Ecrire un sous-programme qui reçoit deux valeurs signées (mots) dans D0 et D1 et qui retourne le max des deux dans D2. Notez le code qui fonctionne !

Idem question l avec des noms de registres différents.



p) Ecrire un sous-programme ABS qui reçoit une valeur signée dans D0 et qui retourne sa valeur absolue dans D1. Testez votre programme pour des valeurs positive et négative de D0.



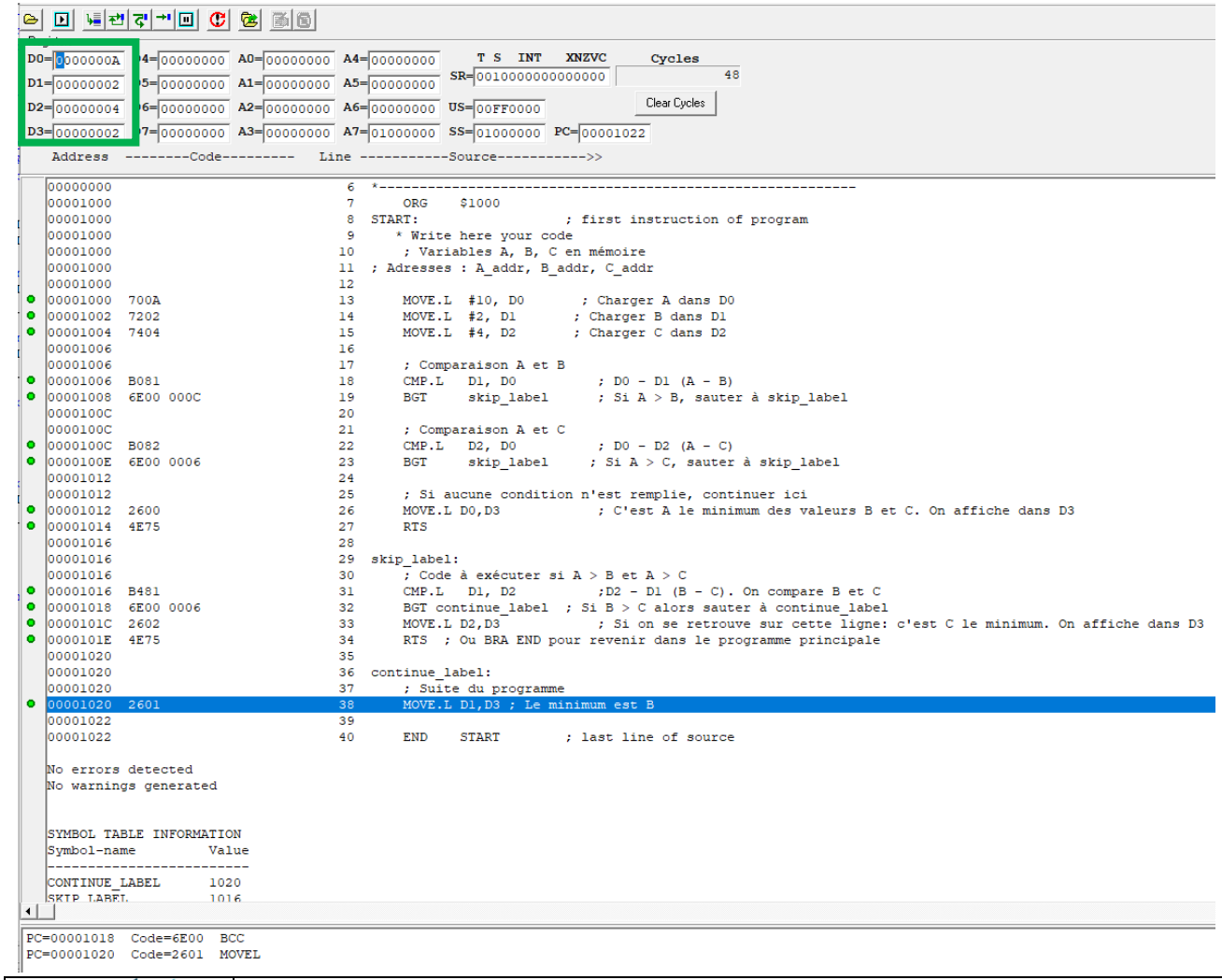
```
*-----------------------------------------------------------
* Title      :Valeur Absolue
* Written by :Rody
* Date       :
* Description:
*-----------------------------------------------------------
    ORG    $1000
START:                  ; first instruction of program

    MOVE.W #-11, D0      ; Charger la valeur à tester
    MOVE.W  D0, D1       ; Charger la valeur dans D1
    TST.W   D1           ; Tester si D1 est négatif
    BMI     NEGATIVE     ; Si négatif, sauter à NEGATIVE
    ; Si positif ou zéro, la valeur est déjà la valeur absolue
    BRA     FIN
NEGATIVE:
    NEG.W   D1           ; Negation (complément à deux)
FIN:
    ; D1 contient maintenant la valeur absolue

    END    START        ; last line of source
```

```
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 01/05/2025 17:34:32

00000000                             1  *-----------------------------------------------------------
00000000                             2  * Title      :Valeur Absolue
00000000                             3  * Written by :Rody
00000000                             4  * Date       :
00000000                             5  * Description:
00000000                             6  *-----------------------------------------------------------
00001000                             7      ORG    $1000
00001000                             8  START:                  ; first instruction of program
00001000                             9
00001000  303C FFF5                 10      MOVE.W #-11, D0      ; Charger la valeur à tester
00001004  3200                      11      MOVE.W  D0, D1       ; Charger la valeur dans D1
00001006  4A41                      12      TST.W   D1           ; Tester si D1 est négatif
00001008  6B00 0006                 13      BMI     NEGATIVE     ; Si négatif, sauter à NEGATIVE
0000100C                            14      ; Si positif ou zéro, la valeur est déjà la valeur absolue
0000100C  6000 0004                 15      BRA     FIN
00001010                            16  NEGATIVE:
00001010  4441                      17      NEG.W   D1           ; Negation (complément à deux)
00001012                            18  FIN:
00001012                            19      ; D1 contient maintenant la valeur absolue
00001012                            20
00001012                            21      END    START        ; last line of source

No errors detected
No warnings generated


SYMBOL TABLE INFORMATION
Symbol-name         Value
-------------------------
FIN                 1012
NEGATIVE            1010
START               1000
```


1 ORG $1000
2 START:
3 MOVE.W #-11, D0 ; Charger la valeur à tester
4 MOVE.W D0, D1 ; Charger la valeur dans D1
5 TST.W D1 ; Tester si D1 est négatif
6 BMI NEGATIVE ; Si négatif, sauter à NEGATIVE
7 ; Si positif ou zéro, la valeur est déjà la valeur absolue
8 BRA FIN
9 NEGATIVE:
10 NEG.W D1 ; Negation (complément à deux)
11 FIN:
12 ; D1 contient maintenant la valeur absolue
13 END: * Jump here to end the program
X N Z V C
1 0 0 0 1
L W B
Address 1000
PC 00001018
Registers
D0 0000 fff5
D1 0000 000b
D2 0000 0000
00 01 02 03 04 05 06 07
1000 FF FF FF FF FF FF FF FF
1010 FF FF FF FF FF FF FF FF
1020 FF FF FF FF FF FF FF FF
1030 FF FF FF FF FF FF FF FF

q) Ecrire un sous-programme MIN qui reçoit trois valeurs signées (mots) dans D0, D1 et D2 et qui retourne le minimum des trois dans D3.

```
*-----------------------------------------------------------
* Title      :Comparaison 3 Variables et afficher le minimum dans D3
* Written by :Rody
* Date       :
* Description:
*-----------------------------------------------------------
    ORG    $1000
START:                  ; first instruction of program
   * Write here your code
    ; Variables A, B, C en mémoire
; Adresses : A_addr, B_addr, C_addr

    MOVE.L  #10, D0        ; Charger A dans D0
    MOVE.L  #2, D1        ; Charger B dans D1
    MOVE.L  #4, D2        ; Charger C dans D2

    ; Comparaison A et B
    CMP.L   D1, D0          ; D0 - D1 (A - B)
    BGT     skip_label      ; Si A > B, sauter à skip_label

    ; Comparaison A et C
    CMP.L   D2, D0          ; D0 - D2 (A - C)
    BGT     skip_label     ; Si A > C, sauter à skip_label

    ; Si aucune condition n'est remplie, continuer ici
    MOVE.L D0,D3            ; C'est A le minimum des valeurs B et C. On affiche dans D3
    RTS

skip_label:
    ; Code à exécuter si A > B et A > C
    CMP.L   D1, D2          ;D2 - D1 (B - C). On compare B et C
    BGT continue_label  ; Si B > C alors sauter à continue_label
    MOVE.L D2,D3            ; Si on se retrouve sur cette ligne: c'est C le minimum. On affiche dans D3
    RTS

continue_label:
    ; Suite du programme
    MOVE.L D1,D3 ; Le minimum est B

    END    START        ; last line of source
```


D0=0000000A
D1=00000002
D2=00000004
D3=00000002
A0=00000000
A1=00000000
A2=00000000
A3=00000000
A4=00000000
A5=00000000
A6=00000000
A7=01000000
SR=0010000000000000
US=00FF0000
SS=01000000
PC=00001022
T S INT XNZVC
Cycles
48
Clear Cycles
Address --------Code--------- Line ----------Source----------->>
00000000 6 *---
00001000 7 ORG $1000
00001000 8 START: ; first instruction of program
00001000 9 * Write here your code
00001000 10 ; Variables A, B, C en mémoire
00001000 11 ; Adresses : A_addr, B_addr, C_addr
00001000 12
00001000 700A 13 MOVE.L #10, D0 ; Charger A dans D0
00001002 7202 14 MOVE.L #2, D1 ; Charger B dans D1
00001004 7404 15 MOVE.L #4, D2 ; Charger C dans D2
00001006 16
00001006 17 ; Comparaison A et B
00001006 B081 18 CMP.L D1, D0 ; D0 - D1 (A - B)
00001008 6E00 000C 19 BGT skip_label ; Si A > B, sauter à skip_label
0000100C 20
0000100C 21 ; Comparaison A et C
0000100C B082 22 CMP.L D2, D0 ; D0 - D2 (A - C)
0000100E 6E00 0006 23 BGT skip_label ; Si A > C, sauter à skip_label
00001012 24
00001012 25 ; Si aucune condition n'est remplie, continuer ici
00001012 2600 26 MOVE.L D0,D3 ; C'est A le minimum des valeurs B et C. On affiche dans D3
00001014 4E75 27 RTS
00001016 28
00001016 29 skip_label:
00001016 30 ; Code à exécuter si A > B et A > C
00001016 B481 31 CMP.L D1, D2 ;D2 - D1 (B - C). On compare B et C
00001018 6E00 0006 32 BGT continue_label ; Si B > C alors sauter à continue_label
0000101C 2602 33 MOVE.L D2,D3 ; Si on se retrouve sur cette ligne: c'est C le minimum. On affiche dans D3
0000101E 4E75 34 RTS ; Ou BRA END pour revenir dans le programme principale
00001020 35
00001020 36 continue_label:
00001020 37 ; Suite du programme
00001020 2601 38 MOVE.L D1,D3 ; Le minimum est B
00001022 39
00001022 40 END START ; last line of source
No errors detected
No warnings generated
SYMBOL TABLE INFORMATION
Symbol-name Value
CONTINUE_LABEL 1020
SKIP_LABEL 1016
PC=00001018 Code=6E00 BCC
PC=00001020 Code=2601 MOVEL

```
1  ORG $1000
2  START:
3      * Write here your code
4      ; Variables A, B, C en mémoire
5  ; Adresses : A_addr, B_addr, C_addr
6
7      MOVE.L  #10, D0       ; Charger A dans D0
8      MOVE.L  #2, D1        ; Charger B dans D1
9      MOVE.L  #4, D2        ; Charger C dans D2
10
11     ; Comparaison A et B
12     CMP.L   D1, D0          ; D0 - D1 (A - B)
13     BGT     skip_label      ; Si A > B, sauter à skip_label
14
15     ; Comparaison A et C
16     CMP.L   D2, D0          ; D0 - D2 (A - C)
17     BGT     skip_label     ; Si A > C, sauter à skip_label
18
19     ; Si aucune condition n'est remplie, continuer ici
20     MOVE.L D0,D3            ; C'est A le minimum des valeurs B et C. On affiche dans D3
21     BRA END
22
23 skip_label:
24     ; Code à exécuter si A > B et A > C
25     CMP.L   D1, D2          ;D2 - D1 (B - C). On compare B et C
26     BGT continue_label  ; Si B > C alors sauter à continue_label
27     MOVE.L D2,D3            ; Si on se retrouve sur cette ligne: c'est C le minimum. On affiche dans D3
28     RTS  ; Ou BRA END pour revenir dans le programme principale
29
30 continue_label:
31     ; Suite du programme
32     MOVE.L D1,D3 ; Le minimum est B
33
34 END: * Jump here to end the program
```

X	N	Z	V	C
0	0	0	0	0

L	W	B
○	●	●

Address 1000

PC 00001038

Registers

D0	0000000a
D1	00000002
D2	00000004
D3	00000002

Tr	00	01	02	03	04	05	06
1000	FF	FF	FF	FF	FF	FF	FF
1010	FF	FF	FF	FF	FF	FF	FF
1020	FF	FF	FF	FF	FF	FF	FF
1030	FF	FF	FF	FF	FF	FF	FF
1040	FF	FF	FF	FF	FF	FF	FF

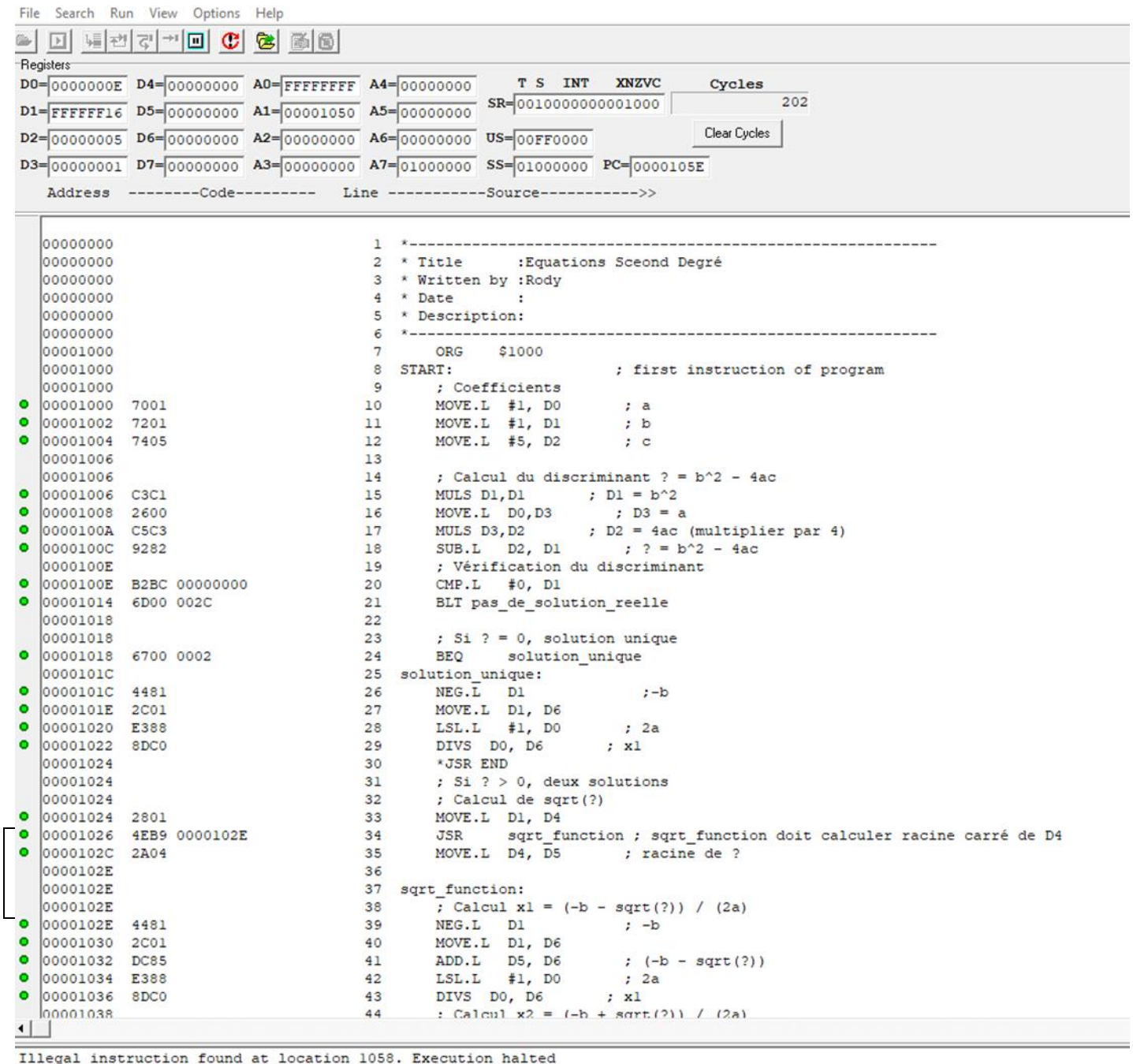
La plus petite des valeurs de l'exemple 10, 2 et 4 est bien D3= 2.

r) Ecrire un sous-programme FONC qui reçoit une variable x dans D0.W et qui retourne dans D1.L la valeur : $x^2 + x + 5$

```
*-----------------------------------------------------------
* Title      :Equations Second Degré
* Written by :Rody
* Date       :
* Description:
*-----------------------------------------------------------
    ORG    $1000
START:                  ; first instruction of program
    ; Coefficients
    MOVE.L  #1, D0       ; a
    MOVE.L  #1, D1       ; b
    MOVE.L  #5, D2       ; c

    ; Calcul du discriminant ? = b^2 - 4ac
    MULS D1,D1       ; D1 = b^2
    MOVE.L  D0,D3       ; D3 = a
    MULS D3,D2       ; D2 = 4ac (multiplier par 4)
    SUB.L   D2, D1       ; ? = b^2 - 4ac
    ; Vérification du discriminant
    CMP.L   #0, D1
    BLT pas_de_solution_reelle

    ; Si ? = 0, solution unique
    BEQ     solution_unique
solution_unique:
    NEG.L   D1             ;-b
    MOVE.L  D1, D6
    LSL.L   #1, D0       ; 2a
    DIVS  D0, D6       ; x1
    *JSR END
    ; Si ? > 0, deux solutions
    ; Calcul de sqrt(?)
```

```
    MOVE.L  D1, D4
    JSR     sqrt_function ; sqrt_function doit calculer racine carré de D4
    MOVE.L  D4, D5        ; racine de ?

sqrt_function:
    ; Calcul x1 = (-b - sqrt(?)) / (2a)
    NEG.L   D1            ; -b
    MOVE.L  D1, D6
    ADD.L   D5, D6        ; (-b - sqrt(?))
    LSL.L   #1, D0        ; 2a
    DIVS  D0, D6        ; x1
    ; Calcul x2 = (-b + sqrt(?)) / (2a)
    NEG.L   D1
    MOVE.L  D1, D7
    SUB.L   D5, D7        ; (-b + sqrt(?))
    LSL.L   #1, D0
    DIVS  D0,D7        ; x2
    *JSR END

pas_de_solution_reelle:
    LEA MESSAGE, A1 ; chargement du message dans le registre A1
    MOVE.B #14,D0 ;
    MOVE.B D0,D1 ;
    TRAP #15
MESSAGE: DC.B    'PAS DE SOLUTION REELLES !',0

FIN:
; Résultats dans D6 et D7
    END    START        ; last line of source
```

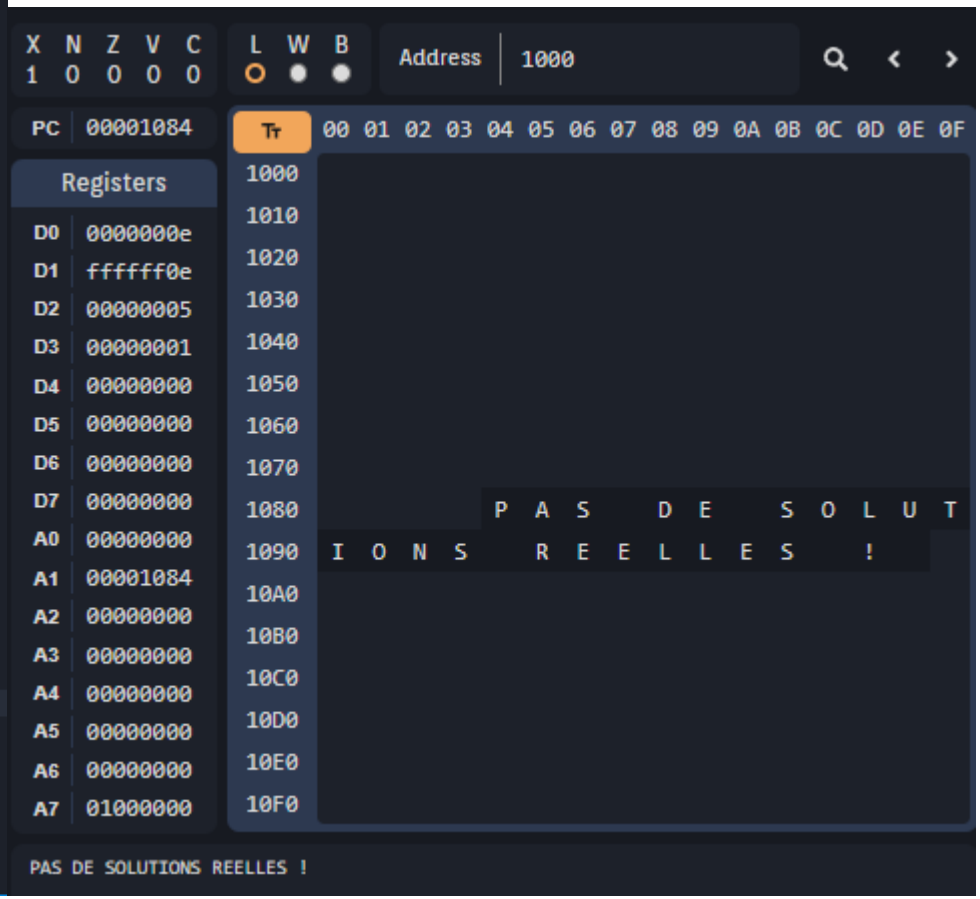
TD/Cours Architecture des ordinateurs	BACHELOR EN CYBERSECURITE	PRENOM : NOM :



```
File  Search  Run  View  Options  Help

Registers
D0=0000000E  D4=00000000  A0=FFFFFFFF  A4=00000000     T S  INT   XNZVC      Cycles
D1=FFFFFF16  D5=00000000  A1=00001050  A5=00000000  SR=0010000000001000            202
D2=00000005  D6=00000000  A2=00000000  A6=00000000  US=00FF0000     Clear Cycles
D3=00000001  D7=00000000  A3=00000000  A7=01000000  SS=01000000  PC=0000105E
   Address  --------Code---------   Line -----------Source----------->>

00000000                            1  *-----------------------------------------------------------
00000000                            2  * Title      :Equations Sceond Degré
00000000                            3  * Written by :Rody
00000000                            4  * Date       :
00000000                            5  * Description:
00000000                            6  *-----------------------------------------------------------
00001000                            7      ORG    $1000
00001000                            8  START:                  ; first instruction of program
00001000                            9      ; Coefficients
00001000  7001                     10      MOVE.L  #1, D0       ; a
00001002  7201                     11      MOVE.L  #1, D1       ; b
00001004  7405                     12      MOVE.L  #5, D2       ; c
00001006                           13
00001006                           14      ; Calcul du discriminant ? = b^2 - 4ac
00001006  C3C1                     15      MULS D1,D1       ; D1 = b^2
00001008  2600                     16      MOVE.L  D0,D3       ; D3 = a
0000100A  C5C3                     17      MULS D3,D2       ; D2 = 4ac (multiplier par 4)
0000100C  9282                     18      SUB.L   D2, D1       ; ? = b^2 - 4ac
0000100E                           19      ; Vérification du discriminant
0000100E  B2BC 00000000            20      CMP.L   #0, D1
00001014  6D00 002C                21      BLT pas_de_solution_reelle
00001018                           22
00001018                           23      ; Si ? = 0, solution unique
00001018  6700 0002                24      BEQ     solution_unique
0000101C                           25  solution_unique:
0000101C  4481                     26      NEG.L   D1              ;-b
0000101E  2C01                     27      MOVE.L  D1, D6
00001020  E388                     28      LSL.L   #1, D0       ; 2a
00001022  8DC0                     29      DIVS  D0, D6       ; x1
00001024                           30      *JSR END
00001024                           31      ; Si ? > 0, deux solutions
00001024                           32      ; Calcul de sqrt(?)
00001024  2801                     33      MOVE.L  D1, D4
00001026  4EB9 0000102E            34      JSR     sqrt_function ; sqrt_function doit calculer racine carré de D4
0000102C  2A04                     35      MOVE.L  D4, D5       ; racine de ?
0000102E                           36
0000102E                           37  sqrt_function:
0000102E                           38      ; Calcul x1 = (-b - sqrt(?)) / (2a)
0000102E  4481                     39      NEG.L   D1           ; -b
00001030  2C01                     40      MOVE.L  D1, D6
00001032  DC85                     41      ADD.L   D5, D6       ; (-b - sqrt(?))
00001034  E388                     42      LSL.L   #1, D0       ; 2a
00001036  8DC0                     43      DIVS  D0, D6       ; x1
00001038                           44      ; Calcul x2 = (-b + sqrt(?)) / (2a)

Illegal instruction found at location 1058. Execution halted
```

TD/Cours Architecture des ordinateurs	BACHELOR EN CYBERSECURITE	PRENOM : NOM :

```
ORG $1000
START:
  ; Coefficients
MOVE.L  #1, D0       ; a
MOVE.L  #1, D1       ; b
MOVE.L  #5, D2       ; c
; Calcul du discriminant Δ = b^2 - 4ac
MULS D1,D1       ; D1 = b^2
MOVE.L  D0,D3       ; D3 = a
MULS D3,D2       ; D2 = 4ac (multiplier par 4)
SUB.L   D2, D1       ; Δ = b^2 - 4ac
; Vérification du discriminant
CMP.L   #0, D1
BLT     pas_de_solution_reelle
; Si Δ = 0, solution unique
BEQ     solution_unique
solution_unique:
NEG.L   D1           ;-b
MOVE.L  D1, D6
LSL.L   #1, D0       ; 2a
DIVS  D0, D6       ; x1
JMP END
; Si Δ > 0, deux solutions
; Calcul de sqrt(Δ)
MOVE.L  D1, D4
JSR     sqrt_function ; sqrt_function doit calculer racine carré de D4
MOVE.L  D4, D5       ; racine de Δ

sqrt_function:
; Calcul x1 = (-b - sqrt(Δ)) / (2a)
NEG.L   D1           ; -b
MOVE.L  D1, D6
ADD.L   D5, D6       ; (-b - sqrt(Δ))
LSL.L   #1, D0       ; 2a
DIVS  D0, D6       ; x1
; Calcul x2 = (-b + sqrt(Δ)) / (2a)
NEG.L   D1
MOVE.L  D1, D7
SUB.L   D5, D7       ; (-b + sqrt(Δ))
LSL.L   #1, D0
DIVS  D0,D7       ; x2
JMP END
pas_de_solution_reelle:
LEA MESSAGE, A1 ; chargement du message dans le registre A1
MOVE.B #14,D0 ;
MOVE.B D0,D1 ;
TRAP #15
MESSAGE: DC.B    'PAS DE SOLUTION REELLES !',0
FIN:
; Résultats dans D6 et D7
```



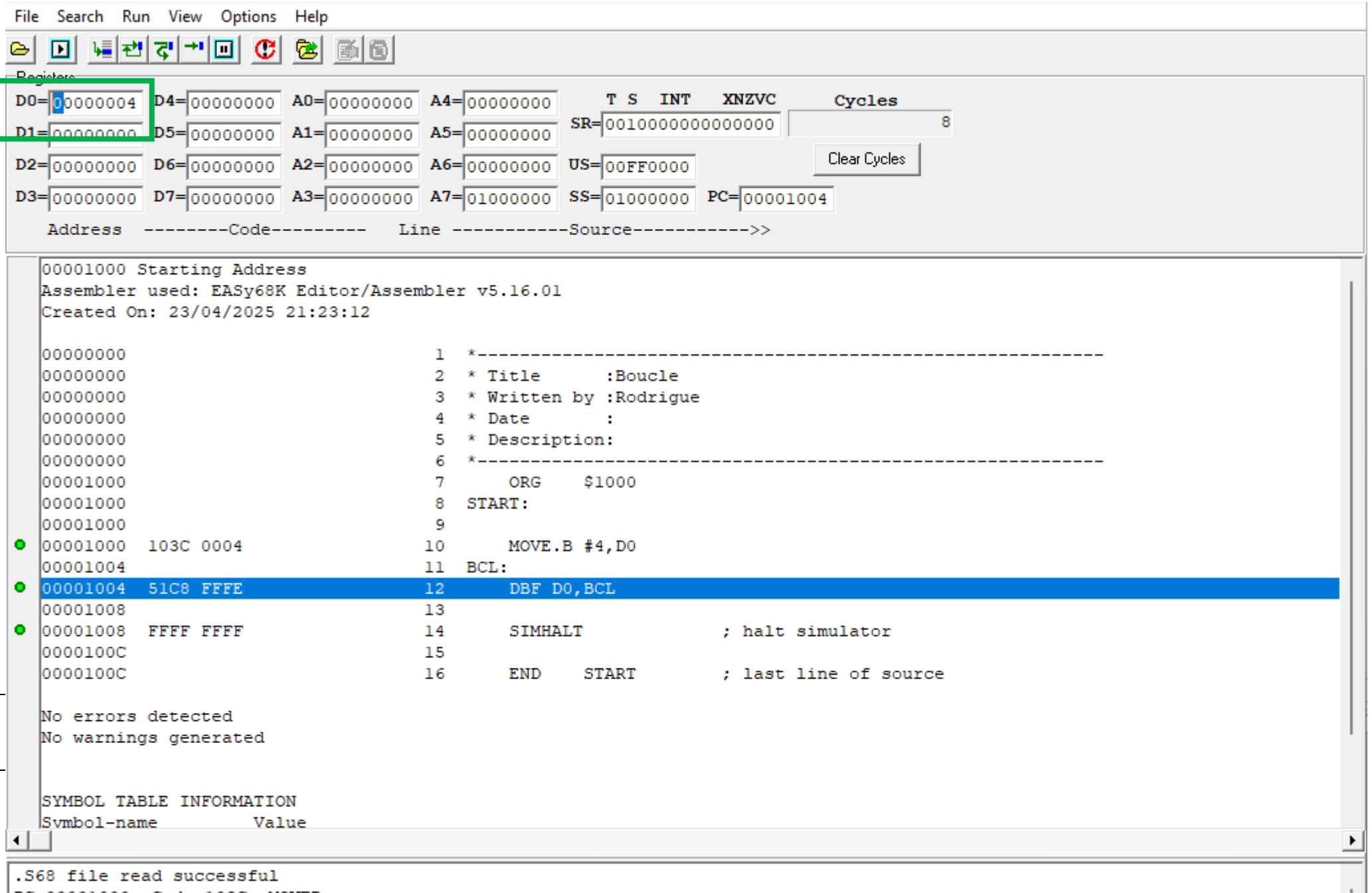
s) Exercice final

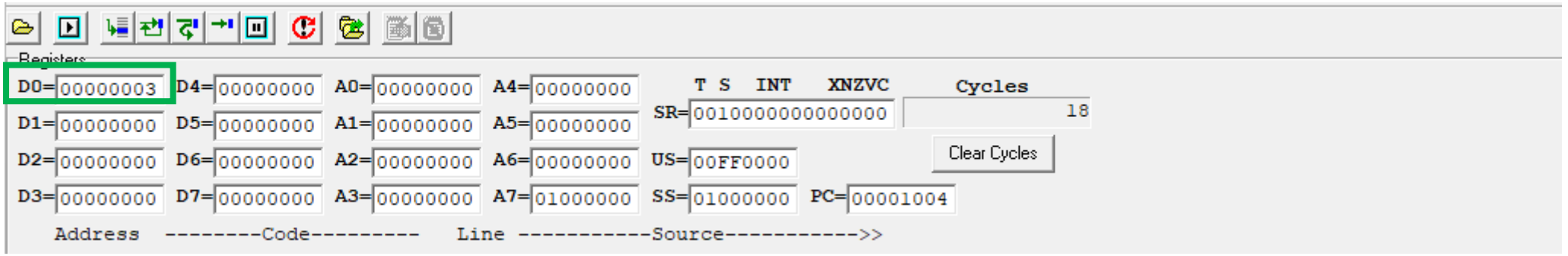
Soit le code suivant :

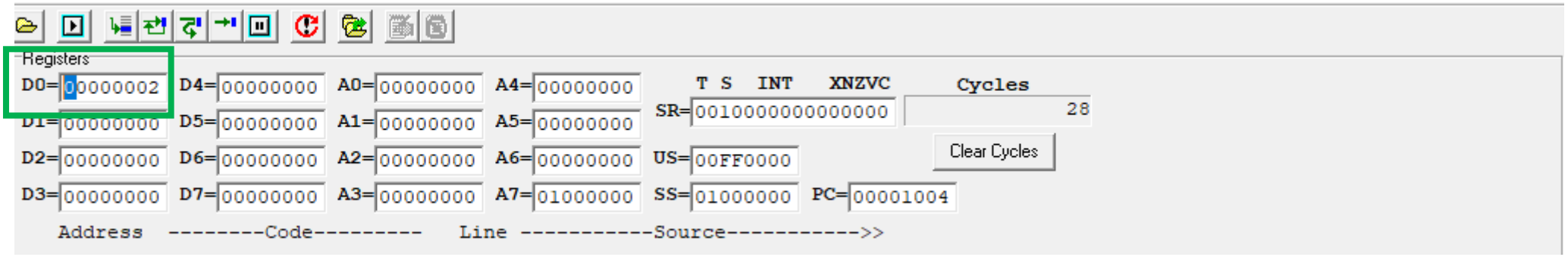
```
MOVE.B #5,D0
BCL:
DBF D0,BCL
```

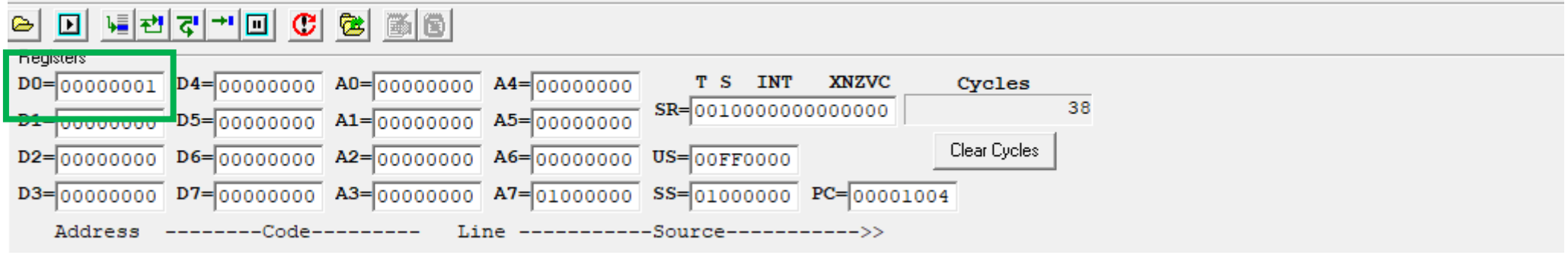
Décrire le programme et applique le fonctionnement en mode pas à pas.

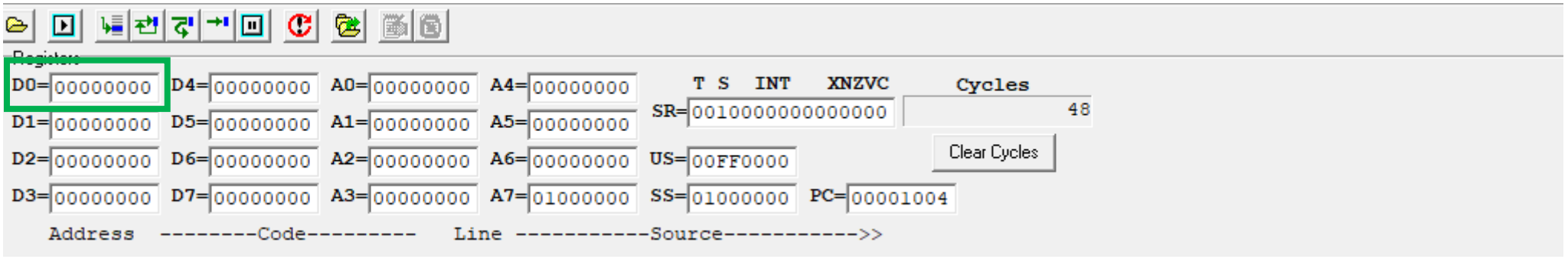
TD/Cours Architecture des ordinateurs	BACHELOR EN CYBERSECURITE	PRENOM : NOM :

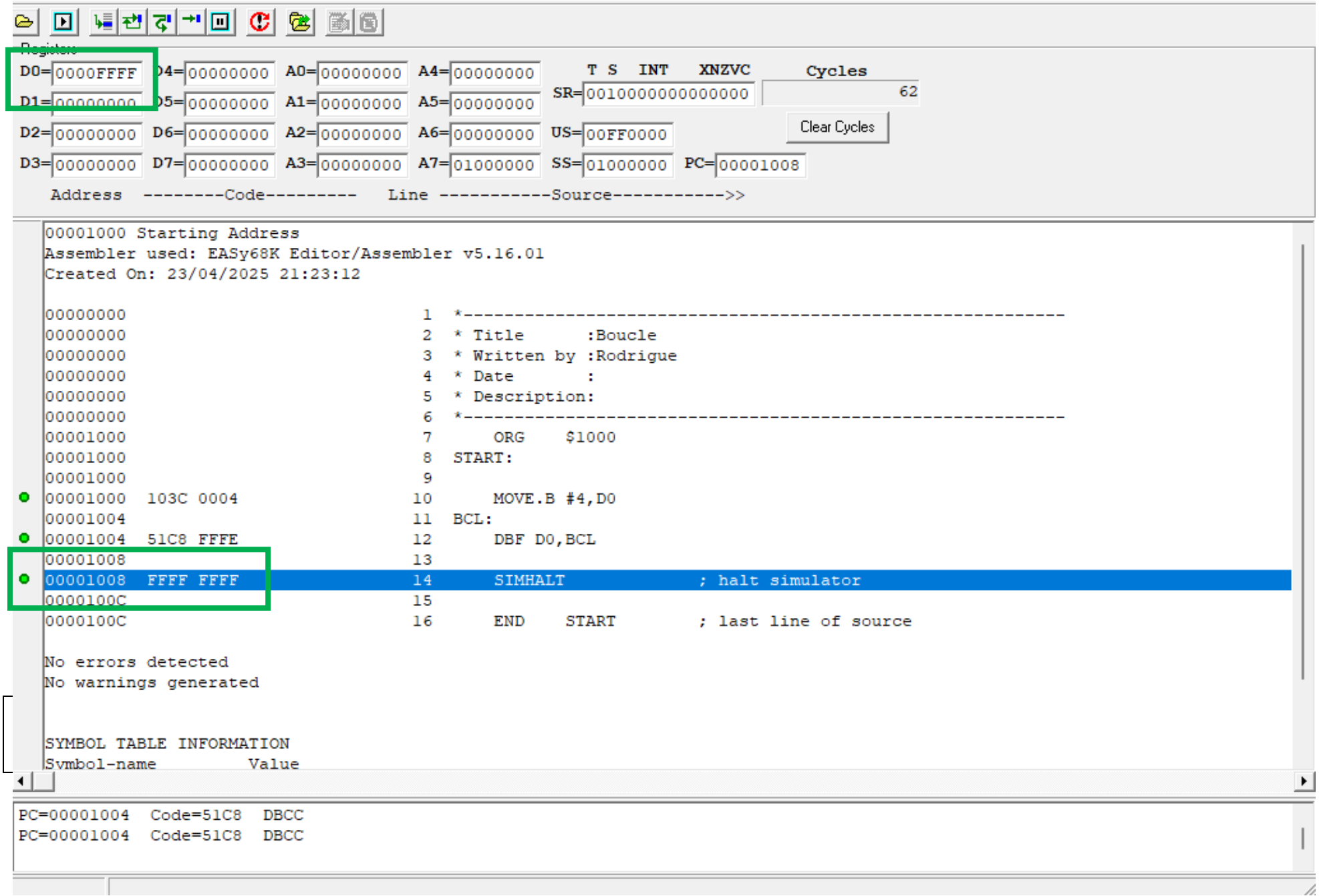


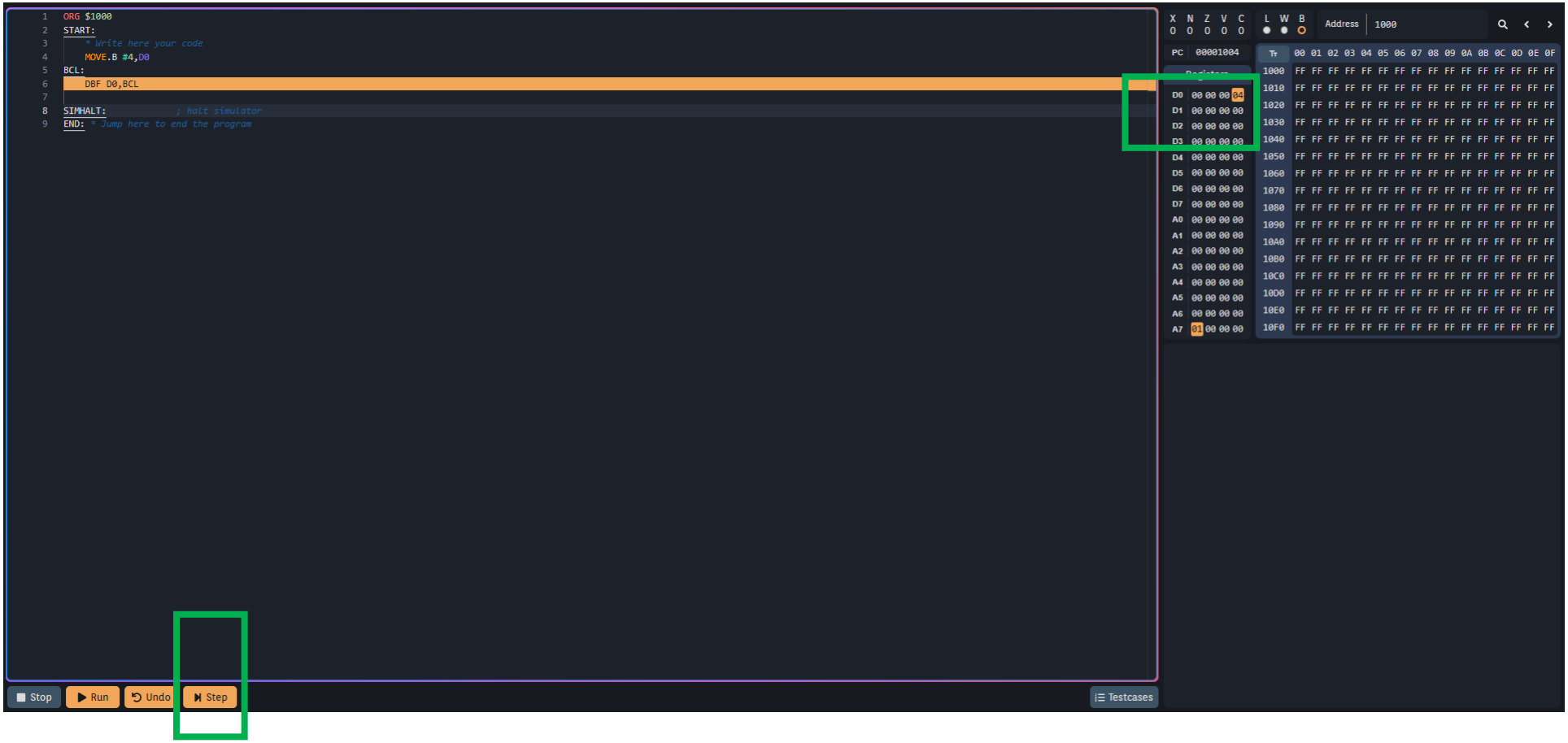

Registers
D0= 00000003 D4= 00000000 A0= 00000000 A4= 00000000
D1= 00000000 D5= 00000000 A1= 00000000 A5= 00000000
D2= 00000000 D6= 00000000 A2= 00000000 A6= 00000000
D3= 00000000 D7= 00000000 A3= 00000000 A7= 01000000
T S INT XNZVC
SR= 0010000000000000
US= 00FF0000
SS= 01000000
PC= 00001004
Cycles
18
Clear Cycles
Address --------Code--------- Line -----------Source----------->>

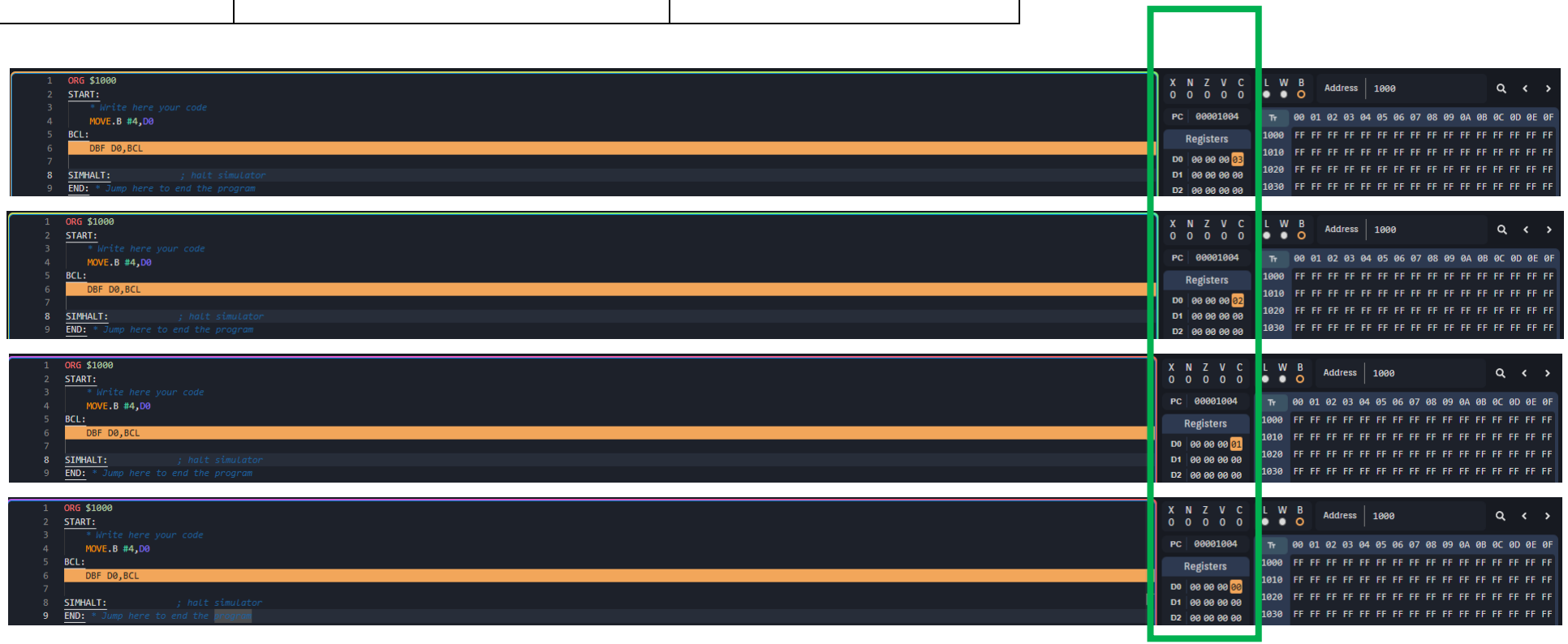

Registers
D0= 00000002 D4= 00000000 A0= 00000000 A4= 00000000
D1= 00000000 D5= 00000000 A1= 00000000 A5= 00000000
D2= 00000000 D6= 00000000 A2= 00000000 A6= 00000000
D3= 00000000 D7= 00000000 A3= 00000000 A7= 01000000
T S INT XNZVC
SR= 0010000000000000
US= 00FF0000
SS= 01000000
PC= 00001004
Cycles
28
Clear Cycles
Address --------Code--------- Line -----------Source----------->>


Registers
D0= 00000001 D4= 00000000 A0= 00000000 A4= 00000000
D1= 00000000 D5= 00000000 A1= 00000000 A5= 00000000
D2= 00000000 D6= 00000000 A2= 00000000 A6= 00000000
D3= 00000000 D7= 00000000 A3= 00000000 A7= 01000000
T S INT XNZVC
SR= 0010000000000000
US= 00FF0000
SS= 01000000
PC= 00001004
Cycles
38
Clear Cycles
Address --------Code--------- Line -----------Source----------->>


Registers
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
T S INT XNZVC
SR= 0010000000000000
US= 00FF0000
SS= 01000000
Cycles
48
Clear Cycles
PC= 00001004
Address --------Code--------- Line -----------Source----------->>


Registers
D0=0000FFFF D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC
SR=0010000000000000
Cycles
62
Clear Cycles
US=00FF0000
SS=01000000
PC=00001008
Address --------Code--------- Line -----------Source----------->>
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 23/04/2025 21:23:12
00000000 1 *---
00000000 2 * Title :Boucle
00000000 3 * Written by :Rodrigue
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *---
00001000 7 ORG $1000
00001000 8 START:
00001000 9
00001000 103C 0004 10 MOVE.B #4,D0
00001004 11 BCL:
00001004 51C8 FFFE 12 DBF D0,BCL
00001008 13
00001008 FFFF FFFF 14 SIMHALT ; halt simulator
0000100C 15
0000100C 16 END START ; last line of source
No errors detected
No warnings generated
SYMBOL TABLE INFORMATION
Symbol-name Value
PC=00001004 Code=51C8 DBCC
PC=00001004 Code=51C8 DBCC


ORG $1000
START:
* Write here your code
MOVE.B #4,D0
BCL:
DBF D0,BCL
SIMHALT: ; halt simulator
END: * Jump here to end the program
X N Z V C
0 0 0 0 0
L W B
Address 1000
PC 00001004
D0 00 00 00 04
D1 00 00 00 00
D2 00 00 00 00
D3 00 00 00 00
D4 00 00 00 00
D5 00 00 00 00
D6 00 00 00 00
D7 00 00 00 00
A0 00 00 00 00
A1 00 00 00 00
A2 00 00 00 00
A3 00 00 00 00
A4 00 00 00 00
A5 00 00 00 00
A6 00 00 00 00
A7 01 00 00 00
Tr 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
1000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1080 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1090 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10A0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10B0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10D0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10E0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
10F0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
Stop
Run
Undo
Step
Testcases


ORG $1000
START:
* Write here your code
MOVE.B #4,D0
BCL:
DBF D0,BCL
SIMHALT: ; halt simulator
END: * Jump here to end the program
X N Z V C
0 0 0 0 0
PC 00001004
Registers
D0 00 00 00 03
D1 00 00 00 00
D2 00 00 00 00
Address 1000
ORG $1000
START:
* Write here your code
MOVE.B #4,D0
BCL:
DBF D0,BCL
SIMHALT: ; halt simulator
END: * Jump here to end the program
X N Z V C
0 0 0 0 0
PC 00001004
Registers
D0 00 00 00 02
D1 00 00 00 00
D2 00 00 00 00
Address 1000
ORG $1000
START:
* Write here your code
MOVE.B #4,D0
BCL:
DBF D0,BCL
SIMHALT: ; halt simulator
END: * Jump here to end the program
X N Z V C
0 0 0 0 0
PC 00001004
Registers
D0 00 00 00 01
D1 00 00 00 00
D2 00 00 00 00
Address 1000
ORG $1000
START:
* Write here your code
MOVE.B #4,D0
BCL:
DBF D0,BCL
SIMHALT: ; halt simulator
END: * Jump here to end the program
X N Z V C
0 0 0 0 0
PC 00001004
Registers
D0 00 00 00 00
D1 00 00 00 00
D2 00 00 00 00
Address 1000

Les captures des écrans sont explicites sur le fonctionnement de cette boucle.