

# TRAVAUX DIRIGES

## ASSEMBLEUR



**Architecture des  
ordinateurs**  
**(Bachelor en cybersécurité)**



Formateurs Jean-Michel BUSCA & Rodrigue MALEOMBHO

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

## 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 connaître 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>

	Formateurs Jean-Michel BUSCA & Rodrigue MALEOMBHO
---	---

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

## 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

```

HelloWorld.X68
*-----*
* Title      :Hello World
* Written by :Rodrigue
* Date       :
* Description:
*-----*
        ORG      $1000
START
    LEA MESSAGE, A1 ; chargement du message dans le registre
    MOVE.B   #14, D0 ;
    TRAP     #15

MESSAGE DC.B    'HELLO Rodriguez !',0
        SIMHALT          ; halt simulator

* Put variables and constants here

END      START          ; last line of source

```

2: 26 Modified Insert

The screenshot shows the EASy68K Editor/Assembler interface. The assembly code is as follows:

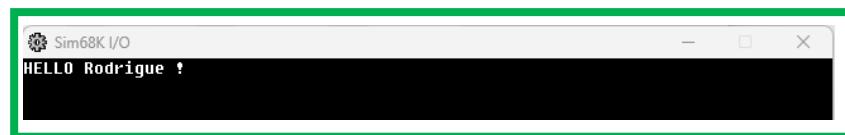
```

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 LER 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



The screenshot shows the 68000 Memory window. The assembly code from the previous screenshot is dumped into memory starting at address 00001000. The memory dump shows the bytes corresponding to the assembly instructions.

Address	From: \$00000000 To: \$00000000 Bytes: \$00000000	Copy	Fill
00000FDO	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 0123456789ABCDEF		
00000FD0:	FF -----		
00000FE0:	FF -----		
00000FF0:	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 FF O Rodrigue !----		
00001020:	FF -----		
00001030:	FF -----		
00001040:	FF -----		
00001050:	FF -----		
00001060:	FF -----		
00001070:	FF -----		

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

The screenshot shows an assembly editor interface with the following details:

- Code Area:** Displays the assembly code:

```
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
```
- Registers Area:** Shows the state of registers from PC to A7.
- Memory Dump Area:** Shows memory starting at address 1000, with the value 'HELLO Rodrigue !' highlighted in green at address 1000.



Formateurs Jean-Michel BUSCA & Rodrigue MALEOMBHO

The screenshot shows a debugger interface with the following sections:

- Assembly View:** Displays the assembly code for the program. The code includes:
  - ORG \$1000
  - START:
  - \* Write here your code
  - LEA MESSAGE, A1 ; chargement du message dans le registre A1
  - MOVE.B #14, D0 ;
  - TRAP #15
  - MESSAGE: DC.B 'HELLO Rodrigue !',0
  - SIMHALT; ; halt simulator
  - END: \* Jump here to end the program
- Registers View:** Shows the state of various registers. The PC register is at 0000100c. The Tr register has values 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F. Other registers (D0-D7, A0-A7) are all 00 00 00 00.
- Memory Dump View:** Shows the memory dump starting at address 1000. The memory contains the string "Hello Rodrigue !".

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

**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  $45_{10} = 2D_{16}$

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

0000010000	103C	002D
00001004	0600	0055

	Formateurs Jean-Michel BUSCA & Rodrigue MALEOMBHO
---	---

File Search Run View Options Help

**Registers**

D0=00000000	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	0
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=00FF0000	<input type="button" value="Clear Cycles"/>
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=00001000

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

```

00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 26/04/2025 16:57:50

00000000          1  *-----
00000000          2  * Title      :Additionl
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 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 END      START           ; last line of source

No errors detected
No warnings generated

SYMBOL TABLE INFORMATION
Symbol-name      Value
-----
START           1000

```

PC=00001008 Code=4E4F TRAP  
TRAP exception occurred at location 1008. Execution halted

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

```

The screenshot shows the EA68K Editor/Assembler interface. The top section displays the Registers window with various memory locations (D0-D3, A0-A7, SR, US, SS, PC) and their current values. The bottom section shows the assembly code listing, symbol table, and memory dump.

**Registers:**

D0=0000002D	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	8
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=00FF0000	<input type="button" value="Clear Cycles"/>
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=00001004

**Assembly Code Listing:**

```

00001000 Starting Address
Assembler used: EA68K Editor/Assembler v5.16.01
Created On: 26/04/2025 16:57:50

00000000          1 *-----
00000000          2 * Title      :Additional
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 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 END     START;   ; last line of source

No errors detected
No warnings generated

```

**Symbol Table Information:**

Symbol-name	Value
START	1000

**Memory Dump:**

```

PC=00001000 Code=103C MOVEB
PC=00001000 Code=103C MOVEB

```

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

The screenshot shows the EA5y68K Editor/Assembler interface. The top section displays assembly code with some lines highlighted in green. The bottom section shows the register state.

**Registers:**

D0=00000082	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000001010	16
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=0OFF0000	
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=00001008

**Assembly Code:**

```

00001000 Starting Address
Assembler used: EA5y68K Editor/Assembler v5.16.01
Created On: 26/04/2025 16:57:50

00000000          1  *-
00000000          2  * Title      :Additionl
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 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 END      START      ; last line of source

No errors detected
No warnings generated

SYMBOL TABLE INFORMATION
Symbol-name      Value
-----
START           1000

```

**Registers (bottom):**

PC=00001000	Code=103C	MOVEB
PC=00001004	Code=0600	ADDI

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

```

Registers
D0=00000082 D4=00000000 A0=00000000 A4=00000000 T S INT XNZVC Cycles
D1=00000000 D5=00000000 A1=00000000 A5=00000000 SR=0010000000001010 16
D2=00000000 D6=00000000 A2=00000000 A6=00000000 US=00FF0000 Clear Cycles
D3=00000000 D7=00000000 A3=00000000 A7=01000000 SS=01000000 PC=0000100A

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

00001000 Starting Address
Assembler used: EA68K Editor/Assembler v5.16.01
Created On: 26/04/2025 16:57:50

00000000          1 *-----
00000000          2 * Title      :Addition
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 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 END START; last line of source

No errors detected
No warnings generated

SYMBOL TABLE INFORMATION
Symbol-name      Value
-----
START           1000

PC=00001008 Code=4E4F TRAP
TRAP exception occurred at location 1008. Execution halted

```

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.

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

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>

The screenshot shows the EASY68K assembly editor interface. The code area contains the following assembly code:

```
1 ORG $1000
2 START:
3     * Write here your code
4     MOVE.B #45,D0
5     ADD.B #85,D0
6     TRAP #15
7 END: * Jump here to end the program
```

The "Step" button in the toolbar at the bottom is highlighted with a green box.

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

```
1 ORG $1000
2 START:
3     * Write here your code
4     HLT
5 ADD.B #85,00
6 TRAP #15
7 END: * Jump here to end the program
```

X	N	Z	V	C	L	W	B
PC	00001008				●	●	○
	Registers				Tr	00	E
D0	00	00	00	00	1910	FF	F
D1	00	00	00	00	1920	FF	F
D2	00	00	00	00	1930	FF	F
D3	00	00	00	00	1940	FF	F
D4	00	00	00	00	1950	FF	F
D5	00	00	00	00	1960	FF	F
D6	00	00	00	00	1970	FF	F
D7	00	00	00	00	1980	FF	F
A0	00	00	00	00	1990	FF	F
A1	00	00	00	00	19A0	FF	F
A2	00	00	00	00	19B0	FF	F
A3	00	00	00	00	19C0	FF	F
A4	00	00	00	00	19D0	FF	F
A5	00	00	00	00	19E0	FF	F
A6	00	00	00	00	19F0	FF	F
A7	01	00	00	00			



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

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

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

X	N	Z	V	C
0	1	0	1	0
PC   00001008	Registers			
DO   00 00 00 82				

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.



Formateurs Jean-Michel BUSCA & Rodrigue MALEOMBHO

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

The screenshot shows the EASY68K software interface. At the top, there's a menu bar with File, Search, Run, View, Options, Help. Below the menu is a toolbar with various icons. The main window is divided into several sections:

- Registers:** Shows registers D0-D7, A0-A7, T, S, INT, XNZVC, and Cycles (set to 16). The D1 register is highlighted with a green box.
- Code Window:** Displays assembly code:

```
00001000 Starting Address
Assembler used: EASY68K Editor/Assembler v5.16.01
Created On: 27/04/2025 17:20:32

00000000      1  -----
00000000      2  * Title      :Additionl
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 303C 0159      9  MOVE.W #345,D0
● 00001004 0640 075D      10 ADD.W #1885,D0
00001008      11 END      START          ; last line of source

No errors detected
No warnings generated

SYMBOL TABLE INFORMATION
Symbol-name      Value
-----
START           1000
```
- Message Window:** Shows the message: "S0 = 68KPROG 20CREATED BY EASY68K .S68 file read successful".

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

```

X	N	Z	V	C	L	W	B	Address	1000
0	0	0	0	0	●	○	●		
PC	00001008	Tr	00	01	02	03	04	05	06
Registers		1000	FF	FF	FF	FF	FF	FF	FF
D0	0000 08b6	1010	FF	FF	FF	FF	FF	FF	FF
D1	0000 0000	1020	FF	FF	FF	FF	FF	FF	FF

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				



Formateurs Jean-Michel BUSCA & Rodrigue MALEOMBHO

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

The screenshot shows the EASy68K Editor/Assembler interface. The top section displays the CPU registers:

Registers	D0=00000021	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1	00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	16
D2	00000000	D6=00000000	A2=00000000	A6=00000000	US=0OFF0000	<input type="button" value="Clear Cycles"/>
D3	00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=0000100C

The bottom section shows the assembly code:

```
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 27/04/2025 17:30:12

00000000          1  *-
00000000          2  * Title      :Additionl
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 103C 002D  9  MOVE.B #45,D0
00001004 0400 000C  10 SUB.B #12,D0
00001008          11 END      START      ; last line of source
```

The line `MOVE.B #45,D0` is highlighted with a blue background.

```

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

```

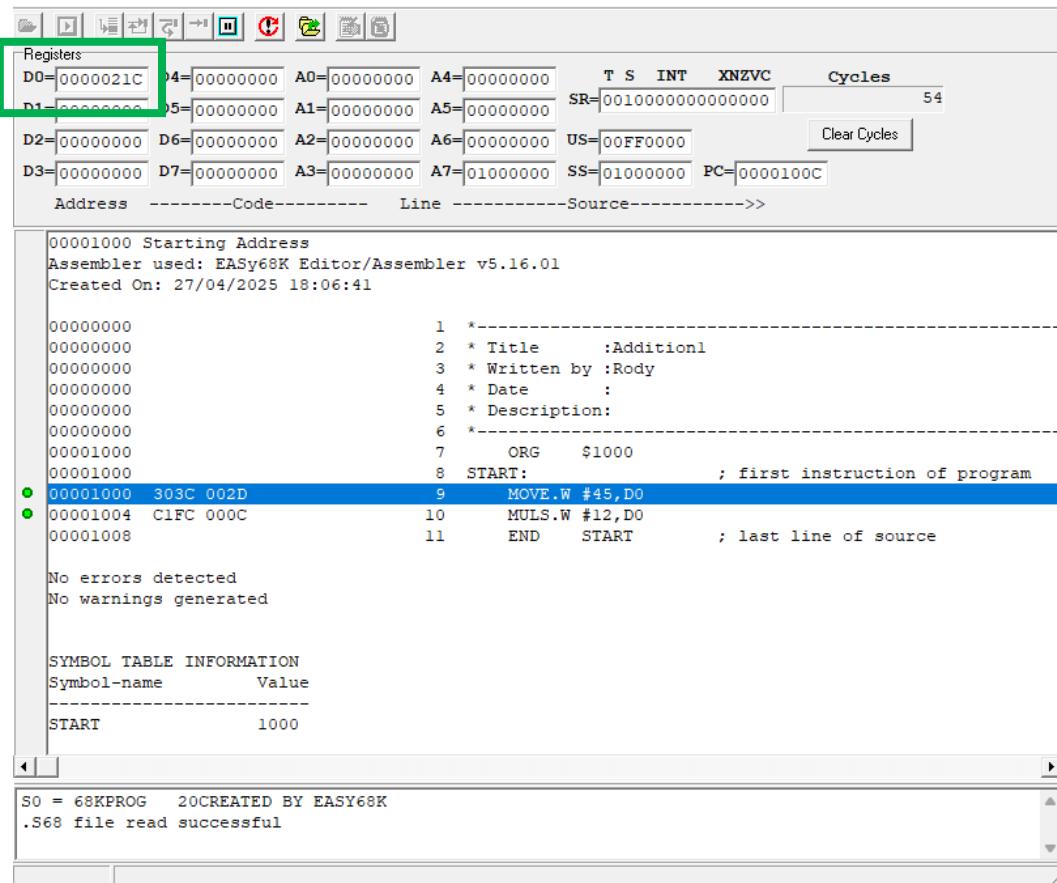
X	N	Z	V	C	L	W	B	Address	1000	
0	0	0	0	0	●	●	○			
PC	00001008	T	00	01	02	03	04	05	06	07
Registers		1000	FF	FF	FF	FF	FF	FF	FF	FF
D0	00 00 00 21	1010	FF	FF	FF	FF	FF	FF	FF	FF
		1020	FF	FF	FF	FF	FF	FF	FF	FF

### 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	
Hexa.	$2^3$	$2^2$	$2^1$	$2^0$	$2^3$	$2^2$	$2^1$	$2^0$	33
	2				1				

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

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



The screenshot shows the EASY68K Editor/Assembler interface. The top section displays the CPU registers:

D0=0000021C	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	54
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=00FF0000	
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=0000100C

Below the registers is a status bar with fields for Address, Code, Line, Source, and a Clear Cycles button.

The main window shows the assembly code:

```
00001000 Starting Address
Assembler used: EASY68K Editor/Assembler v5.16.01
Created On: 27/04/2025 18:06:41

00000000      1  *
00000000      2  * Title      :Additionl
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  303C 002D      9  MOVE.W #45,D0
● 00001004  C1FC 000C      10  MULS.W #12,D0
00001008          11  END     START      ; last line of source

No errors detected
No warnings generated

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

The assembly code includes comments and labels. Instructions 9 and 10 are highlighted with blue bars. The bottom status bar indicates: S0 = 68KPROG 20CREATED BY EASY68K .S68 file read successful.

```

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

```

The screenshot shows a debugger interface for a Motorola 68000 processor. At the top, there are status indicators for X, N, Z, V, C, L, W, and B, followed by an Address field set to 1000. Below this is a PC register showing 00001008. A memory dump window shows bytes at address 1000 and 1010. The Registers window is highlighted with a green border and shows the following values:

Register	Value
D0	0000 021c

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  $45 * 12$ .

File Search Run View Options Help

Registers

D0=0000021C	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	54
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=00FF0000	<input type="button" value="Clear Cycles"/>
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=0000100C

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

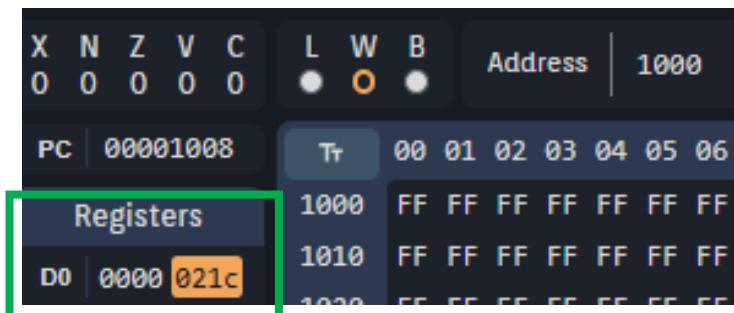
```
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 27/04/2025 18:34:48

00000000          1  *
00000000          2  * Title      :Additionl
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 303C 002D          9  MOVE.W #45,D0
● 00001004 C0FC 000C          10 MULU.W #12,D0
00001008          11 END      START        ; last line of source

No errors detected
No warnings generated

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

```
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.

File Search Run View Options Help

Registers

D0=FFFFFDE4	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000001000	54
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=0OFF0000	
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=0000100C

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

```
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 27/04/2025 18:47:18

00000000          1  *-----
00000000          2  * Title      :Additionl
00000000          3  * Written by :Rody
00000000          4  * Date       :
00000000          5  * Description:
00000000          6  *-
00000100          7  ORG    $1000
00000100          8  START:           ; first instruction of program
● 00001000  303C FFD3          9  MOVE.W # -45,D0
● 00001004  C1FC 000C          10  MULS.W # 12,D0
00001008          11  END    START      ; last line of source

No errors detected
No warnings generated

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

```

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

```

X	N	Z	V	C	L	W	B	Address	1000
0	1	0	0	0	●	○	●	PC	00001008
								T <sub>r</sub>	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
Registers								D0	ffff fde4
								D1	0000 0000

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				Page 25

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

File Search Run View Options Help

Registers

D0=0000001B	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	32
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=00FF0000	
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=00001012

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

```
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 27/04/2025 19:11:47

00000000          1 *-----
00000000          2 * Title      :Additionl
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 303C 002D  9 MOVE.W #45,D0
00001004 0640 000C 10 ADD.W #12,D0
00001008 0640 FF00 11 ADD.W #-48,D0
0000100C 0640 0012 12 ADD.W #18,D0
00001010 4E40      13 TRAP #0
00001012          14 END      START    ; last line of source

No errors detected
No warnings generated

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

```
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
```

X	N	Z	V	C	L	W	B	Address	1000
0	0	0	0	0	●	○	●		
PC	00001010	Tr	00	01	02	03	04	05	06
		1000	FF	FF	FF	FF	FF	FF	FF
Registers	1010	FF	FF	FF	FF	FF	FF	FF	FF
D0	0000 001b	1020	FF	FF	FF	FF	FF	FF	FF

Capture des registres et expliquer

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

## Une autre façon de faire

File Search Run View Options Help

Registers

D0=0000001B	D4=00000000	A0=00001016	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	40
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=00FF0000	<input type="button" value="Clear Cycles"/>
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=0000100E

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

```
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 27/04/2025 22:37:02

00000000          1 *-----
00000000          2 * Title      :Additionl
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 41FA 000C          9 LEA TAB(PC),A0 A0 pointe à la première donnée
00001004 3018             10 MOVE.W (A0)+,D0 D0 contient 45
00001006 D058             11 ADD.W (A0)+,D0 D0 contient 57
00001008 D058             12 ADD.W (A0)+,D0 D0 contient 9
0000100A D058             13 ADD.W (A0)+,D0 D0 contient 27
0000100C 4E40             14 TRAP #0
0000100E= 002D 000C FF00 0012 15 TAB: DC.W 45,12,-48,18
00001016          16 END     START      ; last line of source

No errors detected
No warnings generated
```

```
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	L	W	B	Address	1000
0	0	0	0	0	●	●	○		
PC	00001014								
Registers									
D0	00 00 00 1b								
1000	FF	FF	FF	FF	FF	FF	FF	FF	FF
1010	FF	FF	FF	FF	FF	FF	FF	FF	FF
1020	FF	FF	FF	FF	FF	FF	FF	FF	FF

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 !

File Search Run View Options Help

Registers

D0=000000DD	D4=00000000	A0=00001026	A4=00000000	T S INT XNZVC	Cycles
D1=0000FFFF	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	190
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=00FF0000	<input type="button" value="Clear Cycles"/>
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=00001012

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

```

00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 27/04/2025 22:58:32

00000000          1 *-----
00000000          2 * Title      :Autre
00000000          3 * Written by :Rody
00000000          4 * Date        :
00000000          5 * Description:
00000000          6 *-----
000001000         7 ORG      $1000
000001000         8 START;           ; first instruction of program
000001000 41FA 0010 9 LEA TAB(PC),A0 A0 pointe à la première donnée
000001004 3018 10 MOVE.W (A0)+,D0 D0 contient 45
000001006          11 * Il faut répéter «ADD.W» 9 fois
000001006 323C 0008 12 MOVE.W #8,D1 compteur initialisé à 9-1
00000100A D058 13 BCL ADD.W (A0)+,D0 Accumulation des données
00000100C 51C9 FFFC 14 DBRA D1,BCL
000001010 4E40 15 TRAP #0
000001012= 002D 000C FF00 0... 16 TAB: DC.W 45,12,-48,18,-5,81,12,35,-18,89
000001026 4E4F 17 TRAP #15
000001028          18 END     START      ; last line of source

No errors detected
No warnings generated

```

```
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
```

```
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	L	W	B	Address	1000	Q	<	>
0	0	0	0	0	●	○	●					
PC	00001000	Tr	00	01	02	03	04	05	06	07	08	09
			0A	0B	0C	0D	0E	0F				
		Registers	1000	FF	FF	FF	FF	FF	FF	FF	FF	FF
			1010	FF	FF	FF	FF	FF	FF	FF	0	2D
				0	0	C	0	23	FF	EE	0	59
		D0	0000 0000	1020	FF	FB	0	51	0	C	0	23
		D1	0000 0000		FF	EE	0	59	FF	FF	FF	FF
		D2	0000 0000	1030	FF	FF	FF	FF	FF	FF	FF	FF

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

```
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
```

```
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
```



Formateurs Jean-Michel BUSCA & Rodrigue MALEOMBHO

```

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	L	W	B	Address	1000			
0	0	0	0	0	●	○	●			Q	<	>
PC	0000100c	Tr	00	01	02	03	04	05	06	07	08	09
D0	0000 002d	1000	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D1	0000 0008	1010	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D2	0000 0000	1020	FF	FB	0	51	0	C	0	23	FF	EE
D3	0000 0000	1030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D4	0000 0000	1040	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D5	0000 0000	1050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D6	0000 0000	1060	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D7	0000 0000	1070	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A0	0000 101a	1080	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A1	0000 0000	1090	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A2	0000 0000	10A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A3	0000 0000	10B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A4	0000 0000	10C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A5	0000 0000	10D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A6	0000 0000	10E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A7	0100 0000	10F0	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	L	W	B	Address	1000			
0	0	0	0	0	●	○	●			Q	<	>
PC	00001010	Tr	00	01	02	03	04	05	06	07	08	09
D0	0000 0039	1000	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D1	0000 0008	1010	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D2	0000 0000	1020	FF	FB	0	51	0	C	0	23	FF	EE
D3	0000 0000	1030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D4	0000 0000	1040	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D5	0000 0000	1050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D6	0000 0000	1060	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
D7	0000 0000	1070	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A0	0000 101c	1080	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A1	0000 0000	1090	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A2	0000 0000	10A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A3	0000 0000	10B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A4	0000 0000	10C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A5	0000 0000	10D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A6	0000 0000	10E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
A7	0100 0000	10F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF



The screenshot shows a debugger interface with two main panes. The left pane displays assembly code with comments in French. The right pane shows a memory dump from address \$1000 to \$10F0.

```

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

```

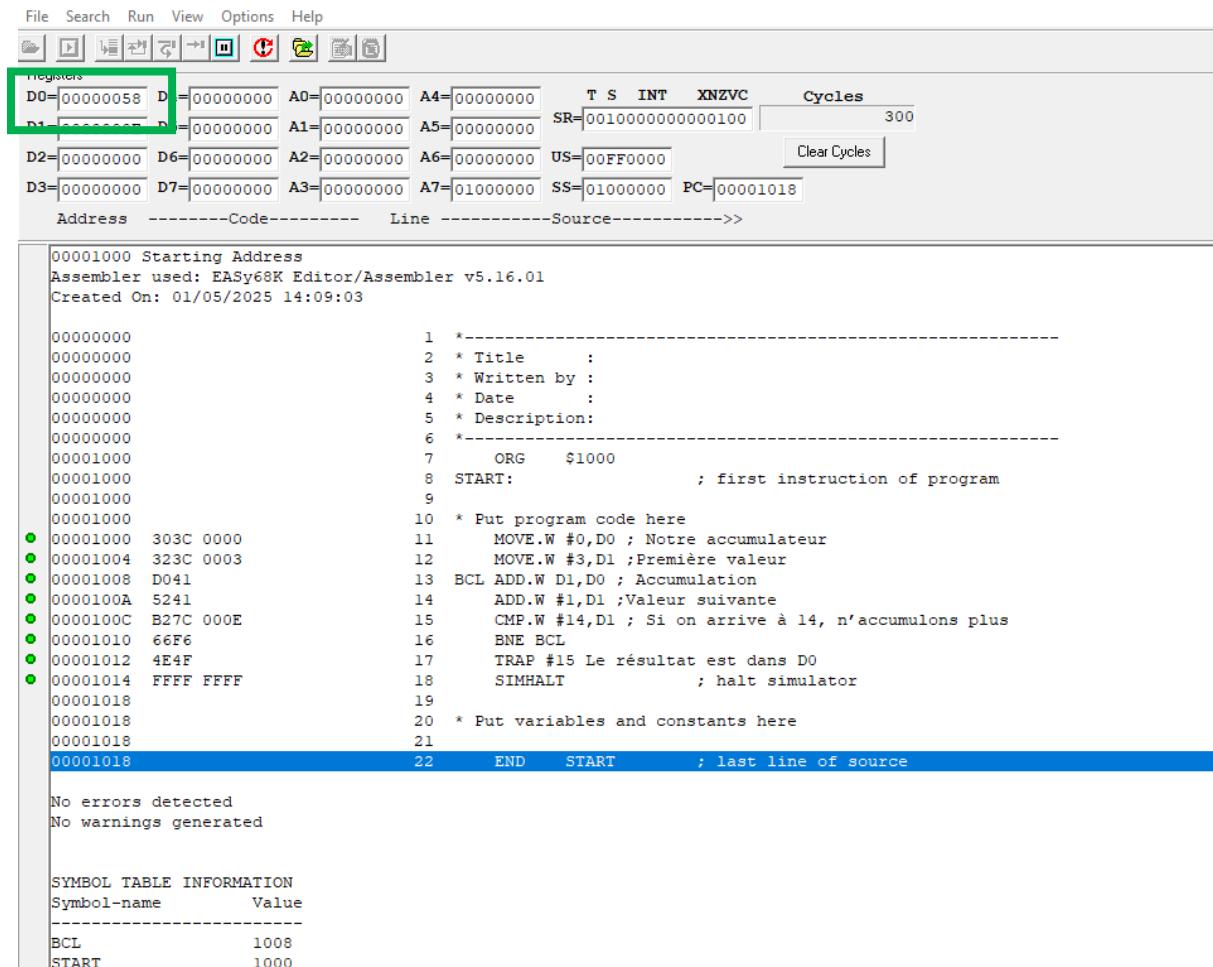
**Registers:**

	X	N	Z	V	C	L	W	B	Address	1000
PC	00001014								T <sub>r</sub>	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
D0	0000 00dd					1000	FF			FF
D1	0000 ffff					1010	FF 0 2D 0 C FF D0 0 12			FF
D2	0000 0000					1020	FB 0 51 0 C 0 23 FF EE 0 59 FF FF FF FF FF FF FF			FF
D3	0000 0000					1030	FF			FF
D4	0000 0000					1040	FF			FF
D5	0000 0000					1050	FF			FF
D6	0000 0000					1060	FF			FF
D7	0000 0000					1070	FF			FF
A0	0000 102c					1080	FF			FF
A1	0000 0000					1090	FF			FF
A2	0000 0000					10A0	FF			FF
A3	0000 0000					10B0	FF			FF
A4	0000 0000					10C0	FF			FF
A5	0000 0000					10D0	FF			FF
A6	0000 0000					10E0	FF			FF
A7	0100 0000					10F0	FF			FF

**Unknown interrupt: 221**

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

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



The screenshot shows the EASy68K Editor/Assembler interface. The top part displays the CPU registers (D0-D3, A0-A7, T, S, INT, XNZVC, SR, PC, Cycles) with values corresponding to the assembly code execution. The bottom part shows the assembly source code:

```

00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 01/05/2025 14:09:03

00000000          1 *-----
00000000          2 * Title   :
00000000          3 * Written by :
00000000          4 * Date    :
00000000          5 * Description:
00000000          6 *-----
00001000          7 ORG     $1000
00001000          8 START:      ; first instruction of program
00001000          9
00001000          10 * Put program code here
00001000 303C 0000 11 MOVE.W #0,D0 ; Notre accumulateur
00001004 323C 0003 12 MOVE.W #3,D1 ;Première valeur
00001008 D041 13 BCL ADD.W D1,D0 ; Accumulation
0000100A 5241 14 ADD.W #1,D1 ;Valeur suivante
0000100C B27C 000E 15 CMP.W #14,D1 ; Si on arrive à 14, n'accumulons plus
00001010 66F6 16 BNE BCL
00001012 4E4F 17 TRAP #15 Le résultat est dans D0
00001014 FFFF FFFF 18 SIMHALT      ; halt simulator
00001018          19
00001018          20 * Put variables and constants here
00001018          21
00001018          22 END      START      ; last line of source

```

No errors detected  
No warnings generated

SYMBOL TABLE INFORMATION

Symbol-name	Value
BCL	1008
START	1000



```
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
```



The screenshot shows a debugger interface with three main panes:

- Left pane (Assembly Code):** Displays the assembly code for the program. Lines 10 and 11 are highlighted in red, while line 12 is highlighted in blue.
- Middle pane (Registers):** Shows the CPU registers. Register D0 contains the value 0000 0058, which is highlighted with a green box.
- Right pane (Memory Dump):** Shows memory starting at address 1000. The first byte of D0 (address 1000) is also highlighted with a green box.

```
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
```

Unknown interrupt: 88

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

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.

File Search Run View Options Help

Registers

D0=00000000	D1=0000FFFF	D2=0000000A	D3=0000000A	A0=00000000	A1=00000000	A2=00000000	A3=00000000	A4=00000000	A5=00000000	A6=00000000	A7=01000004	T=SR=0010000000000000	S=US=0OFF0000	INT=SS=01000004	XNZVC=PC=FFFFFF	Cycles=56
-------------	-------------	-------------	-------------	-------------	-------------	-------------	-------------	-------------	-------------	-------------	-------------	-----------------------	---------------	-----------------	-----------------	-----------

Code Line Source

```

00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 01/05/2025 15:50:45

00000000          1 *-----
00000000          2 * Title      :Compa
00000000          3 * Written by :
00000000          4 * Date       :
00000000          5 * Description:
00000000          6 *-----
00001000          7 ORG     $1000
00001000          8 START:           ; first instruction of program
00001000
00001000          9
00001000          10    * Write here your code
00001000          11    * Write here your code
00001000          12    MOVE.W #1,D1 ; Valeur de D1
00001004          13    MOVE.W #10,D2 ; Valeur de D2
00001008          14    MAX:    MOVE.W D1,D3 ; On suppose que D1 est la max
0000100A          15    CMP.W D2,D1 ; On fait D2-D1 >= 0
0000100C          16    BGE    OK ; Si c'est bien D1 max, on sort du sous programme
00001010          17    MOVE D2,D3 ;Si on est la, c'est D2 la valeur max
00001012          18    OK:    RTS
00001014          19    END    START           ; last line of source

No errors detected
No warnings generated

SYMBOL TABLE INFORMATION
Symbol-name      Value
MAX              1008
OK               1012
START            1000

```



```
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 là, c'est D2 la valeur max
10 OK:   RTS
11 END: * Jump here to end the program
```

X	N	Z	V	C	L	W	B	Address	1000	Q	<	>
0	0	0	0	0	.	O	.					
PC	ffffffffff							00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F				
D0	0000 0000				1000	FF						
D1	0000 ffff				1010	FF						
D2	0000 000a				1020	FF						
D3	0000 000a				1030	FF						
D4	0000 0000				1040	FF						
D5	0000 0000				1050	FF						
D6	0000 0000				1060	FF						
D7	0000 0000				1070	FF						
A0	0000 0000				1080	FF						
A1	0000 0000				1090	FF						
A2	0000 0000				10A0	FF						
A3	0000 0000				10B0	FF						
A4	0000 0000				10C0	FF						
A5	0000 0000				10D0	FF						
A6	0000 0000				10E0	FF						
A7	0100 0004				10F0	FF						

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.

File Search Run View Options Help

Registers

D0=00000003	I4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000001	I5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	50
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=0OFF0000	
D3=00000000	D7=00000000	A3=00000000	A7=01000004	SS=01000004 PC=FFFFFFF	

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

```

00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 01/05/2025 16:41:38

00000000          1  *
00000000          2  * Title      :PAIR / IMPAIR
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
00001000          9
00001000         10  START:
00001000         11  * Write here your code
● 00001000 303C 0003          12  MOVE.W #3, D0 ; Valeur de D0
● 00001004 0800 0000          13  BTST #0, D0 ; on contrôle le bit de poids faible
● 00001008 6700 0008          14  BEQ BITZERO
● 0000100C 323C 0001          15  MOVE.W #1,D1 ;La valeur est testée impaire. On envoie 1 dans D1
● 00001010 4E75              16  RTS
00001012          17  BITZERO:
● 00001012 323C 0000          18  MOVE.W #0,D1 ; la valeur est paire on envoie 0 dans D1
00001016          19
00001016          20  END     START           ; last line of source

No errors detected
No warnings generated

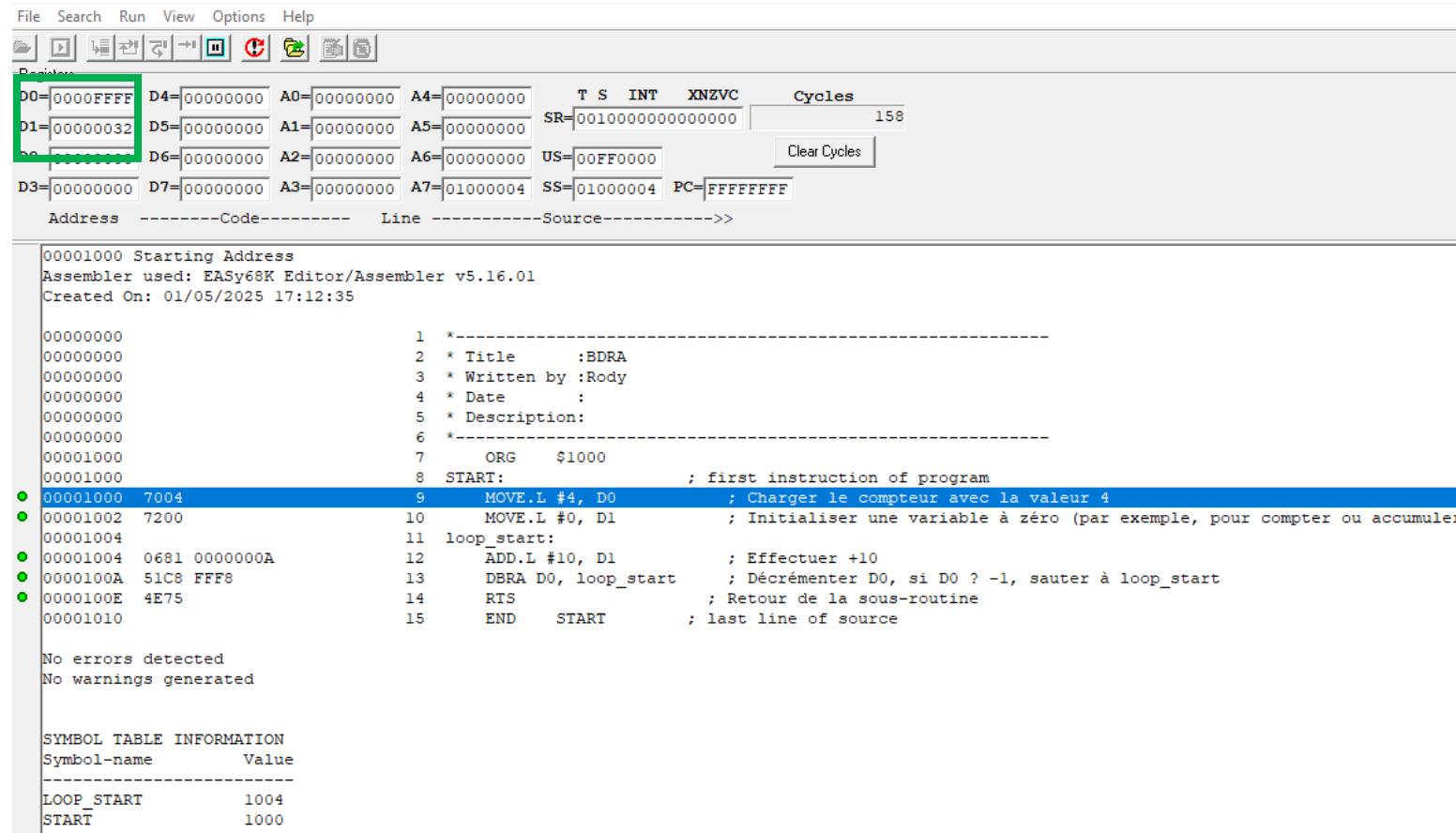
SYMBOL TABLE INFORMATION
Symbol-name      Value
-----
BITZERO          1012
START            1000

```

```
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 valeur 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 !



The screenshot shows the EASy68K Editor/Assembler interface. The assembly code is as follows:

```

00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v5.16.01
Created On: 01/05/2025 17:12:35

00000000          1  *-----
00000000          2  * Title      :BDRA
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 7004      9  MOVE.L #4, D0   ; Charger le compteur avec la valeur 4
00001002 7200      10 MOVE.L #0, D1  ; Initialiser une variable à zéro (par exemple, pour compter ou accumuler)
00001004          11 loop_start:
00001004 0681 0000000A 12 ADD.L #10, D1   ; Effectuer +10
0000100A 51C8 FFF8 13 DBRA D0, loop_start ; Décrémenter D0, si D0 > -1, sauter à loop_start
0000100E 4E75      14 RTS      ; Retour de la sous-routine
00001010          15 END      START     ; last line of source

No errors detected
No warnings generated

SYMBOL TABLE INFORMATION
Symbol-name      Value
-----
LOOP_START        1004
START            1000

```

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

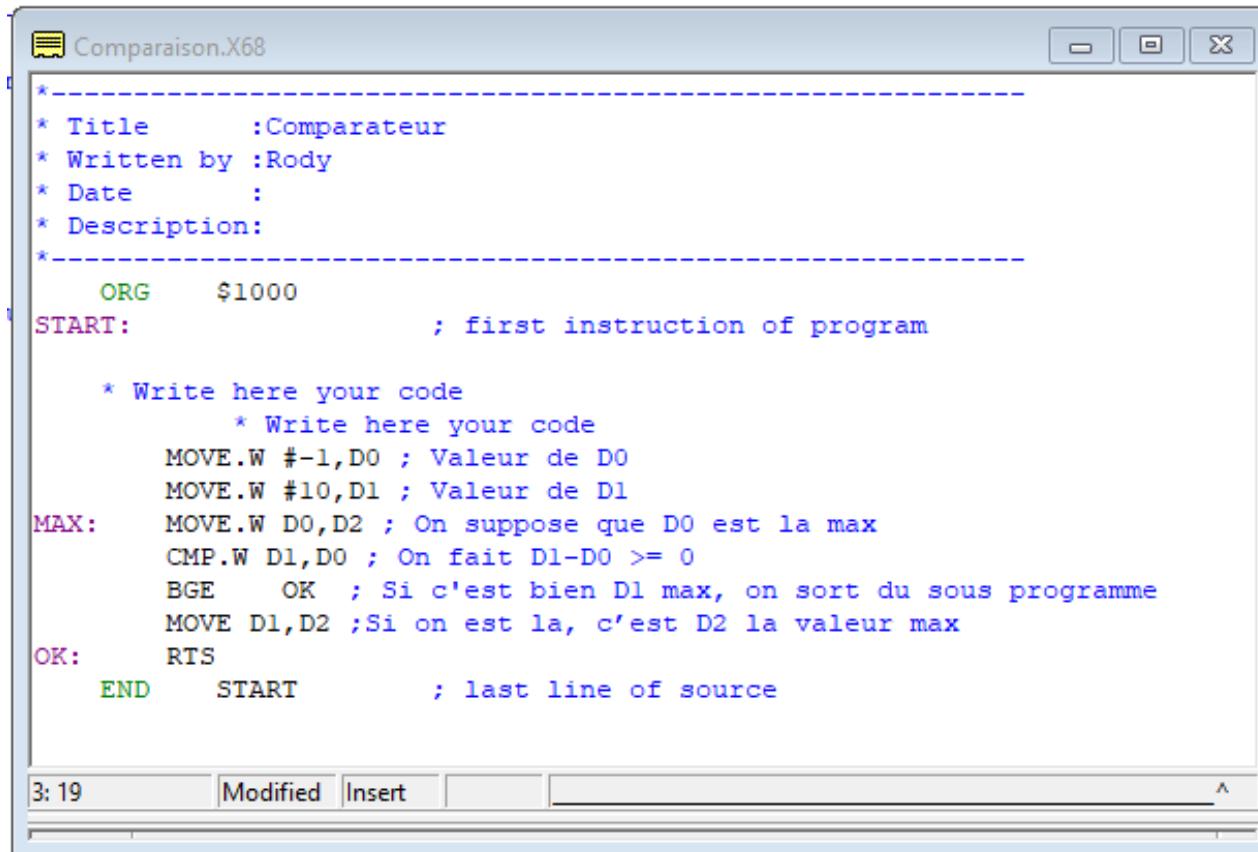
```
1 ORG $1000
2 START:
3     MOVE.L #4, D0          ; Charger le compteur avec la valeur 4
4     MOVE.L #0, D1          ; Initialiser une variable à zéro (par exemple, pour compter ou accumuler)
5 loop_start:
6     ADD.L #10, D1          ; Effectuer +10
7     DBRA D0, loop_start    ; Décrémenter D0, 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 1 avec des noms de registres différents.*

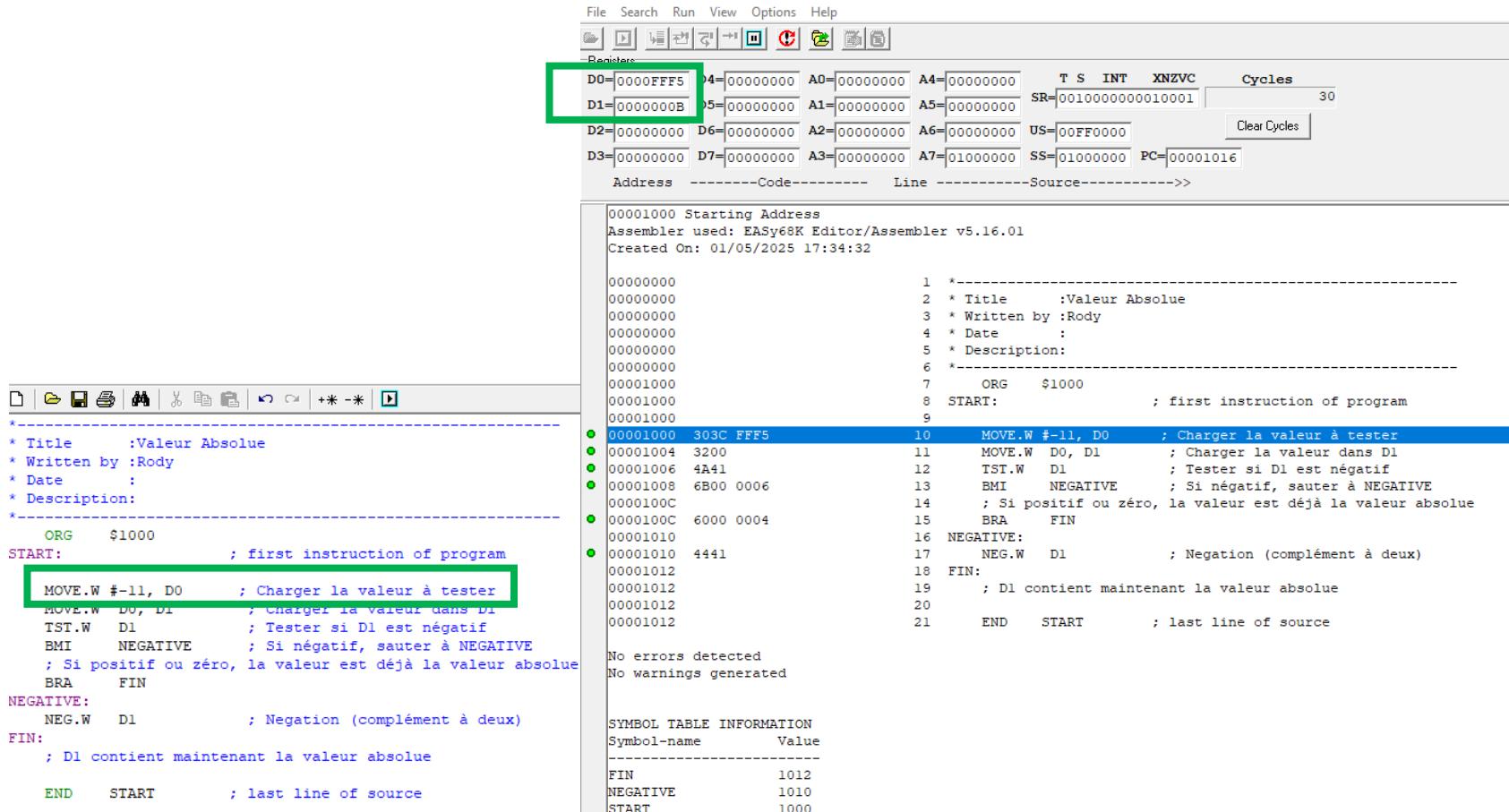


The screenshot shows a Microsoft Notepad window with the title "Comparaison.X68". The code is written in 6809 assembly language:

```
*-----  
* Title      :Comparateur  
* Written by :Rody  
* Date       :  
* Description:  
*-----  
ORG    $1000  
START:           ; first instruction of program  
  
    * Write here your code  
        * Write here your code  
        MOVE.W #-1,D0 ; Valeur de D0  
        MOVE.W #10,D1 ; Valeur de D1  
MAX:   MOVE.W D0,D2 ; On suppose que D0 est la max  
        CMP.W D1,D0 ; On fait D1-D0 >= 0  
        BGE    OK    ; Si c'est bien D1 max, on sort du sous programme  
        MOVE D1,D2 ;Si on est la, c'est D2 la valeur max  
OK:    RTS  
END    START           ; last line of source
```

The status bar at the bottom shows "3: 19" and buttons for "Modified" and "Insert".

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.



The screenshot shows the EA68K Editor/Assembler interface. The assembly code is as follows:

```

*-----*
* 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

```

The Registers window shows:

D0	0000FFFF	D4	00000000	A0	00000000	A4	00000000	T	S	INT	XNZVC	Cycles
D1	0000000B	D5	00000000	A1	00000000	A5	00000000	SR	00100000000010001		30	
D2	00000000	D6	00000000	A2	00000000	A6	00000000	US	00FF0000			
D3	00000000	D7	00000000	A3	00000000	A7	01000000	SS	01000000	PC	00001016	

The Stack window shows:

```

00001000 Starting Address
Assembler used: EA68K 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         10  MOVE.W #11, D0    ; Charger la valeur à tester
00001004         11  MOVE.W D0, D1    ; Charger la valeur dans D1
00001006         12  TST.W  D1    ; Tester si D1 est négatif
00001008         13  BMI     NEGATIVE ; Si négatif, sauter à NEGATIVE
0000100C         14  ; Si positif ou zéro, la valeur est déjà la valeur absolue
0000100C         15  BRA     FIN
00001010         16  NEGATIVE:
00001010         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	L	W	B	Address	1000
PC		Tr							
1	0	0	0	1	.	O	.	1000	
1000	FF	1000							
1010	FF	1010							
1020	FF	1020							
1030	FF	1030							

Registers

D0	0000	ffff5
D1	0000	000b
D2	0000	0000

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

**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
```

	Formateurs Jean-Michel BUSCA & Rodrigue MALEOMBHO
---	---

The screenshot shows a debugger interface with several panes:

- Registers:** Shows registers D0-D3 with values 000000A, 00000002, 00000004, and 00000002 respectively. Below the registers are fields for Address, Code, Line, and Source.
- Memory Dump:** A table showing memory starting at address 00000000. It lists addresses from 00000000 to 00001022, their corresponding memory values, and descriptions of the code.
- Assembly Code:** The main pane displays assembly code with numbered comments. The code initializes variables A, B, and C, compares them, finds the minimum, and prints it. Labels like START, skip\_label, continue\_label, and END are used.
- Symbols:** A table titled "SYMBOL TABLE INFORMATION" showing symbols like CONTINUE\_LABEL and SKIP\_LABEL with their addresses.
- Registers:** A summary table at the bottom showing PC, Code, and Value for registers D0 and D3.

```

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
00001000  00001000 700A      12
00001000  00001002 7202      13 MOVE.L #10, D0    ; Charger A dans D0
00001000  00001004 7404      14 MOVE.L #2, D1    ; Charger B dans D1
00001000  00001006          15 MOVE.L #4, D2    ; Charger C dans D2
00001006
00001006  00001006 B081      16
00001006  00001008 6E00 000C 17 ; Comparaison A et B
0000100C      18 CMP.L D1, D0    ; D0 - D1 (A - B)
0000100C      19 BGT skip_label ; Si A > B, sauter à skip_label
0000100C
0000100C  0000100C B082      20
0000100C  0000100E 6E00 0006 21 ; Comparaison A et C
0000100E      22 CMP.L D2, D0    ; D0 - D2 (A - C)
0000100E      23 BGT skip_label ; Si A > C, sauter à skip_label
00001012      24
00001012  00001012 2600      25 ; Si aucune condition n'est remplie, continuer ici
00001012  00001014 4E75      26 MOVE.L D0,D3    ; C'est A le minimum des valeurs B et C. On affiche dans D3
00001016      27 RTS
00001016
00001016  00001016 B481      28
00001016  00001018 6E00 0006 29 skip_label:
00001018      30   ; Code à exécuter si A > B et A > C
00001018  0000101C 2602      31 CMP.L D1, D2    ; D2 - D1 (B - C). On compare B et C
0000101C      32 BGT continue_label ; Si B > C alors sauter à continue_label
0000101C  0000101E 4E75      33 MOVE.L D2,D3    ; Si on se retrouve sur cette ligne: c'est C le minimum. On affiche dans D3
00001020      34 RTS ; Ou BRA END pour revenir dans le programme principale
00001020      35
00001020  00001020 2601      36 continue_label:
00001020  00001022          37   ; Suite du programme
00001022      38 MOVE.L D1,D3    ; Le minimum est B
00001022      39
00001022  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  MOVE.L

```



```

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	L	W	B	Address
0	0	0	0	0	○	●	●	1000
PC	Tr							1000
00001038	00	01	02	03	04	05	06	
Registers								
D0	0000000a							1000 FF FF FF FF FF FF FF
D1	00000002							1010 FF FF FF FF FF FF FF
D2	00000004							1020 FF FF FF FF FF FF FF
D3	00000002							1030 FF FF FF FF FF FF FF
								1040 FF FF FF FF FF FF FF

TD/Cours Architecture des ordinateurs
--

BACHELOR EN CYBERSECURITE
---------------------------

PRENOM : NOM :
-------------------

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



Formateurs Jean-Michel BUSCA & Rodrigue MALEOMBHO

**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(?)
```



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

```

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

```

File Search Run View Options Help

Registers

D0=0000000E	D4=00000000	A0=FFFFFFF	A4=00000000	T S INT XNZVC	Cycles
D1=FFFFFFFFFF16	D5=00000000	A1=00001050	A5=00000000	SR=0010000000001000	202
D2=00000005	D6=00000000	A2=00000000	A6=00000000	US=0OFF0000	
D3=00000001	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=0000105E

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

```

00000000          1 *-----
00000000          2 * Title      :Equations Second 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          10 MOVE.L #1, D0    ; a
00001000          11 MOVE.L #1, D1    ; b
00001000          12 MOVE.L #5, D2    ; c
00001006          13
00001006          14 ; Calcul du discriminant ? = b^2 - 4ac
00001006          15 MULS D1,D1    ; D1 = b^2
00001008          16 MOVE.L D0,D3    ; D3 = a
0000100A          17 MULS D3,D2    ; D2 = 4ac (multiplier par 4)
0000100C          18 SUB.L D2, D1    ; ? = b^2 - 4ac
0000100E          19 ; Vérification du discriminant
0000100E          20 CMP.L #0, D1
00001014          21 BLT pas_de_solution_reelle
00001018          22
00001018          23 ; Si ? = 0, solution unique
00001018          24 BEQ     solution_unique
0000101C          25 solution_unique:
0000101C          26 NEG.L D1      ;-b
0000101E          27 MOVE.L D1, D6
00001020          28 LSL.L #1, D0    ; 2a
00001022          29 DIVS D0, D6    ; x1
00001024          30 *JSR END
00001024          31 ; Si ? > 0, deux solutions
00001024          32 ; Calcul de sqrt(?)
00001024          33 MOVE.L D1, D4
00001026          34 JSR     sqrt_function ; sqrt_function doit calculer racine carré de D4
0000102C          35 MOVE.L D4, D5    ; racine de ?
0000102E          36
0000102E          37 sqrt_function:
0000102E          38 ; Calcul x1 = (-b - sqrt(?)) / (2a)
0000102E          39 NEG.L D1      ;-b
00001030          40 MOVE.L D1, D6
00001032          41 ADD.L D5, D6    ; (-b - sqrt(?))
00001034          42 LSL.L #1, D0    ; 2a
00001036          43 DIVS D0, D6    ; x1
00001038          44 ; Calcul x2 = (-b + sqrt(?)) / (2a)

```

Illegal instruction found at location 1058. Execution halted

```

1  ORG $1000
2  START:
3  ; Coefficients
4  MOVE.L #1, D0      ; a
5  MOVE.L #1, D1      ; b
6  MOVE.L #5, D2      ; c
7  ; Calcul du discriminant  $\Delta = b^2 - 4ac$ 
8  MULS D1,D1      ; D1 =  $b^2$ 
9  MOVE.L D0,D3      ; D3 = a
10 MULS D3,D2      ; D2 =  $4ac$  (multiplier par 4)
11 SUB.L D2, D1      ;  $\Delta = b^2 - 4ac$ 
12 ; Vérification du discriminant
13 CMP.L #0, D1
14 BLT pas_de_solution_reelle
15 ; Si  $\Delta = 0$ , solution unique
16 BEQ solution_unique
17 solution_unique:
18 NEG.L D1          ; -b
19 MOVE.L D1, D6
20 LSL.L #1, D0      ; 2a
21 DIVS D0, D6      ; x1
22 JMP END
23 ; Si  $\Delta > 0$ , deux solutions
24 ; Calcul de  $\sqrt{\Delta}$ 
25 MOVE.L D1, D4
26 JSR sqrt_function ; sqrt_function doit calculer racine carré de D4
27 MOVE.L D4, D5      ; racine de  $\Delta$ 
28
29 sqrt_function:
30 ; Calcul  $x_1 = (-b - \sqrt{\Delta}) / (2a)$ 
31 NEG.L D1          ; -b
32 MOVE.L D1, D6
33 ADD.L D5, D6      ;  $(-b - \sqrt{\Delta})$ 
34 LSL.L #1, D0      ; 2a
35 DIVS D0, D6      ; x1
36 ; Calcul  $x_2 = (-b + \sqrt{\Delta}) / (2a)$ 
37 NEG.L D1
38 MOVE.L D1, D7
39 SUB.L D5, D7      ;  $(-b + \sqrt{\Delta})$ 
40 LSL.L #1, D0
41 DIVS D0,D7      ; x2
42 JMP END
43 pas_de_solution_reelle:
44 LEA MESSAGE, A1 ; chargement du message dans le registre A1
45 MOVE.B #14,D0 ;
46 MOVE.B D0,D1 ;
47 TRAP #15
48 MESSAGE: DC.B    'PAS DE SOLUTION REELLES !',0
49 FIN:
50 ; Résultats dans D6 et D7

```

X	N	Z	V	C	L	W	B	Address	1000	Q	<	>													
					PC	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F																			
					Registers																				
1	0	0	0	0	1000	0000000e	1010	1020	1030	1040	1050	1060	1070	1080	P	A	S	D	E	S	O	L	U	T	
2	0	●	●	0	A0	00000000	I	O	N	S	R	E	E	L	L	E	S	!	1090	10A0	10B0	10C0	10D0	10E0	10F0
3					A1	00001084																			
4					A2	00000000																			
5					A3	00000000																			
6					A4	00000000																			
7					A5	00000000																			
8					A6	00000000																			
9					A7	01000000																			
10					PAS DE SOLUTION REELLES !																				

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

## 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.



Formateurs Jean-Michel BUSCA & Rodrigue MALEOMBHO

File Search Run View Options Help



Registers

D0=	00000004	D4=	00000000	A0=	00000000	A4=	00000000	T S INT XNZVC	Cycles
D1=	00000000	D5=	00000000	A1=	00000000	A5=	00000000	SR=	0010000000000000 8
D2=	00000000	D6=	00000000	A2=	00000000	A6=	00000000	US=	0OFF0000
D3=	00000000	D7=	00000000	A3=	00000000	A7=	01000000	SS=	01000000 PC=00001004

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

.S68 file read successful  
00000000 00000000 00000000 00000000

Registers

D0=00000003	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	18
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=0OFF0000	<input type="button" value="Clear Cycles"/>
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=00001004

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

Registers

D0=00000002	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	28
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=0OFF0000	<input type="button" value="Clear Cycles"/>
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=00001004

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

Registers

D0=00000001	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	38
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=0OFF0000	<input type="button" value="Clear Cycles"/>
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=00001004

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

Registers

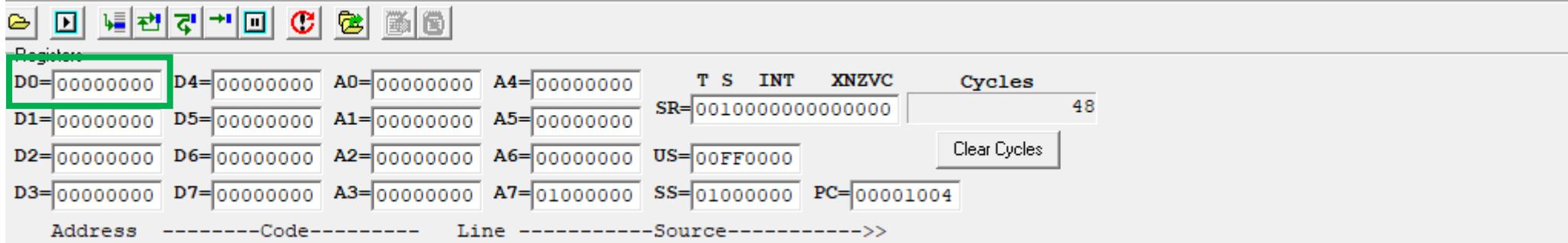
D0=00000001	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	38
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=0OFF0000	<input type="button" value="Clear Cycles"/>
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=00001004

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

Registers

D0=00000000	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	48
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=00FF0000	<input type="button" value="Clear Cycles"/>
D3=00000000	D7=00000000	A3=00000000	A7=01000000	SS=01000000	PC=00001004

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



Registers

D0=0000FFFF	D4=00000000	A0=00000000	A4=00000000	T S INT XNZVC	Cycles
D1=00000000	D5=00000000	A1=00000000	A5=00000000	SR=0010000000000000	62
D2=00000000	D6=00000000	A2=00000000	A6=00000000	US=00FF0000	<input type="button" value="Clear Cycles"/>
D3=00000000	D7=00000000	A3=00000000	A7=01000000	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
00000000
00000000
00000000
00000000
00000000
00001000
00001000
00001000
● 00001000 103C 0004
● 00001004 51C8 FFFE
00001008
● 00001008 FFFF FFFF
0000100C
0000100C

1 *-
2 * Title      :Boucle
3 * Written by :Rodrigue
4 * Date       :
5 * Description:
6 *-
7 ORG     $1000
8 START:
9
10 MOVE.B #4,D0
11 BCL:
12 DBF D0,BCL
13
14 SIMHALT      ; halt simulator
15
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
```

The screenshot shows a debugger interface with the following components:

- Assembly View:** On the left, the assembly code is displayed. It includes labels like `START`, `BCL`, and `SIMHALT`, and instructions such as `MOVE.B #4,D0` and `DBF D0,BCL`.
- Registers View:** A green box highlights the `D0` register, which contains the value `04`.
- Memory Dump View:** The right side shows a memory dump from address `1000` to `10F0`. The `D0` register's value (`04`) is highlighted in orange at address `1004`. The `D0` column is also highlighted in orange.
- Control Buttons:** At the bottom left are buttons for `Stop`, `Run`, `Undo`, and `Step`. At the bottom right is a `Testcases` button.

```
1 ORG $1000
2 START:
3     * Write here your code
4     MOVE.B #4,D0
5 BCL:
6     DBF D0,BCL
7
8 SIMHALT:           ; halt simulator
9 END: * Jump here to end the program
```

```
1 ORG $1000
2 START:
3     * Write here your code
4     MOVE.B #4,D0
5 BCL:
6     DBF D0,BCL
7
8 SIMHALT:           ; halt simulator
9 END: * Jump here to end the program
```

```
1 ORG $1000
2 START:
3     * Write here your code
4     MOVE.B #4,D0
5 BCL:
6     DBF D0,BCL
7
8 SIMHALT:           ; halt simulator
9 END: * Jump here to end the program
```

```
1 ORG $1000
2 START:
3     * Write here your code
4     MOVE.B #4,D0
5 BCL:
6     DBF D0,BCL
7
8 SIMHALT:           ; halt simulator
9 END: * Jump here to end the program
```

X	N	Z	V	C	L	W	B	Address	1000	Q	<	>
0	0	0	0	0	●	●	●	Tr	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F			
PC	00001004				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

X	N	Z	V	C	L	W	B	Address	1000	Q	<	>
0	0	0	0	0	●	●	●	Tr	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F			
PC	00001004				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

X	N	Z	V	C	L	W	B	Address	1000	Q	<	>
0	0	0	0	0	●	●	●	Tr	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F			
PC	00001004				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

X	N	Z	V	C	L	W	B	Address	1000	Q	<	>
0	0	0	0	0	●	●	●	Tr	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F			
PC	00001004				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



The screenshot shows a debugger interface with two main panes. The left pane displays assembly code:

```
1 ORG $1000
2 START:
3     ; Write here your code
4     MOVE.B #4,D0
5 BCL:
6     DBF D0,BCL
7
8 SIMHALT:           ; halt simulator
9 END: * Jump here to end the program
```

The right pane shows a memory dump from address 1000 to 10F0. A green box highlights the first few bytes of memory, which are filled with FF FF FF FF. The registers pane shows the PC at 00001008 and various registers (D0-D7, A0-A7) containing 00 00 00 00.

X	N	Z	V	C	L	W
0	0	0	0	0	●	●

Address	1000
PC	00001008
D0	00 00 ff ff
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

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