

TP1 Assembleur : premiers pas

P. Carreno - P. Portejoie

Antoine Gicquel**A2**

Les TP se déroulent sous Windows.

Vous utiliserez le simulateur d'assemblage SMS32v50. Vous pouvez y accéder par le lecteur G: (pensez à créer un raccourci y conduisant depuis votre espace personnel sinon vous ne pourrez pas enregistrer vos travaux). Vous pouvez aussi télécharger le dossier :

G:\prof\1tin01\ASR\M2101\ASM\ (récupérable aussi sur Moodle)



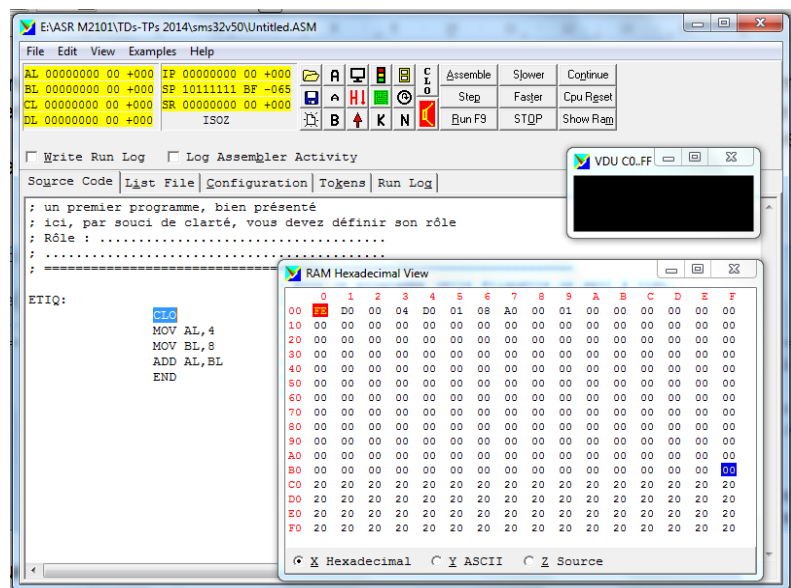
Pour chaque exercice votre compte-rendu devra comprendre une copie écran de votre code source assemblé (vous pourrez utiliser l'onglet List File du simulateur) et de son chargement en mémoire (format hexadécimal), ainsi qu'un compte-rendu d'exécution.

NB :

- pour les prochains Tps vos codes sources devront être commentés (pas de List File)
- les compte-rendus de TP doivent être rendus sur Moodle au plus tard le dimanche suivant le TP, avant 23h55

Travail préliminaire

Lancez le simulateur et observez-en les caractéristiques ; assurez-vous d'en comprendre la plupart des composants. N'hésitez pas à poser des questions à votre enseignant.



Exercice 1-1

- Reprenez la dernière version du programme de l'exercice 1-1 du TD et faites-le fonctionner ; observez bien le comportement des registres, en particulier IP. Vous vérifierez la conformité de l'assemblage fait manuellement en TP (onglet *ListFile*), ainsi que la conformité des résultats (observez la RAM ainsi que les registres AL et BL).

The screenshot shows an 8086 assembler interface. The main window displays the following assembly code:

```
AL 00001100 0C +012 IP 00001010 0A +010
BL 00001000 08 +008 SP 10111111 BF -065
CL 00000000 00 +000 SR 00000000 00 +000
DL 00000000 00 +000 IS0Z
```

The status bar indicates "END Program has halted." and "Log Assembler Activity" is checked. The "RAM Source Code View" window is open, showing the memory contents in hexadecimal, ASCII, and source code views. The source code view shows the program's execution flow, with the final state of the registers AL and BL highlighted.

Source Code View:

```
CLO
MOV AL,4
MOV BL,8
ADD AL,BL
END
```

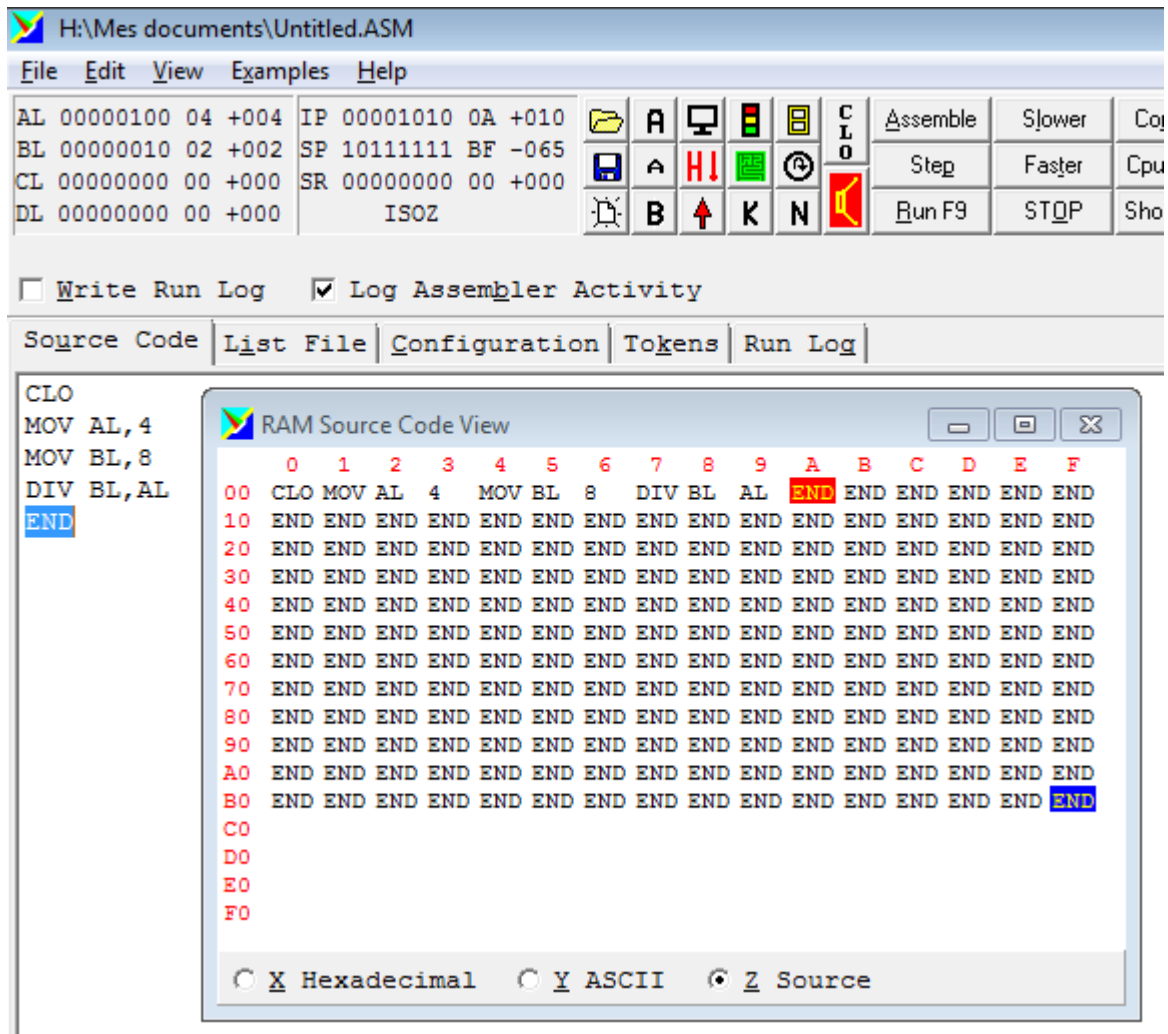
RAM Source Code View:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	CLO	MOV	AL	4	MOV	BL	8	ADD	AL	BL	END	END	END	END	END	END
10	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
20	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
30	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
40	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
50	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
60	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
70	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
80	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
90	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
A0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
B0	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END	END
C0																
D0																
E0																
F0																

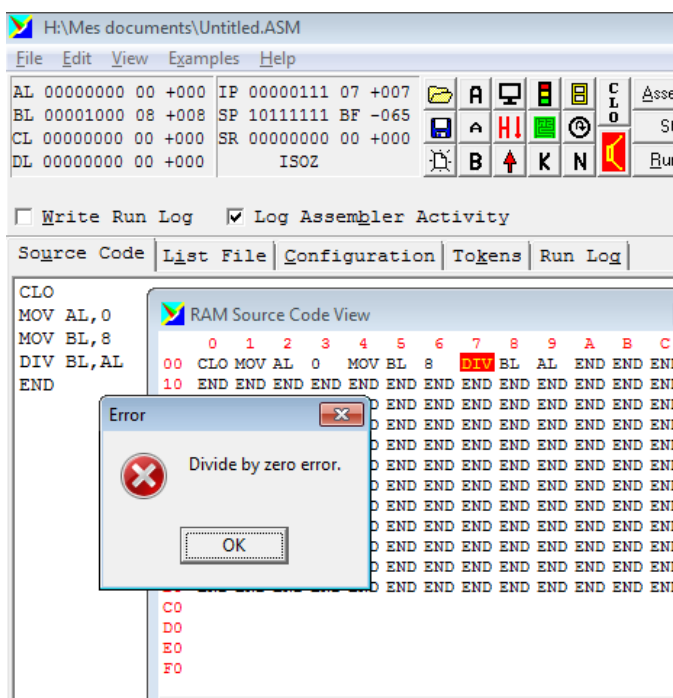
Hexadecimal: X ASCII: Y Source: Z

On voit que le registre AL contient bien 12 c'est à dire l'addition de 4 et 8 donc le programme s'est correctement exécuté.

- Modifiez-le de façon à ce qu'il fasse une division, puis testez le cas d'une division par 0.



On divise 8 par 4 et stock le résultat dans le registre BL. BL contient 2 donc le programme s'est correctement effectué.



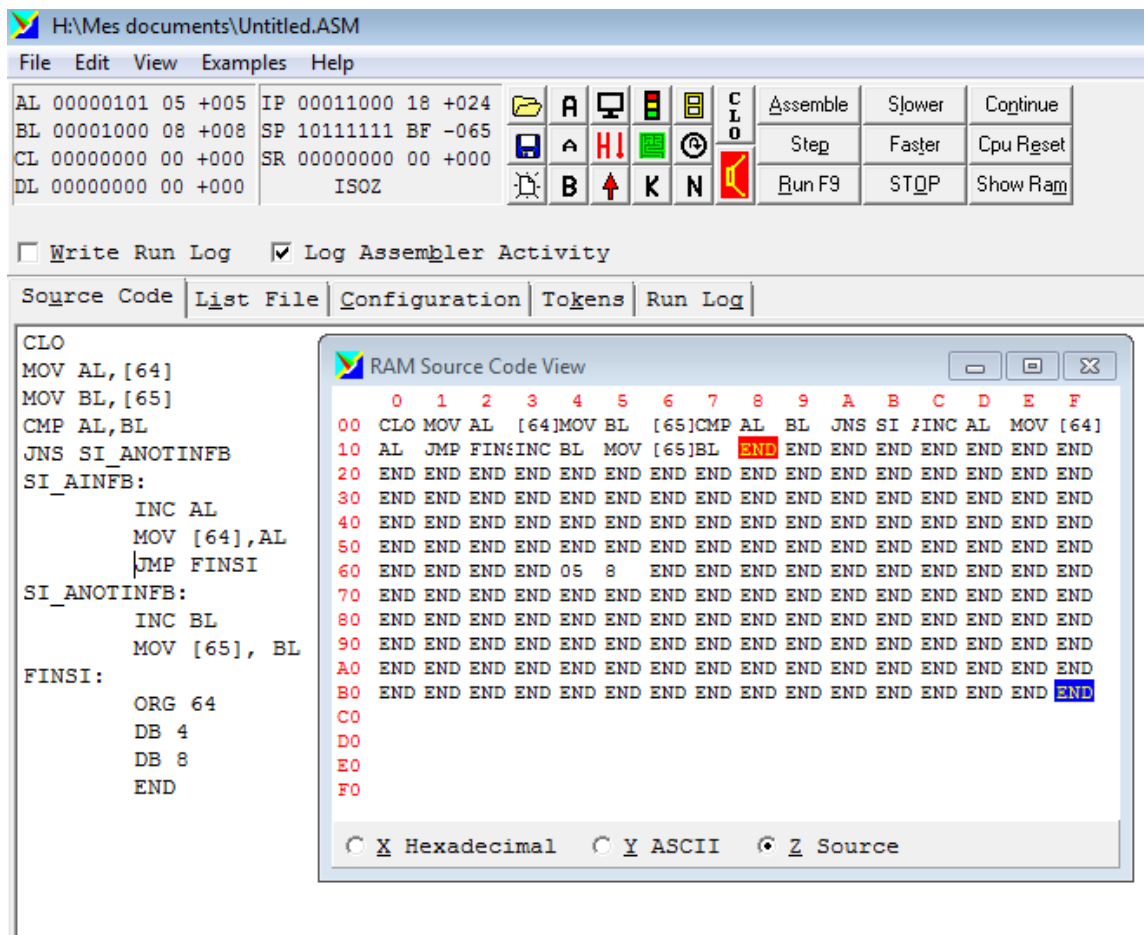
Lors d'une division par zéro, on obtient une erreur lors de l'exécution.

Exercice 1-3

- Reprenez le programme de l'exercice 1-3 du TD et faites-le fonctionner. Vérifiez la conformité de l'assemblage fait manuellement en TD, surtout en ce qui concerne le calcul des sauts. Faites les tests d'exécution appropriés afin de comprendre le fonctionnement du programme et de pouvoir indiquer ci-dessous l'emplacement du bit S du registre d'état.

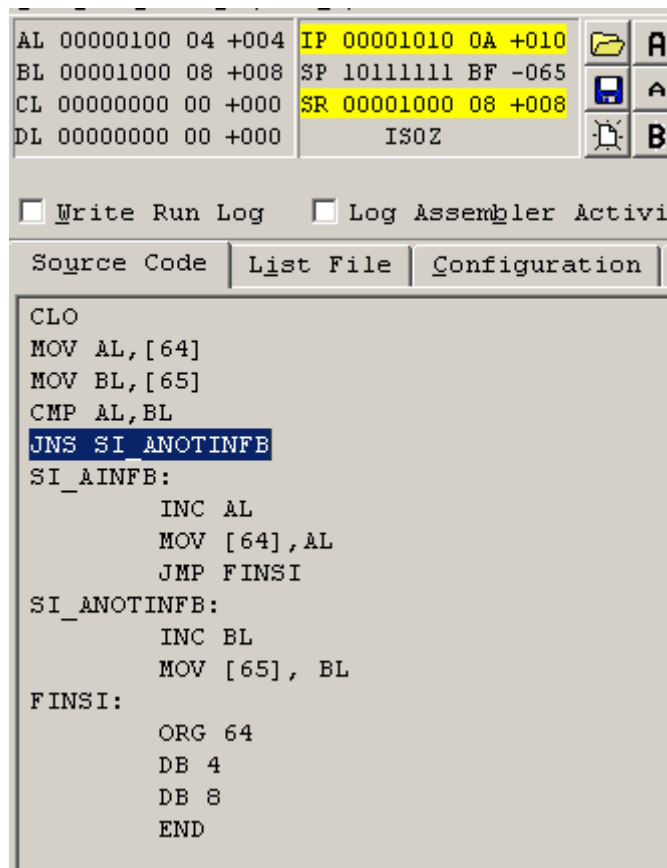
SR

0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---



AL à l'adresse 64 était la plus petite valeur (4<8) et donc été incrémenter de 1. Le programme s'est bien exécuté.

- Faites un test d'exécution approprié afin de pouvoir indiquer ci-dessus l'emplacement du bit Z du registre d'état (vous indiquerez le test effectué).



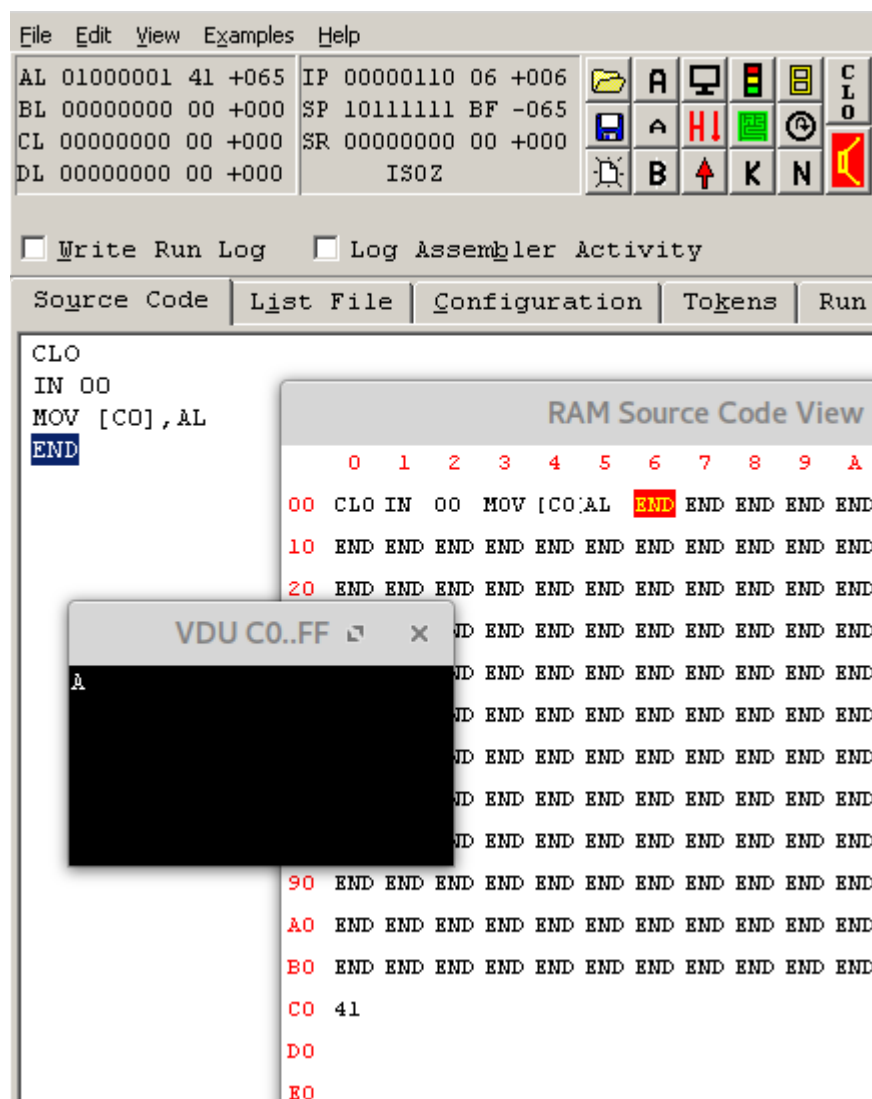
The screenshot shows an assembly editor interface. At the top, there are two columns of register values: AL (00000100 04 +004), BL (00001000 08 +008), CL (00000000 00 +000), and DL (00000000 00 +000) on the left; and IP (00001010 0A +010), SP (10111111 BF -065), SR (00001000 08 +008), and ISOZ on the right. Below the registers are checkboxes for 'Write Run Log' and 'Log Assembler Activi'. A tabbed interface at the bottom has three tabs: 'Source Code', 'List File', and 'Configuration'. The 'Source Code' tab is active, displaying the following assembly code:

```
CLO
MOV AL,[64]
MOV BL,[65]
CMP AL,BL
JNS SI_ANOTINFB
SI_AINFB:
    INC AL
    MOV [64],AL
    JMP FINSI
SI_ANOTINFB:
    INC BL
    MOV [65], BL
FINSI:
    ORG 64
    DB 4
    DB 8
    END
```

SR affiche un 1 pour S donc AL est inférieur à BL.

Exercice 1-4

- Ecrivez un programme qui permet la saisie au clavier d'un caractère et qui l'affiche sur le terminal virtuel.
- Pour la saisie au clavier vous utiliserez l'instruction *IN n°port* (consultez le récapitulatif du jeu d'instructions), sachant que le port correspondant au clavier est 00
- Pour l'affichage du résultat vous aurez simplement à ranger la valeur à afficher en mémoire, à l'adresse C0
- Testez votre programme en utilisant d'autres valeurs à la place de C0 (par exemple E6 ou encore FF). Qu'observez-vous ? Concluez.



La valeur de l'adresse modifie l'emplacement du caractère. Donc C0 est la première case de la matrice, tandis que E6 est dans le milieu.

Exercice 1-5

- Modifiez le programme de l'exercice 1-3 précédent (incrément de 1) afin de pouvoir saisir les données au clavier (uniquement des chiffres, mais vous n'avez pas à en programmer la vérification) et afficher le résultat sur le terminal virtuel.

NB : vous aurez beau chercher, vous ne trouverez pas pour ce simulateur d'instruction de transfert entre registres (par exemple MOV BL,AL). Vous devrez pour cela utiliser la pile (voir cours). Notez bien que cela n'est pas dû à une limitation due à la simulation, mais bien un choix de construction.

- Testez le programme avec les jeux de données suivants : 4 et 8, puis 9 et 9. Décrivez ce que vous observez dans le dernier cas et donnez-en une explication. Proposez brièvement une solution, mais sans chercher à la coder, cela fera l'objet d'un TD-TP ultérieur.

Après avoir rentré respectivement les valeurs 3 et 6 :

The screenshot shows the 8086 simulator interface. At the top, the register window displays the following values:

AL	00110110	36	+054	IP	00011100	1C	+028
BL	00110100	34	+052	SP	10111111	BF	-065
CL	00000000	00	+000	SR	00000000	00	+000
DL	00000000	00	+000	IS02			

Below the register window are checkboxes for "Write Run Log" and "Log Assembler Activity". The main window is divided into two panes. The left pane shows the assembly code:

```

CLO
IN 00
PUSH AL
POP BL
IN 00
CMP AL,BL
JNS SI_ANNOTINFB
SI_ANNOTINFB:
    ADD AL,1
    JMP FINSI
SI_ANNOTINFB:
    ADD BL,1
FINSI:
    MOV [C0],AL
    MOV [CF],BL
    END
  
```

The right pane shows the "RAM Source Code View" with a table of memory addresses and their contents:

Address	Content
00	CLO IN 00 PUSHAL POP BL IN 00 CMP AL
10	1 JMP FINSI ADD BL 1 MOV [C0],AL MOV [CF]
20	END END END END END END END END END ENI
30	END END END END END END END END END ENI
40	END END END END END END END END END ENI
50	END END END END END END END END END ENI
60	END END END END END END END END END ENI
70	END END END END END END END END END ENI
80	END END END END END END END END END ENI
90	END END END END END END END END END ENI
A0	END END END END END END END END END ENI
B0	END END END END END END END END END ENI
C0	END END END END END END END END END ENI
D0	END END END END END END END END END ENI
E0	END END END END END END END END END ENI
F0	END END END END END END END END END ENI

At the bottom, a VDU window titled "VDU C0..FF" is open, showing a black screen with the number "6" in the top left corner and "4" in the top right corner. The bottom status bar shows the output format set to "Hexadecimal".