Architecture des ordinateurs

Travaux pratiques N°2

‘Programmation Assembleur M68K - First Step’

Objectif de cette séance

Le but de ce TP est d'introduire l’édition, la compilation et la simulation avec EASy68K.

Cette fiche est à faire en binôme. Il faudra :

1. Réaliser un rapport à rendre au format **pdf** contenant les réponses aux questions de cette fiche ;
2. Envoyer le rapport à l'adresse de votre chargé de TP : mahmoudr@esiee.fr

1. L’environnement de travail ‘EASy68K’

L’environnement ‘EASy68K’ est un ensemble de programme permettant de manipuler, apprendre, tester … des programmes en assembleur 68000 (68K) de Motorola. Ce simulateur est gratuit, open source et sous License GNU – General Public Use Licence.

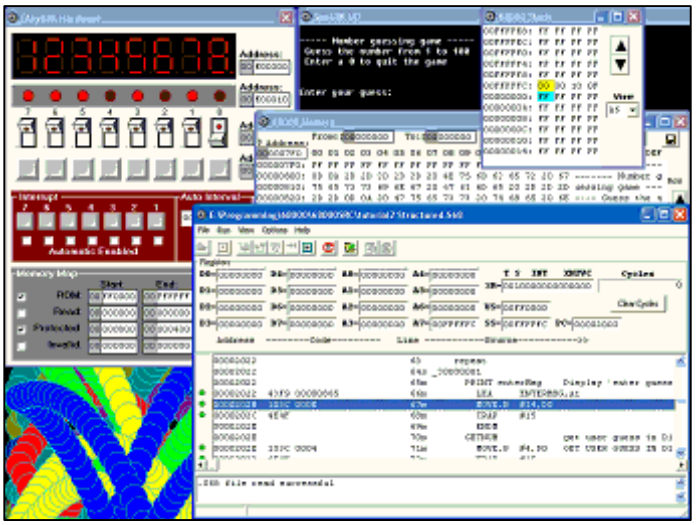

Fig. 1. Un aperçu des outils offerts par l’environnement EASy68K

Pour pouvoir l’utiliser sous Windows, il suffit de télécharger puis lancer l’installateur SetupEASy68K.exe.

Télécharger **SetupEASy68K.exe** depuis **http://www.easy68k.com/**

Sachant que les utilisateurs sous Linux ou Mac peuvent l’installer sous l’émulateur Windows, en tant que super utilisateur, avec la commande :

wine ./SetupEASy68K.exe

Une fois cet environnement est installé, vous disposeriez d'un menu similaire à celui-ci :

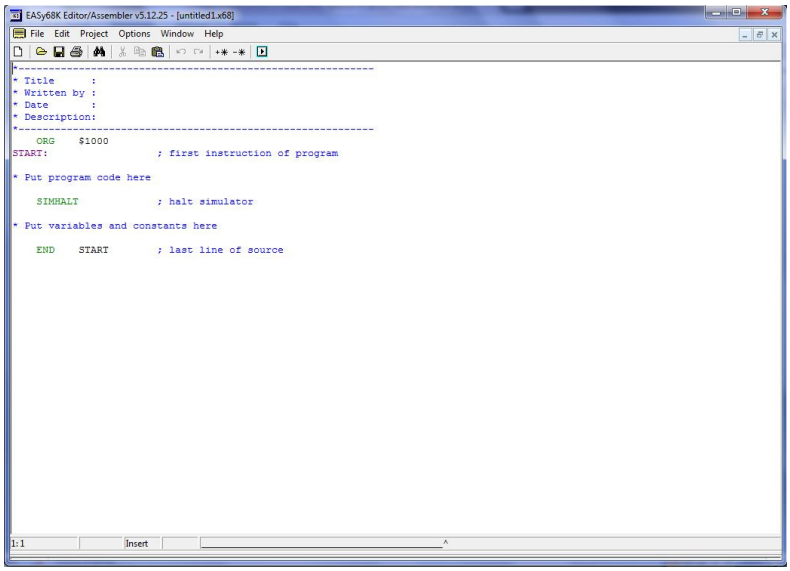

Fig. 2. Le menu de l'environnement EASy68K sous Windows

L'environnement **EASy68K** comporte un éditeur EASy68K, un Help, un simulateur Sim68K et un éditeur binaire **EASyBIN** qui permet de manipuler des zones mémoires ou des fichiers S-Record. Ces fichiers peuvent être manipulés en hexadécimal ou en mode texte.

1.1. Les fichiers manipulés

Afin de tester un programme assembleur, vous devez suivre plusieurs étapes. Le diagramme de la figure suivante illustre les étapes par lesquelles passe un programme écrit en assembleur 68000.

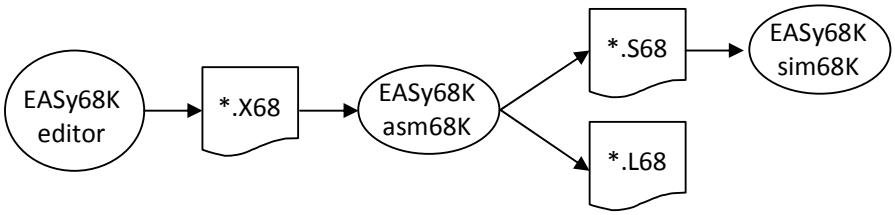

Fig.3. Les fichiers manipulés lors de la programmation assembleur

1) La première étape est d'éditer un fichier (EASy68K) contenant le code du programme assembleur dont l'extension doit être '.X68'
2) Une fois le fichier '.X68' est enregistré, vous pouvez le compiler. Cette étape génère deux fichier : un pour la simulation (exécutable) d'extension '.S68' et un autre pour le debuggage d'extension '.L68'
3) La simulation de ce programme est assurée par le programme Sim68K.

1.2. Description d'une ligne de code

Un programme en assembleur 68000 est constitué par des lignes de la forme :

Label	Opcode	Operand	Comment
.loop	ADD	D0,D1	Add two numbers
	BMI	.loop	Loop while negative

Fig. 4. Les champs d'une ligne de code assembleur 68000

- Label : une étiquette définie par le programme pour marquer un endroit du programme
- Opcode : une instruction que le processeur peut exécuter
- Operand : les données nécessaires à l'exécution des instructions
- Comments : des explications pour documenter le programme.

2. Édition, compilation et simulation

2.1. Création et édition d'un programme

Lancer l'éditeur EASy68K et créer un nouveau fichier. Saisissez puis enregistrer le programme suivant dans le répertoire Assembleur68K de votre espace de travail sous le nom Exemple_1.X68.

```
                    $1000       The program will load into address $1000
* Display HELLO massage
* See Help / Simulator IO for a complete list of task numbers
START               MOVE     #14,D0          Put text display task numbers
                    LEA      HELLO,A1        Load address of string to display into A1
                    TRAP     #15             Activates input/output task

* Display the contents of register D1
* task number 3 is used to display the contents of D1.L as number
                    MOVE.L   #12345678,D1    Put a number in D1 so we can display it
                    MOVE     #3,D0           Task number 3 in D0
                    TRAP     #15             Display number in D1

* Stop execution
                    MOVE.B   #9,D0
                    TRAP     #15                        Halt Simulator

HELLO               DC.B     'Hello World', $D, $A, 0   Null terminated string with null

                    END      START
```

Description du programme:

La première instruction spécifie l'adresse à partir de la quelle le programme sera chargé dans la mémoire. Le $ signifie que l'adresse est en hexadécimal.

```
                $1000       The program will load into address $1000
```

Les lignes qui commencent par * sont des commentaires. Exemple :

```
* Display HELLO massage
```

Mettre 14 dans le registre D0 (14 est le numéro identifiant la tâche d'affichage). Cette ligne est référencier par l'étiquette START.

```
START           MOVE    #14,D0          Put text display task numbers
```

Charger l'adresse de début du message à afficher, dans le registre A1, sachant que l'adresse du message est référencier par l'étiquette HELLO :

```
                LEA     HELLO,A1        Load address of string to display into A1
```

L'instruction suivante active la routine 'task' d'entrée/sortie du simulateur Sim68K, sachant que le numéro de la tâche à exécuter est dans le registre D0.

```
                TRAP    #15             Activates input/output task
```

Les lignes suivantes permettent de charger le nombre '12345678' dans le registre D1 puis de l'afficher[1] en utilisant la tâche numéro 3 de la routine TRAP #15.

```
                MOVE.L  #12345678,D1    Put a number in D1 so we can display it
                MOVE    #3,D0           Task number 3 in D0
                TRAP    #15             Display number in D1
```

Les deux lignes suivantes indiquent au simulateur d'arrêter la simulation :

```
                MOVE.B  #9,D0
                TRAP    #15                         Halt Simulator
```

La chaîne de caractère nommée HELLO est définit par la déclaration suivante. Cette variable contient le message 'Hello World' et se termine par un retour à la ligne ($D) poursuivi par une terminaison nulle.

```
HELLO           DC.B    'Hello World', $D, $A, 0    Null terminated string with null
```

La dernière ligne du programme indique la fin du programme, sachant que START est l'étiquette à partir de la quelle le simulateur commence l'exécution.

```
                END     START
```

[1] Afficher le contenu du registre D1.L en tant que nombre

2.2. Compilation, exécution et debuggage

Le bouton (project -> Assemble source) permet de compiler le fichier que vous avez sauvegardé. Si vous n'avez pas fait de fautes lors de la saisie du programme, vous aurez '0 Warnings' (des erreurs qui n'arrête pas la compilation)[2] et '0 Erros' (Des erreurs qui causent l'arrêt de la compilation et la non génération du fichier .S68)[3]. La figure suivante illustre ce cas de compilation sans erreurs.

Fig.6. Compilation – Fenêtre des erreurs

Cette fenêtre permet d'afficher le fichier .L68 (Load L68) ou de lancer le simulateur via le bouton exécute (Execute). Une fois le simulateur est lancé, vous avez le choix entre plusieurs mode d'exécution. Ces modes seront très utiles pour le débuggage de vos programmes. L'exécution de ce programme se termine par l'affichage de la fenêtre donnée par la figure suivante :

Fig.7. Fenêtre d'E/S du simulateur

Dans le cas ou le programme contient des erreurs, des messages d'erreurs localisant ces fautes seront affichés dans la partie basse de l'éditeur. Il suffit de cliquer sur un message pour être diriger vers la aligne contenant l'erreur. Le fichier log .L68 permet aussi de localiser l'erreur avec plus de détails.

[2] Des messages auxquels il faut faire attention, car ils peuvent induire à des erreurs sur les résultats
[3] Les erreurs syntaxiques, les fautes de frappe …

2.3. Simulation du programme

La figure suivante affiche la fenêtre principale du simulateur permettant de suivre l'évolution de l'exécution du programme.

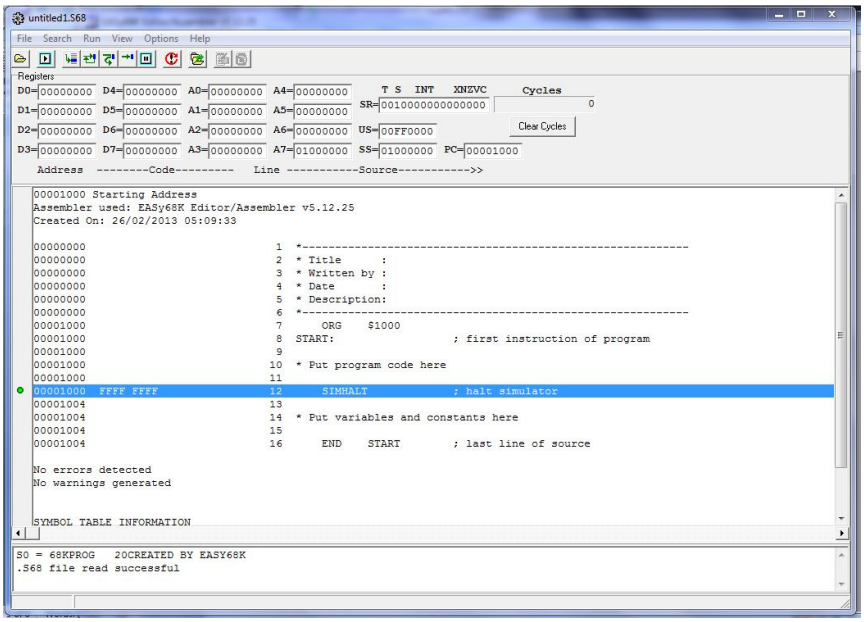

Fig.8. La fenêtre principal du simulateur

Cette fenêtre permet de suivre l'évolution des registres D0…D7, A0…A7, SS, US, SR et PC au cours de la simulation d'un programme. Elle offre aussi divers outils pour la manipulation et la suivie du contenu de la mémoire, de la pile, ainsi qu'une maquette de simulation 'hardware'.

Cette fenêtre affiche aussi le fichier '.L68' au cours de la simulation. Ce fichier comporte plusieurs colonnes qui simplifient la compréhension et la suivie de l'exécution du programme :

- Adress : l'adresse à partir de la quelle l'instruction commence ;
- Code : le code de l'opération en hexadécimal
- Line : Le numéro de la ligne correspondante ;
- Source : le contenu du fichier source du programme

La barre d'outils offre aussi plusieurs boutons permettant d'interagir avec le simulateur, en sélectionnant le mode de simulation ainsi que l'opération à réaliser.

Vous pouvez consulter l'aide du logiciel pour plus d'informations sur les fonctionnalités offertes par le simulateur.

3. Travail demandé

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

Q1. Faites une exécution en mode ligne par ligne et dresser un tableau contenant les registres qui changent à chaque étape.

Q2. Afficher le contenu de la mémoire,

Q3. Donner les adresses de début et de fin de la partie code et de la partie données de ce programme.

Q4. Localiser ka variable Hello dans la mémoire et donner la liste des codes ascii correspondant aux caractères de ce message.

Q5. Comment peut-ton définir la tâche que doit effectuer l'instruction **trap #15**, et comment elle récupère les paramètres nécessaires.

4. Bonus

Q1. Modifier le code précédent pour afficher plusieurs lignes de texte. Vous pouvez vous inspirer de l'exemple donné dans le 'Quick Start Programs' à l'adresse suivante :

http://www.easy68k.com/easy68kexamples.htm

Q2. Tourner puis décrire les résultats renvoyés par les exemples de codes de Lab2 donnés en téléchargement à l'adresse suivante :

http://www.esiee.fr/~mahmoudr/architecture.html

5. Bibliographie

[1] TP programmation assembleur 68000 sous l'environnement « EASy68K », S. Bazine et I. Ben Ameur.
[2] http://www.easy68k.com/
[3] http://mycorner.no-ip.org/68k/easy68k/index.html