Architecture – Fiche de TP - n°04
Structures de contrôle en assembleur 68000 : branchements et CMP

Objectif : Mise en œuvre des instructions de branchement et utilisation des instructions de comparaison (CMP et CMPI). Comme toujours, la documentation technique est nécessaire.

Rappel : Dans l'écriture d'un programme en assembleur à partir d'un algorithme, vous veillerez à ce que votre programme respecte bien l'algorithme proposé, en particulier :

- L'ordre des opérations algorithmiques doit être respecté dans le programme en assembleur.
- Toutes les instructions nécessaires à l'exécution d'une opération algorithmique doivent être effectuées avant de passer à l'opération algorithmique suivante.
- En particulier, n'oubliez pas l'affectation de la variable du résultat d'un calcul lorsque cela est nécessaire.

1. Tant que

1.1. Créez un nouveau répertoire et recopiez le fichier :

`/users/info/pub/commun/r1-03archi/tp04/tp04.s`

Ce fichier contient le début de la traduction en assembleur de l'algorithme suivant :

```
Solde := 20 ;
Depenses := 40 ;
tantque (Solde <= 20)
     et (Depenses < 50)
faire
     Solde:= Solde-5;
     Depenses:=Depenses+5;
finfaire;
```

1.2. Réfléchissez tout d'abord sur l'algorithme (comme vu en cours) pour indiquer, pour chaque clause, la condition du branchement assembleur (qui doit donc être associée à la rupture de séquence) et sa cible.

Pour vous aider, les étiquettes reprenant la structure du « tant que » sont indiquées dans le fichier tp04.s.

Ecrivez le programme correspondant en assembleur : n'oubliez pas les commentaires à côté des instructions, en particulier pour les branchements.

Compilez votre programme (Rappel : **`68kasm -l tpxx.s`**).

Et faites-le exécuter au pas à pas pour suivre son bon déroulement, en observant « simultanément » :

- via l'évolution des valeurs des 2 variables en mémoire centrale ;

- via le fichier .lis qui permet de voir l'instruction qui va s'exécuter ;

- via la fenêtre Trace qui permet de voir l'instruction qui s'est exécutée ;

- via l'évolution du PC (Compteur Ordinal) qui évolue à chaque instruction.

2. Tant que… un peu plus complexe

Procédez de même (et en créant, bien sûr, un autre fichier tpxx.s) pour l'algorithme suivant :

```
        a := 5 ; entier naturel en .W
        b := 4 ; entier naturel en .W
        c := 'S' ; 1 caractère par DC.B

tant_que (2 < a)
        et
        [ (3>=b)
          ou
          (c='S')]
faire
        c:='N' ; b:= b-1 ; a := a-1 ;
finfaire;
fin
```

N'oubliez pas, en première étape, d'analyser (sur le papier) chaque condition du tantque :

- Indiquer la cible de la rupture de séquence
- Indiquer la condition de cette rupture de séquence

3. PGCD de deux entiers naturels

Écrivez un programme qui calcule le Plus Grand Commun Diviseur de deux **entiers naturels** A et B, codés sur **32 bits**, selon l'algorithme suivant :

```
déclarer et initialiser A et B : entiers naturels


copier A dans le registre D1
et B dans le registre D2
tant que D1≠D2 faire
                  si D1>D2
                      alors D1:=D1-D2
                      sinon D2:=D2-D1
                  finsi
          finfaire;
D0 := D1     ; la valeur du PGCD apparaît dans D0
```

Testez votre programme pour différentes valeurs de A et B (il faut à chaque fois réassembler et recharger le nouveau programme .h68) :

pgcd(4,4)=4 ; pgcd(12,4)=4 ; pgcd(56,241)=1 ;

pgcd(481,1313)=13 ; pgcd (10165,3745)=535 ;

pgcd(221,143)= ? ; pgcd(348,2225)= ?

4. Si..alors..sinon : pour entraînement et révisions

4.1 ET

Soit le programme suivant :

```
a := 1 ;entier naturel en .W
b := 10 ; entier naturel en .W
cas := 'init' ; 4 caractères par DC.L
début
si (a > b) et (a < 4)
     alors cas := 'alo1' ;
     sinon cas := 'sin1' ;
finsi;
fin
```

Le traduire en assembleur sans oublier de mettre des commentaires.

Tester votre programme pour différentes valeurs de a et b mettant bien en évidence les différents cas.

4.2 OU

Idem avec d'autres conditions, par exemple :

```
si (a > b) ou (a < 4)
     alors cas := 'alo2' ;
     sinon cas := 'sin2' ;
finsi;
```

Le traduire en assembleur sans oublier de mettre des commentaires.

Tester votre programme pour différentes valeurs de a et b mettant bien en évidence les différents cas.