Système Embarqué

Journal - TP.02 : Introduction à l'assembleur

Temps effectué hors des heures de classe

Nous avons travaillé jusqu'à 18h30, donc 2h de plus.

Synthèse des acquis

Acquis :
- Tableaux en assembleur

Acquis, mais à exercer encore :
- Utilisation de Git
- Debugging du code dans Eclipse
- Principes de bases de l'Assembleur

Questions

Citer les modes d'adressage courants supportés par le µP ARM.

- Traitement de données, qui se base sur l'utilisation d'opérande de décalage et de rotation.
- Echange de données avec la mémoire, qui se base sur l'utilisation d'instruction ldr ou str indexées (pré ou post).
- Echange de données avec les coprocesseurs.

Quel outil permet de transformer le code assembleur en code objet ?

On transforme un programme écrit en langage d'assemblage en du code objet avec un outil nommé assembleur. À chaque mnémonique (instruction au nom simple) du langage d'assemblage pour un processeur donné, combiné à ses paramètres correspond une instruction-machine. De même, les paramètres des instructions en langage d'assemblage rejoignent ceux de leurs équivalents en langage machine.

-h-deb.clg.qc.ca

Quel outil permet de créer l'application à partir des codes objet précédemment générés ?

L'éditeur de liens.

Pourrait-on optimiser l'algorithme du crible d'Eratosthène ? Si oui, comment ?

Oui, il suffit de réaliser le crible d'Eratosthène, mais en ignorant tous les entiers multiples des plus petits nombres premiers (2, 3, 5, 7...). L'exécution du crible s'en retrouve ainsi fortement améliorée.

Autre méthode, mais moins efficace, supprimer la boucle qui remplit le tableau de 1, puis dans la deuxième boucle qui mettait des 0 dans les nombres non premiers, mettre la valeur 1.
Ensuite, il ne reste plus qu'à imprimer les nombres avec 0 comme valeur à leur indice. Il s'agit de cette dernière méthode que nous avons implémentée en assembleur comme amélioration.

Pourrait-on réduire la taille de votre code en assembleur ? Si oui, comment ?

En réduisant au maximum l'utilisation de variables et constantes, et en diminuant la taille du tableau contenant les nombres premiers.

Remarques

La déclaration et l'utilisation de tableau en assembleur se résument de la façon suivante :

```
1   // Create boolean table[100]
2
3   table:  .space 2*100    // 100 cases of 2 byte
4   table:  .space 100      // 100 cases of 1 byte
5
6   // Find table[51]
7
8   ldr r0, =table          // r0 = First element address
9   add r0, #10             // r0 = Eleven element address
10
11  mov r2, #100
12  str r2, [r0]            // table[11] = 100
13
14  mov r1, #2
15  str r2, [r0, r1]        // table[11+2] = 100
16  str r2, [r0, -r1]       // table[11-2] = 100
17
18  // To read, use ldr instead of str
```

Feedback

Une introduction à certains outils de debugging, comme la lecture de la mémoire, serait utile pour nous aider à comprendre et débugger nos codes.

Référence

Transformation assembleur en code objet