Charlotte Junod et Nicolas Fuchs
charlotte.junod@edu.hefr.ch et nicolas.fuchs@edu.hefr.ch

Systèmes embarqués I - Journal
TP0 :

Heures de travail en dehors des cours : 2

Synthèse et feedback

Nous avons pu voir comment faire des tableaux en assembleur et comment implémenter des boucles for.

Par contre, ce tp aurait été plus facile si nous avions vu la théorie sur l'adressage avant.

Aussi, nous n'avons pas pu réellement tester la fonction printf car nous ne savions pas comment afficher les sorties dans la console.

Questions

Citer les modes d'adressage courant supportés par le µP ARM

Il y a trois modes :

Immédiat : en indiquant une adresse de départ et un offset (entier)

Par registre : en indiquant une adresse de départ et un registre contenant la valeur de l'offset

Par registre et décalage : en indiquant une adresse de départ, un registre contenant une valeur et un entier. L'offset est déterminé par la soustraction du registre avec l'entier donnés.

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

Il s'agit du compilateur, qui traduit notre code en byte code, ce qui génère nos fichiers « .o ».

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

Il s'agit de l'éditeur de liens, qui permet de lier (d'où son nom) les différents objets.

Pourrait-on optimiser l'algorithme d'Eratosthène ?

Oui, lors de la deuxième boucle, lorsqu'on « élimine » les nombres qui sont des multiples. Avant de rentrer dans la boucle for intérieure (celle dont la variable d'incrémentation est j dans l'exemple du TP), on pourrait tester si i a déjà été déclaré non premier (autrement dit, s'il est déjà multiple d'autre chose) et si c'est le cas, ne pas exécuter la boucle intérieure.

Une implémentation de cette méthode est proposée dans le fichier main.S rendu avec le tp, en commentaire au-dessous du code « de base ».

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

Oui, en appliquant la méthode expliquée ci-dessus. La taille du texte constituant le code augmenterait, mais le temps d'exécution serait réduit.