#Livrable 3#
Projet Informatique
Emulateur-Arm

I-Désassemblage :

- Nous arrivons maintenant à lire toutes les instructions ainsi qu'à détecter les valeurs des opérandes avec toutes les fonctions de mise en forme (définies dans format-o.h/.c), ces fonctions nous permettent de traiter tous les cas possibles des valeurs immédiate , elles nous permettront aussi d'élaborer quelques commandes pour l'exécution comme les rotation et le décalage.
- On stocke toutes ces instruction dans un tableau d'instruction qui va contenir le code de l'instruction, les valeurs des opérandes, ainsi que toute les informations nécessaires.
- Nous avons créé une fonction **lecture_txt** qui nous permet de lire le segment {.text} et d'en extraire toutes les instructions désassembler sous forme d'un tableau d'instruction.
- Cette fonction sera alors appelé dans la commande '**disasm'** pour but d'afficher, et dans la commande '**run'** pour exécuter.

Il reste des erreurs à corriger en ce qui concerne l'allocation et la libération de cette structure.

Il nous reste d'inclure ce tableau d'instruction dans la structure inter afin de l'avoir durant toute l'exécution de l'émulateur.

Nous ne disposons pas encore des fonction pour élaborer les instructions du langage assembleur.

→ Block IT :

Notre programme arrive à distinguer les différents cas des blocs IT ainsi que les conditions associés. On se limite en ce moment à l'affichage , en changeant la mnémonique. Nous n'avons pas pu traiter l'exécution.

II-Exécution :

Dans cette partie nous présentons notre perspective pour élaborer l'exécution ;

Cette partie n'est pas incluse dans l'émulateur, parce que nous n'avons pas encore fini les tests associés.

→ Création d'un fichier run.c et run.h
dans **run.c** :
fonction runcmd : int **_runcmd**(interpreteur inter, unsigned int adresse)

cette fonction est constituée d'une boucle infinie comportant un switch sur les différentes valeurs possibles de l'exécution. On définie dans **'run.h'** RUN et PAUSE comme des entiers (0 pour RUN, 1 pour PAUSE).
Dans la fonction **_runcmd**, lorsque l'état vaut RUN, la fonction désassemble à partir de l'adresse fournie en paramètre et accède au tableau stockage_inst qui contient toutes les instructions sous forme de structure de type DESASM_INST. La fonction fait alors appel à la fonction find_inst pour chaque éléments du tableau :
int **find_inst**(interpreteur inter, DESASM_INST instruction).
La fonction compare l'identifiant grâce à un switch l'identifiant de l'instruction pour pouvoir faire appel à la bonne fonction d'exécution correspondante : stockage_inst[i].id. Enfin la fonction d'exécution correspondante à l'instruction trouvée exécute à proprement parler les opérations sur les registres créés et stockés dans inter.

Les choses à modifier : dans les fonctions _runcmd et find_inst il faut rajouter inter en paramètre. Dans les dicos il faut rajouter une colonne ou tu mets un unique numéro pour chaque instruction (correspondant au int id que j'ai rajouté dans la structure DESASM_inst) pour pouvoir ensuite dans find_inst faire un switch sur id.

III-Répartition du travail :

Ali Saghiran : - Fonction de mise en forme des opérandes.

- Amélioration désassemblage.

Damien Chabannes : - Lecture des valeurs des opérandes

-Blocs IT + conditions