Institut National des Sciences Appliquées

Projet Systèmes Informatiques

Du compilateur vers le microprocesseur

https://github.com/Thxmasb/Projet_Systeme_Info

Thomas BALLOTIN, Terani LUQUE

Département Génie Électrique et Informatique
135, Avenue de Rangueil
31077 Toulouse Cedex 4
Émail : noms@etud.insa-toulouse.fr

1 Développement d'un compilateur en utilisant LEX et YACC

1.1 Programmation de différentes fonctions C pour gérer la table des symboles

Pour gérer la table des symboles nous avons décidé de faire un programme C.

Ce programme comporte une structure comprenant le nom du symbole et un booléen qui permet de savoir si un symbole à été initialisé.

Il y a également 4 fonctions :

1. Une fonction "ajouterSymbole" permettant d'ajouter des symboles dans la table ;
2. "indexSymbole" qui cherche l'index d'un symbole dans la table ;
3. "initialSymbole" qui permet de savoir si le symbole fourni est initialisé ou non ;
4. "initialiserSymbole" permet d'initialiser le symbole.

1.2 Choix d'un fichier pour programme ASM

Nous avons décidé d'écrire le code assembleur de notre programme en C dans un fichier texte. Ainsi, nous pouvons travailler avec grâce aux fonctions ftell(), fprintf() et fseek().

1.3 Implémentation de la fonction *If* et *While*

Pour la programmation du *If* et *While* nous avons choisi de faire des non-terminaux *If*, *While* et *BodyIf*.

Nous avons choisi de faire un *BodyIf* different du Body "normal" d'un programme car dans le body d'un if nous pouvions avoir seulement des Instructions et pas seulement des déclarations suivies d'instructions comme dans un programme.

Nous avons également mis en place un compteur d'instructions pour savoir combien d'instruction ASM il y a eu depuis le début du programme.

Nous avons mis un type nombre au token t_IF pour pouvoir lui attribuer des valeurs. Lors du début d'un if nous y stockons la valeur de la fonction ftell qui permet de savoir ou le curseur de notre fichier est actuellement.

Ensuite, nous écrivons l'instruction ASM *JMF* avec en argument :

- le numéro d'instruction de la condition du IF
- notre ligne actuelle dans le fichier (ftell)

Ensuite, nous écrivons les instructions ASM du body du if et à la fin nous faisons un fseek pour pouvoir se remettre à la ligne où JMF prenait place.

Puis nous écrasons cette même ligne à l'aide d'un fprintf avec un JMF où nous changeons le deuxième argument par le compteur d'instruction final + 1.

2 Conception d'un microprocesseur de type RISC avec pipeline

2.1 Choix d'utilisation de processus

Nous avons choisi d'implémenter un processus dans certaines entités, notamment pour les entités qui ont besoin d'être activé sur un front montant d'horloge. Ces entités sont les suivantes :

- le compteur
- le processeur
- la mémoire d'instruction
- le banc de registre
- la mémoire de donnée

2.2 Processeur

Il s'agit maintenant d'intégrer tous les éléments précédemment implémentés afin de créer une première version du processeur. Il fallait implémenter les instructions les unes après les autres. Premièrement, nous avons défini tous les composants du processeur puis tous les signaux nécessaires. Chaque "ligne" entre deux registres sur le schéma correspond à un signal. Dans un deuxième temps, nous avons déclaré les port map de chaque composant. Enfin, dans un processus, on définit les échanges internes dans le chemin de données.

3 Problèmes rencontrés

Plusieurs fois durant le projet la syntaxe fut un obstacle au bon déroulement de nos programmes. Nous avons aussi passé beaucoup de temps à implémenter les expressions conditionnelles du If et du While. En effet, l'utilisation des fonctions de gestion d'un fichier texte ne nous était pas familière et il nous a fallu du temps pour les appréhender.