Université de Strasbourg
L2 Informatique - S3 Printemps - 2020
Groupe: TP2

Architecture des Ordinateurs
Louis DISTEL
Paul MEYER

Donne moi ton numéro

de carte bancaire

Donne moi ton numéro de carte bancaire est un projet en Assembleur MIPS qui contient deux fonctions, l'une pour générer et l'autre pour vérifier un numéro de carte bancaire.

Pour commencer nous avons réussi à faire le projet et toutes ses améliorations proposées dans le sujet, ce qui est le résultat d'un travail d'équipe régulier dont les étapes de conceptions sont disponibles sur GitHub. "Donne moi ton numéro" sera rendu publique une fois le projet rendu.

Sachant que nous sommes un binôme, et qu'il y'a deux fonctionnalités très distinctes, nous avons décidé de diviser le travail en deux. Paul s'est donc occupé de la partie "vérification de numéro de CB" et Louis de la partie "génération de numéro de CB" (CB, alias carte bancaire).

Pour le travail en binôme, nous avons choisi de faire un repository GitHub, avec chacun notre branche pour travailler. Nous avons ensuite fusionner le tout, après avoir chacun fini nos modifications personnels. De plus, nous avons tout deux commit notre travail le plus régulièrement possible pour se tenir au courant des dernières modifications.

Les fonctions basiques nécessaires au bon fonctionnement du programme ont été satisfaites sans poser de problèmes particuliers que ce soit au niveau de l'algorithme ou au niveau de la retranscription en MIPS. Nous avons naturellement effectué cela en respectant toutes les contraintes imposées par le sujet.
En revanche, la partie consacrée à l'amélioration a demandé plus de temps et a posé plus de problèmes pour les deux parties, que ce soit pour générer ou pour vérifier un code. Cependant, nous avons finalement réussi à implémenter toutes les améliorations proposées dans le sujet à savoir la génération et la vérification d'un code en prenant compte de sa longueur et du type de carte.

1. Partie vérification (Paul)

La vérification de numéro de carte bancaire n'a pas posé de problèmes particuliers. Cependant, pour trouver l'émetteur de la carte bancaire, dans la deuxième partie, nous avons rencontré quelques difficultés et cela a nécessité plus de reflection.

L'idée de base était de trouver une forme de stockage brute adaptée pour les deux parties du programme, mais finalement nous avons décidé d'utiliser chacun notre propre forme. Certes, cela fait des dupliqués d'entrées brutes, mais il était difficile voir impossible, de faire autrement.

Pour stocker les premiers chiffres des émetteurs, j'ai d'abord utilisé un tableau d'entiers (.word). Cependant, un problème s'est vite présenté. Les premiers chiffres n'ont pas tous la même longueur (parfois 1, 3, 4 chiffres) et il n'existe pas de fonction pour compter cette longueur. Il est donc impossible (sans cette fonction) de prendre les 1, 3, ou 4 premiers chiffres du numéro que l'on souhaite vérifier pour les comparer. J'ai finalement décidé de stocker les chiffres de chaque émetteur dans des chaînes de caractères (.asciiz). Chaque nombre est séparé par une virgule, ce qui permet de les différencier. Pour trouver le bon émetteur, j'ai donc comparé chaque chiffre en débutant du premier chiffre avec les chiffres du code de carte bancaire à vérifier.
Par exemple, si le chiffre à la position 1 n'est pas égal à la position 1 du code de CB, alors on passe à la virgule suivante, et on compare. Si on atteint la virgule d'après sans qu'il y ait eu de faute, alors le numéro de début est valide.

Pour les plages, j'ai convertis les 4 ou 6 premiers chiffres du code de CB en entier. J'ai multiplié chaque chiffre par une puissance de 10 en partant de la fin. (par exemple 35 = 5*10^0 + 3*10^1). Puis j'ai vérifié s'il se trouvait entre les bornes minimum et maximum de la plage (stockées en brut).

2. Partie génération (Louis)

Pour générer un code aléatoire, j'ai commencé par traiter le problème en ne stockant aucune valeur. Je les affichais pour ensuite leur appliquer des opérations, dans le but de générer le dernier chiffre de vérification. Cette étape ne m'a pas posée de problèmes et était relativement courte à réaliser.

J'ai, cependant, rencontré quelques difficultés pour la génération de code en prenant en compte des débuts types.
En effet, j'ai réussi à faire en sorte qu'on puisse générer un code de carte en choisissant le type de carte souhaité (ce qui change donc le début), puis précisément ou aléatoirement la taille du code souhaité.

En revanche, même si la génération fonctionne dans le code du programme en lui même, j'ai fait trop de redondances dont je n'ai pas réussi à me défaire. Je fais ici référence aux neuf fonctions dans la séquence intitulée "CHOIX DES CARTES" où de petits éléments changent à chaque fois. Cependant, je n'ai pas réussi à n'en faire qu'une seule fonction pour en éviter les redondances.
Cela compte aussi pour la séquence intitulée "SÉQUENCE DE GESTION DU CHOIX DE LA TAILLE" où je traite un nombre en lui soustrayant à chaque fois la valeur de sa grandeur. Par exemple, pour 34 972, je stocke le 3 dans une autre variable ce qui me laisse ensuite uniquement 4 972.
Mais ce fonctionnement m'a créé un grand nombre de répétitions dont j'ai essayé de me défaire. J'ai tenté de résoudre ce problème, en vain. Ce sont des points négatifs dont je suis conscient.

En conclusion, malgré les redondances, la génération de code fonctionne et se sert de presque tous les registres temporaires. Ces derniers possédant eux-mêmes une propriété individuelle à chaque registres, explicitée dans les commentaires du programme.

Pour en savoir plus sur le projet et sa réalisation voici le lien GitHub pour y accéder:
- https://github.com/SchawnnDev/ProjetMIPS