PROJET - Architecture des ordinateurs

Renaud Vanhoutrève (vanhoutreve@unistra.fr)

13 octobre 2014

1 Bonnes pratiques

Pour vous préparer au monde de l'entreprise, les pratiques de développement suivantes sont valables pour les exercices :

- l'ensemble des fonctions doit être commenté avant la définition de cette dernière :
  - l'intérêt de la fonction doit être résumé ;
  - l'ensemble des paramètres et leurs registres doivent être expliqué ;
  - la valeur retournée et le registre utilisé doivent être précisée.
- l'ensemble des fonctions doivent être d'au maximum 40 lignes ;
- l'ensemble des fonctions doivent avoir une indentation identique entre elles ;
- il ne doit pas y avoir plus d'une instruction par ligne ;

2 Principe

Le sudoku est un jeu en forme de grille défini en 1979 par l'Américain Howard Garns, mais inspiré du carré latin, ainsi que du problème des 36 officiers du mathématicien suisse Leonhard Euler.
Le but du jeu est de remplir la grille avec une série de chiffres (ou de lettres ou de symboles) tous différents, qui ne se trouvent jamais plus d'une fois sur une même ligne, dans une même colonne ou dans une même sous-grille. La plupart du temps, les symboles sont des chiffres allant de 1 à 9, les sous-grilles étant alors des carrés de $3 * 3$. Quelques symboles sont déjà disposés dans la grille, ce qui autorise une résolution progressive du problème complet.

3 Sujet

Le but de ce projet est :

- lire un fichier contenant un sudoku partiellement rempli.
- afficher a l'écran toutes les solutions possibles. Les sudokus disponibles (livres d'énigme, journaux...) ont généralement qu'une seule solution, ceci amenant de la difficulté pour la personne qui le résout. Cependant il est assez facile de générer des sudoku qui admettent plusieurs solutions.

Les contraintes sont les suivantes :

- le projet sera réalisé en MIPS ;
- L'algorithme de recherche des solutions sera récursif et utilisera une version "brute force" amélioré.
- Le programme devra se lancer à l'aide de la commande suivante :

  ```
  spim -file <votre_fichier>.asm <sudoku_file>
  ```

  Voici une version graphique de ce qui est demandé :
  `http://codertao.com/projects/ai-sudoku/index.php?page=7`

4 Format du fichier d'entrée

Le fichier d'entrée aura la forme suivante :

415638972362479185789215364926341758138756429574982631257164893843597216691823547

ce qui correspond à :

4	1	5	6	3	8	9	7	2
3	6	2	4	7	9	1	8	5
7	8	9	2	1	5	3	6	4
9	2	6	3	4	1	7	5	8
1	3	8	7	5	6	4	2	9
5	7	4	9	8	2	6	3	1
2	5	7	1	6	4	8	9	3
8	4	3	5	9	7	2	1	6
6	9	1	8	2	3	5	4	7

Il est bien évident que le précédent sudoku est complet et ne néccessite donc pas de résultat autre que lui-même.
Les cases dont la valeur est inconnu sur le sudoku seront défini comme ayant la valeur 0.
Voici un exemple :

415638972362479185789215364926341758138756429574982631257164000843597000691823000

ce qui correspond, selon le formalisme spécifié ci-dessus, à :

4	1	5	6	3	8	9	7	2
3	6	2	4	7	9	1	8	5
7	8	9	2	1	5	3	6	4
9	2	6	3	4	1	7	5	8
1	3	8	7	5	6	4	2	9
5	7	4	9	8	2	6	3	1
2	5	7	1	6	4	0	0	0
8	4	3	5	9	7	0	0	0
6	9	1	8	2	3	0	0	0

5 Résultat du programme

5.1 Format de la sortie

Le programme produira un résultat de la forme suivante :
$< sudoku\_initial >< CR >$
$< sudoku >< CR >$
$< sudoku >< CR >$
...
$< sudoku >< CR >$
$< sudoku >< CR >$

avec :
$< sudoku\_initial >$ est le sudoku fourni en entrée du programme (tel qu'il est dans le fichier)
$< sudoku >$ est un sudoku possible ayant le même formalisme que le $< sudokuinitial >$
$< CR >$ signifie "carriage return" ou saut de ligne symbolisé par le caractère '\n'.

5.2 Exemple

Supposons que dans le fichier de référence vous avez le sudoku suivant :

120056789690078215587291463352184697416937528978625341831542976269713854745869132

alors le résultat devra être :

120056789690078215587291463352184697416937528978625341831542976269713854745869132
123456789694378215587291463352184697416937528978625341831542976269713854745869132
124356789693478215587291463352184697416937528978625341831542976269713854745869132

6 Les fonctions

6.1 Les fonctions en assembleur

L'assembleur ne possède pas a proprement dit la notion de fonction. Cependant il est possible de découper en block que l'on appellera fonction. En assembleur, les fonctions :

- commencent par un label afin de pouvoir faire un branchement (jump) dessus.
- ont le premier argument dans le registre $a0, le second argument dans $a1, le troisième argument dans $a2.
- fournissent le permier résultat dans $v0, le second résultat dans $v1
- n'ont pas de contexte séparé. Les registres peuvent être vu comme une variable global, ainsi si votre fonction utilise le registre $t0 et qu'elle appelle une autre fonction qui utilise le même registre alors vous risquez d'avoir un bug. Notez qu'il est possible de sauvegarder des valeurs dans la pile.

6.2 Les fonctions fournies

Le fichier start.asm, qui se trouve au même emplacement que le sujet, contient les fonctions suivantes :

newLine saute une ligne à l'écran.

openfile ouvre un fichier dans le mode précisé par ses arguments et renvoie le descripteur de fichier.

closefile ferme un descripteur de fichier.

extractionValue extrait l'ensemble du sudoku à partir du fichier précisé en paramêtre.

changeArrayAsciiCode transforme les valeurs ascii obtenu à partir du fichier en entier.

printArray affiche à l'écran la grille du sudoku.

modulo fait le modulo (a%b).

Ces fonctions peuvent être utilisées comme bon vous semble.

6.3 Les fonctions attendues

Les fonctions suivantes devront être implémenté :

- colonne_n_valide qui test la validité de la n-ième colonne.
- ligne_n_valide qui test la validité de la n-ième ligne.
- carre_n_valide qui test la validité du n-ième carré.
- colonnes_valides qui test la validité de l'ensemble des colonnes.
- lignes_valides qui test la validité de l'ensemble des lignes.
- carres_valides qui test la validité de l'ensemble des carres.
- sudoku_valides qui test si le sudoku est valide.
- recherche_algorithme qui recherche l'ensemble des solutions du sudoku.

Votre programme devra utilisé ces fonctions afin de fournir la solution. D'autres fonctions pourront être rajouté afin d'améliorer la visibilité.

6.3.1 colonne_n_valide, ligne_n_valide, carre_n_valide

Ces fonctions testent si le sudoku accepte une solution malgrè un remplissage partiel. Le remplissage incrémental de la grille peut la rendre non-viable, il n'est alors pas intéressant de continuer à la remplir. D'un point de vue implémentation, ces fonctions doivent retourné faux lorsqu'il y a collision entre deux cases non-vides (contenant des valeurs de 1 à 9).

7 Structure de donnée

Actuellement dans le fichier start.asm, le sudoku est modélisé par un tableau à une dimension de 81 cases. Cette structure de donnée n'est pas imposée, vous pouvez la changer/améliorer sans aucune pénalité. Cependant, si la structure de la grille de sudoku venait à être modifié, rien ne garantit que les fonctions fournis continueront à fonctionner.

8 Rendu

Chaque étudiant devra rendre sa version du projet. Il est bien évident que toute tentative de triche sera pénalisé.

Vous recevrez par mail le jour du rendu (en décembre) et les modalités.

9 Annexes

9.1 FAQ

Vous trouverez une FAQ au même endroit que vous avez trouvé le sujet. Lorsque vous avez des questions, vérifiez que cette question n'est pas encore traité dans le fichier. Si la FAQ ne réponds pas totalement à votre question, envoyez moi un mail.

9.2 MIPS

La page wikipédia anglaise de MIPS donne l'ensemble des instructions et les registres a utilisé.
`http://en.wikipedia.org/wiki/MIPS_instruction_set#MIPS_assembly_language`