POLYTECH SORBONNE

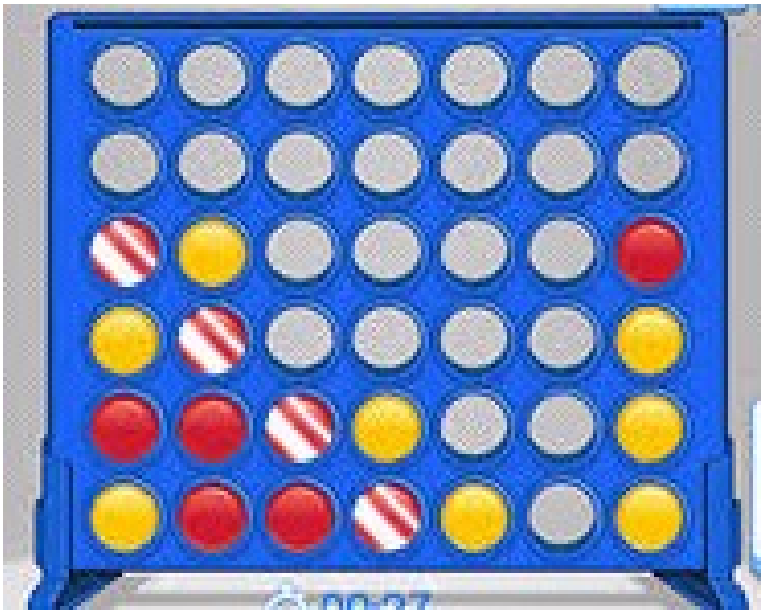
ARCHITECTURE DES ORDINATEURS

SORBONNE UNIVERSITÉ

RAPPORT

Projet MIPS - Puissance 4

Rabhi Aissam *Sennoun Naêl*



1 Introduction

Le jeu

Le puissance 4 est un jeu opposant deux joueurs à l'aide d'une grille de 42 cases (6 lignes, 7 colonnes). Au début du jeu le premier joueurs choisi une colonne, le jeton tombe alors tout en bas de celle ci. Le deuxième joueur insère à son tour son jeton dans la colonne de son choix. Et ainsi de suite jusqu'à obtenir une rangée de 4 jetons identiques.
Le premier joueur à aligner 4 jetons horizontalement, verticalement ou en diagonale remporte la partie.

Le projet

Le but de ce projet est de programmer ce jeu en langage assembleur MIPS. Nous avons commencé par coder le jeu en C pour avoir une idée des fonctions à mettre en place avec MIPS afin d'avoir un point de départ. Cependant la retranscription des structures de données du C vers l'assembleur donne un code et des fonctions un peu différentes.

2 Élaboration du code MIPS

2.1 Initialisation

La première difficulté rencontrée est l'initialisation de la grille, on passe d'un tableau à deux dimensions (6x7) à un tableau de 42 cases. Chacune de ces cases contient la valeurs 0, 1 ou 2 associée à une case vide, ou remplie par un des deux jetons. On initialise la grille en donant la valeur 0 aux 42 cases suivantes à l'aide de la fonction FCT_REFRESH_A2

2.2 Affichage

En entrée de la fonction affichage (FCT_PRIINT) nous avons le registre a2 qui contient notre "grille" de puissance 4 et on ne renvoie rien.
La grille de puissance 4 sera sous le format ci dessous (Ici lorsque nous parlons de a2 nous faisons référence à son adresse:

$$\begin{Vmatrix} a2+0 & a2+1 & \cdots & a2+6 \\ a2+7 & \ddots & \vdots & a2+13 \\ \vdots & \vdots & \ddots & \vdots \\ a2+35 & \cdots & \cdots & a2+41 \end{Vmatrix}$$

Si à l'adresse a2 se trouve un 0 alors la case est vide : ' '
Si il y a un 1 alors il se trouve le pion du joueur 1 : 'O'
Si il y a un 2 alors il se trouve le pion du joueur 2 : 'X'

On commence tout d'abbord à afficher la ligne 1 jusqu'à la 7ème en mettant en place un compteur et on décremente a2 de 42 à la fin pour le remettre à l'adresse de la première case pour le réutiliser plutard dans d'autre fonction.

2.3 Jouer

- Le programme ouvre une boîte de dialogue demandant à l'utilisateur de choisir un numéro de colonne, elle accepte seulement les valeurs comprises entre 1 et 7.

- On ajoute 34 à la colonne choisie, la somme des deux est ajoutée à l'adresse de début de la grille pour se retrouver sur la ligne du bas.

- On vérifie si la case choisie est vide : si oui on y place le pion, sinon on retire 7 à l'adresse de a2 pour se placer juste au dessus et répéter l'opération.

- On place la valeur [Adresse de grille +7] dans un registre permanent pour vérifier si une colonne est pleine. Dans ce cas le programme demande à l'utilisateur de choisir une autre colonne.

- On replace le registre a2 au début de la grille, et on laisse la main à la fonction d'affichage.

Vérifications

Les fonctions de vérification sont systématiquement appelées après la fonction jouer, à chaque fois que la grille est mise à jour.

2.3.1 Match nul

La fonction FCT_VERIF_DRAW est une fonction de parcours permettant de vérifier s'il y'a match nul. Elle prend en argument l'adresse a2 et retourne sa valeur dans $v1. Elle parcourt toutes les cases de la grille. Si une seule case est vide (valeur 0) alors elle renvoie 1, si non elle renvoie 0. Ce qui implique que toutes les cases sont pleines, et qu'il y a donc match nul.

2.3.2 Victoire verticale

La fonction FCT_CALCUL_JETONS permet, à partir du dernier jeton placé, de calculer la somme de jetons identiques alignés. Le nombre de jetons identiques sur la même colonne est calculé en ajoutant 7 à l'adresse de la case jouée. Par défaut les cases se trouvant à une adresse supérieure à a2 + 41 sont nulles, il n'y a donc pas de risque de "victoire cachée" du à une erreur des valeurs non visibles.

2.3.3 Victoire horizontale

La vérification de victoire horizontale compte le nombre de jetons à droite et gauche du dernier placé. On ajoute 1 à l'adresse dans un premier temps, puis en enlève 1 pour "revenir" dans le cas où le jetons se trouve au milieu d'une rangée de 4. Tout cela en faisant bien attention à arrêter de compter dès qu'on atteint les colonnes 1 ou 7. La principale difficulté se trouve dans cette partie. Les étiquettes sont laissés dans le code mais nous ne les avons pas intégrée au programma car elles ne fonctionnent pas comme on l'attend.

Principe : On a 7 étiquettes différentes, si on choisit la colonne 3 par exemple, la fonction récupère et compare les 2 cases a gauche du pions posé, si oui on compte à partir du jeton situé sur la colonne 1, vers la gauche donc, si on atteint 4 alors il' y'a victoire.

2.3.4 Victoire diagonale

La vérification de victoire diagonale utilise le même principe que la victoire horizontale, avec 4 cas différents cette fois, voici les options de parcours

- haute-droite -6
- basse-droite +8
- haute-gauche -8
- basse-gauche +6

Cette fonction prend en argument l'adresse $a2 du dernier jetons placé et renvoie le resultat (0 ou 1) dans le registre $s5 qui sera évalué dans le main.

2.4 Bruitage

On a décidé de rajouter une fonction bruitage pour éviter la monotonie du jeu.
Ca ne sert peut être pas a grand chose mais on a découvert comment faire du bruit avec un ordinateur grâce à un synthétiseur. L'important dans un jeu est son aspect visuel (sur lequel on n'est pas aller chercher plus loin), mais aussi son aspect auditif. On a du faire appel a un syscall MIDI out pour synthétiser les sons et un syscall sleep afin de rythmer les bruitages.

Conclusion

Ce projet nous a permis de mettre en oeuvre les connaissances acquise au cours du semestre en langage assembleur. On a ainsi pu approfondir ce que nous avons pu voir en cours de façon très théorique. Ce projet nous a également montré l'importance de l'utilisation des fonction en MIPS. Ainsi que l'importance de bien commenter son code, car en assembleur il est beaucoup plus long que dans n'importe quel langage de haut niveau. Après quelque jours d'absence ont a du mal à bien comprendre le code que nous avons nous même écrit s'il n'est pas bien commenté

Ressources

Pour le programme C :

https://zestedesavoir.com/tutoriels/755/le-langage-c-1/1043_aggregats-memoire-et-fichiers/5147_tp-un-puissance-4/