Projet d'assembleur SPARC n° 1 : Un petit morpion Groupe P1

Benoît Meister

November 26, 2004

Principe

Un morpion est un jeu qui se joue sur un plateau de 9 cases (représentées sur la figure 1). Chaque case du plateau peut être dans l'un des trois états suivants:

- marqué d'une croix : état n° 1,
- marqué d'un rond : état n° 2,
- vierge, c'est à dire que personne n'a encore marqué cette case : état n° 0.

L'état de chaque case peut être codé sur 2 bits, comme $1_{10} = 01_2$, $2_{10} = 10_2$, et $0_{10} = 00_2$. Il suffit donc de $9 \times 2 = 18$ bits pour coder l'état d'un plateau de jeu. Comme on n'est pas radins, on pourra utiliser un mot de 32 bits, qui a l'avantage de tenir exactement sur un registre, pour coder l'état du plateau de jeu. Les 18 bits de poids le plus faible de ce mot de 32 bits feront l'affaire.

Un jeu est fait de neuf tours (de 0 à 8) maximum. Le joueur n° 1 joue les tours pairs, et le joueur n° 2 les tours impairs.

1 Jeu à deux joueurs

Faites un jeu de morpion en assembleur SPARC, à l'aide :

a- d'une fonction **affiche** d'affichage,

b- d'une fonction **joue** qui, à partir d'un plateau p (un mot de 32 bits) , d'un numéro n de case (de 0 à 8) et d'un mot x de 2 bits ($\times \rightarrow 01$ ou $O \rightarrow 10$), modifie la n-ième case de p pour la mettre à la valeur x,

c- d'une fonction **vainqueur** qui indique quel joueur a gagné. Cette fonction doit renvoyer 0 si personne n'a gagné (pour le moment), 1 si le joueur 1 a gagné, et 2 si le joueur 2 a gagné.

Le jeu affichera le plateau, avant de demander un numéro de case au joueur dont c'est le tour. A la fin, on affichera le plateau final et le numéro du vainqueur.

0	1	2
3	4	5
6	7	8

Figure 1: Un plateau de morpion

2 Jeu contre l'ordinateur

Faites un jeu dont le joueur n ° 2 est l'ordinateur. Pour savoir quel coup jouer, l'ordinateur peut tester les différentes coups qu'il peut jouer et leur donner une *valeur*. Cette valeur doit refléter les chances de victoire de l'ordinateur. L'ordinateur choisira le coup (c'est à dire la case à cocher) qui aura la plus grande valeur, c'est à dire la plus grande chance de le mener à la victoire. Il nous faut donc faire deux nouvelles fonctions:

a- une fonction **chances_de_gagner** qui calcule la *valeur* associée à un coup, étant donné un certain état du plateau,

b- une fonction **meilleur_coup** qui parcourt tous les coups possibles, les évalue, et renvoie le meilleur coup, c'est à dire le numéro de case qui a la plus grande valeur.

L'algorithme de la fonction **chances_de_gagner** est laissé à votre imagination. Pour un coup donné, vous avez par exemple la possibilité :

- de tester toutes les combinaisons possibles à partir du coup considéré ($7 * 6 * 5$ au premier coup, ce qui est largement faisable). Cela peut éventuellement être fait de manière récursive.
- ou de compter le nombre paires de ronds contigus (il y a 12 configurations possibles). Naturellement, si vous trouvez que 3 ronds sont alignés, la valeur du coup devra être nettement supérieure à celle d'un plateau ayant le maximum de paires de ronds contigus. Remarquez qu'au maximum, on ne peut avoir que deux paires de ronds contigus sur un même plateau.
- ou de compter le nombre de paires de ronds qui pourront donner lieu à un alignement de 3 ronds. Il y a 16 possibilités si on ne tient pas compte des croix présentes.

Remarques génerales

Commentez votre code, de facon à expliquer clairement ce que vous faites, et à montrer que vous comprenez bien ce que vous faites.

Vous avez toute liberté d'écrire des fonctions supplémentaires à celles qui sont demandées.

Ce projet est à rendre au plus tard le

L'appel à **gcc -S** n'est pas autorisé pour générer votre programme ou même des morceaux de votre programme. L'utilisation de **gcc -S** sera en principe sanctionnée par une note nulle (au moins pour la fonction qui a été générée ainsi).