// Guilherme Costa Vieira Nº USP: 9790930
// Gabriel Kazuyuki Isomura Nº USP: 9793673
// Victor Chiaradia Gramuglia Araujo Nº USP: 9793756

Mudança no OPERANDO:

Agora o OPERANDO não é mais um int mas sim uma struct com um tipo e uma union, o que ocupa a union depende do tipo. Agora temos o tipo NUM que é simplesmente um número que sera usado, e o tipo ACAO que se refere a uma syscall.

Em nossa implementação não é necessário falar o tipo do argumento de uma instrução pois tentamos inferir o que é desejado, uma chamada de sistema sempre será do tipo ACAO, só faz sentido fazer aritméticas com números, logo sempre será uma operação entre dois números.

Arena:

Nossa arena é uma struct global que possui o número de exércitos no jogo (é possível ter tantos exércitos quanto for desejável), um contador de turnos, um vetor da struct army e uma matriz quadrada N.

A matriz é feita de nodes, cada node possui um ponteiro para o robô ocupante, um número que indica qual exército ocupa aquela região, um indicador se existe uma base (HQ) na posição e um número x de cristais.

Nossa struct army possui o nome do exército, o número total de robôs no início do exército (pode ser qualquer número), a posição do HQ do exército e um vetor com todos os robos do exercito.

As funções associadas a arena são relativamente simples. Como a implementação das funções desejadas não deve afetar a iteração do usuário final com o jogo houve uma mudança na função Sistema(). Em nossa implementação ela recebe o robô que a chamou e não o tipo de chamada de sistema que foi realizado (isso é empilhado).

Novas instruções:

-ATR: Foi decidido que a ATR iria só ser possível devolver informações sobre áreas adjacentes ao robô que a execução, isso faz com que o nosso jogo possua uma "fog of war" (https://en.wikipedia.org/wiki/Fog_of_war ver parte 3.1). Para o uso da instrução é necessário empilhar um número de 0 a 5 que se refere a um vizinho de seu robo.

-Chamadas de sistema:

-Na interpretação do grupo uma chamada de sistema é uma instrução que por um dado momento retira o controle do robô. Assim decidimos não criar uma instrução SYS e simplesmente criar instruções MOV, ATK, FETCH e DEPO que iriam retirar o controle da máquina e assim seriam chamadas de sistema. Como a ATR essas instruções funcionam usando um sistema de coordenadas relativas.

-MOV n: O robo se movimenta para a posição n do mapa. Se o robo decidir entrar em um "swamp" ele deve gastar um cristal para sair, se ele não possui cristais ele pode sair sem problemas.

-FETCH n: Tira um cristal da posição n e o coloca no robo.

-DEPO n: Coloca todos os cristais do robô na posição n.

-ATK n: Ataca a posição relativa ao robo dada por n (número entre 0 e 5).

No nosso jogo um ataque a uma robô com cristais rouba esses cristais porém se o atacado não possuir nenhum cristal ele é destruído. Temos "friendly fire on" (https://www.giantbomb.com/friendly-fire/3015-2243/) já que este é presente em jogos de RTS como StarCraft 2 e Age of Empires.