Relatório Projeto IAC

Descrição do projeto

O programa é composto por duas funções principais, a `geracacto` e a `atualizajogo`, que juntas vão atualizando os valores de `4000H` a `404FH` com valores aleatórios entre 0 e 4. Para executar as funções fornecemos o programa `projetoiac.as`. Diminuindo a clock speed do processador permite observar melhor as mudanças da tabela. Se quiser verificar individualmente, basta colocar um breakpoint nas linhas 16 e 41 (não ao mesmo tempo)

atualizajogo

A função `atualizajogo` é uma função que vai pegar o valor de endereço de memória `n` e movê-lo para o valor de memória `n-1`, realizando esta ação 80 vezes, como se trata de um deslocamento à esquerda de 80 posições de memória.

A função guarda em `R1` o valor do endereço de memória `404Fh`, e em `R2` o valor 4. De seguida chama a função `geracacto`, a qual gera um número de 0 a 4. Após isto `R2` passa a conter o valor 80.

Para realizar a primeira transição, a função `atualizajogo` vai guardar o valor do endereço de memória guardado em `R1` no registo `R4`, de seguida vai guardar o valor guardado em `R3` (retorno da função `geracacto`), e movê-lo para o endereço de memória em `R1`. Depois diminui em 1 `R2` (contador do número de deslocações realizadas), e `R1` (fica a guardar o valor de memória `n-1`), guarda o valor do endereço de memória em `R1` em `R3`, guarda o valor em `R4` no endereço de memória guardado em `R1`, e depois guarda o valor guardado em `R3` em `R4`, sendo que este loop vai ser repetido até que o valor guardado em `R2` seja 0.

Quando isso acontece, os valores de `R4` e `R7` vão ser restaurados, e a função reinicia-se a si mesma (pode facilmente ser alterado para voltar ao main, basta alterar `JMP atualizajogo` para `JMP R7`, fizemos esta implementação por efeitos de estética, é mais interessante ver a alteração rápida de números).

geracacto

A função `geracacto` gera um número aleatório a partir de uma semente, que neste caso escolhemos inicializar a 5, como no pseudocódigo que nos foi fornecido em Python.
Usamos o registo R2 para a altura máxima (neste caso 4), e o R3 para o retorno da função.
A implementação da função em si segue bastante rigorosamente o pseudo-código em Python, recorrendo a duas subrotinas, `ChangeBit` (que altera potencialmente qualquer bit de x) e `Funcao`, que devolve um 0 95% das vezes.
Apesar de tudo, existe uma ligeira mudança na implementação, preferimos usar `BR.N` e `29491` em vez de `BR.C` e `62258`, pois achámos que facilita a legibilidade e compreensão do código.