TÉCNICO LISBOA

Instituto Superior Técnico

Licenciatura em Engenharia Informática e de Computadores

Introdução à Arquitetura de Computadores

Projeto 2020/2021

Grupo 10 — João Vieira Antunes (99257), Rafael Serra e Oliveira (99311)

Introdução

O nosso projeto consiste numa adaptação do jogo "dino" (disponibilizado de forma icónica pelo *browser* Google Chrome) para o Pequeno Processador Pedagógico com *Pipeline* (P4) introduzido na cadeira, cumprindo todas as especificações definidas no enunciado fornecido.

Fizemos várias escolhas na implementação do enunciado, tal como a coloração da área de jogo em preto no branco, de forma idêntica ao jogo "dino" original e (após consultar o Prof. Baltasar Dinis) permitir que o salto do dinossauro seja ativado pela tecla "seta para cima" do teclado para além de através do botão embutido no P4, de modo a facilitar a jogabilidade no simulador do ex-bolseiro Dinis Madeira.

Estrutura e Fluxo

O código do nosso programa está dividido em cinco secções básicas: **cabeçalho** (definição de constantes e declaração de variáveis), **corpo de controlo** (centrais ao funcionamento do programa), **funções de jogo** (invocadas oportunamente para controlo do jogo em si), **funções auxiliares** (usadas pelo resto do código) e **código de interrupções** (executado quando estas são desencadeadas).

O fluxo básico do programa é ritmado pelo temporizador do P4 com uma duração de `100ms` e consiste no **corpo de controlo** verificar sempre que possível se existem interrupções geradas pelo temporizador por tratar, caso em que executa as **funções de jogo** para que o estado interno do programa seja atualizado.

Implementação

Cabeçalho: várias constantes são definidas, incluindo parâmetros de configuração e endereços da memória que representam vários portos para manipulação de periféricos. Seguem algumas diretivas a fim de indicar a reserva de espaço em memória para diversas variáveis globais contendo toda a informação de estado do programa. Note-se que cuidado deve ser tomado a alterar os seus valores iniciais pois estas são reinicializadas no início de cada partida, sendo portanto apenas garantidamente seguro alterar estes valores através das constantes definidas para o efeito. Destaca-se ainda a exceção das três últimas diretivas que reservam espaço para *strings* a serem apresentadas ao utilizador, nunca sendo alteradas.

Corpo de Controlo: conjunto de instruções (a primeira identificada pelo rótulo `MAIN`) que inicia uma partida (inicializando variáveis de estado e repondo o terminal), aguarda que o utilizador sinalize a sua intenção de começar a jogar (premindo o botão `0` no P4) e corre o ciclo de controlo principal do programa indefinidamente, invocando a subrotina `processtick` se houverem interrupções do *timer* pendentes.

Funções de Jogo: múltiplas subrotinas que implementam o jogo em si:

- **`processtick`:** chamada em cada *tick*, não recebe argumentos nem devolve nenhum valor. Invoca, por esta ordem, as funções **`atualizajogo`** (para atualizar o terreno e gerar um novo cacto), **`escrevejogo`** (que desenha o terreno no terminal) e **`dinossauro`** (que controla todo o movimento e interação da personagem *dino*). Incrementa a pontuação e chama a subrotina **`update7SD`**, que atualiza o valor escrito no mostrador de 7 segmentos.
- **`atualizajogo`:** desloca todos os elementos do terreno uma posição para a esquerda, adicionando um cacto obtido por invocação da função auxiliar **`geracacto`**. Recebe como argumentos o primeiro endereço do terreno e a sua dimensão, não devolvendo nada. *(*)*
- **`geracacto`:** devolve um número pseudo-aleatório entre `0` e o seu argumento, com base na semente dada pela variável global **`x`** (que esta função altera em cada invocação). *(*)*
- **`escrevejogo`:** recebe como argumentos a primeira posição do terreno e a sua dimensão; não devolve nenhum valor. Desenha toda a área de jogo (com 2 `*` **`ALTURA_MAX`** de altura).
- **`dinossauro`:** não recebe argumentos nem devolve nada. Se houver alguma tecla para ler no teclado, chama **`verifsalto`**. Lê **`saltoDino`** e **`direcaoDino`** e sobe/desce uma linha a personagem *dino* (por **`linhaDino`**) se necessário, invocando de seguida **`escrevedino`** para que este seja representado no terminal. Se houver alguma colisão entre o *dino* e um cacto, termina a partida e aguarda que o utilizador comece uma nova, transferindo nesse caso o controlo de execução para o rótulo **`MAIN`** (corpo de controlo do programa).
- **`verifsalto`:** não recebe argumentos. Se a última tecla premida no terminal for `TECLA_SALTO` (seta para cima), escreve 1 para **`saltoDino`**. Não devolve nada para mais fácil integração com o método normal de salto (botão seta para cima no P4).
- **`limpa`:** recebe a primeira posição do terreno e a sua dimensão; não devolve nada. Remove todos os cactos do vetor, apaga tudo no terminal e faz **`pontuacao`** `= 0`.
- **`escrevedino`:** desenha a personagem *dino* no seu argumento. Não devolve nada.

(*) Estas funções estão documentadas em mais detalhe no documento submetido na Parte 1 do projeto.

Funções Auxiliares: subrotinas utilizadas pelo resto do programa sem relação direta com o jogo:

- **`update7SD`:** atualiza o *display* de 7 segmentos com o seu único argumento, convertendo-o para este ser mostrado como um número decimal e não hexadecimal. Usa **`divint`** para a conversão e divide o mostrador em duas partes para facilitar a implementação, devido à numeração pouco intuitiva dos portos. Não devolve nada.
- **`divint`:** recebe um número e devolve a divisão inteira (por **R3**) e o seu resto (pela pilha).
- **`escrevecentrado`:** recebe a primeira posição de uma cadeia de caracteres e uma linha na forma `XY00`, escrevendo essa *string* no centro do terminal (nessa linha). Não devolve nada.
- **`strlen`:** recebe a primeira posição de uma cadeia de caracteres e devolve o seu tamanho.

Código de Interrupções: instruções devidamente posicionadas na memória (através de diretivas `ORIG`). No caso de interrupções devido ao botão "seta para cima" no P4, atualiza **`saltoDino`** para 1. Para interrupções relativas ao temporizador, invoca a função auxiliar especial **`timerinterrupt`** que, para além do contemplado na convenção de chamada de funções, tem que preservar os registos **R1** a **R3**, visto não haver espaço suficiente na memória de instruções para o fazer dentro do *interrupt handler* em si. Esta função **`timerinterrupt`** reativa o temporizador e incrementa o número de interrupções pendentes através de **`timerPending`**, acedida pelo corpo de controlo. Não recebe argumentos nem devolve nada.