Chuva de Meteoros

Projeto de Introdução à Arquitetura de Computadores

Realizado por:

Alexandre Telo Antunes Duarte (ist1102948)
David Faia Nunes (ist1102890)
Érik Stelvin da Silva Bianchi (ist1103580)

(Grupo 44)

Sob Supervisão de:

Prof. Fernando Ramos
Prof. Hugo Miguel Tavares Matias

INSTITUTO SUPERIOR TÉCNICO

JUNHO 2022

Tabela de Conteúdos:

1. Introdução

Visão Global

Este relatório visa desconstruir o resultado do trabalho realizado no desenvolvimento do projeto da cadeira de Introdução à Arquitetura de Computadores, no qual era requerido o desenvolvimento dum videojogo embutido numa máquina virtual cujo processador é o PEPE-16 - Processador Especial Para Ensino de 16 bits. Para este fim, foi necessário realizar exclusivamente programação de baixo nível com Assembly próprio do processador mencionado, dando uma melhor compreensão da arquitetura básica dum computador e dos comandos mais simples que a este podemos impor, sendo que cada linha programada em Assembly representa uma única ação, ao contrário de linhas em linguagens de programação de alto-nível.

Motivação

Sendo uma linguagem de baixo nível, uma linguagem Assembly tem uma correspondência muito forte com as instruções em código máquina específicas do processador em questão. Apesar da prevalência das linguagens de alto nível utilizadas no desenvolvimento de quase todo o software, a importância das linguagens Assembly no mundo de hoje não pode ser subestimada. Um programador pode ainda ganhar muito ao aprender as ideias transversais a estas linguagens e a sua implementação. Com Assembly é permitida a manipulação direta de hardware, assim como a abordagem de problemas críticos que dizem respeito a performance e o acesso a instruções especiais para processadores. Deste modo, este projeto foi um ótimo exercício para nos aumentar a bagagem de recursos, adicionando uma ferramenta tão importante para qualquer engenheiro informático.

Objetivo

O objetivo da versão final do projeto era um jogo que corresse numa máquina virtual com o PEPE-16:

1. Tratamento do input do teclado da máquina virtual.
2. Mecanismos que permitam que o sinal captado de cada tecla seja correspondido com uma determinada ação no ecrã, passando para este fim pela animação dos sprites do jogo - a personagem do jogador, os meteoros bons, os meteoros maus e o tiro - e pelo controlo de três displays de sete segmentos.
3. Tratamento das mudanças do valor da vida/energia da personagem do jogador, e sua conversão de hexadecimal (intrínseco à forma como o computador opera e,

portanto, à própria linguagem de programação usada) para decimal (muito mais natural para o utilizador).

4. Gasto periódico de energia.
5. Verificação de colisões entre sprites do jogo.

2. Implementação

Componente Multimédia

Toda a realização deste projeto foi possibilitada pela providência do simulador do PEPE-16, no qual uma das partes integrantes da máquina virtual é o MediaCenter. Este módulo multimédia inclui um ecrã de 32 x 63 pixels e a capacidade de controlar a reprodução de ficheiros de áudio.

O nosso grupo selecionou e editou imagens para as várias necessidades do jogo: um menu inicial, um menu de fim de jogo, um menu de pausa e uma imagem de fundo para o decorrer normal da parte jogável. Para além disto, também foram selecionados efeitos sonoros para o tiro, para o pausar/despausar do jogo, para o início/reinício do jogo, entre outros. Todas estas componentes audiovisuais foram controladas a partir de instruções próprias do PEPE-16, listadas no seu manual.

Embora desenhados pixel-a-pixel diretamente no ecrã do MediaCenter, os sprites do jogo foram previamente concebidos numa folha de Excel, na tentativa de obter um Pikachu™, um pato e uma Pokéball™ como personagem jogável, “meteoro bom” e “meteoro mau”, respetivamente.

Relógios e Interrupções

O circuito empregado no projeto contém uma secção de componentes crucial para a implementação utilizada: os relógios. Com estes mecanismos foi possível a realização ciclicamente temporizada de certas ações: a diminuição do valor da vida da personagem do jogador a cada 3 segundos, a descida contínua dos meteoros e a subida contínua dos tiros (mencionados como “mísseis”, no enunciado do projeto) do jogador. Porém, para permitir realmente este acionamento contínuo foi necessário recorrer a interrupções, uma vez que estas permitem ao hardware realizar ações que requerem atenção sensível ao tempo.

Decidimos criar variáveis com valores booleanos que indicam se num relógio foi percorrido algum ciclo cujas supostas consequências ainda não foram verificadas. Ou seja, quando o relógio percorre um ciclo a variável registra o valor 1(True) e, apenas após as ações que o relógio deve acionar serem realizadas, o valor retorna para 0(False). Este mecanismo explica a existência de “meteoros_interrupt”, “tiros_interrupt” e de “vida_interrupt”.

Valor da Vida

O valor da vida (mencionado como valor da “energia”, no enunciado do projeto) é o único valor divulgado explicitamente ao jogador, ainda mais, em tempo real. Para permitir isto, foi desenvolvido um processo “display” que permite a exibição do valor da vida guardado num registro específico. Este valor é, por sua parte, tratado através da rotina “aumenta_vida” que permite a sua diminuição a cada ciclo do “relógio energia” e o seu aumento - cada vez que o jogador apanha um meteoro bom ou destroi um meteoro mau.

Sendo o sistema binário intrínseco à forma como o computador opera, a linguagem Assembly trabalha maioritariamente com números na forma hexadecimal. Por outro lado, sendo um jogo um pedaço de software feito para o utilizador comum, a conversão deste valor hexadecimal para decimal era mandatória. Foi portanto criada a rotina “converte_hex_dec” que visa tratar desta tarefa.

Teclado

O funcionamento do jogo como experiência interativa coagiu-nos a implementar um processo denominado “teclado”. Deste modo, todas as teclas funcionam, no sentido em que, ao analisar todas as combinações linha-coluna, o processo cria um output que identifica qual a tecla a ser premida, e, caso nenhuma esteja a ser premida, devolve 0. Isto permite o desencadeamento das ações certas no momento em que o jogador pretende. Para mais tarde ser possível identificar as teclas sem fazer referência a “números mágicos”, foram criadas constantes que definem as teclas relevantes ao jogo.

Uma das secções cuja importância é mais surpreendente do processo “teclado” é o “espera_nao_tecla” que permite que certas teclas não tenham output contínuo. Deste modo, cada tecla cuja verificação esteja associada ao “espera_nao_tecla” tem outputs discretos, um por clique.

Personagem do Jogador

A personagem do jogador já esteticamente concebida foi definida numa tabela que contém a sua largura (tendo uma forma quadrada, a largura e a altura são equivalentes, e, portanto, foi evitada a dupla referência ao valor na tabela) e os valores ARGB de cada pixel do sprite (da esquerda para a direita, de cima para baixo). Este processo foi repetido para cada sprite.

A manipulação dos sprites dependeu também do desenvolvimento de rotinas como “desenha_boneco” e “apaga_boneco”, autoexplicativas. Qualquer animação de sprites no nosso programa faz uso destas duas rotinas: apaga-se o boneco nas suas coordenadas

atuais, seguido duma mudança das coordenadas(linha e coluna) e do traçado do boneco na posição nova - com base no tamanho, cores e ordem de cores definidos na tabela do boneco a ser tratado.

No caso específico da personagem do jogador, foi criado um processo que permitir ao utilizador o controlo do seu movimento lateral através das teclas 0 (cujo output no teclado foi registrado como constante TECLA_ESQUERDA) e 2 (cujo output no teclado foi registrado como constante TECLA_DIREITA), analisando se alguma destas ações é requerida e, se sim, realizá-la. A rotina “testa_limites” verifica se a personagem do jogador se encontra nos limites laterais do ecrã, para impedir que se ultrapasse estes.

Meteoros

Tal como para a personagem jogável, os meteoros também têm associado um processo. Este processo tem código responsável pelo movimento descendente dos sprites do meteoro, por apagar o meteoro quando este é destruído pelo tiro do jogador e por variar a tabela utilizada para o traçado do meteoro entre a tabela do “meteoro bom” e a tabela do “meteoro mau”. Esta escolha tinha de ser aleatória porém com probabilidade fixa (cada meteoro que surge tem 25% de probabilidade de ser um “meteoro bom” e 75% de ser um “meteoro mau”) portanto elaborámos o “escolhe_meteoro” que, a partir dum número aleatório criado pela deslocação de bits dum número associado ao teclado, faz essa escolha pseudo-aleatória.

Tiros

Os tiros do jogador têm uma trajetória ascendente até atingirem um qualquer meteoro ou até chegarem ao alcance máximo (definido na constante “ALCANCE_TIRO”). Este movimento foi implementado de forma análoga à do meteoro.

Colisões

Para verificar as colisões entre os tiros ou o jogador com os meteoros, formámos a rotina “verifica_colisao” que: primeiro verifica se houve colisão do meteoro com o jogador; se não, verifica se houve colisão com o tiro, mas, se sim, prossegue a verificar se o meteoro é bom ou mau; caso nos encontremos neste último caso, são realizadas as ações necessárias dependendo da natureza do meteoro.

Menus

Decidimos implementar a passagem entre menus(menu de pausa, menu inicial, menu de jogo, etc.) através da criação de variáveis cujo valor é booleano e indica se certo menu é o que deve estar exibido. Deste modo, cada processo (do boneco, do meteoro, do tiro, etc.) tem uma secção onde as medidas certas são tomadas, estando detetado um certo ecrã cuja exibição é suposta. Por exemplo, se o ecrã de pausa for exibido, não é permitido o movimento da personagem jogável.

O MediaCenter tem vários ecrãs o que possibilitou que cada sprite esteja desenhado num ecrã próprio. Deste modo, quando estamos num menu onde o jogo propriamente dito está parado, todos os ecrãs são escondidos (com auxílio da rotina “esconder_ecras”), quando retornamos à parte jogável todos os ecrãs são trazidos de volta (com auxílio da rotina “mostrar_ecras”).

3. Bugs Encontrados e Suas Soluções

Conversão Hexadecimal para Decimal

Aquando da conversão de valores em base hexadecimal para valores em base decimal, surgiram *outputs* decimais com letras (sendo que em base decimal não existem dígitos assim). Isto deveu-se à implementação utilizada que resultava na acumulação esporádica de unidades duma certa casa hexadecimal na casa posterior. Ou seja, em vez de 16(em hexadecimal) ser convertido em 22(em decimal), era convertido em 1C, porém o próprio C(em hexadecimal) representa 12(em decimal), portanto criou-se um processo que a cada dígito do output da conversão que seja superior a 9 (ou seja, uma letra) retira-se A(correspondente a 10 em decimal) e soma se uma unidade à casa hexadecimal anterior. Logo, no nosso exemplo, ficaria 1C → 10 + 12 = 22.

Testagem de Limites

O nosso grupo denotou ao testar o movimento da personagem jogável que esta atravessava o limite direito do ecrã, embora o limite esquerdo estivesse a ser bem verificado, impedindo este de ser atravessado. Isto deveu-se a estarmos a comparar a coluna esquerda do sprite com o limite. Facilmente este bug foi corrigido, passando a testar com a coluna direita da personagem.

Movimento dos Meteoros

Tal como anteriormente descrito, os meteoros devem todos movimentar-se ao ritmo do “relógio meteoros”. Porém, deparámo-nos com um bug ao reparar que os meteoros estavam a cair sem sincronia. Isto deveu-se ao facto de estarem ligados ao mesmo interruptor e foi solucionado criando uma tabela com variáveis para cada meteoro que aparece no ecrã.