

Adventures of lolo

João Pedro Felício Pereira

Gustavo Lopez Dezan

Universidade de Brasília

Departamento de Ciência da Computação, Brasília-DF

Resumo: Esse projeto teve como objetivo desenvolver um game em **Assembly RISC-V**, interpretado pelo simulador RARS, utilizando de ferramentas como **Bitmap Display**, **Keyboard MMIO** e **interface MIDI**. Esse projeto tem como base o game *Adventures of lolo*.

1 Introdução

Adventures of lolo foi lançado em 1989 pela **HAL laboratory**, originalmente para o console **Nintendo Entertainment System(NES)** e posteriormente foi relançado na **Virtual Console da Wii** e posteriormente na **Wii U** e **Nintendo 3DS**.



Figure 1: Capa do jogo para NES

2 Metodologia

Para criar esse jogo utilizamos o simulador da **ISA** do processador **Assembly RISC-V**, **RARS**, nele utilizamos ferramentas como **Bitmap Display**, **Keyboard MMIO** e **interface MIDI** para fazer uma integração do jogo com os dispositivos de saída e entrada.

2.1 Movimentação e interação com os elementos

Movimento e interação com os elementos No uso de macros, o grupo buscou produzi-las de forma modular, de modo que elas se encaixassem em diversas situações, o que dificultou. Os primeiros estágios da programação para facilitar os estágios mais avançados. Para a movimentação do personagem Lolo, foram criadas funções que o deslocavam de acordo com um plano cartesiano implementado também por código. Além disso, para checar por colisões o personagem encontra sua posição na matriz e busca pelos valores dos elementos correspondentes. Os códigos de cada ele-

mento do jogo foram pensados como objetos, com suas "funções" entrelaçadas para que fosse possível alcançar os resultados desejados, o que pode ser notado no momento em que o Lolo colide com um coração, onde ele detecta a posição do objeto colidido e chama a função de apagar um sprite na posição específica do coração. De modo semelhante, o poder disparado por Lolo checa inicialmente por sua própria existência, de modo que caso ela seja verdadeira, ele se propaga pelo mapa buscando por colisões e, no caso de ele acertar um "objeto" inimigo, ele chama o código que passará para o inimigo o estado de **LIVE = FALSE**. O inimigo, por sua vez, só poderá se redesenhar no mapa a cada loop do jogo caso ele possua seu estado de live como true.

2.2 Manipulação de imagens

Foi necessário utilizar programas externos para construir as imagens dos elementos do jogo e o mapa, como o **paint.net** e **piskel**. Após isso, utilizamos um programa para converter o arquivo de imagem para **.data**. Sempre reparando em detalhes como deixar os arquivos em dimensões, cujo o **x** e **y** são pares, pois como o **RARS** trabalha em **byte addressing** facilitaria o cálculo e o funcionamento mais dinâmico do jogo. Para a criação do mapa, foi criado um **tile-set** com todos os objetos estáticos do jogo, e a partir disso o mapa foi montado como um conjunto de blocos de **16x16 pixels**. Para as imagens que possuem mais de um estado, ou uma animação completa (que de modo geral são os sprites), utilizou-se uma imagem com todas as posições que o sprite poderia assumir ordenadas por suas direções, desse modo, foi criado um script que itera por essas posições encontrando a direção do sprite e o último frame que ele assumiu. Para objetos que o Lolo "consumiria", eles foram gerados como fazendo parte da imagem "mapa" e foram printados fechados por cima dela.

2.3 Trilha Sonora

Foi utilizada duas macros, uma para efeitos sonoros e outra para a trilha principal. A macro de **sound effects** recebe quatro parâmetros, todos são registradores de argumento, **a1**, **a2**, **a3** e **a4**, correspondente a nota, duração, instrumento e volume, respectivamente. A macro de **play music** é um pouco mais complexa por ter várias notas e a duração para cada uma delas. Sendo assim, tem de ser calculado a quantidade de notas para servir como comparação com o contador para saber quando parar de tocar tal música. A **interface MIDI** conta, também, com as syscalls **31** e **33**, **MidOut** e **MidOutSync** que são os serviços que vão nos auxiliar para tocar a trilha.

3 Resultados Obtidos

3.1 O jogo

O resultado obtido foi bem satisfatório, apesar do tempo ter sido um grande obstáculo, fizemos um game com efeitos sonoros, animação, tratamento de colisões e outras funções, e isso nos deixou bastante animados, pois nunca tínhamos feito um projeto em grupo dessa magnitude.

3.2 Problemas

Programar em Assembly não é algo muito atrativo. Durante o programa, o computador de um dos integrantes do grupo sofreu sérios problemas mecânicos por causa da execução de uma música. Além disso, problemas como "**branch target**" acabou nos tomando bastante tempo e não conseguimos implantar diversas funções no código por causa desse erro. Somando isso com as vezes que o simulador simplesmente parava de executar um programa que ele executava com facilidade anteriormente.

4 Conclusão

Esse projeto foi de imensa importância para a nossa aprendizagem em assembly. No início, o maior desafio foi o desânimo, pois não

References

MIDI number	Note name	Keyboard	Frequency Hz		Period ms
21	22	A0	27.500	36.36	
23	24	B0	30.868	32.40	
24	25	C1	32.703	30.58	
26	27	D1	36.708	27.24	
28	29	E1	41.203	24.27	
29	30	F1	43.654	22.91	
31	32	G1	48.999	20.41	
33	34	A1	55.000	18.18	
35	36	B1	61.735	16.20	
36	37	C2	65.406	15.29	
38	39	D2	73.416	13.62	
40	41	E2	82.407	12.13	
41	42	F2	87.307	11.45	
43	44	G2	97.999	10.20	
45	46	A2	110.00	9.091	
47	48	B2	123.47	8.099	
48	49	C3	130.81	7.645	
50	51	D3	146.83	6.811	
52	53	E3	164.81	6.068	
53	54	F3	174.61	5.727	
55	56	G3	196.00	5.102	
57	58	A3	220.00	4.545	
59	60	B3	246.94	4.050	
60	61	C4	261.63	3.822	
62	63	D4	293.67	3.405	
64	64	E4	329.63	3.034	
65	66	F4	349.23	2.865	
67	68	G4	392.00	2.551	
69	70	A4	440.00	2.273	
71	71	B4	493.88	2.025	
72	73	C5	523.25	1.910	
74	75	D5	587.33	1.703	
76	76	E5	659.26	1.517	
77	78	F5	698.46	1.432	
79	80	G5	783.99	1.276	
81	82	A5	880.00	1.136	
83	83	B5	987.77	1.012	
84	85	C6	1046.5	0.9556	
86	86	D6	1174.7	0.8513	
88	87	E6	1318.5	0.7584	
89	88	F6	1396.9	0.7159	
91	89	G6	1568.0	0.6378	
93	92	A6	1760.0	0.5682	
95	94	B6	1975.5	0.5062	
96	96	C7	2093.0	0.4778	
98	97	D7	2349.3	0.4257	
100	98	E7	2637.0	0.3792	
101	102	F7	2793.0	0.3580	
103	104	G7	3136.0	0.3189	
105	106	A7	3520.0	0.2841	
107		B7	3951.1	0.2531	
108		C8	4186.0	0.2389	

