Universidade de São Paulo - ICMC
Bacharelado em Ciências de Computação
SSC0147 - Tópicos Especiais em Sistemas de Computação I
Professores: Dr. Francisco José Monaco e Dr. Cláudio Fabiano M. Toledo
Ministrantes: Leonardo T. Pereira, Gustavo Ceccon e Rafael Miranda

Arquitetura Nintendo 64

Rafael Monteiro	Nº USP: 9293095
William Ferreira	Nº USP: 9293421
Giovanna Guimarães	Nº USP: 9293693
Lucas Alexandre Soares	Nº USP: 9293265

São Carlos - SP

21 de novembro de 2016

Capa feita por Marcos Cesar Ribeiro de Camargo

Sumário

1 Introdução

O Nintendo 64 foi o terceiro console de mesa da empresa japonesa Nintendo lançado para o mercado internacional e sucessor do Super Famicon (no Japão) ou SNES (Super Nintendo Entertainment System, no exterior). O nome lhe foi dado devido à sua CPU de 64-bits pelo game designer Shigesato Itoi. Lançado no Japão em junho de 1996, o N64 chegou ao Brasil apenas em dezembro de 1997 e disputou mercado com os demais consoles da quinta geração: Sony Playstation e Sega Saturn. Apesar de seu grande sucesso na América do Norte e na Europa, principalmente, o Nintendo 64 foi ultrapassado em vendas pelo Playstation.

O console chegou ao fim de suas vendas com poucos jogos em sua biblioteca devido à falta de apoio de desenvolvedoras, uma vez que a produção de cartuchos, utilizados pelo Nintendo 64, era extremamente mais complicada em relação aos CDs utilizados pelos concorrentes. Apesar disso, o 64 continua como um dos video games mais reconhecidos da história, tendo jogos como Super Mario 64, The Legend of Zelda: Ocarina of Time e GoldenEye 007, que ainda causam impacto na indústria dos games.

Nesta monografia serão apresentados conceitos relativos à arquitetura desse icônico console da Nintendo, buscando uma melhor compreensão sobre como eram desenvolvidos os jogos eletrônicos numa época em que a tecnologia computacional, como um todo, era muito mais limitada.

2 Técnicas e Tecnologias

2.1 CPU

O processador do Nintedo 64 é o **NEC MIPS R4300i**, uma versão mais barata do **MIPS 4400**. É um processador RISC com 5 estágios de pipeline operando a 93.75MHz de clock, com uma unidade de ponto flutuante integrada e 24KB de cache L1 (sendo 16KB para instruções e 8KB para dados), que provê aproximadamente 20% de incremento de performance. É um processador escalar no sentido de que as operações são feitas uma a uma, em oposição à processadores vetoriais, como utilizados normalmente em GPU's. Ele possui modo de operação de 32 e 64 bits porém poucos jogos aproveitavam das instruções 64 bits uma vez que as de 32 bits gastam menos memória e são mais rápidas de serem executadas. A interface externa é feita por um barramento de 32 bits, o que desencoraja ainda mais o uso de instruções 64 bits.

O chip conta com uma tabela de tradução de endereços chamada de Translation Lookaside Buffer (TLB). Esta tabela possui 32 entradas e cada entrada mapeia um endereço virtual na página de dois endereços físicos. Os endereços das páginas são variáveis e podem ser manejados independentemente de cada entrada. O endereçamento de memória é feito em formato 32 bits, mas a CPU é incapaz de realizar acesso direto à memória, precisando primeiro passar pelo RCP antes de conseguir enviar ou buscar dados da RAM.

Os principais papéis da CPU eram, comumente, processar lógica do jogo, algumas vezes uma parte do processamento de áudio, IA e gerenciamento de *input*, enquanto o RCP realiza os demais processamentos.

2.2 Reality Coprocessor

O *Reality Coprocessor* (RCP) é um processador paralelo à CPU, que opera a 62.5 MHz, realizando o papel de controlador de memória. Pelo fato de que o acesso à memória é controlado pelo RCP, os outros componentes são incapazes de acessá-la diretamente, fazendo com que os dados sempre passem pelo RCP antes de chegarem ao seu destino.

Os módulos de entrada e saída também ficam contidos neste coprocessa-

dor. O chip possui portas de saída de áudio e vídeo, entrada e saída de dados com a CPU, memória interna, Game Pak (cartucho) e entrada de dados serial (típicamente os controles) e paralela (típicamente os *game paks*). Este chip é basicamente o comunicador universal do console, todos os dados passam por ele. Sejam dados da CPU, dados de I/O ou acesso à memória, seja o próprio chip que está acessando ou algum outro módulo que requer dados da memória.

Todo processamento gráfico e de áudio do Nintendo 64 é realizado pelo RCP. Isto é possível graças à duas unidades importantes do chip: o *Reality Drawing Processor* (RDP) e *Reality Signal Processor* (RSP). O barramento de comunicação entre essas duas unidades possui 128bits e é capaz de transmitir até 1GB/s de dados (o que era uma alta capacidade de processamento para a época), permitindo a renderização de gráficos 3D de boa qualidade em tempo real.

2.2.1 Reality Display Processor

Responsável pelas operações a nível de pixel, o RDP é encarregado da rasterização de objetos e de manipular o Z-Buffer do console para gráficos 3D. Ele é capaz de renderizar geometria com profundidade, processar sombras, aplicar *Anti-Aliasing* sobre as texturas, aplicar texturas e calcular transparência de pixels (canal alpha). Para isso, o RDP possuí um *rasterizador*, uma unidade de textura, uma unidade de filtro de texturas, um combinador de cor, um *blender* e uma interface de memória. Os dados processados são enviados ao *frame buffer* para depois serem imprimidos na tela.

O RDP possui apenas 4KB de memória de texturas, permitindo armazenamento interno de até 8 texturas a qualquer momento. Em comparação, o Playstation possui 1MB de memória de vídeo dedicada (tamanho variável para texturas), permitindo texturas mais realistas em oposição às texturas típicamente mais cartoonistas e de propósito geral utilizadas pelo jogos de Nintendo 64. Além disso, se for utilizado a técnica de *mipmapping* para otimizar efeitos de *aliasing*, a memória de texturas pode ser reduzido até pela metade, pois *mipmapping* consiste em armazenar várias cópias da mesma textura porém com dimensões diferentes, com diferentes níveis de detalhes. Assim é possível inter-

polar entre duas imagens para obter o resultado com maior qualidade possível com pouco tempo de renderização, porém com alto custo de memória.

2.2.2 Reality Signal Processor

O RSP é encarregado de realizar cálculos 3D de áudio e vídeo. Ele é capaz de realizar *Edge Anti-Aliasing*, mapeamento de texturas, cálculo de profundidade para renderização (Z-Buffering), cálculos de luz para o mapeamento de iluminação, *culling*, transforms e posicionamento de triângulos.

O RSP também possui 4KB de memória de dados e 4KB de memória de instruções, permitindo micro-programação de até 1024 instruções. Estas instruções são chamadas de microcódigo e permitem que o processador seja mais flexível, podendo criar ou otimizar efeitos.

2.3 Memória

O Nintendo 64 foi um dos primeiros consoles a implementar uma arquitetura de memória unificada (UMA, em inglês). Ao invés de haver um banco de memória dedicado à cada tarefa, como áudio ou gráficos, há apenas um banco universal de 4MB de RAM dinâmica. Nesta arquitetura, o tempo de acesso à memória é independente de quem está acessando e a quantidade de memória a ser usada para cada trabalho pode variar de acordo com o necessário para o jogo, tornando-a mais flexível. A memória pode ser expandida para 8MB com o uso do *Expansion Pak*.

A memória utilizada pelo console era uma **RDRAM** da *Rambus*, com barramento de dados de 9 bits operando a 500MHz, provendo uma taxa de transferência de 562.5MB/s. Entretanto, esta memória possui uma latência de acesso muito alta, e combinado com o fato do console não ser capaz de realizar acesso direto à memória, o custo de acessar a memória é extremamente alto. Dessa forma, o impacto positivo da alta taxa de transferência da memória é fortemente prejudicado.

O design do chip de memória se torna mais simples por ter um barramento pequeno, de apenas 9 bits, tornando esta memória mais barata. Este barramento também permitiu aos desenvolvedores usar técnicas de design mais

simples no console como um todo, deixando-o mais barato.

3 Desenvolvimento

3.1 Gráficos

O processamento gráfico começa no RSP, um processador vetorial projetado para realizar cálculos 3D com alta performance. Ele é responsável pela maior parte do pré-processamento de vídeo e parte do áudio no Nintendo 64.

A aplicação do jogo deve compilar uma *Display List* com os objetos que precisam ser processados e enviá-la ao RSP. Pode-se aninhar múltiplas *Display Lists*, com profundidade máxima de 10, para tentar obter o tamanho ideal de 32 elementos por lista e otimizar a etapa de renderização.

Cada *Display List* é composta por objetos e *tasks*. As *tasks* descrevem o que precisa ser processado em cada objeto, carregando os devidos segmentos de microcódigo a serem executados.

O pré-processamento das listas conta com 6 etapas: pilha de matrizes, conversão de geometria 3D, cálculo dos vetores de luz, cortar objetos fora da visão, mapeamento de iluminação e reflexão para aplicação de shaders e preparação de polígonos e linhas para a rasterização.

A pilha de matrizes serve para que objetos relacionados sejam processados relativamente entre si, ao invés de processar em coordenadas globais. Isso quer dizer que se há um objeto que deve movimentar outros objetos quando movido, por exemplo um chassi de carro que deve mover as rodas, essa pilha de matrizes pode calcular a transformada relativa para todas as outras partes juntas, acelerando o processo todo. As outras etapas são todas para a criação de uma imagem 3D: projeção de texturas em modelos 3D, vetorização da luz para verificar a incidência de luz, aplicação da física de luz com reflexão e shaders para efeitos especiais e desenhar os polígonos na tela.

Para o processamento 3D, o RSP conta com uma unidade de vértices capaz de realizar cálculos com até 16 vértices que podem ser carregados quantas vezes quiser consecutivamente e podem começar em qualquer posição. Caso o processamento da pilha de matriz e da iluminação já estejam terminados, seus resultados são automáticamente aplicados aos vértices afetados.

Na etapa de aplicação de iluminação, é possível aplicar efeito de névoa para

evitar de objetos aparecerem e sumirem de repente durante o carregamento da cena. Assim os objetos aparecem de forma mais quando entram no cone de visão do jogador.

Na inicialização, o RSP executa um microcódigo de boot padrão que carrega o primeiro elemento da lista, a *task* e o microcódigo aclopados ao elemento e, então, começa o pré-processamento deste elemento.

Todos os endereços da lista são endereços segmentados e podem ser mapeados utilizando macros fornecidos pelo sistema. Este processo de mapeamento é importante pois a aplicação deve informar ao RSP os endereços necessários para carregar os dados necessários. Também há um cache de 16 segmentos e seus endereços base, utilizando *Translation Lookaside Buffer*, para acessar a memória de dados mais rapidamente. Vale notar que essa cache é alinhada em 32 comandos, então os tamanhos otimizados são multiplos de 32 objetos por lista.

É possível acontecer um problema de concorrência caso a CPU e o RSP estejam manipulando os mesmos dados, pois há uma cache implementada com mecanismo de *write-back*. Dessa forma, alterações feitas nos dados pela CPU não são escritos na memória imediatamente, então é pode acontecer de o RSP ter que esperar a CPU atualizar a memória principal para poder recuperar o dado necessário.

Ao fim da conversão, todos os dados do RSP são transferidos para o RDP para rasterizar triângulos e retângulos e produzir pixels de alta qualidade, texturizados, com aplicações de *antialising* (redução de serrilhamento) e *Z-buffer* (manipulação das camadas de profundidade de uma imagem 3D).

3.1.1 Blocos de *Pipeline* do RDP

O RDP é constituído por inúmeros sub-blocos de *pipeline*. Dentre eles, os seis principais blocos lógicos são:

- **RS (Rasterizer)**: É responsável pela geração das coordenadas dos pixels (constituídas por X e Y).
- **TX (Texturizing Unit)**: Contém a memória de texturas e realiza amos-

tragem das mesmas baseado em qual *texel* (pixel de textura) representa o pixel sendo processado no *pipeline*.

- **TF (Texture Filter)**: executa um filtro bilinear 4x1 em 4 amostras de *texel* para produzir um único texel bilinearmente filtrado.
- **CC (Color Combiner)**: Realiza mistura entre duas fontes de cor através de interpolação linear entre duas cores.
- **BL (Blender)**: combina os pixels processados pelo *pipeline* com os pixels provenientes do *framebuffer*, além de poder realizar transparências e operações sofisticadas de *antialising*.
- **MI (Memory Interface)**: Realiza os verdadeiros ciclos de leitura, escrita e modificação do *framebuffer* e para o *framebuffer*.

O RDP possui quatro principais configurações nas quais as unidades de processamento individuais do *pipeline* trabalham juntas para gerar os pixels necessários para renderização de imagens. Tais configurações são denominadas "modos de ciclos de desenho", pois eles indicam quantos pixels são desenhados por ciclo de *clock*. Os quatro modos são:

- **FILL**: Neste modo, o RDP desenha os pixels no *framebuffer* utilizando a cor especificada no registrador de cor de preenchimento. Em seu máximo desempenho (geralmente atingido para retângulos), dois pixels de 32 bits, ou quatro de 16 bits são desenhados por ciclo.
- **COPY**: No modo de cópia, o RDP transfere as texturas da memória de texturas (TMEM) para o *framebuffer*. Em máximo desempenho, dois pixels de 32 bits, ou quatro de 16 bits são desenhados por ciclo.
- **1CYCLE** Neste modo, o RDP desenha um máximo de 1 pixel por ciclo de *clock*. Contudo, o pixel gerado no modo 1CYCLE é de qualidade muito superior aos demais: podem ser corrigidos de acordo com a perspectiva, bilinearmente filtrados, transparentes, possuir textura modular, dentre outros recursos. Este modo utiliza cada uma das unidades do *pipeline* do RDP citadas acima de uma só vez. O desempenho máximo é

raramente atingido, exceto em retângulos, pois o *framebuffer* é organizado em linhas. Um diagrama do *pipeline* deste modo é mostrado abaixo.

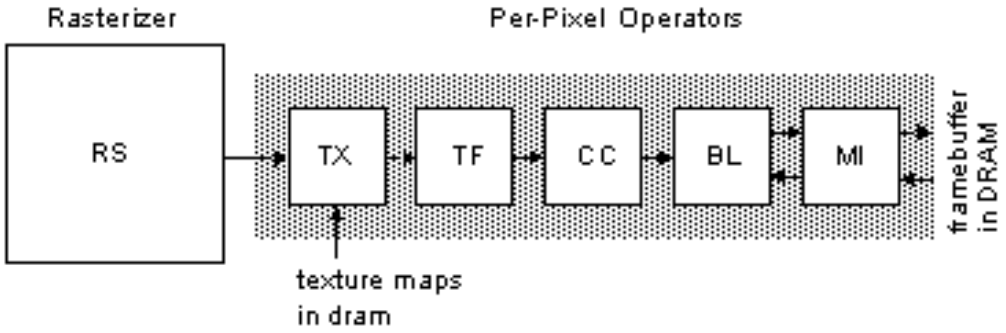

Figura 1: Diagrama do *pipeline* do modo 1CYCLE do RDP.
Fonte: *https* : *//level*42.*ca/projects/ultra*64/*Documentation/man/pro* − *man/pro*12/

- **2CYCLE** No modo 2CYCLE, as unidades de *pipeline* do RDP são reconfiguradas para excutarem funções adicionais a uma velocidade máxima reduzida de um pixel a cada 2 ciclos de *clock*. Todas as unidades são utilizadas duas vezes (exceto o rasterizador). Neste modo, os pixels de alta qualidade gerados têm todas as características dos que podem ser obtidos no modo anterior, com adição de névoa e "MIP-mapping". Um diagrama do *pipeline* deste modo é mostrado abaixo.

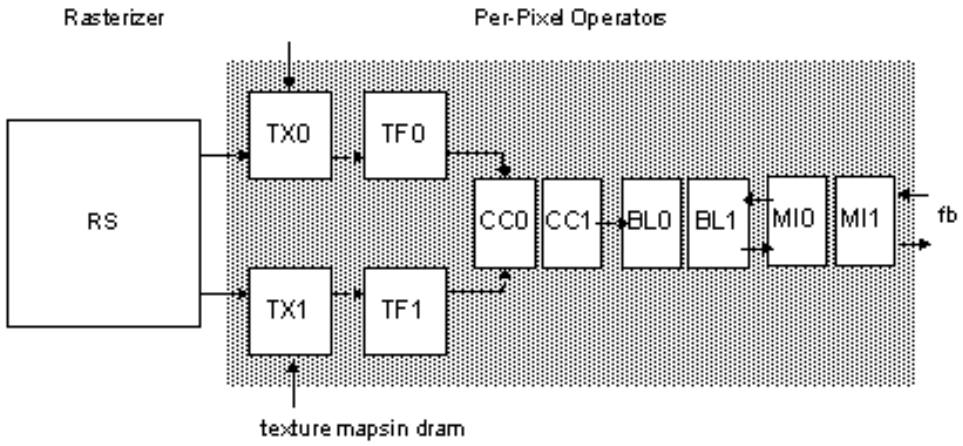

Figura 2: Diagrama do *pipeline* do modo 2CYCLE do RDP.
Fonte: *https* : *//level*42.*ca/projects/ultra*64/*Documentation/man/pro* − *man/pro*12/

3.1.2 Gouraud Shading

Devido ao espaço limitado para texturas do RDP, muitas vezes desenvolvedores se voltaram ao Gouraud Shading. A técnica consiste em interpolar o vetor normal de um polígono a partir dos vetores normais em seus vértices, permitindo uma variação suave de iluminação. Um exemplo da aplicação desse shader pode ser encontrado na Figura 3.

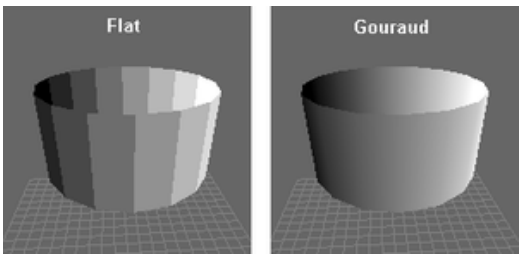

Figura 3: Comparação entre iluminação tradicional e com o uso de Gouraud. Ambos os objetos têm a mesma malha.
Fonte: *https : //en.wikipedia.org/wiki/Gouraud_shading*

Por outro lado, efeitos localizados (como luz especular) não são renderizados corretamente. Isso ocorre porque o efeito não afeta polígonos adjacentes. Esse efeito pode ser visto na Figura 4. A Figura 5 representa o resultado desejado.

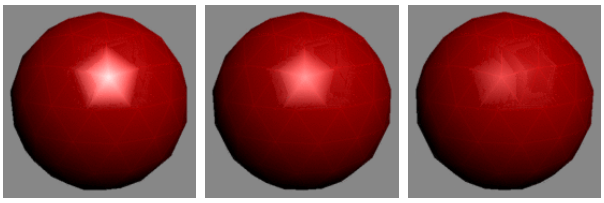

Figura 4: Efeito de luz com o Gouraud shading.
Fonte: https://en.wikipedia.org/wiki/Gouraud_shading

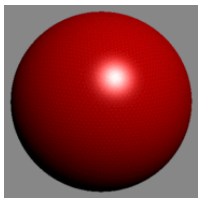

Figura 5: Efeito de luz desejado.
Fonte: https://en.wikipedia.org/wiki/Gouraud_shading

3.1.3 Microcódigo do RSP

O microcódigo é um interpretador de instruções de máquina que cria uma interface entre as instruções em nível de software e em nível de máquina, permitindo a reprogramação das funções que o RSP deve realizar. Isto permite otimizar qualidade ou velocidade de acordo com o que for desejado para o jogo. É possível reprogramar o processador para uma tarefa mais específica, melhorando sua performance em comparação com seu microcódigo genérico.

É possível programar microcódigo tanto para áudio como para gráficos, tornando-o muito flexível.

Apesar de ser possível reprogramar o RSP, poucos desenvolvedores se aproveitaram disso devido à falta de documentação e a dificuldade de se programar sem ferramentas oficiais. As ferramentas oficiais de desenvolvimento de microcódigo só foram fornecidas no fim do ciclo de vida do console, até então os desenvolvedores usavam ferramentas rudimentares para microprogramação, oque dificultava a tarefa imensamente. Alguns desenvolvedores que escreveram seus próprios microcódigos conseguiram uma qualidade gráfica melhor sem perda de desempenho. Um dos melhores exemplos é o jogo da Factor 5, *Indiana Jones and the Infernal Machine*, originalmente de PC, seu porte para o Nintendo 64 teve resultados gráficos melhores do que a versão original de PC, com texturas mais detalhadas e melhores efeitos de sombra e luz.

3.2 Game Pak

O conteúdo dos jogos do Nintendo 64 eram armazenados em cartuchos de ROM (sigla para *Read-only Memory*, ou memória somente para leitura) chamados de **Game Pak**. A utilização dos mesmos foi planejada para que o custo final do sistema fosse o menor possível, enquanto a velocidade do mesmo era maximizada. Os cartuchos, entretanto, possuíam uma capacidade de armazenamento muito inferior à dos CD-ROM, bem como um maior custo de produção.

3.2.1 Características

O tamanho de cada cartucho variava entre 4MB (tal como o de *Dr. Mario 64*) e 64MB (*Resident Evil 2* e *Conker's Bad Fur Day*), menos de dez vezes a capacidade do CD-ROM utilizado pelos consoles concorrentes, que armazenavam cerca de 650MB. Os game paks podiam conter três tipos de memória interna para armazenar o progresso dos jogos:

- **EEPROM (*electrically erasable programmable read-only memory*)**

  Um tipo de memória não volátil capaz de armazenar uma quantidade de dados relativamente pequena, permitindo que bytes sejam individualmente apagados e reprogramados.

- **Memória Flash**

  Um tipo de memória que pode ser eletricamente apagada e reprogramada, além de permitir que a escrita e leitura sejam feitas em blocos geralmente muito menores do que o dispositivo como um todo. Diferente das EPROMs, que precisavam ter seu conteúdo completamente apagado antes de serem reescritas.

- **RAM com backup de bateria**

  Uma memória RAM convencional plugada a uma bateria para impedir que os dados se perdessem quando o sistema fosse desligado.

Quando nenhum destes tipos eram utilizados, o progresso dos jogos era salvo em um componente a parte denominado *Controller Pak*, cujas características serão detalhadas no tópico 3.3.1.

Com relação à durabilidade, os Cartuchos se mostravam mais vantajosos, ainda que fossem menos resistentes aos efeitos do ambiente a longo prazo. Não havia a necessidade de mantê-los guardados em estojos protetores, uma vez que não eram suscetíveis a arranhões e seus subsequentes erros de leitura, como eram os CDs.

3.2.2 Performance

Os jogos do Nintendo 64 possuíam um tempo de carregamento muito menor quando comparados aos demais jogos em CD. Isso porque a velocidade de um cartucho podia chegar até 300MB/s, enquanto os CD-ROM dos consoles concorrentes não ultrapassavam a velocidade de 300KB/s, além de possuírem uma alta latência. Como consequência, podemos observar que as longas telas de *loading*, tão comuns nos demais sistemas, eram praticamente inexistentes no N64. Graças a essa alta velocidade de transmissão de dados, os game paks muitas vezes eram utilizados como uma extensão da memória RAM, sendo possível recuperar dados de lá quase com o mesmo tempo de acesso à memória principal, maximizando a eficiência da RAM do sistema.

3.2.3 Proteção contra cópias

Cada Game Pak do Nintendo 64 possuía um chip de segurança para impedir que cópias não autorizadas dos mesmos fossem criadas. Cada um desses chips continha um valor de semente utilizado para o cálculo da soma de verificação (conhcido como *Checksum*) do código de boot do jogo. Visando tornar a estratégia mais eficiente, cinco diferentes versões do chip foram criadas.

Durante o processo de boot - e ocasionalmente durante a execução dos jogos - o console computava a soma de verificação e realizava a checagem com o chip de segurança do Game Pak, interrompendo o boot caso o procedimento falhasse.

3.3 Principais periféricos

3.3.1 Controller Pak

O *Controller Pak* era o *memory card* do Nintendo 64, semelhante aos encontrados no Playstation e nos demais consoles com jogos de CD-ROM. Eles foram divulgados como um meio de promover a troca de dados de jogos com outros donos do N64, uma vez que a informação salva em um *Game Pak* não podia ser transferida para outro.

O *Controller Pak* era plugado na parte traseira do controle do N64 e somente alguns jogos permitiam que seus dados fossem salvos nele. Os modelos originais,

distribuídos pela Nintendo, possuíam uma memória do tipo SRAM alimentada por bateria, com uma capacidade de 32KB, limitada a armazenar apenas 16 arquivos contendo o progresso dos jogos.

3.3.2 Jumper Pak

O *Jumper Pak* era um preenchedor plugado na porta de expansão de memória do Nintendo 64. Não possuí nenhum propósito funcional, a não ser o de terminar o barramento da memória principal na ausência do *Expansion Pak*. Os primeiros consoles fabricados vinham com o *Jumper Pak* incluso e já instalado, uma vez que ele era necessário para o funcionamento dos mesmos quando estes não possuiam o *Expansion Pak*. Caso nenhum dos dois estivesse presente, o N64 não era capaz de exibir nenhuma imagem na tela.

3.3.3 Expansion Pak

Este periférico consiste em 4MB RAM que eram utilizados para duplicar a memória principal do Nintendo 64. O *Expansion Pak* era instalado numa porta localizada no topo do console e sua presença era obrigatória para que os títulos *Donkey Kong 64* e *The Legend of Zelda: Majora's Mask* pudessem ser jogados. Já os demais games podiam utilizar o periférico para melhorar seus apectos visuais e acrescentar opções de gameplay.

3.3.4 Rumble Pak

O *Rumble Pak* fazia com que o controle do console vibrasse quando o jogador realizasse certas ações em alguns jogos. Ele foi lançado especialmente para o jogo *Starfox 64*.

3.3.5 Transfer Pak

O *Transfer Pak* era um acessório utilizado para transferir os dados de jogos do *Game Boy*, console portátil da Nintendo, para o N64. Ele era plugado na parte traseira do controle e acompanhava o jogo *Pokémon Stadium*, para o que o jogador pudesse transferir os pokémon capturados no portátil. Assim, o Nintendo 64 já apresenta integração entre os outros consoles da própria empresa.

3.4 Agendamento de Áudio e Vídeo

Os jogos do Nintendo 64 possuem ambos processos gráficos e de áudio. O fluxo do processo gráfico é da CPU para o RSP, depois para o RDP e por fim aos barramentos de vídeos no banco de I/O, enquanto o fluxo do proceso de áudio faz o mesmo caminho, porém, sem passar pelo RDP.

Por vezes, o processamento gráfico leva mais de uma frame para finalizar, porém o processo de áudio deve ser entregue todas as frames, sem falha. Caso isso não aconteça, o som do jogo terá uma pausa inapropriada, causando ruídos desagradáveis e uma música não fluída. Para evitar isso, o console possuí uma *thread* chamada **Scheduler** (Agendador). O **Scheduler** é visto como um processo de alta prioridade, podendo tomar o controle da CPU, RSP ou RDP a qualquer momento. Ele pode ser invocado utilizando uma interrupção de sincronização vertical, e então checar o estado do RSP. Caso o RSP esteja em processamento gráfico, o **Scheduler** salva o atual estado do processo gráfico na memória, carrega o microcódigo de áudio e muda o estado do RSP para processamento de áudio. Quando o processamento de áudio termina, o estado do RSP é restaurado para quando foi interrompido, o microcódigo de gráficos é recarregado e assim o processamento gráfico pode continuar de onde parou. Dessa forma é possível garantir que o áudio não será afetado pelo processamento gráfico, que é mais demorado.

Assim, o **Scheduler** é capaz de controlar o tráfego de dados nos processadores, podendo alterar o que será processado a qualquer momento.

4 Conclusão

Sobre a arquitetura do Nintendo 64, pode-se concluir que o desempenho final do console foi prejudicado por más decisões de projeto. Dentre elas, a escolha da CPU não possuir acesso direto à memória, que já possuia alta latência de acesso, acabou tornando a recuperação de dados um processo muito lento. Contudo, a CPU realiza principalmente cálculo de física e o RCP possuí acesso direto à memória, tornando o processamento gráfico muito poderoso. Apesar disso, a RAM é extremamente rápida para acesso aleatório e barata, considerando suas especificações.

Outro grande problema fora a disponibilização tardia de ferramentas oficiais para a elaboração de microcódigo, dificultando seu desenvolvimento. Isso atrapalhou o aproveitamento do potencial máximo do console, o qual dependia muito do processamento gráfico do RSP.

Por fim, a maior limitação do N64 era sua memória. Ainda que os *Game Paks* fossem muito superiores em relação à performance, sua falta de espaço para o armazenamento de jogos e texturas impediu que muitos games fossem lançados para o console. Por outro lado, essas desvantagens motivaram o surgimento de soluções criativas como o *Gouraud Shading*, que levaram os desenvolvedores a utilizarem gráficos mais cartoonescos que, até hoje, são mantidos pela Nintendo.