

Instituto de Matemática e Estatística
Monografia dos curso Organização de Computadores

Arquitetura do Console Nintendo 64

Professor: Siang Wun Song

Alunos: Antonio Augusto Abello
Gustavo Estrela de Matos
Lucas Romão Silva

São Paulo, 4 de Dezembro de 2016

Conteúdo

1	Introdução	2
2	Principais Componentes do Nintendo64	3
3	Chip NEC VR4300	3
3.1	Principais Componentes do Processador VR4300	3
3.2	Formato de instrução	5
4	Reality Coprocessor (RCP)	7
4.1	RSP	8
4.2	RDP	9
5	Interface de Vídeo	10

1 Introdução

O Nintendo64 se consiste dum conjunto de componentes de hardware que trabalham em conjunto para o processamento gráfico e de áudio. O componente chave do sistema é o RCP, que será abordado nas próximas seções. A CPU e o RCP rodam concorrentemente; na CPU, os processos são rodados em *threads* ao passo em que no RCP os processos são rodados em *tasks*.

Na CPU, os processos rodam no conjunto de *threads* onde cada uma possui sua própria pilha de execução. Diferentemente dos sistemas UNIX, no console não existe o conceito de relógio de swap ou de *round-robin* para o escalonamento, isto é, uma *thread* que possui CPU não é interrompida, caso exista outra de mesma prioridade a CPU só será removida da mesma quando ela acabar sua execução ou caso outra *thread* de maior prioridade.

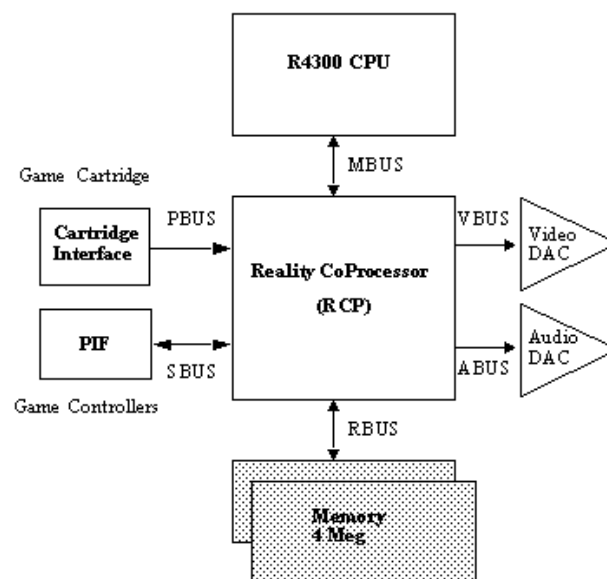


Figura 1: Diagrama de blocos do hardware do Nintendo 64

No aspecto do software, a camada é responsável pela tradução de funções lógicas e argumentos para as configurações específicas do hardware enquanto boa parte do gerenciamento dos recursos era deixada para os jogos, sendo as únicas camadas já fornecidas pelo console sendo de E/S e de áudio.

Nas seções que seguem, será falado mais a fundo sobre os componentes do console e suas respectivas funções no processamento de um jogo.

2 Principais Componentes do Nintendo64

3 Chip NEC VR4300

O chip NEC VR4300 é o principal processador no Nintendo64, responsável principalmente por processar a lógica dos jogos e, também áudio. Esse processador foi desenvolvido pela empresa japonesa NEC e implementa a arquitetura de conjunto de instruções MIPS, desenvolvida pela empresa de mesmo nome. A arquitetura MIPS define um conjunto de instruções do tipo RISC, *reduced instruction set computer*.

O processador VR4300 possuía uma arquitetura compatível com instruções de 64 bits, apesar de grande parte das instruções do Nintendo 64 serem de apenas 32 bits. Especificamente nesse console, o processador da NEC trabalhava a uma frequência de 93,75 MHz.

3.1 Principais Componentes do Processador VR4300

A figura 3.1 mostra um diagrama com os principais componentes do processador VR4300 e como eles se comunicam. Nesta seção, discutiremos o papel de cada um desses componentes.

- **System interface** define a interface do chip com os componentes externos. Essa interface é formada por um bus de 32 bits e vários outros bits para controle de bits, interrupções, relógio, etc. No Nintendo 64, alguns desses bits se conectam à memória RAM do console e, também diretamente ao cartucho com o jogo.
- **Clock generator** é responsável por determinar o clock interno do chip, que é o clock do pipeline do processador. Esse componente recebe um clock externo e define o clock de pipeline como uma fração do anterior. Um dos modos de operação do VR4300 é ter o relógio interno funcionando com 2 ciclos a cada ciclo do relógio externo.
- **Instruction cache** é um cache que guarda a instrução que está sendo executada. No contexto do pipeline, manter uma cópia da instrução lida dentro do processador é essencial para aumentar eficiência e permitir operações como desvios e interrupções.

- **Execution unit** é uma parte do hardware que é especializada em realizar operações aritméticas. Na arquitetura MIPS, a *Execution unit* tem o papel de *COP1* (coprocessador 1).
- **CP0** é o componente responsável por fazer o controle da memória (MMU), isto é, permite endereçamento virtual, controla o acesso a pedaços de memória diferentes, aloca páginas de memória e outras funções. Para ajudar na eficiência de acesso a memória, o CP0 possui uma *TLB* (*Translation Lookaside Buffer*) que é, basicamente, um cache para a tradução de endereços virtuais para físicos.
- **Data cache** é um cache para dados da memória. Os dados armazenados nesse componente são indexados pelos endereços virtuais. Podemos ver, no diagrama, uma conexão entre *Execution Unit* e esse componente, essa ligação é utilizada quando um dado necessário em uma computação já está no cache. Quando isso acontece poupamos o trabalho de buscar dados na memória ou qualquer outro componente pois ele já está disponível dentro do chip.
- **Instruction address** tem o papel de calcular o endereço da próxima instrução. Esse circuito deve ser capaz de incrementar o *Program Counter*, dando sequência as execuções de instruções, e também deve ser capaz de atualizar o *Program Counter* quando ocorrem desvios (*branch* e *jump*).
- **Pipeline control** é o circuito que controla a execução do pipeline do processador. O conceito de pipeline consiste em dividir a execução de uma tarefa de maneira que mais de uma tarefa possa ser executada simultaneamente, desde que em diferentes etapas e não dependentes. O pipeline do processador VR4300 consiste em cinco etapas diferentes:
 - IC (Instruction Cache Fetch): faz a leitura da instrução.
 - RF (Register Fetch): deixa valores de registradores prontos para serem usados em cálculos.
 - EX (Execution): executa operações aritméticas.
 - DC (Data Cache Fetch): armazena no cache os dados utilizados na instrução.

- WB (Write Back): atualiza valores na memória ou registradores.

O que ocorre em cada etapa do pipeline também depende muito da instrução que está sendo executada, portanto o que está descrito logo acima é uma visão superficial de cada uma dessas etapas.

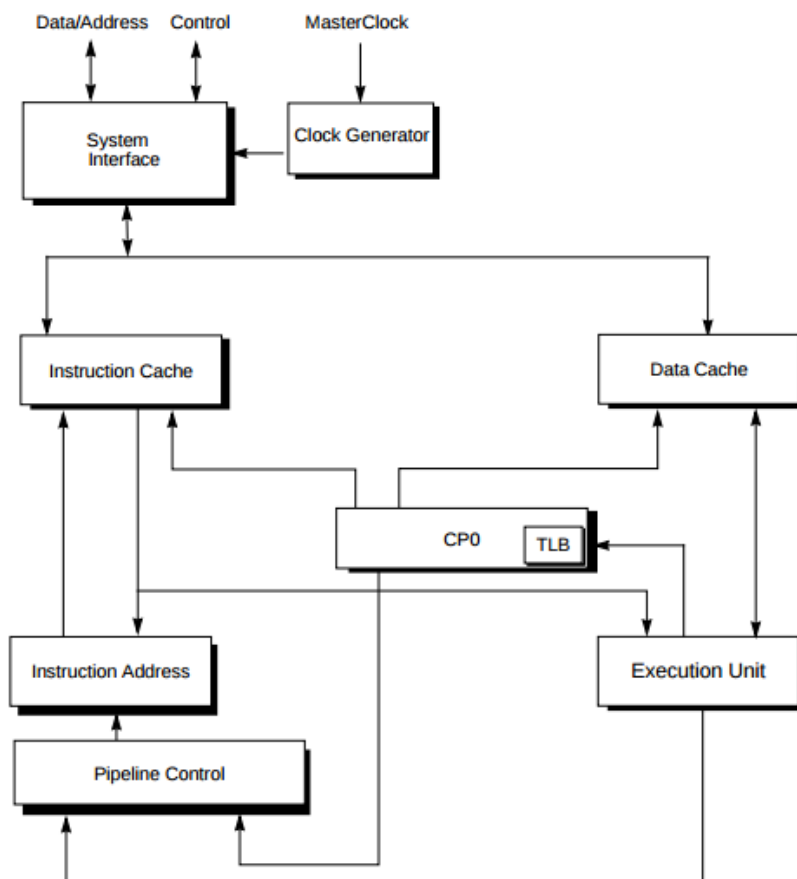


Figura 2: Um diagrama com os principais componentes do processador VR4300

3.2 Formato de instrução

Cada instrução do processador é formada por 32 bits, e elas podem ser separadas em três categorias: *I-type*, *J-Type* e *R-Type*.

As instruções do tipo *I-Type* são formadas por 5 bits, *op*, que determinam a operação; 5 bits em *rs* e mais 5 em *rt*, que determinam os registradores que estão sendo operados; e mais 16 bits, *immediate*, que pode representar ou um endereço ou uma constante. Exemplos de instruções desse tipo são as instruções *load* e *store*.

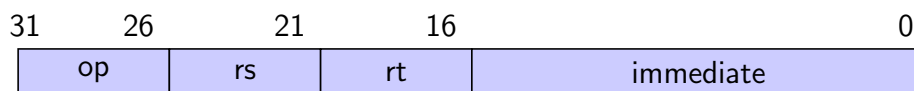


Figura 3: Formato de uma instrução *I-Type*

As instruções do tipo *J-Type* são usadas para controlar o fluxo do programa. Para isso, esse tipo de instrução pode pular para um pedaço específico do código por via de um *jump* ou um *branch*. Quando uma instrução do tipo *jump* é executada, o desvio sempre acontece, ao contrário da instrução *branch* na qual é possível determinar uma condição para o desvio. No VR4300 essas instruções são formadas por 5 bits, *op*, que determinam a operação; e mais 26 bits *target*, que determinam o endereço do possível desvio. Quando a instrução é um desvio obrigatório, os 26 bits estão todos disponíveis para determinar o endereço de destino, mas no caso de um *branch* o valor de *target* só pode determinar um *offset* de 16 bits relativo ao registrador *PC*.



Figura 4: Formato de uma instrução *J-Type*

As instruções do tipo *R-Type* envolvem apenas o uso de registradores. Exemplos de instruções desse tipo são aquelas que fazem operações aritméticas entre dois registradores e guardam o resultado em um terceiro registrador. Essas instruções são formadas por 5 bits *op*; 5 bits para cada um dos três registradores *rs*, *rt* e *rd*; 5 bits *sa* que definem um *shift* para o resultado; e mais 6 bits para *function*.

Code	Operation
0	Add
1	Subtract
2	Multiply
3	Divide
4	Square root
5	Absolute value
6	Transfer
7	Sign reverse
8	Convert to 64-bit fixed-point, rounded to nearest/even
9	Convert to 64-bit fixed-point, rounded toward zero
10	Convert to 64-bit fixed-point, rounded to $+\infty$
11	Convert to 64-bit fixed-point, rounded to $-\infty$
12	Convert to 32-bit fixed-point, rounded to nearest/even
13	Convert to 32-bit fixed-point, rounded toward zero
14	Convert to 32-bit fixed-point, rounded to $+\infty$
15	Convert to 32-bit fixed-point, rounded to $-\infty$
16–31	Reserved
32	Convert to single floating-point
33	Convert to double floating-point
34	Reserved
35	Reserved
36	Convert to 32-bit fixed-point
37	Convert to 64-bit fixed-point
38–47	Reserved
8–63	Floating-point compare

Tabela 1: Lista de todas as possíveis funções em uma instrução do tipo *R-Type* [1].

4 Reality Coprocessor (RCP)

O chip RCP (Reality Coprocessor) é, na verdade, uma coleção de processadores, interfaces de memória e lógica de E/S. Seus dois processadores são o RSP (Reality Signal Processor) e o RDP (Reality Display Processor). Ambos possuem responsabilidades parecidas (cuidam de partes distintas do processo de geração de imagem e o RSP cuida sozinho do áudio) e são separados para aumentar a performance do sistema ao realizar suas tarefas em paralelo. A parte de E/S conecta o RCP com a CPU, os inputs, o cartucho e também possui saídas *write-only* de áudio e vídeo. Essas conexões são necessárias porque a parte mais lógica do processamento de áudio e vídeo é feita pela CPU. A figura abaixo mostra um esquema dos componentes do RCP.

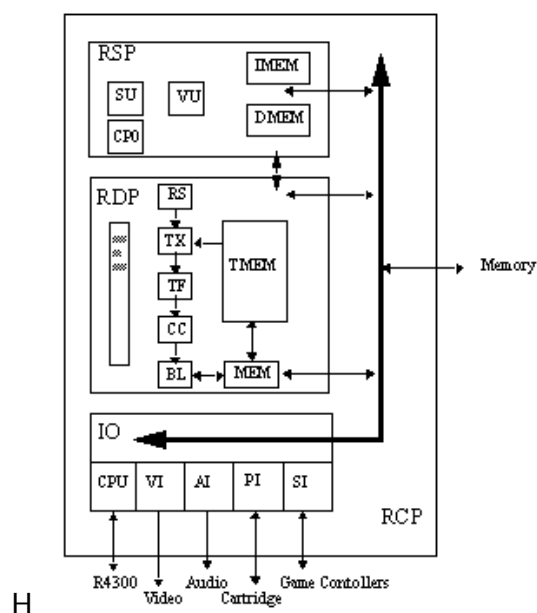


Figura 5: Um diagrama com os principais componentes do co-processador RCP

4.1 RSP

O RSP (*Reality Signal Processor*) é um processador que se especializa na execução das transformações geométricas além de fornecer visibilidade para as funcionalidades do RCP.

Outra função também realizada pelo RSP é executar a decodificação do áudio, apesar de este procedimento também poder ser feito pela CPU. Dentre os consoles da época, o Nintendo 64 era o único que não possuía um chip dedicado ao processamento de áudio, cabendo então ao programador decidir se o processamento do áudio seria feito na CPU ou no RSP. Por um lado, isso permitia que o sistema fosse flexível às necessidades de cada jogo, por outro lado, isso exigia que o programador soubesse usar corretamente as funcionalidades do console a fim de melhorar a performance do console.

O RSP é totalmente reprogramável através do microcódigo (μ code). Através deste, era possível adicionar novas funcionalidades, novas operações geométricas, novos efeitos além de otimizações para velocidade ou qualidade de processamento. Entretanto, o acesso ao μ code só foi liberado pela Nintendo no final do ciclo de vida do console.

Unidades de processamento do RSP

- Scalar Unity (SU): Usa um subconjunto de R4000 instruções para execução
- Vector Unity (VU): possui mecanismos para operação soma-produto de 16 bits

- Instruction Memory (IMEM): Memória que armazena o microcódigo
- Data Memory (DMEM): Espaço de memória usado pelo microcódigo

4.2 RDP

O RDP é um processador que se especializa na parte final de processamento de vídeo, possuindo diversos componentes para realizar essa tarefa. Recebe a maioria de suas informações e instruções do RSP, mas também ocasionalmente recebe instruções via código para alterar seu modo de execução. Conta ainda com uma memória específica para guardar texturas (TMEM) de 4KB e ao final de seu pipeline escreve na memória (na parte reservada ao *framebuffer* os pixels prontos. Ele possui 6 blocos lógicos (que em hardware são implementados como vários blocos, mas agem como um só)

- Rasterizador (RS): rasteriza triângulos e retângulos
- Unidade de Textura (TX): realiza *sample* das texturas carregadas em TMEM
- Unidade de Filtro de Textura (TF): filtra os *samples* de textura
- Combinador de Cor (CC): interpola e combina duas cores (por exemplo, da textura e da iluminação)
- Blender (BL): mistura os pixels gerados com os pixels atuais no buffer (para transição), realiza operações relacionadas à profundidade (*z-buffer*) para produzir o 3D e o *anti-aliasing* (redução de serrilhados)
- Interface de Memória (MI): realiza as operações de leitura, escrita e modificação dos pixels um a um no *framebuffer* (em 1 ou 0.5 pixel-por-clock). Também carrega a TMEM usando os endereços físicos e lê/escreve o *z-buffer*

5 Interface de Vídeo

Ao final de todo o processamento da CPU, RSP, RDP, a interface de vídeo lê o *framebuffer* e gera os sinais RGB, podendo trabalhar tanto em PAL ou NTSC. Ela ainda realiza um segundo

passo do algoritmo *anti-alias* e pode alterar a escala das imagens. O processamento de imagens, então, passa por quase todo hardware, começando na CPU, com a parte mais lógica, passando ao RSP com a geometria e terminando no RDP com a renderização, texturização e finalização

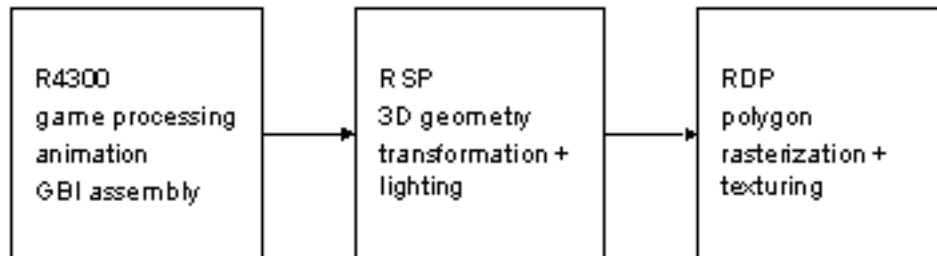


Figura 6: Esquema geral de processamento de vídeo

Referências

- [1] NEC V_R4300, V_R4305, V_R4310 64-bit processor User's Manual 7th edition. Japan, 2000.
- [2] Nintendo 64 Programming Manual. Digitalizado por Project Ultra 64. Acessível em: <http://level42.ca/projects/ultra64/Documentation/man/>