Lucas Ibiapino Alves

Um Estudo Comparativo entre Arquiteturas MIPS e IPNOSYS

Picos - PI
Junho, 2017

Lucas Ibiapino Alves

Um Estudo Comparativo entre Arquiteturas MIPS e IPNOSYS

Trabalho de Conclusão de Curso submetido à coordenação de Sistemas de Informação, da Universidade Federal do Piauí, no período de 2017.1 para aquisição do título de Bacharel em Sistemas de Informação, sob orientação do Professor Esp. Ivenilton Alexandre de Souza Moura.

Universidade Federal do Piauí
Campus Senador Helvídio Nunes de Barros
Bacharelado em Sistemas de Informação

Picos - PI
Junho, 2017

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

A474e Alves, Lucas Ibiapino.

Um estudo comparativo entre arquiteturas MIPS E IPNSYS / Lucas Ibiapino Alves .– 2017.
CD-ROM : il.; 4 ¾ pol. (45 f.)
Trabalho de Conclusão de Curso (Curso Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí, Picos, 2017.
Orientador(A): Prof. Esp. Ivenilton Alexandre de Souza Moura

1. Arquitetura MIPS. 2. Arquitetura IPNosys. 3. Arquitetura de Computadores. I. Título.

CDD 004.22

UM ESTUDO COMPARATIVO ENTRE ARQUITETURAS MIPS E IPNOSYS

LUCAS IBIAPINO ALVES

Monografia aprovada como exigência parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI, 22 de junho de 2017

Prof. Esp. Ivenilton Alexandre de Souza Moura
Orientador

Profª. Ma. Alcilene Dalilia de Sousa
Membro

Profª. Ma. Patricia Medyna Lauritzen de Lucena Drumond
Membro

Agradecimentos

Primeiramente agradeço a Deus pelas bênçãos que proporciona em minha vida, pela coragem, determinação, paciência e pela oportunidade de aprender cada vez mais. Todas as conquistas que obtive dedico a ti senhor.

Ao meu orientador, o professor Esp. Ivenilton Alexandre, na qual o chamo carinhosamente de mestre, por me guiar nesse campo antes desconhecido e por ser meu mentor, me mostrado os caminhos por onde devo seguir. Obrigado professor por me auxiliar sempre nas horas complicadas, por puxar minha orelha quando as vezes me permitia fraquejar e por me levantar em todas as vezes que caí, vou levar isso sempre comigo.

Ao professor Dr. Sílvio Fernandes, da UFERSA, pela ajuda e co-orientações sobre a arquitetura IPNoSys e pelas dicas de programação para o desenvolvimento dos códigos deste trabalho, muito obrigado professor pela ajuda.

Agradeço a todos os meus professores de ensino médio do Colégio Santa Rita, por me dar base para todos os desafios que encontrei na universidade. Agradeço em especial ao professor Denildo, carinhosamente chamado de "Denildão" por ter me inserido na área de física eletrônica e computação, meu muito obrigado professor.

A todos os meus professores do curso de Sistemas de Informação por transmitirem seus conhecimentos e conselhos, especialmente à professora Ma. Patrícia Medyna Lauritzen de Lucena Drumond pelas dicas que tive ao longo do curso e por ter sido uma "mãe", me ensinando e me aconselhando em decisões que tomei em minha vida acadêmica e pessoal. À professora Ma. Alcilene Dalília de Sousa, por orientações, dicas e pela positividade. Aos professores Thiago José e Allan Jheyson pelo companheirismo em projetos de *hardware* e pelas diversas vezes que debatemos sobre essa área que me encanta todos os dias.

À Universidade Federal do Piauí por conceder todo o espaço e fomento para realizar meus estudos e obter mais conhecimento. A todos os professores e funcionários dessa belíssima instituição. Muito obrigado a todos!

Aos meus pais, Lusiene Ibiapino Veras Alves e Francisco de Assis Alves, pela força e apoio que sempre tive em minha vida, pela dedicação, pelas vezes que tiveram que abrir mão de tudo para me ajudar, pelos seus ensinamentos e pelo o exemplo que são, com certeza levarei para minha vida. À minha irmã Iasmim Ibiapino Alves, por se mostrar presente sempre nas horas mais difíceis, disposta a me ajudar e aconselhar. Devo minha vida a vocês, muito obrigado.

À toda minha família por ter se mostrado presente nas vezes que precisei, tanto na vida acadêmica, quanto na minha vida pessoal.

Aos meus amigos Maura Géssica, Marcos Raniere, Antonino Calisto, Thayane Simões, Mirla Santana, Felipe Santos, Emerson Silva, Gabriel Moura, Anne Lívia, Letícia Araújo, Nathana Lucena, Otília Santos, Jaqueline Campelo, Davi Luís e Alexandre Ramos, que me

ajudaram no decorrer de minha caminhada acadêmica, muito obrigado pelos conselhos, dicas, palavras de força e pela disponibilidade em me ajudar quando precisei. Com certeza nossa amizade será levada além da UFPI.

A todos que contribuíram de forma direta e indireta para minha formação, muito obrigado.

Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível.

Charles Chaplin

Resumo

Com a crescente demanda de informação, surge a necessidade de construir novas arquiteturas com poder de processamento rápido e que haja menos gastos de energia, ociosidade do processador e menos perdas de dados. Com a ampliação no uso de mais núcleos de processamento, são necessários estudos de novas arquiteturas para que se possa substituir ou melhorar as já existentes. Atualmente, existe um grande paradigma que entrelaça duas diferentes vertentes: o uso de *pipeline* e uso de unidades de processamento e roteamento, dentre os modelos de arquiteturas que são candidatas a serem utilizadas hoje em dia. Este trabalho propõe a construção de um modelo comparativo entre a arquitetura MIPS e IPNoSys, determinando qual arquitetura se comportou em testes algorítmicos, apresentando resultados que serão comparados entre ambos. O modelo apontou que a arquitetura IPNoSys se sobressai da arquitetura MIPS em quesitos como velocidade e eficiência de processamento na resolução de problemas, mas a arquitetura MIPS se mostrou mais escalável do que a arquitetura IPNoSys.

Palavras-chaves: Arquitetura MIPS. Arquitetura IPNoSys. Arquitetura de Computadores.

Abstract

With the growing demand for information, the need arises to build new architectures with fast processing power and less energy costs, processor idleness and less data loss. With the increase in the use of more processing cores, studies of new architectures are needed to be able to replace or improve existing ones. Currently, there is a great paradigm that interweaves two different strands: the use of pipeline and the use of processing and routing units, among the models of architectures that are candidates for use today. This work propose the construction of a comparative model between the MIPS and IPNoSys architecture, determining which architecture behaved in algorithmic tests, presenting results that will be compared between both. The model pointed out that the IPNoSys architecture stands out from the MIPS architecture in terms of speed and processing efficiency in problem solving, but the MIPS architecture proved to be more scalable than the IPNoSys architecture.

***Keywords**: MIPS Architecture. IPNoSys Architecture. Computer Architecture.*

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BI	Busca de Instrução
BO	Busca dos Operandos
CISC	Complex Instruction Set Computer
CO	Cálculo dos Operandos
CPU	Central Processing Unit
DI	Decodificação de Instrução
EI	Execução da Instrução
EO	Execução de Operando
GHz	GigaHertz
IPNoSys	Integrated Processing NoC System
IPS	Instruções Por Segundo
MARS	MIPS Assembler and Runtime Simulator
MAU	Memory Access Unit
MIPS	Microprocessor without Interlocked Pipeline Stages
MPSoC	MultiProcessors System on Chip
NoC	Network on Chip
PDL	Package Description Language
RISC	Reduced Instruction Set Computer
RPU	Routing and Processing Unit
SoC	System on Chip
US	Unidade de Sincronização
ULA	Unidade de Lógica e Aritmética
VHDL	VHSIC Hardware Description Language

Sumário

1 Introdução

Arquitetura de computadores é a forma de referenciar componentes visíveis de um computador. Com o passar do tempo, foram surgindo novas arquiteturas e componentes que mudaram a forma de como os dados são processados. Com ênfase em aumentar o poder de processamento, duas vertentes são utilizadas: o aumento da frequência e o paralelismo (de dados e de instruções).

Segundo Moore et al. (1965), o número de transistores dobra a cada 2 anos, ou seja, com o avanço tecnológico, a frequência de processamento aumentou ao passo em que o uso desses transistores foi ampliado, consequentemente aumentando a quantidade de energia e calor, causando problemas. Nesse contexto, o paralelismo tornou-se uma alternativa viável de aumentar a performance computacional, sem a dependência do aumento da frequência de processamento.

Com o surgimento dos computadores multinúcleo, ou seja, a integração de mais núcleos em um chip surgiu a capacidade de processar uma ou mais instruções por ciclo de *clock*, fazendo com que o paralelismo ficasse em maior uso nos dias atuais.

Unidades de Processamento com arquiteturas de 64 e 32 *bits*, utilizaram o aumento do ciclo de *clock* como forma de melhorar a performance. Atualmente essa nova concepção de sistema está sendo amplamente difundido no mundo, graças a esses pressupostos surgidos para processamento em nível de instrução e dados, surgindo novas ideias para a construção e uso de sistemas menores como sistemas embarcados.

Um sistema embarcado possui como elemento principal, a CPU (*Central Processing Unit*), onde os cálculos e operações são feitos. Esses sistemas poderão usar arquiteturas de mono ou multinúcleo, dependendo muito da aplicação. Com o uso de sistemas embarcados, estão sendo desenvolvidas muitas arquiteturas multinúcleo, tendo como exemplo o MIPS (*Microprocessor without Interlocked Pipeline Stages*) e IPNoSys (*Integrated Processing NoC System*).

Arquiteturas do tipo MIPS utilizam como organização o RISC (*Reduced Instruction Set Computer*), sendo que ciclos de pequenas instruções são favorecidas com o uso desses processadores (PATTERSON, 2005). Com o surgimento dos processadores multinúcleo e a diminuição dos transistores, foi criado um conceito novo chamado MPSoC (*Multi Processor System on Chip*), onde em um único chip possui muitos núcleos de processamento.

Com o advento dos MPSoC's, outras possibilidades puderam ser pensadas para a construção de sistemas capazes de comportar, em um único chip, muitos núcleos de processadores. IPNoSys é uma arquitetura baseada em uma rede de MPSoC chamada NoC (*Network on Chip*), onde os chips interligados nessa rede formam uma malha de processamento capaz de comportar muitos e diferentes ciclos de instruções (FERNANDES et al., 2008).

Nos estudos realizados na literatura para a elaboração deste trabalho, não foram encontrados modelos de comparação entre essas arquiteturas, portanto esse trabalho se mostra inédito na literatura.

A importância deste trabalho mostra uma comparação direta de eficiência entre uma arquitetura existente e comumente utilizada em projetos e em equipamentos eletrônicos, com uma arquitetura protótipo e candidata a substituir as arquiteturas existentes, concluindo que a tecnologia que existe atualmente está sendo otimizada, dando espaço a novos projetos de arquiteturas.

Existem muitos estudos na área de MPSoCs, mas não existe um estudo aprofundado e comparativo entre arquiteturas multinúcleo e arquiteturas comuns.

1.1 Objetivos

O objetivo desse trabalho é apresentar um modelo comparativo entre as arquiteturas MIPS e IPNoSys elencando os problemas e as diferenças entre as arquiteturas dentro de um ambiente computacional, desenvolvendo e analisando códigos simples e recursivo, na qual será mensurado sua escalabilidade, velocidade e eficiência.

1.2 Organização do Trabalho

O trabalho em questão está dividido em 7 (sete) capítulos, que estão organizados da seguinte forma:

1. Capítulo 2 - Referencial Teórico: São apresentados os conceitos relacionados às arquiteturas propostas e todo o conjunto utilizado para a base da pesquisa.

2. Capítulo 3 - Trabalhos Relacionados: São listados os trabalhos relacionados com a área de atuação da pesquisa proposta.

3. Capítulo 4 - Metodologia: São descritos e apresentados a importância do trabalho, materiais, métodos e modelos utilizados na pesquisa.

4. Capítulo 5 - Resultados e Discussão: São apresentados e discutidos os resultados obtidos de acordo com a metodologia aplicada na pesquisa.

5. Capítulo 6 - Conclusão e Trabalhos Futuros: São apresentadas as considerações finais e conclusão da pesquisa com base nos resultados gerados a partir das simulações.

6. Apêndices: São apresentados o guia de programação utilizado para o desenvolvimento dos códigos.

2 Referencial Teórico

Neste capítulo são apresentados e desenvolvidos assuntos relacionados e que servem de base para a elaboração deste trabalho de conclusão de curso, com a finalidade de apresentar uma concepção mais clara dos assuntos abordados. Foram considerados os conceitos e aplicações de ciclo de *clock*, conjuntos de instruções, CPU, *pipeline*, memória, linguagens de programação, programação e arquiteturas paralelas, MPSoC, Arquitetura MIPS e IPNoSys.

2.1 Ciclo de Clock

Para a execução dos programas de um computador, é necessário que as instruções que compõe o programa, sejam executadas passo a passo, obedecendo as ordens de precedência determinadas pelo algoritmo e pelo sistema operacional. Das instruções utilizadas pelo processador, é necessário que se haja ciclos para que uma instrução seja executada ou encaminhada para um determinado local.

Segundo Hennessy (2008), ciclo de *clock* é um intervalo de tempo determinado por impulsos elétricos que o processador utiliza para se comunicar com os componentes do computador e processar instruções. Esse intervalo de tempo é medido em *hertz* por segundo, que indica o número de oscilações dentro de uma determinada medida (Figura 1).

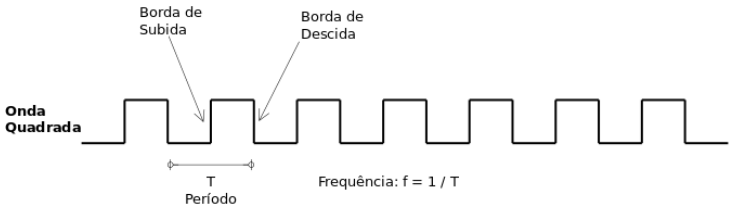

Figura 1: Representação do Pulso de *Clock*

Fonte: Hennessy (2008)

Hoje em dia, os processadores trabalham na faixa dos GHz (*GigaHertz*) ou seja, em uma frequência de 2,4 GHz significa dizer que esse processador comporta cerca de 2 bilhões e 400 milhões de ciclos por segundo, mostrando uma grande evolução.

Nos processadores com *pipeline*, a cada ciclo, uma instrução percorre um estágio, como mostrada na Figura 1, sendo assim um processador que comporta muitos núcleos. Um ciclo de *clock* haverá processamento de muitos dados por segundo, dando mais desempenho. Mas com o aumento da quantidade de ciclos por segundo, haverá também maior demanda

de energia, pois com o aumento da quantidade de núcleos, o impulso elétrico percorrerá mais área de chip.

2.2 Conjuntos de Instruções

A execução de tarefas e programas de um computador se deve a partir de sua construção ser feita através de algoritmos. Algoritmos, segundo Cormen (2009), é uma sequência de instruções finitas e bem definidas, que busca executar de forma eletrônica ou mecânica ações, gerando uma resposta. Em outras palavras, são passos definidos para a execução de comandos capazes de interagir com usuários ou com o ambiente.

Existem vários tipos de conjuntos de instruções, na qual cada componente suporta um determinado tipo. Nesse trabalho é definido apenas dois conjuntos, o CISC (*Complex Instruction Set Computer*) e RISC (*Reduced Instruction Set Computer*).

Conjunto de instruções CISC é uma linha de arquitetura de processadores capaz de executar centenas de instruções complexas diferentes por ciclo de *clock*, ou seja, os processadores possuem a capacidade de processar e executar um conjunto grande de instruções, dando uma alta escalabilidade e complexidade à arquitetura (STALLINGS, 2010).

Arquiteturas do tipo CISC são mais robustos e executam instruções complexas que, normalmente, processadores do tipo RISC demorariam muitos ciclos de *clock* para executar. Como foi definido por Silva e Antunes (2013), uma das características dos processadores CISC é a utilização de microcódigo e micro programação, sendo uma das características primordiais que permitiam aos projetistas a implementação de instruções complexas em *hardware*. A rapidez na execução das instruções em processadores portadoras dessa arquitetura, se deve pelo fato de haver uma parte da instrução gravadas dentro do processador, facilitando a execução.

A vantagem de utilizar processadores com a arquitetura CISC seria pelo fato dela reduzir o tamanho do código executável, pois possuem instruções já pré-carregadas no processador. A desvantagem é a impossibilidade de se alterar alguma instrução complexa, sem falar no espaço ocupado pelas instruções dentro do processador, podendo comprometer o seu desempenho.

Já o conjunto de instruções RISC, é uma linha de arquitetura de processadores que possui um conjunto pequeno de instruções que, ao contrário dos CISC, executam as instruções de forma mais simples (MONTEIRO, 2002). Computadores que comportam essa linha de arquitetura, não possuem uma micro programação gravada dentro do processador, ou seja, as instruções não são executadas diretamente pelo *hardware*.

Arquiteturas desse tipo são amplamente utilizadas por serem mais baratas e mais viáveis para produzir. Elas estão presentes na maioria dos produtos eletrônicos, principalmente as que possuem processadores MIPS, onde uma arquitetura baseada em registradores é utilizada para armazenar a maioria das operações.

Por ser um conjunto reduzido de instruções, arquiteturas RISC executam apenas uma instrução por ciclo de *clock*, dando uma menor escalabilidade em relação com arquiteturas CISC. No entanto, processadores que comportam a RISC possuem, segundo (STALLINGS, 2010), modos de endereçamento mais simples. É por possuir um formato de instrução mais simples, seu endereçamento será menos robusto.

De acordo com a Tabela 1, pode-se elencar uma comparação entre as arquiteturas de processadores do tipo RISC *versus* CISC:

Tabela 1: Comparação entre RISC *versus* CISC

Arquitetura	**RISC**	**CISC**
Implementação do Controle	*hardware*	Micro programação
Comprimento das Instruções	fixo	variável
Número de Registradores	tipicamente alto (32 a 128)	tipicamente baixo (4 a 16)
Execução das Instruções	Alta superposição baseada em *pipeline*	Baixa superposição
Número de Instruções	Médio (tipicamente 64)	Alta (mais de 100)
Instruções de Desvio	Desvio de atraso do *hardware*, para predição de desvio	Normal (condicional ou absoluto)

Fonte: Autor

O conjunto de instruções das arquiteturas propostas são demonstradas no Apêndice deste trabalho, onde é apresentada uma Figura contendo as principais instruções MIPS e uma Tabela com as principais instruções IPNoSys, que foram utilizadas na construção dos códigos.

2.3 Central Processing Unit

Dentro da construção principal de qualquer ambiente computacional, pode se elencar estruturas essenciais para seu funcionamento. Dentre várias estruturas a CPU (conhecido popularmente como processador) não é diferente das demais. Segundo Stallings (2010), a CPU é um dos componentes mais importantes de um computador, é nela que todas as funcionalidades, ciclos de instruções, distribuição de recursos do computador são executadas e controladas.

A arquitetura básica de um processador consiste em memórias de acesso a dados e a instruções como memórias *cache*, registradores, memória principal, unidade lógica e aritmética para cálculos matemáticos, unidade de controle e dispositivos periféricos de entrada e saída de dados. Todas essas unidades são interconectadas através de barramentos, que são vias de alta velocidade onde os dados trafegam.

Os processadores nos dias atuais podem trabalhar com um ou mais núcleos. Quanto maior a velocidade do processador, mais rápido ele executará instruções, processando

maior quantidade de dados. Segundo Patterson (2005), o que faz determinar a velocidade de processamento é a frequência de energia que é percorrida por área de chip em um ciclo de *clock*, medido em *hertz* (ciclos por segundo).

Um grande advento para melhorar ainda mais o processamento de instruções foi a implementação do pipeline, que organiza ao mesmo tempo, várias instruções diferentes em vários estágios como se fosse uma linha de montagem. De acordo com Stallings (2010), "À medida que os sistemas computacionais evoluem, um melhor desempenho pode ser obtido tirando vantagens das melhorias, como por exemplo a construção de circuitos mais rápidos".

Nos processadores que utilizam a arquitetura MIPS em sua construção, possui um processador com a capacidade de processar instruções simples, mostrando que ele não possui uma grande complexidade no seu funcionamento. Ele possui a implementação de estágios de *pipeline* para que o processador não fique muito tempo ocioso.

A Figura 2 mostra a construção interna de um *chip* que possui a arquitetura MIPS, note que as estruturas se assemelham com os processadores convencionais.

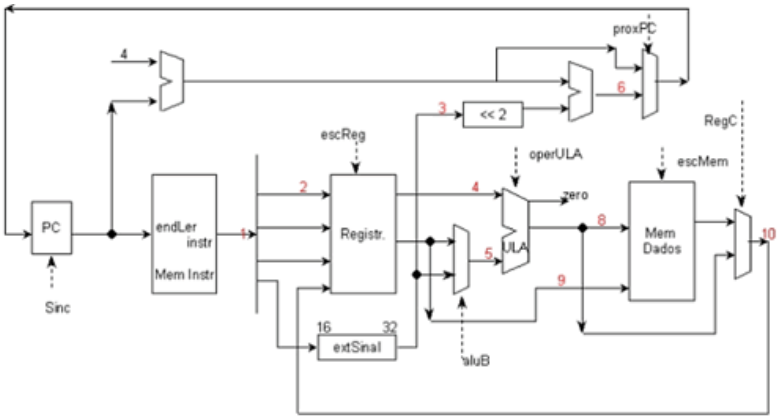

Figura 2: Representação Interna de um MIPS

Fonte: Patterson (2005)

Já a arquitetura IPNoSys possui as RPUs (*Routing and Processing Unit*), que são unidades de processamento e roteamento de pacotes de instruções desta arquitetura. A construção desse tipo de processador é representada na Figura 3. Dentro da RPU possui uma US (Unidade de Sincronização), onde os pacotes são armazenados para serem sincronizados entre as RPUs ou entre as MAUs (*Memory Access Unit*). Possui também uma ULA (Unidade de Lógica e Aritmética), *buffers* nas entradas, um *crossbar* e um árbitro nas saídas. O *crossbar* é uma via de interconexão dentre os componentes da RPU e o árbitro é um seletor de pacotes, ou seja, ele seleciona a prioridade e a quantidade de pacotes que serão injetados na rede de processamento (Figura 4).

A construção interna desses processadores são específicas para processar as instruções definidas pela gramática da linguagem da arquitetura.

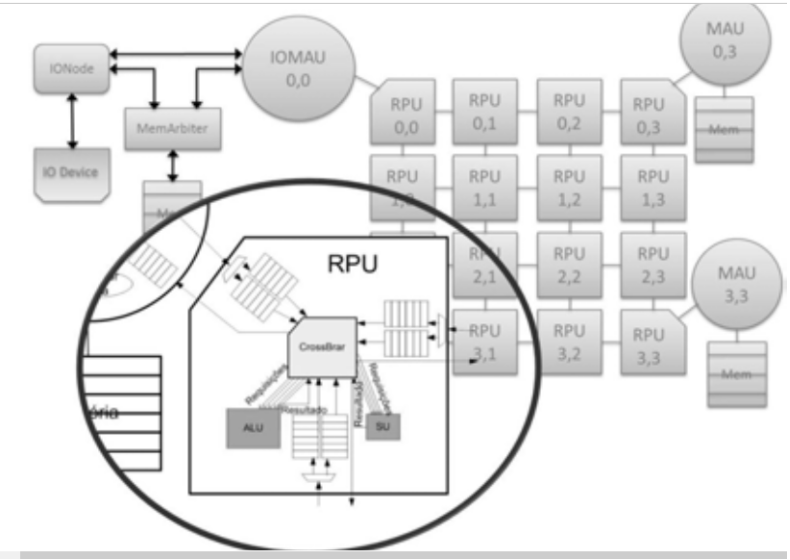

Figura 3: Representação Interna de uma RPU IPNoSys

Fonte: Araújo (2012)

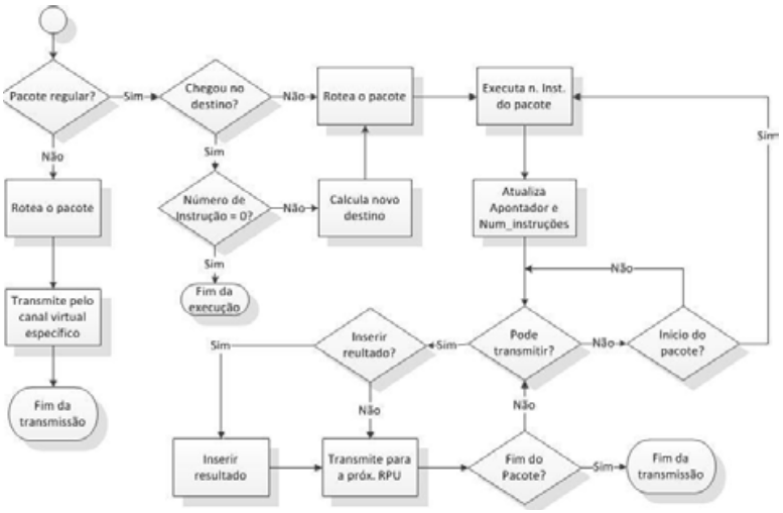

Figura 4: Fluxograma de Funcionamento de uma RPU IPNoSys

Fonte: Araújo (2012)

2.4 Memória

Memórias, segundo Stallings (2010), são unidades de armazenamento temporárias ou permanentes de dados, programas ou instruções para uso posterior do processador. São lugares com espaços de armazenamento de *bits* ou conjunto de *bits* (palavra) que o processador busca e faz referência quando necessita.

Existem basicamente dois tipos de memórias: memórias internas e externas ao processador. As internas são voláteis, perdem os dados quando há ausência de energia, como por exemplo, as memórias *cache*, os registradores e a memória principal. As externas não

são voláteis, como exemplo as memórias ROM (*Ready-Only Memory*), que armazenam os dados permanentemente, não havendo a possibilidade de alteração e exclusão.

Em alguns processadores, existem as memórias *cache* e os registradores, elas são pequenas e de alta velocidade de acesso. Funcionam como um *buffer* de instruções e dados que o processador precisa com mais emergência e rapidez. Na mesma rota está a memória principal, mais lenta e de maior capacidade que as memórias cache e registradores, na qual seu acesso é randômico e é usado quando o processador não encontra as instruções dentro das memórias próximas a ele (Figura 5).

Na arquitetura MIPS, as memórias são os registradores, onde as instruções são guardadas para serem executadas. Já na arquitetura IPNoSys, as instruções em formato de pacotes, são armazenadas nas MAUs, esperando o roteamento por parte das RPUs.

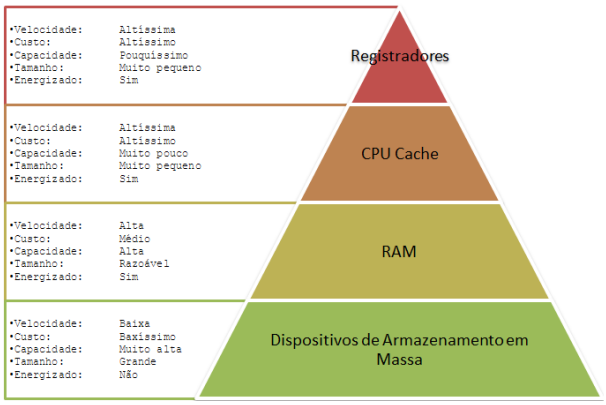

Figura 5: Hierarquia de Memória

Fonte: Stallings (2010)

2.5 Pipeline

Com o avanço dos processadores e da capacidade de aumentar os ciclos de *clock*, se viu necessário criar novas técnicas para que se possa aproveitar o tempo ocioso do processador. Com isso a escalabilidade, eficiência e velocidade de processamento obteve um grande aumento nos processadores atuais.

Antes do surgimento das técnicas de *pipeline*, o processamento das instruções seguia de forma sequencial (Figura 6), e as seções do processador se mantinham ociosas enquanto esperavam por novais instruções. Por conta disso, foi necessário o desenvolvimento de novas técnicas para diminuir a ociosidade do processador e aumentar o desempenho, surgindo assim as técnicas de *pipeline* (STALLINGS, 2010).

De acordo com Shen e Lipasti (2013), o *pipeline* é uma técnica de *hardware* que organiza e executa instruções em estágios, parecidos como uma linha de montagem organizada conforme apresentada na Figura 7. Ele divide o processador em setores e as instruções

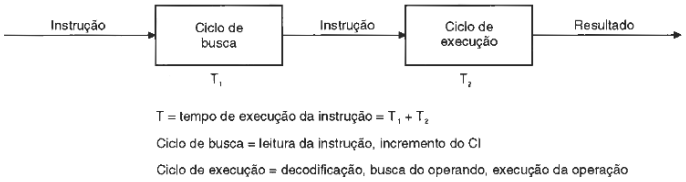

Figura 6: Processamento Sequencial

Fonte: Stallings (2010)

seguem a linha até obter seu processamento. A cada espaço do processador é atribuído uma função específica e diferente das demais, para que ao término de uma instrução, seja capaz de comportar outra de forma imediata.

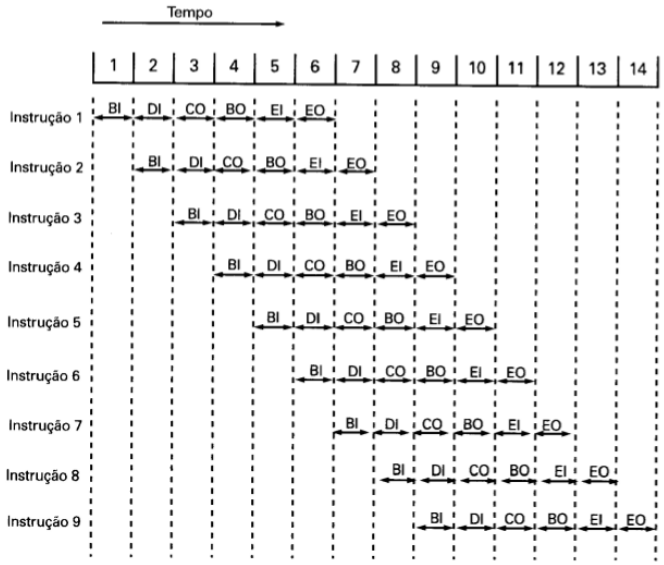

Figura 7: Representação de Pipeline

Fonte: Hennessy (2008)

Na Figura 7 pode-se observar como é dividido o processador, cada lacuna numerada equivale a um estágio em determinado tempo e, cada componente inserido dentro dessas lacunas de tempo, uma instrução. De acordo com Shen e Lipasti (2013), o uso de *pipeline* proporciona ganhos de desempenho significativos, para haver maior ganho é necessário desenvolver mais estágios pois, quanto maior o número de estágios, maior a quantidade de instruções processadas. As instruções são divididas como:

1. Busca de Instrução (BI)
2. Decodificação da Instrução (DI)
3. Cálculo dos Operandos (CO)

4. Busca de Operandos (BO)
5. Execução da Instrução (EI)
6. Escrita do Operando (EO)

A BI busca na memória principal a próxima instrução a ser executada pelo processador e armazena em um *buffer*, o DI determina o código de operação da instrução e obtém referência para cada operando selecionado. O CO determina o endereço de cada operando através de cálculos de endereçamento, para que a BO possa buscar na memória o valor de cada operando. Já a EI executa as instruções e guarda as informações em um *buffer* até que a EO escreva o resultado na memória.

Em arquiteturas MIPS, o *pipeline* possui 5 estágios: busca na memória da próxima instrução, leitura dos registradores enquanto uma instrução é decodificada, execução do processamento, acesso a um operando de memória e escrita do resultado. Segundo Patterson (2005), as instruções são executadas em porções diferentes e especializadas do processador, em um só ciclo são capazes de processar simultaneamente as instruções em todos os estágios, com isso diminuindo a ociosidade do processador.

Em arquiteturas IPNoSys, apresenta um modelo diferente de *pipeline*. Segundo Fernandes et al. (2008), no IPNoSys apresenta uma arquitetura de processadores ligados entre si através de uma rede NoC, abstraindo os estágios de *pipeline* conhecidos nos demais processadores. A transmissão dos pacotes de instruções dentro dessa arquitetura, possibilita a comunicação simultânea entre as RPUs de acordo com a construção desse pacote, demonstrando assim um *pipeline*.

Ambas as arquiteturas descritas neste trabalho trabalham com *buffers*, onde parte das instruções (caso do MIPS) ou pacotes (caso do IPNoSys) são armazenadas para que possam ser processadas ou roteadas para a memória ou outros roteadores.

2.6 Programação e Arquiteturas Paralelas

As arquiteturas de computadores estão em constante evolução, com isso foram criados novos paradigmas que estão sendo utilizados por novas arquiteturas. Com base nisso, os modelos de programação e de arquiteturas paralelas estão tomando mais espaço dentro do mundo tecnológico.

De acordo com MONTEIRO (2002), as arquiteturas com paralelismo surgiram com o objetivo de aumentar o desempenho dos computadores e devido ao fato de que, a limitação de um único processador não poder aumentar indefinidamente seus recursos computacionais. Com o aumento do ciclo de *clock* e com a miniaturização dos circuitos, aumentou a busca por essa característica.

Em arquiteturas MPSoCs, o paralelismo é atrelado em sua construção, fazendo com que seu processamento dê a ideia de ser instantâneo. Assim, arquiteturas desse tipo tendem a tomar lugar das arquiteturas que são, na prática, mais lentas.

Com base nisso, o paralelismo pode ser introduzido em vários níveis como: paralelismo a nível de *chip*, instrução, múltiplas CPUs, multiprocessadores entre outros (MONTEIRO, 2002).

O paralelismo a nível de *chip*, é a capacidade de executar muitas tarefas ao mesmo tempo, destacando duas técnicas: a *multithreading*, que é segundo Stallings (2010), a capacidade de um programa ou processo ser executado em mais de um usuário ao mesmo tempo; e o *multicore*, onde é um único componente capaz de comportar em sua construção, duas ou mais CPUs independentes.

O paralelismo a nível de instrução é baseada no conceito de *pipeline*, ou seja, é a execução simultânea de mais de uma instrução pela CPU, sendo que cada instrução se encontra em um estágio diferente da CPU (MONTEIRO, 2002).

A arquitetura IPNoSys apresenta, de acordo com Araújo (2012), o paralelismo a nível de instrução e *chip*, na qual é explorada a partir dos modelos de programação e dos recursos oferecidos pela arquitetura. O processamento das instruções permite que a arquitetura possa escolher quais RPUs e quais MAUs possam ser utilizadas simultaneamente durante o processamento dos pacotes, dando mais dinamismo e liberdade de processamento.

2.7 Linguagens de Programação

Um conjunto de instruções, como foi abordado anteriormente, é a execução de algoritmos construídos para realizar determinada ação. A construção desses algoritmos segue regras estabelecidas de acordo com metodologias na qual chamamos de linguagens de programação.

Linguagens de programação são, segundo Silva e Assis (1988), métodos padronizados para comunicar um conjunto de instruções para um processador, em outras palavras, é o conjunto de regras sintáticas e semânticas pré-determinadas de acordo com o paradigma trabalhado.

Existem centenas de linguagens de programação, divididas em paradigmas tais como: orientação a objetos, estruturada, modular e linear. Pode se elencar também as linguagens de descrição de *hardware* como SystemC, ArchC e VHDL, linguagens de descrição de pacotes como PDL (*Package Description Language*) e linguagens de máquina, conhecida como *assembly*.

As linguagens de programação em geral, utilizam o compilador para serem executadas. O compilador, segundo Grune et al. (2001), é um programa de computador capaz de traduzir um código fonte escrito em uma linguagem de programação, em um código de

máquina, ou seja, ele irá converter uma linguagem de fácil entendimento do programador para uma linguagem de fácil entendimento do *hardware*.

Elas podem ser divididas em duas categorias: as linguagens de programação de alto nível, ou seja, aquelas que são mais amigáveis ao programador, por possuir elevado nível de abstração, longe do código de máquina e mais perto da linguagem humana; e as linguagens de baixo nível, em outras palavras, com menos níveis de abstração, ligadas diretamente com a arquitetura do computador.

Nesse trabalho, foram estudadas as linguagens PDL e a linguagem *assembly* MIPS, onde foram utilizados compiladores e montadores próprios de cada linguagem. Cada linguagem possui características específicas e diferentes, dependendo de cada arquitetura, ou seja, o código de máquina produzido para MIPS é diferente do código produzido para IPNoSys.

Na Figura 8 e 9 pode-se observar as estruturas sintáticas e semânticas de cada linguagem.

```
1   .text
2
3       la  $a0, info
4       lw  $t0, length
5       sll $t0, $t0, 2
6       add $a1, $a0, $t0
7       jal mergesort
8       b   sortend
9
10  mergesort:
11
12      addi    $sp, $sp, -16
13      sw  $ra, 0($sp)
14      sw  $a0, 4($sp)
15      sw  $a1, 8($sp)
```

Figura 8: Representação da Linguagem *Assembly* MIPS

Fonte: Autor

2.8 MultiProcessor System on Chip

Com o aumento recorrente do número de transistores encapsulados dentro de um único chip, surgiu o desafio de construir novas arquiteturas capazes de comportar e processar grande números de dados, sem haver desperdício de energia e acúmulo de calor. Com isso, foram surgindo os MPSoC's.

O MPSoC é, de acordo com Moreira (2009), uma tecnologia embarcada que possui, em um único chip, muitos núcleos de processamento independentes e interconectados por barramentos ou por uma rede chamada NoC, sendo capaz de processar uma gama enorme de instruções. Esse advento tecnológico surgiu na necessidade de construir sistemas mais rápidos e com grande capacidade de processamento por ciclo de *clock*.

```
PROGRAM prog_0
DATA
    nota1 = 5
    nota2 = 5
    nota3 = 5
    nota4 = 5
    mediaFinal
    aprovado = 0

PACKAGE pac_1
ADDRESS MAU_1
    LOAD MAU_1 n1; //nota 1
        nota1;
    LOAD MAU_1 n2; //nota 2
        nota2;
    LOAD MAU_1 n3; //nota 3
        nota3;
        MUL p1;
        n1  2;
    MUL p2;
        n2  3;
    MUL p3;
        n3  4;
    ADD p4;
        p1 p2;
    ADD p5;
        p3 p4;
    DIV parcial1 parcial2;
```

Figura 9: Representação da Linguagem PDL

Fonte: Autor

Os núcleos são interconectados e os dados a serem processados são divididos entre si, obtendo mais rapidez e dinamismo na execução das instruções. Novas arquiteturas como IPNoSys surgem como candidatas a melhorias, viabilizando mais velocidade e capacidade de processamento. Numa rede NoC, todos os núcleos estão conectados através de uma malha de processamento em rede e fazem na forma de roteamento de pacotes, nesse caso, pacotes de instruções (FERNANDES et al., 2008).

Com o objetivo de atender esta demanda de mercado, os projetos de sistemas embarcados direcionaram-se para o desenvolvimento de MPSoC's com a combinação do paralelismo do processamento com o alto poder de integração proporcionado pela tecnologia system-on-chip (SoC) (GOMES; BARROS, 2007).

A maioria dos sistemas embarcados que existem hoje em dia, utilizam em sua construção, um sistema MPSoC. Segundo Moreira (2009), pode se dizer que a maioria dos MPSoCs tendem a ser heterogêneos, podendo ter vários elementos de processamento, memórias, entre outros.

2.9 Arquitetura MIPS

A arquitetura MIPS foi idealizado por David Patterson em 1980 e projetado por John Hennessy em 1981. Esse processador foi criado com base na concepção de que, ciclos de instruções reduzidos é a melhor forma de criar todas as operações (PATTERSON, 2005).

Os processadores MIPS são, segundo Patterson (2005), trabalhados para suportar instruções do tipo RISC e são construídos em uma arquitetura com uso de registradores (Apêndice guia de programação). São dispostos nesse processador 32 registradores, para

que as instruções utilizem e realizem determinadas operações, mas nem todos os registradores estão à disposição do programador, pois são de uso exclusivo do processador.

Esse tipo de processador é muito utilizado na construção de projetos embarcados e na elaboração de projetos multinúcleo, por ser simples e didático, é utilizado em muitas aplicações como celulares, videogames portáteis entre outros.

As instruções MIPS são bem parecidas com as instruções reconhecidas pelos outros processadores com a diferença que, por ser simples, realiza operações complexas com o uso de operações simples em série. As instruções podem ser de aritmética, lógica, uso de memória, comparações e controle de fluxo (TANENBAUM, 2003), mostradas na Figura 10.

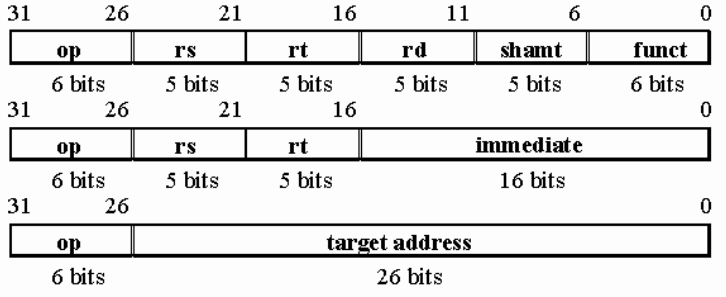

Figura 10: Instruções MIPS

Fonte: Patterson (2005)

A arquitetura converte as instruções em 32 bits, para que assim possam ser utilizadas de forma mais otimizadas. A programação da arquitetura é de forma direta, em conjunto com os registradores, ou seja, a qualquer momento, o programador pode utilizar em seu código, os registradores que são permitidos para uso.

Os registradores em MIPS possuem diferentes funções: o registrador $zero é a constante zero (0), onde o programador pode utilizar para definir como espaço vazio. Registradores do tipo $v são de retorno valores, os de tipo $a são de argumentos, de tipo $t são registradores temporários, que armazenam valores temporários e que serão substituídos ao longo do processamento. Registradores de tipo $s são de variáveis salvas após o processamento, as de tipo $k são de operações temporárias do Sistema Operacional.

Existem outros tipos de variáveis que são utilizadas pelo sistema para armazenamento de *flags, breakpoints* e valores que são decifradas somente pela CPU, fazendo com que o programador não tenha acesso total aos registradores reservados.

O modelo de programação da arquitetura MIPS obedece o paradigma da linguagem de programação de baixo nível, o *assembly* MIPS, onde as instruções estão listadas na Figura 18 no Apêndice guia de programação, mostrando as divisões das instruções MIPS, em quatro categorias: aritmética, transferência de dados, lógico e desvio condicional e incondicional.

2.10 Arquitetura IPNoSys

Nos sistemas de processamento que existentes atualmente, o processamento de dados e o paralelismo (de instruções e dados) são determinantes para que o desempenho seja constante. Com o estudo de novas arquiteturas multinúcleo, se vem mostrando que esse avanço alcançou novas possibilidades de concepção de sistemas completos.

O desenvolvimento da arquitetura IPNoSys, com o uso do avanço da capacidade de comportar muitos núcleos em um chip, é uma realidade viável, dando a possibilidade de construir sistemas complexos e quem sabe, uma arquitetura que irá substituir as que existem atualmente.

A arquitetura IPNoSys é uma das recentes arquiteturas baseadas em MPSoC, que na sua concepção, funciona como uma grande malha de chips de processamento interconectados pela rede NoC com a diferença de um MPSoC apenas no conteúdo dos pacotes de roteamento e na organização dos componentes (ARAÚJO, 2012).

Segundo Fernandes et al. (2008), "No sistema IPNoSys, os roteadores se tomaram RPUs, pois também possuem uma ULA (Unidade Lógica e Aritmética) e uma SU (Unidade de Sincronização) para executar as instruções das aplicações". Os roteadores da rede NoC são capazes de, além de rotear os pacotes da rede para os núcleos de processamento, processar as instruções que estão contidas dentro desses pacotes, fazendo com que possua uma grande rede de processamento e comunicação.

A arquitetura e organização do IPNoSys segue um modelo de malha 2D, o modelo mais simples utiliza uma matriz 4x4 com o uso de uma unidade de processamento e roteamento (Figura 11). Em cada ponta dessa topologia, existe um núcleo responsável para acessar a memória principal do sistema e resgatar os pacotes. Nos quatro cantos da rede, existem estruturas de acesso à memória chamadas de MAU (*Memory Access Unit*), onde as RPUs buscam os pacotes.

O sistema de entrada e saída do IPNoSys (*IONode* e *MemArbiter*) também é ligado a uma das MAUs, a IOMAU. Os pacotes deixam de serem apenas mensagens de requisições e respostas para se tornarem uma estrutura contendo instruções e operandos a serem processadas durante seu percurso (ARAÚJO, 2012).

A arquitetura IPNoSys como utiliza um modelo de malha 2D, suas linhas e colunas são iguais, o roteamento dos pacotes de instruções dessa arquitetura é chamado *Spiral Complement*. Esse algoritmo é capaz de rotear novamente o pacote, quantas vezes for necessário, até que seja garantida a execução de todas as instruções, independentemente do número de RPUs (ARAÚJO, 2012). O roteamento da arquitetura IPNoSys é chamada de Roteamento XY, onde o X e Y são as coordenadas dos roteadores na rede (Figura 12).

Na Figura 12, pode se observar todas as possibilidades de roteamento de pacotes dentro da arquitetura IPNoSys, mostrando o dinamismo do roteamento. Em cada caso, o padrão de roteamento é determinado pela composição do pacote que está trafegando entre as RPUs.

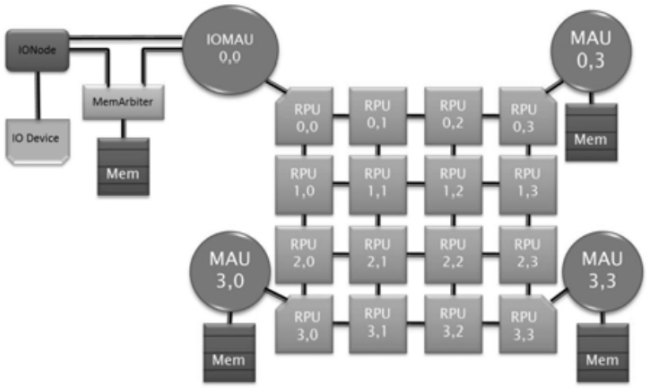

Figura 11: Arquitetura IPNoSys

Fonte: Araújo (2012)

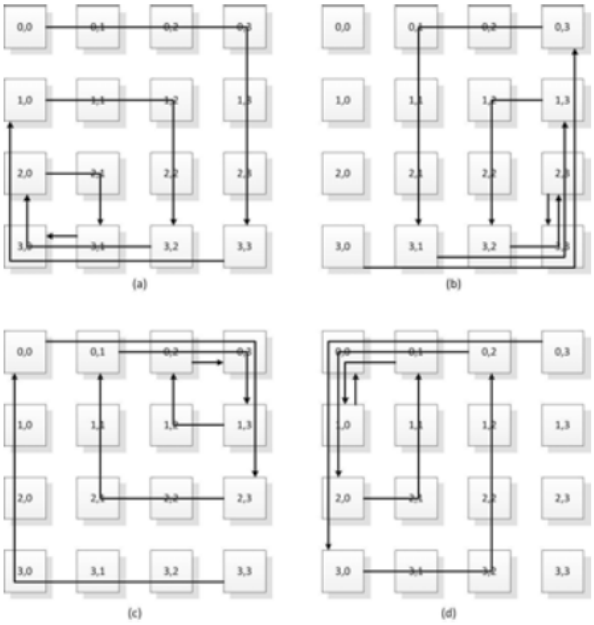

Figura 12: Roteamento XY

Fonte: Araújo (2012)

Fernandes et al. (2008) aponta que esse tipo de arquitetura não utiliza o modelo proposto por John von Neumann, mas utiliza a ideia de paralelismo em nível de instrução e dados e o uso de *pipeline*, fazendo com que os pacotes se tornem uma sequência de instruções sendo distribuídas em cada núcleo de roteamento e processamento.

Na Figura 13 é apresentado o formato geral dos pacotes que são utilizados pela arquitetura para seu funcionamento. As RPUs decodificam os pacotes e processam as instruções contidas.

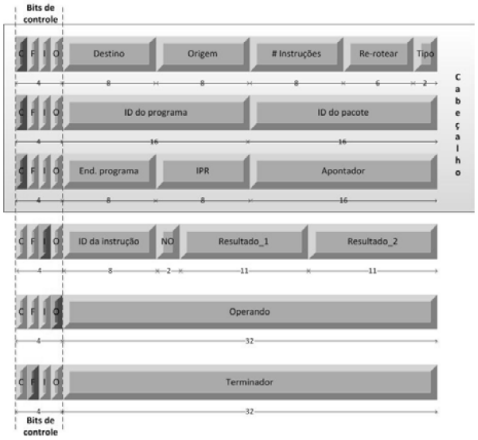

Figura 13: Formato dos Pacotes

Fonte: Araújo (2012)

O conjunto de instruções da arquitetura IPNoSys é representada na seção Apêndice guia de programação, onde a Tabela 7 elenca as principais instruções que são utilizadas para a programação desta arquitetura.

3 Trabalhos Relacionados

Na literatura, muitos trabalhos estão sendo desenvolvidos visando obter maior variedade de projetos com o uso de sistemas multinúcleo. Os assuntos pesquisados para a construção deste trabalho, nenhum apresentou um estudo comparativo entre arquiteturas de processadores diferentes mais utilizadas para o mesmo propósito: agilizar o processamento de dados e/ou instruções.

Dentro da área especifica de arquiteturas que utilizam MPSoC e redes NoC, apresentam alguns trabalhos que são relevantes para introduzir o estudo em questão, conforme apresentado abaixo.

Na pesquisa de (FERNANDES et al., 2008), é apresentado o desenvolvimento da arquitetura IPNoSys dentro de um projeto de sistemas integrados baseados em redes NoC. Segundo essa pesquisa, é expandido o uso e funcionalidades dos roteadores da rede NoC para que seja capaz de executar instruções fragmentadas em pacotes.

É apresentada em (ARAÚJO, 2012), um modelo de programação que comporta a arquitetura proposta, onde dá a capacidade para que, qualquer programador, possa entender e programar na linguagem PDL. O trabalho apresenta todas as características da arquitetura com o sistema operacional, com arquiteturas *Dataflow*, programação paralela entre outros. O trabalho obteve a construção de um compilador e montador da linguagem de programação PDL e do simulador da arquitetura IPNoSys, mostrando a composição da malha de processamento.

Na pesquisa de (OLIVEIRA; SILVA, 2010), descreve uma comparação de modelos de memórias em plataformas MPSoC utilizando a linguagem de descrição de *hardware* systemC, avaliando o impacto dos modelos de memória distribuída e compartilhada dentro dessa arquitetura.

No trabalho de (GOMES; BARROS, 2007), é apresentado um *framework* chamado de PDESIGNER, onde possui um conjunto de ferramentas baseadas em ArchC e SystemC que possibilita a modelagem, simulação e análise de plataformas SoC em diferentes níveis de abstração.

Para estudo da arquitetura MIPS, foi utilizado um artigo e quatro livros conhecidos no meio computacional para ensino de arquitetura de computadores. Eles foram importantes para o entendimento geral da arquitetura de processadores e conjunto de instruções.

O artigo de (MOORE et al., 1965) descreve o funcionamento do ciclo de crescimento dos processadores, desde a década de 60 aos dias atuais. De como a integração entre milhares de transistores em um único chip poderia causar um aumento exponencial de desempenho.

4 Metodologia

A metodologia desenvolvida para a elaboração do modelo comparativo entre as arquiteturas MIPS e IPNoSys foi empregada inicialmente o levantamento dos requisitos para os testes realizados, tais como velocidade de processamento, escalabilidade, consumo de recursos do computador e eficiência em algoritmos de cálculo de média aritmética, algoritmos recursivos e cálculo de fatorial, soma de matrizes e cálculo de quantidade de elementos em um vetor, ambos convertidos em *assembly* MIPS e PDL.

Posteriormente, foram definidos as ferramentas de simulação das arquiteturas, tais como: simulador MARS (*MIPS Assembler and Runtime Simulator*) para a arquitetura MIPS e o simulador IPNoSys IDE, para a arquitetura IPNoSys.

Para a execução dos testes foi utilizado um computador com processador Intel Core i3-3110M com 2.40GHz de frequência de *clock* possuindo 4 núcleos, totalizando 8 *Threads* de processamento, memória *RAM* de 4GB com 1600MHz de velocidade.

4.1 Coleta dos Dados

A coleta dos dados a serem processados pelos algoritmos se deu de forma manual, ou seja, sem a utilização de funções para gerar números aleatórios, sendo que isso teria dificuldade na medição precisa das arquiteturas.

Com os números inseridos, os algoritmos utilizados para a realização dos testes nas duas arquiteturas foram executados e, no final, foi gerado um relatório de consumo de processamento e tempo, onde foi feita a análise de escalabilidade, velocidade e eficiência.

4.2 Simuladores

Os simuladores utilizados no desenvolvimento dos testes foram: o MARS e o IPNoSys IDE, ambas com suas linguagens *assembly* MIPS e PDL, respectivamente.

O simulador MARS foi elaborado pela *Missouri State University* em fevereiro de 2013 e é o simulador dessa arquitetura mais estável e utilizado. Esse simulador possui várias ferramentas de medição como: simulador de dados em *cache*, representação de ponto flutuante, estatísticas de instruções em execução entre outros (Figura 14).

Este simulador apresenta, também, a demonstração em tempo real do uso dos registradores que a arquitetura comporta, de acordo com o projeto desenvolvido, podendo ser acessadas pelo programador a qualquer momento. Na Figura 15 é mostrada a parte do simulador destinada para o acompanhamento dos registradores utilizados.

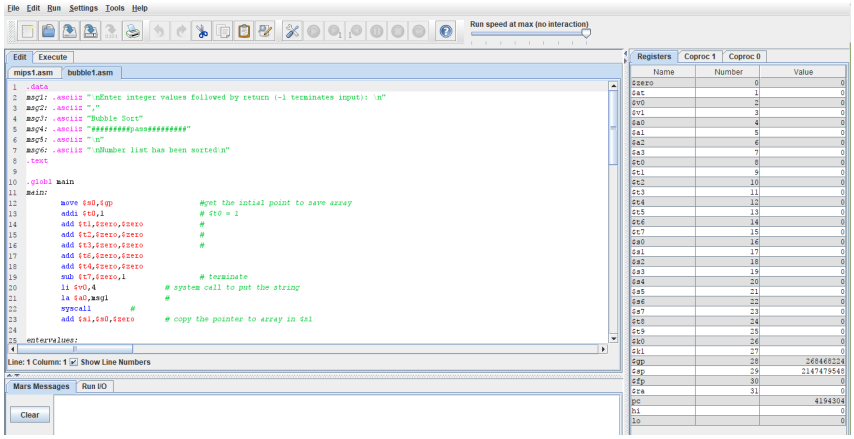

Figura 14: Simulador MARS - Parte de Edição

Fonte: Autor

Na Figura 15 é mostrada a parte de execução das instruções MIPS, mostrado passo a passo as instruções sendo executadas. Também é possível observar a inserção dos resultados de acordo com a execução das instruções.

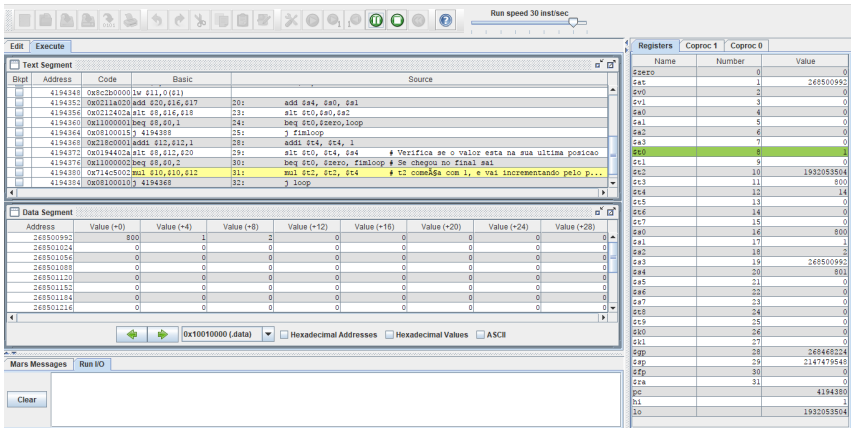

Figura 15: Simulador MARS - Parte de Execução

Fonte: Autor

Já o IPNoSys IDE foi elaborado pela Universidade Federal do Rio Grande do Norte, pelo professor Dr. Sílvio Roberto Fernandes de Araújo. Esse simulador é o único que realiza as simulações da arquitetura IPNoSys e foi elaborada para montar e simular o funcionamento dessa arquitetura. Ela possui a representação gráfica da topologia de rede utilizada pela arquitetura e mostra, em tempo real, o funcionamento e o deslocamento dos pacotes de instruções pelas RPUs, mostrando o quão dinâmica essa arquitetura se apresenta (Figura 16).

Na Figura 17, é descrita a área de trabalho do simulador, onde é possível desenvolver o código fonte do projeto, alterar a IPS (Instruções por Segundo) e as partes essenciais do projeto como salvar, compilar e executar, onde o usuário poderá alterar de acordo com a

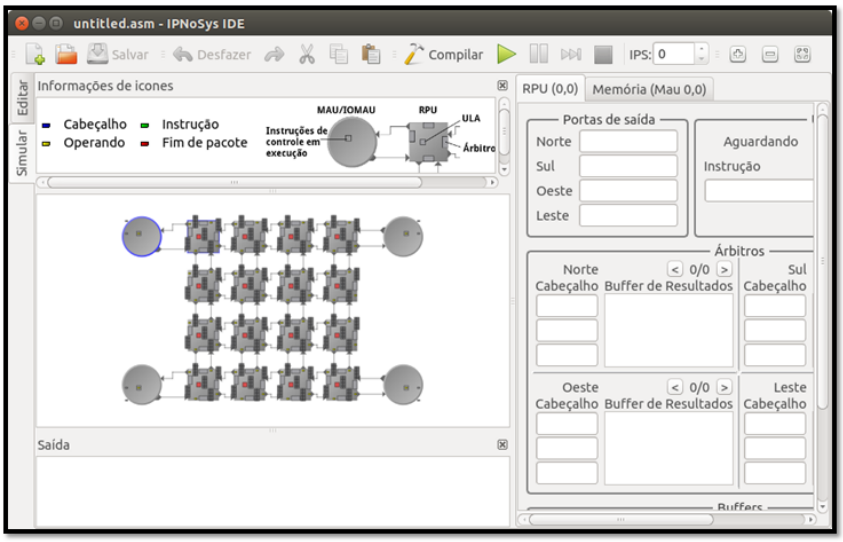

Figura 16: Simulador IPNoSys - Parte de Simulação e Execução

Fonte: Autor

necessidade da simulação.

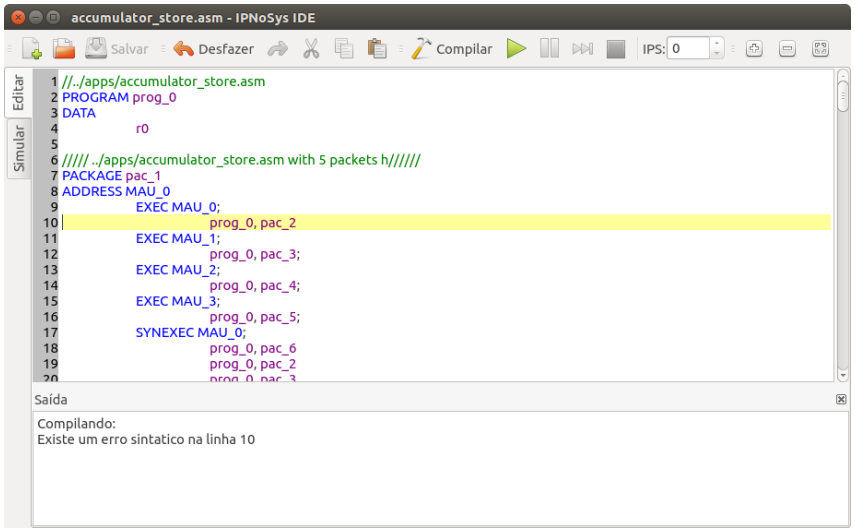

Figura 17: Simulador IPNoSys - Parte de Edição

Fonte: Autor

Os resultados gerados pela simulação, são gravados em um arquivo de texto, onde estão inseridos a quantidade de memória utilizada, ciclos por segundo, MAUs e RPUs utilizadas, entre outros dados.

Além disso, é gerado um arquivo objeto de extensão ipn, onde é mostrada o resultado do processamento das operações matemáticas, em conjunto com a seção de código, com números decimais de 36 bits, descrevendo os pacotes de instruções processadas.

5 Resultados e Discussões

Os testes foram realizados mudando alguns parâmetros essenciais para as arquiteturas. Ambas as arquiteturas foram definidas a quantidade de instruções executadas por segundo como 30, pois assim seria melhor o acompanhamento do processamento pelas arquiteturas.

Nos testes na arquitetura MIPS, foram feitas alterações na quantidade e tamanho de entradas, dando maior *workload* (Carga de trabalho) para o processador. Já na arquitetura IPNoSys além das entradas, foi explorado a quantidade de MAUs utilizadas.

Na execução dos testes, foi medida a quantidade de instruções executadas por cada arquitetura, memória consumida, eficiência média de execução das instruções por ciclo e a quantidade percentual de recursos consumidos pelo computador.

Na execução dos testes, foi observado uma grande diferença na quantidade de memória consumida por ambas as arquiteturas, isso se deve pelo fato de o conjunto de instruções das arquiteturas serem diferentes, já que a arquitetura MIPS converte suas instruções em 32 *bits* e a arquitetura IPNoSys converte em 36 *bits* (32 *bits* para instruções e 4 *bits* para o controle dos pacotes).

Outra característica determinante é o fato de que a arquitetura IPNoSys trabalha com pacote de dados e não com dados simples, como a arquitetura MIPS. A arquitetura IPNoSys processa o pacote contendo as instruções e roteia o restante para as demais RPUs, criando assim uma maior alocação de memória.

Os resultados encontrados pelos testes utilizando os algoritmos descritos na metodologia se deram da seguinte forma:

5.1 Algoritmo de Média Aritmética

Esse algoritmo faz a média aritmética de 5 números, retornando o resultado correto. Na medição dos recursos consumidos pelo processador foram utilizadas ferramentas encontrados no simulador, no caso da arquitetura MIPS, e no relatório gerado pela arquitetura IPNoSys.

Na Tabela 2 mostra a execução do teste na arquitetura MIPS e IPNoSys, mostrando os valores encontrados na execução dos algoritmos de média aritmética.

De acordo com a Tabela 2, a arquitetura MIPS executou 35 instruções, enquanto que a arquitetura IPNoSys executou 20 instruções. A memória consumida foi de 14 *bytes* pelo MIPS e 184 *bytes* pelo IPNoSys. O tempo médio de execução tiveram valores aproximados, mas a arquitetura MIPS se mostrou melhor tanto no tempo médio, quanto no consumo dos recursos do computador.

Tabela 2: Resultados Obtidos - Média Aritmética

Parâmetros	**Código MIPS de Média**	**Código IPNoSys de Média**
Qtd. Instruções Executadas	35 instruções	20 instruções
Memória Consumida (*bytes*)	14 *bytes*	184 *bytes*
Tempo Médio de Execução (inst/ciclo)	1.16 inst/ciclo	0.6 inst/ciclo
Uso Médio dos Recursos do Computador (%)	6.43%	8.31%

Fonte: Autor

5.2 Algoritmo de Recursividade e Cálculo Fatorial

Este teste demonstra a execução de algoritmos que utilizam a recursividade como característica na sua construção e, também, na execução de cálculos fatorial em ambas as arquiteturas, sem recursividade.

Em *assembly* MIPS, foi elaborado um algoritmo de cálculo de combinação de números como, por exemplo, considere um conjunto com seis elementos que são tomados dois a dois, o algoritmo calcula o fatorial dos números, seguindo a regra da combinação simples e retorna o resultado correto.

Na arquitetura IPNoSys, é feito o cálculo de fatorial, onde o algoritmo calcula o fatorial do número escolhido pelo usuário e retorna o resultado correto. Os algoritmos são diferentes um do outro somente por um ciclo de instrução a mais por parte do *assembly* MIPS, mas isso não foi impecílio para a execução dos testes.

Foi incluída também, o algoritmo de cálculo fatorial em *assembly* MIPS e em PDL (para IPNoSys) de forma otimizada e sem recursividade, para que se possa adquirir os dados necessários para a elaboração dos resultados.

Os resultados são apresentados na Tabela 3 e 4. Os códigos apresentados em ambas as arquiteturas, apresentaram a mesma lógica na sua construção, mudando apenas a gramática e a forma de otimização. A ideia base para a construção dos algoritmos, foi baseada na mesma premissa para as duas arquiteturas.

As Tabelas 3 e 4 mostram os resultados da execução de algoritmos com e sem recursão. Na utilização de algoritmos recursivos, a arquitetura MIPS executou 273 instruções, enquanto que a arquitetura IPNoSys executou apenas 42 instruções, isso mostra o nível de otimização de cada arquitetura. A quantidade de memória consumida foi de 73 *bytes* pelo MIPS e 175 *bytes* pelo IPNoSys. Quanto o tempo médio de execução, a arquitetura MIPS se mostrou mais rápida, enquanto que no desempenho médio, a arquitetura IPNoSys utilizou menos recursos do computador.

Tabela 3: Resultados dos Testes - *assembly* MIPS com e sem recursão

Parâmetros	***assembly* MIPS recursivo**	***assembly* MIPS sem recursão**
Qtd. Instruções Executadas	273 instruções	60 instruções
Memória Consumida (*bytes*)	73 *bytes*	59 *bytes*
Tempo médio de Execução (inst/ciclo)	9.1 instr/ciclo	2 instr/ciclo
Uso Médio de Recurso do Computador	11.25%	8.46%

Fonte: Autor

Tabela 4: Resultados dos Testes - Código PDL com e sem recursão

Parâmetros	**PDL recursivo**	**PDL sem recursão**
Qtd. Instruções Executadas	42 instruções	38 instruções
Memória Consumida (*bytes*)	175 *bytes*	164 *bytes*
Tempo médio de Execução (inst/ciclo)	1.4 instr/ciclo	1.2 instr/ciclo
Uso Médio de Recurso do Computador	8.96%	7.23%

Fonte: Autor

Na utilização de algoritmos sem recursão, a arquitetura MIPS executou 60 instruções, enquanto que a arquitetura IPNoSys executou 38 instruções. A quantidade de memória consumida foi de 59 *bytes* pelo MIPS e 164 *bytes* pelo IPNoSys. Quanto o tempo médio de execução, os valores foram bem parecidos, dando destaque a arquitetura MIPS, que executou 1.4 instruções por ciclo. Já o desempenho médio, destaque para a arquitetura IPNoSys, que utilizou 7.23% dos recursos do computador.

5.3 Algoritmo de Contagem de Elementos em um Vetor

O algoritmo proposto faz a contagem dos elementos componentes de um vetor de inteiros. A construção foi feita de forma simples, sem uso de recursividade e já otimizado nas linguagens *assembly* MIPS e PDL. O vetor inicialmente estará preenchido e a capacidade do vetor é flexível, podendo haver expansão ou retração da quantidade de elementos. No vetor, há um ponteiro que percorre as posições e incrementa em um contador. No final, retorna a quantidade de elementos que compõe o vetor. Os resultados são apresentados na Tabela 5.

Tabela 5: Resultado dos Testes - Contagem de Elementos em um Vetor

Parâmetros	**Código *assembly* MIPS**	**Código PDL IPNoSys**
Qtd. Instruções Executadas	68 instruções	98 instruções
Memória Consumida ({bytes})	87 *bytes*	216 *bytes*
Tempo Médio de Execução (inst/ciclo)	2.26 inst/ciclo	3.2 inst/ciclo
Uso Médio dos Recursos do Computador (%)	7.68%	7.91%

Fonte: Autor

A Tabela 5 mostra que foram executadas 68 instruções pelo MIPS e 98 instruções pelo IPNoSys, com o consumo de memória de 87 *bytes* pelo MIPS e 216 *bytes* pelo IPNoSys. O tempo médio de execução mostrou que a arquitetura IPNoSys executou mais instruções por ciclo, mas em questão de desempenho médio, a arquitetura MIPS utilizou menos recursos do que a IPNoSys.

5.4 Algoritmo de Soma de Matrizes

O algoritmo proposto carrega na memória uma matriz de números inteiros e retorna a soma dos elementos da matriz. A construção deste algoritmo, tanto na linguagem PDL, quanto na linguagem *assembly* MIPS foi de forma otimizada e sem recursividade. Os resultados encontrados na realização dos testes são representados na Tabela 6.

Tabela 6: Resultado dos Testes - Soma de Matrizes

Parâmetros	**Código *assembly* MIPS**	**Código PDL IPNoSys**
Qtd. Instruções Executadas	98 instruções	109 instruções
Memória Consumida (*bytes*)	85 *bytes*	322 *bytes*
Tempo Médio de Execução (inst/ciclo)	3.2 inst/ciclo	3.7 inst/ciclo
Uso Médio dos Recursos do Computador (%)	8.32%	6.85%

Fonte: Autor

A Tabela 6 mostra que foram executadas 98 instruções pelo MIPS e 109 instruções pelo IPNoSys, com o consumo de memória de 85 *bytes* pelo MIPS e 322 *bytes* pelo IPNoSys. O

tempo médio e desempenho tiveram valores aproximados, mas com uma vantagem para a arquitetura IPNoSys, pelo fato dela processar mais instruções por ciclo e usar menos recursos do computador.

De acordo com os resultados apresentados, pode-se observar que a arquitetura IP-NoSys, obteve maior eficiência na execução dos algoritmos com uso de recursão e soma de matrizes, ou seja, a arquitetura se mostrou mais eficiente na resolução de problemas complexos e que demandam mais uso de recursos.

Já a arquitetura MIPS obteve maior eficiência nos algoritmos de média aritmética e quantidade de elementos de um vetor, mostrando que a arquitetura é mais eficiente em problemas simples, mas que também se mostra capaz de resolver problemas mais complexos com velocidade e eficiência.

6 Conclusão

Com base nos resultados apresentados, pode se concluir que a arquitetura IPNoSys foi melhor em algoritmos que demandam maior uso de processamento. O uso dessa arquitetura veio para aumentar a velocidade de processamento das instruções e pôr em prática o conceito de paralelismo a nível de instruções, com o uso de redes NoC.

Também de acordo com os testes, pode-se observar que a demanda de desempenho das duas arquiteturas é aproximada e bem parecida, mas quanto menor for o uso de recursos do computador, mais rápido será a execução e, mais uma vez, a arquitetura IPNoSys obteve superioridade em comparação com a arquitetura MIPS.

A arquitetura MIPS se comportou melhor em algoritmos simples, mostrando que também é eficiente em projetos pequenos. Um ponto positivo para a construção dos projetos com MIPS é a facilidade na programação do código, já que o programador tem a oportunidade de construir com otimização.

Com isso em questão de eficiência de processamento, uso de recursos computacionais, a arquitetura IPNoSys superou a arquitetura MIPS. Mas na categoria de escalabilidade, a arquitetura MIPS se mostrou melhor.

Referências

ARAÚJO, S. R. F. d. *Projeto de Sistemas Integrados de Propósito Geral Baseados em Redes em Chip Expandindo as Funcionalidades dos Roteadores para Execução de Operações: A plataforma IPNoSys*. Dissertação (Dissertação de Mestrado), 2012. Citado 6 vezes nas páginas 19, 23, 27, 28, 29 e 30.

CORMEN, T. H. *Introduction to algorithms*. [S.l.]: MIT press, 2009. Citado na página 16.

FERNANDES, S. et al. Ipnosys: uma nova arquitetura paralela baseada em redes em chip. *Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD)*, p. 53–60, 2008. Citado 6 vezes nas páginas 13, 22, 25, 27, 28 e 30.

GOMES, M. d. A. A.; BARROS, E. N. da S. Pdesigner: Framework para modelagem de mpsocs. 2007. Citado 2 vezes nas páginas 25 e 30.

GRUNE, D. et al. Projeto moderno de compiladores: implementação e aplicações. *Rio de Janeiro: Campus*, 2001. Citado na página 23.

HENNESSY, J. L. *Arquitetura de Computadores: Uma Abordagem Quantitativa*. [S.l.]: Elsevier, 2008. Citado 2 vezes nas páginas 15 e 21.

MONTEIRO, M. A. Introdução à organização de computadores, 2a edição. *São Paulo, LTC*, 2002. Citado 3 vezes nas páginas 16, 22 e 23.

MOORE, G. E. et al. Cramming more components onto integrated circuits. *Electronics Magazine*, v. 86, n. 1, p. 114–117, 1965. Citado 2 vezes nas páginas 13 e 30.

MOREIRA, O. *Implementação e avaliação de desempenho de um MPSoC homogêneo interconectado por NoC*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, 2009. Citado 2 vezes nas páginas 24 e 25.

OLIVEIRA, B. C. de; SILVA, I. S. Comparação de modelos de memória para plataformas mpsoc usando systemc. 2010. Citado na página 30.

PATTERSON, D. A. *Organização e Projeto de Computadores: a interface hardware/software*. [S.l.]: Elsevier Brasil, 2005. Citado 5 vezes nas páginas 13, 18, 22, 25 e 26.

SHEN, J. P.; LIPASTI, M. H. *Modern processor design: fundamentals of superscalar processors*. [S.l.]: Waveland Press, 2013. Citado 2 vezes nas páginas 20 e 21.

SILVA, J. C. G. da; ASSIS, F. S. G. de. *Linguagem de programação: conceitos e avaliação*. [S.l.]: McGraw-Hill, 1988. Citado na página 23.

SILVA, L. F.; ANTUNES, V. J. M. Comparação entre as arquitecturas de processadores risc e cisc. *Cidade do Porto, Portugal*, p. 2–6, 2013. Citado na página 16.

STALLINGS, W. *Arquitetura e Organização de Computadores 8a Edição*. [S.l.]: Prentice-Hall, Pearson, 2010. Citado 7 vezes nas páginas 16, 17, 18, 19, 20, 21 e 23.

TANENBAUM, A. *Sistemas operacionais modernos*. [S.l.]: Pearson Education Inc., 2003. Citado na página 26.

Apêndices

APÊNDICE A – Guia de Programação MIPS

Nesta seção, serão descritas as instruções que têm a execução direta na ferramenta de simulação. Isto é, ao contrário das pseudo-instruções que são traduzidas em instruções reais múltiplas antes de ser montada. As instruções serão mostradas na Figura 18.

O termo CONST define o termo constante (imediato) dentro da instrução MIPS.

Categoria	Instrução	Descrição
Aritmética	add	Adiciona dois registros
	addu	Adiciona dois registros sem sinal
	addi	Utilizado para adicionar constantes
	addiu	Utilizado para adicionar constantes sem sinal
	sub	Subtrai dois registradores
	subu	Subtrai dois registradores sem extensão de sinal
	mult	Multiplica dois registradores
	div	Divide dois registradores
Transferência de Dados	lw	Carrega o termo armazenado
	lh	Carrega o meio termo armazenado e o byte seguinte
	lb	Carrega o byte armazenado
	lbu	Carrega o byte armazenado sem extensão de sinal
	sw	Armazena um termo nos registradores
	sh	Armazena a primeira metade em um registrador e o byte seguinte
	sb	Armazena o primeiro quarto de um registro
	lui	Carrega um operador imediato de 16 bits para os 16 bits superiores
	mfhi	Move um valor alto a um registrador
	mflo	Move um valor baixo a um registrador
Lógico	and	Operação lógica E
	andi	Operação E com inserção imediata
	or	Operação lógica OU
	ori	Operação OU com inserção imediata
	xor	Operação OU Exclusivo
	nor	Operação NAO OU
	slt	Testa se um registrador é menor do que o outro
	slti	Testa se um registrador é menor do que uma constante
	srl	Muda o número CONST de bits para a direita - os zeros são deslocados
	sra	Muda o número CONST de bits, o bit de sinal é deslocado
Desvios Condicional e Incondicional	beq	Vai para a instrução no endereço especificado se for igual
	bne	Vai para a instrução no endereço especificado se for diferente
	j	Pula incondicionalmente para o endereço especificado
	jr	Pula para o endereço contido no registrador
	jal	Usada para chamar uma subrotina retornando o endereço

Figura 18: Guia de Programação MIPS

Fonte: Autor

APÊNDICE B – Guia de Programação IP-NoSys

Nesta seção, é mostrada o guia de programação utilizada na arquitetura IPNoSys em geral, mostrando algumas semelhanças com a arquitetura MIPS, mas possuindo instruções de roteamento e paralelismo específicas como EXEC, SYNEXEC e SYNC.

Tabela 7: Guia de Programação IPNoSys

Código	**Instrução**	**Operando**	**Tipo**	**Descrição**
0	ADD	2	Aritmética	Soma
1	SUB	2	Aritmética	Subrai
2	MUL	2	Aritmética	Multiplica
3	DIV	2	Aritmética	Divide
4	NOT	1	Lógico	Negação
5	AND	2	Lógico	Conjunção
6	OR	2	Lógico	Disjunção
7	XOR	2	Lógico	Ou-Exclusivo
8	RSHIFT	2	Deslocamento	Desloca a direita
9	LSHIFT	2	Deslocamento	Desloca a esquerda
10	LOAD	1	Acesso a memória	Solicita um valor da memória
11	STORE	Vários	Acesso a memória	Armazena um valor na memória
12	EXEC	1	Sincronização	Injeção imediata de um pacote
13	SYNEXEC	Vários	Sincronização	Injeção de um pacote após sincronização
14	SYNC	1	Sincronização	Sinal de sincronização
15	RELOAD	1	Acesso a memória	Retorna um valor carregado na memória
16	BE	2	Condicional	Desvia se igual
17	BNE	2	Condicional	Desvia se diferente
18	BL	2	Condicional	Desvia se menor
19	BG	2	Condicional	Desvia se maior
20	BLE	2	Condicional	Desvia se menor ou igual
21	BGE	2	Condicional	Desvia se maior ou igual
22	JUMP	0	Incondicional	Desvio incondicional
23	COPY	1	Auxiliar	Copia instrução para o mesmo pacote
24	NOP	0	Auxiliar	Sem operação
25	SEND	2	Sincronização	Envia um valor a ser inserido no pacote
26	IN	3	Entrada/Saída	Receber *bytes*
27	OUT	3	Entrada/Saída	Enviar *bytes*
28	WAKEUP	1	Sistema	Reinjeta pacote interrompido
29	NOTIFY	1	Sistema	Notifica estado de um pacote
30	CALL	Vários	Procedimento	Faz a chamada de uma função/procedimento
31	RETURN	2	Procedimento	Retorna o resultado de uma função

Fonte: Autor

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, Lucas Ibiapino Alves, autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação Um Estudo Comparativo entre Arquiteturas MIPS e IPNoSys de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI 29 de Junho de 2017.

Lucas Ibiapino Alves

Assinatura