ORGANIZAÇÃO DE COMPUTADORES

TRABALHO 2

Iuri Rodrigues Seifriz
Instituto Federal do Rio Grande do Sul
Tecnologia em Análise e Desenvolvimento de Sistemas – 2° Semestre

Introdução

O MIPS (Microprocessor without Interlocked Pipeline Stages) é uma arquitetura de processador com um conjunto de instruções RISC (Reduced Instruction Set Computing), bastante usada em sistemas embutidos, simuladores e como plataforma educacional. O modelo monociclo de MIPS descreve a execução de uma instrução dentro de um único ciclo de relógio. A seguir, apresento um resumo sobre o caminho de dados e as instruções no MIPS monociclo, com foco nas operações, controle e uso de registradores.

1. Caminho de Dados Monociclo

Em uma implementação monociclo de MIPS, todas as instruções são executadas em um único ciclo de relógio, com a condição de que o tempo do ciclo deve ser suficiente para acomodar a operação mais lenta. O caminho de dados é simplificado e pode ser descrito pelas seguintes unidades:

1.1. Busca de Instrução

- **Memória de Instruções:** As instruções são armazenadas em memória e são recuperadas pelo **PC** (contador de programa). O PC é incrementado para buscar a próxima instrução após cada ciclo.
- **Somador:** Um somador calcula o endereço da próxima instrução para alimentar o PC.

1.2. Banco de Registradores

- O MIPS tem um conjunto de 32 registradores de 32 bits, com operações de leitura e escrita.
- Cada registrador pode ser acessado diretamente, e o valor de um registrador pode ser usado como entrada para outras operações.

1.3. ULA (Unidade Lógica e Aritmética)

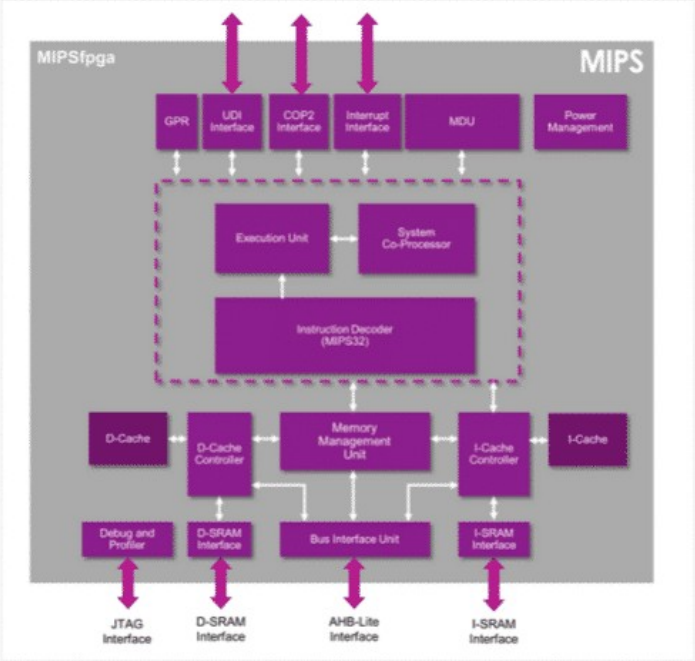
- Realiza operações aritméticas e lógicas, como soma (para `add` e `lw`), subtração (para `beq`), e comparações (para `slt`).
- A ULA é controlada por sinais de controle que são derivados do tipo de instrução.

1.4. Memória de Dados

- A memória de dados é usada para armazenar e recuperar dados (tipicamente com instruções `lw` e `sw`).
- As instruções de load e store são usadas para transferir dados entre registradores e memória.

1.5. Controle

- O controle é realizado por uma unidade de controle que gera sinais baseados no tipo de instrução. Isso determina quais unidades do caminho de dados serão ativadas.
- Exemplos de controle incluem **RegWrite**, **MemRead**, **MemWrite**, **ALUOp**, e **ALUSrc**.



2. Tipos de Instruções em MIPS

Existem três tipos principais de instruções no MIPS, que se diferenciam pelo formato de seus campos:

2.1. Instruções Tipo-R

Essas instruções envolvem operações aritméticas ou lógicas entre registradores, como:

- `add`, `sub`, `and`, `or`, `slt` (set less than).

Formato de instrução tipo-R:

```
op (6 bits) | rs (5 bits) | rt (5 bits) | rd (5 bits) | shamt (5 bits) | funct (6 bits)
```

Onde:

- `rs` e `rt` são registradores fonte.
- `rd` é o registrador de destino.
- `shamt` é utilizado para instruções de deslocamento.
- `funct` especifica a operação (por exemplo, soma ou subtração).

Exemplo:

```
add $s0, $s1, $s2
```

Isso soma os valores de `$s1` e `$s2`, armazenando o resultado em `$s0`.

2.2. Instruções Tipo-I

Essas instruções são usadas para operações com um imediato (valor constante), como as instruções de carregamento e armazenamento, e comparações. Exemplos incluem `lw` (load word) e `sw` (store word).

Formato de instrução tipo-I:

```
op (6 bits) | rs (5 bits) | rt (5 bits) | imediato (16 bits)
```

Exemplo:

```
lw $t0, 32($s3)
```

Isso carrega o valor da memória no endereço calculado como `32 + $s3` para o registrador `$t0`.

2.3. Instruções Tipo-J

Essas instruções são usadas para desvio (jumps). O campo de destino contém um endereço que será concatenado com os bits mais altos do PC.

Formato de instrução tipo-J:

```
op (6 bits) | endereço (26 bits)
```

Exemplo:

```
j 1000
```

Isso faz com que o PC seja atualizado para o endereço 1000, efetuando um salto incondicional.

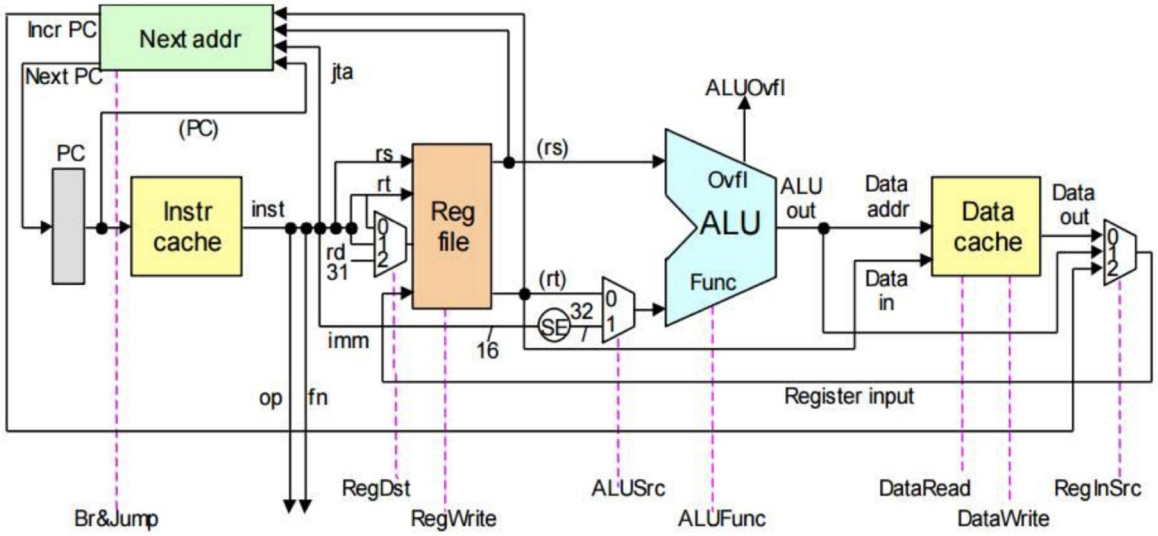
3. Controle da ULA

A Unidade Lógica e Aritmética (ULA) tem um controle que depende do campo `funct` e de um sinal chamado **ALUOp**. Esse controle determina a operação que a ULA deve realizar (como soma ou comparação).

Tabela de Controle para a ULA

A ULA pode realizar operações como:

- **Soma** (para `lw`, `sw`).
- **Subtração** (para `beq`).
- **Operações aritméticas e lógicas** (para instruções tipo-R, como `add`, `sub`, `and`).



4. Controle de Fluxo

O controle de fluxo é feito através de instruções de desvio:

- **Condicionais:** `beq` (branch if equal), `bne` (branch if not equal).
- **Incondicionais:** `j` (jump).

Exemplo:

```
beq $s1, $s2, Label
```

Isso compara `$s1` e `$s2`. Se forem iguais, o PC será atualizado para o endereço de `Label`.

5. Considerações de Implementação

- **Tempo do Ciclo:** O ciclo de relógio precisa ser longo o suficiente para acomodar a instrução mais lenta. Isso significa que todas as unidades de processamento devem ser sincronizadas para garantir que todas as operações necessárias sejam realizadas no mesmo ciclo.
- **Memória de Instruções e Dados:** O MIPS monociclo usa duas memórias separadas: uma para armazenar as instruções e outra para os dados. Isso é necessário para garantir que a execução da instrução não interfira no acesso à memória.

6. RISC-V e sua Implementação no Processador Milk V

A **arquitetura RISC-V** (Reduced Instruction Set Computing, versão 5) é uma arquitetura aberta e padronizada, projetada para ser simples, flexível e altamente escalável. Lançada inicialmente em 2010 pela Universidade da Califórnia, Berkeley, a RISC-V surgiu como uma alternativa de código aberto às arquiteturas proprietárias como ARM e x86. Seu design modular permite que os desenvolvedores escolham conjuntos de instruções específicos para suas necessidades, desde sistemas embarcados até servidores de alto desempenho.

Características Principais da Arquitetura RISC-V

1. **Conjunto de Instruções Modular e Extensível**: A arquitetura RISC-V possui um conjunto base de instruções que pode ser expandido por meio de extensões. Isso permite que ela seja customizada para diferentes tipos de dispositivos e aplicações.
2. **Simplicidade e Eficiência**: O design de RISC-V foca em operações simples e eficientes, otimizadas para execução rápida e com menor consumo de energia.
3. **Aberta e Livre de Royalties**: Ao ser uma arquitetura aberta, RISC-V permite que qualquer pessoa ou empresa implemente seus próprios processadores sem custos com licenciamento, o que a torna atraente tanto para a academia quanto para o mercado de tecnologia.
4. **Alta Escalabilidade**: A arquitetura foi projetada para ser escalável, desde sistemas embarcados com recursos limitados até servidores de grande porte e supercomputadores.

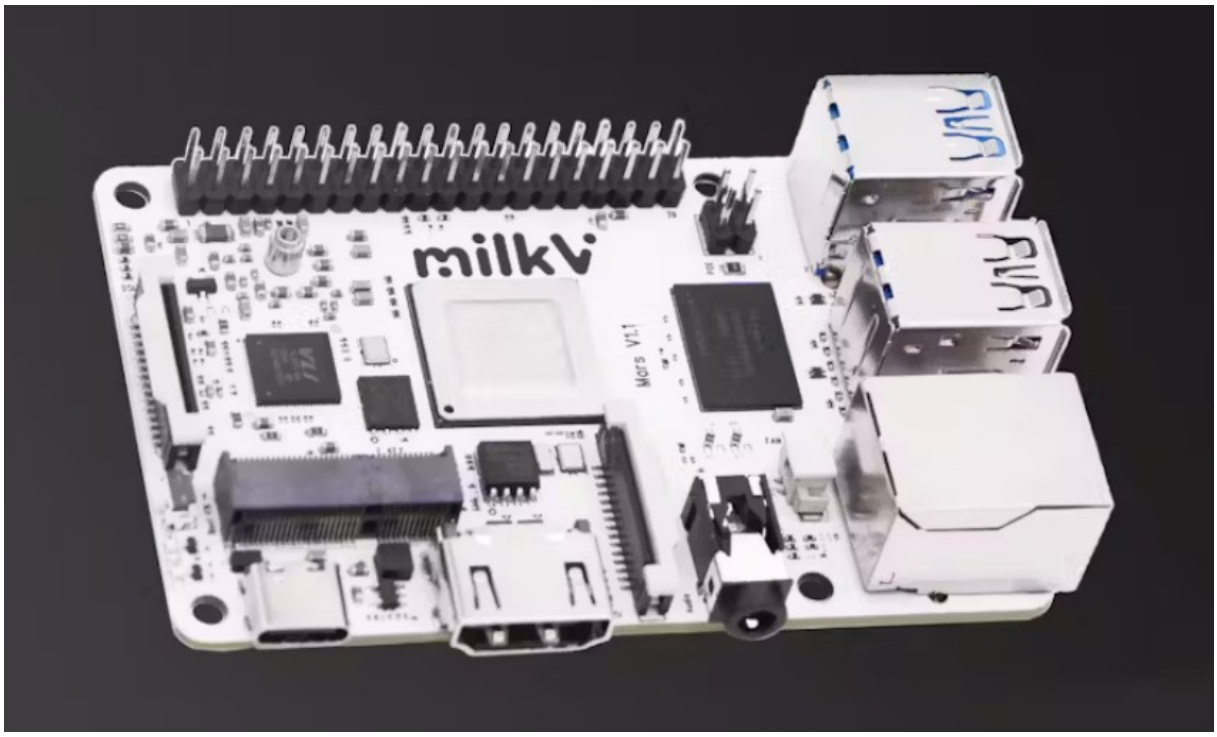
A Implementação do Processador Milk V

O **Milk V** é uma implementação específica da arquitetura RISC-V, desenvolvida para demonstrar o potencial dessa arquitetura em ambientes de alto desempenho e em dispositivos de consumo. Seu design foi inspirado pela necessidade de criar um processador de alto desempenho, com baixo consumo de energia e alto nível de personalização.

O Milk V é projetado para suportar uma ampla gama de operações e aplicações, podendo ser utilizado em dispositivos móveis, servidores e até em sistemas embarcados. Ele se destaca por integrar várias extensões da arquitetura RISC-V, o que permite um equilíbrio entre poder de processamento e eficiência energética. Além disso, a implementação do Milk V enfatiza:

- **Compatibilidade com as Extensões RISC-V**: O Milk V é compatível com as principais extensões da arquitetura RISC-V, como a extensão "V" para vetores e a extensão "M" para operações de multiplicação e divisão, permitindo maior flexibilidade de uso em uma variedade de cenários.
- **Design de Núcleo Escalável**: O processador Milk V possui um núcleo altamente escalável, capaz de ser configurado para diferentes necessidades, seja em termos de número de núcleos ou de características específicas de processamento.
- **Suporte a Operações em Baixa Potência**: Como a arquitetura RISC-V é projetada com eficiência energética em mente, o Milk V é particularmente eficiente em termos de consumo de energia, tornando-o ideal para aplicações em dispositivos móveis e IoT (Internet das Coisas).
- **Open Source e Personalização**: Uma das grandes vantagens do Milk V é que ele segue a filosofia aberta do RISC-V, permitindo que empresas ou desenvolvedores possam modificar e personalizar o processador conforme suas necessidades.

Em resumo, a implementação do **Milk V** baseado na arquitetura **RISC-V** representa um avanço significativo na adoção de processadores abertos, oferecendo uma plataforma flexível, eficiente e adaptável a uma variedade de aplicações modernas.



7. Conclusão

A arquitetura MIPS monociclo é simples e eficaz para entender os conceitos fundamentais de um processador RISC. A execução das instruções ocorre de forma direta, com cada unidade funcional envolvida na execução de cada instrução. No entanto, o design monociclo apresenta limitações de desempenho devido à necessidade de que todas as unidades funcionais sejam sincronizadas para um único ciclo de relógio, o que pode resultar em um tempo de ciclo mais longo.

Além da RISC-V com sua proposta open source, temos Ryzen, baseada em múltiplos núcleos e ARM, para produtos de baixo consumo energético, como smartphones. Modelos de arquiteturas modernas, que buscam melhorar o desempenho de processadores, são diversas, e podem oferecer diferentes soluções para determinadas necessidades de um mundo tão dependente de microprocessadores.