Departamento de Engenharia Elétrica e de Computação

SEL 606 – Lab. de Sistemas Digitais

Prof. Dr. Maximiliam Luppe

Kollins Gabriel Lima

PRÁTICA Nº3

Circuitos Sequenciais – Máquinas de Estados Finitos

Problema

Implementar uma Unidade de Controle de um processador baseado na arquitetura MIPS utilizando uma Máquina de Estados Finitos - MEF.

Arquitetura MIPS de 32 bits

A arquitetura MIPS é um exemplo de arquitetura com conjunto reduzido de instruções (RISC) desenvolvido na década de 1980. Ela tem instruções e barramento de dados de 32 bits e trinta e dois registradores de 32 bits de uso geral.

A execução de uma instrução pela arquitetura exige que ela passe por diversas etapas até que se chegue a um resultado. Para tanto, se faz necessário o controle dos diversos componentes para que estas etapas sejam realizadas da maneira correta. A Unidade de Controle (UC) é a parte responsável por fornecer todos os sinais necessários e gerenciar os demais componentes e será o último grande módulo do MIPS a ser montado.

A arquitetura MIPS tem apenas três formatos de instrução: **I**, **R** e **J-Format**. Somente as instruções **I-Format** referenciam operandos da memória. Instruções **R-Format** realizam operações somente em dados nos registradores. Elas requerem dois registradores de operandos, Rs e Rt. O resultado da operação é armazenado num terceiro registrador, Rd. Os campos Shift e Function da **R-format** são usados como um campo de instrução estendida. Instruções **J-Format** incluem as instruções de desvio. Por questões de simplificação, a arquitetura terá um barramento de dados de 8 bits e um barramento de instruções de 16 bits. Na tabela 1 é apresentado o formato, também simplificado, das instruções.

Tabela 1 – Formatos de Instruções do MIPS 32 bits

Field Size	5-bits	2-bits	2-bits	2-bits	2-bits	4-bits
R-Format	Opcode	Rs	Rt	Rd	Shift	Function
I-Format	Opcode	Rs	Rt	Address/Immediate value		
J-Format	Opcode	Branch target address				

O conjunto das instruções básicas da arquitetura simplificada MIPS que será implementada é mostrado na tabela 2.

Tabela 2 – Instruções básicas do processador MIPS

Mnemonic	Format	Opcode Field	Function Field	Instruction
Add	**R**	**0 (00000)**	**8 (1000)**	**Add**
Addi	**I**	**8 (01000)**	**-**	**Add Immediate**
Sub	**R**	**0 (00000)**	**10 (1010)**	**Subtract**
And	**R**	**0 (00000)**	**12 (1100)**	**Bitwise And**
Or	**R**	**0 (00000)**	**13 (1101)**	**Bitwise Or**
Slt	**R**	**0 (00000)**	**14 (1110)**	**Set If Less Than**
Lw	**I**	**19 (10011)**	**-**	**Load Word**
Sw	**I**	**27 (11011)**	**-**	**Store Word**
Beq	**I**	**4 (00100)**	**-**	**Branch on Equal**
Bne	**I**	**5 (00101)**	**-**	**Branch on Not Equal**
J	**J**	**2 (00010)**	**-**	**Jump**

Implementação Ciclo Único

Uma implementação do processador MIPS, baseado no exemplo do livro *Computer Organization and Design The Hardware/Software Interface*, de Patterson e Hennessy, é mostrado na figura 1. Esta implementação do MIPS realiza a busca, decodificação e execução de instruções em um único ciclo de clock. O Contador de Programa (PC) é utilizado para buscar o próximo endereço na memória de instrução. Uma vez que a memória é endereçada byte a byte, o PC é incrementado de 4 para buscar a próxima palavra de 32 bits na memória. Ao mesmo tempo em que a instrução é buscada, um somador (acima da memória) é usado para somar 4 ao PC para gerar o próximo endereço. A saída da memória de instrução é a próxima instrução de 32 bits.

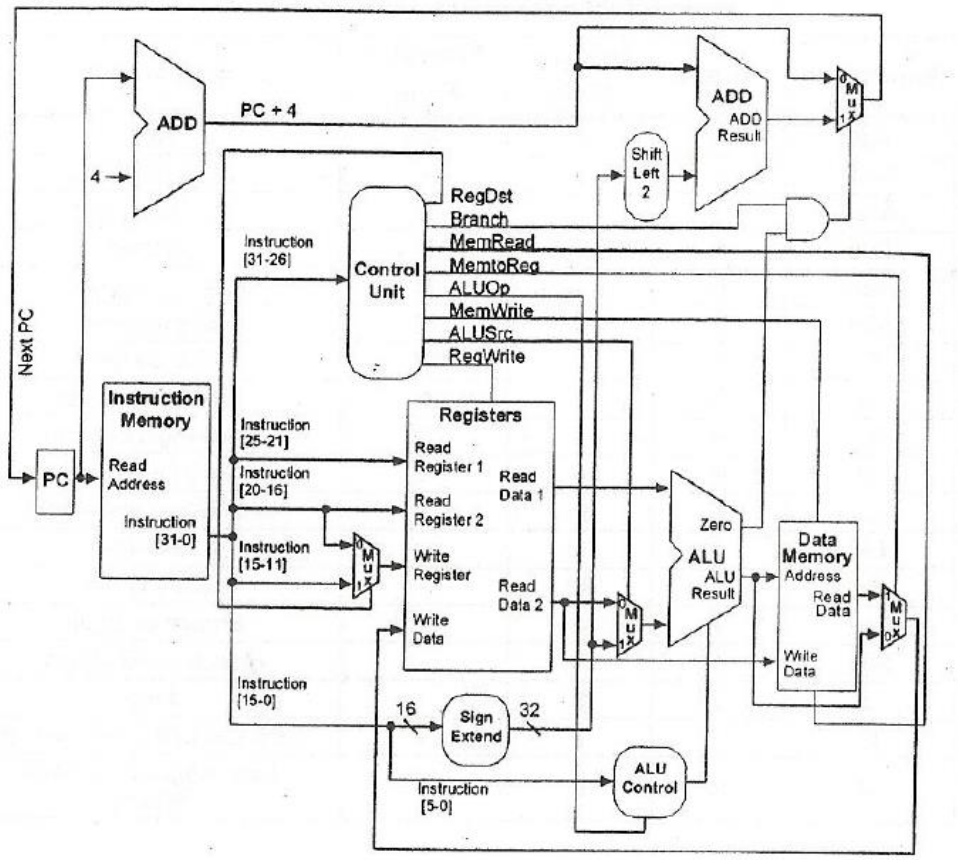

Figura 1 - Esquema do processador MIPS (*Computer Organization and Design The Hardware/Software Interface*)

O código da instrução é enviado à unidade de controle e o código da função é enviado à unidade de controle da ULA. Os campos de endereço de registradores da instrução são usados para endereçar o banco de registradores de duas portas. Este banco pode realizar duas leituras e uma escrita independentes em um único ciclo de clock. Isto implementa a decodificação de instrução.

As duas saídas do banco de registradores alimentam as entradas de dados da ULA. As unidades de controle definem a operação da ULA necessária para a execução da instrução. Depois, as instruções LOAD e STORE leem ou escrevem na memória de dados. Instruções **R-Format** passam diretas pela memória de dados utilizando um multiplexador. Por último, as instruções de leitura e **R-Format** escrevem de volta um novo valor no banco de registradores.

Instruções de desvios relativos ao PC usam o somador e o multiplexador mostrados acima da ULA na figura 1 para calcular o endereço de desvio. O multiplexador é necessário para operações de desvio condicional. Depois que todas as saídas tiverem sido estabilizadas, o próximo clock carrega o novo valor do PC e o processo repete para a próxima instrução.

Além da implementação de ciclo único, o MIPS pode ser implementado em *pipeline* (as instruções são executadas em cinco etapas concorrentes: *Fetch*, *Decode*, *Execute*, *Memory* e *Write back*) e multiciclo (as instruções são executadas em vários ciclos de clock).

Implementação Multiciclo

Na figura 2 está a representação completa da arquitetura MIPS, sem simplificação. Ela é capaz de executar as seguintes instruções: add, sub, and, or, slt, lw, sw, beq, addi, e j. O processador multiciclo é dividido em três unidades: controle (formado pela *Control Unit* e lógica adicional), memória (formado pela *Instr/Data Memory*), e *datapath* (demais circuitos). Observe que a unidade de memória será responsável pelo armazenamento tanto das instruções e como dos dados. Observe também que a unidade de controle contém o decodificador principal (que tem como entrada os sinais Op5:0) e o decodificador da ULA (que tem como entrada os sinais ALUOp1:0 e Funct5:0). Os sinais Op5:0, ALUOp1:0 e Funct5:0 são oriundos da instrução que o processador irá executar e são fornecidos pelo Registrador de Instruções. A unidade de controle também inclui a lógica necessária para produzir o sinal de habilitação PCEn, para o registrado PC.

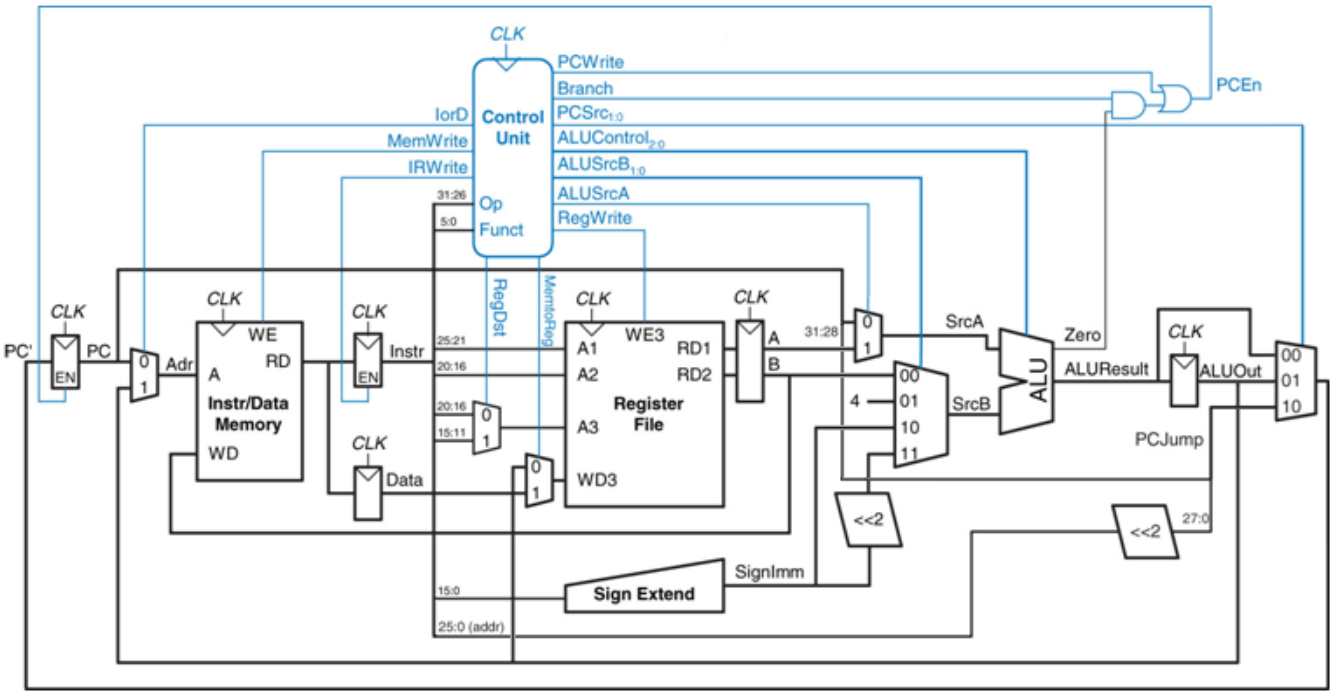

Figura 2 - Esquema do processador MIPS multiciclo (*Digital Design and Computer Architecture*)

Para a arquitetura MIPS multiciclo, a UC será uma máquina de estados finitos que recebe parte da instrução como entrada (opcode) e como saída possui os seguintes sinais:

Tabela 3 – Sinais de Controle

Sinal	Efeito se Desativado	Efeito se Ativado
Branch		Habilita PC para escrita se flag Zero da ULA estiver ativado
PCWrite		Habilita PC para escrita
IorD	PC é utilizado para endereçar a memória	ALUOut é utilizado para endereçar memória
MemRead		Conteúdo da memória endereçado é colocado na saída de dados
MemWrite		Habilita Memória para Escrita
MemtoReg	Entrada de dados do Banco de Registradores é o ALUOut	Entrada de dados do Banco de Registradores é o MDR
IRWrite		Habilita escrita no registrador IR
RegWrite		Habilita escrita no Banco de Registradores
RegDst	Endereço de escrita no Banco de Registradores vem do campo rt da instrução	Endereço de escrita no Banco de Registradores vem do campo rd da instrução
PCSource	Seleciona fonte de dados para escrita no PC	
ALUOp	Seleciona operação da ULA (com auxílio do ALUControl)	
ALUSrcA e ALUSrcB	Selecionam operandos para ULA	

Tabela 3

Para ilustrar como a UC controla o processador para executar uma instrução tome como exemplo a instrução LW (Load Word) que lê um dado da memória e o coloca no banco de registrador. Primeiro é preciso realizar a leitura da instrução e colocá-la no IR. O PC também é incrementado neste ciclo. Para que os dados possam realizar este caminho, é necessário:

- PCWrite = 1
- MemRead = 1
- ALUSrcA = 0
- IorD = 0
- IRWrite = 1
- ALUSrcB = 01
- ALUOp = 00
- PCSource = 00

A instrução precisa depois ser decodificada e isso será feito pela UC no segundo ciclo. Neste meio tempo, os operandos são carregados e o endereço de *branch* é calculado (caso seja esta a instrução, ela poderá ser completada no próximo ciclo).

- ALUSrcA = 0
- ALUSrcB = 11
- ALUOp = 00

Depois da decodificação, cada instrução segue uma sequência de etapas diferentes. Para a LW, chega o momento de calcular o endereço da memória em que o dado se encontra, para isso é utilizado o registrador A como base e o campo imediato da instrução. Repare na figura que a partir deste momento, as funções da ULA (ALU Control) são decididas com base no ALUOp e no campo de função das instruções. Os sinais de controle serão:

- ALUSrcA = 1
- ALUSrcB = 10
- ALUOp = 00

Com o endereço calculado, é realizado o acesso à memória para buscar o dado solicitado. O dado é armazenado temporariamente no MDR para poder ser escrito no banco de registradores no próximo ciclo. Serão ativados apenas os sinais:

- MemRead = 1
- IorD = 1

Por fim, o dado buscado e armazenado no MDR é escrito no banco de registradores, no endereço passado pela instrução (bits 20-16). A instrução então completa sua execução e o ciclo recomeça. Nesta última etapa serão necessários:

- RegWrite = 1
- MemtoReg = 1
- RegDst = 0

Problema

Como visto, as instruções passam por etapas diferentes em função do seu tipo e cabe a UC guiar cada instrução ao longo de sua execução. A figura 3 apresenta o diagrama de estados da UC do MIPS para as instruções tipo R, LW, SW, BEQZ e J. Repare que os primeiros 2 estados são comuns para todas as instruções.

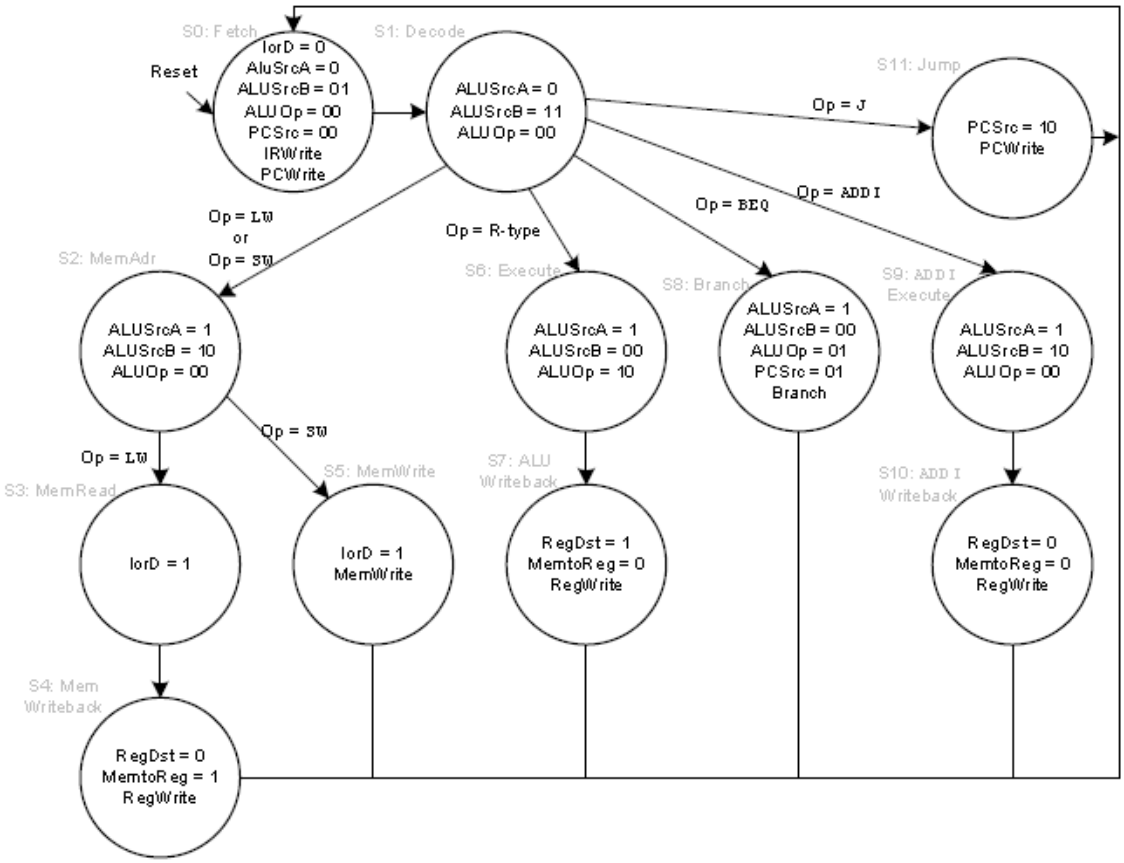

Figura 3 - Diagrama de Transição de Estados da arquitetura MIPS multiciclo

ERRATA: No estado 4, MemtoReg = 1 e RegDst = 0

Nesta e na próxima prática será projetado e implementada a UC de uma arquitetura simplificada MIPS multiciclo para implementar um processador completo. Algumas das estruturas necessárias para isso já foram implementadas em práticas passadas, como a ULA e o Banco de Registradores. Nesta, daremos continuidade às práticas de PBL com a implementação da unidade de controle da arquitetura simplificada MPIS por meio de uma MEF.

Portanto, o problema proposto nesta prática é a construção de uma máquina de estados seguindo o diagrama apresentado pela figura 3. A máquina deve conter uma entrada de 5 bits que representará o opcode da instrução as saídas contidas na tabela 1. Inclua também um reset (em nível alto) para o sistema.

Gerando os Sinais de Controle

Antes de começar a implementação da máquina de estados para a arquitetura simplificada do MIPS multiciclo, é necessário determinar os corretos sinais de controle para cada estado no Diagrama de Transição de Estados do processador multiciclo, apresentado na Figure 3.

Complete os dados de decodificação de saída do Decodificador Principal na Tabela 5, na folha de respostas. Obtenha a palavra de controle da MEF em hexadecimal para cada estado. As duas primeiras linhas estão preenchidas como exemplo. Tenha cuidado com este passo. Demora muito mais tempo para corrigir um circuito errado do que projetá-lo corretamente na primeira vez.

Implementação da Unidade de Controle

A Unidade de Controle (figura 4) é a parte mais complexa do processador multiciclo. Ela consiste de duas partes, o Decodificador Principal (*Main Decoder*) e o Decodificador da ULA (*ALU Decoder*). O Decodificador Principal tem como entrada o campo Opcode da instrução e deve produzir as saídas descritas na tabela 4, conforme o estado na EMF da figura 3.

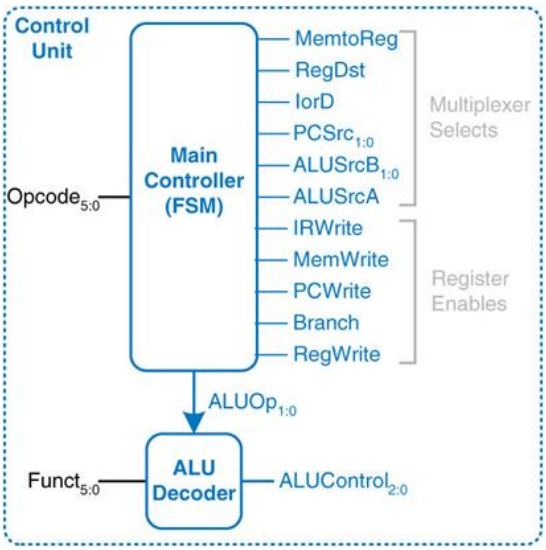

Figura 4 – Unidade de controle

ERRATA: $Opcode_{4:0}$ e $Funct_{3:0}$

O Decodificador da ULA tem como entradas o campo Funct da instrução e os sinais ALUOp do Decodificador Principal e deve gerar as saídas, para o correto funcionamento da ULA, conforme a instrução decodificada, de acordo com a tabela 4 abaixo.

Tabela 4 – Conjunto de operações da ULA

$ALUOp_{1:0}$	**$Funct_{3:0}$**	**$ALUControl_{2:0}$**	**Função**
00	**-**	-	Soma
01	**-**	-	Subtração
10	**12 (1100)**	000	A and B

10	**13 (1101)**	001	A or B
10	**8 (1000)**	010	A + B
10	**10 (1010)**	110	A – B
10	**14 (1110)**	111	SLT
11	-	-	-

Departamento de Engenharia Elétrica e de Computação

SEL 606 – Lab. de Sistemas Digitais

Prof. Dr. Maximiliam Luppe

Kollins Gabriel Lima

FOLHA DE RESPOSTAS: PRÁTICA Nº3

NOTA:

Circuitos Sequenciais – Máquinas de Estados Finitos

DATA:

NOMES:	Nº USP

Tabela 5 – Sinais de saída da FSM

State (Name)	PCWrite	MemWrite	IRWrite	RegWrite	ALUSrcA	Branch	IorD	MemtoReg	RegDst	ALUSrcB[1:0]	PCRsc[1:0]	ALUOp[1:0]	FSM Control Word
0 (Fetch)	1	0	1	0	0	0	0	0	0	01	00	00	0x5010
1 (Decode)	0	0	0	0	0	0	0	0	0	11	00	00	0x0030
2 (MemAdr)													
3 (MemRd)													
4 (MemWB)													
5 (MemWr)													
6 (RtypeEx)													
7 (RtypeWB)													
8 (BeqEx)													
9 (AddiEx)													
10 (AddiWB)													
11 (JEx)													