

Processador

Profa. Débora Christina Muchaluat Saade
debora@midia.com.uff.br

<http://www.midia.com.uff.br/debora>

1

Processador

- ✓ Capítulo 6 do Livro do Mario Monteiro
- ✓ Capítulo 5 (5.1, 5.2, 5.3 e 5.4) do Livro do Patterson

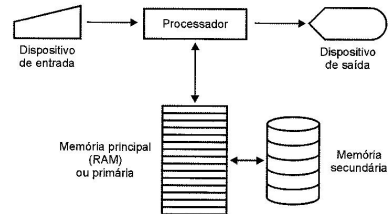


Figura 2.1 Componentes básicos de um computador.

2

Ciclo de Instrução Básico

✓ Processador executa instruções

- CPU (central processing unit)
- UCP (unidade central de processamento)

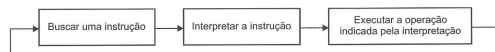


Figura 6.3 Funcionamento do hardware do processador – ciclo de instrução básico.

3

Ciclo de Instrução Básico

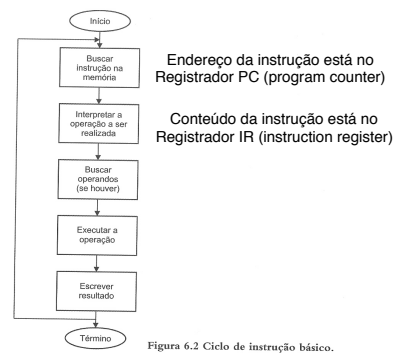


Figura 6.2 Ciclo de instrução básico.

Funções do Processador

- ✓ Funções de processamento
- ✓ Funções de controle

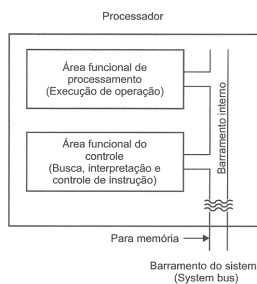


Figura 6.4 Diagrama em bloco (básico) de um processador.

Funções de Processamento

- ✓ Unidade Lógica e Aritmética (ULA, UAL, ALU)
- ✓ Registradores de dados e registradores especiais de estado (PSW – program status word)

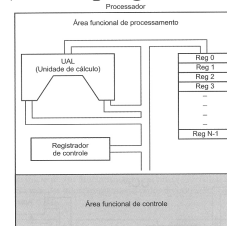
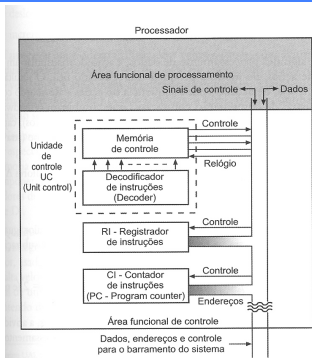


Figura 6.6 Componentes da área funcional de processamento.

Funções de Controle

Fundamentos de Arquiteturas de Computadores



- ✓ Unidade de controle, relógio (clock)
- ✓ Registradores especiais
 - PC – program counter (CI – contador de instruções)
 - IR – instruction register (RI – registrador de instrução)
 - REM – registrador de endereços de memória (MAR)
 - RDM – registrador de dados de memória (MDR)

Figura 6.8 Componentes básicos da área funcional de controle.

Unidade de Controle

Fundamentos de Arquiteturas de Computadores

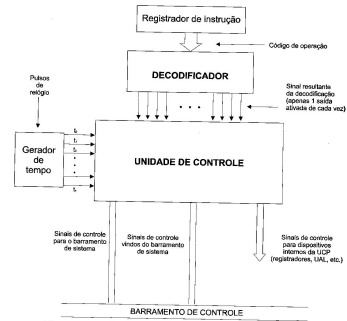
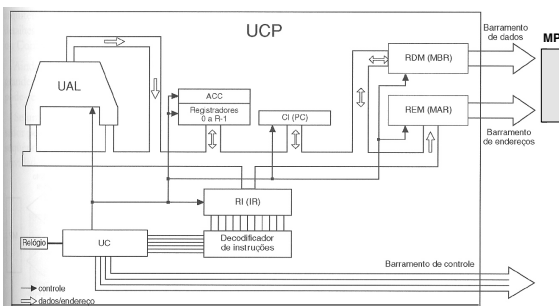


Figura 6.15 Diagrama em bloco simplificado da função controle.

Esquema Simplificado da CPU

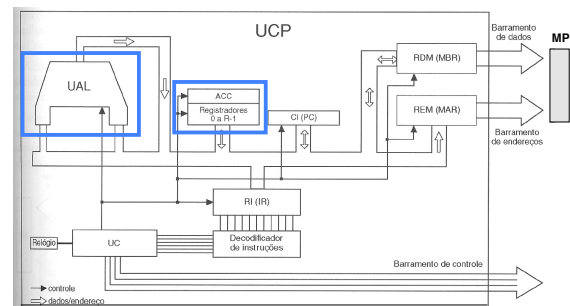
Fundamentos de Arquiteturas de Computadores



9

Componentes com Função de Processamento

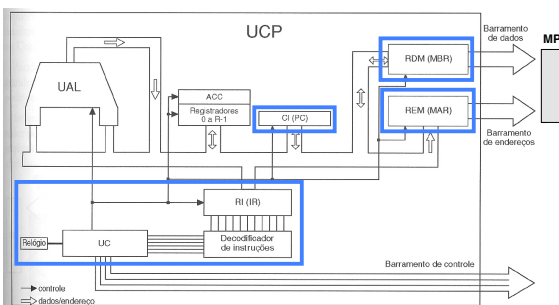
Fundamentos de Arquiteturas de Computadores



10

Componentes com Função de Controle

Fundamentos de Arquiteturas de Computadores



11

Implementação da UCP

Fundamentos de Arquiteturas de Computadores

- ✓ Construção do caminho de dados
- ✓ Controle
- ✓ Implementação monociclo

12

Projeto Lógico

Fundamentos de Arquiteturas de Computadores

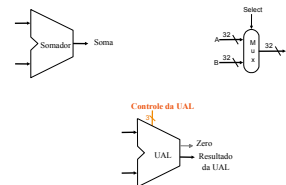
✓ Duas definições importantes

- **Elemento Combinacional** - saída depende somente das entradas
 - Exemplo: ALU
- **Elemento Sequencial**: elementos contêm informações de estado
 - Exemplo: Registradores

19

Elementos Combinacionais

Fundamentos de Arquiteturas de Computadores



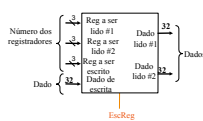
20

Banco de Registradores

Fundamentos de Arquiteturas de Computadores

✓ Contém 8 registradores

- **Dois barramentos de 32 bits de saída**
 - Dado lido #1 e Dado lido #2
- **Um barramento de 32 bits de entrada**
 - Dado a ser escrito
- **Registrador 0 tem o valor 0**
- **Registrador selecionado por**
 - Reg a ser lido #1
 - Reg a ser lido #2
 - Reg a ser escrito seleciona registrador que recebe Dado a ser escrito quando EscReg=1

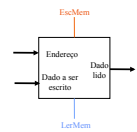


21

Memória

Fundamentos de Arquiteturas de Computadores

- ✓ **Um barramento de entrada: Dado a ser escrito**
- ✓ **Um barramento de saída: Dado lido**
- ✓ **Seleção de endereço**
 - **Endereço seleciona a palavra a ser colocada em Dado lido**
 - **Para escrever no endereço, seta EscMem para 1**
 - **Para ler do endereço, seta LerMem para 1**



22

Passos do Projeto

Fundamentos de Arquiteturas de Computadores

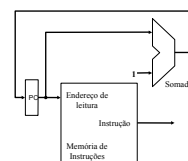
- ✓ De acordo com a arquitetura do conjunto de instruções, define-se uma estrutura organizacional macro (número de unidades funcionais, por exemplo)
- ✓ Essa estrutura é refinada para definir os componentes do caminho de dados, sua interconexão e pontos de controle
- ✓ Estrutura de controle é definida
- ✓ O projeto do caminho de dados e controle é refinado para projeto físico e validação funcional

23

Busca da Instrução

Fundamentos de Arquiteturas de Computadores

- ✓ Busca a instrução na memória, cujo endereço está no contador de programa PC
- ✓ Incrementa o contador de programa PC de 1



24

Instrução de Soma

Fundamentos de Arquiteturas de Computadores

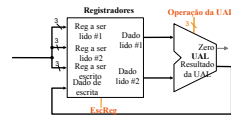
- ✓ **add regA regB destreg**
 - $Mem[PC]$ *Obtém instrução da memória*
 - $R[destreg] \leftarrow R[regA] + R[regB]$ *Executa operação de soma*
 - $PC \leftarrow PC + 1$ *Calcula próximo endereço*

25

Caminho de Dados para Instruções do tipo R

Fundamentos de Arquiteturas de Computadores

- ✓ $R[destreg] \leftarrow R[regA] \text{ op } R[regB]$
 - Controle da UAL e de EscReg baseado na instrução decodificada
 - Reg a ser lido #1, Reg a ser lido #2, Reg a ser escrito são regA, regB, destreg



26

Instrução de Carga

Fundamentos de Arquiteturas de Computadores

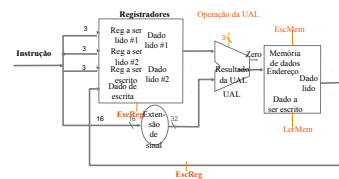
- ✓ **lw regA regB imm**
 - $Mem[PC]$ *Busca instrução na memória*
 - $End \leftarrow R[regA] + SignExt(imm)$ *Calcula o endereço da memória*
 - $R[regB] \leftarrow Mem[End]$ *Carrega os dados no registrador*
 - $PC \leftarrow PC + 1$ *Calcula o próximo endereço*

7 bits	3 bits	3 bits	3 bits	16 bits
0	op	regA	regB	imediato

27

Caminho de Dados para Instrução de Carga

Fundamentos de Arquiteturas de Computadores



28

Instrução de Armazenamento

Fundamentos de Arquiteturas de Computadores

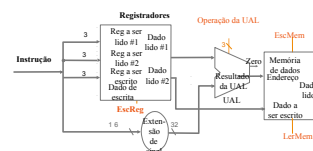
- ✓ **sw regA regB imm**
 - $Mem[PC]$ *Busca instrução na memória*
 - $End \leftarrow R[regA] + SignExt(imm)$ *Calcula o endereço da memória*
 - $Mem[End] \leftarrow R[regB]$ *Carrega os dados na memória*
 - $PC \leftarrow PC + 1$ *Calcula o próximo endereço*

7 bits	3 bits	3 bits	3 bits	16 bits
0	op	regA	regB	imediato

29

Caminho de Dados para Instrução de Armazenamento

Fundamentos de Arquiteturas de Computadores



30

Instrução de Desvio Condicional

Fundamentos de Arquiteturas de Computadores

✓ beq regA regB imm

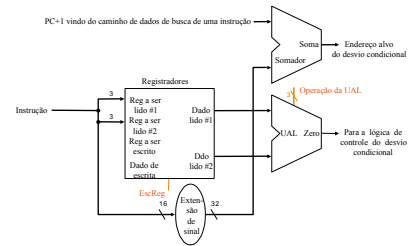
- $mem[PC]$ *memória* *Busca instrução na memória*
- $Cond \leftarrow R[regA]-R[regB]$ *Calcula a condição de desvio*
- $if(Cond eq 0)$
 - $PC \leftarrow PC+1 + SignExt(imm)$ *Calcula endereço PC relativo*
- *else*
 - $PC \leftarrow PC+1$ *Calcula o próximo endereço*

7 bits	3 bits	3 bits	3 bits	16 bits
0	op	regA	regB	imediato

31

Instrução de Desvio Condicional

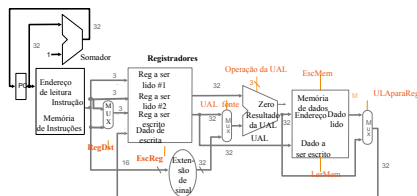
Fundamentos de Arquiteturas de Computadores



32

Busca de Instrução e Instruções Aritm. e Lóg. e de Referência à Memória

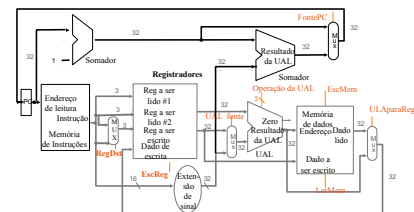
Fundamentos de Arquiteturas de Computadores



33

Caminho de Dados para Suportar Subconjunto das Instruções

Fundamentos de Arquiteturas de Computadores

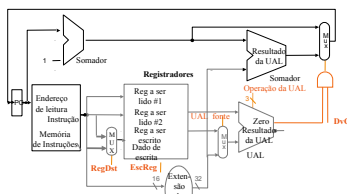


34

Caminho de Dados para Carregar PC

Fundamentos de Arquiteturas de Computadores

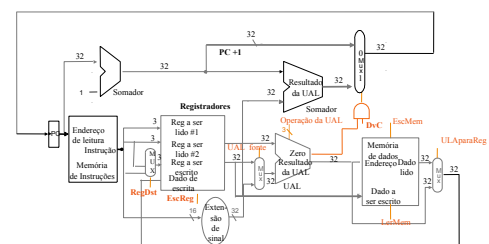
- ✓ PC incrementado normalmente
- ✓ Se instrução for beq
 - *pode adicionar imm a PC + 1*



35

Juntando Todas as Partes

Fundamentos de Arquiteturas de Computadores



36

Inserindo o Controle

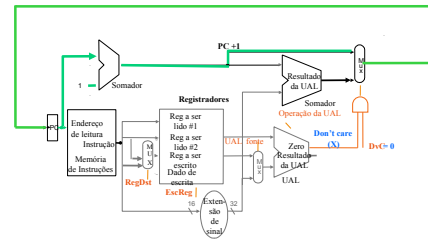
Fundamentos de Arquiteturas de Computadores

- ✓ Identifica pontos de controle no caminho de dados
 - *Busca da instrução*
 - *Operações lógicas e aritméticas*
 - *Memória*
- ✓ Identifica tipo de controle do sinal
 - *Fluxo de dados através de multiplexadores*
 - *Escrita de dados*
- ✓ Deriva sinais de controle para cada instrução
- ✓ Coloca todos os sinais juntos

37

Busca da Instrução

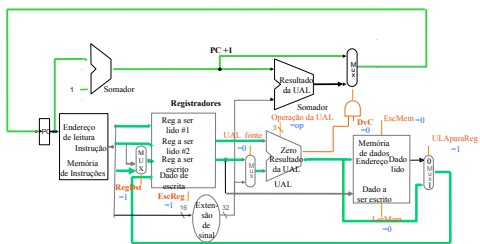
Fundamentos de Arquiteturas de Computadores



38

Controle para Operação Aritmética

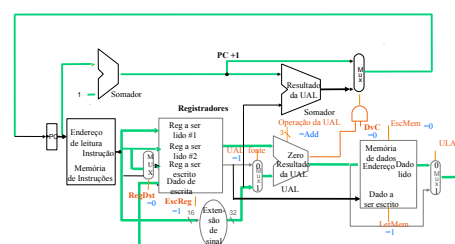
Fundamentos de Arquiteturas de Computadores



39

Controle para Operação de Carga

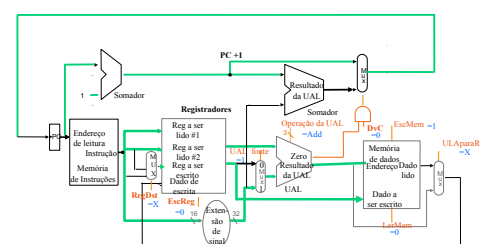
Fundamentos de Arquiteturas de Computadores



40

Controle para Operação de Armazenamento

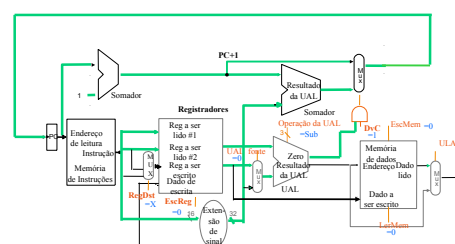
Fundamentos de Arquiteturas de Computadores



41

Controle para Operação de Desvio Condicional (beq)

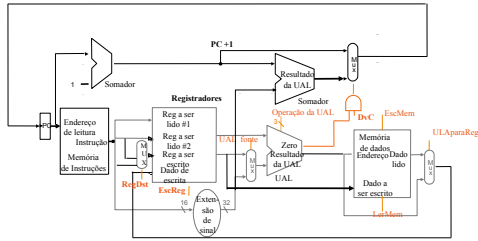
Fundamentos de Arquiteturas de Computadores



42

Sinais de Controle

Fundamentos de Arquiteturas de Computadores



43

Resumo dos Sinais de Controle

Fundamentos de Arquiteturas de Computadores

op	000	010	011	100
add	add	lw	sw	beq
RegDst	1	0	X	X
UALfonte	0	1	1	0
ULAparaReg	1	0	X	X
EscReg	1	1	0	0
LerMem	0	1	0	0
EscMem	0	0	1	0
DvC	0	0	0	1
UALOp	Add	Add	Add	Sub

44

Tabela-verdade para os Três Bits de Controle da UAL

Fundamentos de Arquiteturas de Computadores

Código de Operação	Op2	Op1	Op0	Operação da UAL
lw	0	1	0	010 (soma)
sw	0	1	1	010 (soma)
beq	1	0	0	110 (subtração)
Tipo R(add)	0	0	0	010 (soma)

45

Função de Controle para Implementação Monociclo

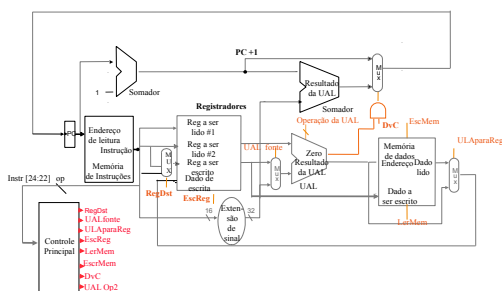
Fundamentos de Arquiteturas de Computadores

op	000	010	011	100
add	add	lw	sw	beq
RegDst	1	0	X	X
UALfonte	0	1	1	0
ULAparaReg	1	0	X	X
EscReg	1	1	0	0
LerMem	0	1	0	0
EscMem	0	0	1	0
DvC	0	0	0	1
UALOp0	0	0	0	0
UALOp1	1	1	1	1
UALOp2	0	0	0	1

46

Juntando as Partes

Fundamentos de Arquiteturas de Computadores



48

Implementação Monociclo

Fundamentos de Arquiteturas de Computadores

- ✓ **Vantagens**
 - Um ciclo de relógio por instrução torna lógica mais simples
- ✓ **Desvantagens**
 - Ciclo de clock determinado pela instrução que leva maior tempo
 - Instrução de carga utiliza cinco unidades funcionais em série
 - tempo de acesso à memória de instruções +
 - tempo de acesso ao banco de registradores +
 - retardo da UAL +
 - tempo de acesso à memória de dados +
 - tempo de estabilidade dos dados para o banco de registradores
 - Duplicação de unidades funcionais

49

Execução de Programas

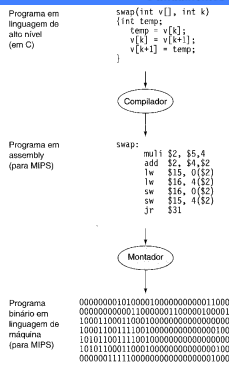
Profa. Débora Christina Muchaluat Saade
debora@midia.com.uff.br

<http://www.ic.uff.br/~debora/fac>

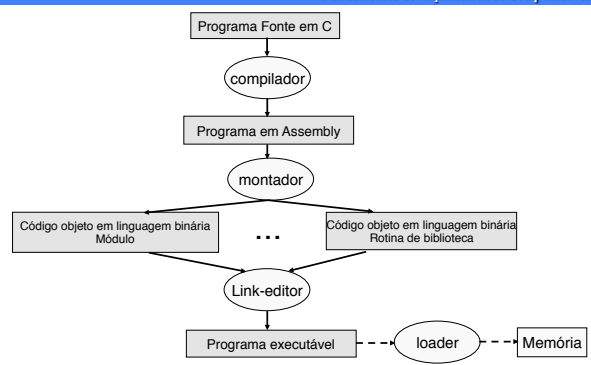
Execução de Programas

- ✓ Um programa é escrito em uma linguagem que a máquina não entende
 - *Linguagem de alto nível*
- ✓ É necessário traduzir o programa para a linguagem binária

Exemplo



Geração do executável



Compilação

- ✓ **Compilador traduz o programa escrito em linguagem de alto nível para linguagem de montagem**
- ✓ **Compilador executa três funções:**
 - **Análise léxica:**
 - decompõe o programa fonte em seus elementos individuais distintos (comandos, operadores, variáveis) e verifica se estão de acordo com as regras da linguagem
 - **Análise sintática:**
 - cria as estruturas para os comandos e verifica a correção das estruturas (por exemplo, tem int antes de main, tem abre e fecha parênteses)
 - **Análise semântica:**
 - verifica as regras semânticas estáticas da linguagem (por exemplo, estar atribuindo valores do tipo correto para uma variável)
 - Ex.: *Parcela1=1.5*, poderia dar erro pois variável *Parcela1* é do tipo *int*

Montagem

- ✓ **Montador realiza a tradução de um programa em linguagem de montagem (código fonte) para linguagem binária (código objeto)**
- ✓ **Funções básicas:**
 - *Substituir códigos de operações simbólicos por valores numéricos*
 - *Substituir nomes simbólicos de endereços por valores numéricos*
 - *Reservar espaço de memória para armazenamento de instruções e dados*
 - *Converter valores de constantes para binários*
 - *Examinar a correção de cada instrução*

Link-edição ou ligação

Fundamentos de Arquiteturas de Computadores

- ✓ Link-editor ou ligador
- ✓ Algumas rotinas já existem codificadas no sistema, por exemplo, rotinas para entrada e saída de dados, de modo que o programador não precisa codificá-las.
- ✓ Exemplo:
 - *printf()*,
 - *readln()*,
 - *rotinas matemáticas como seno e cosseno.*
- ✓ Estas rotinas são armazenadas em bibliotecas que são incluídas no código objeto gerado pelo compilador.
- ✓ Para que o código da subrotina externa seja anexado ao código objeto utiliza-se o processo de ligação.
 - *Este processo examina todo o código objeto e procura por referências externas não resolvidas assinaladas pelo compilador.*
 - *Ele procura pela rotina no sistema e, se a encontra, inclui seu código no código objeto, caso contrário, gera uma mensagem de erro.*

56

Programas interpretados

Fundamentos de Arquiteturas de Computadores

- ✓ Linguagem interpretada
- ✓ Realiza as três fases (compilação, ligação e execução) comando a comando do programa fonte.
- ✓ Um programa fonte é executado pelo programa interpretador.

57

Compilação x Interpretação

Fundamentos de Arquiteturas de Computadores

- ✓ Erros são detectados de forma mais fácil na interpretação
- ✓ Utiliza-se mais memória na interpretação porque interpretador tem que estar na memória
- ✓ Na interpretação, loop tem que ser traduzido em todas as iterações
- ✓ Programas muito utilizados são traduzidos toda vez que são executados na interpretação

58