

Processador

Profa. Débora Christina Muchaluat Saade

debora@midiacom.uff.br

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

Processador

Fundamentos de Arquiteturas de Computadores

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

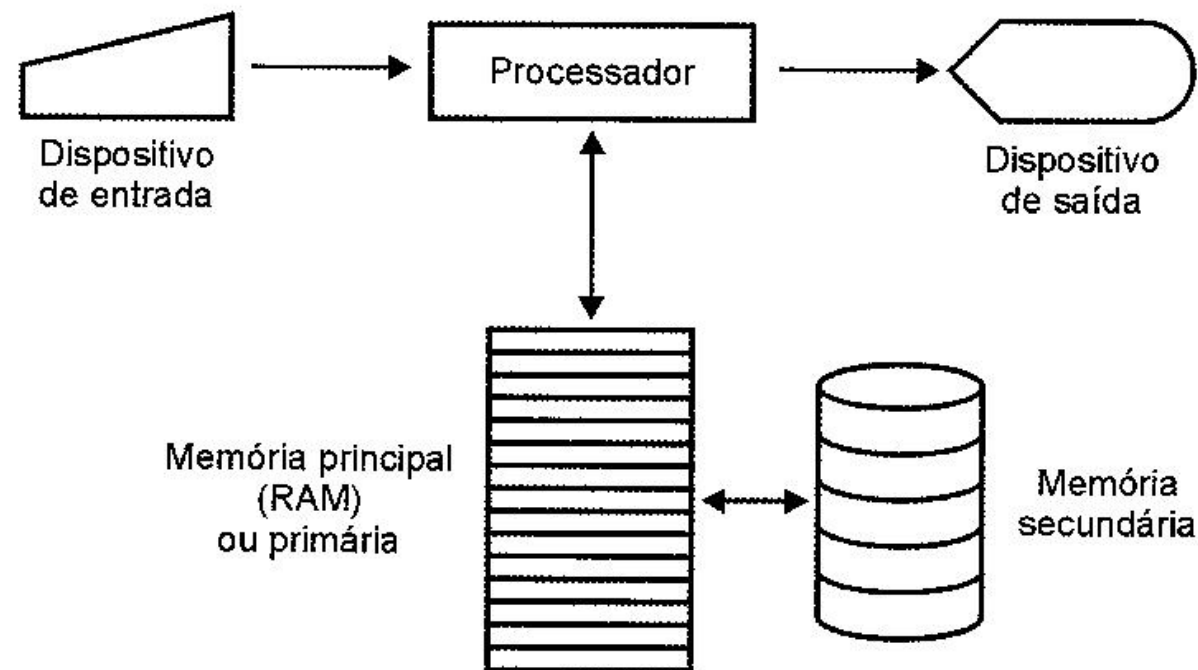


Figura 2.1 Componentes básicos de um computador.

Ciclo de Instrução Básico

✓ Processador executa instruções

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

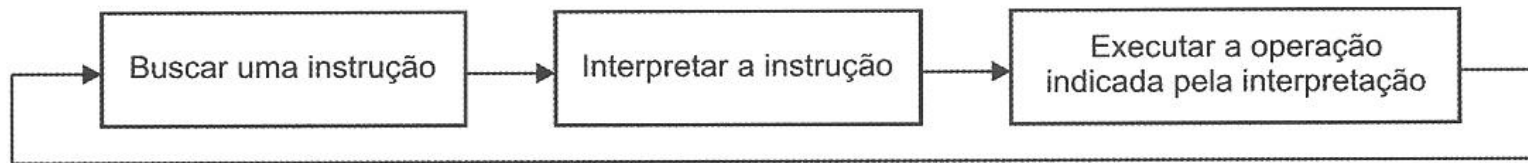
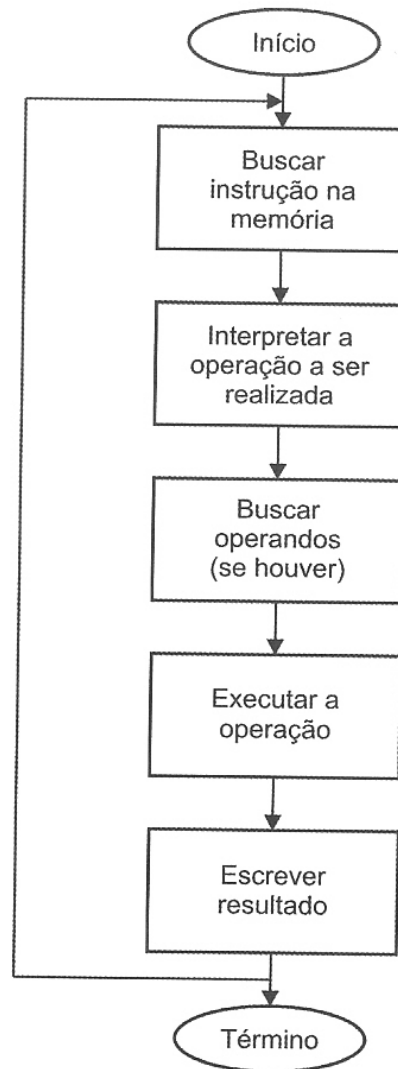


Figura 6.3 Funcionamento do hardware do processador – ciclo de instrução básico.

Ciclo de Instrução Básico

Fundamentos de Arquiteturas de Computadores



Endereço da instrução está no Registrador PC (program counter)

Conteúdo da instrução está no Registrador IR (instruction register)

Figura 6.2 Ciclo de instrução básico.

Funções do Processador

Fundamentos de Arquiteturas de Computadores

- ✓ **Funções de processamento**
- ✓ **Funções de controle**

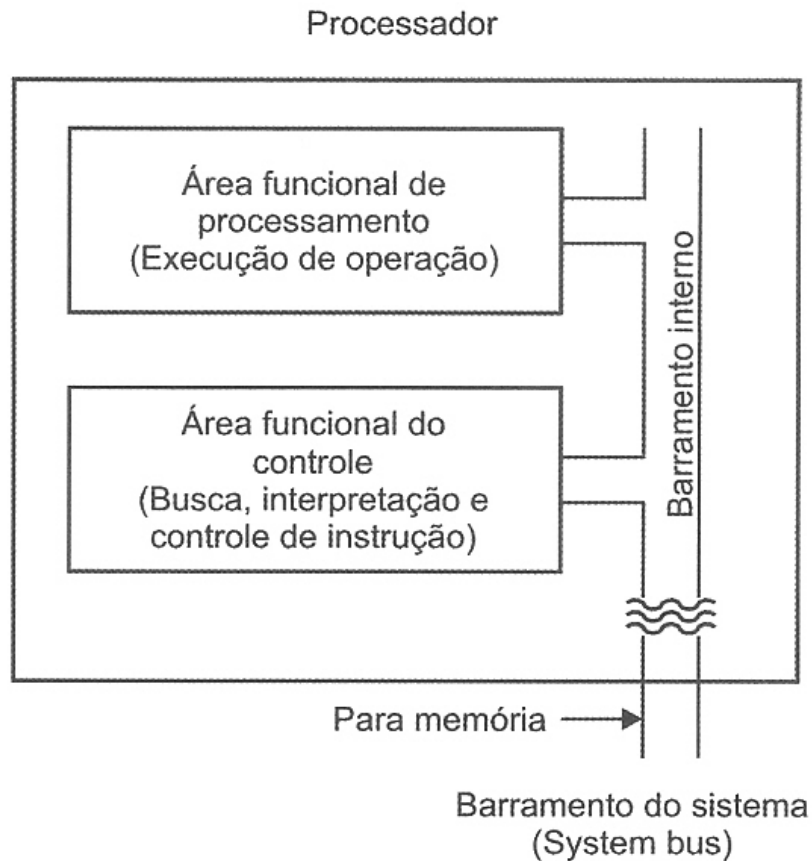


Figura 6.4 Diagrama em bloco (básico) de um processador.

Funções de Processamento

Fundamentos de Arquiteturas de Computadores

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

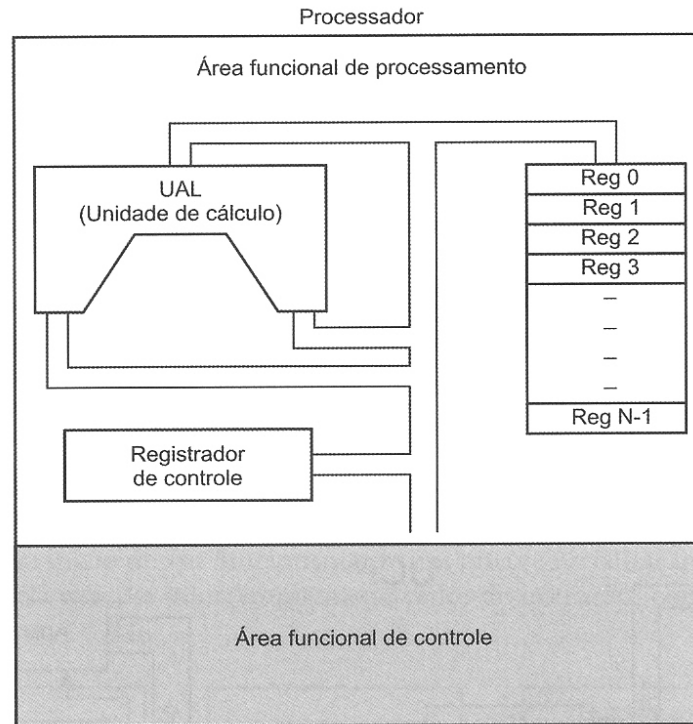
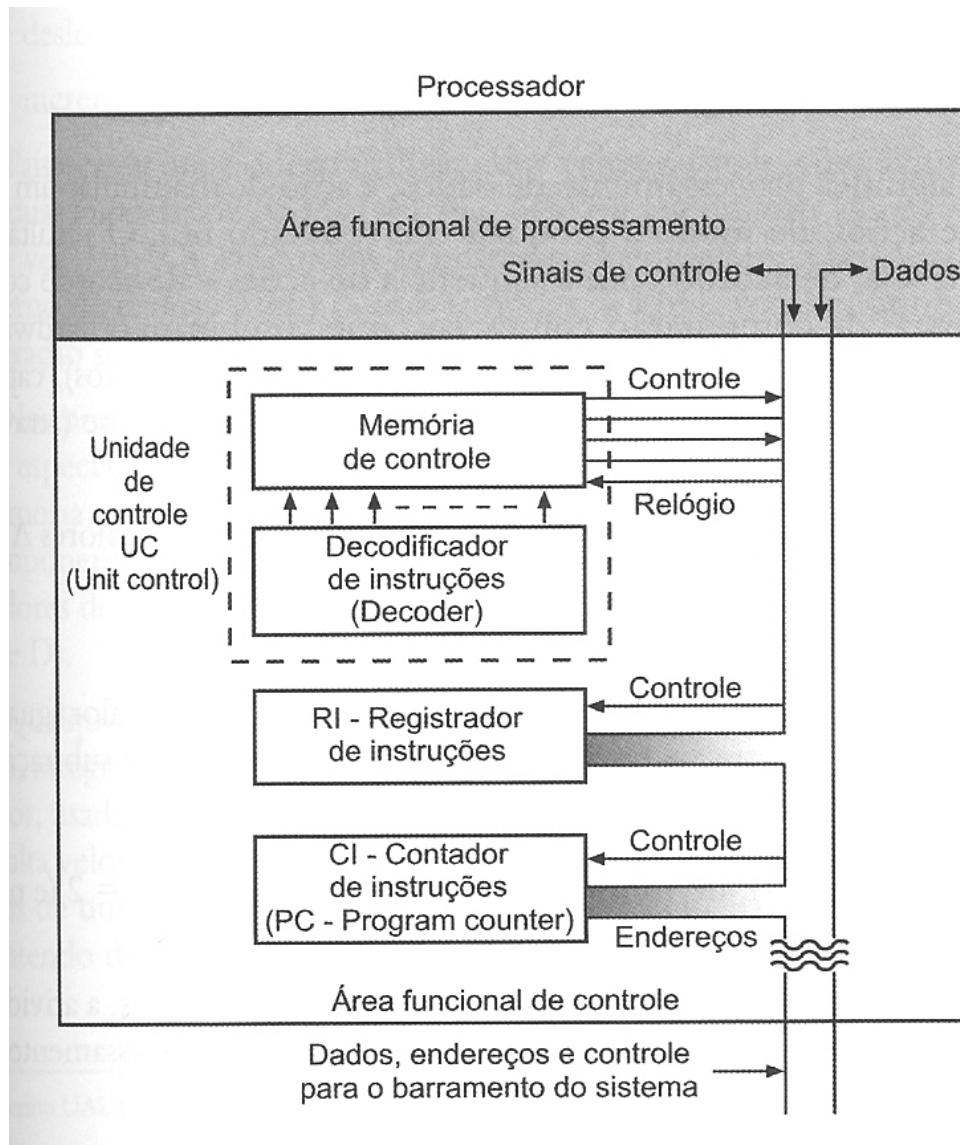


Figura 6.6 Componentes da área funcional de processamento.

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

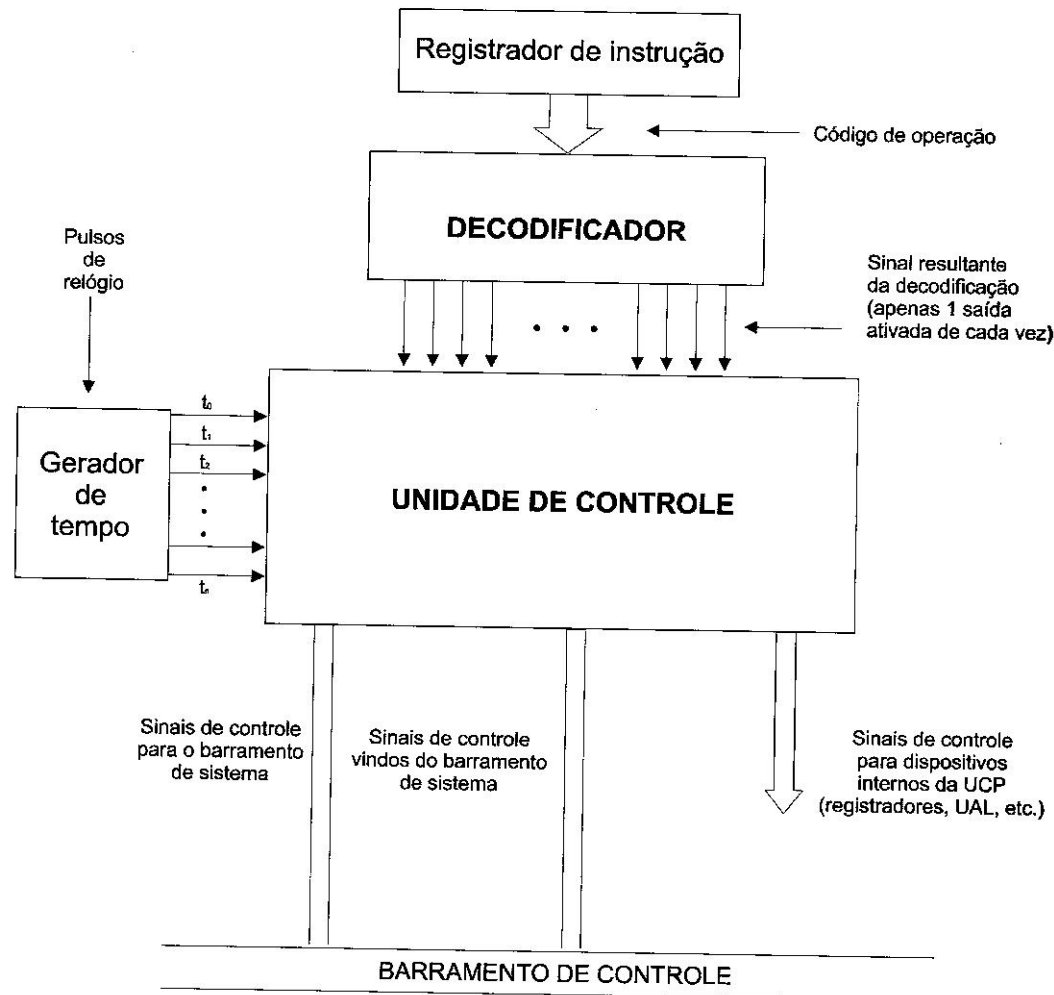
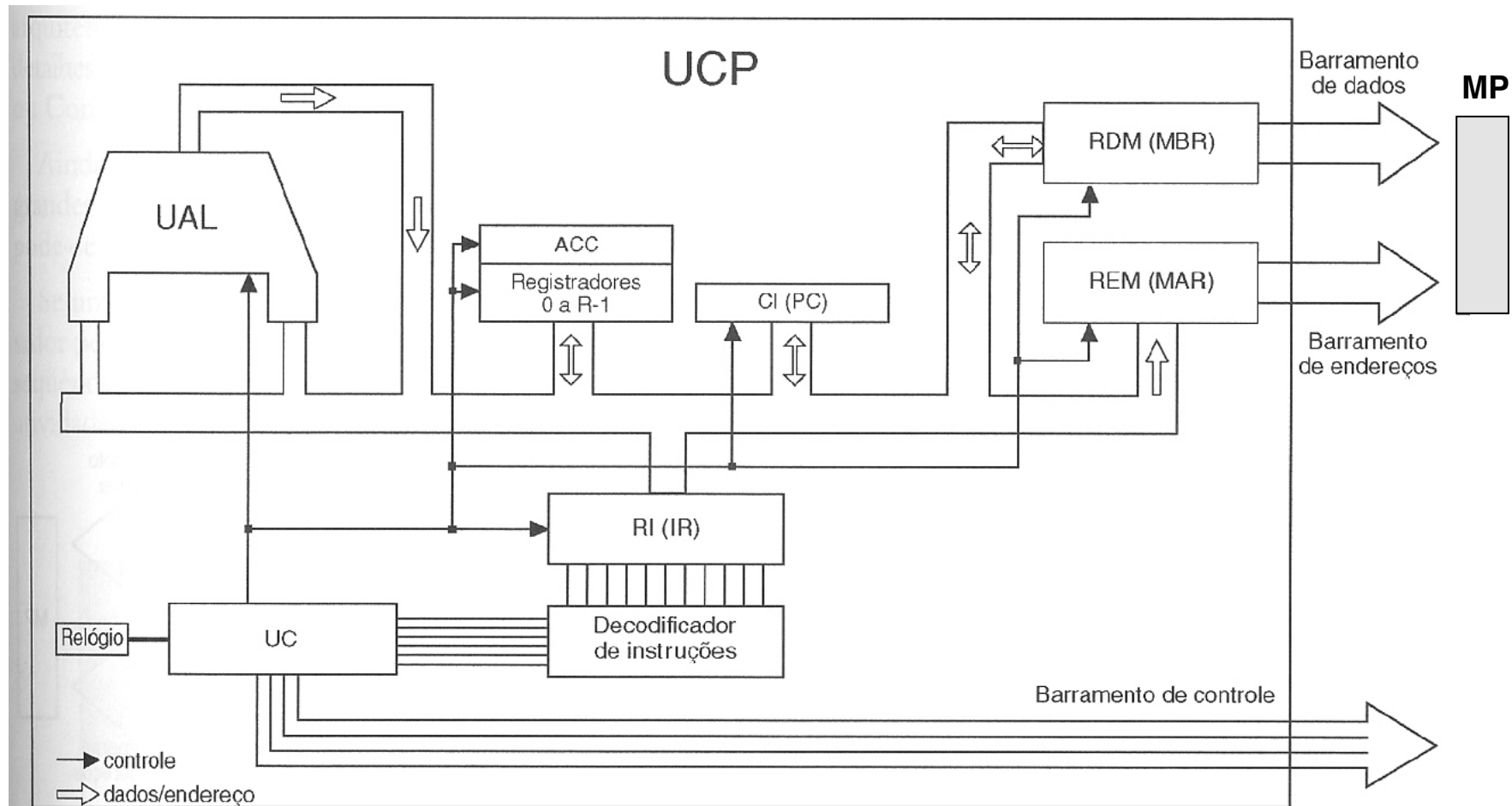


Figura 6.15 Diagrama em bloco simplificado da função controle.

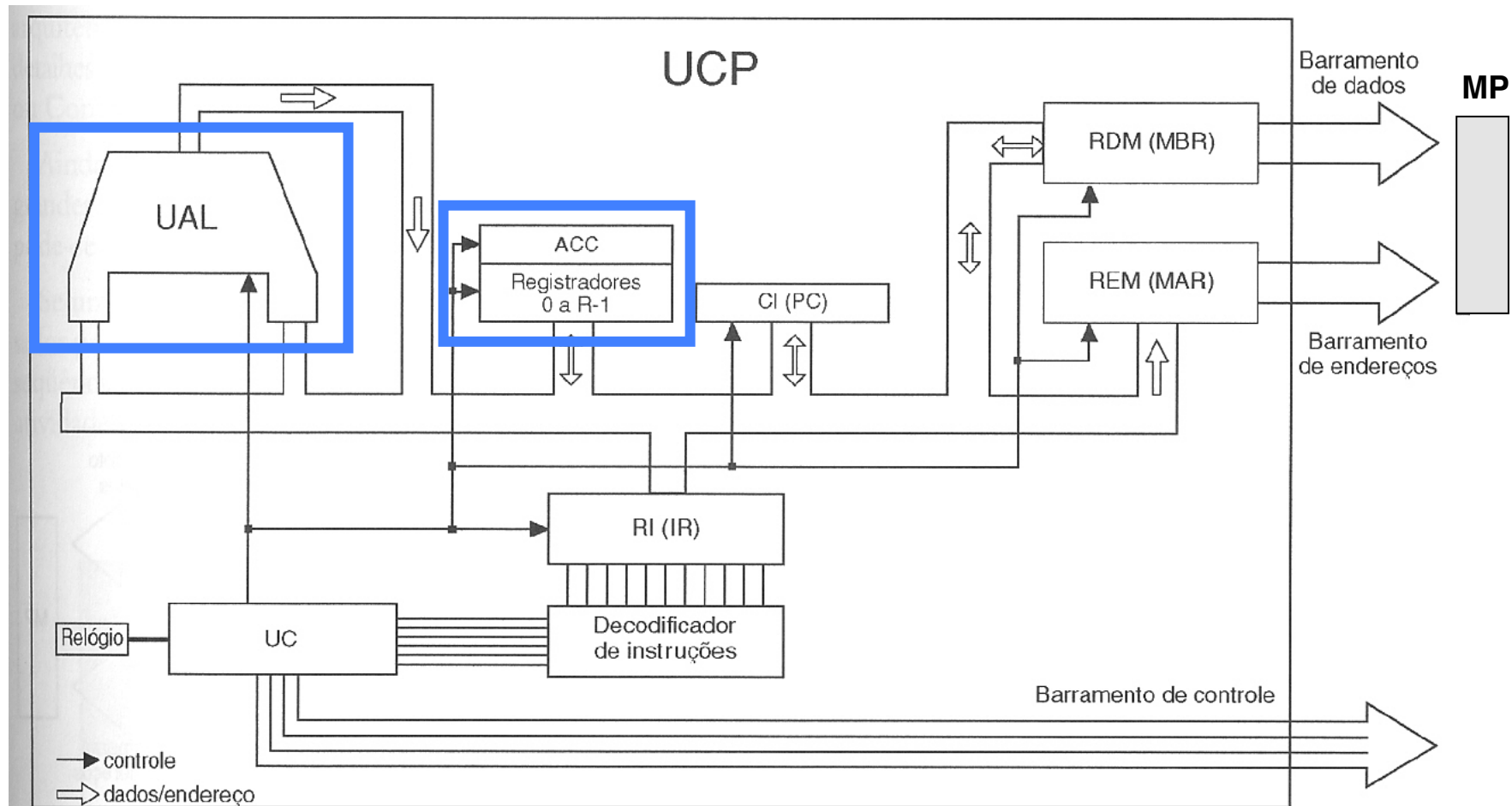
Esquema Simplificado da CPU

Fundamentos de Arquiteturas de Computadores



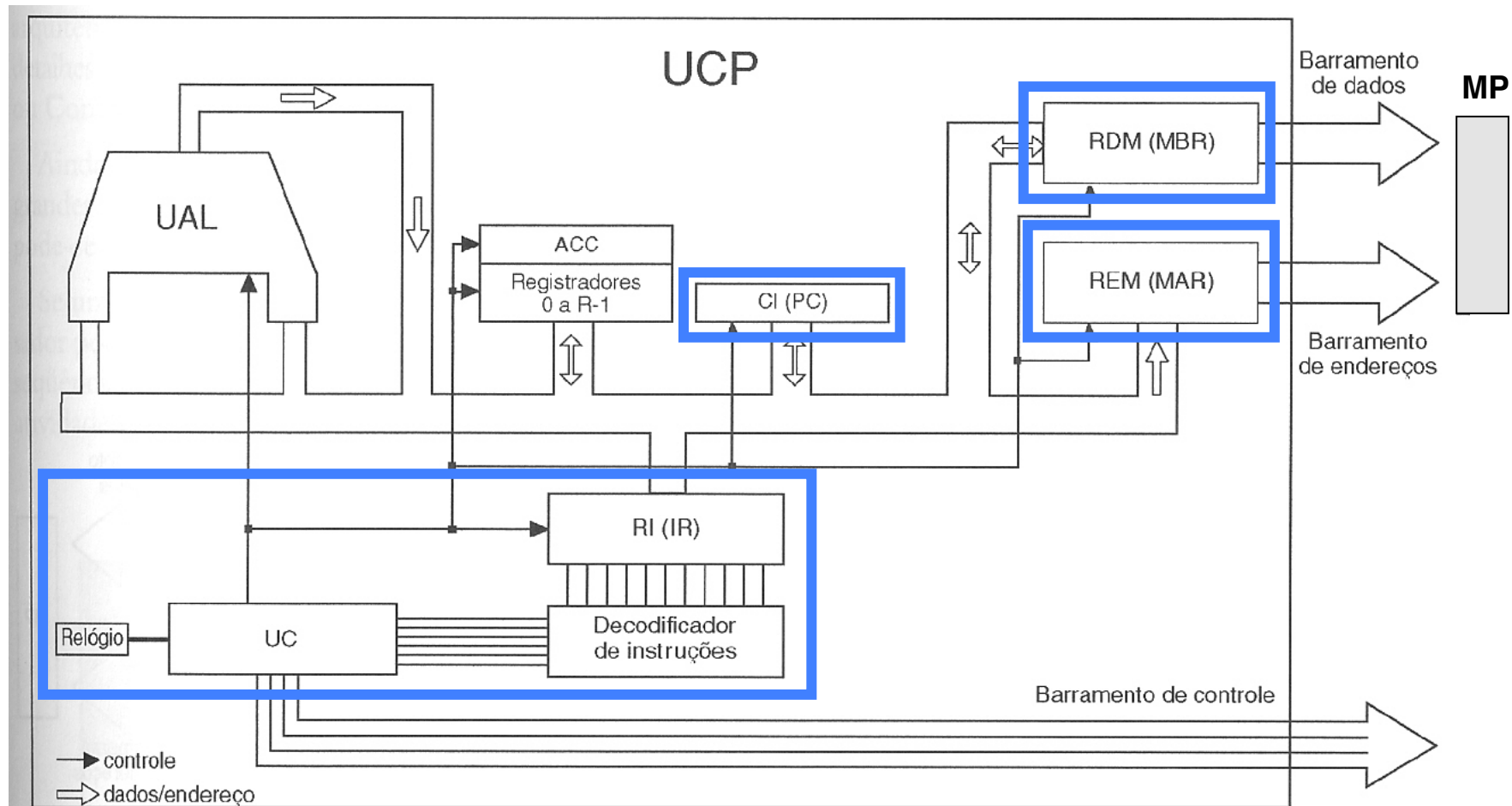
Componentes com Função de Processamento

Fundamentos de Arquiteturas de Computadores



Componentes com Função de Controle

Fundamentos de Arquiteturas de Computadores



Implementação da UCP

Fundamentos de Arquiteturas de Computadores

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

Conceitos Gerais para Implementação do Processador

Fundamentos de Arquiteturas de Computadores

- ✓ **Conceito de caminho de dados e controle**
- ✓ **Caminho dos bits de instrução e dados**
- ✓ **Utilização de clock, lógica combinacional e seqüencial**
- ✓ **Começa com uma implementação simples e iterativamente vai melhorando**

Performance

- ✓ Medida de performance baseada em:
 - *número de instruções*
 - *período do clock*
 - *ciclos de clock por instrução (CPI)*
- ✓ O primeiro é um fator do programa, mas os dois últimos são baseados na implementação do processador

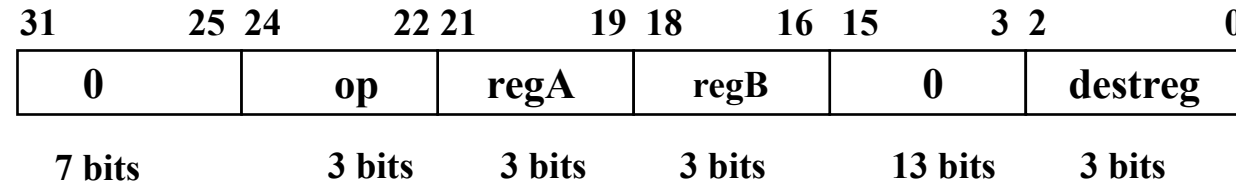
Subconjunto de instruções

- ✓ Para simplificar o estudo do projeto do processador, o foco será dado em um subconjunto de instruções do MIPS:
 - *Memória: lw e sw*
 - *Aritmética: add e addi*
 - *Desvio: beq*

Revisão do Formato das Instruções

✓ R-FORMAT (tipo R – registrador)

- *add regA regB destreg*



op: código de operação

regA: registrador com primeiro operando fonte

regB: registrador com segundo operando fonte

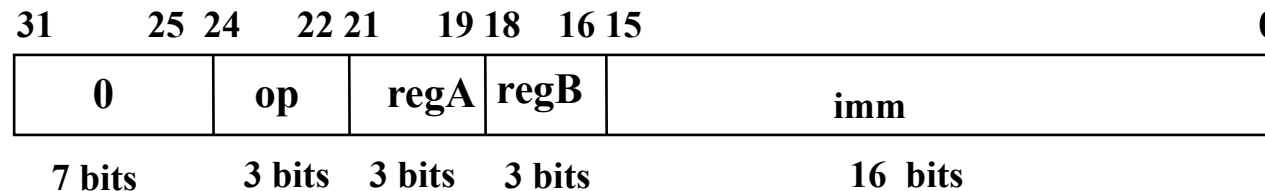
destreg: registrador que guarda resultado da operação

Revisão do Formato das Instruções

✓ I-Format (tipo I – imediato)

- *lw regA regB imm*
- *sw regA regB imm*
- *beq regA regB imm*
- *addi regA regB imm*

✓ Desvio utiliza endereço PC relativo ($PC + 1 + imm$)



Função básica da CPU

Fundamentos de Arquiteturas de Computadores

- ✓ **Buscar uma instrução na memória**
- ✓ **Interpretar qual operação é representada pela instrução**
- ✓ **Trazer (se for o caso) os operandos para a CPU**
- ✓ **Executar a operação**
- ✓ **Armazenar (se for o caso) os dados de saída**
- ✓ **Repetir o processo com uma nova instrução**

 *etapas do Ciclo de Instrução*

Projeto Lógico

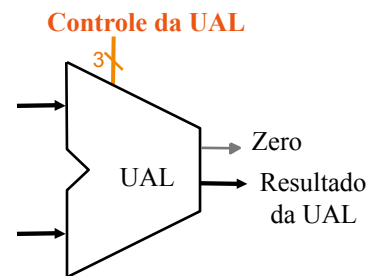
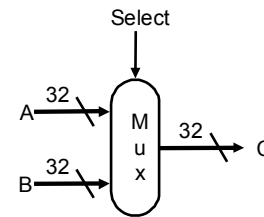
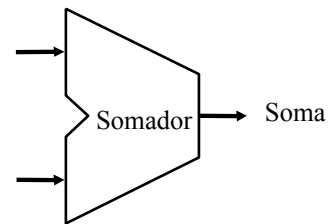
Fundamentos de Arquiteturas de Computadores

✓ Duas definições importantes

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

Elementos Combinacionais

Fundamentos de Arquiteturas de Computadores



Banco de Registradores

✓ 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

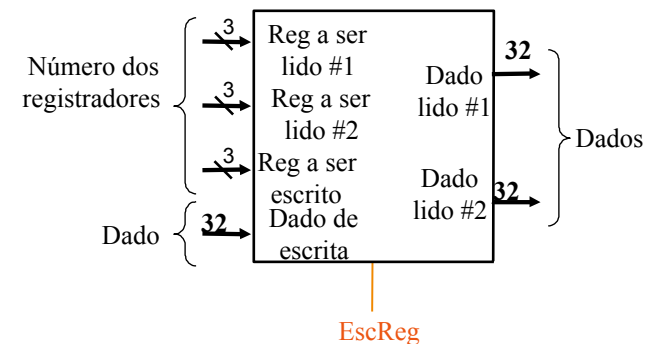
- *Registrador 0 tem o valor 0*

- *Registrador selecionado por*

- Reg a ser lido #1

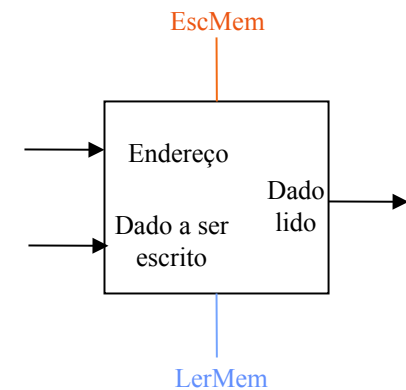
- Reg a ser lido #2

- Reg a ser escrito seleciona registrador que recebe Dado a ser escrito quando EscReg=1



Memória

- ✓ 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*

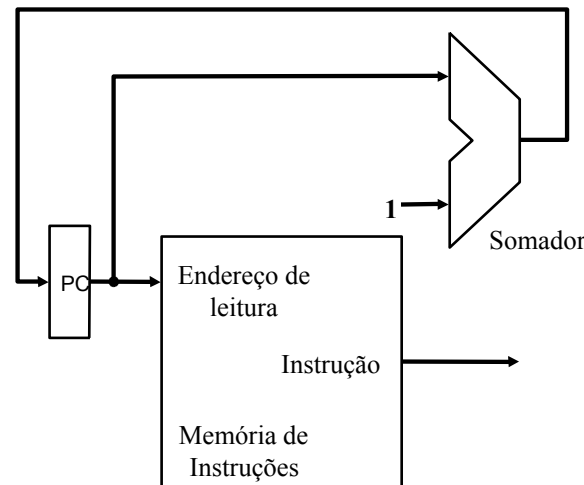


Passos do Projeto

- ✓ **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**

Busca da Instrução

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



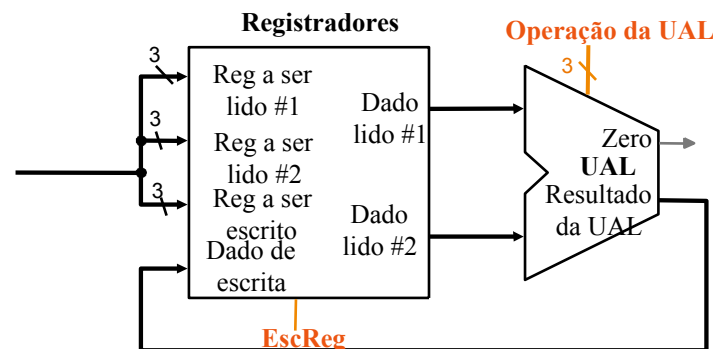
Instrução de Soma

✓ **add regA regB destreg**

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

Caminho de Dados para Instruções do tipo R

- ✓ $R[\text{destreg}] \leftarrow R[\text{regA}] \text{ op } R[\text{regB}]$
 - *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*



Instrução de Carga

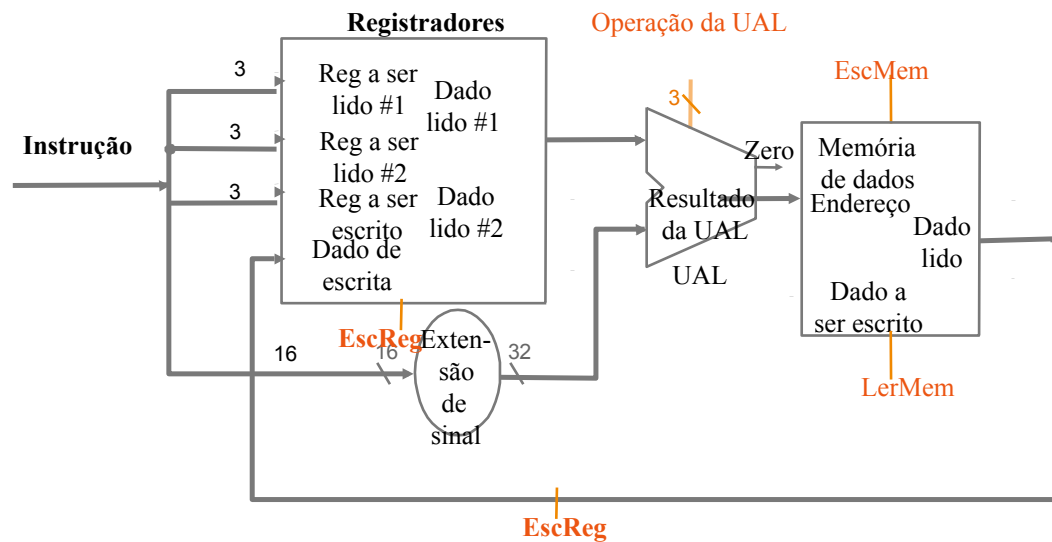
✓ lw regA regB imm

- *mem[PC]* *memória* *Busca instrução na*
- *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

Caminho de Dados para Instrução de Carga

Fundamentos de Arquiteturas de Computadores



Instrução de Armazenamento

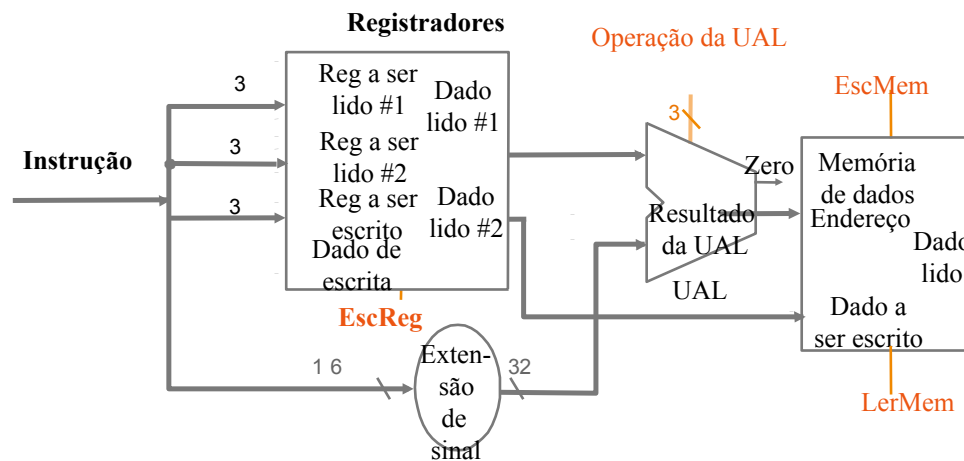
✓ **sw regA regB imm**

- *mem[PC]* *Busca instrução na memória*
- *End* $\leftarrow R[\text{regA}] + \text{SignExt}(\text{imm})$ *Calcula o endereço da memória*
- *Mem[End]* $\leftarrow R[\text{regB}]$ *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

Caminho de Dados para Instrução de Armazenamento

Fundamentos de Arquiteturas de Computadores



Instrução de Desvio Condicional

✓ beq regA regB imm

- *mem[PC]*
memória

Busca instrução na

- *Cond* $\leftarrow R[\text{regA}] - R[\text{regB}]$
desvio

Calcula a condição de

- *if (Cond eq 0)*

- $PC \leftarrow PC + 1 + \text{SignExt}(\text{imm})$

Calcula endereço PC relativo

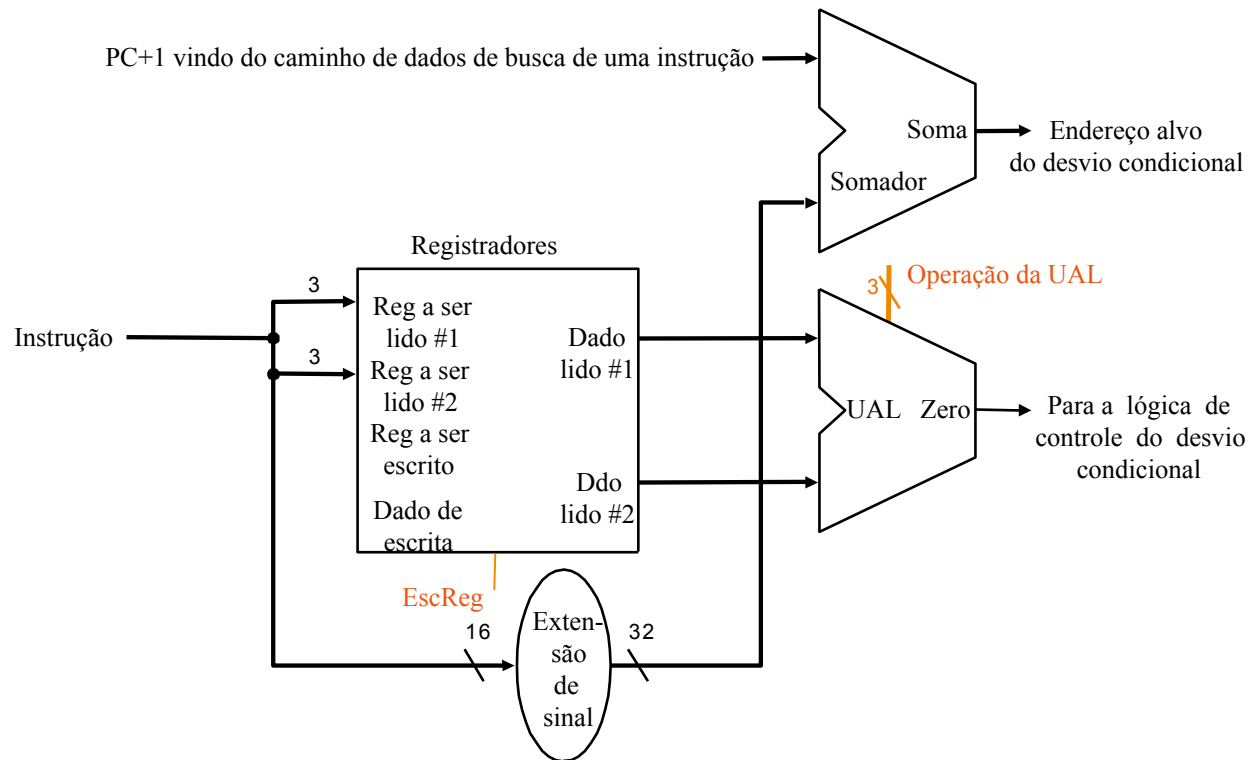
- *else*

- $PC \leftarrow PC + 1$

Calcula o próximo endereço

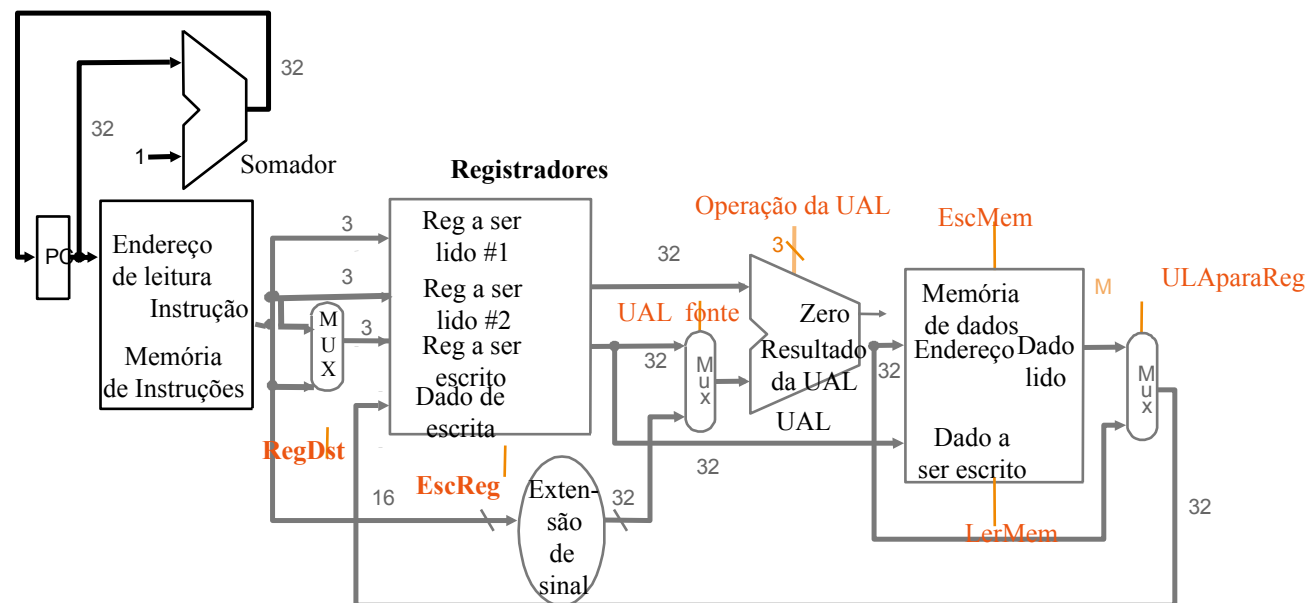
7 bits	3 bits	3 bits	3 bits	16 bits
0	op	regA	regB	imediato

Instrução de Desvio Condicional



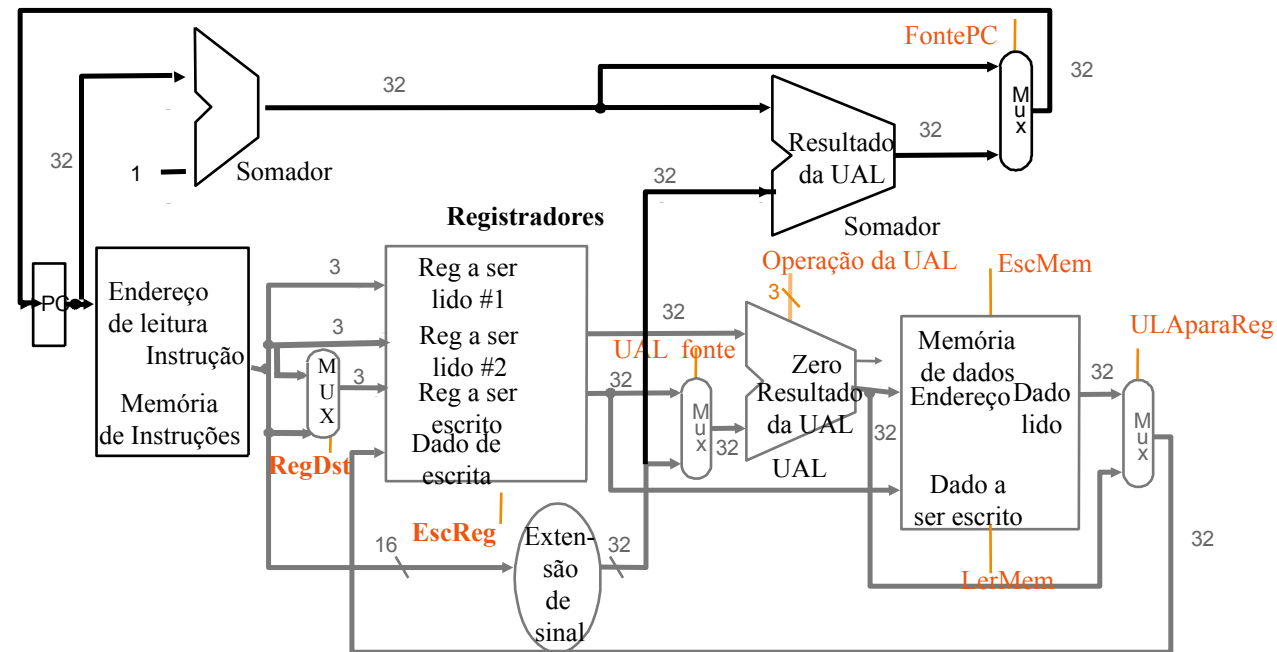
Busca de Instrução e Instruções Aritm. e Lóg. e de Referência à Memória

Fundamentos de Arquiteturas de Computadores



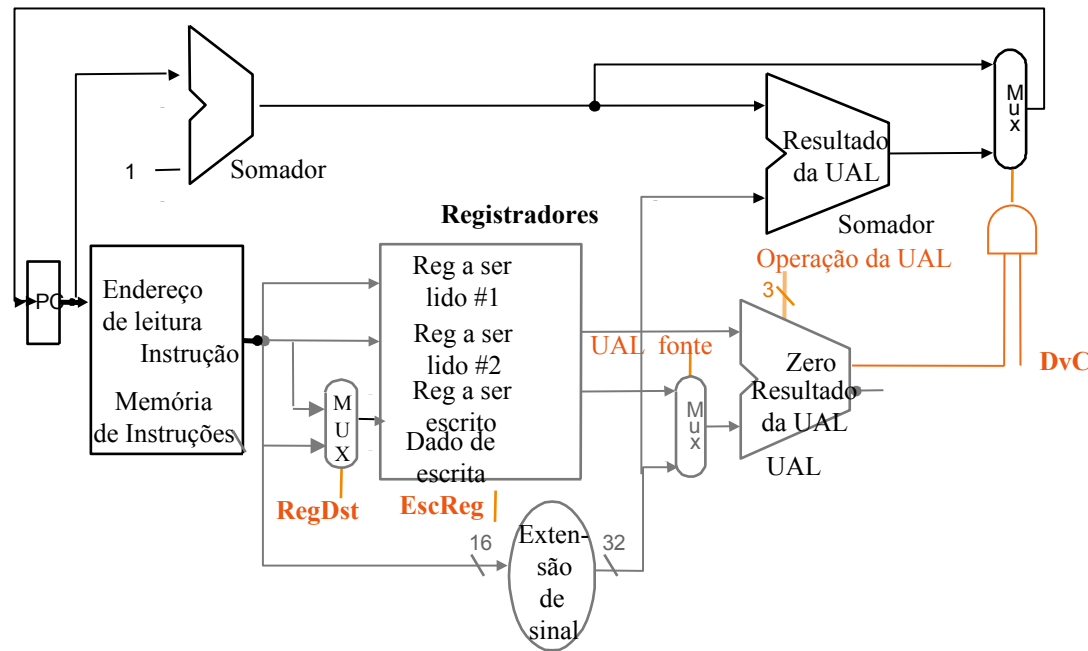
Caminho de Dados para Suportar Subconjunto das Instruções

Fundamentos de Arquiteturas de Computadores

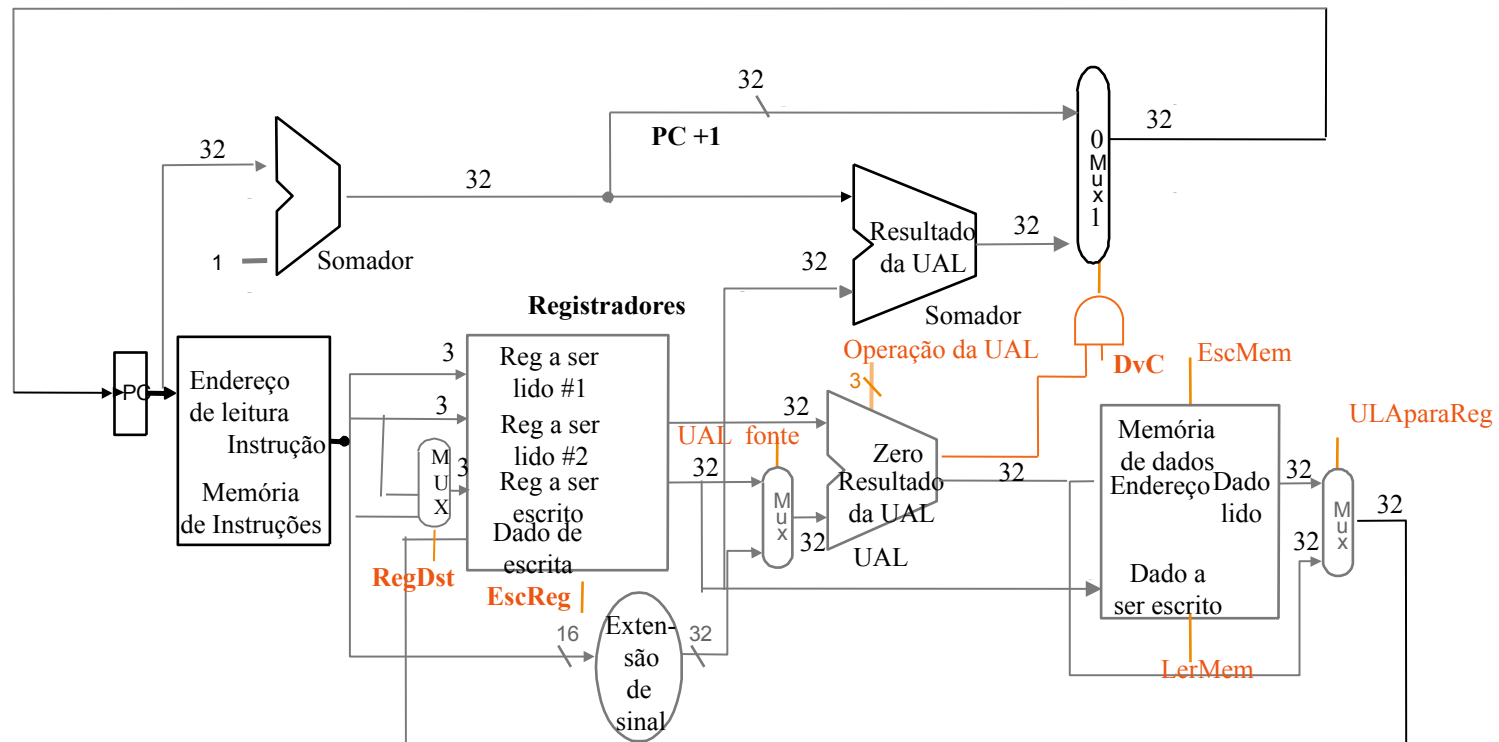


Caminho de Dados para Carregar PC

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



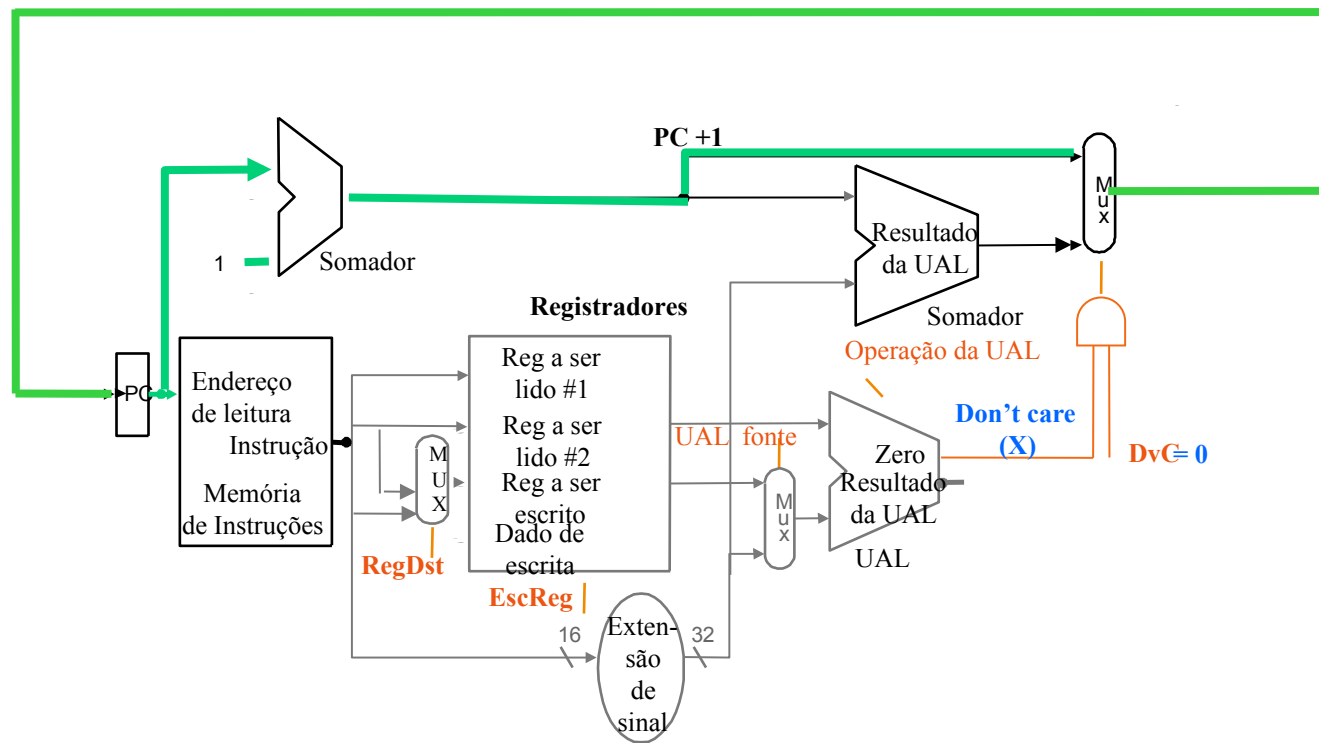
Juntando Todas as Partes



Inserindo o Controle

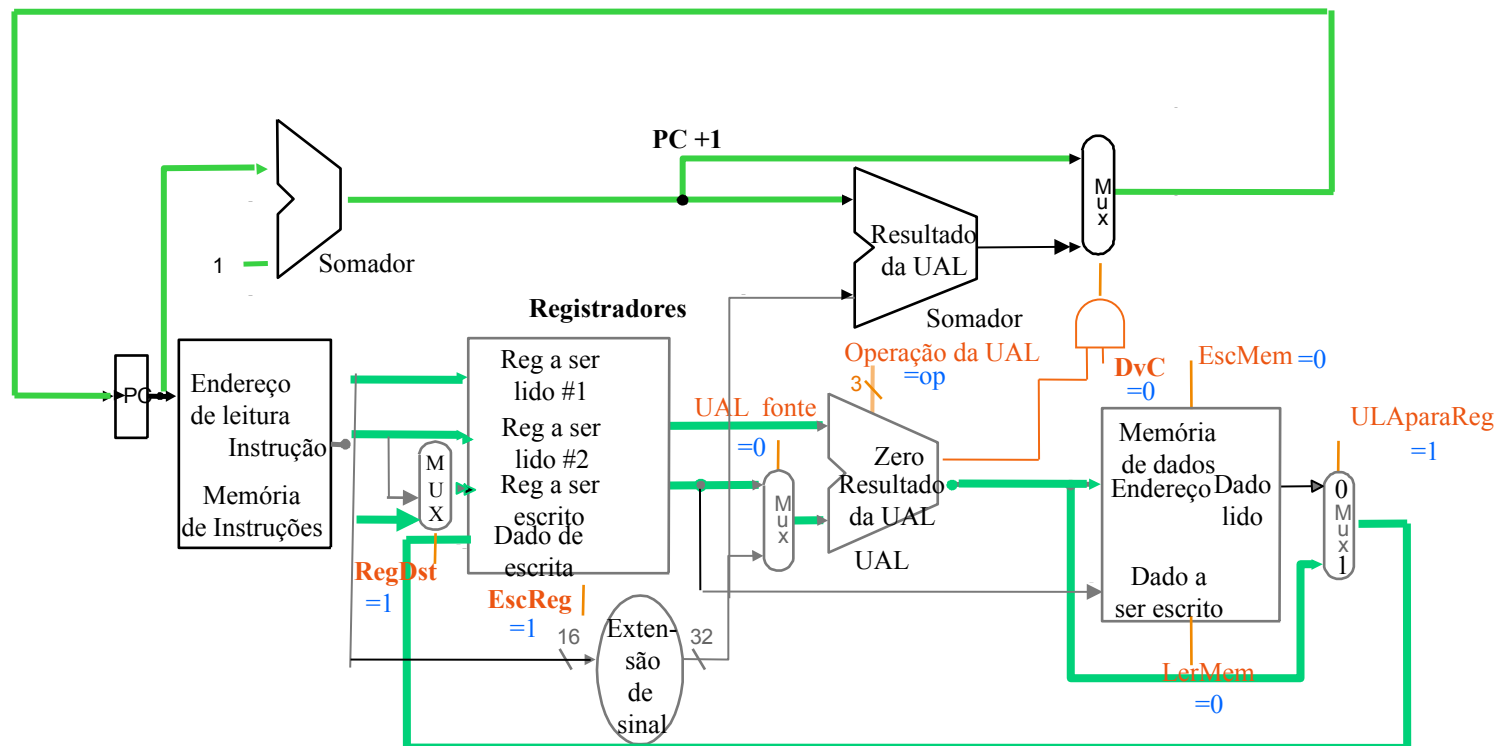
- ✓ **Identifica pontos de controle no caminho de dados**
 - *Busca da instrução*
 - *Operações lógicas e aritméticas*
 - *Memória*
- ✓ **Identifica tipo de controle do sinal**
 - *Fluxo de dados através de multiplexadores*
 - *Escrita de dados*
- ✓ **Deriva sinais de controle para cada instrução**
-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

Busca da Instrução



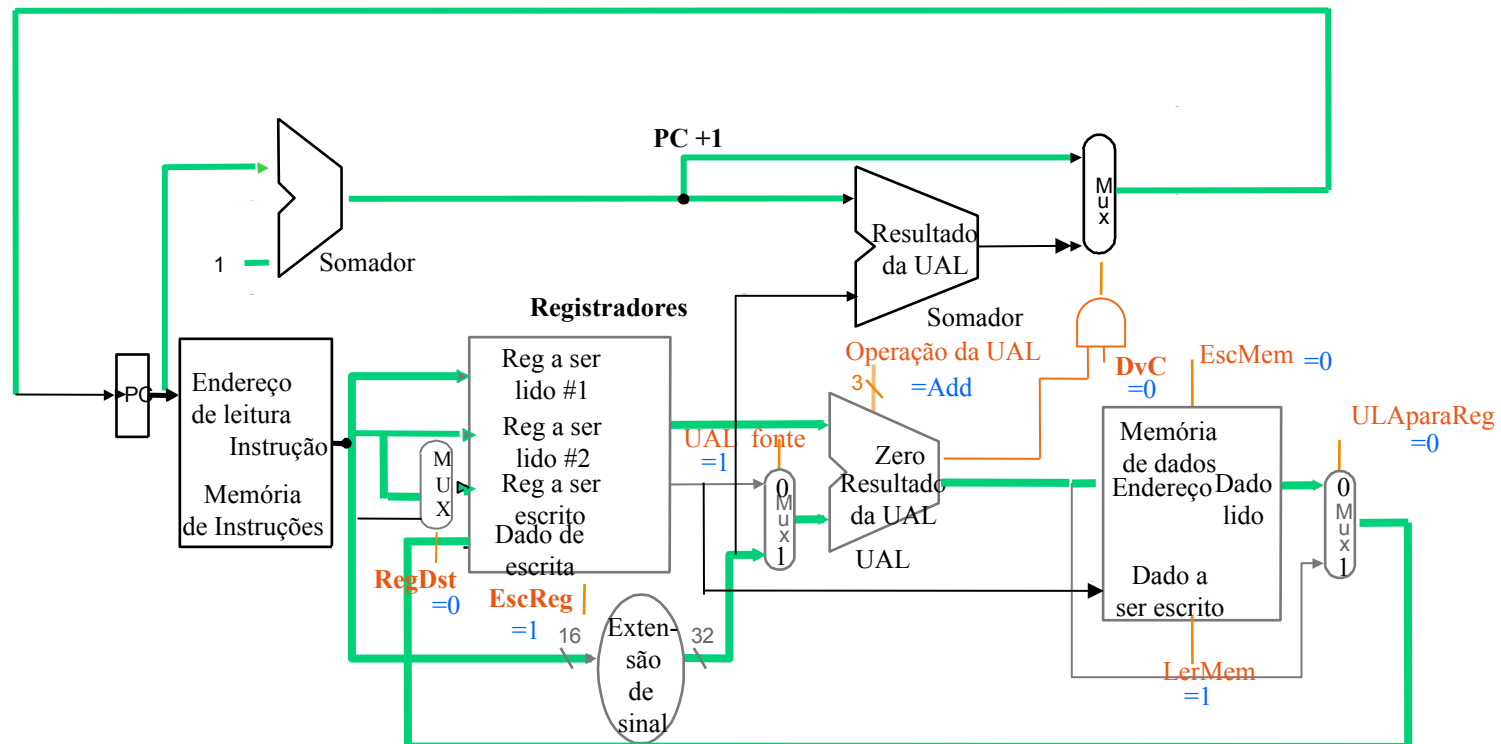
Controle para Operação Aritmética

Fundamentos de Arquiteturas de Computadores



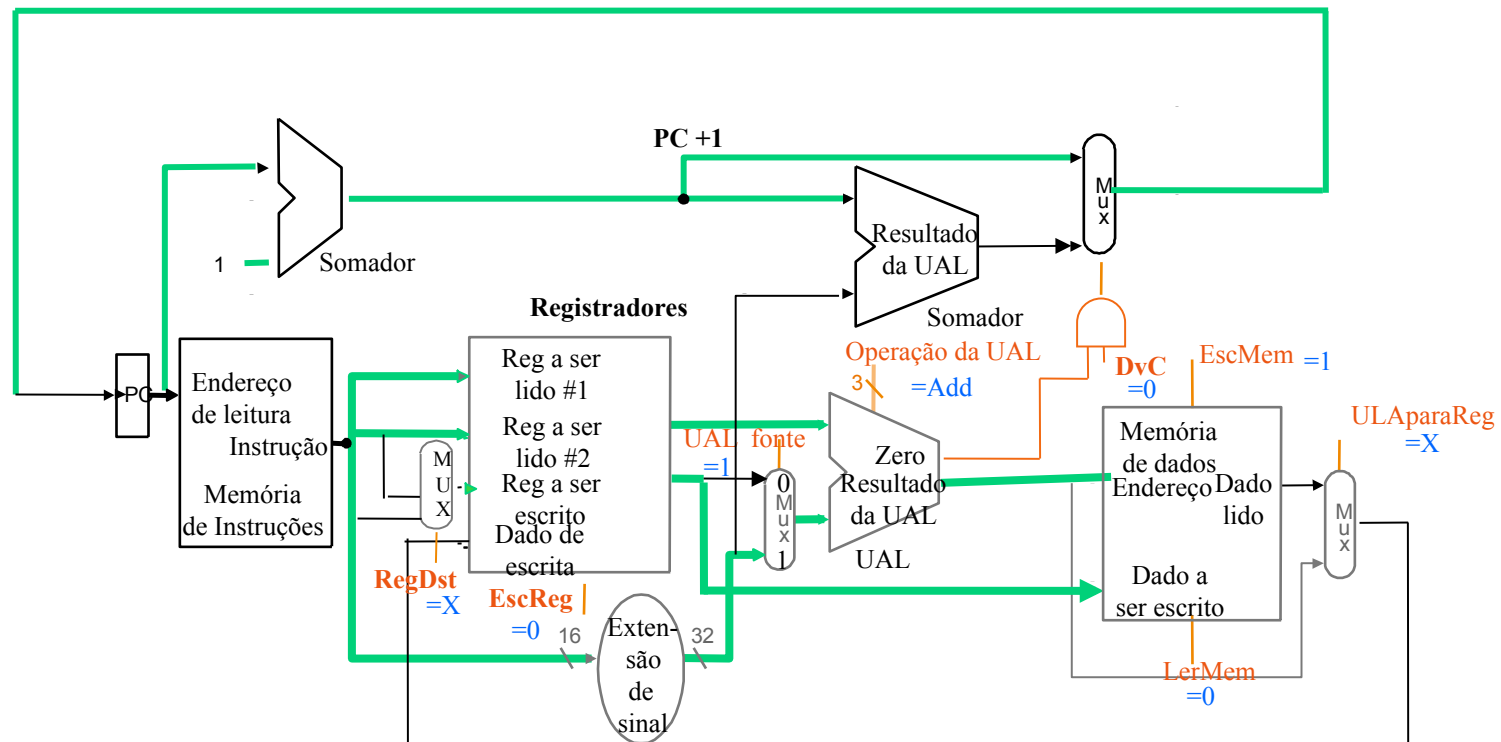
Controle para Operação de Carga

Fundamentos de Arquiteturas de Computadores



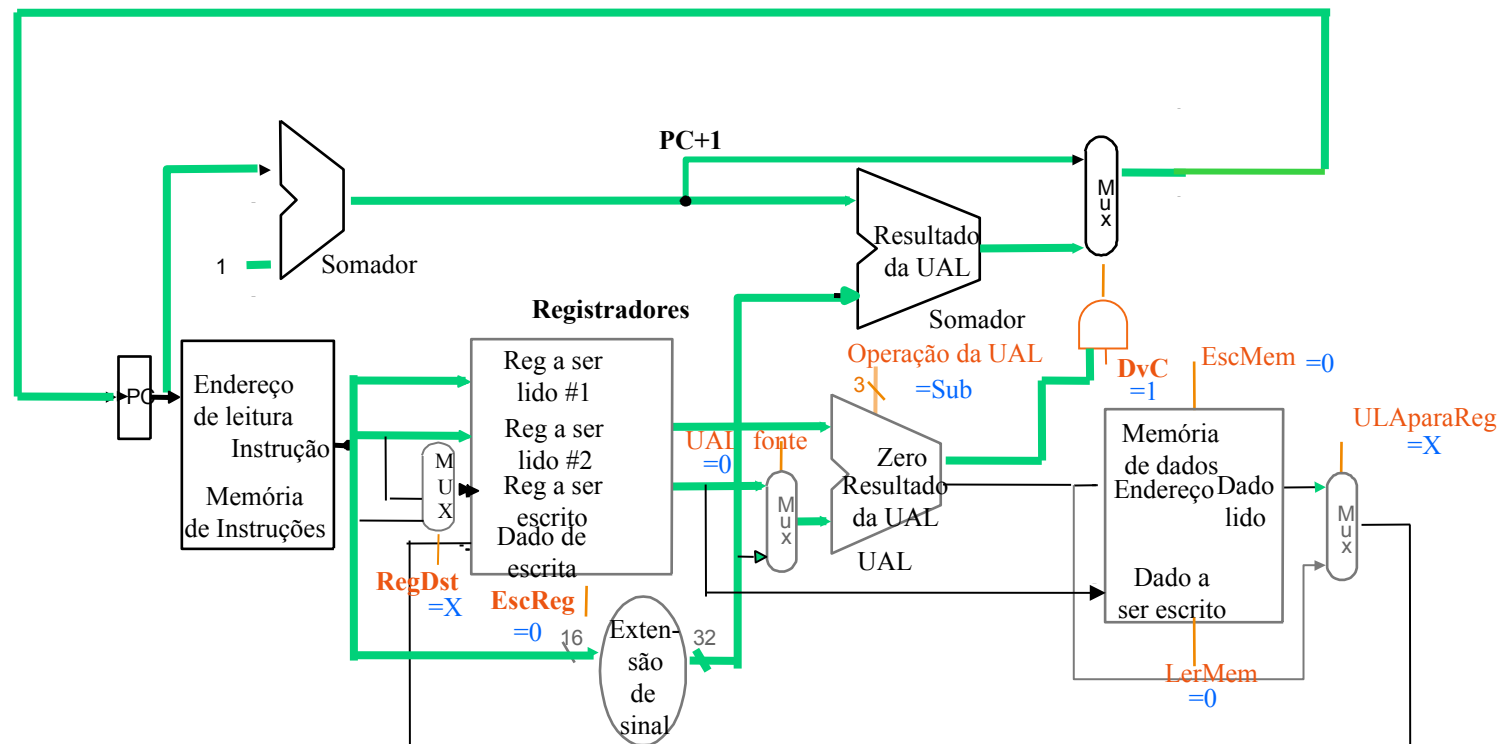
Controle para Operação de Armazenamento

Fundamentos de Arquiteturas de Computadores



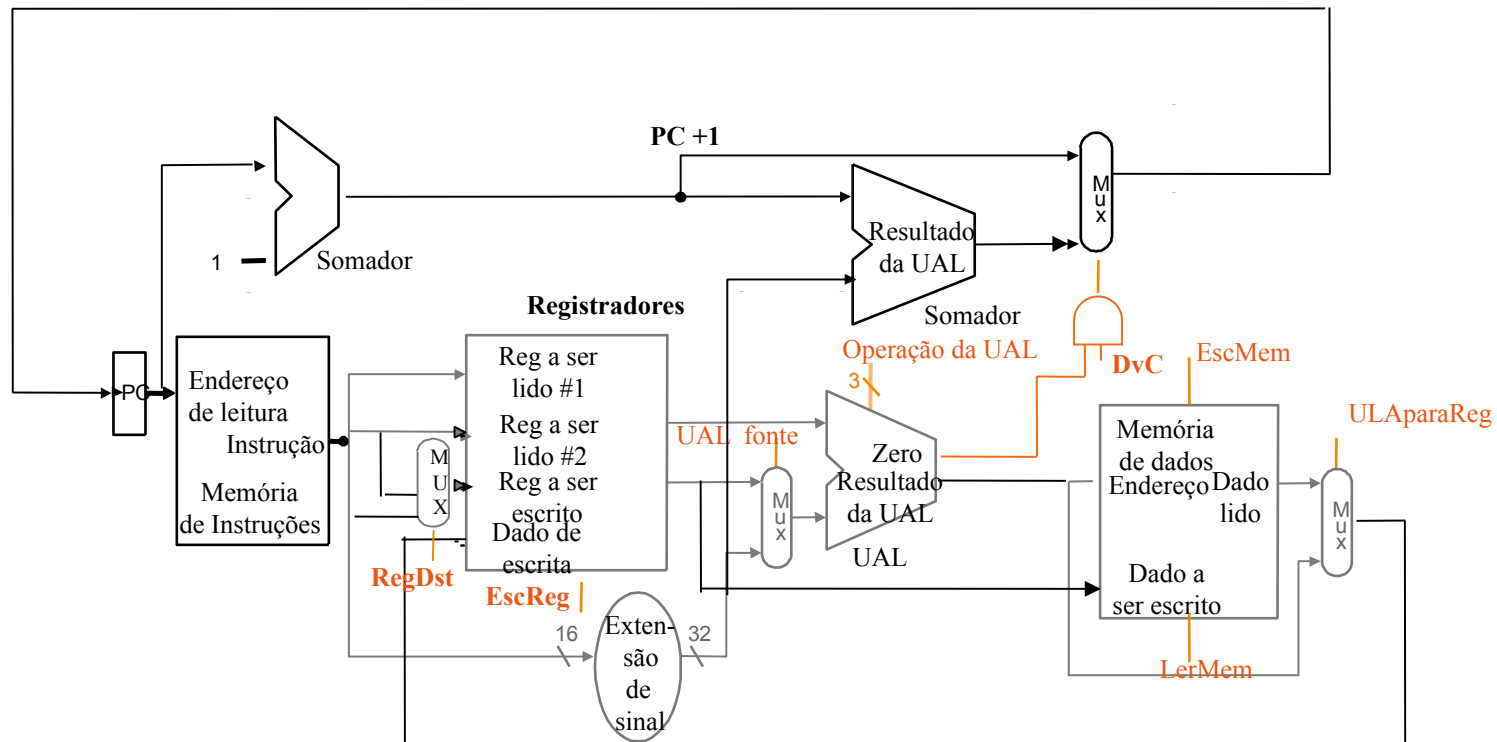
Controle para Operação de Desvio Condicional (beq)

Fundamentos de Arquiteturas de Computadores



Sinais de Controle

Fundamentos de Arquiteturas de Computadores



Resumo dos Sinais de Controle

Fundamentos de Arquiteturas de Computadores

op	000	010	011	100
	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

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)

Função de Controle para Implementação Monociclo

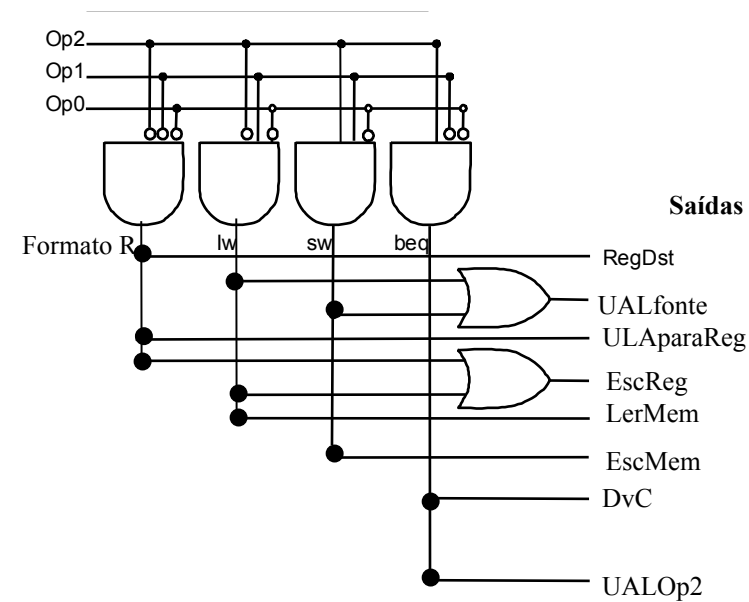
Fundamentos de Arquiteturas de Computadores

op	000	010	011	100
	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

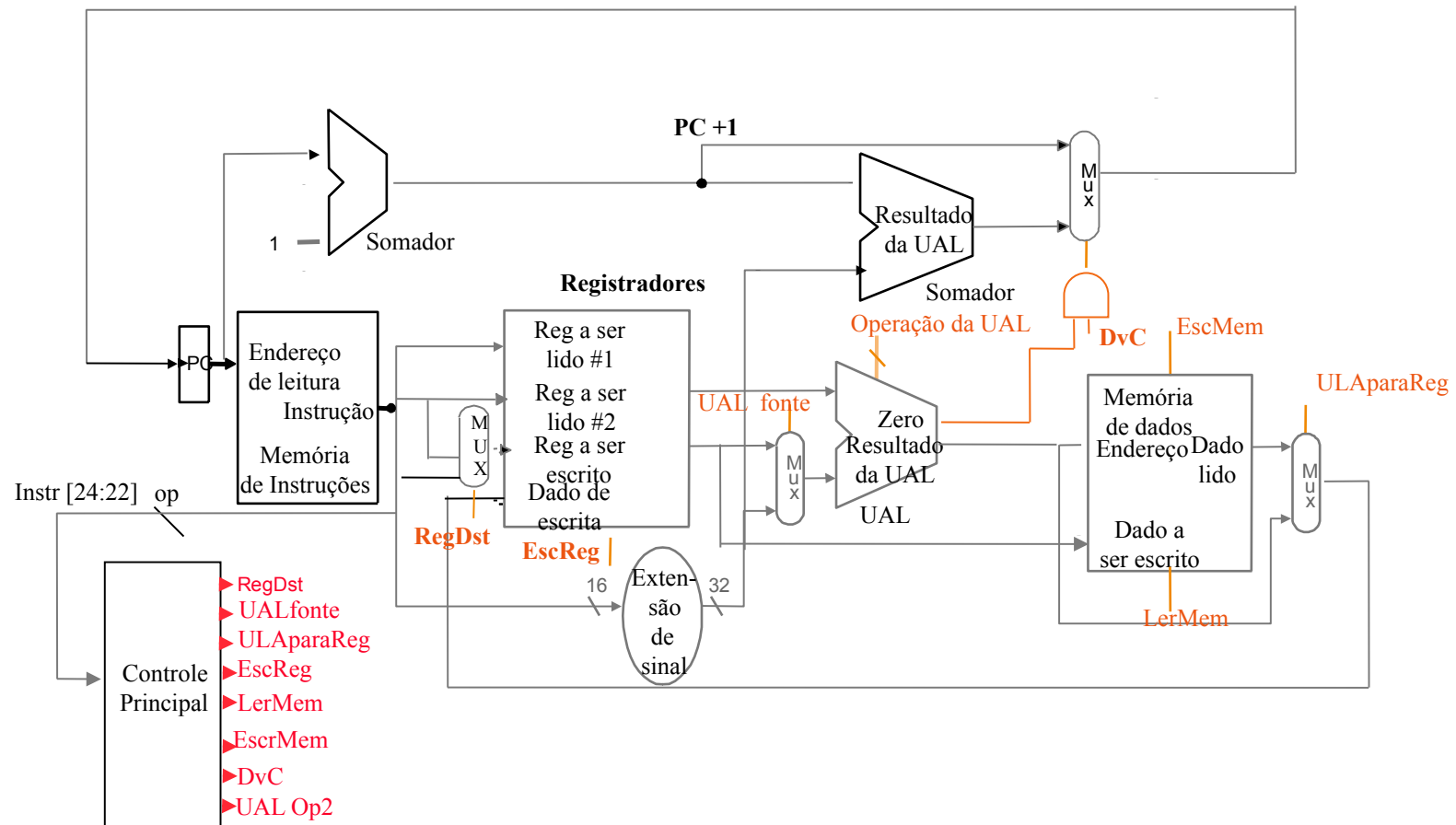
Implementação da Função de Controle (PLA)

Fundamentos de Arquiteturas de Computadores

Entradas



Juntando as Partes



Implementação Monociclo

✓ 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*

Execução de Programas

Profa. Débora Christina Muchaluat Saade

debora@midiacom.uff.br

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

Execução de Programas

Fundamentos de Arquiteturas de Computadores

- ✓ 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

Fundamentos de Arquiteturas de Computadores

Programa em
linguagem de
alto nível
(em C)

```
swap(int v[], int k)
{
    int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

↓
Compilador

Programa em
assembly
(para MIPS)

```
swap:
    muli $2, $5, 4
    add  $2, $4, $2
    lw   $15, 0($2)
    lw   $16, 4($2)
    sw   $16, 0($2)
    sw   $15, 4($2)
    jr   $31
```

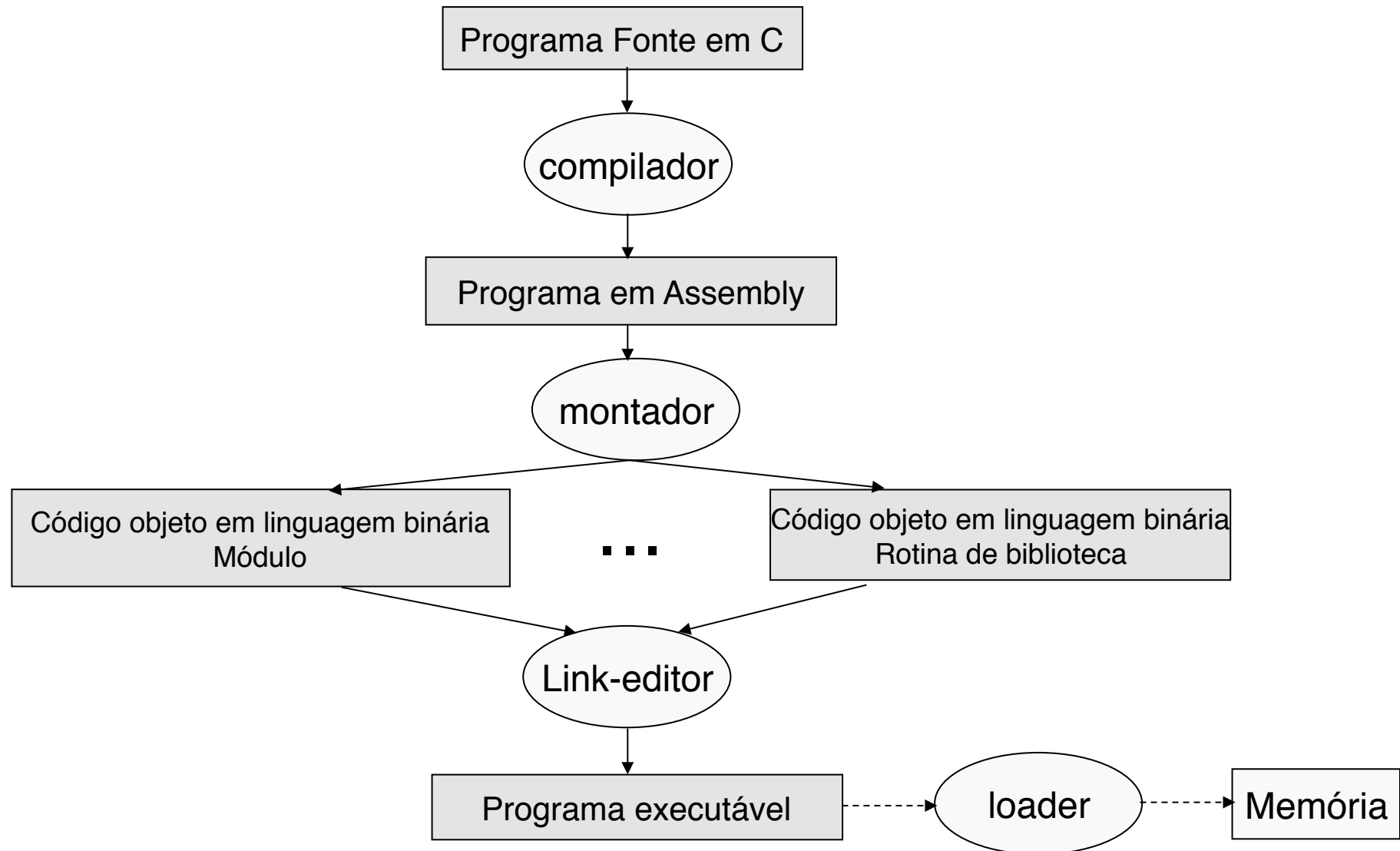
↓
Montador

Programa
binário em
linguagem de
máquina
(para MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
0000001111100000000000000001000
```

Geração do executável

Fundamentos de Arquiteturas de Computadores



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

- ✓ 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.*

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.**

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**