UCP: Construindo um Caminho de Dados (Parte III)

Cristina Boeres

Instituto de Computação (UFF)

Fundamentos de Arquiteturas de Computadores

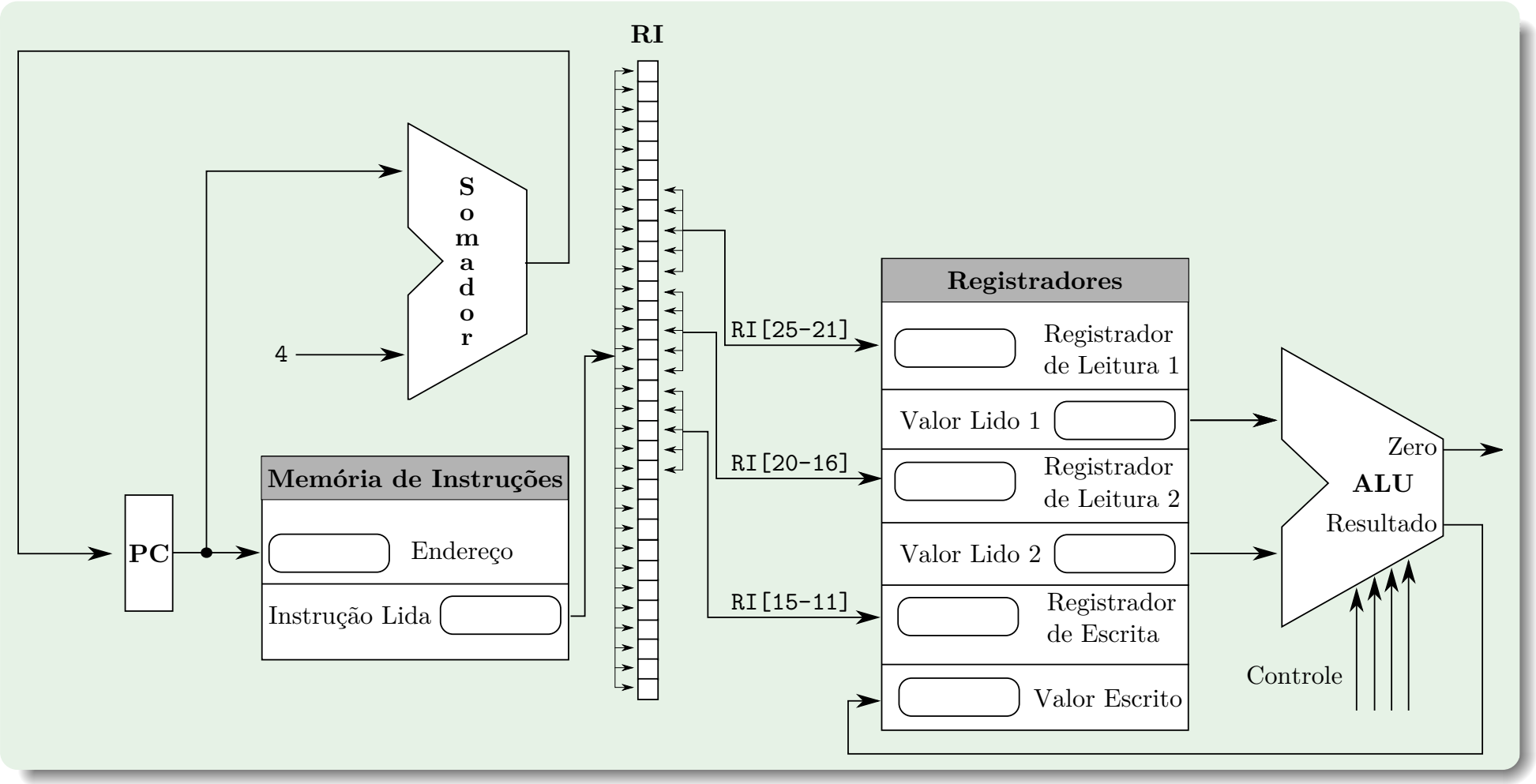
Material baseado nos slides do Fernanda Passos

Continuando a construir a arquitetura básica

Interconectando componentes de forma incremental

- Seguindo a estratégia de desenvolvimento incremental, começamos do básico
  - Como implementar a etapa de busca de uma instrução
  - Fase comum a qualquer tipo de instrução
- Demais etapas são específicas de cada tipo de instrução
  - Como executar uma instrução do tipo R.

Busca de Instrução e Execução de Instrução do Tipo R



Continuando a construir a arquitetura

- Este **esquema** é quase completo para a execução de instruções do tipo R
- Mas falta uma coisa:
  - Determinar as linhas de controle da ALU – ainda falta
- Mas note que operações lógicas/aritméticas também são necessárias para outros tipos de instrução
  - Diferentes do tipo R, como as de desvio condicional
- Por isso, antes disso, veremos como implementar outras instruções

Instruções de Desvio Condicional

A instrução *branch on equal*

- Nosso processador simplificado possui apenas uma instrução deste tipo
- Semântica:
  - Compara dois valores (em registradores)
  - Se forem iguais, altera o valor do PC para o valor especificado na instrução
  - Se forem diferentes, PC continua apontando para a próxima instrução
    - LEMBRE: PC já está sendo incrementado no mesmo ciclo de relógio

Instruções de Desvio Condicional

Formato (relembrando)

000100 *sssss ttttt iiiiiiiiiiiiiiii*

- *sssss*: indica o registrador do primeiro operando
- *ttttt*: indica o registrador do segundo operando
- *iiiiiiiiiiiiiiii*: 16 bits (Complemento a Dois)

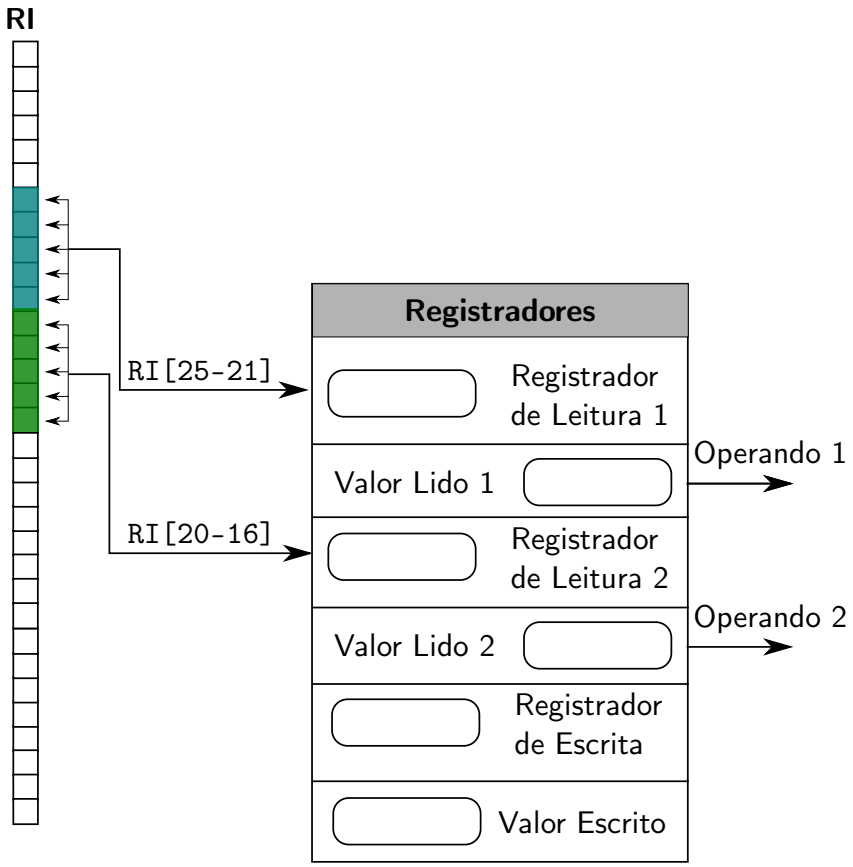
Alguns detalhes importantes:

- A instrução possui formato I.
  - Recebe três operandos: 2 registradores, 1 imediato (constante).
- Se desvio for realizado, PC é incrementado no valor dado pelo imediato
  - **Depois do incremento inicial**
  - **E em unidades do tamanho da instrução**

$$PC \leftarrow PC + (imm + 1) \times 4$$

Branch on Equal: Busca de Operandos

- A instrução compara dois valores
  - nos registradores especificados na instrução
- Logo, o banco de registradores será utilizado
- Note que a instrução não escreve em registrador do banco



Branch on Equal: Execução

como comparar dois conteúdos de registrador?

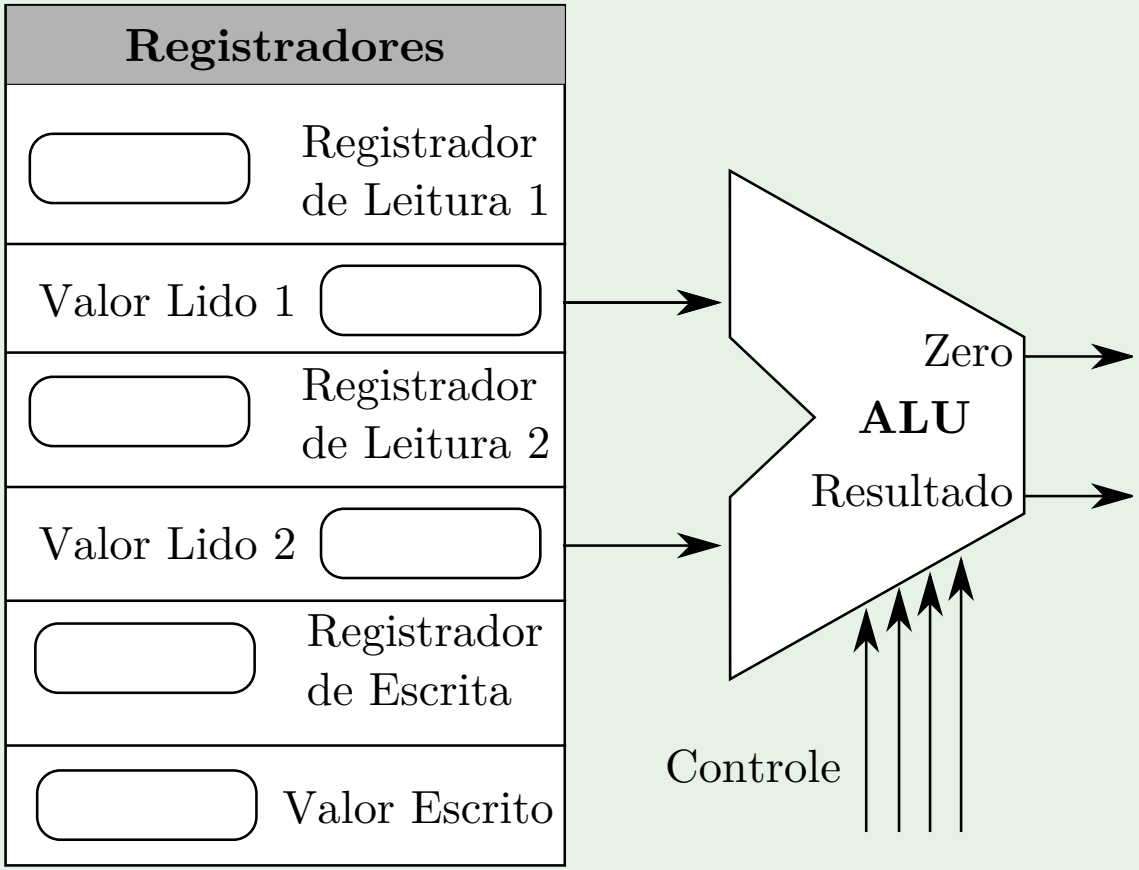
- Note que nenhum componente visto até aqui realiza comparações
  - Nem mesmo a ALU tem essa funcionalidade
- Mas podemos alcançar este mesmo efeito usando um truque
  - Repare que nossa ALU tem duas saídas: resultado e o *bit zero*
  - Bit zero indica se operação realizada deu zero
  - Uma **subtração** dá zero **se, e somente se,** os operandos forem iguais

Solução para comparar dois registradores

- Usamos a ALU para subtrair os dois valores
- Se o bit zero ficar ativo, valores são iguais

Branch on Equal: Execução

A interconexão entre o banco de registradores e a ALU é idêntica àquela das instruções R



Branch on Equal: Execução

Realizada a comparação: resolver o valor do PC

- É preciso calcular o novo valor do PC
  - Caso a condição seja verdadeira.
- Novo valor do PC:
  - $PC + (imm + 1) \times 4$.
  - Ou: $PC + 4 + imm \times 4$.
- LEMBRE: na busca da próxima instrução , $PC + 4$ já era calculado
  - Aproveitando aquele resultado, é preciso apenas somá-lo com $imm \times 4$

Branch on Equal: Execução

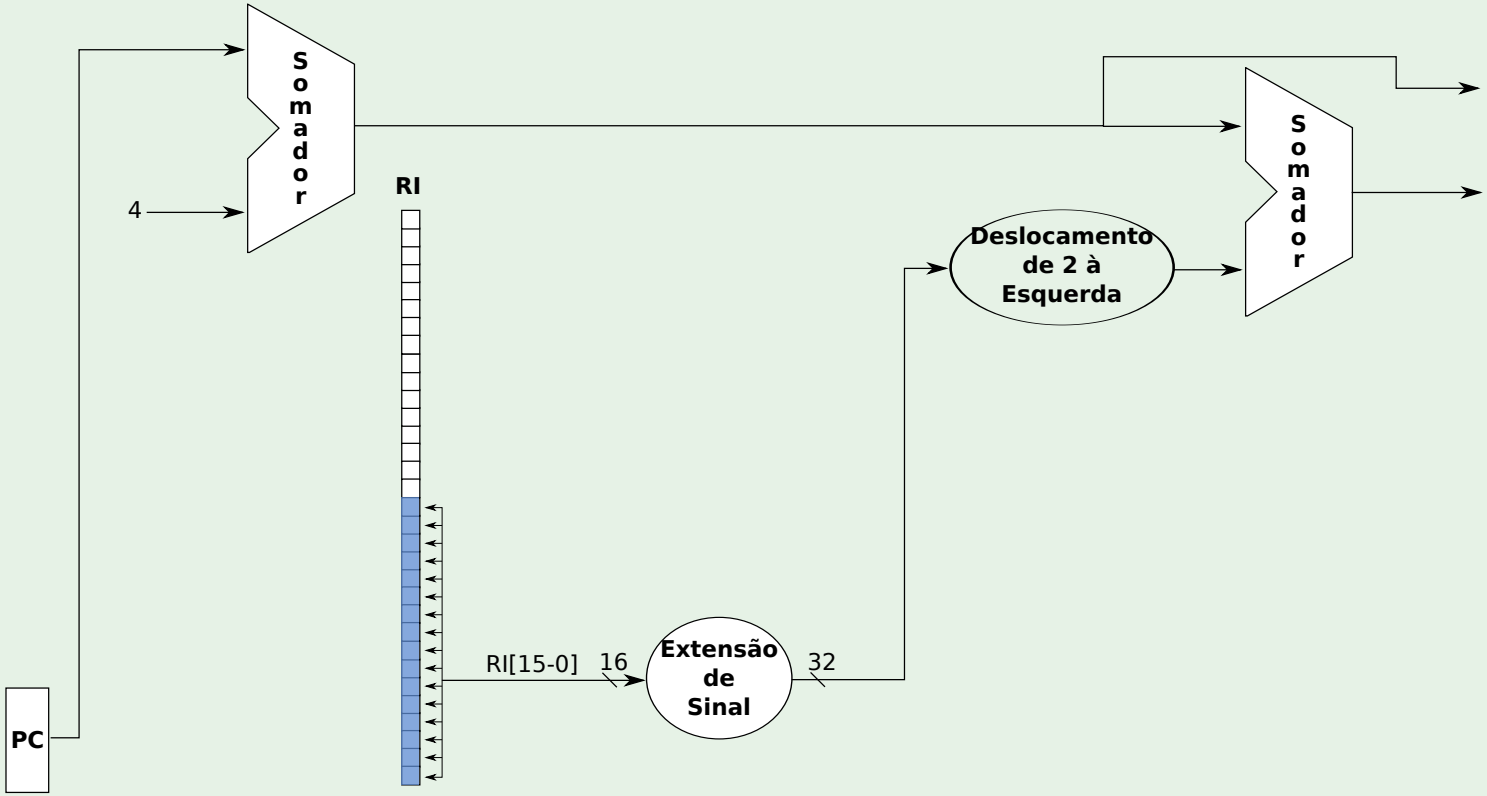
Atualizando o valor do PC

- *imm* vem dos 16 bits menos significativos da instrução
- para multiplicá-lo por 4:
  - Basta deslocá-lo 2 bits para a esquerda
  - precisa-se de um deslocador

- Para somá-lo com PC + 4, poderíamos usar a ALU
- Mas não vamos porque:
  - já estamos usando a ALU para fazer a comparação dos dois valores
  - será utilizado um somador simples

Branch on Equal: Execução

A lógica da atualização do PC

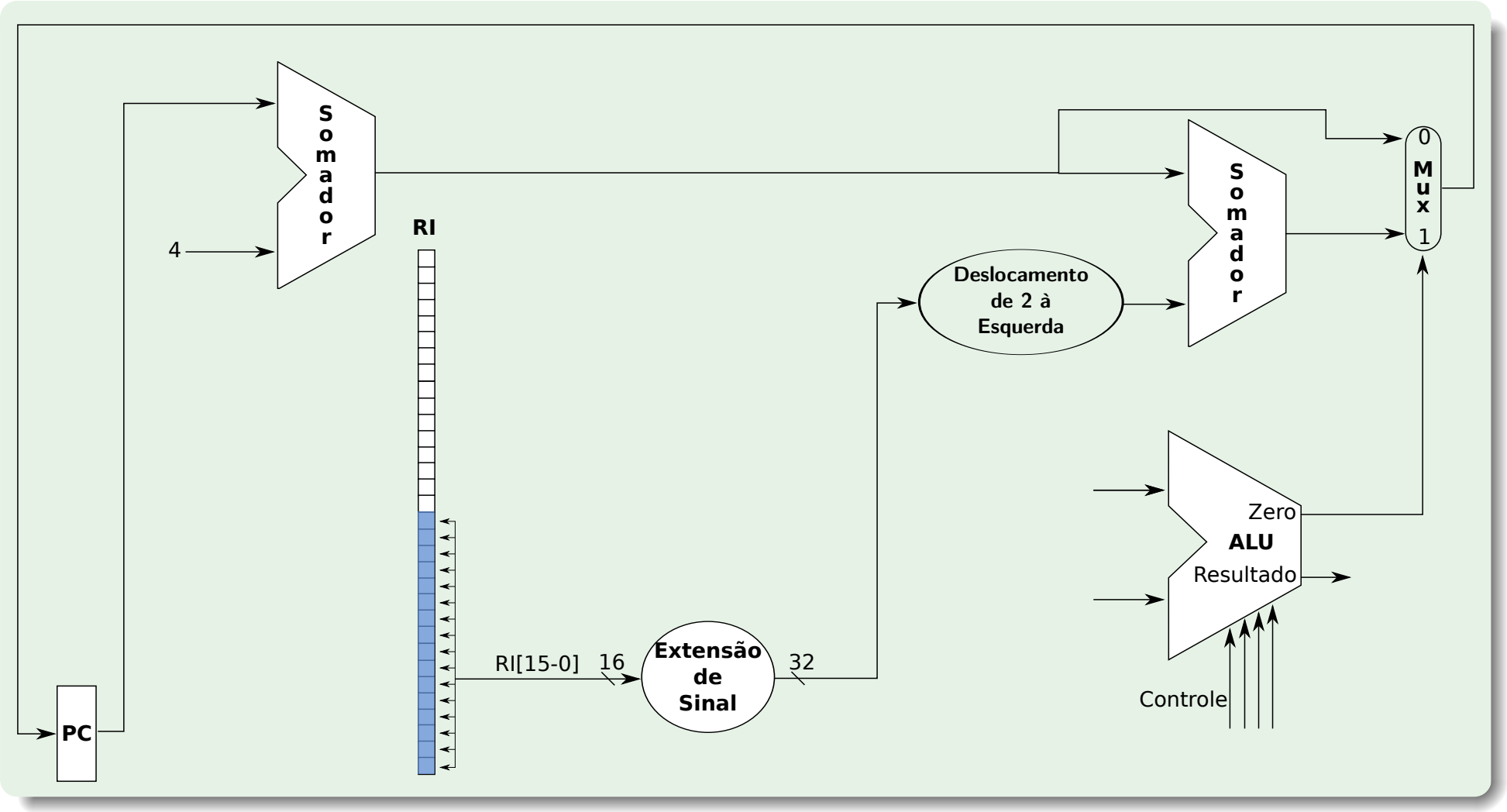


Branch on Equal: Armazenando o Resultado

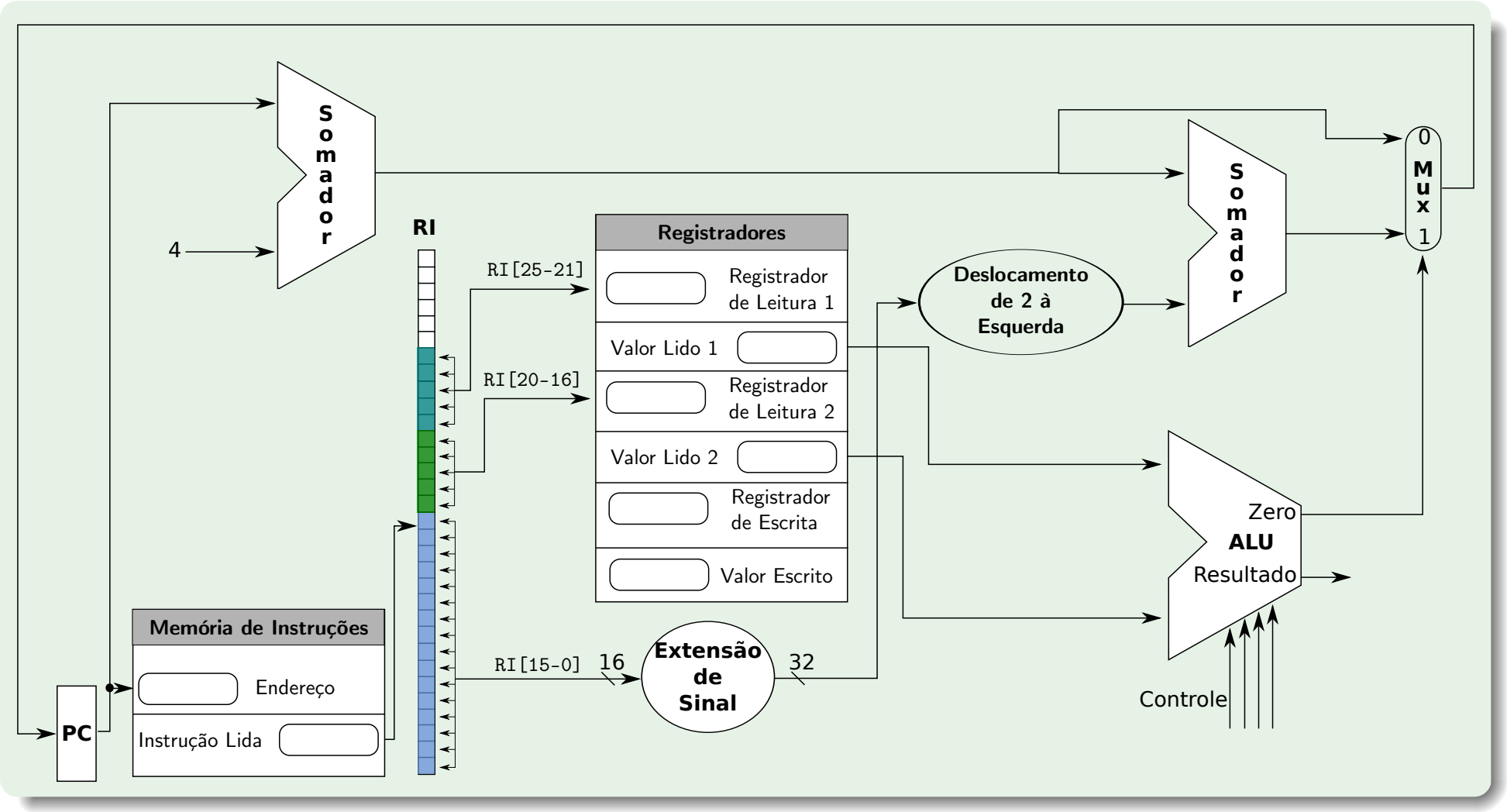
Novo valor de PC

- De acordo com o resultado da instrução de desvio
  - $PC + 4$, ou
  - $PC + 4 + 4 \times imm$
- Ao final da instrução este valor deve ser armazenado de volta no PC
- Na etapa de armazenamento de resultado, portanto, precisamos:
  - Decidir entre os dois valores
  - Escrevermos este valor de volta no PC
- A decisão entre os dois valores: através do bit zero da saída da ALU
  - é setado se o resultado da subtração (utilizado para comparar) é zero
  - se for zero, os dois valores são iguais

Branch on Equal: Armazenando o Resultado (II)



Branch on Equal: Juntando Tudo



Branch on Equal: Juntando Tudo (II)

- Assim como ocorreu para as instruções R, este caminho de dados está ainda incompleto.
  - **FALTA: linhas de controle com a ALU**

Instrução *Add Immediate*

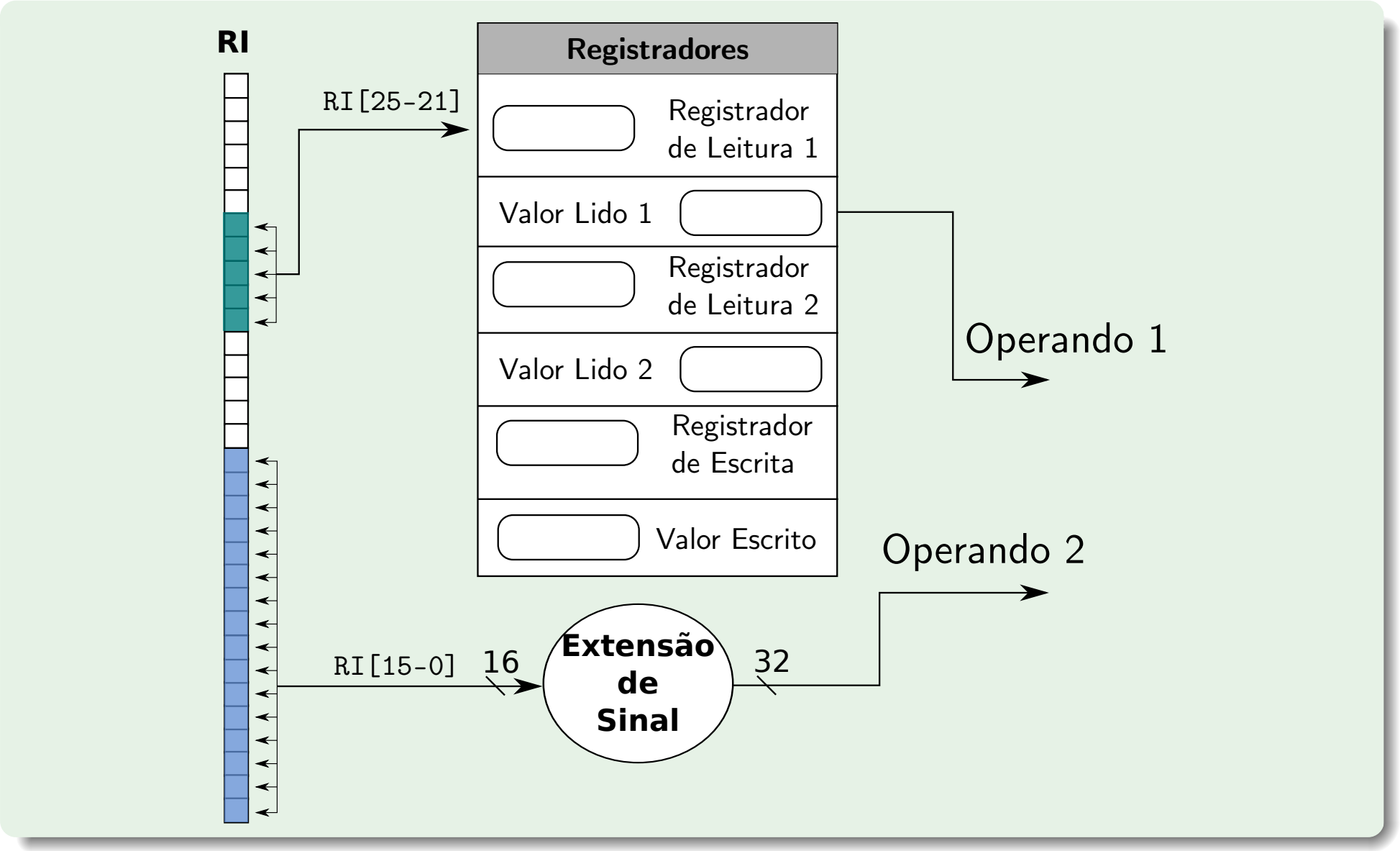
Formato:

001000 *sssss ttttt iiiiiiiiiiiiiiii*

Instrução no formato I

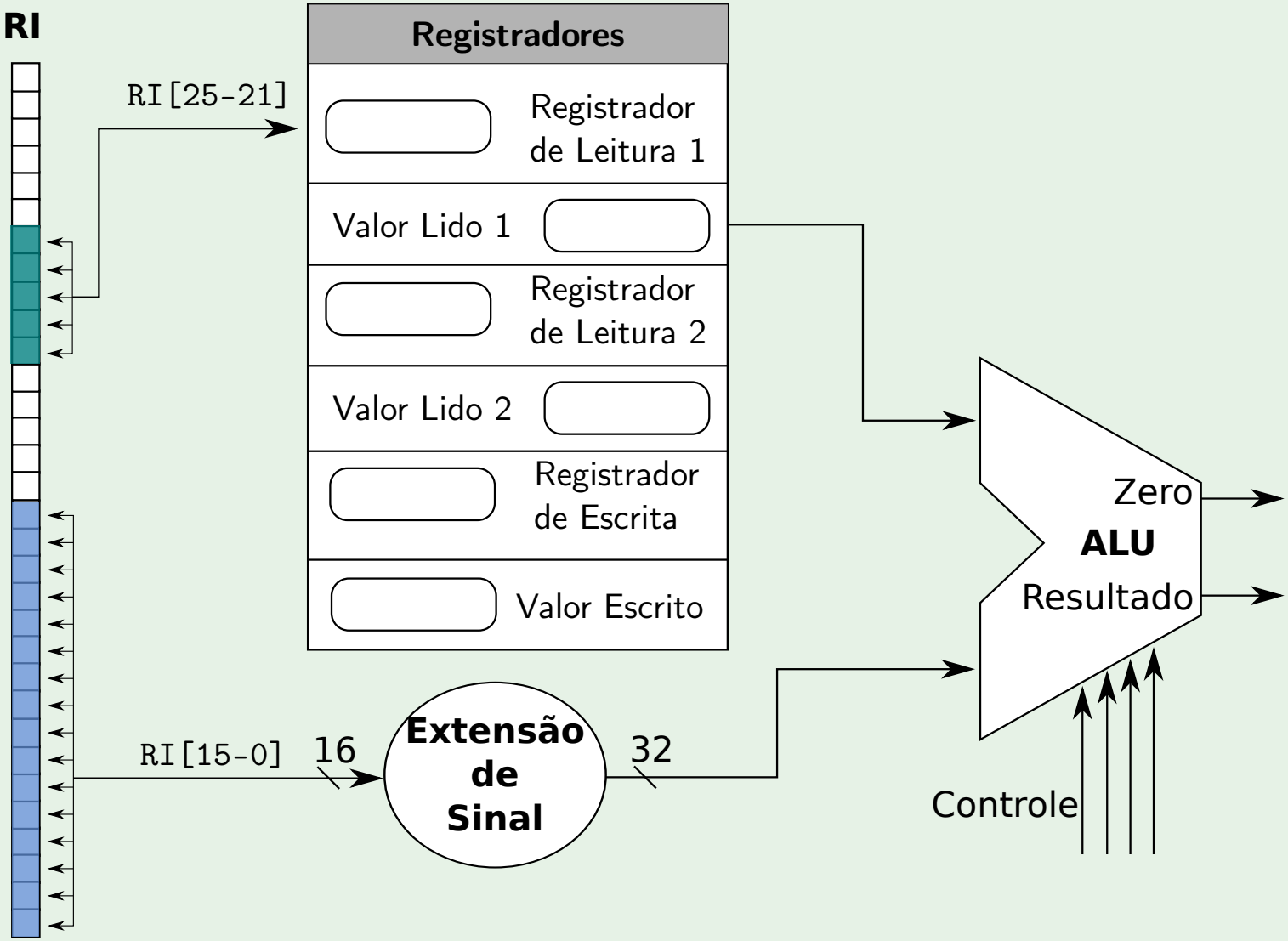
- Assim como a *branch on equal* que acabamos de ver
- Soma valor de um registrador com o valor do operando imediato
- Armazena resultado em um registrador
- De certa forma, essa instrução é uma mistura das instruções R com a instrução de desvio condicional
  - Usa o mesmo formato da de desvio
  - Mas executa uma operação aritmética e armazena resultado em registrador

Instrução *Add Immediate*: Busca de Operandos



Instrução *Add Immediate*: Execução

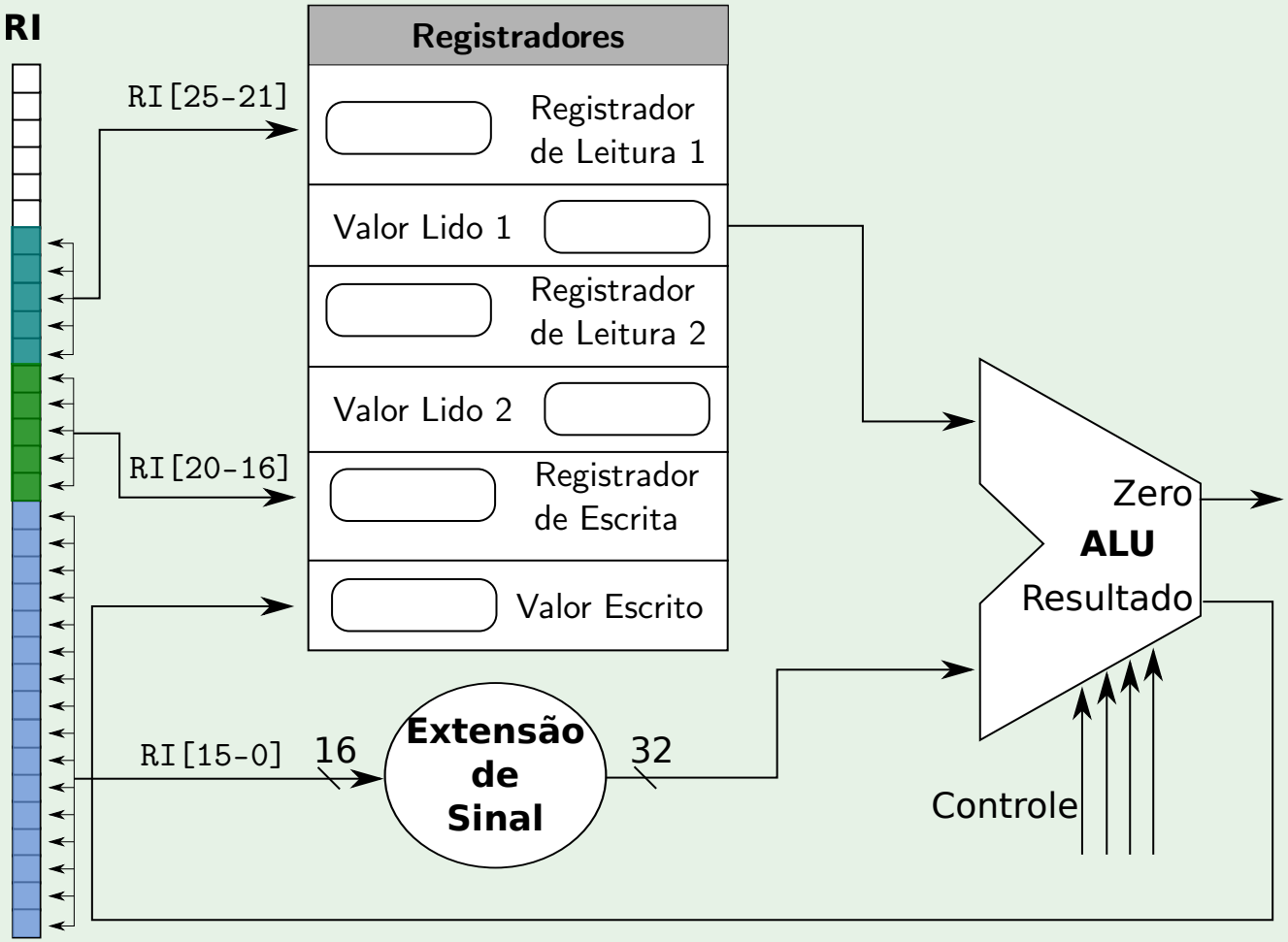
a soma realizada pela ALU



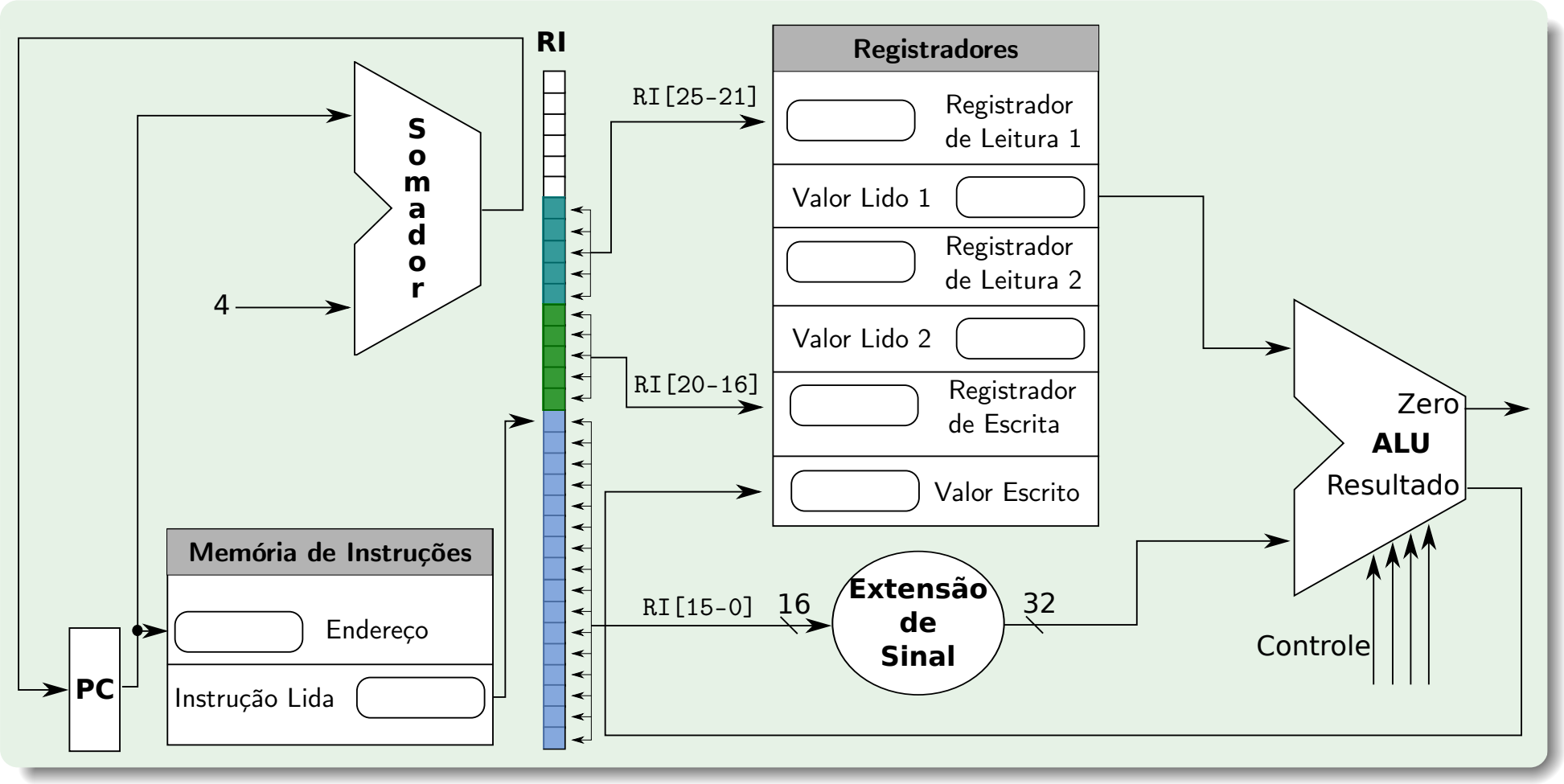
Instrução *Add Immediate*: Armazenando o Resultado

Saída da ALU deve alimentar o banco de registradores

- ID do registrador para escrita vem diretamente dos bits 20–16 do RI



Instrução *Add Immediate*: Juntando Tudo



Instrução *Add Immediate*: Juntando Tudo

- Este caminho de dados é mais um que utiliza a ALU.
- Novamente, não indicamos de onde virão as linhas de controle.
- Assim como fizemos nos outros casos, resolveremos isso depois.

Instruções de Transferência de Memória

load word e *store word*

$$\textit{loadword} \rightarrow 100011\ \textit{sssss ttttt iiiiiiiiiiiiiiii}$$

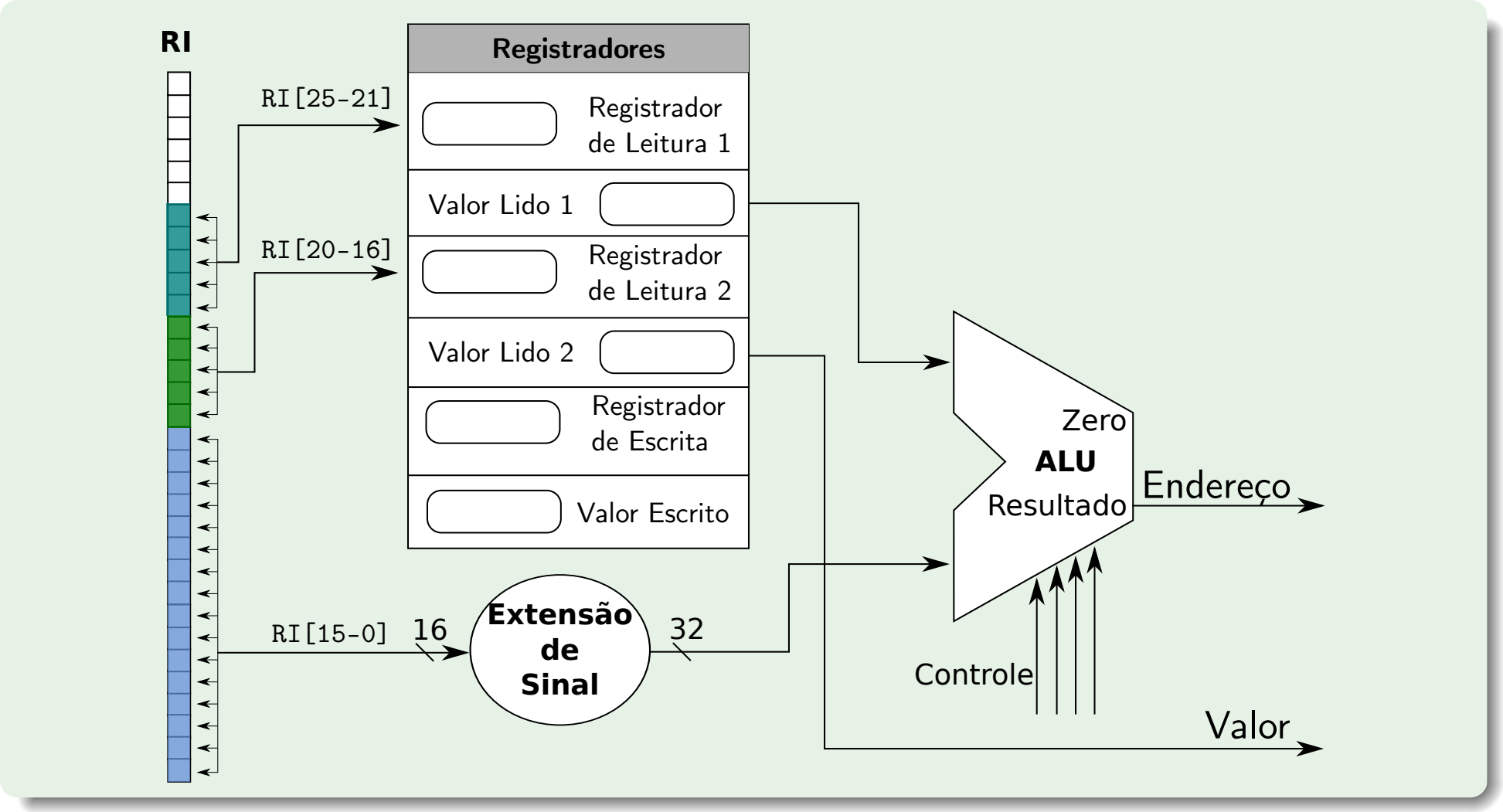
$$\textit{storeword} \rightarrow 101011\ \textit{sssss ttttt iiiiiiiiiiiiiiii}$$

- Formato I: dois registradores e um imediato
- O endereço da MP a ser acessado é dado por uma soma:
  - Valor do primeiro registrador somado ao operando imediato
- O segundo registrador é a origem/destino do dado a ser transferido

Inst. de Transferência de Memória: Busca de Operandos

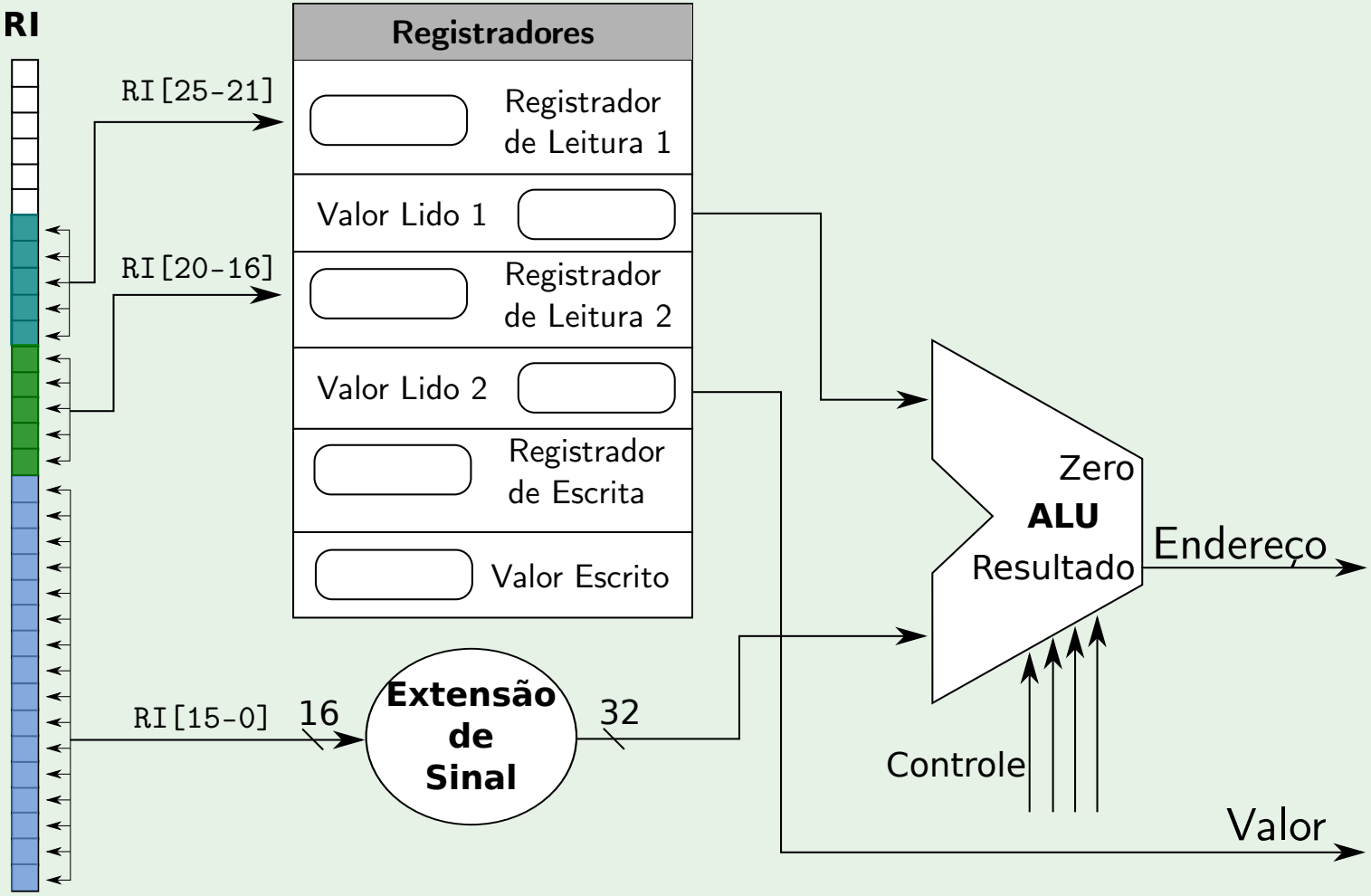
- Na instrução *load word*, antes da execução, precisamos determinar o endereço a ser acessado
- Na *store word*, adicionalmente, precisamos do valor a ser escrito
- O endereço a ser acessado precisa ser calculado
  - Usa-se a ALU para isso
  - Ela não será necessária para mais nada nestas instruções
- Para obter o valor a ser escrito na MP, basta passar o ID do registrador adequado para o banco de registradores

Inst. de Transferência de Memória: Busca de Operandos

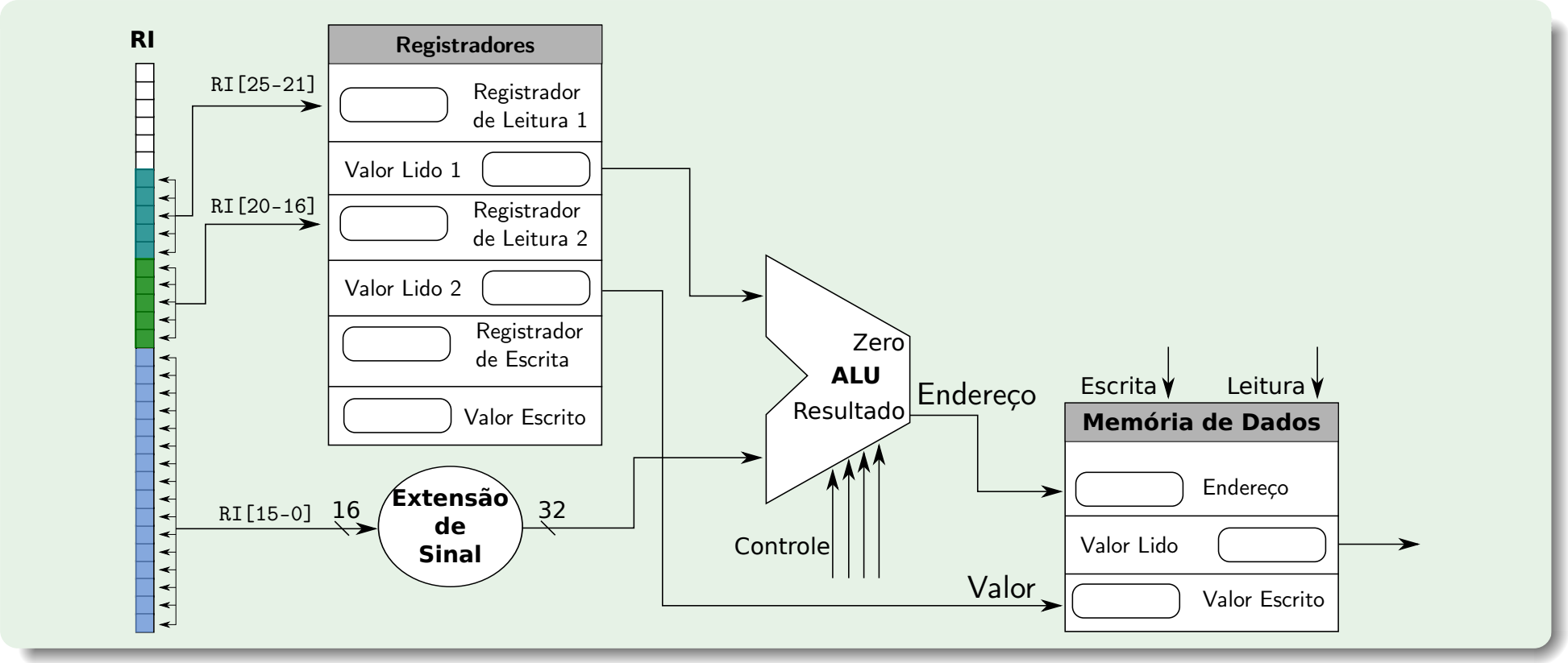


Instruções de Transferência de Memória: Execução

Com endereço a ser acessado e o valor a ser escrito (*store word*) – a execução é trivial



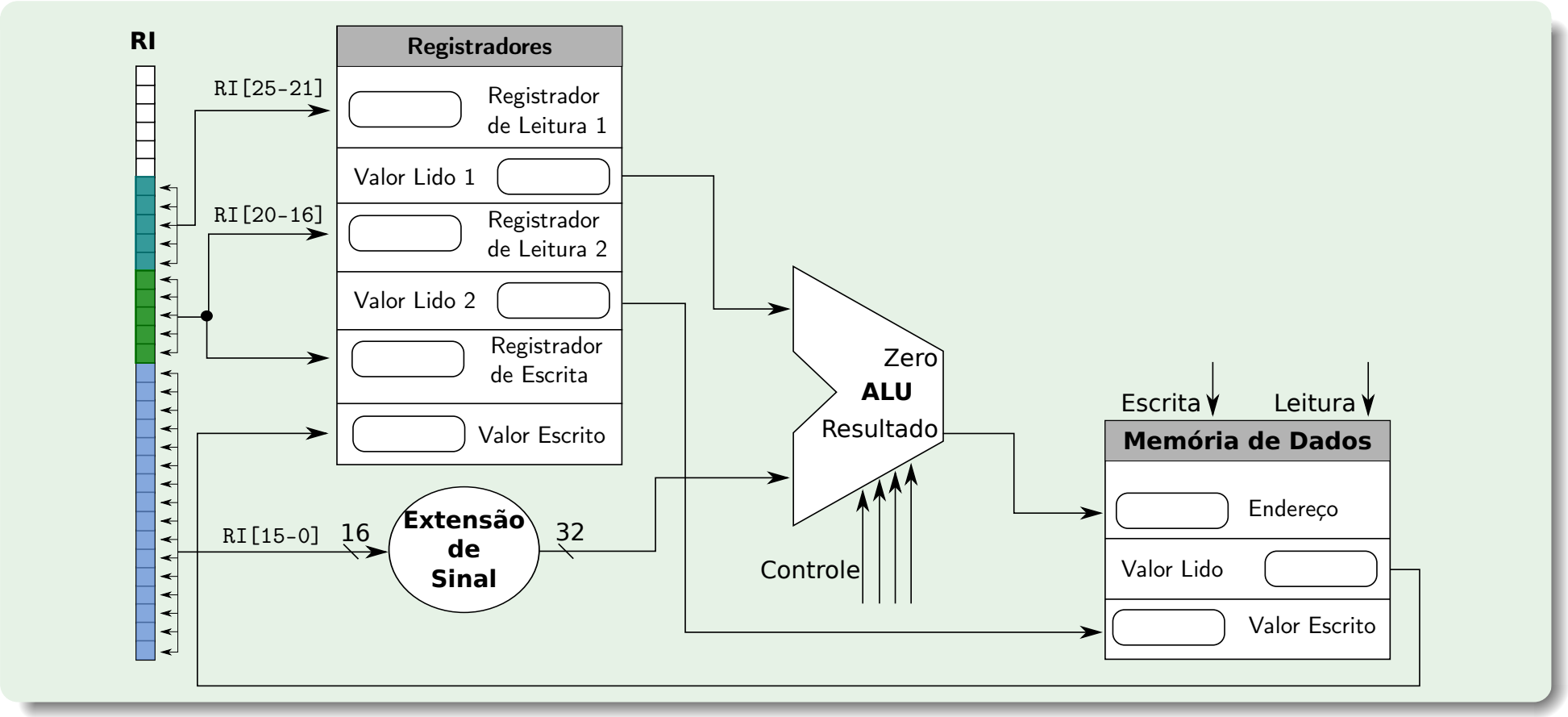
Instruções de Transferência de Memória: Execução



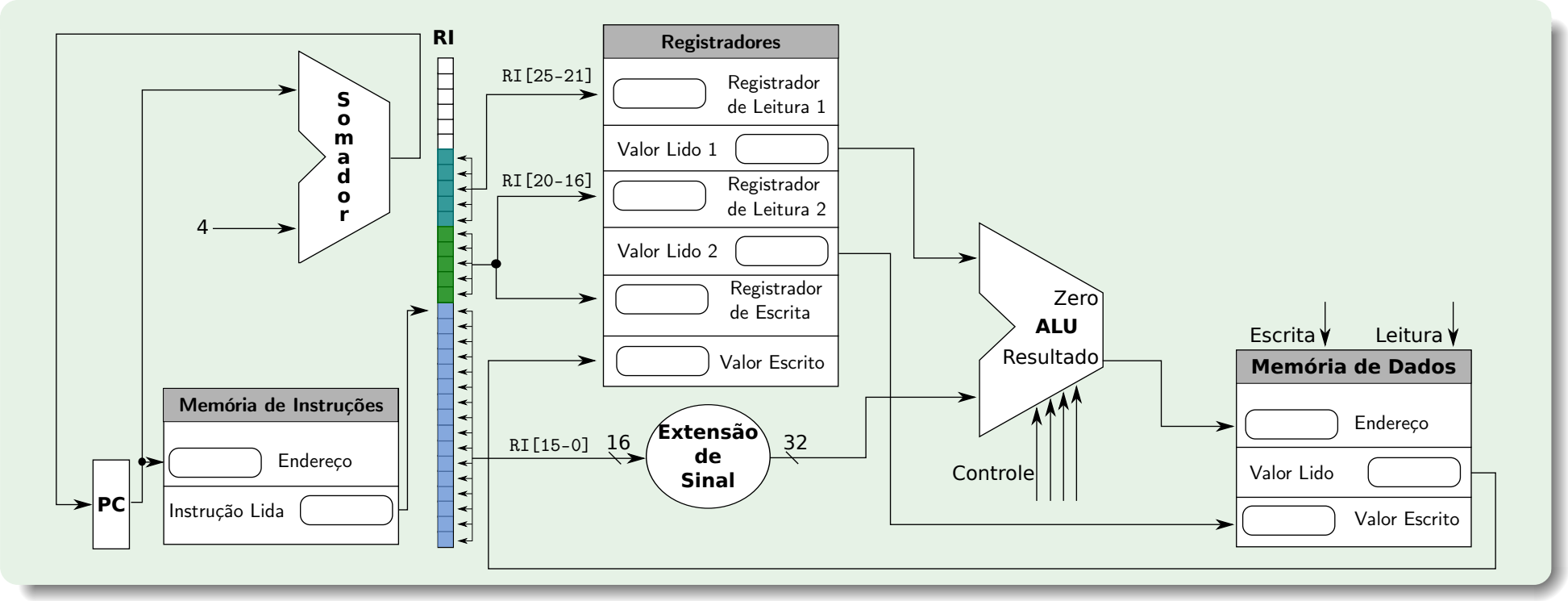
Instruções de Transferência de Memória: Armazenando os Resultados

- No caso da *load word*, a interface com a MP retornará o valor lido.
  - Precisa ser armazenado no registrador adequado
- No caso da *store word* o único armazenamento é a própria escrita do dado na MP
  - Tarefa realizada pela interface com a MP.
- Logo, nesta etapa, para a CPU, basta conectar a saída da interface com a MP, à entrada do banco de registradores
  - O ID do registrador vem diretamente dos bits do RI

Instruções de Transferência de Memória: Execução



Instruções de Transferência de Memória: Juntando Tudo



Instruções de Transferência de Memória: Juntando Tudo

- Novamente, ainda não indicamos de onde vêm as linhas de controle da ALU
- Ainda, neste caso temos mais duas a serem especificadas
  - As linhas de escrita e leitura da interface com a MP