UCP: Caminho de Dados (Parte II)

Cristina Boeres

Instituto de Computação (UFF)

Fundamentos de Arquiteturas de Computadores

Material baseado nos slides de Fernanda Passos

Na Aula Passada...

- Definimos alguns conceitos:
  - Caminho de dados: *hardware* que efetivamente executa as instruções.
    - ★ Realiza a **transformação** dos dados.
  - Unidade de controle: *hardware* que comanda o caminho de dados.
    - ★ Diz o que este tem que fazer.
    - ★ Define as linhas de controle dos seus componentes.
- Ilustraremos o funcionamento destes dois macro-componentes projetando passo a passo um processador.
  - Simplificado.
  - Mas capaz de rodar programas interessantes.

Estrutura Organizacional Macroscópica

Objetivo desta etapa

- Determinar mais claramente algumas especificações de *hardware*.
- Funcionamento de certos componentes
  - *e.g.*, quais linhas de controle eles possuem
  - *e.g.*, quantos bits em cada entrada/saída
- Também definiremos algumas **quantidades** de componente
  - como quantos registradores no processador

Número de Registradores

- Quantos/quais registradores serão necessários ao nosso processador?
- Vamos dividir a questão em duas partes:
  - Registradores especiais (*i.e.*, que tem papel no controle ou execução de tarefas do internas ao processador)
  - Demais registradores

Registradores Especiais

- Vamos manter o mínimo necessário para o funcionamento do nosso processador:
  - PC: (*program counter*) - o contador de programa
    - ★ armazena sempre o endereço da próxima instrução a ser executada
  - IR: registrador de instrução
    - ★ Necessário para guardarmos a instrução atual
    - ★ Precisaremos fazer coisas como decodificar o opcode
    - ★ Bits da instrução precisam ser guardados em algum lugar
  - MAR e o MBR
    - ★ Associados a interface com a memória

Registradores de Propósito Geral

- Registradores usados pelos programas
- Há um compromisso:
  - Mais registradores $\rightarrow$ mais informação é armazenada dentro da CPU
  - Menos registradores $\rightarrow$ menor custo, menor complexidade de projeto

- Se tivermos a **liberdade** de realizar esta escolha de projeto, precisamos **pesar** este compromisso

Registradores de Propósito Geral

- Ao especificar o formato de instruções e quantos bits são necessários para identificar um registrador:
  - O número de registradores está sendo **implicitamente** definido
- Por que?
  - No formato adotado, 5 bits identificam um registrador
  - 5 bits $\rightarrow 2^5 = 32$ combinações possíveis
  - Não adianta colocar mais registradores
  - Ter menos, é um desperdício de bits.

Registradores de Propósito Geral

- Por outra lado, o arquiteto pensa na arquitetura e daí, especifica o formato de instuções
  - Para ter n registradores, precisaremos de $\log_2 n$ bits no formato de instrução
- Conclusão:
  - O formato de instrução está interconectado com a organização do processador

Tamanho da Palavra

- Qual o tamanho da palavra do nosso processador?
  - a quantidade de bits armazenada em cada endereço/célula
- Outra vez, podemos analisar a especificação das instruções. Por exemplo, *add immediate*:
  - a constante especificada (imediato) é de 16 bits
  - então, uma informação/valor é armazenada em 16 bits
  - Logo, **não faz sentido** termos uma palavra de **menos** de 16 bits.
    - ★ Registradores, barramentos, *etc*, devem, ao menos, suportar o número de bits da constante

Tamanho da Palavra

- No entanto, ainda temos que analisar outro fato: as instruções têm 32 bits
  - **Idealmente**, o barramento de comunicação com a MP deve ser **largo o suficiente** para trazer **de uma só vez** todos os bits da instrução
    - ⋆ Não é estritamente necessário, mas é vantajoso
  - Além disso, o IR **necessariamente** terá 32 bits
    - ⋆ Manter os demais registradores do mesmo tamanho parece razoável/fácil para o projeto

- Enfim, temos um **processador de 32 bits**

Tamanho da Palavra

- Dado o tamanho da palavra, definimos os tamanhos de algumas estruturas do processador/computador:
  - Largura do barramento de dados com a memória: 32 bits.
  - Largura de todos registradores: 32 bits.
  - Largura das entradas/saída da ALU: 32 bits.
  - Número de bits usados para endereçar a MP: 32 bits.

- **Importante**:
  - Neste projeto, estamos decidindo usar o tamanho da palavra para todas estas estruturas.
  - **Nem todos os processadores são assim.**
    - ★ *e.g.*, alguns registradores maiores/menores.
    - ★ *e.g.*, barramentos mais largos.

Codificação de Números Negativos

- Decisão que também já apareceu nos formatos de instrução.
  - Especificamente, nas instruções *load word*, *store word* e *branch on equal*.
- Vamos utilizar sempre Complemento a 2.

A Unidade Lógica-Aritmética

- Com um conjunto de instruções é restrito, a ALU poderia ser simplificada
  - Cinco tipos de operação lógica/aritmética: soma, subtração, *Ou*, *E* e *set on less than*.
- Trataremos a ALU como uma caixa preta
  - Saberemos quais são as entradas e saídas, mas não como as funcionalidades são implementadas

ALU

- Entradas: duas, cada uma de 32 bits
- Manipulação de números com sinal: Complemento a 2
- Saídas:
  - Resultado de operação lógica/aritmética (32 bits)
  - Bit indicando se resultado é zero

A Unidade Lógica-Aritmética

- Linhas de controle:
  - Quatro.
  - Vide tabela ao lado.
- Note que nem todas as funcionalidades disponibilizadas pela ALU são úteis para o nosso conjunto de instruções.
  - As operações desnecessárias (como a NOR) serão apenas ignoradas.
  - Utilizaremos apenas as demais.

Linhas de Controle	Função
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

Banco de Registradores

- No nosso conjunto de instruções, as que acessam mais registradores são as do tipo R.
  - Dois registradores lidos e um escrito.
- Adicionalmente, note que:
  - Nenhuma instrução lê mais que dois registradores.
  - Nenhuma instrução escreve mais que um registrador.
- Logo, o banco de registradores precisa dar suporte a:
  - Leitura de dois registradores.
  - Escrita em um.

Banco de Registradores (II)

- Resumindo:
  - Quatro entradas.
    - ★ Identificadores dos dois registradores de leitura.
    - ★ Identificador do registrador para escrita.
    - ★ Valor a ser escrito.
  - Duas saídas:
    - ★ Valores dos registradores lidos.
- Linhas de controle:
  - Discutidas posteriormente.

Interconectando Componentes

Fase de implementar a lógica de funcionamento da arquitetura

- Precisamos garantir que cada instrução faça tudo o que deve fazer
  - *e.g.*, uma instrução de branch não deve alterar registradores
- Nosso projeto será realizado de forma **incremental**.
  - Inicialmente: funcionalidades básicas
  - Adicionaremos mais complexidade/funcionalidade aos poucos
- De forma concomitante:
  - Partes da unidade de controle.
  - Partes do caminho de dados.

Busca da Próxima Instrução

Qual é a primeira etapa no ciclo de execução de uma instrução?

- Busca da instrução na MP

Passos que devem ser realizados:

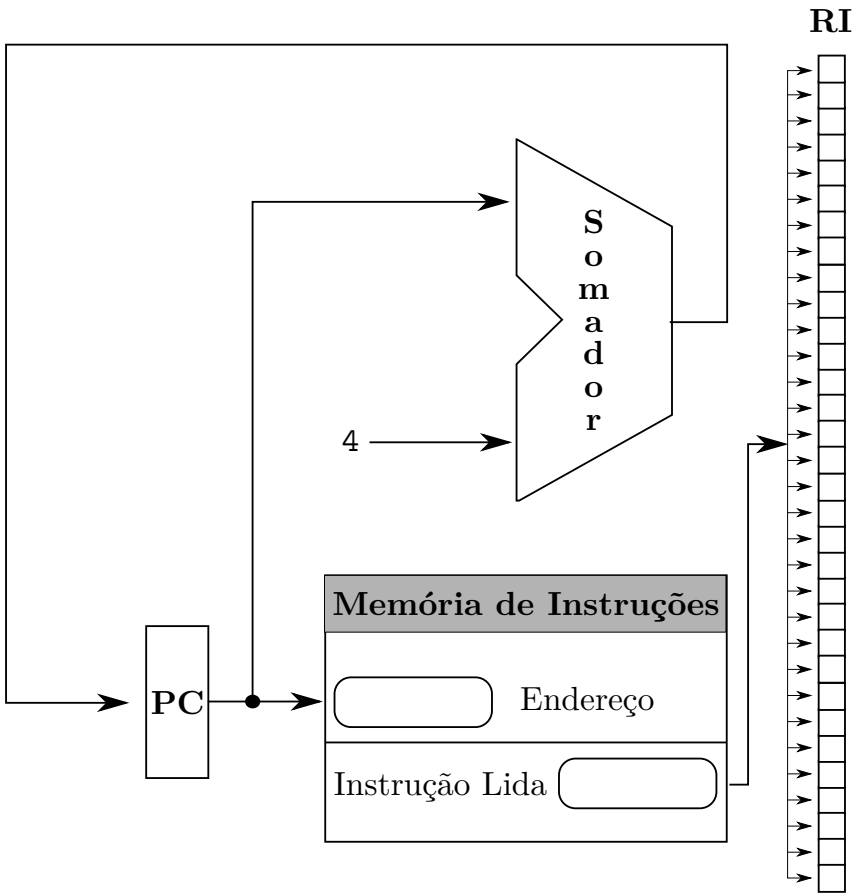
- Instrução apontada pelo PC é trazida da MP para o IR
  - **leitura da MP no endereço indicado por PC**
- Ao mesmo tempo, **PC é incrementado**.
  - Pode-se aproveitar e projetar a arquitetura para que as duas funções sejam realizadas
    - Incrementar o PC é apontar para a próxima instrução a ser executada
    - Na maioria das vezes, a próxima instrução está no endereço seguinte.

Busca da Próxima Instrução

Componentes para implementar esta funcionalidade

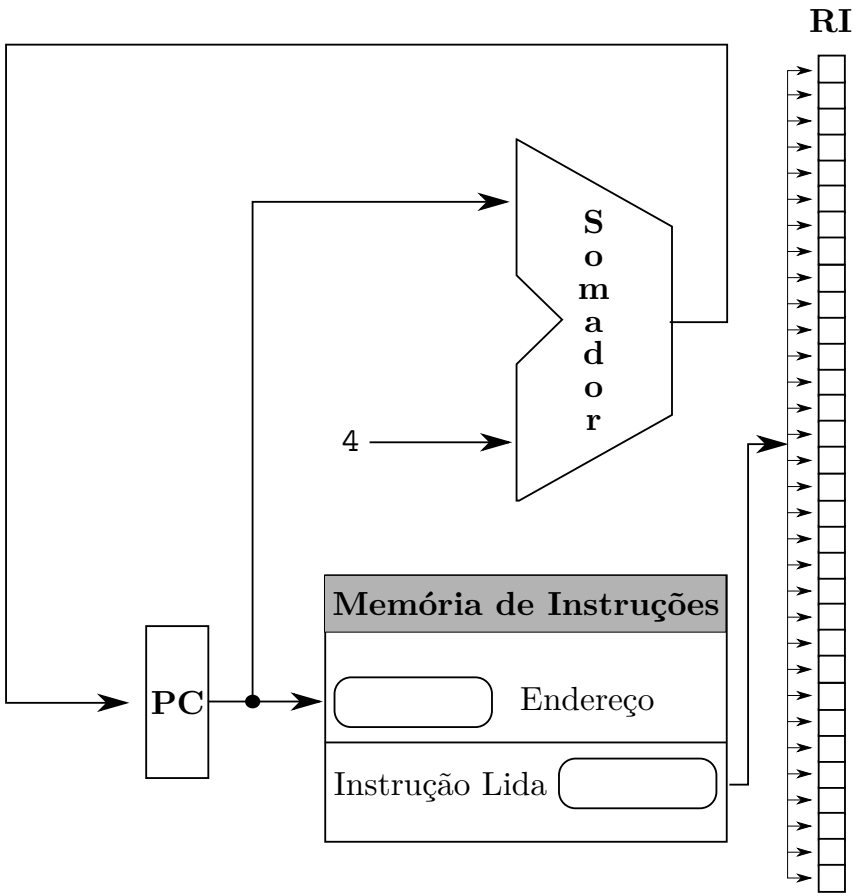
- Obviamente, o registrador PC
  - Endereço é lido dele
  - Ele é incrementado
- Para incrementar o PC, também precisamos de um **somador**
- Também precisamos acessar a memória, *i.e.*, de uma interface com a MP

Busca da Próxima Instrução



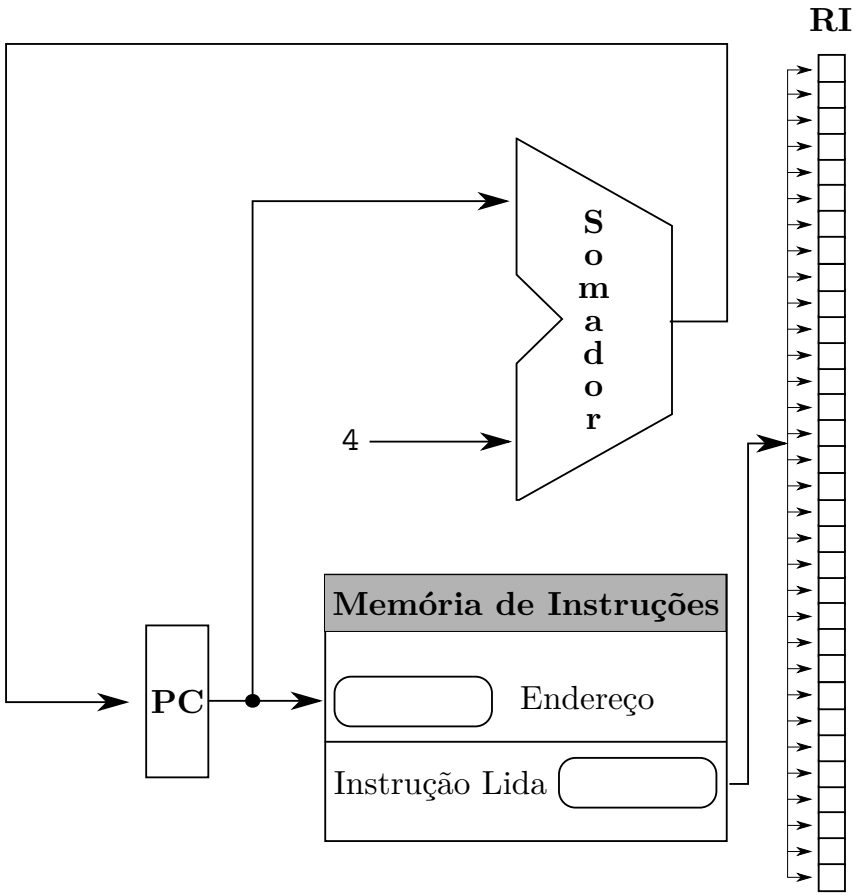
- Conectamos a saída do PC a duas entradas:
  - interface com a memória
  - um somador
- A interface com a memória lê a instrução e a coloca na saída.
- A outra entrada do somador é conectada a um sinal constante, representando o número 4.
  - Por que 4, e não 1?

Busca da Próxima Instrução



- Conectamos a saída do PC a duas entradas:
  - interface com a memória
  - um somador
- A interface com a memória lê a instrução e a coloca na saída.
- A outra entrada do somador é conectada a um sinal constante, representando o número 4.
  - Por que 4, e não 1?
  - Porque **nossa memória é endereçada em bytes** e uma instrução tem 4 bytes (32 bits)

Busca da Próxima Instrução: Outros Detalhes



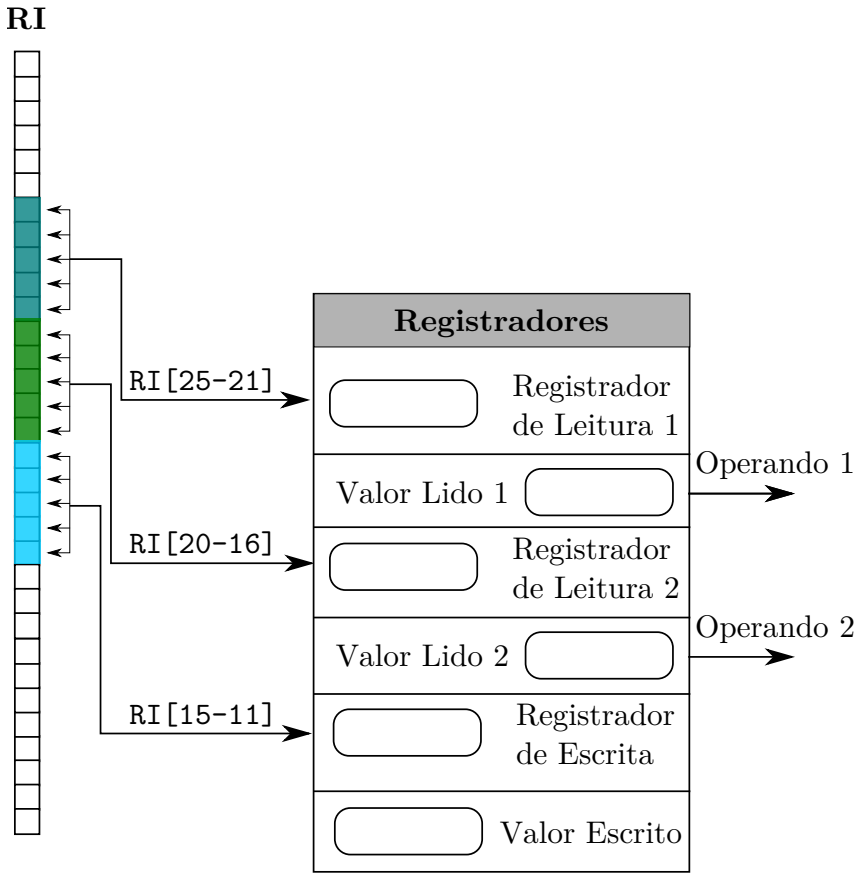
- Note que estamos realimentando o PC com base no seu próprio valor (mais 4)
- Para que isto funcione, este diagrama está subentendendo alguns detalhes
  - Existe um gerador de clock
  - No início de um período do clock, PC coloca seu valor na saída
  - Apenas depois ocorre a atualização para o valor na entrada

Execução de instruções do Tipo R

Instruçao foi buscada: é do tipo R

- São as instruções lógicas/aritméticas.
  - Operação é realizada na ALU
  - Os dois operandos de entrada: estão em registrador
  - A saída também é armazenada em registrador

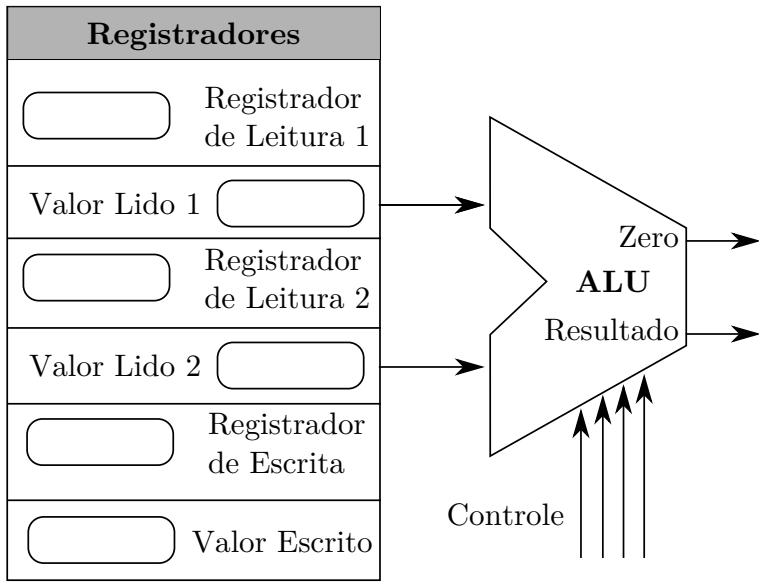
Instruções do Tipo R: Busca de Operandos



- Todos os operandos são registradores.
- E todos estão identificados por conjuntos de 5 bits na instrução:
  - Operando 1: bits 21 a 25.
  - Operando 2: bits 16 a 20.
  - Resultado: bits 11 a 15.
- Temos que "separar" estes bits e usá-los como entrada do banco de registradores.

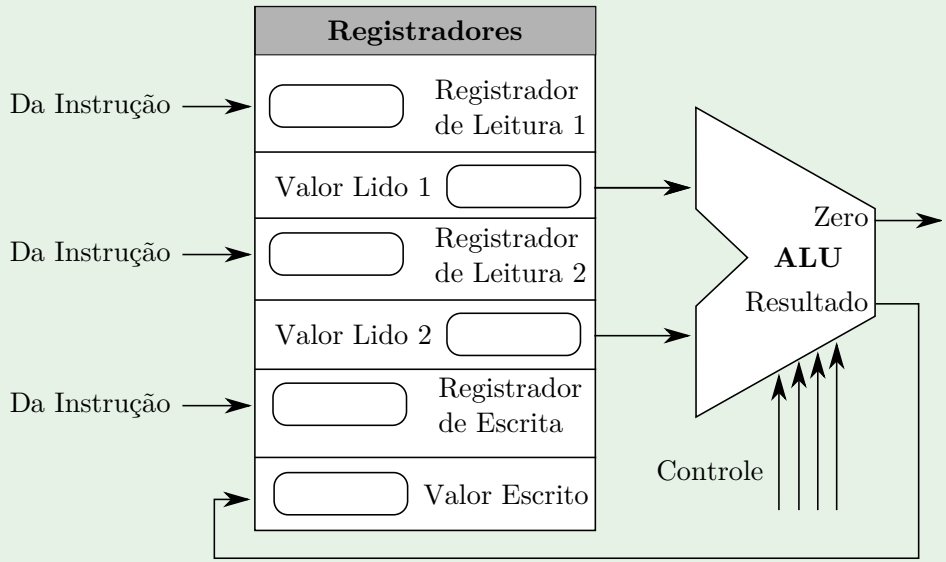
Instruções do Tipo R: Execução

- Identificados os operandos, a operação é realizada pela ALU
- Logo, temos que conectar as saídas do banco de registradores às entradas da ALU.



Instruções do Tipo R: Armazenando o Resultado

- Em instrução tipo R, resultado deve ser armazenado em um registrador
- Este resultado é gerado pela ALU – é a saída da ALU
  - Identificador especificado em bits da instrução
- Logo, temos que conectar a saída da ALU à entrada do banco de registradores



Instruções do Tipo R: Juntando Tudo