Processador: Conceitos Básicos e Componentes

Cristina Boeres

Instituto de Computação (UFF)

Fundamentos de Arquiteturas de Computadores

Material baseado nos slides de Fernanda Passos

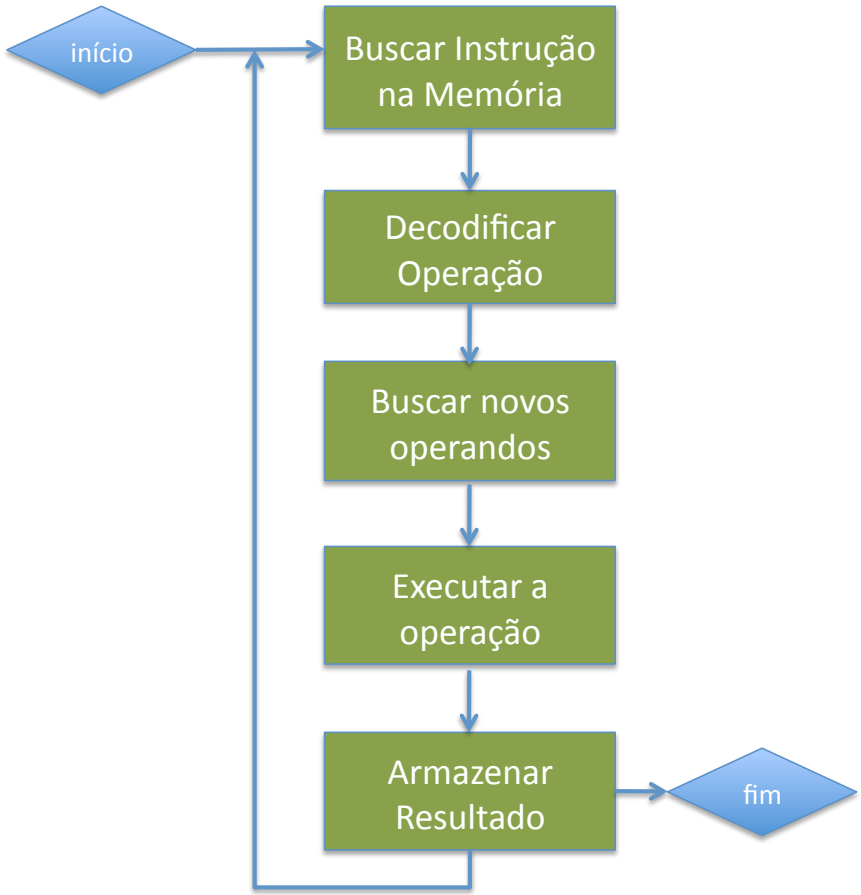
Processador: Linhas gerais

- Todo processador é construído de modo a ser capaz de realizar algumas operações:
  - Somar, subtrair, multiplicar ou dividir números
  - Fazer operações lógicas
  - Mover dados da memória para a CPU
  - Transferir um valor para um dispositivo de E/S
- Processador executa instruções: ele é um interpretador de um programa executável

Ciclo de Instrução

Execução de uma instrução de máquina passa por várias etapas:

- Buscar instrução na MP
- Decodificar o conteúdo
- Buscar operando, se necessário
- Executar a operação
- Armazenar resultado, se necessário



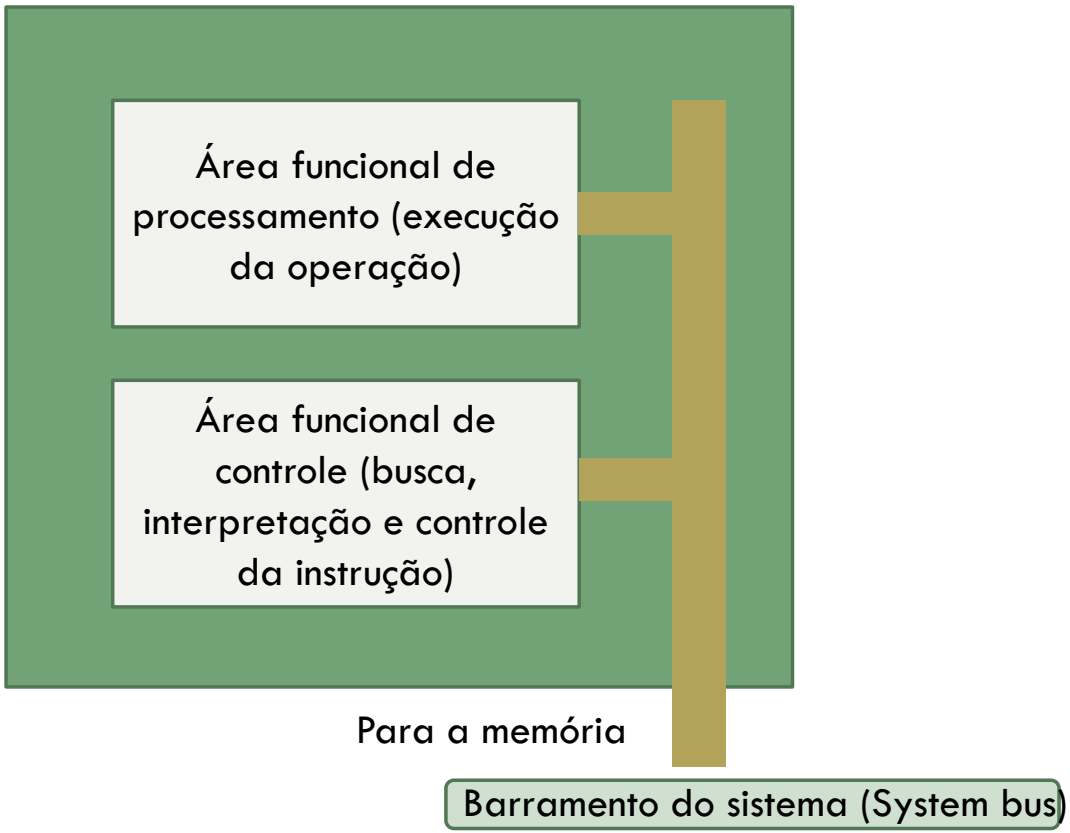
Processador

Opera em ciclos. Sempre repete a sequencia de etapas

- Mas para instruções/operações diferentes
- sobre operandos diferentes

As diferenças em instruções e operandos geram resultados diferentes

Áreas funcionais de um processador

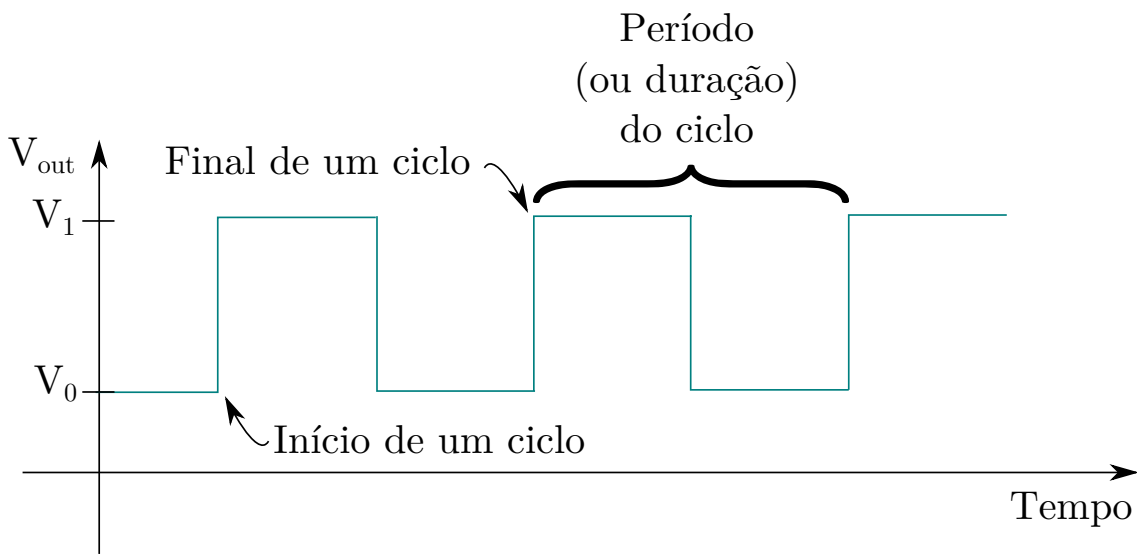


Componente de Controle: Relógio

- Processador opera em ciclos
- Ciclos têm início e fim
- **Algum componente precisa demarcar estes eventos!**
  - *i.e.*, indicar quando começa e quando termina um ciclo.
- Este componente é o **gerador de clock/relógio**.

Componente de Controle: Relógio

- Como o processador é um dispositivo eletrônico, indicação precisa ser baseada em **sinais elétricos**
- Função do gerador de clock é apenas criar este sinal.
  - Sinal periódico.
  - Ondas quadradas.



Componente de Controle: Relógio

Sinal possui transições

- Baixo para alto e de alto para baixo

Transições sincronizam funcionamento de outros componentes

- Certas ações são realizadas nas transições, ou começam a ser realizadas.

Frequência de um processador

- **inverso do período** do sinal gerado pelo clock
- Exemplo: Frequência de 1 GHz
  - Período de $\frac{1}{1GHz} = 1$ ns: cada ciclo dura 1 ns.

Frequência de Clock

Frequências típicas atuais

- processadores operam hoje (tipicamente) de 1 a 3 GHz
- Vários fatores **limitam** (na prática) essa **frequência**:
  - Consumo energético
  - Dissipação de calor
  - Tempo de propagação

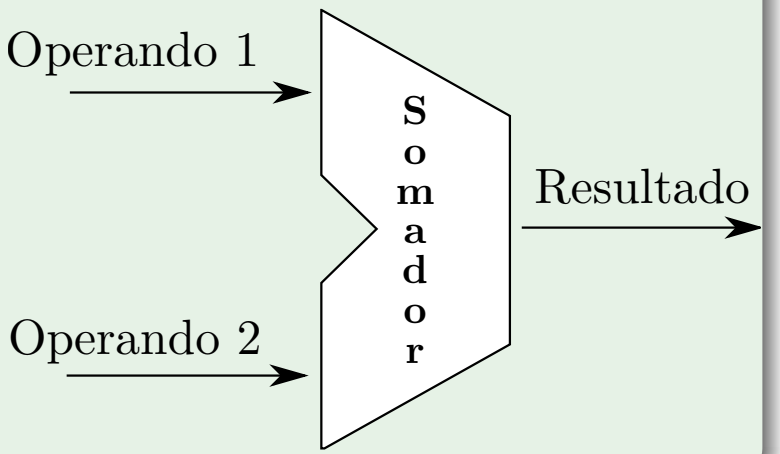
Somadores

Somadores são extremamente importantes

- Em um único ciclo de relógio:
  - Incrementar o PC
  - Realizar operações de soma
  - incrementadores de 1 unidade

Entrada e Saída de um somador

- Cada entrada é um conjunto de n bits
- A saída também é composta por n bits
  - Como se os bits representassem valores binários positivos

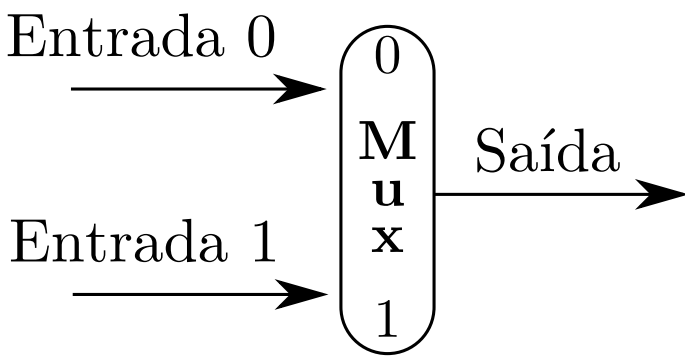


Multiplexadores

Multiplexadores são componentes que implementam escolhas

- Recebem duas ou mais entradas
- Recebem uma ou mais **linhas de controle**
- Repetem na saída a entrada selecionada

- Exemplo de multiplexador com 2 entradas de 1 bit cada:



Entrada 0	Entrada 1	Controle	Saída
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Multiplexadores

- Multiplexadores com 2^k entradas têm k linhas de controle
- Exemplo:
  - Multiplexador de 8 entradas para 1 saída:
  - precisam de 3 bits para identificar a entrada – Três linhas de controle (000 a 111)
- Obs.: cada entrada e cada saída pode ter múltiplos bits (são barramentos)

Em suma:

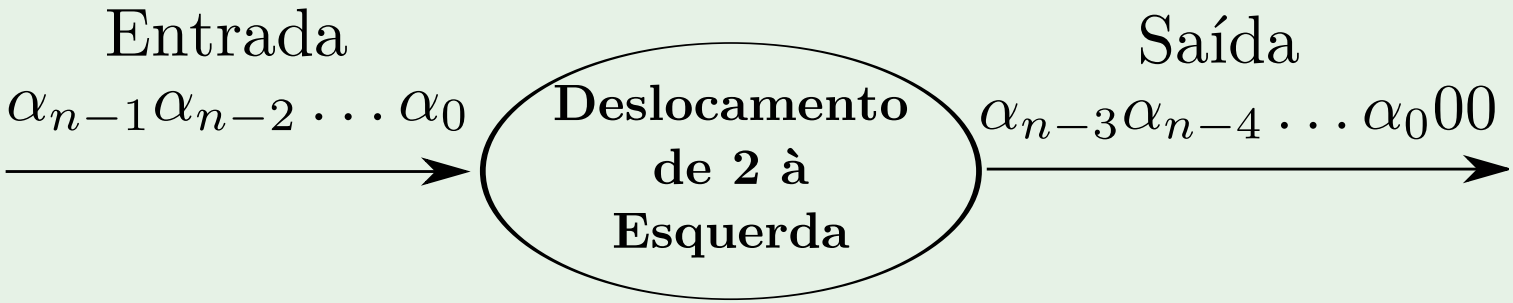
- multiplexidores são conectores de várias entradas (caminhos) para uma saída

Deslocadores

Recebem uma entrada: um conjunto de bits. A saída:

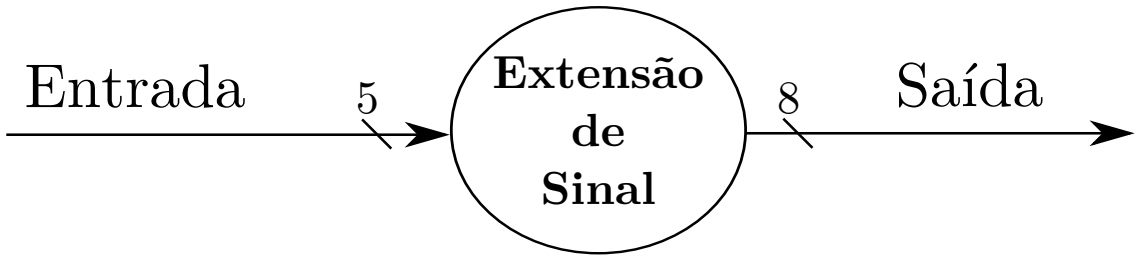
- deslocamento dos bits para esquerda (ou direta), de k bits:
  - os k bits mais (ou menos) significativos (dependendo do deslocamento) são perdidos
  - São adicionados k bits 0 ao início (ou fim) do valor original

Deslocando dois bits



Extensores de Sinal

- Extensores de sinal servem para aumentar o número de bits de um conjunto representando um número inteiro.
- Exemplo:
  - Temos um conjunto de 5 bits representando um número inteiro.
  - Número representado em Complemento a Dois.
  - Queremos transformá-lo no mesmo valor, mas representado com 8 bits.
  - Temos que **repetir o bit do sinal** mais 3 vezes à esquerda.
- Representação:



Extensores de Sinal (II)

- De forma genérica, extensores de sinal se preocupam em aumentar o número de bits sem alterar a correção da representação do número.
- Podemos construir extensores de sinal para qualquer representação.
  - Complemento a dois
  - Representação em excesso
  - Sinal e Magnitude
  - ...
- Assumiremos aqui extensores de sinal para Complemento a Dois

Registradores

Pequenas unidades de memória

- Número de bits geralmente dado pelo **tamanho da palavra**
- Mas alguns registradores podem ser maiores ou menores, dentro do mesmo processador (depende de sua função)

Registradores armazenam estado

- Valores não são perdidos de um ciclo para outro.
- Resultado de um ciclo depende da instrução executada, dos operandos **e do estado interno do processador**

Registradores: Tipos

Existem registradores de propósito geral

- Podem ser utilizados da forma que o programador preferir
- armazenam dados e acumulam resultados

Existem registradores de utilidade específica

- MAR: armazena endereço da MP a ser acessado
- MDR: armazena dado recém lido ou a ser escrito na MP
- PC: armazena endereço da próxima instrução a ser buscada

Registradores: Quantidade

Processador só pode operar sobre valores que estão em registradores

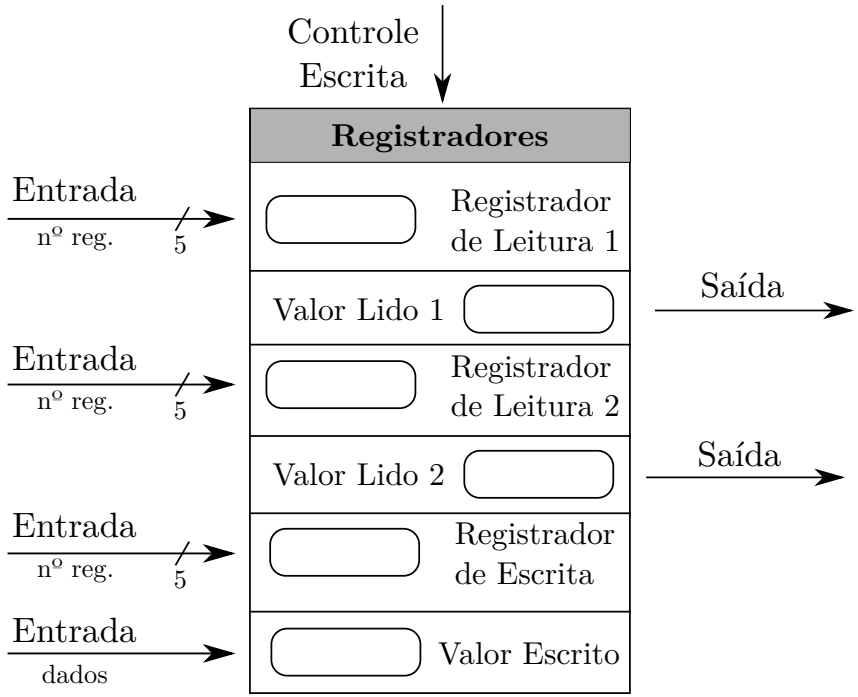
- Dado que não está em um registrador precisa ser trazido previamente para um
- Se os dados do nosso programa estiverem em registradores
  - Maior rapidez de execução
  - Quanto mais registradores, maior eficiência de execução
- Mas a adição de registradores tem um custo
  - O que torna o número de registradores relativamente limitado.

Registradores e Arquiteturas

- Arquiteturas diferentes possuem quantidades diferentes de registradores.
- Exemplos:
  - Arquitetura MIPS:
    - ★ 32 registradores de propósito geral (inteiros).
  - Arquitetura x86_64:
    - ★ 16 registradores de propósito geral *acessíveis ao programador*.
    - ★ Embora, fisicamente, CPU possa implementar mais.

Banco de Registradores

- Engloba um conjunto de registradores (de propósito geral)
  - identificados de 0 a 31
- Dois registradores serão somente de leitura
- Uma linha de controle que determina se escrita deve ser realizada ou não nos restantes



Interface com a Memória

As interfaces com a memória recebem os seguintes bits de entrada

- Endereço da MP
- Valor a ser escrito
- dois bits de controle
  - Especificam se a leitura e a escrita devem ser realizadas
  - Em um dado ciclo, não podemos fazer leitura e escrita

Por que não usar apenas uma linha de controle para selecionar entre leitura e escrita?

- Por que queremos também uma terceira opção: nem ler, nem escrever.

Decodificadores

- Vimos até aqui vários componentes que dependem de linhas de controle.
  - Interfaces com a memória
  - Banco de registradores
  - Multiplexadores
  - ...
- Em um processador, o estado de cada um destas linhas é determinado de acordo com:
  - Tipo de instrução
  - Operandos
  - Estado interno do processador
- Considere o tipo de instrução:
  - Como, a partir dela, determinamos o estado de um conjunto de linhas de controle?

Decodificadores

- Em geral, utilizamos uma parte dos bits da instrução
  - Como o opcode.
- A partir destes bits, sabemos a operação a ser realizada
  - O que determina uma série de linhas de controle
  - não ativar a escrita em registradores para uma instrução de escrita em MP
- Estas linhas de controle poderiam ser mapeadas diretamente para bits do opcode
- Mas:
  - Em geral, há um número grande de linhas de controle.
  - E nem todas as combinações são válidas.
    - ★ *e.g.*, linhas de leitura e escrita da interface com a memória ligadas simultaneamente;
    - ★ ou fazemos exclusivamente a escrita (ativa escrita) ou fazemos exclusivamente leitura (ativa leitura).

Decodificadores

Opcode: Compactando bits de controle

- Ao invés da instrução do programa conter todos os bits de controle de todos os componentes da processador
  - opcode: código de operação, que codifica todos esses bits
  - economia de bits, no tamanho do processo
- Logo precisa ser transformado para o tal conjunto de bits de controle:
  - através do decodificador
- Cada valor do código está associado a um conjunto de valores para as linhas de controle

Decodificadores

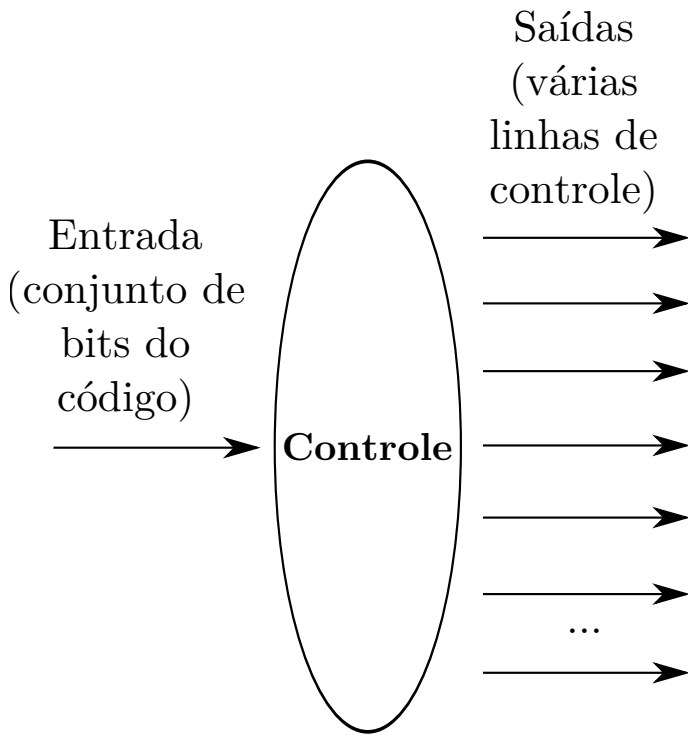
Em uma arquitetura:

Código	Ação
00	escrita em registrador **desativada** leitura em MP **desativada** escrita em MP **desativada**
01	escrita em registrador **desativada** leitura em MP **ativada** escrita em MP **desativada**
10	escrita em registrador **desativada** leitura em MP **desativada** escrita em MP **ativada**
11	escrita em registrador **ativada** leitura em MP **desativada** escrita em MP **desativada**

Conclusão: com 2 bits, conseguimos controlar 3 linhas de controle

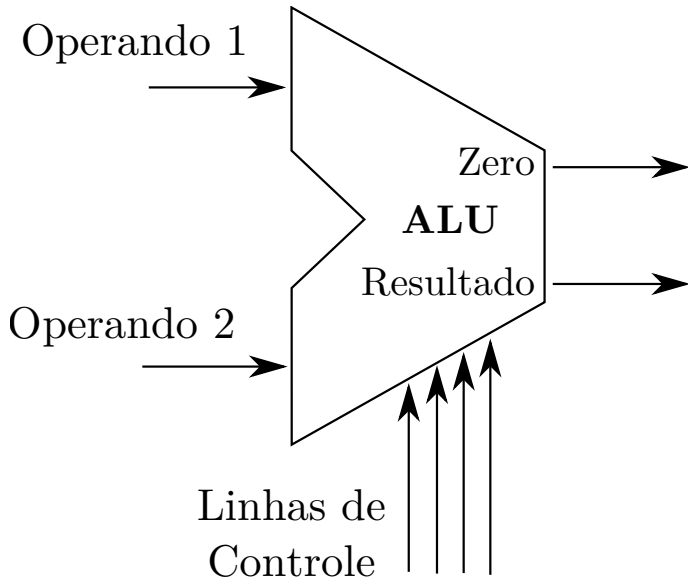
Decodificadores

- O componente que realiza a conversão dos bits de código para as linhas de controle é chamado **decodificador**.
- Recebe um código como entrada.
  - Composto por um conjunto de bits.
- Retorna um conjunto de linhas de controle.



A Unidade Lógica-Aritmética

- Também chamada pela sigla ULA.
  - Ou ALU, da sigla em inglês.
- Um dos principais componentes do processador.
- Executa operações lógicas ou aritméticas sobre par de operandos.
- Saída consiste no resultado da operação.
  - E, algumas vezes, em **flags** de controle.



ALU: Operações

- Dependendo do processador, a ALU pode ser mais ou menos complexa.
  - Implementar mais ou menos operações.
- Algumas operações típicas:
  - Soma.
  - Subtração.
  - Operação de E bit a bit
  - Operação de **Ou-Exclusivo** bit a bit.
  - ...
- Também contém um conjunto de linhas de controle.
  - Controlam operação a ser realizada.

ALU: Operandos e Saídas.

- A ALU tem dois operandos e uma saída para o resultado.
- Tanto os operandos, quanto o resultado possuem o mesmo número de bits.
  - Tipicamente, do tamanho dos registradores de propósito geral.
  - Igual ao tamanho da palavra.
- Saída pode, ainda, conter algumas **flags**.
  - Bits isolados, indicando a ocorrência de algum evento.
  - Alguns exemplos comuns:
    - ★ Resultado é zero.
    - ★ Resultado é negativo.
    - ★ Houve *overflow*.

ALU: Por Dentro

- Projetar uma ALU está fora do escopo desta aula.
- Mas alguns detalhes:
  - Linhas de controle da ALU geralmente são interpretadas em conjunto, como um código.
  - Decodificador é usado para interpretar código em linhas de controle internas.
    - ★ Que modificam o comportamento do componente.
  - Normalmente, a ALU realiza várias operações em paralelo.
    - ★ Operação certa é selecionada na saída.

Exercício

- Suponha uma ALU que realiza as seguintes operações:
  - Soma.
  - Subtração.
  - Operação lógica *E* bit a bit.
  - Operação lógica *Ou Exclusivo* bit a bit.
  - Multiplicação.
- Com base apenas nestas informações, qual é o número **mínimo** de linhas de controle necessárias a esta ALU?