Arquitetura de Computadores

Noções básicas de arquitetura de computadores - Aula 2

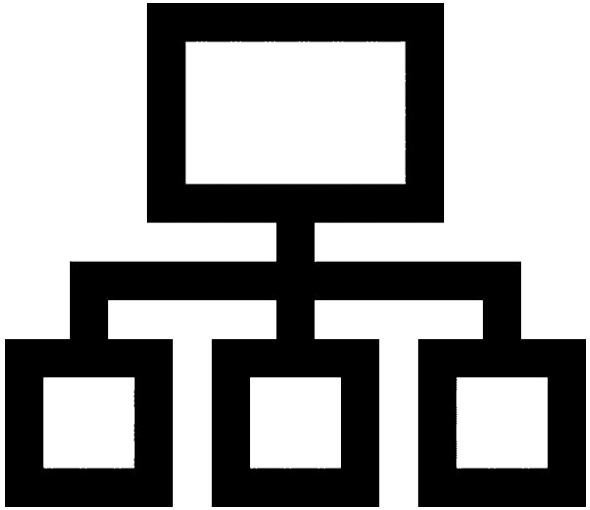
UNIVERSIDADE FEDERAL DE SERGIPE

Professor
Calebe Conceição

Aprofunde-se no livro texto!

Leia o capítulo 11 do Livro do Uyemura.

Na última aula



- Aprofundamos a noção de projeto hierárquico
  - Especialmente na descrição de funcionamento de um computador
- Vimos como a noção hierárquica é coberta nas HDLs
  - Especialmente na linguagem Verilog
- Entendemos as principais etapas de execução de um programa
  - Discutimos os tipos de instruções e quais blocos funcionais que são ativados ao executá-las

Nesta aula

- Veremos em mais detalhes o funcionamento de um processador. O esquemático geral será nosso guia de referência!
  - Suas características gerais
  - Estudo inicial da Unidade Central de Processamento
  - Componentes da Via de Dados (*Datapath*)
  - Instruções e Via de Dados
  - Unidade de Controle
  - Arquiteturas RISC e CISC

Características Gerais

Principais componentes de um computador

Circuitos de Entrada: Fornece dados para o computador via Teclado, mouse, unidades de disco, CD-ROM, scanners...

Circuitos de Saída: Codificam os resultados binários das operações para apresentá-los via monitor, unidade de disco no modo escrita

Memória: Armazena programas, dados e tudo mais necessário, como o Sistema Operacional.

Via de dados: Representa o caminho que o dado percorre durante os eventos de processamento.

Unidade de Controle: Garante que o dado seja enviado para conjunto correto de circuitos, nas mais diversas operações suportadas na via de dados.

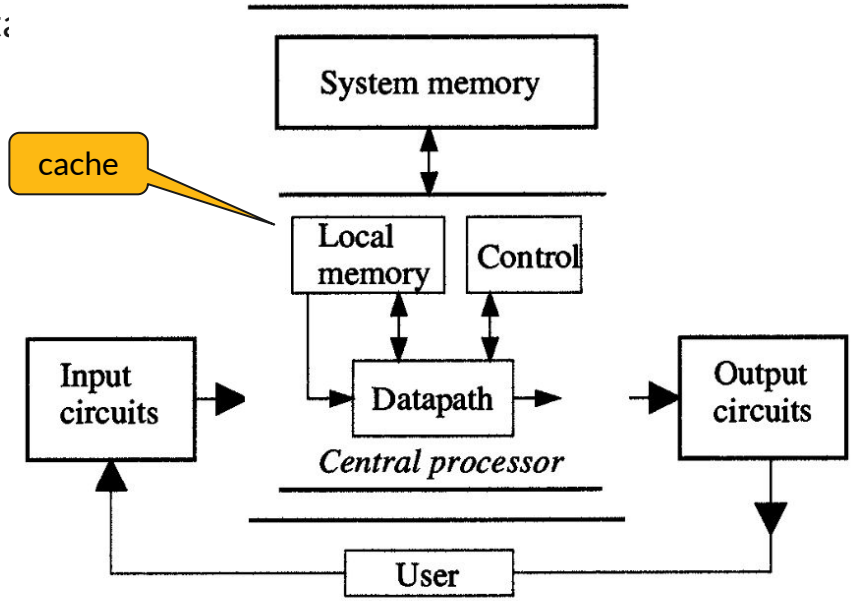


Figure 11.1 Major components of a computer

O que o computador pode fazer?

Podemos simplificar as operações que o computador realiza em duas principais:

- Movimentação de dados
- Execução de operações binárias
    - Funções lógicas: NOT, AND, XNOR, ... – usadas nas decisões
    - Funções aritméticas: soma, subtração, multiplicação...

Instrução → Cada operação que o computador realiza

Conjunto de Instruções → é o conjunto de todas as instruções suportadas (ISA - *Instruction Set Architecture*)

Modelo de von Neumann

Composto por dois blocos principais:

- **Memória** → Armazena programa (lista de instruções) e dados
- **Unidade Central de Processamento** →via de dados + unidade de controle (+ registrador de instrução).

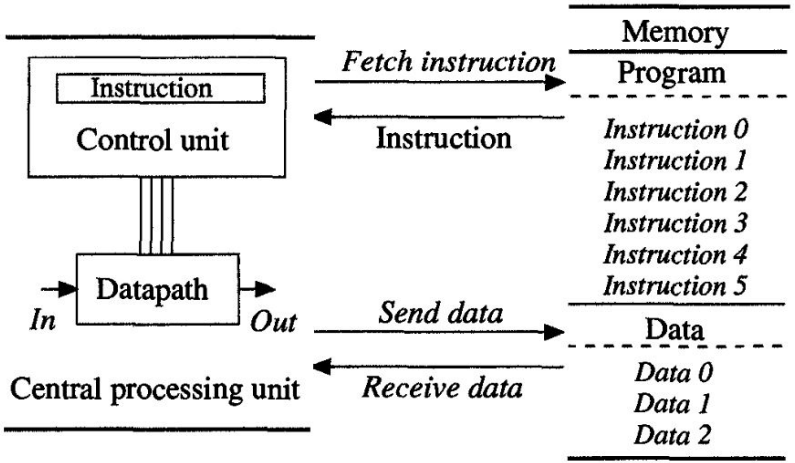


Figure 11.2 The von Neumann model of a programmable computer

Ciclos do Modelo von Neumann

O modelo von Neumann de um computador é baseado na repetição de um procedimento de quatro ciclos para execução do programa,. São eles:

1. **Busca de instrução** → Busca na memória a instrução que será armazenada na Unidade de Controle;
2. **Decodificação de instrução** →Processo de interpretar a instrução para determinar o que precisa ser feito dentro da CPU; A unidade de controle então "informa" à via de dados o que fazer;
3. **Execução da instrução** → A via de dados executa a operação, acessando as entradas, calculando, e apresentando os resultados;
4. **Armazenamento** →o resultado das operações é guardado novamente na memória.

$$tInst = t_{BI} + t_{DI} + t_{EX} + t_{A}$$

Programação

Programa: lista ordenada de comandos que ditam uma sequência de operações necessárias para efetuar uma determinada tarefa.

Linguagem de programação: permite descrever o programa em uma sintaxe específica, com uma semântica atrelada.

O computador só "entende" linguagem de máquina (0s e 1s).

Programas especiais, como o compilador, fazem a tradução da linguagem de programação para a linguagem de máquina.

A linguagem Assembly possui uma correspondência direta com cada comando binário em linguagem de máquina, facilitando seu entendimento.

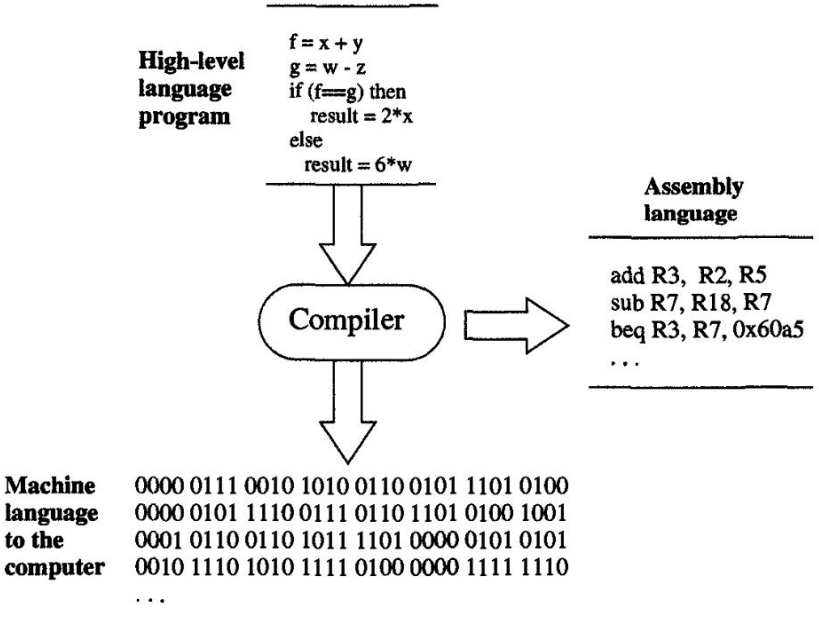


Figure 11.3 Levels of programming languages

Registradores do Computador

Registrador → é um conjunto unitário de célula de memória. São agrupados para armazenar toda uma palavra binária.

Atente-se à notação:

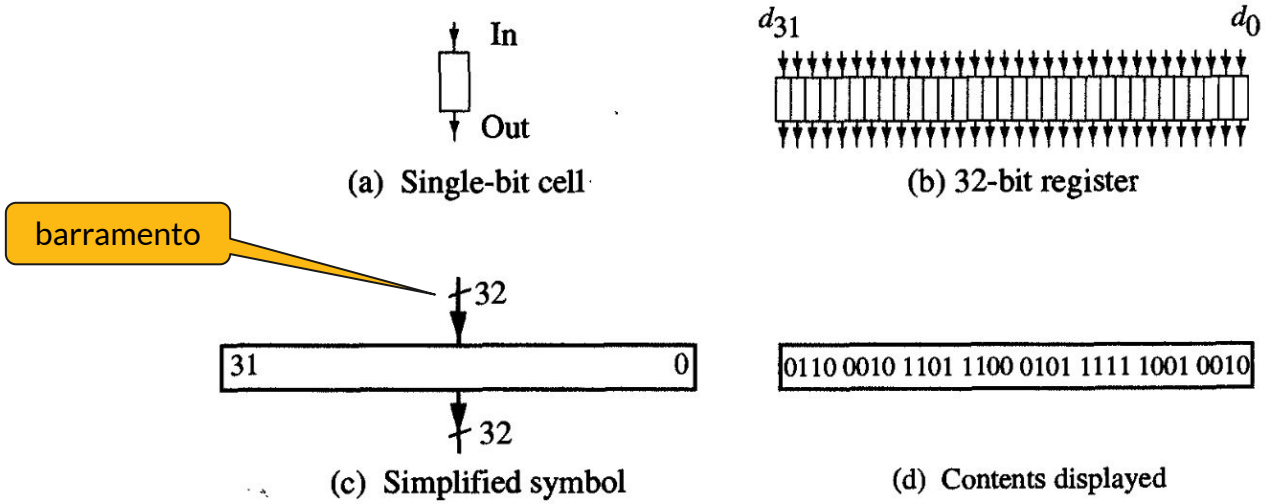


Figure 11.4 Schematic symbols used to represent registers

Estudo inicial da Unidade Central de Processamento

Circuito de busca de informação

Considere a sequência de instruções de um programa armazenado na memória, conforme segue

Address	Binary Instruction	Order
0400	01101100 11111010 11110000 11110000	*Inst* 0
0404	11000101 10110101 00001111 11110000	*Inst* 1
0408	01000110 10011111 10101010 10101010	*Inst* 2
0412	10010011 01101110 00110011 00110011	*Inst* 3
0416	10101001 01000101 11100011 11100011	*Inst* 4
0420	10001000 10001101 10011001 10011001	*Inst* 5
0424	11100010 10101001 11100010 10101010	*Inst* 6
0428	00100111 01101010 00110010 01011000	*Inst* 7
0432	10010010 11010011 10010011 01001001	*Inst* 8

Figure 11.5 A program sequence stored in memory

Circuito de busca de informação

O **Registrador de Instrução (IR)** guarda a palavra binária da instrução corrente
O **Contador de Programa (PC)** guarda o endereço de memória da próxima instrução que será buscada, controlando assim o fluxo de execução.

Para obter a próxima instrução, calculamos:

PC ← PC + X, (*onde X normalmente é 4*)

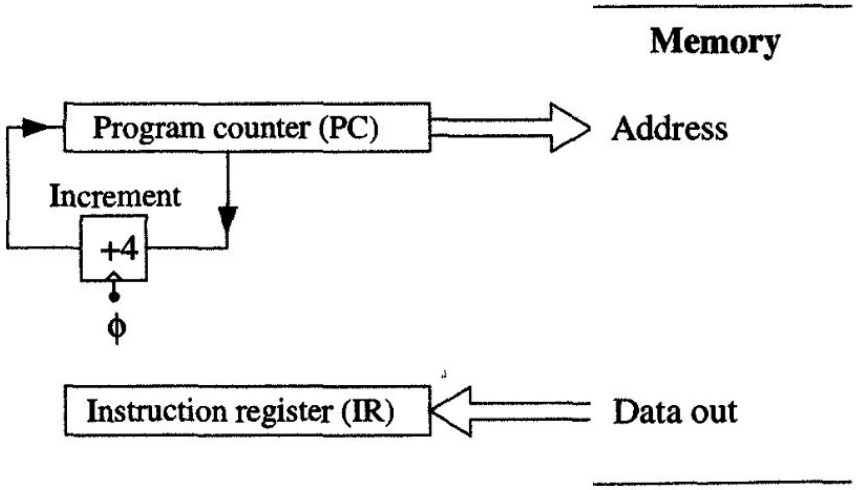
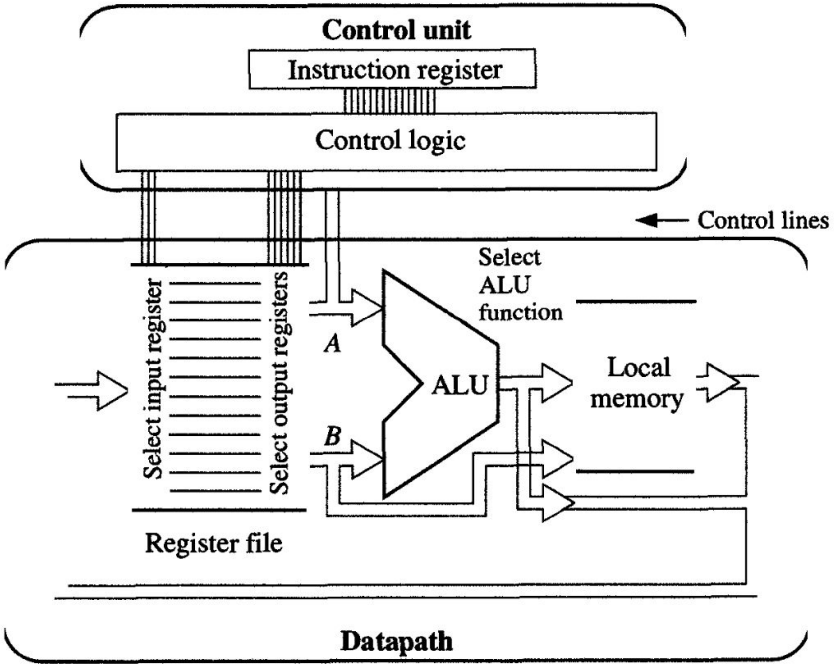


Figure 11.6 Operation of the instruction fetch (IF) network

Conceito de via de dados

A Unidade de Controle **acessa diretamente** o Registrador de Instrução.
Os bits da instrução determinam quais **sinais de controle** a Lógica de Controle irá passar para as **linhas de controle**, determinando como se dará o fluxo na Via de Dados.



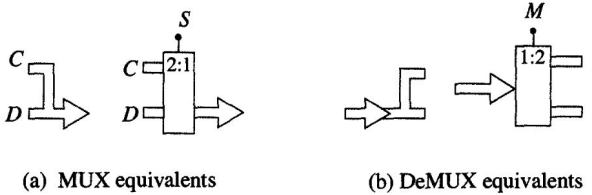
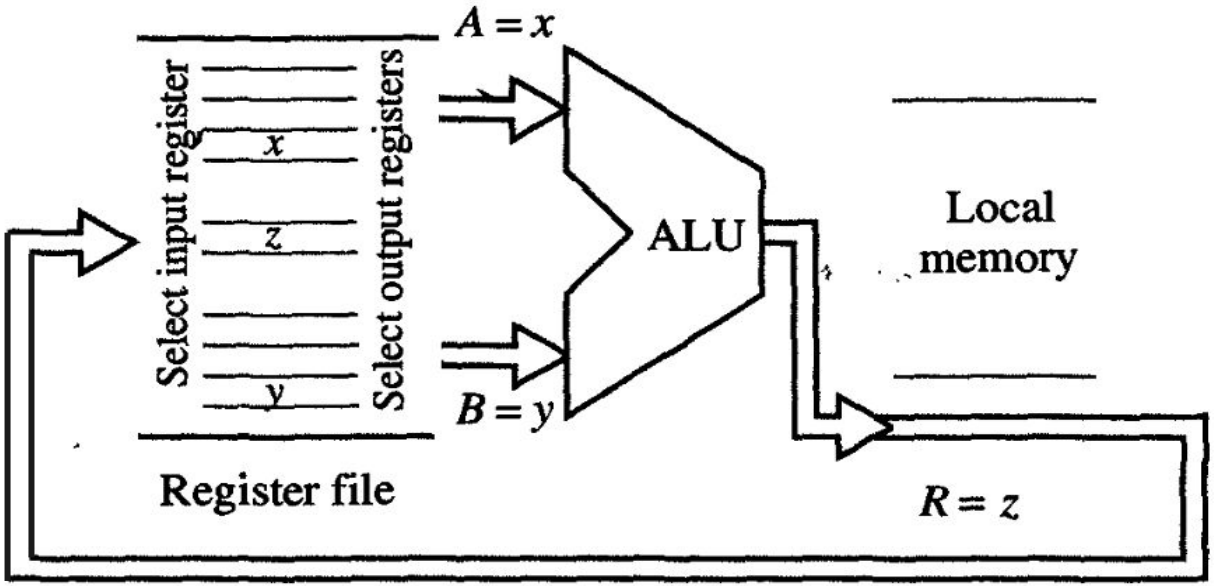


Figure 11.7 The central processor consists of the datapath and the control unit

Operações da via de dados

Operações de Registrador para Registrador



$$R = A + B$$
$$R = A - B$$
$$R = A \cdot B$$
$$R = \overline{A}$$

Figure 11.9 Datapath for a register-to-register operation where data originates from the registers and the result is stored back into the registers

Operações da via de dados

Carregar (Load)

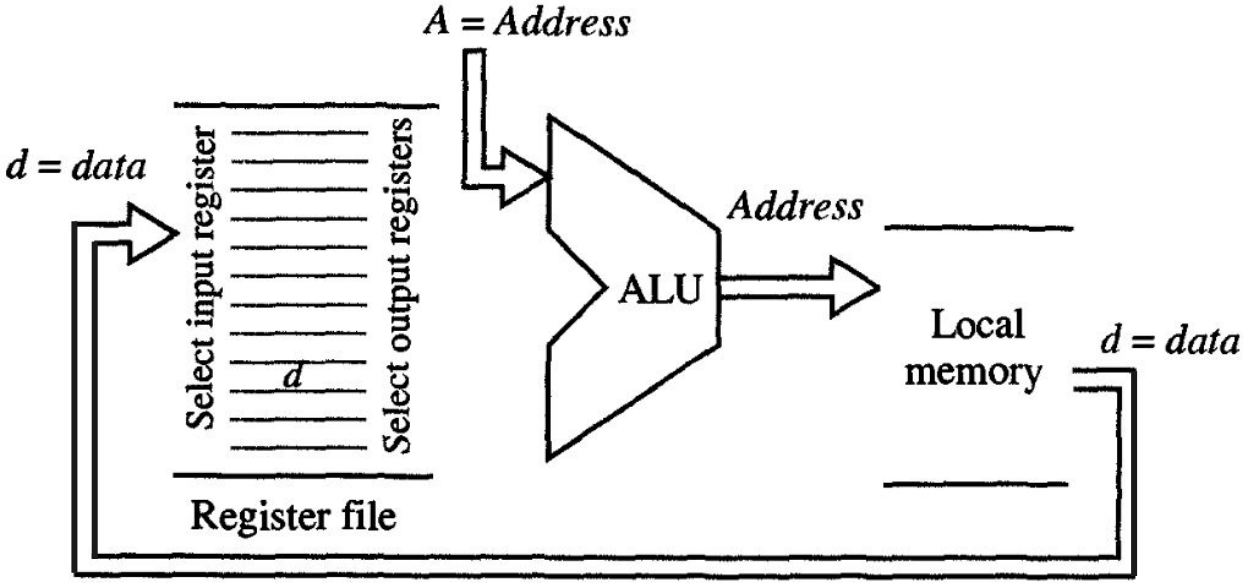


Figure 11.10 The load word instruction allows us to move a data word from the memory to a particular register

Operações da via de dados

Armazenar (Store)

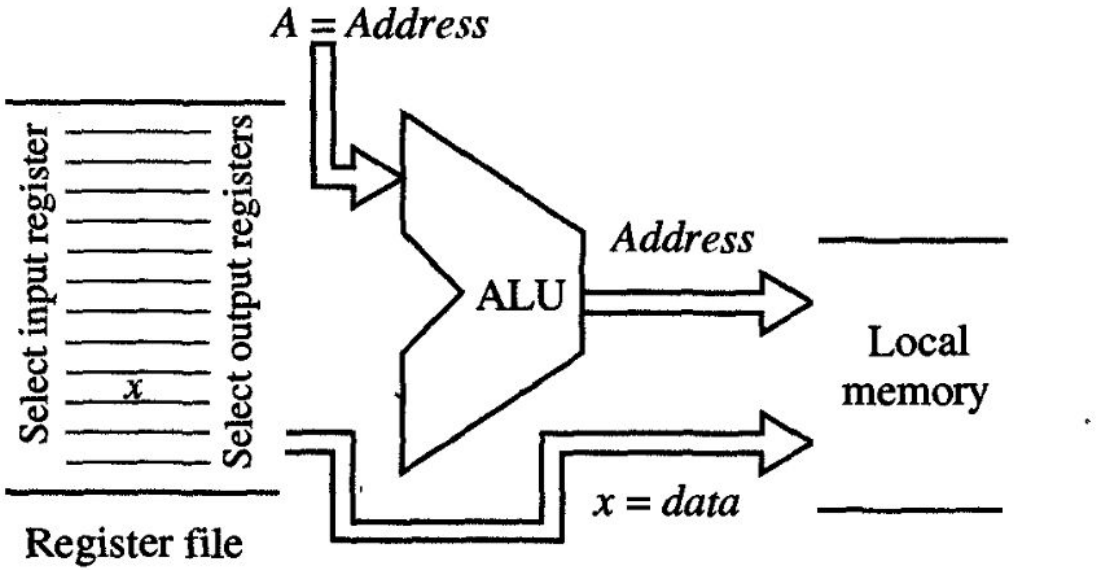


Figure 11.11 The store word instruction allows us to move a data word from a register and store it in the memory

Componentes da Via de Dados (*Datapath*)

Arquivo de Registradores (ou Banco de

No exemplo, um banco de registradores com 32 registradores de 32 bits cada, e barramentos de endereço de 5 bits.

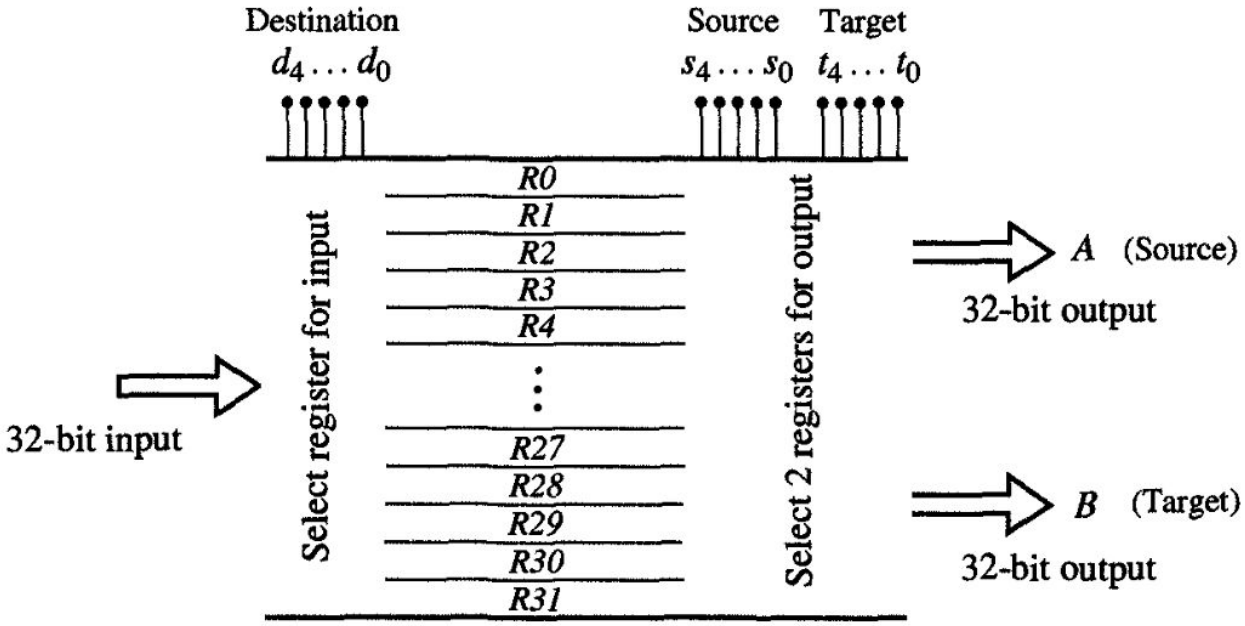


Figure 11.12 Structure of the register file

Arquivo de Registradores (ou Banco de

Detalhando um pouco mais a estrutura…

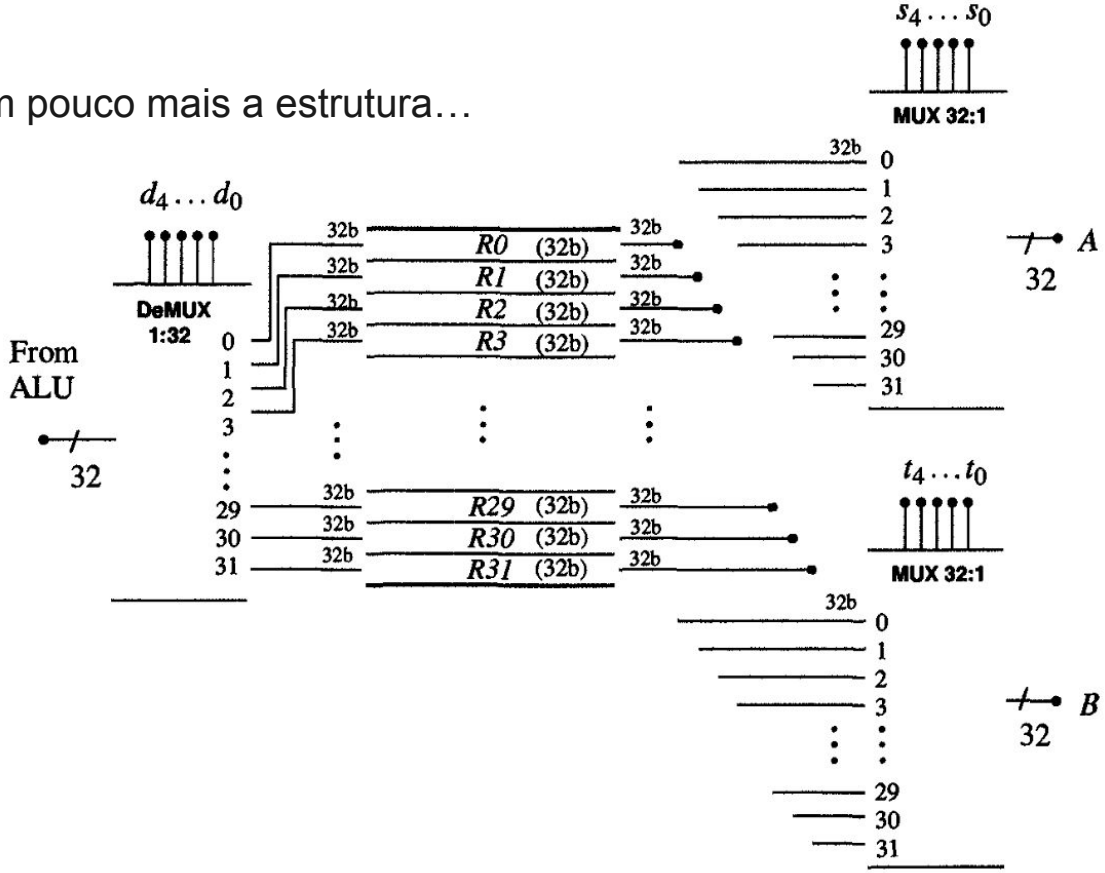


Figure 11.13 A unit-level view of the register file structure

Unidade lógica e Aritmética

Leiam com atenção toda a seção 11.3.2.

Seção responsável por executar as funções aritméticas requeridas (ADD, SUB, etc.) e todas as operações lógicas (NOT, AND, XOR, etc.)
Operações variam de processador para processador, mas muitas operações são comuns a todos.
A ULA é controlada pela palavra de **seleção de função** f.
No exemplo, assumimos que todos os operandos têm 32 bits.

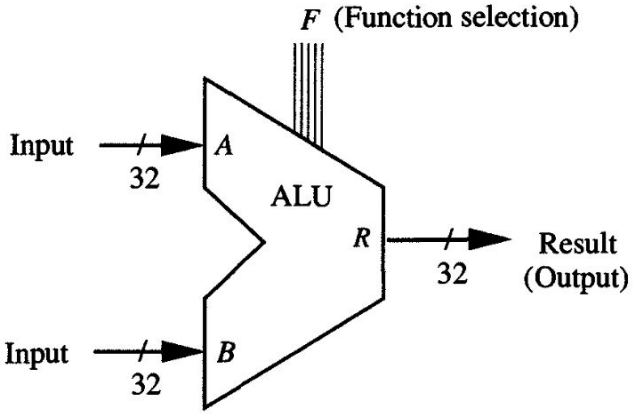


Figure 11.15 Symbol for the arithmetic and logic unit (ALU)

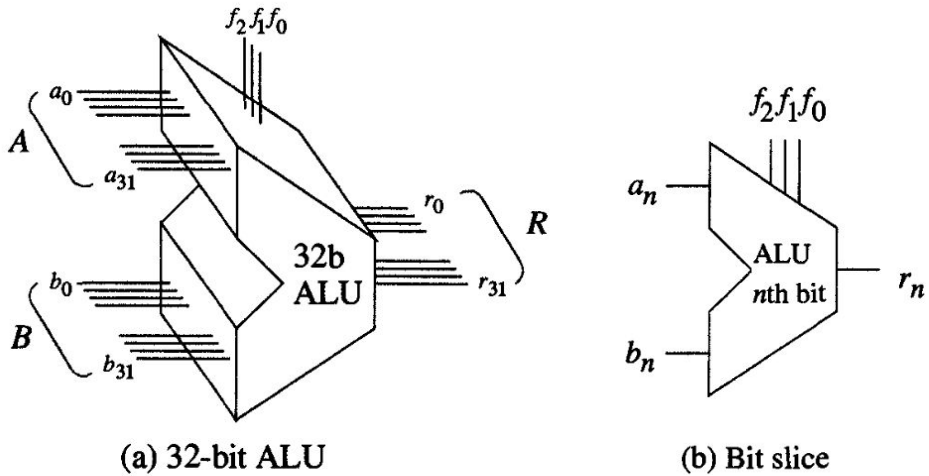


Figure 11.17 Concept of a bit slice of the ALU

UNIVERSIDADE FEDERAL DE SERGIPE

Memória Local

Memória utilizada para operações rápidas de *load* e *store*, genericamente chamada de **cache**.

É conectada entre o caminho do fluxo de dados e a memória principal.

Operacionalmente idêntica a uma matriz genérica de memória escrita/leitura. O valor do sinal R/W especifica qual operação será executada.

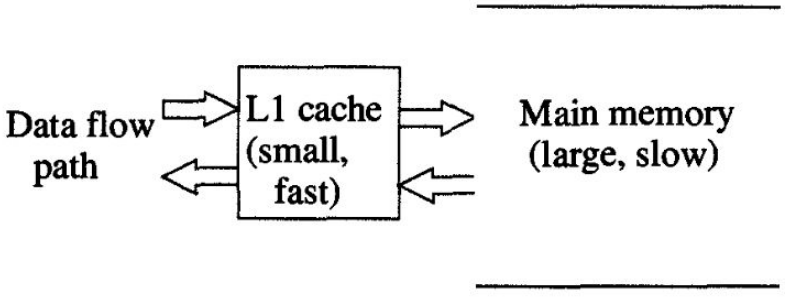


Figure 11.20 Concept of local memory

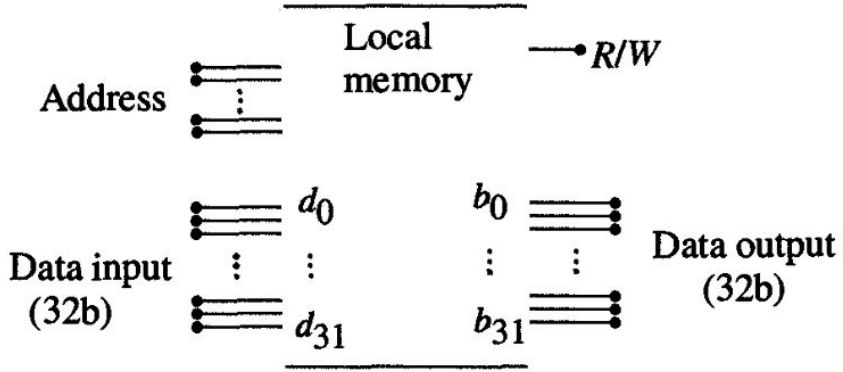


Figure 11.21 Operational model for the local memory

Instruções e Via de Dados (*Datapath*)

Continuaremos na próxima aula

Arquiteturas RISC e CISC

Continuaremos na próxima aula

Referências

UYEMURA, J. P., Sistemas Digitais: uma abordagem integrada. Editora Thompson-Pioneira, Brasil, 2008 (Cap. 11)

Hora-Trabalho de Hoje

Leia o capítulo 11 do livro do Uyemura, para reforço de aprendizagem.

Trabalho 1

Leia a seção 11.3.2 e implemente em Verilog um módulo que modela o funcionamento de uma Unidade Lógica e Aritmética (ULA).

Requisitos:

Implemente o testbench e entregue junto com o módulo.

Critérios:

Corretude; Implementações que priorizam a descrição estrutural serão melhor avaliadas. Completude da rotina de testbench, com saídas adequadas para acompanhamento da execução.

Dúvidas?

Na próxima aula...

Continuação; Introdução ao assembly do processador MIPS;

Não falte! 😉

Obrigado pela atenção

UNIVERSIDADE
FEDERAL DE
SERGIPE