UNIOESTE – Universidade Estadual do Oeste do Paraná
Departamento de Engenharias e Ciências Exatas
Campus de Foz do Iguaçu

Microprogramação - introdução

Profs.: Newton Spolaôr e Fabiana Frata

Apoio: Camile Bordini

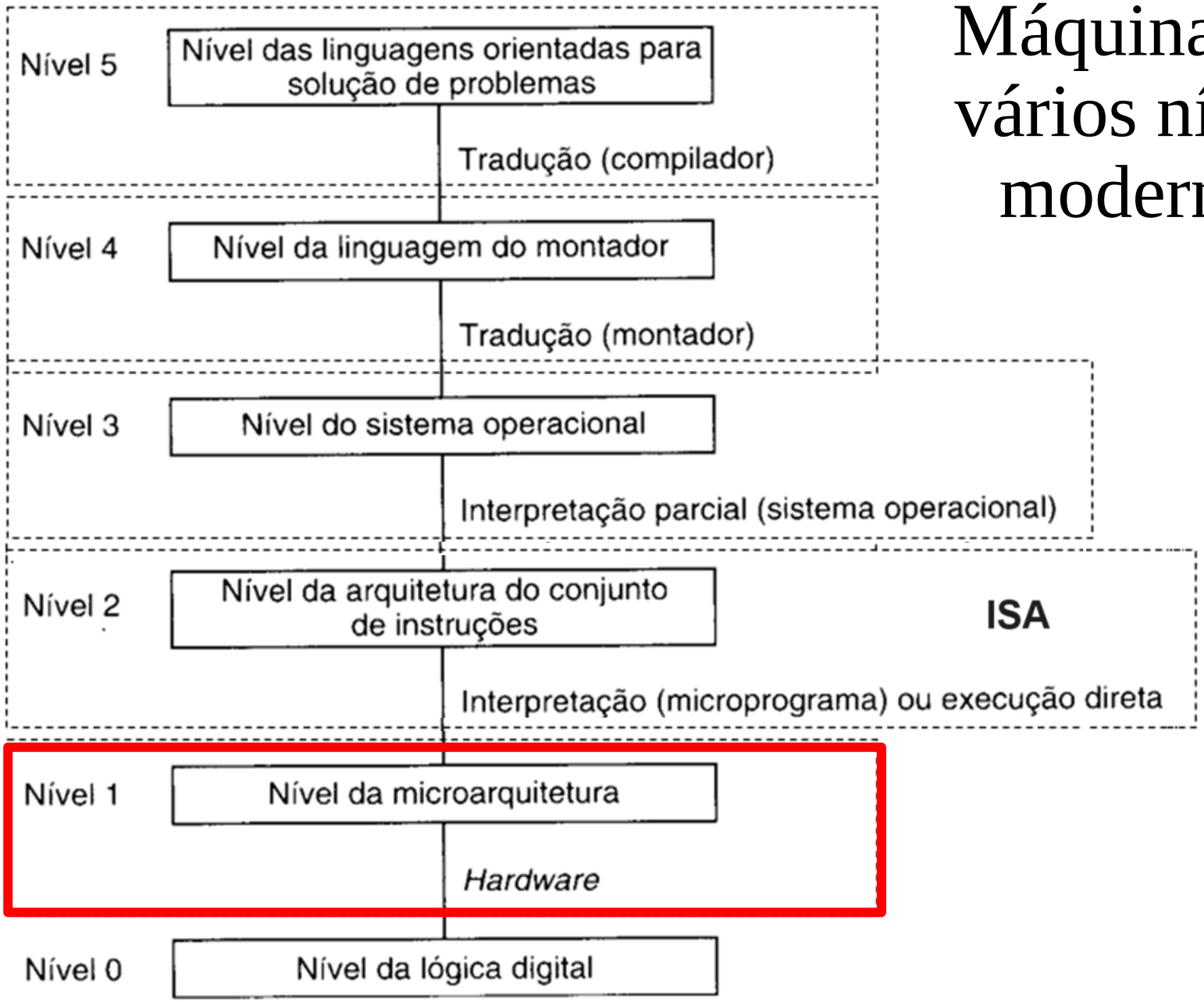
Adaptações de: R. Siegfried (U. Adelphi), C. Merimovich (Tel-Aviv Acad. Coll.),

M. Malek (Humboldt-Universität Zu Berlin)

Relembrando…

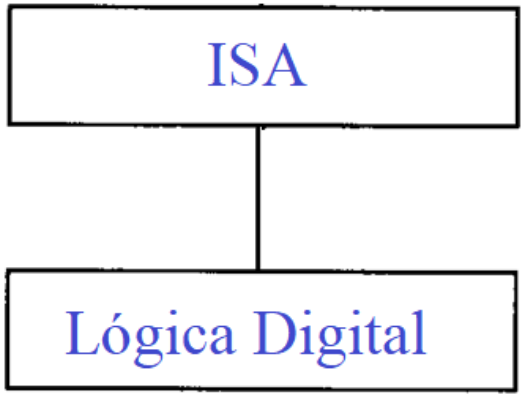
A invenção da Microprogramação

Máquinas de vários níveis modernas



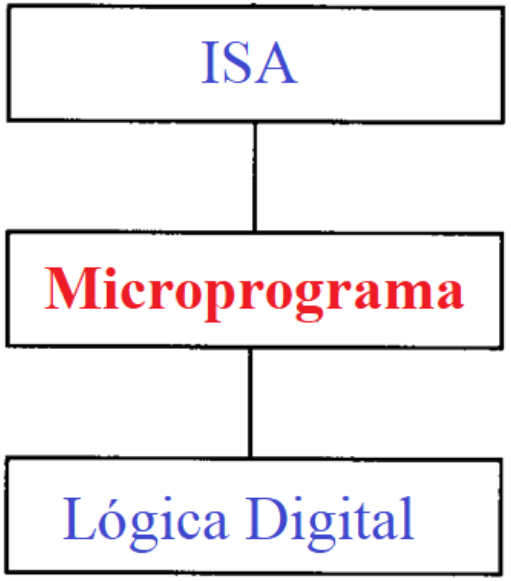
A invenção da microprogramação

- Os primeiros computadores digitais (~1940) tinham somente 2 níveis:
  - **Nível ISA**, onde toda a programação era feita
  - **Nível da lógica digital**, onde os programas eram executados (circuitos complicados e pouco confiáveis)



A invenção da microprogramação

- Em **1951** (Maurice Wilkes): ideia de construir um computador com **3 níveis** para simplificar o hardware:
  - **Necessário um interpretador (microprograma)** para executar programas no nível ISA



A invenção da microprogramação

Sir Maurice Wilkes e o computador EDSAC, um dos primeiros computadores que seguem o conceito de **programa armazenado (em memória)**

Fonte: http://alchetron.com/Maurice-Wilkes-1022304-W#-

A invenção da microprogramação

•Após visitar um computador americano *hardwired* nos anos 50, ele imaginou uma solução baseada em microprogramação;

"... Acho que somente quando voltei a Cambridge percebi que a solução era tornar a unidade de controle em um computador em miniatura ..." [Wilkes]

•Essa ideia estava à frente de seu tempo, pois para um bom funcionamento, a microprogramação dependia de uma memória rápida, indisponível naquela época;

•Logo, o controle *hardwired* era ainda a melhor opção vivável naquela época.

A invenção da microprogramação

•A IBM valorizou a microprogramação nos anos 60 com a família de máquinas IBM 360;

•Para viabilizar essa ideia, ela incorporou tecnologia de memória dentro da empresa;

•Algumas tecnologias que se popularizaram a partir daquela época incluem ROM e RAM;

•Com a oferta de memórias mais rápidas e a motivação de vantagens da microprogramação, como portabilidade para diferentes máquinas, houve uma popularização dessa ideia nos anos 60 e 70.

A invenção da microprogramação

- Portanto: **hardware passa a executar somente microprogramas** (*que possuem conjunto de instruções bastante limitado*), ao invés de programas no nível ISA (*com muito mais instruções*!)

  - **Necessário bem menos circuitos eletrônicos!**

menos válvulas = maior confiabilidade

A invenção da microprogramação

- Algumas dessas máquinas de 3 níveis foram construídas ao longo dos **anos 1950**

- **Década de 1960**: quantidade de máquinas produzidas dessa forma aumentou bastante
  - 1964: Introdução da família de processadores System/360 da IBM
    - Primeira arquitetura comercial de computadores que usava **microprogramação** (**interpretação** de instruções de arquitetura para execução em hardware).

A invenção da microprogramação

- **Década de 1970**: tornou-se prática comum ter um nível ISA interpretado por um microprograma, em vez de ser executado diretamente por circuitos eletrônicos
  - Utilização de **conjunto de instruções complexas**
  - Cuja implementação era muito simples devido ao emprego do **interpretador**
  - Quase ninguém pensava em projetar máquinas “mais simples”

A invenção da microprogramação

- Projetistas logo observaram que poderiam acrescentar novas instruções ao conjunto de instruções do processador simplesmente **expandindo o microprograma**!

  - **Explosão no conjunto de instruções**!
  - Projetistas disputando na produção de conjuntos maiores e melhores!

A invenção da microprogramação

- Foram adicionadas **várias outras instruções** ao conjunto de instruções por meio do microprograma:
  - para multiplicação e divisão de números inteiros
  - para aritmética em ponto flutuante
  - para chamada e retorno de procedimentos
  - para acelerar a execução de loops
  - para manipular *strings* de caracteres

0011111	01011	01010	001	01000	1100111
imm[12:6]	rs2	rs1	funct3	imm[5:1]	opcode

A invenção da microprogramação

- Consequências da "***era de ouro da microprogramação***" entre décadas de 60 e 70?

  - **Os microprogramas cresceram muito e tornaram-se lentos!**

- Pesquisadores começaram a estudar os **efeitos de projetar máquinas SEM usar microprogramação!**

E aí surge a filosofia RISC que já conhecemos.

RISC – Reduced Instruction Set Computer

- Exemplos de arquiteturas **RISC**
  - Alpha, da DEC
  - MIPS
    - Arquitetura simples e didática que serve como base para alguns conceitos da disciplina
    - Desenvolvida sob coordenação de J. Hennessy
  - ARM: base para smartphones e sistemas embarcados atuais
  - RISC V: design de microprocessador de código aberto

CISC – Complex Instruction Set Computers

- Modelo de arquiteturas que...
  - Contém instruções complexas, as quais realizam uma sequência de operações em baixo nível
  - Exibem um conjunto de instruções grande

- Exemplo : add #1004, BX, #1000
  - Carregar dados da memória
  - Executar operações na ULA
  - Salvar resultado na memória

CISC – Complex Instruction Set Computers

- Exemplos de arquiteturas **CISC**
  - Grandes mainframes IBM
    - System/360: pioneiro em microprogramação
  - Intel
    - 8088: primeiros processadores Intel, base para alguns conceitos da disciplina
    - Pentium
  - Família VAX, da DEC (em praticamente em todas a universidades da época)

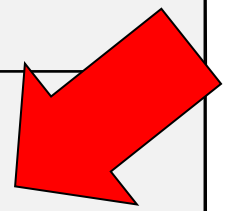
RISC vs CISC

RISC vs CISC

RISC	CISC
As instruções levam em média 1 ciclo de clock para serem executadas (80% delas)	Instruções complexas levando vários ciclos de clocks
Apenas `load` e `store` referenciam a memória; poucos modos de endereçamentos	Qualquer instrução acessa a memória
Instruções com formato fixo	Instruções com formato variado
Poucas instruções	Muitas instruções
Instruções executadas pelo hardware	Instruções interpretadas pelo microprograma
Complexidade está no compilador	Complexidade esta no microprograma
Altamente pipelined	Pouco pipelined

RISC vs CISC

RISC	CISC
As instruções levam em média 1 ciclo de clock para serem executadas (80% delas)	Instruções complexas levando vários ciclos de clocks
Apenas `load` e `store` referenciam a memória; poucos modos de endereçamentos	Qualquer instrução acessa a memória
Instruções com formato fixo	Instruções com formato variado
Poucas instruções	Muitas instruções
Instruções executadas pelo hardware	**Instruções interpretadas pelo microprograma**
Complexidade está no compilador	Complexidade esta no microprograma
Altamente pipelined	Pouco pipelined



Unidade de controle pode variar!

Unidade de controle

(instruções executadas pelo hardware – RISC)

- A **unidade de controle** é vista como um **grande circuito**;
- Entradas incluem a **instrução** a ser executada e o **clock**;
- As saídas são ilustradas por **sinais de controle**;
- A representação da unidade pode ser realizada via **máquina de estados**;
- A operação pode ser **rápida**;
- Alguns desses conceitos já foram abordados em aulas anteriores.

Unidade de controle

(instruções interpretadas pelo microprograma – CISC)

- A unidade de controle passa a conter **dispositivos como memória**;
- Dentro dela há a execução de ***microinstruções***:
  - Uma microinstrução pode ativar sinais de controle;
  - Microinstruções atuam em um nível mais baixo do que as instruções do conjunto RISC-V;
  - Microinstruções compõem um ***microprograma***, (assim como instruções de máquina compõem um código Assembly).
- Operação mais lenta, mas mais suporte para reprojetar a unidade;

Microprogramação

(introdução)

- Para o subconjunto de instruções RISC-V visto até agora, uma máquina de estados finitos é suficiente

- Contudo, vale lembrar que o conjunto completo de instruções RISC-V contém **mais de 100 instruções**

- Esse fator, sozinho, já aumenta a complexidade do controle de instruções (mais estados, mais sinais de controle, mais conflitos...).

Microprogramação

(introdução)

•E quando lidamos com um conjunto de instruções ainda maior, como o IA-32 (CISC)?

Microprogramação

(introdução)

•E quando lidamos com um conjunto de instruções ainda maior, como o IA-32 (CISC)?

- Várias centenas de instruções de classes muito distintas;
- **Unidade de controle** teria milhares de estados com centenas de sequências diferentes;
- Dificuldade (ou impossibilidade) de obter representação gráfica dessa unidade.

E agora?

Microprogramação

(introdução)

- Uma solução surge quando usamos **ideias de programação** para obter um controle mais simples
- Para tanto, considere o conjunto de sinais de controle que precisam ser ativados como uma **instrução a ser executada na via de dados**
- Essa instrução de baixo nível é denominada, de agora em diante, **microinstrução**, (para evitar confusão com as instruções RISC-V)
- Logo, executar uma **microinstrução** levaria à ativação dos sinais que ela especifica.

Microprogramação

(introdução)

- Assim como precisamos definir o próximo estado na máquina de estados da RISC-V, também precisamos saber **qual é a próxima microinstrução a ser executada**

- Assim como ocorre em programas de alto nível, por padrão as **microinstruções** são executadas **em sequência**

- Quando necessário, um **desvio explícito** do fluxo de execução é indicado em **programas** e **microprogramas**.

Microprogramação

(introdução)

- Como mencionado, um **microprograma** é uma representação de um grupo de **microinstruções**

- Assim como instruções de máquina, as **microinstruções** lidam com campos como registradores e valores imediatos

- O **microprograma** pode representar valores ativados nos sinais de controle simbolicamente, como veremos nas próximas aulas

Microprogramação

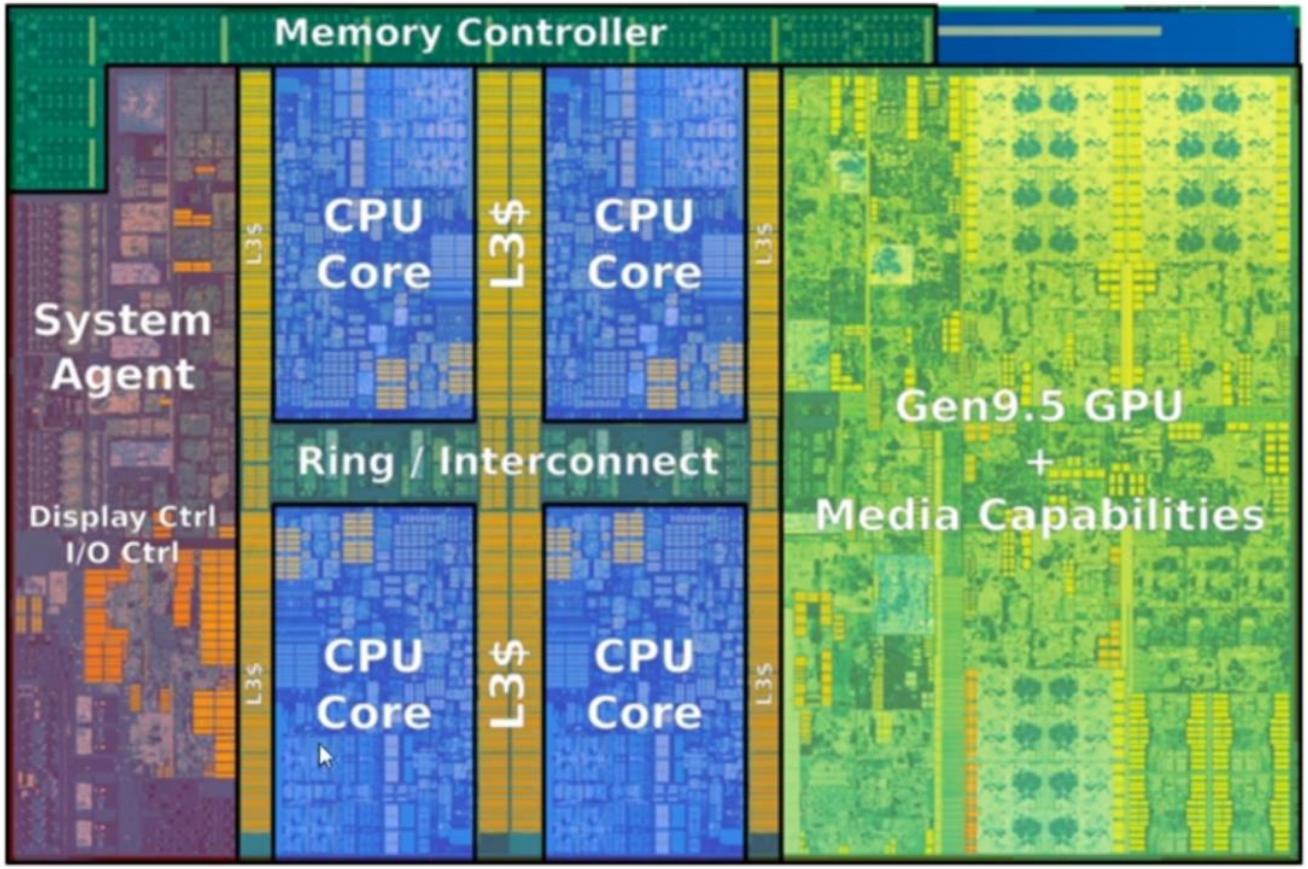
(vantagens e aplicações)

•Quais as vantagens da **microprogramação**?

- Sistematização do controle via programação
- Há a possibilidade de ganhar desempenho (ex.: executando algumas microinstruções em paralelo)
- Compatibilidade entre conjuntos de microinstruções em máquinas de uma mesma série (ex.: Intel 286 e 386)
- Emulação: interpretação de instruções de uma máquina em outra máquina.

Caminho de dados

Microarquitetura



- Representação do chip de um processador com microarquitetura Kaby Lake da Intel (7ª, 8ª geração)

Caminho de dados microprogramado

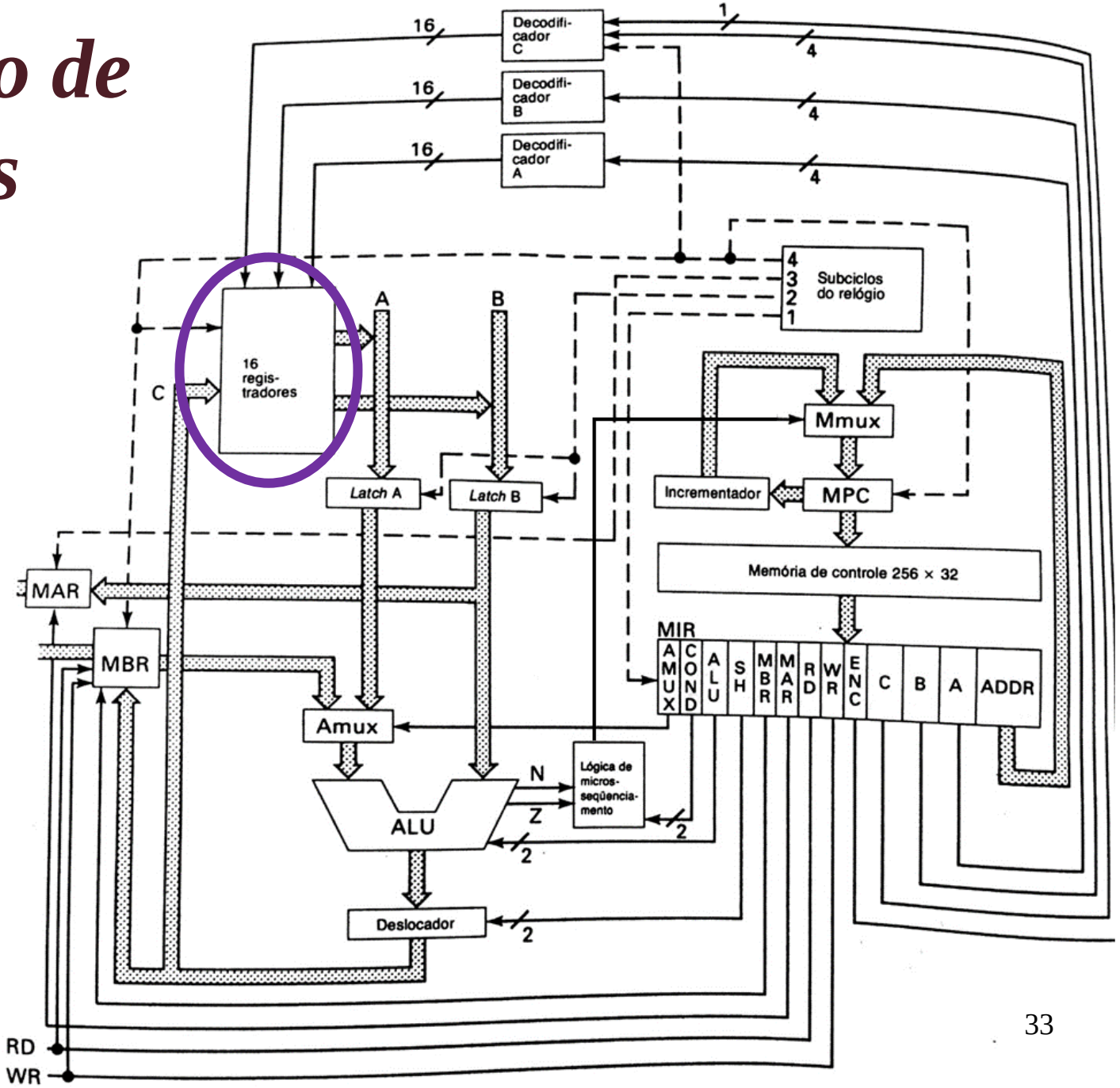
(exemplo de microarquitetura)

•Em geral, arquiteturas no nível de microprogramação (microarquiteturas) seguem a abordagem CISC e são complexas;

•Nesta disciplina é adotado um exemplo mais simples e didático de microarquitetura, **MIC**, criada por Tanenbaum:

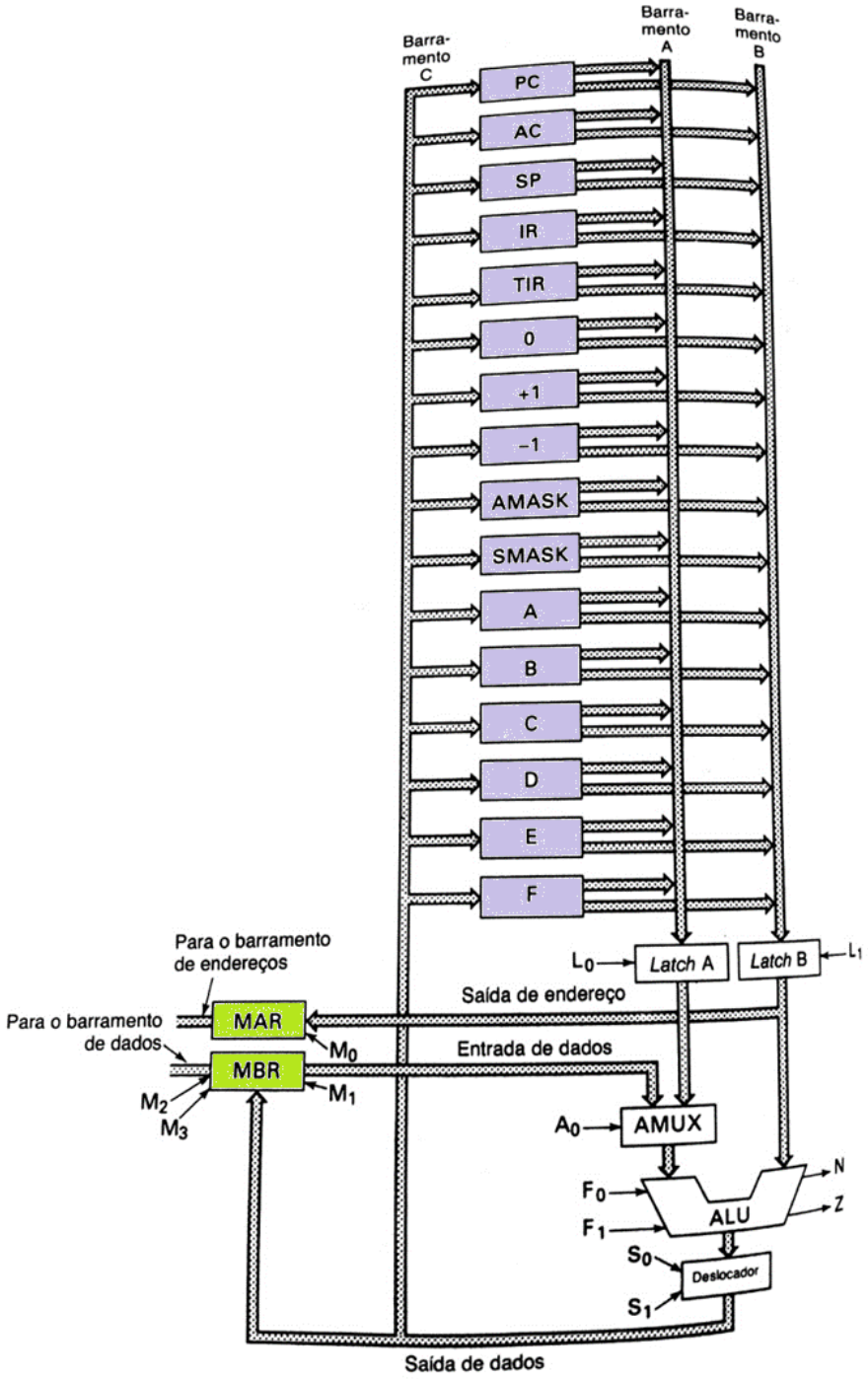
- Similar a algumas outras microarquiteturas de processadores antigos da família x86
- Possui **registradores de 16 bits**
- Baseada em caminhos de dados – porção do processador com ALU, suas entradas e saídas.

Caminho de dados



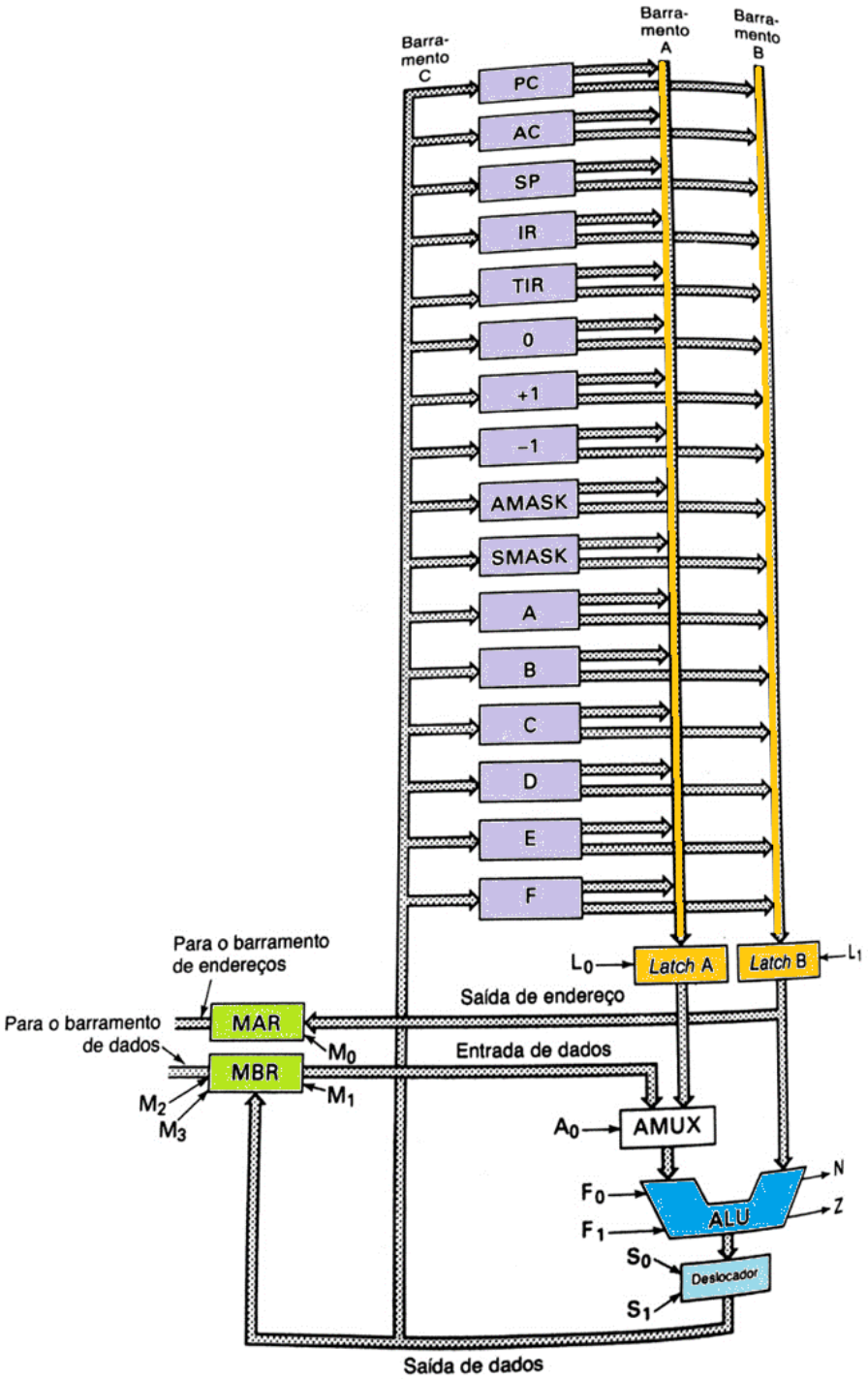
- **16 registradores de 16 bits**:

Endereço	Registrador	Comentários
0000	**PC**	contador de programa
0001	**AC**	Acumulador utilizado para movimentação de dados, cálculos e outros propósitos
0010	**SP**	Apontador de pilha
0011	**IR**	Registrador de instrução
0100	**TIR**	Registrador temporário de instrução
0101	**0**	Constante
0110	**1**	Constante
0111	**-1**	Constante
1000	**AMASK**	é a máscara de endereços 0000111111111111 e é utilizado para separar opcode de bits de endereço
1001	**SMASK**	é a máscara de pilha 0000 000011111111 e é utilizado nas instruções INSP e DESP para isolar a distância de 8 bits
1010	**A**	Registrador de propósito geral
1011	**B**	Registrador de propósito geral
1100	**C**	Registrador de propósito geral
1101	**D**	Registrador de propósito geral
1110	**E**	Registrador de propósito geral
1111	**F**	Registrador de propósito geral



Caminho de dados microprogramado

- Além dos **16 registradores**, existem outros a destacar:
  - **MAR**: armazena endereço de memória
    (registradores → memória)
  - **MBR**: armazena dado da ALU/deslocador ou da memória
    (memória ↔ registradores)
- Note os sinais de controle.



Caminho de dados microprogramado

- Pequeno exemplo de ciclo:
  1. Colocar valores nos barramentos **A** e **B**
  2. Armazená-los nos *latches*
  3. Processá-los na **ALU** e no **deslocador**
  4. Armazenar o resultado em **registrador** ou **MBR**.

Caminho de dados microprogramado

(registradores)

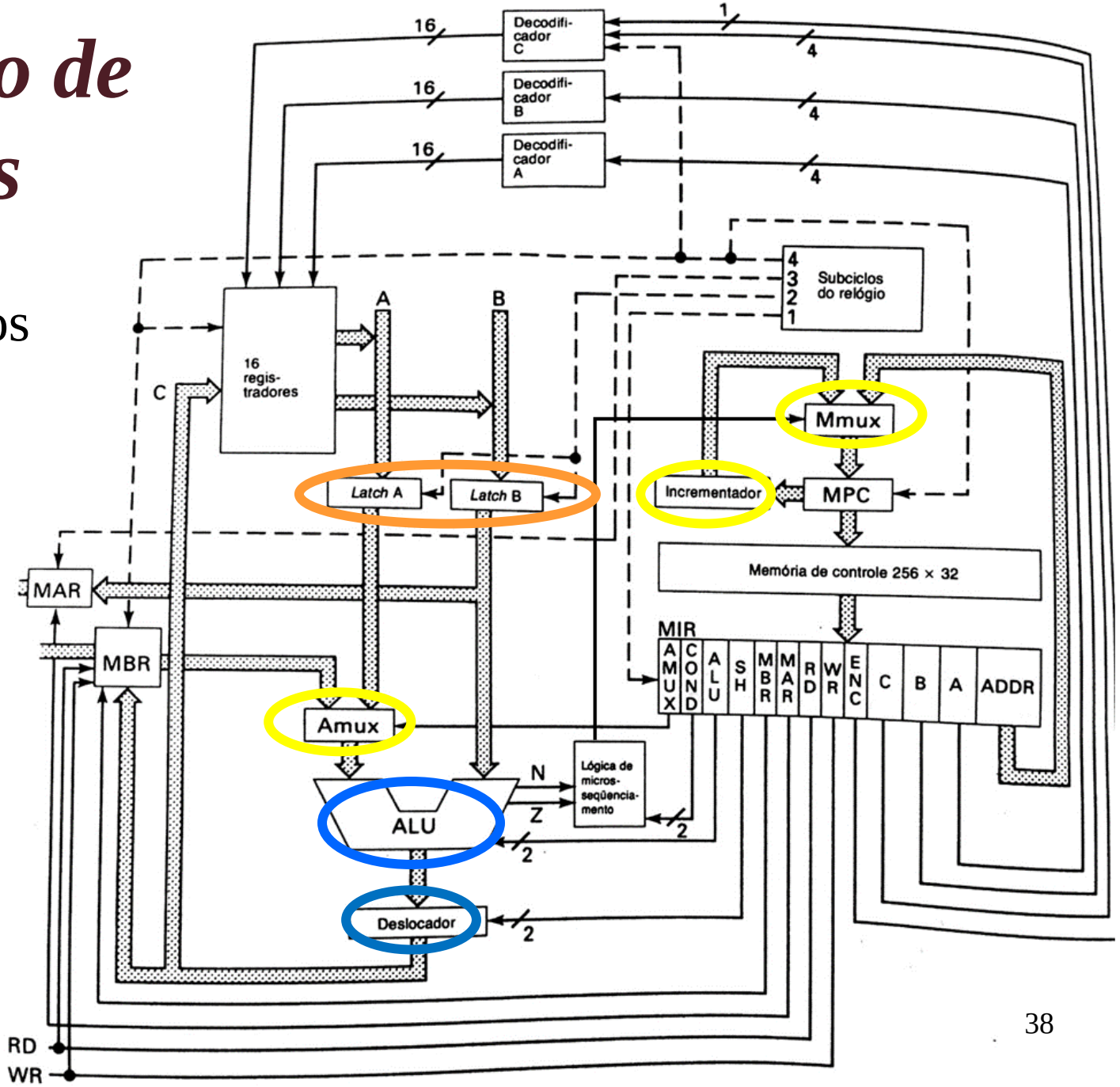
- **Registradores** como **PC**, **IR**, **A** e **B** também existem no caminho de dados **RISC-V**;

- Uma diferença importante do exemplo de arquitetura considerada consiste na maior oferta de constantes simples e máscaras

- Em compensação, o **banco de registradores** é menor em termos de **número de registradores** e de **quantidade de bits**.

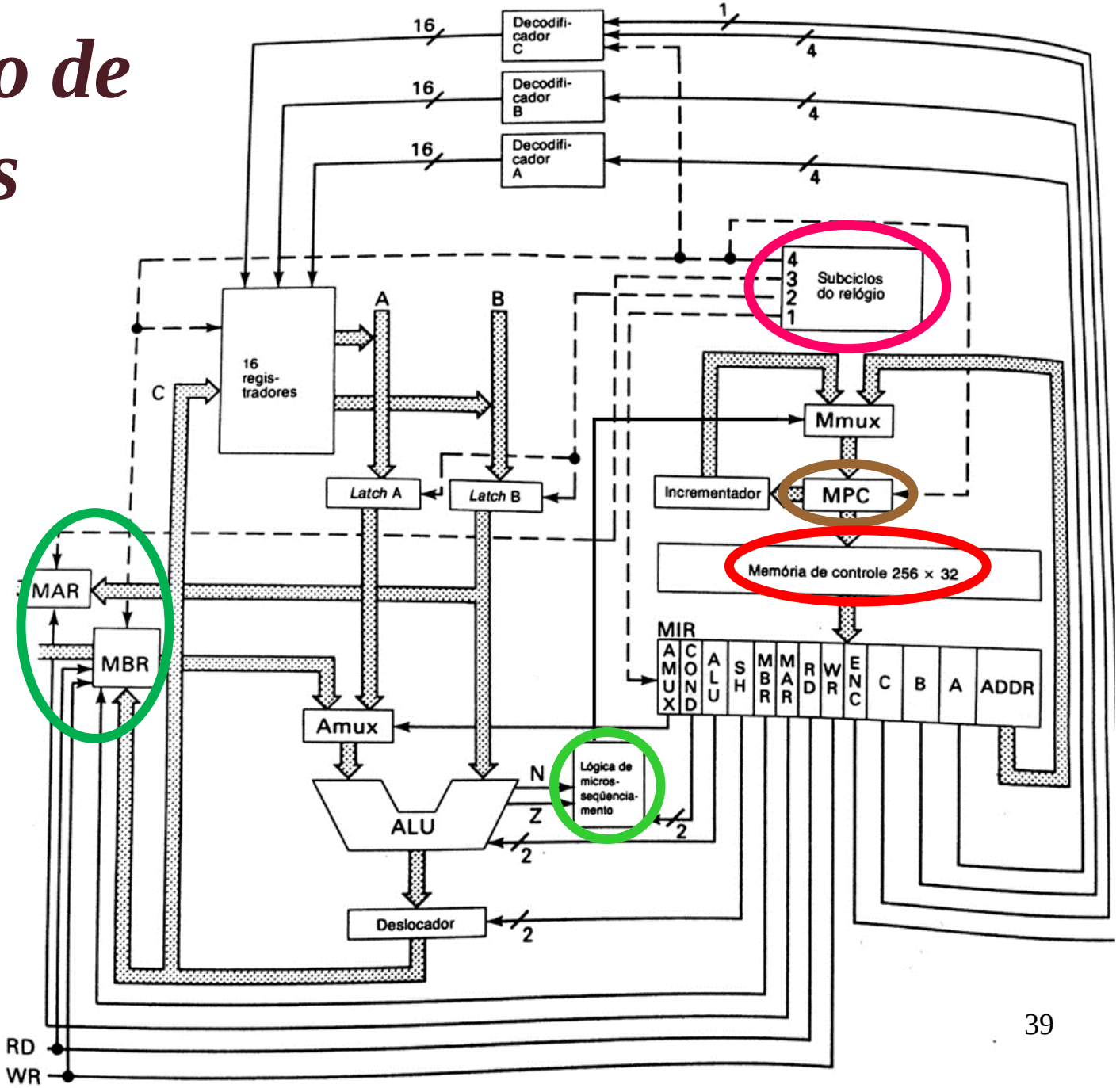
Caminho de dados

Existem vários elementos similares aos que vimos na RISC-V



Caminho de dados

E quais elementos são **novos**?



Caminho de dados microprogramado

(alguns elementos)

- **Registrador de microinstrução** (**MIR**): "*IR para microinstruções*"
- **Contador de microprograma** (**MPC**): "*PC para microinstruções*"
- **Memória de controle**: "*memória do microprograma*";
- **Unidade lógica de microsequenciamento**: definição da sequência de microinstruções a serem executadas
- **Temporizador**: relógio que define subciclos de um ciclo
- **Decodificadores**: unidades que auxiliam no controle de registradores.