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

Microprogramação – microinstruções

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)

Microinstruções

(lembrete sobre microinstrução)

•Uma **microinstrução** pode ser entendida como uma instrução para controle em baixo nível

•Por meio de uma microinstrução, é possível **ativar sinais de controle** da microarquitetura

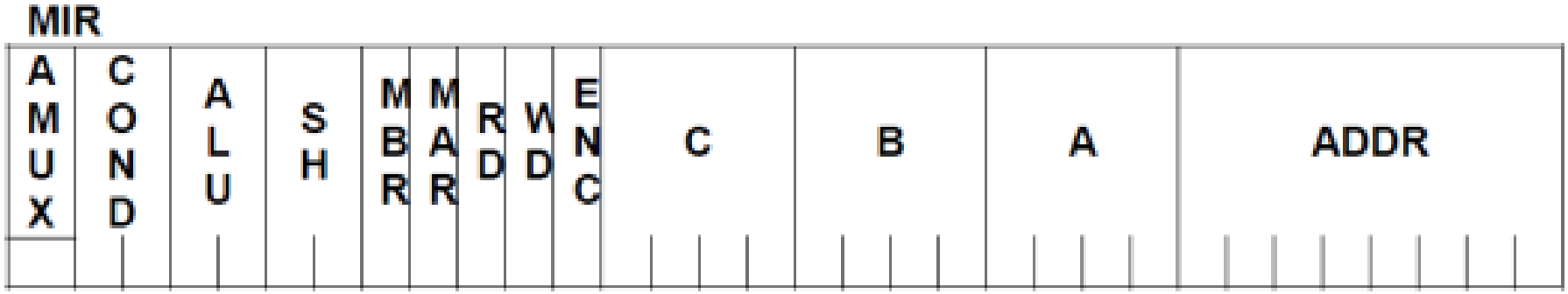
•**Microinstruções** compõem um ***microprograma***, (assim como instruções de máquina compõem um código Assembly)

•Assim como ocorrem com as instruções para arquitetura RISC-V, as microinstruções possuem um formato pré-definido

•Na microarquitetura exemplo, esse formato é dado por **32 bits** distribuídos em **13 campos**.

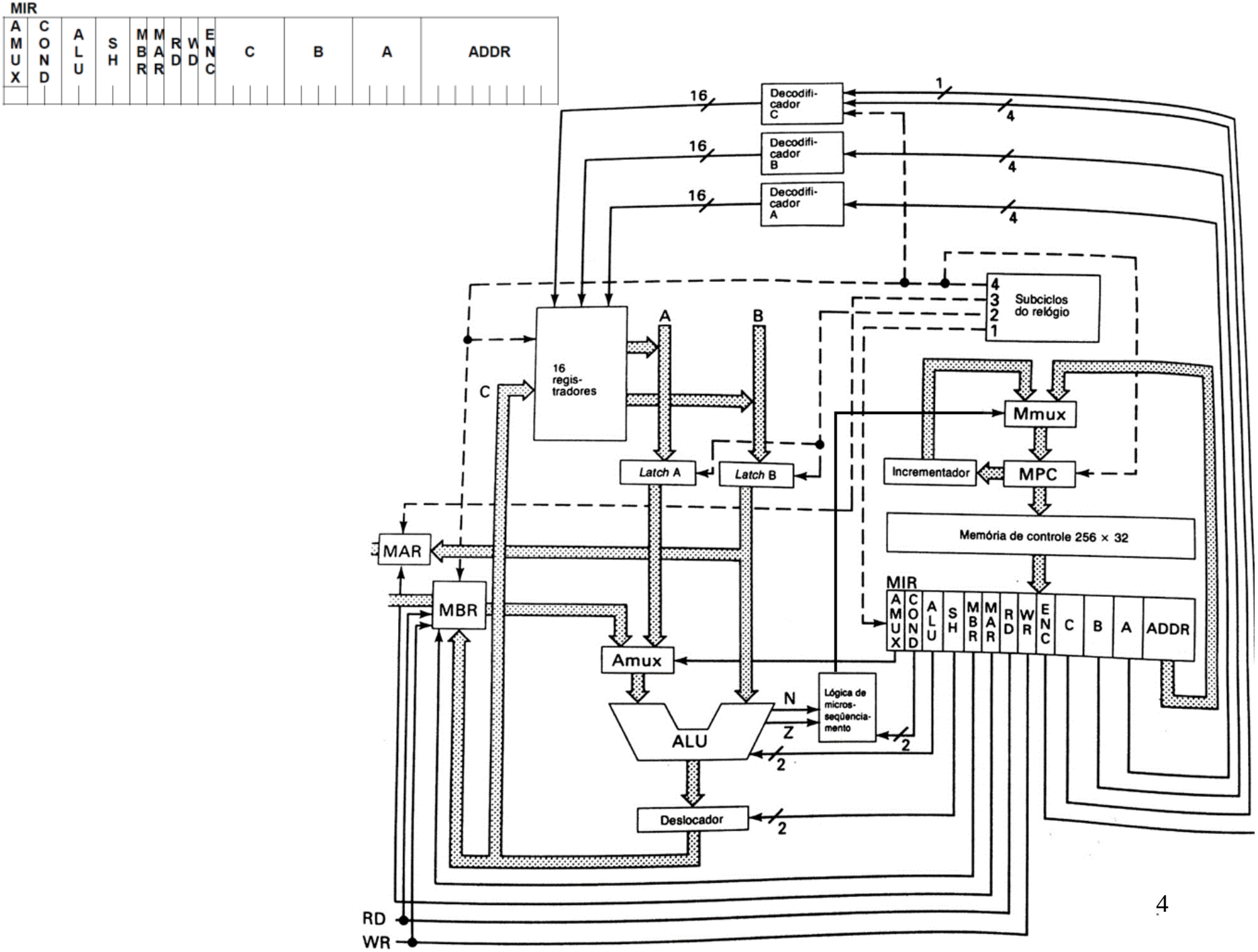
Microinstruções

(layout de microinstrução)



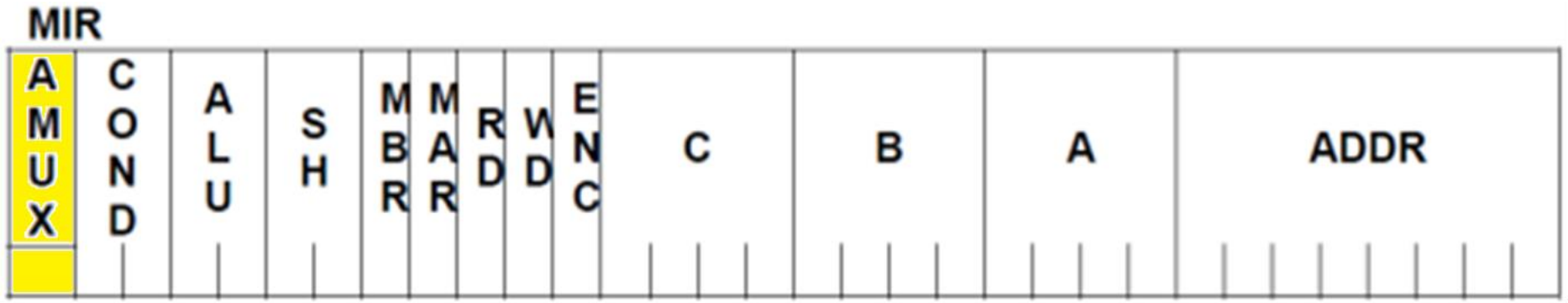
Na sua opinião, para que serve cada um dos 13 campos?

(vide caminho de dados a seguir para facilitar)


MIR
AMUX
COND
ALU
SH
MBR
MAR
RD
WD
ENC
C
B
A
ADDR
16
Decodificador C
Decodificador B
Decodificador A
1
4
Subciclos do relógio
16 registradores
A
B
C
Latch A
Latch B
Mmux
Incrementador
MPC
Memória de controle 256 × 32
MAR
MBR
Amux
ALU
N
Z
Lógica de microsseqüenciamento
Deslocador
RD
WR

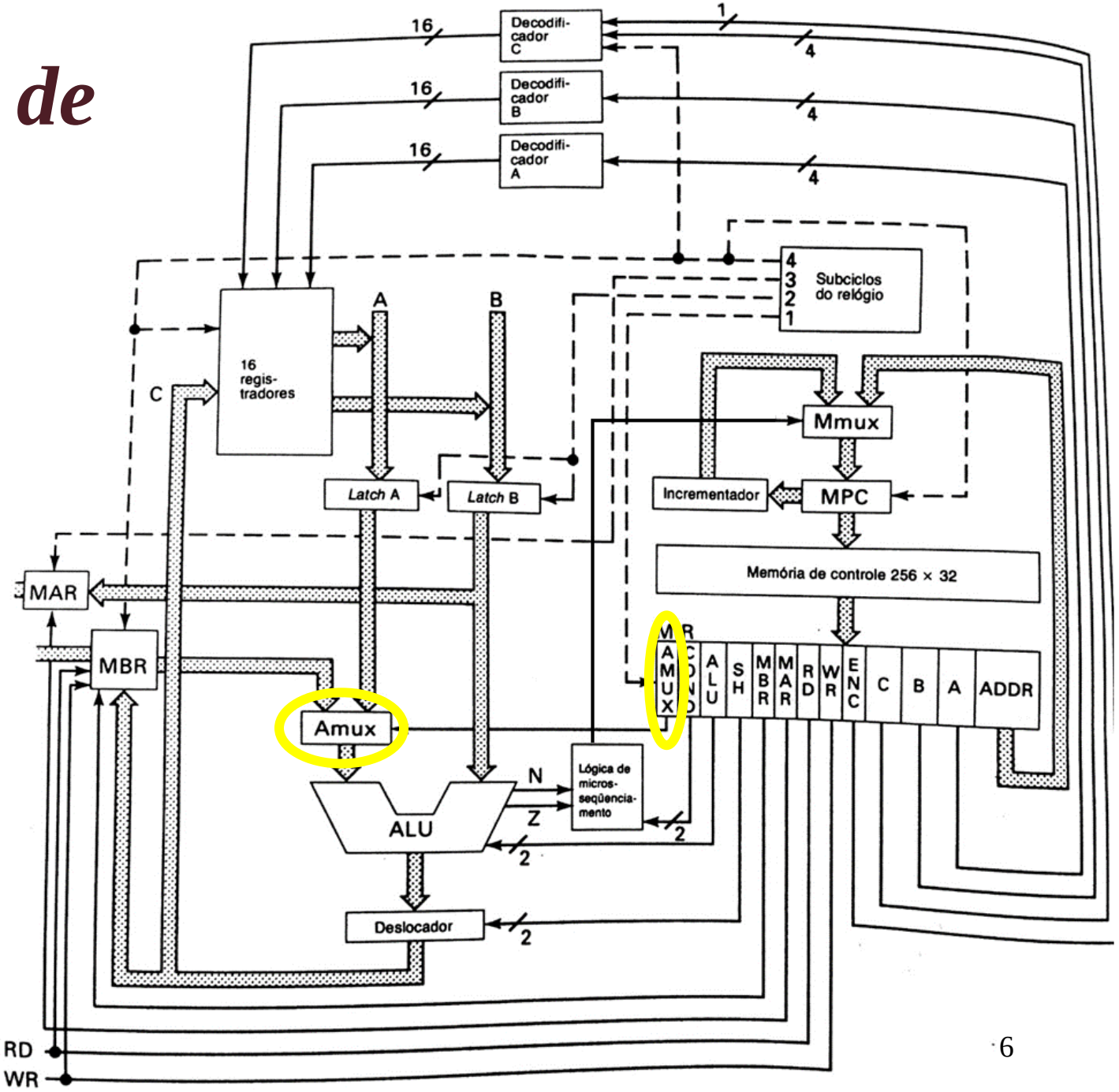
Microinstruções

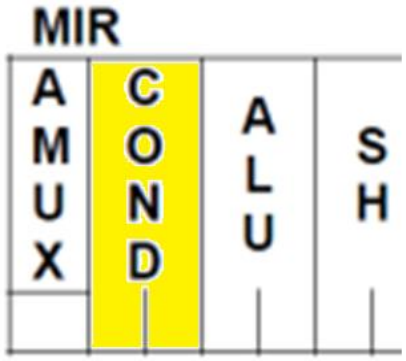
(layout de microinstrução)



- **AMUX**:
  - Multiplexador que controla a **entrada do operando esquerdo da Unidade Lógica Aritmética** (*Latch* **A**);
  - Seu valor pode ser: **0**: quando a entrada for o conteúdo de um registrador; e **1**: quando a entrada for um dado da memória que estará armazenado no registrador **MBR**.

Caminho de dados





N=1: se bit de mais alta ordem do resultado da ALU é 1

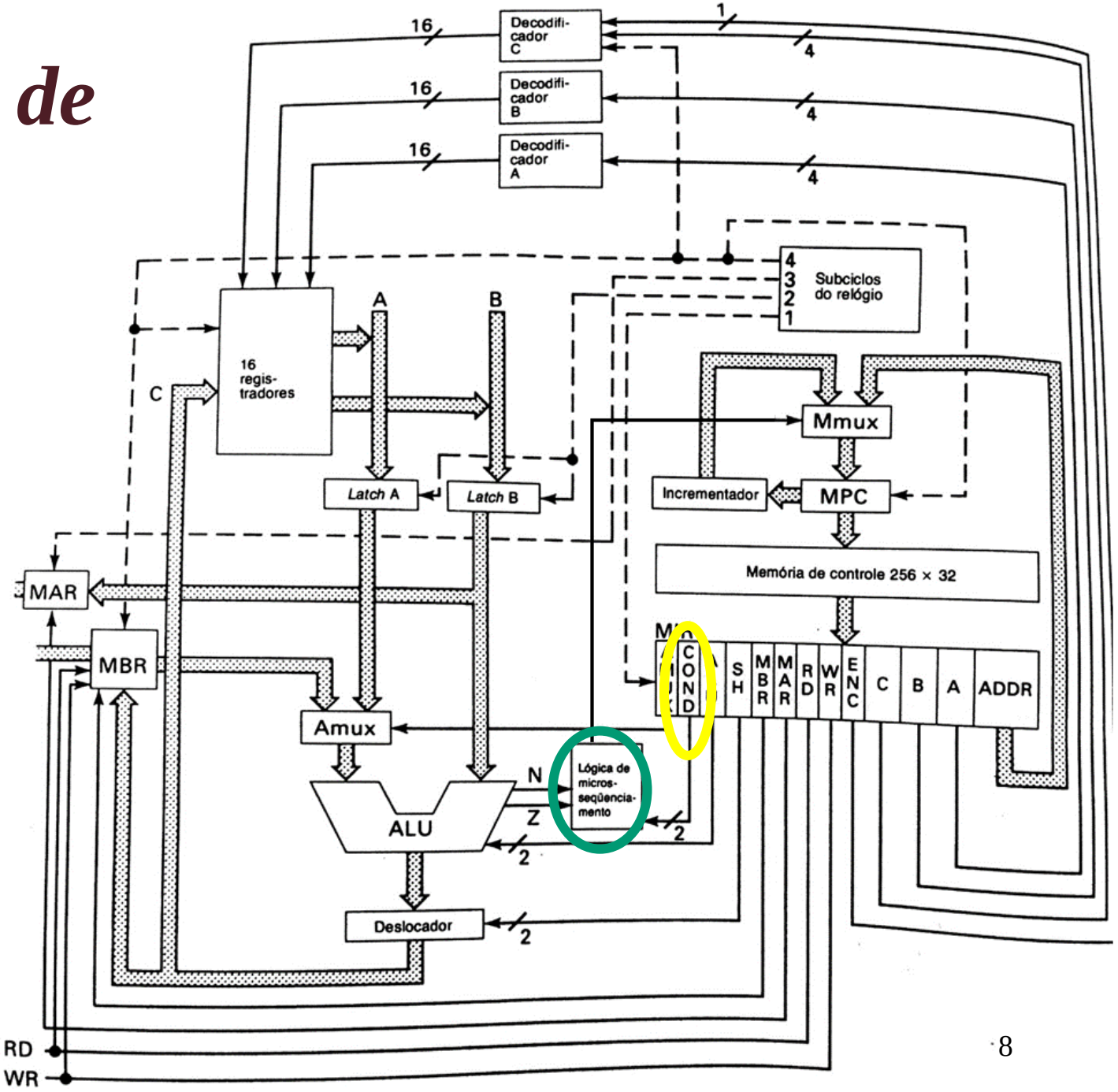
N=0: caso contrário

Z=1: se resultado da ALU é zero

Z=0: caso contrário

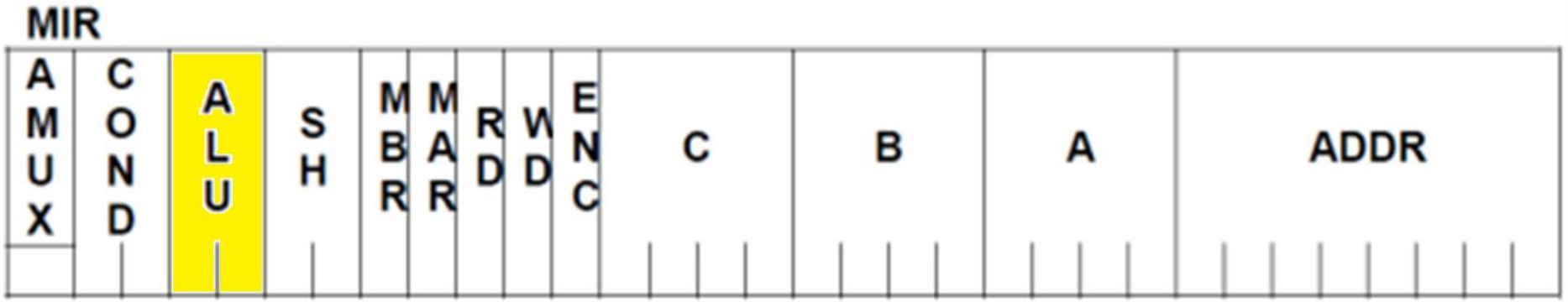
- **COND**:
  - **Sinal de controle que entra na unidade lógica de microsequenciamento**, a fim de informar se deve ou não acontecer um desvio dentro do microcódigo
  - Considera os **sinais denominados N** (**negative**) e **Z** (**zero**) **do resultado gerado pela ALU**
  - Os valores que este sinal de controle pode assumir são:
    - **0**: não desvie
    - **1**: desvie se **N =1**
    - **2**: desvie se **Z=1**
    - **3**: desvie sempre

Caminho de dados



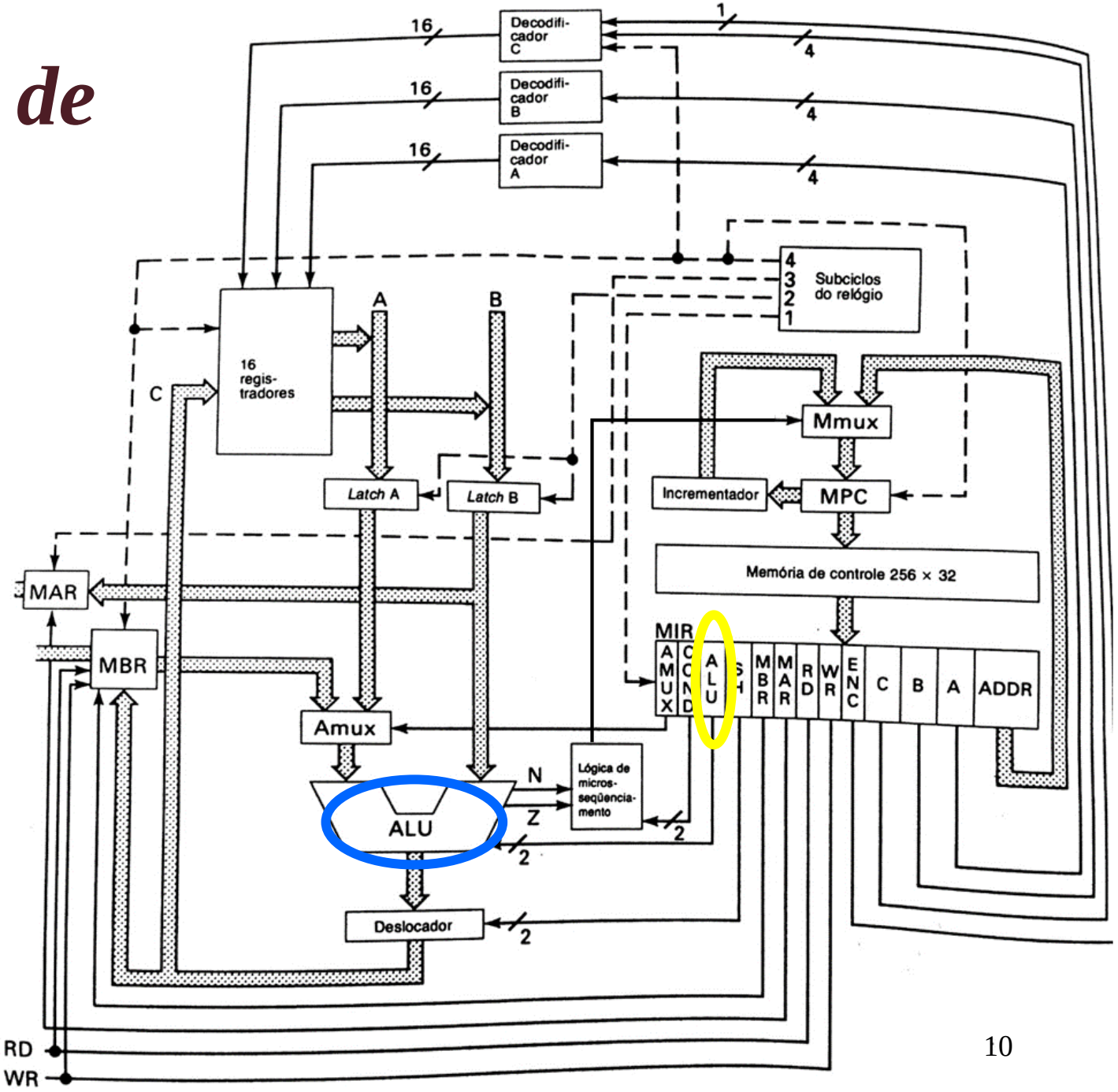
Microinstruções

(layout de microinstrução)



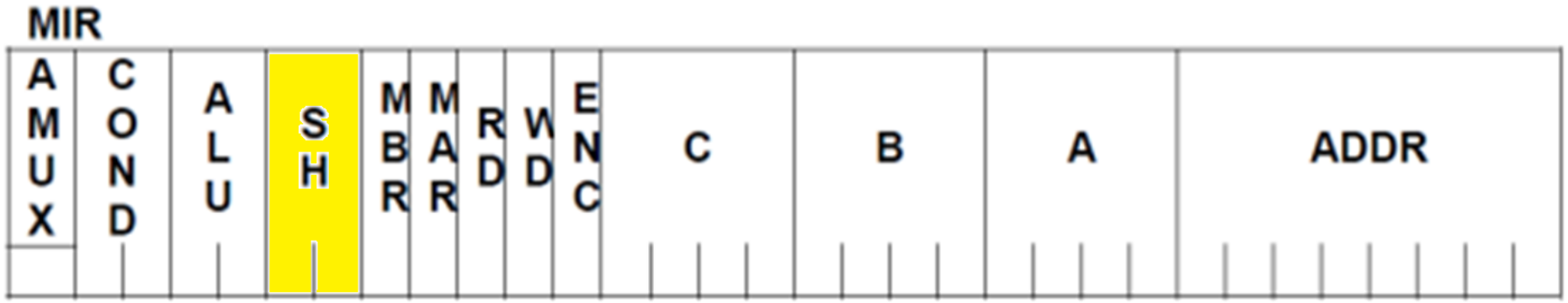
- **ALU**:
  - Sinal de controle que indica **a operação que a ALU deverá executar** neste ciclo de clock.
  - Os valores que este sinal pode assumir são:
    - **0**: soma (A + B)
    - **1**: and (A AND B)
    - **2**: repetir a entrada (A)
    - **3**: complemento da entrada (NOT A)

Caminho de dados



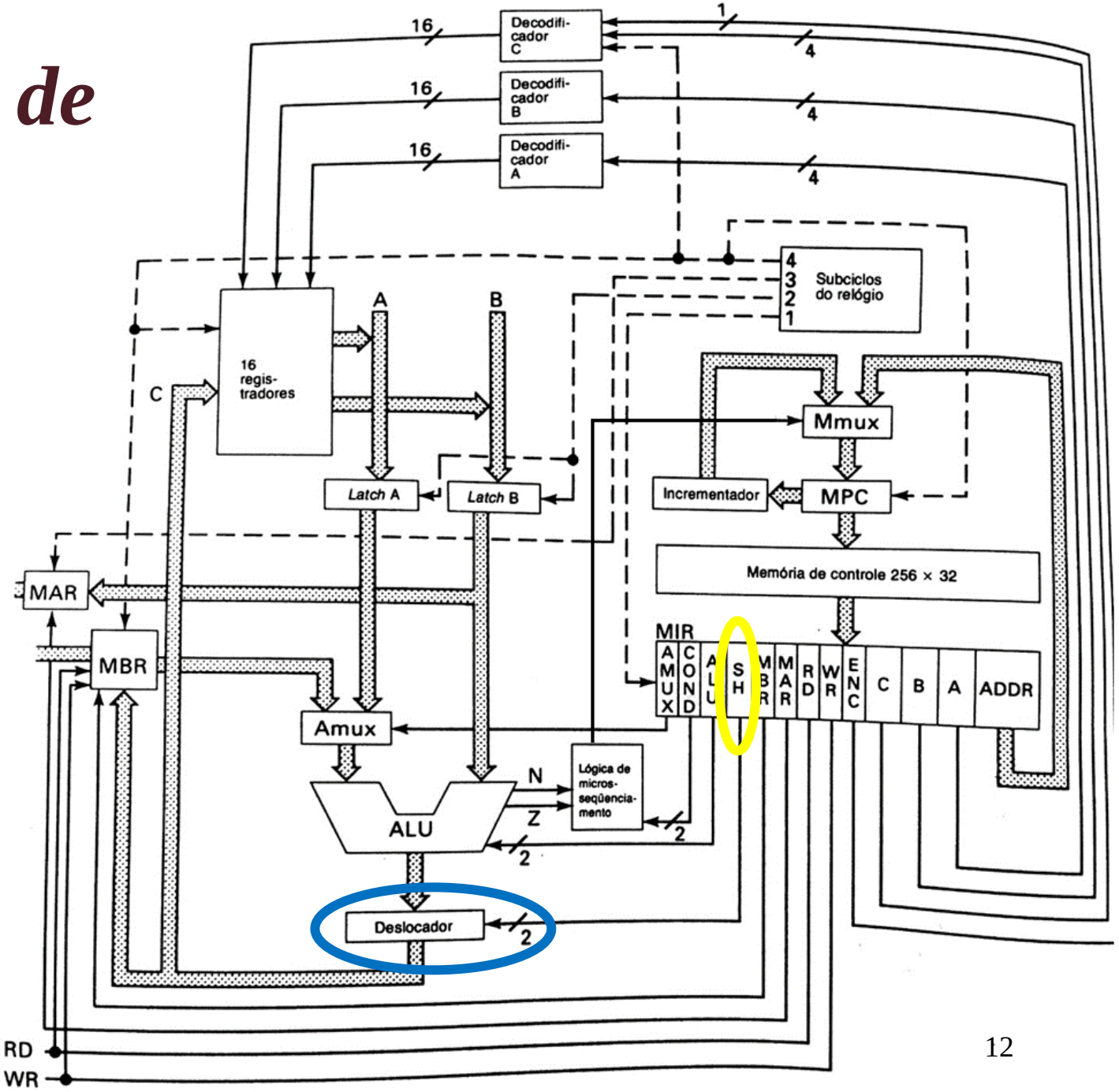
Microinstruções

(layout de microinstrução)



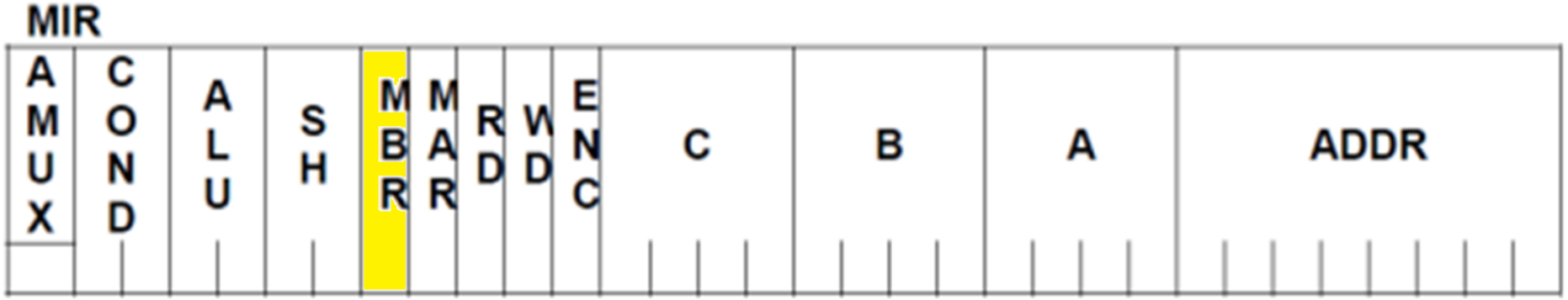
- **SH**:
  - Sinal que indica **quando irá ocorrer um deslocamento**.
  - Seus valores podem ser:
    - **0**: não desloque
    - **1**: desloque 1 bit à esquerda (SHIFT LEFT)
    - **2**: desloque 1 bit à direita (SHIFT RIGHT)

Caminho de dados



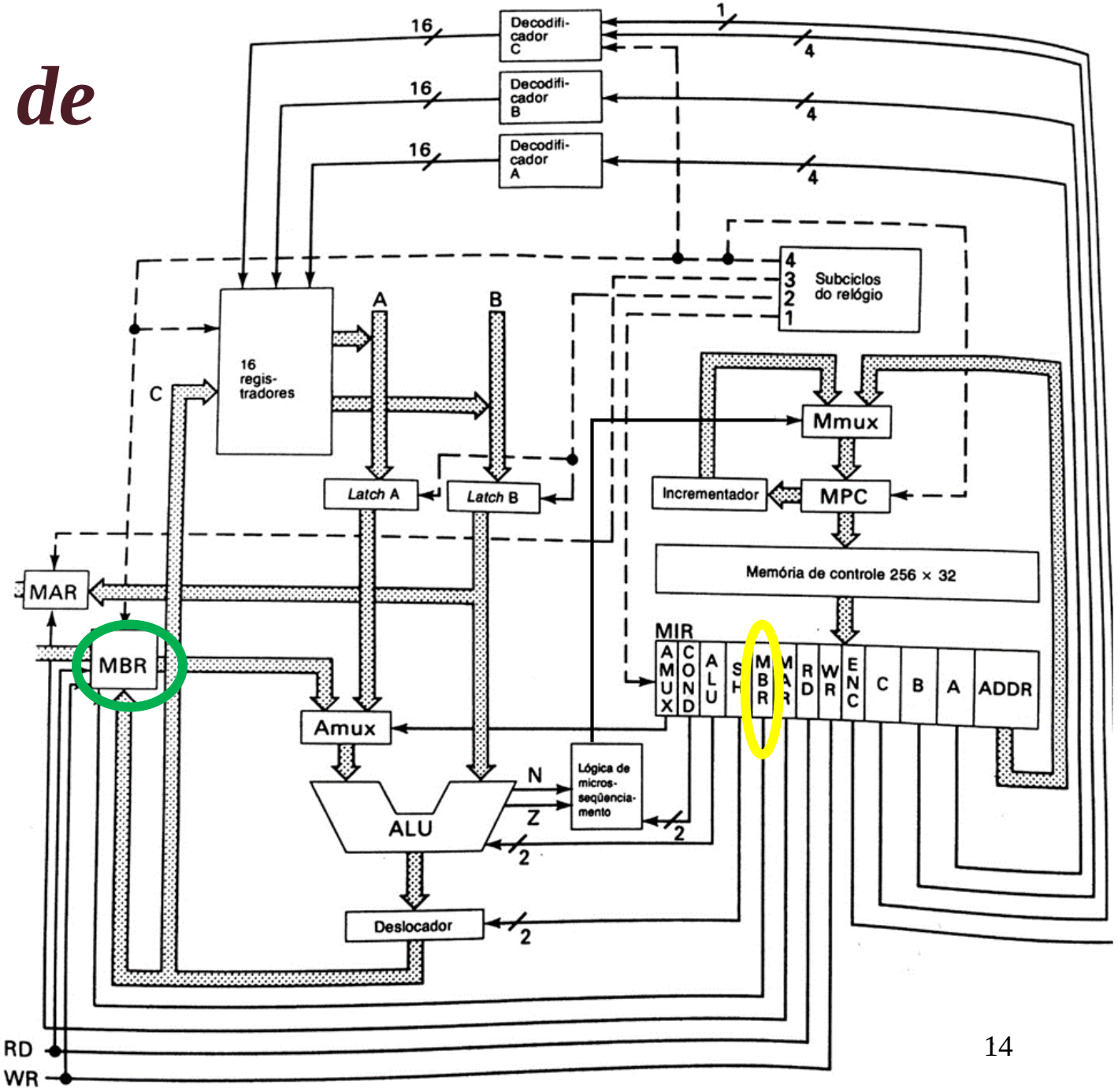
Microinstruções

(layout de microinstrução)



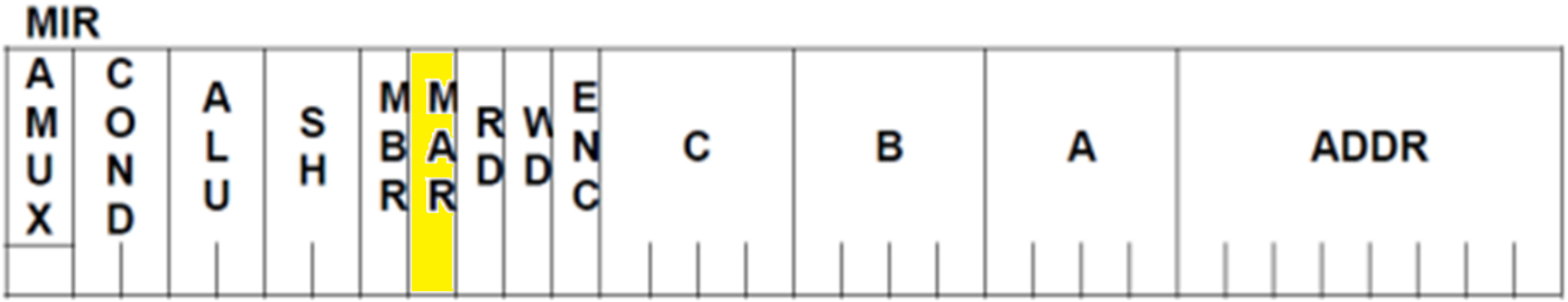
- **MBR**:
  - Sinal de controle que assume os seguintes valores:
    - **1**: indica que o registrador **MBR** será carregado com o valor de saída do deslocador
    - **0**: indica que não vai ser carregado.

Caminho de dados



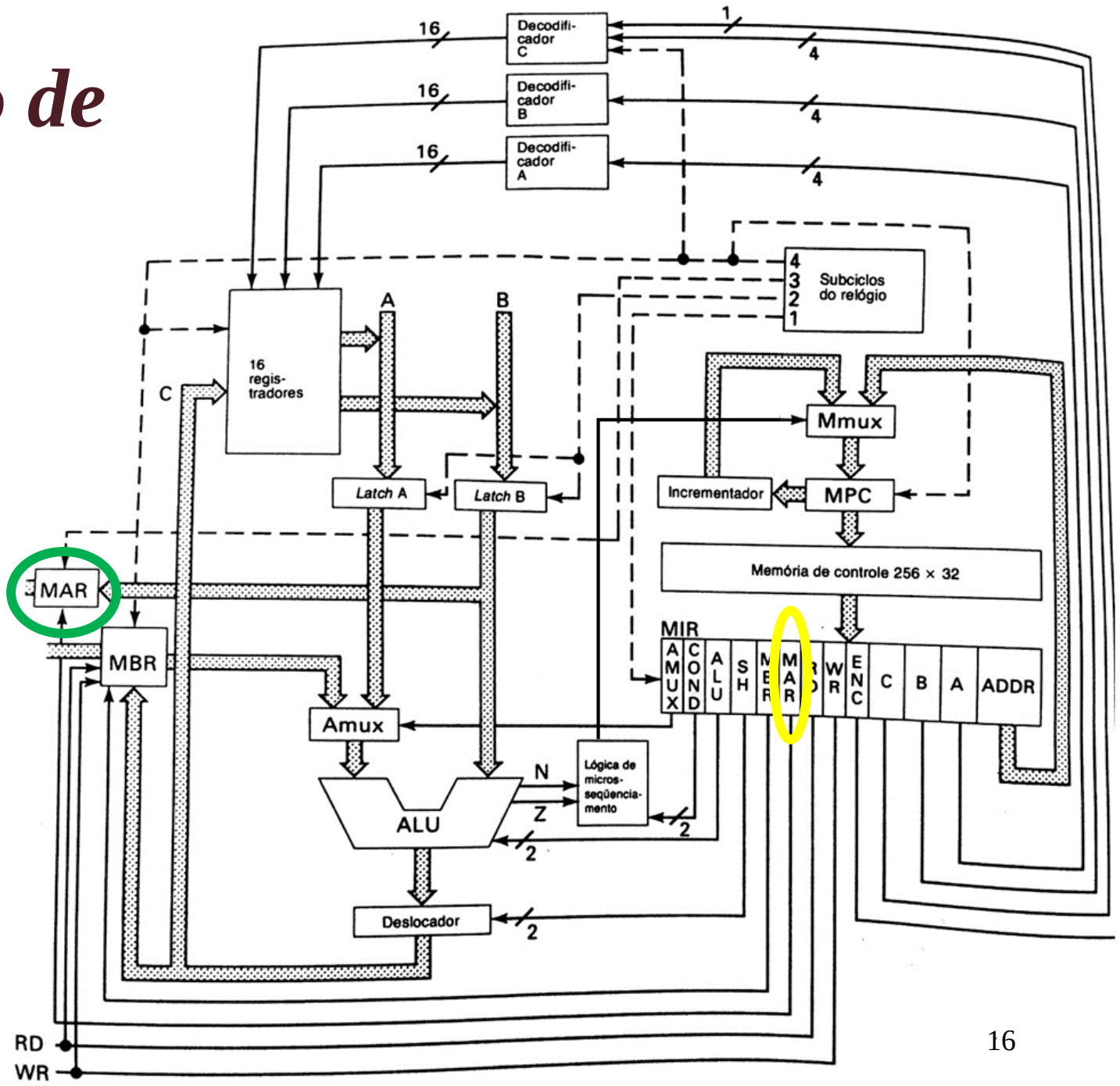
Microinstruções

(layout de microinstrução)



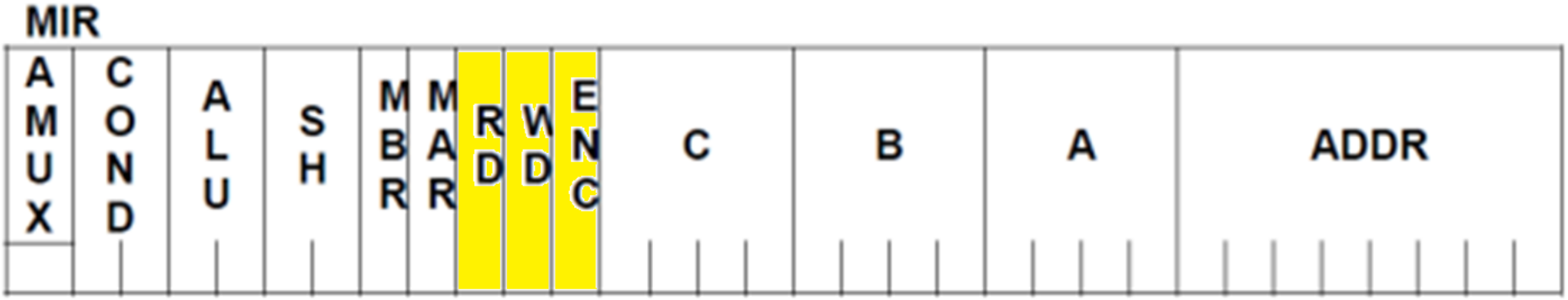
- **MAR**:
  - Sinal de controle que assume os seguintes valores:
    - **1**: indica que o registrador **MAR** será carregado com o valor contido no **registrador B**
    - **0**: indica que não vai ser carregado.

Caminho de dados



Microinstruções

(layout de microinstrução)

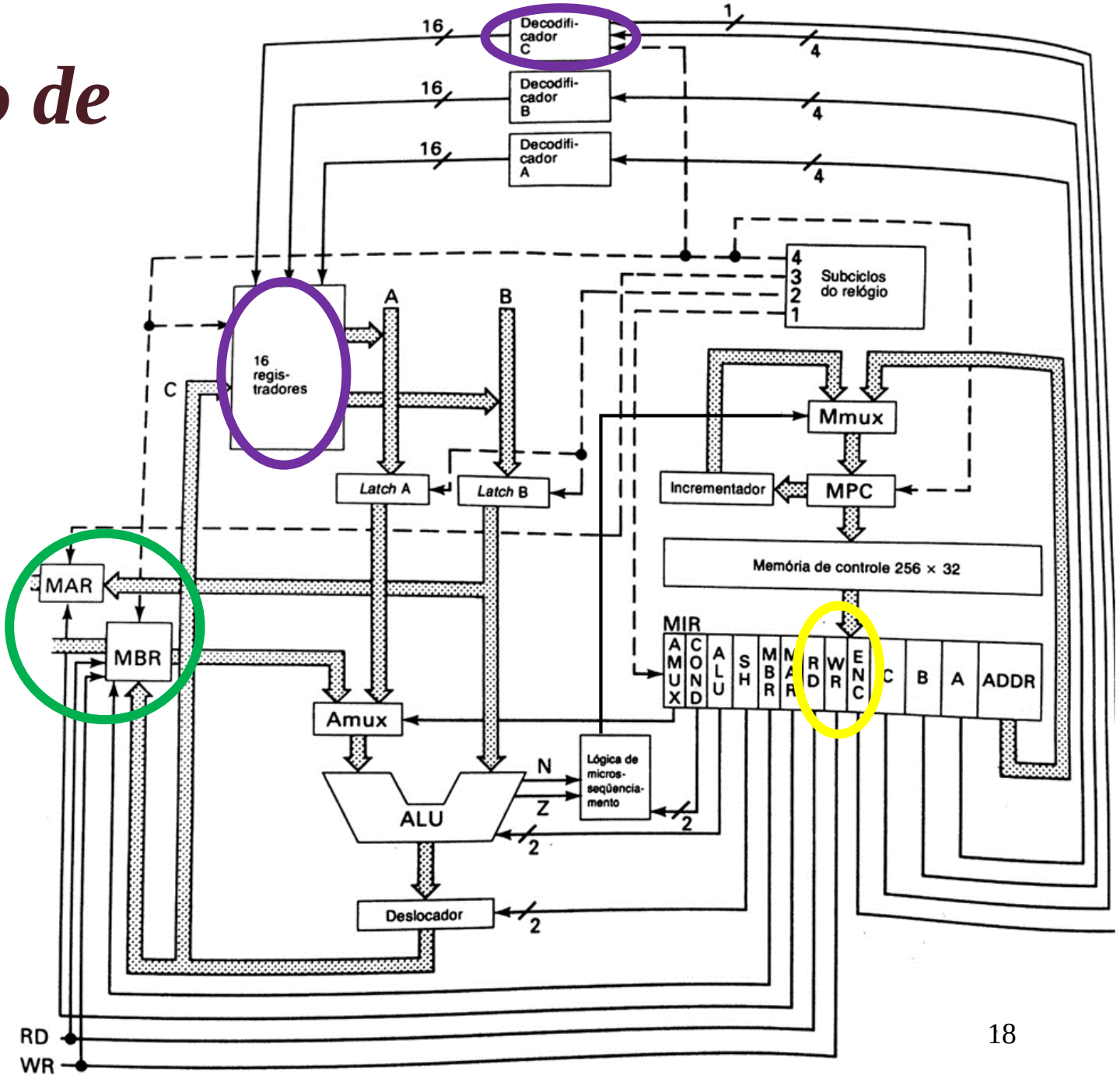


•**RD**: quando este sinal de controle está com valor **1**, o **dado,** contido no endereço de memória especificado pelo registrador **MAR**, **é lido da memória e colocado em MBR** (READ).

•**WR**: quando este sinal de controle está com valor **1**, o **conteúdo do registrador MBR é escrito na memória**, no endereço especificado pelo registrador **MAR** (WRITE, ou STORE).

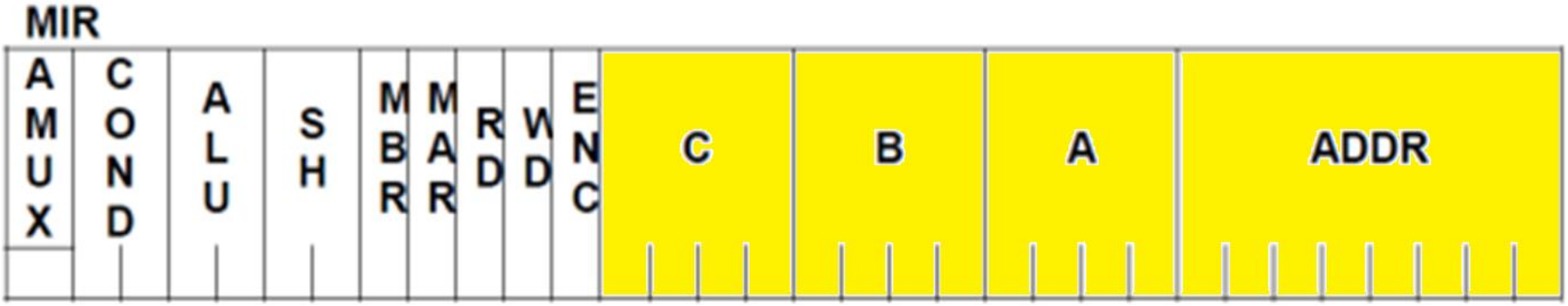
•**ENC**: indica se vai haver escrita no **registrador destino**.

Caminho de dados



Microinstruções

(layout de microinstrução)



- **C**: indica o endereço do **registrador destino**
- **B**: indica o endereço do **segundo operando da ALU**
- **A**: indica o endereço do **primeiro operando da ALU**
- **ADDR**: **endereço** para onde será desviado o microcódigo se a condição for aceita.

Temporização das microinstruções

Cada ciclo de clock é dividido em **4 subciclos**, onde os eventos chave durante cada um dos subciclos são:

1 – Carregar a **próxima microinstrução** a ser executada no registrador **MIR** (Registrador de microinstrução)

2 – Colocar os **conteúdos dos operandos** dentro dos registradores **A** e **B**

3 – Dar um tempo para que a **ALU** execute uma operação com os operandos e para que o **deslocador** produza uma saída estável e se necessário, carregar o registrador **MAR**

4 – Armazenar o valor existente no **barramento C** no **registrador destino** e carregar **MBR**, se for necessário.

Exemplo de microprograma

Ver material complementar: "CISC zerada.doc"

Análise de microinstrução

- Por exemplo, vamos analisar a microinstrução

MAR:=PC; RD;

- Objetivos:
  - Registrador **MAR** deve receber o conteúdo de **PC**
  - Além disso, deve ser habilitado o sinal de leitura (**RD**) em memória
- Para isso ocorrer, quais os bits em cada um dos campos da microinstrução?

Comando	AMUX	COND	ALU	SH	MBR	MAR	RD	WR	ENC	C	B	A	ADDR
MAR:=PC; rd;													

Análise de microinstrução

- AMUX:

AMUX
X

  - Observando o caminho de dados, nenhuma operação na ALU é necessária para realizar a microinstrução, portanto o bit desse campo pode ser qualquer um (*“don’t care”*)

- COND:

COND
00

  - Não há nenhuma condição a ser avaliada na microinstrução, logo, não deve ser realizado nenhum desvio após a microinstrução terminar, seguindo para a microinstrução seguinte. Portanto, os bits devem ser 00 (“não desvie”)

Análise de microinstrução

- ALU e SH:

ALU	SH
xx	xx

  - Como nenhuma operação na ALU é necessária, os bits desses campos não importam (*"don't care"*)

- MBR:

MBR
0

  - Sinal 0 pois MBR não deve receber nenhum valor

- MAR:

MAR
1

  - Sinal 1 pois o regsitrador MAR deve receber algum valor (endereço de PC, contido no registrador Latch B)

Análise de microinstrução

- RD:

RD
1

  - Sinal 1 pois deseja-se habilitar a memória para leitura

- WR:

WR
0

  - Sinal 0 pois não deseja-se habilitar a memória para escrita

- ENC

ENC
0

  - Sinal 0 pois não deve haver escrita no banco de registradores

- C:

C
xxxx

  - “*Don’t care*” pois não há registrador de destino

- B:

B
0000

  - Latch B deve receber o endereço de PC (0000)

- A:

A
xxxx

  - Não importa qual será o 1º operando da ALU

- ADDR:

ADDR
xxxxxxx

  - “*Don’t care*” pois ocorrerá desvios

Análise de microinstrução

- Resultado

Comando	AMUX	COND	ALU	SH	MBR	MAR	RD	WR	ENC	C	B	A	ADDR
MAR:=PC; rd;	x	00	xx	xx	0	1	1	0	0	xxxx	0000	xxxx	xxxxxxx