ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

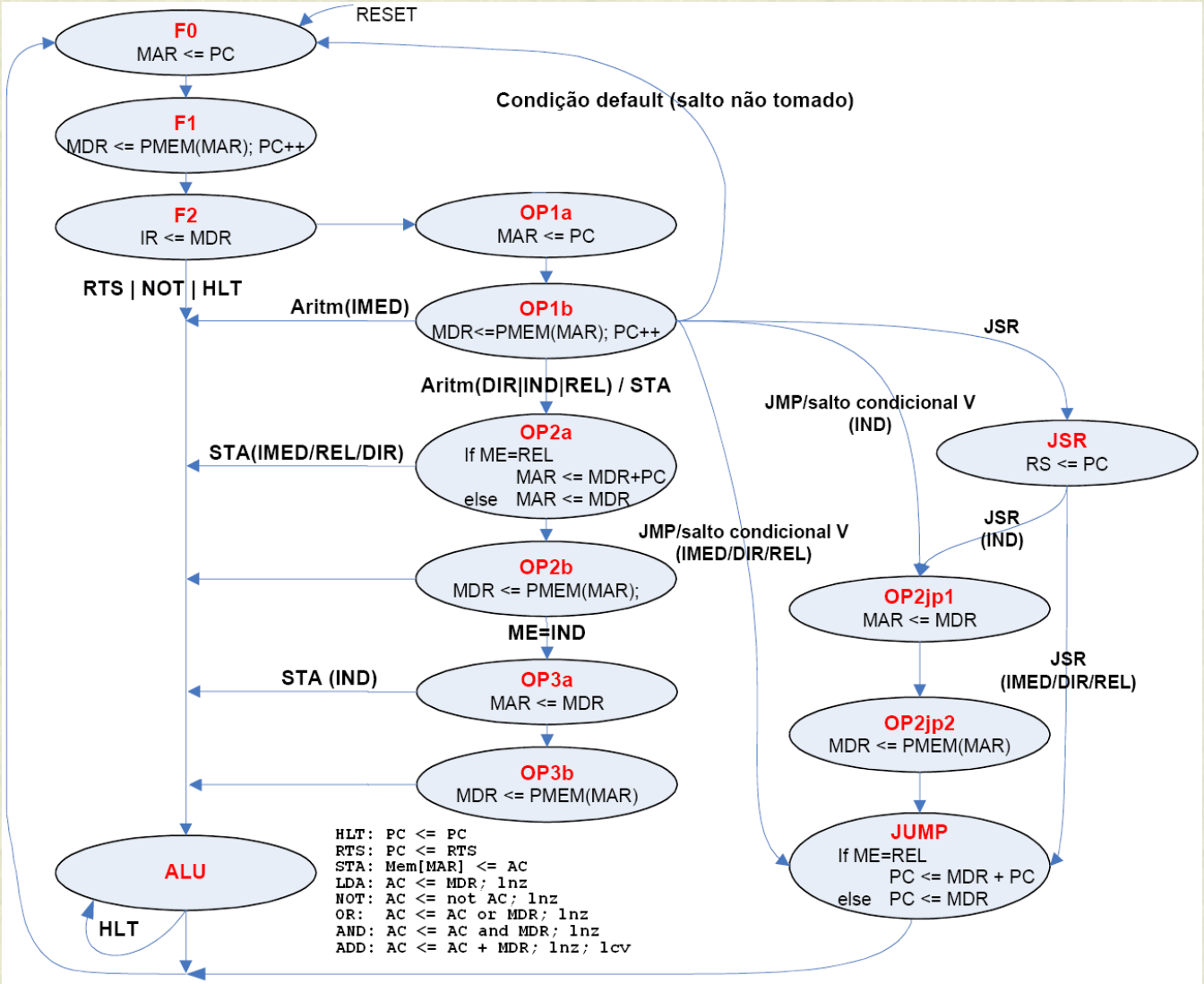
Computador Cleópatra

Assembly

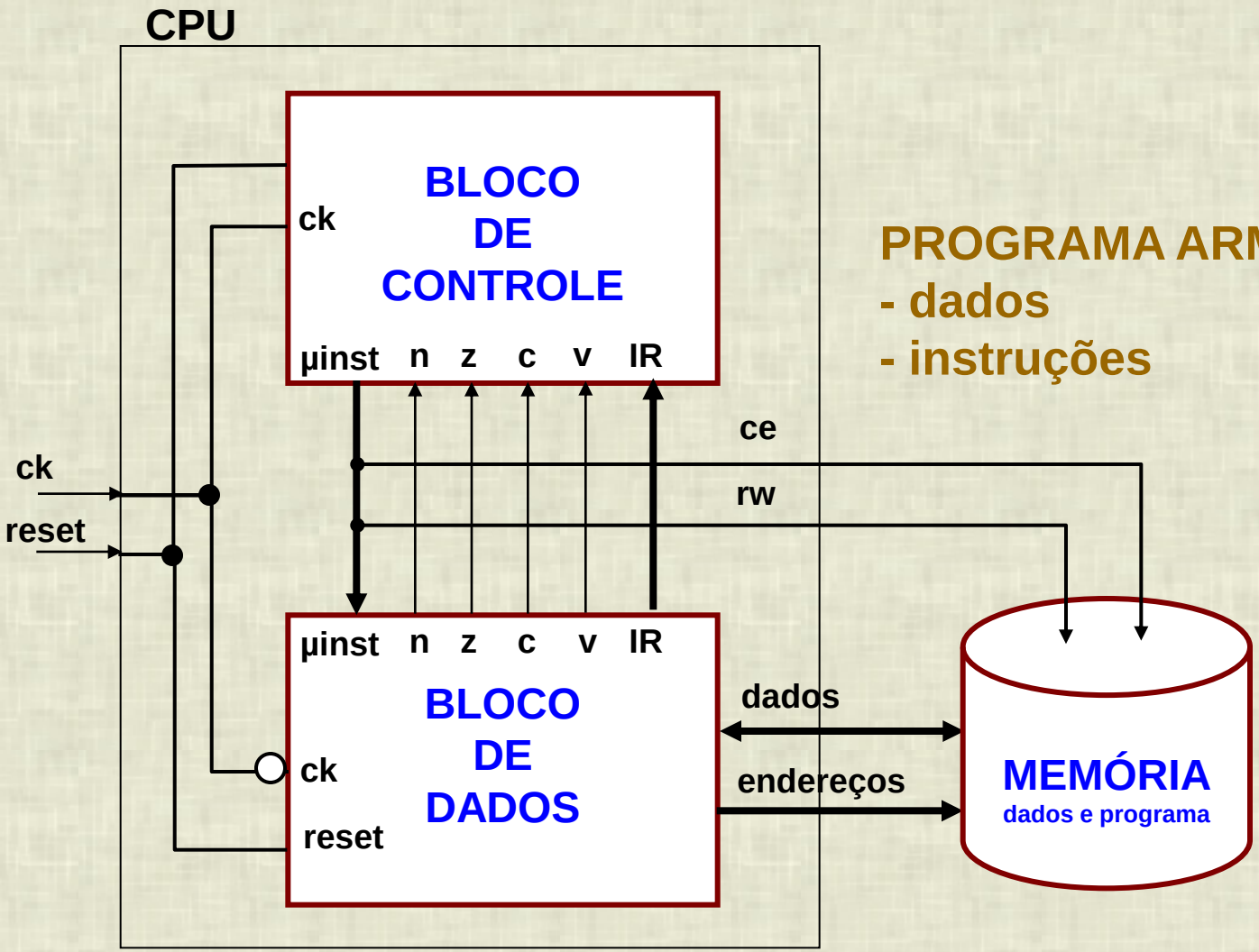
Alexandre Amory
Edson Moreno

Na Aula Anterior …

- **Vimos a máquina de estados da Cleo**

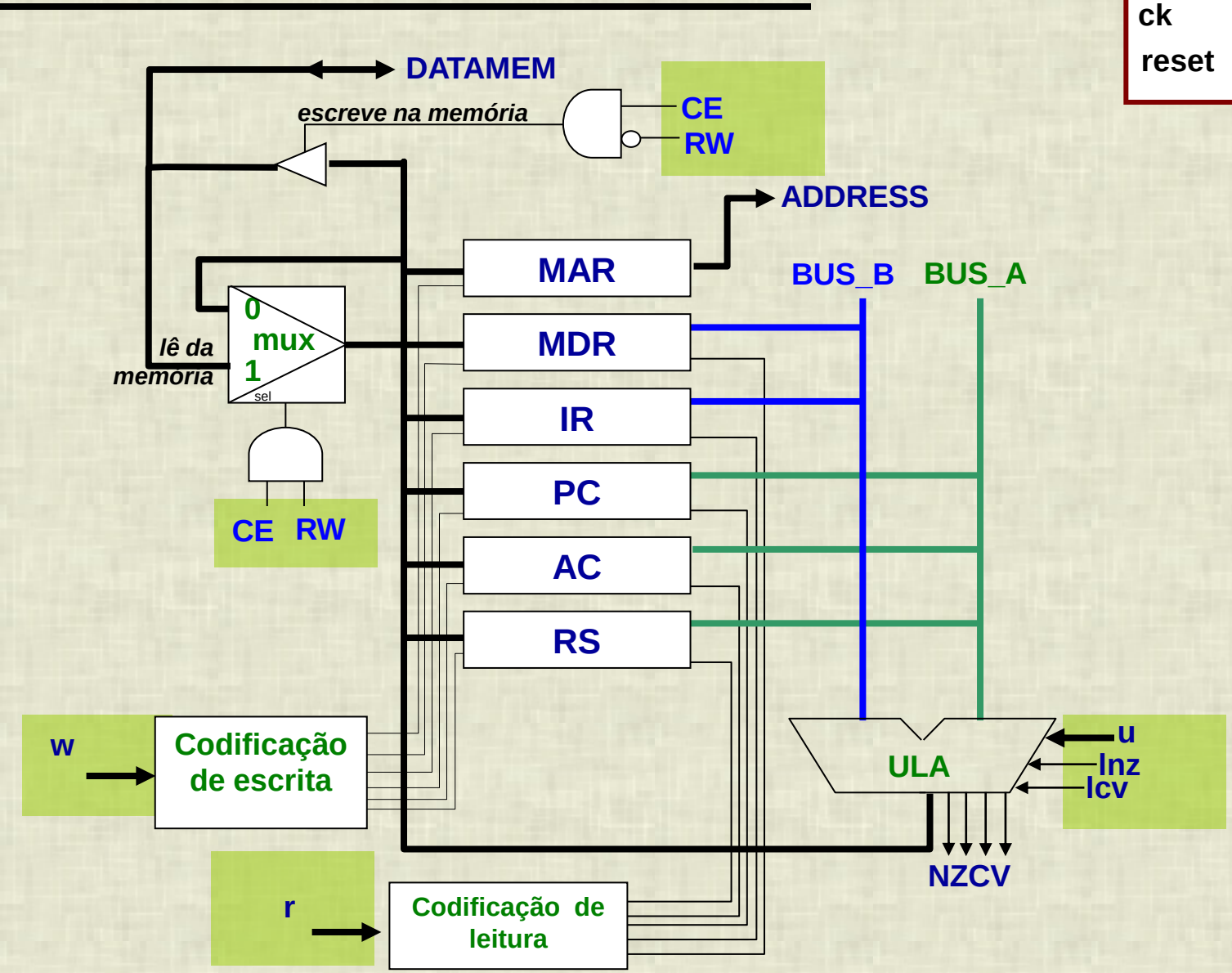


Arquitetura Cleópatra - Von Neumann

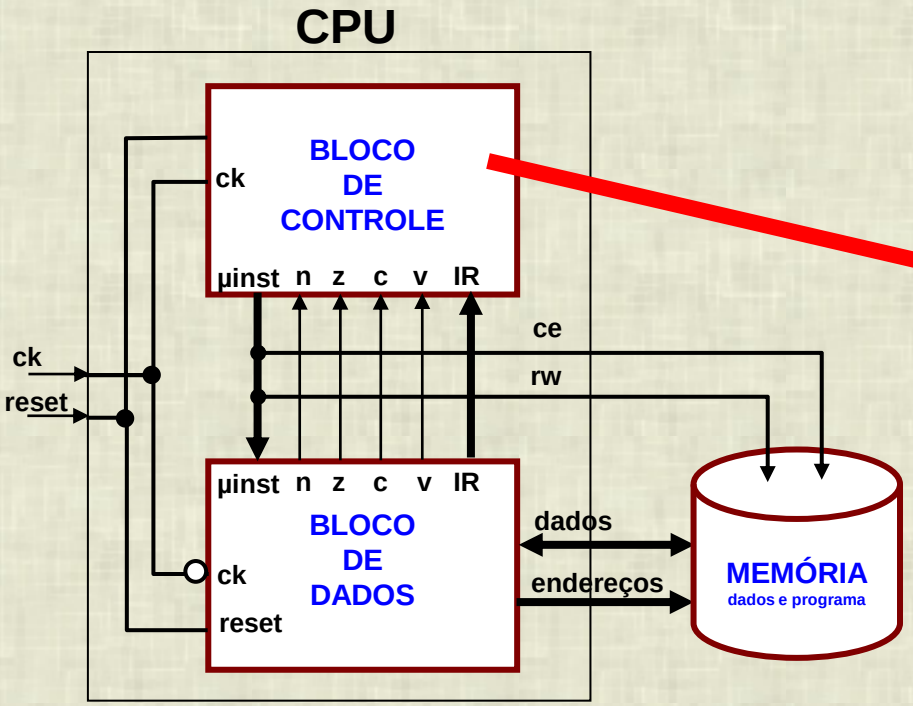


PROGRAMA ARMAZENADO
- dados
- instruções

Bloco de Dados



Bloco de Controle



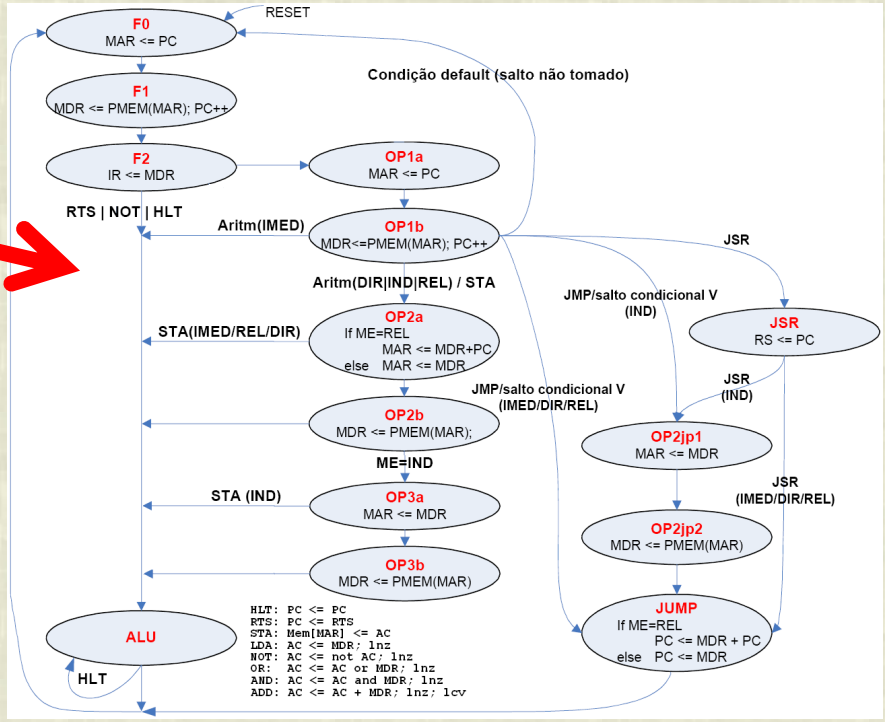
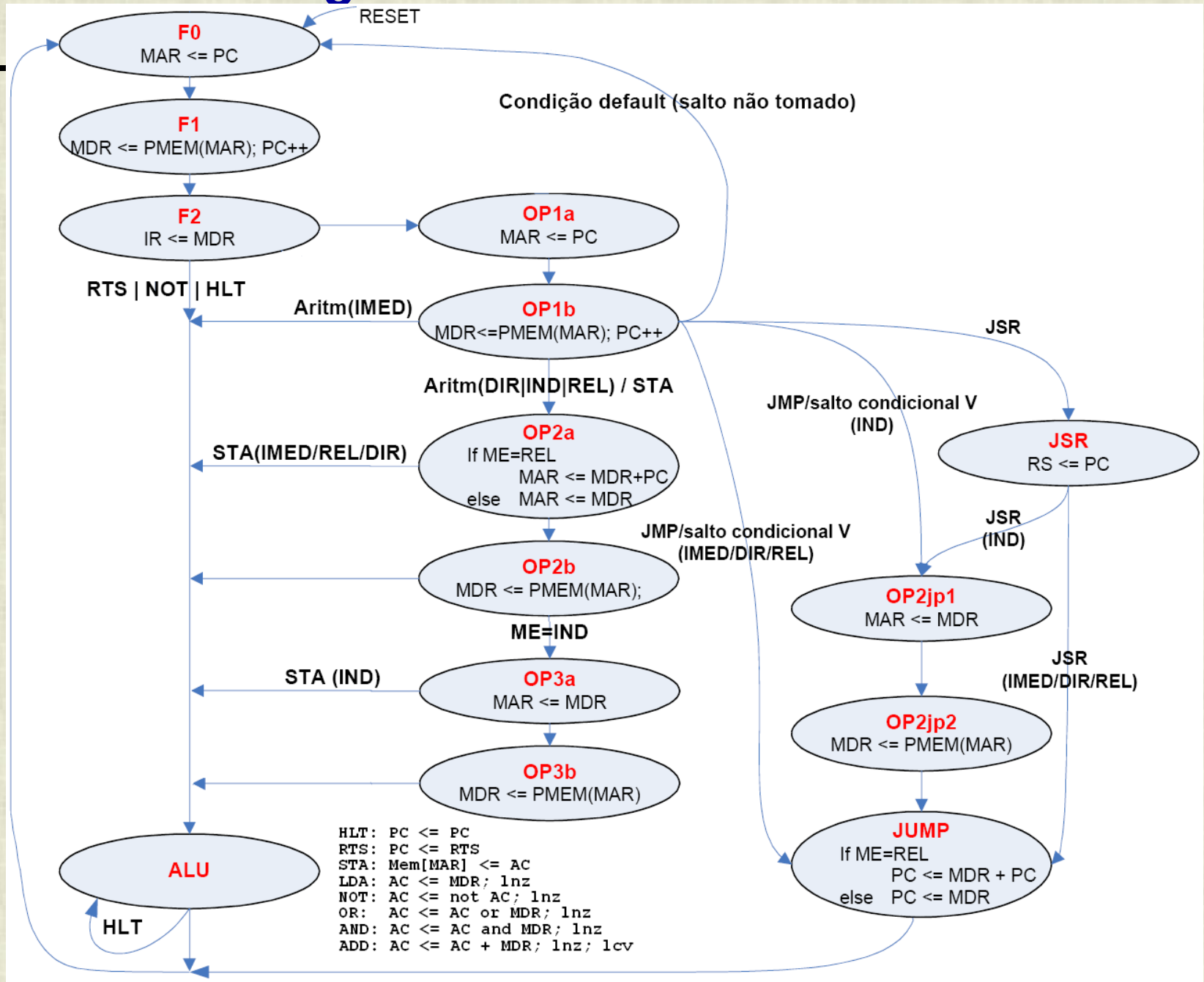


Diagrama de Estados da Cleo



Modo de Endereçamento

- **Imediato**
  - Operando é o próprio dado
  - Usado para representar constantes
    - Ex.: Número **45** do trecho de código C
- **Direto**
  - Operando é endereço do dado
  - Usado para representar variáveis
    - Ex. Variável **entrada** do trecho de código C
- **Indireto**
  - Operando é endereço do endereço do dado
  - Usado para representar ponteiros
    - Ex. Ponteiro ***p** (que aponta para valor) do trecho de código C

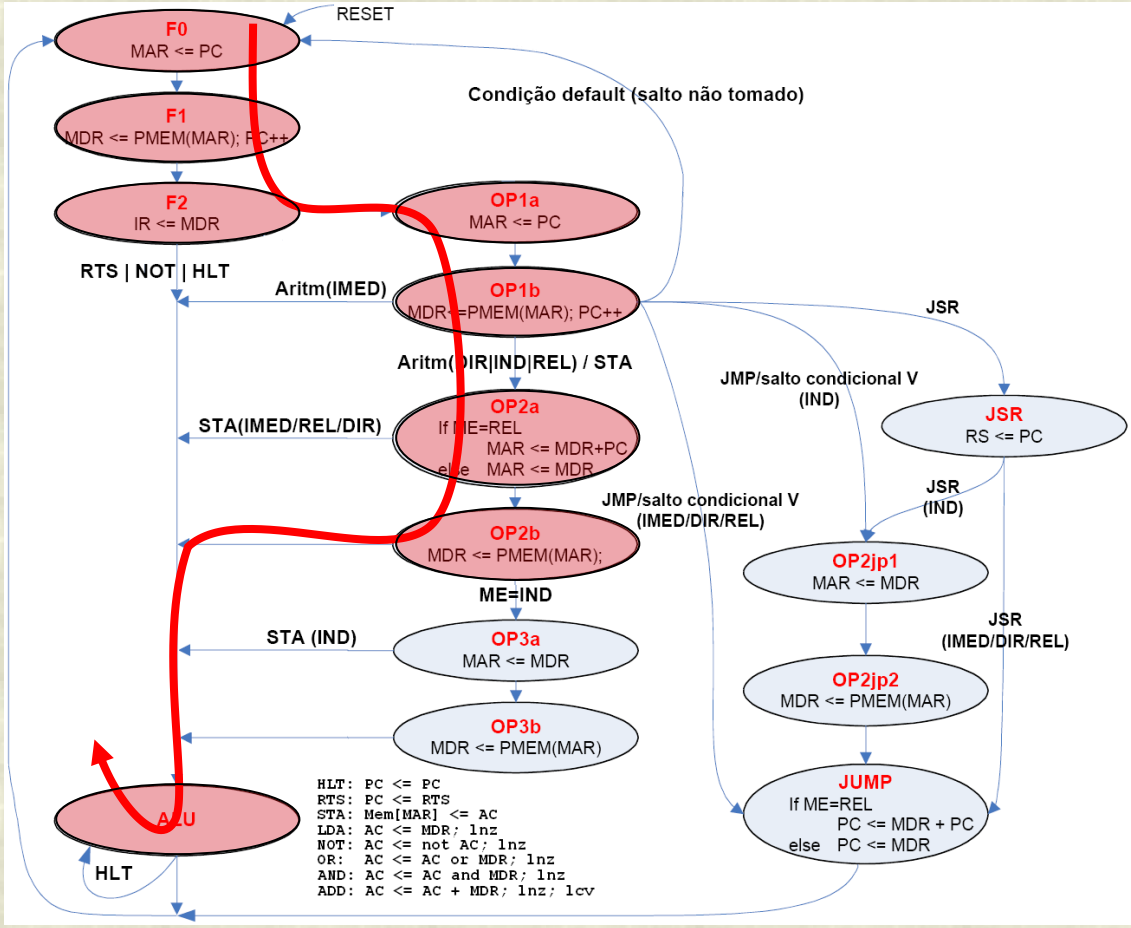
```
int valor;
int *p = &valor;
int entrada = 8;

*p = entrada + 45;
```

Descrição das Instruções

Mnemônico	Operação
NOT	Complementa (inverte) todos os bits de AC.
STA oper	Armazena AC na memória dada por oper.
LDA oper	Carrega AC com conteúdos de memória da posição dada por oper.
ADD oper	Adiciona AC ao conteúdo da memória dada por oper.
OR oper	Realiza OU lógico do AC com conteúdo da memória dada por oper.
AND oper	Realiza E lógico do AC com conteúdo da memória dada por oper.
JMP oper	PC recebe dado especificado por oper (desvio incondicional).
JC oper	Se C=1, então PC recebe valor dado por oper (desvio condicional).
JV oper	Se V=1, então PC recebe valor dado por oper (desvio condicional).
JN oper	Se N=1 então PC recebe valor dado por oper (desvio condicional).
JZ oper	Se Z=1, então PC recebe valor dado por oper (desvio condicional).
JSR oper	RS recebe conteúdo de PC e PC recebe dado de oper (subrotina).
RTS	PC recebe conteúdos de RS (retorno de subrotina).
HLT	Suspende processo de busca e execução de instruções.

Meu Primeiro Prog Assembly

- **Linguagem alto nível (e.g. C, JAVA)**
  - *C = A+3
- **Assembly da Cleo**
  - LDA A
  - ADD #3
  - STA C,I



Na Aula de Hoje …

- Exemplos dos outros modos de endereçamento
- Exemplo de IF
- Exemplo de LAÇO
- Exemplo de programa completo

Estrutura de um Programa Assembly Completo

CÓDIGO

Programa

```
     Rótulo    Mnem      Operando
1    .CODE
2              ORG       #00H
3    INIT:     LDA       END1
4              ADD       END2,I
5              STA       END3
6              AND       #0FH
7              JZ        FIM,R
8              NOT
9    FIM:      HLT
10   .ENDCODE
```

Dados

```
     Rótulo    Mnem      Operando
1    .DATA
2              ORG       #90H
3    END1:     DB        #30H
4    END2:     DB        #END4
5    END3:     DB        #00H
6    END4:     DB        #5BH
7    .ENDDATA
```

Estrutura de um Programa Assembly Completo

CÓDIGO

Programa

```
     Rótulo    Mnem      Operando
1    .CODE
2              ORG       #00H
3    INIT:     LDA       END1
4              ADD       END2,I
5              STA       END3
6              AND       #0FH
7              JZ        FIM,R
8              NOT
9    FIM:      HLT
10   .ENDCODE
```

início do corpo do programa (0)

corpo do programa

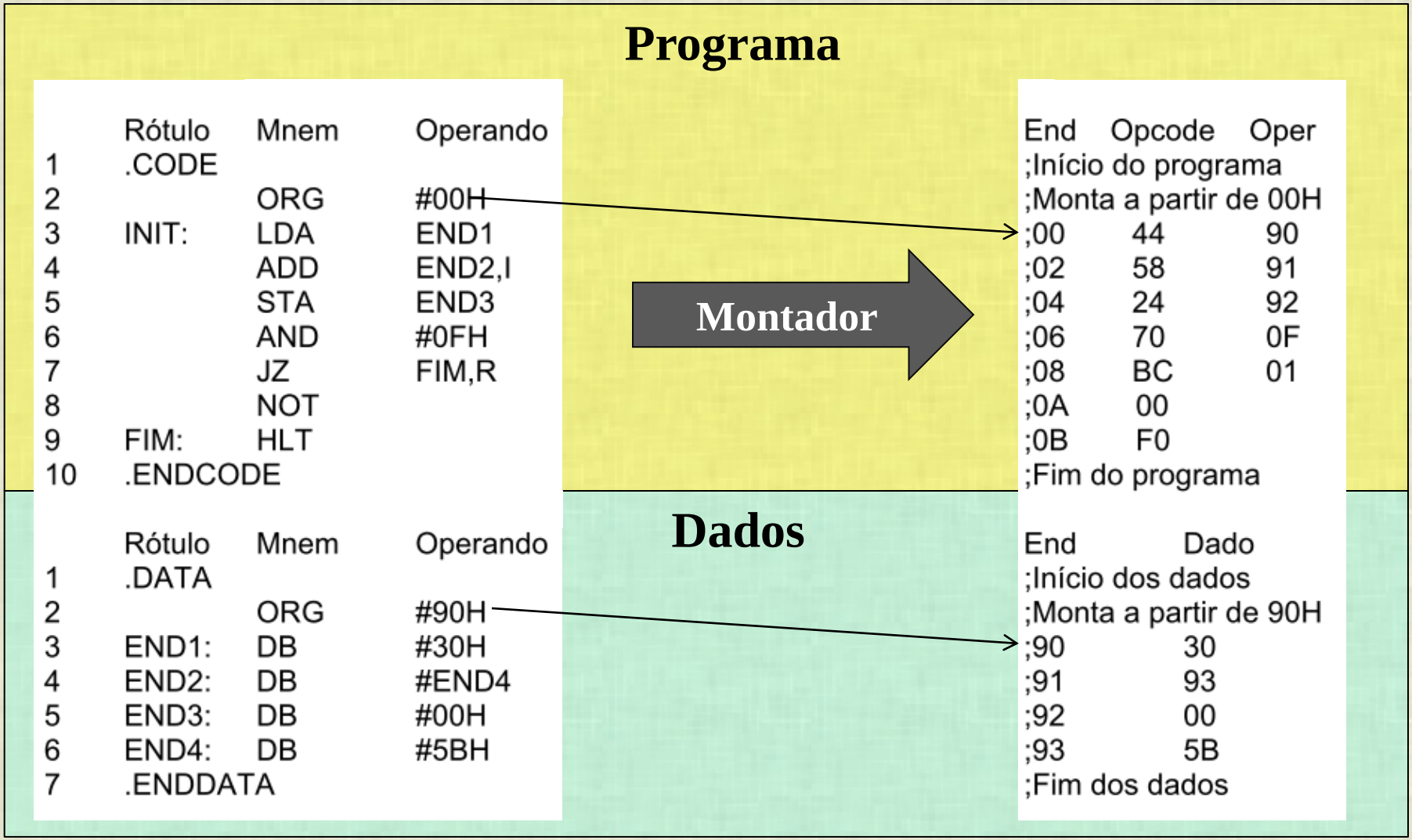
Dados

```
     Rótulo    Mnem      Operando
1    .DATA
2              ORG       #90H
3    END1:     DB        #30H
4    END2:     DB        #END4
5    END3:     DB        #00H
6    END4:     DB        #5BH
7    .ENDDATA
```

início da área de dados (90H)

variáveis

Estrutura de um Programa Assembly Completo



Estrutura de um Programa Assembly Completo

- **.CODE**
  - ORG #00H → início do código na posição 0 de memória
  - ...
- **.ENDCODE:**
  - fim da área de código

- **.DATA**
  - ORG #90H → início das variáveis na posição 90H de memória
  - END1 : DB #30H → inicialização de variáveis
  - END2 : DB #END4 → inicialização de variáveis
  - END3 : DB #00H → inicialização de variáveis
  - END4 : DB #5BH → inicialização de variáveis
- **.ENDDATA**
  - fim da área de dados

Programa Subtração

Linguagem de alto nível

C = A - B

ASSEMBLY DA CLEÓPATRA

```
.code
  lda B
  not
  add #01h
  add A
  sta C
  hlt
.endcode

.data
  A: db #04h
  B: db #01h
  C: db #00h
.enddata
```

AC

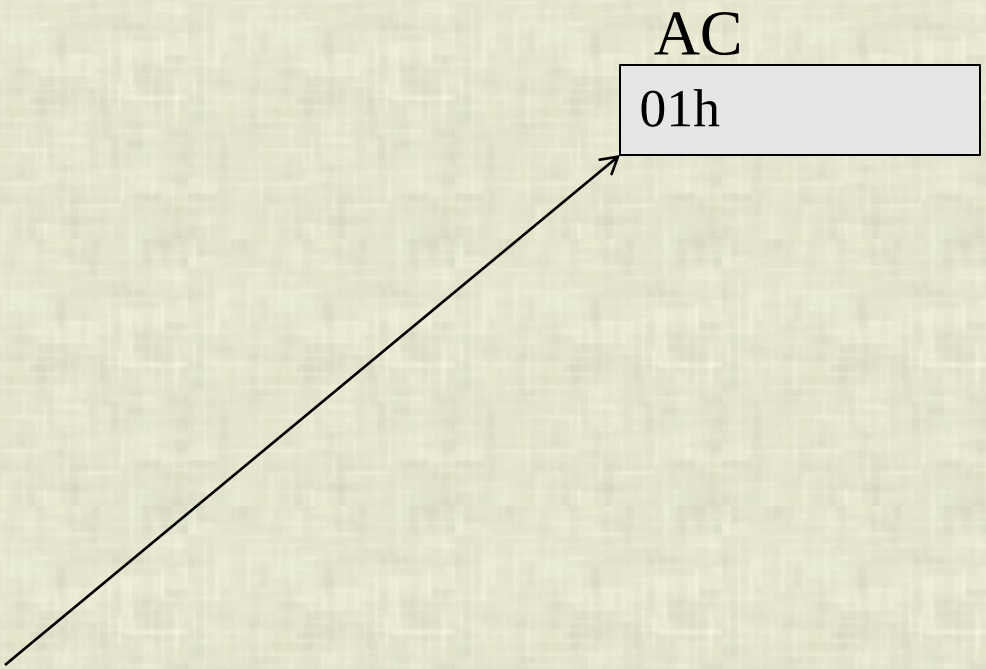
Programa Subtração

Linguagem de alto nível

C = A - B

ASSEMBLY DA CLEÓPATRA

```
.code
  lda B
  not
  add #01h
  add A
  sta C
  hlt
.endcode

.data
  A: db #04h
  B: db #01h
  C: db #00h
.enddata
```



Programa Subtração

Linguagem de alto nível

C = A - B

ASSEMBLY DA CLEÓPATRA

```
.code
  lda B
  not
  add #01h
  add A
  sta C
  hlt
.endcode

.data
  A: db #04h
  B: db #01h
  C: db #00h
.enddata
```

AC

FEh

Programa Subtração

Linguagem de alto nível

C = A - B

ASSEMBLY DA CLEÓPATRA

```
.code
  lda B
  not
  add #01h
  add A
  sta C
  hlt
.endcode

.data
  A: db #04h
  B: db #01h
  C: db #00h
.enddata
```

AC

FFh

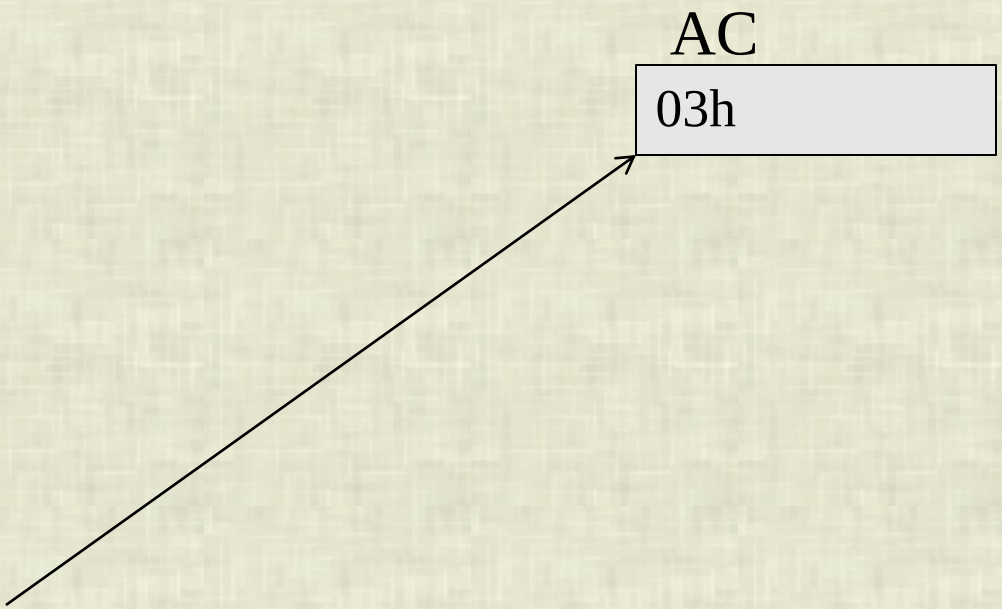
Programa Subtração

Linguagem de alto nível

C = A - B

ASSEMBLY DA CLEÓPATRA

```
.code
  lda B
  not
  add #01h
  add A
  sta C
  hlt
.endcode


.data
  A: db #04h
  B: db #01h
  C: db #00h
.enddata
```



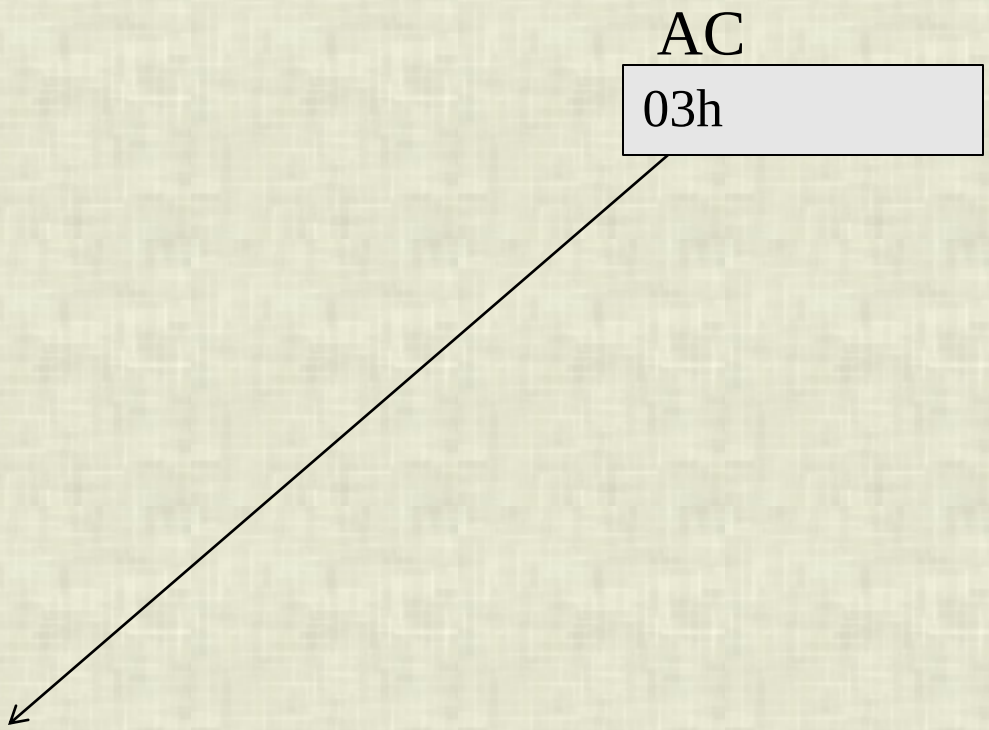
Programa Subtração

Linguagem de alto nível

C = A - B

ASSEMBLY DA CLEÓPATRA

```
.code
  lda B
  not
  add #01h
  add A
  sta C
  hlt
.endcode

.data
  A: db #04h
  B: db #01h
  C: db #03h
.enddata
```



Programa Subtração

Linguagem de alto nível

C = A - B

ASSEMBLY DA CLEÓPATRA

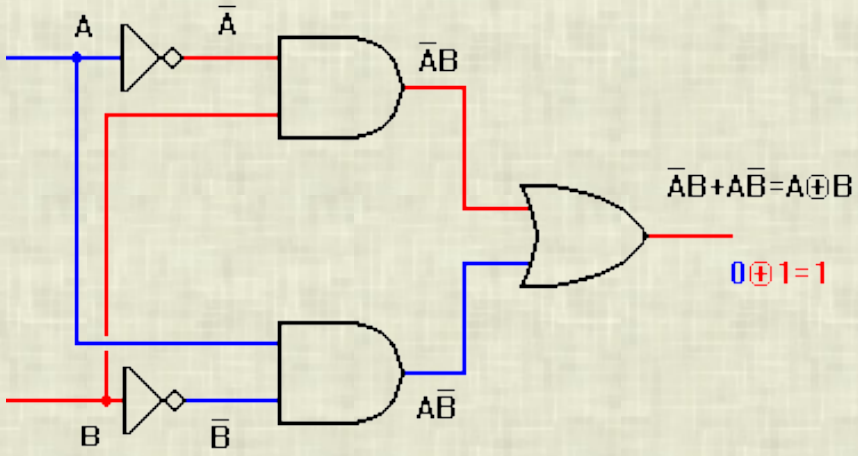
```
.code
  lda B
  not
  add #01h
  add A
  sta C
  hlt
.endcode

.data
  A: db #04h
  B: db #01h
  C: db #03h
.enddata
```

AC

03h

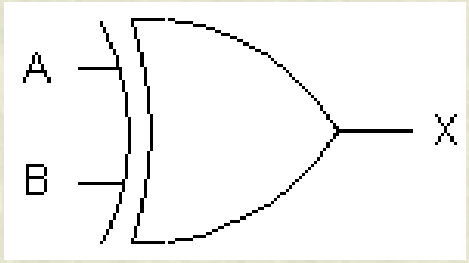
Programa XOR

```
.code
  lda A
  not
  and B
  sta Y
  lda B
  not
  and A
  or Y
  sta Y
  hlt
.endcode
```




```
.data
  A: db #33h
  B: db #91h
  Y: db #00h
.enddata
```

- **Representação gráfica**

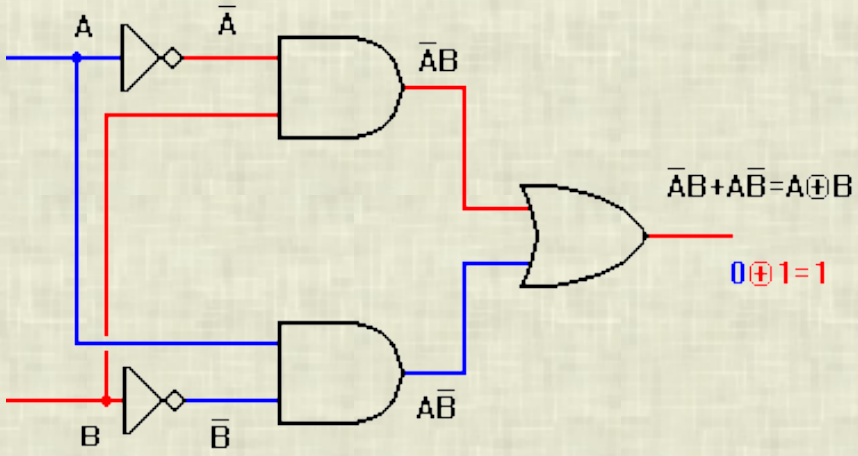


- **Tabela Verdade**

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Programa XOR

```
.code
  lda A
  not
  and B
  sta Y
  lda B
  not
  and A
  or Y
  sta Y
  hlt
.endcode
```




```
.data
  A: db #33h
  B: db #91h
  Y: db #00h
.enddata
```

$Y=A \oplus B$

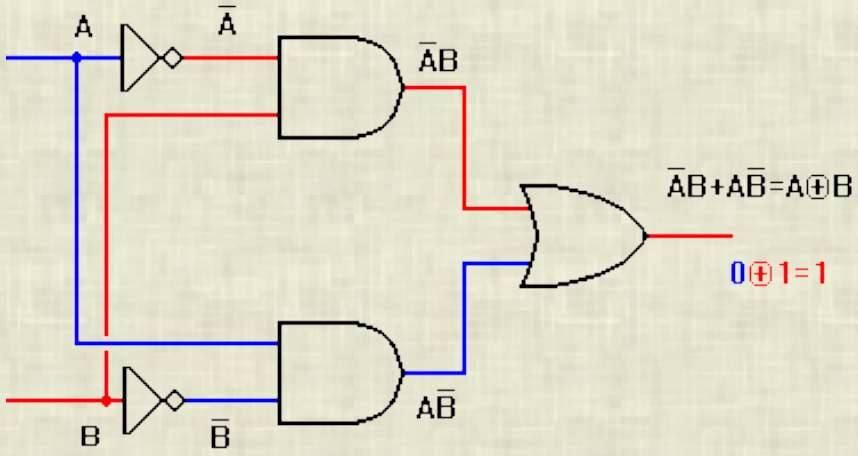
$Y=33h \oplus 91h$

$Y=00110011 \oplus 10010001$

$Y=$?????

Programa XOR

```
.code
  lda A
  not
  and B
  sta Y
  lda B
  not
  and A
  or Y
  sta Y
  hlt
.endcode
```



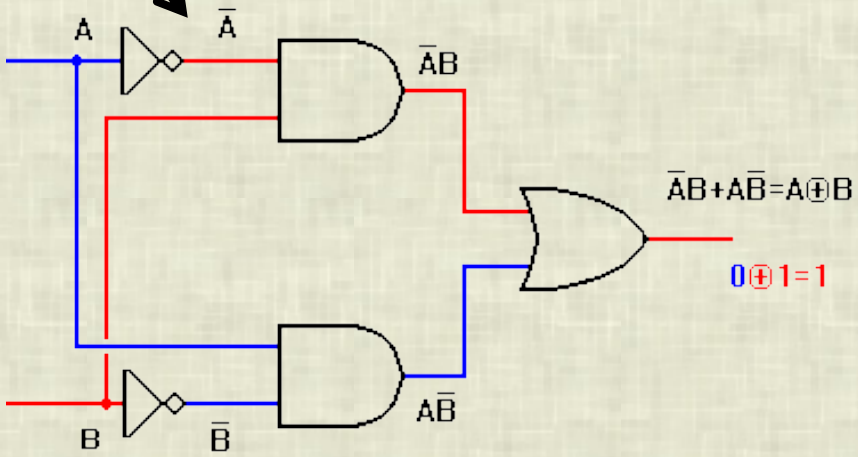

```
.data
  A: db #33h
  B: db #91h
  Y: db #00h
.enddata
```

$$Y=A \oplus B$$
$$Y=33h \oplus 91h$$
$$Y=00110011 \oplus 10010001$$
$$Y= 10100010 = A2h$$

Programa XOR

```
.code
  lda A
  not
  and B
  sta Y
  lda B
  not
  and A
  or Y
  sta Y
  hlt
.endcode
```

!A



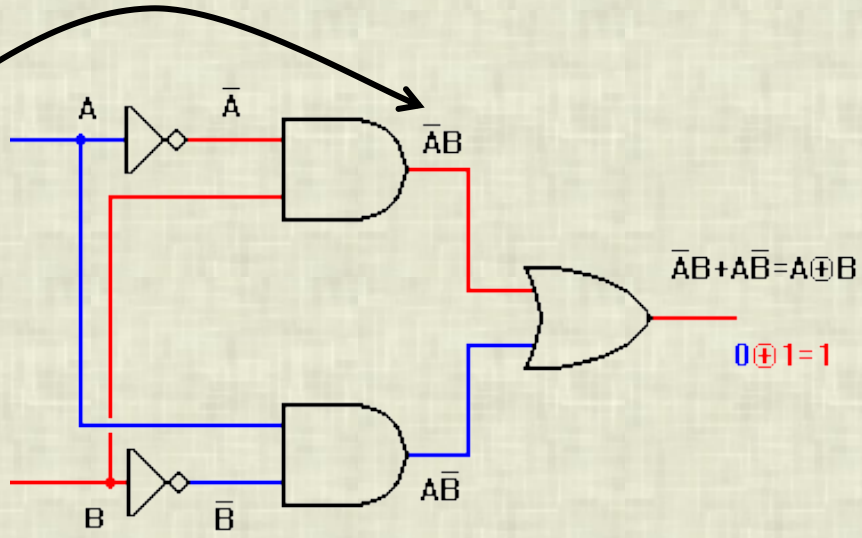

```
.data
  A: db #33h
  B: db #91h
  Y: db #00h
.enddata
```

Programa XOR

```
.code
  lda A
  not
  and B
  sta Y
  lda B
  not
  and A
  or Y
  sta Y
  hlt
.endcode
```

!A

Y=!AB




```
.data
  A: db #33h
  B: db #91h
  Y: db #00h
.enddata
```

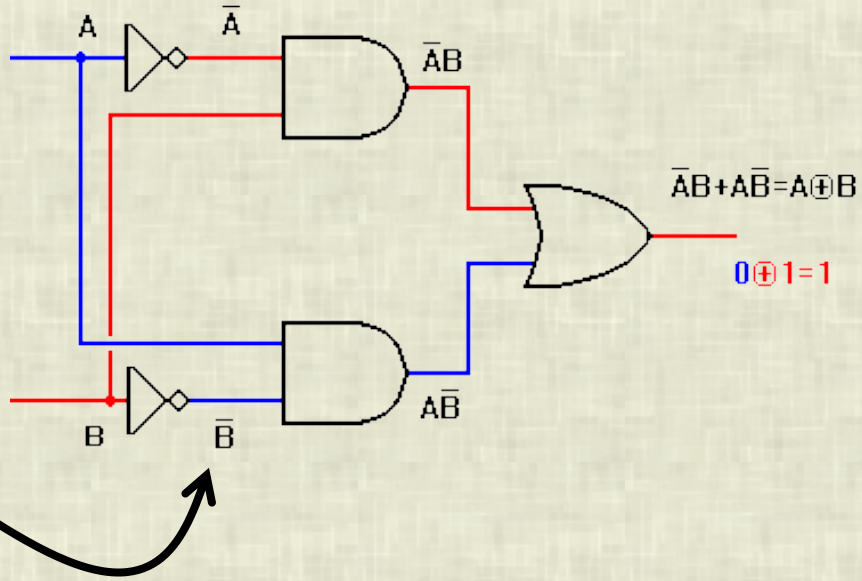
Programa XOR

```
.code
  lda A
  not
  and B
  sta Y
  lda B
  not
  and A
  or Y
  sta Y
  hlt
.endcode
```

!A

!B

Y=!AB




```
.data
  A: db #33h
  B: db #91h
  Y: db #00h
.enddata
```

Programa XOR

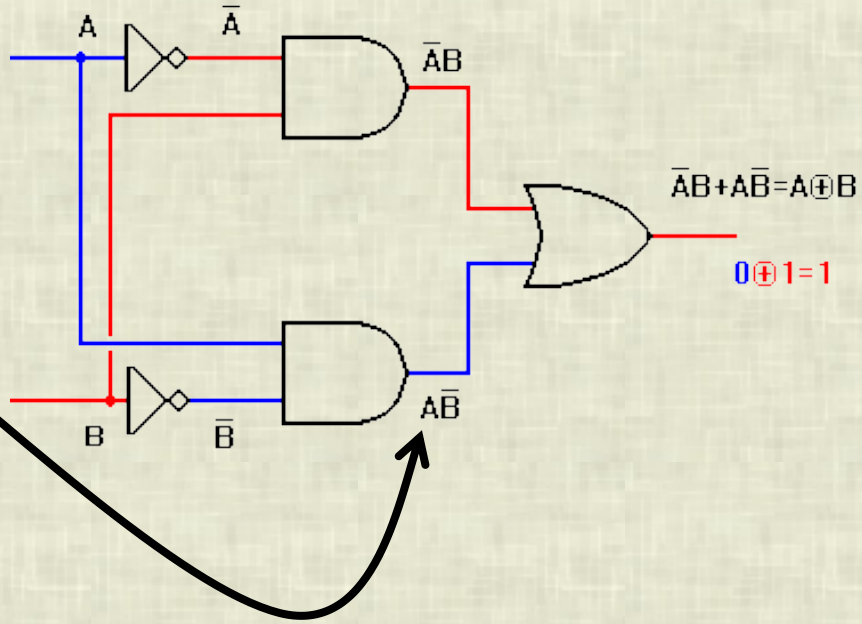
```
.code
  lda A
  not
  and B
  sta Y
  lda B
  not
  and A
  or Y
  sta Y
  hlt
.endcode
```

!A

Y=!AB

!B

AC=A!B




```
.data
  A: db #33h
  B: db #91h
  Y: db #00h
.enddata
```

Programa XOR

```
.code
  lda A
  not
  and B
  sta Y
  lda B
  not
  and A
  or Y
  sta Y
  hlt
.endcode
```

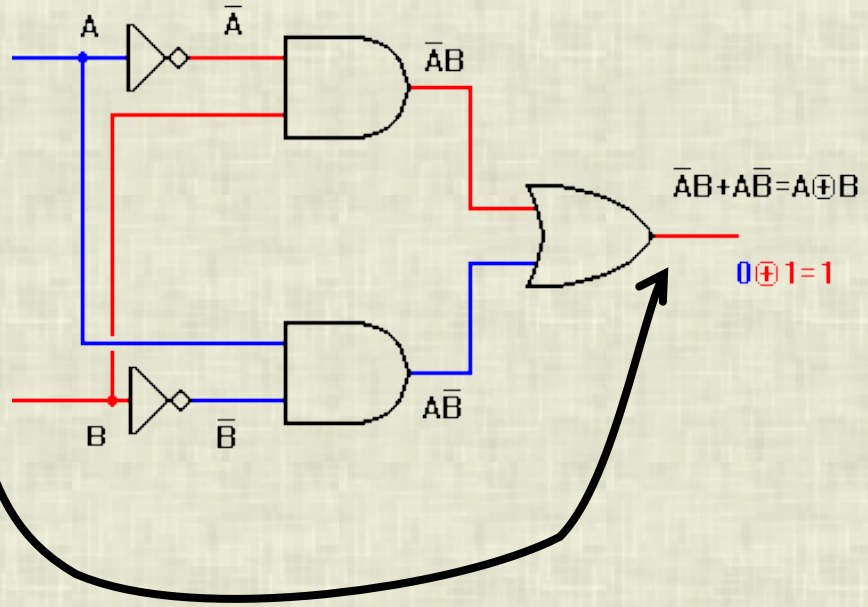
!A

!B

Y=!AB

AC=A!B

AC=AC+Y




```
.data
  A: db #33h
  B: db #91h
  Y: db #00h
.enddata
```

Programa XOR

```
.code
  lda A
  not
  and B
  sta Y
  lda B
  not
  and A
  or Y
  sta Y
  hlt
.endcode
```

!A

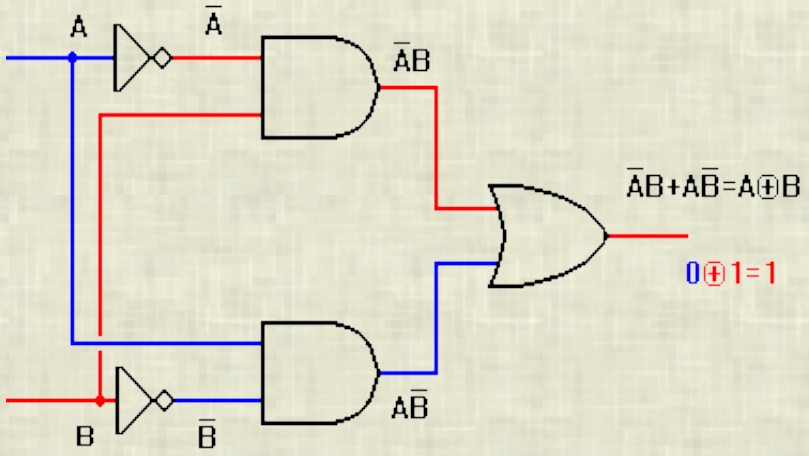
Y=!AB

!B

AC=A!B

AC=AC+Y

Y = AC




```
.data
  A: db #33h
  B: db #91h
  Y: db #00h
.enddata
```

$Y = A \oplus B$

$Y = 00110011 \oplus 10010001$

$Y = 10100010 = A2h$

IF-THEN-ELSE em Assembly

```
.code
  lda in
  jz jp_then
  lda #01h
  jmp end_if
jp_then:
  lda #02h
end_if:
  sta s
  hlt
.endcode

.data
  in: db #01h
  s: db #00h
.enddata
```

-O que faz esse programa ?
-Qual é o valor de ‘s’
ao final do programa ?

IF-THEN-ELSE em Assembly

```
.code
  lda in
  jz jp_then
  lda #01h
  jmp end_if
jp_then:
  lda #02h
end_if:
  sta s
  hlt
.endcode

.data
  in: db #01h
  s: db #00h
.enddata
```

```
if in == 0 then
   s = 2
else
   s = 1
```

LAÇO em Assembly

```
.code
  lda cnt
inicio_laco:
  jz fim_laco
  lda s
  add s
  sta s
  lda cnt
  add #0ffh
  sta cnt
  jmp inicio_laco
fim_laco:
  hlt
.endcode

.data
  cnt: db #05h
  s: db #01h
.enddata
```

-O que faz esse programa ?
-Qual é o valor de ‘s’
ao final do programa ?

LAÇO em Assembly

```
.code
  lda cnt
inicio_laco:
  jz fim_laco
  lda s
  add s
  sta s
  lda cnt
  add #0ffh
  sta cnt
  jmp inicio_laco
fim_laco:
  hlt
.endcode

.data
  cnt: db #05h
  s: db #01h
.enddata
```

```
int cnt = 5;
int s = 1;
while(cnt!=0){
    s = s + s
    cnt--;
}
```

s == 32

Exercício em Sala de Aula

- **Traduza os seguinte trechos de código C em Assembly (Cleo):**

1.
```
if (a<b) then
   s--;
else
   s++;
```

2.
```
for(i=0;i<5;i++)
   s = s + 3;
```

3.
```
for(i=0;i<=5;i++)
   s = s + 3;
```

4.
```
for(i=5;i>=0;i--)
   s = s + 3;
```

Resumo

- **Descrever em Assembly estruturas tais como**
  - If-then-else, laço, programa completo

O trabalho sobre assembly terá exercícios semelhantes aos desta aula !!!