ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

Computador Cleópatra

Interface Hardware e Software

Alexandre Amory
Edson Moreno

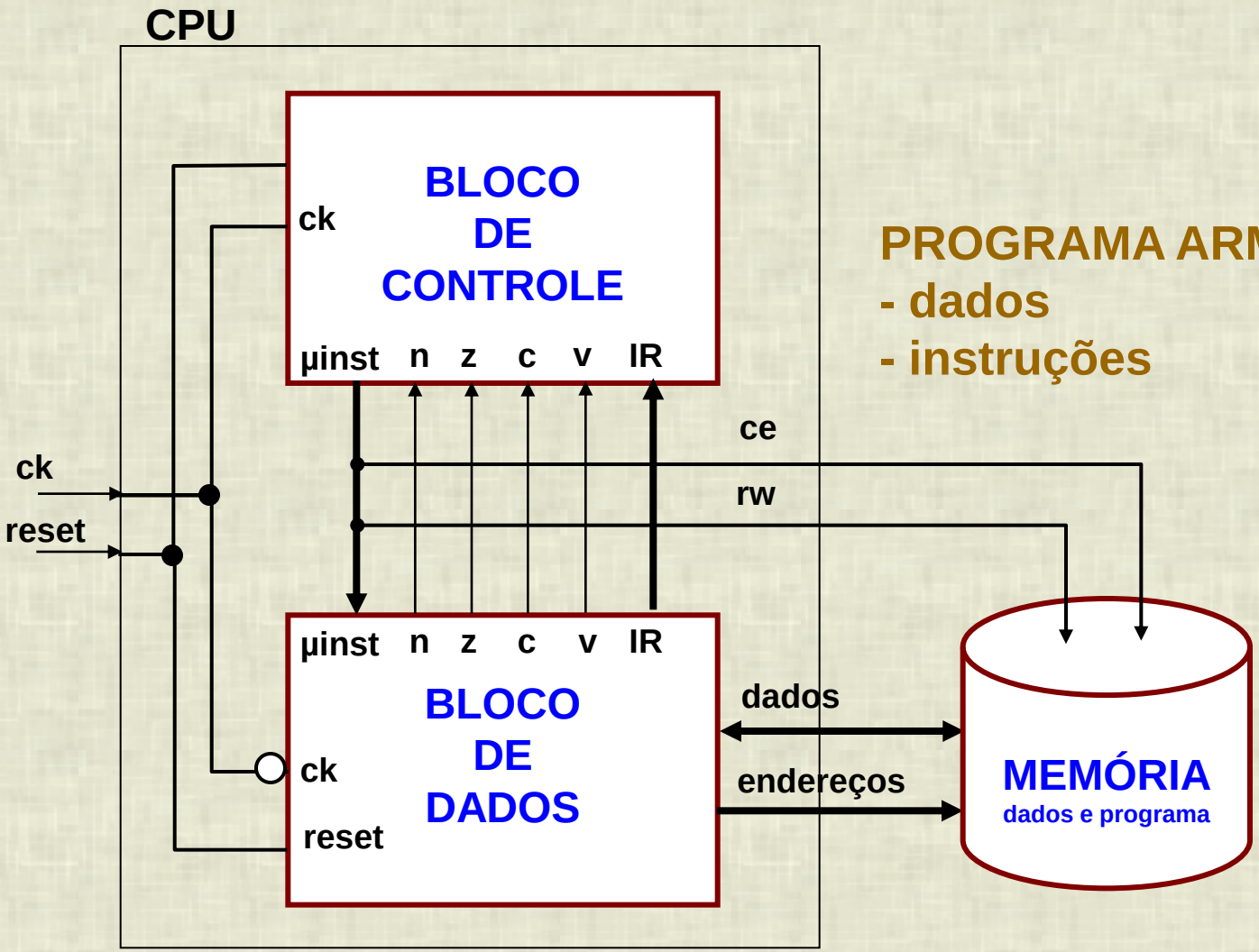
Nas Aulas Anteriores …

- **Vimos como descrever e implementar circuitos combinacionais**
- **Como descrever a ULA da Cleo usando lógica combinacional**
- **Vimos como descrever e implementar circuitos sequenciais e máquinas de estados**

Na Aula de Hoje …

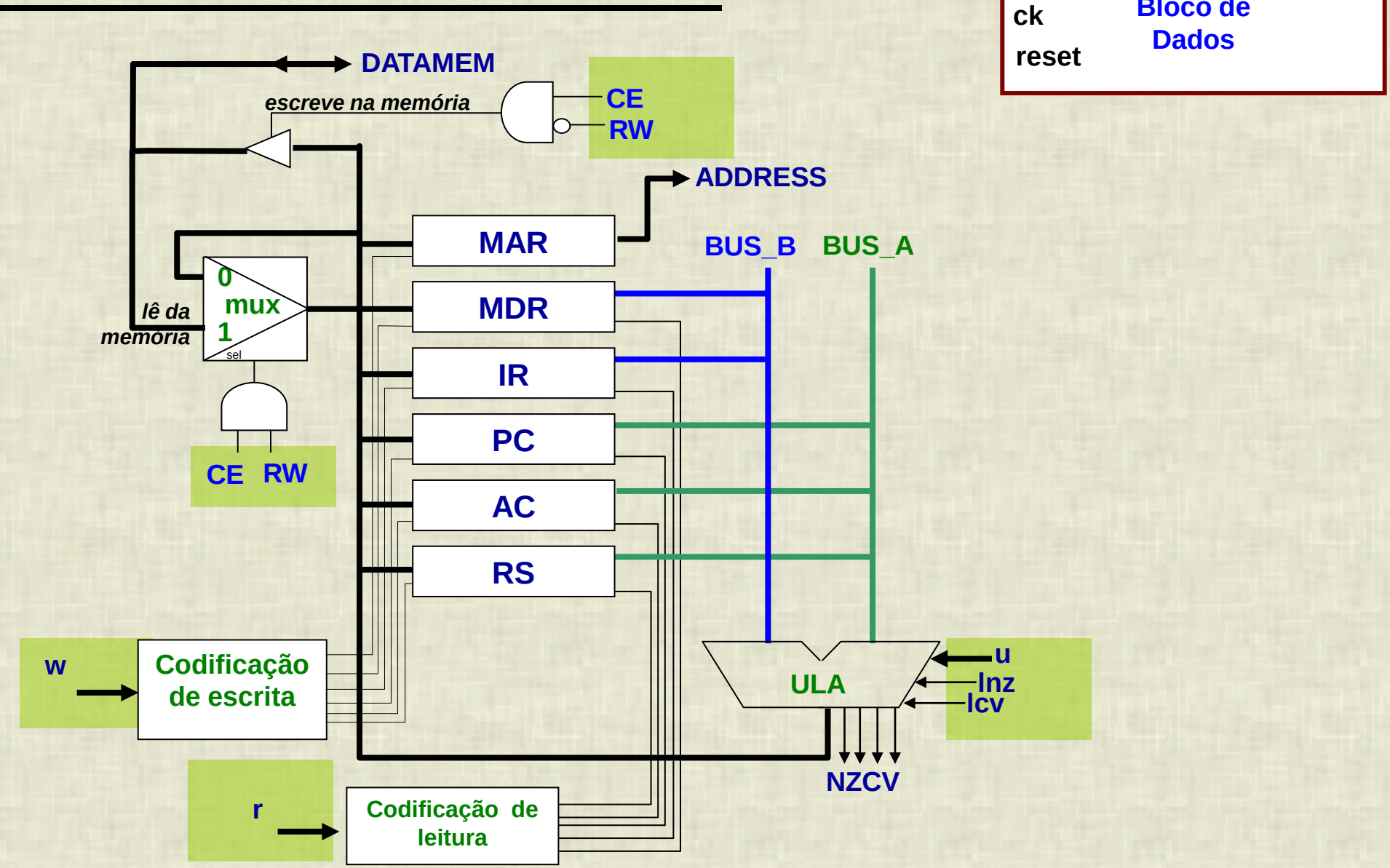
- **Retomamos a Arquitetura Cleópatra**
- **Foco no fluxo de dados/instruções internamente ao processador**
- **Como as instruções Assembly são executadas**
  - Relação: instrução+ME com a máquina de estados e ULA

Arquitetura Cleópatra - Von Neumann

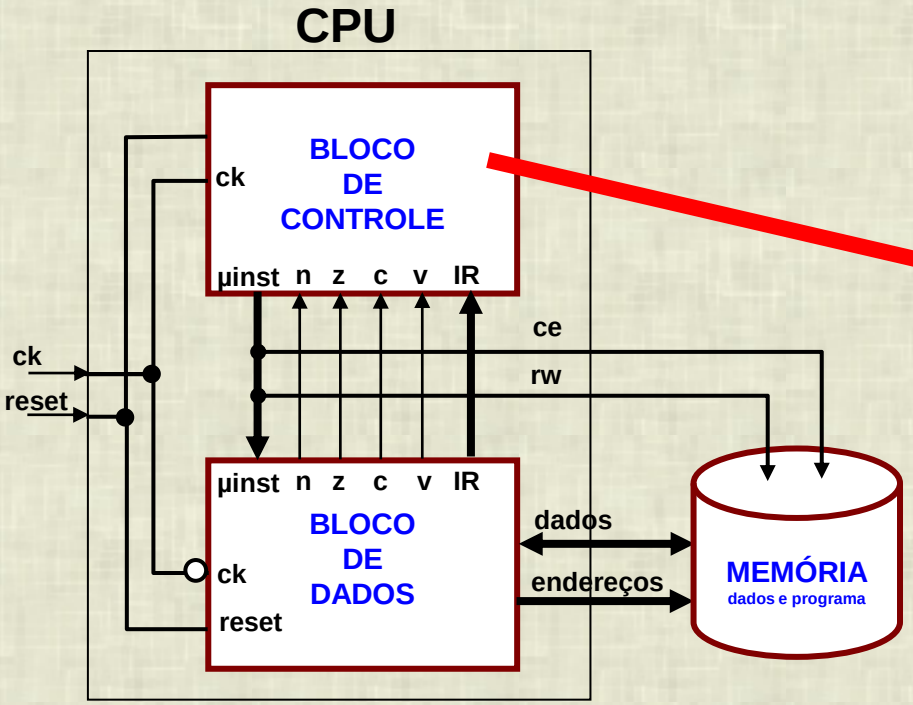


PROGRAMA ARMAZENADO
- dados
- instruções

Bloco de Dados



Bloco de Controle



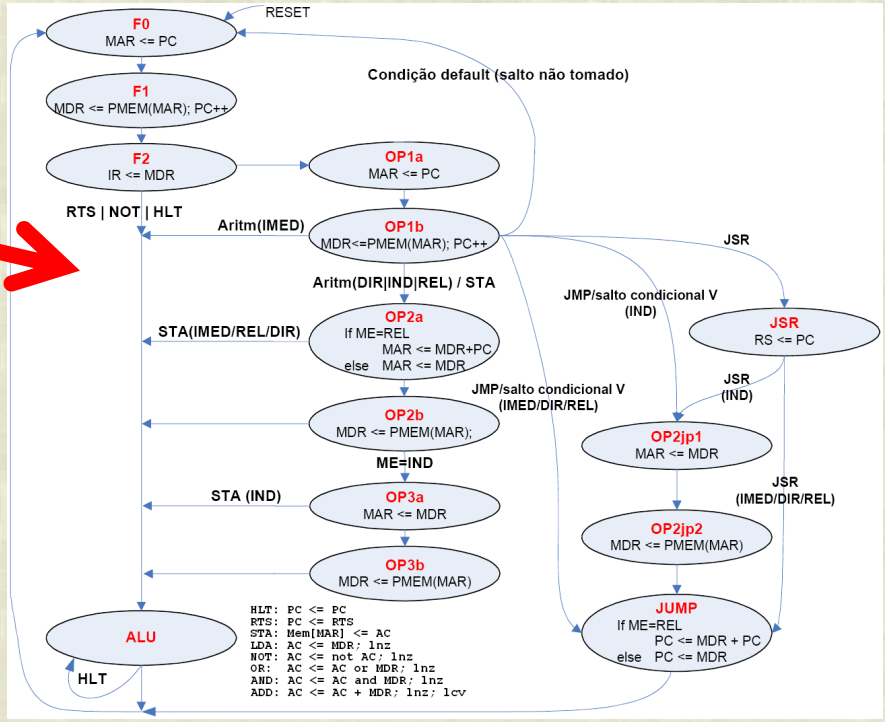
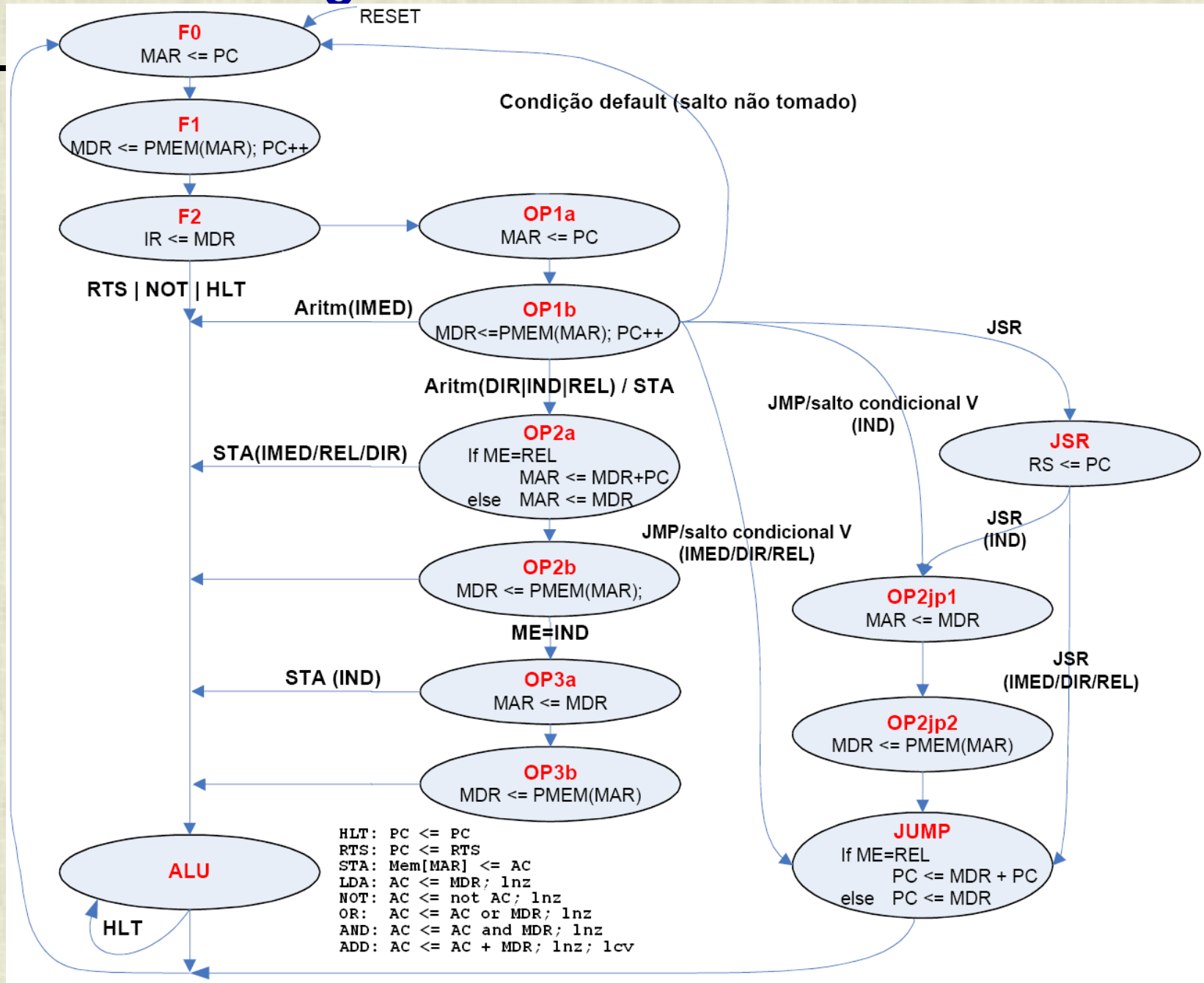
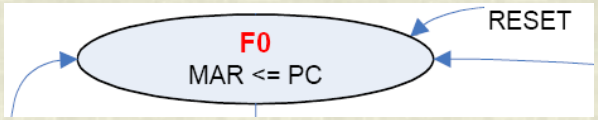
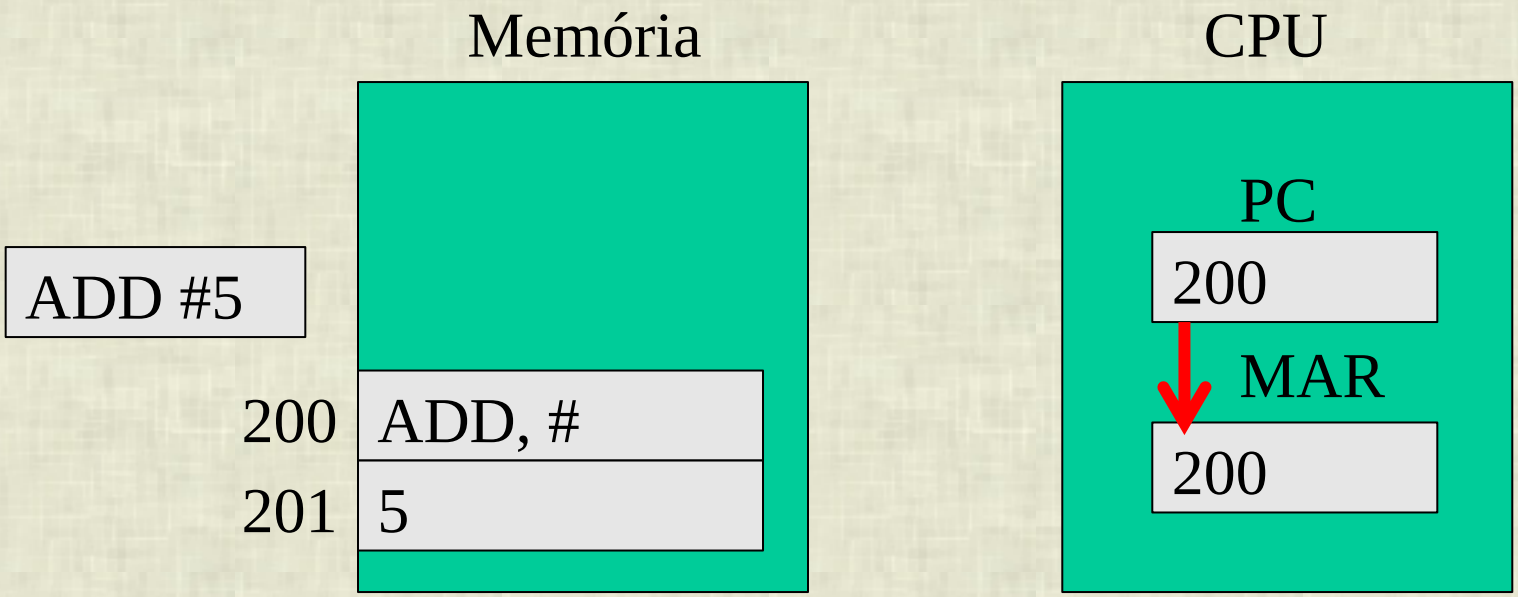


Diagrama de Estados da Cleo

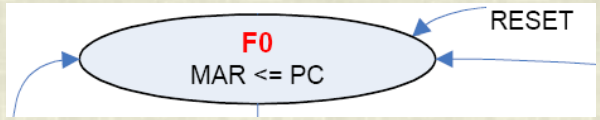


Cleo: Carrega Endereço da Instrução



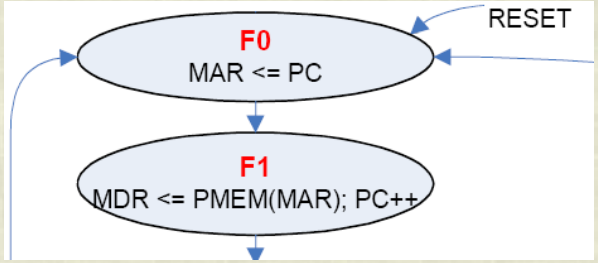


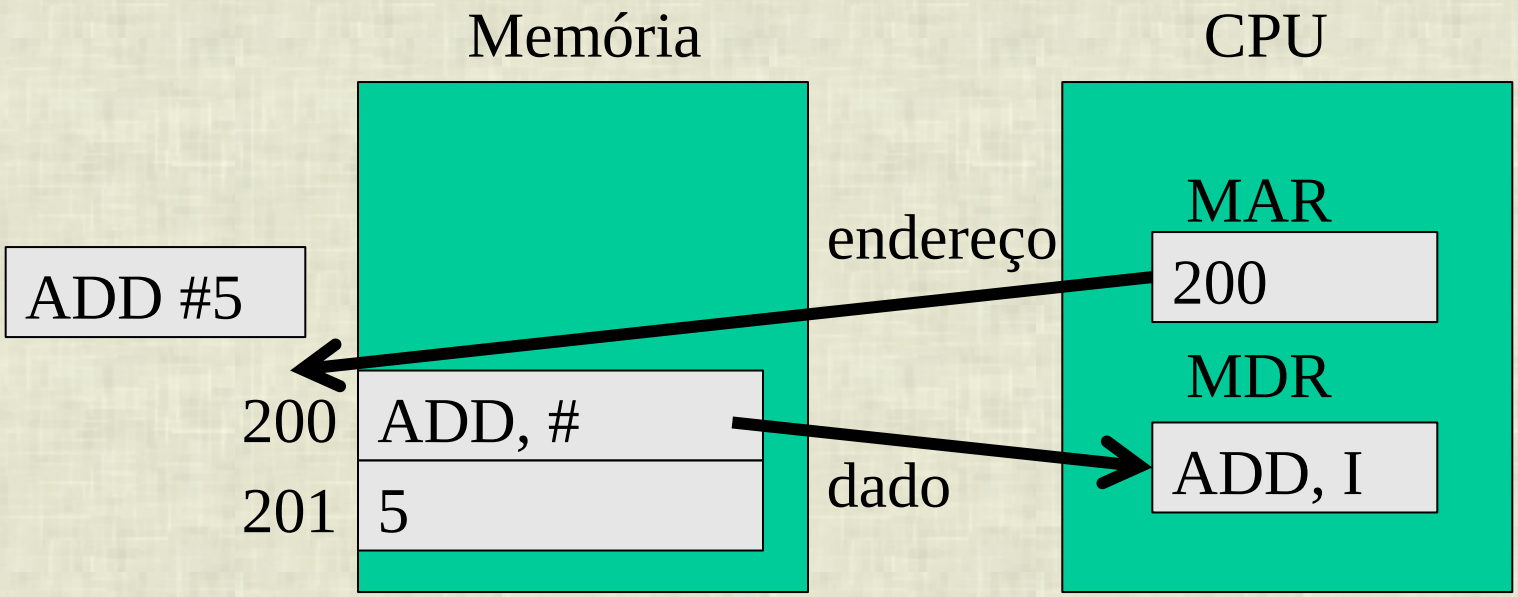
Cleo: Carrega Endereço da Instrução



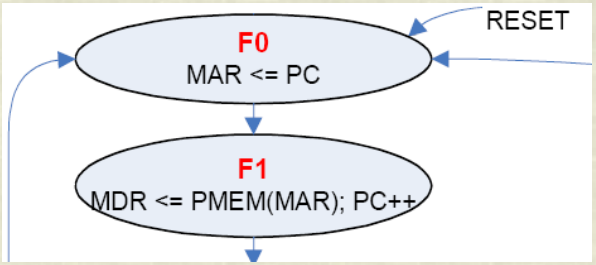
DATAMEM
escreve na memória
CE
RW
ADDRESS
MAR
BUS_B
BUS_A
lê da memória
0
mux
1
sel
MDR
IR
PC
AC
RS
CE
RW
w
Codificação de escrita
ULA
u
lnz
lcv
NZCV
r
Codificação de leitura

Cleo: Lê Instrução da Memória



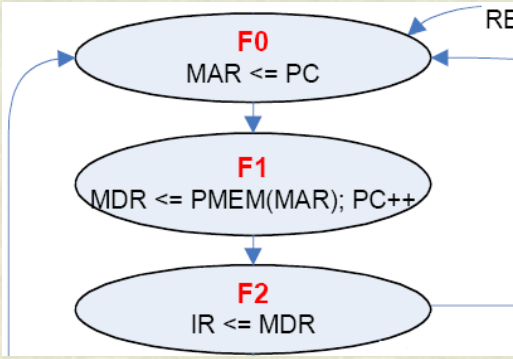


Cleo: Lê Instrução da Memória



DATAMEM
escreve na memória
CE
RW
ADDRESS
MAR
BUS_B
BUS_A
lê da memória
0
mux
1
sel
MDR
IR
PC
AC
RS
CE
RW
w
Codificação de escrita
ULA
u
lnz
lcv
NZCV
r
Codificação de leitura

Cleo: Decodifica Instrução



Memória

CPU

ADD #5

200 ADD, #

201 5

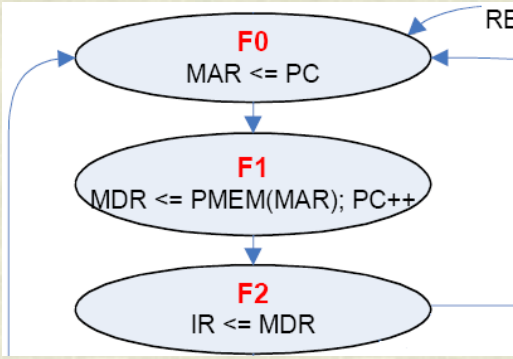
IR

ADD, I

MDR

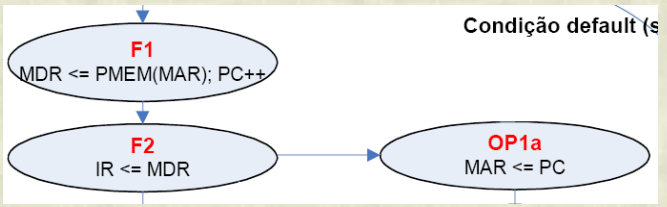
ADD, I

Cleo: Decodifica Instrução



DATAMEM
escreve na memória
CE
RW
ADDRESS
MAR
BUS_B
BUS_A
lê da memória
0
mux
1
sel
MDR
IR
PC
AC
RS
CE
RW
w
Codificação de escrita
ULA
u
lnz
lcv
NZCV
r
Codificação de leitura

Cleo: Inicia Busca do Operando



Memória

CPU

ADD #5

200	ADD, #
201	5

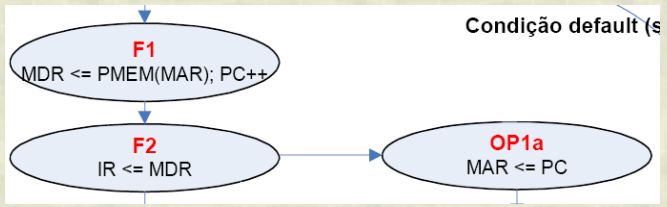
PC

201

MAR

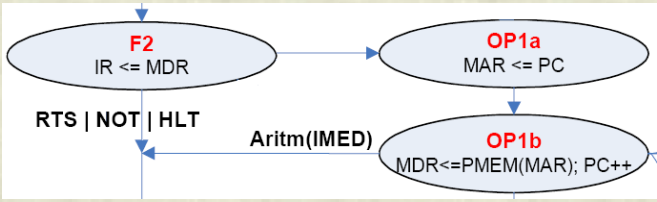
201

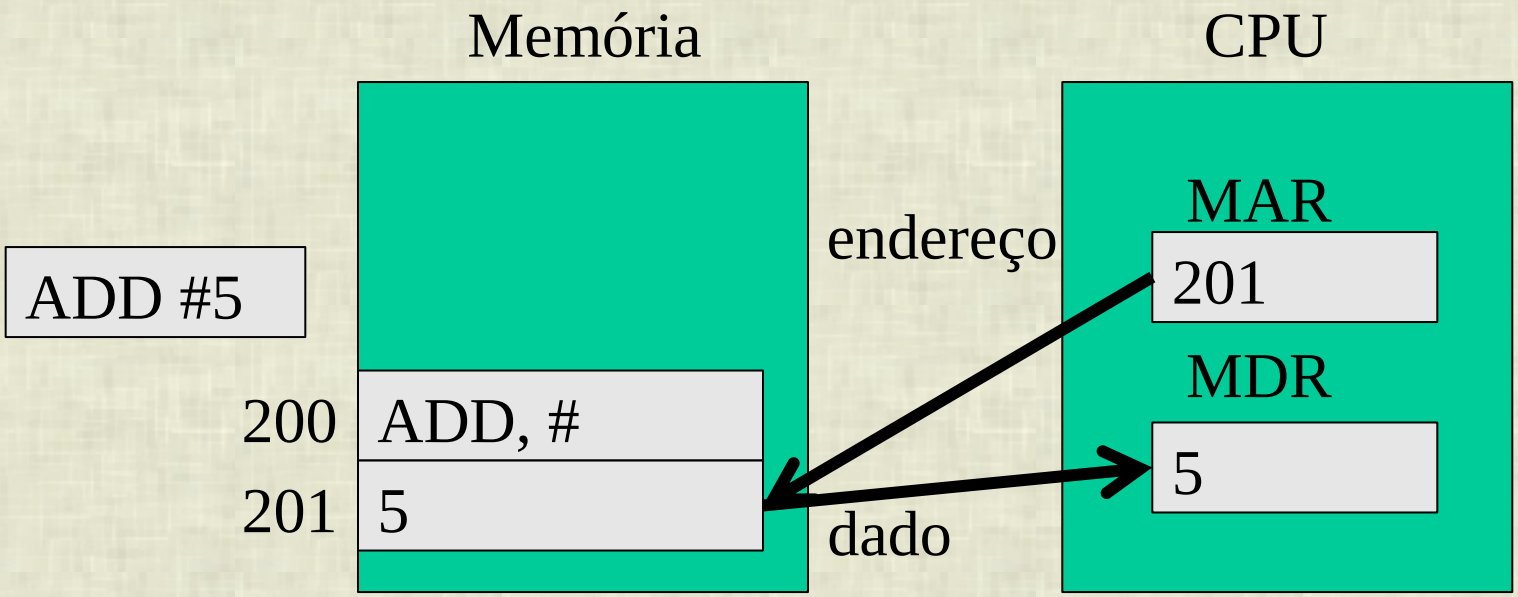
Cleo: Inicia Busca do Operando



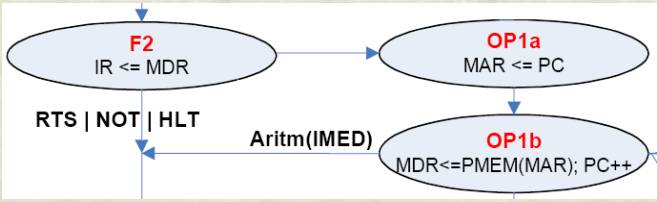
DATAMEM
escreve na memória
CE
RW
ADDRESS
MAR
BUS_B
BUS_A
0
mux
1
sel
lê da memória
MDR
IR
PC
AC
RS
CE
RW
w
Codificação de escrita
ULA
u
lnz
lcv
NZCV
r
Codificação de leitura

Cleo: Leitura do Operando



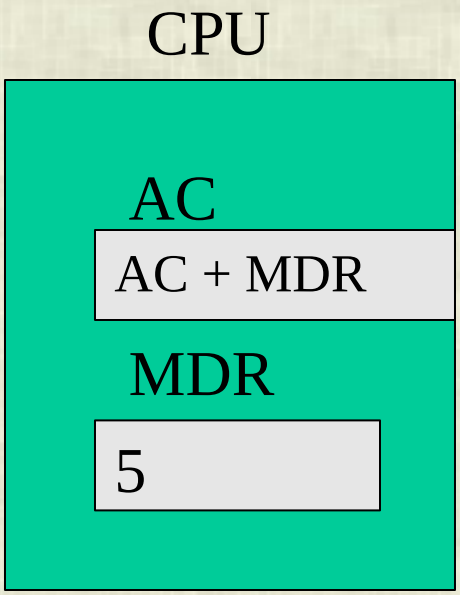


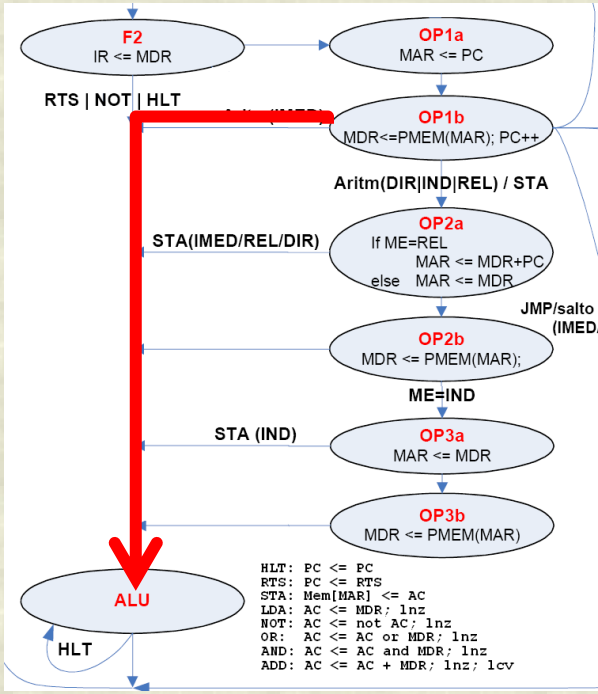
Cleo: Leitura do Operando



DATAMEM
escreve na memória
CE
RW
ADDRESS
MAR
BUS_B
BUS_A
lê da memória
0
mux
1
sel
MDR
IR
PC
AC
RS
CE
RW
w
Codificação de escrita
ULA
u
lnz
lcv
NZCV
r
Codificação de leitura

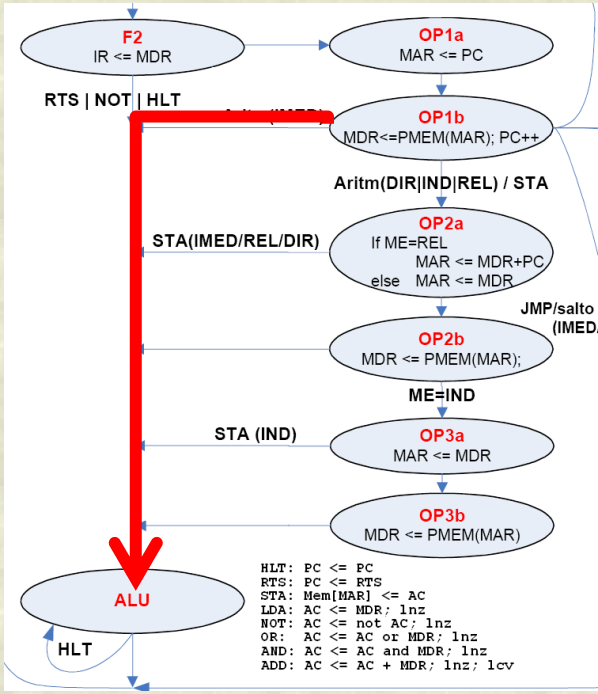
Cleo: Realiza a Operação



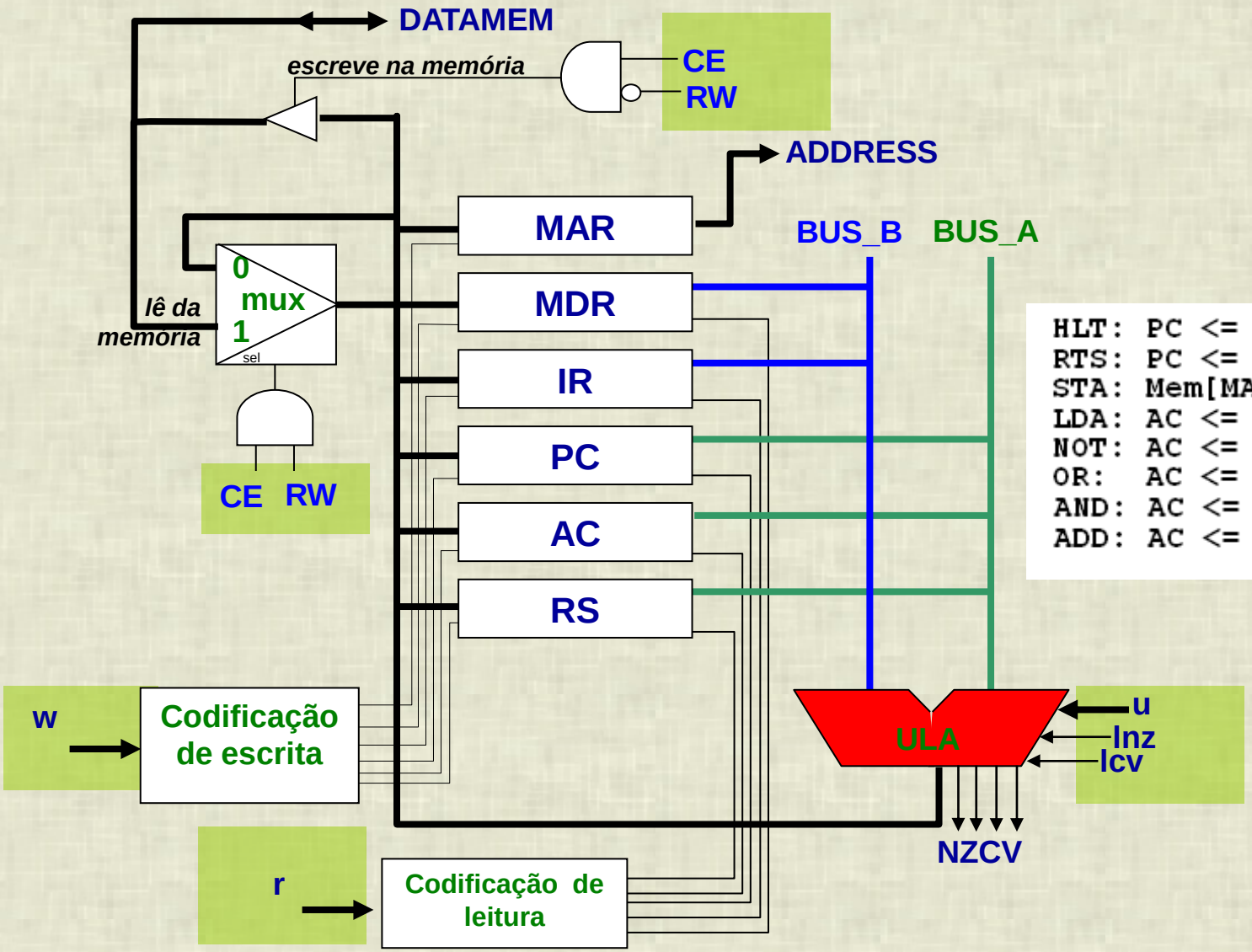


Cleo: ADD Imediato

Exemplo:
ADD #5

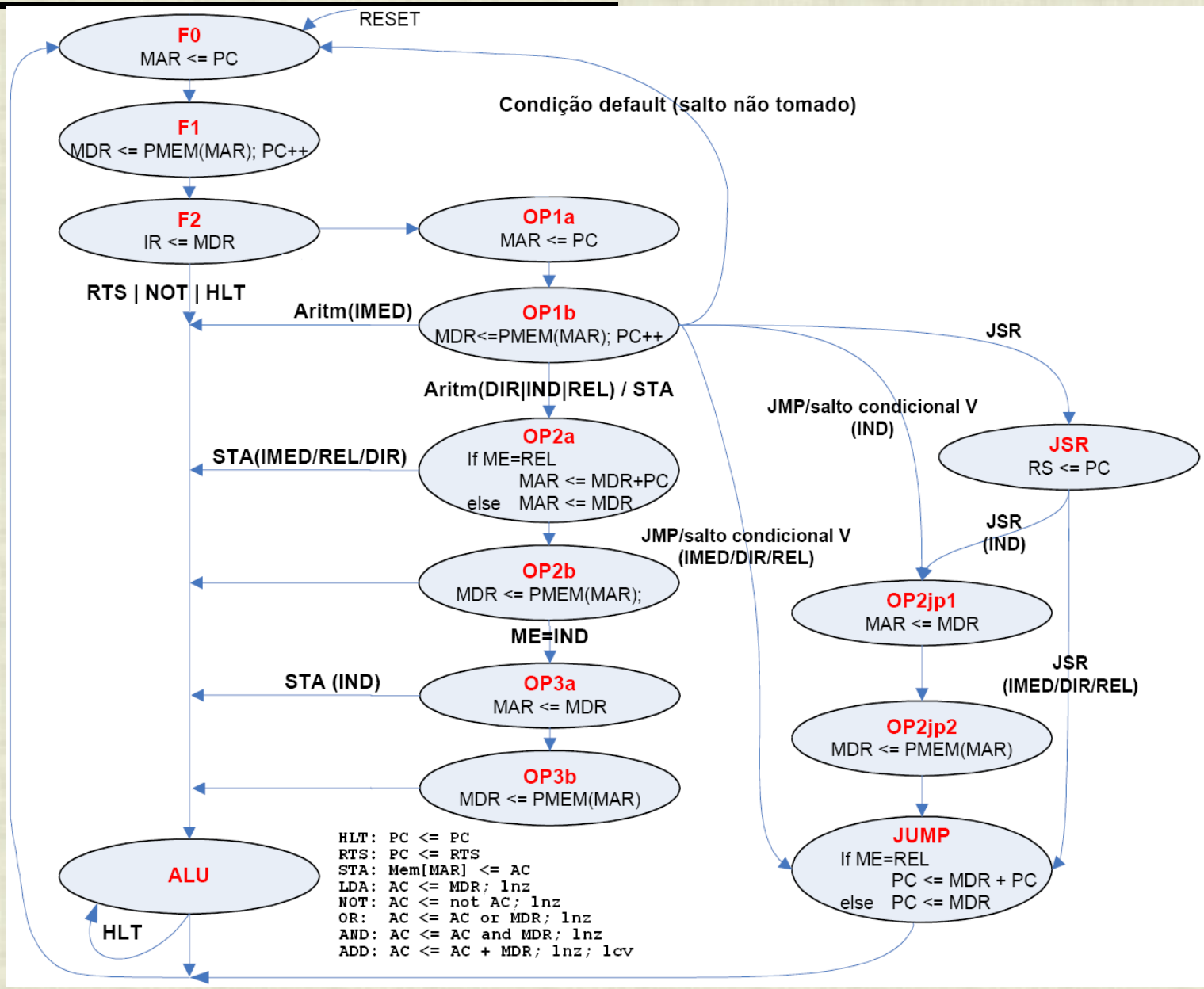


Datapath Cleo – ULA




```
HLT: PC <= PC
RTS: PC <= RTS
STA: Mem[MAR] <= AC
LDA: AC <= MDR; lnz
NOT: AC <= not AC; lnz
OR:  AC <= AC or MDR; lnz
AND: AC <= AC and MDR; lnz
ADD: AC <= AC + MDR; lnz; lcv
```

Bloco de Controle - Máquina de Estados Finita



Modo de Endereçamento

- **Imediato**
  - Operando é o próprio dado
  - Usado para representar constantes
    - Ex.: Número **45** do trecho de código C

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

*p = entrada + 45;
```

- **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

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

Meu Primeiro Prog Assembly

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

Carrega a variável A no reg interno

Meu Primeiro Prog Assembly

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

Soma o conteúdo do reg interno (A) com a constante 3. o resultado é salvo no reg interno. Utiliza Modo Imediato.

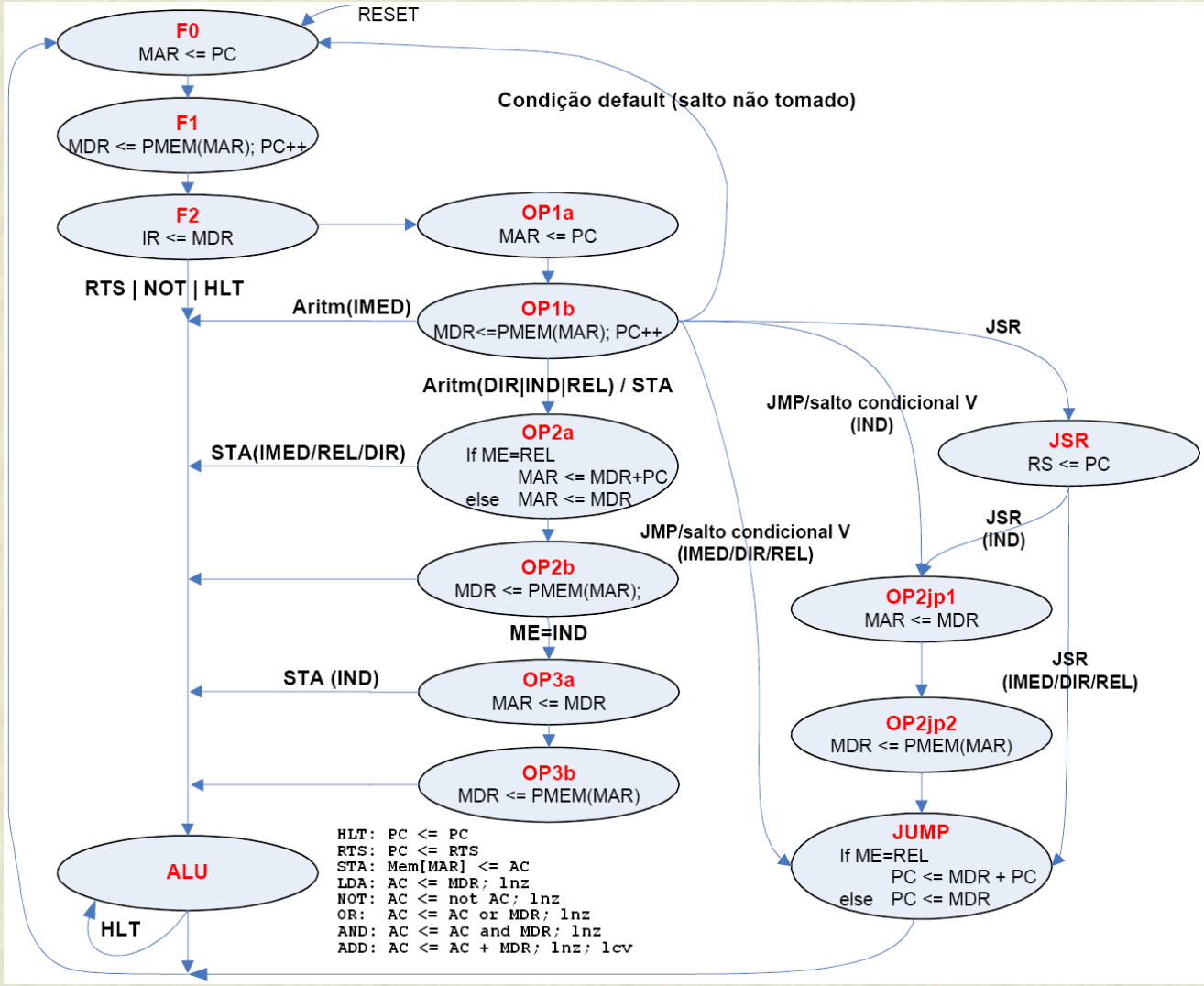
Meu Primeiro Prog Assembly

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

Carrega conteúdo do reg interno (A+3) na variável C

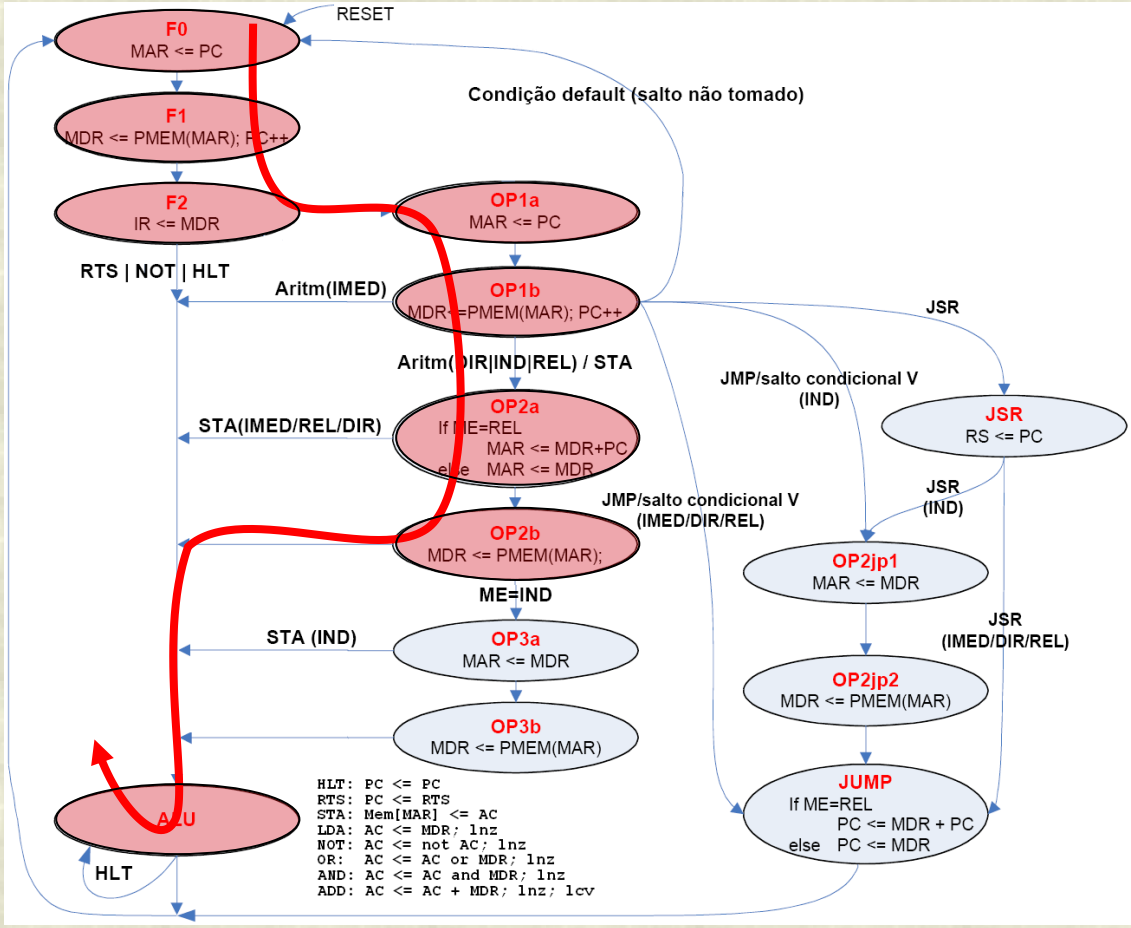
Meu Primeiro Prog Assembly

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



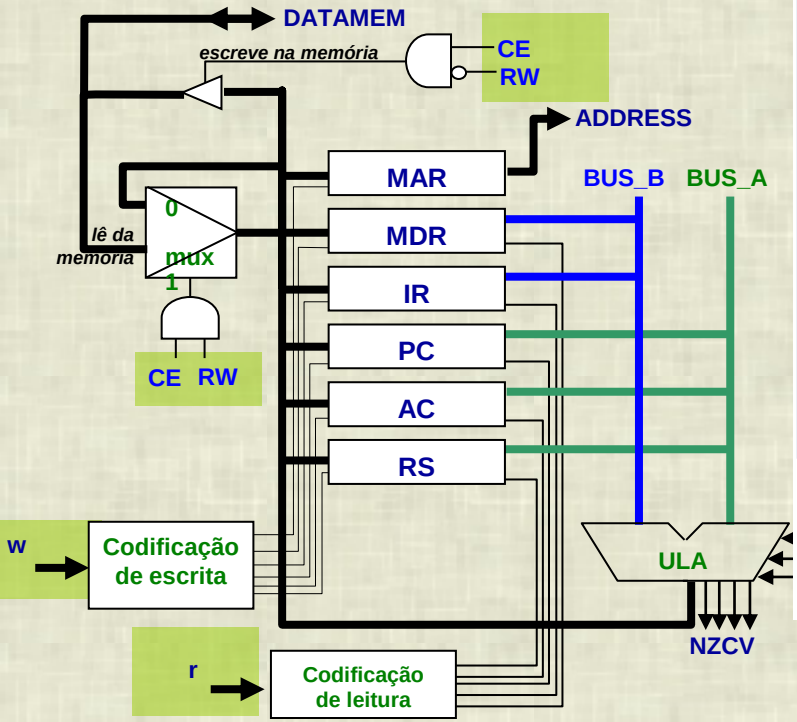
Meu Primeiro Prog Assembly

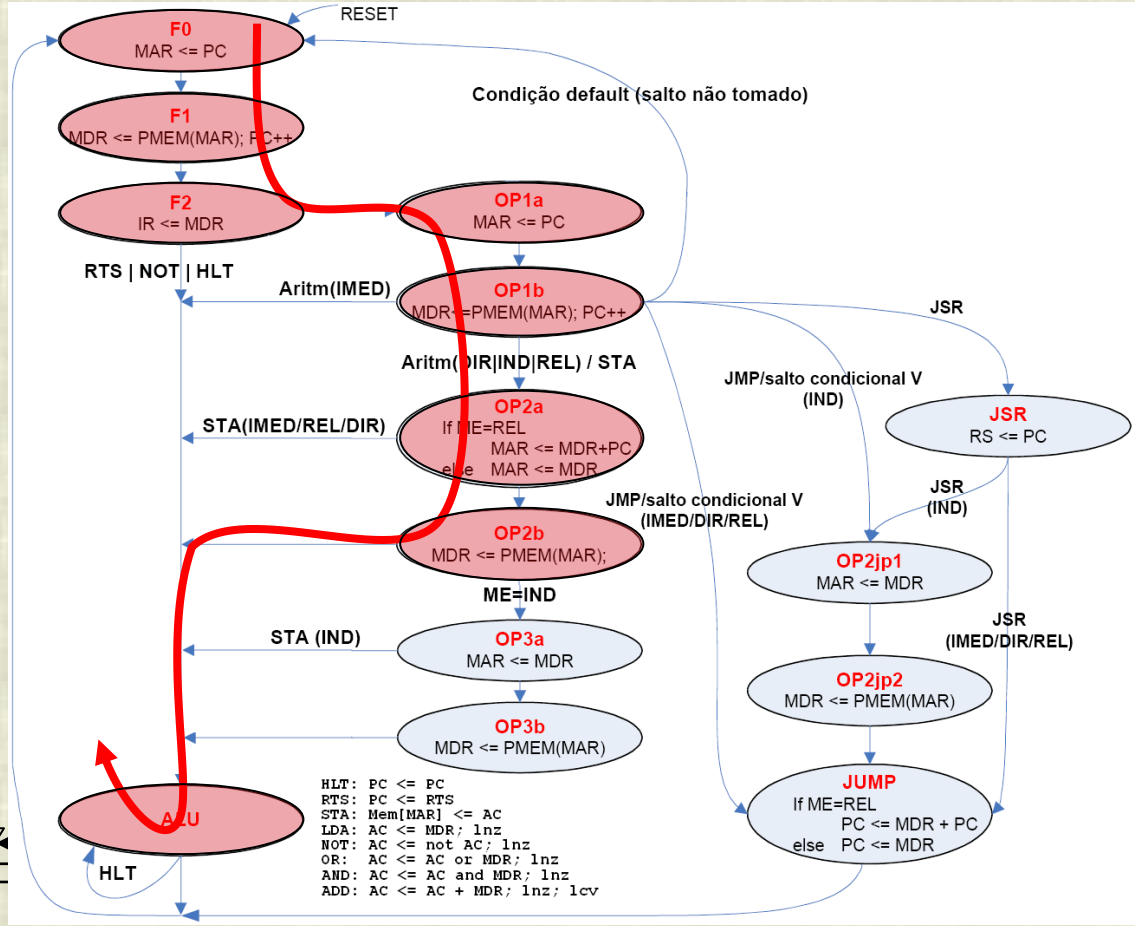
- **Linguagem alto nível (e.g. C)**
  - *C = A+3
- **Assembly da Cleo**
  - LDA A
  - ADD #3
  - STA C,I



Meu Primeiro Prog Assembly

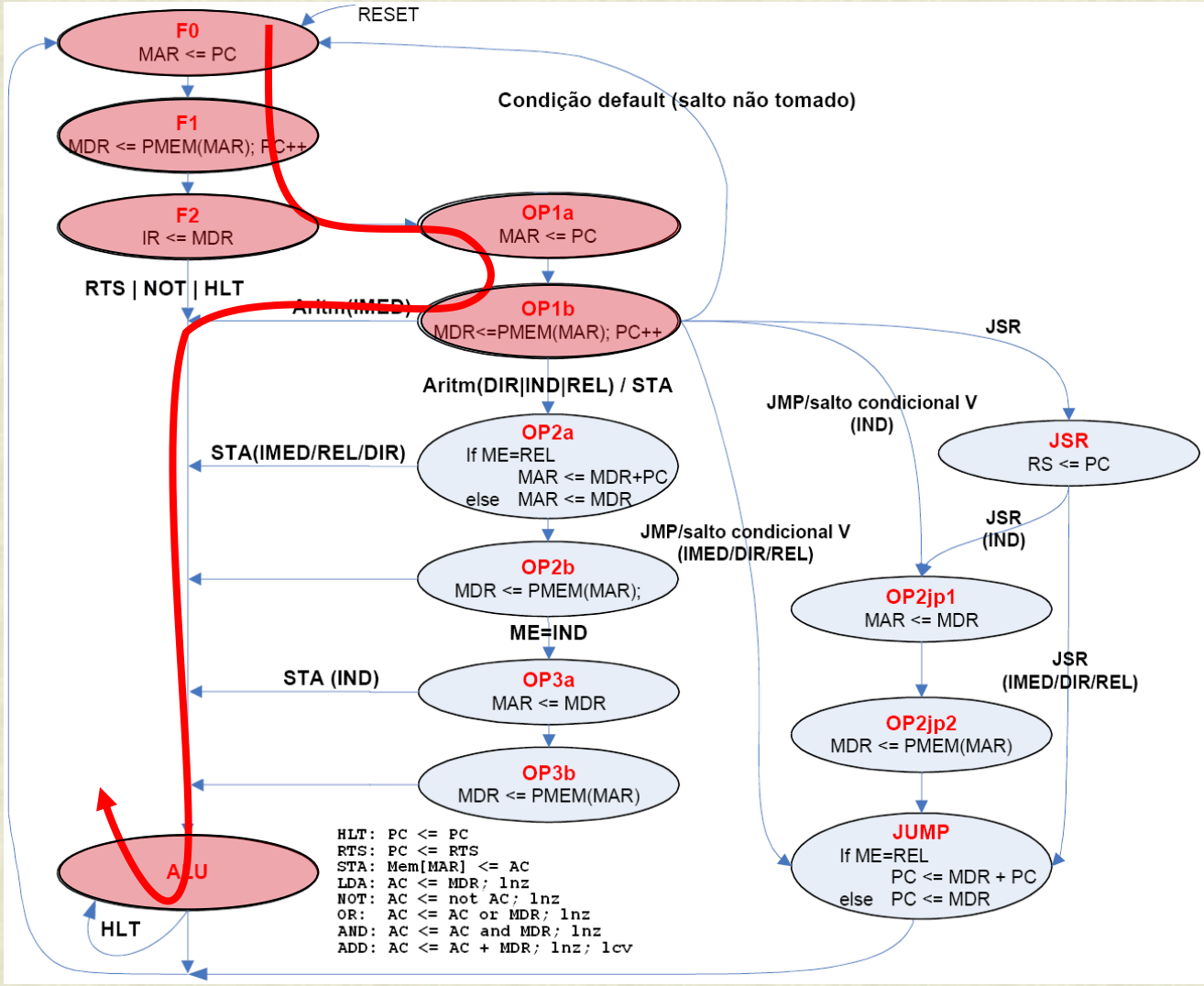
- **Linguagem alto nível (e.g. C)**
  - *C = A+3
- **Assembly da Cleo**
  - LDA A
  - ADD #3
  - STA C,I





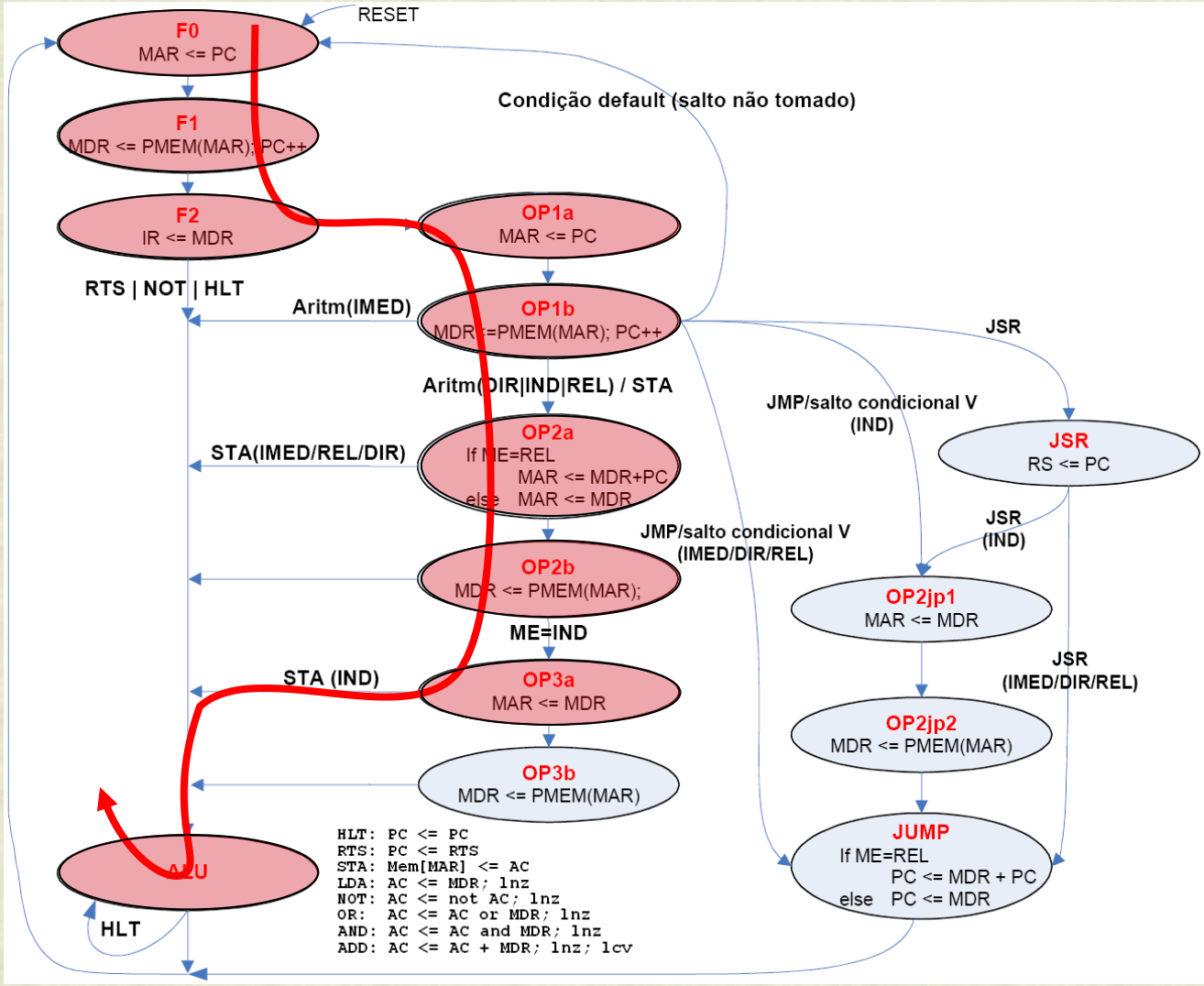
Meu Primeiro Prog Assembly

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



Meu Primeiro Prog Assembly

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



Resumo

- **Como um programa Assembly é executado**
  - O que acontece dentro do processador durante a execução de uma instrução