Uma Arquitetura Concreta para a Máquina de von Neumann

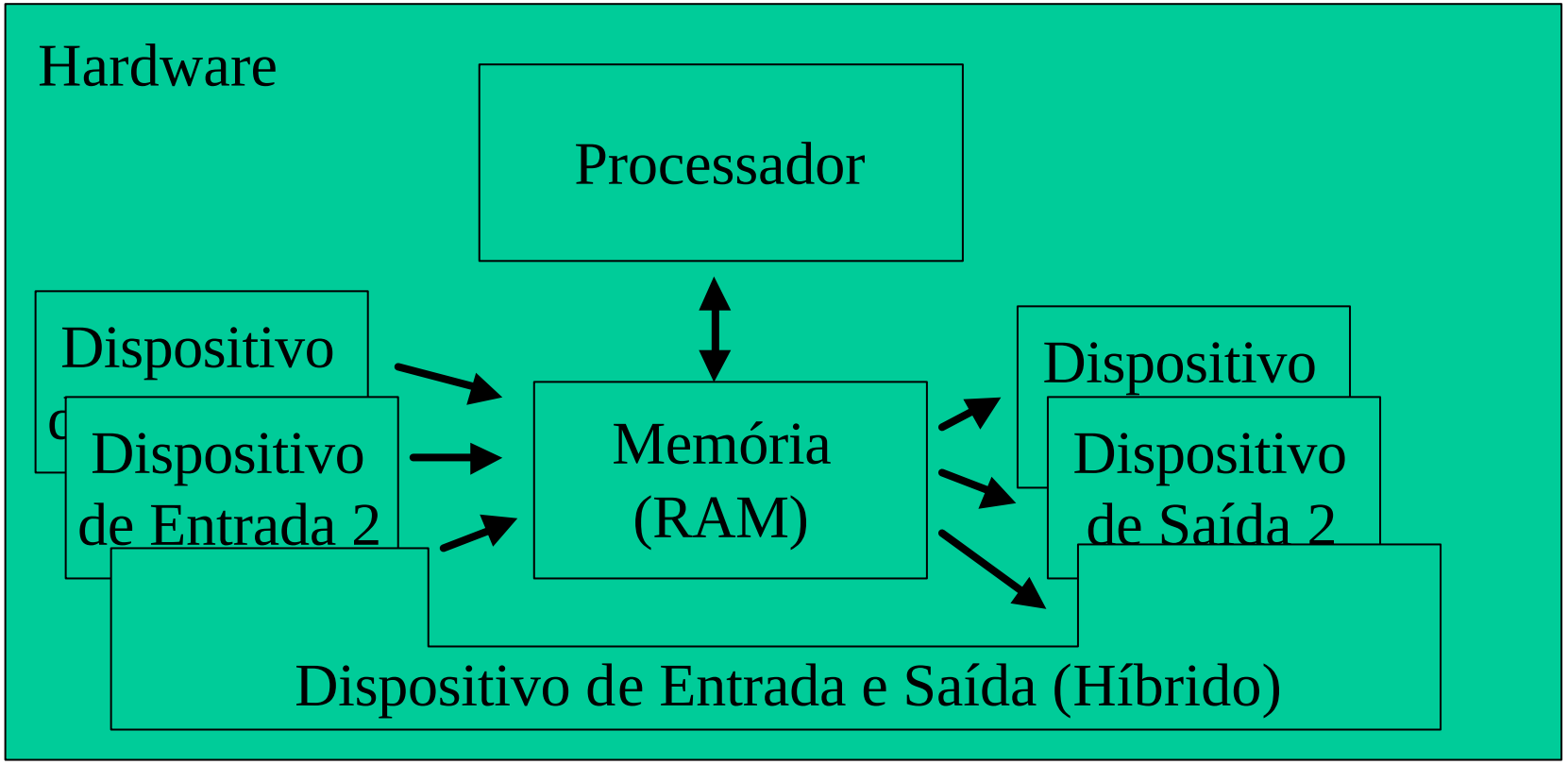
Jorge Fernandes

Fevereiro de 2005

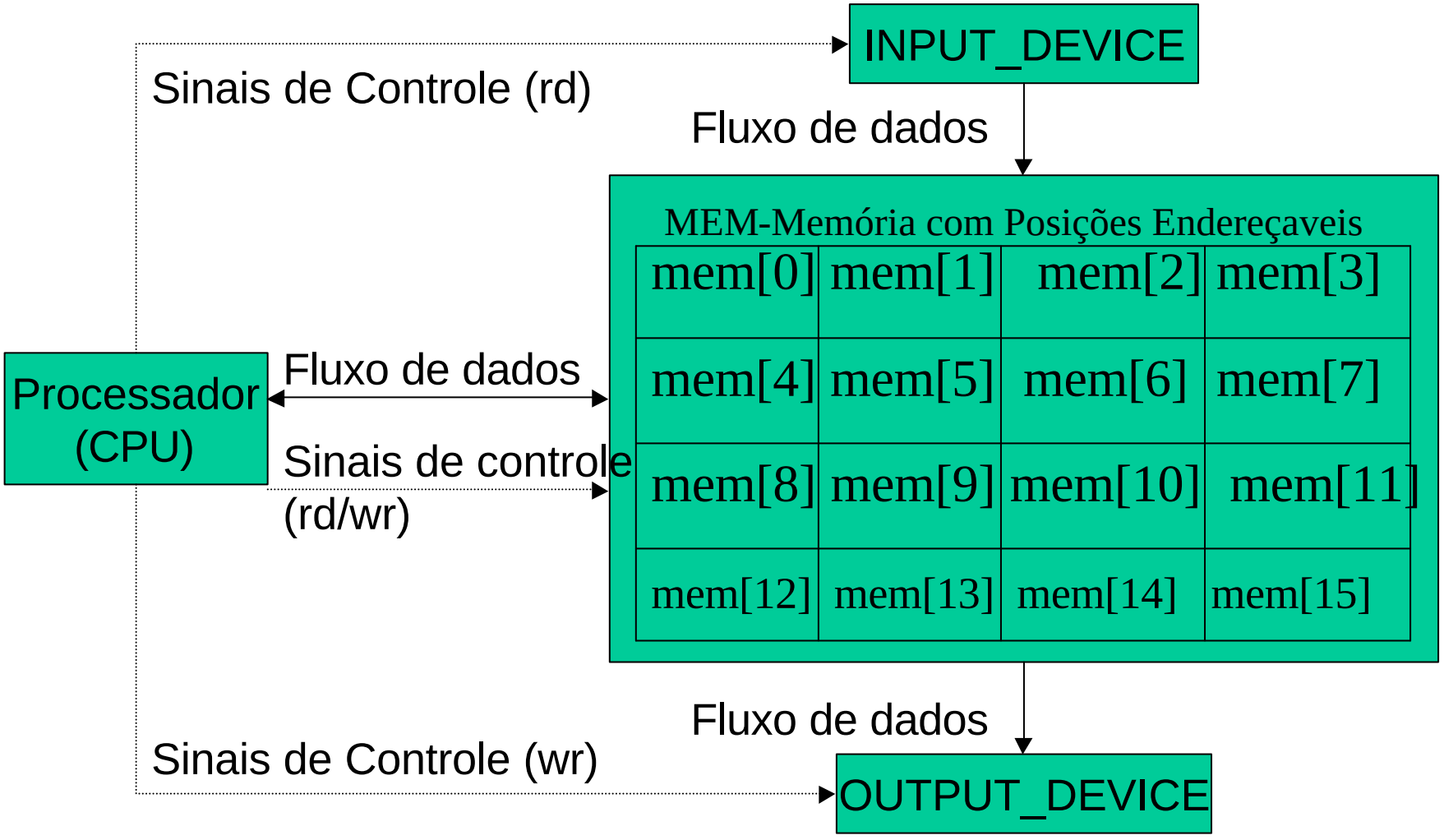
Uma Arquitetura Concreta para a Máquina de von Neumann

- Proposta em 1946
- Máquina composta por órgãos:
  - Memória (MEM)
    - Armazena dados, onde alguns destes dados são interpretados como instruções de um programa a ser executado pela CPU
  - Unidade Central de Processamento
    - Unidade de controle (UC)
    - Unidade de cálculos aritméticos e lógicos (UAL)
    - Registrador de instrução (IR)
    - Ponteiro de instrução (IP)
    - Registrador de uso geral (acumulador - ACC)
  - Dispositivos de Entrada e Saída
    - INPUT_DEVICE
    - OUTPUT_DEVICE

Organização Geral de um Computador de von Neumann



Um “Computador Visível” para Entender como o Sofware Funciona



O Processador
ou
Central Processing Unit – CPU
ou
Unidade Central de Processamento – UCP

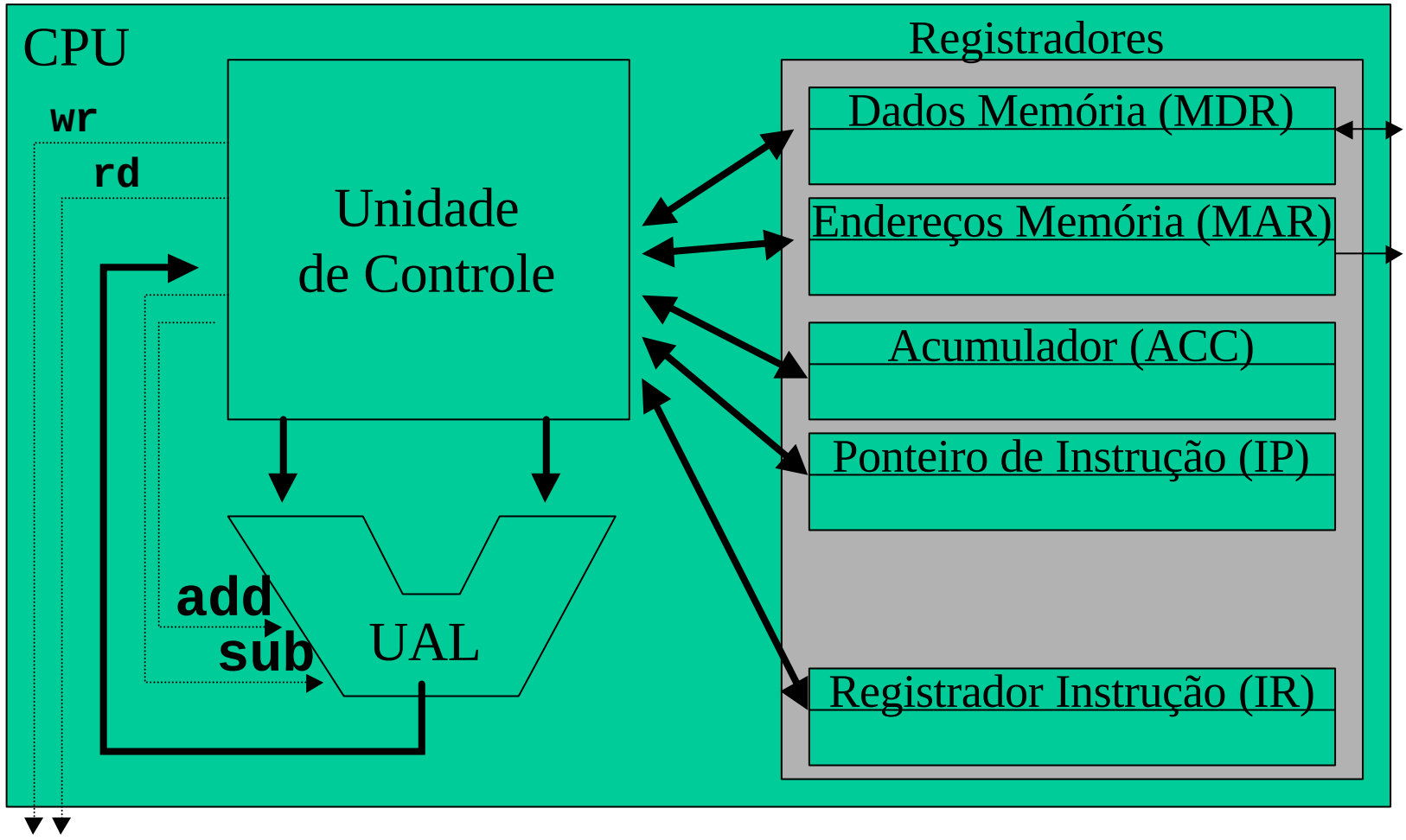
Atenção!

O termo CPU é muitas vezes usado de forma errônea para se referir ao GABINETE de um computador pessoal, que contém, além da CPU, dispositivos de armazenamentos, entrada e saída de dados, memória RAM, etc

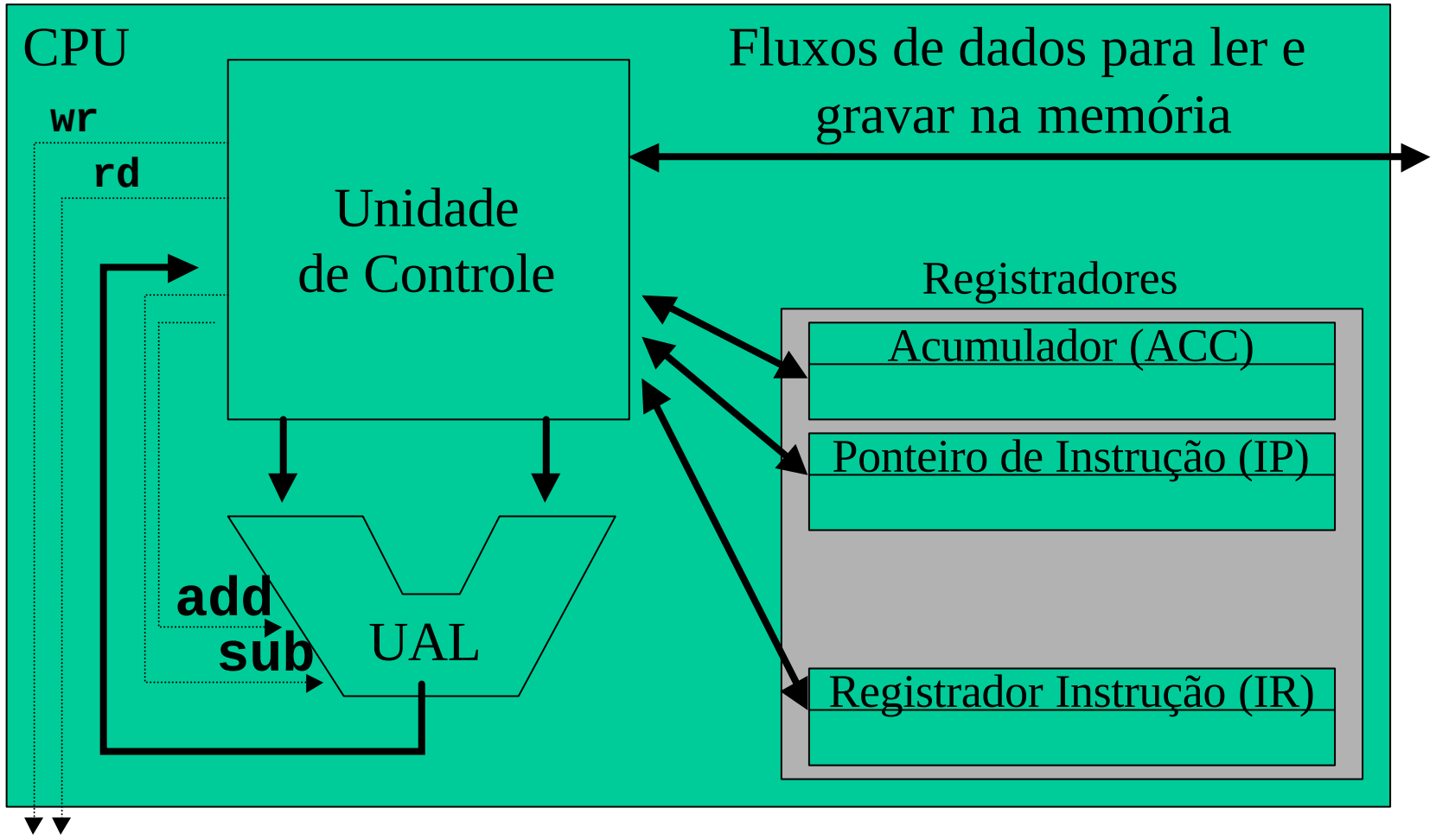
O Processador é Classicamente Organizado em Três Partes

- Registradores
  - Pequeno conjunto de células com nomes específicos, para leitura e gravação de dados em **baixíssimo** tempo.
- Unidade Aritmética Lógica (UAL)
  - Efetua transformações sobre dados (soma, subtração, operações lógicas - and, ou, etc.)
- Unidade de Controle
  - Interpretar os comandos ou instruções do software, controlando as transformações sobre dados e o fluxo destes dados entre todas as outras partes do hardware.

Organização interna do Processador (CPU)



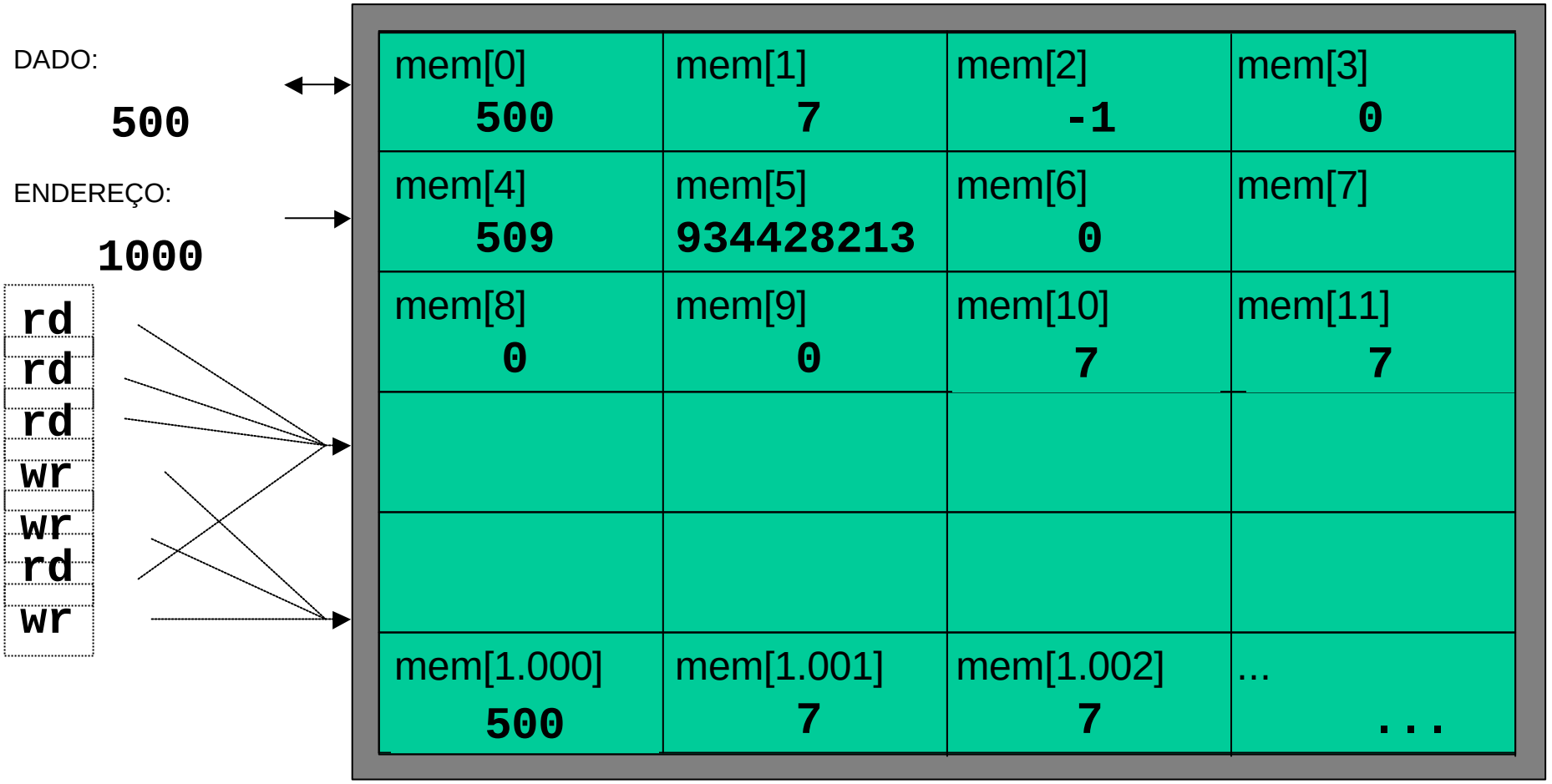
Organização simplificada do Processador



Memória RAM

- Conjunto de Células para Leitura (rd) e Gravação (wr) de Dados
- Diretamente Endereçáveis
  - Não há diferença no tempo de leitura, qualquer que seja a posição lida
  - Não há diferença no tempo de gravação, qualquer que seja a posição gravada
- Alta Velocidade no Armazenamento e Recuperação

Operações sobre Memória RAM (execute a animação para ver ciclos de leitura e escritura)



Programando na máquina de von Neumann

Instruções Mínimas para Interação com a Máquina: Linguagem de Entrada e Saída de Dados: LM_IO

- STOP
  - Pára execução do programa
- INPUT x
  - MEM[x] <- INPUT_DEVICE
    - Carrega na posição de memória indicada pelo valor x, o que está no dispositivo de entrada de dados
- OUTPUT x
  - OUTPUT_DEV ICE <- MEM[x]
    - Carrega no dispositivo de saída o valor que está armazenado na posição de memória indicada pelo valor x

Programa Eco:
apresenta no dispositivo de saída o que foi digitado na entrada

INPUT 3 --Lê entrada e guarda na memória 3

OUTPUT 3 --Imprime o valor guardado em memória 3

STOP -- pára a execução do programa

Semântica = significado das instruções, definido operacionalmente, através de fluxo de dados e controle entre os elementos da arquitetura

INPUT 3 significa MEM[3] <- INPUT_DEVICE

OUTPUT 3 significa OUTPUT_DEVICE <- MEM[3]

STOP significa Pára execução do programa

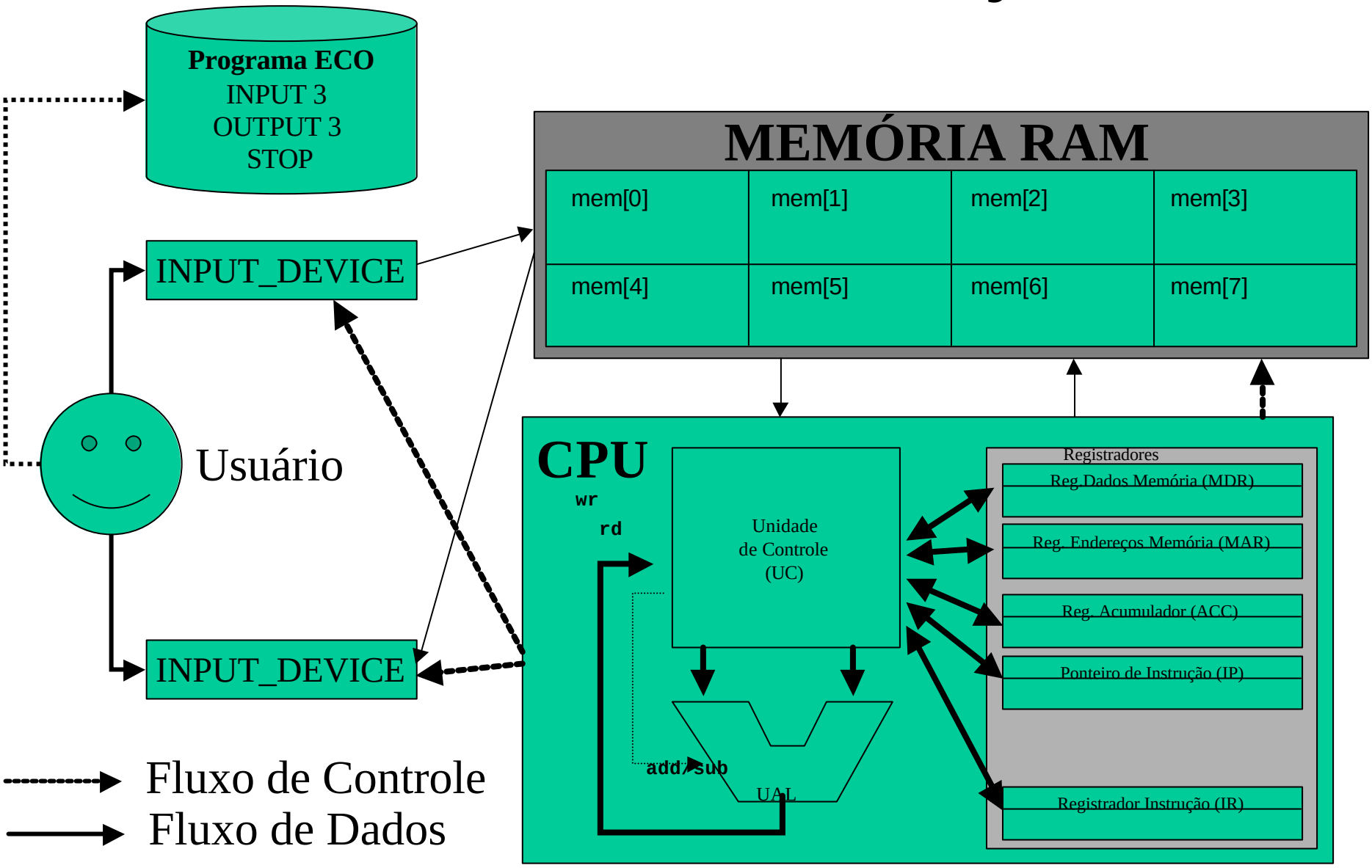
Executando programas na máquina de von Neumann

Ciclo de Execução de Programas

- Etapa 1– carga do programa na memória
- Etapa 2 – reset
  - IP := 0
- Etapa 3 – Ciclo de execução de instruções, até que a instrução STOP seja encontrada
  - Etapa 3.1 – FECTH – Busca a instrução na memória
    - IR := MEM[IP]
  - Etapa 3.2 – INC – Incrementa o IP
    - IP := IP + 1
  - Etapa 3.3 – DECODE – decodifica a instrução, e caso a instrução não seja STOP, volta à etapa 3.1

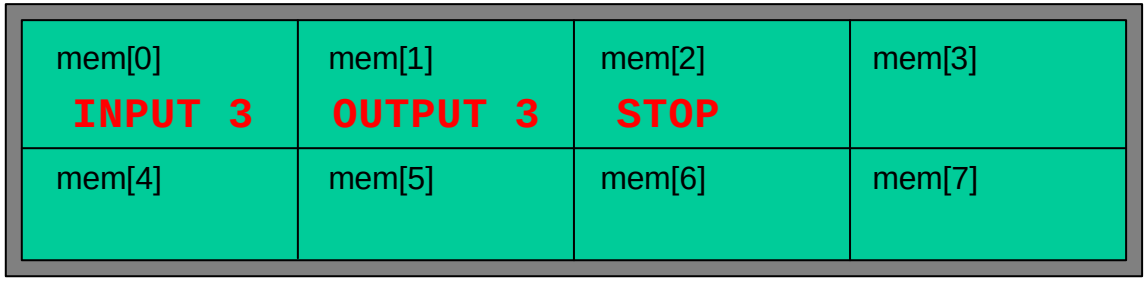
Executando o Programa ECO para observar um valor digitado pelo usuário, por exemplo o numeral 50

O Cenário de Execução



Executando o Programa ECO
Etapa 1 – Carga do Programa na Memória

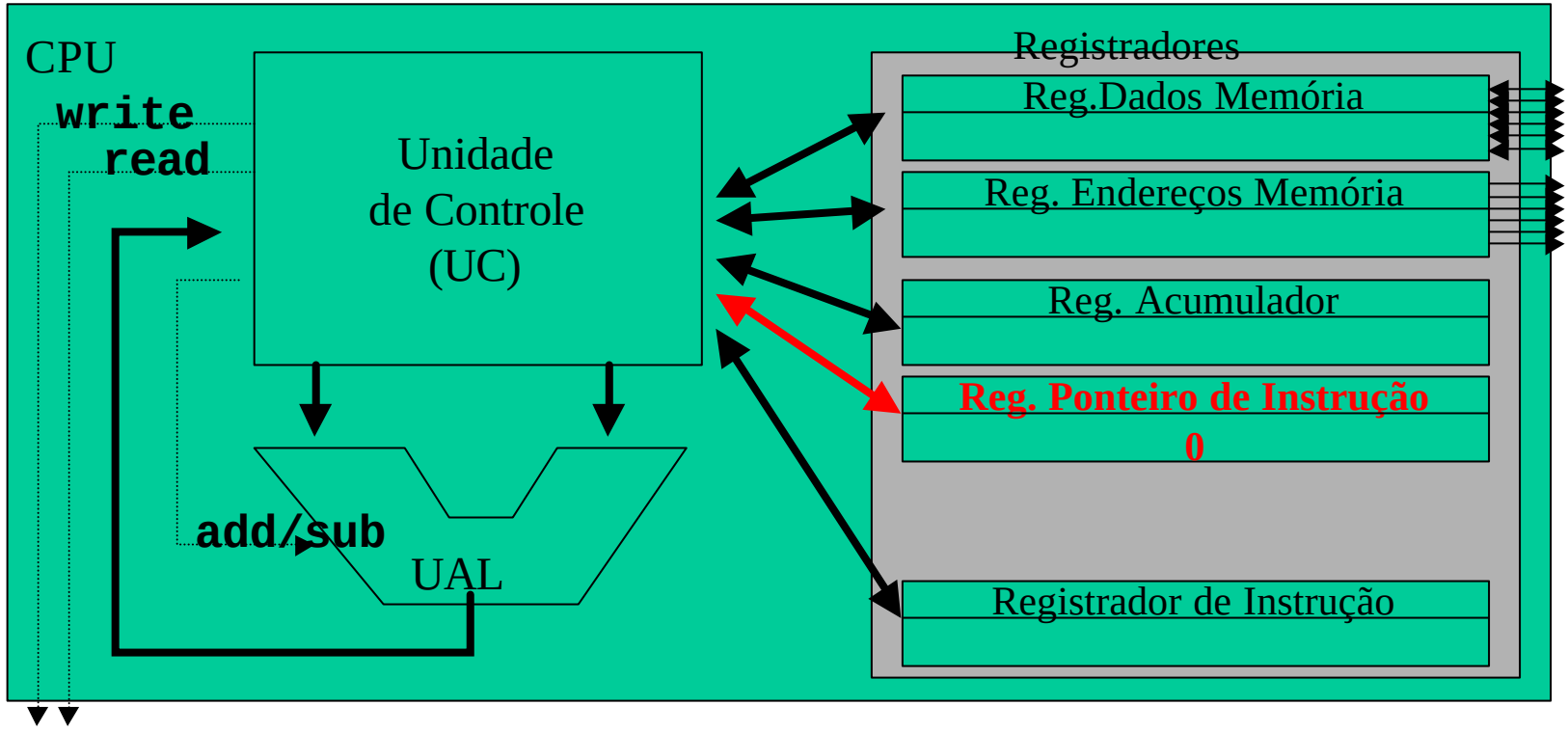
Programa Eco carregado numa memória RAM com 8 palavras de tamanho



Um programa do computador chamado loader cuida da carga do programa, que é feita usando os dispositivos de entrada de dados mas cujos detalhes serão omitidos. O resultado final é que o programa estará armazenado na Memória após a carga.

Executando o Programa ECO
Etapa 2 – Reset do IP

RESET do Ponteiro de Instruções (IP)



Executando o Programa ECO
Etapa 3 –ciclo de execução de instruções, até que a instrução STOP seja encontrada

3.1 – FECTH – Busca a instrução na memória

IR := MEM[IP]

3.2 – INC – Incrementa o IP

IP := IP + 1

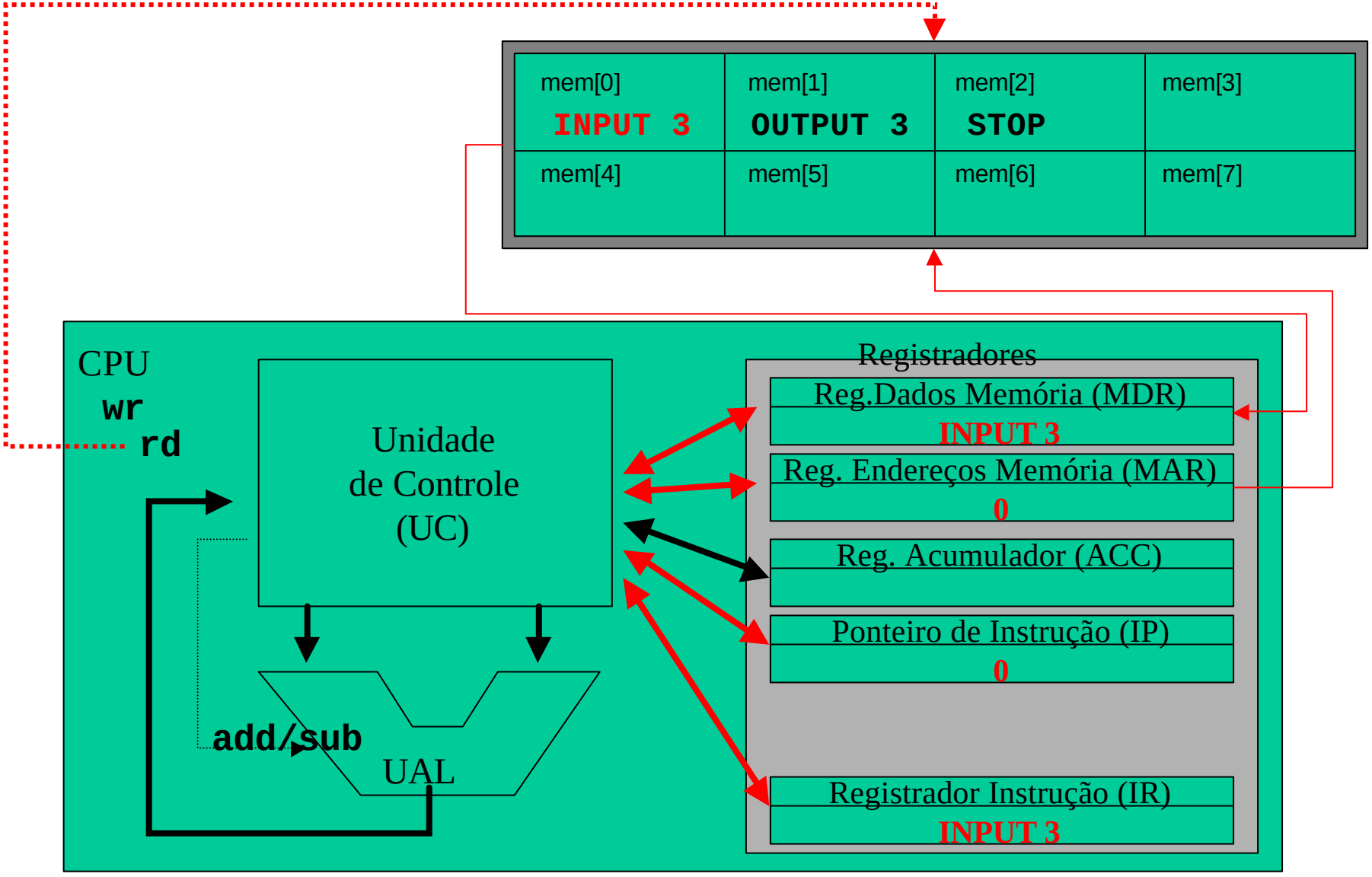
3.3 – DECODE – decodifica a instrução

Executando a 1a instrução:
INPUT 3

3.1 FETCH: Busca instrução na memória

FECTH – Busca a instrução na memória

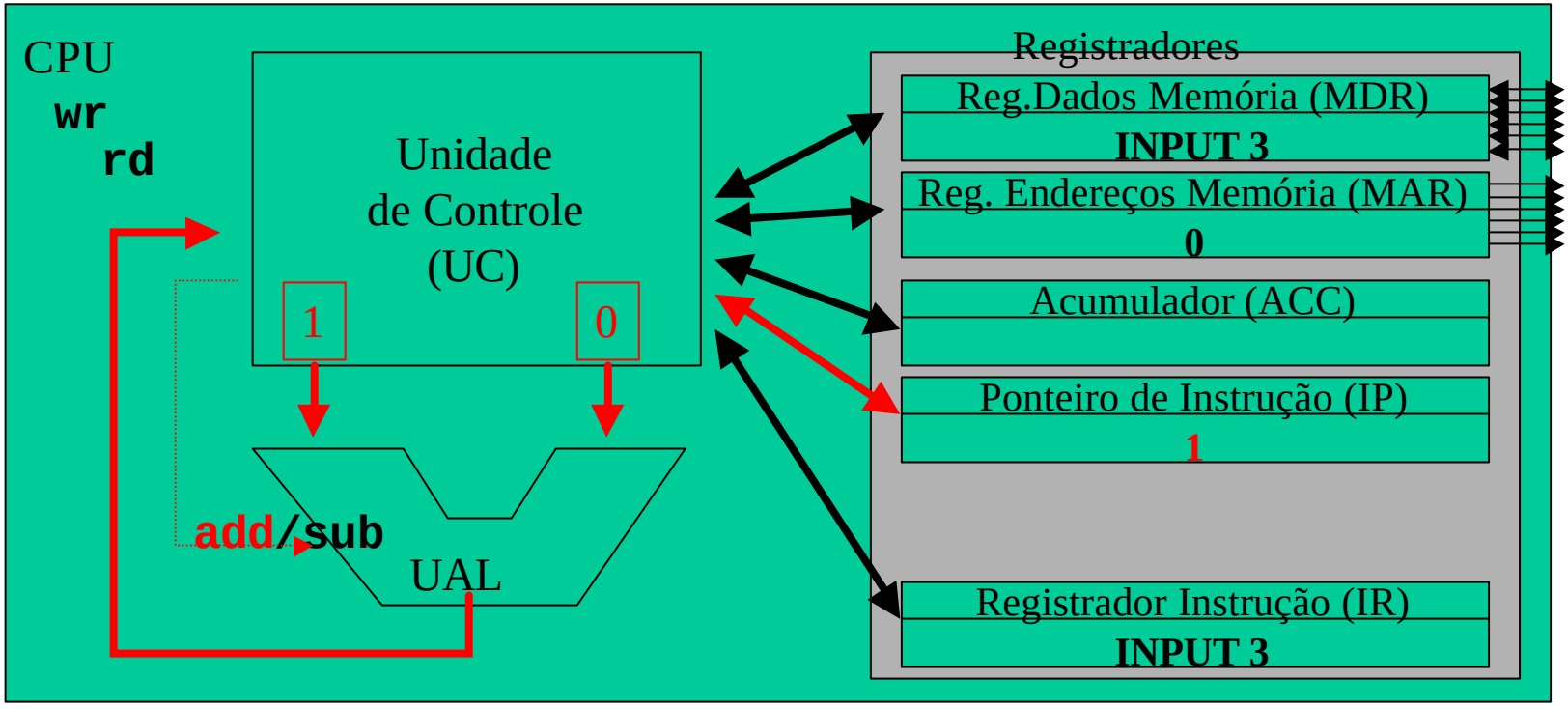
IR := MEM[IP]



3.2 INC: Incrementa ponteiro de instrução

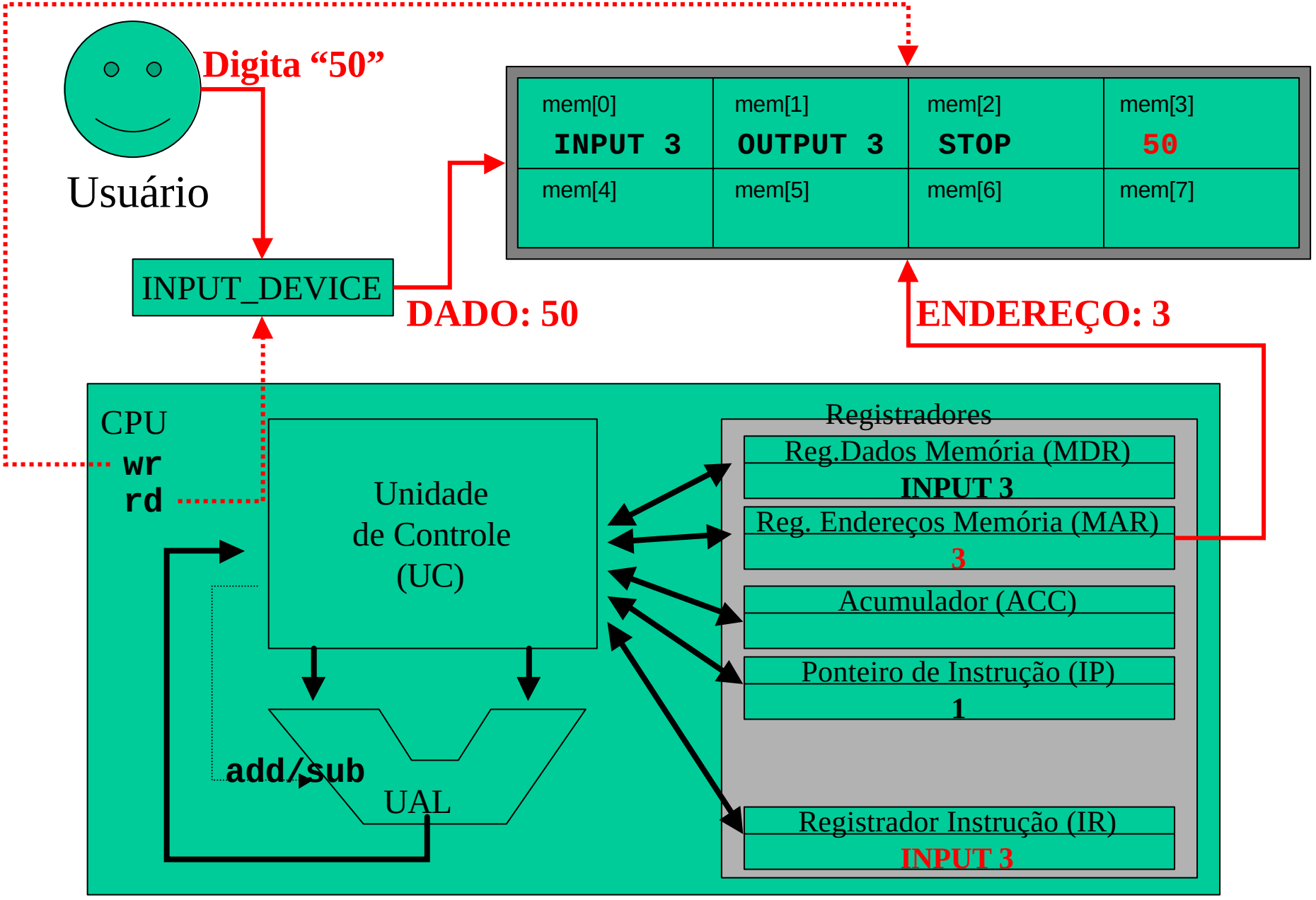
INC – Incrementa IP: IP := IP + 1

mem[0] **INPUT 3**	mem[1] **OUTPUT 3**	mem[2] **STOP**	mem[3]
mem[4]	mem[5]	mem[6]	mem[7]



3.3 DECODE: Decodifica instrução

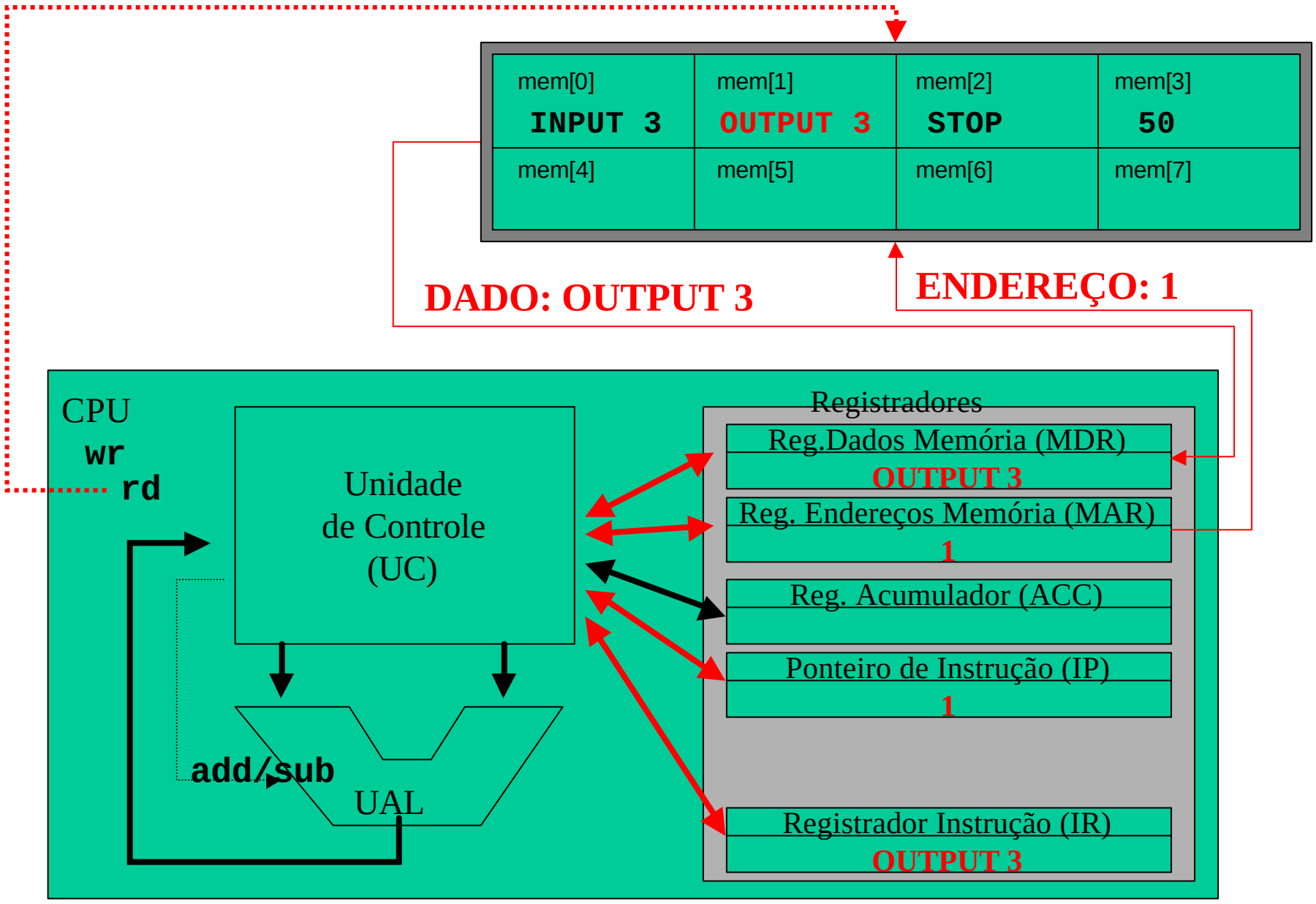
DECODE – decodifica a instrução



Executando a 2a instrução:
OUTPUT 3

3.1 FETCH: Busca instrução na memória

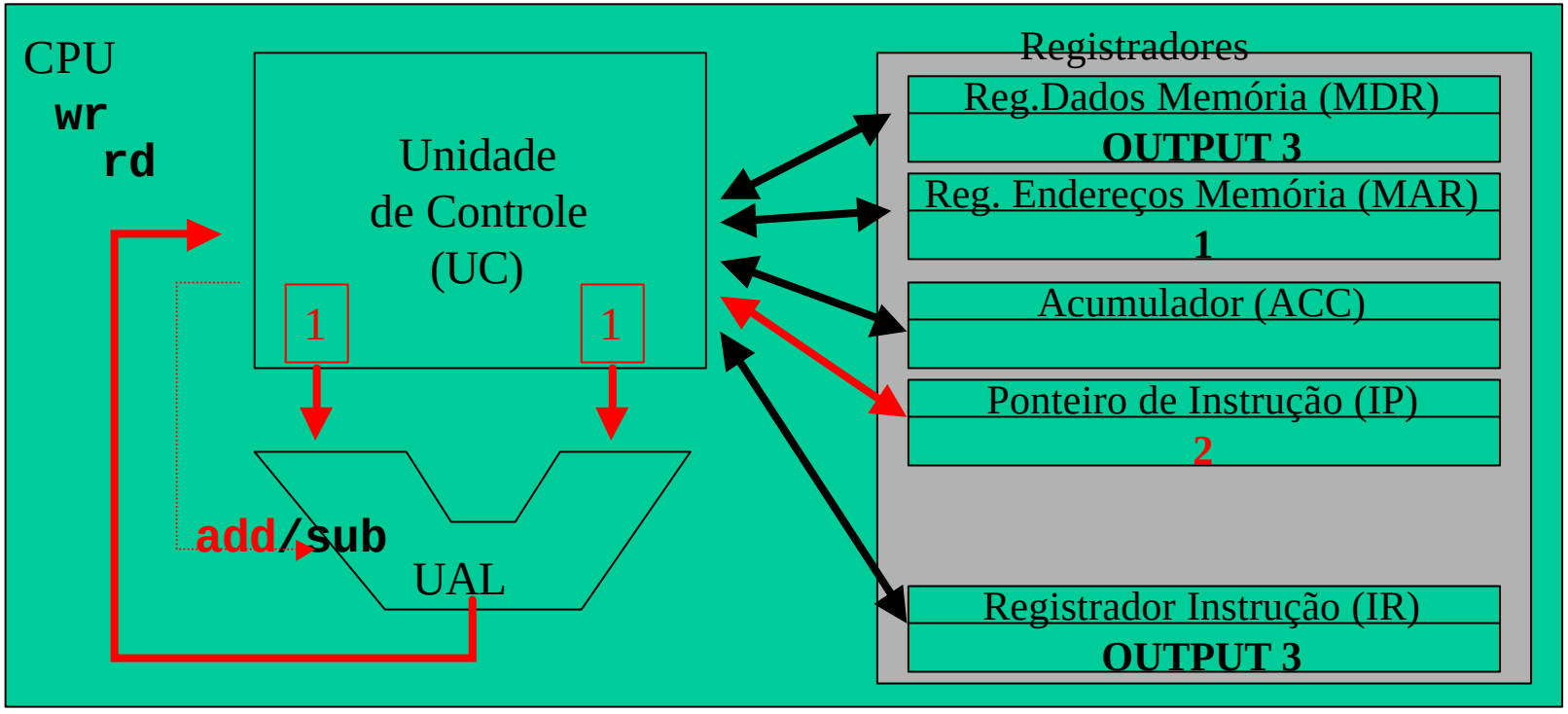
FECTH – Busca a instrução na memória (IR := MEM[IP])



3.2 INC: Incrementa ponteiro de instrução

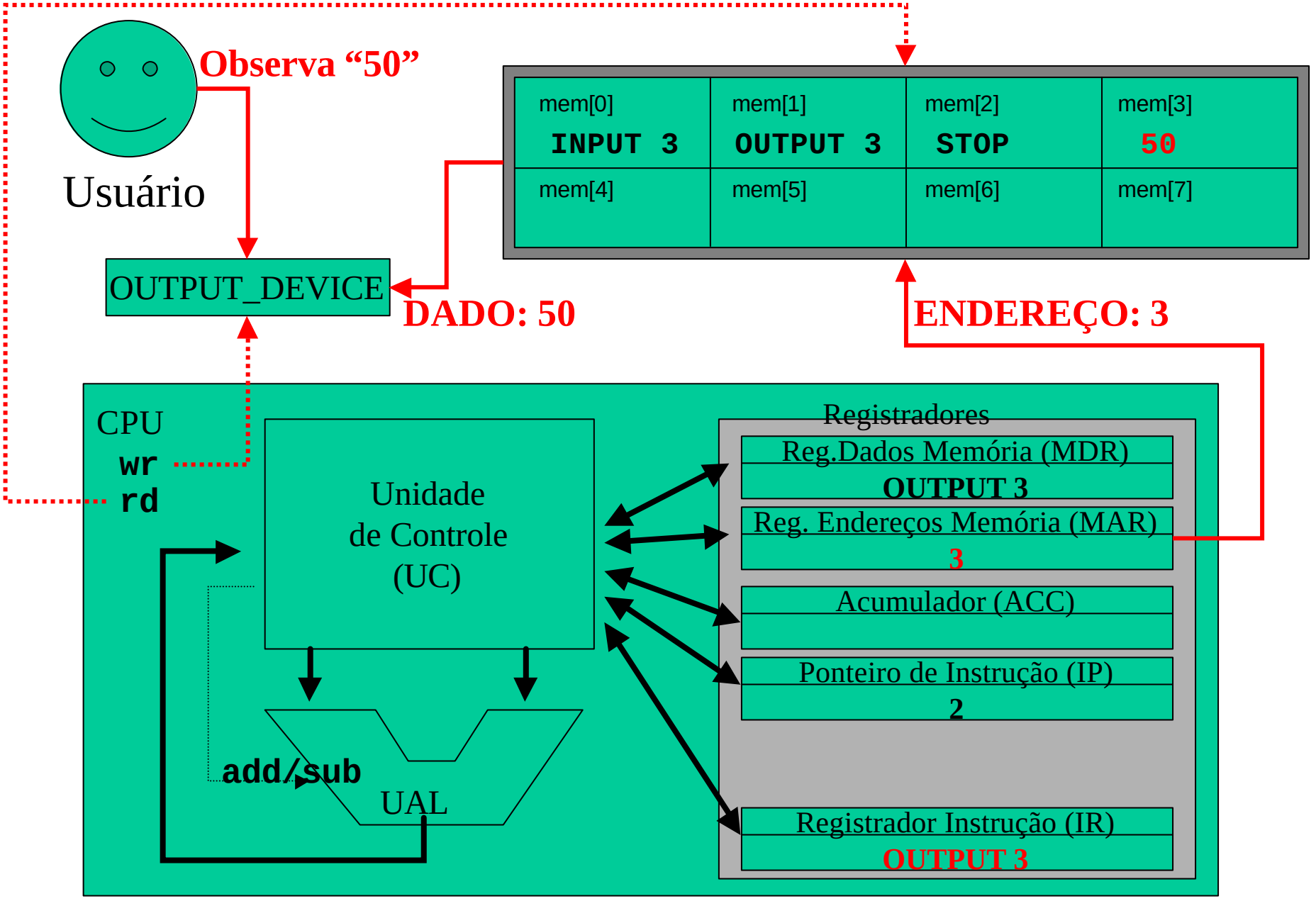
INC – Incrementa IP: IP := IP + 1

mem[0] **INPUT 3**	mem[1] **OUTPUT 3**	mem[2] **STOP**	mem[3]
mem[4]	mem[5]	mem[6]	mem[7]



3.3 DECODE: Decodifica instrução

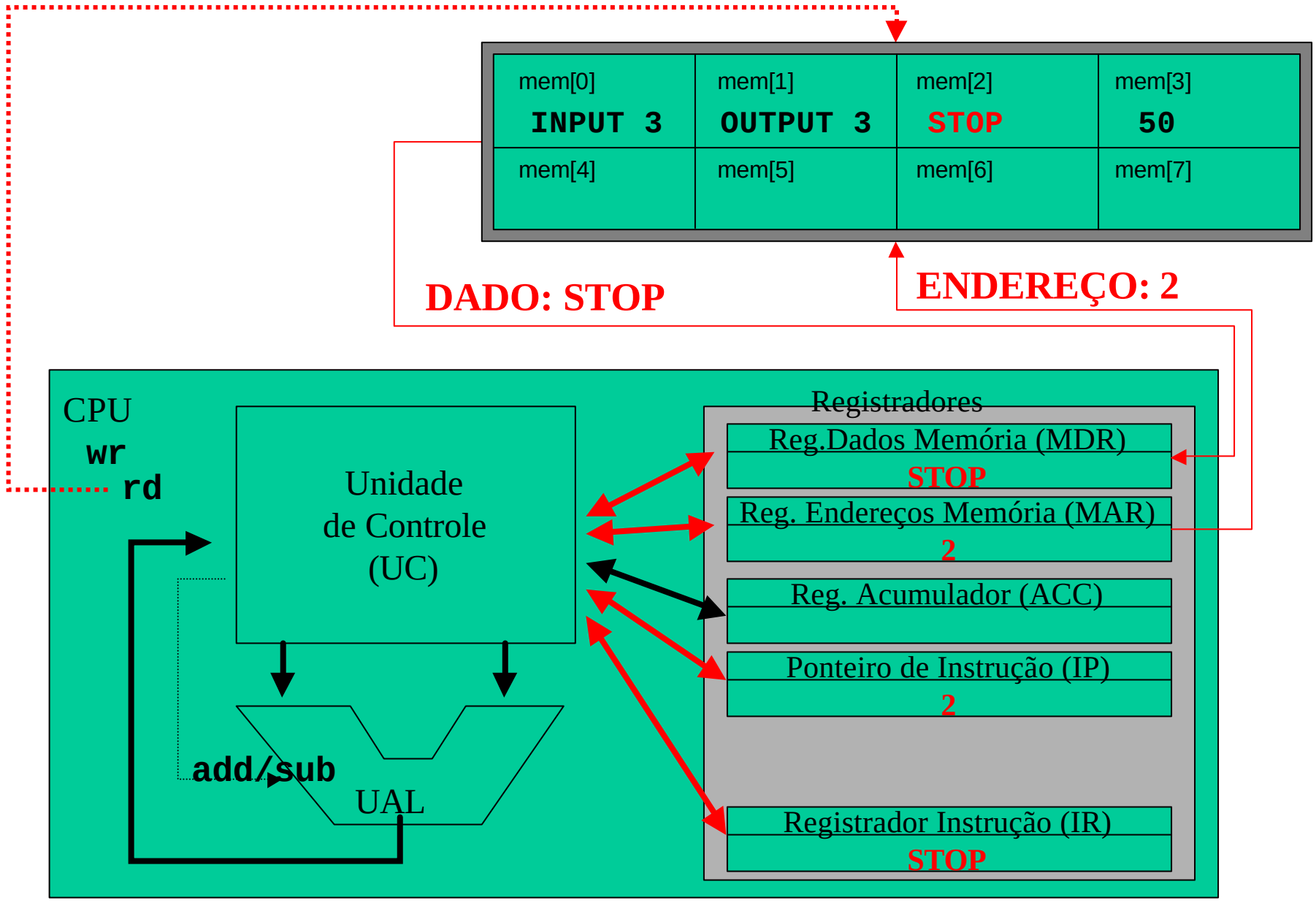
DECODE – decodifica a instrução



Executando a 3a instrução: STOP

3.1 FETCH: Busca instrução na memória

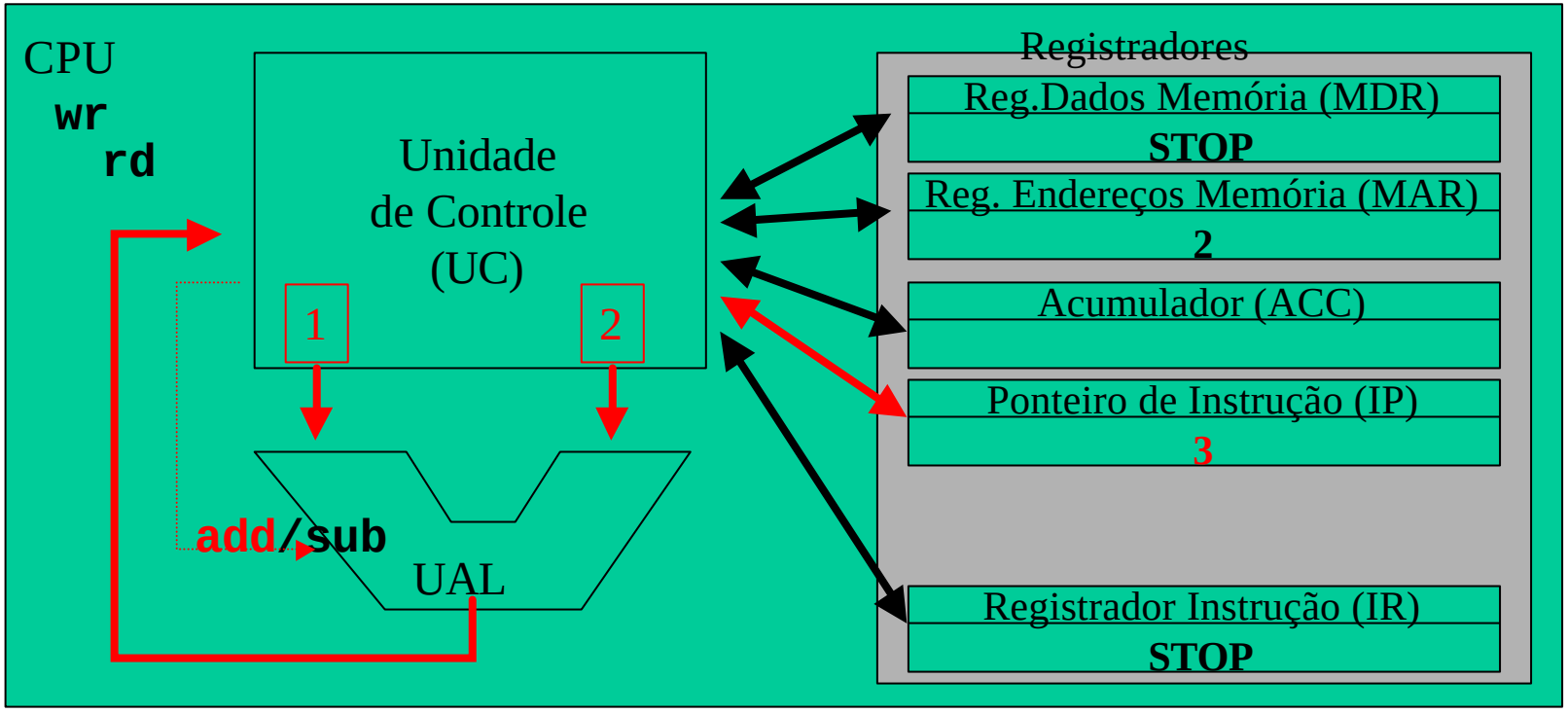
FECTH – Busca a instrução na memória (IR := MEM[IP])



3.2 INC: Incrementa ponteiro de instrução

INC – Incrementa IP: IP := IP + 1

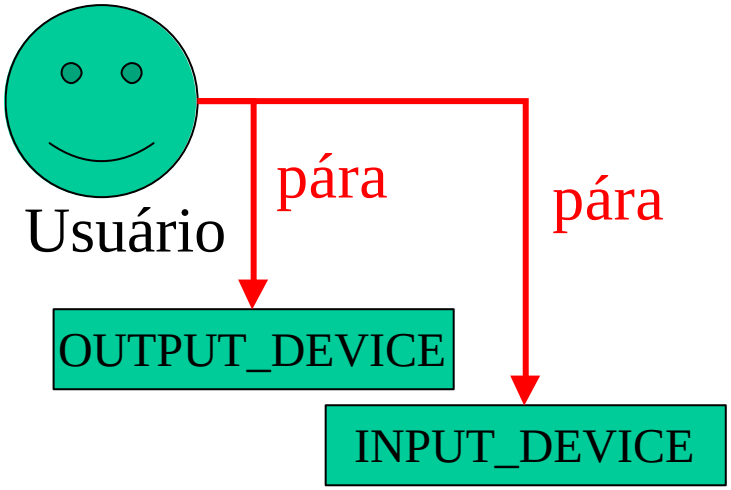
mem[0] **INPUT 3**	mem[1] **OUTPUT 3**	mem[2] **STOP**	mem[3]
mem[4]	mem[5]	mem[6]	mem[7]



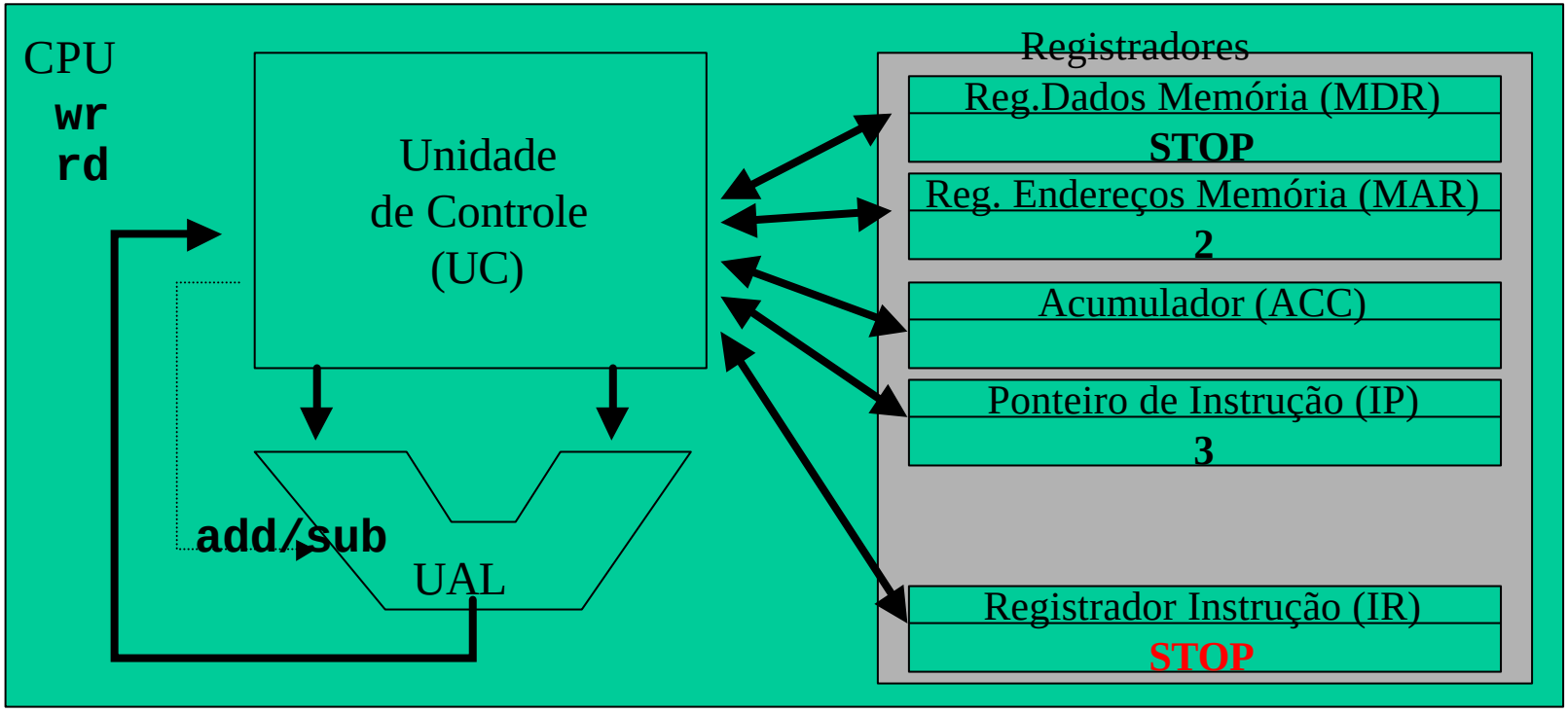
3.3 DECODE: Decodifica instrução

DECODE – decodifica a instrução

Observa fim de execução



mem[0]	mem[1]	mem[2]	mem[3]
INPUT 3	**OUTPUT 3**	**STOP**	**50**
mem[4]	mem[5]	mem[6]	mem[7]



Uma Arquitetura Concreta para a Máquina de von Neumann

Jorge Fernandes