Organização de Computadores
(revisão)

André Tavares da Silva
andre.silva@udesc.br

Conceitos Básicos

- **Microcomputador** é um sistema computacional que possua como CPU um microprocessador.
- **Microprocessador** é qualquer componente que implemente “on chip” as funções de uma unidade central de processamento.
- **Microcontrolador** é qualquer componente que incorpore “on chip” a maioria das unidades de um microcomputador ou seja: CPU, memória, portas e periféricos de E/S.
- **DSP** é qualquer microcontrolador que adicione funções avançadas para condicionamento e processamento digital de sinais (DSP - Digital Signal Processor).

Arquitetura X Organização

de Computadores

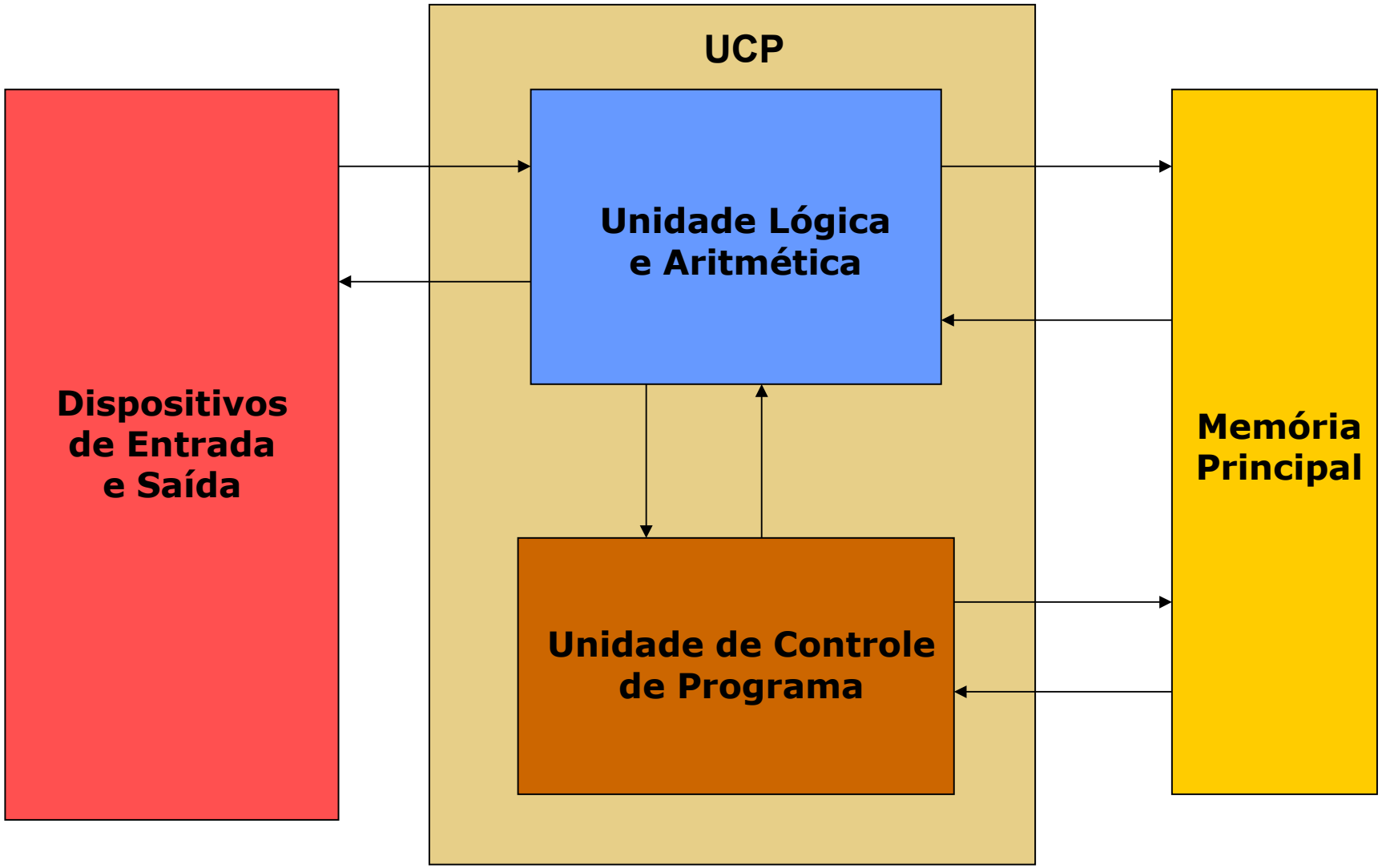
“John” Von Newmann

- *Margittai Neumann János Lajos*
- Nascido em 26/12/1903
- Budapeste / Hungria
- Morreu em 08/02/1957

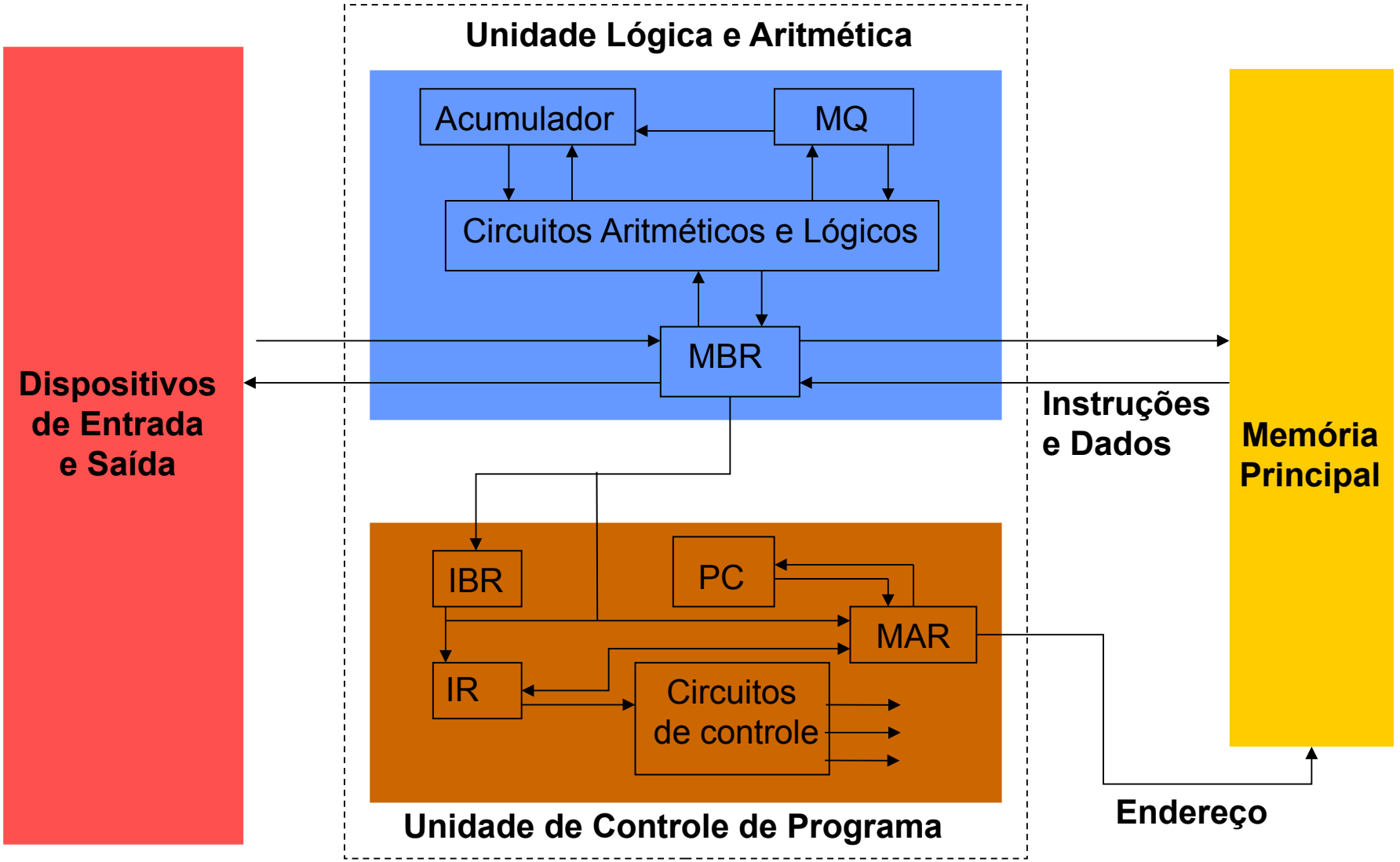
“John” Von Newmann - Contribuições

- Teoria dos conjuntos
- Análise funcional
- Mecânica quântica
- Ciência da computação
- Economia
- Teoria dos jogos
- Análise numérica
- Estatística
- Matemática

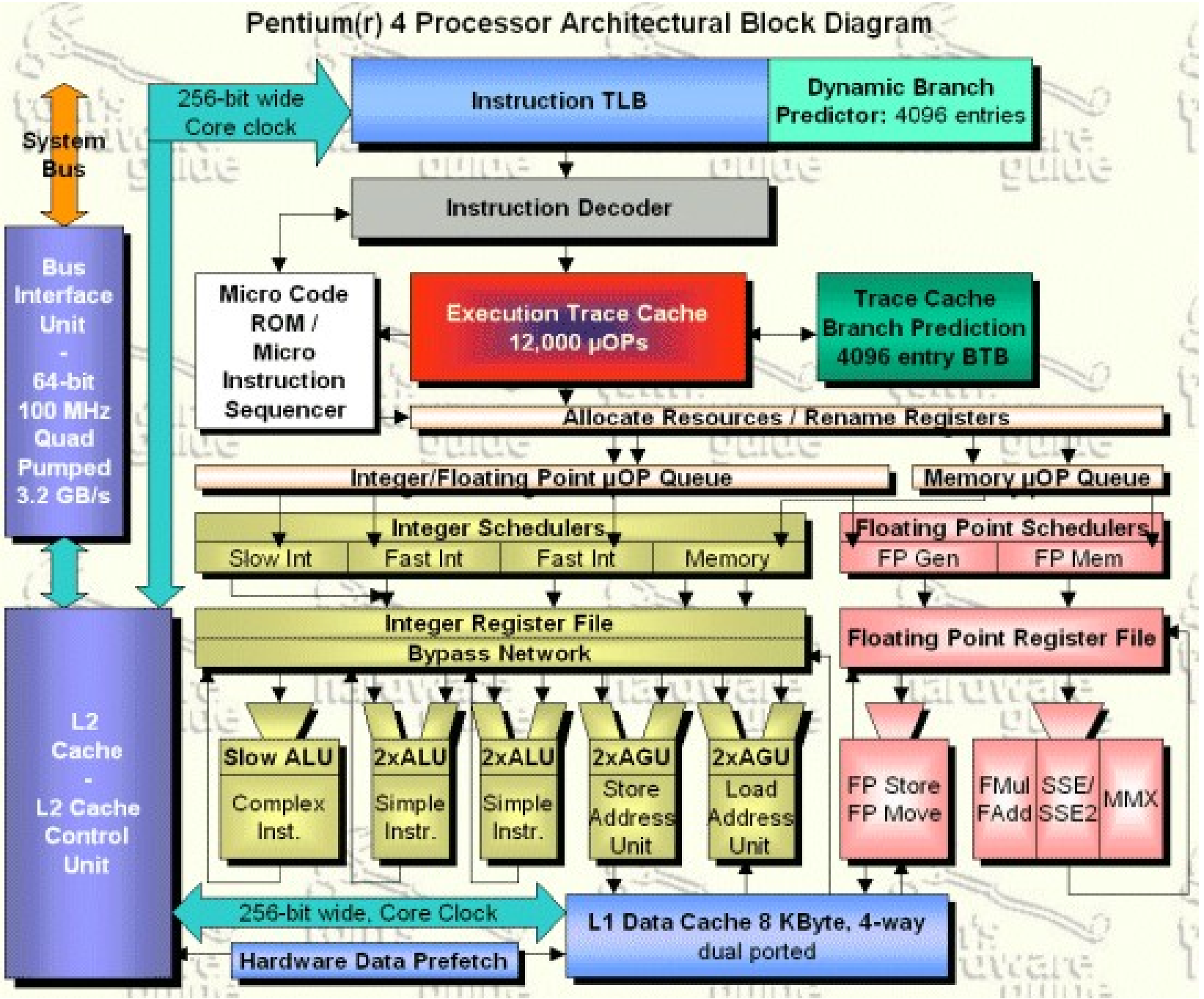
Modelo de Computador



Modelo de Computador



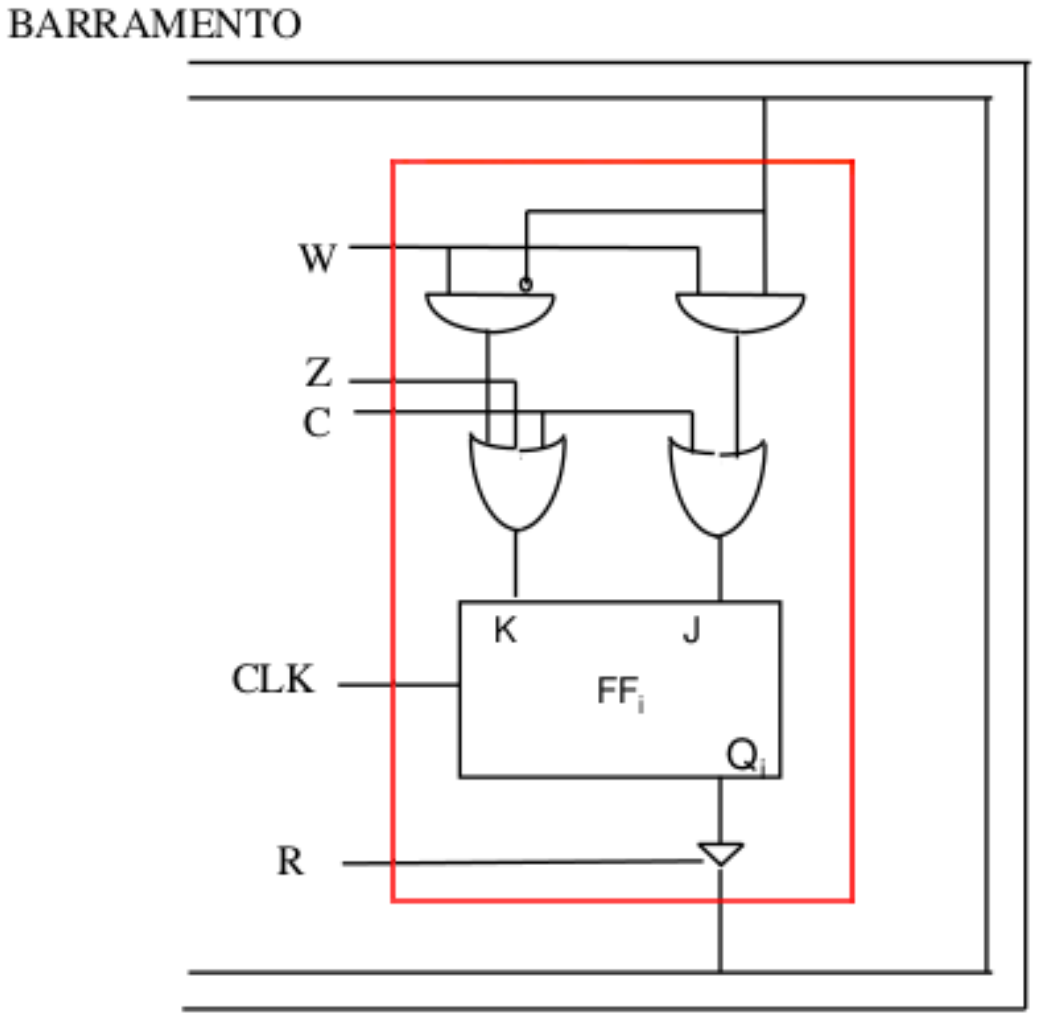
Exemplo - P4

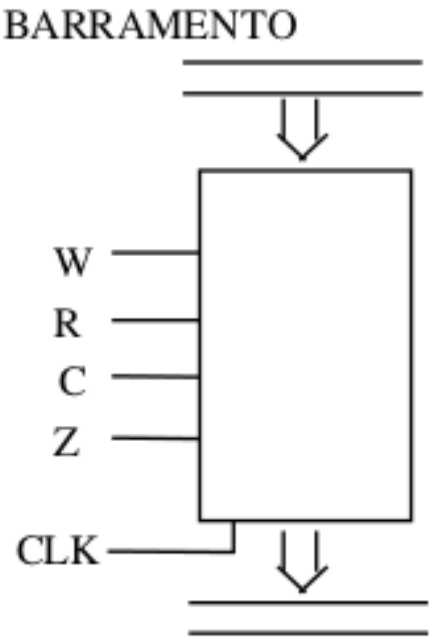

Pentium(r) 4 Processor Architectural Block Diagram
System Bus
256-bit wide Core clock
Instruction TLB
Dynamic Branch Predictor: 4096 entries
Instruction Decoder
Bus Interface Unit - 64-bit 100 MHz Quad Pumped 3.2 GB/s
Micro Code ROM / Micro Instruction Sequencer
Execution Trace Cache 12,000 µOPs
Trace Cache Branch Prediction 4096 entry BTB
Allocate Resources / Rename Registers
Integer/Floating Point µOP Queue
Memory µOP Queue
Integer Schedulers
Slow Int
Fast Int
Fast Int
Memory
Floating Point Schedulers
FP Gen
FP Mem
Integer Register File
Bypass Network
Floating Point Register File
L2 Cache - L2 Cache Control Unit
Slow ALU
Complex Inst.
2xALU
Simple Instr.
2xALU
Simple Instr.
2xAGU
Store Address Unit
2xAGU
Load Address Unit
FP Store FP Move
FMul FAdd
SSE/ SSE2
MMX
256-bit wide, Core Clock
Hardware Data Prefetch
L1 Data Cache 8 KByte, 4-way dual ported

“Revisão” de flip-flops e registradores

- Operações básicas:
  - Transferência
  - Complementação
  - Deslocamento
  - Incremento/Decremento
  - Set/Reset

Operações múltiplas entre registradores





Registrador Sensível a Múltiplos Comandos (Sinais de Controle)

W - Transferir do barramento para o registrador (Write)

R - Transferir do registrador para o barramento (Read)

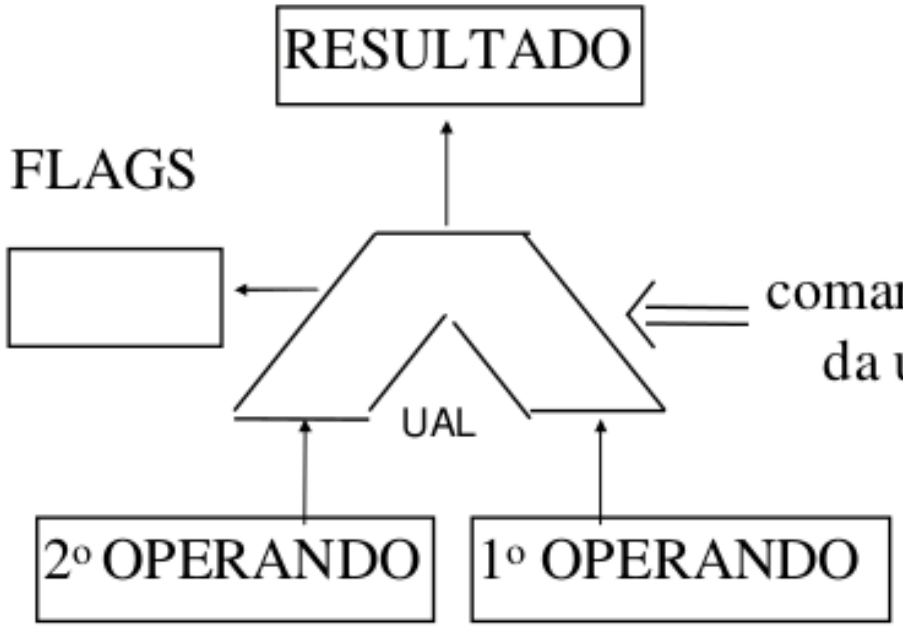
C - Complementar o registrador

Z - Zerar o registrador

Unidade Lógica e Aritmética- ULA

- A ULA faz parte da CPU, sendo responsável pela execução de todas as operações sobre os dados.
- A ULA pode ser classificada, dependendo de como devem ser especificados os operandos e resultados, em :
  - máquina de 3 endereços,
  - máquina de 2 endereços,
  - máquina de 1 endereço,
  - máquina de zero endereços.

Máquina de 3 endereços



Formato da instrução

C.O.	OP1	OP2	RES
código de operação	end. 1º operando	end. 2º operando	endereço resultado

RES ← (OP1) operação (OP2)

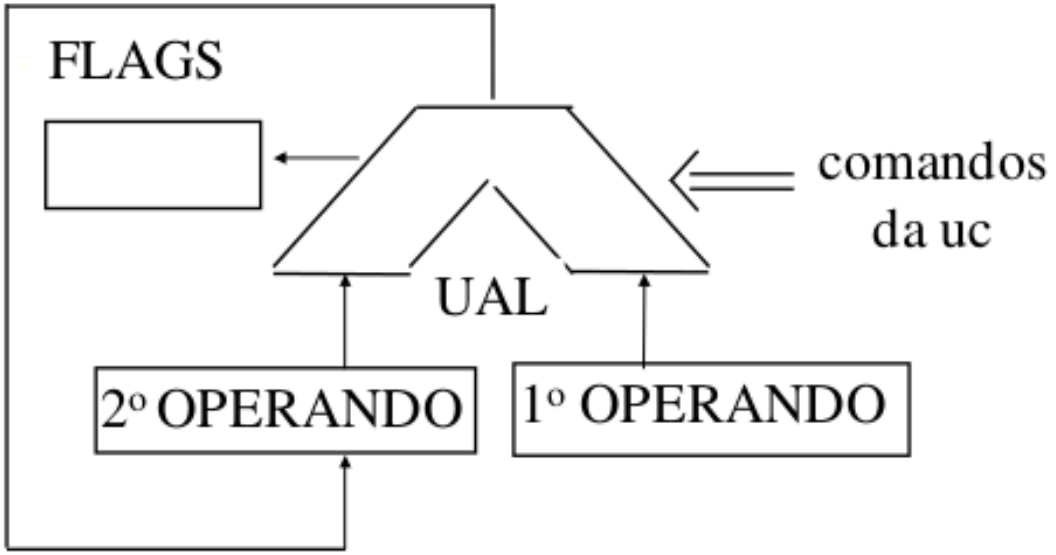
Exemplo de adição: R = S1 + S2 ⇒ R ← (S1) + (S2)

ADD			
código de operação	end. 1º operando	end. 2º operando	endereço resultado

Em linguagem simbólica (Assembly) teríamos, por exemplo:

ADD S1, S2, R ⇒ R ← (S1) + (S2)

Máquina de 2 endereços



Formato da instrução

C.O.	OP1	OP2

código de operação — end. 1º operando — end. 2º operando

OP2 ← (OP1) operação (OP2)

Exemplo de adição: R = S1 + S2 ⇒ R ← (S1) + (S2)

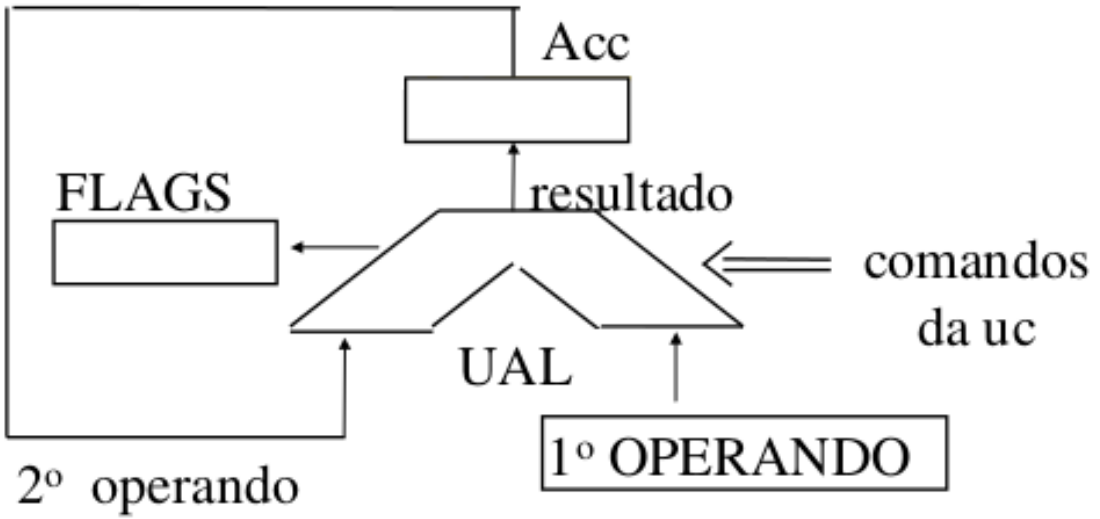
ADD		

código de operação — end. 1º operando — end. 2º operando

Em linguagem simbólica (Assembly) teríamos, por exemplo: (necessitamos de 2 instruções)

MOVE S1, R ⇒ R ← (S1)
ADD S2, R ⇒ R ← (R) + (S2)

Máquina de 1 endereço



Formato da instrução

código de operação — endereço operando

Acc ← (Acc) operação (OP1)

Exemplo de adição: R = S1 + S2 ⇒ R ← (S1) + (S2)

código de operação — endereço operando

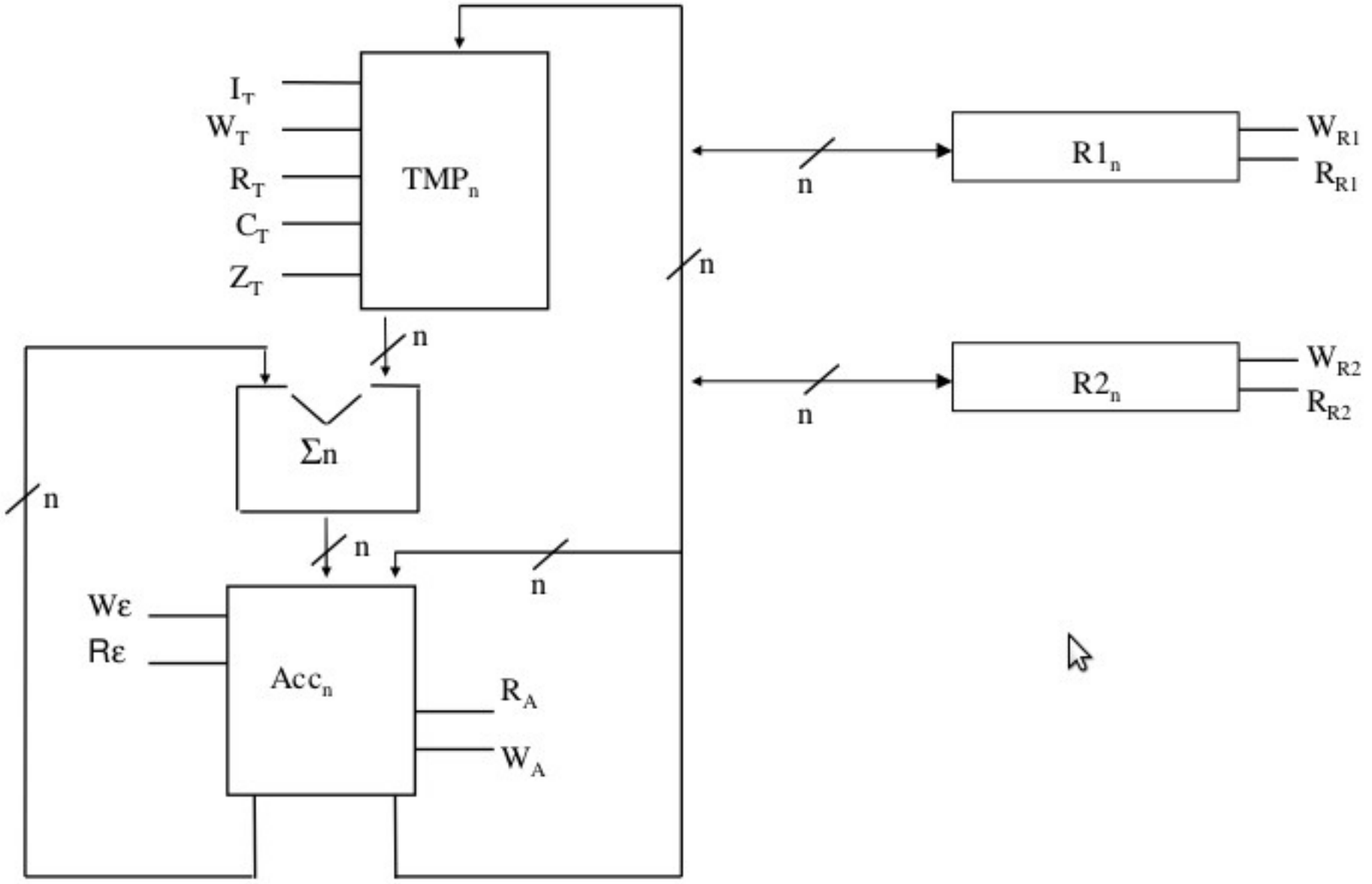
Em linguagem simbólica (Assembly) teríamos, por exemplo: (necessitamos de 3 instruções)

LOAD	S1	⇒	Acc ← (S1)
ADD	S2	⇒	Acc ← (Acc) + (S2)
STA	R	⇒	R ← (Acc)

Máquina de zero endereços

- Máquinas de zero endereços também denominadas “stack machines” ou máquinas por pilha, armazenam os operandos em uma pilha necessitando assim somente da operação no corpo da instrução.

Um controlador simples



- Unidade de Controle é a sequenciadora e controladora da execução das instruções de máquina, via um conjunto de microoperações que atuam sobre regs., memória, etc.

Sinais de controle (microcomandos)

- **W** - transferir do barramento para o registrador: R1 ← (bus)
- **R** - transferir do registrador para o barramento: bus ← (Acc)
- **Wε**- transferir a saída do somador para o Acc
- **Rε**- transferir o conteúdo do registrador Acc para a entrada do Σn
- $\mathbf{I_T}$ - incrementar o conteúdo do registrador TMP: TMP ← (TMP) + 1
- $\mathbf{C_T}$ - complementar o conteúdo do registrador TMP: TMP ← ($\overline{TMP}$)
- $\mathbf{Z_T}$ - zerar (reset) o registrador TMP: TMP ← "0"

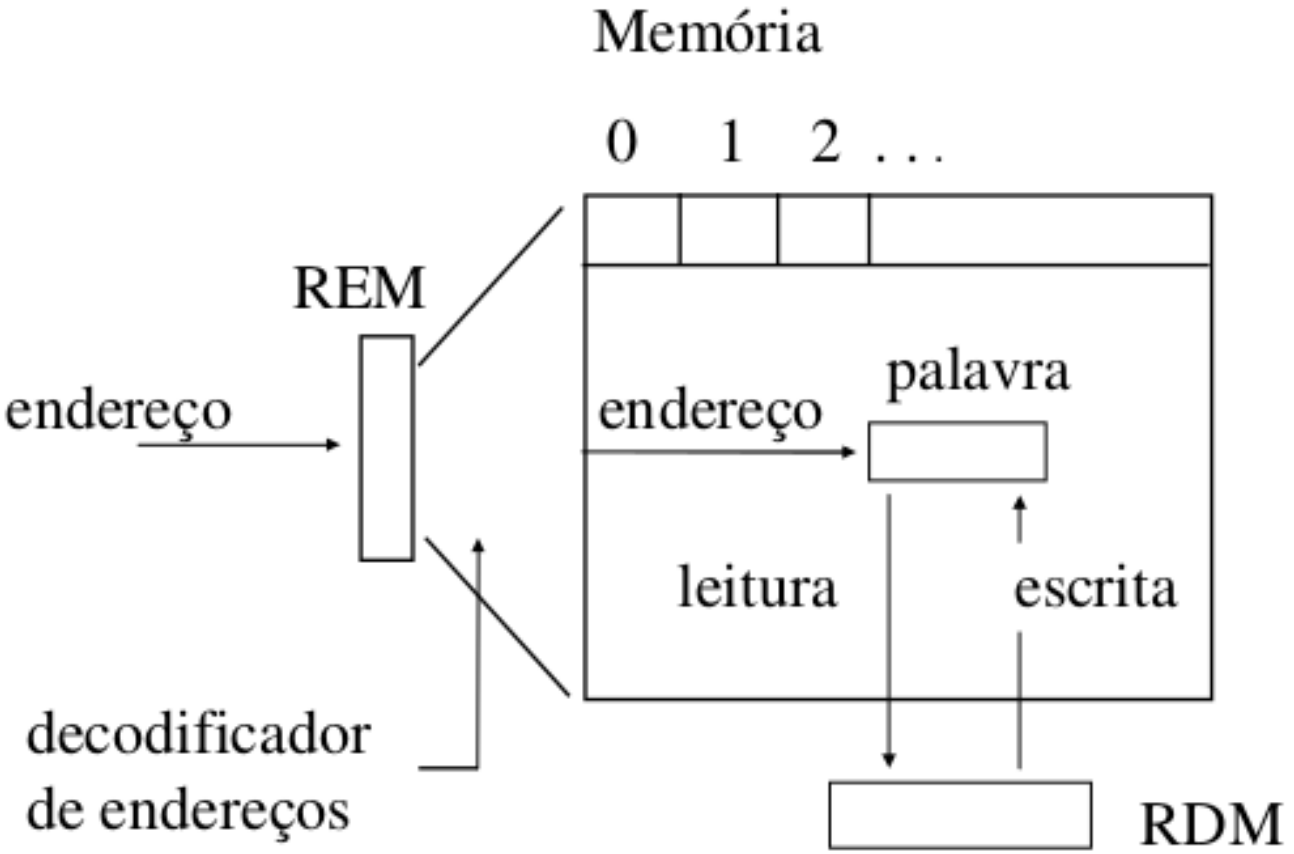
- $\mathbf{R2_n}$,$\mathbf{R1_n}$ – registradores de dados com n bits
- $\mathbf{TMP_n}$ – registrador de complemento/incremento com n bits
- $\mathbf{Acc_n}$ – registrador acumulador com n bits
- **Σn** – somador combinacional de n bits
- **n-/-** – barramento de n bits

Microcomandos (exemplo)

$$R1 \leftarrow (R1) + (R2)$$

Passos para Adição	Sinais de Controle (microcomandos)	Pulso do Relógio	Notação Simbólica de Microoperações
1. Transferir o conteúdo de R1 para Acc	R_{R1} , W_A	1	bus ← (R1) Acc ← (bus)
2. Transferir o conteúdo de R2 para TMP	R_{R2} , W_T	2	bus ← (R2) TMP ← (bus)
3. Somar saída Σn ← (TMP) + (Acc)	R_T , Rε	3	saída Σn ← soma
4. Armazenar a soma Acc ← saída Σn	Wε	4	Acc ← soma
5. Armazenar resultado R1 ← (Acc)	R_A , W_{R1}	5	bus ← (Acc) R1 ← (bus)

Representação da memória principal



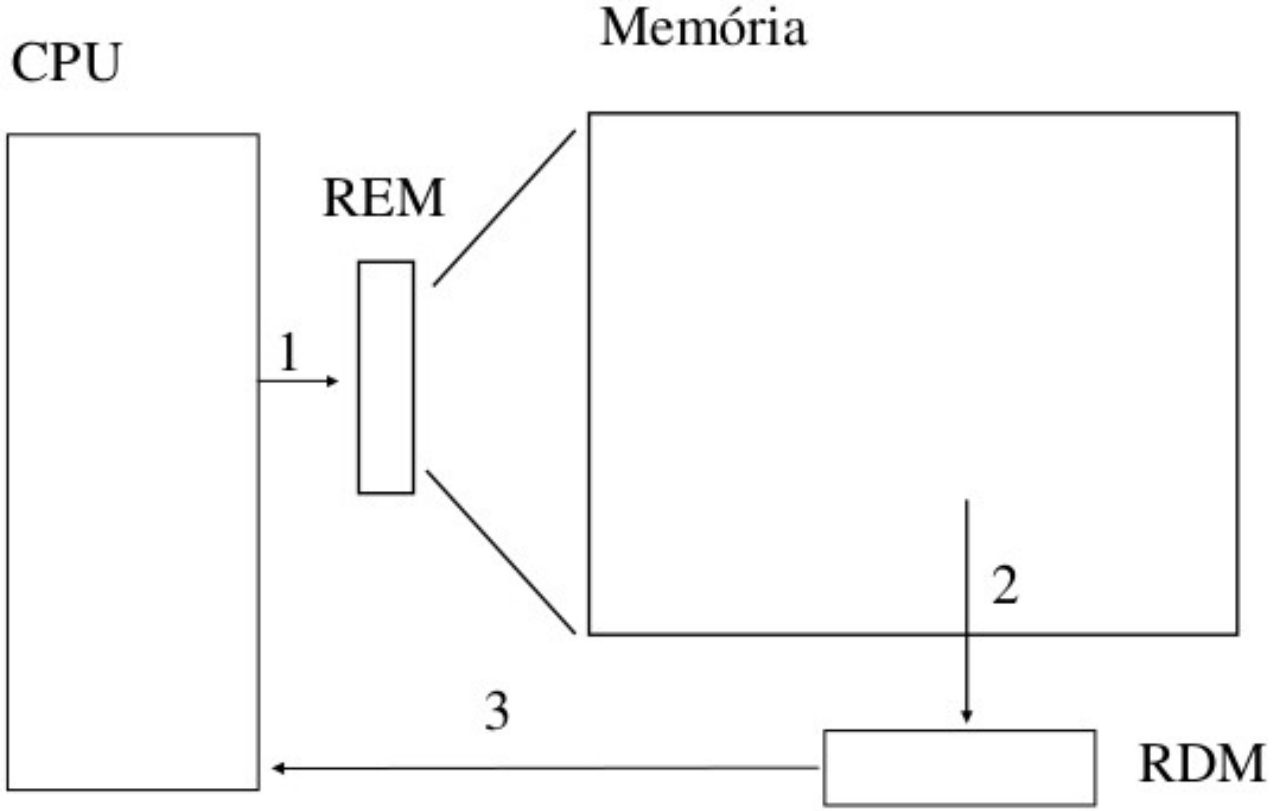
REM - registrador de endereços da memória

RDM - registrador de dados da memória

Obs: **X** – endereço

(X) – conteúdo do endereço **X**

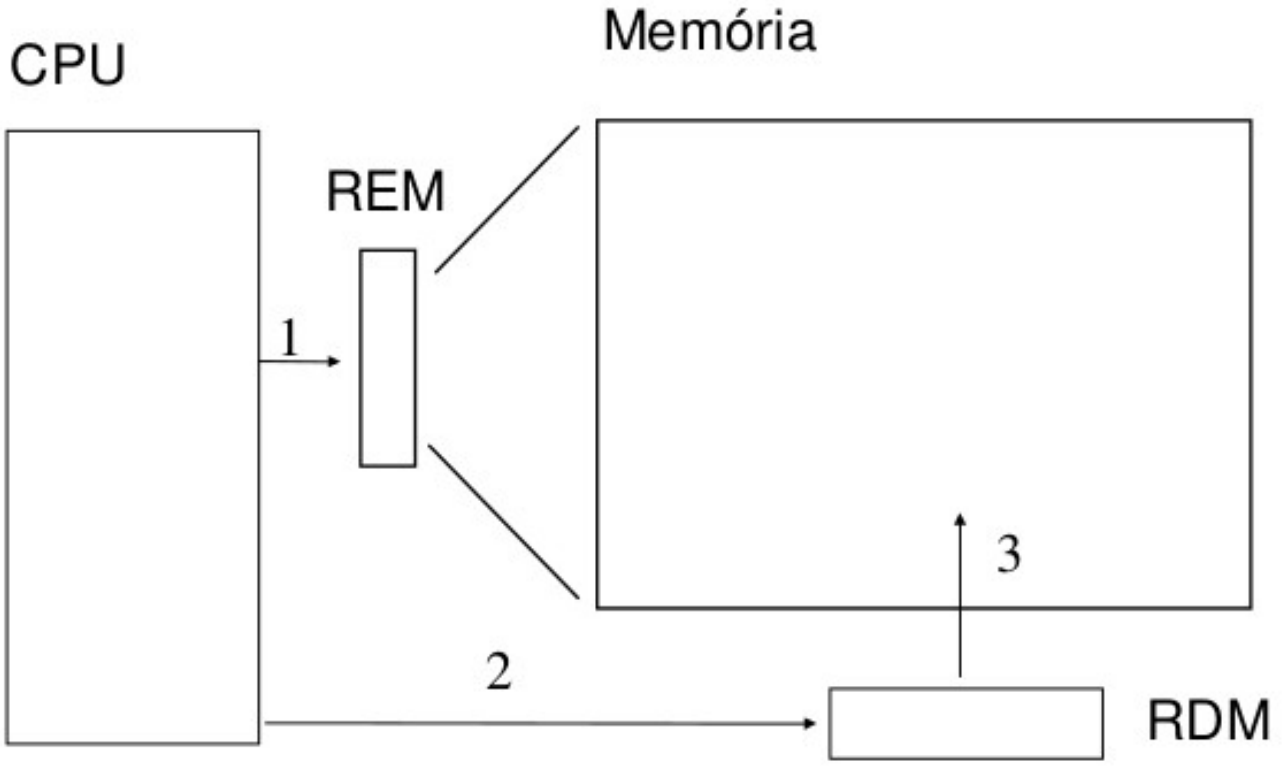
Memória – Ciclo de leitura



Pulso	Microoperação
1	REM ← endereço
2	leitura :
	RDM ← ((REM)) ou
	RDM ← (m)
3	CPU ← (RDM)

onde m é o endereço de uma posição de memória

Memória – Ciclo de escrita



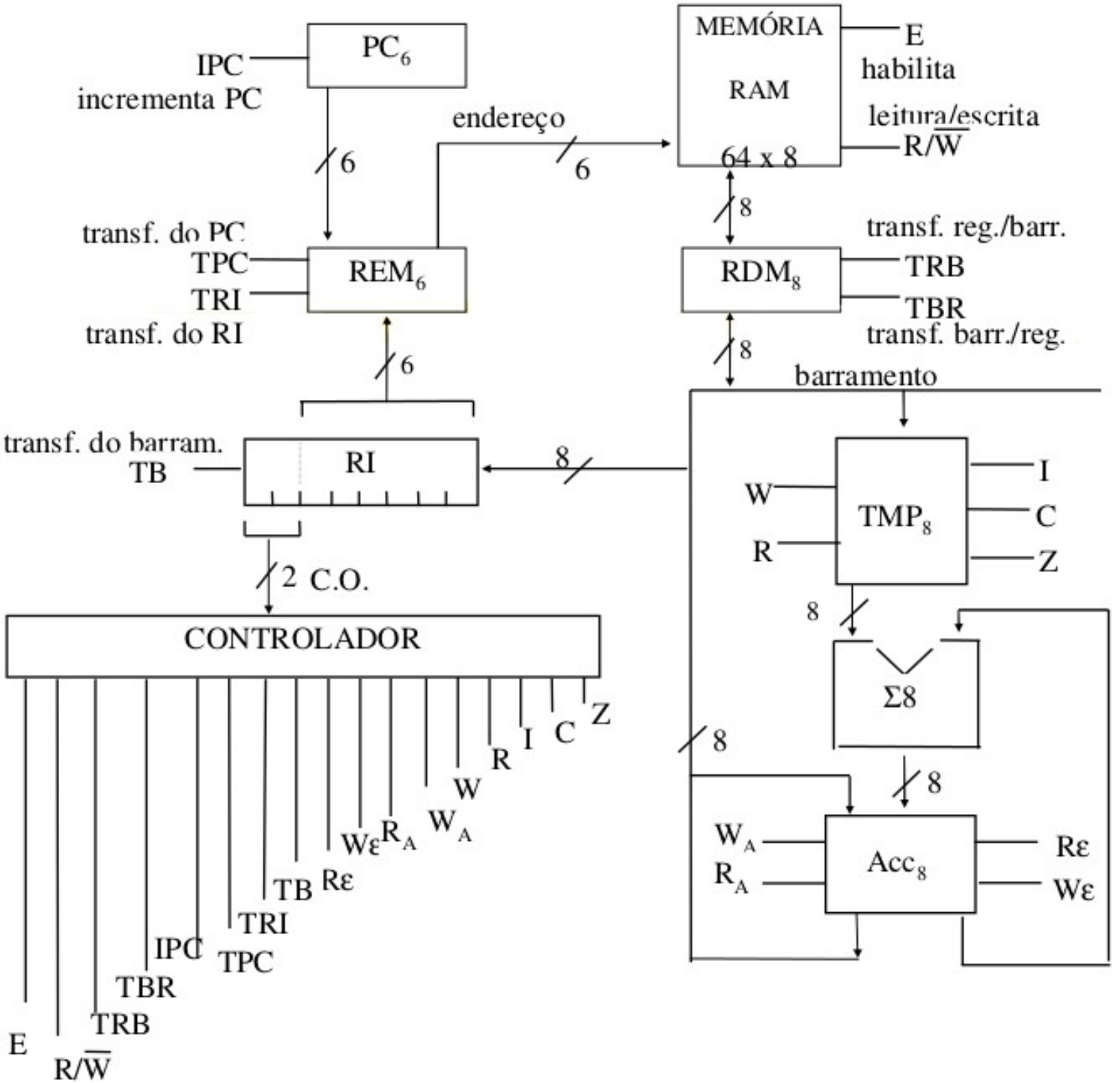
Pulso	Microoperação
1	REM ← endereço
2	RDM ← dado
3	escrita : (REM) ← (RDM) ou m ← (RDM)

onde "**(REM) ←**" significa armazenar no endereço contido no REM

Memória – outros tipos

- **Memória associativa**: o acesso é realizado pelo conteúdo e não pelo endereço.
- **Memória cache**: normalmente colocada entre memória principal e CPU e tendo característica de alta velocidade, proporciona um acesso mais rápido.
- Etc.

Computador Simples



Instruções de máquina (tipos)

- 1. Instruções de transferência de dados: transferem dados, ou blocos de dados, entre diferentes registradores ou regiões de memória.

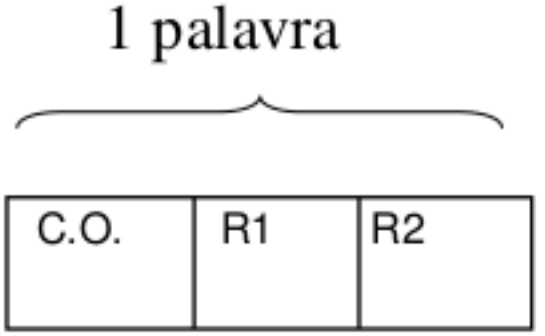
  Ex.: MOVE R1, R2 R2 ← (R1)

- 2. Instruções de tratamento de operandos: realizam operações aritméticas ou lógicas.
  - Ex.: ADD R1, R2 R2 ← (R1) + (R2)
- 3. Instruções de desvio: determinam desvio no fluxo do programa, isto é, controlam a sequência de execução do programa.
  - Ex.: JMP, Jcond, JSR - salto p/ subrotina,
    RET - retorno de subrotina
- 4. Instruções de Entrada/Saída: realizam a comunicação entre a UCP e as Interfaces de entrada e saída.
  - Exemplo: IN port Acc ← (port)

Instruções de máquina - comprimento

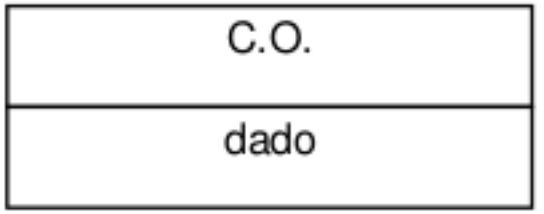
Instruções de 1 palavra :

MOVE R1, R2 ;R2 ← (R1)

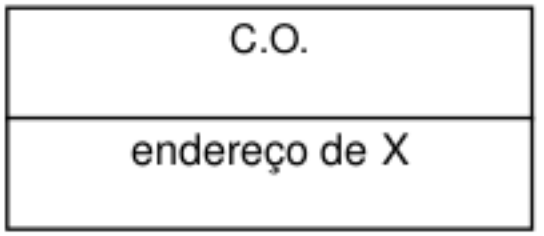


Instruções de 2 palavras :

a. ADI dado ; Acc ← (Acc) + dado



b. ADD X ; Acc ← (Acc) + (X)

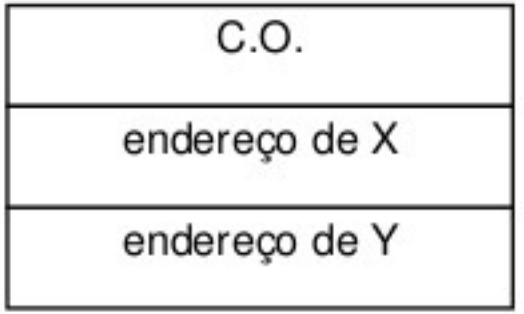


Instruções de máquina - comprimento

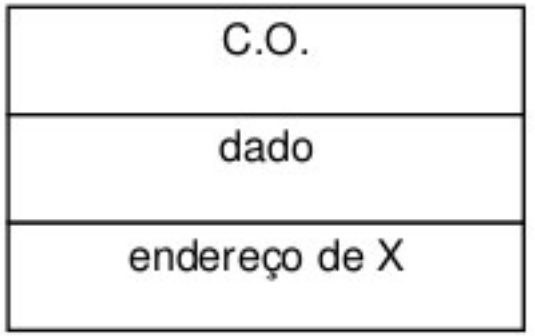
1 palavra

Instruções de 3 palavras :

a. ADD X, Y ; $Y \leftarrow (X) + (Y)$



b. ADD #dado, X ; $X \leftarrow (X) + dado$



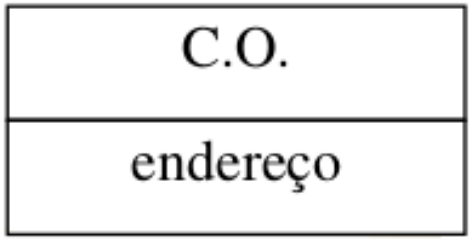
Execução de uma instrução de máquina

- **Fase de busca**: compreende o Ciclo de Máquina para a leitura do código de operação (C.O.), ou seja, da primeira palavra da instrução. Desta forma, a Fase de Busca é idêntica para qualquer instrução.
- **Fase de execução**: compreende a execução dos Ciclos de Máquina necessários para a leitura das palavras restantes da instrução, se existirem, e da execução da operação identificada pela instrução. A fase de execução é diferente para cada tipo de instrução.

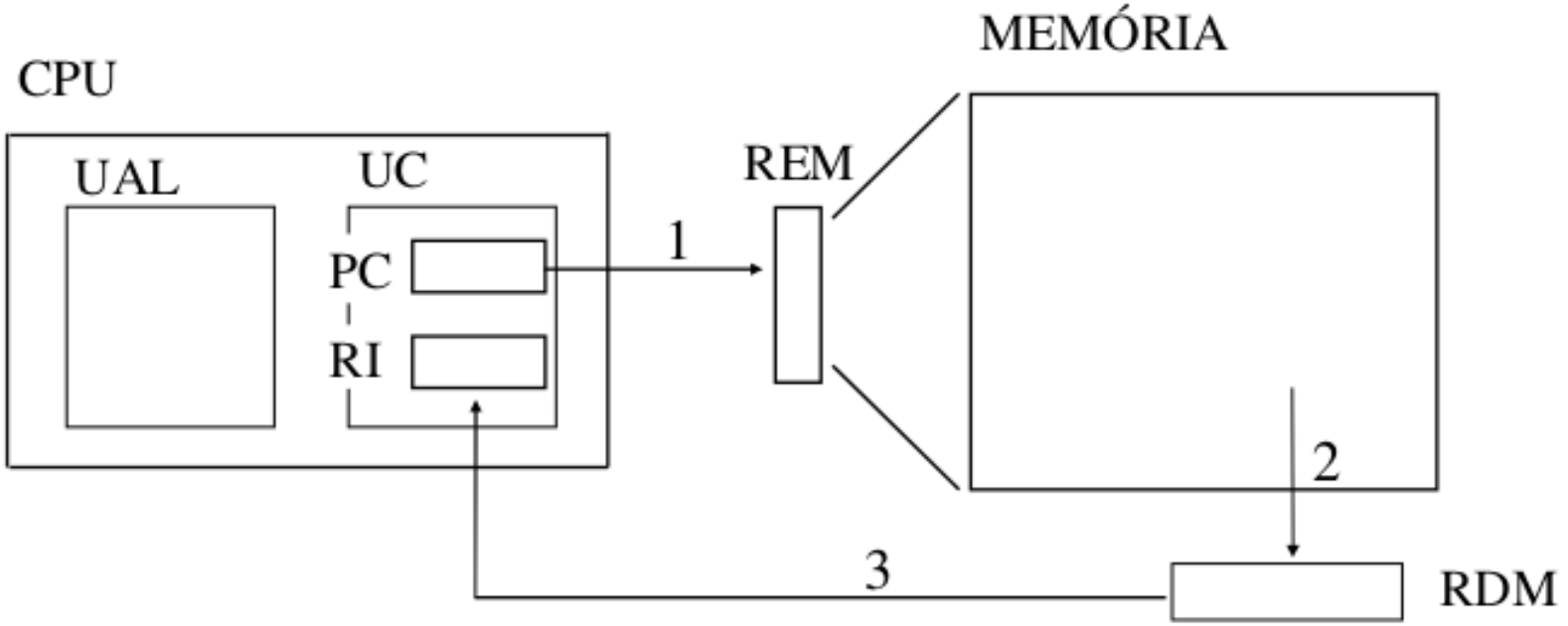
Exemplo: execução da instrução STA end

STA end ⇒ end ← (Acc)

formato da instrução



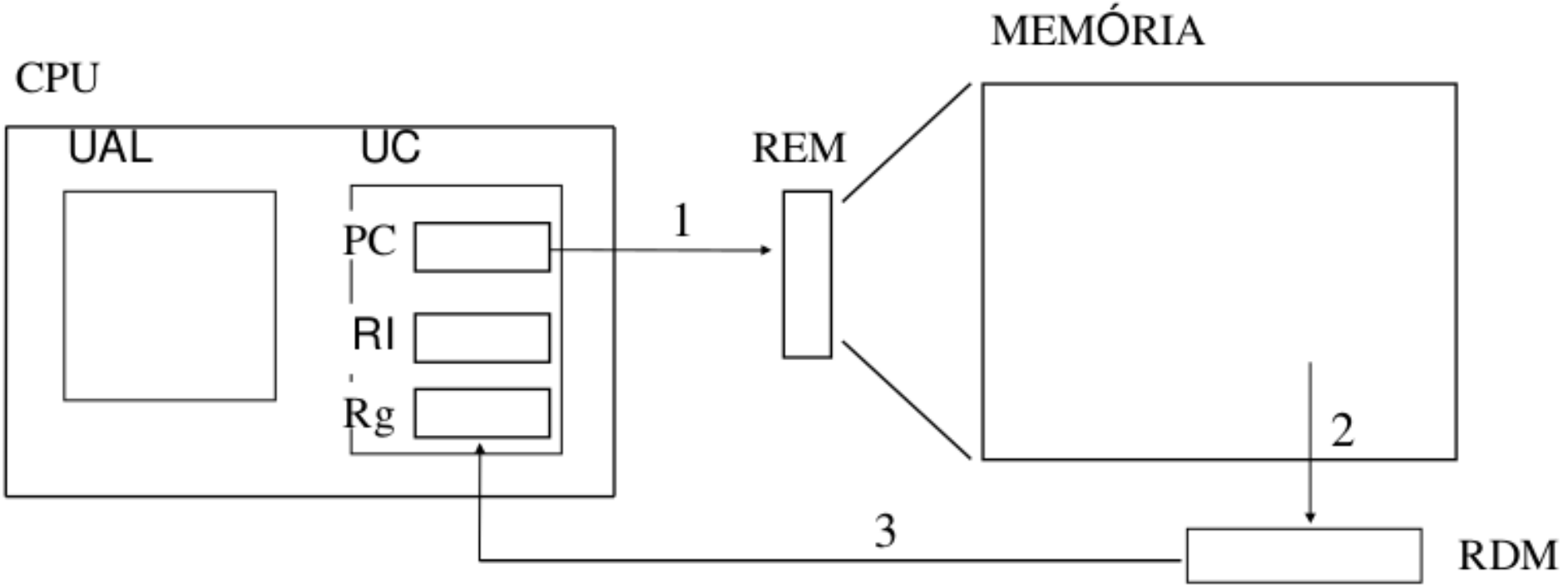
1º ciclo de máquina : FASE DE BUSCA - busca do C.O.



PULSO	**MICROOPERAÇÃO**
1.	REM ← (PC)
2.	leitura: RDM ← ((REM)) ou RDM ← (m)
	PC ← (PC) + 1 ; operação interna
3.	RI ← (RDM)

Exemplo: execução da instrução STA end

2º ciclo de máquina : Leitura do endereço

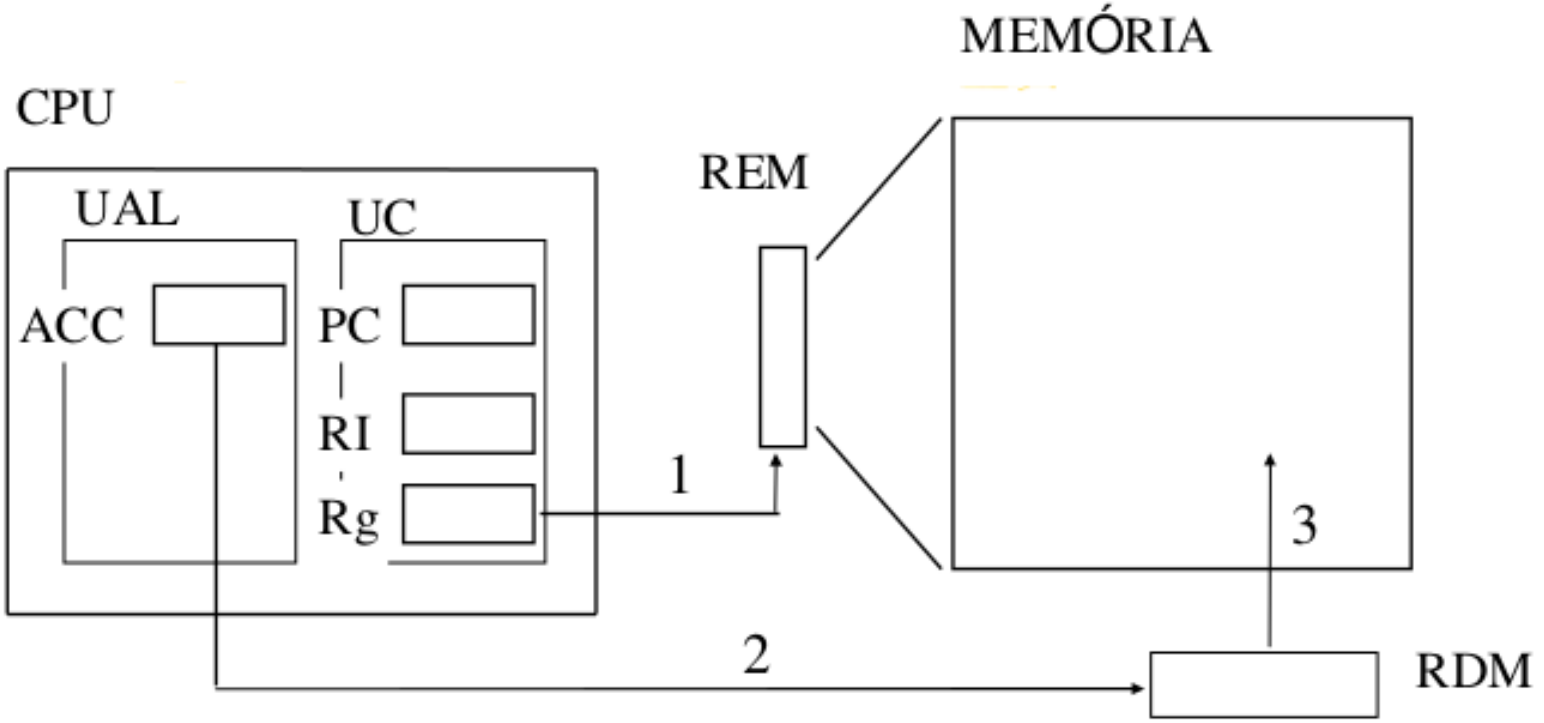


PULSO MICROOPERAÇÃO

1. REM ← (PC)
2. leitura: RDM ← ((REM)) ou RDM ← (m)
   PC ← (PC) + 1
3. Rg ← (RDM)

Exemplo: execução da instrução STA end

3º ciclo de máquina : Transferência do conteúdo de Acc para a memória (endereço end)



PULSO	MICROOPERAÇÃO
1.	REM ← (Rg)
2.	RDM ← (Acc)
3.	escrita: (REM) ← (RDM) ou m ← (RDM)

Microprograma e microinstrução

Instrução em Linguagem de Máquina	Microprograma	Microinstrução	Microoperação	Microcomando
uma instrução de máquina	um micro programa	1	REM ← (PC)	*TPC*
		2	RDM ← (m)	*E, R/$\overline{W}$,TBR*
			PC ← (PC) + 1	*IPC*
		3	:	:
		:	:	:

- Um ou um conjunto de sinais de controle definem uma **microinstrução**; uma ou mais microinstruções definem um **microprograma**.

Programa exemplo (trecho)

end.	
0	ADD 61
1	ADD 59
2	SUB 60
3	SUB 62

endereço	
000000	01111101
000001	01111011
000010	10111100
000011	10111110

59	30
60	-22
61	-8
62	18

111011	00011110
111100	11101010
111101	11111000
111110	00010010

Execução da instrução: SUB 60

- Ciclo de busca:

Microoperação	**Pulso**	**Microcomando**
REM ← (PC)	1	*TPC*
RDM ← (m) PC ← (PC) + 1	2	*E, R/W* *IPC*
RI ← (RDM)	3	*TRB, TB*

Execução da instrução: SUB 60

- Ciclo de execução:

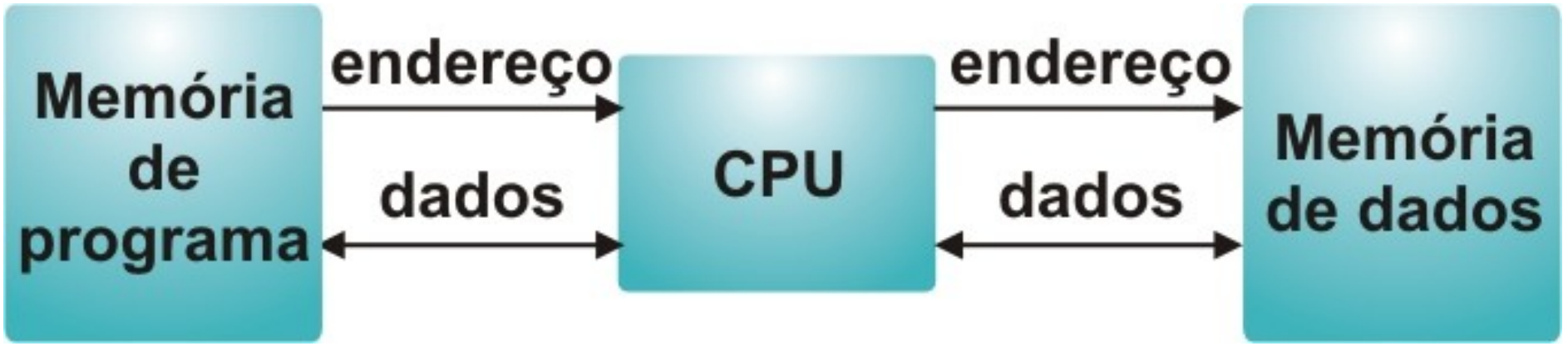
Microoperação	Pulso	Microcomando
REM ← (RI.end)	1	*TRI*
RDM ← (m)	2	E, R/W
TMP ← (RDM)	3	*TRB, W*
TMP ← $\overline{(TMP)}$	4	*C*
TMP ← (TMP) + 1	5	*I*
Σ ← (Acc) + (TMP)	6	*R, Rε*
Acc ← Σ	7	*Wε*

Faça você mesmo

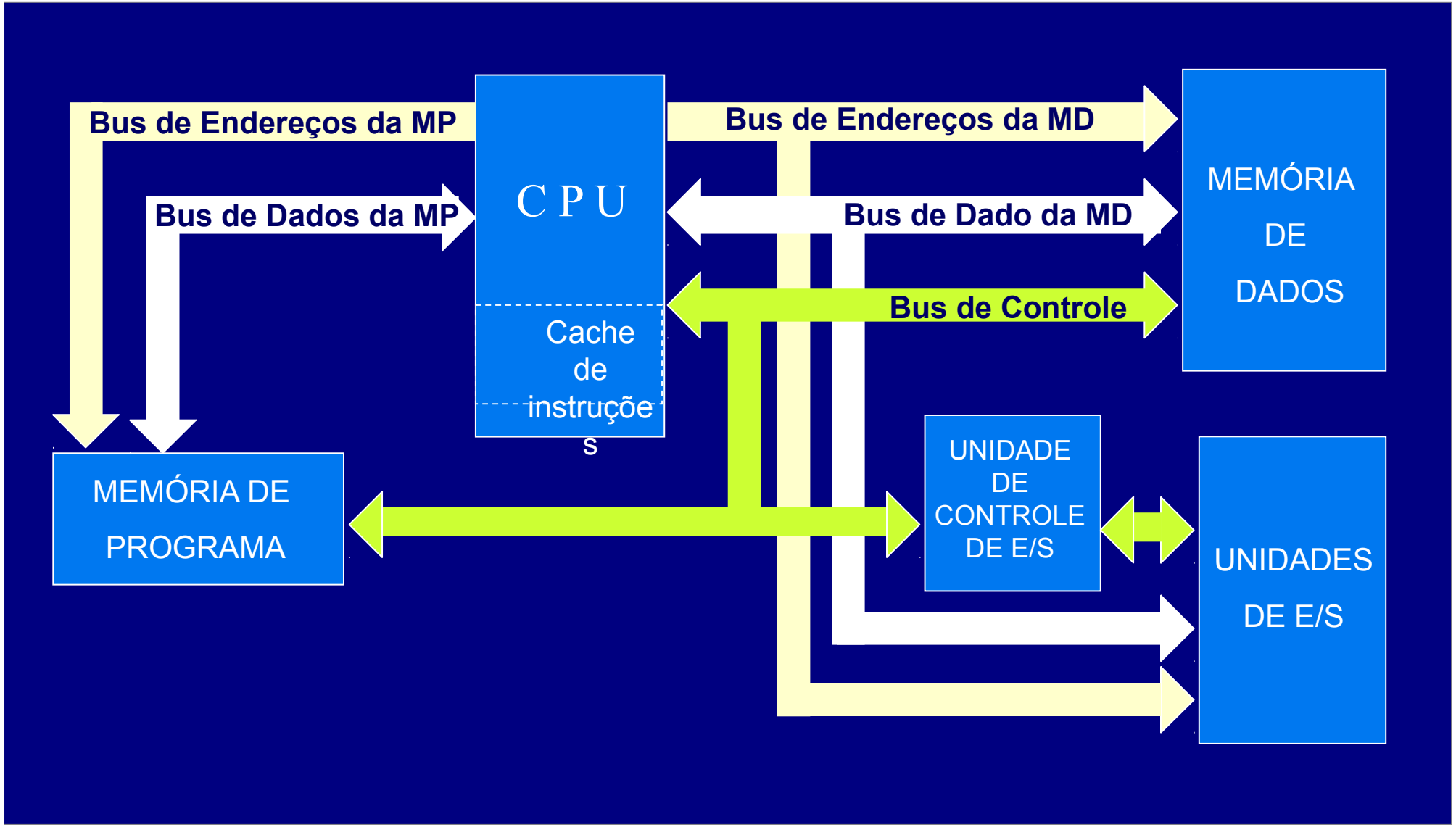
- Logisim:
  - Ferramenta educacional para projeto e simulação de circuitos digitais.
  - http://ozark.hendrix.edu/~burch/logisim/
- Fritzing:
  - Outra ferramenta para projeto de circuitos digitais. Inclui suporte ao Arduino.
  - http://fritzing.org/

Modelo de Harvard

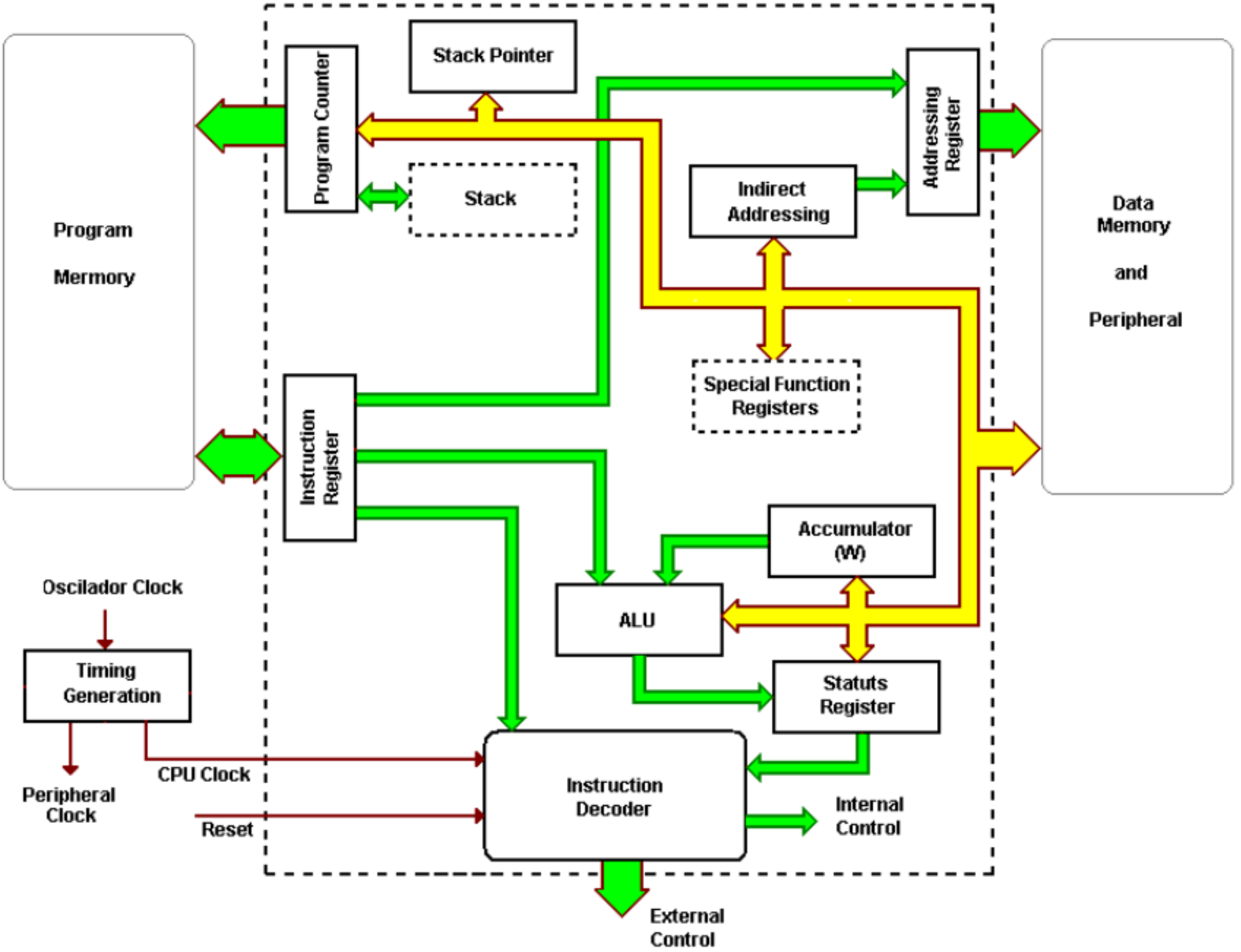
- A principal característica é ter a memória de programa separada da memória de dados e acessada por um barramento independente. A maioria dos DSPs usa Harvard pela sua maior largura de banda.



Modelo Harvard
(Analog Devices)



Exemplo de CPU (Harvard)



Defina e explique alguns conceitos relacionados à Microprogramação

- Formato de microinstruções: *horizontais*, *verticais*, *diagonais* (explicar as vantagens e desvantagens de cada formato).
- Classificação das microinstruções quanto à execução: monofásicas e polifásicas.
- Defina a diferença entre microinstruções polifásicas síncronas e assíncronas.