Computador Hipotético Ramsés
Parte II

Disciplina: Introdução à Arquitetura de Computadores

Luciano Moraes Da Luz Brum

Universidade Federal do Pampa – Unipampa – Campus Bagé

Email: lucianobrum18@gmail.com

Tópicos

- **Instruções de desvio;**
- **Modo de Endereçamento Indexado;**
- **Instruções por tipo e Códigos de Condição;**
- **Subrotinas;**
- **Passagem de parâmetros;**
- **Comparações entre arquiteturas hipotéticas;**
- **Organização do processador hipotético Ramsés;**

Instruções de Desvio

- Desvios nos modos de endereçamento:
  - Direto;
  - Indireto;
  - Indexado;

Instruções de Desvio

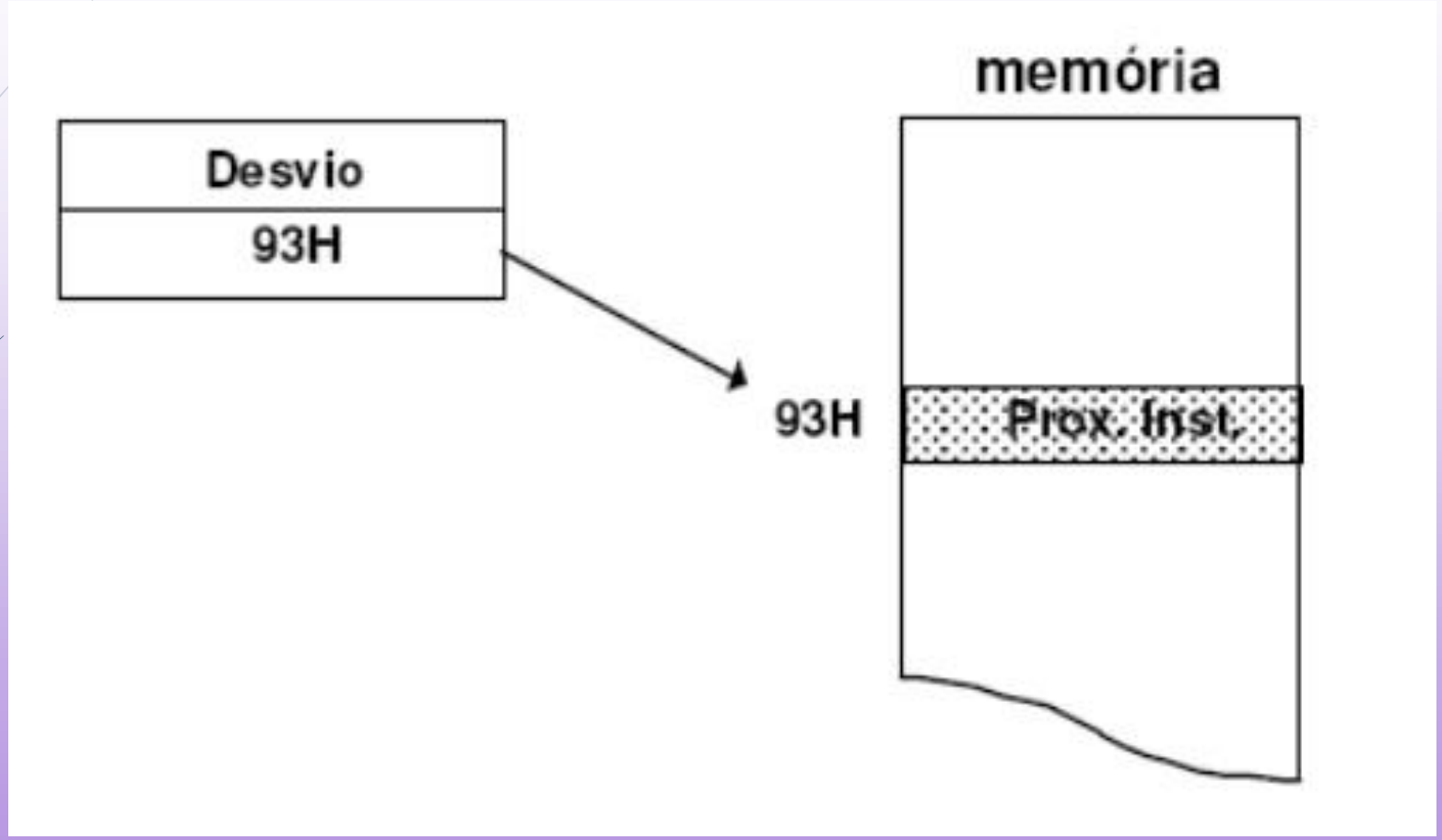


Figura 1: Modo de endereçamento direto.

Instruções de Desvio

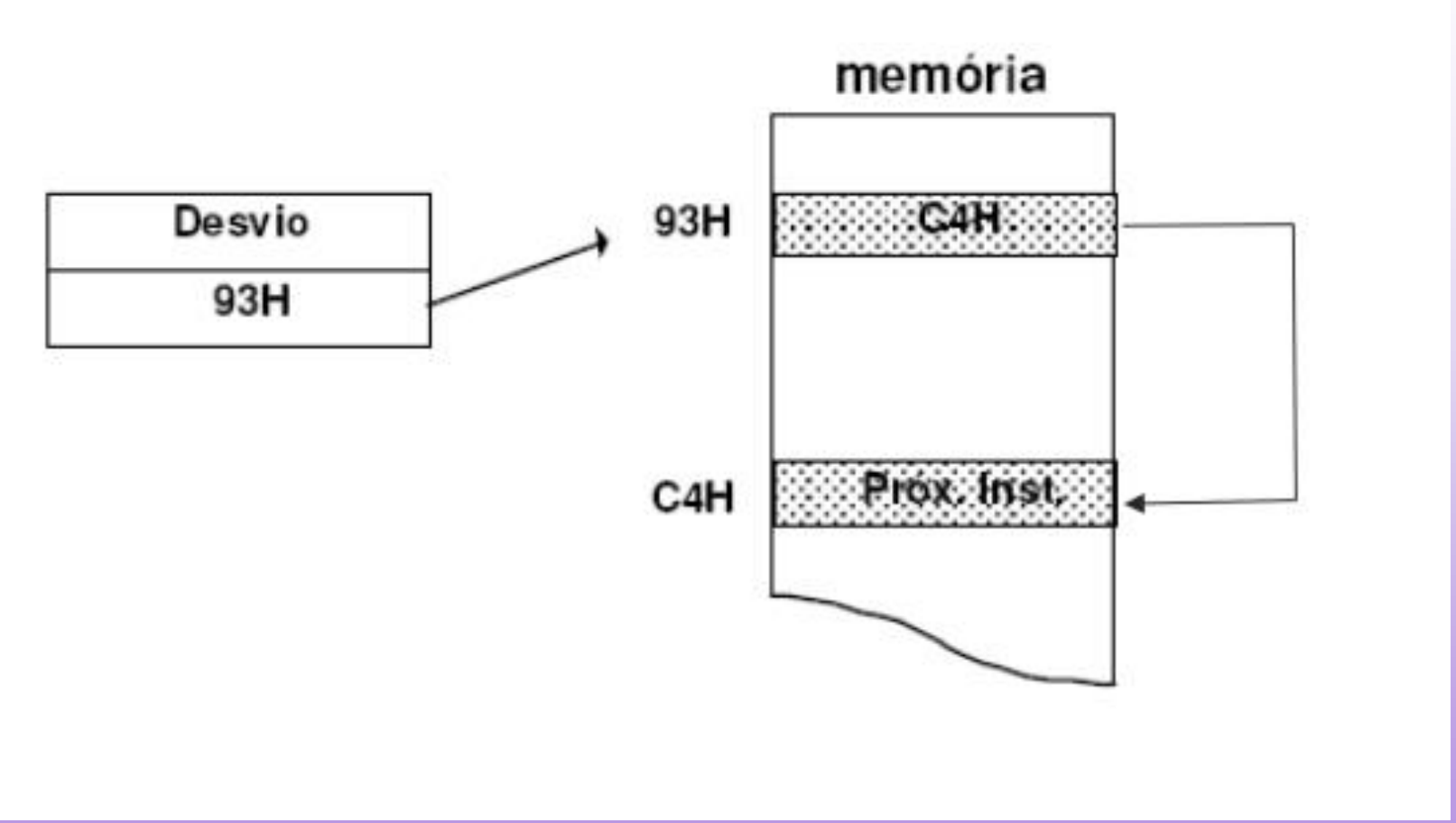


Figura 2: Modo de endereçamento indireto.

Instruções de Desvio

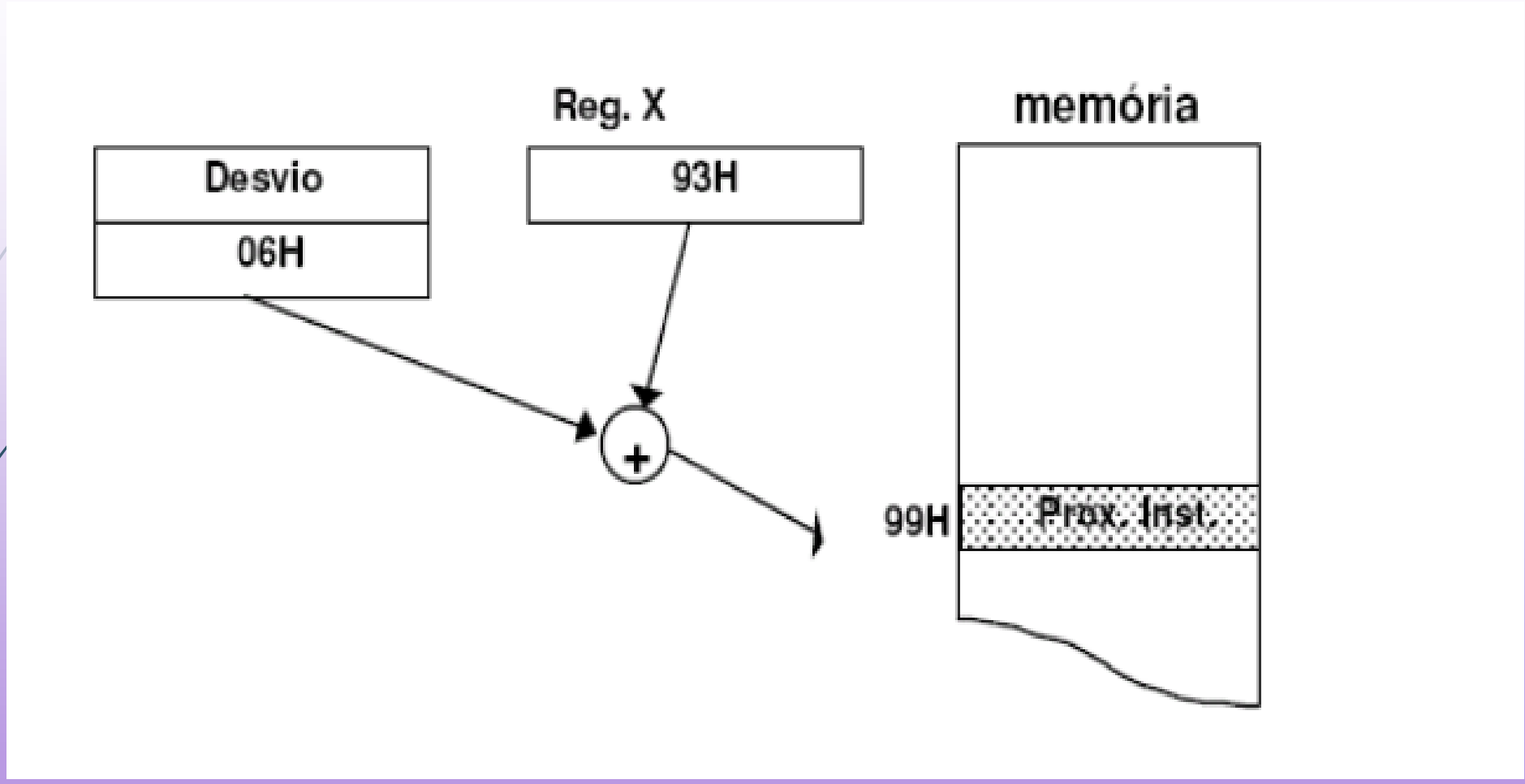


Figura 3: Modo de endereçamento indexado.

Tópicos

- ~~Instruções de desvio;~~
- **Modo de Endereçamento Indexado;**
- **Instruções por tipo e Códigos de Condição;**
- **Subrotinas;**
- **Passagem de parâmetros;**
- **Comparações entre arquiteturas hipotéticas;**
- **Organização do processador hipotético Ramsés;**

Modos de Endereçamento

- Exemplos com endereçamento indexado:

- Acessar elemento de um array de n bytes, indexado de 0 a n-1.

- Quatro situações:
  - Endereço de vetor fixo, índice fixo
  - Endereço de vetor fixo, índice variável
  - Endereço de vetor variável, índice fixo
  - Endereço de vetor variável, índice variável

Modos de Endereçamento

- Situação (1): Endereço de vetor fixo, índice fixo.
  - Exemplo: vetor inicia no endereço 132, e o índice é 7
  - Endereço final do operando fixo: 132+7=139
  - Usa-se modo direto:
    - LDR A 139

Modos de Endereçamento

- Situação (2): Endereço de vetor fixo, índice variável.
  - Exemplo: vetor inicia no endereço 132, e o índice é dado pela variável i
  - Endereço final do operando fixo: 132+i
  - Usa-se modo indexado:
    - LDR X i
    - LDR A 132,X

Modos de Endereçamento

- Situação (3): Endereço de vetor variável, índice fixo.
  - Exemplo: vetor inicia no endereço e, e o índice é 7
  - Endereço final do operando fixo: e+7
  - Usa-se modo indexado:
    - LDR X e
    - LDR A 7,X

Modos de Endereçamento

- Situação (4): Endereço de vetor variável, índice variável.
  - Exemplo: vetor inicia no endereço e, e o índice é dado pela variável i
  - Endereço final do operando fixo: e+i
  - Usa-se modo indexado:
    - LDR X e
    - ADD X i
    - LDR A 0,X

Tópicos

- ~~Instruções de desvio;~~
- ~~Modo de Endereçamento Indexado;~~
- **Instruções por tipo e Códigos de Condição;**
- **Subrotinas;**
- **Passagem de parâmetros;**
- **Comparações entre arquiteturas hipotéticas;**
- **Organização do processador hipotético Ramsés;**

Ramsés: Instruções por tipo

- Aritiméticas e lógicas:
  - ADD r end ; r ← MEM(end) + r
  - SUB r end
  - OR r end
  - AND r end
  - NOT r ; r ← NOT(r) – neg. bit-a-bit
  - NEG r ; r ← -r

Ramsés: Instruções por tipo

- Acesso à memória:
  - STR r end ; MEM(end) ← r
  - LDR r end ; r ← MEM(end)

- Instruções de controle de fluxo
  - JMP end ; PC ← end
  - JN end ; IF N=1 THEN PC ← end
  - JZ end ; IF Z=1 THEN PC ← end
  - JC end ; IF C=1 THEN PC ← end
  - JSR end ; MEM(end) ← PC; PC ← end+1

Ramsés: Instruções por tipo

- Instrução de deslocamento de bits
  - SHR r ; r ← r/2

- Considerar as limitações impostas pelo formato de instrução adotado.

Ramsés: Códigos de Condição

Instrução	Códigos de Condição
0010 – LDR	N, Z
0011 – ADD	N, Z, C
0100 – OR	N, Z
0101 – AND	N, Z
0110 – NOT	N, Z
0111 – SUB	N, Z, C Obs.: carry=1: não houve borrow; carry=0: houve borrow
1101 – NEG	N, Z, C
1110 – SHR	N, Z, C Obs.: carry = bit menos significativo (deslocado para fora do registrador)

Figura 4: Instruções e códigos de condição afetados.

Tópicos

- ~~Instruções de desvio;~~
- ~~Modo de Endereçamento Indexado;~~
- ~~Instruções por tipo e Códigos de Condição;~~
- **Subrotinas;**
- **Passagem de parâmetros;**
- **Comparações entre arquiteturas hipotéticas;**
- **Organização do processador hipotético Ramsés;**

Subrotinas

- Chamada: JSR end (Jump to SubRoutine).
- Retorno: JMP end,I (desvio com endereço indireto).
- O endereço de retorno é armazenado na primeira palavra da subrotina.
- Execução de JSR end:
  - MEM(end) ← PC+1 ;
  - PC ← end + 1 ;

Subrotinas

- Uma subrotina para trocar o sinal de um número (complemento de B):
  - End. Instrução
  - 60 NOP ; #aqui ficará o end. de retorno
  - 61 NOT B
  - 62 ADD B #1
  - 64 JMP 60,I ; #retorno da subrotina

- Chamada da subrotina: JSR 60

Subrotinas

...

10 JSR 60 ; #guarda end. 12 na posição 60 da mem

12 ... ; #instrução a executar após o retorno

...

16 JSR 60 ; #guarda end. 18 na posição 60 da mem

18 ... ; #instrução a executar após o retorno

...

60 NOP ; #posição onde será guardado o end. retorno

61 xxx ; #posição onde fica a primeira instrução da rotina

- Em ambos os casos, após JSR, PC = 61

Subrotinas: Limitações

- Não permite recursividade;
- Não permite reentrância;
- Mas permite chamadas aninhadas;

Tópicos

- ~~Instruções de desvio;~~
- ~~Modo de Endereçamento Indexado;~~
- ~~Instruções por tipo e Códigos de Condição;~~
- ~~Subrotinas;~~
- **Passagem de parâmetros;**
- **Comparações entre arquiteturas hipotéticas;**
- **Organização do processador hipotético Ramsés;**

Passagem de Parâmetros

- Duas formas:
  - Por valor: Passando o valor de uma variável.
  - Por nome: Passando o endereço da variável.

Passagem de Parâmetros

- Duas formas:
  - Por valor: Passando o valor de uma variável.
  - Por nome: Passando o endereço da variável.

Passagem de Parâmetros

- Exemplo: Multiplicação:
  - Programa principal usa três endereços:
    - Primeiro_operando.
    - Segundo_operando.
    - Resultado.
  - Subrotina usa três endereços:
    - Op1.
    - Op2.
    - Resultado.

Passagem de Parâmetros

- Por registradores

Programa principal

```
LDR A primeiro_operando
LDR B segundo_operando
JSR multiplica
STR A resultado
```

Subrotina

```
NOP
STR A op1
STR B op2
<multiplicação>
LDR A resul
JMP multiplica,I
```

Passagem de Parâmetros

- Por posições de memória

Programa principal

```
LDR A primeiro_operando
STR A param1
LDR A segundo_operando
STR A param2
JSR multiplica
LDR A param3
STR A resultado
```

Subrotina

```
NOP
LDR A param1
STR A op1
LDR A param2
STR A op2
<multiplicação>
LDR A resul
STR A param3
JMP multiplica,I
```

Passagem de Parâmetros

- Por posições de memória

Programa principal

JSR multiplica
<valor do primeiro operando>
<valor do segundo operando>
<endereço do resultado>
<< instrução seguinte >>

Passagem de Parâmetros

```
multiplica:   NOP
              LDR A multiplica,I     ; obtém valor do primeiro operando
              STR A op1
              LDR A multiplica       ; atualiza endereço de retorno
              ADD A #1
              STR A multiplica

              LDR A multiplica,I     ; obtém valor do segundo operando
              STR A op2
              LDR A multiplica       ; atualiza endereço de retorno
              ADD A #1
              STR A multiplica

              <multiplicação>

              LDR A multiplica,I     ; obtém endereço do terceiro operando
              STR A ponteiro
              LDR B resul            ; obtém resultado da rotina
              STR B ponteiro, I      ; salva resultado no endereço do terceiro parâmetro
              LDR A multiplica       ; atualiza endereço de retorno
              ADD A #1
              STR A multiplica

              JMP multiplica, I      ; retorna da subrotina
```

Passagem de Parâmetros

```
multiplica:   NOP
              LDR X multiplica      ; X aponta para área de parâmetros

              LDR A 0,X             ; obtém valor do primeiro operando
              STR A op1

              LDR A 1,X             ; obtém valor do segundo operando
              STR A op2

              <multiplicação>

              LDR A 2,X             ; obtém endereço do terceiro parâmetro
              STR A ponteiro        ; ponteiro para guardar resultado
              LDR B resul           ; obtém produto calculado pela subrotina
              STR B ponteiro, I     ; salva resultado no endereço apontado

              ADD X #3
              STR X multiplica      ; atualiza endereço de retorno

              JMP multiplica, I     ; retorna da subrotina
```

Tópicos

- ~~Instruções de desvio;~~
- ~~Modo de Endereçamento Indexado;~~
- ~~Instruções por tipo e Códigos de Condição;~~
- ~~Subrotinas;~~
- ~~Passagem de parâmetros;~~
- **Comparações entre arquiteturas hipotéticas;**
- **Organização do processador hipotético Ramsés;**

Exercício: o que faz esse código?

Endereço	Instrução	
0	LDA 132	; inicializa (zera) o total
2	STA 130	
4	LDA 129	; inicializa ponteiro
6	STA 17	
8	LDA 128	; inicializa contador
10	STA 131	
12	JZ 34	; testa se contador é zero
14	LDA 130	; carrega total no acumulador
16	ADD 17	; soma com posição de memória
18	STA 130	; atualiza total
20	LDA 17	; incrementa ponteiro
22	ADD 134	
24	STA 17	
26	LDA 131	; decrementa contador
28	ADD 133	
30	STA 131	
32	JMP 12	; retorna ao início do laço
34	HLT	
128	n	número de posições
129	e	endereço inicial
130	total	total
131	contador	contador
132	0	constante zero
133	255	constante -1
134	1	constante 1

Exercício: o que faz esse código?

Endereço	Instrução	
0	LDR A #0	; inicializa (zera) o total
2	LDR X 129	; inicializa ponteiro
4	LDR B 128	; inicializa contador
6	JZ 16	; testa se contador é zero
8	ADD A 0,X	; soma com posição de memória
10	ADD X #1	; incrementa ponteiro
12	SUB B #1	; decrementa contador
14	JMP 6	; retorna ao início do laço
16	STR A 130	; atualiza total
18	HLT	
128	n	número de posições
129	e	endereço inicial
130	total	

Comparação: Neander vs Ramsés

Tabela 1: Comparação de execução Neander vs Ramsés.

	Neander	Ramsés
Programa (bytes)	**35**	**19**
Dados (bytes)	**7**	**3**
Instruções	**11*n+8**	**5*n + 6**
Leituras	**27*n + 18**	**11*n + 13**
Escritas	**4*n + 3**	**1**
Instruções (n=30)	**338**	**156**
Acessos à memória(n=30)	**951**	**344**

Fonte: Weber, 2001.

Comparação: Neander vs Ramsés

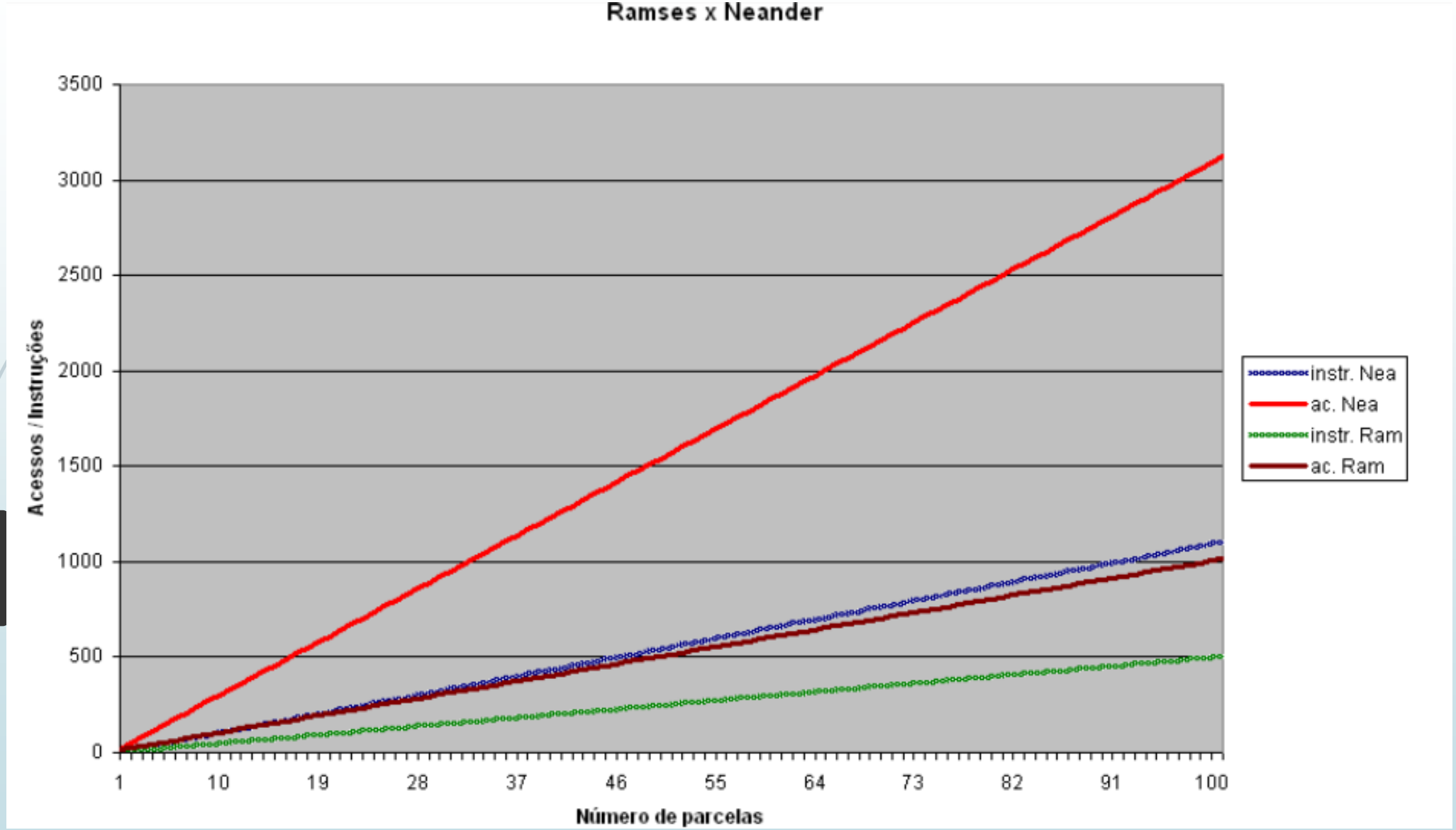


Figura 5: Comparação de execução Neander vs Ramsés.

Problemas com Neander

- Arquitetura do Neander – Críticas?
  - Possui apenas 1 modo de endereçamento (Direto Absoluto).
  - Possui apenas 1 registrador de uso geral (Acumulador).
  - Possui apenas 2 flags de status da ULA (Flip-flops N e Z).
  - Possui apenas 11 instruções de máquina (incluindo NOP e HLT).
  - Não possui flags de “vai-um” (Carry In, Carry Out).
  - Não possui instruções de desvio/retorno de sub-rotina (JSR, RTS).
  - Não possui uma pilha auxiliar para dados/endereços (Push, Pop).
  - Não possui instruções de acesso imediato a memória (LDA #).
  - Não possui instruções de acesso indexado a memória (LDA $,X).
  - Não possui instruções dedicadas de E/S (In, Out).

Comparações entre Diferentes Arquiteturas

Evolução do Neander... Ahmes, Ramses, Cesar

Quadro comparativo

Arquitetura	Endereços	Dados	Nro. Instruções	Registradores
NEANDER	8 bits 256 bytes	8 bits Compl.2	11 instruções (OpCode: 4bits)	AC, PC, IR, Flags (N,Z) REM, RDM
AHMES	8 bits	8 bits	24 instruções (Neander ext.)	PC, IR, REM, RDM Flags (N, Z, C, B, V)
RAMSES	8 bits	8 bits	Modos de End. 4 modos x 16 instr.	PC, IR, RA, RB, RX Flags (N, Z, V, C)
CESAR	16 bits 64 Kbytes	16 bits	Inúmeras	R0 a R6 (uso geral) R7 (PC)

Figura 6: Comparação entre arquiteturas hipotéticas.

Tópicos

- ~~Instruções de desvio;~~
- ~~Modo de Endereçamento Indexado;~~
- ~~Instruções por tipo e Códigos de Condição;~~
- Subrotinas;
- Passagem de parâmetros;
- Comparações entre arquiteturas hipotéticas;
- **Organização do processador hipotético Ramsés;**

Arquitetura de Computador

- Trata do comportamento funcional de um computador do ponto de vista do programador. São os atributos do sistema que são visíveis a este.

- Conjunto de instruções, número de bits usados para representação de dados, mecanismos de E/S, técnicas de endereçamento, etc.

- ex: Existe uma instrução de multiplicação?

Organização de Computador

- Trata dos aspectos transparentes ao programador. Refere-se às unidades operacionais e suas interconexões que implementam sua arquitetura:

- Sinais de controle, interfaces entre o computador e os periféricos, tecnologia de memória, tamanho da memória física, frequência de clock, etc.

- ex: Existe uma unidade de multiplicação no hardware ou esta é implementada através do uso sucessivo do mecanismo de soma?

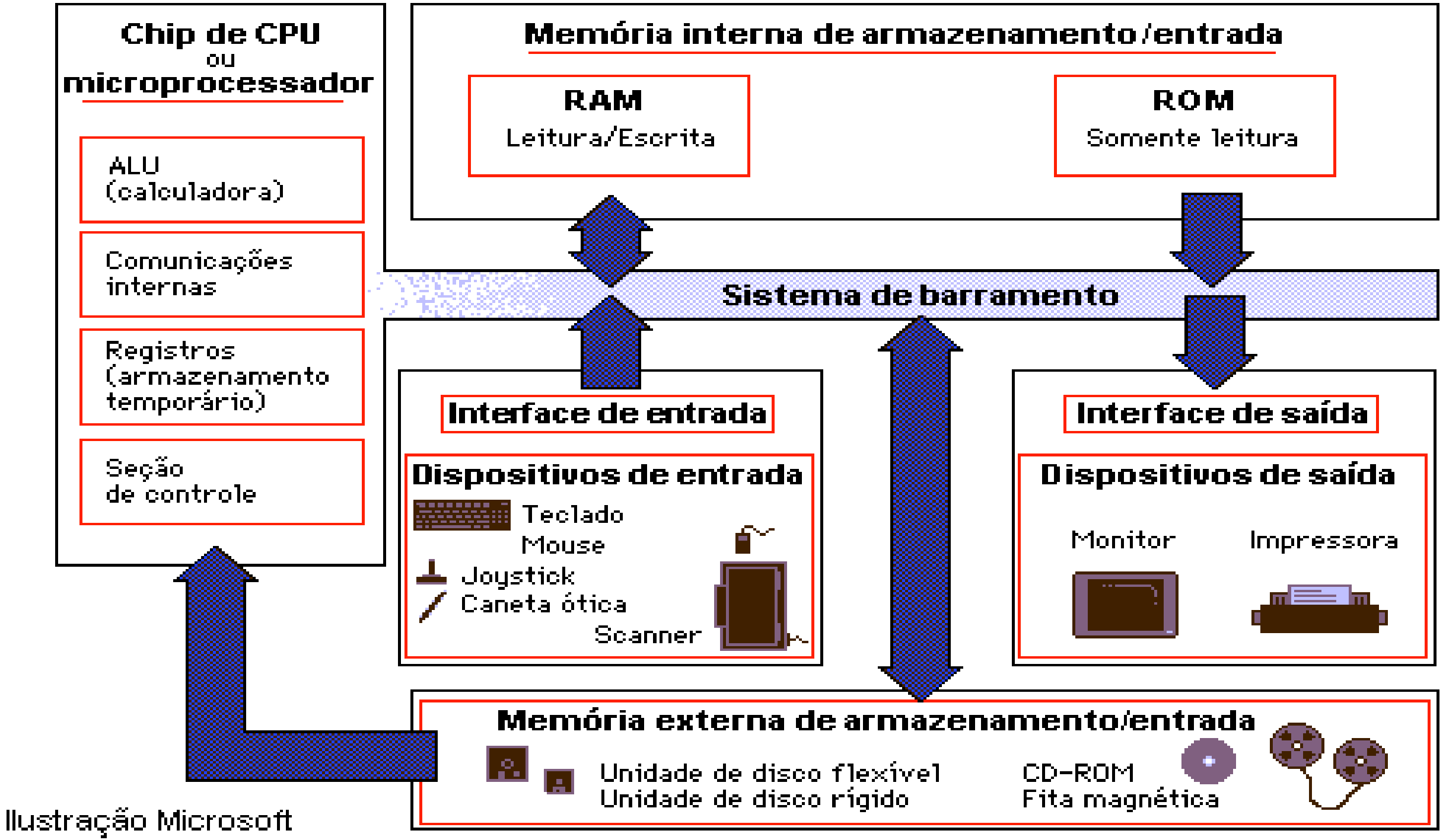
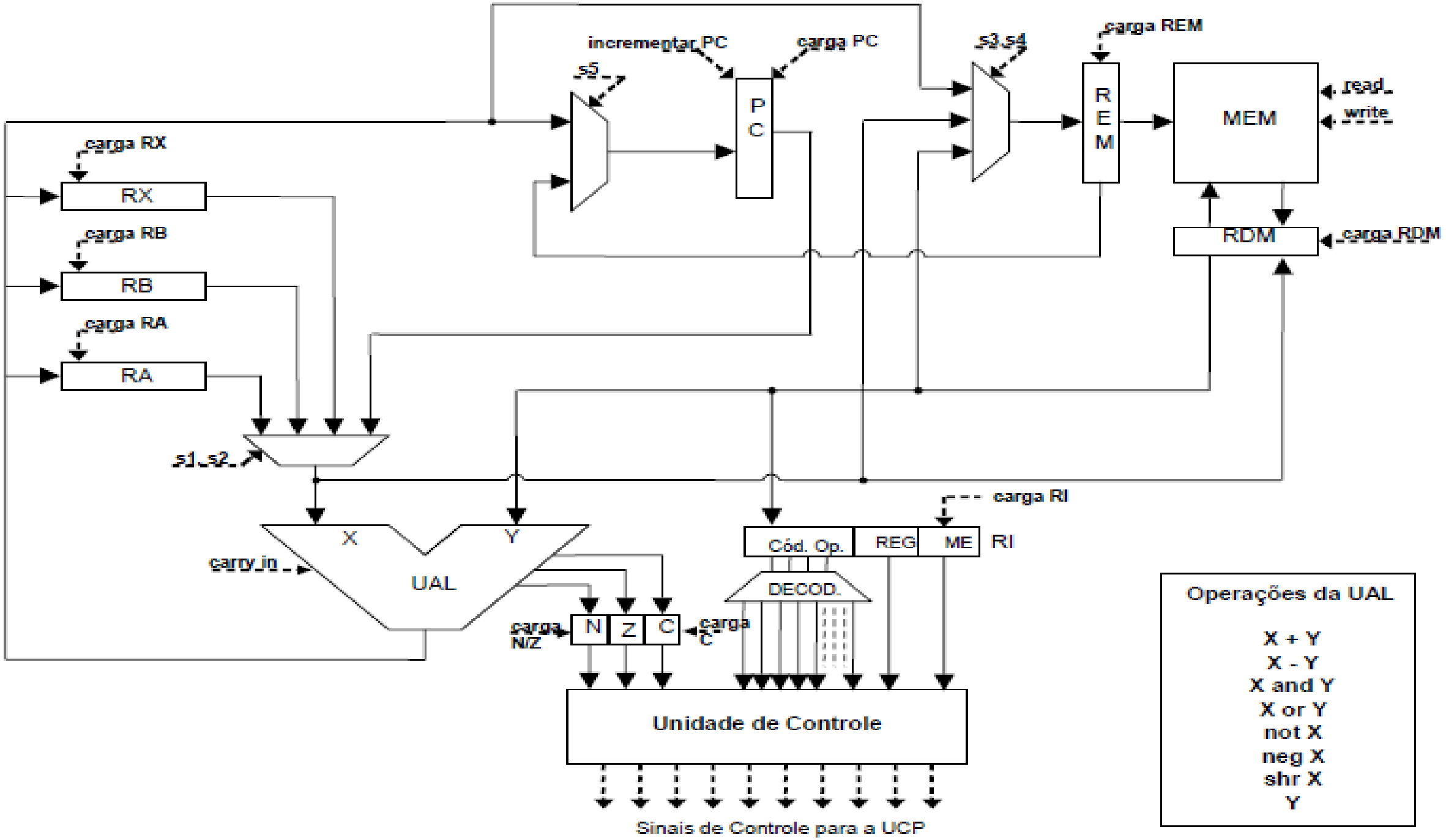
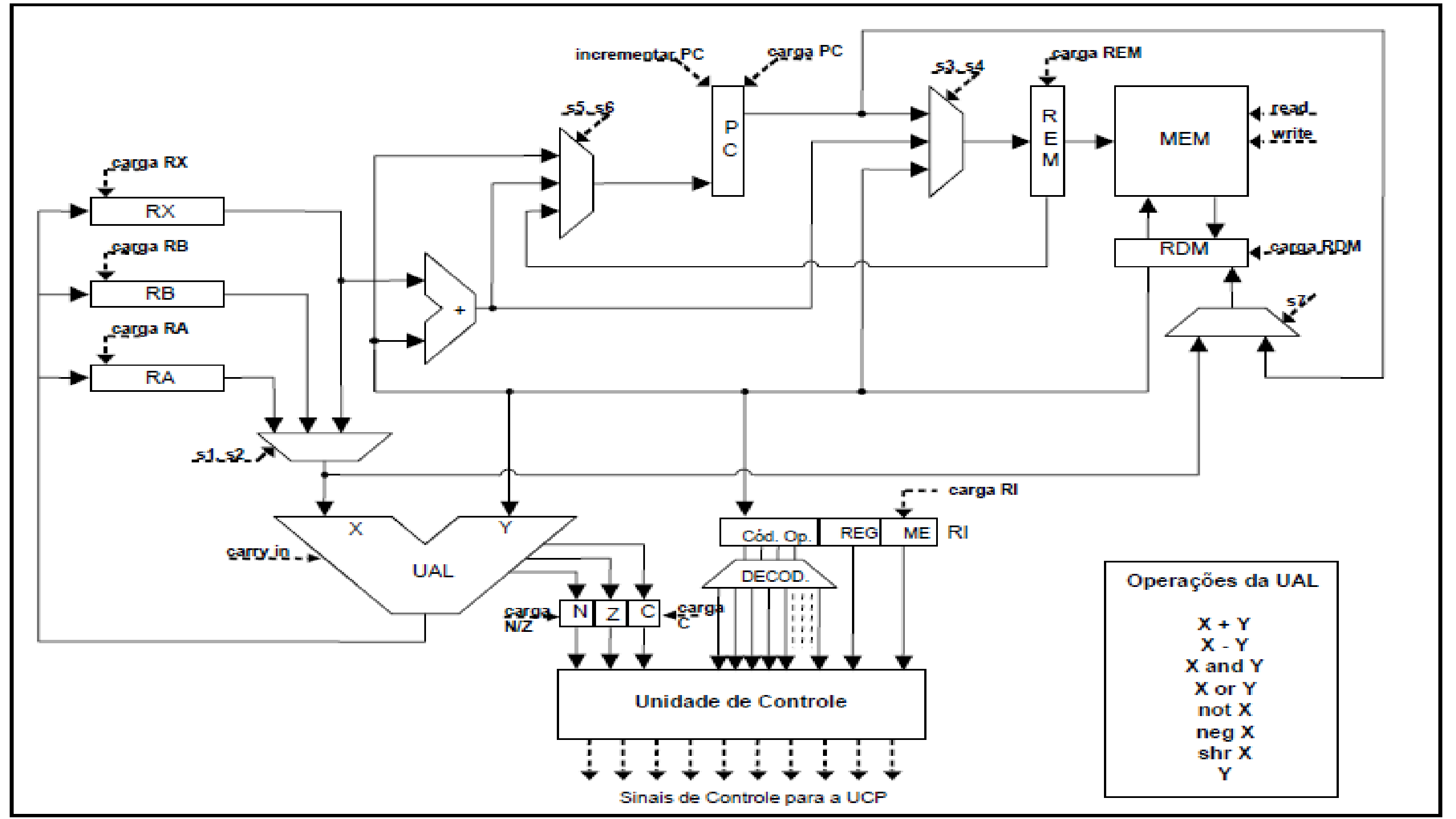
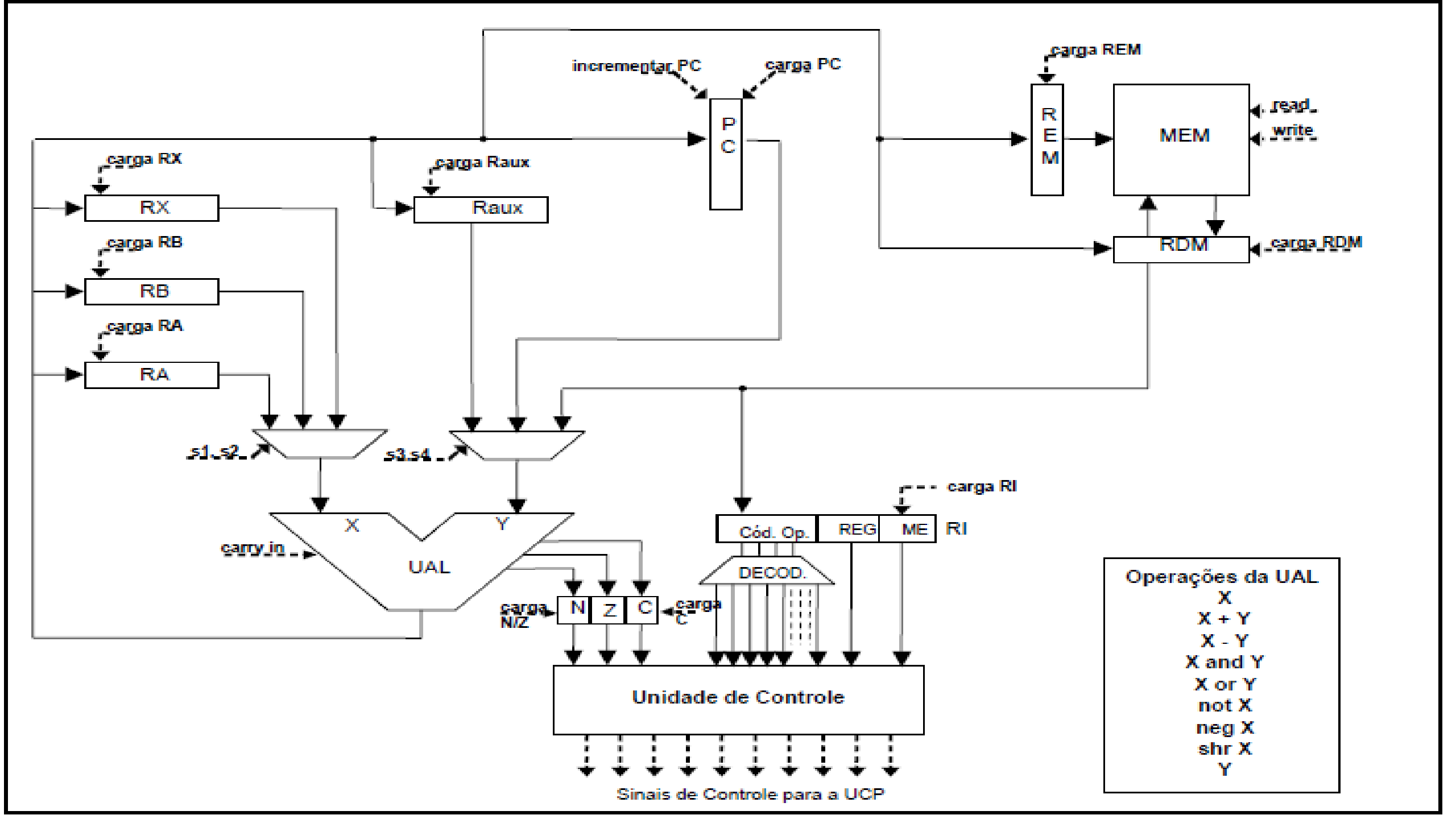


Ilustração Microsoft


incrementar PC
carga PC
s5
s3 s4
carga REM
P
C
R
E
M
MEM
read
write
carga RX
RX
carga RB
RB
carga RA
RA
RDM
carga RDM
s1 s2
carga RI
X
Y
carry_in
UAL
Cód. Op.
REG
ME
RI
DECOD.
carga N/Z
N
Z
C
carga C
Unidade de Controle
Sinais de Controle para a UCP
Operações da UAL
X + Y
X - Y
X and Y
X or Y
not X
neg X
shr X
Y


incremeptar PC
carga PC
s3_s4
carga REM
s5_s6
P
C
R
E
M
MEM
read
write
carga RX
RX
carga RB
RB
carga RA
RA
+
RDM
carga RDM
s7
s1_s2
carga RI
X
Y
Cód. Op.
REG
ME
RI
carry_in
UAL
DECOD.
carga N/Z
N
Z
C
carga C
Unidade de Controle
Sinais de Controle para a UCP
Operações da UAL
X + Y
X - Y
X and Y
X or Y
not X
neg X
shr X
Y


incrementar PC
carga PC
carga REM
PC
REM
MEM
read
write
carga RX
RX
carga Raux
Raux
carga RB
RB
carga RA
RA
RDM
carga RDM
s1,s2
s3,s4
X
Y
carry in
UAL
carga RI
Cód. Op.
REG
ME
RI
DECOD.
carga N/Z
N
Z
C
carga C
Unidade de Controle
Sinais de Controle para a UCP
Operações da UAL
X
X + Y
X - Y
X and Y
X or Y
not X
neg X
shr X
Y

Finalizando...

- **Próximas Aulas:**

  - **Organização de Computadores e o ciclo de busca-decodificação-execução de instruções: juntando e misturando tudo !!**

Exercícios

1. **Explique a diferença entre as seguintes instruções:**
   - **LDR B 128**
   - **LDR B 128,I**
   - **LDR B #128**
   - **LDR B 128,X**

2. **Se a posição 128 tem o valor 130, a 130 tem o valor 128 e o reg. X tem o valor 2, qual o conteúdo dos registradores após a execução de cada uma das instruções acima?**

Exercícios

3. **Faça um programa que multiplique duas variáveis de 8 bits positivas e forneça o resultado em 16 bits.**

- **Posição 128: multiplicando.**
- **Posição 129: multiplicador.**
- **Posição 130: Resultado (byte mais significativo).**
- **Posição 131: Resultado (byte menos significativo).**

Exercícios

4. **Faça um programa que multiplique duas variáveis de 8 bits positivas e forneça o resultado em 8 bits, indicando overflow.**

- **Posição 128: multiplicando.**
- **Posição 129: multiplicador.**
- **Posição 130: Resultado.**
- **Posição 131: conteúdo = 0H quando não ocorreu overflow**
  conteúdo = FFH quando ocorreu overflow

Exercícios

5. **Subrotinas: Faça um programa que conte o número de bits com o valor 1 em uma palavra cujo endereço está armazenado na posição 128 de memória.**

- **A subrotina deve devolver o resultado no registrador X.**

Referências e Material de Apoio

- Leitura do capítulo 11 do livro Fundamentos de Arquitetura de Computadores (Raul Fernando Weber).

- Materiais de Apoio da UFRGS:
  http://www.inf.pucrs.br/flash/orgarq/apoio.html

- Canal do prof. Dr. Sandro Camargo:
  https://www.youtube.com/user/scamargo10/videos

Dúvidas ?