4. 6 Um processador microprogramado simples

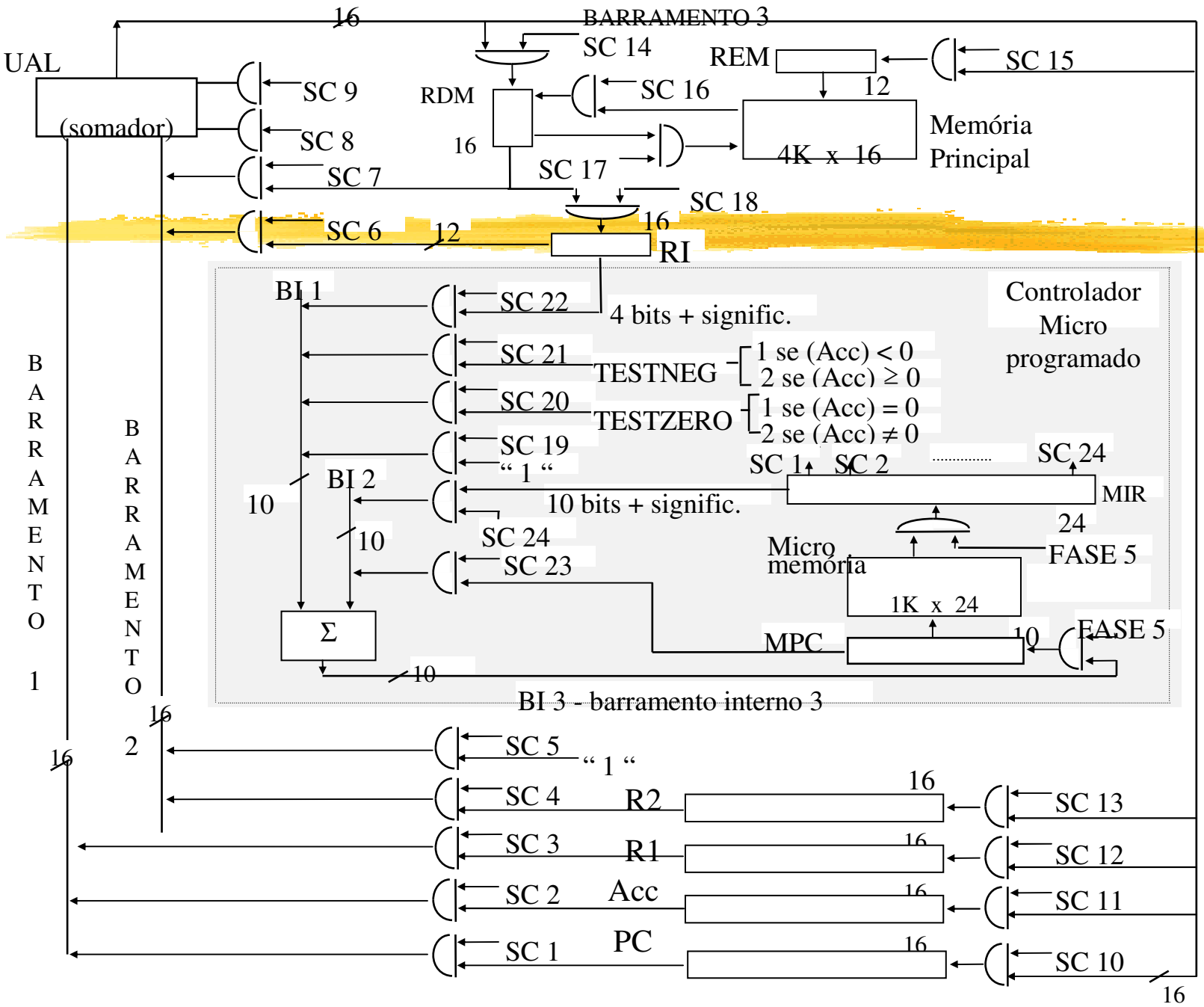
Apresentaremos aqui um processador microprogramado e sua operação com uma memória principal formando um computador simples.

Características da arquitetura:

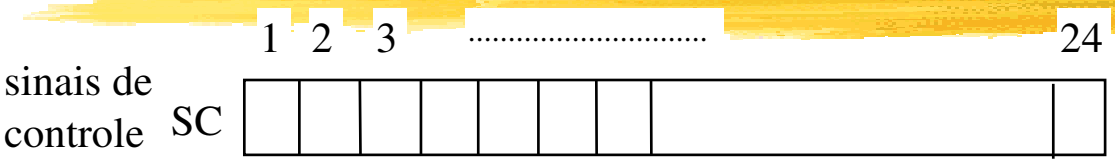
- máquina de 16 bits com 3 barramentos e 4 registradores: PC, Acc, R1 e R2, de 16 bits;
- os barramentos estão identificados como BARRAMENTO 1, BARRAMENTO 2 e BARRAMENTO 3 e, por construção, transportam tanto dados como endereços;
- memória principal de 4K palavras de 16 bits. Registradores associados: REM de 12 bits (4K) e RDM de 16 bits; registrador de instruções RI de 16 bits;
- UAL composta das seguintes funções: ADIÇÃO, SUBTRAÇÃO e SHIFT LEFT;
- quando não existe nenhum sinal de controle a UAL tem a função de ADIÇÃO, as funções de SUBTRAÇÃO e SHIFT LEFT são selecionadas por dois controles distintos;

Características da arquitetura: (continuação)

- BARRAMENTO DE CONTROLE (não mostrado na figura) de 24 bits, que transporta os sinais de controle SC1 a SC24;
- CONTROLADOR MICROPROGRAMADO (área hachurada na figura) composto por:
  - MICROMEMÓRIA com 1K palavras de 24 bits.
  - Registradores associados: MIR, registrador de microinstruções, de 24 bits e
  - MPC, registrador contador de microinstruções, de 10 bits;
  - barramentos internos BI1, BI2, BI3 de 10 bits e somador Σ para controle interno do microprograma;
  - MICROINSTRUÇÃO horizontal polifásica com o formato dado na transparência 251.


16
BARRAMENTO 3
UAL
(somador)
SC 14
REM
SC 15
SC 9
RDM
SC 16
12
SC 8
16
4K x 16
Memória
Principal
SC 7
SC 17
SC 18
SC 6
12
16
RI
BI 1
SC 22
4 bits + significic.
Controlador
Micro
programado
SC 21
TESTNEG
1 se (Acc) < 0
2 se (Acc) ≥ 0
SC 20
TESTZERO
1 se (Acc) = 0
2 se (Acc) ≠ 0
SC 19
" 1 "
SC 1
SC 2
SC 24
BI 2
10
10 bits + significic.
MIR
24
SC 24
10
Micro
memória
FASE 5
SC 23
1K x 24
Σ
MPC
10
FASE 5
10
BI 3 - barramento interno 3
BARRAMENTO 1
BARRAMENTO 2
16
16
SC 5
" 1 "
SC 4
R2
16
SC 13
SC 3
R1
16
SC 12
SC 2
Acc
16
SC 11
SC 1
PC
16
SC 10
16

A microinstrução deste processador é definida polifásica :



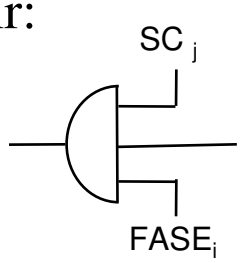
Fase do Relógio	Bits Associados
1	1 a 9
2	10 a 15
3	16, 17, 18
4	19 a 24
5	entrada do MIR e MPC

Observações :

Os sinais de controle, SC1 a SC24, habilitam transferência de informação de/para registradores, barramentos, memória principal e micromemória, além de selecionar operações na UAL.

Os sinais de controle são gerados a partir do MIR para as portas AND especificadas na figura. O $bit_i = 1$ corresponde ao sinal de controle SC_i ativo na fase apropriada do relógio. Isto implica que, junto com cada sinal de controle SC_i deve entrar, na porta AND, a fase de relógio associada. Por exemplo, a Fase 3 deve entrar nas portas com os sinais SC16, SC17 e SC18.

É importante ter presente na Figura que todas as portas, em função de diminuir a complexidade da figura, têm implícitos o sinal de Fase, como a seguir:



O bit 24 da microinstrução tem função semelhante ao bit D/I da arquitetura de Wilkes ; porém, não existe o campo de próximo endereço, se desvio. O funcionamento do desvio será explicado a seguir.

Descrição de alguns sinais:

SC 2, 7, 15: REM ← (Acc) + (RDM)
SC 2, 7, 12, 15: R1 ← (Acc) + (RDM)
REM ← (Acc) + (RDM)
SC 5: BARRAMENTO 2 ← "1"
SC 8: BARRAMENTO 3 ← (BARRAMENTO 1) - (BARRAMENTO 2)
SC 9: BARRAMENTO 3← SHIFT LEFT((BARRAMENTO 1)+(BARRAMENTO 2))
SC 16: READ (REM contém o endereço), ou seja RDM ← ((REM))
SC 17: WRITE ou (REM) ← (RDM), (RDM contém o dado e REM contém o endereço),
SC 20: BARRAMENTO INTERNO 1 (BI1) ← "1" , se (Acc) = 0
BARRAMENTO INTERNO 1 (BI1) ← "2" , se (Acc) ≠ 0
(utilizado para implementar SKIP, isto é, "pulo" da próxima microinstrução)
SC 21: BARRAMENTO INTERNO 1 (BI1) ← "1" , se (Acc) < 0
BARRAMENTO INTERNO 1 (BI1) ← "2" , se (Acc) ≥ 0
(utilizado para implementar SKIP)
SC 22: BARRAMENTO INTERNO 1 (BI1) ← os 4 bits + significativos de RI (C.O.)
SC 24: BARRAMENTO INTERNO 2 (BI2) ← os 10 bits mais significativos de MIR e inibe os outros sinais de controle.

Resumo – Fases e Microoperações

FASE 1 (bits de 1 a 9): adição/subtração/shift ou incremento do conteúdo dos registradores PC, Acc, R1, R2 e/ou RDM (transferência para BARRAMENTOS 1 e 2 e SOMADOR).

FASE 2 (bits de 10 a 15): armazenamento em PC, Acc, R1, R2, REM e/ou RDM (transferência do SOMADOR via BARRAMENTO 3).

FASE 3 (bits 16, 17 e 18): leitura/escrita de dados (RDM) da/na memória principal ou transferência de instrução para o RI.

FASE 4 (bits de 19 a 24): preparação do endereço da próxima microinstrução a ser executada.

FASE 5 (interno): armazenamento do conteúdo do BARRAMENTO INTERNO 3 no MPC e leitura do conteúdo da micromemória.

Observações:

1. Os bits 1 a 24 do MIR, quando iguais a 1, geram sinais para os respectivos controles SC1 a SC24 durante a fase apropriada do relógio; **EXCETO QUANDO O BIT 24 TÊM VALOR** 1.

2. Tendo o bit 24 valor 1, ocorre a transferência dos 10 bits mais significativos do MIR para o BARRAMENTO INTERNO 2. Esta operação permite que estes bits sejam transferidos para o MPC, através do somador Σ, gerando um JUMP no microprograma.

3. Isto acontece porque no formato da microinstrução não existe campo de endereço como mostrado no modelo de Wilkes. Resumindo, o bit 24 indica se os bits da microinstrução são interpretados como sinais de controle SCi ou como um endereço para um salto no microprograma.

Exemplo 1

Implementar no computador microprogramado: pos-mem-1 ← (Acc) **.** (R1)

Especificação da Multiplicação :

posição 1 da memória principal ← (Acc) x (R1);
o RDM será utilizado como registrador de trabalho e
o resultado da operação já estará carregado no RDM
para armazenamento na memória.

Algoritmo:

RDM ← "0" ;
repetir (Acc) vezes
    RDM ← (RDM) + (R1)
REM ← "1"; write ;

Traduzindo o algoritmo para microoperações da máquina-exemplo, temos:

```
        RDM ← "0";
TST:    (Acc) = "0" ?
                sim: jump FIM;                          while (Acc) ≠ 0
                não: RDM ← (RDM) + (R1);                RDM ← (RDM) + (R1);
                Acc ← (Acc) - 1;                        Acc ← (Acc) - 1;
                jump TST;
FIM:    REM ← "1";
        write;
```

```
        RDM ← "0";
TST:    (Acc) = "0" ?
                sim: jump FIM;
                não: RDM ← (RDM) + (R1);
                     Acc ← (Acc) - 1;
                     jump TST;
FIM:    REM ← "1";
        write;
```

Iniciaremos o microprograma na posição zero da micromemória; isto não é mandatório e sim uma simples opção.

Endereço da micromemória	Microoperações da microinstrução	SC	Comentário
0	RDM ← 0 + 0	14	RDM ← “0”
	MPC ← (MPC) + 1	19, 23	próxima microinstrução

```
        RDM ← "0";
TST:    (Acc) = "0" ?
                sim: jump FIM;
                não: RDM ← (RDM) + (R1);
                Acc ← (Acc) - 1;
                jump TST;
FIM:    REM ← "1";
        write;
```

A próxima operação do algoritmo é o teste do conteúdo do registrador Acc - realizado pelo sinal de controle TESTZERO (SC20):

se (Acc) = 0 então BI 1 ← 1

se (Acc) ≠ 0 então BI 1 ← 2

1	MPC ← (MPC) + TESTZERO	20, 23	loop : testa se (Acc)="0"
2	MPC ← 0 + ?	24, ?	Acc=0, desvia para o fim do programa
3	RDM ← (RDM) + (R1)	3, 7, 14	Acc≠0, inicia o tratamento
	MPC ← (MPC) + 1	19, 23	

```
        RDM ← "0";
TST:    (Acc) = "0" ?
                sim: jump FIM;
                não: RDM ← (RDM) + (R1);
                     Acc ← (Acc) - 1;
                     jump TST;
FIM:    REM ← "1";
        write;
```

A próxima microoperação é : Acc ← (Acc) - 1

4	Acc ← (Acc) - 1	2, 5, 8, 11	decrementa contador
	MPC ← 0 + 1	19	retorna ao loop

```
        RDM ← "0";
TST:    (Acc) = "0" ?
            sim: jump FIM;
            não: RDM ← (RDM) + (R1);
            Acc ← (Acc) - 1;
            jump TST;
FIM:    REM ← "1";
        write;
```

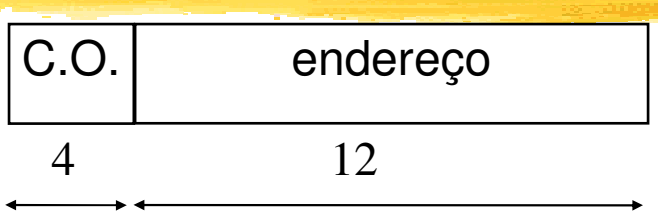
End. micromemo	Microoperações da Microinstrução	SC	Comentários
0	RDM ← 0 + 0	14	RDM ← “0”
	MPC ← (MPC) + 1	19, 23	próxima microinstr.
1	MPC ← (MPC) + TESTZERO	20, 23	loop : testa se Acc = “0”
2	MPC ← 0 + 5	8, 10, 24	Acc=0, desvia para 5
3	RDM ← (RDM) + (R1)	3, 7, 14	Acc≠ 0, soma R1
	MPC ← (MPC) + 1	19, 23	próxima microinstr.
4	Acc ← (Acc) - 1	2, 5, 8, 11	decrementa contador
	MPC ← 0 + 1	19	retorna ao loop
5	REM ← 0 + 1	5, 15	escreve na posição
	write	17	1 de memória
	MPC ← (MPC) + 1	19, 23	próxima microinstr.

>> end. final é 5

O conteúdo da micromemória será:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0														1					1				1	
1																				1			1	
2								1		1														1
3			1				1							1					1				1	
4		1			1			1			1								1					
5					1										1		1		1				1	

Exemplo 2: Linguagem de máquina para o processador microprogramado



C.O.	Instrução	Significado
0001	LOAD	Acc ← (m)
0010	STORE	m ← (Acc)
0011	ADD	Acc ← (Acc) + (m)
0100	SUBTRACT	Acc ← (Acc) - (m)
0101	MULTIPLY	Acc ← (Acc) * (m)
0110	DIVIDE	Acc ← (Acc) / (m)
0111	JUMP	desvio para m
1000	JUMPZERO	se (Acc) = 0, desvio para m
1001	JUMPNEG	se (Acc) < 0, desvio para m

Microprograma interpretador de comandos

	Endereço na Micromemória	Microoperações da Microinstrução	SC	Comentários
B	0	REM ← (PC) + 0	1, 15	end. da próxima instr.
U		RDM ← (m)	16	instrução no RDM
S		MPC ← (MPC) + 1	19, 23	incrementa MPC
C	1	PC ← (PC) + 1	1, 5, 10	incrementa PC
		RI ← (RDM)	18	instrução no RI
A		MPC ← (MPC) + (RI.co)	22, 23	decodifica instrução
M	2	MPC ← 0 + 11	7, 9, 10, 24	jump LOAD
A	3	MPC ← 0 + 13	7, 8, 10, 24	jump STORE
P	4	MPC ← 0 + 15	7, 8, 9, 10, 24	jump ADD
E	5	MPC ← 0 + 17	6, 10, 24	jump SUBTRACT
A	6	MPC ← 0 + 19	6, 9, 10, 24	jump MULTIPLY
M	7	MPC ← 0 + 36	5, 8, 24	jump DIVIDE
E	8	MPC ← 0 + 53	5, 6, 8, 10, 24	jump JUMP
N	9	MPC ← 0 + 54	5, 6, 8, 9, 24	jump JUMPZERO
TO	10	MPC ← 0 + 57	5, 6, 7, 10, 24	jump JUMPNEG

Posições 0 e 1: Estas posições, por decisão de implementação contém as microoperações/SCs que implementam a busca da instrução a ser executada.

posição 0: leitura da memória principal (busca da instrução atual)
posição 1: incremento do PC (aponta para a próxima instrução)
armazena instrução atual no registrador de instrução
inicia "decodificação" do CO.

Posições 2 a 10: Estas posições contém o mapeamento para a área da micromemória que contém o microprograma desejado (ou seja, definido pelo CO).

Aqui a decodificação é substituída por um mapeamento, usando o fato de que com o bit 24 podemos realizar um desvio de execução na micromemória.

Microprograma interpretador de comandos – cont.

	Endereço na Micromemória	Microoperações da Microinstrução	SC	Comentários
LOAD	11	REM ← 0 + (RI.end) RDM ← (m) MPC ← (MPC) + 1	6, 15 16 19, 23	REM ← endereço dado no RDM incrementa MPC
	12	Acc ← 0 + (RDM) MPC ← 0 + 0	7, 11	dado no Acc retorno à busca
STORE	13	REM ← 0 + (RI.end) MPC ← (MPC) + 1	6, 15 19, 23	REM ← endereço incrementa MPC
	14	RDM ← (Acc) + 0 (REM) ← (RDM) MPC ← 0 + 0	2, 14 17	dado no RDM dado na memória retorno à busca
ADD	15	REM ← 0 + (RI.end) RDM ← (m) MPC ← (MPC) + 1	6,15 16 19,23	REM ← endereço dado no RDM incrementa MPC
	16	Acc ← (Acc) + (RDM) MPC ← 0 + 0	2,7,11	Acc ← (Acc) + (end) retorno à busca

Microprograma interpretador de comandos – cont.

	Endereço na Micromemória	Microoperações da Microinstrução	SC	Comentários
SUBTR	17	REM ← 0 + (RI.end) RDM ← (m) MPC ← (MPC) + 1	6,15 16 19,23	REM ← endereço dado no RDM incrementa MPC
	18	Acc ← (Acc) - (RDM) MPC ← 0 + 0	2,7,8,11	Acc ← (Acc) - (end) retorno à busca
	:	:	:	:
JUMP	53	PC ← 0 + (RI.end) MPC ← 0 + 0	6,10	PC ← endereço retorno à busca
JZ	54	MPC ← (MPC) + TESTZERO	20,23	(Acc) = 0 ?
	55	PC ← 0 + (RI.end) MPC ← 0 + 0	6,10	sim, PC ← endereço retorno à busca
	56	MPC ← 0 + 0		não, retorno à busca
JN	57	MPC ← (MPC) + TESTNEG	21,23	(Acc) < 0 ?
	58	PC ← 0 + (RI.end) MPC ← 0 + 0	6,10	sim, PC ← endereço retorno à busca
	59	MPC ← 0 + 0		não, retorno à busca

Posições 11 a 59: Estas posições contêm os microprogramas responsáveis pela execução de cada tipo de instrução. No caso temos microprogramas para a execução de cada uma das 9 instruções existentes no processador.

Exemplo: posições 11 e 12 – alcançadas através do mapeamento de CO=0001, ou seja a instrução **LOAD**.

posição 11: transfere o endereço do operando para REM
faz a leitura da posição de memória indicada por REM para RDM
transfere execução para a posição 12 da micromemória
posição 12: transfere o operando, contido em RDM, para o Acc
transfere a execução para a posição 0 (inicia nova busca)

Microprograma do interpretador

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	1														1	1			1				1	
1	1				1					1								1				1	1	
2							1		1	1														1
3							1	1		1														1
4							1	1	1	1														1
5						1				1														1
6						1			1	1														1
7					1			1																1
8					1	1		1		1														1
9					1	1		1	1															1
10					1	1	1			1														1
11						1									1	1			1				1	
12							1				1													
13						1									1				1				1	
14		1												1			1							
15						1									1	1			1				1	
16		1					1				1													
17						1									1	1			1				1	
18		1					1	1			1													

Microprograma do interpretador – cont.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
18		1					1	1			1													
53						1				1														
54																				1			1	
55						1				1														
56																								
57																					1		1	
58						1				1														
59																								
:																								

Microprogramação
Utilização

Vantagens

Projeto Sistemático da Unidade de Controle

Alterabilidade da Arquitetura

Flexibilidade

Adaptabilidade

Compatibilidade com o software

Economia

Diagnosticabilidade

Facilidade de experimentação

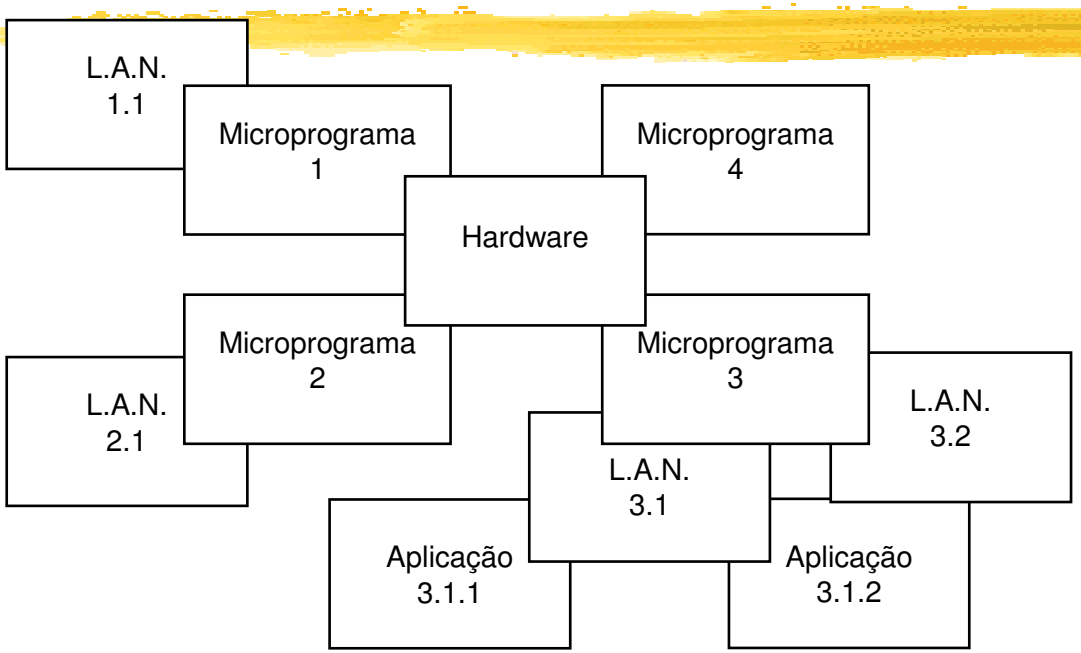
Desvantagens

Antieconômico para sistemas simples

Possível perda de velocidade

Custo do software de suporte

Microprogramação - versatilidade



Quatro máquinas diferentes são implementadas a partir do mesmo hardware, cada uma com o seu repertório de instruções de máquina a partir do qual são desenvolvidos os tradutores de linguagens de alto nível e as aplicações escritas nestas linguagens.

4.7 Exercícios sobre o processador microprogramado simples

1. Resolva os exercícios (18) e (19) do livro texto.

4.8 Exercícios com o simulador do computador microprogramado simples

1) a) Escreva um microprograma que execute a seguinte instrução:

   MULTIPLY end Acc ← (Acc)*(end)

   onde os números a serem multiplicados são inteiros e positivos.

   b) Escreva um programa que utilize esta instrução e verifique se está correta.

2) a) Escreva um microprograma que execute a seguinte instrução:

   DIVIDE end Acc ← (Acc)/(end)

   onde os números a serem divididos são inteiros e positivos.

   b) Escreva um programa que utilize esta instrução e verifique se está correta.

3) No simulador, utilizando a instrução JUMP é possível escrever um programa que comece em uma posição diferente da posição 0 da memória RAM. Utilizando esta instrução escreva e execute um programa que comece na posição 5 da memória RAM.

4) Simule o seguinte programa:

```
LOAD    3000
ADD     3001
STORE   3002
STOP
```

a. Nas posições de memória utilizadas pelo programa grave o conteúdo que achar conveniente.

b. Quantas microinstruções são executadas?

c. Quais microoperações são executadas e quais sinais de controle são gerados?

5) Simule o programa abaixo:

RAM	conteúdo	
0	LOAD	3000h
1	JUMPZERO	4
2		
3		
4	STOP	
...		
3000	0	

a. Quantas microinstruções são executadas?

b. Quais microoperações são executadas e quais sinais de controle são gerados?

c. Simule e descreva o que acontece se o conteúdo da posição 3000 for diferente de zero.

6) Simule o programa abaixo:

RAM	conteúdo	
0	LOAD	3000h
1	SUBTRACT	3001h
2	JUMPZERO	4
3	STOP	
4	STORE	3002h
5	STOP	
...		
3000	-4	
3001	-3	

a. Quantas microinstruções são executadas?

b. Quais microoperações são executadas e quais sinais de controle são gerados?

c. Simule e descreva o que acontece com diversos conteúdos das posições 3000 e 3001.

7) Simule o programa abaixo:

RAM	conteúdo	
0	0	
1	LOAD	3000h
2	SUBTRACT	3001h
3	JUMPZERO	5
4	STOP	
5	STORE	3002h
6	STOP	
...		
3000	-4	
3001	-3	

a. O programa acima funciona no simulador ?

b. Se não funcionar corrija-o e simule.

8) Simule o programa abaixo:

RAM	conteúdo	
0	LOAD	3000h
1	ADD	3001
2	JUMPNEG	6
3	SUBTRACT	3002h
4	STORE	3003h
5	STOP	
6	ADD	3004h
7	STORE	3005h
8	STOP	
...		
3000	a	
3001	b	
3002	c	
3003	d	

a. Atribua diferentes valores às constantes a, b, c, d de modo a que a instrução JUMPNEG gere ou não desvio. Simule todos os casos.

b. Para um dos casos responda: quantas instruções são executadas e mostre as microoperações e sinais de controle.