## CONTEÚDO DO CAPÍTULO 10 –(SAP-1)

**SAP-1**

O computador SAP ( Simples – Quanto - Possível = Simple-As-Possible) foi projeta para você, o principiante. A principal finalidade do SAP consiste em introduzir todas as idéias cruciais além da operação do computador sem sobrecarregá-lo com detalhes desnecessários.Mas até mesmo um simples computador como o SAP engloba muitos conceitos avançados. Para evitar bombardear você com excesso de informações todas de uma vez, examinaremos três diferentes gerações do computador SAP.

SAP-1 é o primeiro estágio na evolução com vistas aos modernos computadores .Embora primitivo, o SAP-1 é um grande passo para um principiante.Assim, neste capítulo, entregue-se ao estudo com determinação; procure adquirir um perfeito conhecimento ou prática do SAP-1, de sua arquitetura, de sua programação e de seus circuitos. Depois você estará prepara o para estudar o SAP-2.

**10.1 ARQUITETURA**

A Fig.10-1 mostra a arquitetura ( estrutura ) do SAP-1, um computador organizado em barramentos . Todas as saídas dos registradores para o barramento W são de três estados; isto possibilita a transferência de dados ordenadamente. Todas as outras saídas dos registradores são de dois estados; estas saídas comandam continuamente as caixas às quais elas estão conectadas.

O diagrama da Fig.10-1 enfatiza os registradores usados no SAP-1. Por esta razão, nenhuma tentativa tem sido feita para guardar todos os circuitos de controle em um bloco chamado unidade de controle, todos os circuitos de entrada-saída em um outro bloco chamado unidade de E/S (I/O) etc.

Muitos dos registradores da Fig.10-1 já são conhecidos desde os primeiros exemplos e discussões. O que segue é uma breve descrição de cada caixa; explicações detalhadas vêm mais tarde.

**Contador de Programa**

O programa é armazenado no começo da memória com a primeira instrução no endereço binário 0000, a segunda instrução no endereço 0001, a terceira no endereço 0010 etc. O contador de programa, que é parte da unidade de controle, conta de 0000 a 1111. Sua tarefa é enviar à memória o endereço da instrução seguinte a ser buscada e executada. Ele faz isto como segue.

O contador de programa é restabelecido (reset) a 0000 antes de cada processamento no computador. Quando começa o processamento ou execução no computador, o contador de programa envia o endereço 0000 à memória. O contador de programa é então incrementado para se obter 0001. Depois da primeira instrução ser buscada e executada, o contador de programa envia o endereço 0001 à memória. Novamente o contador de programa é incrementado. Depois da segunda instrução ser buscada e executada, o contador de programa envia o endereço 0010 à memória. Desta maneira, o contador de programa está acompanhando o desenvolvimento da próxima instrução a ser buscada e executada.

O contador de programa é como alguém que aponta um dedo em uma lista de instruções,dizendo para fazer isto em primeiro lugar, fazer isto em segundo, fazer isto em terceiro etc. É por isto que o contador de programa às vezes é chamado ponteiro (pointer); ele aponta ou indica um endereço na memória onde algo importante está sendo armazenado.

**Entrada e REM**

Abaixo do contador de programa está o bloco ***entrada e REM***.Ele inclui os registradores de chaves de dados e de endereço discutidos na secção 9-4. Estes registradores de chaves, que são parte da unidade de entrada, permitem-nos enviar 4 bits de endereço e 8 bits de dados à RAM. Conforme lembramos, as palavras de dados e de instrução são escritas na RAM antes de um processamento no computador.

O registrador de endereços na memória (REM) é parte da memória do SAP-1. Durante um processamento no computador, o endereço no contador de programa é retido no REM. Um bit mais tarde, o REM aplica este endereço de 4 bits à RAM, onde uma operação de leitura é realizada.

**A RAM**

A RAM é uma RAM TTL estática de 16 x 8. Conforme discutido na secção 9-4, podemos programar a RAM por meio dos registradores de chaves de dados e de endereços. Isto nos permite armazenar um programa e os dados na memória antes de um processamento do computador.

Durante um processamento do computador, a RAM recebe endereços de 4 bits do REM e é executada uma operação de leitura. Desta maneira, a instrução ou palavra de dados armazenada na RAM é colocada no barramento W para uso em alguma outra parte do computador.

**Registrador de Instruções**

O registrador de instruções constitui parte da unidade de controle. Para buscar uma instrução da memória o computador realiza uma operação de leitura da memória. Isto coloca o conteúdo do local de memória endereçado no barramento W. Ao mesmo tempo, o registrador de instruções é preparado para carregamento na próxima transição positiva de relógio.

O conteúdo do registrador de instruções é dividido em dois nibbles (meios-bytes). O nibble superior é uma saída de dois estados que vai diretamente ao bloco rotulado "controlador-sequencializador". O nibble inferior é uma saída de três estados que é lida no barramento W quando necessário.

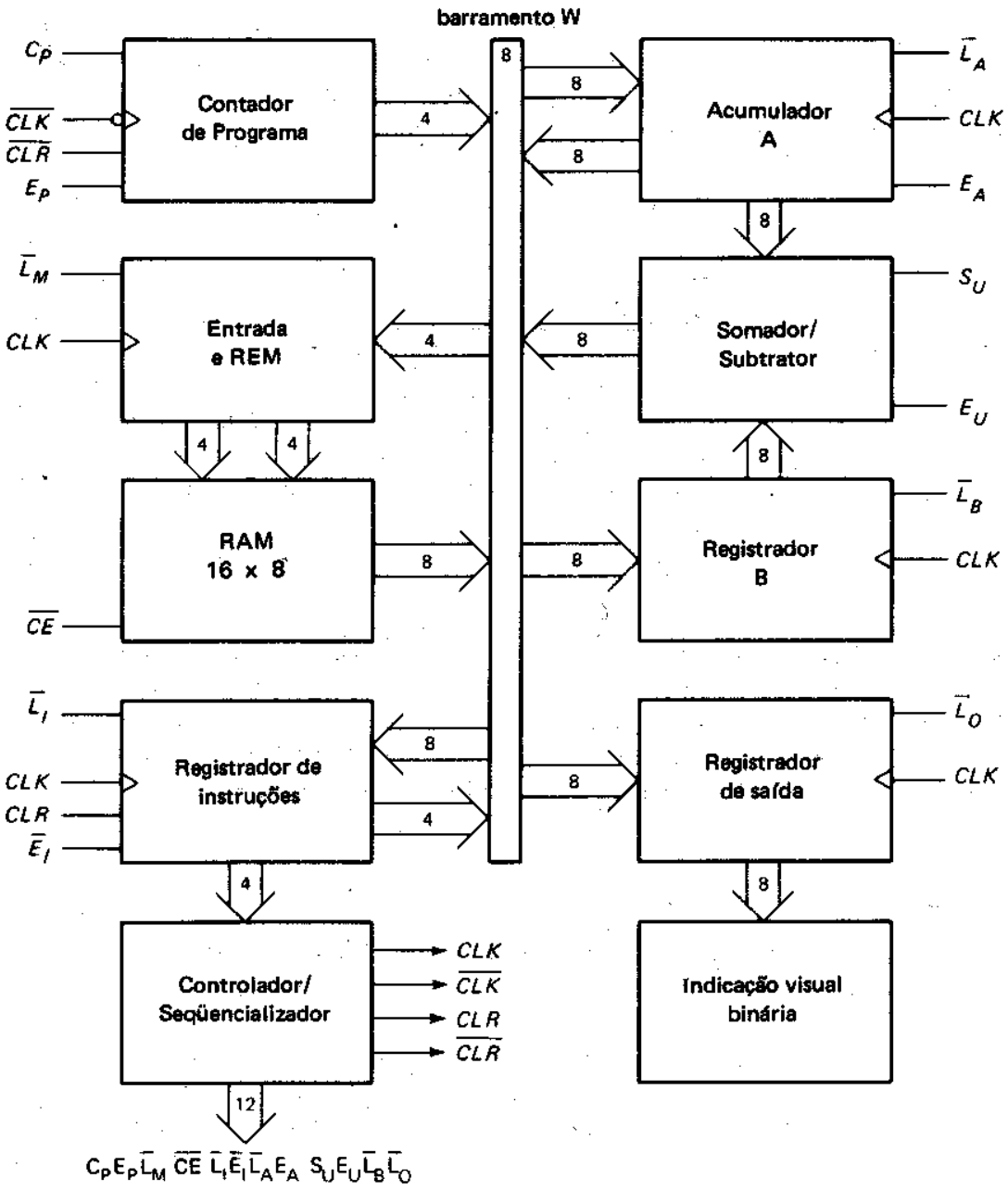



**Fig10-1 Arquitetura do SAP-1**

### Controlador – Sequencializador

O bloco inferior esquerdo contém o controlador - sequencializador. Antes de cada processamento no computador, um sinal $\overline{CLR}$ é enviado ao contador de programa e um sinal CLR é enviado ao registrador de instruções. Isto restabelece o contador de programa em 0000 e elimina a última instrução no registrador de instruções.
Um sinal CLK de relógio é enviado a todos os registradores de memória intermediária; isto sincroniza a operação do computador, assegurando que as coisas acontecem quando elas são passíveis de

acontecer. Em outras palavras, todas as transferências do registrador ocorrem na transição positiva de um sinal CLK comum. Observemos que um sinal $\overline{CLR}$ também vai ao contador de programa.

Os 12 bits que vêm para fora do controlador- sequencializador formam uma palavra que controla o resto do computador (como um supervisor que diz aos outros o que fazer). Os 12 fios que transportam a palavra de controle são chamados ***barramento de controle***.

A palavra de controle tem o formato de

$$\mathbf{CON=}\ C_P E_P \overline{L}_M \overline{CE}\ \overline{L}_I \overline{E}_I \overline{L}_A E_A\ S_U E_U \overline{L}_B \overline{L}_O$$

Esta palavra determina como os registradores reagirão à próxima transição positiva de relógio (CLK). Por exemplo, um $E_p$ alto e um $\overline{L}_M$ baixo significam que o conteúdo do contador de programa é retido no REM na próxima transição positiva de relógio. Como outro exemplo, um baixo $\overline{CE}$ e um baixo $\overline{L}_A$ significam que a palavra da RAM endereçada será transferida para o acumulador na próxima transição positiva de relógio. Mais tarde, examinaremos os diagramas de temporização para verificar exatamente quando e como ocorrem estas transferências de dados.

**Acumulador**

O acumulador (A) é um registrador de memória intermediária que armazena respostas intermediárias durante um processamento no computador. Na Fig. 10-1 o acumulador tem duas saídas. A saída de dois estados vai diretamente ao somador-subtrator. A saída de três estados vai ao barramento W. Portanto, a palavra do acumulador de 8 bits continuamente comanda o somador-subtrator; a mesma palavra aparece no barramento W quando $E_A$ está alto.

**O Somador-subtrator**

SAP-1 usa um somador-subtrator de complemento de 2. Quando $S_U$ for baixo na Fig. 10.1, a soma fora do somador-subtrator será

**S=A+B**

Quando $S_U$ for alto, aparecerá a diferença:

**S = A + B'**

O somador-subtrator é assincrono (não-sincronizado); isto significa que seu conteúdo pode variar logo que as palavras de entrada variem. Quando $E_U$ for alto, estes conteúdos aparecerão no barramento W.

**Registrador B**

O registrador B é um outro registrador de memória intermediária. Ele é usado em operações aritméticas. Um baixo $L_B$ e uma transição positiva de relógio carregam a palavra do barramento W dentro do registrador B. A saída de dois estados do registrador B comanda o somador-subtrator, fornecendo o número a ser adicionado ou subtraído do conteúdo do acumulador.

**Registrador de Saída**

O Exemplo 8-1 tratou do registrador de saída. No final de um processamento do computador, o acumulador contém a resposta ao problema que está sendo resolvido. Neste ponto, necessitamos transferir a resposta para o mundo exterior. Isto é onde é usado o registrador de saída. Quando $E_A$

for alto e $\overline{L}_0$ for baixo, a próxima transição positiva de relógio carregará a palavra do acumulador no registrador de saída.

O registrador de saída muitas vezes é chamado porta de sa(da porque os dados processados podem sair do computador através deste registrador. Em microcomputadores as portas de saída são conectadas aos ***circuitos de interface*** que comandam dispositivos periféricos como as impressoras, os tubos de raios catódicos, as teleimpressoras etc. (Um circuito de interface prepara os dados para comandar cada dispositivo.)

**Indicador Visual em Binário**

O ***indicador visual em binário*** é uma fileira de oito diodos emissores de luz (LEDs). Em virtude de cada LED conectar-se a um biestável da porta de saída, o indicador visual em binário mostra-nos o conteúdo da porta de saída. Portanto, depois de termos transferido uma resposta do acumulador para a porta de saída, podemos ver a resposta em forma binária.

**Resumo**

A unidade de controle do SAP-1 consiste no contador de programa, no registrador de instruções e no controlador-sequencializador que produz a palavra de controle, os sinais de limpar(ou restabelecer) e os sinais de relógio. A ULA do SAP-1 consiste em um acumulador, em um somador-subtrator e em um registrador B. A memória do SAP-1 tem um REM e uma RAM de 16x8. A unidade de E/S inclui as chaves de programação de entrada, a porta de saída e o indicador visual em binário.

**10-2 CONJUNTO DE INSTRUÇÕES**

Um computador é um amontoado inútil de ***hardware*** até que alguém o programe. Isto significa o carregamento de instruções passo-a-passo na memória antes do início de um processamento no computador. Antes que possamos programar um computador, no entanto, devemos aprender seu conjunto de instruções, as operações básicas que ele pode executar .Segue-se o conjunto de instruções do SAP-1.

**LDA**

Conforme descrito no Capítulo 9, as palavras na memória podem ser simbolizadas por $\mathbf{R_0}$, $\mathbf{R_1}$, $\mathbf{R_2}$ etc. Isto significa que $R_0$ é armazenada no endereço **OH**, $\mathbf{R_1}$ no endereço **1H**, $\mathbf{R_2}$ no endereço 2H etc.

**LDA** significa "carregar o acumulador" (Load the Accumulator). Uma instrução **LDA** completa inclui o endereço hexadecimal dos dados a serem carregados. **LDA 8H**, por exemplo,significa "carregar o acumulador com o conteúdo do local **8H** da memória". Portanto, dado

$$\mathbf{R_8=11110000}$$

a execução de **LDA 8H** resulta em

**A= 1111 0000**

Similarmente, **LDA AH** significa "carregar o acumulador com o conteúdo do local **AH** da memória", **LDA FH** significa "carregar o acumulador com o conteúdo do local **FH** da memória " etc.

**ADD**

**ADD** é uma outra instrução do SAP-1. Uma instrução ADD completa inclui o endereço da palavra a ser acrescentada. Por exemplo, **ADD 9H** significa "acrescentar o conteúdo do local **9H** " da memória ao conteúdo do acumulador"; a soma substitui o conteúdo original do acumulador.

Eis aqui um exemplo. Suponhamos que o decimal 2 esteja no acumulador e que o decimal , no local **9H** da memória. Então

**A= 00000010**
**$R_9$ = 00000011**

Durante a execução de **ADD 9H**, acontece o seguinte. Primeiro, $R_9$ é carregada no registrador B para a se obter

**B= 00000011**

e quase instantaneamente o somador-subtrator forma a soma de **A** e **B**.

**SUM = 00000101**

Segundo, esta soma é carregada no acumulador para se obter

**A= 00000101**

A rotina precedente é usada em todas as instruções **ADD**; a palavra RAM endereçada vai ao registrador B e a saída do somador-subtrator vai para o acumulador. Isto ocorre porque a execução de **ADD 9H** acrescenta **$R_9$** ao conteúdo do acumulador, a execução de **ADD FH** acrescenta $R_F$ ao conteúdo do acumulador etc.

**SUB**

SUB é uma outra instrução do SAP-1. Uma instrução **SUB** completa inclui o endereço da palavra a ser subtraída. Por exemplo, **SUB CH** significa "subtrair o conteúdo do local **CH** da memória do conteúdo do acumulador"; a diferença para fora do somador-subtrator depois substituir o conteúdo original do acumulador.

Como um exemplo concreto, admitamos que o decimal 7 esteja no acumulador e que o decimal 3, no local **CH** da memória. Então

**A =00000111**
**$R_c$ = 00000011**

A execução de **SUB CH** ocorre como segue. Primeiro, **$R_c$** é carregada no registrador B para se obter

**B = 00000011**

e quase que instantaneamente o somador-subtrator forma a diferença de **A** e **B**:

**DIFF** = 0000 0100

Segundo, esta diferença é carregada dentro do acumulador e

**A= 00000100**

A rotina precedente aplica-se a todas as instruções **SUB**; a palavra RAM endereçada vai ao registrador **B** e a saída do somador-subtrator vai ao acumulador. Isto ocorre porque a execução de **SUB CH** subtrai **$R_c$** do conteúdo do acumulador, a execução de **SUB EH** subtrai **$R_E$** do acumulador etc.

**OUT**

A instrução **OUT** diz ao computador SAP-1 para transferir o conteúdo do acumulador para a porta de saída. Depois de **OUT** ter sido executada, podemos ver a resposta ao problema que está sendo resolvido.

**OUT** é completa por si própria; isto é, não temos que incluir um endereço quando usamos **OUT** porque a instrução não envolve dados na memória.

**HLT**

**HLT** significa parar (halt). Esta instrução diz ao computador para parar o processamento de dados. **HLT** assinala o fim de um programa, similar à maneira com que um ponto assinala o fim de uma frase ou sentença. Devemos usar uma instrução **HLT** no fim de cada programa do SAP-1; do contrário, obtemos refugos (trash) no computador (respostas sem significado causadas pelo processamento descontrolado).

**HLT** é completa por si própria; não temos que incluir uma palavra de RAM ao usarmos HLT porque esta instrução não envolve a memória.

**Instruções de Referência à Memória**

**LDA**, **ADD** e **SUB** são chamadas instruções de referência à memória porque elas usam dados armazenados na memória. **OUT** e **HLT**, por outro lado, não são instruções de referência à memória porque elas não envolvem dados armazenados na memória.

**Mnemônicos**

**LDA**, **ADD**, **SUB**, **OUT** e **HLT** são o conjunto de instruções do SAP-1. Instruções abreviadas como estas são chamadas ***mnemônicos*** (auxílios à memória). Mnemônicos são conhecidos no trabalho dos computadores porque eles nos lembram da operação que ocorrerá quando a instrução for executada. A Tabela 10-1 resume o conjunto de instruções do SAP-1.

**Os Microprocessadores 8080 e 8085**

O primeiro microprocessador amplamente usado foi o 8080. Ele tem 72 instruções. O 8085 é uma versão ampliada do 8080 tendo essencialmente o mesmo conjunto de instruções.Para tornar o SAP prático, as instruções do SAP terão compatibilidade ascendente com o conjunto de instruções do 8080/8085. Em outras palavras, as instruções **LDA**, **ADD**, **SUB**, **OUT** e **HLT** são instruções do 8080/8085. Analogamente, as instruções do SAP-2 e do SAP-3 serão parte do conjunto de instruções do 8080/8085. Conhecendo as instruções do SAP estaremos preparados para o 8080 e para o 8085, dois microprocessadores amplamente usados. Uma vez que aprendemos o conjunto de instruções do 8080/8085, podemos nos desviar para outros microprocessadores.

**TABELA 10-1. CONJUNTO DE INSTRUÇÕES DO SAP-1**

| Mnemônicos | Operação |
|---|---|
| **LDA** | Carregue os dados da RAM no acumulador. |
| **ADD** | Some os dados da RAM com o acumulador. |
| **SUB** | Subtraia os dados da RAM do acumulador . |
| **OUT** | Carregue os dados do acumulador no registrador de saída. |
| **HLT** | Pare o processamento. |

EXEMPLO 10-1

Eis aqui um programa do SAP-1 em forma de mnemônico:
Endereço Mnemônicos

| Endereço | Mnemônicos |
|---|---|
| 0H | LDA 9H |
| 1H | ADD AH |
| 2H | ADD BH |
| 3H | SUB CH |
| 4H | OUT |
| 5H | HLT |

Os dados na memória superior são

| Endereço | Dados |
|---|---|
| 6H | FFH |
| 7H | FFH |
| 8H | FFH |
| 9H | 01H |
| AH | 02H |
| BH | 03H |
| CH | 04H |
| DH | FFH |
| EH | FFH |
| FH | FFH |

O que faz cada instrução?

**SOLUÇÃO**

O programa está na memória inferior, localizado nos endereços OH a 5H. A primeira instrução carrega o acumulador com o conteúdo do local 9H da memória, e assim o conteúdo do acumulador toma-se

**A=** 01H

A segunda instrução soma o conteúdo do local **AH** da memória ao conteúdo do acumulador para produzir um novo total do acumulador de

**A** = 01H + 02H = 03H

Similarmente, a terceira instrução soma o conteúdo do local **BH** da memória

**A=** 03H + 03H = 06H

A instrução **SUB** subtrai o conteúdo do local **CH** da memória para produzir

**A=** 06H -04H = 02H

A instrução **OUT** carrega o conteúdo do acumulador para a porta de saída; portanto, o indicador visual em binário mostra

0000 0010

A instrução **HLT** interrompe o processamento dos dados.

### 10.3 PROGRAMAÇÃO DO SAP-1

Para carregar palavras de dados e instruções na memória do SAP-1 temos que usar alguma espécie de código que o computador possa interpretar. A Tabela 10-2 mostra o código usado no SAP-1. O número 0000 significa **LDA**, 0001 significa **ADD**, 0010 significa **SUB**, 1110 significa **OUT** e 1111 significa **HLT**.Em virtude deste código dizer ao computador que operação executar, ele é chamado código de operação (op code).

Conforme tratado inicialmente-as chaves de dados e de endereços da Fig. 9-7 nos possibilitam programar a memória do SAP-1. Pelo projeto, estas chaves produzem um 1 na posição para cima **(U)** e um 0 na posição para baixo **(D)**. Quando programando as chaves de dados com uma instrução, o código de operações entra no nibble superior, e o operando (o resto da instrução) entra no nibble inferior.

**TABELA 10-2. CÓDIGO OP DO SAP-1**

| Mnemônicos | Código op |
|---|---|
| LDA | 0000 |
| ADD | 0001 |
| SUB | 0010 |
| OUT | 1110 |
| HALT | 1111 |

Por exemplo, suponhamos que queremos armazenar as seguintes instruções:

| Endereço | Instrução |
|---|---|
| 0H | LDA FH |
| 1H | ADD EH |
| 2H | HLT |

Primeiro, convertemos cada instrução em binário como segue:

LDA FH = 0000 1111
ADD EH = 00011110
H L T = 1111 XXXX

Na primeira instrução. 0000 é o código op de **LDA** e 1111 é o equivalente binário de **FH**. Na segunda instrução, 0001 é o código op de **ADD** e 1110 é o equivalente binário de **EH**. Na terceira instrução, 1111 é o código op de **HLT** e XXXX são saídas irrelevantes porque a HLT não é uma instrução de referência da memória.

A seguir, montemos as chaves de dados e de endereços como segue:

| Endereços | Dados |
|---|---|
| DDDD | DDDDUUUU |
| DDDU | DDDUUUUD |
| DDUD | UUUUXXXX |

Depois de cada palavra de dados e de endereço ser ajustada, comprimimos o botão de escrita.Uma vez que **D** armazena um binário 0 e **U** armazena um binário 1, os três primeiros locais da memória têm agora este conteúdo:

| Endereços | Conteúdo |
|---|---|
| 0000 | 0000 1111 |
| 0001 | 00011110 |
| 00 10 | 1111 XXXX |

Um detalhe final. A ***linguagem de montagem*** (Assembly language) envolve o trabalho com mnemônicos quando se escreve um programa. A ***linguagem de máquina*** envolve o trabalho com cordões de 0s e 1s. Os seguintes exemplos assinalam a distinção entre as duas linguagens.

EXEMPLO 1 0-2

Traduzir o programa do Exemplo 10-1 em linguagem de máquina do SAP-1.

**SOLUÇÃO**

Eis o programa do Exemplo 10-1:

| Endereço | Instrução |
|---|---|
| 0H | LDA 9H |
| 1H | ADD AH |
| 2H | ADD BH |
| 3H | SUB CH |
| 4H | OUT |
| 5H | HLT |

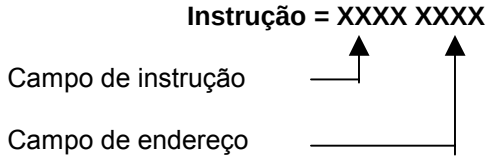
Este programa está em linguagem de montagem conforme ele se acha agora. Para obtê-lo em linguagem de máquina, nós o traduzimos em 0s e 1s conforme segue:

| Endereço | Instrução |
|---|---|
| **0000** | 00001001 |
| 0001 | 00011010 |
| 0010 | 00011011 |
| 0011 | 00101100 |
| 0100 | 1110 XXXX |
| 0101 | 1111 XXXX |

Agora o programa está em linguagem de máquina.

Qualquer programa como o precedente que esteja escrito em linguagem de máquina é chamado ***programa objeto***. O programa original com mnemônicos é chamado ***programa fonte***. Em SAP-1, o operador traduz o ***programa fonte*** em um ***programa objeto*** quando da programação das chaves de dados e de endereços.

Um detalhe final. Os quatro **MSBs** de uma instrução em linguagem de máquina SAP-1 especificam a operação, e os quatro **LSBs** dão o endereço. As vezes nós nos referimos aos **MSBs** como o campo de instrução e aos **LSBs** como o campo de endereço. Simbolicamente,




EXEMPLO 10-3

Como devemos programar SAP-1 para resolver este problema de aritmética?

16 + 29 + 24- 32

Os números estão em forma decimal.

SOLUÇÃO

Uma das maneiras consiste em usar o programa do exemplo precedente, armazenando os dados (6, 20, 4, 32) nos locais da memória **9H** a **CH**. Com o Apêndice 1, podemos converter os dados decimais em hexadecimais para obter esta versão em linguagem de montagem:

| Endereço | Conteúdo |
|---|---|
| 0H | LDA 9H |
| 1H | ADD AH |
| 2H | ADD BH |
| 3H | SUB CH |
| 4H | OUT |
| 5H | HLT |
| 6H | XX |
| 7H | XX |
| 8H | XX |
| 9H | 10H |
| AH | 14H |
| BH | 18H |
| CH | 20H |

A versão em linguagem de máquina é

| Endereço | Conteúdo |
|---|---|
| 0000 | 0000 1001 |
| 0001 | 00011010 |
| 0010 | 00011011 |
| 0011 | 0010 1100 |
| 0100 | 1110 XXXX |
| 0101 | 1111 XXXX |
| 0110 | XXXX XXXX |
| 0111 | XXXX XXXX |
| 1000 | XXXX XXXX |
| 1001 | 0001 0000 |
| 1010 | 0001 0100 |
| 1011 | 0001 1000 |
| 1100 | 0010 0000 |

Observemos que o programa é armazenado à frente dos dados. Em outras palavras, o programa está na memória inferior e os dados na memória superior. Isto é essencial em SAP-1 porque o contador de programa aponta para o endereço 0000 para a primeira instrução, 0001 para a segunda instrução e assim por diante.

EXEMPLO 10-4

Compactar o programa e os dados do exemplo precedente convertendo em abreviação hexadecimal.

SOLUÇÃO

| Endereço | Conteúdo |
|---|---|
| 0H | 09H |
| 1H | 1AH |
| 2H | 1BH |
| 3H | 2CH |
| 4H | EXH |
| 5H | FXH |
| 6H | XXH |
| 7H | XXH |
| 8H | XXH |
| 9H | 10H |
| AH | 14H |
| BH | 18H |
| CH | 20H |

Esta versão do programa e dados é ainda considerada linguagem de máquina.

A propósito, os dados negativos são carregados em forma de complemento de 2. Por exemplo, **-03H** é introduzido como **FDH**.

**10-4 CICLO DE BUSCA (FETCH)**

A ***unidade de controle*** constitui a chave para uma operação automática do computador. A ***unidade de controle*** gera as palavras de controle que buscam e executam cada instrução. Enquanto cada instrução for buscada e executada, o computador passará por diferentes ***estados de temporização*** (estados T), períodos durante os quais mudam os conteúdos dos registradores. Investiguemos mais alguma coisa a respeito destes estados T.

**Contador em Anel**

Inicialmente, analisamos o contador em anel do SAP-1 (ver Fig. 8-16 para o diagrama esquemático). A Fig. 10-2a simboliza o contador em anel, que tem uma saída

$$\mathbf{T = T_6T_5T_4T_3T_2T_1}$$

No começo de um processamento do computador, a palavra anel é

$$\mathbf{T} = 000001$$

Pulsos de relógio sucessivos produzem palavras de anel de

$$\mathbf{T} = 000010$$
$$\mathbf{T} = 000100$$
$$\mathbf{T} = 001000$$
$$\mathbf{T} = 010000$$
$$\mathbf{T} = 100000$$

Então, contador em anel se restabelece (reset) em 000001 e o ciclo se repete. Cada palavra de anel representa um estado T.

A Fig. 10-2b mostra os pulsos de temporização fora do contador em anel. O estado inicial $\mathbf{T_1}$ começa com uma transição negativa de relógio e termina com a próxima transição negativa de relógio. Durante este estado **T**, o bit $\mathbf{T_1}$ fora do contador em anel está alto.

Durante o próximo estado, $\mathbf{T_2}$ está alto; o estado seguinte tem um $\mathbf{T_3}$ alto; depois um $\mathbf{T_4}$ alto e assim por diante. Conforme podemos ver, o contador em anel produz seis estados **T**: Cada instrução é buscada e executada durante estes seis estados **T**.

Observemos que uma transição **CLK** positiva ocorre a meio caminho em cada estado **T**. A importância disto será ressaltada mais tarde.

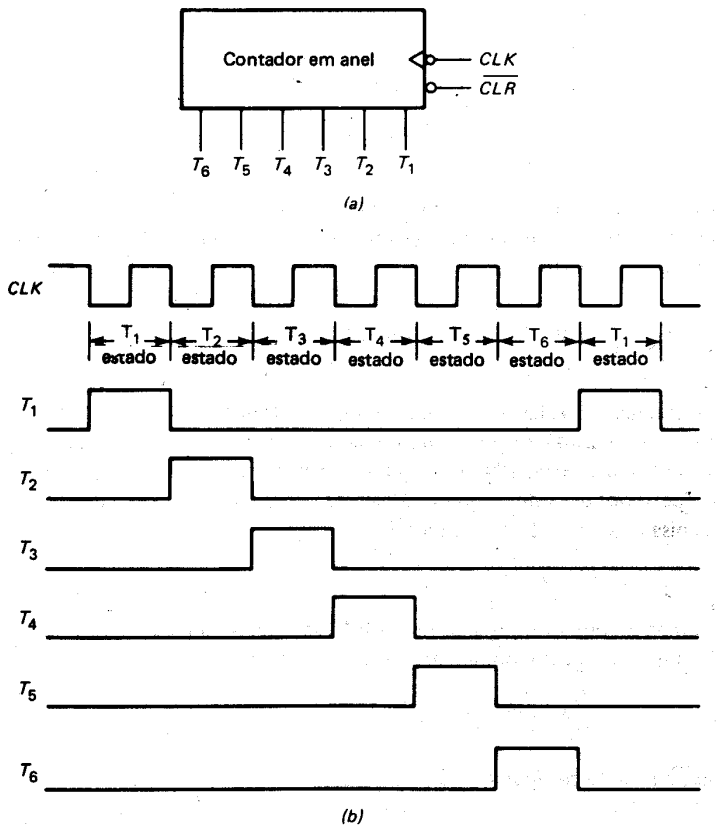



**Fig. 10-2 Contador em anel: (a)símbolo;(b) sinais de relógio e de temporização**.

### Estado de Endereço

O estado $\mathbf{T_1}$ é chamado ***estado de endereço*** porque o endereço no contador **(PC)** de programa é transferido para o registrador **(REM)** de endereços da memória durante este estado. A Fig. 10-3a mostra as secções do computador que estão ativas durante este estado (as partes ativas estão claras; as partes inativas estão escuras).

Durante O estado de endereço, $E_P$ e $\overline{L}_M$ estão ativos; todos os outros bits de controle estão inativos. Isto significa que o controlador-sequencializador está enviando para fora uma palavra de controle de

$$\begin{aligned} CON= {} & C_P E_P \overline{L}_M \overline{CE}\ \ \overline{L}_I \overline{E}_I \overline{L}_A E_A\ \ S_U E_U \overline{L}_B \overline{L}_O \\ = {} & 0\ 1\ 0\ \ 1\ \ \ 1\ 1\ 1\ 0\ \ \ 0\ 0\ 1\ 1 \end{aligned}$$

durante este estado.

### Estado de Incremento

A Fig. 10-3b mostra as partes ativas de SAP-1 durante o estado $\mathbf{T_2}$. Este estado é chamado ***estado de incremento*** porque o contador de programa é incrementado. Durante o ***estado de incremento***, o controlador-sequencializador está produzindo uma palavra de controle de

$$\mathbf{CON} = C_P E_P \overline{L}_M \overline{CE}\ \ \overline{L}_I \overline{E}_I \overline{L}_A E_A\ \ S_U E_U \overline{L}_B \overline{L}_O$$
$$= 1\ 0\ 1\ 1\ \ \ 1\ 1\ 1\ 0\ \ 0\ 0\ 1\ 1$$

Conforme vemos, o bit $C_P$ é ativo.

**Estado de Memória**

O estado **$T_3$** é chamado ***estado de memória*** porque a instrução de RAM endereçada é transferida da memória para o registrador de instrução. A Fig. 10-3c mostra as partes ativas de SAP-1 durante o ***estado de memória***. Os únicos bits de controle ativos durante este estado são $\overline{CE}$ e $\overline{L}_I$, e a palavra fora do controlador-sequencializador é

$$\mathbf{CON} = C_P E_P \overline{L}_M \overline{CE}\ \ \overline{L}_I \overline{E}_I \overline{L}_A E_A\ \ S_U E_U \overline{L}_B \overline{L}_O$$
$$= 0\ 0\ 1\ 0\ \ \ 0\ 1\ 1\ 0\ \ 0\ 0\ 1\ 1$$

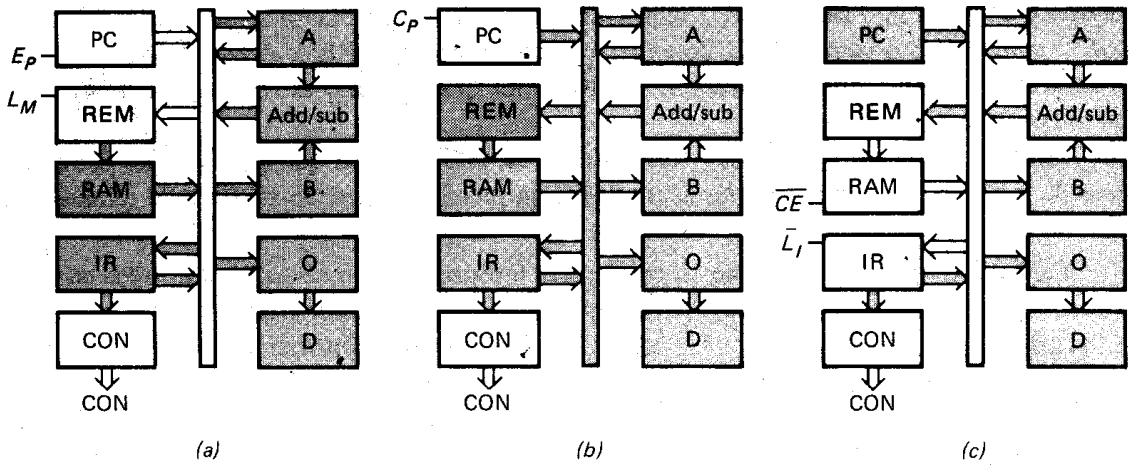



**Fig. 10-3 Ciclo de busca:(a) estado $T_1$;(b) estado $T_2$ ;(c)estado $T_3$.**

**Ciclo de Busca**

Os estados de endereço, de incremento e de memória são chamados ciclo de busca do SAP-1. Durante O estado de endereço, $E_P$ e $\overline{L}_M$ estão ativos; isto significa que o contador de programa monta o REM através do barramento W. Conforme mostrado inicialmente na Fig. 10-2b uma transição positiva de relógio ocorre a meia distância através do estado de endereço; isto carrega o REM com o conteúdo do PC.

$C_P$ é o único bit de controle ativo durante o estado de incremento. Isto prepara o contador de programa para contar transições positivas de relógio. A meia distância através do estado de incremento, uma transição positiva de relógio atinge o contador de programa e avança a contagem de 1.

Durante o estado de memória, $\overline{CE}$ e $\overline{L}_I$ estão ativos. Portanto, a palavra de RAM endereçada prepara o registrador de instruções através do barramento W. A meia distância através do estado de memória, uma transição positiva de relógio carrega o registrador de instruções com a palavra de RAM endereçada.

## 10-5 CICLO DE EXECUÇÃO

Os três estados seguintes ( $T_4$ , $T_5$ e $T_6$) constituem o ciclo de execução do SAP-1 . As transferências do registrador durante o ciclo de execução dependem da instrução particular que está sendo executada. Por exemplo, **LDA 9H** requer transferências de registrador diferentes das de **ADD BH**. O que se segue são as ***rotinas de controle*** para as diferentes instruções do SAP-1.

### Rotina LDA

Para um estudo concreto, admitamos que o registrador de instruções tenha sido carregado com **LD A 9H**:

**IR** = 0000 1001

Durante o estado $T_4$, o campo 0000 de instrução vai para o controlador-sequencializador, onde ele é decodificado; o campo 1001 de endereços é carregado no REM. A Fig. 10-4a mostra as partes ativas de SAP-1 durante o estado $T_4$. Notemos que $\overline{E}_I$ e $\overline{L}_M$ são ativos; todos os outros bits de controle são inativos.

Durante o estado $T_5$ $\overline{CE}$ e $\overline{L}_A$ tomam-se altos. Isto significa que a palavra de dados endereçada a RAM será carregada no acumulador na próxima transição positiva de relógio (ver Fig. 10-4b).

$T_6$ é um estado sem operação. Durante este terceiro estado de execução, todos os registradores estão inativos (Fig. 10-4c). Isto significa que o controlador-sequencializador está enviando para fora uma palavra cujos bits são todos inativos. **Nop** (pronunciado noop) significa "sem operação" ou “inoperante". O estado $T_6$ da rotina **LDA** é um ***nop***.

A Fig.10-5 mostra o diagrama de temporização das rotinas **LDA** e de busca. Durante o estado $T_1$, $E_P$ e $\overline{L}_M$ estão ativos; a transição positiva de relógio a meio caminho através deste estado transferirá o endereço no contador de programa para o REM. Durante o estado $T_2$, $C_P$ está ativo e o contador de programa é incrementado na transição positiva de relógio. Durante o estado $T_3$ , $\overline{CE}$ e $\overline{L}_I$ estão ativos; quando ocorre a transição positiva de relógio, a palavra da RAM endereçada é transferida para o registrador de instruções. A execução de **LDA** começa com o estado $T_4$ , onde $\overline{L}_M$ e $\overline{E}_I$ estão ativos; na transição positiva de relógio o campo de endereço no registrador de instruções é transferido para o REM. Durante o estado $T_5$ , $\overline{CE}$ e $\overline{L}_A$ estão ativos; isto significa que a palavra de dados de RAM endereçada é transferida para o acumulador na transição positiva de relógio. Conforme sabemos, o estado $T_6$ da rotina **LDA** é um nop.

### Rotina ADD

Suponhamos que no fim do ciclo de busca o registrador de instruções contenha **ADD BH**:

**IR** = 0001 1011

Durante o estado $T_4$ o campo de instruções vai para o controlador-sequencializador e o campo de endereços para o REM (ver Fig.10-6a). Durante este estado $\overline{E}_I$ e $\overline{L}_M$ estão ativos.

Os bits de controle $\overline{CE}$ e $\overline{L}_B$ estão ativos durante o estado $T_5$.Isto permite que a palavra de RAM endereçada prepare (sete) o registrador B (Fig. 10-6b). Usualmente, o carregamento ocorre a meio caminho através do estado quando a transição positiva de relógio atinge a entrada CLK do registrador B.

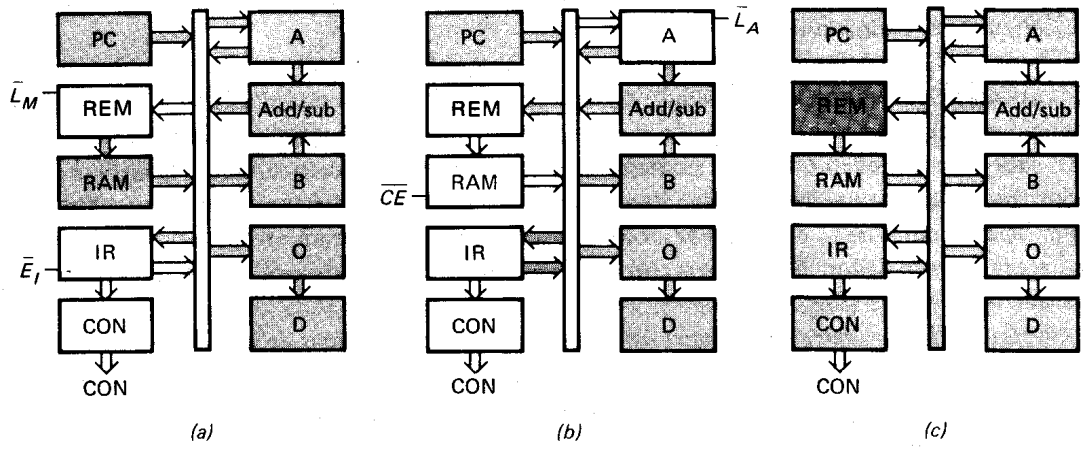



**Fig. 10-4 Rotina LDA:(a) estado $T_4$;(b) estado $T_5$ ;(c)estado $T_6$.**

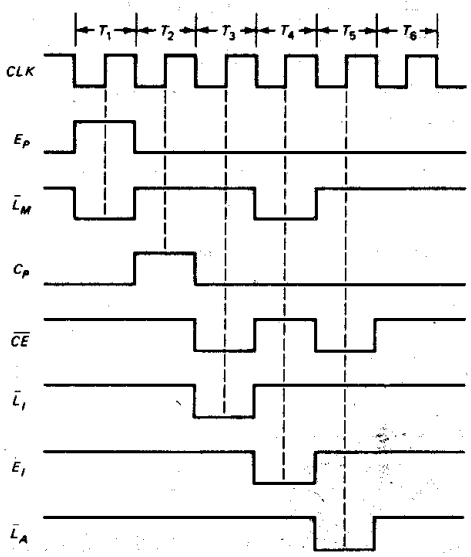



**Fig. 10-5 Diagrama de temporização de busca e LDA**

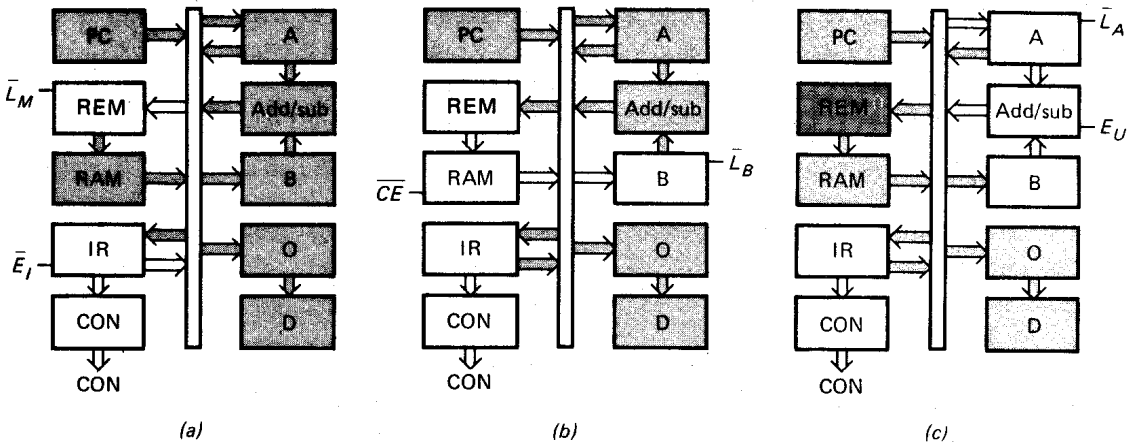



**Fig. 10-6 Rotinas ADD e SUB: (a)estado $T_4$ ;(b) estado $T_5$ ; (c) estado $T_6$.**

Durante o estado **$T_6$** ,$E_U$ e $\overline{L}_A$ estão ativos; portanto, o somador-subtrator estabelece ou prepara o acumulador (Fig. 10-6c). A meio caminho através deste estado, a transição positiva de relógio carrega a soma no acumulador.

A propósito, o tempo de preparação e o tempo de retardo de propagação evitam a corrida do acumulador durante este estado final de execução. Quando ocorre a transição positiva de relógio na Fig.10.6c, o conteúdo do acumulador se modifica, forçando o conteúdo do Somador-subtrator a mudar. Os novos conteúdos retomam à entrada do acumulador, mas os novos conteúdos não chegam lá até dois retardos de propagação após a transição positiva de relógio(um para o acumulador e um para o somador-subtrator), A esse tempo é demasiado tarde para preparar o acumulador. Isto evita a corrida do acumulador ( carregamento mais de uma vez na mesma transição de relógio).

A Fig.10-7 mostra o diagrama de temporização para as rotinas de busca e **ADD**. A rotina de busca é a mesma que a anterior: o estado **$T_1$** carrega o endereço **PC** no REM; o estado **$T_2$** incrementa o contador de programa; o estado **$T_3$** envia a instrução endereçada para o registrador de instrução.

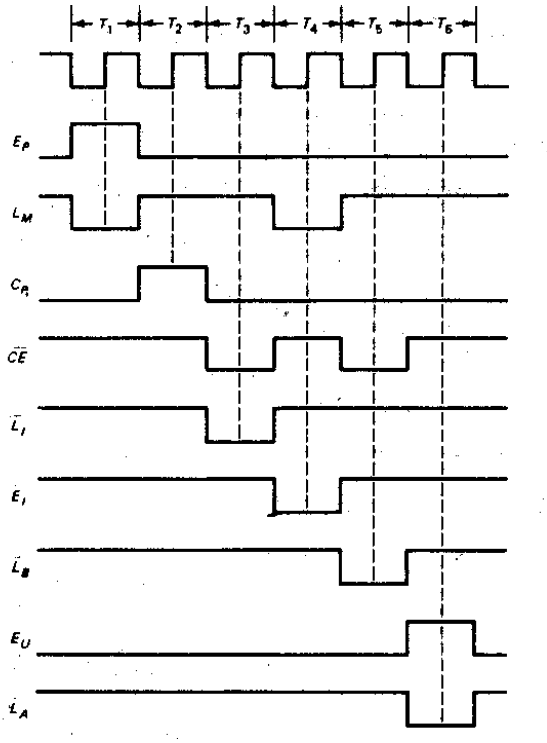



**Fig. 10-7 Diagrama de temporização de busca e ADD.**

Durante o estado **$T_4$** , $\overline{E}_I$ e $\overline{L}_M$ estão ativos; na próxima transição positiva de relógio, o campo de endereços no registrador de instrução vai para o REM. Durante o estado **$T_5$**. $\overline{CE}$ e $\overline{L}_B$ estão ativos; portanto, a palavra de RAM endereçada é carregada no registrador B a meio caminho através do estado. Durante o estado **$T_6$**, $\overline{E}_U$ e $\overline{L}_A$ estão ativos; quando ocorre a transição positiva de relógio, a soma fora do somador-subtrator é armazenada no acumulador .

**Rotina SUB**

A rotina **SUB** é similar à rotina **ADD**. A Fig.10-6a e b mostra as partes ativas de SAP-1 durante os estados $\mathbf{T_4}$ e $\mathbf{T_5}$. Durante o estado $\mathbf{T_6}$ , um $S_U$ alto é mandado ao somador-subtrator da Fig.10-6c. O diagrama de temporização é quase idêntico à Fig.10-7. Visualizamos $S_U$ baixo durante os estados $\mathbf{T_1}$ a $\mathbf{T_5}$ e $S_U$ alto durante o estado $\mathbf{T_6}$.

**Rotina OUT**

Suponhamos que o registrador de instruções contenha a instrução **OUT** no fim de um ciclo de busca. Então

**IR** = 1110 XXXX

O campo de instrução vai ao controlador-sequencializador para decodificação. Então o controlador-sequencializador emite a palavra de controle necessária para carregar o conteúdo do acumulador no registrador de saída.

A Fig. 10-8 mostra as secções ativas de SAP.1 durante a execução de uma instrução **OUT**.Uma vez que $E_A$ e $\overline{L}_0$ estão ativos, a próxima transição positiva de relógio carrega o conteúdo do acumulador no registrador de saída durante o estado $\mathbf{T_4}$ . Os estados $\mathbf{T_5}$ e $\mathbf{T_6}$ são nops.

A Fig.10.9 é o diagrama de temporização das rotinas **OUT** e de busca. Novamente, o ciclo de busca é o mesmo: estado de endereço, estado de incremento e estado de memória. Durante o estado $\mathbf{T_4}$ , $E_A$ e $\overline{L}_0$ estão ativos; isto transfere a palavra do acumulador para o registrador de saída quando ocorre a transição positiva de relógio.

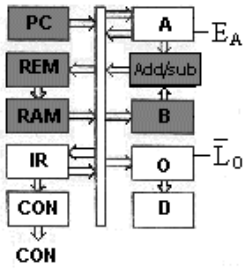



**Fig. 10-8 Estado $T_4$ da instrução.**

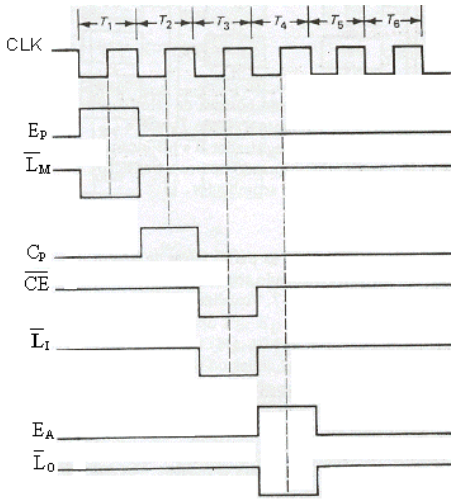



**Fig. 10-9 Diagrama de temporização de busca e de ADD.**

**HLT**

**HLT** não requer uma rotina de controle porque não há registradores envolvidos na execução uma instrução **HLT** .Quando o **IR** contiver.

**IR** = 1111 XXXX

o campo 1111 de instrução avisará ao controlador-sequencializador para interromper o processamento dos dados. O controlador-sequencializador pára o computador desligando o relógio(conjunto de circuitos estudado mais tarde).

**Ciclo de Máquina e Ciclo de Instrução**

SAP-1 tem seis estados T (três de busca e três de execução). Estes seis estados são chamados ***ciclo de máquina*** (ver Fig. 10-10a).É necessário um ciclo de máquina para buscar e executar cada instrução. O relógio SAP-1 tem uma freqüência de 1 kHz, equivalente a um período de 1 ms.Portanto, são necessários 6 ms para ocorrer um ciclo de máquina do SAP-1.

SAP-2 é ligeira diferente porque algumas de suas instruções levam mais do que um ciclo de máquina para buscar e executar. A Fig. 10-10b mostra a temporização (formas-de-onda) para uma instrução que requer dois ciclos de máquina. Os três primeiros estados **T** constituem o ciclo de busca; no entanto, o ciclo de execução requer os próximos nove estados **T**. É por isto que uma instrução de dois ciclos de máquina é mais complicada e necessita daqueles estados **T** extras para completar a execução.

O número de estados **T** necessário para buscar e executar uma instrução é chamado ***ciclo de instrução***. No SAP-1 o ciclo de instrução é igual ao ciclo de máquina. Enquanto no SAP-2 e em outros microcomputadores o ***ciclo de instrução*** pode ser igual a dois ou mais ciclos de máquina, conforme mostrado na Fig. 10.10b.

Os ciclos de instrução para o 8080 e para o 8085 levam de um a cinco ciclos de máquina(posteriormente, mais detalhes sobre isto).

EXEMPLO 10.5

O manual de programação do 8080/8085 diz que são necessários treze estados T para buscar e executar a instrução **LDA**.

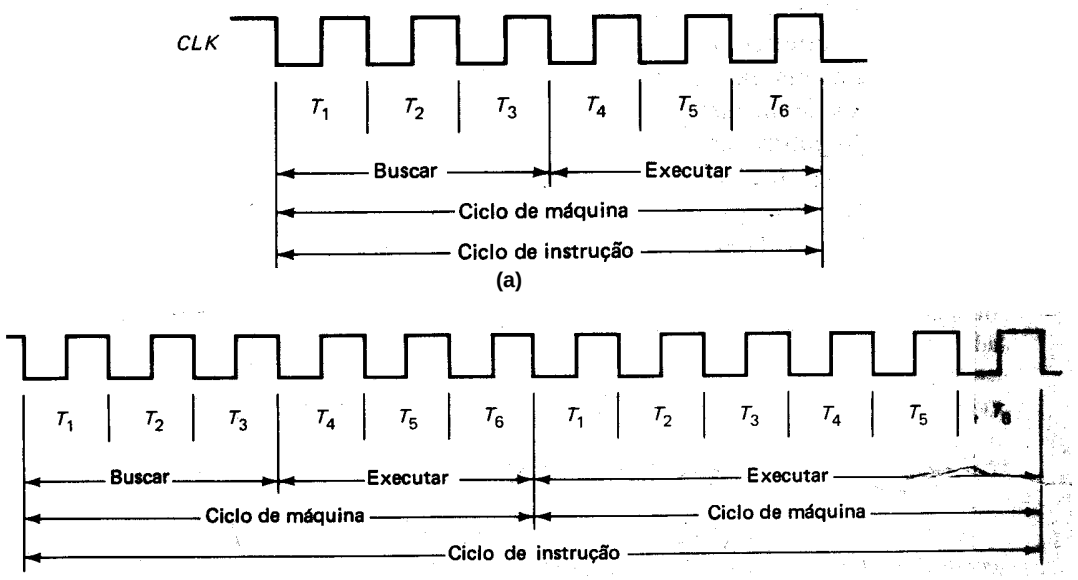



**Fig. 10-10(a) Ciclo de instrução do SAP-1;(b)ciclo de instrução com dois ciclos de máquina.**

Se o relógio do sistema tiver uma freqüência de 2,5 MHz, quanto tempo corresponderá a um ciclo de instrução?

SOLUÇÃO

O período do relógio é

$$T = \frac{1}{f} = \frac{1}{2{,}5MHz} = 400ns$$

Portanto, cada estado **T** dura 400 ns. Uma vez que são necessários treze estados **T** para buscar e executar a instrução **LDA**, o ciclo de instrução dura

13 x 400 ns = 5 200 ns = 5,2μs

EXEMPLO 10-6

A Fig. 10-11 mostra os seis estados **T** do SAP-1. A transição positiva de relógio ocorre a meio caminho em cada estado. Por que isto é importante?
SOLUÇÃO

SAP-1 é um computador organizado em barramentos (atualmente o tipo comum). Isto permite que seus registradores se comuniquem através do barramento W. Mas o carregamento confiável de um registrador ocorre apenas quando os tempos de preparação e de retenção (***hold***) forem satisfeitos. A espera de um meio-ciclo antes do carregamento do registrador satisfaz o tempo de preparação; a espera de um meio-ciclo após o carregamento satisfaz o tempo de retenção. É por isto que a transição positiva de relógio é projetada para bater nos registradores a meio ciclo.
em cada estado **T**(Fig. 10-11).

Há uma outra razão para esperar meio-ciclo antes do carregamento de um registrador.Quando a entrada ***HABILITA*** do registrador que emite tornar-se ativa, o conteúdo será subitamente descarregado no barramento W. Capacitância espúria e indutância da fiação impedem que as linhas do duto alcancem seus níveis de tensão corretos imediatamente. Em outras palavras obtemos transitórios no barramento W e temos que esperar que eles desapareçam para garantir dados válidos no instante do carregamento. O retardo de meio-ciclo antes da sincronização pernecessitamos sumarizar a execução das instruções do SAP-1 em uma tabela chamada ***microprograma***.

## 10-6 O MICROPROGRAMA DO SAP-1

Logo estaremos analisando o diagrama esquemático do computador SAP-1, mas primeiro necessitamos sumarizar a execução das instruções do SAP-1 em uma tabela chamada ***microprograma***.

### Microinstruções

O controlador-sequencializador emite palavras de controle, uma durante cada estado **T** ou ciclo de relógio. Estas palavras são como ordens ou instruções que dizem ao restante do computador o que fazer. Em virtude de produzir uma pequena etapa no processamento de dados, cada palavra de controle é chamada uma ***microinstrução***. Quando olhamos o diagrama-bloco do SAP-1(Fig. 10-1 ), podemos Visualizar uma corrente uniforme de microinstruções que flui do controlador-sequencializador para os outros circuitos do SAP-1.

### Macroinstruções

As instruções com que temos estado programando (**LDA**, **ADD**, **SUB**, ...) são às vezes chamadas macroinstruções para distingui-las das microinstruções. Cada macroinstrução do SAP-1 é formada

de três microinstruções. Por exemplo, a macroinstrução **LDA** consiste nas microinstruções na Tabela 10-3. Para simplificar o aspecto destas microinstruções, podemos usar compactação hexadecimal conforme mostrado na Tabela 10-4.

A Tabela 10-5 mostra o microprograma do SAP-1, uma listagem de cada macroinstrução e as microinstruções necessárias para executá-las. Esta tabela resume as rotinas de execução para as instruções do SAP-1. Uma tabela semelhante pode ser usada com conjuntos de instruções mais avançadas.

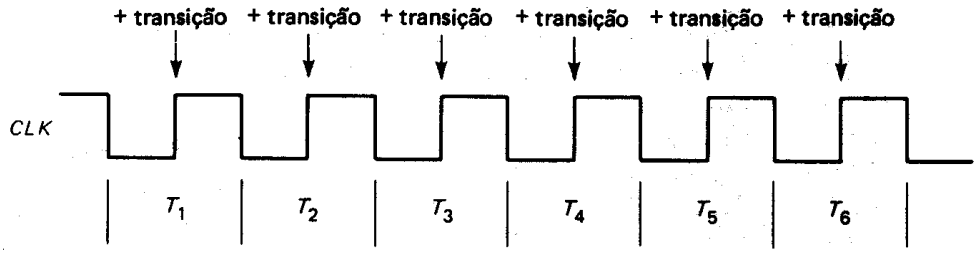



**Fig. 10-11 Transições positivas de relógio ocorrem a meio caminho nos estados T.**

### 10-7 O DIAGRAMA ESQUEMÁnco DO SAP-1

Nesta secção examinaremos o diagrama esquemático completo do SAP-1. As Figs.10-12 a 10.15 mostram todos os circuitos integrados (pastilhas), fios e sinais. Consultaremos estas figuras em toda a explicação seguinte. O Apêndice 3 apresenta detalhes adicionais de algumas das pastilhas (**CIs**) mais complicadas.

TABELA 10-3

| Macro | Estado | $C_P$ | $E_P$ | $\overline{L}_M$ | $\overline{CE}$ | $\overline{L}_I$ | $\overline{E}_I$ | $\overline{L}_A$ | $E_A$ | $S_U$ | $E_U$ | $\overline{L}_B$ | $\overline{L}_0$ | Ativo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **LDA** | $T_4$ | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | $\overline{L}_M$ , $\overline{E}_I$ |
| | $T_5$ | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | $\overline{CE}$ $\overline{L}_A$ |
| | $T_6$ | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | **Nada** |

TABELA 10-4

| Macro | Estado | CON | Ativo |
|---|---|---|---|
| **LDA** | $T_4$ | 1A3H | $\overline{L}_M$ , $\overline{E}_I$ |
| | $T_5$ | 2C3H | $\overline{CE}$ , $\overline{L}_A$ |
| | $T_6$ | 3E3H | **Nada** |

TABELA 10-5. **MICROPROGRAMA DO SAP-1***

| Macro | Estado | CON | Ativo |
|---|---|---|---|
| **LDA** | **$T_4$** | 1A3H | $\overline{L}_M, \overline{E}_I$ |
| | **$T_5$** | 2C3H | $\overline{CE}, \overline{L}_A$ |
| | **$T_6$** | 3E3H | **Nada** |
| **ADD** | **$T_4$** | 1A3H | $\overline{L}_M, \overline{E}_I$ |
| | **$T_5$** | 2E1H | $\overline{CE}, \overline{L}_B$ |
| | **$T_6$** | 3C7H | $\overline{L}_A, E_U$ |
| **SUB** | **$T_4$** | 1A3H | $\overline{L}_M, \overline{E}_I$ |
| | **$T_5$** | 2E1H | $\overline{CE}, \overline{L}_B$ |
| | **$T_6$** | 3CFH | $\overline{L}_A, S_U, E_U$ |
| **OUT** | **$T_4$** | 3F2H | $E_A, \overline{L}_0$ |
| | **$T_5$** | 3E3H | **Nada** |
| | **$T_6$** | 3E3H | **Nada** |

* $\mathrm{CON} = C_P E_P \overline{L}_M \overline{CE}\ \overline{L}_I \overline{E}_I \overline{L}_A E_A\ S_U E_U \overline{L}_B \overline{L}_0$

**Contador Programa**

As pastilhas **C1**, **C2** e **C3**. da Fig. 10-12 constituem o contador de programa. A pastilha **C1**,um **74LS107**, é um biestável duplo J-K mestre-escravo, que produz os 2 bits de endereço superiores. A pastilha **C2**, um outro **74LS107**, produz os 2 bits de endereço inferiores. A pastilha **C3** é um **74LS126**, uma chave quádrupla de três estados normalmente aberta; ela dá ao contador de programa uma saída de três estados.

No inicio de um processamento de computador, um $\overline{CLR}$ baixo restabelece o contador de programa 0000. Durante o estado **$T_1$** , um $E_P$ alto coloca o endereço no barramento W.

Durante o estado **$T_2$** . um $C_P$ alto é aplicado ao contador de programa a meio caminho através deste estado, a transição $\overline{CLR}$ negativa (equivalente à transição CLK positiva) incrementa o contador de programa.

O contador de programa é inativo durante os estados **$T_3$** a **$T_6$** .

**REM**

A pastilha **C4**, um **74LS173**, é um registrador de memória intermediária (buffer) de 4 bits; ele se como o REM. Observemos que os pinos 1 e 2 estão ligados à terra; isto converte a saída de três estados em uma saída de dois estados. Em outras palavras, a saída do REM não está conectada ao barramento W, e portanto não há necessidade de usar a saída de três estados.

**Multiplexador de 2-para-l**

A pastilha **C5** é um **74LS157**, um ***multiplexador de nibble de 2-para-1***. O *nibble* da esquerda (pinos 14,11 ,5, 2) vem do registrador (**$S_1$**) de chaves de endereço. O *nibble* da direita(pinos 13, 0, 6, 3) vem do REM. A chave (**$S_2$**) **RUN-PROG** seleciona o nibble para alcançar a saída de **C5**. Quando **$S_2$** estiver na posição **PROG**; o nibble fora do registrador de chaves de endereço será selecionado. Por outro lado, quando **$S_2$** estiver na posição RUN, a saída do REM será selecionada.

**RAM de 16 x 8**

As pastilhas **C6** e **C7** são **74189s**. Cada pastilha é uma RAM estática de 16 x 4. Juntas, elas nos dão uma ***memória de leitura-escrita*** de 16 x 8. $S_3$ é o registrador (8 bits) da chave de dados e $\mathbf{S_4}$ é a chave de leitura-escrita (uma chave com botão de calcar). Para programar a memória, $\mathbf{S_2}$ é posta na posição **PROG**; isto toma baixa a entrada $\overline{CE}$ (pino 2). As chaves de dados e de endereços são depois ajustadas com as palavras de dados e de endereços corretas.Calcando-se momentaneamente a chave de leitura-escrita, toma-se $\overline{WE}$ baixa (pino 3) e carrega-se a memória.

Depois do programa e dos dados estarem na memória, a chave ($S_2$) **RUN-PROG** será posta na posição RUN em preparação ao processamento no computador .

**Registrador de Instruções**

As pastilhas **C8** e **C9** são **74LS173s**. Cada pastilha é um registrador de memória intermediária de três estados e de 4 bits. As duas pastilhas constituem o ***registrador de instruções***. Ligando-se à terra os pinos 1 e 2 de **C8**, converte-se a saída de três estados em uma saída de dois estados, $\mathbf{I_7I_6I_5I_4}$. Este nibble vai ao decodificador de instruções no controlador-seqiiencializador. O sinal $\overline{E}_I$ controla a saída de **C9**, o nibble inferior no registrador de instruções. Quando $\overline{E}_I$ for baixo, este nibble será colocado no barramento W.

**Acumulador**

As pastilhas **C10** e **C11**, **74LS173s**, constituem o *acumulador* (Ver. Fig. 10-13). Os pinos 1 e 2 são ligados à terra em ambas as pastilhas para produzir uma saída de dois estados no somador-subtrator. As pastilhas **C12** e **C13** são **74LS126s**; estas chaves de três estados colocam o conteúdo do acumulador no barramento W quando $E_A$ está alto.

**Somador-subtrator**

As pastilhas **C14** e **C15** são **74LS86s**. Estas portas **EXCLUSIVE-OR** constituem um inversor controlado. Quando $S_U$ é baixo, o conteúdo do registrador B é transmitido. Quando $S_U$ é alto, o complemento de 1 é transmitido e um 1 é acrescentado ao **LSB** para formar o complemento de 2.

As pastilhas **C16** e **C17** são **74LS83s**. Estes somadores totais de 4 bits se combinam para produzir uma soma ou diferença de 8 bits. As pastilhas **C18** e **C19**, que são **74LS126s**, convertem esta resposta de 8 bits em uma saída de três estados para comandar o barramento W.

**Registrador B e Registrador de Saída**

As pastilhas **C20** e **C21**, que são **74LS173s**, formam o registrador B. Ele contém os dados a serem adicionados ou subtraídos do acumulador. Ligando-se à terra os pinos 1 e 2 de ambas as pastilhas, produz-se uma saída de dois estados para o somador-subtrator.

As pastilhas **C22** e **C23** são **74LS173s** e formam o registrador de saída. Ele comanda o indicador visual binário e nos permite ver os dados processados.

**Eliminador de Trepidação de Iniciar-Limpar"("Clear-Start Debouncer")**

Na Fig. 10-14, o ***eliminador de trepidação de iniciar-limpar*** produz duas saídas: **CLR** para o registrador de instruções e $\overline{CLR}$ para o contador de programa e para o contador em anel. **CLR** também vai a **C29**, o biestável de relógio-início (clock-start flip-flop). $S_5$ é uma chave de botão de comprimir. Quando comprimida (abaixada), ela vai para a posição **CLEAR**, gerando um alto **CLR** e

um baixo $\overline{CLR}$. Quando $S_5$ é liberada, ela retoma à posição **START**, produzindo um **CLR** baixo e um $\overline{CLR}$ alto.

Observemos que metade de **C24** é usada no eliminador de trepidação de iniciar-limpar e a outra metade no eliminador de trepidação de etapa única. A pastilha **C24** é uma porta **NAND** quádrupla de 2 entradas, do tipo 7400.

**Eliminador de trepidação de Etapa Única**

SAP-1 pode realizar processamento em qualquer um dos dois modos: manual ou automático. No modo manual, comprimimos e liberamos **$S_6$** para gerar um pulso de relógio. Quando **$S_6$** for comprimida, **CLK** será alto; quando estiver liberada, **CLK** será baixo. Em outras palavras, o eliminador de trepidação de etapa única da Fig.10-14 gera os estados **T** um de cada vez conforme comprimimos e liberamos o botão. Isto nos permite atravessar os diferentes estados **T** enquanto reparamos defeitos ou fazemos depuração. (Depuração significa procurar erros em nosso programa. Procuramos defeitos no ***hardware*** e depuramos o ***software***).

**Eliminador de trepidação Manual-automático**

A chave $S_7$ é uma chave de inversão monopolar (**SPDT**) que pode permanecer quer na posição **MANUAL** quer na posição **AUTO**. Quando na **MANUAL**, o botão de um único estágio está ativo. Quando em **AUTO**, o computador realiza o processamento automaticamente. Duas das portas **NAND** em **C26** são usadas para eliminar trepidação na chave **MANUAL-AUTO**. As duas outras portas **C26 NAND** constituem parte de uma estrutura **NAND/NAND** que orienta (ou guia) relógio de etapa única ou o relógio automático para as saídas finais **CLK** e $\overline{CLR}$.

**Memórias Intermediárias de Relógio ("Clock Buffers")**

A saída do pino 11, **C26**, comanda as ***memórias intermediárias de relógio***. Conforme vemos Fig.10-14, dois inversores são usados para produzir a saída final **CLK** e um inversor para produzir a saída $\overline{CLR}$. Ao contrário da maioria das outras pastilhas, **C27** é **TTL** padrão em vez de um Schottky de baixa potência (ver SAP-1, Lista das Partes, Apêndice 4). **TTL** padrão é usado porque ele pode comandar 20 cargas **TTL Schottky** de baixa potência, conforme indicado na Tabela 4-5.

Se verificarmos as folhas de características de dados do **74LS107** e do **74LSl73** quanto às correntes de entrada, seremos capazes de contar as seguintes cargas **TTL Schottky** (**LS**) de baixa potência nos sinais de relógio e de limpar (levar-a-0):

CLK = 19 LS cargas

$\overline{CLR}$ = 2 LS cargas

CLR = 1 LS cargas

$\overline{CLR}$ = 20 LS cargas

Isto significa que os sinais CLK e $\overline{CLR}$ fora de **C27** (**TTL** padrão) são adequados para comandar as cargas **TTL Schottky** de baixa potência. Além disso, os sinais CLR e $\overline{CLR}$ fora de **C24** (TTL padrão) podem comandar suas cargas.

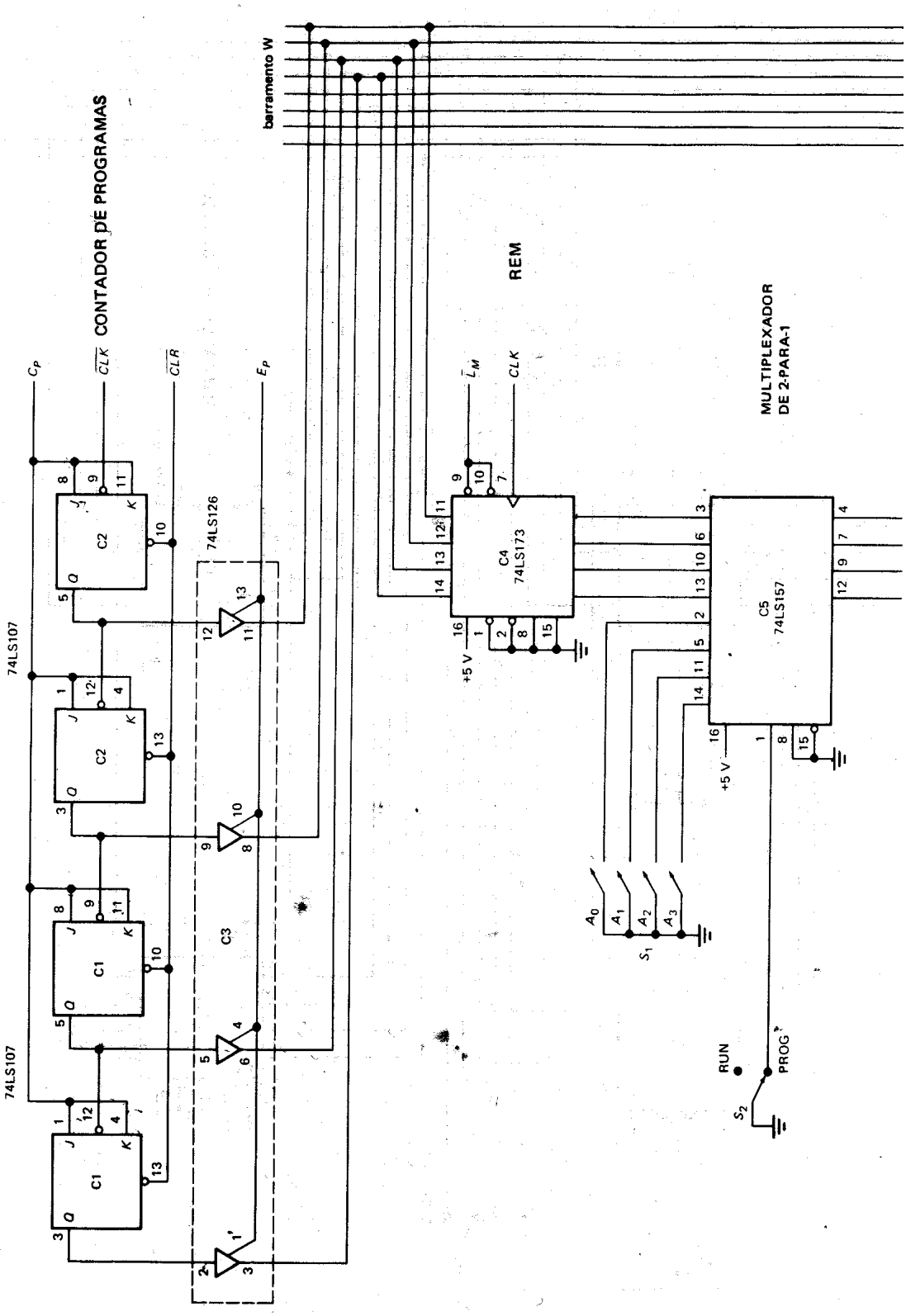



**Fig. 10-12 Contador de programa, memória e registrador de instruções do SAP-1.**

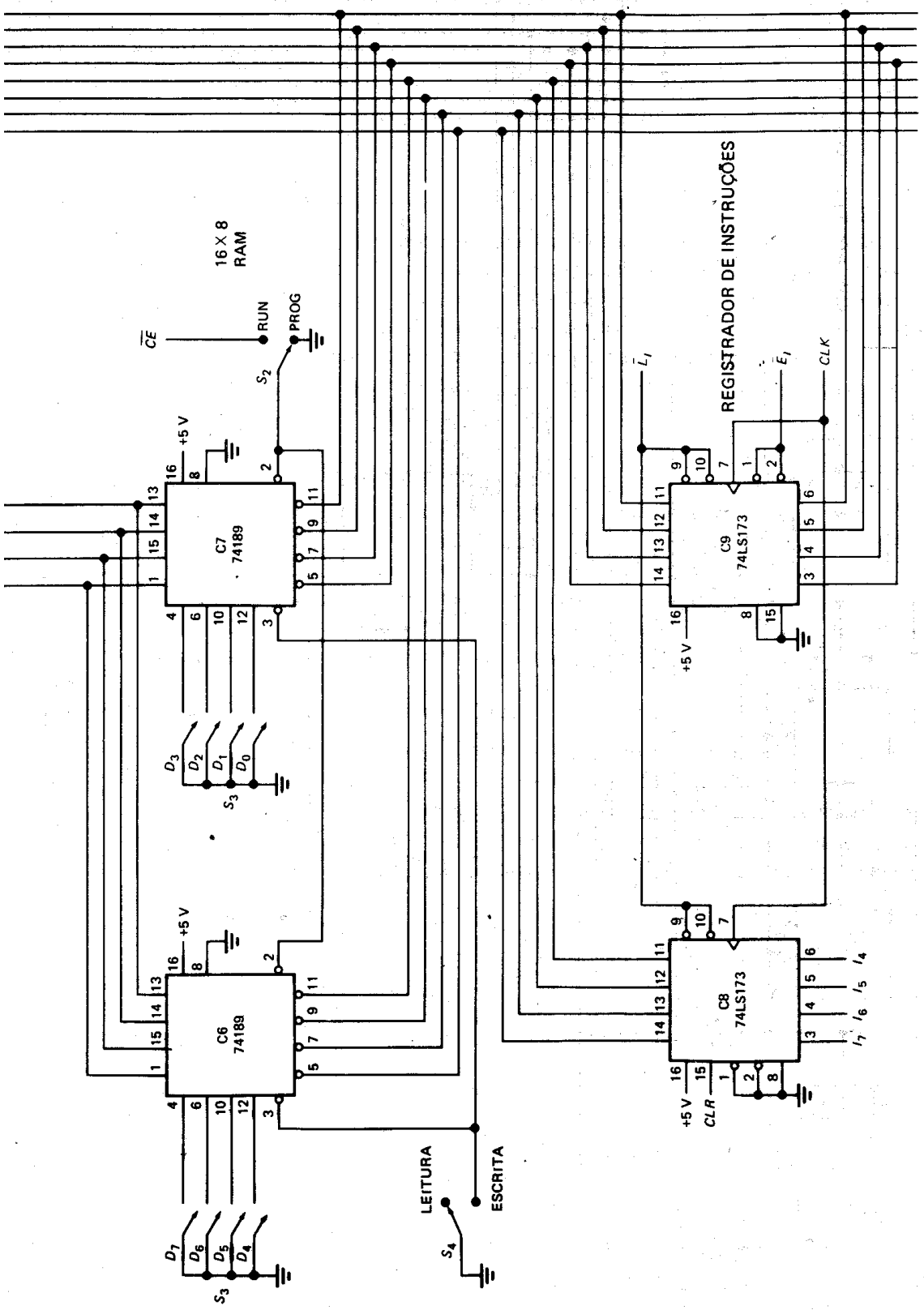



**Fig. 10-12 (Continuação )**

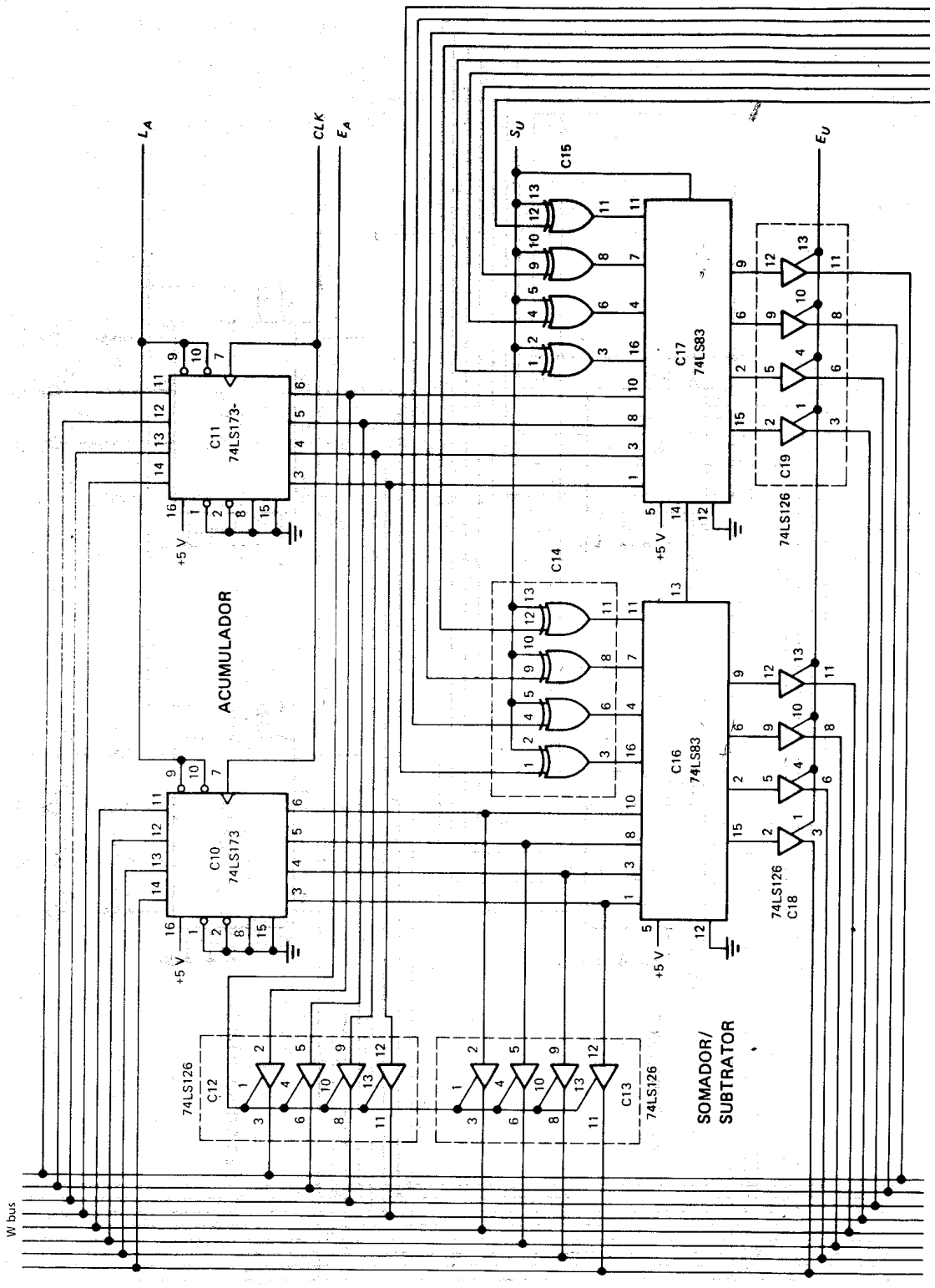



**Fig. 10-13 Registradores A e B, somador-subtrator e circuitos de saída.**

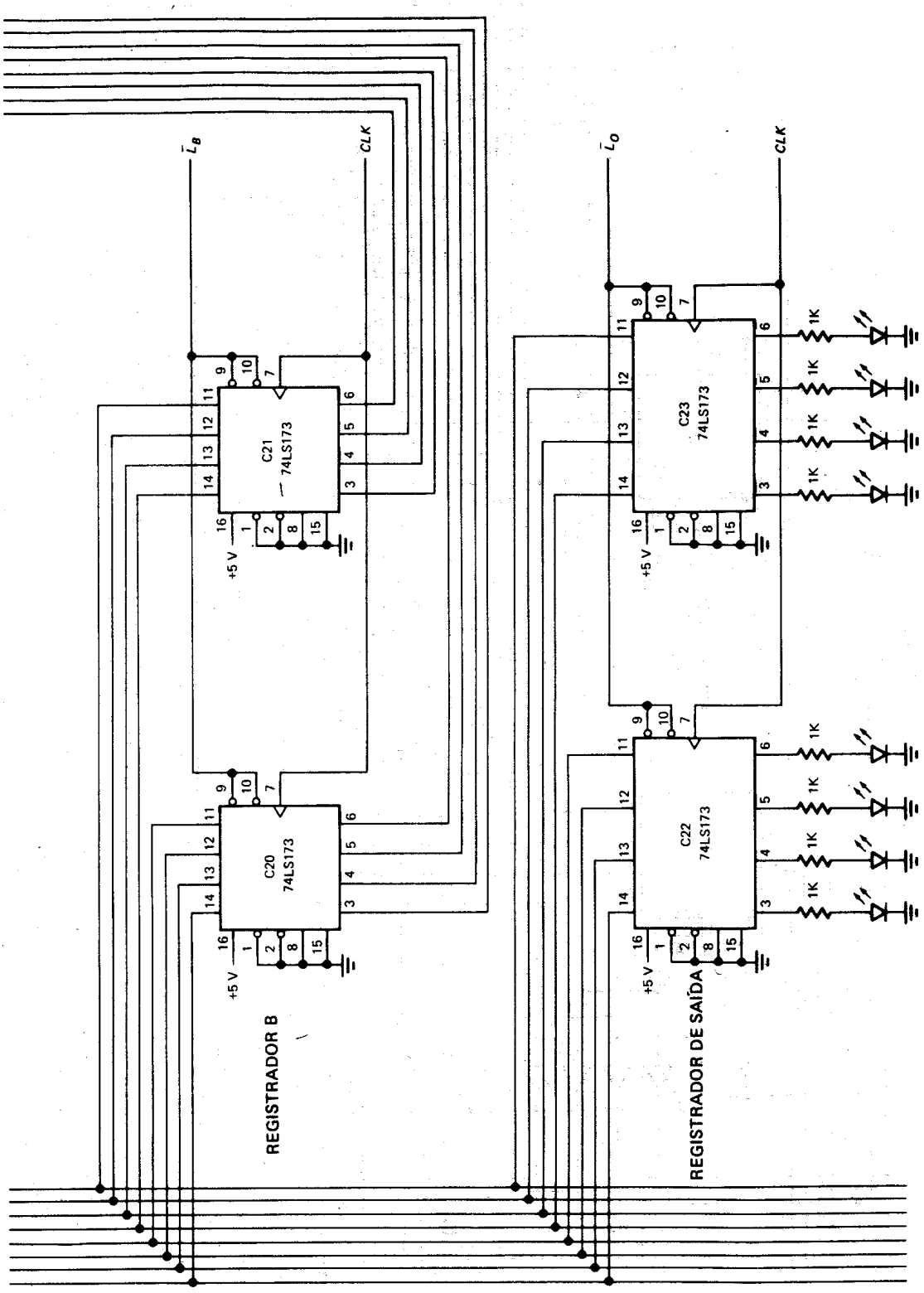



**Fig. 10-13 (Continuação)**

### Circuitos de Relógio e Fonte de Alimentação

A pastilha **C28** é um temporizador 555. Este **CI** produz uma saída retangular de **2 kHz** com um ciclo de atividade de 75 por cento. Conforme estudado anteriormente, um ***biestável*** (**C29**) de iniciar o relógio divide o sinal em **1 kHz** e ao mesmo tempo produz um ciclo de atividade de 50 por cento.

A fonte de alimentação consiste em um retificador em ponte de onda completa operando com um filtro a capacitor de entrada. A tensão cc no capacitor de 1000 µF é de aproximadamente 20V . A pastilha **C30**, um **LM340T-5**, é um regulador de tensão que produz uma saída estável de +5 V.

### Decodificador de Instrução

A pastilha **C31** , um inversor hexadecimal, produz complementos dos bits em código op,$I_7I_6I_5I_4$ (ver Fig. 10-15). Depois as pastilhas **C32**, **C33** e **C34** decodificam o código op, para produzir cinco sinais de saída: LDA, ADD, SUB, OUT e $\overline{HLT}$ . Lembremo-nos de que somente um destes sinais está ativo de cada vez. ($\overline{HLT}$ é ativo quando baixo; todos os outros são ativos quando altos.)

Quando a instrução **HLT** estiver no registrador de instruções, os bits $I_7I_6I_5I_4$ serão 1111 e $\overline{HLT}$ é baixo. Este sinal retomará a **C25** (relógio de etapa única) e a **C29** (relógio automático ).O relógio pára e o processamento no computador terminará, quer no modo **MANUAL** quer no **AUTO**.

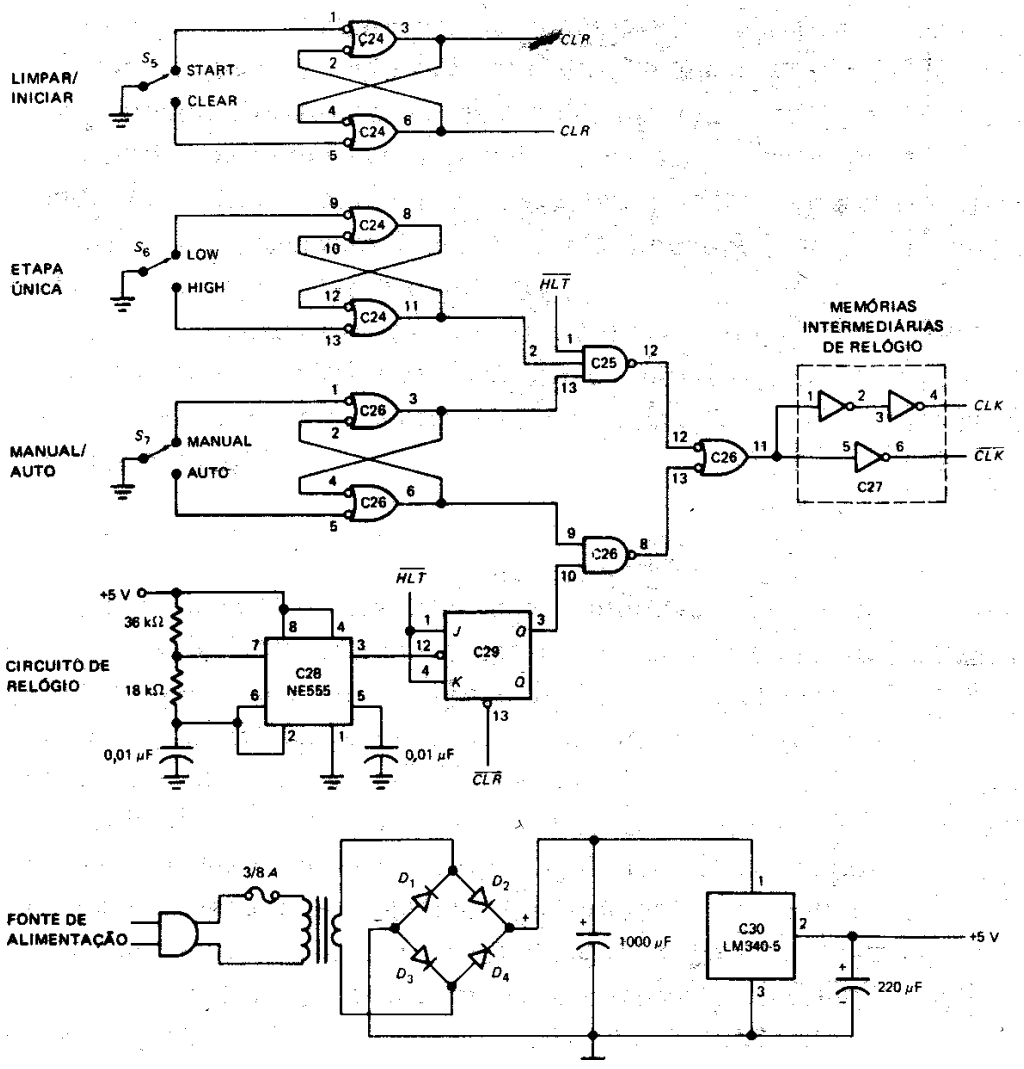



**Fig. 10-14 Circuitos da fonte de alimentação, de relógio e de limpar(zerar).**

**Contador em Anel**

O contador em anel, às vezes chamado ***contador de estados***, consiste em três pastilhas, C36, C37 e 38. Cada uma destas pastilhas é um 74LS107, um biestável duplo mestre-escravo **JK**. Este contador será restabelecido (levado-a-0) quando o botão ($S_5$) de limpar-iniciar for comprimido. O biestável $Q_0$ é invertido de modo que sua saída $\overline{Q}$ (pino 6, **C38**) comande a entrada $j$ do biestável $Q_1$ (pino 1, **C38**). Por causa disto, a saída $\mathbf{T_1}$ inicialmente alta.

O sinal **CLK** comanda uma entrada baixa ativa. Isto significa que a transição negativa do sinal **CLK** inicia cada estado **T**. Meio-ciclo mais tarde, a transição positiva do sinal **CLK** produz o carregamento do registrador, conforme descrito anteriormente.

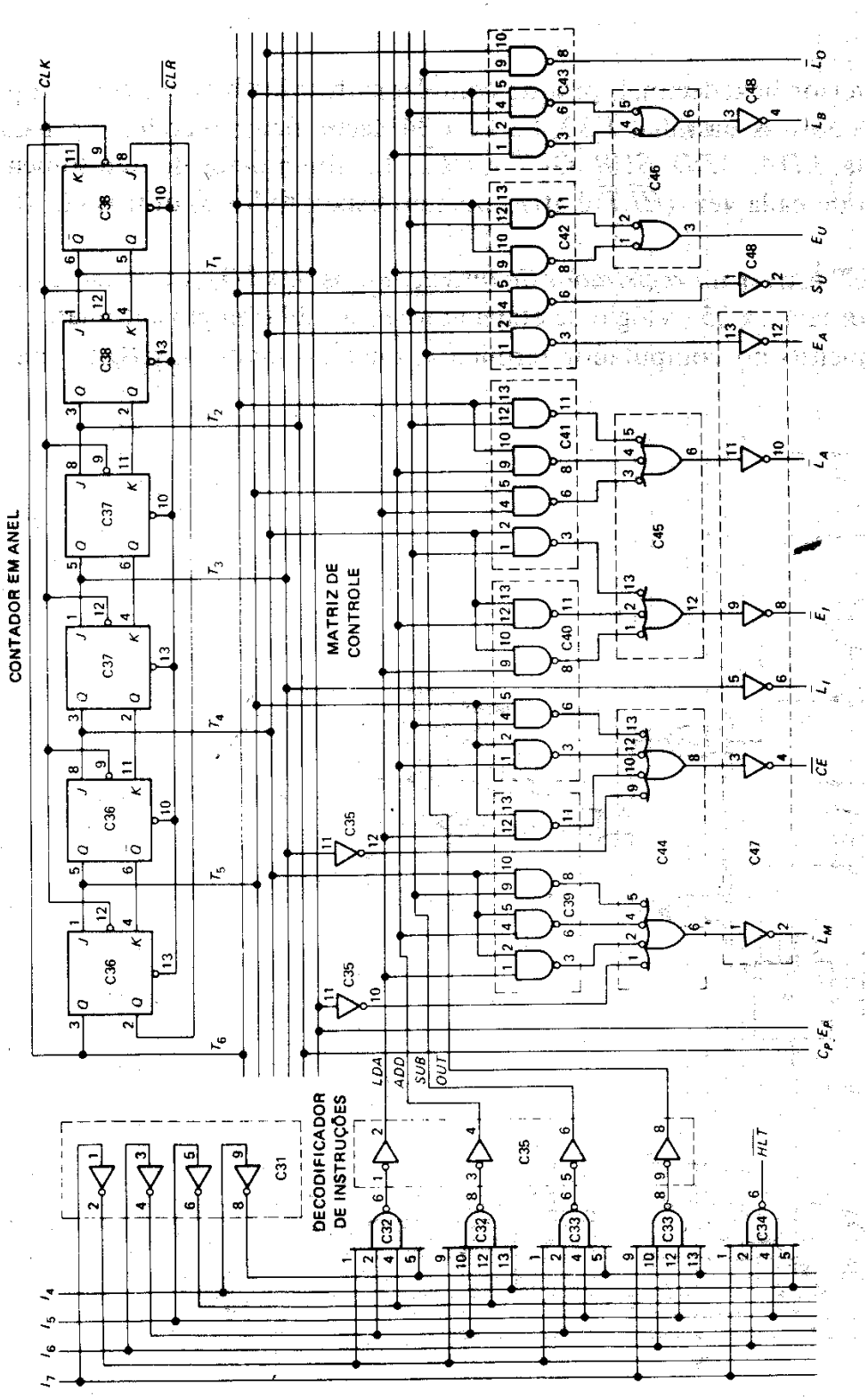



**Fig. 10-15 Decodificador de instruções, contador em anel e matriz de controle.**

**Matriz de Controle**

Os sinais **LDA**, **ADD**, **SUB** e **OUT** do decodificador de instruções comandam a matriz de controle, **C39** a **C48**. Ao mesmo tempo, os sinais do contador em anel, $\mathbf{T_1}$ a $\mathbf{T_6}$ , estão comandando a matriz (um circuito que recebe dois grupos de bits de diferentes fontes). A matriz produz **CON**, uma microinstrução de 12 bits que diz ao restante do computador o que fazer .

Na Fig. 10-15, $\mathbf{T_1}$ toma-se alto, depois $\mathbf{T_2}$, depois $\mathbf{T_3}$ etc. Analisemos a matriz de controle e eis o que encontraremos. Um $\mathbf{T_1}$ alto produz um $E_P$ alto e um $\overline{L_M}$ baixo (estado de endereços); um $\mathbf{T_2}$ alto resulta num $C_P$ alto (estado de incremento); e um $\mathbf{T_3}$ alto produz um $\overline{CE}$ baixo e um $\overline{L}_I$ baixo (estado de memória). Os três primeiros estados **T**, portanto, são sempre ciclo de busca no SAP-1. Em notação compactada, as palavras **CON** para o ciclo de busca são

| Estado | CON | Bits ativos |
|---|---|---|
| $T_1$ | 5E3H | $E_P, \overline{L}_M$ |
| $T_2$ | BE3H | $C_P$ |
| $T_3$ | 263H | $\overline{CE}, \overline{L}_I$ |

Durante os estados de execução, $\mathbf{T_4}$ a $\mathbf{T_6}$ tomam-se altos em sucessão. Ao mesmo tempo, somente um dos sinais codificados (**LDA** até **OUT**) está alto. Por causa disto, a matriz automaticamente guia os bits ativos para as linhas de controle da saída correta.

Por exemplo, quando **LDA** for alto, as únicas portas **NAND** de 2 entradas habilitadas serão a primeira, a quarta, a sétima e a décima. Quando $\mathbf{T_4}$ estiver alto, ele ativará a primeira e a sétima porta, resultando em $\overline{L_M}$ baixo e $\overline{E}_I$ baixo (carregam REM com o campo de endereço).Quando $\mathbf{T_5}$ estiver alto, ele ativará as quarta e décima porta **NAND**, produzindo um baixo $\overline{CE}$ e um baixo $\overline{L}_A$ (carregam dados de RAM no acumulador). Quando $\mathbf{T_6}$ tomar-se alto, nenhum dos bits de controle estarão ativos (nop).

Devemos analisar a ação da matriz de controle durante os estados de execução das possibilidades restantes: **ADD** alto, **SUB** alto e **OUT** alto. Depois concordaremos em que a matriz de controle poderá gerar as microinstruções **ADD**, **SUB** e **OUT** mostradas na Tabela 10-5 (microprograma do SAP-1).

**Operação**

Antes de cada processamento no computador, o operador introduz o programa e os dados na memória do SAP-1. Com o programa na memória inferior e os dados na memória superior, o operador comprime e libera o botão de limpar (zerar). Os sinais $CLK$ e $\overline{CLK}$ comandam os registradores e contadores. A microinstrução fora do controlador-se-quencializador determina o que acontece em cada transição CLK positiva.

Cada ciclo de máquina do SAP-1 começa com um ciclo de busca. $T_1$ é o estado de endereço, $T_2$ é o estado de incremento e $\mathbf{T_3}$ é o estado de memória. No fim do ciclo de busca a instrução é armazenada no registrador de instruções. Depois do campo de instrução ter sido decodificado, a matriz de controle automaticamente gerará a rotina de execução correta. Ao término do ciclo de execução, o contador em anel se restabelecerá (é zerado) e começará o próximo ciclo de máquina.

O processamento dos dados terminará quando uma instrução HLT for carregada no registrador de instruções.

## 10.8 MICROPROGRAMAÇÃO

A matriz de controle da Fig. 10-15 é uma das maneiras de gerar as microinstruções necessárias para cada ciclo de execução. Com maiores conjuntos de instruções, a matriz de controle torna-se muito complicada e requer centenas ou até mesmo milhares de portas. É por isto que o ***controle por fios fixos*** (***hardwired control*** = portas da matriz soldadas juntas) forçou os projetistas a procurar um modo alternativo de produzir palavras de controle que executam o processamento em um computador.

A ***microprogramação*** é a alternativa. A idéia básica consiste em armazenar microinstruções em uma **ROM** em vez de produzi-las com uma matriz de controle. Este processo simplifica o problema da construção de um controlador-sequencializador.

### Armazenamento do Microprograma .

Atribuindo-se endereços e incluindo-se a rotina de busca, podemos apresentar as microinstruções do SAP-1 mostradas na tabela 10-6. Estas microinstruções podem ser armazenadas em uma ROM de controle com a rotina de busca nos endereços **0H** a **2H**, a rotina **LDA** nos endereços **3H** a **5H**, a rotina **ADD** em **6H** a **8H**, a rotina **SUB** em **9H** e **BH** e a rotina **OUT** em **CH** a **EH**.

Para ter acesso a qualquer rotina, precisamos fornecer os endereços corretos. Por exemplo, para obter a rotina **ADD**, necessitamos fornecer os endereços **6H**, **7H** e **8H**. Para se obter a rotina **OUT** , fornecemos os endereços **CH**, **DH** e **EH**. Portanto, ter acesso a qualquer rotina requer três etapas:

1. Conhecimento do endereço de partida da rotina.
2. Escalonamento através dos endereços da rotina.
3. Aplicação dos endereços à ROM de controle.

### ROM de Endereços

Fig. 10-16 mostra como microprogramar o computador SAP-1. Ele tem uma ***ROM de endereço*** , um contador *pré-ajustável* e uma ROM de controle. A ROM de endereço contém os endereços de partida de cada rotina na Tabela 10-6. Em outras palavras, a ROM de endereços contém os dados listados na Tabela 10.7. Conforme mostrado, o endereço de partida da rotina **LDA** é **0000** etc. O endereço de partida da rotina **ADD** é **0110** etc.

Quando os bits $\mathbf{I_7I_6I_5I_4}$ do código op comandam a ROM de endereço, o endereço de partida é gerado. Por exemplo, se a instrução **ADD** estiver sendo executada, $\mathbf{I_7I_6I_5I_4}$ se **0001**. Esta é a entrada para a ROM de endereço; a saída desta ROM é 0110.

### Contador Pré-ajustável

Quando $\mathbf{T_3}$ for alto, a entrada de carga do ***contador pré-ajustável*** será alta e o contador carregará o endereço de partida da ROM de endereços. Durante os outros estados **T**, o contador contará.

Inicialmente, um sinal **CLR** alto proveniente do eliminador de trepidação de limpar-iniciar é diferenciado para se obter um estreito pico positivo. Isto restabelece (zera) o contador. Quando começa o processamento no computador, a saída do contador é 0000 durante o estado $\mathbf{T_1}$, 0001 durante o estado $T_2$ e 00 10 durante o estado $\mathbf{T_3}$ . Cada ciclo de busca é o mesmo, porque 0000, 0001 e 0010 saem do contador durante os estados $\mathbf{T_1}$, $\mathbf{T_2}$ e $T_3$.

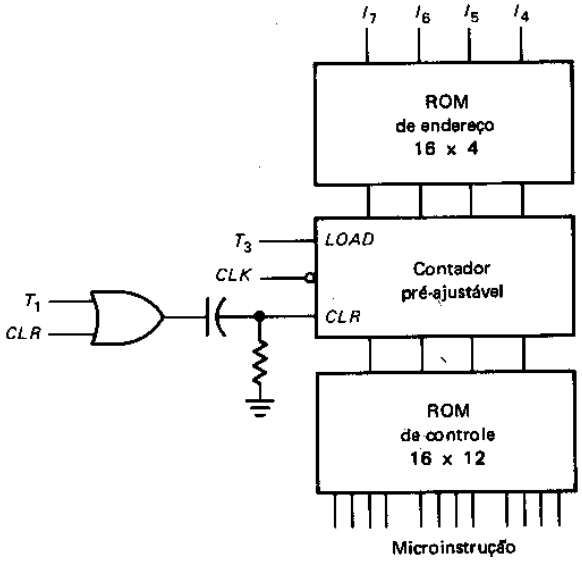



**Fig. 10-16 Controle microprogramado do SAP-1.**

**TABELA 10-6. ROM DE CONTROLE DO SAP-1***

| Endereço | Conteúdos | Rotina | Ativo |
|---|---|---|---|
| 0H | 5E3H | Fetch | $E_P, \overline{L}_M$ |
| 1H | BE3H | | $C_P$ |
| 2H | 263H | | $\overline{CE}, \overline{L}_A$ |
| 3H | 1A3H | LDA | $\overline{L}_M, \overline{E}_I$ |
| 4H | 2C3H | | $\overline{CE}, \overline{L}_A$ |
| 5H | 3E3H | | **Nada** |
| 6H | 1A3H | ADD | $\overline{L}_M, \overline{E}_I$ |
| 7H | 2E1H | | $\overline{CE}, \overline{L}_B$ |
| 8H | 3C7H | | $\overline{L}_A, E_U$ |
| 9H | 1A3H | SUB | $\overline{L}_M, \overline{E}_I$ |
| AH | 2E1H | | $\overline{CE}, \overline{L}_B$ |
| BH | 3CFH | | $\overline{L}_A, S_U, E_U$ |
| CH | 3F2H | OUT | $E_A, \overline{L}_0$ |
| DH | 3E3H | | **Nada** |
| EH | 3E3H | | **Nada** |
| **FH** | **X** | **X** | **Não usado** |

* $\mathbf{CON} = C_P E_P \overline{L}_M \overline{CE}\ \overline{L}_I \overline{E}_I \overline{L}_A E_A\ S_U E_U \overline{L}_B \overline{L}_0$

| Endereço | Conteúdos | Rotina |
|---|---|---|
| 0000 | 0011 | LDA |
| 0001 | 0110 | ADD |
| 0010 | 1001 | SUB |
| 0011 | XXXX | Nada |
| 0100 | XXXX | Nada |
| 0101 | XXXX | Nada |
| 0110 | XXXX | Nada |
| 0111 | XXXX | Nada |
| 1000 | XXXX | Nada |
| 1001 | XXXX | Nada |
| 1010 | XXXX | Nada |
| 1011 | XXXX | Nada |
| 1100 | XXXX | Nada |
| 1101 | XXXX | Nada |
| 1110 | 1100 | OUT |
| 1100 | XXXX | Nada |

O código op no registrador de instruções controla o ciclo de execução. Se uma instrução **ADD** tiver sido buscada, os bits $\mathbf{I_7I_6I_5I_4}$ serão 0001. Estes bits do código op comandam a ROM de endereços, produzindo uma saída de 0110 (Tabela 10-7). Este endereço de partida é a entrada para o contador pré-ajustável. Quando $T_3$ estiver alto, a próxima transição negativa de relógio carregará 0110 no contador pré-ajustável. O contador está agora ajustado (levado-a-1), e a contagem pode resumir-se no endereço de partida da rotina **ADD**. A saída do contador é 0110 durante o estado $\mathbf{T_4}$,0111 durante o estado $\mathbf{T_5}$ e 1000 durante o estado $\mathbf{T_6}$.

Quando começa o estado $\mathbf{T_1}$, a transição frontal do sinal $\mathbf{T_1}$ é diferenciada para produzir um estreito o pico positivo que restabelece o contador em 0000, o endereço de partida da rotina de busca.Começa então um novo ciclo de máquina.

**ROM de Controle**

A **ROM** de controle armazena as microinstruções do SAP-1. Durante o ciclo de busca, ela recebe os endereços 0000, 0001 e 0010. Portanto, suas saídas são

**5E3H**
**BE3H**
**263H**

Estas microinstruções, listadas na Tabela 10-6, produzem o estado de endereços, o estado de incremento o estado de memória.

Se uma instrução **ADD** estiver sendo executada, a **ROM** de controle receberá endereços 0110,0111 e 1000 durante o ciclo de execução. Suas saídas são

**1A3H**
**2E1H**
**3C7H**

Estas microinstruções executam a adição conforme discutido previamente.

Como um outro exemplo. suponhamos que a instrução **OUT** esteja sendo executada. Então o código op é 1110 e o endereço de partida é 1100 (Tabela 10-7). Durante o ciclo de execução, a saída do contador é 1100, 1101 e 1110. A saída da **ROM** de controle é 3F2H,3E3H e 3E3H (Tabela 10-6). Esta rotina transfere o conteúdo do acumulador para a porta de saída.

### Ciclo Variável de Máquina

A microinstrução 3E3H na Tabela 10-6 é uma nop. Ela ocorre uma vez na rotina **LDA** e duas vezes na rotina **OUT** .Estas nops são usadas em SAP-1 para se obter um ***ciclo fixo de máquina*** para todas as instruções. Em outras palavras, cada ciclo de máquina dura exatamente seis estados **T**, não importa qual a instrução. Em alguns computadores um ciclo de máquina fixo constitui uma vantagem. Mas quando a velocidade for importante, as nops constituirão um desperdício de tempo e poderão ser eliminadas.

Um modo de acelerar a operação do SAP-1 consiste em saltar qualquer estado **T** com uma nop. Reprojetando-se o circuito da Fig. 10-16 podemos eliminar os estados nop. Isto abreviará o ciclo de máquina da instrução **LDA** para cinco estados (**$T_1$**, **$T_2$**, **$T_3$**, **$T_4$** e **$T_5$**) .Também encurta-se o ciclo de máquina da instrução **OUT** para quatro estados (**$T_1$**, **$T_2$**, **$T_3$**, **$T_4$** e **$T_5$**) .

A Fig. 10-17 mostra um modo de se obter um ***ciclo de máquina variável***. Com uma instrução **LDA**, a ação é a mesma que anteriormente durante os estados **$T_1$** a **$T_5$**.Quando começa o estado **$T_6$**, a ROM de controle produz uma saída de 3E3H (a microinstrução nop). A porta **NAND** detecta esta nop instantaneamente e produz um sinal $\overline{NOP}$ de saída baixa. $\overline{NOP}$ é realimentado para o contador em anel através de uma porta **AND**, conforme mostrado na Fig. 10-18.Isto restabelece o contador em anel no estado **$T_1$**, e começa um novo ciclo de máquina. Isto reduz o ciclo de máquina da instrução **LDA** de seis estados para cinco.

Com a instrução **OUT** , ocorre a primeira nop no estado **$T_5$** .Neste caso, logo após começar o estado **$T_5$** , a **ROM** de controle produz uma saída de 3E3H, que é detectada pela porta **NAND**.O baixo sinal $\overline{NOP}$ então restabelece o contador em anel para o estado **$T_1$**. Desta maneira, reduzimos o ciclo de máquina da instrução **OUT** de seis estados para quatro.

Ciclos variáveis de máquina são comumente usados com microprocessadores. No 8085,por exemplo, os ciclos de máquina duram de dois a seis estados **T** porque todos os estados nop indesejáveis são ignorados.

### Vantagens

Uma vantagem da microprogramação é a eliminação do decodificador de instruções e da matriz de controle; ambos tomam-se muito complicados em maiores conjuntos de instruções.Em outras palavras, é muito mais fácil armazenar microinstruções em uma ROM do que montar um decodificador de instruções e matriz de controle.

Além disso, uma vez que montamos um decodificador de instruções e a matriz de controle, a única maneira pela qual podemos alterar o conjunto de instruções é desconectando e montando novamente. Isto não é necessário com o controle microprogramado; tudo o que temos a fazer é modificar a **ROM** de controle e a **ROM** de endereço de partida. Isto será uma grande vantagem se estivermos tentando melhorar o equipamento vendido inicialmente.

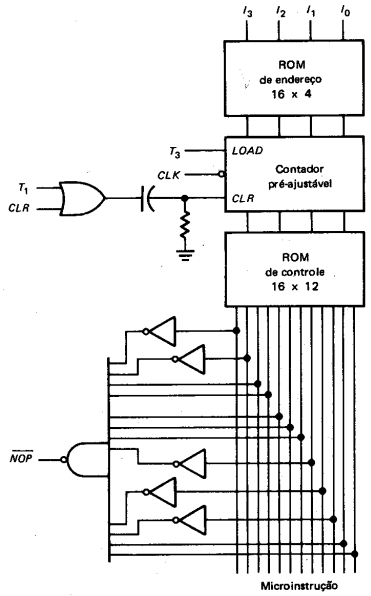



**Fig. 10-17 Ciclo de máquina variável.**

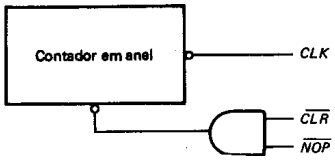



**Fig. 10-18**

**Resumo**

Concluindo, a maioria dos computadores construídos atualmente usa controle microprogramado em vez de controle por fios fixos. As tabelas e os circuitos de microprogramação costumam ser mais complicados do que os do SAP-1, mas a idéia é a mesma. As microinstruções são armazenadas numa ROM de controle e tem-se acesso a elas pela aplicação do endereço da microinstrução desejada.

**GLOSSÁRIO**

**Acumulador**: O local onde estão acumuladas as respostas às operações aritméticas e lógicas.Ás vezes chamado registrador A.

**Ciclo de busca (fetch)**: A primeira parte do ciclo de instrução. Durante o ciclo de busca, o endereço é mandado à memória, o contador de programa é incrementado e a instrução é transferida da memória para o registrador de instruções.

**Ciclo de instruções**: Todos os estados necessários para buscar e executar uma instrução.

**Ciclo de máquina**: Todos os estados gerados pelo contador em anel.

**Código op**: Código de operação. Aquela parte da instrução que diz ao computador que operação efetuar.

**Conjunto de instruções**: As instruções a que respondem um computador.

**Contador de programas**: Um registrador que conta em binário. Seu conteúdo é o endereço da próxima instrução a ser buscada na memória.

**Estado de endereços**: O estado $\mathbf{T_1}$. Durante este estado, o endereço no contador de programa é transferido para o **REM**.

**Estado de incremento**: O estado $\mathbf{T_2}$. Durante este estado, o contador de programa é incrementado.

**Estado da memória**: O estado $\mathbf{T_3}$. Durante este estado, a instrução na memória é transferida para o registrador de instruções.

**Instrução de consulta à memória**: Uma instrução que solicita uma segunda operação da memória para se ter acesso aos dados.

**LDA**: Mnemônico para carregar o acumulador.

**Linguagem de máquina**: Os cordões de 0s e de 1s usados em um programa.

**Linguagem de montagem**: Os rnnemônicos usados na escrita de um programa.

**Macroinstrução**: Uma das instruções no conjunto de instruções.

**Microinstrução**: Uma palavra de controle fora do controlador-sequencializador. A menor etapa no processamento de dados.

**Nop**: **Sem operação**. Um estado durante o qual nada acontece.

**Programa fonte**: Um programa escrito em mnemônicos.

**Programa objeto**: Um programa escrito em linguagem de máquina.

**RAM**: Memória de acesso aleatório. Uma melhor nome é memória de leitura-escrita. A RAM armazena o programa e os dados necessários a um processamento no computador.

**Registrador B**: Um registrador auxiliar que armazena os dados a serem adicionados ou subtraídos do acumulador.

**Registrador de instruções**: O registrador que recebe a instrução da memória.

**Registrador de saída**: O registrador que recebe dados processados do acumulador e comanda o indicador visual de saída do SAP-1. Também chamado porta de saída.

**REM**: Registrador de endereços da memória. Este registrador recebe o endereço dos dados a que se vai ter acesso na memória. O **REM** fornece este endereço à memória.

**EXERCÍCIOS DE FIXAÇÃO**

Completar as lacunas. A resposta aparece no início da próxima pergunta.

1. O contador de_____ que constitui parte da unidade de controle, conta de 0000 a 1111. Ele envia à memória o _____ da próxima instrução.
2. (*programa, endereço*) O **REM**, ou registrador de _____ , retém o endereço do contador de programas. Um bit mais tarde, o **REM** aplica este endereço à _____, onde é efetuada uma operação de leitura.
3. (*endereços da memória, RAM*) O registrador de instruções constitui parte da unidade de controle. O conteúdo do registrador de _____ é dividido em dois nibbles (meios-bytes). O nibble superior vai para o _____.
4. (*instrução, controlador-sequencializador*) O controlador-seqiiencializador produz uma palavra de 12 bits que controla o restante do computador. Os 12 fios que transportam esta _____ são chamados _____ .
5. (*palavra de controle, barramento de controle*) O é um registrador de memória intermediária que armazena somas ou diferenças. Sua saída de dois estados vai para o somador-subtrator .O _____ produzirá a soma quando $S_U$ for baixo e a diferença quando $S_U$ for alto. O registrador de saída é às vezes chamado _____ .
6. (*acumulador, somador-subtrator, porta de saída*) O conjunto de _____ do **SAP-1** é **LDA**, **ADD**, **SUB**, **OUT**, e **HLT**; **LDA**, **ADD** e **SUB** são chamadas instruções de _____ porque elas usam dados armazenados na memória.
7. (*instruções, consulta à memória*) O 8080 foi o primeiro microprocessador amplamente usado. O _____ é uma versão ampliada do 8080 tendo essencialmente o mesmo conjunto de instruções.
8. (*8085*) **LDA**, **ADD**, **SUB**, **OUT** e **HLT** são codificadas como cordões de 4 bits de 0s e 1s.Este código é chamado _____. A linguagem _____ usa mnemônicos quando escrevendo um programa. A linguagem _____ usa cordões de 0s e 1s.
9. (*código op, de montagem, de máquina*) **SAP-1** tem _____ estados **T**, períodos durante os quais o conteúdo dos registradores se modificam. O contador em anel, ou contador _____ produz estes estados **T**. Estes seis estados **T** representam um um ciclo de máquina. No **SAP-1** o ciclo de instruções tem apenas um ciclo de máquina.

Nos microprocessadores como o 8080 e o 8085, o ciclo de _____ pode ter de um a cinco ciclos de máquina.

**10.** (*seis, de estados, instrução*) O controlador-sequencializador envia palavras de controle, uma durante cada estado **T** ou ciclo de relógio. Cada palavra de controle é chamada _____.Instruções como **LDA**, **ADD**, **SUB** etc. são chamadas _____.Cada macroinstrução **SAP-1** é constituída de três _____ .

**11.** (*microinstrução, macroinstruções, microinstruções*) Com maiores conjuntos de instruções, a matriz de controle torna-se muito complicada. Isto é porque o controle por fios fixos está sendo substituído pela _____.A idéia básica consiste em armazenar as _____numa **ROM** de controle.

**12.** (*microprogramação, microinstruções*) **SAP-1** usa um ciclo de máquina fixo em todas as instruções. Em outras palavras, cada ciclo de máquina leva exatamente seis estados **T**. Os microprocessadores como o 8085 têm ciclos de máquina variáveis porque todos os estados nop indesejáveis são eliminados.

**PROBLEMAS**

**10.1**. Escrever um programa **SAP-1** usando mnemônicos (semelhantes ao Exemplo 10.1) que apresente o resultado de

5+4-6

Usar endereços **DH**, **EH** e **FH** para os dados.

**10.2**. Converter a linguagem de montagem do Probl. 10.1 em linguagem de máquina **SAP-1**.Mostrar a resposta em forma binária e em forma hexadecimal.

**10.3**. Escrever um programa em linguagem de montagem que efetue esta operação:

8+4-3+5-2

Usar endereços **BH** a **FH** para os dados.

**10.4**. Converter o programa e os dados do Probl. 10.3 em linguagem de máquina. Expressar o resultado tanto em forma binária quanto em forma hexadecimal.

**10.5**. A Fig. 10-19 mostra o diagrama de temporização para a instrução **ADD**. Desenhar o diagrama de temporização para a instrução **SUB**.

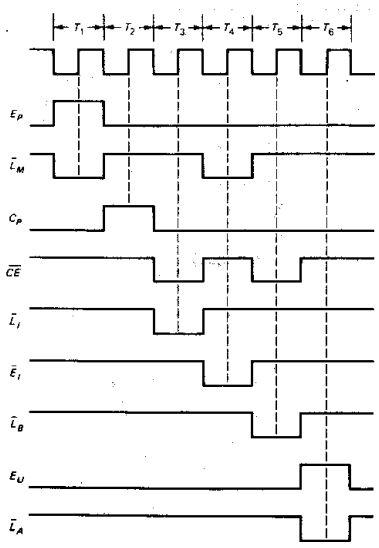



**Fig. 10-19**

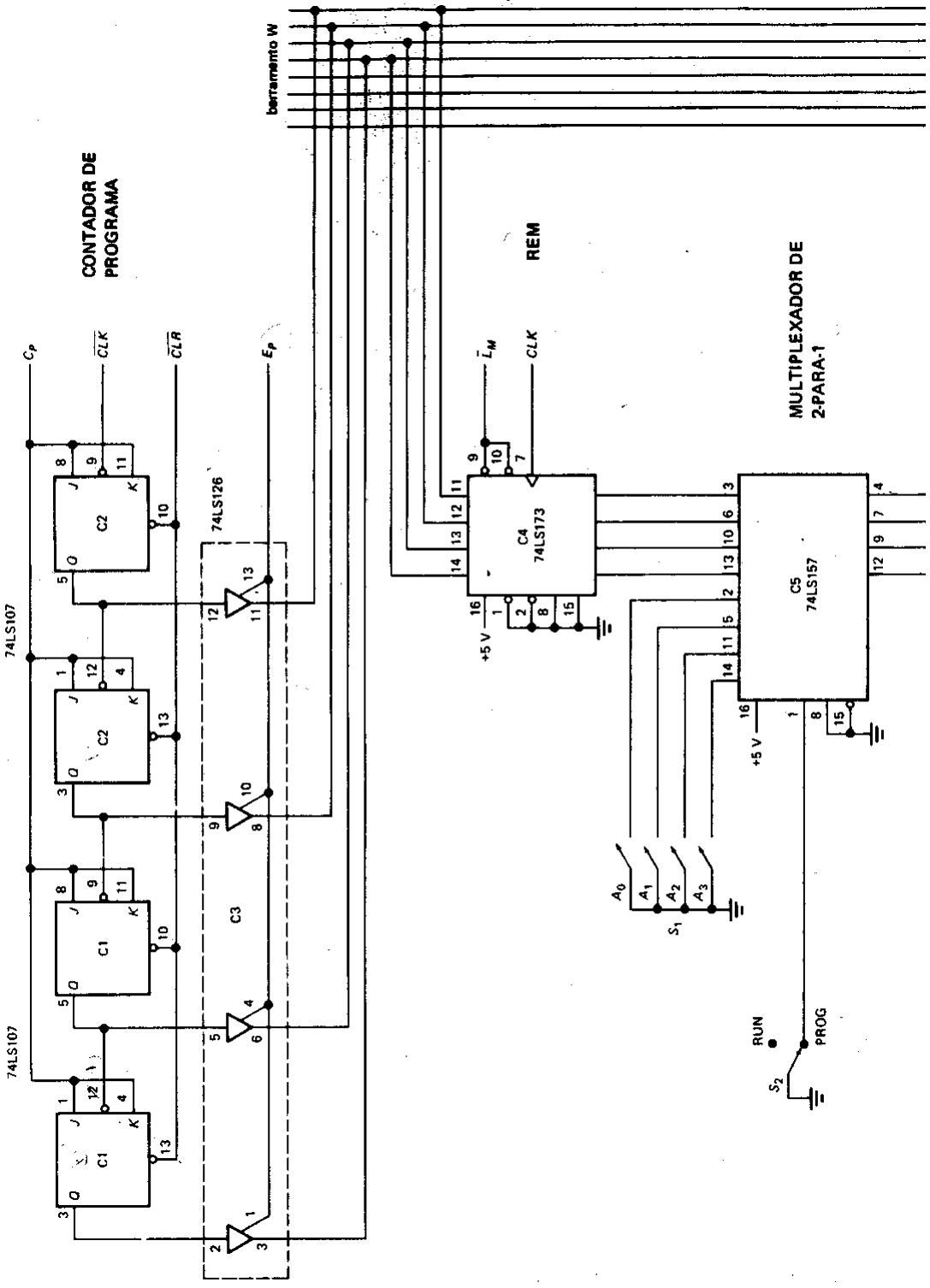

barramento W
CONTADOR DE PROGRAMA
REM
MULTIPLEXADOR DE 2-PARA-1
74LS107
74LS107
74LS126
C1
C2
C3
C4 74LS173
C5 74LS157
+5 V
RUN
PROG

Fig. 10-20

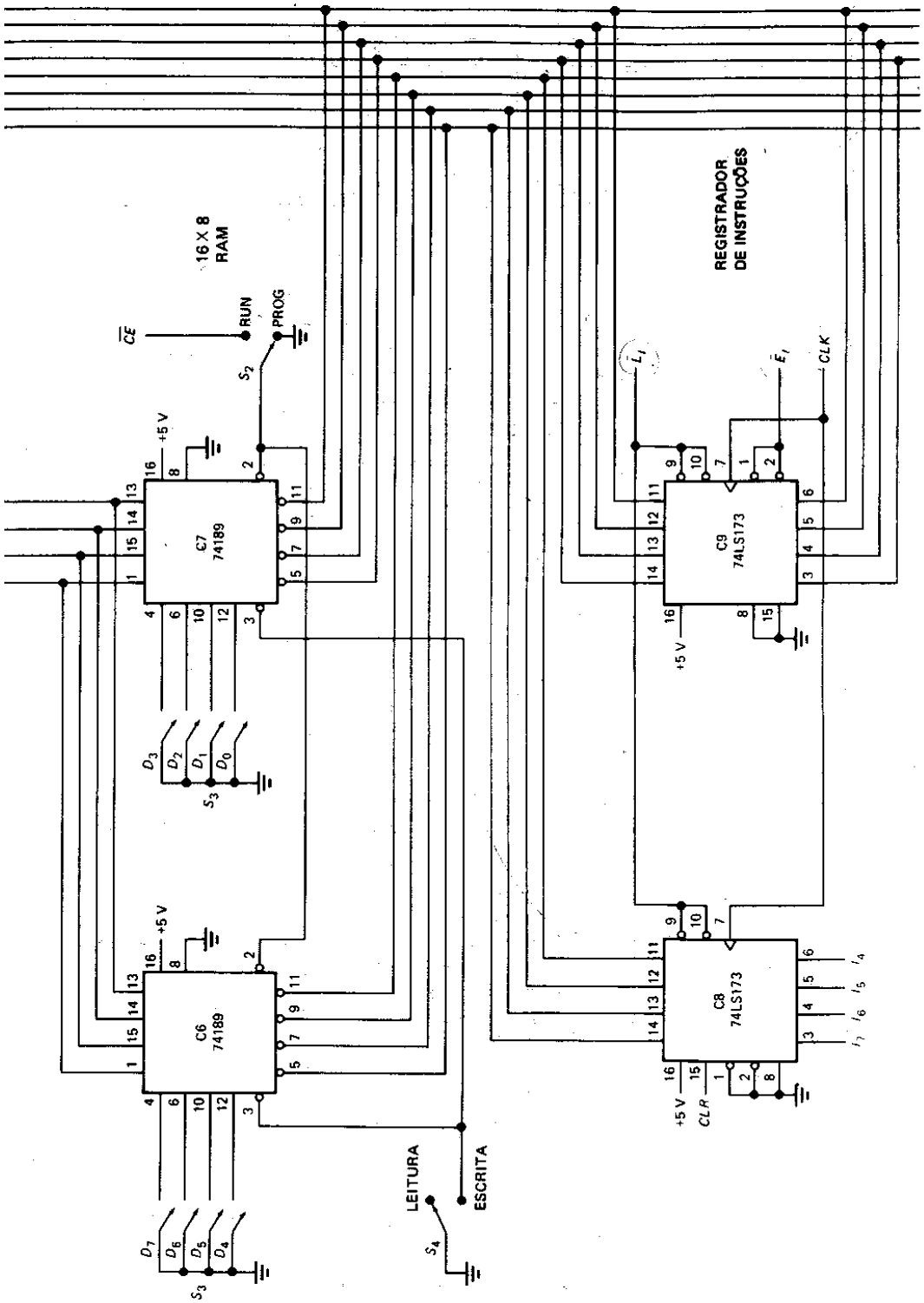



**Fig 10-20 (continuação)**

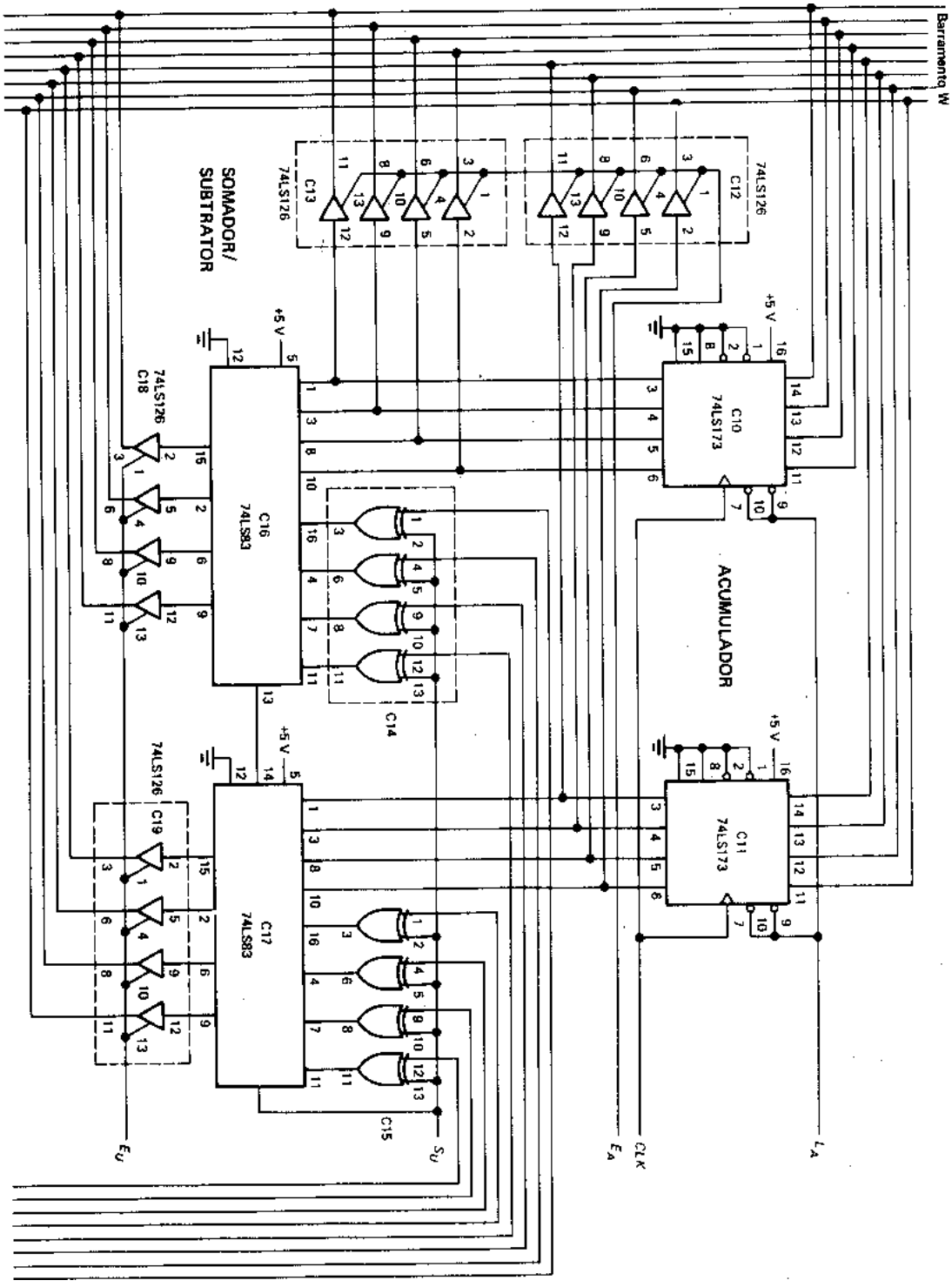

Barramento W
SOMADOR/
SUBTRATOR
ACUMULADOR
74LS126
C12
C13
74LS126
C10
74LS173
C11
74LS173
C16
74LS83
C17
74LS83
C14
C15
C18
74LS126
C19
74LS126
+5 V
$E_U$
$S_U$
$E_A$
CLK
$L_A$

Fig 10-21

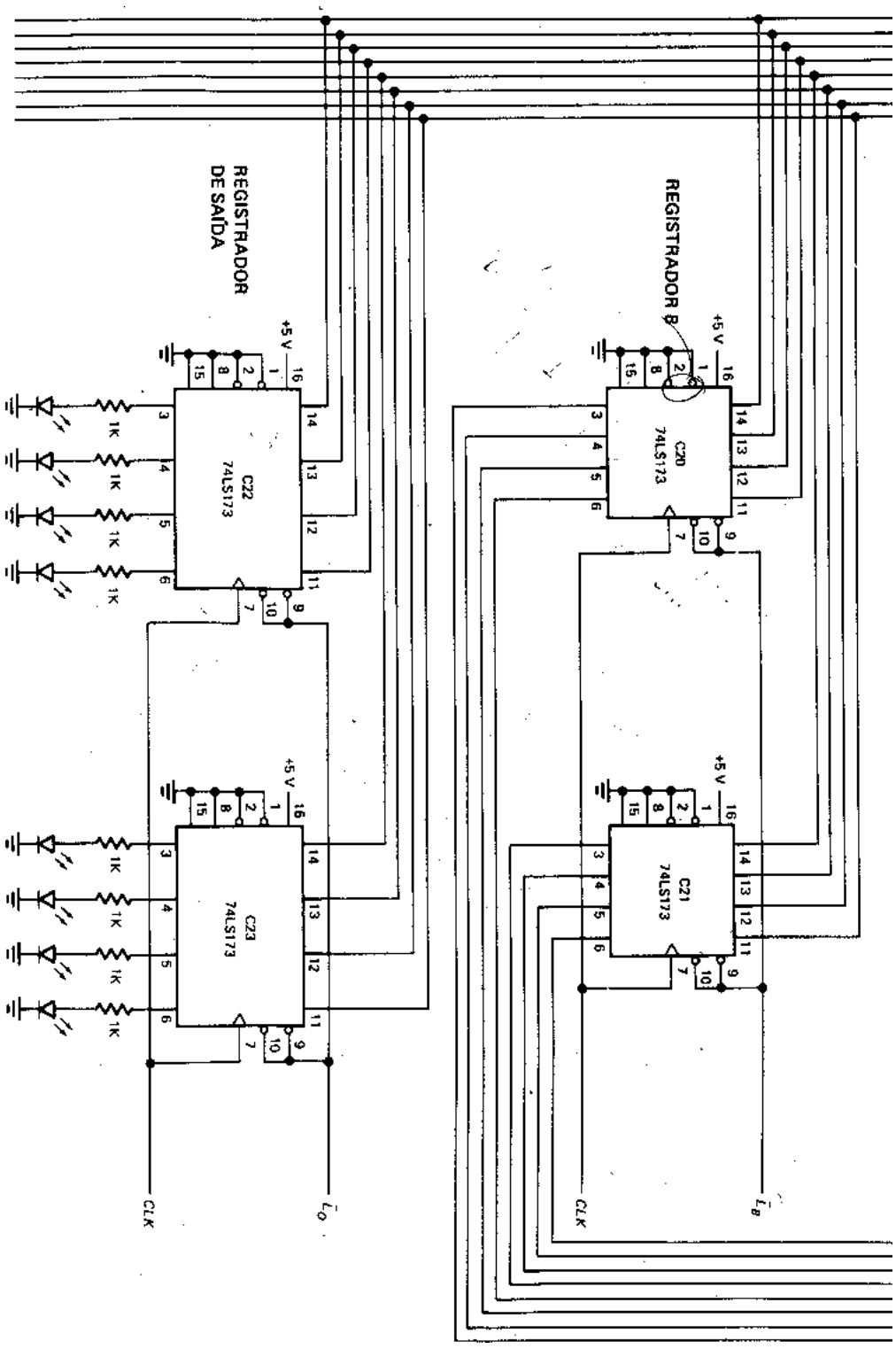



**Fig 10-21(Continuação)**

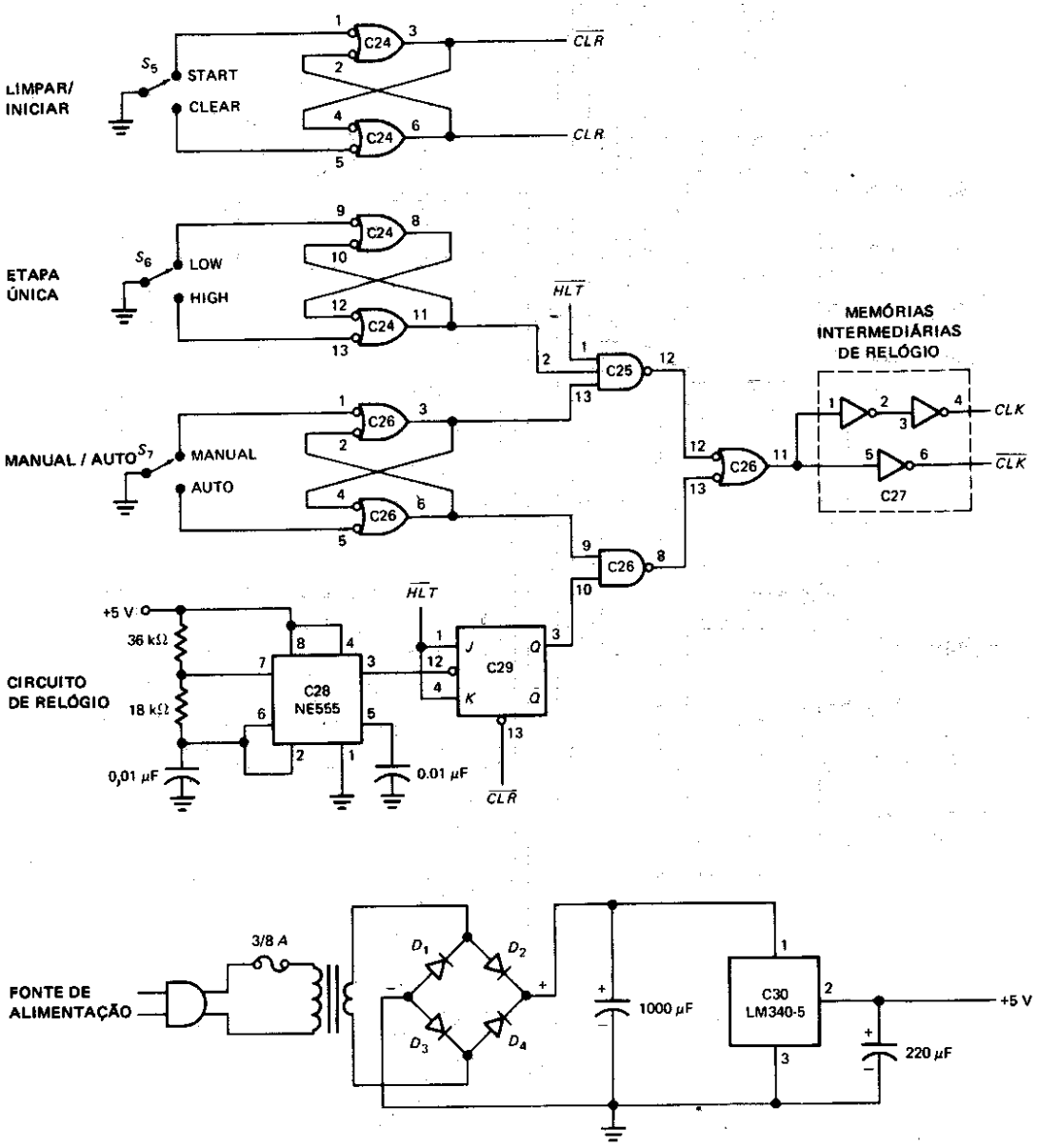

LIMPAR/
INICIAR
$S_5$
START
CLEAR
C24
$\overline{CLR}$
ETAPA
ÚNICA
$S_6$
LOW
HIGH
$\overline{HLT}$
MEMÓRIAS
INTERMEDIÁRIAS
DE RELÓGIO
C25
C26
C27
CLK
$\overline{CLK}$
MANUAL / AUTO
$S_7$
MANUAL
AUTO
+5 V
36 kΩ
18 kΩ
0,01 μF
C28
NE555
C29
0.01 μF
CIRCUITO
DE RELÓGIO
FONTE DE
ALIMENTAÇÃO
3/8 A
$D_1$
$D_2$
$D_3$
$D_4$
1000 μF
C30
LM340-5
220 μF
+5 V

Fig. 10-22

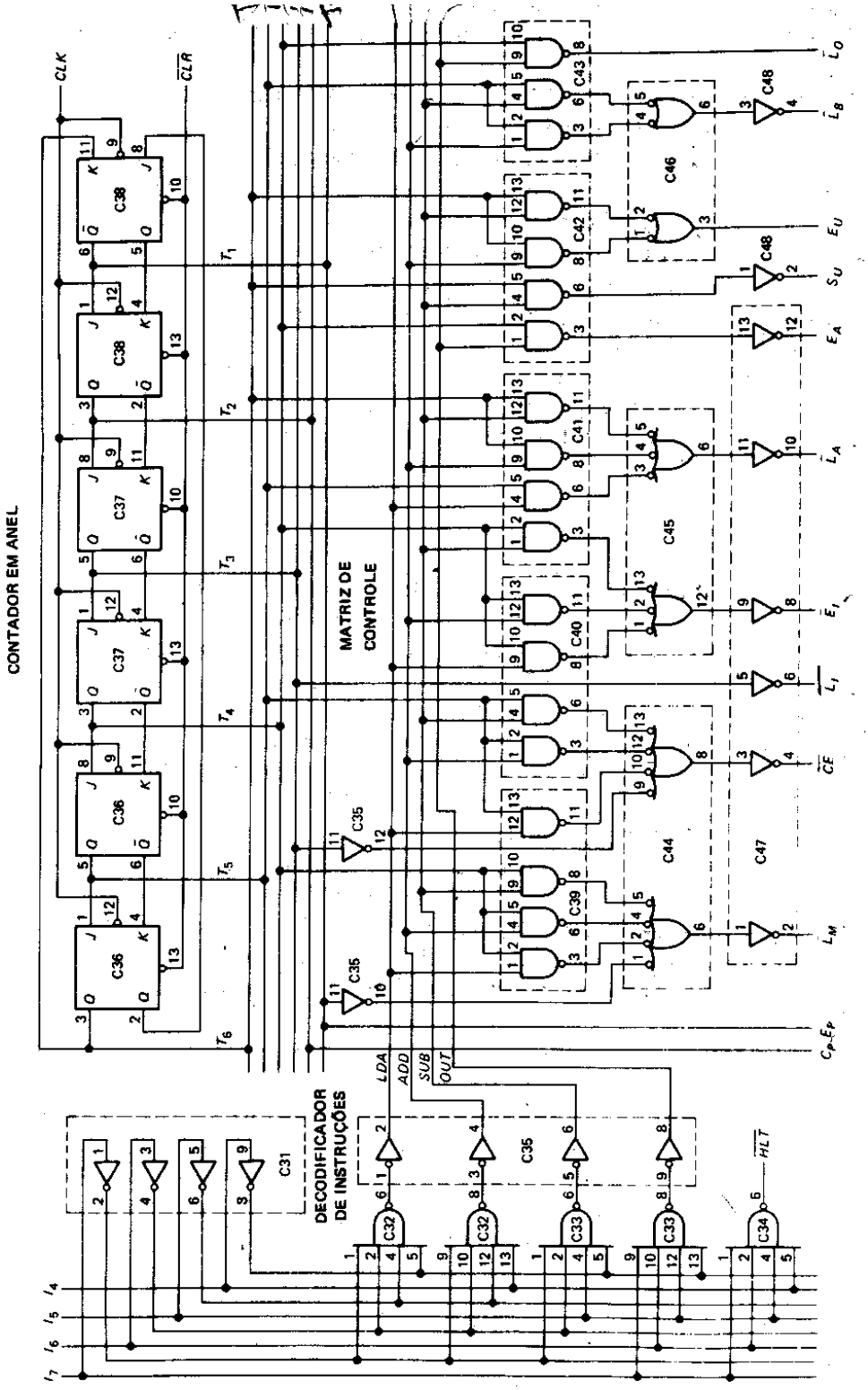

CONTADOR EM ANEL
MATRIZ DE CONTROLE
DECODIFICADOR DE INSTRUÇÕES
CLK
CLR
LDA
ADD
SUB
OUT
HLT

Fig 10-23

**10-6**. Suponhamos que um 8085 use uma frequência de relógio de 3 MHz. A instrução **ADD** de um 8085 leva quatro estados **T** para buscar e executar. Quanto tempo representa isto ?

**10-7**. Quais são as microinstruções SAP-1 para a rotina **LDA**? E para a sub-rotina **SUB**? Expressar as respostas em forma binária e em hexadecimal.

**10-8**. Suponhamos que queremos transferir o conteúdo do acumulador para o registrador B.Isto requer uma nova microinstrução. Qual é esta microinstrução? Expressar sua resposta em forma hexadecimal e em forma binária.

**10.9**. Observar a Fig. 10-20 e responder às seguintes perguntas:

a. Os conteúdos do contador de programas são modificados na transição negativa ou na transição positiva do sinal $\overline{CLK}$ ? Neste instante, o sinal CLK está em sua transição crescente ou decrescente?
b. Para incrementar o contador de programa, $C_P$ tem que ser baixo ou alto?
c. Para limpar (levar-a-0) o contador de programa, $\overline{CLK}$ tem que ser baixo ou alto?
d. Para colocar o conteúdo do contador de programa no barramento W, $E_P$ deve ser baixo ou alto?

**10-10**. Consulte a Fig. 10-21:

a. Se $\overline{L}_A$ for alto, o que acontecerá ao conteúdo do acumulador na próxima transição positiva de relógio?
b. Se A= 0010 1100 e B = 1100 1110,o que estará sobre o barramento W quando $E_A$ for alto?
c. Se A= 0000 1111,B = 0000 0001 e $S_U$ = 1, o que estará sobre o barramento W quanto $E_U$ for alto?

**10-11**. Responder às seguintes perguntas quanto à Fig. 10-22:

a. Com $S_5$ ,na posição **CLEAR**, a saída $\overline{CLK}$ é baixa ou alta?
b. Com $S_6$ na posição **LOW**, a saída é baixa ou alta no pino 11, **C24**?
c. Para ter um sinal de relógio no pino 3 de **C29**, $\overline{HLT}$ deve ser baixo ou alto?

**10-12**. Com referência à Fig. 10-23, responder ao seguinte:

a. Se $\mathbf{I_7I_6I_5I_4}$ = 1110, somente um dos pinos de saída em **C35** está alto. Que pino é este?
b. $\overline{CLR}$ torna-se alto. Qual é o sinal de temporização ($T_1$ a $T_6$) que se torna alto?
c. **LDA** e $\mathbf{T_5}$ estão altos. A tensão está baixa ou alta no pino 6, **C45**?
d. **ADD** e $T_4$ estão altos. O sinal no pino 12, **C45**, é baixo ou alto?