

NEMO-80

Versão 1.0

SISTEMA MONITOR PARA COMPUTADORES CEDM-80 V1 e
CEDM-80 V1 Mod

AVISO:

Esse documento e software são distribuídos SEM QUALQUER GARANTIA EXPRESSA OU IMPLICITA, INCLUÍDO COMERCIALIZAÇÃO, QUALIDADE SATISFATÓRIA OU FORMATAÇÃO PARA QUALQUER PROPÓSITO PARTICULAR.

Esse documento e software têm APENAS PROPÓSITO DIDÁTICO E O AUTOR NÃO PODERÁ SER RESPONSABILIZADO, EM QUAISQUER CIRCUNSTÂNCIAS, PELAS CONSEQUÊNCIAS DE SEUS USOS. SE V.Sa. OS USAR, SERÁ POR SUA CONTA E RISCO.

DISCLAIMER:

This documentation and software are distributed WITHOUT ANY EXPRESS OR IMPLIED WARRANTY, INCLUDING OF MERCHANTABILITY, SATISFACTORY QUALITY AND FITNESS FOR A PARTICULAR PURPOSE.

This document and software have ONLY DIDACTIC PURPOSES, AND THE AUTHOR CANNOT BE HELD LIABLE IN ANY CIRCUMSTANCES FOR THE CONSEQUENCES OF THEIR USAGE. IF YOU USE THEM, DO IT AT YOUR OWN RISK.

REVISÕES

Revisão	Data	Observações
A, B e C	Fev a 05/Maio/24	- Revisões iniciais. - Lançado ao público a partir da versão C
D	25/Maio/24	- Criada esta tabela de revisões - Corrigida tabela da sub-rotina “sys_disp_clear”(item 7.5) - Corrigida tabela de particionamento de memória (item 6.2)
E	26/Maio/24	- Completada a tabela com os códigos de teclas do item 7.4
F	08/Abril/2025	Correções de textos e erros de digitação

Sumário

1.	INTRODUÇÃO	6
2.	INSTALAÇÃO DO NEMO-80	6
3.	EQUIVALÊNCIA DE TECLAS CEDM-80 V1 E CEDM-80 V1 MOD	8
4.	OPERAÇÃO DO NEMO-80	9
4.1.	Visualização do conteúdo de dados na memória	9
4.2.	Alteração do conteúdo de dados na memória.....	10
4.3.	Visualização do conteúdo dos registradores da CPU	11
4.4.	Alteração do conteúdo dos registradores da CPU	11
4.5.	Executar um programa armazenado na memória	12
4.6.	Parar a execução de um programa	12
5.	EXEMPLO DE ESCRITA E EXECUÇÃO DE UM PROGRAMA	13
5.1.	Preparação do programa	13
5.2.	Escrita do programa no computador	14
5.3.	Execução e verificação do programa	16
5.4.	Programa padrão de teste do NEMO-80.....	18
6.	ESPECIFICAÇÕES DO SISTEMA.....	19
6.1.	Hardware necessário.....	19
6.2.	Particionamento de memória.....	19
7.	SUB-ROTINAS DO NEMO-80 VERSÃO 1.0 PARA O USUÁRIO (Mini BIOS)	20
7.1.	Nomenclatura de partes do hardware para as sub-rotinas	20
7.2.	Principais endereços de memória de sistema e nome de referência	20
7.3.	Tabela de códigos para exibição de caracteres no display	20
7.4.	Tabela de códigos das teclas	21
7.5.	Sub-rotina “sys_disp_clear” (CALL 0262H)	22
7.6.	Sub-rotina “sys_disp_data” (CALL 0272H)	22
7.7.	Sub-rotina “sys_disp_addr” (CALL 0281H).....	23
7.8.	Sub-rotina “sys_wait_keypress” (CALL 02B2H)	23
7.9.	Sub-rotina “sys_wait_keyrelease” (CALL 02BEH)	24
7.10.	Sub-rotina “sys_in_data” (CALL 02CAH).....	25
7.11.	Sub-rotina “sys_in_addr” (CALL 031AH).....	26

7.12.	Sub-rotina “sys_conv_hexdisp” (CALL 0389H)	27
7.13.	Sub-rotina “sys_sftl_addr_disp ” (CALL 039CH).....	27
7.14.	Sub-rotina “sys_sftl_data_disp” (CALL 03B6H)	28
7.15.	Sub-rotina “sys_keyb_disp” (CALL 03D0H)	29
7.16.	Sub-rotina “sys_delay_ms ” (CALL 0478H)	30
7.17.	Sub-rotina “menu_reg” (CALL 0045H)	30
8.	CÓDIGO-FONTE DO NEMO-80	31
8.1.	Ambiente de programação e compilação utilizado	31
8.2.	Código-fonte	31

1. INTRODUÇÃO

O software NEMO-80 (**NEw MO**nitor for **Z80**) é um software monitor desenvolvido para o computador didático CEDM-80 V1 ou a versão modificada CEDM-80 V1 Mod (de Thiago Turcato do Rego, referência: https://github.com/turcato1/CEDM-80_modificado).

O software monitor permite ao usuário interação básica com um computador para entrada de programas em sua memória, visualização/alteração do conteúdo dos registradores internos do processador, visualização/alteração do conteúdo posições de memória e colocação em execução de programas armazenados nas memórias RAM ou ROM.

Para a operação do NEMO-80 versão 1.0, são utilizados o teclado de 24 teclas e os 6 displays presentes nas variações do CEDM-80 V1.

Adicionalmente, o NEMO-80 foi desenvolvido de forma a permitir que as sub-rotinas utilizadas no software monitor, também possam ser chamadas (utilizadas) nos programas desenvolvidos pelo usuário. Sub-rotinas com funções como a limpeza do conteúdo dos displays, exibição de valor numérico no display e leitura do que é pressionado no teclado podem ser acessadas pelo usuário.

2. INSTALAÇÃO DO NEMO-80

Para instalação do NEMO-80, o usuário deve gravar o arquivo binário NEMO-80.bin em uma memória EPROM 27C16, para o caso do CEDM-80 V1, 27C512 ou EEPROM 28C64 (chip com encapsulamento DIP28) para o caso do CEDM-80 V1 Mod. A EPROM ou EEPROM deve ser instalada nos slots ilustrados nas Figuras a seguir:

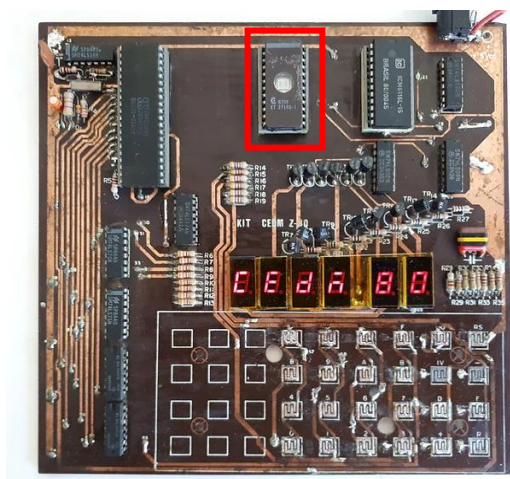


Figura 1: CEDM-80 e a posição onde a memória EPROM 27C16 com o NEMO-80 deve ser instalada (vermelho).

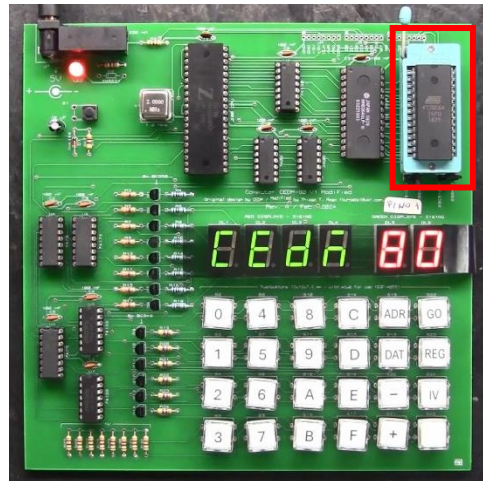


Figura 2: CEDM-80 V1 Mod e a posição onde a memória EPROM 27C512 ou EEPROM 28C64 com o NEMO-80 deve ser instalada (vermelho).

ATENÇÃO: No caso do CEDM-80 V1 Mod, os jumpers existentes na parte inferior da memória, conforme Figura 3, devem ser posicionados de acordo com o tipo de memória a ser utilizado. Não configurar corretamente esses jumpers, pode danificar a memória ou o computador.

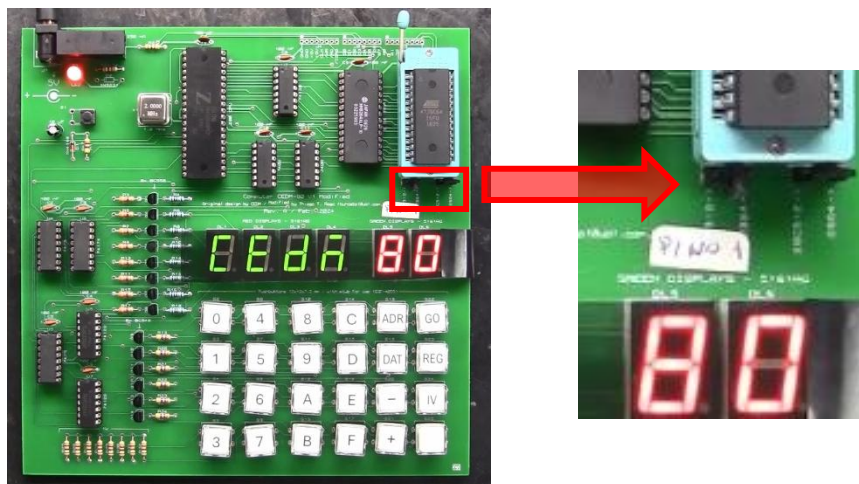


Figura 3: CEDM-80 V1 Mod e os jumpers de configuração do tipo de memória ROM utilizada.

A memória deve ser instalada, em qualquer caso, com o computador desligado e desenergizado.

3. EQUIVALÊNCIA DE TECLAS CEDM-80 V1 E CEDM-80 V1 MOD

Nesse manual serão utilizadas nas explicações as teclas do computador CEDM-80 V1 MOD.

Caso esteja utilizando o CEDM-80 V1, observe abaixo a equivalência de teclas de diferentes nomenclaturas para operação:

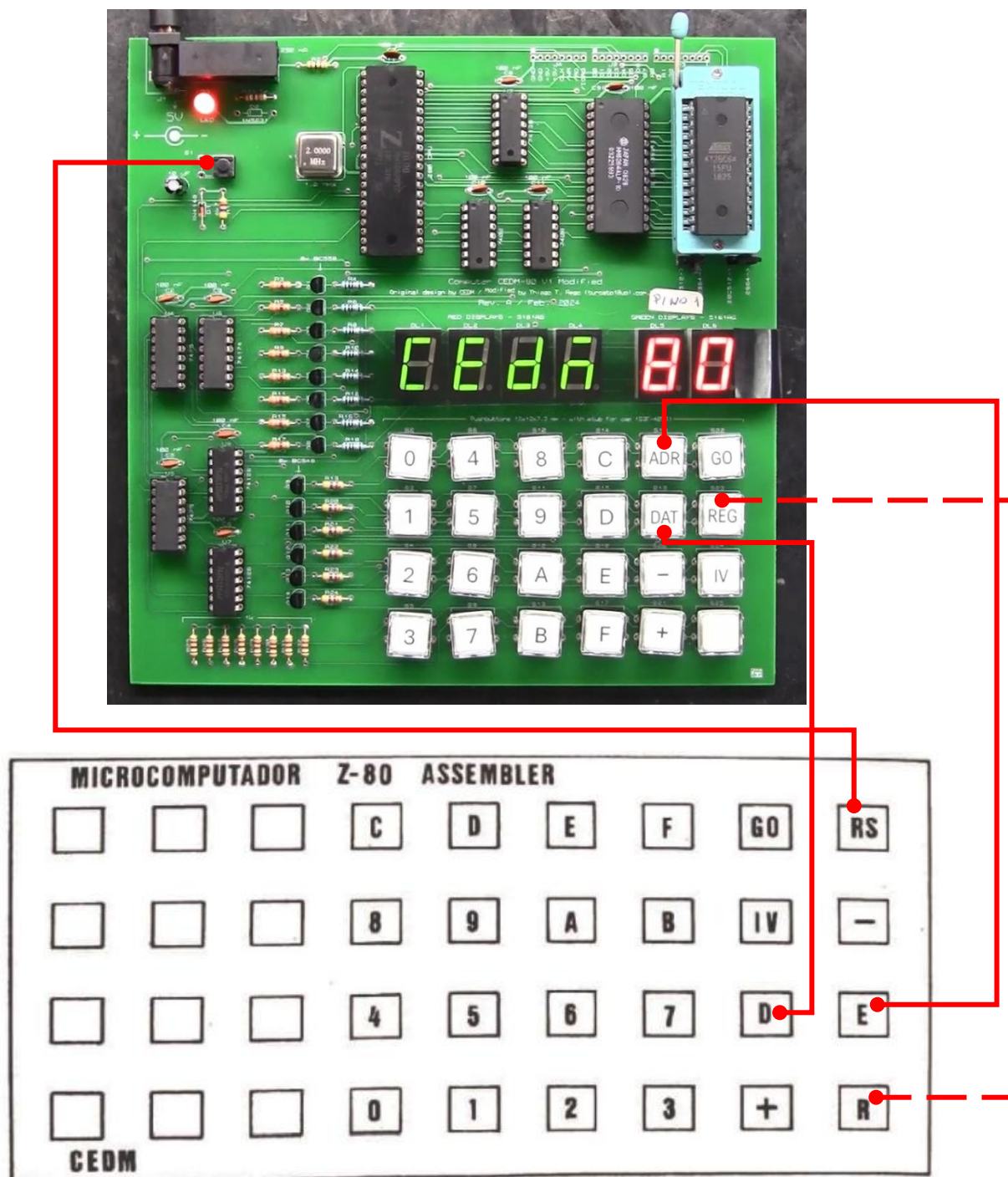


Figura 4: Equivalência de teclas diferentes entre CEDM-80 V1 e CEDM-80 V1 Mod.

4. OPERAÇÃO DO NEMO-80

Com o NEMO-80 já instalado, como explicado no capítulo 2, ligue o computador e a mensagem de boas-vindas “NEMO 80” deverá aparecer no display, conforme ilustra a Figura 5 (abaixo).

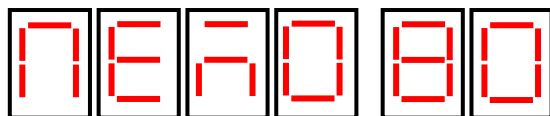


Figura 5: Mensagem de boas-vindas “NEMO 80” ao ligar o CEDM-80.

Neste capítulo, onde houver um texto entre aspas, é o nome da tecla. Exemplos: “REG” é a tecla de nome REG, “0” é a tecla 0 (zero).

4.1. Visualização do conteúdo de dados na memória

Para visualizar o conteúdo da memória siga o seguinte procedimento:

Passo	Pressionar/digitar	O que aparecerá no display
1	<div>ADR</div>	<div></div> <div></div> <div></div> <div></div> <div></div> <div></div>
2	(endereço desejado, usando o teclado numérico, ex.: endereço 2000H, digite: “2” “0” “0” “0”)	<div>2</div> <div>0</div> <div>0</div> <div>0</div> <div></div> <div></div>
3	<div>DAT</div>	<div>2</div> <div>0</div> <div>0</div> <div>0</div> <div>5</div> <div>A</div> <p>(valor nos displays à direita poderão variar de acordo com o conteúdo na memória)</p>

No passo 2, caso o endereço digitado seja errado, continue digitando o endereço correto pois o valor do display será deslocado a cada dígito pressionado para a esquerda. Ao aparecer o endereço correto dos 4 displays da esquerda, siga para o passo 3.

Para visualizar o conteúdo dos endereços seguintes de memória, pressione a tecla “+” repetidamente. Para visualizar endereços anteriores de memória, use o mesmo procedimento, porém com a tecla “-”.

4.2. Alteração do conteúdo de dados na memória

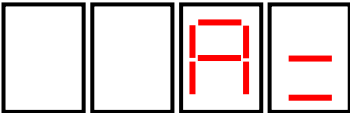
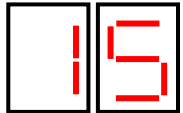
Para alterar o conteúdo de um endereço de memória, siga o seguinte procedimento:

Passo	Pressionar/digitar	O que aparecerá no display
1	<div>ADR</div>	<div></div> <div></div> <div></div> <div></div> <div></div> <div></div>
2	(endereço desejado, usando o teclado numérico, ex.: endereço 2000H, digite: “2” “0” “0” “0”)	<div>2</div> <div>0</div> <div>0</div> <div>0</div> <div></div> <div></div>
3	<div>DAT</div>	<div>2</div> <div>0</div> <div>0</div> <div>0</div> <div>S</div> <div>A</div> (valor nos displays à direita poderão variar de acordo com o conteúdo na memória)
4	(valor numérico que deseja escrever no conteúdo da memória para o endereço exibido, ex. valor 76H, digite: “7” “6”)	<div>2</div> <div>0</div> <div>0</div> <div>0</div> <div>7</div> <div>6</div>

Após o passo 4, o valor já estará escrito no endereço de memória indicado no display. Para continuar escrevendo em próximos endereços, pressione a tecla “+” e, sem a necessidade de pressionar DAT (ou D), digite o dado a ser escrito nesse endereço seguinte de memória.

4.3. Visualização do conteúdo dos registradores da CPU

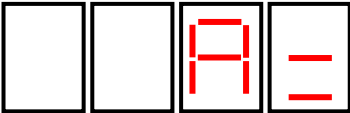
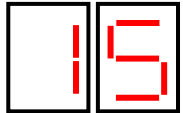
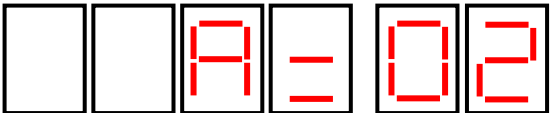
Para visualizar o conteúdo de um registrador, siga o seguinte procedimento:

Passo	Pressionar	O que aparecerá no display
1	REG	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> <p>Registrador</p>  </div> <div style="text-align: center;"> <p>Valor atual do registrador</p>  </div> </div> <p>(valor nos displays da direita poderão variar)</p>

Use as teclas “+” e “-” para visualizar o conteúdo de outros registradores. Os registradores exibidos são A, B, C, D, E, H, L, I, R e SP (*stack pointer* ou ponteiro de pilha).

4.4. Alteração do conteúdo dos registradores da CPU

Para alterar o conteúdo de um registrador, siga o seguinte procedimento:

Passo	Pressionar	O que aparecerá no display
1	REG	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> <p>Registrador</p>  </div> <div style="text-align: center;"> <p>Valor atual do registrador</p>  </div> </div> <p>(valor nos displays da direita poderão variar)</p>
2	(digite o valor numérico que deseja escrever no conteúdo do registrador exibido, ex. 02H, digite: “0” “2”)	

Após o passo 2, o valor será escrito no registrador ao sair do menu de registradores (por exemplo, ao pressionar a tecla ADR).

4.5. Executar um programa armazenado na memória

Para executar um programa armazenado na memória, é preciso saber primeiramente o endereço inicial onde está o programa.

De posse dessa informação, siga o procedimento abaixo:

Passo	Pressionar	O que aparecerá no display
1	<div>ADR</div>	<div></div> <div></div> <div></div> <div></div> <div></div> <div></div>
2	(endereço desejado, usando o teclado numérico, ex.: endereço 2000H, digite: “2”“0”“0”“0”)	<div>2</div> <div>0</div> <div>0</div> <div>0</div> <div></div> <div></div>
3	<div>GO</div>	(A exibição no display irá variar, de acordo com o programa executado)

4.6. Parar a execução de um programa

Para parar um programa em execução, pressione a tecla RS ou o reset, dependendo da variação de computador, conforme a Figura 6 (abaixo).

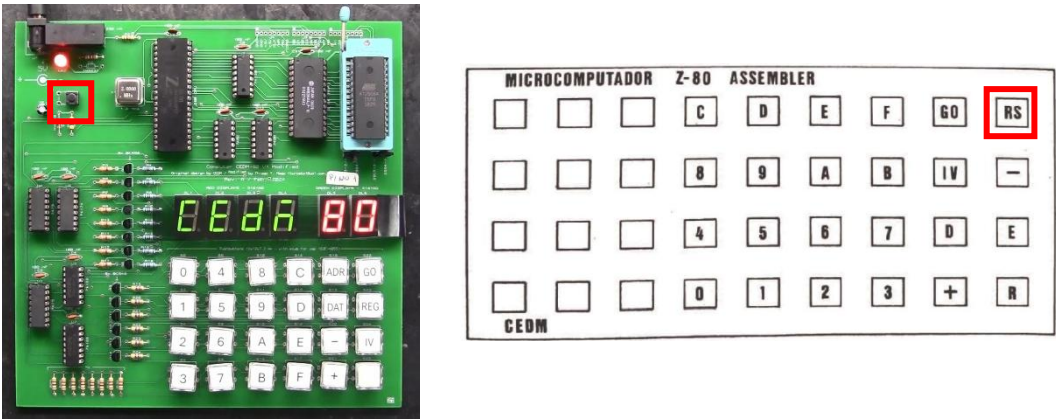


Figura 6: Reset no CEDM-80 V1 Mod (à esquerda) e CEDM-80 V1 (à direita), marcado em vermelho.

Após o reset, a execução voltará para o programa monitor e a mensagem “NEMO 80” aparecerá no display.

5. EXEMPLO DE ESCRITA E EXECUÇÃO DE UM PROGRAMA

5.1. Preparação do programa

Suponha que o programa abaixo vai ser escrito e executado no CEDM-80 (Mod).

```
LD A,(2100H)      ; Lê o valor contido no endereço 2100 hexa
LD B,04H          ; Armazena o valor 04 hexadecimal em B
ADD A,B           ; Soma o valor de A e B, resultado em A
LD (2101H),A      ; Armazena o resultado no endereço 2100 hexa
HALT              ; Pára a execução
```

Primeiramente é preciso compilar o programa manualmente, convertendo as instruções *assembly* em *opcodes* para que o programa possa ser escrito no computador pelo NEMO-80. Os *opcodes* podem ser encontrados no manual de usuário do processador Z80.

Assembly	Opcode (hexa)
LD A,(2100H)	3A 00 21
LD B,04H	06 04
ADD A,B	80
LD (2101H),A	32 01 21
HALT	76

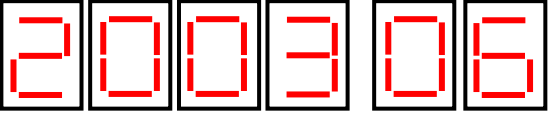

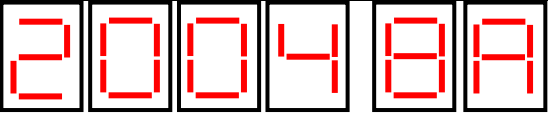
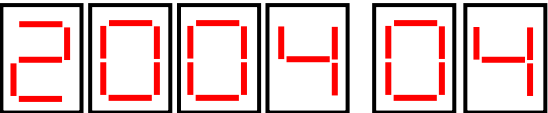

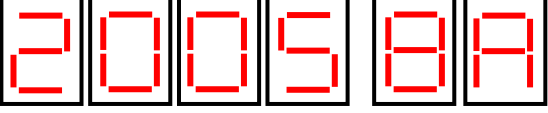
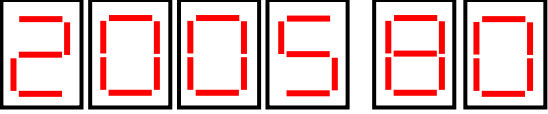

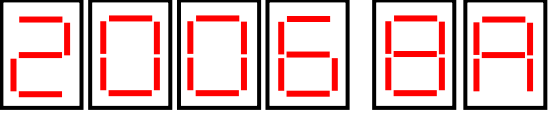
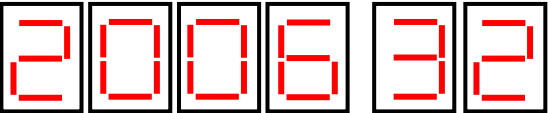

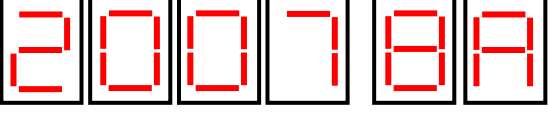
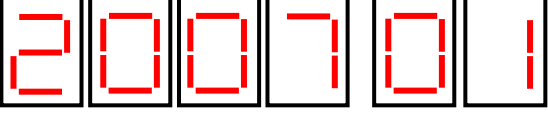

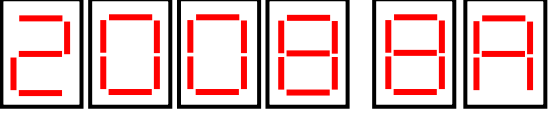
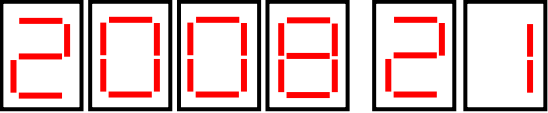
Determine o local de memória onde o programa será armazenado, alocado um byte por endereço de forma sequencial (a memória suporta 1 byte ou 8 bits de dados por endereço). A escrita de dados via NEMO-80 no CEDM-80 (Mod) é permitida apenas na memória RAM (endereços na faixa de 2000H a 25FFH).


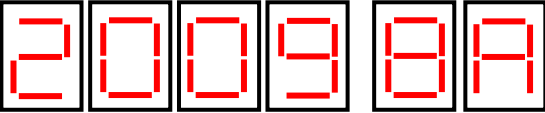
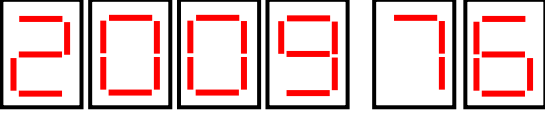
Assembly	Opcode (hexa)	Endereço de memória (hexa)
LD A,(2100H)	3A	2000
	00	2001
	21	2002
LD B,04H	06	2003
	04	2004
ADD A,B	80	2005
LD (2101H),A	32	2006
	01	2007
	21	2008
HALT	76	2009

5.2. Escrita do programa no computador

Agora, para escrever o programa no computador, siga o procedimento abaixo. A partir do passo 4, é a digitação/escrita dos *opcodes* do programa em si.

Passo	Pressionar/digitar	O que aparecerá no display
1	<div>ADR</div>	<div></div> <div></div> <div></div> <div></div> <div></div> <div></div>
2	"2" "0" "0" "0"	<div>2</div> <div>0</div> <div>0</div> <div>0</div> <div></div> <div></div>
3	<div>DAT</div>	<div>2</div> <div>0</div> <div>0</div> <div>0</div> <div>5</div> <div>A</div> (valor nos displays à direita poderão variar de acordo com o conteúdo na memória)
4	"3" "A"	<div>2</div> <div>0</div> <div>0</div> <div>0</div> <div>3</div> <div>A</div>
5	<div>+</div>	<div>2</div> <div>0</div> <div>0</div> <div>1</div> <div>8</div> <div>A</div> (valor nos displays à direita poderão variar de acordo com o conteúdo na memória)
6	"0" "0"	<div>2</div> <div>0</div> <div>0</div> <div>1</div> <div>0</div> <div>0</div>
7	<div>+</div>	<div>2</div> <div>0</div> <div>0</div> <div>2</div> <div>8</div> <div>A</div> (valor nos displays à direita poderão variar de acordo com o conteúdo na memória)
8	"2" "1"	<div>2</div> <div>0</div> <div>0</div> <div>2</div> <div>2</div> <div>1</div>
9	<div>+</div>	<div>2</div> <div>0</div> <div>0</div> <div>3</div> <div>8</div> <div>A</div> (valor nos displays à direita poderão variar de acordo com o conteúdo na memória)

10	“0” “6”	
11		 (valor nos displays à direita poderão variar de acordo com o conteúdo na memória)
12	“0” “4”	
13		 (valor nos displays à direita poderão variar de acordo com o conteúdo na memória)
14	“8” “0”	
15		 (valor nos displays à direita poderão variar de acordo com o conteúdo na memória)
16	“3” “2”	
17		 (valor nos displays à direita poderão variar de acordo com o conteúdo na memória)
18	“0” “1”	
19		 (valor nos displays à direita poderão variar de acordo com o conteúdo na memória)
20	“2” “1”	

21		 (valor nos displays à direita poderão variar de acordo com o conteúdo na memória)
22	"7" "6"	



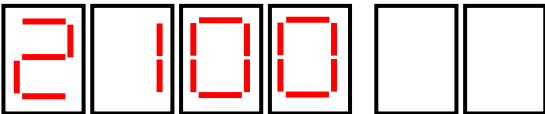

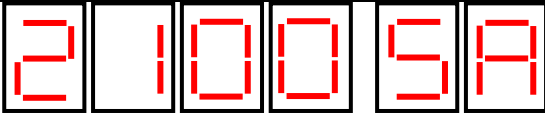
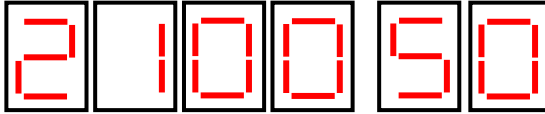
Agora, o programa já está na memória do computador, a partir do endereço 2000H.

5.3. Execução e verificação do programa



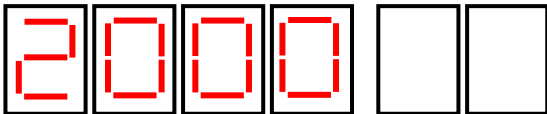


O programa de exemplo das seções anteriores realiza a soma de um valor armazenado no endereço de memória 2100H com 04H e o resultado é gravado no endereço de memória 2101H

Para testar a execução do programa, os passos abaixo instruem um exemplo da gravação do valor 50H no endereço 2100H, fazendo com que o resultado após a execução do programa seja 54H no endereço de memória 2101H.

Assim, primeiramente, é necessário gravar o valor 50H no endereço 2100H.

Passo	Pressionar/digitar	O que aparecerá no display
1		
2	"2" "1" "0" "0"	
3		 (valor nos displays à direita poderão variar de acordo com o conteúdo na memória)
4	"5" "0"	


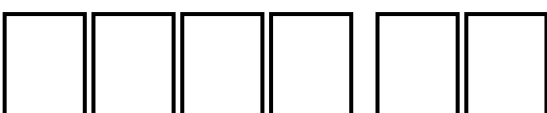
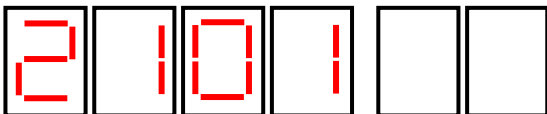

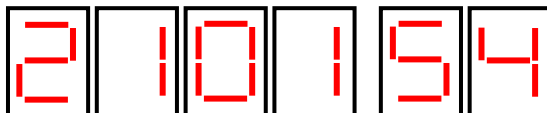
Agora, o programa deve ser posto em execução, entrando com o endereço de memória 2000H, que é o endereço do início do programa.

Passo	Pressionar	O que aparecerá no display
1		
2	(endereço desejado, usando o teclado numérico, ex.: endereço 2000H, digite: “2” “0” “0” “0”)	
3		

Nada deverá aparecer no display após a execução do programa. Agora, pressione RESET (vide seção 4.6, “Parar a execução de um programa”, caso não tenha localizado a tecla RESET) para parar a execução do programa e devolver o controle ao programa monitor.

ATENÇÃO: NÃO DESLIGUE O COMPUTADOR, POIS ISSO CAUSA A PERDA DO RESULTADO DA EXECUÇÃO DO PROGRAMA ARMAZENADO NA MEMÓRIA RAM!

Para saber se o programa foi executado corretamente, deve-se ler o valor armazenado no endereço de memória 2101H. O valor nessa posição de memória deverá ser 54H, caso o programa tenha sido executado corretamente. Para visualizar o conteúdo da memória em 2101H:

Passo	Pressionar/digitar	O que aparecerá no display
1		
2	“2” “1” “0” “1”	
3		

No passo 3, os displays da direita devem mostrar “54” se o programa foi executado corretamente.

5.4. Programa padrão de teste do NEMO-80

Há um programa padrão de teste dentro do NEMO-80, que pode ser utilizado para verificar o funcionamento do computador. Esse teste fará com que um número incrementado a cada 0,5 segundo, aproximadamente, apareça nos dois displays da direita. Esse programa, no NEMO-80 versão 1.0, está no alocado no endereço de memória 049CH.

Para executar o programa padrão de teste:

Passo	Pressionar	O que aparecerá no display
1	<div>ADR</div>	<div><div></div><div></div><div></div><div></div><div></div><div></div></div>
2	“0” “4” “9” “C”	<div><div>0</div><div>4</div><div>9</div><div>C</div><div></div><div></div></div>
3	<div>GO</div>	<div><div></div><div></div><div></div><div></div><div>0</div><div>0</div></div> <p>Após aprox. 0,5 segundo:</p> <div><div></div><div></div><div></div><div></div><div>0</div><div>1</div></div> <p>O valor se incrementa a cada 0,5 segundo de 00 até FF e, depois, retorna para 00, reiniciando o ciclo de contagem.</p>

6. ESPECIFICAÇÕES DO SISTEMA

6.1. Hardware necessário

Computador CEDM-80 V1 ou CEDM-80 V1 Mod, com, no mínimo, 527 bytes de memória RAM e 1536 bytes de memória ROM disponível para o NEMO-80.

6.2. Particionamento de memória

O NEMO-80 utiliza parte da memória RAM e ROM para sistema, com a seguinte organização. Em negrito, as áreas disponíveis ao usuário.

Endereço	Alocação		Tipo de memória
FFFFH	<div>Não usar (não alocado em hardware)</div>		<div>RAM</div> <div>2048 bytes</div>
2800H			
27FFH	<div>Stack (128 bytes)</div> <div>RAM para displays (16 bytes)</div> <div>RAM para teclado (3 bytes)</div> <div>Variáveis temporárias de sistema + reserva (365 bytes)</div> <div>Variáveis de execução do programa monitor (16 bytes)</div> <div>Reservado para o sistema NEMO-80 (528 bytes)</div>		
2780H			
277FH			
2770H			
276FH			
276DH			
276CH			
2600H			
25FFH	<div>RAM disponível para usuário (1520 bytes)</div>		
25F0H			
25EFH	<div>RAM disponível para usuário (1520 bytes)</div>		
2000H			
1FFFH	<div>Não usar (não alocado em hardware)</div>		
0800H			
07FFH	<div>ROM disponível para usuário (512 bytes)</div>		<div>ROM</div> <div>2048 bytes</div>
0600H			
05FFH	<div>Sistema NEMO-80 versão 1.0 (1536 bytes)</div>		
0000H			

7. SUB-ROTINAS DO NEMO-80 VERSÃO 1.0 PARA O USUÁRIO (Mini BIOS)

Neste capítulo são apresentadas as sub-rotinas disponíveis para o usuário chamar utilizando a instrução CALL do *assembly* do Z80. O objetivo é facilitar o acesso ao hardware, semelhante à função de uma BIOS em um computador de maior porte.

É importante que a chamada dessas sub-rotinas seja feita rigorosamente conforme as instruções para evitar o travamento de execução ou efeito indesejável/inesperado.

7.1. Nomenclatura de partes do hardware para as sub-rotinas

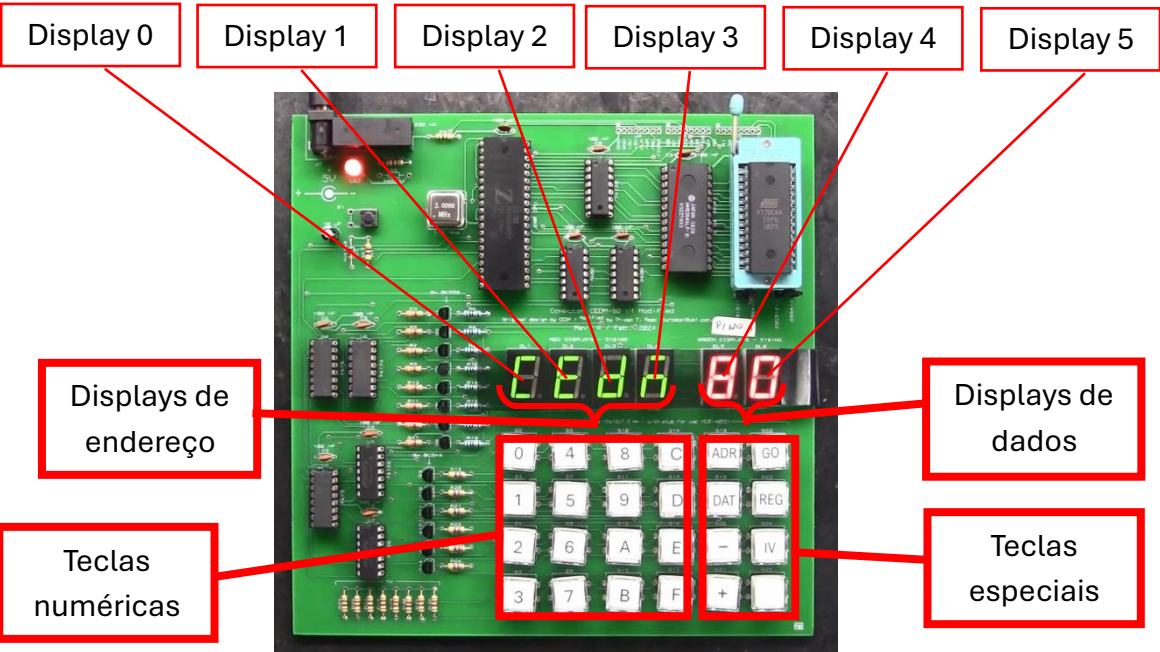


Figura 7: Nomenclatura das partes do hardware referenciadas na documentação das sub-rotinas.

7.2. Principais endereços de memória de sistema e nome de referência

Nome de referência	Endereço de memória	Descrição								
RAM_DISPLAY+0	2770H	Caractere a exibir no Display 0								
RAM_DISPLAY+1	2771H	Caractere a exibir no Display 1								
RAM_DISPLAY+2	2772H	Caractere a exibir no Display 2								
RAM_DISPLAY+3	2773H	Caractere a exibir no Display 3								
RAM_DISPLAY+4	2774H	Caractere a exibir no Display 4								
RAM_DISPLAY+5	2775H	Caractere a exibir no Display 5								
RAM_KEYB_CONV	276DH	<div>Bit7<div>Bit0</div><table><tr><td>P</td><td>X</td><td>X</td><td>C</td><td>C</td><td>C</td><td>C</td><td>C</td></tr></table><p>P: 1 = Há tecla pressionada; 0 = nenhuma tecla pressionada. C: Código da tecla (5 bits, 00H a 17H). X: Irrelevante (não usado).</p></div>	P	X	X	C	C	C	C	C
P	X	X	C	C	C	C	C			

7.3. Tabela de códigos para exibição de caracteres no display

Os códigos a seguir devem utilizados para gravação de caracteres na memória de display.

Caractere	Código
0	0C0H
1	0F9H
2	0A4H
3	0B0H
4	099H
5	092H
6	082H
7	0F8H
8	080H
9	090H
(apagado)	0FFH

Caractere	Código
A	088H
B	083H
C	0C6H
D	0A1H
E	086H
F	08EH
G	082H
H	089H
I	0F9H
J	0E1H
L	0C7H
M	0AAH
N	0C8H
O	0C0H
P	08CH
Q	098H
R	0AFH
S	092H
T	087H
U	0C1H
=	0B7H

7.4. Tabela de códigos das teclas

Tecla numérica	Código
0	00H
1	01H
2	02H
3	03H
4	04H
5	05H
6	06H
7	07H
8	08H
9	09H
A	0AH
B	0BH
C	0CH
D	0DH
E	0EH
F	0FH

Tecla especial	Código
ADR	10H
DAT	11H
-	12H
+	13H
GO	14H
REG	15H
IV	16H
(vazia)	17H

7.5. Sub-rotina “sys_disp_clear” (CALL 0262H)

Quando executada, essa sub-rotina limpa o conteúdo da RAM_DISPLAY+0 a +5, escrevendo valor 0FFH (display apagado) nessas posições de memória.

Variáveis de entrada:
Nenhuma
Variáveis de saída:
Nenhuma
Registradores afetados:
Pilha de memória utilizada

7.6. Sub-rotina “sys_disp_data” (CALL 0272H)

Quando executada, essa sub-rotina exibe nos displays de dados, o valor numérico em hexadecimal de 8 bits (1 byte, 2 dígitos hexadecimais) contido na posição de memória apontada pelo par de registradores HL. Deve ser chamada em loop para manter o valor exibido no display. Essa sub-rotina atualiza a variável de leitura de teclado RAM_KEYB_CONV.

Variáveis de entrada:
HL: Endereço de memória onde está o dado a ser exibido no display
Variáveis de saída:
(RAM_KEYB_CONV): Tecla apertada e seu código
Registradores afetados:
Registrador A Pilha de memória utilizada

Exemplo de uso:

```
LD HL,2100H
loop:
CALL 0272H
JR loop
```

O valor gravado no endereço de memória 2100H é exibido em formato hexadecimal nos displays de dados.

7.7. Sub-rotina “sys_disp_addr” (CALL 0281H)

Quando executada, essa sub-rotina exibe nos displays de endereço o valor numérico em hexadecimal de 16 bits (2 bytes, 4 dígitos hexadecimais) contidos nas posições de memória apontadas pelo par de registradores HL e HL+1. Deve ser chamada em loop para manter o valor exibido no display. Essa sub-rotina atualiza a variável de leitura de teclado RAM_KEYB_CONV.

Variáveis de entrada:
HL: Endereço de memória onde está o byte menos signif. a ser exibido no display HL +1: End. de memória onde está o byte mais signif. a ser exibido no display
Variáveis de saída:
(RAM_KEYB_CONV): Tecla apertada e seu código
Registradores afetados:
Registrador A Pilha de memória utilizada

Exemplo de uso:

```
LD HL, 2100H
loop:
CALL 0281H
JR loop
```

Os valores gravados nas posições de memória 2100H e 2101H são exibidos em formato hexadecimal nos displays de endereço.

7.8. Sub-rotina “sys_wait_keypress” (CALL 02B2H)

Quando executada, essa sub-rotina retém a execução do programa e, quando alguma tecla for pressionada, libera a continuação da execução a partir do próximo endereço de memória após a chamada da sub-rotina. Essa sub-rotina atualiza a variável de leitura de teclado RAM_KEYB_CONV.

Variáveis de entrada:
Nenhuma
Variáveis de saída:
(RAM_KEYB_CONV): Tecla apertada e seu código
Registradores afetados:
Registrador A Pilha de memória utilizada

Exemplo de uso:

```
LD A, 55H
CALL 02B2H
ADD A, B
```

Após a instrução LD A,55H ser executada, o processamento fica retido na sub-rotina chamada por CALL 02B2H, não avançando para o próximo passo enquanto uma tecla qualquer não for pressionada. Ao ser pressionada uma tecla, o processamento é liberado e a instrução seguinte, ADD A,B no exemplo acima, é executada.

7.9. Sub-rotina “sys_wait_keyrelease” (CALL 02BEH)

Quando executada, essa sub-rotina retém a execução do programa apenas se uma tecla estiver sendo pressionada e, quando a tecla pressionada for solta (não houver nenhuma tecla pressionada), libera a continuação da execução a partir do próximo endereço de memória após a chamada da sub-rotina. Essa sub-rotina atualiza a variável de leitura de teclado RAM_KEYB_CONV.

Variáveis de entrada:
Nenhuma
Variáveis de saída:
(RAM_KEYB_CONV): Tecla pressionada e seu código
Registradores afetados:
Registrador A
Pilha de memória utilizada

pulso_soma_um:

```
CALL 02BEH      ; sub-rotina “sys_wait_keyrelease”, Ponto A
CALL 02B2H      ; sub-rotina “sys_wait_keypress”, Ponto B
INC C
JR pulso_soma_um
```

No exemplo acima, o registrador C é incrementado de uma unidade, cada vez que uma tecla qualquer for pulsada. No “Ponto A” o processamento aguarda a tecla ser solta, no “Ponto B”, aguarda uma tecla ser pressionada e volta para o “Ponto A”, aguardando a tecla ser solta para um próximo pulso que repetirá a instrução de incremento INC C.

7.10. Sub-rotina “sys_in_data” (CALL 02CAH)

Essa sub-rotina é usada para receber um valor digitado pelo usuário em formato hexadecimal, exibindo o que está sendo digitado nos displays de dados (2 dígitos, 1 byte). O valor digitado é escrito no endereço de memória apontado pelo par de registradores HL.

Quando executada, essa sub-rotina limpa o display e fica em execução ciclicamente, retendo o processamento enquanto um valor é digitado. O processamento só é liberado quando uma das teclas especiais (teclas de código 10H a 17H, vide seções 7.1 e 7.4) é pressionada. Após isso, o valor exibido anteriormente nos displays de dados será escrito no endereço de memória apontada pelo par de registradores HL. Caso nada tenha sido digitado, o valor 00H será escrito. Caso apenas um número tenha sido digitado, o dígito mais significativo será considerado 0H e o valor será escrito na memória.

Essa sub-rotina atualiza a variável de leitura de teclado RAM_KEYB_CONV.

Variáveis de entrada:
HL: Endereço de memória que receberá o valor a ser digitado
Variáveis de saída:
(HL): Valor digitado na execução da sub-rotina (RAM_KEYB_CONV): Última tecla apertada e seu código
Registradores afetados:
Registrador A Pilha de memória utilizada

Exemplo de uso:

```
LD HL,2300H
CALL 02CAH
ADD A,B
```

Ao chamar a sub-rotina com CALL 02CAH, o processamento fica retido nessa sub-rotina aguardando a digitação do valor. O valor digitado será escrito no endereço de memória 2300H. Ao se pressionar uma tecla especial qualquer, o processamento segue para a instrução ADD A,B.

7.11. Sub-rotina “sys_in_addr” (CALL 031AH)

Essa sub-rotina é usada para receber um valor digitado em formato hexadecimal pelo usuário, exibindo o que está sendo digitado nos displays de endereço (4 dígitos, 2 bytes). O valor digitado é escrito nas posições de memória apontadas pelo par de registradores HL e HL +1.

Quando executada, essa sub-rotina limpa o display e fica em execução ciclicamente, retendo o processamento para aguardar o valor digitado. O processamento só é liberado quando uma das teclas especiais (teclas de código 10H a 17H, vide seções 7.1 e 7.4) é pressionada. Após isso, o valor exibido anteriormente nos displays de dados será escrito no endereço de memória apontada pelo par de registradores HL e a consecutiva (HL +1). Caso nada tenha sido digitado, o valor 0000H será escrito. Caso menos do que 4 números tenham sido digitados, os dígitos mais significativos serão considerados 0H e o valor será escrito na memória.

Essa sub-rotina atualiza a variável de leitura de teclado RAM_KEYB_CONV.

Variáveis de entrada:
HL: Endereço de memória que receberá o valor do byte menos signif. digitado HL +1: End. de memória que receberá o valor do byte mais signif. digitado
Variáveis de saída:
(HL): Valor do byte menos significativo digitado na execução da sub-rotina (HL +1): Valor do byte mais significativo digitado na execução da sub-rotina (RAM_KEYB_CONV): Última tecla apertada e seu código
Registradores afetados:
Registrador A Pilha de memória utilizada

Exemplo de uso:

```
LD HL,2300H
CALL 031AH
ADD A,B
```

Ao chamar a sub-rotina com CALL 031AH, o processamento fica retido nessa sub-rotina aguardando a digitação do valor. O valor digitado será escrito nos endereços de memória 2300H e 2301H. Ao se pressionar uma tecla especial qualquer, o processamento segue para a instrução ADD A,B.

7.12. Sub-rotina “sys_conv_hexdisp” (CALL 0389H)

Essa sub-rotina é usada para converter um dígito hexadecimal no código do caractere correspondente para exibição no display, ou seja, converter um valor como 0FH no caractere “F” a ser exibido no display.

Variáveis de entrada:
A: Valor de um dígito hexadecimal de 00H a 0FH
Variáveis de saída:
C: Código de caractere para exibição no display, correspondente ao valor hexadecimal dado pela variável de entrada, registrador A.
Registradores afetados:
Pilha de memória utilizada

Exemplo de uso:

```
LD A,0EH
CALL 0389H
LD (2770H),C      ; 2770H = End. de memória do Display 0
```

Ao chamar a sub-rotina com CALL 0389H, o valor carregado no registrador A, 0EH é convertido para o valor 86H, correspondente à exibição do caractere “E” no display de 7 segmentos. Esse valor convertido é armazenado no registrador C. No exemplo acima, esse valor convertido é escrito no endereço de memória 2770H, para que o caractere “E” possa ser exibido no Display 0. Para que o conteúdo da memória de display seja efetivamente exibido, é necessário chamar a função de varredura do display e teclado, “sys_keyb_disp”.

7.13. Sub-rotina “sys_sftl_addr_disp” (CALL 039CH)

Essa sub-rotina é usada para deslocar o valor exibido nos displays de endereço um dígito à esquerda. O valor contido no registrador C é escrito na posição aberta no display mais à direita (Display 3), com o deslocamento do valor dos displays de endereço.

Variáveis de entrada:
C: Valor do código de caractere a ser inserido na posição direita
Variáveis de saída:
Nenhuma
Registradores afetados:
(RAM_DISPLAY+0) à (RAM_DISPLAY+3)
Pilha de memória utilizada

Exemplo de uso:

```
LD C,0E1H    ; Código de caractere para letra "J"  
CALL 039CH
```

Ao chamar a sub-rotina com CALL 039CH, o valor de exibição nos displays armazenado em RAM_DISPLAY+0 a RAM_DISPLAY+3 é deslocado para a esquerda e o espaço aberto em RAM_DISPLAY+3 é preenchido com o código do caractere "J".

7.14. Sub-rotina "sys_sftl_data_disp" (CALL 03B6H)

Essa sub-rotina é usada para deslocar o valor exibido nos displays de dados um dígito à esquerda. O valor contido no registrador C é escrito na posição aberta no display mais à direita (Display 5), com o deslocamento do valor dos displays de endereço.

Variáveis de entrada:
C: Valor do código de caractere a ser inserido na posição direita
Variáveis de saída:
Nenhuma
Registradores afetados:
(RAM_DISPLAY+4) e (RAM_DISPLAY+5) Pilha de memória utilizada

Exemplo de uso:

```
LD C,0B0H    ; Código de caractere para letra "3"  
CALL 03B6H
```

Ao chamar a sub-rotina com CALL 03B6H, o valor de exibição nos displays armazenado em RAM_DISPLAY+4 e RAM_DISPLAY+5 é deslocado para a esquerda e o espaço aberto em RAM_DISPLAY+5 é preenchido com o código do caractere "3".

7.15. Sub-rotina “sys_keyb_disp” (CALL 03D0H)

Essa sub-rotina é usada para realizar a varredura de acionamento de todos os displays (0 a 5) e de leitura do teclado. O valor escrito nos displays durante a varredura tem origem na memória reservada de sistema RAM_DISPLAY+0 a +5 e a memória reservada de sistema RAM_KEYB_CONV recebe o código da última tecla pressionada (bits 0 a 4) e o status se há ou não uma tecla pressionada (bit 7 de RAM_KEYB_CONV).

Variáveis de entrada:
(RAM_DISPLAY+0) a (RAM_DISPLAY+5): Caracteres a serem exibidos nos displays
Variáveis de saída:
(RAM_KEYB_CONV): Código e status de tecla pressionada
Registradores afetados:
Registrador A
Pilha de memória utilizada

Exemplo de uso:

```
LD A,08CH          ; Caractere B
LD (RAM_DISPLAY+0),A
LD A,0F9H          ; Caractere I
LD (RAM_DISPLAY+1),A
LD A,087H          ; Caractere T
LD (RAM_DISPLAY+2),A
varre_display:
CALL 03D0H
JR varre_display
```

Ao chamar a sub-rotina com CALL 03D0H ciclicamente, a palavra “BIT” é exibida no display.

7.16. Sub-rotina “sys_delay_ms ” (CALL 0478H)

Essa sub-rotina é usada para realizar um atraso operativo na execução (delay) pelo tempo aproximado de milissegundos dado pelo valor inteiro sem sinal de 8 bits armazenado no registrador B.

Variáveis de entrada:
B: tempo de atraso aproximado em milissegundos (valor inteiro sem sinal de 8 bits)
Variáveis de saída:
Nenhuma
Registradores afetados:
Pilha de memória utilizada

Exemplo de uso:

inicio:

```
LD A,08CH          ; Caractere B
LD (RAM_DISPLAY+0),A
LD A,0F9H          ; Caractere I
LD (RAM_DISPLAY+1),A
LD A,087H          ; Caractere T
LD (RAM_DISPLAY+2),A
LD B,100
```

varre_display:

```
CALL 03D0H
DJNZ varre_display
LD B,150
CALL 0478H
JR inicio
```

A palavra “BIT” pisca ficando 150 ms apagada e um tempo determinado por B, no ciclo de execução, aceso.

7.17. Sub-rotina “menu_reg” (CALL 0045H)

Essa sub-rotina exibe o menu de visualização e alteração de registradores.

Variáveis de entrada:
A, B, C, D, E, F, H, L, I, R, SP
Variáveis de saída:
Nenhuma
Registradores afetados:
Pilha de memória utilizada

8. CÓDIGO-FONTE DO NEMO-80

8.1. Ambiente de programação e compilação utilizado

O código do NEMO-80 foi desenvolvido utilizando-se o editor Visual Studio Code® da Microsoft® versão 1.88.1, com as seguintes extensões instaladas:

- Z80 Instruction Set v1.2.3 by Maziac
- Z80 Macro-Assembler v0.7.10 by Mboric
- ASM Code Lens v2.6.0 by Maziac
- Simulador DeZog v3.3.2 by Maziac
- Hex Hover Converter v1.2.2 by Maziac
- Sjasmplus v1.20.3 by Sjoerd Mastijn

8.2. Código-fonte

O código-fonte está disponível em:

https://github.com/turcato1/CEDM-80_modificado

(Último acesso em 25/05/2024).