Assembler
Disassembler
Compiler
Monitor

BY ZX SOFTWARE

Índice

Introdução

No cassette que acompanha este manual, lado A, estão gravados os programas:

1. Check Load
2. Assembler
3. Disassembler
4. Compiler
5. Monitor

Considerando que voce já deve estar familiarizado com os procedimentos de LOAD, passamos a descrever os programas, iniciando com o Check Load, um teste da fita.

Check Load

Como os programas Assembler, Disassembler, Compiler e Monitor possuem cada um mais de tres minutos de gravação, incluimos,no início da fita, este pequeno programa denominado Check Load, com aproximadamente trinta segundos, destinado a testar a eficiência .de carga da fita fornecida com o seu gravador/micro.

Primeiramente, veja, ou melhor, escute se a gravação produz sons agudos, ou mesmo estridentes. Não ? Produz somente sons graves e abafados ? Então o "azimute", a posição da cabeça do gravador em relação a fita de seu cassette não corresponde ao mesmo do aparelho em que a fita foi gravada. São comuns variações de cassette para cassette, sendo perfeitamente tolerável com músicas, mas muitas vezes é o principal responsável pelas impossibilidades de LOAD.

O cabeçote de todos os cassettes é preso por dois parafusos, sendo que um destes, juntamente com o cabeçote, está montado sobre uma mola.Apertando-se ou soltando-se este parafuso,altera-se a posição do cabeçote em relação à fita.

Caso tudo esteja OK, carregue o programa por LOAD "" e aparecerá:

CHECK LOAD OK

Após algum segundos o programa aciona um LOAD "". Desta forma, se o Check Load produziu a mensagem OK, não é necessário mexer no micro, para que se carregue o segundo programa, o Assembler.

Assembler

DESCRIÇÃO

O ZX ASSEMBLER é uma rotina destinada a facilitar a edição de programas no Assembly Z80. Ocupa menos de 7K de memória no Ram Top e compõem-se de tres partes principais. A saber:

1. EDITOR

   O EDITOR permite a edição de programas diretamente no Assembly do Z80. Possui cursor próprio, permite inserir/apagar linhas/caracteres, auto repetição de teclas, identificação de labels, strings, etc....

2. ASSEMBLER

   O ASSEMBLER converte os mnemônicos nos códigos, interpreta os labels, números em hex ou decimal, além de textos.

3. MONITOR

   O MONITOR oferece facilidades para testar, alterar e rodar seus programas em código, além de sub-rotinas utilitárias como PRINT, Keyboard Input, etc...

CARREGANDO O PROGRAMA

Carregue-o a partir do CHECK LOAD ou use LOAD "" . O tempo de carga é de aproximadamente 3 minutos. Se a operação de carga tiver sucesso, será mostrado no vídeo

<<<ZX ASSEMBLER>>>
BY ZX SOFTWARE

Obrigatoriamente digite NEW. Para rodar o programa use:

RAND USR 30000 ou, RAND USR 3E4

Novamente será mostrado

<<< ZX ASSEMBLER >>>
BY ZX SOFTWARE

Porém, desta vez, no topo da tela. Nesta circunstância, para retornar ao Basic,digite **Q** duas vezes. Como vamos iniciar experimentando o EDITOR, digite **E**.

O EDITOR

O editor possui um cursor próprio, dado por um espaço inverso.Quando este aparecer, voce poderá digitar a sua rotina no Assembly Z80, usando diretamente os mnemônicos, como por exemplo:

```
LD A,20
PUSH BC
PUSH DE
PUSH HL
CALL 7EAA
POP HL
POP DE
etc...
```

A única forma de abandonar o EDITOR é digitando SHIFT Q. As facilidades de edição oferecidas podem ser acessadas por:

SHIFT **6**	move o cursor uma linha abaixo.
SHIFT **7**	move o cursor uma linha acima.
SHIFT **8**	move o cursor uma posição a direita
SHIFT **5**	move o cursor uma posição a esquerda
SHIFT **0**	apaga o último caractere
SHIFT **9**	para inserir um caractere
SHIFT **A**	para inserir um LABEL. Move o cursor até o extremo esquerdo da tela. Somente quando o cursor estiver na 7ª posição (início da linha) e não houver LABEL.

SHIFT **D**	apaga a linha em edição
SHIFT **E**	insere uma linha
SHIFT **G**	imprime o texto a partir do cursor, na impressora.
SHIFT **Q**	repete a linha em edição, quando , por exemplo, após a inserção de algum erro.
SHIFT **S**	procura uma STRING a partir da posição do cursor. SHIFT A para procurar um LABEL. Se não for encontrado,o cursor volta a posição original.
SHIFT **T**	coloca o cursor no topo do texto.

Todas as teclas possuem uma AUTO REPETIÇÃO após pressionadas por mais de um segundo.Uma linha se inicia com o cursor na sétima posição, reservando as primeiras para os LABELs. Para mudar de linha usa-se NEW LINE (ENTER).

LABELs

As rotinas podem ser identificadas por nomes formados de letras ou números, desde que começados por uma letra e seguidos de um sinal de igual, se a seguir vier uma constante numérica. Podem estar diretamente seguidos dos mnemônicos ou completados por espaços. Por exemplo:

```
INICIO LD BC,0000       ; apenas letras como LABEL
MEM 1  PUSH BC          ; LABEL com número-
MEM 2 = 9900            ; LABEL para constante numérica
etc...
```

COMENTÁRIOS

Após os mnemônicos, colocando-se um ponto e virgula ; como no exemplo acima, pode-se inserir comentários que serão ignorados pela rotina de ASSEMBLER.

NÚMEROS

Os números podem ser entrados na forma :

DECIMAL se precedidos de um sinal de + ou de menos. O valor máximo admissível é equivalente a 16 bits.

HEXADECIMAL de Ø a FFFF.

Números fora de faixa ou tentativa de carregar register com valores fora de faixa param o ASSEMBLER com código de erro.

TEXTOS

Pode-se entrar diretamente textos, desde que colocados entre aspas. Para imprimi-lo podemos chamar a sub-rotina PRINT TEXTO, se o mesmo for terminado pelo código FF. Por exemplo.

```
CALL 7E99          ; CALL PRINT TEXTO
"TESTE DE SPRINT"FF
RET                ; retorne
```

OS MNEMÔNICOS PADRÃO E AS EXCEÇÕES

Os mnemônicos devem ser digitados como se escrevem. Isto é, LD A,Ø possui um espaço entre o D e o A, assim como várias outras instruções. Quanto ao término da linha, pode-se digitar apenas NEW LINE, ou completá-la com espaços. Mais de 31 caracteres em uma linha interrompe o programa.

O ASSEMBLER interpreta todos os mnemônicos no Z80, em sua forma padrão, exluindo-se os abaixo:

EX AF,AF'	use	EX AF
IM 0	use	IM0
IM 1	use	IM1
IM 2	use	IM2
JR,+N	use	JR+N

Todas as instruções JR e JP devem ser escritas sem virgulas.

O ASSEMBLER

Desde que voce já tenha digitado um programa em Assembly, saia do EDITOR digitando Q. Para acionar o ASSEMBLER digite A e será mostrado ASSEMBLE. Digite o endereço onde os códigos deverão ser colocados, na forma hexadecimal de quatro dígitos, ou apenas digite ENter/New Line para colocá-los apartir do endereço 4082 hex,16514 em decimal. É o endereço da primeira linha de programa em Basic.EM caso de erro é dada a mensagem. Digite E .

As possibilidades de erro são:

LABEL

Label indefinido ou com definição errada. Por exemplo : iniciando com número, linha com definição de Label removida por engano,etc..

NÚMEROS

Utilização de números fora da faixa permitida, inclusive quando em intruções como LOAD B,560 .

ESGOTAMENTO DE MEMÓRIA

Pode ocorrer no Editor com uma linha longa demais ou,no Assembler, quando não houver espaço para armazenar os códigos produzidos.

JUMP RELATIVE

Se o JUMP indicado estiver fora da faixa. -128 a +127.

SINTAXE

Todos os demais erros, inclusive os de escrita dos mnemônicos.

STACK

Se muito espaço da memória foi ocupado, ou o mesmo que, deixar pouco espaço para o STACK.

COMPENSAÇÃO

Quando carregar um register com um LABEL somado ou subtraído em mais que 9 ou nos registers IX, IY maior que + 127 ou menor que - 128. Por exemplo. LD BC,(TEST+13). O máximo seria LOAD BC,(TEST+9).

O MONITOR

Esta rotina apresenta facilidades para voce testar, corrigir e rodar os seus programas em código. Para acioná-la, com o programa na situação original, digite M. Todos os comandos oferecidos por esta rotina tornam-se disponíveis com o acionamento de uma única tecla, como os demais já citados. Os comando são.

C copia um bloco de memória de uma posição para outra. COPY BLOCK:
FIRST ADDRESS primeiro endereço a ser copiado.
LAST ADDRESS último endereço a ser copiado
TO ADDRESS endereço de destino

M modo de Edição direta na memória. Fornece uma coluna com 24 endereços de 2 bytes e o conteúdo de cada um deles. O cursor no meio da coluna indica o endereço que pode ser alterado. Digite o novo valor em hex, ou apenas New Line. Neste modo ainda existem os comandos:

J calcula o deslocamento de JUMP RELATIVO. Entre os dois últimos dígitos do endereço de destino Digite New Line e o deslocamento será dado pelo endereço do cursor.

New Line move a listagem uma linha.

L significa List, desce o cursor continuamente.

O move o cursor continuamente para cima.

P move o cursor para cima em uma linha.

R repete a entrada de um valor. Entre o valor e digite New Line. O endereço para o qual o cursor se deslocar pode receber este valor. Podem ser usadas as facilidades de deslocamento do cursor. Para cancelar a repetição digite Q.

I inspeção e modificação de registers. Apresenta o conteúdo dos registers AF, BC, DE e HL. O register F é também apresentado em binário,com as marcações das bandeiras S=sign, Z=zero, H=half-carry, O=parity/overflow, N=negative ou minus e C=carry. O cursor move-se por New Line.

S search for procura um número de 2 bytes.
from address a partir do endereço dado.
procura em toda a memória, exceto no STACK. Fornece o endereço de onde encontrou, ou indica"NOT FOUND",não encontrado. Digite Q para retornar ao modo de comando.

R roda a rotina em código que esteja no endereço 16514, 4082h ou no endereço dado. Não é necessário terminar as suas rotinas por RETURN. o Assembler sempre termina com JUMP para o monitor.

SUB-ROTINAS (EM CÓDIGO) "PRONTAS PARA USAR"

7F50	Scroll up.
7F6A	Scroll down.
7E99	Imprime o texto que estiver a seguir de CALL 7E99 e for terminado pelo código FF.
7E7E	Fornece o valor do register A, na forma hexadecimal.
7AEE	Fornece o endereço de uma linha em Basic. Antes de aplicar CALL 7AEE, coloque o número da linha em Basic no register BC. O endereço voltará em HL.
7B76	Apaga uma linha do programa em Basic, cujo número esteja no register BC.
7F8A	New Line.
7A3C	Endereço de retorno para o Assembler.

Disassembler

DESCRIÇÃO

Este programa lê códigos de máquina no assembly do Z80. Fornece os endereços em decimais, com os códigos em hexadecimais seguidos dos mnemônicos completos. Possui rotina para impressora e interpreta todas as instruções do Z80.

O PROGRAMA

Carregue-o por LOAD "". Aparecerá no vídeo:

<<< ZX DISASSEMBLER >>>
BY ZX SOFTWARE

STARTING ADDRESS?

Starting address?,é o endereço de início da rotina a se produzir o disassembly. Informe 0 (zero)e voce terá o disassembly da ROM a partir do endereço zero. Para se "disassemblar" as demais rotinas da ROM é preciso conhecer os seus endereços de início.

A tela é preenchida com os endereços em decimal, os códigos em hex e os mnemônicos completos com o uso de scroll, até completar vinte e duas linhas. Neste ponto voce terá os comandos

R	RUN ou recomeça. Volta a starting address?
C	CONTINUE. Continua o disassembly.
P	PRINT ou COPY. Copia a tela para a impressora.

ATENÇÃO: NUNCA use RUN. Use GOTO 22 (em caso de interrupção).

Compiler

DESCRIÇÃO:

Este programa transforma os programas em Basic em programas no código da máquina, com o principal objetivo de rodá-los até 50 vezes mais rápido. Para simplificar a construção e utilização deste programa, bem como para não torná-lo muito longo, foi preciso limitar certos recursos, ficando as seguintes restrições:

1. Só opera com números inteiros na faixa de - 32768 a + 32767 .
2. Não aceita operações lógicas e com Strings.

COMO COMPILAR:

Inicialmente carregue o COMPILER por LOAD "" . Se carregado com sucesso aparecerá no vídeo:

<<<ZX COMPILER>>>
BY ZX SOFTWARE

Escreva o seu programa em Basic, a ser "compilado", sem contudo apagar as linhas de números 1 a 5, as quais contém o programa COMPILER. As demais, de números 6 a 25, devem ser apagadas.É obrigatório. Observe as restrições da página 14 , referentes aos diferentes comandos do Basic Sinclair.

Para compilar, coloque como última linha do programa em Basic, um comando STOP e, a seguir, digite:

SLOW e New Line e,
LET L=USR 17389

A compilação é executada duas vezes, apresentando simultaneamente a listagem em Basic. Para obter um SCROLL da listagem e continuar a compilação, mantenha a tecla 0 pressionada. Ao final aparecerá u

ma interrogação invertida ? e, a seguir 0/0 indicando o término.Caso apareça algum S invertido S , houve erro.

Os erros mais comuns são:

1. Erro no programa em Basic, indicado por um S inverso, próximo ao mesmo.
2. Inexistência de STOP ao final do programa em Basic.
3. Uso de número fracionário ou E.
4. Nome de variável com mais de uma letra ou matriz com nome diferente de Z .
5. Uso de INKEY$ sem CODE.
6. Uso de operações aritméticas sem LET, exceto quando com parenteses.
7. Instrução ilegal ou não interpretável pelo COMPILER.

COMO RODAR OS PROGRAMAS COMPILADOS

Para rodar um programa já compilado, digite:

```
LET L=USR 18823
```

COMO COMPILAR PROGRAMAS CARREGADOS DO CASSETTE

Para compilar programas já escritos e arquivados em cassette, use o seguinte procedimento:

1. Carregue o COMPILER e, a seguir digite NEW.
2. Carregue o programa,em Basic, a ser compilado.
3. Digite LET L=USR 32577 para recuperar o COMPILER do RamTop.
4. Utilize o COMPILER como descrito, observando as restrições da página 11

RESTRIÇÕES AOS COMANDOS DO BASIC SINCLAIR

ABS - permitido com o uso de LET.

CHR$ - permitido com o uso de PRINT.

DIM Z(X) - somente aceita uma matriz denominada Z .

CLS - normal

COPY - use USR 2153

FAST - normal

FOR/NEXT - somente com incremento de 1 em 1.

GOSUB - chama a linha indicada e demais como uma sub-rotina.

GOTO - desvio incondicional do programa para a linha de nº indicado.

IF/THEN - aceita as condicionais < > , = , = > , < =, < ou > . a seguir de THEN aceita qualquer comando interpretável pelo COMPILER.

INKEY$ - somente com o uso de CODE e LET.

INT - veja LET

LET - aceita as operações (/ * + -), além de
PEEK X
USR X
RND - gera um número pseudo-randomico entre Ø e 32767.
ABS X
INT X
CODE INKEY$

NEW - use USR Ø

PAUSE X - causa uma pausa de X/50 segundos, sem aquela "piscada" no vídeo, caracteristica da PAUSE em Basic, mesmo quando em SLOW. Máximo valor de X: 32768 .

PEEK X - uso com LET

PLOT X,Y - normal

POKE X,Y - normal

PRINT - aceita:
PRINT "string"
PRINT AT
PRINT variável
PRINT CHR$

RAND - veja RND, LET

REM - normal

RETURN - normal

RND - veja LET

SCROLL - normal, exceto que apresenta um deslocamento uniforme, devido a velocidade superior a 2 linhas por segundo.

SLOW - normal

STOP - obrigatóriamente só deve ser usado uma vez, ao final do programa.

UNPLOT - normal

USR X - veja LET

OBSERVAÇÕES

Antes de recompilar um programa corrigido, apague a linha 2.

Nos programas compilados, o código de erro do Basic Sinclair 5/,de vídeo esgotado, foi substituído por uma interrogação invertida. Quando esta é apresentada, voce tem a sua disposição os seguintes comandos:

1. CONT - executa um CLS e continua o programa. Mantendo CONT pressionado haverá uma auto-repetição do mesmo.
2. D - executa um SCROLL a velocidade máxima de 2 linhas/segundo.
3. Z - executa a COPY, do vídeo na impressora.
4. QUALQUER TECLA PRESSIONADA - executa SCROLL na velocidade máxima.
5. BREAK - para o programa

Monitor

DESCRIÇÃO

Este programa, em código, é um utilitário e destina-se ao estudo de outros programas em código. Permite listar, entrar, testar e alterar rotinas em código na forma hexadecimal. Possui também um conversor de hex para decimal e vice-versa.

CARREGANDO O PROGRAMA

Use LOAD "" e aparecerá no vídeo:

<<< ZX MONITOR >>>
BY ZX SOFTWARE

Após alguns segundos o vídeo se apaga voltando ao cursor K. Coloque como uma linha de programa

1 LET L=USR 30848

Use RUN para rodar o MONITOR. Voce estará então no modo COMANDO.

OS COMANDOS DO MONITOR

A xxxx n Altera memória

Este comando permite alterar qualquer byte da RAM, no endereço informado em hex (4 digitos), para qualquer valor em hex (2 digitos). No final de uma página usa-se New Line para voltar ao modo COMANDO ou X para continuar.

B xxxx n Breakpoint

Permite inserir uma ordem de interrupção em qualquer ponto da rotina em código. Um contador memoriza quantas vezes foi executada a instrução e somente executa a interrupção após o número de vezes informado.

Os comandos G e C, respectivamente GOTO e CONTINUE zeram o contador. O uso de um Breakpoint ocupa 3 bytes da memória devido ao uso de uma instrução CALL. Por esta razão não se deve usar dois Breakpoints a menos que estejam distantes entre si mais de tres bytes. As instruções destes tres bytes serão normalmente executadas se não ocorrer a interrupção. Para interromper na 1ª execução da instrução do endereço 16516 use: B 4084 00

C Continua a execução após uma interrupção.

C xxxx n

Este comando recarrega os registers AF, BC, DE, HL. Rearmazena o Stack em 32 bytes da memória e continua a execução do programa a partir do último Breakpoint. O contador do mesmo é zerado.

D Decimal para Hex - conversão

Converter qualquer valor decimal entre 00000 e 65535 em seu equivalente em hex. Números acima de 65535 provocam a interrupção com a mensagem de sobrecarga - "overflow".

G xxxx GOTO

GOTO ao endereço indicado em hex, 4 dígitos.

H Hex para Decimal - conversão

Converte qualquer valor em hex de 0000 a FFFF em seu equivalente decimal.

N Negate Breakpoint

Elimina o último Breakpoint e recoloca as instruções dos últimos tres bytes ocupados pelo Breakpoint. Para cada Breakpoint eliminado é mostrado o seu endereço.

P xxxx n

Este comando lista uma página da memória. Para continuar a listar digite qualquer tecla, exceto New Line, que é usado para voltar ao modo COMANDO.

R Lista os registers.

As bandeiras são apresentadas na forma hexadecimal e binária.Para se alterar os registers usa-se A (Altera) nos endereços 7E17 hex até 7E1E hex, onde eles estão armazenados.

S Stop

Este comando retorna ao Basic.