ASSEMBLER Z80

Este programa tem por objetivo compilar um programa fonte escrito em Assembler (do processador Z80) para "código de máquina".
Este manual não pretende ensiná-lo a programar em linguagem Assembler, mas sim descrever os princípios básicos de operação do programa ASSEMBLER Z80. Caso você queira aprender a programar Assembler, recomendamos o livro "INTRODUÇÃO A LINGUAGEM DE MÁQUINA PARA O TK - ASSEMBLY Z80" de autoria de Flavio Rossini, em dois volumes, das Editôras MICROMEGA e MODERNA.
Para utilizar este programa a primeira providência é carregá-lo, da maneira usual. O programa entrará com o código de erro 1/0. Não se preocupe, esta é a sua maneira normal de carga.
Ao ser carregado, o programa altera automaticamente o RTP e ocupa os 5K de memória imediatamente acima.
Para entrar um programa em Assembly você deve digitar as instruções em linhas REM, respeitando o padrão de mnemônicos, ZILOG Z80, exceto as vírgulas que devem ser substituidas por pontos. É permitido mais de uma instrução por linha, desde que estejam separadas por ":" (ponto e vírgula).
Com o objetivo de delimitar a área de montagem a primeira linha REM de seu programa fonte deve conter um "(" (abre parenteses), e a última deve conter um ")" (fecha parenteses).
Você pode utilizar-se de até 256 labels (pontos de referência), de L0 a L255, os quais serão identificados pela presença de ":" (dois pontos), antecedendo-os. Comentários podem ser inseridos à vontade, desde que sejam precedidos por um "*" (asterisco).
Os valores numéricos serão assumidos como decimais a menos que precedidos por "$" (cifrão), quando serão interpretados como hexadecimais.
Não utilize em seu programa fonte, números de linha 900 ou superiores, pois estes são reservados para o programa ASSEMBLER.

EXEMPLOS DE PROGRAMA

1 - Esta subrotina, quando executada em FAST, produzirá um atraso de 0,1 a 25,6 segundos conforme o valor carregado no registrador "B", na linha 20.

```
10 REM (
20 REM LD B.10 ;LD DE.$FFFF
30 REM:L1LD HL.14130;:L2 ADD HL.DE
40 REM JR C.L2;DJNZ.L1;RET
50 REM )
```

2 - Executando esta subrotina obteremos o SCROLL no sentido inverso ao SCROLL executado pelo BASIC.

```
10 REM (
20 REM LD HL.($400 C);PUSH HL;LD BC.$2D6
30 REM ADD HL.BC;EX DE.HL;LD BC.693
40 REM POP HL;ADD HL.BC
50 REM LDDR;LD B.$20 ;:L10 INC HL
60 REM LD (HL).0; DJNZ.L10; RET
70 REM )
```

Após ter introduzido o seu programa em Assembler, o passo seguinte é traduzí-lo em código de máquina e inseri-lo na área de memória desejada. Para que isto seja feito, você deve indicar ao programa o local onde ele deverá colocar o código gerado. Na maioria das vezes isto é feito em uma linha REM no início de seu programa. Portanto introduza uma linha do tipo:

```
1 REM XXXXXX ... XXXXX
```

onde a quantidade de caracteres "X" seja suficiente para conter o código de máquina correspondente ao seu programa fonte.
Feito isto, digite GOTO 900 e, ao receber o cursor, introduza o valor 16514 (primeiro caractere "X" de sua linha REM), ou qualquer outro que você tenha escolhido para o seu programa.
Então a tradução e montagem do código de máquina terá início.
A cada instrução processada pelo programa ASSEMBLER Z80, você terá no vídeo uma mensagem do tipo:

```
100   40 8A   76   HALT
```

Isto significa que uma instrução HALT está na linha número 100 de seu programa fonte, que o código de máquina para HALT é 76 (HEXA) e que este código foi armazenado no endereço de memória 40 8A (HEXA).
Este programa é um Assembler de dois passos, portanto, durante sua execução, a listagem será apresentada duas vezes.
Quando a tela estiver cheia, pressione qualquer tecla para continuar ou BREAK para interromper a montagem. Ao terminar a montagem lhe será mostrado, no canto inferior esquerdo, um código de êrro com os seguintes significados:

0 — Nenhum erro
1 — Não existe "("
2 — Não existe ")"
3 — Label ilegal
4 — Instrução ilegal
5 — Número fora de faixa
6 — Jump relativo fora de faixa

Após terminada uma montagem sem erros, sugerimos que seja feito um "SAVE" do programa fonte junto com o código de máquina, pois é comum, utilizando-se linguagem de máquina, ocorrerem situações em que não é possível a recuperação do controle do computador

OS MNEMÔNICOS

Várias abreviações são utilizadas na relação de mnemônicos:

-r	é um registrado qualquer
-dis	é um byte na faixa de - 128 a 127
-dado	qualquer dado de um ou dois bytes
-mem	qualquer posição de memória ou também (HL), (IY + dis), (IX + dis).

As primeiras três letras da maioria das instruções são muito importantes, e você deve digitá-las corretamente especialmente os espaços

· ADC A,mem	ADC A,r	ADC A, dado	ADC. HL,BC
· ADC HL,DE	ADC HL,HL	ADC HL,SP	ADD A,mem
· ADD A,r	ADD A,dado	ADD HL,BC	ADD HL,DE
ADD HL,HL	ADD HL,SP	ADD IX,BC	ADD IX,DE
· ADD IX,IX	ADD IX,SP	ADD IY,BC	ADD IX,DE
· ADD IY,IY	ADD IY,SP	AND mem	AND r
· AND dado	BIT 0,mem	BIT 0,r	BIT 1,mem
· BIT 1,r	BIT 2,mem	BIT 2,r	BIT 3,mem
BIT 3,r	BIT 4,mem	BIT 4,r	BIT 5,mem
· BIT 5,r	BIT 6,mem	BIT 6,r	BIT 7,mem
· BIT 7,r	CALL endereço	CALLC. endereço	CALLM. endereço
· CALLNC. endereço	CALLNZ. endereço	CALLP. endereço	CALLPE. endereço
· CALLPO. endereço	CALLZ. endereço	CCF	CP mem
CP r	CP dado	CPD	CPDR
· CPI	CPIR	CPL	DAA
DEC mem	DEC r	DEC BC	DEC DE
· DEC HL	DEC IX	DEC IY	DEC SP
· DI	DJNZ,dis	EI	EX (SP),HL
· EX (SP),IX	EX (SP),IY	EX AF,AF	EX DE,HL
· EXX	HALT	IMO	IMI
IM2	IN r,(C)	IN A,porta	INC mem
· INC r	INC BC	INC DE	INC HL
· INC IX	INC IY	INC SP	IND
· INDR	INI	INIR	JP (HL)
· JP (IX)	JP (IY)	JP endereço	JP C,endereço
· JP M, endereço	JP NC, endereço	JP NZ, endereço	JP P, endereço
· JP PE, endereço	JP PO, endereço	JP Z, endereço	JR dis
· JR C,dis	JR NC,dis	JR NZ,dis	JR Z,dis
· LD (endereço),A	LD (endereço),BC	LD (endereço),DE	LD (endereço),HL
· LD (endereço),IX	LD (endereço),IY	LD (endereço),SP	LD (BC),A
· LD (DE),A	LD mem,r	LD mem,dado	LD A, (endereço)
· LD A,(BC)	LD A,(DE)	LD r,r	LD r,dado
· LD r,mem	LD A,I	LD A,R	LD BC, (endereço)
· LD BC,dado	LD DE, (endereço)	LD DE,dado	LD HL, (endereço)
· LD HL,dado	LD I,A	LD IX, (endereço)	LD IX,dado
· LD IY, (endereço)	LD IY,dado	LD R,A	LD SP, (endereço)
LD SP,dado	LD SP,HL	LD SP,IX	LD SP,IY
LDD	LDDR	LDI	LDIR
· NEG	NOP	OR mem	OR r
· OR dado	OTDR	OTIR	OUT (C) ,r
· OUT porta,A	OUTD	OUTI	POP AF
· POP BC	POP DE	POP HL	POP IX
· POP IY	PUSH AF	PUSH BC	PUSH DE
· PUSH HL	PUSH IX	PUSH IY	RES 0,mem
· RES 0,r	RES 1,mem	RES 1,r	RES 2,mem
· RES 2,r	RES 3,mem	RES 3,r	RES 4,mem
· RES 4,r	RES 5,mem	RES 5,r	RES 6,mem
· RES 6,r	RES 7,mem	RES 7,r	RET
· RET C	RET M	RET NC	RET NZ
· RET P	RET PE	RET PO	RET Z
· RETI	RETN	RL mem	RL r
RLA	RLC mem	RLC r	RLCA
· RLD	RR mem	RR r	RRA
· RRC mem	RRC r	RRCA	RRD
· RST 00	RST 08	RST 10	RST 18
· RST 20	RST 28	RST 30	RST 38
· SBC A,mem	SBC A,r	SBC A,dado	SBC HL,BC
· SBC HL,DE	SBC HL,HL	SBC HL,SP	SCF
· SET 0,mem	SET 0,r	SET 1,mem	SET 1,r
· SET 2,mem	SET 2,r	SET 3,mem	SET 3,r
· SET 4,mem	SET 4,r	SET 5,mem	SET 5,r
· SET 6,mem	SET 6,r	SET 7,mem	SET 7,r
· SLA mem	SLA r	SRA mem	SRA r
· SRL mem	SRL r	SUB mem	SUB r
· SUB dado	XOR mem	XOR r	XOR dado