

BETÖLTÉSE:

Az assembler/editor gépi kódú program, betöltése a SYSTEM utasítással történik, file neve EDI, belépési pontja 17152. Így betöltéskor a π ? promtra EDI választ π ? kell adni, betöltés után π ? /17152-vel indítható és újraindítható. Ezután a ?> INPUT prompt jelenik meg.

INPUT:

A parancsokat a promtra válaszként kell megadni. Először a parancs betűjét, majd a címet/címeket kell megadni. A címek decimálisak, hexadecimálisak /végén H/ vagy oktálisak lehetnek/ végén O/. Ha több paraméter szükséges, akkor mindegyikre külön prompt jelenik meg. A gépelési hibák visszalépéssel /balra nyíl/ javíthatók. A promptra NEW LINE választ adva az assembler/editor vissza-tér az Input parancshoz.

POINTER PARANCSONK:

Az assembler/editor sor-orientált, miképp maga a Z 80 assembler is. A forrásfile aktuális sorára belső pointer mutat.

- T - top - a pointer a file elejére megy
- B - botton - a pointer a file végére megy
- Dn - down - a pointer n sorral lejjebb megy /default n=1/
- Un - up - a pointer n sorral feljebb megy /default n=1/
- Fn - print - a képernyőre n sort listáz /default n=1/, a pointer ezután az utolsó listázott sorra mutat
- L - locate - a forrásfileben megkeres egy karaktersringet, a pointert a megtalált stringre állítja, ha nem talál ilyen stringet, a pointer a file végére áll

ÁLTALÁNOS PARANCSONK:

- H - how big - kiírja a forrásfile start és end címeit
- K - kill - törli a forrásprogramot a memóriából
- S - sort - az előző assemblálás során keletkezett szimbólumtáblát rendezi és kiírja, a parancs után egy betűt írva csak a megadott betűvel kezdődő szimbólumokat írja ki
opció? V=video/képernyő, E=external/printer
- A -assembler - a startól az end pszeudó utasításig terjedő forrásprogramot lefordítja
opció? V=video/képernyő, E=external/printer, C=cassette /object program/
6 esetén újabb kérdés: address? file név?
default a leggyorsabb mód, amíg a program hibás

KAZETTA PARANCSONK:

A forrás- és object programok save-elhetők és load-olhatók, object program esetén a parancs után O betűt kell írni.

- W - write - file-t ír a kazettára, forrás módban a memóriában lévő forrásfile-t írja ki, object módban prompt: address?, mindkét módban prompt: file név?
- R - read - kazettáról file-t olvas, forrásmódban a memóriában lévő kurrens file után tölt /egyszerű merge/, object módban a file-t címek szerint tölti be, mindkét módban prompt: file név?
load-olás közben "L" látható, hibás load után "E", hibátlan load után a file méret jelenik meg
- V - valid - ellenőrzi, hogy a kiírt file hibátlan-e /ugyanaz, mint a read, de nem tölt a memóriába/

AZ ASSEMBLER:

A forrásprogram utasításai Z 80 assembler utasítások. Minden programsor legfeljebb négy mezőt tartalmaz:

CIMKE; OPERÁTOR OPERANDUS; MEGJEGYZÉS

CIMKE:

Az utasításokat címkével lehet megjelölni. Ha olyan operátort használunk, mint JP vagy CALL, akkor operandusként meg kell jelölni a célt /pl. hogy hova kell ugrani/. Az assembler megengedi szimbolikus nevek címkékként történő alakalmazását.

Szimbolikus nevek kétféleképpen adhatók meg: az utasítás elejére írva /az assembler a programszámláló értékét rendeli hozzá/, és EQUATE PSZEUDO utasítással /ezzel tetszés szerinti értéket lehet a szimbólumokhoz rendelni, pl. BACKSPACE: EQU 8/.

A szimbólum első karaktere csak betű lehet, egyébként tartalmazhat számokat hossza legfeljebb 7 karakter. Operátor nevek, regiszter nevek és feltétel kód nevek nem lehetnek szimbólumok.

OPERÁTOROK:

Az assemblernek a Z 80 utasításkészlet 74 operátorán kívül néhány pszeudo utasítása is van, mégpedig:

- END - end assembly, operandus nincs
ezzel kell lezárni a forrásprogramot, különben hiba üzenetet kapunk
- DS - define storage, egy operandus
megadott számú object byte-ot üresen hagy, pl. szöveg buffer, stack, stb. számára helyfoglalás
- DW - define word, egy operandus
az object file-ban egy kétbyte-os szót generál, a byte-okat fordított sorrendben, ahogy az a Z 80 utasítások számára szükséges, pl. BUFFER: DW VALUE
- DB - define byte s, több operandus
az operandus értékét az object file-ba generálja, több operandus lehet vesszőkkel elválasztva, pl.: DB 6, 93H, 'I', 80H
az operandusok kifejezések is lehetnek, a program számláló minden operandus után növekszik, mintha azok külön sorban lennének, ASCII stringek is lehetnek az operandusok, pl.: MSG: DB 'JO REGGELT'
- EQU - equate, egy operandus
szimbólumhoz értéket rendel, pl. NEW LINE: EQU 12
- ORG - orgin, egy operandus
az object fileban kezdő helyét határozza meg, a programban tetszés szerint többször alkalmazható a kódrészek különböző kezdőpontokra helyezéshez
- LOAD - load memory, egy operandus
object kód memóriába betöltése az előállításkor megadott címre, ez a load teljesen független az assembler indításkor megadott output opciótól, a load-olás a következő ORG-ig tart

OPERANDUSOK:

Az operandusok száma nulla /pl. NOP/, egy /pl. CP/, kettő /pl. BIT/, és egy vagy kettő /pl. JR/ lehet. Az operandusok lehetnek regiszter nevek /pl. A, B, DE, IX, .../ és feltétel nevek /pl. Z, NZ, C, M, .../, ASCII literálok /'vagy " köze zárt karakterek/, számjegyek /számjeggyel kezdődő decimális, hexadecimális, oktális értékek/, programszámláló /futáskor PC-t módosítja/.

Az előző adattípusokból matematikai műveletekkel kifejezések alkothatók, amelyek mindenütt használhatók számok helyett.

MEGJEGYZÉS:

Megjegyzések az assemblerben bárhol elhelyezhetők, a megjegyzés vége a sor vége.

HIBAKEZELÉS:

Ha az assembler hibát talál a forrásprogramban, megáll, hibaüzenetet ad, a hibás sort kiírja, a pointert erre a sorra állítja.

DOUBLE SYMBOL	- többször deklarált szimbólum
RSVD	- szimbólum neve lefoglalt szó /reserved word/
FULL	- szimbólum tábla betelt
ORG	- nincs kezdőpont /ORG utasítás/
OPND	- operandus hiba
UNDEF	- hiányzó szimbólum
EOF	- hiányzó END
HUH?	- sületlenség /syntax error/

ASSEMBLER LISTA:

Képernyőre vagy printere készülhet. A lista oldalanként készül, rövid szünetekkel. A szünet ideje alatt bármely billentyűt megnyomva az oldal bármely következő billentyű megnyomásáig a képernyőn marad. Az oldalméret képernyőn 15, printeren 16 sor.

EDITOR PARANCSONK:

Zn - ZAP - a pointer helyétől kezdve n sort töröl /default n=1/
 E - ENTER - beszúrás a pointer helye elé, sorszámokat ad, beszúrás vége csak "." karaktert tartalmazó sor
 N - NEW - a pointer sorát új sorra cseréli
 A sorszámok csupán az editálást segítik, a filenak nem részei.

DEBUG /programbelövő/ PARANCSONK:

G - GOTO - vezérlés átadás a felhasználói programnak, prompt? belépési cím? breakpoint cím? /default az aktuális programszámláló/, breakpoint default nincs, ha van, akkor egy cím felhasználásával a byte megőrződik és egy RST 38H utasítás kerül besúrára
 C - COPY - kezdőcímtől végcímig terjedő memória blokk áthelyezése cél-címre
 F - FILL - kezdőcímtől a végcímig feltölti a memóriát egy 'data' konstanssal
 X - EXAMINE - a felhasználói gépi kód státuszát írja ki, felül a fő regiszterekkel, a mások sorban az alternatív regiszterekkel
 M - MODIFY - a memóriát tartalmát vizsgálja és módosítja, a megadott címet és tartalmát kiírja, a tartalom átírható, majd a következő címre ugrik, kilépés "."-tal
 Q - QUERY - a memóriából 8 byte-ot kiír a következő formában: cím, hexadecimális, ASCII a 7. bit nélkül, ASCII a 7. bittel /a harmadik mező megjeleníti azokat a karaktereket, amelyekben a 7. bit 1, a negyedik mező ezeket grafikus karakterenként mutatja/, a kontroll karakterek "."-tá alakítva jelennek meg. Q a 0000 címtől Qnnn az nnnn címtől kezd a kiírást, majd Q onnan folytatja tovább.

Z 80 ASSEMBLER UTASÍTÁSOK

Rövid áttekintés

A központi egység 1 byte-os regiszterei: A,B,C,D,E,H,L,I,R . Ezek közül:

- A - akkumlátor
- I - interrupt vektor 2. byte-ja
- R - felfrissítő regiszter

a többi általános regiszter, Dupla regiszterei: AF,BC,DE,HL,IX,IY,IV,SP,PC. Ezek közül

- PC - program számláló
- SP - stack pointer
- IV - interrupt vektor
- IX,IY - indexregiszterek
- AF - flageket tartalmazó regiszter / A reg.+flagek/

AF-fel csak egyetlen műveletet lehet végezni. Ezenkívül munkaregiszterei is vannak, mégpedik F',A',B',C',D',E',H',L'.

1 byte-os információk:

- a regiszterek tartalma
- 1 byte-os adat: dd
- egy cím tartalma: (cím)
- dupla regiszterben tárolt cím tartalma: (HL) , (DE) , (BC) ,

2 byte-os információk:

- a dupla regiszterek tartalma
- 2 byte-os adatok: dddd

LOAD /adatátvitel/:

Típusai: regiszterből regiszterbe, mindkét helyen állhat A,B,C,D,E,H,L,I,R, de az I és R csak az A-val együtt lehet

dupla regiszterből dupla regiszterbe: csak SP-be a HL,IX,IY párok-ból

regiszterbe adat: A,B,C,D,E,H,L regiszterbe 1 byte-os adat, BC, DE,HL,IX,IY,SP dupla regiszterbe 2 byte-os adat

regiszterbe cím tartalma: csak A-ba

dupla regiszterbe címtől két byte tartalma: BC,DE,HL,IX,IY,SP?

címre regisztert: csak A-t

címtől 2 byte-ra dupla regisztert: BC,DE,HL,IX,IY,SP

Indirekt címzések:

dupla regiszterben adott címre regisztert: BC,DE,HL,IX+távolság, IY+távolság, de BC,DE csak A-val, a többi B,C,D,E,H,L-lel is

dupla regiszterbe egy dupla regiszterben adott címen álló byte: mint előbb

dupla regiszterben adott címre adat: HL,IX+távolság,IY+távolság

BIT /bit vizsgálat/, SET /bit bekapcsolás/, RES /bit kikapcsolás/:

A megadott sorszámú bittel végez műveletet, regiszterek: A,B,C,D,E,H,L, (HL) , (IX+távolság) , (IY+távolság) . A BIT állít flageket, a SET és RES nem.

ADD /összeadás/, ADC/ maradékos összeadás/

ADD: az A regiszterhez ad egy 1 byte-os információt: B,C,D,E,H,L, (HL) , (IX+távolság) , (IY+távolság) , adat

2 byte-os ADD: HL,IX,IY regiszterhez BC,DE,SP,vagy önmaga hozzáadása

ADC: mint ADD, de paritás, maradék, előjel, nulla flag-eket állítja

2 byte-os ADC: csak HL-hez BC,DE,HL,SP

SUB/ kivonás/. SBC /kivonás átvitel/

SUB: az A-ból egy 1 byte-os információt: A,B,C,D,E,H,L,(HL), (IX+távolság), (IY+távolság), adat

SBC: kivonja A-ból a C-flaget /átvitel/ és a fenti 1 byte-os információk valamelyikét

SBC dupla regiszterre: kivonja HL-ből a C-t, és egy regiszterpárt: BC,DE, HL,SP

INC /növelés 1-gyel/. DEC /csökkentés 1-gyel/

argumentumaik: A,B,C,D,E,H,L,(HL), (IX+távolság), (IY+távolság), BC,DE,HL,SP, IX,IY

Logikai műveletek: AND /és/, OR /vagy/, XOR /kizáró vagy/, CP/hasonlítás/ mindegyiknél az egyik regiszter A, a másik 1 byte-os információ /mint a SUB utasításnál/

Vezérlésadáó utasítások: CALL /szubrutinhívás/, JP /ugrás/, JR /relatív ugrás/, RET /visszatérés/, DJNZ /csökkentés és felt. ugrás/ feltételek: semmi /-/ , nulla /Z/, nem nulla /NZ/, átvitel /C/, nincs átvitel /NC/, páros /PE/, páratlan /PO/, pozitív /P/, negatív /M/ hívásnál és ugrásnál címet, relatív ugrásnál távolságot adunk meg DJNZ csökkeneti B-t és ha az eredmény nem nulla, az adott helyre ugrik, relatív ugrással

DI - kizárja az interrupt lehetőségét
EI - megengedi az interrupt lehetőségét
RETI -visszatér az interrupt módból
RETN - visszatér a DI módból
EX - felcseréli (SP) -t HL,IX,IY valamelyikével, DE-t HL-lel, AF-et AF'-vel
EXX - felcseréli a regisztereket a vesszős regiszterekkel
CPL - A komplemente
DAA - A decimális alakja
NEG - A -l -szerese
NOP - nem csinál semmit
CCF,SCF - váltja ill. 1-re állítja C-t
HLT - megállít
IN,OUT - portról való be- és kivitelt
a C-ben levő adat a port: C , és A,B,C,D,E,H,L lehet a szállítandó, vagy befogadó regiszter, csak A-val szerepelhet a port száma
PUSCH,POP - stackbe való be- illetve kivitelt
regiszterek: AF,BC,DE,HL,IX,IY
RST - 1 byte-os szubrutin hívás, kezdőcím lehet 0,8,10,18,20,28,30,38 hexa
IM - interrupt módok 0,1,2
SLA /balra shift/, SRL /jobbra shift/, SRA /jobbra shift, első bit marad/
SLA,SRL első bit nulla, mindháromnál regiszterek mint SUB de dd nincs
Forgatások: utasításkódban R/rotate/, L/bal/, R/jobb/, C/C-flag/, regiszterek mint előbb
RRD,RLD/jobbra ill. balra forgat fél byte-ok között, a három fél byte A négy utolsó bite, (HL) első ill. utolsó 4 bitje/
CPDR,CPIR,CPD,CPI, /block compare utasítások, HL tartalmát hasonlítja A-éval/
BC-ben számolja, hogy hányszor/
INDR,INIR,IND,INI / a C-ben megadott portról vár adatokat, HL-ben számol, az A-ba kerül az adat
LDDR,LDIR,LDD,LDI /betöltés/
OTDR,OTIR,OUTD,OUTI /mint IN, de nem be hanem kivisz adatokat/

MnemonicLeírásFeltétel kódok

COMPARE

ADC HL,ss	HL←ss+CY-t HL-be	SZPC	
ADC A,r	A←r+CY-t A-ba	SZPC	
ADC A,n	A←n+CY-t A-ba	SZPC	
ADC A,(HL)	A←(HL)+CY-t A-ba	SZPC	
ADC A,(IX+d)	A←(IX+d)+CY-t A-ba	SZPC	
ADC A,(IY+d)	A←(IY+d)+CY-t A-ba	SZPC	
ADD A,n	A←n -t A-ba	SZPC	
ADD A,r	A←r-t A-ba	SZPC	
ADD A,(HL)	A←(HL) -t A-ba	SZPC	
ADD A,(IX+d)	A←(IX+d) -t A-ba	SZPC	
ADD A,(IY+d)	A←(IY+d) -t A-ba	SZPC	
ADD HL,ss	HL←ss-t HL-be		C
ADD IX,pp	IX←pp-t IX-be		C
ADD IY,rr	IY←rr-t IY-ba		
AND r	A ES r-t A-ba	SZP	C=0
AND n	A ES n-t A-ba	SZP	C=0
AND (HL)	A ES (HL) -t A-ba	SZP	C=0
AND (IX+d)	A ES (IX+d) -t A-ba	SZP	C=0
AND (IY+d)	A ES (IY+d) -t A-ba	SZP	C=0
BIT b,r	r b-ik bitjének vizsgálata	SZP	
BIT b,(HL)	(HL) b-ik bitjének vizsgálata	SZP	
BIT b,(IX+d)	(IX+d) b-ik bitjének vizsgálata	SZP	
BIT b,(IY+d)	(IY+d) b-ik bitjének vizsgálata	SZP	
CALL cc,nn	nn szubrutint hívja, ha cc		
CALL nn	nn szubrutint hívja		
CCF	komplement carry flag		C
CP r	A:r hasonlítás	SZPC	
CP n	A:n hasonlítás	SZPC	
CP (HL)	A:(HL) hasonlítás	SZPC	
CB (IX+d)	A:(IX+d) hasonlítás	SZPC	
CP (IY+d)	A:(IY+d) hasonlítás	SZPC	
CPD	blokk hasonlítás ismétlés nélkül	SZP	
CPDR	blokk hasonlítás ismétléssel	SZP	
CPI	blokk hasonlítás ismétlés nélkül	SZP	
CPIR	blokk hasonlítás ismétléssel	SZP	
CPL	A komplemente /egyes komplement/		
DAA	A decimális igazítása	SZP	
DEC r	r-t eggyel csökkenti	SZP	
DEC (HL)	(HL)-t eggyel csökkenti	SZP	
DEC (IX+d)	(IX+d) -t eggyel csökkenti	SZP	
DEC (IY+d)	(IY+d) -t eggyel csökkenti	SZP	
DEC IX	IX-t eggyel csökkenti		
DEC IY	IY-t eggyel csökkenti		
DEC ss	regiszterpárt eggyel csökkenti		
DI	interrupt elnyomás /disable interrupt/		
DJNZ e	B-t csökkenti és JR, ha B≠0		
DECREMENT 3 JUMP NON ZERO DIS	interrupt engedélyezés /enable/		
EI	(SP) és HL felcserélése		
EX (SP),HL	(SP) és IX felcserélése		
EX (SP),IX	(SP) és IY felcserélése		
EX (SP),IY	elsődleges AF-t aktiválja		
EX AF,AF'	DE és HL felcserélése		
EX DE,HL	felcseréli B--L-t B'--L'-vel		
EXX	megállás		
HALT			

MnemonicLeírásFeltétel kódok

IM Ø	Ø-s interrupt módotbeállítja	
IM 1	1-es interrupt módot beállítja	
IM 2	2-es interrupt módot beállítja	
IN A,(n)	(n)-ből inputtal tölti A-t	
IN r,(C)	(C)-ből inputtal tölti r-et	SZP
INC r	r-t egyel növeli	SZP
INC (HL)	(HL)-t egyel növeli	SZP
INC (IX+d)	(IX+d)-t egyel növeli	SZP
INC (IY+d)	(IY+d)-t egyel növeli	SZP
INC IX	IX-t egyel növeli	
INC IY	IY-t egyel növeli	
INC ss	regiszterpárt egyel növeli	
IND	blokk i/o input (C)-ből	SZP
INDR	blokk i/o input ismétléssel	SZP
INI	blokk i/o input (C)-ből	SZP
INIR	blokk i/o input ismétléssel	SZP
JP (HL)	feltétlen ugrás (HL)-re	
JP (IX)	feltétlen ugrás (IX)-re	
JP (IY)	feltétlen ugrás (IY)-ra	
JP cc,nn	nn-re ugrás, ha cc	
JP nn	feltétlen ugrás nn-re	
JR C,e	relatív ugrás, ha van carry	
JR e	feltétlen relatív ugrás	
JR NC,e	relatív ugrás ha nincs carry	
JR NZ,e	relatív ugrás ha nem nulla	
JR Z,e	relatív ugrás ha nulla	
LD A,(BC)	A-t tölti (BC)-vel	
LD A,(DE)	A-t tölti (DE)-vel	
LD A,I	A-t tölti I-vel	SZP
LD A,(nn)	A-t tölti nn címről	
LD A,R	A-t tölti R-rel	SZP
LD (BC),A	A-t tárolja (BC)-re	
LD (DE),A	A-t tárolja (DE)-re	
LD (HL),n	n-t tárolja (HL)-re	
LD dd,nn	regiszterpárt tölti nn-nel	
LD dd,(nn)	regiszterpárt tölti nn címről	
LD HL,(nn)	HL-t tölti nn címről	
LD (HL),r	r-et tárolja (HL)-re	
LD I,A	I-t tölti A-val	
LD IX, nn	IX-et tölti nn-nel	
LD IX,(nn)	IX-et tölti nn címről	
LD (IX+d),n	n-et tárolja (IX+d)-re	
LD (IX+d),r	r-et tárolja (IX+d)-re	
LD IY,nn	IY-t tölti nn-nel	
LD IY,(nn)	IY-t tölti nn címről	
LD (IY+d),n	n-et tárolja (IY+d)-re	
LD (IY+d),r	r-et tárolja (IY+d)-re	
LD (nn),A	A-t tárolja nn címre	
LD (nn),dd	regiszterpárt tárolja nn címre	
LD (nn),HL	HL-t tárolja nn címre	
LD (nn),IX	IX-et tárolja nn címre	
LD (nn),IY	IY-t tárolja nn címre	
LD R,A	R-et tölti A-val	
LD r,r'	r-et tölti r'-vel	
LD r,n	r-et tölti n-nel	
LD r,(HL)	r-et tölti (HL)-l	
LD r,(IX+d)	r-et tölti (IX+d)-vel	
LD r,(IY+d)	r-et tölti (IY+d)-vel	
LD SP,HL	SP-t tölti HL-lel	
LD SP,IX	SP-t tölti IX-szel	
LD SP,IY	SP-t tölti IY-nal	

MnemonicLeírásRelatív kódok

LDD	blokk töltés előre ismétlés nélkül	P	
LDDR	blokk töltés előre ismétléssel		P=0
LDI	blokk töltés hátra ismétlés nélkül	P	
LDIR	blokk töltés hátra ismétléssel		P=0
NEG	A negálja /kettes komplemente/	SZPC	
NOP	üres utasítás		
OR r	A VAGY r-t A-ba	SZP	C=0
OR n	A VAGY n-t A-ba	SZP	C=0
OR (HL)	A VAGY (HL) -t A-ba	SZP	C=0
OR (IX+d)	A VAGY (IX+d) -t A-ba	SZP	C=0
OR (IY+d)	A VAGY (IY+d) -t A-ba	SZP	C=0
OTDR	Blokk output hátra ismétléssel	SZP	
OTIR	blokk output előre ismétléssel	SZP	
OUT (C),r	r kivitele (C)-be		
OUT (n),A	A kivitele az n portra		
OUTD	blokk output hátra ismétlés nélkül	SZP	
OUTI	blokk output előre ismétlés nélkül	SZP	
POP IX	stackból kivesz IX-be		
POP IY	stackból kivesz IY-be		
POP qq	stackból kivesz qq-ba		
PUSH IX	IX-ből stackre tesz		
PUSH IY	IY-ből stackre tesz		
PUSH qq	qq-ből stackre tesz		
RES b,r	r, b-ik bitjét kikapcsolja		
RES b, (HL)	(HL) b-ik bitjét kikapcsolja		
RES b, (IX+d)	(IX+d) b-ik bitjét kikapcsolja		
RES b, (IY+d)	(IY+d) b-ik bitjét kikapcsolja		
RET	szubrutinból visszatér		
RET cc	szubrutinból visszatér, ha cc		
RETI	interruptból visszatér		
RETN	nem maszkolható interruptból visszatér		
RL r	r-t carry-n át balra forgatja	SZPC	
RL (HL)	(HL) -t carry-n át balra forgatja	SZPC	
RL (IX+d)	(IX+d) -t carry-n át balra forgatja	SZPC	
RL (IY+d)	(IY+d) -t carry-n át balra forgatja	SZPC	
RLA	A-t carry-n át balra forgatja	C	
RLC r	r-t balra körbe forgatja	SZPC	
RLC (HL)	(HL) -t balra körbe forgatja	SZPC	
RLC (IX+d)	(IX+d) -t balra körbe forgatja	SZPC	
RLC (IY+d)	(IY+d) -t balra körbe forgatja	SZPC	
RLCA	A-t balra körbe forgatja	C	
RLD	(HL) bcd jegyeit balra forgatja	SZP	
RR r	r-t carry-n át jobbra forgatja	SZPC	
RR (HL)	(HL) -t carry-n át jobbra forgatja	SZPC	
RR (IX+d)	(IX+d) -t carry-n át jobbra forgatja	SZPC	
RR (IY+d)	(IY+d) -t carry-n át jobbra forgatja	SZPC	
RRA	A-t carry-n át jobbra forgatja	C	
RRC r	r-t carry-n át jobbra körbe forgatja	SZPC	
RRC (HL)	(HL) -t jobbra körbe forgatja	SZPC	
RRC (IX+d)	(IX+d) -t jobbra körbe forgatja	SZPC	
RRC (IY+d)	(IY+d) -t jobbra körbe forgatja	SZPC	
RRCA	A-t jobbra körbe forgatja	C	
RRD	(HL) bcd jegyeit jobbra forgatja	SZP	
RST p	újra indítás p címtől		

MnemonicLeírásFeltétel kódok

SBC A,r	A-r-CY-t A-ba	SZPC
SBC A,n	A-n-CY-t A-ba	SZPC
SBC A,(HL)	A-(HL)-CY-t A-ba	SZPC
SBC A,(IX+d)	A-(IX+d)-CY-t A-ba	SZPC
SBC A,(IY+d)	A-(IY+d)-CY-t A-ba	SZPC
SBC HL,ss	HL-ss-CY-t HL-be	SZPC
SCF (SET CARRY FLAG)	carry flag-et állítja	C=1
SET b,(HL)	(HL) b-ik bitjét állítja	
SET b,(IX+d)	(IX+d) b-ik bitjét állítja	
SET b,(IY+d)	(IY+d) b-ik bitjét állítja	
SET b,r	r b-ik bitjét állítja	
SLA r	r balra aritmetikai eltolás	SZPC
SLA (HL)	(HL) balra aritmetikai eltolás	SZPC
SLA (IX+d)	(IX+d) balra aritmetikai eltolás	SZPC
SLA (IY+d)	(IY+d) balra aritmetikai eltolás	SZPC
SRA r	r jobbra aritmetikai eltolás	SZPC
SRA (HL)	(HL) jobbra aritmetikai eltolás	SZPC
SRA (IX+d)	(IX+d) jobbra aritmetikai eltolás	SZPC
SRA (IY+d)	(IY+d) jobbra aritmetikai eltolás	SZPC
SRL r	r jobbra logikai eltolás	SZPC
SRL (HL)	(HL) jobbra logikai eltolás	SZPC
SRL (IX+d)	(IX+d) jobbra logikai eltolás	SZPC
SRL (IY+d)	(IY+d) jobbra logikai eltolás	SZPC
SUB r	A-r-t A-ba	SZPC
SUB n	A-n-t A-ba	SZPC
SUB (HL)	A-(HL)-t A-ba	SZPC
SUB (IX+d)	A-(IX+d)-t A-ba	SZPC
SUB (IY+d)	A-(IY+d)-t A-ba	SZPC
XOR r	A KIZÁRÓ-VAGY r-t A-ba	SZF C=0
XOR n	A KIZÁRÓ-VAGY n-t A-ba	SZF C=0
XOR (HL)	A KIZÁRÓ-VAGY (HL)-t A-ba	SZF C=0
XOR (IX+d)	A KIZÁRÓ-VAGY (IX+d)-t A-ba	SZF C=0
XOR (IY+d)	A KIZÁRÓ-VAGY (IY+d)-t A-ba	SZF C=0

Jelmagyarázat:

Utasítás mező:	b	0-7 bitmező
	c	Feltételkódok /NZ,Z,NC,C,PO,PE,P,M/
	d	+127-től -128 index eltolás /displacement/
	dd	regiszterpár /BC,DE,HL,SP/
	e	relatív ugrás eltolás +127-től -128
	n	közvetlen adat vagy címérték
	pp	regiszterpár /BC,DE,HL,SP/
	qq	regiszterpár /BC,DE,HL,SP/
	r	regiszter /mint r'/
	r'	regiszter /B,C,D,E,H,L,A/
	ss	regiszterpár /BC,DE,HL,SP/
	t	RST mező: cím=tx8

Feltétel kódok:	S	előjel flaget állítja
	Z	nulla flaget állítja
	P	paritás/túlcsordulás flaget állítja
	C	carry flaget állítja

HT 1080 Z memória elrendezése

hexa cím	decimális cím	tartalom
0000-2FFF	00000-12287	Basic interpreter
3000-3BFF	12288-15359	input-output terület /lásd köv. tábla/
3C00-3FFF	15360-16383	video display memória
4000-42E8	16384-17128	Basic munkaterület/ lásd köv. tábla/
42E9-7FFF	17129-32767	Basic program, változók táblázata, tömbváltozók táblázata, stack terület, string terület, védett terület

Címtáblázat

37E0	14304	interrupt address port, bit 7=1:óra, bit 6=1:dis
37E1	14305	disk drive select
37E4	14308	1. vagy 2. magnó választása /0 vagy 1/
37E8	14312	nyomtató address port
37EC	14316	disk parancs/státusz
37ED	14317	disk track választás
37EE	14318	disk szektor választás
37EF	14319	disk adat
3800-3840	14336-14400	billentyűzet address portok
3C00-3FFF	15360-16383	video display memória
4003-4005	16387-16389	RST 10H vezérlés átadási cím
4012-4015	16402-16405	RST 38H vezérlés átadási cím
4015-401C	16405-16412	billentyűzet ellenőrző blokk
401D-4024	16413-16420	video ellenőrző blokk
4025-402C	16421-16428	nyomtató ellenőrző blokk
403D	16445	magnó port és printsize flag /bit3/ másolat
4040-4046	16448-16454	idő-terület /25msec ütés, sec,min,h,yy,dd,mm/
4047-4048	16455-16456	Dos terület kezdete
4049-404A	16457-16458	DOS memória méret
408E-408F	16526-16527	USR kezdőcím
4099	16537	INKEY \$ tárolómező
409A	16538	hibakód
409C	16540	output flag /-1,0 vagy 1/
40A0-40A1	16544-16545	string terület kezdőcím
40A2-40A3	16546-16547	Basic program kurrens sorszáma
40A4-40A5	16548-16549	Basic program kezdőcíme
40AA-40AC	16554-16556	RND mag
40AF	16559	SA random vált. típus flag állítja
40B1-40B2	16561-16562	Basic számára elérhető utolsó memóriacím
40D6-40D7	16598-16599	string terület köv. használható byte-ja
40DF-40E0	16607-16608	system szalag belépési pontja
40E6-40E7	16614-16615	utoljára végrehajtott Basic utasítás terminátora
40EA-40EB	16618-16619	hibás sor száma
40F9-40FA	16633-16634	változótábla kezdete
40FB-40FC	16635-16636	tömbváltozó-tábla kezdete
40FD-40FE	16637-16638	tömbváltozó-tábla utolsó byte-ja
4101-411A	16641-16666	változótípus tábla
411D-4124	16669-16676	SA /software accumulator/
4127-412E	16679-16686	SA, /alternate software accumulator/
4152-41AF	16722-16815	DOS interface terület
41A6-41A8	16806-16808	Basic hibaüzenet terület
40A7-40A8		A billentyűzet -buffer kezdőcíme
40FF-4100		DATA pointer (sor végére mutat!)

BASIC táblázatok

Egyszerű változótábla

Erre mutat 40F9-40FA, a tábla elemei a következő típusuak lehetnek:

Egész változó: typeflag /2/
változónév második karaktere
változónév első karaktere
érték /LSB, alsó byte/
érték /MSB, felső, értékesebb byte/

Egyszeres pontosságú: typeflag /4/
változónév második karaktere
változónév első karaktere
érték /LSB/
érték /2. byte/
érték /MSB/
kitevő

Duplapontosságú: typeflag /8/
változónév második karaktere
változónév első karaktere
érték /7 byte-on: LSB,2.byte-on,...,6.byte,MSB/
kitevő

String változó: typeflag /3/
változónév második karaktere
változónév első karaktere
string hossza
kezdőcím LSB-je
kezdőcím MSB-je

Tömbváltozó tábla:

typeflag
változónév második karaktere
változónév első karaktere
tömb tárolóterület mérete /LSB/
tömb tárolóterület mérete /MSB/
dimenziók száma
első dimenzió LSB-je
első dimenzió MSB-je

BASIC kulcsszó tábla

hexa cím	kulcs- szó	hexa cím	kulcs- szó
4152	CVI	417C	FIELD
4155	FN	417F	GET
4158	CVS	4182	PUT
415B	DEF	4185	CLOSE
415E	CVD	4188	LOAD
4161	EOF	418B	MERGE
4164	LOC	418E	NAME
4167	LOF	4191	KILL
416A	MKI	4197	LSET
416D	MKS	419A	RSET
4170	MKD	419D	INSTR
4173	CMD	41A0	SAVE
4176	TIME	41A3	LINE
4179	OPEN		

Ezek a három byte-os területek a lemez nélküli rendszerekben mind JP 012DH utasítást tartalmaznak és hibaüzenetet adnak. A címeket átírva ezekhez a kulcsszavakhoz tartozó saját rutinokkal bővíthetjük a BASIC-et.

Memóriacím

3801H
14337

0	1	4	8	16	32	64	128
A	A	B	C	D	E	F	G

3802H
14338

H	I	J	K	L	M	N	O
---	---	---	---	---	---	---	---

3804H
14340

P	Q	R	S	T	U	V	W
---	---	---	---	---	---	---	---

3808H
14344

X	Y	Z
---	---	---

3810H
14352

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

3820H
14362

(	)	;	+	<	=	>	?
---	---	---	---	---	---	---	---

3840H
14400

ENT	CLR	BRK	↑	↓	←	→	SPC
-----	-----	-----	---	---	---	---	-----

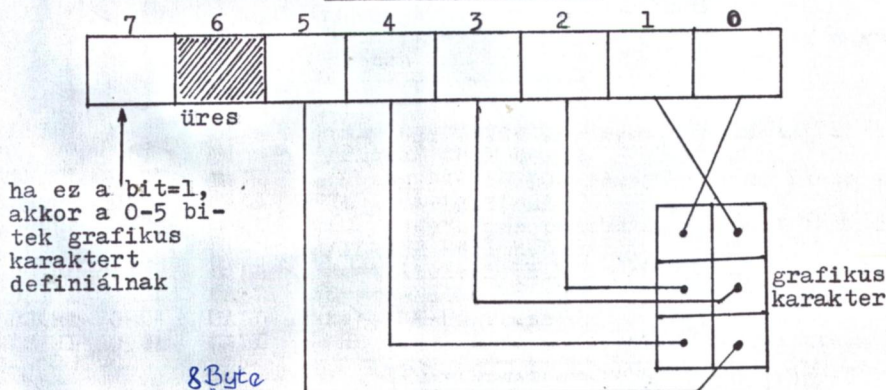
3880H
14464

SH

0	1	2	3	4	5	6	7
00H	02H	04H	08H	10H	20H	40H	80H

Rövidítések: ENT=new line, CLR=clear, BRK=break, SPC=space, SH=shift
 Példa: ha az "S" billentyűt megnyomjuk, a 3804H címen 08H lesz, az összes többi címen 00H

Grafikus karakterek



Példa: a BASIC SET(63,16) utasítása az 5. karakteres sor 31. oszlopában lévő grafikus karakter középső sorának jobb oldali képelemét állítja, tehát a $15360 + 351 = 15711 = 3D5FH$ cím 3-as blokkját, assemblerben mindössze két utasítás:

LD HL, 3D5FH
 SET 3, (HL) 5 Byte

címke	érték	típus	leírás
<u>USR-ből visszatérés /nem RET-tel/</u>			
BSCPAR	0A9A	JP	Basic programba visszatérés paraméterrel
BASIC	1A19	JP	Basic-hez visszatérés /READY üzenet, 1A19 helyett 06CC vagy 0072 lehetséges
GETPAR	0AF7	CALL	az USR(A) A paraméterét betölti HL-be, ez kell hogy a user szubrutin első utasítása legyen
DELAY	0060	CALL	BC-be n-et töltve n üres ciklus várakozás
<u>aritmetikai ROM rutinok</u>			
ADDINT	0BD2	CALL	DE és HL összege, eredmény HL-ben és SA-ban, ha nincs túlsordulás; SA-ban ha van /flag=4/
ADDSP	0716	CALL	RA és SA összege SA-ban
ADDDP	0C77	CALL	SA és SA ₁ összege SA-ban
CPINT	0A39	CALL	DE és HL összehasonlítása
CPDP	0A78	CALL	SA és SA ₁ összehasonlítása
CP16	0018	RST	DE és HL mint 16-bites előjel nélküli egészek összehasonlítása
DIVINT	2490	CALL	DE osztva HL-lel, eredmény SA-ban egyszeres pont.
DIVSP	08A2	CALL	RA osztva SA-val, eredmény SA-ban
DIVDP	0DE5	CALL	SA osztva SA ₁ -gyel, eredmény SA-ban
MLTINT	0BF2	CALL	DE szorozva HL-lel, ha nincs túlsordulás /flag=2/, akkor eredmény HL-ben és SA-ban, ha van /flag=4/, eredmény csak SA-ban
MLTSP	0847	CALL	RA szorozva SA-val, eredmény SA-ban
MLTDP	0DA1	CALL	SA szorozva SA ₁ -gyel, eredmény SA-ban
POWER	13F7	CALL	RA az SA-dik hatványon, eredmény SA-ban
SUBINT	0BC7	CALL	DE mínusz HL, ha nincs túlsordulás /flag=2/, eredmény HL-ben és SA-ban, ha van /flag=4/, csak SA-ban
SUBSP	0713	CALL	RA mínusz SA, eredmény SA-ban
SUBDP	0C70	CALL	SA mínusz SA ₁ , eredmény SA-ban
<u>Basic változók keresése</u>			
VARPTR	260D	CALL	HL a változó névre mutat, eredmény a változó címe DE-ben
<u>Adatmozgatás</u>			
LDDEHL	09D2	CALL	(HL) egyszeres pontosságú számot (DE)-be viszi /FLAGSP kell hozzá/
LDRAHL	09C2	CALL	(HL) egyszeres pontosságú számot RA-ba viszi
LDRAA	09BF	CALL	SA-t RA-ba viszi
LDHLDE	09D3	CALL	(DE) egyszeres pontosságú számot (HL) -be viszi /FLAGSP kell hozzá/
LDHLA	09CB	CALL	SA-t átviszi (HL) -be /HL által mutatott címre/
LDSTA	09A4	CALL	SA-t a stackbe viszi
LDSARA	09B4	CALL	RA-t SA-ba viszi
LDSAHL	09B1	CALL	(HL) egyszeres pontosságú számot SA-ba viszi
<u>Duplapontosságú adatmozgatás</u>			
FLAGDP	0AEC	CALL	SA typeflag-ját duplapontosságúra állítja
MVALT	09FC	CALL	SA-t SA ₁ -be viszi /FLAGDP kell hozzá/
MVDEHL	09D3	CALL	(HL) számot (DE)-be viszi /FLAGDP kell/
MVHLDE	09D2	CALL	(DE) számot (HL) -be viszi /FLAGDP kell/
MVSAHL	09F7	CALL	(HL) -t SA-ba viszi /FLAGDP kell/
SA	411D	EQU	SA első byte-ja
SA1	4127	EQU	SA ₁ első byte-ja
SAFLAG	40AF	EQU	SA typeflag-je
MVVAR	0982	CALL	(DE)-ből /DE által mutatott területről/ a typeflag által mutatott számú byte-ot átvisz HL-be

typeflag kell/

Stringkezelő rutinok

BSCTXT	40A4	EQU
CPSTR	25A1	CALL
MOVEA	09D6	CALL
MOVEB	09D7	CALL
MVSTR	21E3	CALL

ezen a címen kezdődik a Basic program /szöveg/ két string összehasonlítása, HL és BC a két string-re mutat, D és E tartalmazza a stringek hosszát (DE)-ből (HL)-be adatmozgatás, byteszám A-ban mint MOVEA, de byteszám B-ben stringet a stringterületre mozgat, HL a buffer első byte-jára, BC a változónévre mutat

Hibakezelés

ERTRAP	41A6	EQU
ERCODE	409A	EQU

hibakezelés ide ugrik, innen RET a basicbe itt tárolódik a hibakód, ha ezen címen nem OOH van, akkor a READY üzenet után kinyomtatásra kerül a hiba sora és a Basic EDIT módba megy át pointer az utoljára végrehajtott Basic utasítás végére /OO vagy ":"/ áll
HL egy Basic utasítás ":" végére vagy egy sor OOH-végére mutat, a végrehajtás a következő utasításnál folytatódik

BSCSTM	40E6	EQU
RUNSTM	1D1E	JP

Matematikai függvények

1. típus: SA-ban egész, egyszeres vagy kétszeres pontosságú szám, typeflag mutatja, hogy milyen; eredmény SA-ban, typeflaget állítja
2. típus : SA-ban egyszeres pontosságú szám, eredmény SA-ban/

ABS	0977	CALL
ATN	15BD	CALL
COS	1541	CALL
EXP	1439	CALL
FIX	0B26	CALL
FLAGDP	0AEC	CALL
FLAGIN	0A9D	CALL
FLAGSP	0AEF	CALL
INT	0B37	CALL
LOG	0809	CALL
RND	14C9	CALL
RANDOM	01D3	CALL
RSETSA	0778	CALL
SGN	098A	CALL
SIN	1547	CALL
SQR	13E7	CALL
TAN	15A8	CALL

1.típus
2.típus
2.típus
2.típus
2.típus
SA typeflag-jét duplapontosságúra állítja
SA typeflag-jét egész típusúra állítja
SA typeflag-jét egyszeres pontosságúra állítja
1.típus
2.típus
1.típus
randomizál
SA-t kinullázza
1.típus
2.típus
2.típus
2.típus

KONVERZÍÓ

CSAASC	0FBF	CALL
CSAINT	0A7F	CALL
CSASP	0AB1	CALL
CSADB	0ADB	CALL
VAL	1E5A	CALL

SA-t konvertálja ASCII-ra /typeflag áll./, az eredmény /decimálisan/ a 4130H címen kezdődő bufferben, eredmény végén OOH, (HL)=4130
SA-t egészé, eredmény SA-ban /typeflag állítandó/
SA-t egyszeres pontosságúra, eredm. SA-ban /typefl./
SA-t duplapontosságúra, eredmény SA-ban /typeflag/
decimális számot megadó stringet /végén OOH byte/
binárisá konvertál, HL az első karakterre mutat, eredmény DE-ben

CLS	0109	CALL	képernyő törlés
KBDSCN	002B	CALL	billentyűzet vizsgálata /scan/, eredmény A-ban
OUTSEL	409C	EQU	output eszköz választás, -1 a kazetta, 0 a képernyő, 1 a printer
OUTSTR	28A7	CALL	string outputja, a stringet 00H vagy 22H /" kódja/ terminálja, output eszköz választása OUTSEL-lel
OUTCHR	032A	CALL	A regiszterből karakter kivitele / eszközválasztás OUTSEL-lel/
RDSECT	46DD	CALL	lemezolvasás és írás, mindkettőnél C =drive,
WTSECT	46E6	CALL	D =track, E =sector, HL =data buffer, Z=1, ha sikeres volt, egyébként hibaiüzenet kód A-ban

Grafika

GRAPH 0132 JP Használata: visszatérési cím HL-be, PUSH HL, A-ba 0 /POINT/ vagy 1 /RESET/ vagy 80H /SET/, PUSH AF, A-ba x-koordinata, PUSH AF, A-ba y-koordinata, JP GRAPH

Összehasonlítások /CPxxx rutinok/ eredménye

ha egyenlőek	Z=1	Z
ha nem egyenlőek	Z=0	NZ
ha az első kisebb	C=1	C
ha a második kisebb	C=0	NC

v. egyenlő

Software accumulator /SA/

40AF	411D	411E	411F	4120	4121	4122	4123	4124
typeflag -----számok-----								

2					LSB	MSB		
4					LSB	byte2	MSB	EXP
8	LSB	byte2	byte3	byte4	byte5	byte6	MSB	EXP

A 40AFH /16559D/ címen található typeflag értéke szerint értelmeződik SA tartalma. Ha typeflag=2, akkor a 4124/16673/ és 4122/16674/ címeken egész szám van /LSB=less significant byte, alsó, MSB=more significant byte, felső, értékesebb byte/. Ha 4, akkor egyszeres pontosságú szám, ha 8, akkor dupla-pontosságú szám, ezek elhelyezését a táblázat mutatja. SA, másodlagos software accumulator a 4127-412E /16679-16686/ címeken, csak duplapontosságú számokra használható.

Egyszeres pontosságú számok és egész számok regiszterekben is tárolhatók. Egész számok a DE és HL regiszterekben tárolódnak. Egyszeres pontosságiak gyakran az E,D,C,B regiszterekben /sorra LSB, byte2, MSB, EXP/, ezt nevezzük regiszter accumulatornak /RA/.

Megjegyzés: /rr/ jelentése az rr által mutatott cím tartalma /adatmozgatás, duplapontosságú adatmozgatás, stringkezelő rutinok/

Z 80 mikroprocesszor

aritmetikai vagy
logikai művelet
eredménye

aritmetikai és lo-
gikai egység

adatátvitel
memóriából
memóriába

első második
operandus

A	F	A'	F'
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'

általános regiszterek

adatátvitel
regiszterek
között

IX	
IY	
SP	
PC	
I	R

speciális regiszterek

Logikai műveletek

LD A, (HL)	binárisból	LD A, (HL)	2.bitet	LD A, (HL)	2.bitet
OR 30H	ASCII-t	OR 4,A	1-re	AND A, FBH	0-ra
		LD (HL), A		LD (HL), A	állítja

Aritmetikai, logikai és összehasonlító műveletek

Az ábra lényege: 8-bites műveletnél egyik regiszter az A, másik operandus A,H,L,B,C,D,E regiszterből vagy memóriából jöhet, eredmény az A-ba kerül. Lehet regiszterek vagy memóriacímek tartalmát 1-gyel növelni /INC/ vagy csökkenteni /DEC/. 16-bites műveletek HL,IX,IY között. /AZ utasítástáblázatból ki lehet olvasni, hogy pontosan milyen kombinációk lehetnek./

Adatátvitel: LD-vel regiszterből regiszterbe, regiszterből memóriába, memóriából regiszterbe, utasítás operandus mezőjéből regiszterbe, memóriába, regiszterből i/o eszközre, i/o eszközből regiszterbe lehet adatot átvinni.

Példa **PÉLDA**

```

ORG 7000H      ORG 7000H
LOAD 7000H     LOAD 7000H
CALL 01C9H    LD A,"x"
LD A,"Q"      LD HL,3C00H
LD (3D20H),A  LD B,255
HALT          CIKL: LD (HL),A
END           INC HL
              DEC B
              JR NZ,CIKL
              VEGE: JR VEGE
              END
    
```

Példa

```

ORG 7000H
LOAD 7000H
CALL 01C9H;CLS
LD HL,SZOVEG
LD DE,VIDEO+512
LOOP: LD A,(HL)
CP "e"
JP Z,1A19H
LD (DE),A
INC HL
INC DE
JR LOOP
VIDEO: EQU 3C00H
SZOVEG: DB "COMPUTER"
END
    
```

Példa

```

LD HL,0 forrás
LD DE,3C00H cél
LD BC,1024 byteszám
LDIR
    
```

Szorzás

	A	C
LD A, (HL)	17	
SLA A	34	
LD C,A	34	34
SLA A	68	34
SLA A	136	34
ADD A,C	170	34
LD (HL),A	170	34

16-bites összeadás

```

LD A, (IX+1)  alsó byte
ADD A, (IX+1)
LD (IX+1),A
LD A, (IX+0)  felső byte
ADC A, (IX+0) carryt is
LD (IX+0),A   hozzáadja
    
```

Input/output

```

LD A, (3801H) első sor
OR A 0-e? trükk!
JP Z,NOKEY nem nyomták
LD B,FBH meg
K2: INC B
SRL A toljuk,míg nem 0
JP NZ,K2
LD A,B
LD B,0
LD HL,TABLE karakter-
ADD HL,3C táblából tölt
LD A,(HL)
    
```

Címzési módok

ADD A,23 azonnali, A-hoz 23-t hozzáad
LD BC,3700H azonnali, BC d.reg.-be 3700H
ADD A,D regiszter címzés, A-hoz D tartalmát adja hozzá
LD A,(1000H) közvetlen, a regisztert a memóriában (1000H) című tartalommal tölti, a regisztert amegadott címre tárolja
LD HL,1000H regiszter indirekt: HL-be tölti LD A,(HL) címet, majd a HL-ben lévő címről tölti az A-t
JR LOOP relatív címzés /-127...+127/
LD B,(IX+40) indexelt címzés IX+40-ról tölt

LD C, /időtartam/
 CONT: LD B, /frekvencia/
 LD A, 1
 OUT (OFFH), A felső fél hullám
 LOOP1: DJNZ LOOP1 várakozik /frekv./
 LD B, /frekvencia/
 LD A, 2 alsó fél hullám
 OUT (OFFH), A
 LOOP2: DJNZ LOOP2 várakozik
 DEC C időt csökkenti
 JP NZ, CONT következő hullám

Assembler szubrutin

Betöltése előtt Basic számára elérhető
 memória felső vége /16561-16562/ áti-
 randó ORG-nál kisebbre, vagy bekapcsol-
 láskor a READY?-re /memória méret?/ kel-
 ennél kisebbet megadni.
 USR hívása
 16526-16527 USR kezdőcím mezőbe beírni
 az assembler rutin kezdőcímét:
 100 POKE 16526,xxxx:POKE 16527,yyyy
 110 X=USR(A) ALSO
 visszatérést lásd a ROM rutinok elején

Memória elrendezés:

a MEM táblában nem szereplő fontos érték 16416-16417 a cursor kezdő címe

PEEK és POKE

Basic program mérete= vált.tábla kezdete-Basic pgm. kezdőcíme

PRINT PEEK (16634) *256+PEEK (16633) -PEEK (16549) *256-PEEK (16548)

Programsor felépítése:

következő sorra mutató pointer /2/ sorszám /2/ utasítás kódja /1/ szövegrész,
 változók, műveleti jelek stb. /xx/ sorvége jel /1/ file vége jel /2/

Merge

első programot betöltjük, ekkor 40F9-40FA /vált.tábla kezdete/ az első pro-
 gram utáni első címre mutat, ebből kivonunk 2-t /töröljük a file vége jelet/
 átmásoljuk 40A4-40A5-re /basic program kezdőcíme/, betöltjük a második prog-
 ramot, majd az eredeti címet visszairjuk 40A4-40A5-re

?PEEK (16548):?PEEK (16549) leírjuk /x,y/

CLOAD prog 1

?PEEK (16633):?PEEK (16634) kivonunk belőle kettőt /marad v1,v1/

POKE 16548,v1:POKE 16549,v1

CLOAD prog 2

POKE 16548,x:POKE 16549,y visszairjuk az eredeti kezdőcímet

0049H Egy karakter olvasása a billentyűetről az **A** regiszterbe.

0033H Egy karakter írása a képernyőre az **A** regiszterből.

0509H Egy sor behívása a billentyűetről a **HL** által megadott címre
(It max. karaktermódot a **B** regiszter tartalmára)

0212H Az **A** regiszterben megadott maghó motorjának indítása (1 illetve 2)

0211H Az aktuális maghó motorjának kikapcsolása (előző **HL**-be FB00H)

0235H 1 byte olvasása kasettről az **A** regiszterbe

0264H 1 byte írása kasettra az **A** regiszterből