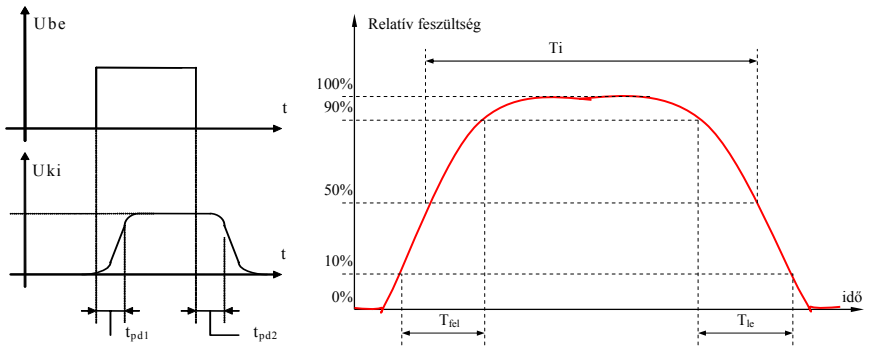
1. Digitális integrált áramkörök

1.1. Logikai függvény elektronikus megvalósítása

Figyelembe vett szempontok:
- legyen könnyen sokszorosítható
- legyenek elemi logikai függvények is (pl. ÉS, -VAGY-, ...)
- rendelkezzen kedvező műszaki tulajdonságokkal (méret, energia-felvétel, környezetei hatások.)
- a két logikai állapot legyen jól megkülönböztethető
- A logikai állapotot: igen-nem, 1-0, valamilyen fizikai jellemző értéke reprezentálja. Elektromos áramköröknél a jellemző lehet:
  - ellenállás (kapcsolók, relék)
  - áram, feszültség (elektronikus áramkör)
  - mágnesezettségi állapot (mágneses tárolók)

- Integrált áramköröknél a logikai érték fizikai megfelelője a feszültség (ami természetesen nem független az áramtól).
- Digitális berendezés megvalósításakor az elemi és az összetett logikai függvényeket realizáló áramkörök (IC-k) ki- és bemeneti pontjait kell megfelelően összekapcsolni.
- Biztosítani kell a berendezés ki- és bemeneti jeleinek illesztését az IC-k ki- és bemeneti jeleihez.

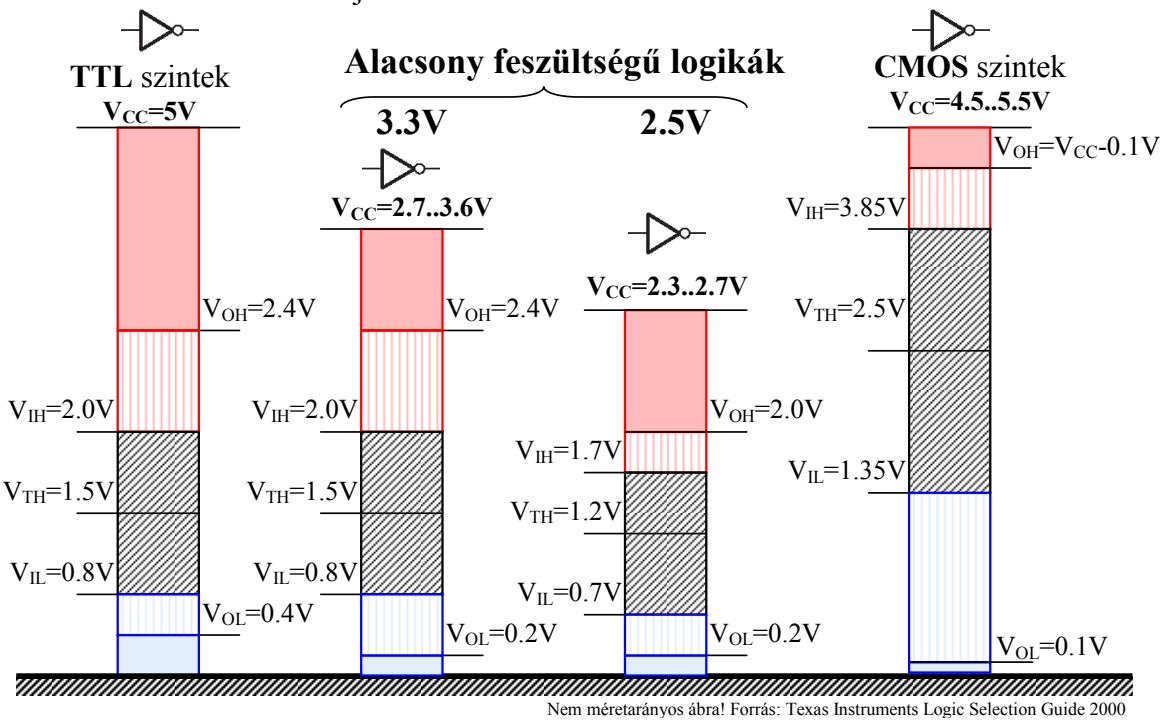
Időbeni viselkedés:
Az ideális kapcsolóelemek átkapcsolási ideje 0s. Valóságos elemek esetén az átkapcsolási idő késleltetésként jelentkezik. Ez a késleltetés a kapcsolások tervezésekor beszámítandó. Figyelembe kell venni a hazárdok kialakulásának lehetőségét, a be/kimeneti változások időfüggvényét, a kimeneti változások érvényességét.



1.1.1. ábra

1.2. Az integrált áramkörök jellemzői:

- Az áramkör logikai feladata, mely megadható logikai egyenlettel, táblázattal (igazságtábla, állapottábla, stb.), ütemdiagrammal
- Sebesség-időjellemzők úgy, mint. Késleltetések, jeltartási idők, időkorlátok
- Teljesítmény jellemzők:. eldisszipált teljesítmény (P_d)
- Tápfeszültség jellemzők: a tápfeszültség névleges értéke és toleranciája
- Logikai értékhatárok: a 0 vagy 1 logikai szintnek megfelelő, érvényesnek tekintett bemeneti és kimeneti feszültséghatárok, melyek a bemeneti és kimeneti minimum, maximum értékeket jelentik



Nem méretarányos ábra! Forrás: Texas Instruments Logic Selection Guide 2000

1.2.1. ábra a különféle logikai szintek

- Pozitív logika esetén a H-szintnek az 1, L szintnek a 0 logikai értéket feleltetjük meg
- Negatív logika esetén a hozzárendelés fordított
- Terhelési és terhelhetőségi jellemzők:. kimeneti terhelhetőség FAN-OUT, bemenetek fogyasztása (terhelés az előző fokozat számára) FAN-IN ezeket az értékeket sokszor egységértékben adják meg (pl.:FAN-OUT=10 TTL, azaz 10 TTL bemenetet tud meghajtani)
- Hőmérsékleti jellemzők, üzemi hőmérséklettartomány, tárolási hőmérséklettartomány
- Csatlakozási topológia, mely tartalmazza a tokozást, IC lábkialakítást, IC belső összeköttetéseire vonatkozó információt

Az áramkör-katalógusokban szereplő adatok a következő felosztásban szerepelnek:

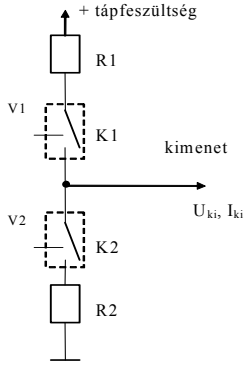
- az áramkör meghibásodását még nem eredményező maximális értékhatárok (absolute maximum ratings),
- üzemi feltételek, az optimális üzemeltetéshez (recommended operating conditions)
- az üzemi feltételek betartása melletti jellemzők: minimum, tipikus és maximum értékek.

1.3. Az integrált áramkörök kimenetei

Az integrált áramkörök kimenete három kapcsolási alapkialakításra épül:

- totem-pole (TP)
- Három állapotú (three-state, 3st)
- Nyitott kollektoros (open-kollektor, OC)

Az áramkörök általános leegyszerűsített kimeneti fokozata:



R1>>R2, R2-t fizikailag nem is építik be, csak a K2 kapcsoló átmeneti ellenállását modellezi.
K1, K2 félvezető eszközökkel megvalósított kapcsoló (kapcsolóüzemű tranzisztorok), amelyeket belső V1 és V2 vezérlőjelek vezérelnek (nyit-zár)

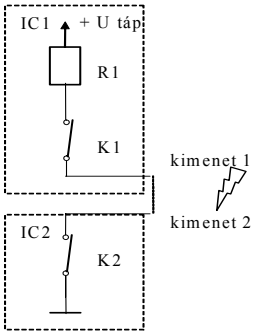
1.3.1. ábra

Totem-pole kimenet:

Az áramkör belső V vezérlőjelével $V_1 = V, V_2 = \overline{V}$

- L szint előállítása: K1 nyit, K2 zár
- H szint előállítása: K1 zár, K2 nyit

A totem-pole kimenet "ellenállásai" olyanok, hogy a kimenetek közvetlen összekötését nem viselik el.

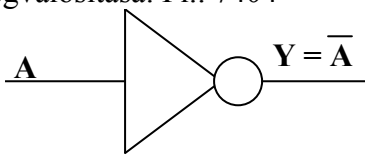


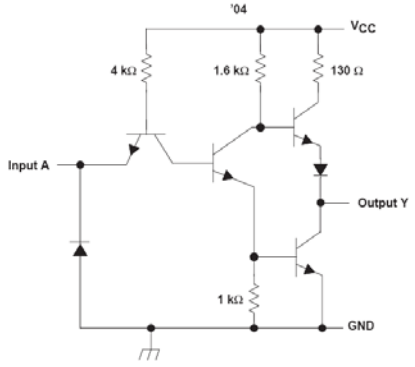
Közvetlen összekötés esetén, ha a két jel eltérő logikai értékű (itt kimenet1 = '1', kimenet2 = '0') a 2. kimeneten keresztül minimálisan

$I_{min} = \frac{U_{t‡p}}{R_1}$ áram folyik, amelyből $I_{min}{}^2 * R_1$ teljesítményt el kell disszipálni.
Ezt az áramkörök nem viselik el, a kimeneti fokozatuk tönkremehet.

1.3.2. ábra

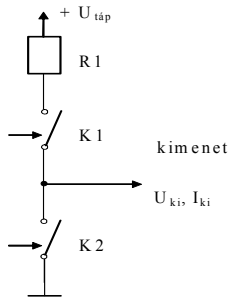
Totem-pole kimenet konkrét megvalósítása: Pl.: 7404





1.3.3. ábra

Three-state kimenet:

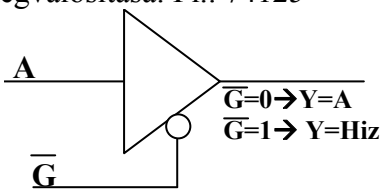


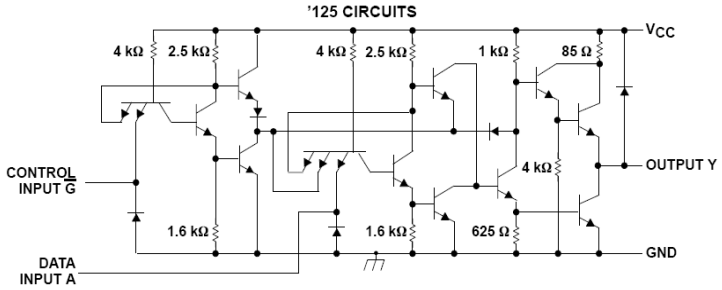
1.3.4. ábra

K1 és K2 külön vezérelhetők.

- L szint előállítása: K1 nyit, K2 zár
- H szint előállítása: K1 zár, K2 nyit
- Nagy impedanciás állapot: K1 nyit, K2 nyit, a kimeneti pont belső meghajtás nélkül lebeg, tehát a vonal más kimenet által történő meghajtását teszi lehetővé. A helyes működés feltétele több kimenet összekötése esetén az, hogy a közvetlenül összekötött kimenetek közül egyidejűleg csak egynek lehet K1 kapcsolója zárt állapotban. Figyelembe kell venni azontúl, hogy a K1, K2 kapcsolók nem ideális kapcsolók, azaz kikapcsolt állapotban az impedanciájuk igen nagy, de nem végtelen.

Three-state pole kimenet konkrét megvalósítása: Pl.: 74125

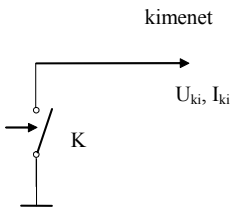




1.3.5. ábra

Nyitott kollektoros kimenet:

Felépítésében a totem-pole kimenettől annyiban tér el, hogy a K1 és R1 elemek elmaradnak

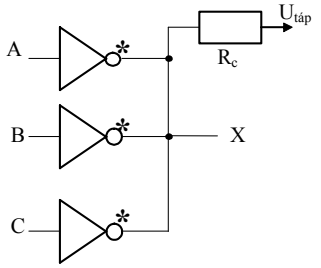


1.3.6. ábra

- L szint előállítása: K zár
- H szint előállítása: K nyit, a tényleges H szintet külső felhúzó ellenállással (Rc) kell biztosítani.

Több open kollektoros kimenet közvetlen összekötésével huzalozott NOR kapcsolás hozható létre:

- Rc-t mindig úgy kell méretezni, hogy ha minden kimenet K2 kapcsolója nyitva van, akkor a közös jel logikai '1' értékű legyen,
- ha bármely, vagy több kimenet K2 kapcsolója zár, akkor a közös jel garantáltan logikai 0 legyen.



1.3.7. ábra Huzalozott NOR kapcsolat

A	B	C	X
1	**-**	**-**	0
-	**1**	**-**	0
-	**-**	**1**	0
0	**0**	**0**	1

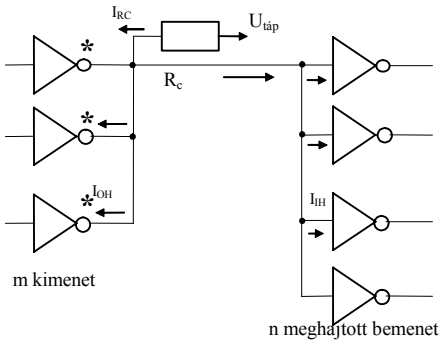
$$X = \overline{A} \cdot \overline{B} \cdot \overline{C} = \overline{A + B + C}$$

Az Rc ellenállás méretezése:
m db nyitott kollektoros kimenet van összekötve, a közös jel n kapubemenetet hajt meg, a közös ellenálláson eső feszültség:

$$U_{RC} = I_{RC} \cdot R_C$$

ahol I_{RC} az R_C ellenálláson átfolyó áram, és az $U_X = U_{táp} - U_{RC}$ feszültségnek a meghajtástól függően a '0' (0V...U_{0Lmax}) logikai, vagy '1' (U_{OHmin}...$U_{táp}$) logikai szintnek megfelelő feszültségsávba kell esnie:

- U_{0Lmax} a '0' szint maximális értéke
- U_{OHmin} az '1' szint minimális értéke



1.3.8. ábra

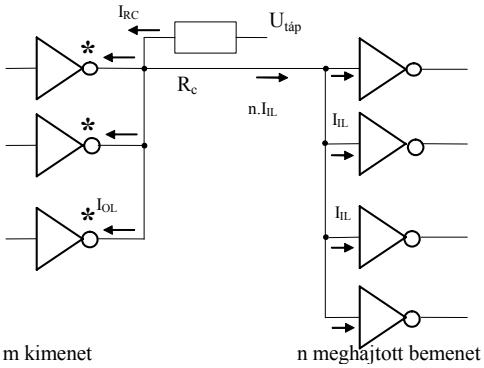
ha minden kimenet '1' állapotban van, akkor az U_{RC} feszültségesés az R_C ellenálláson nem lehet nagyobb, mint a logikai '1'-nek megfelelő feszültségsáv szélessége:
ha ekkor egy bemeneten I_{IH} (input-high) áramnak és egy kimeneten I_{OH} (output-high) áramnak kell folynia

$$U_{táp} - U_{OHmin} \geq R_c I_{RC} = (m \cdot I_{OH} + n \cdot I_{IH}) \cdot R_c$$

R_C maximális értékére:

$$R_{Cmax} \leq \frac{U_{táp} - U_{OHmin}}{m \cdot I_{OH} + I_{IH} n}$$

U_{OHmin}, I_{OH}, I_{IH} az áramköri technológiától függ, ha minden '1' állapotban, kivéve egyet, akkor a közös jelnek logikai '0'-nak kell lennie.



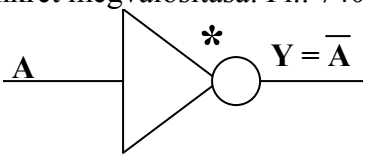
1.3.9. ábra

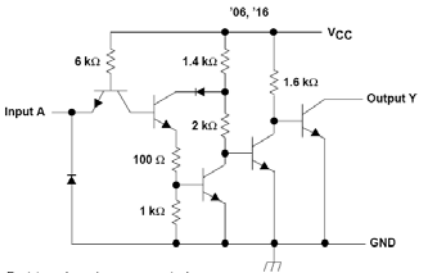
$$U_{táp} \le R_C \cdot I_{RC} = R_C \cdot (I_{OLmax} - n \cdot I_{IL})$$

ha feltételezzük, hogy csak egy kimenet akarja '0'-ba húzni a közös jelet (ha több, akkor a helyzet javul) és a közös pontra áramösszeget.
R_C minimumára a feltétel:

$$R_{C\,min} \ge \frac{U_{táp}}{I_{OL} - n \cdot I_{IL}}$$

az R_{cmin} és az R_{cmax} közötti ellenállásértékek közül választhatunk.
Nyitott kollektoros kimenet
konkrét megvalósítása: Pl.: 7406



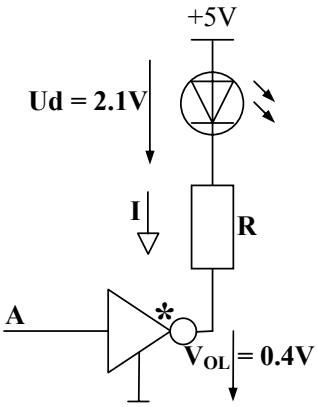


1.3.10. ábra

Kapukimenetek gyakorlati alkalmazása

Kimenet típusa	Alkalmazási példák
TP	Általános célú integrált áramkörök kimenete (pl.: logikai alapműveletek, számlálók, aritmetikai áramkörök, stb.)
3st	Mikroprocesszoros áramkörök és perifériáik Memória áramkörök kimenetei Sín meghajtó áramkörök
OC	Huzalozott függvénykapcsolatok kialakítása (pl.: Wired-NOR) Különböző feszültség logikák közötti szintillesztés TDMA adatátvitel Megszakítás vonalak meghajtása mikroprocesszoros rendszerekben Meghajtás (izzólámpa, relé, LED)

LED meghajtása

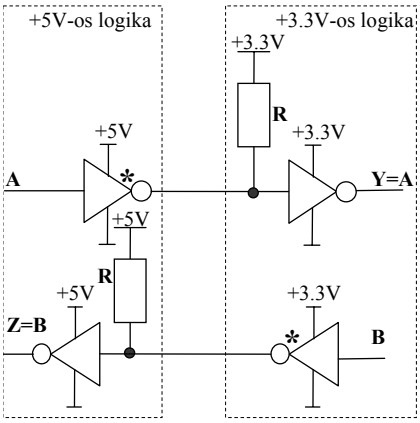


Zöld LED
I = 7mA esetén Ud = 2.1V

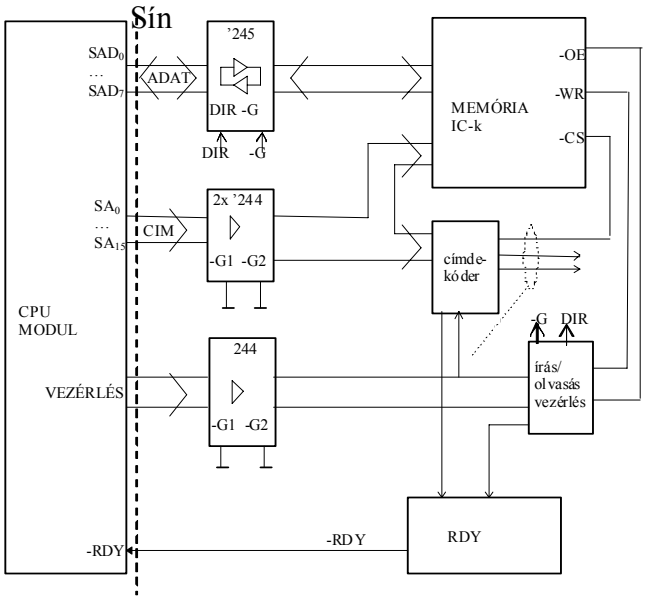
R előtét ellenállás meghatározása:

$$R = \frac{Utáp - Ud - V_{OL}}{I} = \frac{5 - 2.1 - 0.4}{0.007} = 357.142\Omega \longrightarrow \underline{\underline{360\Omega}}$$

Szint illesztés



2-4. Memória illesztése mikroprocesszorhoz



6.1. ábra

Adatmeghajtó:

Célrendszereknél esetfüggően alkalmazva, sín esetén mindig használni kell.

DIR vezérlés kérdései (van-e mögötte is olvasható eszköz), általában az olvasás vezérlőjel hajtja meg,

G = CIMazonosság · parancsazonosság

Címmeghajtó:

Célrendszerek esetén a kapacitív és az egyenáramú terhelések alapján mérlegelendő az alkalmazásuk sínrendszereknél minden esetben kell.

Ready:

Sebesség szinkronizálására.

Hibajelző képesség → watch dog.

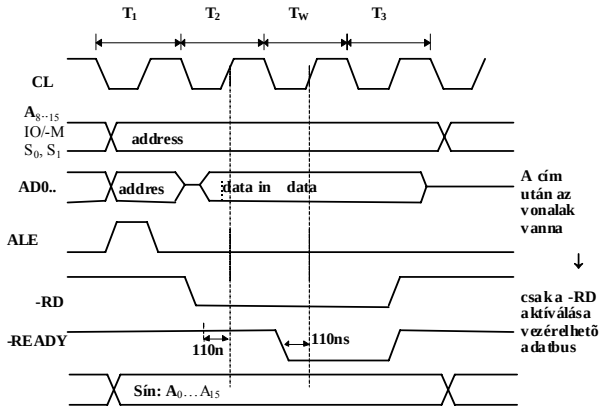
Ready stratégiák:

a) nem kell (mindig kész)

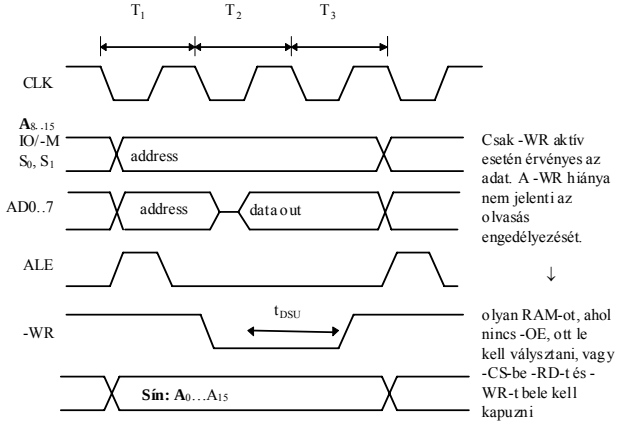
b) ott kell, ahol lassú (itt WAIT-et kér)

c) **mindig kell** (ready-t ad)

1. Időzítési viszonyok a processzor szempontjából



6.2. ábra Tipikus olvasási ciklus

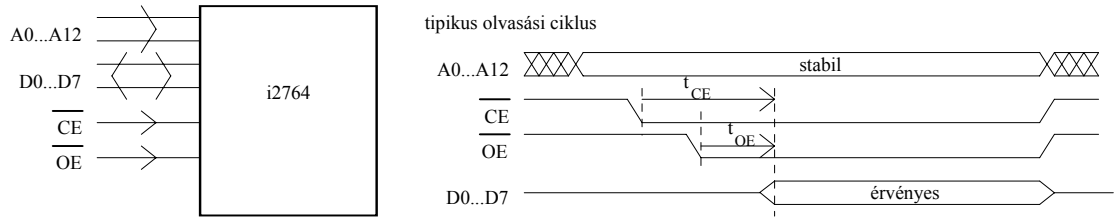


6.3. ábra Tipikus írási ciklus

2. Időzítési viszonyok a memóriaelemek szempontjából – konkrét típusok kapcsán

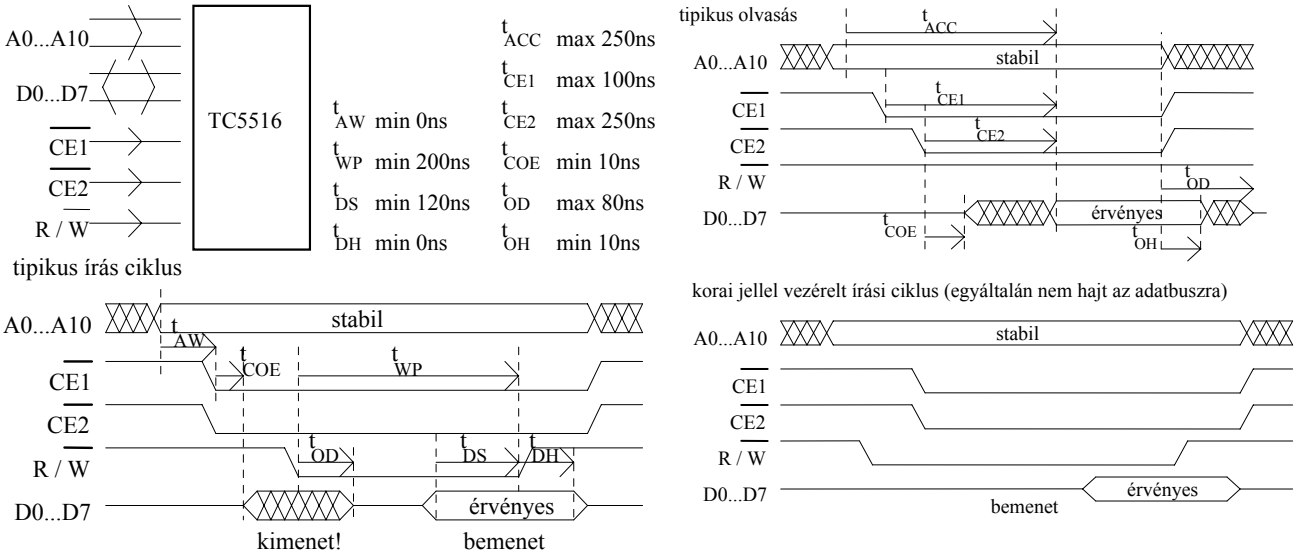
Memóriák: RAM (statikus, dinamikus), ROM, PROM, EPROM, EEPROM

i2764 8kb EPROM



6.4. ábra

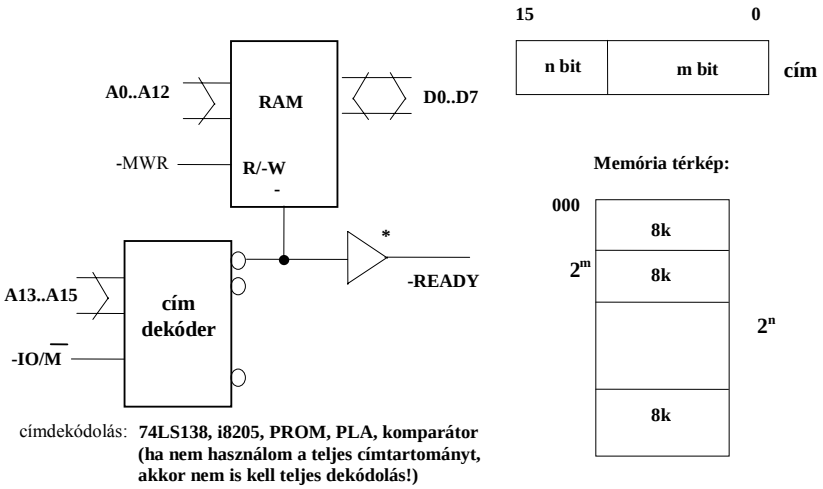
TC5516 2kb statikus RAM



6.5. ábra

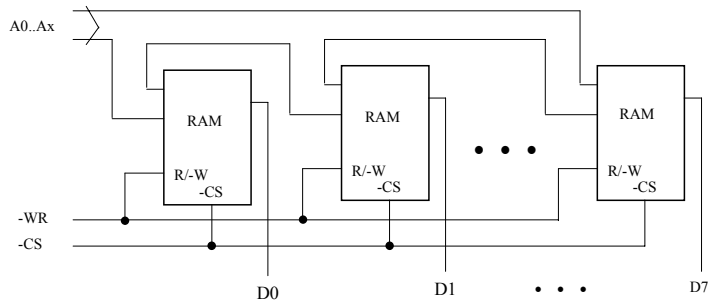
-CS, -CE: a chip-ek működésének engedélyezése, címdekódoló generálja
-OE: a kimeneti erősítők működésének engedélyezése, írás-olvasás vezérlő generálja (tipikusan az -MRD jel)
R/-W: beírás engedélyezése, írás-olvasás vezérlő generálja (tipikusan a -MWR jel)

3. Alapvető illesztés



6.6. ábra

4. Ha a memória adatszélessége keskenyebb (pl. 1bites RAM)



6.7. ábra

5. MSI elemkészlet

Tervezés során csak létező elemeket használjunk.(pl.: nincs 7 bemenetű NAND kapu)

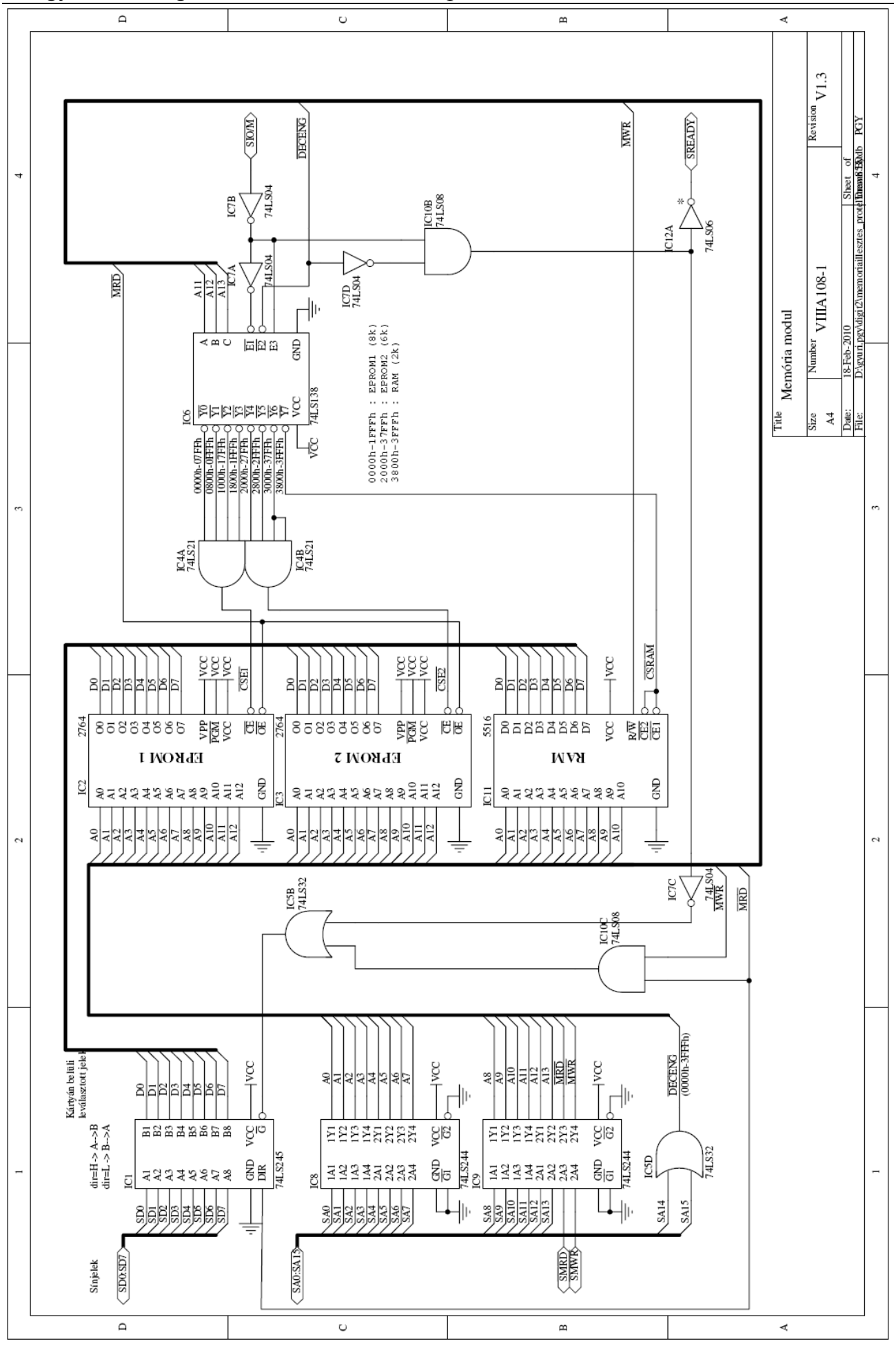
Elemkészlet:

Sorozat:	Belső elemszám:	Funkció:
74LS00	4	2 bemenetű NAND
74LS02	4	2 bemenetű NOR
74LS04	6	invereter
7406	6	OC inverter
7407	6	OC meghajtó
74LS08	4	2 bemenetű AND
74LS21	2	4 bemenetű AND
74LS32	4	2 bemenetű OR
74LS244	2x4	TRI STATE meghajtó
74LS245	1x8	kétirányú TRI STATE meghajtó
74LS373	1x8	DG flip-flop
74LS11	3	3 bemenetű AND

Példa 6.0.: 2 db 2764 EPROM és 1 db 5516 statikus RAM illesztése 8085 processzorhoz.
Memóriakiosztás: EPROM1: 0000h...1FFFh (8kB) EPROM2: 2000h...37FFh(6kB!)
RAM: 3800h...3FFFh (2kB)
Busz: SA0..SA15 címbusz (a címtárolás megtörtént) SD0...SD7 adatbusz
SRAM: nincs -OE vezérlőláb, leválasztó buffert alkalmazunk az adatbuszon (‘245)

A memória modul cím-térképe:

OP. MEM CPU	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	EPROM2 $A_{0.....}A_{12}$	RAM $A_{0.....}A_{10}$
	A /CS előállításához használt címvezetékek						
3FFF						Nem 1FFF	7FF
3800	**0**	**0**	**1**	**1**	**1**	Kihasznált 1800	000
3000	**0**	**0**	**1**	**1**	**0**	1000	
2800	**0**	**0**	**1**	**0**	**1**	0FFF 0800	
2000	**0**	**0**	**1**	**0**	**0**	07FF 0000	
1800	**0**	**0**	**0**	**1**	**1**	EPROM1 1FFF $A_{0.....}A_{12}$ 1800	
1000	**0**	**0**	**0**	**1**	**0**	17FF 1000	
0800	**0**	**0**	**0**	**0**	**1**	0FFF 0800	
0000	**0**	**0**	**0**	**0**	**0**	07FF 0000	


Memória modul
VIIIA108-1
V1.3
18-Feb-2010
D:\gyun_jegy\digit2\memoriailleszt
PCY
0000h-1FFFh : EPROM1 (8k)
2000h-37FFh : EPROM2 (6k)
3800h-3FFFh : RAM (2k)
EPROM 1
EPROM 2
RAM
74LS138
74LS21
74LS04
74LS08
74LS32
74LS245
74LS244
Kártyán belüli kiválasztott jelek
dir=H-> A-->B
dir=L-> B-->A
Sinjelek
SD0:SD7
SA0:SA15
SMRD
SMWR
DECENG
(0000h-3FFFh)
SREADY
MWR
MRD
CSEI
CSE2
CSRAM

Példa 6.1.:

2 db 2764 EPROM és 1 db 5565 statikus RAM illesztése 8085 processzorhoz.

Memóriakiosztás:

EPROM1:	C000h...D7FFh	(6kB!)
RAM:	D800h...E7FFh	(4kB)
EPROM2:	E800h...FFFFh	(6kB!)

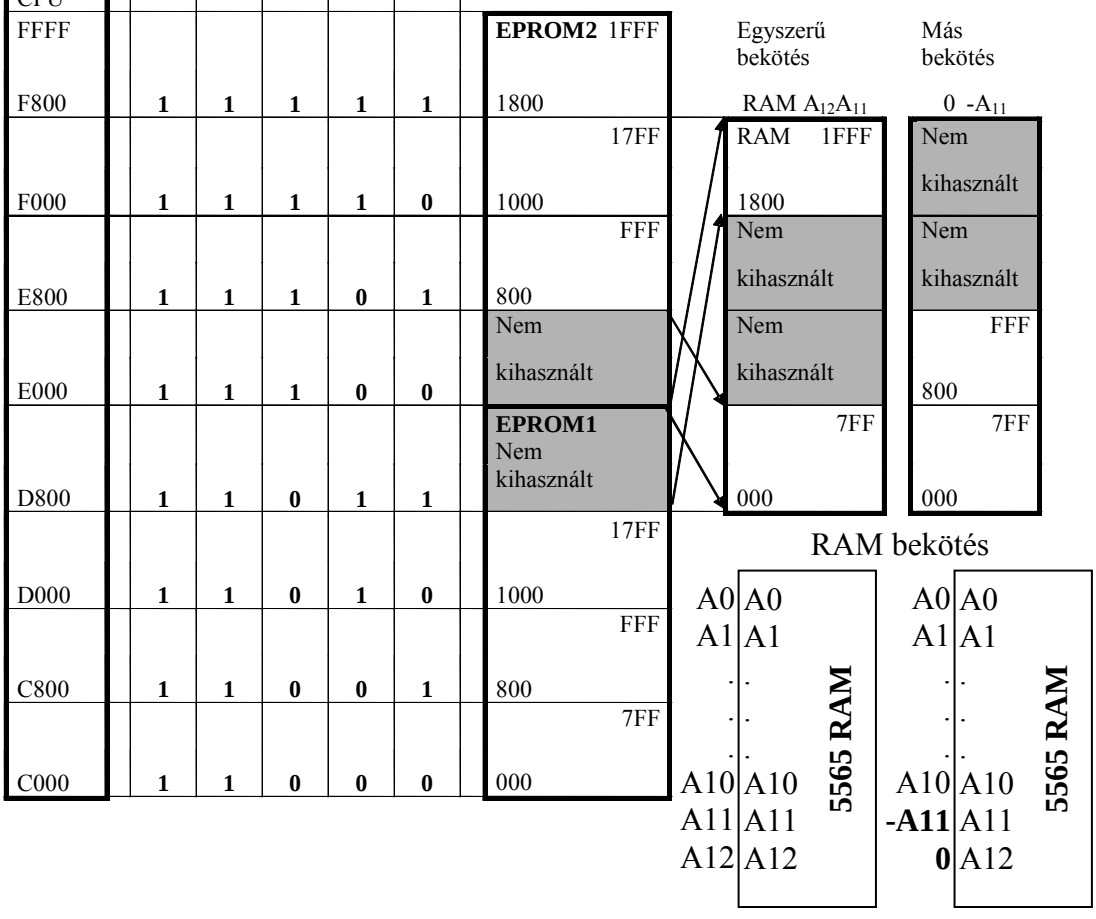
Busz: SA0..SA15 címbusz (a címtárolás megtörtént)
SD0...SD7 adatbusz

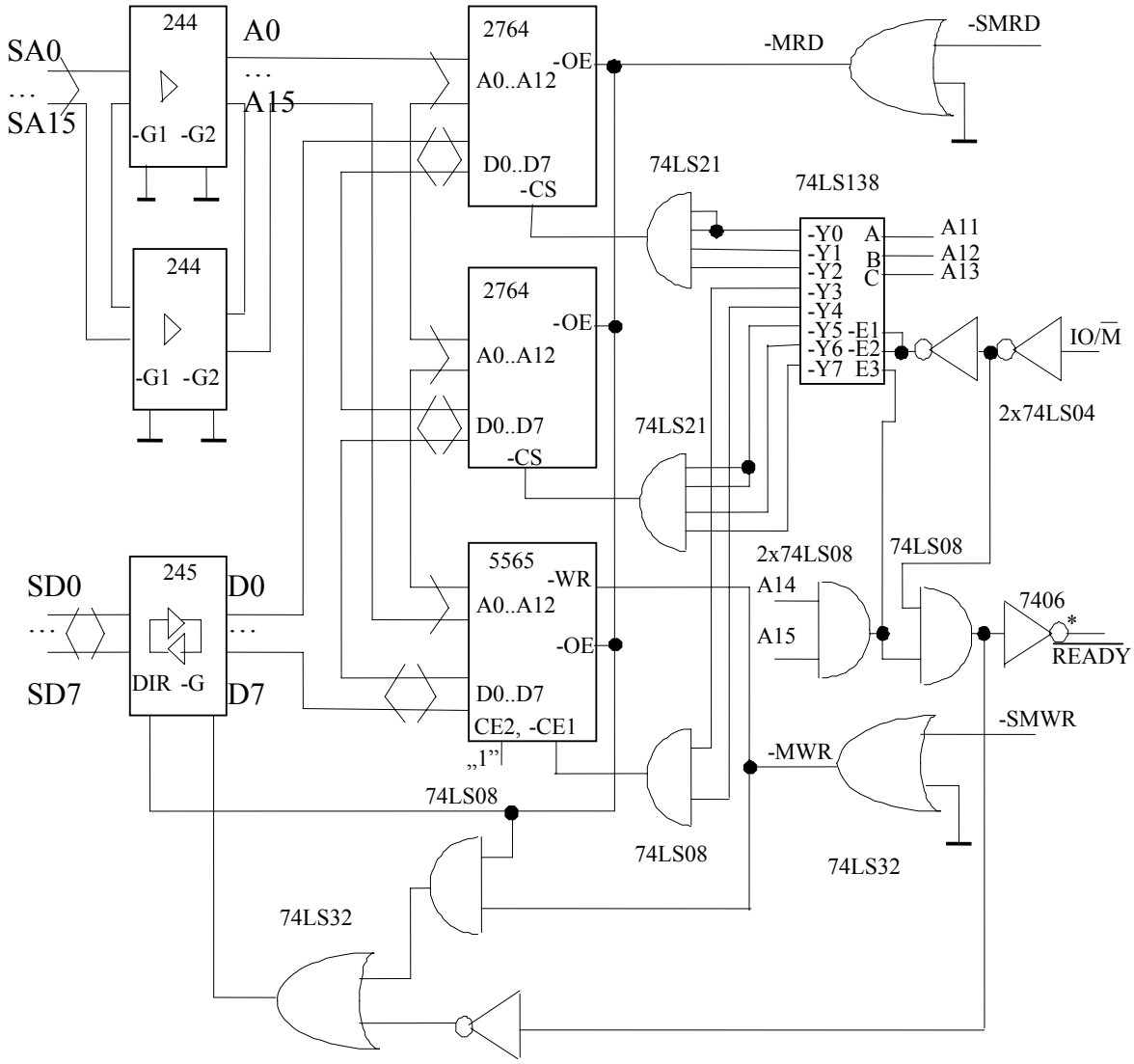
Megoldás 6.1.:

1. a felső 16 kB $A_{14} \cdot A_{15} \cdot \overline{IO / \overline{M}} = CS$,
2. a dekódolt 16kB-ot 8 db 2 kB-os blokkra osztjuk, majd ebből:
   - az első 3 EPROM
   - a második 2 RAM
   - a harmadik 3 EPROM
3. 74LS245 irányát -RD adja meg,
4. 74LS245 engedélyezése -G = -MRD ∗ -MWR + -CS,
5. a buszon csak egységnyi terhelés lehet, ezért 74LS244, 74LS32, 74LS04 áramköröket mint meghajtót alkalmazunk.

A memória modul cím-térképe:

OP. MEM CPU	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	
FFFF						**EPROM2** 1FFF
F800	**1**	**1**	**1**	**1**	**1**	1800
						17FF
F000	**1**	**1**	**1**	**1**	**0**	1000
						FFF
E800	**1**	**1**	**1**	**0**	**1**	800
						Nem kihasznált
E000	**1**	**1**	**1**	**0**	**0**	
						EPROM1 Nem kihasznált
D800	**1**	**1**	**0**	**1**	**1**	
						17FF
D000	**1**	**1**	**0**	**1**	**0**	1000
						FFF
C800	**1**	**1**	**0**	**0**	**1**	800
						7FF
C000	**1**	**1**	**0**	**0**	**0**	000





6.8. ábra

Példa 6.2.:

1 db 27128 EPROM és 1 db 5565 statikus RAM illesztése 8085 processzorhoz.

Megoldás 6.2.:

Memóriakiosztás:

EPROM: 0000h...3FFFh nem teljesen dekódolva

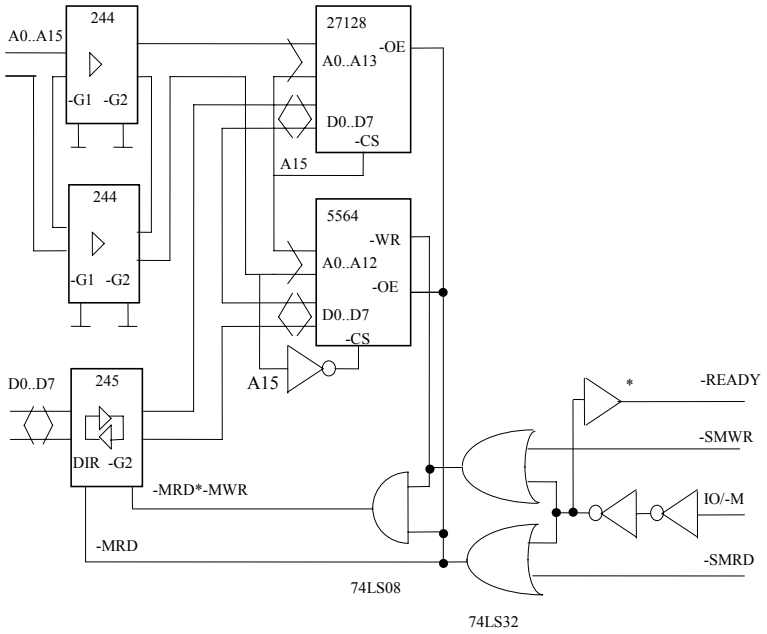
RAM: E000h...FFFFh nem teljesen dekódolva

A rendszerben nincs és nem is lesz több memória és a rendszerben csak közös –RD, -WR jel van. Ha a -CS függvénye a -WD,-RD-nek, az kedvezőtlenebb, mint a -OE vagy a leválasztás, mert a -CS-re lassabban reagál a RAM.

Nem teljesen dekódolt megoldás→ alsó 32kB RAM
felső 32kB ROM

A memóriák nem igényelnek WAIT állapotot, ezért a -READY jelet a legegyszerűbben az IO/-M jelből állítjuk elő.

LS 245-ös engedélyezése minden memória írásra, olvasásra megtörténik.

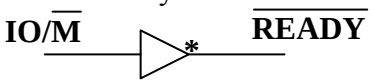


6.9. ábra

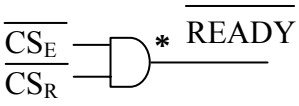
7. $\overline{READY}$ logika tervezése

Írás és olvasás esetén azonos módon működő vezérlés

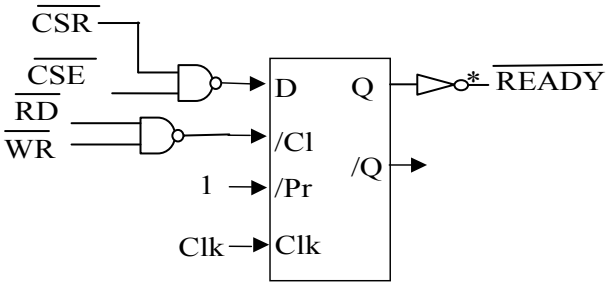
7.1. Legegyszerűbb eset, a rendszerben minden memória áramkör **0WAIT** állapottal működik mind **írás**, mind **olvasás** esetén, nem különböztetünk meg RAM/EPROM memóriát vagy címtartományokat.

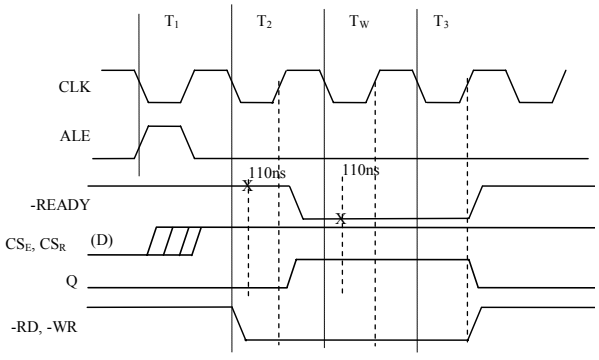


7.2. Tervezzen $\overline{READY}$ logikát, amellyel a RAM és az EPROM egyaránt 0 WAIT állapottal írható/olvasható.



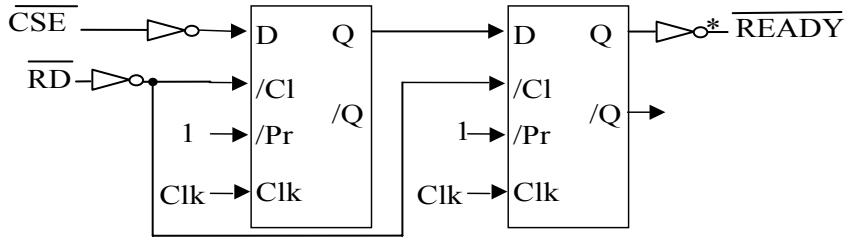
7.3. Tervezzen $\overline{READY}$ logikát, amellyel a **RAM** és az **EPROM 1** WAIT állapottal működik.





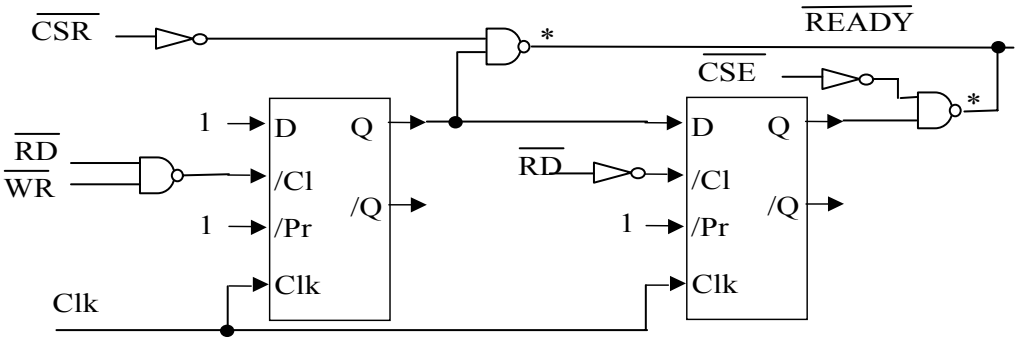
7.4. Tervezzen $\overline{READY}$ logikát, amellyel az **EPROM 2** WAIT állapottal működik.

Az EPROM miatt elegendő csak olvasásra működtetni az áramkört!



Megkülönböztetett írás/olvasás

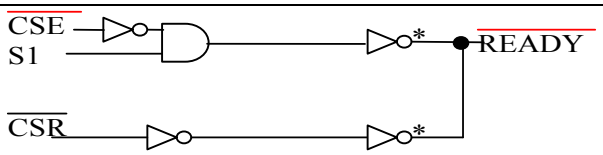
7.5. Tervezzen olyan $\overline{READY}$ áramkört, amely a $\overline{CSR}$ engedélyező jellel rendelkező **RAM** memória számára **íráskor és olvasáskor** 1, a $\overline{CSE}$ engedélyező jellel rendelkező **EPROM olvasáskor** 2 WAIT állapotot kér.



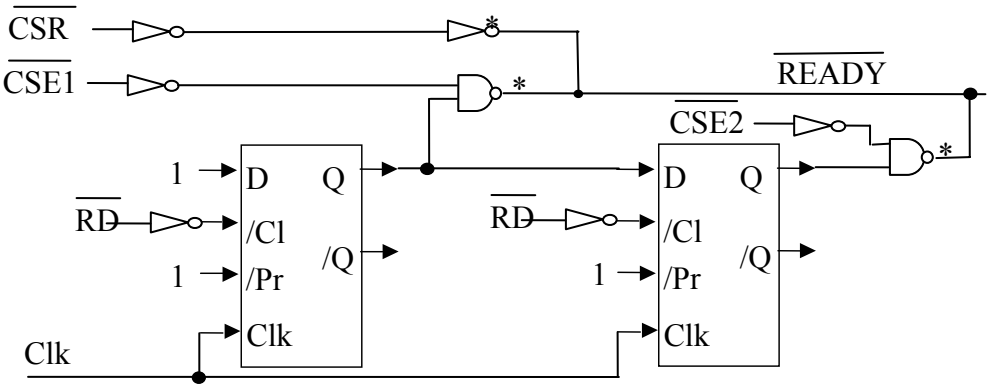
7.6. **Tervezzen** $\overline{READY}$ logikát, ha a RAM **olvasáskor és íráskor 0 WAIT** állapotot igényel! EPROM **csak olvasáskor 0 WAIT** állapottal működik írásakor nem ad $\overline{READY}$ jelet, amelyet S0,S1 állapotjelekkel oldjon meg!

(segédlet 18.o.)

IO/-M	S1	S0	Művelet
0	1	1	**OPkód FETCH**
0	1	0	**Mem. olvasás**
0	0	1	Mem. írás

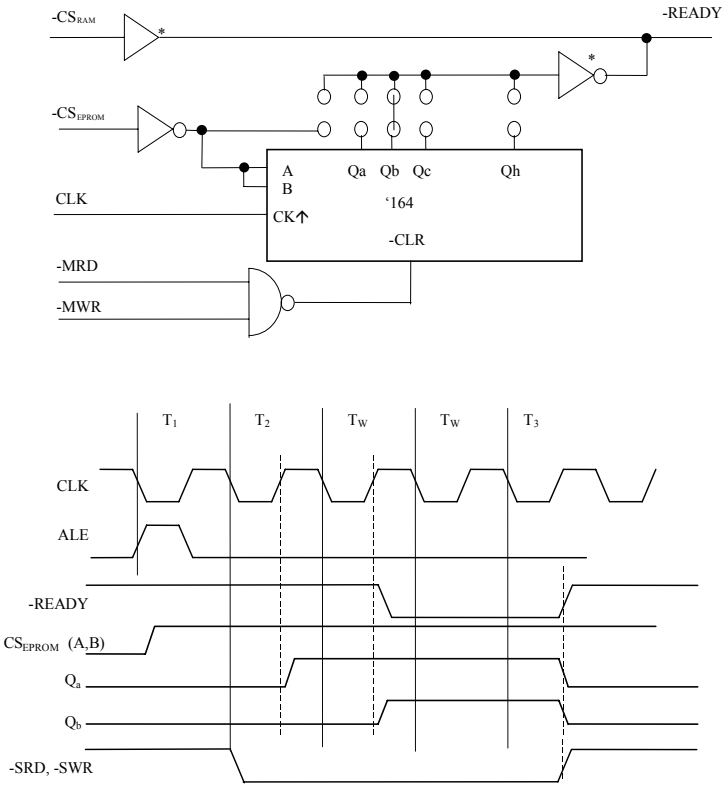


7.7. **Tervezzen** $\overline{READY}$ logikát, ha a RAM (CSR) **olvasáskor és íráskor 0 WAIT** állapotot igényel, az ROM1 (CSE1) **csak olvasáskor 1 WAIT** állapottal működik, az EPROM2 (CSE2) **csak olvasáskor** 2 **WAIT** állapottal működik írásakor egyik EPROM sem adjon $\overline{READY}$ jelet!



7.8. 2-nél több vagy állítható késleltetésű READY logika tervezése:

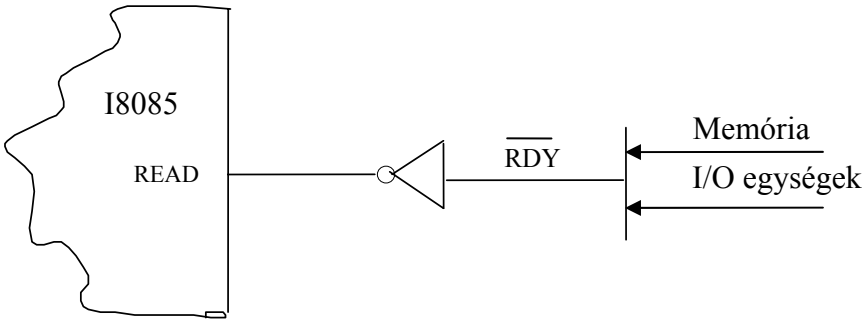
általános megjegyzések:
-lefutó él után nincs biztosítva elég idő a READY-hez pl.: 8085 felfutó órajel előtt legalább 110 ns
-a -RD és a -WR jel figyelembe vétele legalább 1 wait-et jelent



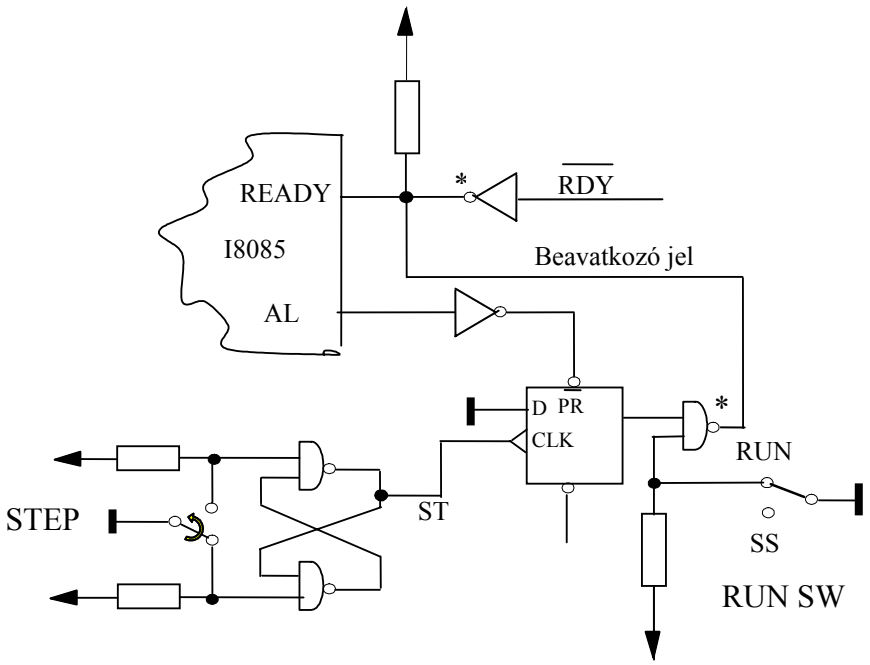
6.10. ábra

4. Pult illesztése 8085 processzoros rendszerekhez

Példa 7.1.: Tervezzen áramkört, amely alkalmas arra, hogy az i8085 processzor futását megállítsa, és egy STEP nyomógombbal gépi ciklusonkénti végrehajtására kényszerítse!
I8085 processzor READY jelének meghajtása:



7.1. ábra



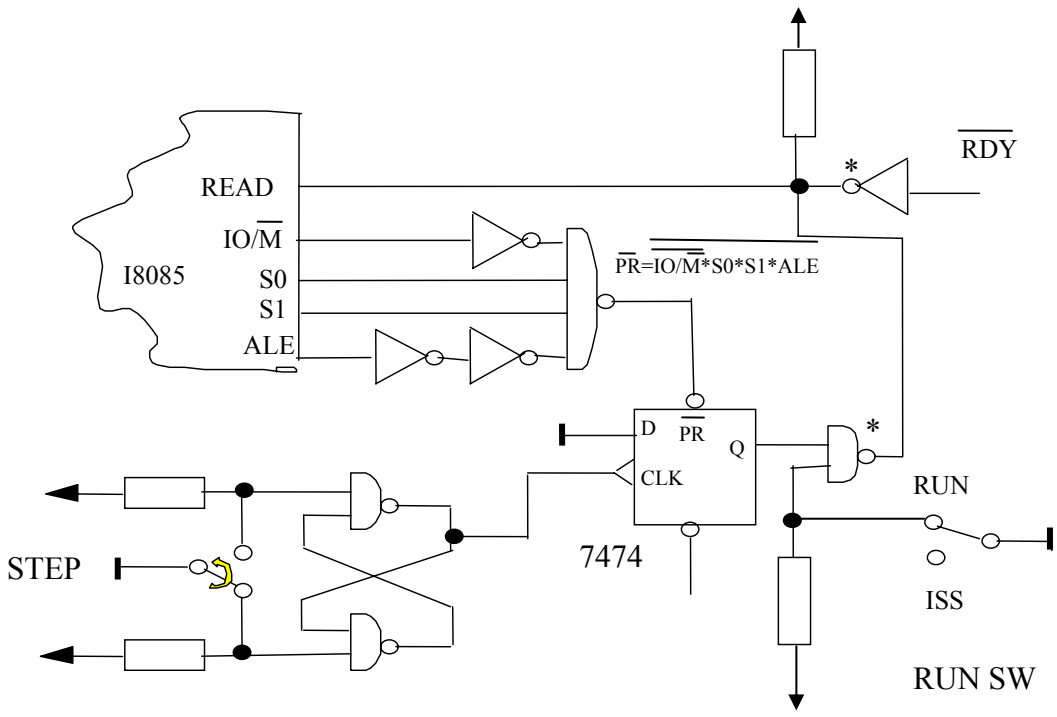
7.2. ábra

Működés: A RUN SW kapcsoló RUN állásban a D flip-flop kimenetről meghajtott NAND kapu kimenete állandóan „H” szinten van (READY). Ilyenkor az –RDY sín jelét a memória vagy az I/O egységek nyitott kollektoros inverterei vezérlik.
A RUN SW kapcsoló SS állásában a D flip-flop kimenete határozza meg a NAND kapu kimenetét. A D flip-flopot az ALE jel a -PR bemenetén keresztül „H” szintre billenti, amely a NAND kapun keresztül „L” szintű READY jelet generál. Ezért a processzor a futó gépi ciklusban WAIT állapotba kerül. Ezt az állapotot a STEP nyomógombbal lehet megszüntetni úgy hogy, a

pergésmentesített ST jel felfutó éle a D flip-flopot „L” szintre billenti. Ha -RDY sín jele „L” szintre kerül, a processzor kilép a WAIT állapotból. Az áramkör a következő gépi ciklus ALE jele után újra megállítja a processzor futását.

Példa 7.2.: Tervezzen áramkört, amely alkalmas arra, hogy az i8085 processzor futását utasításonként (fetch ciklusban) megállítsa és egy STEP nyomógombbal utasításonkénti végrehajtásra kényszerítse! IO/M, S1 és S0 jelek értékei meghatározzák a processzor gépi ciklusának típusát:

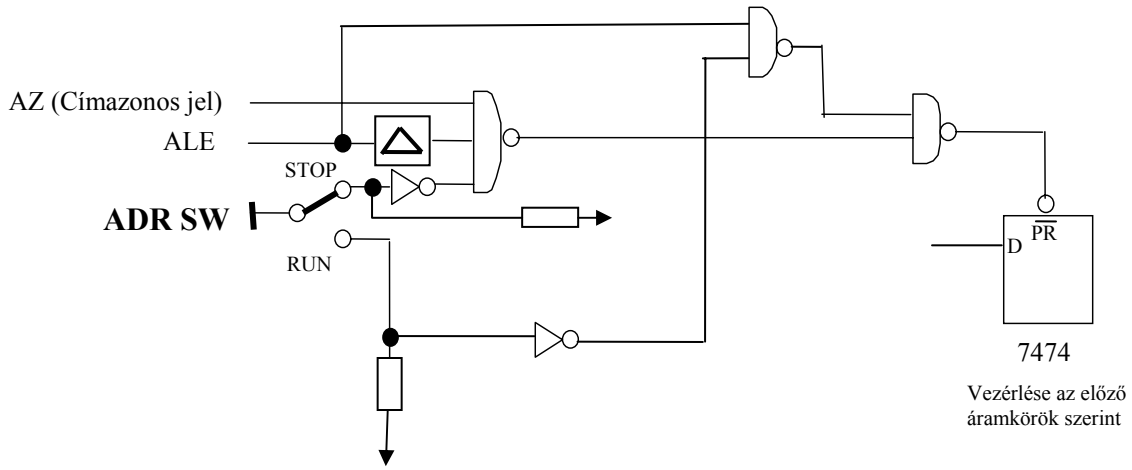
IO/M	S1	S0	Megjegyzés	WR	RD	INTA
TS	0	0	HALT	TS	TS	1
0	0	1	MEMW	0	1	1
0	1	0	MEMRD	1	0	1
0	1	0	*BUS IDLE (DAD)*	1	1	1
0	1	1	**FETCH**	1	0	1
1	0	1	IOW	0	1	1
1	1	0	IORD	1	0	1
1	1	1	INT ACK	1	1	0
1	1	1	*INA (Feles IT-k)*	1	1	1



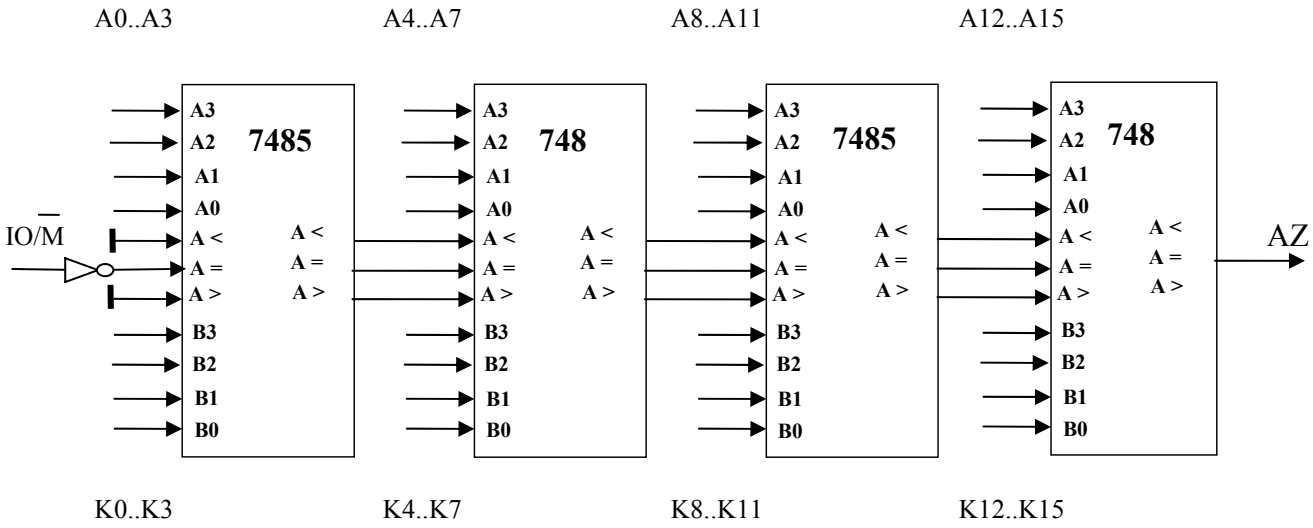
7.3. ábra

A 7.2. ábrán látható SS hálózat annyival bővül, hogy a D flip-flopot az –ALE helyett a -PR függvényt megvalósító kapu jele billenti „H” szintre. A -PR jelet az előző táblázat jelölt sora alapján lehet meghatározni.

Példa 7.3.: Tervezzen áramkört, amely alkalmas arra, hogy a i8085 processzor futását egy kapcsolósoron beállított memória címnél megállítsa, majd egy STEP nyomógomb segítségével gépi ciklusonkénti végrehajtást valósítson meg! Címre való megállást és a megállás utáni lépésenkénti végrehajtást az ADR kapcsoló STOP állása kényszeríti ki
A D flip-flopot vezérlő hálózat:



7.4. ábra



7.5. ábra

Példa 7.4.: Tervezzen áramkört, amely alkalmas arra, hogy az i8085 processzor futását

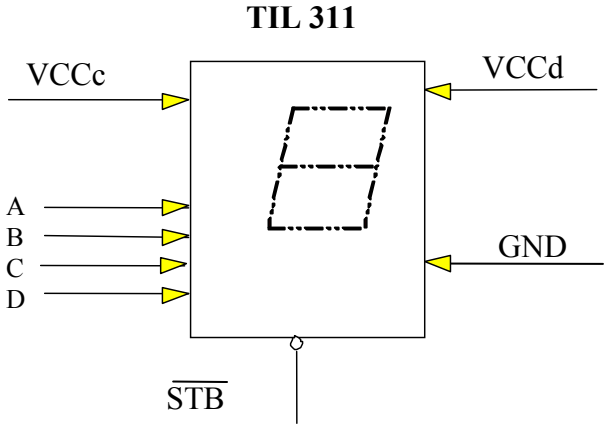
- gépi ciklusonként vagy
- utasításonként (fetch ciklusban) vagy
- beállított memória címnél

állítsa meg, majd egy STEP nyomógombbal lépésenkénti végrehajtásra kényszerítse (címre történő megállás után gépi ciklusonként). Kapcsolók: SS, ISS, ADR (ez prioritás sorrendet is jelent).

Példa 7.5.: Tervezze meg a 7.4 feladatra elkészített áramkör kiegészítését úgy, hogy az áramkör alkalmas legyen:

- a lépésenkénti végrehajtáskor az ADDRESS és DATA sínek hexadecimális megjelenítésére,
- a megállási cím beállítása egy DIS nyomógomb segítségével történjen.

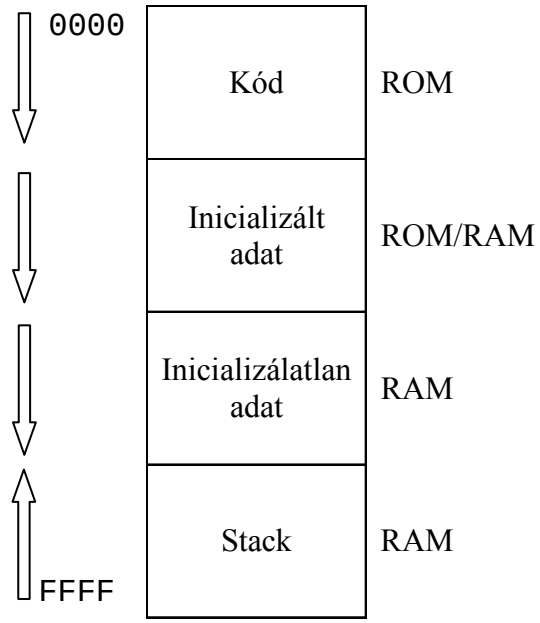
Használja a TIL 311-es kijelzőt



7.6. ábra

5-7. Assembly programozás

Memóriatérkép 8085 rendszer esetén



Kód terület: A működtető program utasításai. Futási időben értékük nem változhat.

Inicializált adat terület: Fordítási időben ismert kezdeti értékkel rendelkező adatok (pl. szövegek, konstansok, a program indulásakor érvényes alapértelmezett értékek).

Konstans: amely a futási időben nem változtatja meg az értékét. (ROM)

Inicializált adat: futási időben változhat az értéke (ROM=>RAM). A program indulásakor át kell tölteni RAM területre. Ez az inicializáló programrész feladata!

Inicializálatlan adat terület: Értékük fordítási időben nem ismert, vagy kezdeti értéke közömbös, és értékét futási időben változtathatja.

Stack terület: ideiglenes adatterület, az alacsonyabb címek felé terjeszkedik

Assembler direktívák segítségével adhatjuk meg, hogy a program melyik része milyen területre kerüljön.

ORG *cím*$_{16}$ A fordító a direktívát követő utasításokat/adatokat a *cím*$_{16}$ abszolút címtől kezdődően helyezi el.

CSEG [**PAGE**] A fordító a direktívát követő utasításokat/adatokat a kódterületen helyezi el. A pontos cím meghatározása a *szerkesztő* (*linker*) feladata. PAGE: a szerkesztő a területet 256-tal osztható kezdőcímre helyezi el.

DSEG [**PAGE**] A fordító a direktívát követő utasításokat/adatokat az adatterületen (tipikusan inicializált adatok) helyezi el. A pontos cím meghatározása a *szerkesztő* (*linker*) feladata.

BSEG [**PAGE**] A fordító a direktívát követő utasításokat/adatokat az adatterületen (tipikusan inicializálatlan adatok) helyezi el. A pontos cím meghatározása a *szerkesztő* (*linker*) feladata.

Adatterület definiálása:

DB adat$_8$ [string]	Kezdeti értékkel rendelkező byte (8 bit) méretű adatok, karakter sorozatok definiálása
DW adat$_{16}$	Kezdeti értékkel rendelkező szó (16 bit) méretű adatok definiálása
DS darab	Kezdeti értékkel rendelkező nem rendelkező *darab* méretű adatterület definiálása.

például:

DB	0,128	2 byte definiálása (decimális értékek)
DB	0FFh	1 byte definiálása (hexadecimális érték)
DB	'A'	1 byte definiálása (karakter)
DB	"Hello"	5 byte definiálása (karakter sorozat)
DW	1234h	1 szó definiálása (byte-ként definiálva: DB 34h,12h)
DS	16	16 byte adatterület lefoglalása (az értékek nem definiáltak)

Konstans definiálása:

konst **EQU** érték

például:

```
I8255A  EQU    0F0H
I8255B  EQU    I8255A+1
I8255C  EQU    I8255A+2
I8255V  EQU    I8255A+3
```

1. Adatmozgatás

Adatmozgató utasítások (nem állítják a flageket)

Címzési módok: közvetlen adat: MVI C,32 ; LXI H, 9000H
közvetlen adatcímzés: LHLD 9000H
indirekt regisztercímzés: STAX D ; MOV A,M
regiszter címzés: MOV A,C

Aritmetikai utasítások

Byte: INR, DCR
Szó: INX, DCX (nem állítják a flageket)

Logikai utasítások

ORA, ANA, XRA (az egyik operandus mindig az akkumulátor)

Elágazások

Feltétel nélküli: JMP
Feltételes: Jcc (Z,NZ,C,NC...)

a.) Írjunk programot, amely a 9000H címről **32 byte**-ot átmásol a 9800h címre (memcpy)

```
 LXI H,9000h     ; forrás mutató
 LXI D,9800h     ; cél mutató
 MVI C,32        ; darabszám - elfér 1 byte-on
ciklus:
 MOV A,M         ; másolás az akkumulátoron keresztül: forrás olvasása
 STAX D          ; cél írása
 INX H           ; forrás mutató növelése
 INX D           ; cél mutató növelése
 DCR C           ; darabszám csökkentése
 JNZ ciklus      ; ha nem nulla, vegyük a következőt
 ...
```

b.) Írjunk programot, amely a 9000H címről **32 szót** (2 byte) átmásol a 9800h címre (memcpy)

```
 LXI H,9000h     ; forrás mutató
 LXI D,9800h     ; cél mutató
 MVI C,32        ; darabszám - elfér 1 byte-on
ciklus:
 MOV A,M         ; másolás az akkumulátoron keresztül: forrás alsó byte olvasása
 STAX D          ; cél alsó byte írása
 INX H           ; forrás mutató növelése
 INX D           ; cél mutató növelése
 MOV A,M         ; forrás felső byte olvasása
 STAX D          ; cél felső byte írása
 INX H           ; forrás mutató növelése
 INX D           ; cél mutató növelése
 DCR C           ; darabszám csökkentése
 JNZ ciklus      ; ha nem nulla, vegyük a következőt
 ...
```

c.) Írjunk programot, amely a 9000H címről **512 byte**-ot átmásol a 9800h címre (memcpy)

```
 LXI H,9000h     ; forrás mutató
 LXI D,9800h     ; cél mutató
 LXI B,512       ; darabszám 2 byte-on fér el
ciklus:
 MOV A,M         ; másolás az akkumulátoron keresztül: forrás olvasása
 STAX D          ; cél írása
 INX H           ; forrás mutató növelése
 INX D           ; cél mutató növelése
 DCX B           ; darabszám csökkentése - nem állítja a flageket!!!
 MOV A,C
 ORA B           ; BC == 0?
 JNZ ciklus      ; ha nem nulla, vegyük a következőt
 ...
```

d.) Írjunk programot, amely a 9000H címről **0 végjelig** másol a 9800h címre (strcpy)

```
 LXI H,9000h     ; forrás mutató
 LXI D,9800h     ; cél mutató
ciklus:
 MOV A,M         ; másolás az akkumulátoron keresztül: forrás olvasása
 STAX D          ; cél írása
 INX H           ; forrás mutató növelése
 INX D           ; cél mutató növelése
 ORA A           ; az átmásolt adat 0?
 JNZ ciklus      ; ha nem nulla, vegyük a következőt
 ...
```

nem állítják a flageket

ezért bárhova helyezhető

2. Minta keresés

Írjunk programot, amely a 9000H címen elhelyezkedő **0 végjelű** karakterstringben megkeresi az első 'A' betűt. A program végén CY=1, ha nem talált.

```
 LXI H,9000h     ; forrás mutató
ciklus:
 MOV A,M         ; aktuális karakter olvasása
 ORA A           ; végjel vizsgálat
 JZ nem          ; nem talált
 CPI 'A'
 JZ talalt       ; ilyenkor cy=0!
 INX H           ; forrás mutató növelése
 JMP ciklus
nem:
 STC             ; cy = 1
talalt:
 ...
```

3. Maximum keresés

a.) Írjunk **szubrutint**, amely a HL regiszterpárban kapott címtől kezdődő a C regiszterben megadott hosszúságú memóriaterületen megkeresi a legnagyobb értéket (**byte méret, abszolút érték**). Megálláskor a HL regiszterpár a maximumra mutat. A szubrutin csak a HL regiszterpár értékét módosítja.

Az összehasonlítás művelet (CMP r) eredményét a következő módon kell értelmezni

JC -- Ugrás, ha A < r , **JNC --** Ugrás, ha A >= r, **JZ --** Ugrás, ha A = r

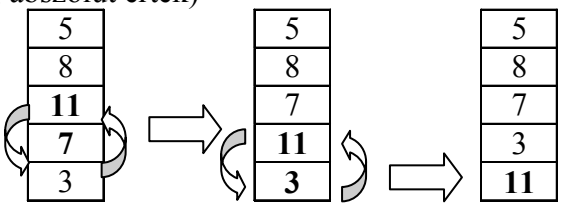
```
max_ker:
 PUSH B          ; regiszterek mentése: BC, DE, PSW
 PUSH D          ; regiszterek mentése: BC, DE, PSW
 PUSH PSW        ; (HL nem, mert ott lesz a visszatérési érték)

 MOV A,M         ; a maximum az első elem
 MOV D,H         ; megjegyezzük a mutatót
 MOV E,L         ; megjegyezzük a mutatót
 DCR C           ; darabszám csökkentés
 JZ  ezaz        ; csak 1 byte volt
ciklus:
 INX H           ; következő elem
 CMP M           ; összehasonlítás: A - M
 JNC kicsi       ; kisebb
 MOV A,M         ; nagyobb, ez lesz az új maximum érték
 MOV D,H         ; megjegyezzük a mutatót
 MOV L,E         ; megjegyezzük a mutatót
kicsi:
 DCR C           ; továbblépünk
 JNZ ciklus      ; még van
ezaz:
 XCHG            ; DE --> HL

 POP PSW         ; Regiszter visszaállítás
 POP D           ; fordított sorrendben!
 POP B
 RET             ; visszatérés!
```

b.) Írjunk **szubrutint**, amely a HL regiszterpárban kapott címtől kezdődő a C regiszterben megadott hosszúságú memóriaterületen megkeresi a legnagyobb értéket (**szó méret, abszolút érték**). Megálláskor a HL regiszterpár a maximumra mutat. A szubrutin csak a HL regiszterpár értékét módosítja.

```
max_ker2:
 PUSH B          ; regiszterek mentése: BC, DE, PSW
 PUSH D          ; regiszterek mentése: BC, DE, PSW
 PUSH PSW        ; (HL nem, mert ott lesz a visszatérési érték)

 PUSH H          ; megjegyezzük a mutatót (a stackben) - a legnagyobb elem az első elem
 MOV E,M         ; maximum alsó byte
 INX H           ;
 MOV D,M         ; maximum felső byte
 INX H           ; következő elem
 DCR C           ; darabszám csökkentés
 JZ  ezaz        ; nincs több
ciklus:
 INX H           ; következő elem felső byte
 MOV B,M         ; B = felső byte, mert lehet, hogy tárolnunk kell
 DCX H           ; mutató vissza az alsó byte-ra
 MOV A,B         ; A = felső byte, hogy komparálni tudjunk
 CMP D           ; összehasonlítás, először a felső byte: A - D
 JZ  also        ; egyenlő, az alsó byte dönt
 JNC nagy        ; az A a nagyobb, meg kell jegyezni
 JMP kicsi       ; kisebb, nem kell az alsó byte-ot vizsgálni
also:
 MOV A,M         ; alsó byte dönt
 CMP E           ; összehasonlítás, alsó byte: A - E
 JC  kicsi       ; az A a kisebb,  megyünk tovább
nagy:            ; nagyobb volt, megjegyezzük
 POP D           ; a régi mutatót eldobjuk ( ADD SP,2 - csak sajnos ilyen utasítás nincs)
 MOV D,B         ; megjegyezzük az értéket: felső byte (B)
 MOV E,M         ; megjegyezzük az értéket: alsó byte (HL mutatott rá)
 PUSH H          ; megjegyezzük a mutatót is
kicsi:
 INX H           ; továbblépünk
 INX H           ;
 DCR C           ;
 JNZ ciklus      ; még van
ezaz:
 POP H           ; a maximum helye a stackből

 POP PSW         ; Regiszter visszaállítás
 POP D           ; fordított sorrendben!
 POP B
 RET             ; visszatérés!
```

4. Rendezés (buborék rendezés)

Írjunk programot, amely a 9000H címtől kezdődő 32 byte-os memóriaterületen nagyság szerint növekvő sorrendbe rendezi az értékeket (byte méret, abszolút érték)



Emlékeztető: buborék rendezés C-ben

```
unsigned char a, b, *p;
for (i = 32; --i > 0; )
 for (j = i, p = KEZD; j > 0; j--) {
   a = *p; p++;
   if (*p < a)
     { b=*p; *p--=a; *p++=b; }
 }
```

```
 MVI C,32          ; tömb mérete ( i )
c1:                ; külső ciklus
 DCR C             ; (-- i )
 JZ allj           ; vége
 MOV B,C           ; még ennyi van (j = i )
 LXI H,9000H       ; itt kezdünk ( p = KEZD )
c2:
 MOV A,M           ; ( a = *p )
 INX H             ; következő elem ( p++ )
 CMP M             ; (*p < a )
 JC mehet          ; (*p > a ), nem kell csere
 JZ mehet          ; (*p = a ), nem kell csere
 MOV E,M           ; csere, a nagyobb hátrébb kerül ; ( b = *p )
 MOV M,A           ; (*p = a )
 DCX H             ; (p-- )
 MOV M,E           ; (*p = b )
 INX H             ; (p++ )
mehet:
 DCR B             ; továbblép (j--)
 JNZ c2            ; belső ciklus, még van elem
 JMP c1            ; külső ciklus
allj:
 ...
```

5. Aritmetika

a.) Írjunk programot, amely összead két, a memóriában elhelyezkedő szót (16 bit)
(9000h) = (9000h) + (9100h)

Összeadás az akkumulátoron keresztül

```
 LXI H,9000H     ; HL: op1 alsó byte címe
 LDA 9100H       ; A: op2 alsó byte
 ADD M           ; A = op1 + op2
 MOV M,A         ; eredmény alsó byte tárolása
 INX H           ; HL: op1 felső byte címe
 LDA 9101H       ; A: op2 felső byte
 ADC M           ; az átvitelt figyelembe kell venni
 MOV M,A         ; eredmény felső byte tárolása
```

Összeadás DAD utasítással

```
 LHLD 9000H      ; HL : op1 (alsó/felső byte)
 XCHG            ; DE : op1
 LHLD 9100H      ; HL : op2 (alsó/felső byte)
 DAD D           ; HL = HL + DE
 SHLD 9000H      ; eredmény tárolása
```

b.) Írjunk programot, amely összead két, a memóriában elhelyezkedő dupla szót (32 bit)
Az összeadást byte-onként kell elvégezni, az átvitel figyelembe vételével

```
 MVI C,4
 LXI H,9000H
 LXI D,9100H     ; az eddigi utasítások nem állították a flageket
 XRA A           ; cy = 0 állítás (ORA A, ANA A is jó lenne)
ciklus:
 LDAX D
 ADC M           ; az első körben cy=0
 MOV M,A         ; eredmény tárolása
 INX H           ; az operandus következő byte-ja
 INX D
 DCR C           ; cy nem valtozik!!!!
 JNZ ciklus
 ...
```

c.) Írjunk programot, amely a 9000H címtől kezdődő 32 byte-os memóriaterülethez modulo 256-os ellenőrző összeget képez.

```
 LXI H,9000H     ; kezdő cím
 MVI C,32        ; darabszám
 XRA A           ; az összeg kezdeti értéke 0 ( MVI A,0 is lehetne - 2 gépi ciklus!)
ciklus:
 ADD M           ; modulo 256, a cy nem kell (az eredmény az A-ban)
 INX H
 DCR C
 JNZ ciklus
 ...
```

d.) Írjunk programot, amely a 9000H címtől kezdődő 32 byte-os memóriaterülethez 16 bites ellenőrző összeget képez.

A 16 bites ellenőrző összeg alsó byte-ját az **A**, felső byte-ját a **B** regiszterben képezzük, végül az alsó byte-ot áttöltjük a **C** regiszterbe

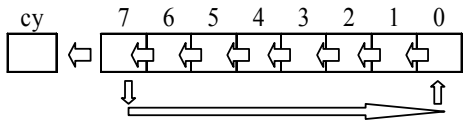
```
 LXI H,9000H      ; kezdő cím
 MVI C,32         ; hossz
 XRA A
 MOV B,A          ; az ellenőrző összeg kezdeti értéke 0
ciklus:
 ADD M            ; alsó byte
 JNC c1
 INR B            ; ha volt átvitel az alsó byte-on, a felső byte-ot növelni kell
c1:
 INX H            ; következő byte címe
 DCR C
 JNZ ciklus       ; van még
 MOV C,A          ; az összeg alsó byte-ja a C regiszterbe
 ...
```

6. Logikai műveletek

Írjunk programot, amely a 9000H címtől kezdődő 32 byte-os memóriaterületen megszámolja, hogy hány darab 1-es bit van.

```
 LXI H,9000H      ; kezdő cím
 MVI C,32         ; hossz
 LXI D,0          ; számláló
c1:
 MVI B,8          ; ennyi bit van egy byte-ban
 MOV A,M          ; aktuális byte az A-ba
c2:
 RLC              ; a bitek vizsgálata a cy-n keresztül
 JNC c3           ; nincs cy, nem kell számolni
 INX D
c3:
 DCR B            ; van még bit?
 JNZ c2
 INX H            ; következő cím
 DCR C            ; van még byte?
 JNZ c1
 ...
```

RLC



7. Bináris - HEXA konverzió

Írjunk szubrutin, amely előállítja az akkumulátorban lévő bináris szám hexadecimális karakteres megfelelőjét a BC regiszterpárba. pl: A=1Eh => B='1' (31h), C='E' (45h). A szubrutin az **A** regiszter és a flag-ek értékét elrontja.

Elv: ha a konvertálandó 4 bit (b) 0 ... 9, akkor a hexa karakter = b + 30h
ha a konvertálandó 4 bit (b) A ... F, akkor a hexa karakter = b + 37h ('A' = 41H)

```
bin2hexa:
 PUSH PSW          ; A mentése, mert először csak 4 bit kell
 ANI 0FH           ; alsó 4 bit (felső 4 bit törlése)
 ADI 30H           ; karakter
 CPI 3AH           ; nagyobb, mint '9'
 JC  c1
 ADI 7             ; még adjunk hozzá 7-et
c1:
 MOV C,A           ; alsó 4 bit hexa karaktere a C-be
 POP PSW           ; A visszaállítása
 RLC               ; a felső 4 bit kell, először odaforgatjuk
 RLC
 RLC
 RLC
 ANI 0FH           ; most már alul van, törölhetjük a felső 4-et
 ADI 30H
 CPI 3AH
 JC c2
 ADI 7
c2:
 MOV B,A           ; felső 4 bit hexa karaktere a C-be
 RET
```

DAA utasítással

```
bin2hexa:
 PUSH PSW
 ANI 0FH
 ADI 90H
 DAA
 ACI 40H
 DAA
 MOV C,A
 POP PSW
 RLC
 RLC
 RLC
 RLC
 ANI 0FH
 ADI 90H
 DAA
 ACI 40H
 DAA
 MOV B,A
 RET
```

8. Programmal ellenőrzött készenlét

a.) A státusz egy bitjén 1-es értékre várunk

```
var1:
 IN 00H            ; státus beolvasás
 ANI 80H           ; legfelső bit kimaszkolása
 JZ var1           ; várunk amíg 0
 ...
```

b.) A státusz két bitjén 1-es értékre várunk

```
var2:
 IN 00H            ; státus beolvasás
 ANI 0C0H          ; felső két bit kimaszkolása
 CPI 0C0H          ; várunk
 JNZ var2          ; amíg mindkettő nem 1-es
 ...
```

c.) A státusz két bitjének valamelyikén 1-es értékre várunk

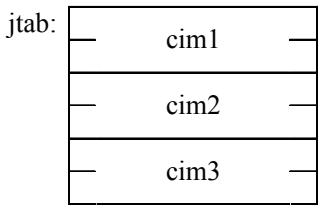
```
var:
 IN 00H          ; státus beolvasás
 ANI 081H        ; alsó és felső bit maszkolása
 JZ var          ; várunk, ha nincs 1-es
 ...
```

d.) A státusz alsó két bitjének valamelyikén 1-es értékre várunk, elágazás a bitek értéke szerint.

```
var:
 IN 00H          ; státus beolvasás
 ANI 3H          ; alsó két bit kimaszkolása
 RRC             ; az alsó bit a cy-be (a másik bit értékét sem ronthatjuk el)
 JC also         ; az alsó 1-es volt (prioritás!, mert először ezt vizsgáltuk)
 ANI 1           ; alsó bit 1-es?
 JZ var          ; nem, tehát várni kell
....             ; a második bit 1-es volt
 ...
also:
....             ; az első bit 1-es volt
 ...
```

e.) elágazás a státus alsó 2 bitje alapján ugró táblával (01 → cim1, 10 → cim2, 11 → cim3)

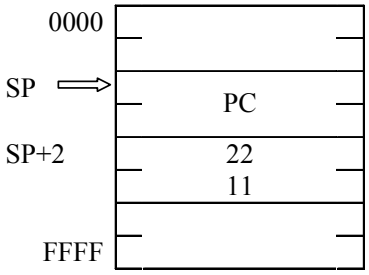
```
var:
 IN 00H          ; státus beolvasás
 ANI 3H          ; alsó két bit kimaszkolása
 JZ var          ; ha nulla várunk
 DCR A           ; A = A - 1
 ADD A           ; A = 2 * A
 MOV E,A         ; DE = offset
 MVI D,0         ;
 LXI H,jtab      ; ugró tábla kezdőcíme
 DAD D           ; (HL) = ugrási cím
 MOV E,M         ; cím alsó byte
 INX H           ;
 MOV D,M         ; DE = ugrási cím
 XCHG            ; HL = ugrási cím
 PCHL            ; PC = HL (indirekt  ugrás)

jtab:
 DW  cim1, cim2, cim3
```



9. Szubrutin paraméter átadás

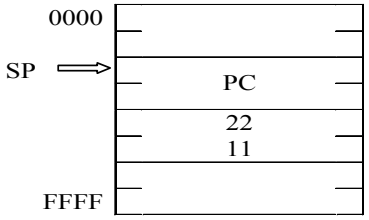
a.) Paraméter átadás a stack-en

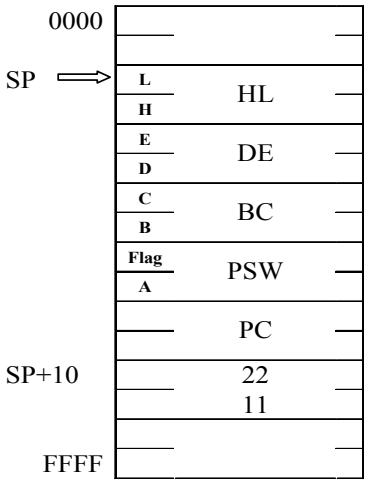
```
 LXI H,1122H   ; ezt az értéket akarjuk átadni
 PUSH H        ; letesszük a stack-be
 CALL rutin    ; szubrutin hívás
 POP H         ; SP visszaállítás
 ...

rutin:
 LXI H,2       ; SP értékéhez csak így tudunk hozzájutni
 DAD SP        ; egyben átlépjük a visszatérési címet
 MOV E,M       ; az átadott paraméter felolvasása
 INX H
 MOV D,M       ; paraméter DE-be
...            ; a szubrutin felülírja a DE regisztert
 RET           ; az általa használt regiszterek értékét
```



b.) Paraméter átadás a stack-en, a szubrutin menti a regisztereket

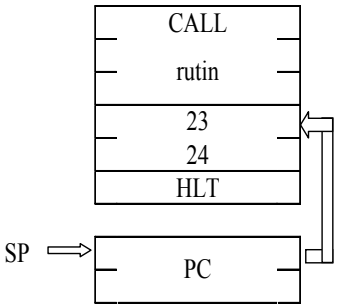
```
 LXI H,1122H   ; ezt az értéket akarjuk átadni
 PUSH H        ; letesszük a stack-be
 CALL rutin    ; szubrutin hívás
 POP H         ; SP visszaállítás
 ...

rutin:
 PUSH PSW      ; regiszter mentés
 PUSH B        ; a sorrend tetszőleges
 PUSH D        ; de a visszaállításnál figyelni kell
 PUSH H
 LXI H,10      ; SP értékéhez csak így tudunk hozzájutni
 DAD SP        ; egyben átlépjük a mentett értékeket
 MOV E,M       ; az átadott paraméter felolvasása
 INX H
 MOV D,M       ; paraméter DE-be
...
 POP H         ; regiszter visszaállítás, fordított sorrendben
 POP D
 POP B
 POP PSW
 RET
```





b.) Paraméter átadás a szubrutin hívó utasítás után a memóriában

```
CALL rutin      ; szubrutin hívás
DB 23, 24       ; paraméterek
...

rutin:
XTHL            ; paraméterek címe a stack tetején
MOV E,M         ; 23 => E
INX H
MOV D,M         ; 24 => D (paraméter DE-be)
INX H
XTHL            ; új visszatérési cím
...
RET
```



10. Állapotgép

Írjunk programot, amely a 9000H címtől kezdődő 0 végjelű karakterstringben megszámolja az 'LY'-okat. (az LLY, LLLY, L....LY sorozatokat 2-nek számolja)

	L	**Y**	**más**	**'\0'**
alap (0)	egyl	alap	alap	--
egyl (1)	ketl	alap/n++	alap	--
ketl (2)	ketl	alap/n+=2	alap	--

```
for (p=KEZD,n=0,b=0; *p!=0 ; p++) {
  switch (b) {
    case 0: if (*p == 'L') b++; break;
    case 1: if (*p == 'L') b++; else {b = 0; if (*p == 'Y') n++;} break;
    case 2: if (*p != 'L') { b = 0; if (*p == 'Y') n+=2 }; break;
  }
}
```

```
; HL: mutató (p)
; DE: számláló (n)
; B: állapot (b)
   LXI H,9000H     ; kezdeti értékadások (p=KEZD)
   LXI D,0         ; itt számolunk (n=0)
   MVI B,0         ; alap állapot (b=0)

cikl:
   MOV A,M
   ORA A           ; végjel vizsgálat
   JZ vege         ; *p!=0
   MOV A,B         ; switch(b)
   ORA A
   JNZ sw1

   MOV A,M         ; 0: (alap állapot)
   CPI 'L'         ; 'L' jött?
   JNZ tovabb
   INR B           ; b=1 (egyl állapot)
   JMP tovabb
sw1:
   DCR A
   JNZ sw2

   MOV A,M         ; 1: (egyl állapot)
   CPI 'L'
   JZ ketl         ; egyl állapotban 'L' jött -> átlépés ketl állapotba
   MVI B,0         ; vissza alap állapotba (b=0)
   CPI 'Y'
   JZ novel1       ; 'Y' jött, a számlálót is növelni kell
   JMP tovabb
ketl:
   INR B
   JMP tovabb

sw2:
```

```
  MOV A,M         ; 2: (ketl állapot)
  CPI 'L'
  JZ tovabb       ; marad az állapot
  MVI B,0         ; vissza alap állapotba (b=0)
  CPI 'Y'         ; *p=='Y' ?
  JNZ tovabb
  INX D           ; számláló növelése (a következő utasítással együtt: n+=2)
novel1:
  INX D           ; számláló növelése (n++)
tovabb:
  INX H           ; mutató növelése (következő karakter)
  JMP cikl
vege:
  ...
```

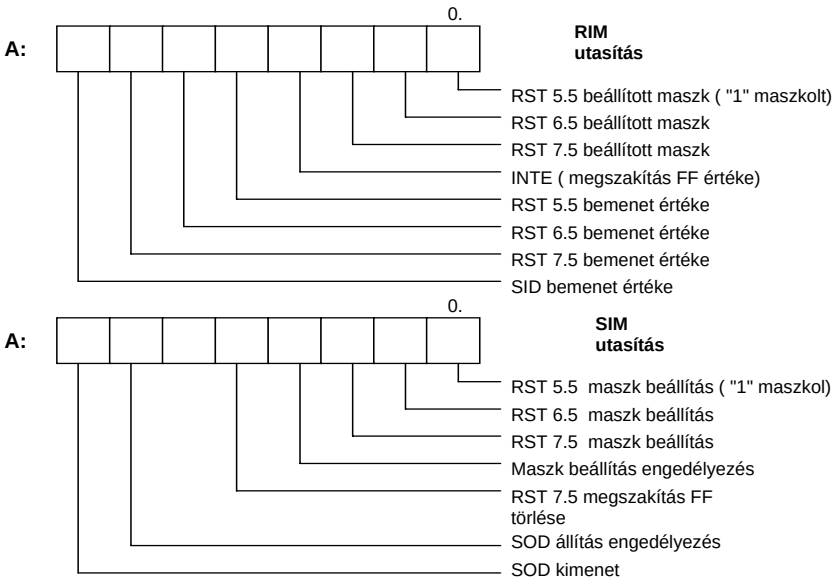
Egy másik megközelítés:

```
for (p=KEZD,n=0,b=0; *p != 0 ; p++)
  switch (*p) {
    case 'L': if (b != 2) b++; break;
    case 'Y': n+=b;  // nincs break!!!! b=0
     default: b=0; break;
   }
```

```
 LXI H,9000H
 LXI D,0
 MVI B,0

ciklus:
 MOV A,M
 ORA A
 JZ vege

 CPI 'L'         ;'L':
 JZ ljott

 CPI 'Y'         ;'Y':
 JZ yjott

def1:
 MVI B,0         ;default

tovabb:
 INX H           ;p++
 JMP ciklus
```

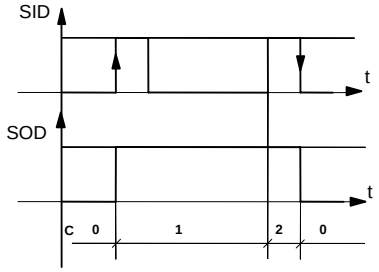
```
ljott:
 MOV A,B
 CPI 2
 JZ tovabb
 INR B
 JMP tovabb

yjott:
 MOV A,B         ;n+=b
 ADD E
 MOV E,A
 JNC def1
 INR D
 JMP def1

vege:
 ...
```

8. SID, SOD vonalak alkalmazása

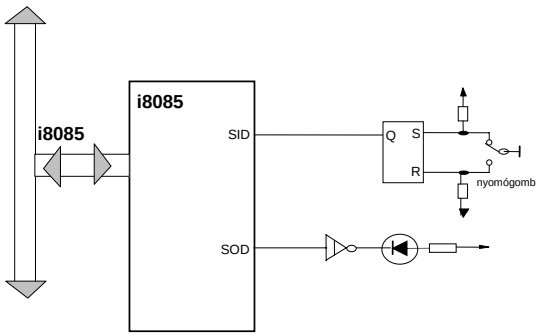


8.1. ábra

Példa 8.1.: A 8.1. ábra szerint egy i8085-ös processzor SID bemenetére pergésmentesített kapcsolót kötöttek.A SOD kimenetről egy LED diódát lehet vezérelni. A processzoron futó programnak a SID bemenetre érkező jeleket kell figyelnie és ez alapján vezérelni a LED diódát. A bekapcsolás után a SID bemenetre érkező jel első felfutó élre a diódának világítani kell, a bemenetre érkező következő impulzus lefutó élére pedig el kell aludnia (8.2. ábra). Készítsen el egy olyan assambly programot, amely ezt a vezérlést ciklikusan ismétli. Aprogram címe legyen 8000H.



8.2. ábra

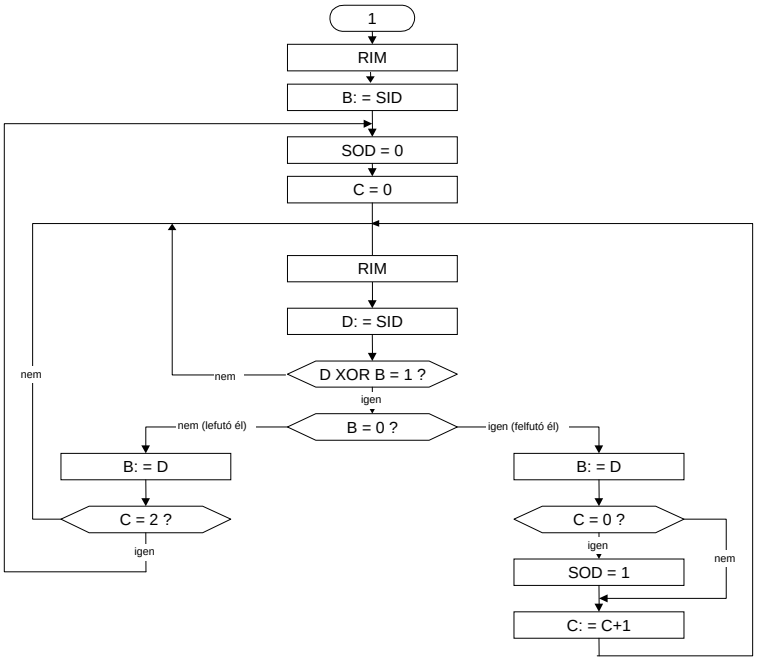


8.3. ábra

Megoldás 8.1.:

A programban alkalmazott regiszterek feladata:

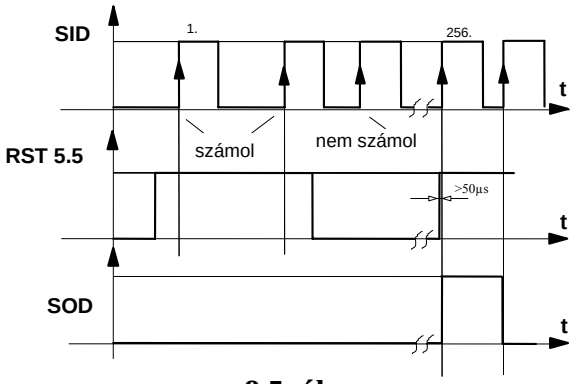
B: Előző mintavételezéskor a SID bemenet értéke,
C: Ciklus állapot-tárolója,
D: Aktuális SID érték



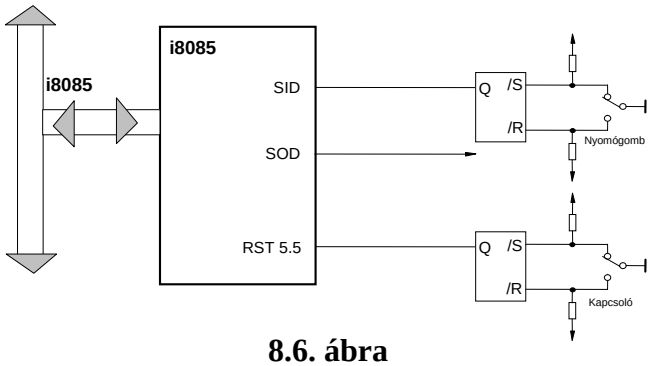
8.4. ábra

```
                ORG     8000H
KEZD:           RIM
                ANI     80H
                MOV     B,A
;---------------------------------------------------------------
FOCIKL:         MVI     A,40H
                SIM
                MVI     C,00H           ;STATUS:=0
;---------------------------------------------------------------
CIKL1:          RIM
                ANI     80H             ;FELSO BITRE MASZK
                MOV     D,A             ;UJ ERTEK MENTESE
                XRA     B               ;A XOR B, FIGYEL
                JZ      CIKL1           ;UGRIK, HA SID NEM VALTOZOTT
                XRA     A               ;A:=0
                ORA     B               ;0+B
                JZ      FELFUT          ;UGRIK, HA FELFUTÓ EL
;---------------------------------------------------------------
LEFUT:          MOV     B,D             ;REGI SID ERTEK FRISSITESE
                MOV     A,C
                CPI     2
                JZ      FOCIKL          ;UGRIK, HOGY SOD LEESSEN
                JMP     CIKL1
;---------------------------------------------------------------
FELFUT:         MOV     B,D             ;REGI SID ERTEK FRISSITESE
                XRA     A
                ORA     C
                JNZ     FEL1            ;UGRIK,HA NEM ELSO EL
                MVI     A,0C0H
                SIM                     ;SOD=1
FEL1:           INR     C               ;C:=C+1
                JMP     CIKL1
;---------------------------------------------------------------------------
                END
```

Példa 8.2.: Mod. 256-os számláló i8085-ös processzorral. A processzor RST 5.5 bemenete a számláló engedélyező jele, és a SID bemenetre érkeznek az impulzusok. Az áramkör az impulzusok felfutó élére működjön mod. 256 számlálóként. Az áramkör kimenete a processzor SOD kimenete, amelyen 256 impulzusonként egy ciklusidőre jelenik meg pozitiv impulzus. Az bemenő impulzusok szélessége nagyobb mint 0.5 ms és a két bemenet változása egymástól 50µs-nál nagyobb távolságra van.

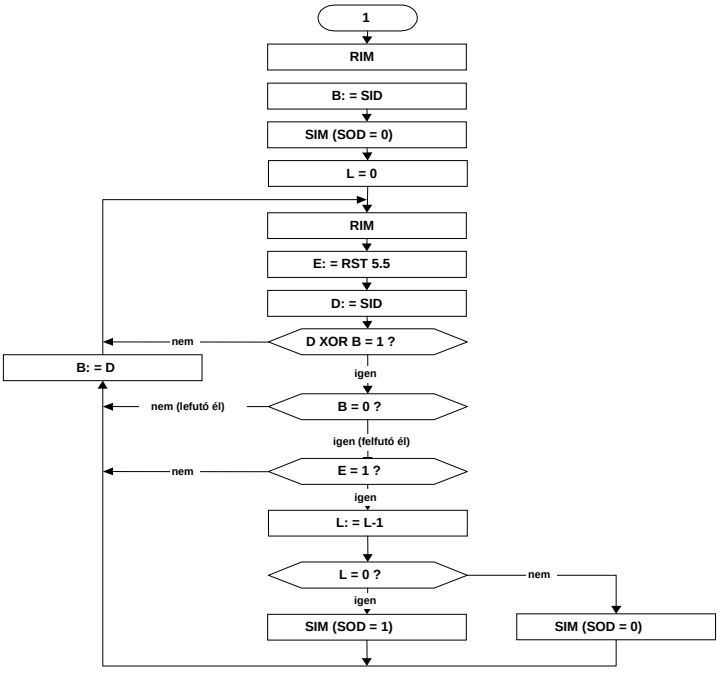


8.5. ábra



8.6. ábra

Megoldás 8.2./1: Az RST 5.5 vonal programozott lekérdezésével.



8.7. ábra

A programban alkalmazott regiszterek feladatai:

B: Előző mintavételi SID érték.

D: Aktuális SID érték.

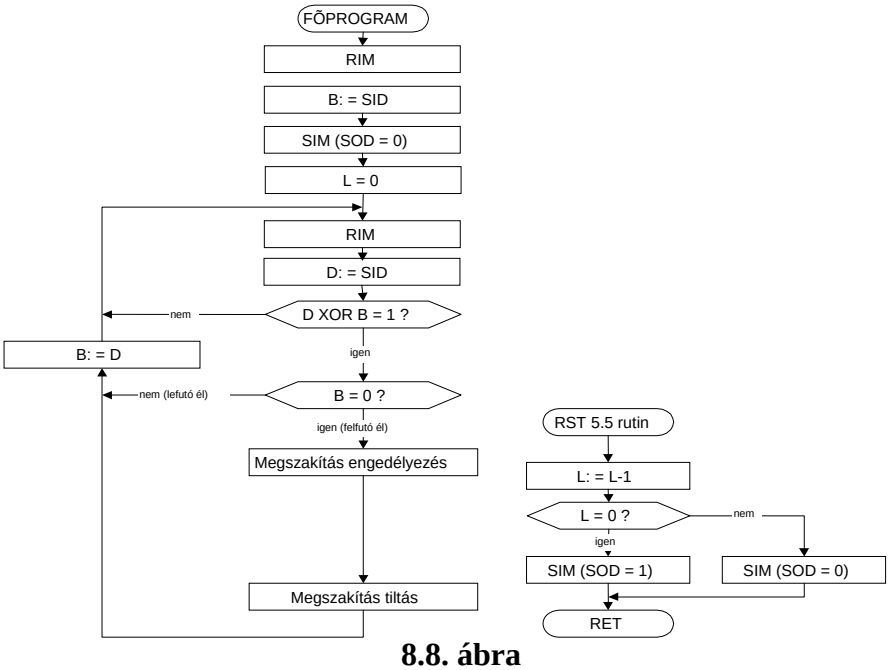
E: Aktuális RST 5.5 érték.

L: Számláló értéke.

Egyidőben történik a mintavételezése a SID és RST 5.5 bemeneteknek, ezért a jelek elvileg egyidőben is változhatnak (az 50µs-os idő a két változás között nincs kihasználva).

```
                ORG     8000H
INIC:           RIM
                ANI     80H
                MOV     B,A             ;ELOZO SOD BEALLITASA A B-BE
                MVI     A,4FH           ;FELES IT-K TILTASA, SOD=0
                SIM
                XRA     A
                MOV     L,A             ;L=0
;-------------------------------------------------------------------------
KEZD:           RIM
                PUSH    PSW
                ANI     10H
                MOV     E,A             ;E=RST 5.5
                POP     PSW
                ANI     80H
                MOV     D,A             ;D=AKTUALIS SID
                XRA     B               ;VOLT SID VALTOZAS ?
                JZ      KEZD            ;UGRIK, HA NEM VOLT
                XRA     A
                ORA     B               ;ELOZO SOD „0” VOLT ?
                JZ      FELFUT          ;URIK, HA FELFUTO EL
;-------------------------------------------------------------------------
LEFUT:          MOV     B,D             ;ELOZO ERTEK FRISSITESE
                JMP     KEZD
;-------------------------------------------------------------------------
FELFUT:         XRA     A
                ORA     E               ;KELL SZAMOLNI?
                JZ      LEFUT           ;UGRIK, HA NEM KELL (RST 5.5=0)
                DCR     L
                JNZ     TORL            ;UGRIK, HA NEM A256. IMP.
                MVI     A,0C0H          ;SOD=1 PARANCS SZO
                SIM
                JMP     LEFUT
TORL:           MVI     A,40H           ;SOD=0 PARANCS SZO
                SIM
                JMP     LEFUT
                END
```

Megoldás8.2./2: RST 5.5 megszakításkérés alkalmazásával.



8.8. ábra

Az RST 5.5 megszakításkérő jel szintjére történik meg a szubrutin meghívása, akkor ha a SID vonal felfutó élére a megszakítások tiltása megszünik (EI). Az RST 5,5 szubrutinban dekrementálás és SOD állítása történik.

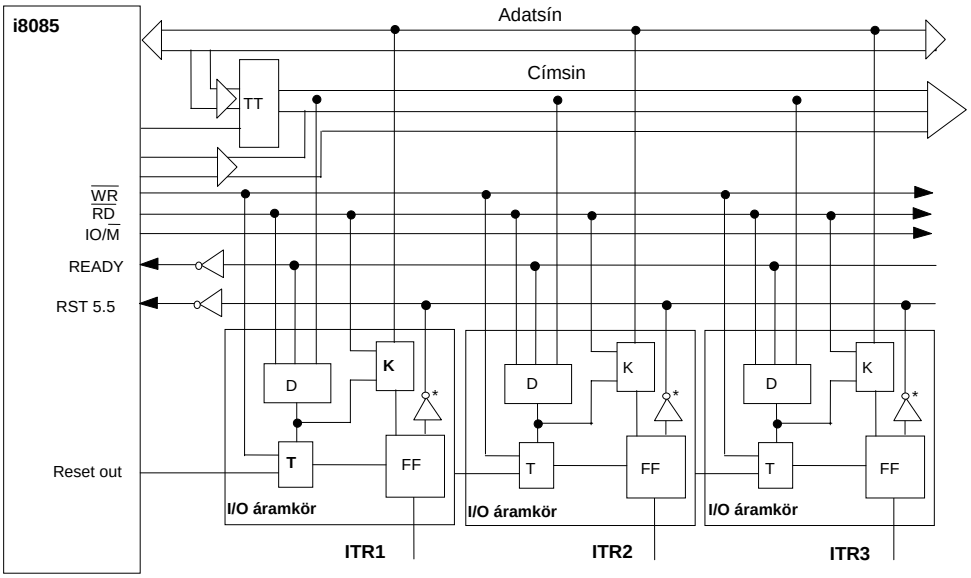
```
                ORG     0H
;
                JMP     INIC
;-------------------------------------------------------------------------------
                ORG     2CH
;
RUTIN:          DCR     L
                JNZ     TORL            ;UGRIK, HA NEM A256. IMP.
                MVI     A,0C0H          ;SOD=1 PARANCS SZO
                SIM
                RET
TORL:           MVI     A,40H           ;SOD=0 PARANCS SZO
                SIM
                RET
;-------------------------------------------------------------------------------
                ORG     8000H
;
INIC:           RIM
                ANI     80H
                MOV     B,A             ;ELOZO SID BEALLITASA A B-BE
                MVI     A,4EH           ;FELES IT-K MASZKOLASA(KIVETEL RST5.5), SOD=0
                SIM
                XRA     A
                MOV     L,A             ;L=0
;-------------------------------------------------------------------------------
KEZD:           RIM
                ANI     80H
                MOV     D,A             ;D=AKTUALIS SID
                XRA     B               ;VOLT SID VALTOZAS ?
                JZ      KEZD            ;UGRIK, HA NEM VOLT
                XRA     A
                ORA     B               ;ELOZO SID „0” VOLT ?
                JNZ     LEFUT           ;UGRIK, HA LEFUTO EL
FELFUT:         EI
                NOP
                DI
;-------------------------------------------------------------------------------
LEFUT:          MOV     B,D             ;ELOZO ERTEK FRISSITESE
                JMP     KEZD
                END
```

9. „Feles” megszakítások alkalmazása.

Példa 9.1.: Készítsen megszakítási vonalakat kezelő áramkört, amely egy 8085-ös processzoron alapuló sín RST 5.5-ös megszakítási vonalára csatlakozva 3 db külső megszakítás fogadására alkalmas. Egy megszakítási impulzusra csak egyszer kérjen megszakítást. Feltételezzük, hogy a megszakítási impulzusok gyakorisága max. 1ms. A megszakítások kezelését külön-külön áramkörök végzik, amelyek alaphelyzetbe állíthatók a 78H, 79H és 7AH címre adott írás paranccsal. A megszakításkérés lekérdezhető a fenti címekre adott olvasás paranccsal. Az RST 5.5-ös megszakítási szubrutinban az IRT1 a „C”, az IRT2 a „D” és az IRT3 az „E” regisztereket inkrementálja.

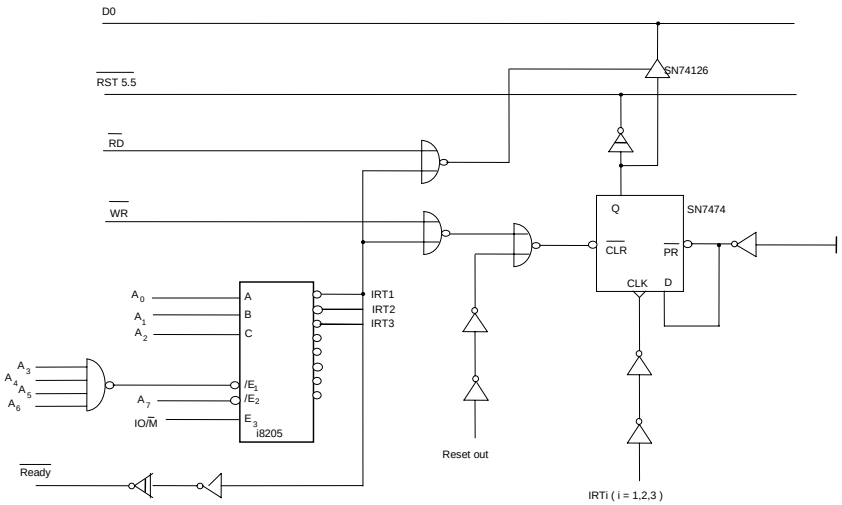
Megoldás 9.1.:

Javasolt blokkvázlat:



9.1. ábra

I/O áramkör



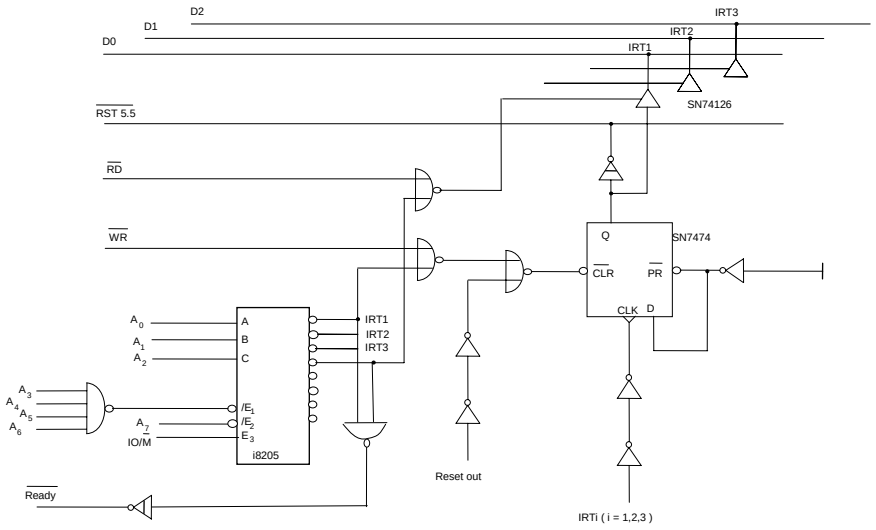
9.2. ábra

```
; FOPROGRAM
RAM             EQU     8000H
STACK           EQU     8100H
IRT1KC          EQU     78H
IRT2KC          EQU     79H
IRT3KC          EQU     7AH
IRTKC           EQU     7BH             ; (9.2. PELDA)
SZAKIT          EQU     7CH             ; (9.3. PELDA)
```

```
MASZK       EQU     0000 1110B      ; RST 6.5, RST 7.5 MASZK PARANCS ( 9.1., 9.2. PELDAK)
MSZ75       EQU     0001 1011B      ; RST 5.5, RST 6.5 MASZK PARANCS (9.3. PELDA) ES RST 7.5 FF TORL.
T75         EQU     0001 0000B      ; RST 7.5 FF. TORLESE (9.3. PELDA)
;
            0RG     0
            JMP INIC
;
            ORG     02CH            ; RST 5.5-OS RUTIN KEZDOCIME
            JMP     RUTIN
;
            ORG     40H
INIC:       LXI     H,STACK
            SPHL                    ; STACK BEALLITASA
            XRA     A
            MOV     C,A
            MOV     D,A
            MOV     E,A             ; C,D,E REGISZTEREK TORLESE
            MVI     A,MASZK
            SIM                     ; RST 6.5, RST 7.5 MASZKOLASA
            EI
HUROK:      JMP     HUROK
;
; RST 5.5 SZUBRUTIN
;
RUTIN:      PUSH    PSW
            IN      IRT1KC          ; IRT1 MEGSZAKITAS ERKEZETT ?
            ANI     1
            JNZ     RUT1
            IN      IRT2KC          ; IRT2 MEGSZAKITAS ERKEZETT?
            ANI     1
            JNZ     RUT2
            INR     E               ; ITR3 LEKEZELESE
            OUT     IRT3KC          ; ITR3 MEGSZAKITASI FF. TORLESE
            POP     PSW
            EI
            RET
RUT1:       INR     C               ; ITR1 LEKEZELESE
            OUT     IRT1KC          ; ITR1 MEGSZAKITASI FF. TORLESE
            POP     PSW
            EI
            RET
RUT2:       INR     D               ; ITR2 LEKEZELESE
            OUT     IRT2KC          ; ITR2 MEGSZAKITASI FF. TORLESE
            POP     PSW
            EI
            RET
            END
```

Példa 9.2.: Készítsék el a *Példa1* megoldását úgy, hogy a megszakítási ff.-ok értékei egy utasítással legyenek lekérdezhetők (cím: 7BH).

Megoldás 9.2.:

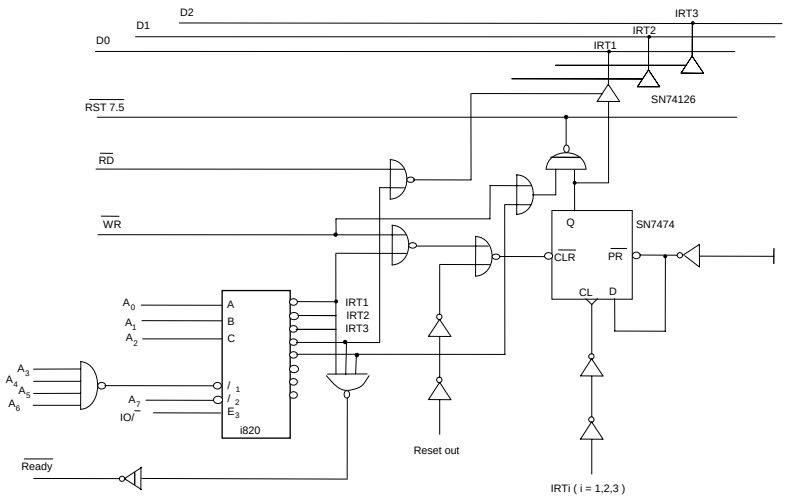


9.3. ábra

```
; RST 5.5 SZUBRUTIN
;
RUTIN:      PUSH    PSW
            PUSH    H
            IN      IRTKC
            MOV     H,A
            ANI     1           ; IRT1 ERKEZETT ?
            JNZ     RUT1
RUT3:       MOV     A,H
            ANI     2           ; IRT2 ERKEZETT?
            JNZ     RUT2
RUT4:       MOV     A,H
            ANI     4           ; IRT3 ERKEZETT?
            JZ      VEGE
            INR     E           ; ITR3 LEKEZELESE
            OUT     IRT3KC      ; ITR3 FF. TORLESE
VEGE:       POP     H
            POP     PSW
            EI
            RET
RUT1:       INR     C           ; ITR1 LEKEZELESE
            OUT     IRT1KC      ; ITR1 MEGSZAKITASI FF. TORLESE
            JMP     RUT3
RUT2:       INR     D           ; ITR2 LEKEZELESE
            OUT     IRT2KC      ; ITR2 MEGSZAKITASI FF. TORLESE
            JMP     RUT4
```

Példa 9.3.: Készítsék el a *Példa2* megoldását RST 7.5-es megszakítás alkalmazásával.

Megoldás 9.3.:



9.4. ábra

```
;
; FOPROGRAM
;
RAM         EQU     8000H
STACK       EQU     8100H
IRT1KC      EQU     78H
IRT2KC      EQU     79H
IRT3KC      EQU     7AH
IRTKC       EQU     7BH         ; (9.2. PELDA)
SZAKIT      EQU     7CH         ; (9.3. PELDA)
MASZK       EQU     0000 1110B  ; RST 6.5, RST 7.5 MASZK
                                ; PARANCS( 9.1., 9.2.PELDAK)
MSZ75       EQU     0001 1011B  ; RST 5.5, RST 6.5 MASZK
                                ; PARANCS(9.3. PELDA)
T75         EQU     0001 0000B  ; RST 7.5 FF. TORLESE
                                ; (9.3. PELDA)
;
            0RG     0
            JMP INIC
;
            ORG     03CH        ; RST 7.5-OS RUTIN KEZDOCIME
            JMP     RUTIN
;
```

```
        ORG     40H
INIC:   LXI     SP,STACK    ; STACK BEALLITASA
        XRA     A
        MOV     C,A
        MOV     D,A
        MOV     E,A         ; C,D,E REG. TORLESE
        MVI     A,MSZ75
        SIM                 ; RST 6.5,  RST 5.5 MASZKOLASA
        EI
HUROK:  JMP     HUROK
;
; RST 7.5 SZUBRUTIN
;
RUTIN:  PUSH    PSW
        PUSH    H
        IN      IRTKC
        MOV     H,A
        ANI     1           ; IRT1 ERKEZETT ?
        JNZ     RUT1
RUT3:   MOV     A,H
        ANI     2           ; IRT2 ERKEZETT?
        JNZ     RUT2
RUT4:   MOV     A,H
        ANI     4           ; IRT3 ERKEZETT?
        JZ      VEGE
        INR     E           ; ITR3 LEKEZELESE
        OUT     IRT3KC      ; ITR3 FF. TORLESE
VEGE:   POP     H
        POP     PSW
        MVI     A,T75
        SIM                 ; RST 7.5 FF. TORLESE (AUTOMATIKUSAN IS TORLODIK)
        OUT     SZAKIT      ; RST 7.5 FELFUTOEL GEN.
        EI
        RET
RUT1:   INR     C           ; ITR1 LEKEZELESE
        OUT     IRT1KC      ; ITR1 FF. TORLESE
        JMP     RUT3
RUT2:   INR     D           ; ITR2 LEKEZELESE
        OUT     IRT2KC      ; ITR2 FF. TORLESE
        JMP     RUT4
        END
```

10. Megszakításkezelők alkalmazása (i8259A)

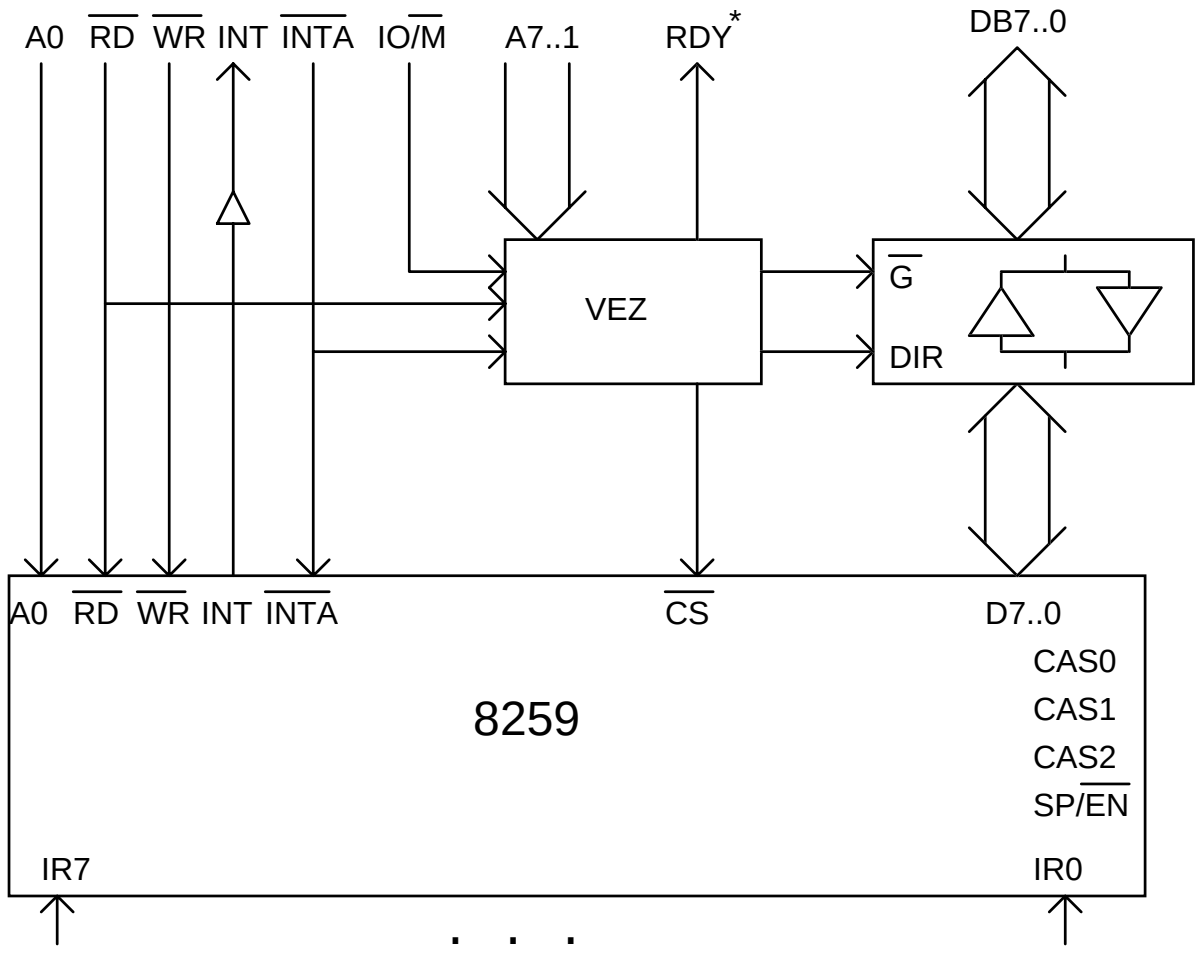
Példa10.1.:
Készítsen áramkört, amely egy 8085-ös processzoron alapuló sínre csatlakozva 8 db külső megszakítás fogadására alkalmas.

A. A processzor modul nem tartalmaz megszakítás vezérlőt, a processzor INT és INTA jelei ki vannak vezetve a buszra. A rendszerben egyéb IT kérő eszköz csak a processzor feles IT vonalait használhatja.
B. A processzor modul tartalmaz 1db 8259-es megszakításvezérlőt. A 8259-es IT bemenetei (IT0..IT7) valamint kaszkád kimenetei (CAS0..CAS2) megtalálhatók a buszon. A kártya az IT2 vezetékre csatlakozhat.

Készítsük el az IT vezérlőket inicializáló rutinokat!
a. Báziscím: 20H, IT tábla cím 1C0H, intervallum: 4 byte.
b. Processzor moduli vezérlő: Bázis cím: 80h, IT tábla cím 180H, intervallum: 4 byte.

I. HARDVER MEGVALÓSÍTÁS

A. KÜLSŐ IT VEZÉRLŐVEL



IR0 .. IR7: Szint vagy él+szint érzékeny, de mindkét esetben a kiszolgálás pillanatában IRi = 1 kell legyen.

1. VEZ feladata:

-CS előállítása

- a 8259-es mint periféria szerepel

$CS = saját\ tcím \cdot IO / \overline{M}$

Adaterősítő vezérlése

- INTA alatt
- pogramozás alatt (8259 mint periféria)

$G = CS + INTA \quad (\overline{G} = \overline{CS} \cdot \overline{INTA})$

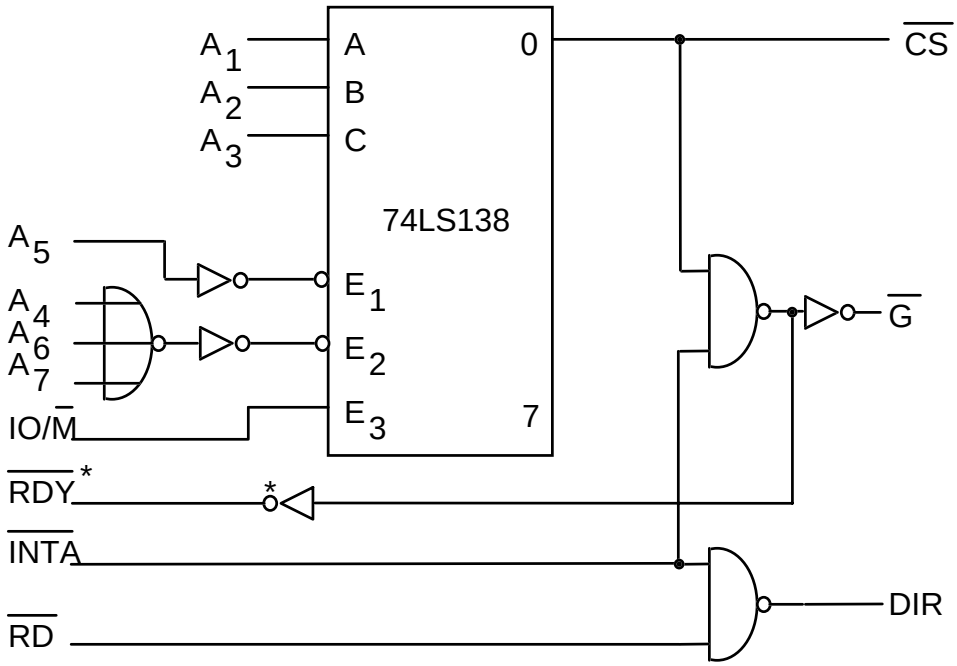
$DIR = IORD + INTA \quad (\overline{DIR} = \overline{IORD} \cdot \overline{INTA})$

RDY* előállítása

- INTA alatt
- pogramozás alatt (8259 mint periféria)

$RDY = CS + INTA$

(Figyelem! $INTA$ alatt $IO / \overline{M} = 1$, más IO eszköz is adhat READY-t)



2. IT RUTIN

Feladatok:
Regiszter mentés
IT kiszolgálás (IT "törlés")
Regiszter visszaállítás
IT engedélyezés

- 8259
- Processzor

a.) Automatikus EOI

A következő IT engedélyezése a processzor
EI utasításával
Használható, ha
Az IT rutinok nem megszakíthatók
Nincs nagy prioritású IT

```
ITRUT0:  PUSH  PSW
         .
         .
         POP   PSW
         EI
         RET
```

b.) Programozott EOI

```
ITRUTi:  PUSH  PSW
         .              ;eddig nem megszakítható
         .
         EI             ;innen kezdve a magasabb
         .              ;prioritású érvényre juthat
         .
         MVI   A,20h
         OUT   X59
         POP   PSW
         RET
```

IT engedélyezése programból
Használható, ha
Az IT rutinok "hosszúak"
Van nagy prioritású IT
Regiszter mentés
"Nem megszakítható" műveletek
EI
További műveletek (IT kiszolgálás)
Regiszter visszaállítás
8259 EOI (OCW2)

B. SLAVE IT VEZÉRLŐVEL

Különbségek:
CAS0..2 vonal kell
Bufferelt üzemmód: INTA helyett $SP / \overline{EN}$

Programozás

ICW1-ben	SNGL = 0	
ICW3-ban	MASTER:	$ICW3_{i.slave} = 1$
	SLAVE:	i
ICW4-ben	MASTER:	M/S = 1
	SLAVE:	M/S = 0

IT rutinban:EOI a MASTER-re és a SLAVE-re is! (Speciális rögzített prioritású működés!!!)

II. PROGRAMOZÁS

8259 felprogramozás (inicilaizálás, maszk állítás), IT tábla kitöltés

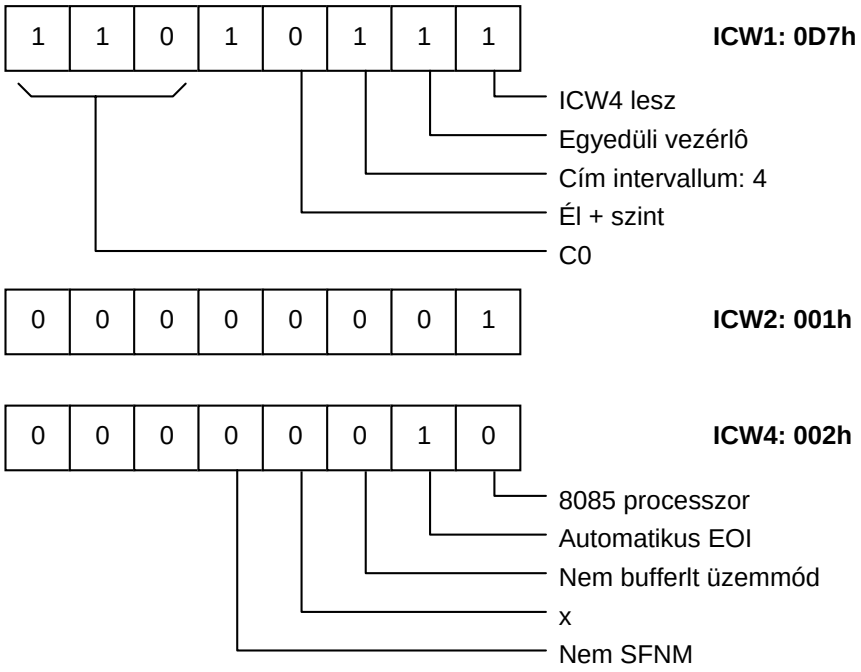
ICW1:
Trigger mód állítás
Tábla cím intervallum
Master/slave konfiguráció
ICW4 szükséges?
IT tábla - alsó byte felső 3 címbit

ICW2:
IT tábla - felső byte 8 címbit

(ICW3:
Master/slave konfigurálás**)**

ICW4:
###P típus
Automatikus EOI
Master/slave
Buffrelt üzemmód
spec. rögzített prioritás

ICW1, ICW2, ICW4: IT tábla báziscím 01C0H



Maszk állítás: Inicilaizálás után: mindenki engedélyezve - ha tiltani kell, M_i=1-re tiltás

```
ITMASK  EQU   0FFh
ICW1    EQU   0D7h
ICW2    EQU   001h
ICW4    EQU   002h
X59     EQU   020h        ; 8259 IO báziscím
ITBASE  EQU   001C0H      ; IT tábla báziscím

ITINIT: DI                ; Inicializálás alatt IT tiltás
        MVI   A,ICW1
        OUT   X59
        MVI   A,ICW2
        OUT   X59+1
        MVI   A,ICW4
        OUT   X59+1       ; 8259 inicializálás
        MVI   A,ITMASK
        OUT   X59+1       ; IT maszk állítás
        LXI   H,ITBASE
        MVI   A,0C3h      ; JMP kódja
        MOV   M,A
        INX   H
        LXI   D,ITRUT0    ; IT0 szubrutin címe

        MOV   M,E
        INX   H
        MOV   M,D
        INX   H
        INX   H           ; 4. byte átugrása
        MOV   M,A         ; JMP kódja
        INX   H
        LXI   D,ITRUT1    ; IT1 szubrutin címe
        .
        EI                ; Inicializálás vége, IT engedélyezés
```

RET

13. Kiemeneti periféria illesztés

Feladat: illesztési felület (interface) és működtető program (handler) kialakítása nyomtatóhoz (perifériához).

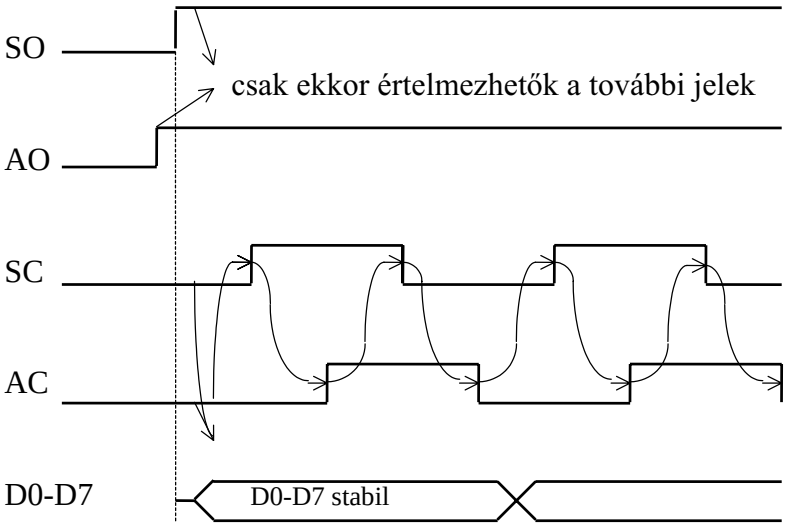
Megvalósítás: szabványos (egységes) interface és protokoll alkalmazása

Példa: BSI interfész és protokoll

Az interfész jelei:

Jel	**Function**	**Funkció**
SO	source operation	forrás üzemképes
SC	soure control	forrás vezérlés
AO	acceptor operation	nyelő üzemképes
AC	acceptor operation	nyelő vezérlés
D0..D7	data lines	adatvonalak
DP	parity	paritásbit

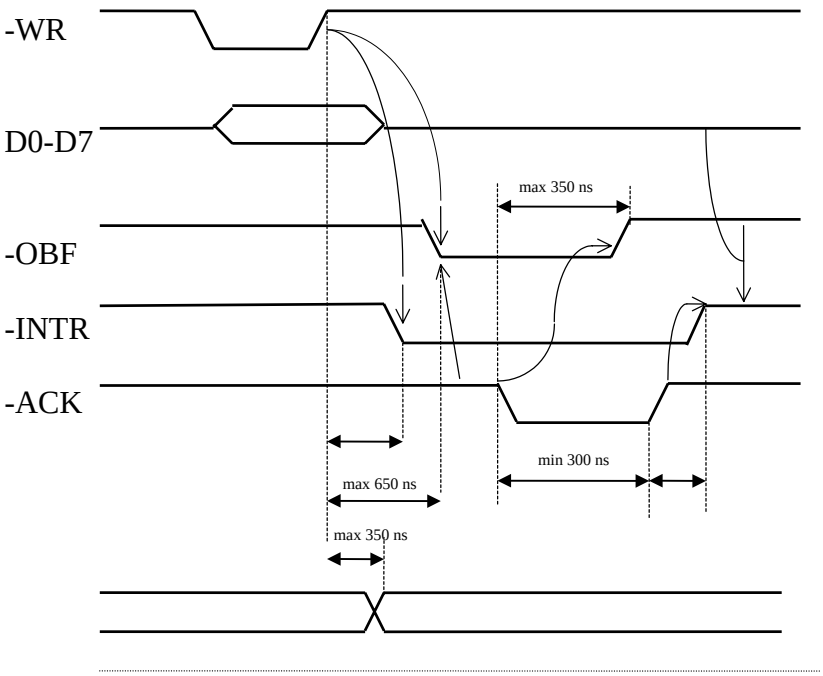
Protokoll:



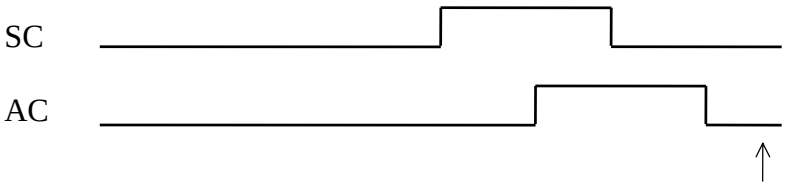
Ez egy un. handshaking (kézfogásos) protokoll, melynek lényege a vezérlőjelek kezdeményezés-visszajelzés párbeszéde.

8255A

1. üzemmódjában a kimenet



Periféria:

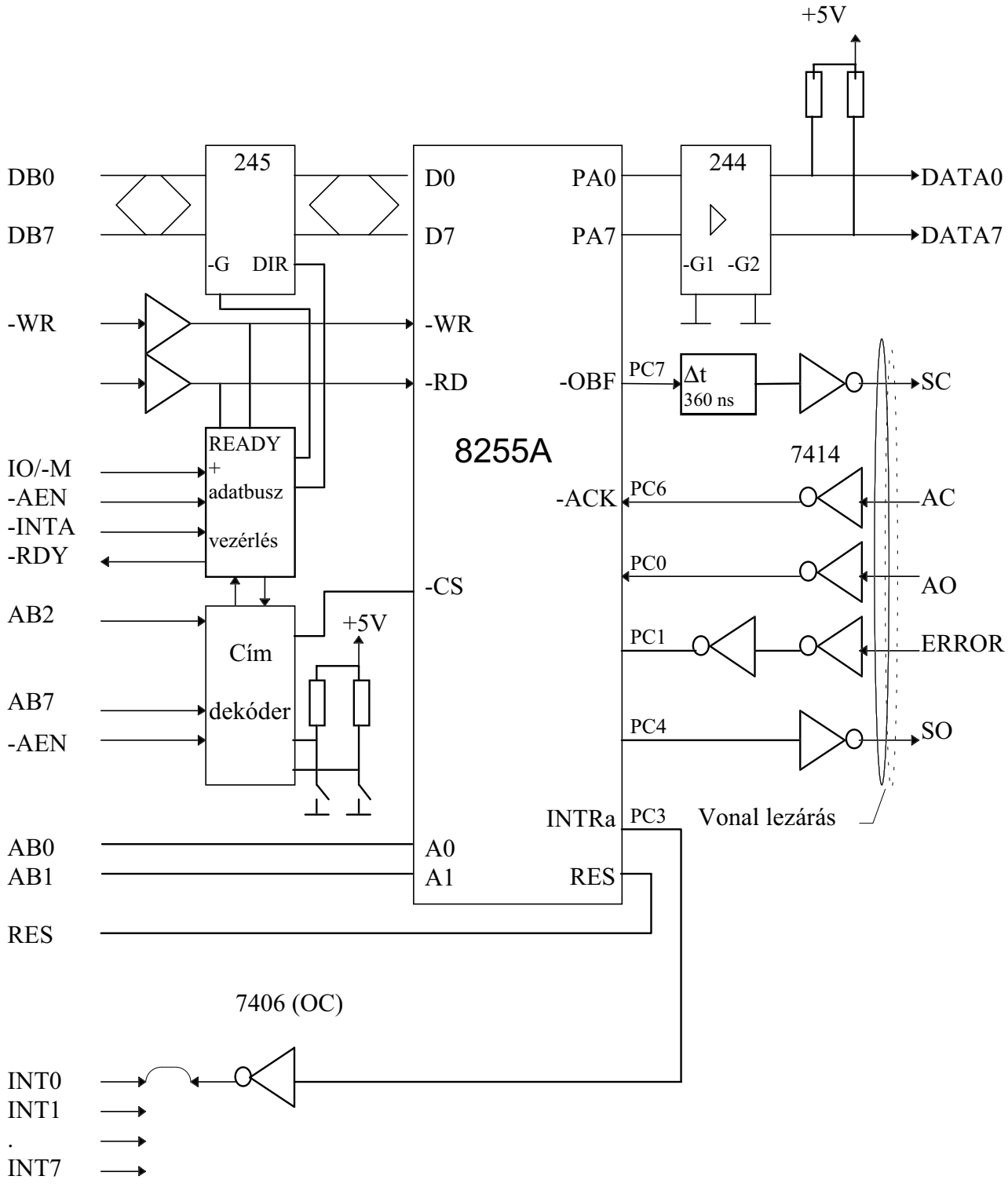


Figyelem! új adatot csak itt lehet kivinni

Irás (-WR) IT-re, vagy programozott státuszellenőrzés után

A port 1. üzemmód kimenetek:

-OBF	PC7
-ACK	PC6
INTRa	PC3

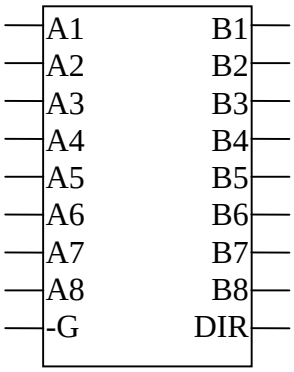


CS= saját cím ∗ IO/-M∗-AEN

-INTA∗CS=ready, hogy IT alatt ne adjon hamis READY-t

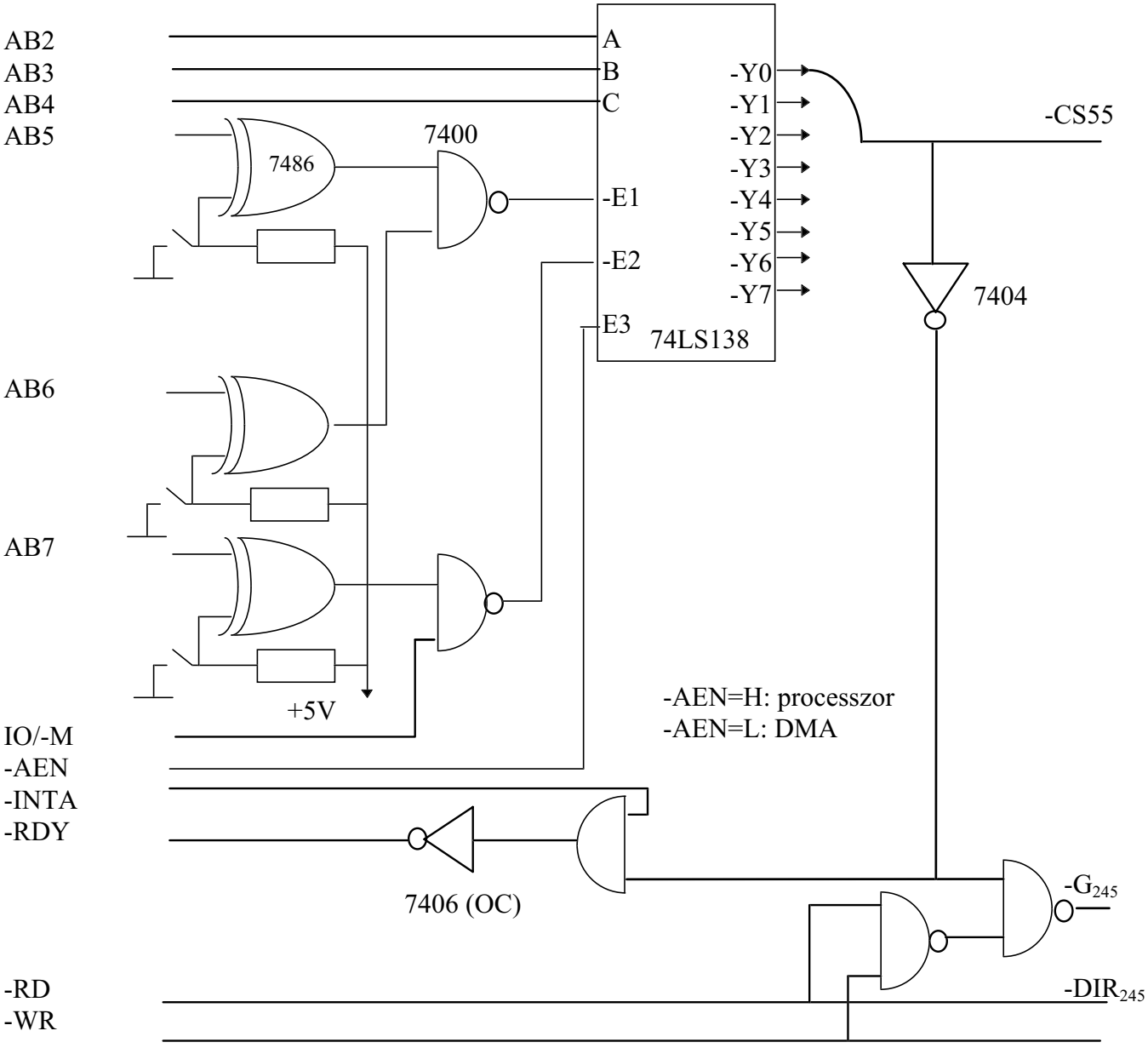
G=CS∗(RD+WR)

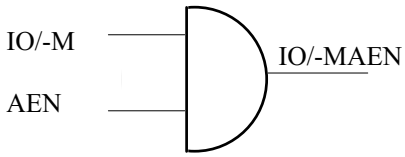
Adatbusz meghajtó:

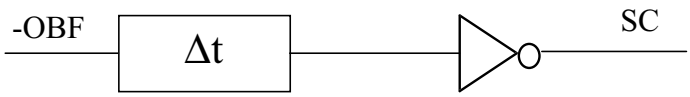


G	DIR	FUNKCIÓ
H	X	TS
L	L	B→A
L	H	A→B

Adatbusz + vezérlő logika:

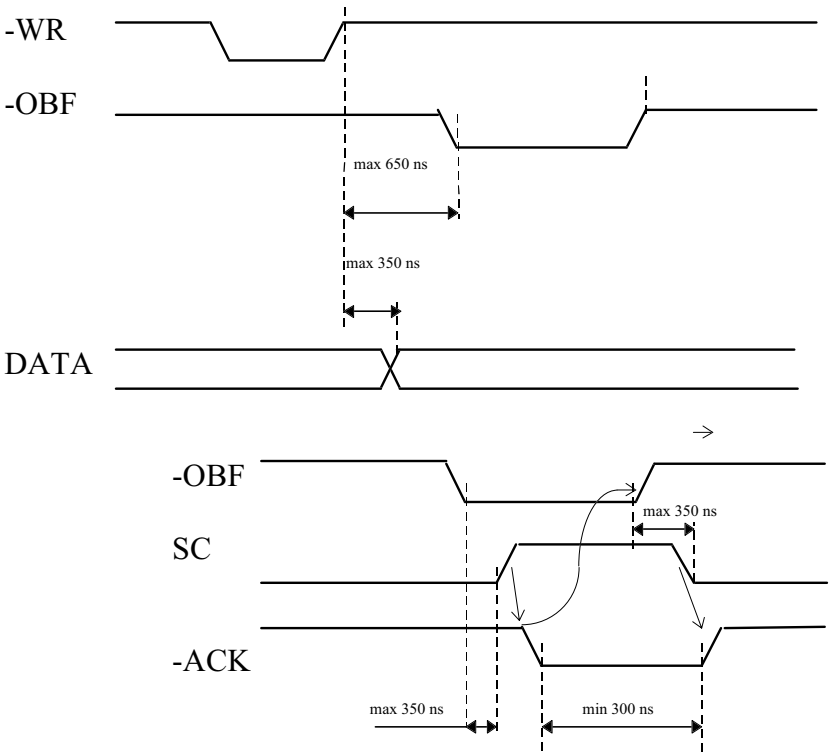






Δt biztosítja hogy DATA0..DATA7 előbb legyen stabil, mint SC 0→1

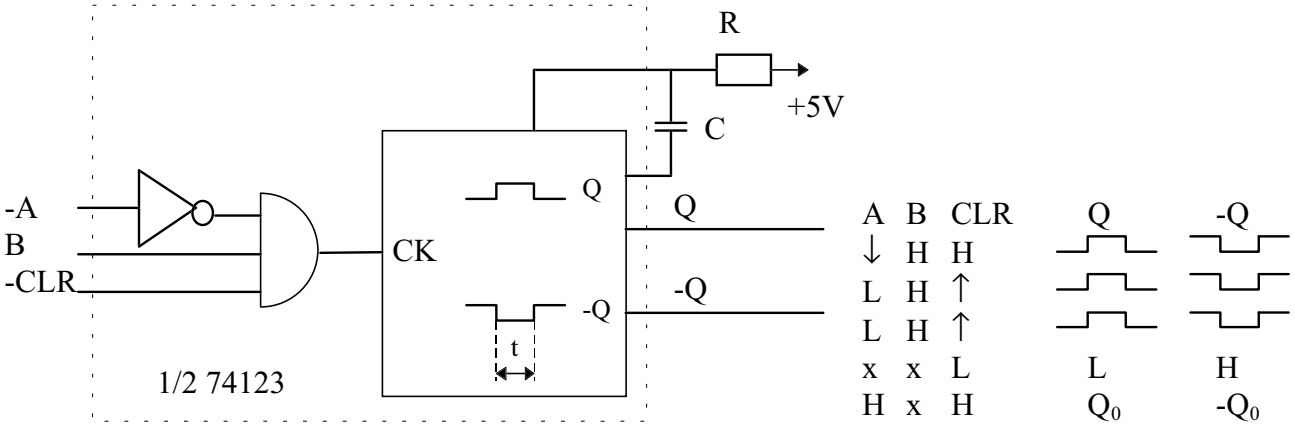
8255 -OBF↓ max 650 ns minimális időről nem ad tájékoztatást

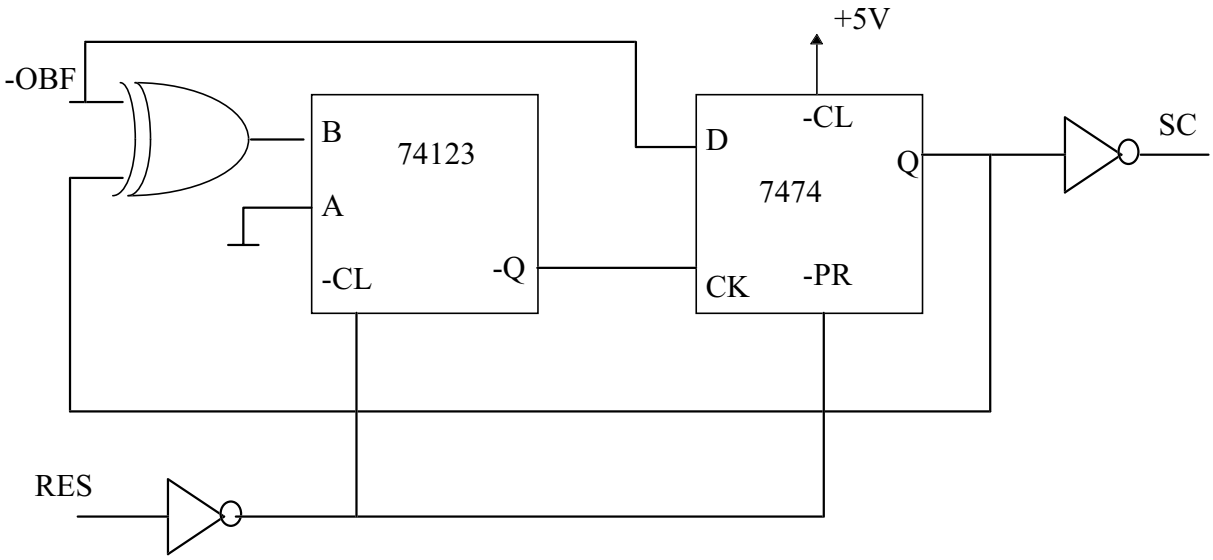


A Δt időzítés előállítása egy 74123 típusú monostabil multivibrátorral (Mono Flop) történik. Az időzítés nagyságát az R és C külső passzív elemekkel állíthatjuk be. Értéket az elemknek diagrammból, vagy az alábbi képlet alapján választhatunk.

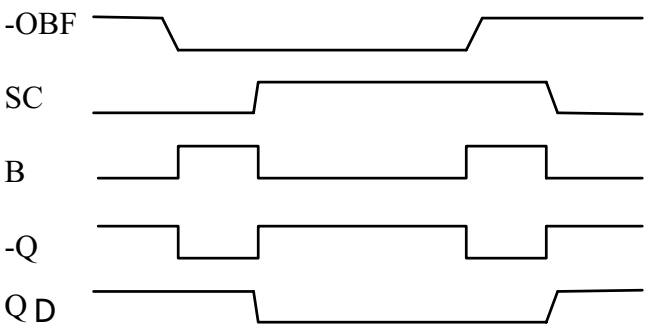
$$t = k \cdot R \cdot C(1 + \frac{0.7}{R})$$

Ahol k=0.28.





A működés idődiagramja:



Szoftver:

INIT (felprogramozás)

Lépések:
. 8255 konfigurálás, munkarekeszek alaphelyzetbe állítása
. az IT rutin címét az táblába beilleszteni

Munka regiszter és jelzők:

KH:

D7							D0
K							H

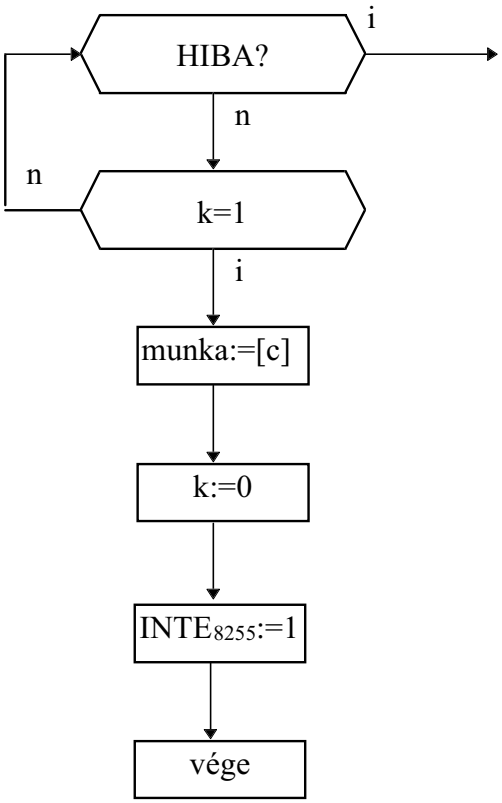
Munka:

kiviendő karakter

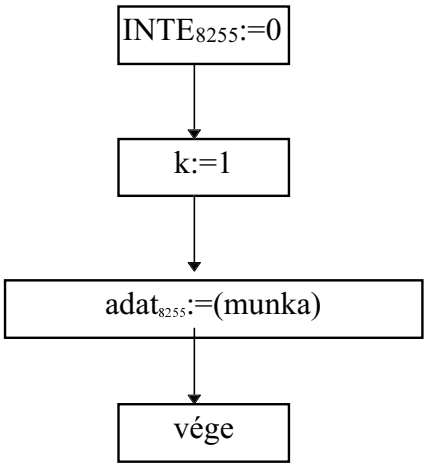
Státusz:

státusz

Kiviteli rutin:



IT rutin:



A 8255A programozása:

a) Regiszterek

Regiszter	Funkció	Cím: A0,A1	Típus
A port	0,1,2 üzem. ki/be	0,0	RW
B port	0,1 üzem. ki/be	1,0	RW
C port alsó	0 ,vezérlőjel A porthoz	0,1	RW
C port felső	0 ,vezérlőjel B porthoz	0,1	RW
Control regiszter	üzemmód állítás, Cport bit set/reset	1,1	W

CW(kontroll regiszter):

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	0	0	1	1

A port
00: 0.üm
01: 1.üm
1x: 2.üm

üzemmód

A port
1=be
0=ki

C felső
1=be
0=ki

B port
1=1 üm
0=0.üm

B port
1=be
0=ki

C alsó
1=be
0=ki

CW=A3h

D7	D6	D5	D4	D3	D2	D1	D0
0	x	x	x	b2	b1	b0	1

bit kiválasztás

bit s/r
1=set
0=reset

PC4	set:	09h
	reset	08h
PC6	set	0Dh
	reset	0Ch

C port (státusz):

D7	D6	D5	D4	D3	D2	D1	D0	irány/üm
IO	IO	IBFa	INTEa	INTRa	INTEb	IBFb	INTRb	be/1
-OBFa	INTEa	IO	IO	INTRa	INTEb	-OBFb	INTRb	ki/1
-OBFa	INTE1	IBF	INTE2	INTRa	B	B	B	/2

A program:

```
;
;                   GLOBAL CONSTANTS

X55                 EQU     0A0H        ;PORT BASE ADDRESS
X55_A               EQU     X55         ;PORT A
X55_B               EQU     X55+1       ;PORT B
X55_C               EQU     X55+2       ;PORT C
X55_CT              EQU     X55+3       ;CONTROL
CW55                EQU     0A3H        ;ONTROL WORD
DISINT              EQU     00CH        ;DISABLE 8255 IT
ENINT               EQU     00DH        ;ENABLE 825 IT
;
ENSO                EQU     008H        ;ENABLE SO-RESET
PC4
DISSO               EQU     009H        ;DISABLE SO- SET PC4
;
KMASK               EQU     080H
HMASK               EQU     001H
PRERR               EQU     003H        ;PRINTER ERROR
IT1MSK              EQU     0FDH        ;IT ENABLE (8259)
EOI                 EQU     020H        ;END OF INTERRUPT(8259)
;
PUBLIC              X55,CW55,DISINT,ENINT,DISSO,ENSO
PUBLIC              KMASK,HMASK,PRERR,IT1MSK
;
;*************************************************************
;                   DATA SEGMENT
;*************************************************************
;                   DSEG    PAGE
                    PUBLIC  KH,MUNKA,HIBA
;
KH:                 DS      1
MUNKA:              DS      1
HIBA:               DS      1

;*************************************************************
;                   PRINTER IT
;*************************************************************

                    CSEG    PAGE
                    PUBLIC  PRINIT
                    EXTERN  ITTAB,X59   ;IT TABLE, 8259 BASE
ADDRESS
;
PRIT:               PUSH    PSW         ;SAVE REGISTER & FLAG
                    MVI     A,DISINT
                    OUT     X55_CT      ;DISABLE IT 8255
                    LDA     MUNKA
                    OUT     X55_A       ;OUTPUT CHARACTER TO A
                    MVI     A,KMASK
                    STA     KH          ;K:=1
                    MVI     A,EOI
                    OUT     X59         ;ENABLE NEXT IT (8259)
                    POP     PSW
                    EI
                    RET
```

```
;*************************************************************
;                       PRINTER INIT
;*************************************************************
PRINIT:
                        MVI         A,CW55
                        OUT         X55_CT
                        MVI         A,DISINT
                        OUT         X55_CT
                        MVI         A,ENSO
                        OUT         X55_CT
                        MVI         A,KMASK
                        STA         KH
;
;                       STORE IT TO ITTABLE
;
                        MVI         A,0C3H          ;JMP CODE
                        STA         ITTAB+4         ;IT 1 POSITION
                        LXI         H,PRIT
                        SHLD        ITTAB+5
;
                        IN          X59+1           ;ENABLE IT (8259)
                        ANI         ITMSK
                        OUT         X59+1
;
                        RET
                        END
;*************************************************************
;                       CHARACTER OUT TO PRINTER
;
;                       IN:     C       =       CHARACTER TO PRINT
;                       OUT:    CY      =       0 - OK
;                               CY      =       1 - ERROR , A = ERROR CODE
;                       DEST:   A,F
;*************************************************************
;
                        CSEG        PAGE
                        PUBLIC      PRO
                        EXTERN      X55,KMASK,HMASK, PRERR
                        EXTERN      KH, MUNKA, HIBA
;
PRO:                    IN          X55_C
                        ANI         PRERR
                        JNZ         P1              ;ERROR IN PRINTER
                        LDA         KH
                        ANI         KMASK
                        JZ          PRO             ;WAIT
                        MOV         A,C             ;PRINT
                        STA         MUNKA           ;NEXT CHAR
                        XRA         A
                        STA         KH              ;       K:=0
                        MVI         A,ENINT
                        OUT         X55_CT          ;ENABLE IT (8255)
                        RET
;
P1:                     STA         HIBA            ;ERROR IN PRINTER
                        MVI         A,HMASK
                        STA         KH
                        LDA         HIBA
                        STC
                        RET
;
                        END
```

Protokoll konverter 8085 mikroprocesszoros rendszerrel

8085 mikroprocesszor és 8255 felhasználásával tervezzünk olyan áramkört, mely a 8255 B portjáról 1-es üzemmódban beolvasott adatokat a szintén 1-es üzemmódba állított A portra továbbítja oly módon, hogy az egymás után érkező azonos adatbájtokat csak egyszer adja tovább. Az adatbeolvasást a processzor RST6.5 vonalán érkező megszakítással, míg a kivitelt programmal ellenőrzött készenléttel oldjuk meg.

Az áramkör álljon két jól elkülöníthető részből, egy CPU kártyából és egy bővítő kártyából.

A CPU kártyán állítsuk elő bufferekkel leválasztott módon az alábbi jeleket:
SA0..SA15, SD0..SD7
$\text{SIO}/\overline{\text{M}}$, $\overline{\text{SRD}}$, $\overline{\text{SWR}}$, $\overline{\text{INTA}}$, SOD, RESETOUT, CLK, HLDA
$\overline{\text{TRAP}}$, $\overline{\text{RST7.5}}$, $\overline{\text{RST6.5}}$, $\overline{\text{RST5.5}}$, $\overline{\text{INT}}$, $\overline{\text{SID}}$, $\overline{\text{HOLD}}$, $\overline{\text{READY}}$
A 8085 processzor órajele 6.144MHz legyen.

Memória térkép

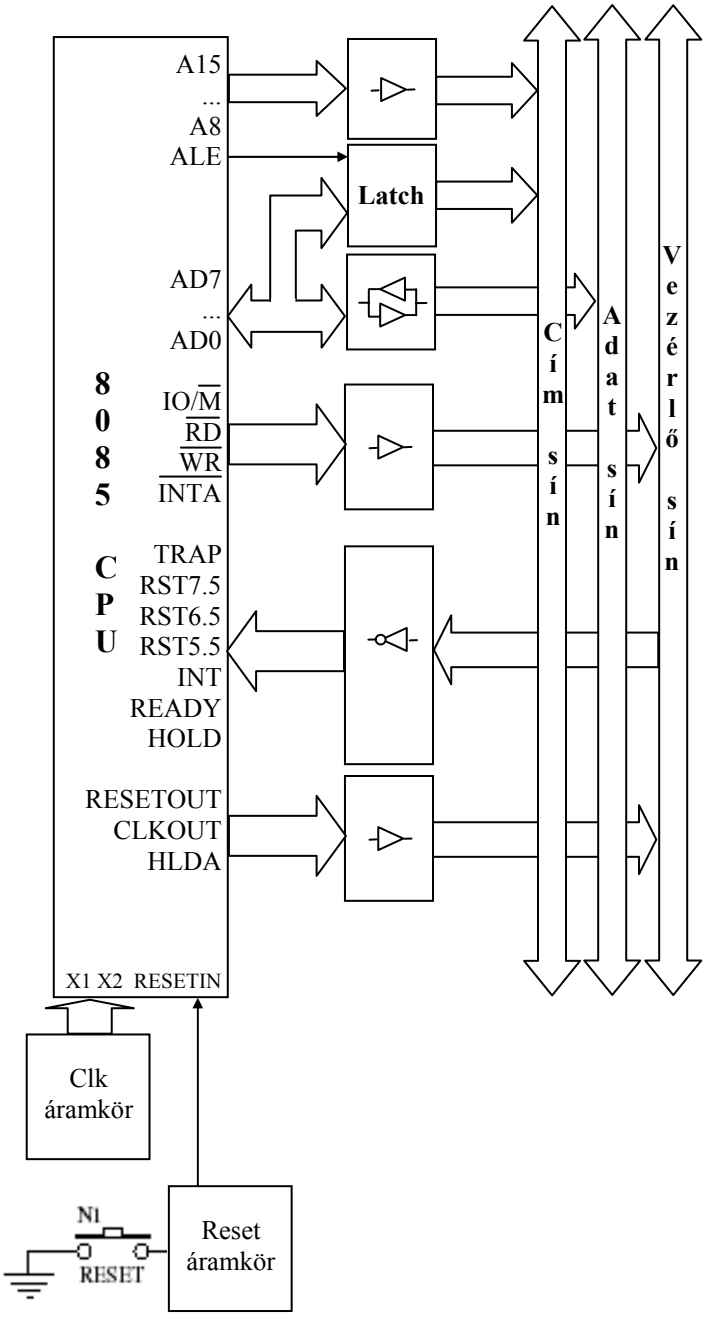
0000h	2k EPROM
07FFh	
0800h	
.. ..	
7FFFh	
8000h	2k RAM
87FFh	
8800h	
.. ..	
FFFFh	

I/O címtartomány

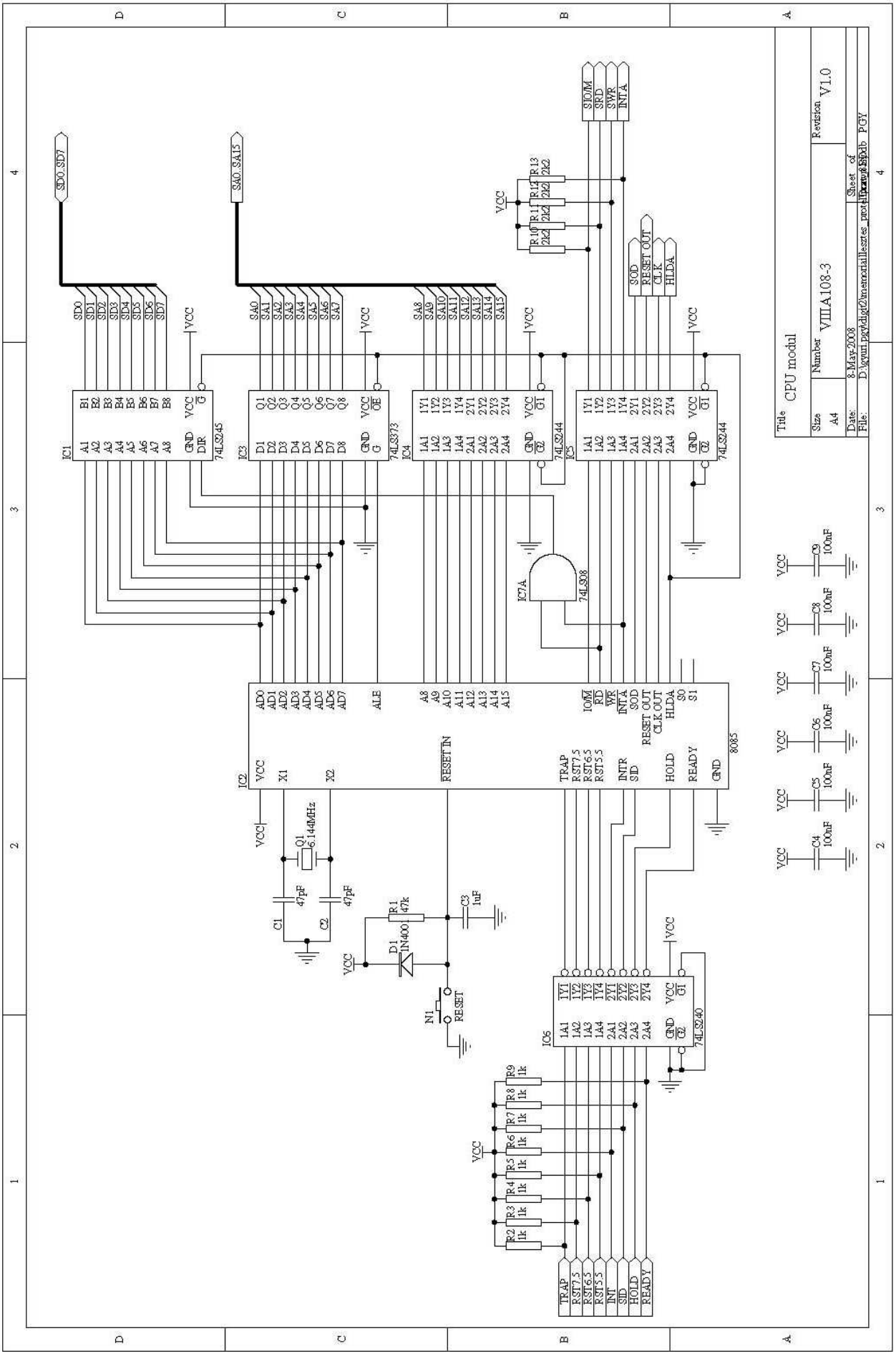
00h	8255 A port
01h	8255 B port
02h	8255 C port
03h	8255 Control
04h	
..	
7Fh	
80h	Fenntartott
..	
FFh	

A vezérlő program tárolásához és működéséhez a bővítő kártyán gondoskodjunk 2k EPROM (2716) és 2k RAM (5516) memóriáról is. Az EPROM memória a 0000h…07FFh, a RAM memória a 8000h…87FFh területeken látszódjon. A tervezés során vegyük figyelembe, hogy a rendszerben semmilyen más memória nincs, és nem is lesz. A memória áramkörök 0 WAIT állapot közbeiktatásával működjenek.
A 8255 áramkör a 00h IO báziscímtől kezdődően legyen elérhető. Az IO címtartományban legyünk tekintettel arra, hogy a 00h…7Fh címtartományban nincs és nem is lesz más áramkör, azonban későbbi bővítésre a 80h..FFh címtartományt fent kell tartani. (A 8255 áramkör szintén 0 WAIT állapotot igényel.)

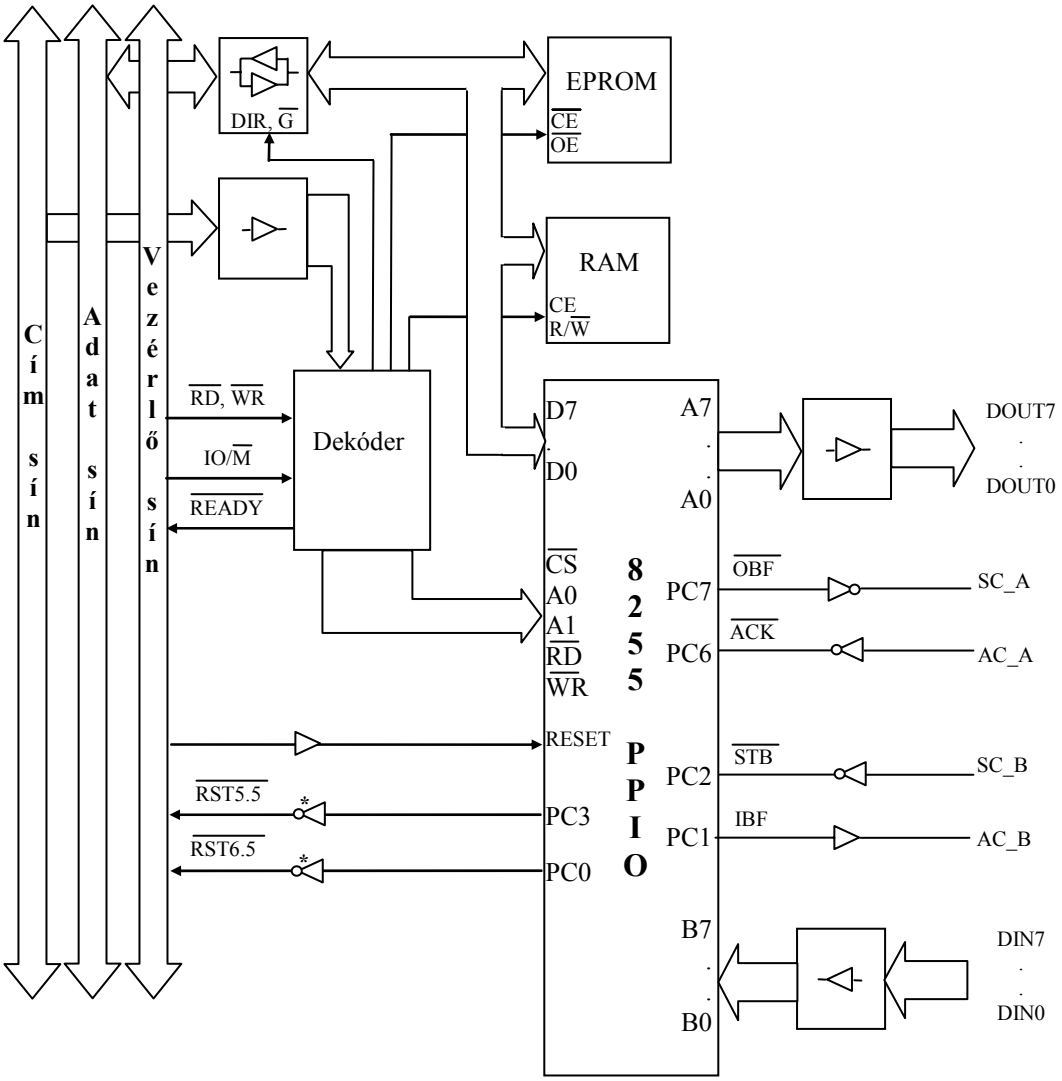
Írjuk meg az EPROM-ba égetendő programot, amely RESET után a specifikációban megadott működést biztosítja.



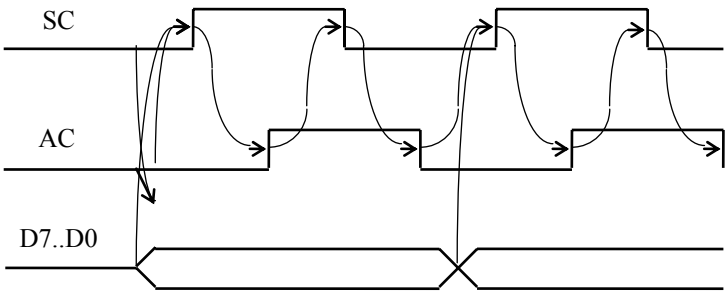
CPU kártya blokk vázlata



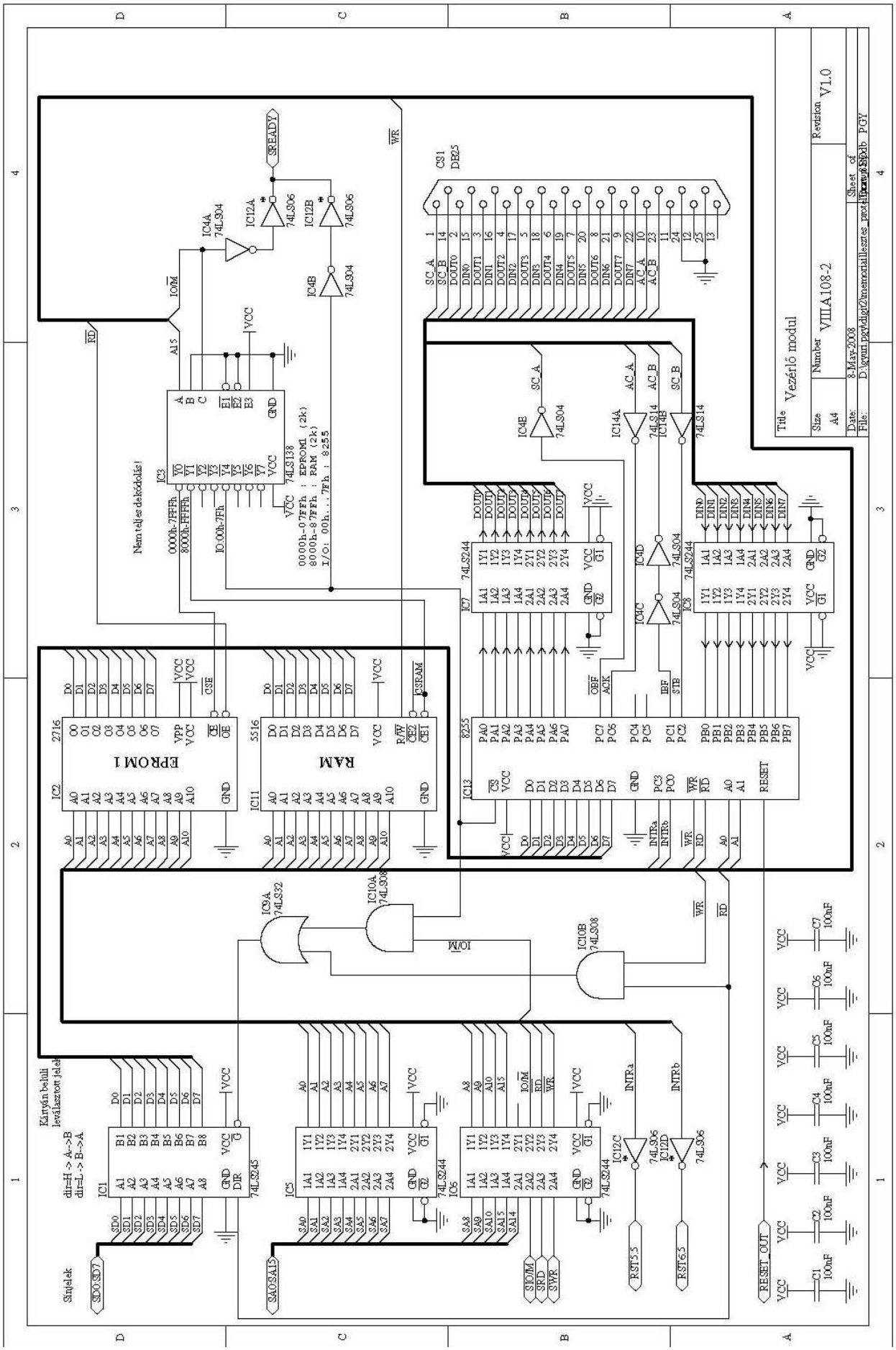
CPU kártya kapcsolási rajza



Bővítő kártya blokk vázlata



A hardver (handshake) protokoll



Bővítő kártya kapcsolási rajza

A működtető szoftver

```
; Feladata:
; A 8255 B portjáról 1-es üzemmódban interrupt-tal beolvasott
; adatokat az A portra 1-es üzemmódban programmal ellenőrzött
; készenléttel adjuk tovább úgy, hogy az egymás után érkezett
; azonos adatbyte-okat csak egyszer továbbítjuk.
;
; (Feltételezzük, hogy a fogadó oldal elég gyors, legfeljebb
; 255 byte lemaradásban lehet.)
;

P8255A    EQU    00h
P8255B    EQU    P8255A+1
P8255C    EQU    P8255A+2
P8255P    EQU    P8255A+3

C_8255    EQU    10101111B    ; A port mode1 output
                              ; B port mode1 input
EN_A_INT  EQU    00001101B    ; A port INTE-ff engedélyezés (PC6)
EN_B_INT  EQU    00000101B    ; B port INTE-ff engedélyezés (PC2)
EN_RST65  EQU    00001101B    ; RST6.5 engedélyezés (többit tilt)

B_INTR    EQU    00000001B    ; B port input mode INTR (Bit0)
A_INTR    EQU    00001000B    ; A port output mode INTR (Bit3)

STACK_    EQU    8800h        ; stack alja (RAM vége+1)
RAM_      EQU    8000h        ; RAM kezdete
EPROM_    EQU    0000h        ; EPROM kezdete
;
; BUFFER: 256 byte körpuffer a beolvasott adatoknak
; WP : írás mutató - a bufferbe utoljára írt adatra mutat
; RP : olvasás mutató - a bufferből utoljára olvasott adatra mutat
;
          org    RAM_
; kihasználjuk, hogy így 256-tal osztható címre kerül
BUFFER:   ds 256
WP:       ds 1               ; írás mutató
RP:       ds 1               ; olvasás mutató

          org    EPROM_       ; 0
          JMP    init

          org    34h          ; RST 6.5 belépési pont
          JMP    behoz        ; RST 6.5 IT rutin
```

```
          org    100h          ; könnyebb a debug, mert 256-tal
                               ; osztható címre kerül

init:     LXI    SP, STACK_    ; Stack pointer inicializálás
          XRA    A
          STA    RP            ; RP inicializálás
          STA    WP            ; WP inicializálás
          MVI    A, EN_RST65
          SIM                  ; RST 6.5 engedélyezés
          MVI    A, C_8255     ; 8255 vezérlő szó
          OUT    P8255P
          MVI    A, EN_A_INT   ; Bit set/reset- nem lesz IT
                               ; de erre a bitre várunk majd
          OUT    P8255P        ; A port INTE ff
          MVI    A, EN_B_INT   ; Bit set/reset
          OUT    P8255P        ; B port INTE ff
          EI

;-------------------------------------------------------
; Főprogram
;-------------------------------------------------------

          LXI    D, BUFFER     ; DE - puffer mutató
          LXI    H, RP         ; HL - RP címére mutat
c_0:      LDA    WP            ; WP akt. értékének felolvasása
                               ; értékét az IT rutin módosítja
          CMP    M             ; WP == RP ?
          JZ     c_0           ; az első byte-ra várunk
          INR    M             ; megérkezett - RP növelés
          MOV    E, M          ; kiviendő byte címe: puffer + RP
          LDAX   D             ; byte felolvasás
c_1:      MOV    C, A          ; paraméter a szubrutinnak
          CALL   kivisz        ; byte kivitele
c_2:      LDA    WP            ; olvasás mutató
          CMP    M             ; WP == RP ?
          JZ     c_2           ; byte-ra várunk
          INR    M             ; megérkezett - RP növelés
          MOV    E, M          ; kiviendő byte címe: puffer + RP
          LDAX   D             ; byte felolvasás
          CMP    C             ; összehasonlítás az előzővel
          JZ     c_2           ; nem kell kivinni
          JMP    c_1           ; ki kell vinni
```

```
;-------------------------------------------------------------
;kivisz
; feladat: 1 byte kivitele az A porton
;          mode 1-ben, programmal ellenőrzött készenléttel
; BE: C - A kiviendő byte
; KI: -
; RONT: A, Flagek
;-------------------------------------------------------------
kivisz:
          IN     P8255C
          ANI    A_INTR
          JZ     kivisz       ; INTR-re várunk, ha 0, az előző
                              ; adatot még nem vették el.
                              ; OBF nem jó, mert lehet, hogy
                              ; az ACK jel még nem került
                              ; alapállapotba
          MOV    A,C
          OUT    P8255A
          RET

;-------------------------------------------------------------
;behoz
; feladat: 1 byte beolvasása a B portról
;          mode 1-ben, IT rutinban
; BE: -
; KI: -
; RONT: - (IT rutin!)
;-------------------------------------------------------------
behoz:
          PUSH   H
          PUSH   PSW
          LXI    H,BUFFER     ; puffer cím előállítása
          LDA    WP           ; buffer írás mutató
          INR    A            ; csak az alsó byte-ot kell
          MOV    L,A          ; állítani
          STA    WP           ; WP növelése
          IN     P8255B       ; adat beolvasás
          MOV    M,A          ; tárolás
          POP    PSW
          POP    H
          EI
          RET
```