HÁLÓZATI RENDSZEREK
ÉS SZOLGÁLTATÁSOK
TANSZÉK

SZÁMÍTÓGÉP ARCHITEKTÚRÁK

Kártyás ajtónyitó tervezése

Horváth Gábor

BME Hálózati Rendszerek és Szolgáltatások Tanszék
ghorvath@hit.bme.hu, belso@hit.bme.hu

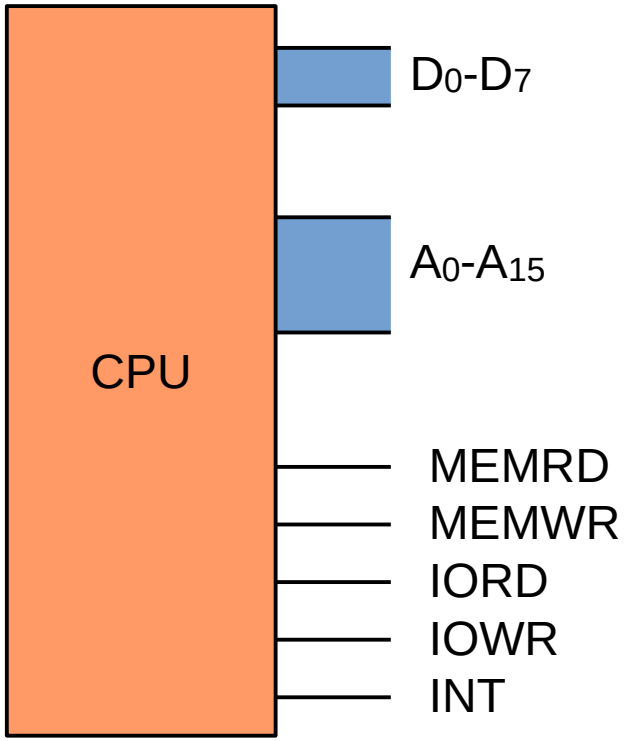
Budapest,
2024.02.09.

MŰEGYETEM 1782

TARTALOMJEGYZÉK

A hardver megtervezése

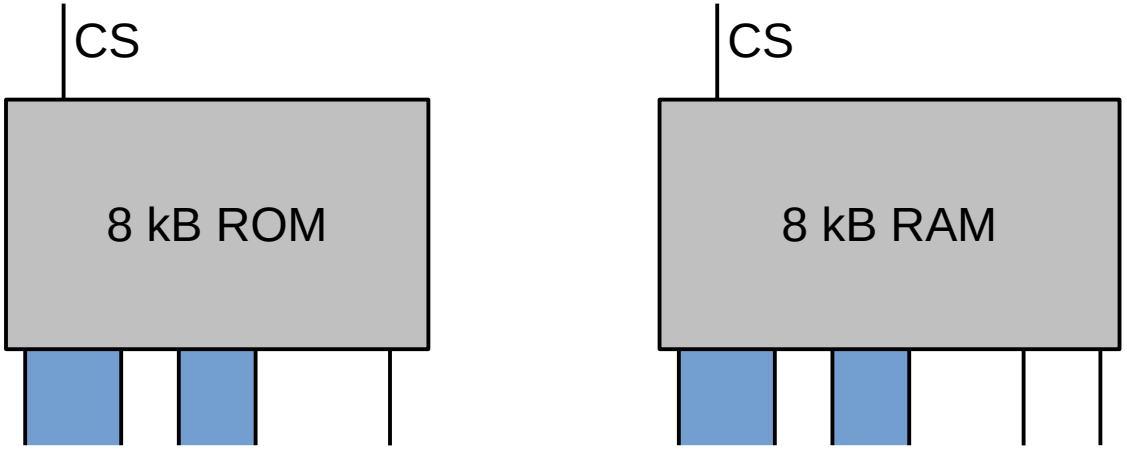
ALKATRÉSZEK



- 8 bites CPU
  - Adatbusz: 8 bit széles
  - Címbusz: 16 bit széles
  - Multiplexált memória és perifériabusz
  - RISC utasításkészlet
  - 3 operandusú utasítások
  - Kezdőcím: **0000h**
  - Megszakításkezelő címe: **1000h**

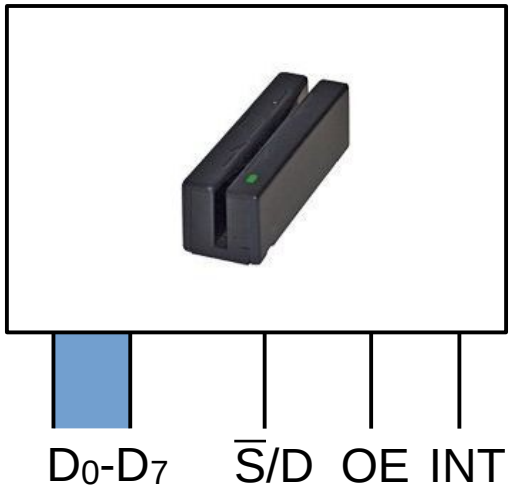
ALKATRÉSZEK

- Memória
  - ROM: **utasításoknak** és konstans adatoknak
  - RAM: **programváltozóknak** és **stack-nek**
- Vegyünk 8kB ROM-ot és 8kB RAM-ot
  - 8 bit adatsín
  - 13 bit címsín



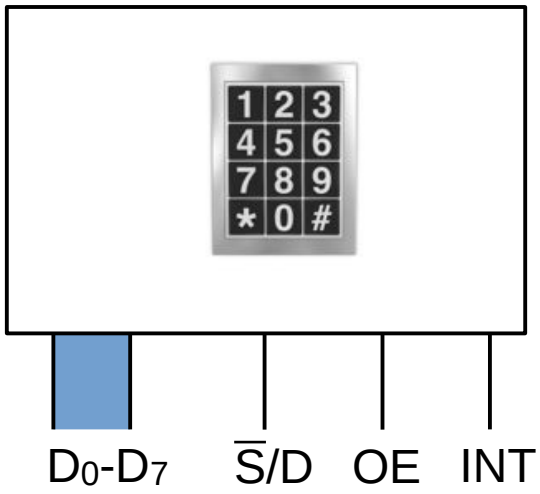
ALKATRÉSZEK

- Kártyaolvasó
  - OE: output enable
  - Van egy 8 bit-es kimenete, D:
    - Ha $\overline{S}$/D = 0: 1-et ad, ha új kártyát látott
    - Ha $\overline{S}$/D = 1: A kártya kódját adja
  - INT: megszakítást kér, ha új kártyát lát



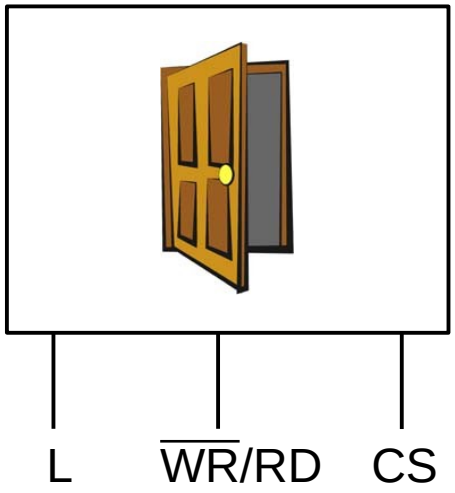
ALKATRÉSZEK

- Billentyűzet
  - OE: output enable
  - Van egy 8 bit-es kimenete, D:
    - Ha $\overline{S}$/D = 0: 1-et ad, ha gombnyomás történt
    - Ha $\overline{S}$/D = 1: A lenyomott gomb kódját adja
  - INT: megszakítást kér, ha gombot nyomtak



ALKATRÉSZEK

- Ajtónyitó
  - CS: chip select
  - $\overline{WR}$/RD
  - Ha $\overline{WR}$/RD = 1: L visszaadja az ajtó állapotát (nyitott=1, zárt=0)
  - Ha $\overline{WR}$/RD = 0: kinyitja az ajtót. Az ajtó egy időzítő lejárja után automatikusan újra bezár.

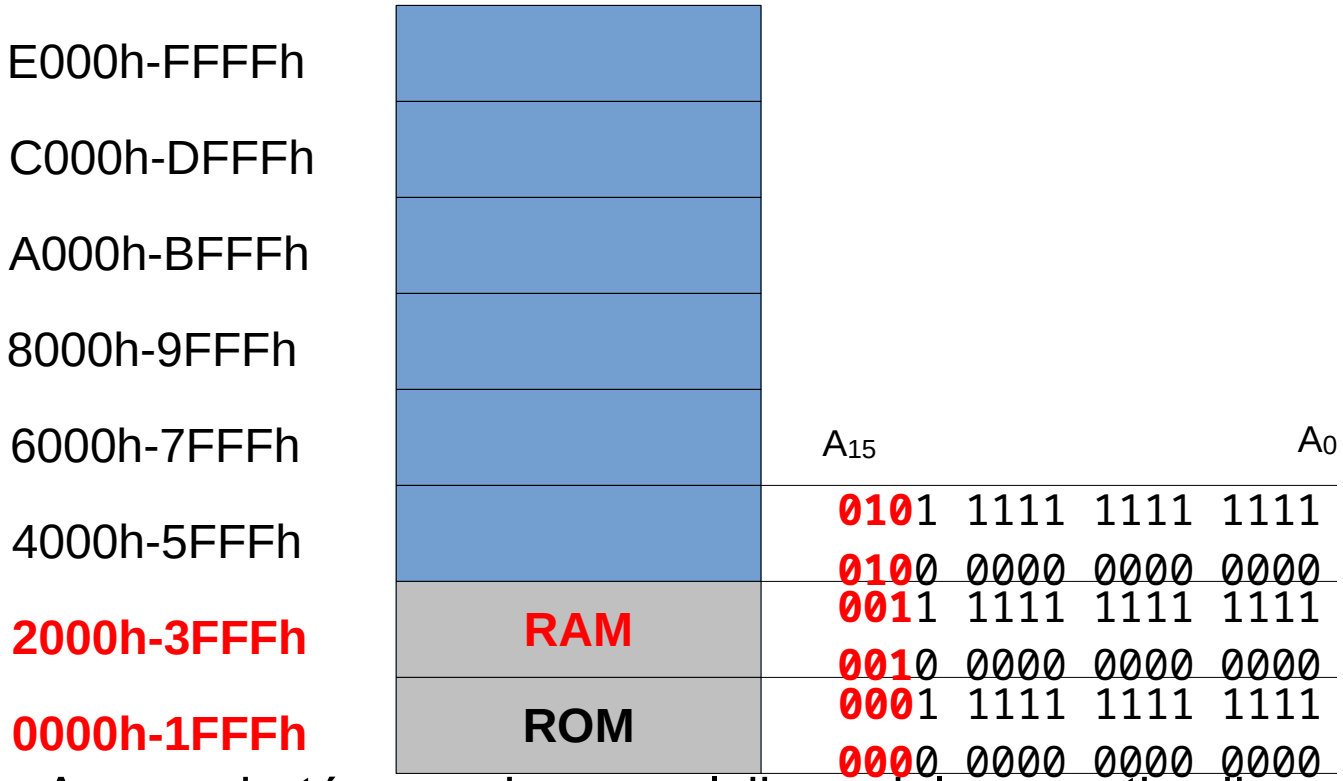


ALKATRÉSZEK

- LED lámpák
  - Az ajtó aktuális nyitottsági állapotát jelzi (piros/zöld)
- További elemek
  - Dekóder – a memóriaillesztéshez
  - Komparátorok – hogy a perifériák figyelni tudják a címüket
  - D flip-flop – a LED-ek állapotának beállításához

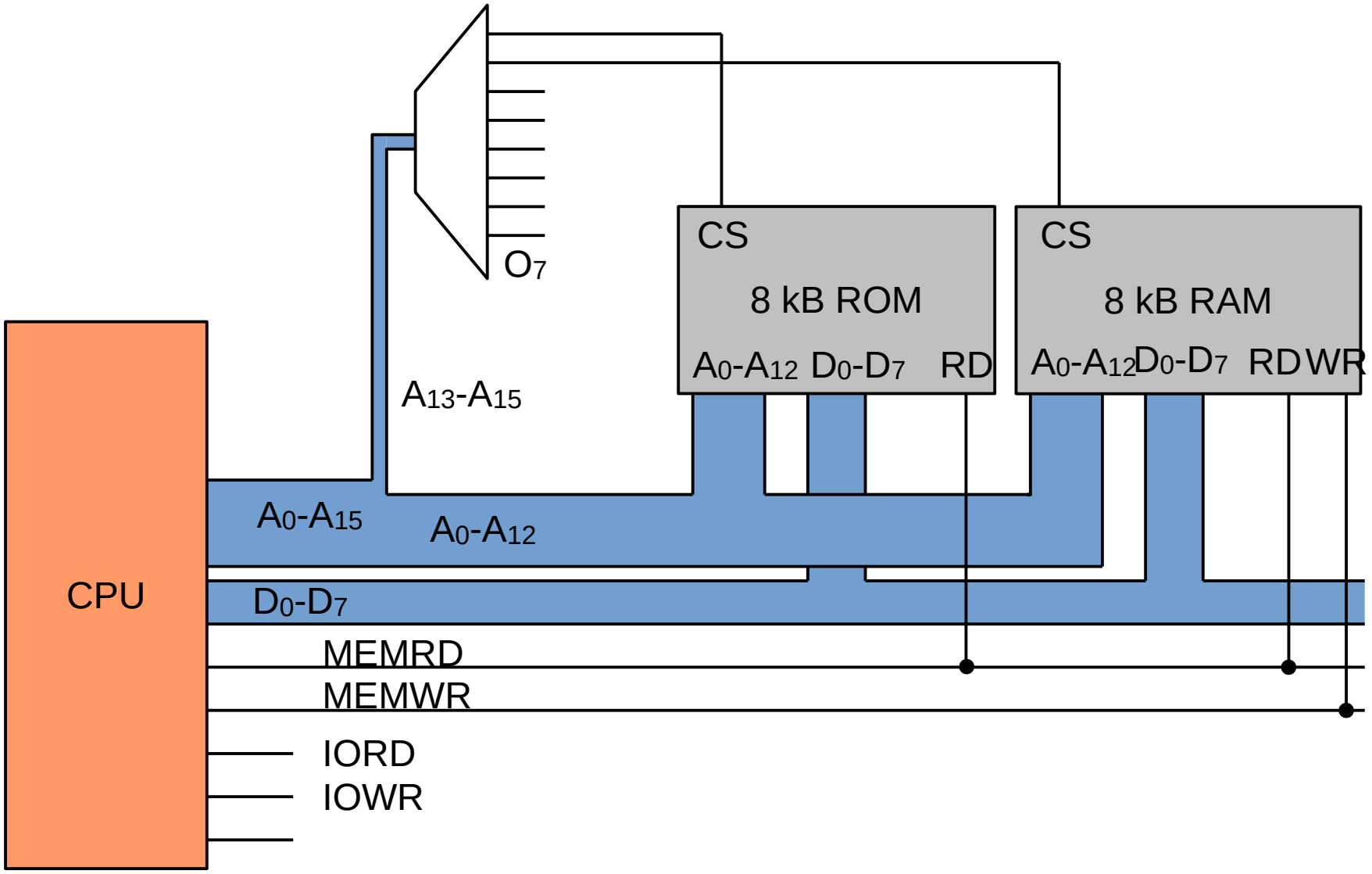
MEMÓRIAILLESZTÉS

- 8 kB ROM + 8 kB RAM
- Memóriatérkép:

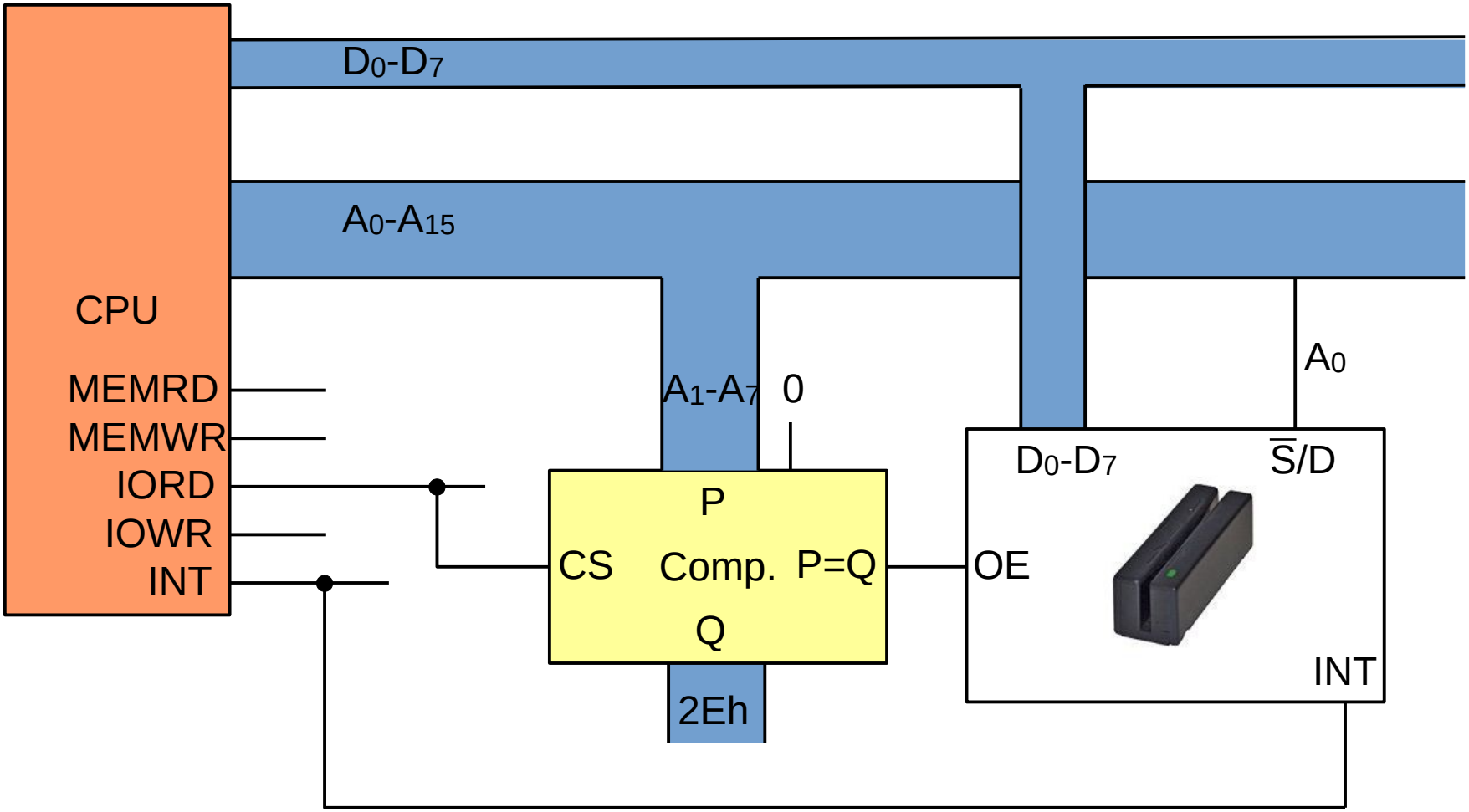


- A_{13}-A_{15} meghatározza, hogy melyik modulra vonatkozik a cím
- A_0-A_{12} a modulon belüli pozíció

MEMÓRIAILLESZTÉS

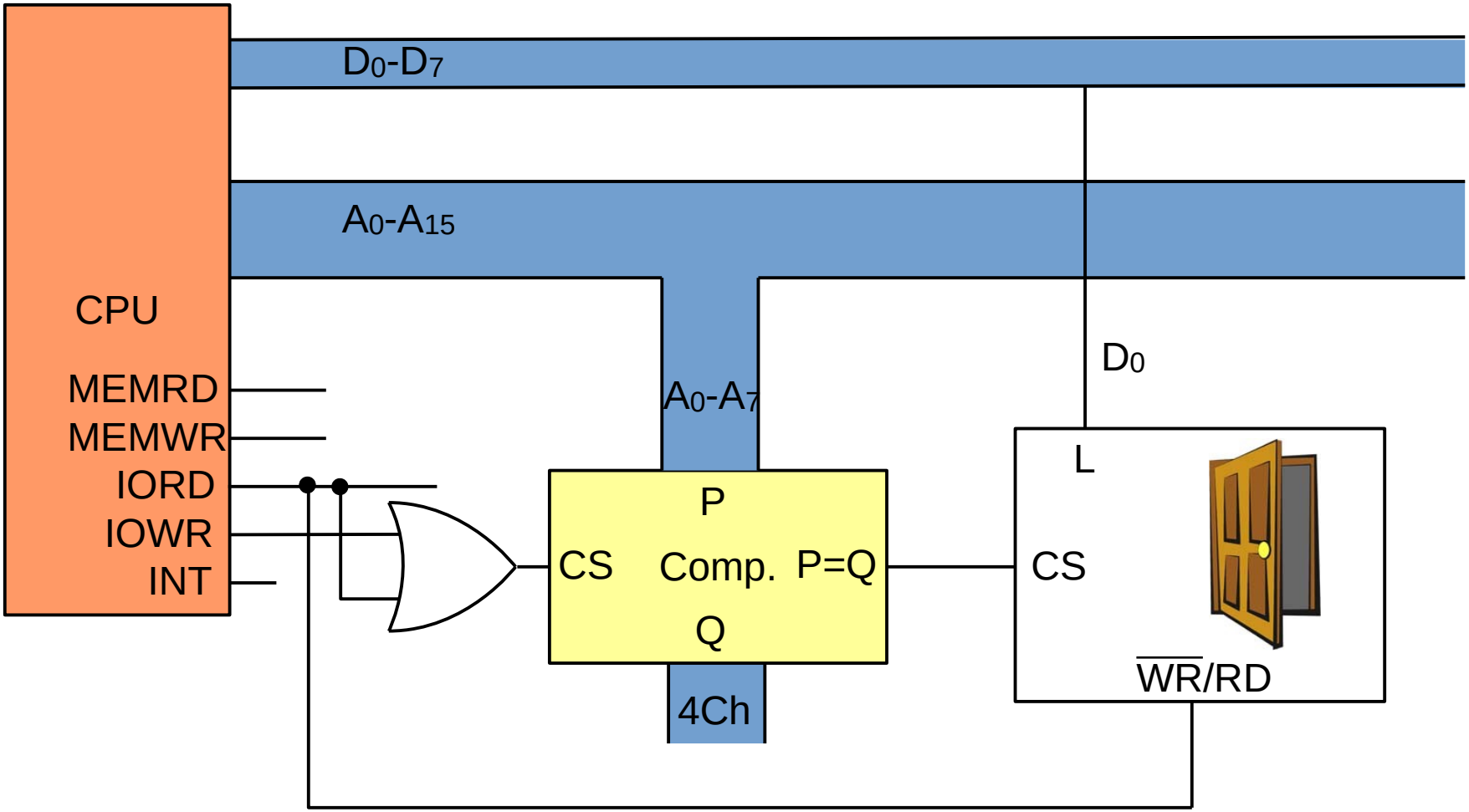


A KÁRTYAOLVASÓ ILLESZTÉSE

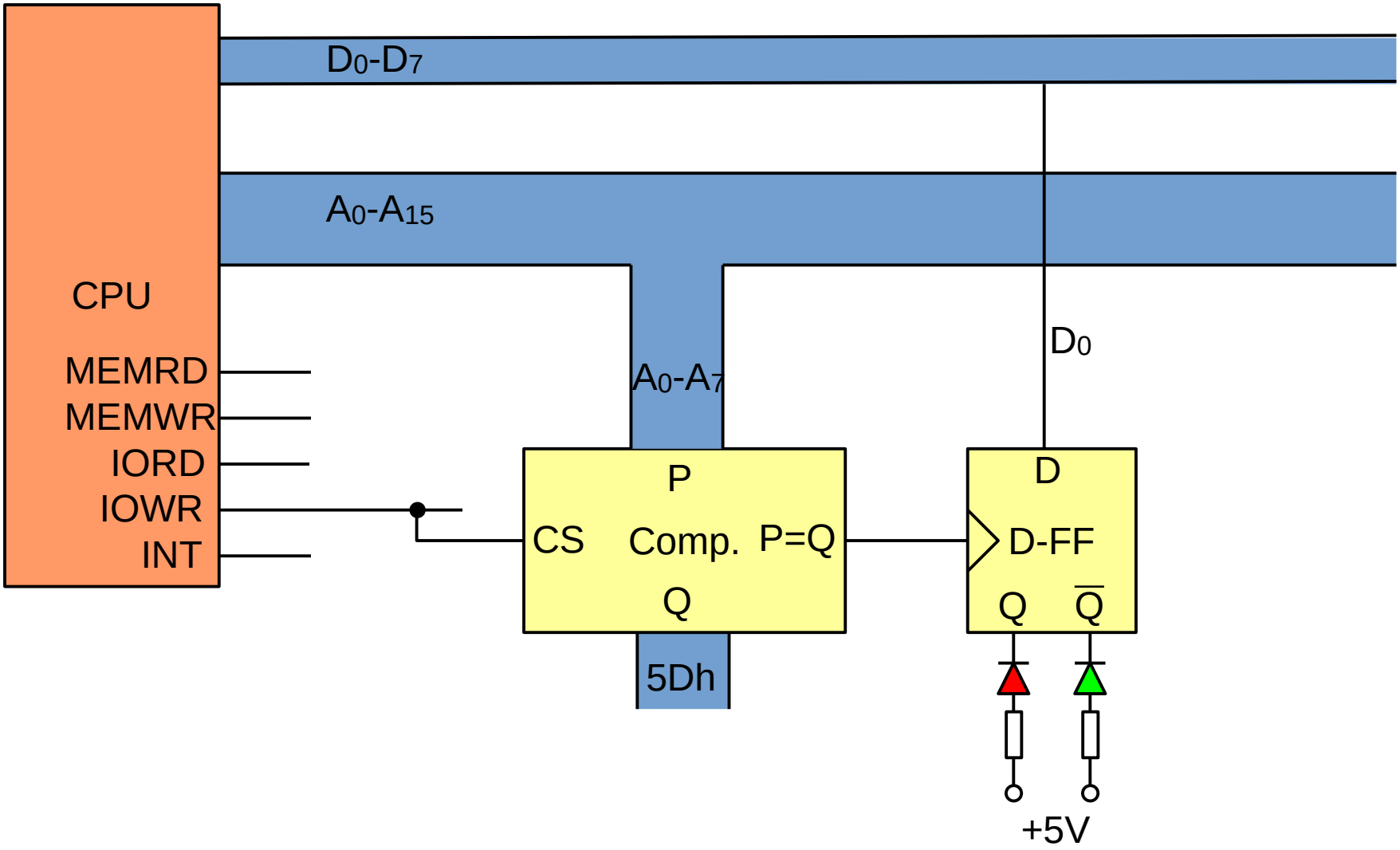


- A billentyűzetet ugyanígy illesztjük, de azt a 3Eh báziscímre

AZ AJTÓNYITÓ ILLESZTÉSE



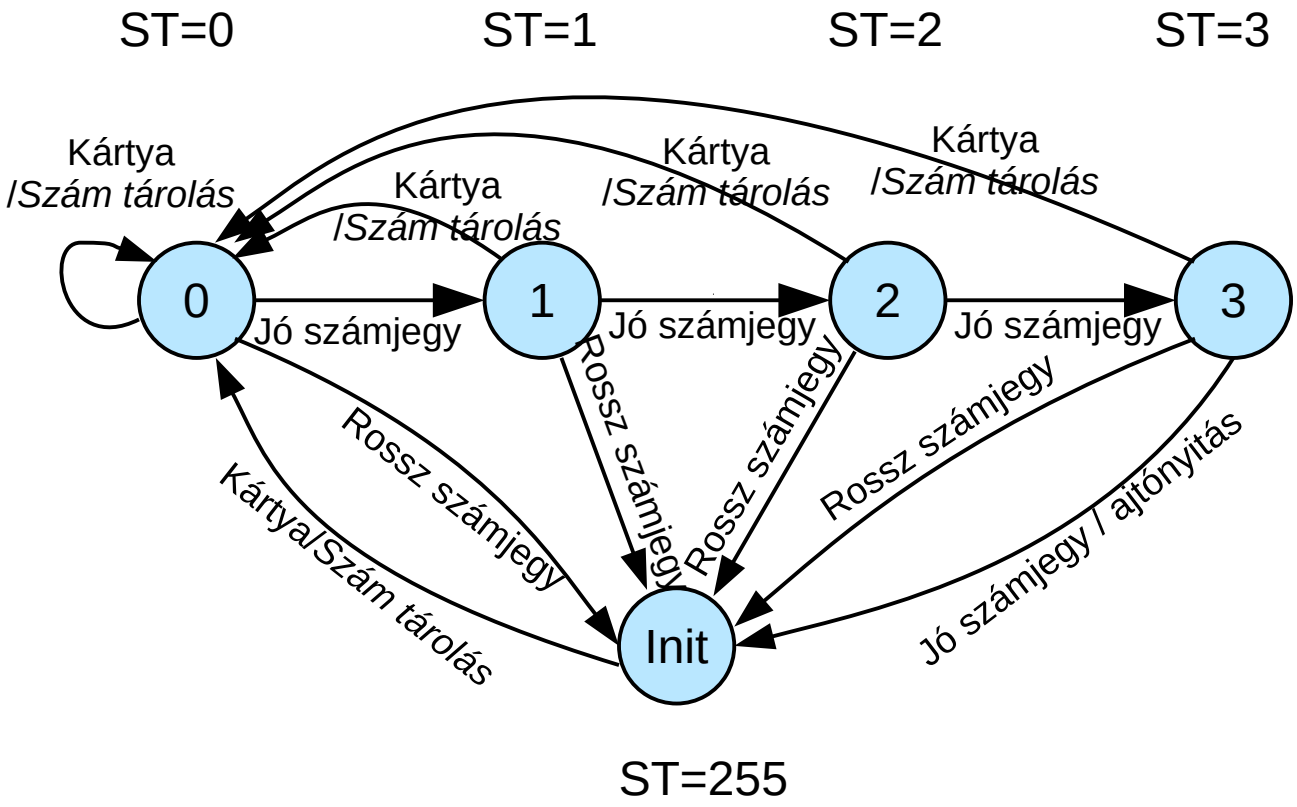
A LED LÁMPÁK ILLESZTÉSE



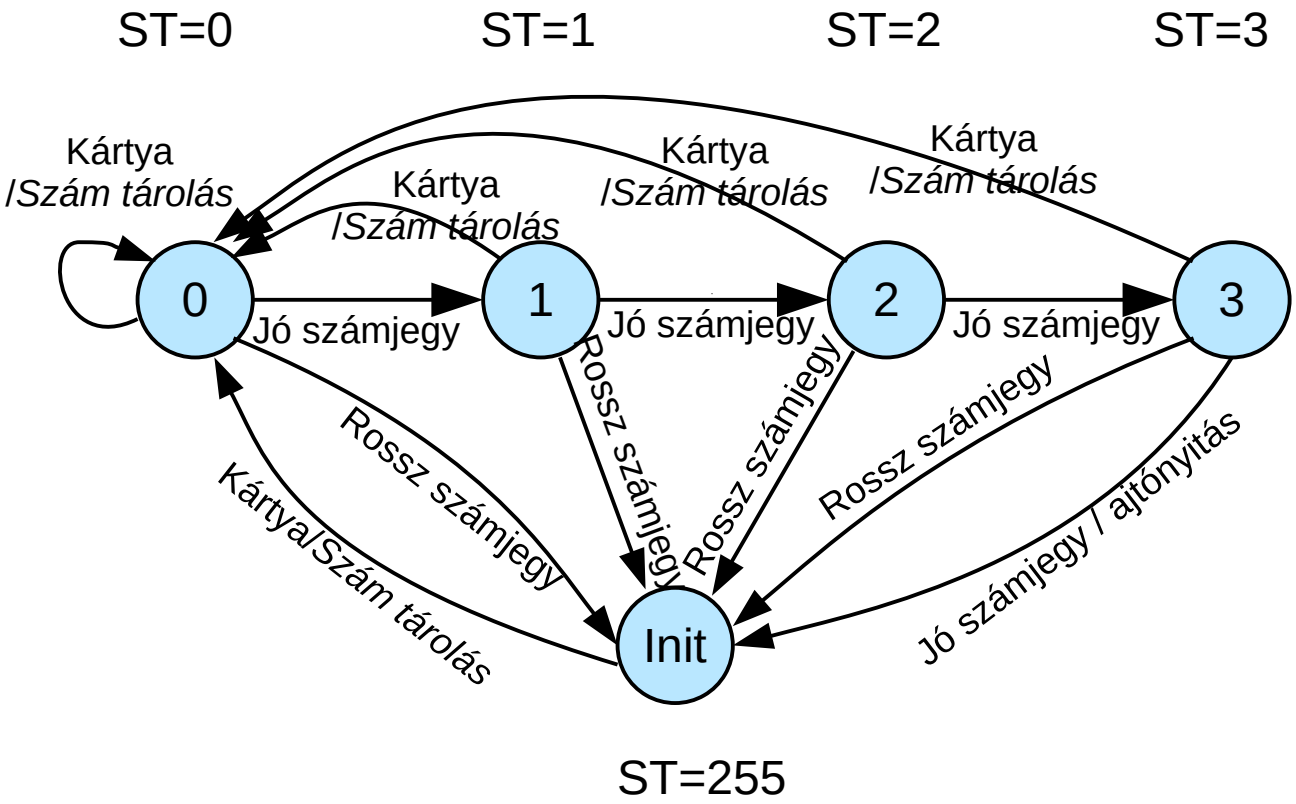
A szoftver megtervezése

ÁLLAPOTGÉP

- ST (állapot) = az eddig lenyomott helyes számjegyek száma



ÁLLAPOTÁTMENETEK



- Kártyalehúzáskor: ST=0, és a kártyaszám tárolás egy lokális változóba
- Jó számjegy megadásakor: ST=ST+1, ha ST nem 255. Ha ST 4 lett, ajtónyitás
- Rossz számjegy megadásakor: ST=255

ADATSZERKEZETEK

- A kártyákhoz tartozó 4 jegyű kódokat tároló tömb
  - A kártyaszám 8 bites → 256 különböző kártyaszám lehet
  - Tömböt használunk
    - 256 elemmel, minden kártyaszámhoz egy elem tartozik
    - Egy elem az adott kártyához tartozó 4 számjegyű kód
  - Teljes memóriafoglaltság: 256*4=1024 bájt (=**400h**)
  - A ***j.*** kártyához tartozó kód ***i.*** számjegyének címe:
    tömb kezdőcím + ***j****4 + ***i***

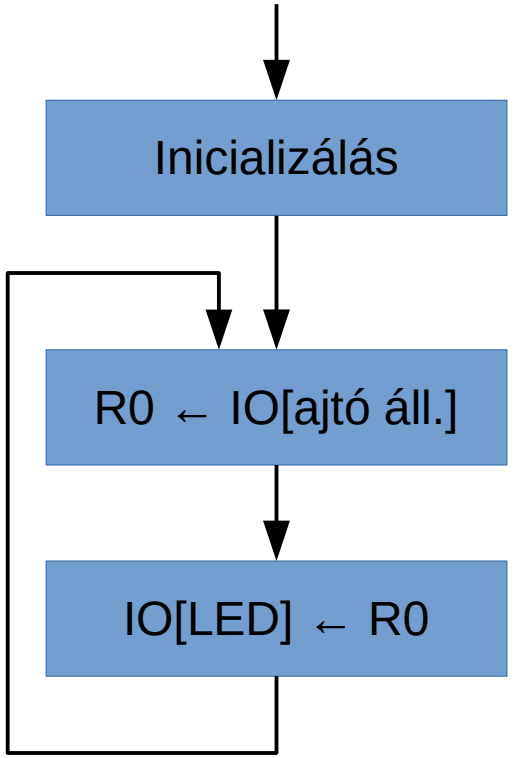
- Lokális változók:
  - ST: 1 bájtos egész szám
  - CARDID: 1 bájtos egész szám

ALGORITMUSOK

- A program két része:
  - Főprogram
  - Megszakításkezelő szubrutin
- A főprogram feladata:
  - Figyeli az ajtó nyitottsági állapotát
  - A LED-eket frissíti az aktuális helyzetnek megfelelően
- A megszakításkezelő feladata:
  - Lekezeli a kártyalehúzás és gombnyomás eseményeket
  - Kinyitja az ajtót, ha helyes a kód

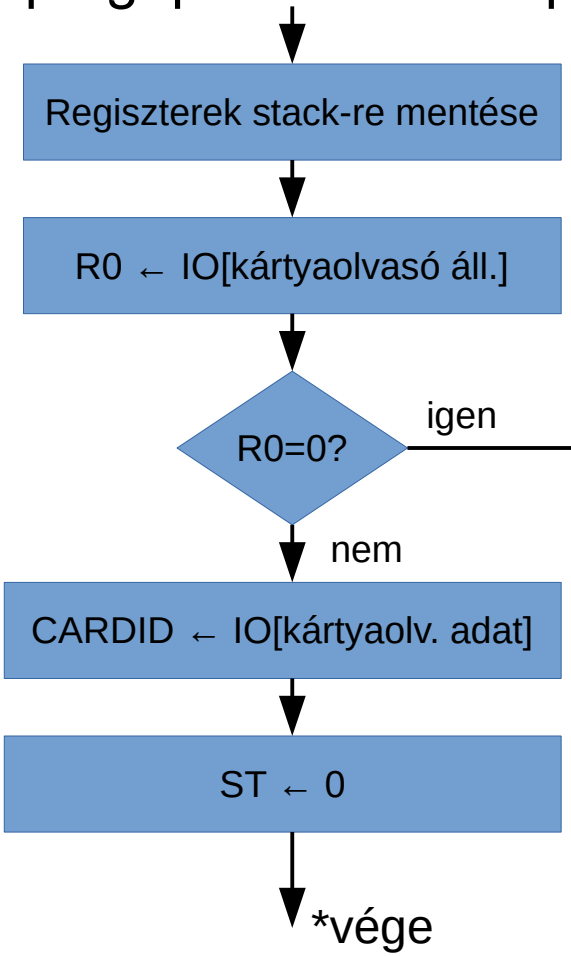
A FŐPROGRAM

- Inicializálás:
  - Stack pointer beállítása
  - Állapotváltozó inicializálása
  - A megszakításkezelés engedélyezése
- Végtelen ciklus:
  - Ajtó állapotának lekérdezése
  - LED-ek beállítása



A MEGSZAKÍTÁSKEZELŐ

- Először kideríti, hogy ki kért megszakítást (polling!)
- Végrehajtja az állapotgép következő lépését



Megnézzük, hogy a kártyaolvasó kérte-e a megszakítást

Tároljuk a kártyaszámot, az állapotgépet ST=0-ba állítjuk, és a megszakítás végére ugrunk

A megszakítás a billentyűzettől érkezett

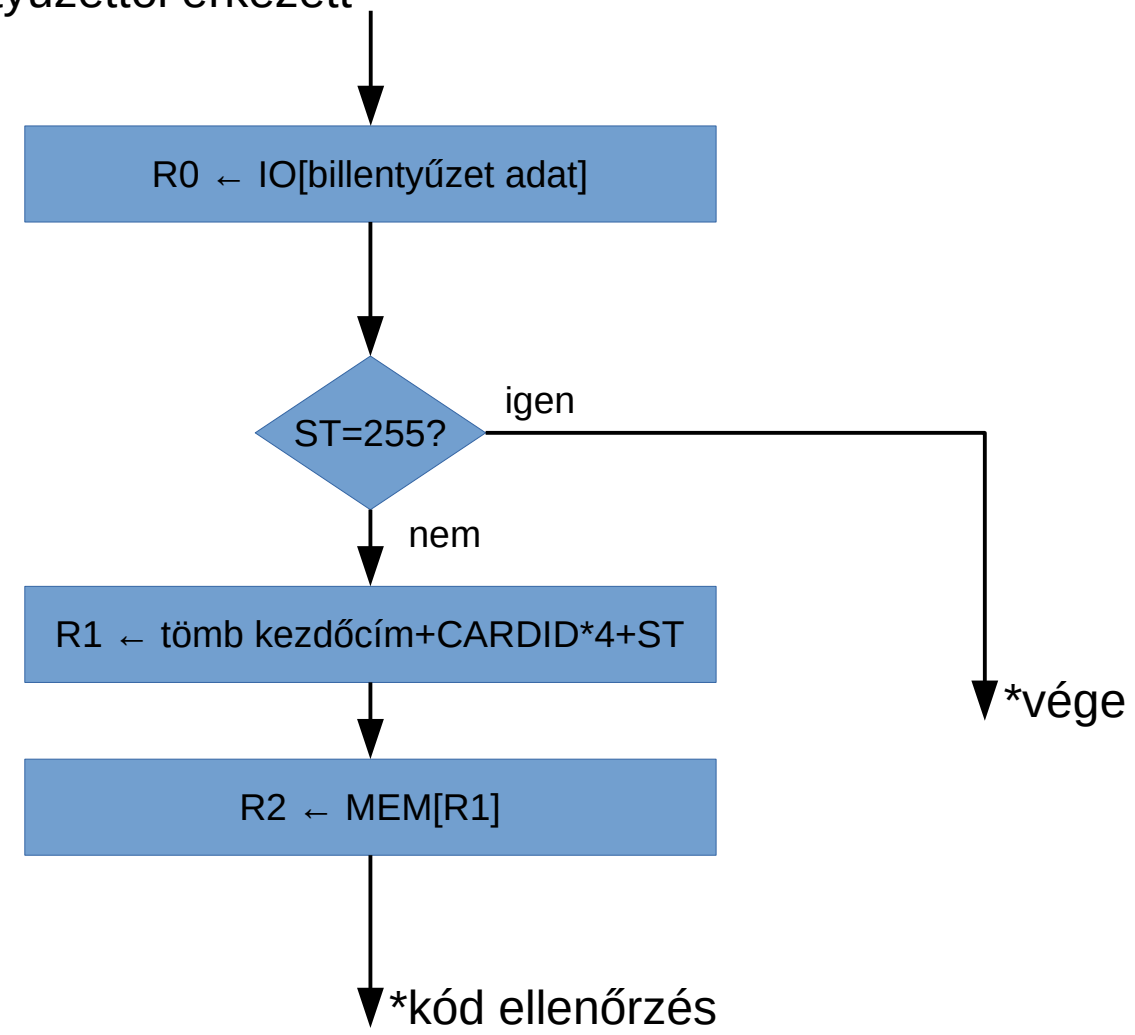
MEGSZAKÍTÁSKEZELŐ

A megszakítás a billentyűzettől érkezett

Beolvassuk az új számjegyet

Biztos, hogy számjegyet várunk?

A memóriából kiolvassuk a várt számjegyet

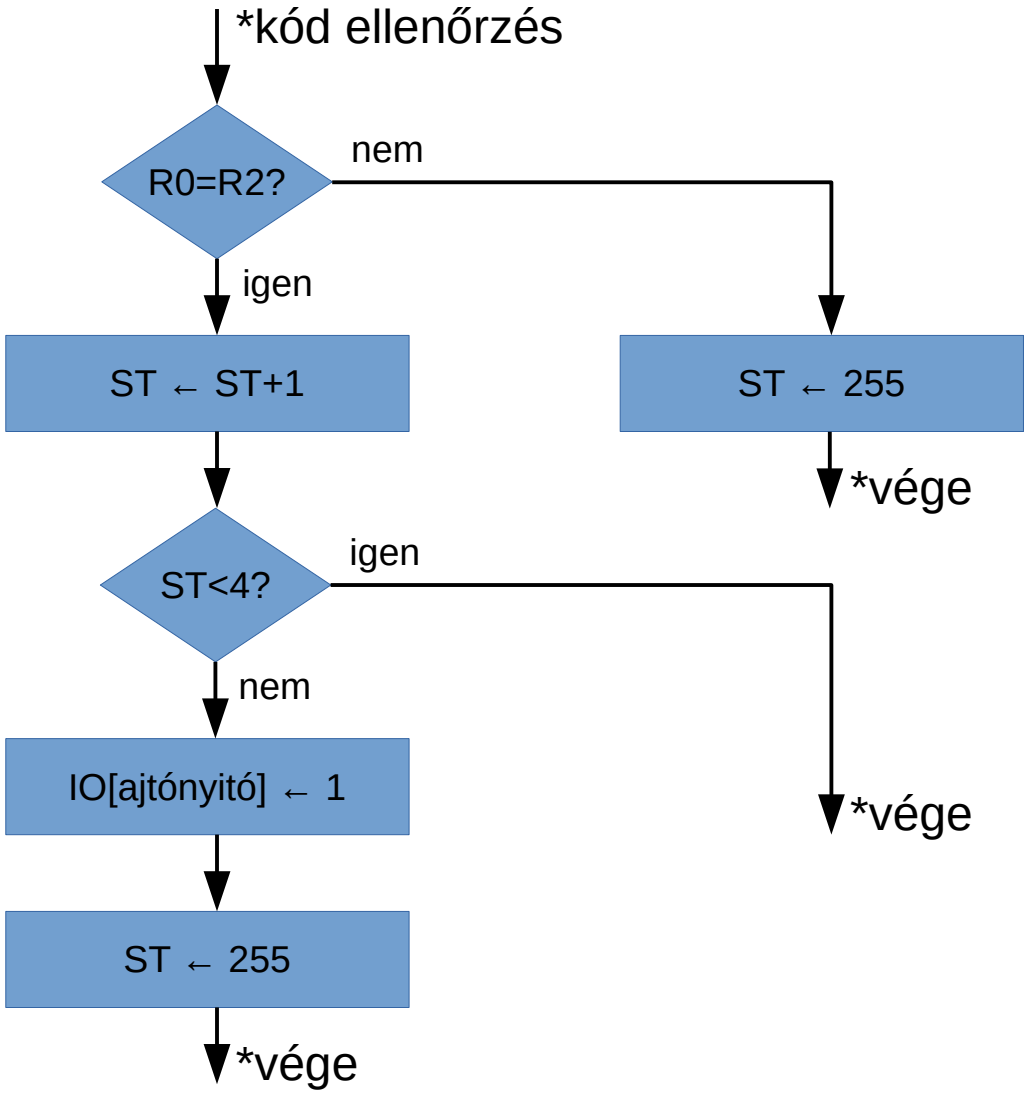


A MEGSZAKÍTÁSKEZELŐ

A várt számjegyet kaptuk?

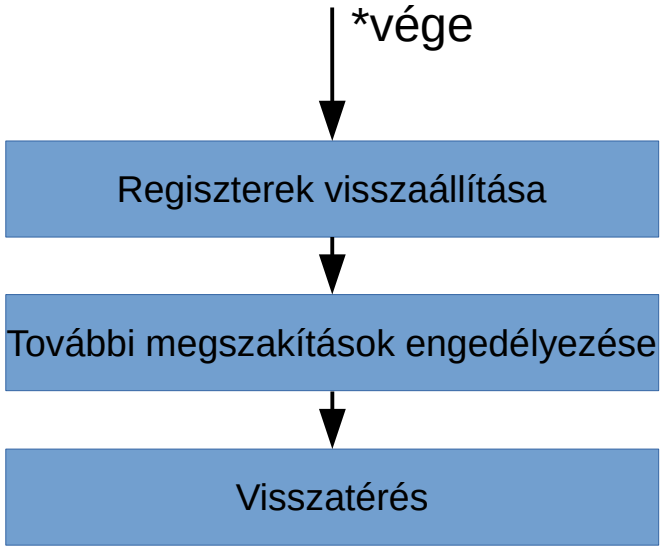
Ha igen, ST-t frissítjük.

Ha mind a 4 szám jó, nyitjuk az ajtót, és alapállapot



A MEGSZAKÍTÁSKEZELŐ

A megszakításkezelő vége:



ELHELYEZÉSI DÖNTÉSEK

- **A RAM tartalma**
  - ST (1 bájtos egész változó) címe: **2000h** (a RAM első bájtja)
  - CARDID (1 bájtos egész változó címe): **2001h** (a RAM második bájtja)
  - A stack mutató kezdő értéke: **3FFFh** (a RAM vége, hiszen visszafelé nő)
- **A ROM tartalma**
  - A főprogram utasításai, kezdőcím: **0000h**
  - A megszakításkezelő utasításai, kezdőcím: **1000h**
  - A helyes kódokat tartalmazó tömb:
    - Mérete: **400h**
    - Bárhová tehetjük, ahol van hely
    - Legyen a kezdőcíme **500h**
      - A főprogramunk kicsi, nem ér el **500h**-ig
      - A tömb vége így **900h**, vagyis nem lóg bele a megszakításkezelő kódjába

ASSEMBLY MEGVALÓSÍTÁS

- A főprogram megvalósítása

```
        ORG 0000h              itt kezdődik az elhelyezés
        SP ← 3FFFh             stack pointer beállítás
        MEM[2000h] ← 255       ST = 255 (alapállapot)
        EI                     interrupt engedélyezés
label:  R0 ← IO[4Ch]           ajtó állapot lekérdezése
        IO[5Dh] ← R0           LED-ek beállítása
        JUMP label             ugrás vissza, új kör
```

ASSEMBLY MEGVALÓSÍTÁS

- A 4 bájtos kódok tömbje

```
ORG 500h
codes: DB 1, 3, 4, 7
       DB 5, 7, 2, 9
       DB 8, 2, 0, 8
       DB 3, 1, 8, 9
       ...
```

ASSEMBLY MEGVALÓSÍTÁS

- A megszakításkezelő megvalósítása:

`ORG 1000h`	CPU ide ugrik megszakítás esetén
`PUSH R0`	R0/R1/R2 regiszter stack-re mentése
`PUSH R1`	
`PUSH R2`	
`R0 ← IO[2Eh]`	kártyaolvasó állapotának lekérdezése
`JUMP keycode IF R0==0`	ugrunk, ha nem az okozott megszakítást
`R0 ← IO[2Fh]`	kártyaszám beolvasása
`MEM[2001h] ← R0`	… és eltárolása a CARDID változóba
`MEM[2000h] ← 0`	ST=0
`JUMP end`	ugrás a megszakításkezelő rutin végére

ASSEMBLY MEGVALÓSÍTÁS

- A megszakításkezelő megvalósítása:

```
keycode: R0 ← IO[3Fh]          R0=érkezett számjegy
         R1 ← MEM[2000h]       R1=ST
         JUMP end IF R1==255   Ugrunk, ha épp nem számjegyre várunk
         R2 ← MEM[2001h]       R2=CARDID
         R2 ← R2 * 4           R2=4*CARDID
         R2 ← R1 + R2          R2=4*CARDID+ST
         R2 ← MEM[R2+codes]    R2=a várt számjegy
         JUMP match IF R2==R0  Ugrunk, ha a számjegy helyes
         MEM[2000h] ← 255      ha nem helyes, az alapállapotba lép
         JUMP end              ...és ugrunk a megszakításkezelő végére
match:   R1 ← R1+1             Ha helyes, növeljük ST-t,
         MEM[2000h] ← R1       ... és elmentjük a memóriába
         JUMP end IF R1<4      Ha nem ez az utolsó számjegy, vége
         IO[4Ch] ← 1           Ha ez volt az utolsó, ajtónyitás
         MEM[2000h] ← 255      Alapállapotba lépünk
         JUMP end              Ugrás a megszakításkezelő végére
```

ASSEMBLY MEGVALÓSÍTÁS

- A megszakításkezelő megvalósítása:

```
end: POP R2        regiszterek visszaállítása fordított sorban
     POP R1
     POP R0
     EI            további megszakítások engedélyezése
     RET           visszatérés a szubrutinból
```

VARIÁCIÓK

- Nem lehet RAM nélkül is megoldani?
  - De.
  - De ekkor elveszítjük a stack-et
    - És ezzel a megszakításkezelést is
    - A főprogramnak kell folyamatosan figyelnie:
      - Az ajtó állapotát (a LED-ek miatt)
      - A kártyaolvasó állapotát
      - A billentyűzet állapotát
  - És elveszítjük a lokális változóinkat is (ST és CARDID)
    - Ezt a kettőt szabad regiszterekben is tárolhatjuk

HÁLÓZATI RENDSZEREK
ÉS SZOLGÁLTATÁSOK
TANSZÉK