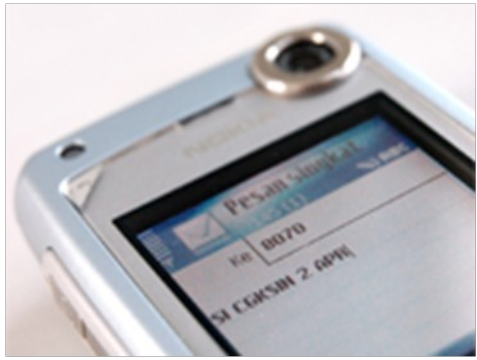




Számítógép architektúrák

Kártyás ajtónyitó tervezése

Mészáros András

BME Hálózati Rendszerek és Szolgáltatások Tanszék
meszarosa@hit.bme.hu

2025. febr. 12.
Budapest

Technikai jellegű kérdések

- Óra kezdése (2 hetente)
- Katalógus/hiányzás/egyéb információ
  http://www.hit.bme.hu/~ghorvath/szgarch/
- Gyakorlatok anyaga:
  http://webspn.hit.bme.hu/~mandras/szga.htm
- “The only stupid question is the one you don't ask.”

Miről lesz szó a félév során?

- Egyszerű rendszer összeállítása (elektronikus beléptető)
- (Perifériák)
- Háttértár (lassú, állandó memória)
  - HDD
  - SSD
- Memória
  - RAM
  - Virtuális memória
- Cache memória (gyors memória)
- Processzor
  - CPU pipeline
  - Elágazásbecslés

A gyakorlat célja

- Egyszerű rendszer összeállítása (elektronikus beléptető)

- „Digitális technikás” megközelítés:
  - Idealizált elemek (CPU, memória, D tároló, stb.)
  - Assembly programozás
  - Digitális technika felfrissítés
  - Ma nem így szokás tervezni

- Mikrokontrolleres megvalósítás:
  - Arduino alapú rendszer
  - Magas szintű programozás
  - Hogy működik egy korszerű megvalósítás?

Tartalomjegyzék

- A feladat specifikációja
- A hardver megtervezése
  - Alkatrészek
  - Memória illesztése
  - Perifériák illesztése
- A szoftver megtervezése
  - Állapotgép modell
  - Folyamatábrák
  - Implementáció

A feladat specifikációja

Elektronikus ajtózár

- Kulcs helyett elektronikusan nyitható ajtó

- Beléptető rendszer:
  - Kártya (RFID)
  - 4 jegyű kód minden kártyához (billentyűzet)

- Zárt állapot: piros LED
- Helyes kód: ajtó nyitása + zöld LED

- Beléptetés pontos módja később (pl. rossz kód)

A hardver megtervezése

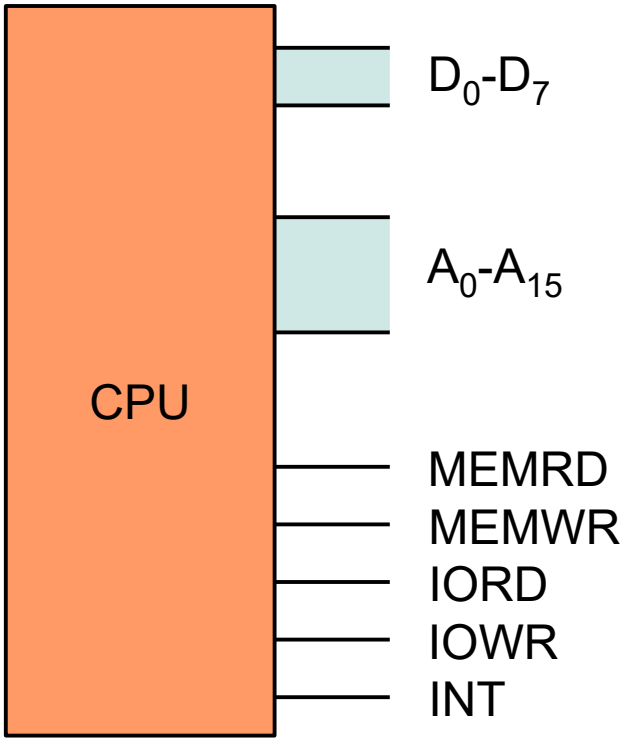
Hexadecimális számábrázolás

- Decimális (10 ujj)
- Bináris (0/1) – gépnek természetes, de hosszú
- Hexadecimális (0,...,9,A,...,F)
  - 0xE2, E2h
  - 4 bit (2^4) 1 hexa: 0001 → 1h
  - 1101 1011 0011 1001 → DB39h
- Példában 16 bites cím: 0000h-FFFFh

Szükséges elemek

- CPU (mikroprocesszor)
- Memória:
  - ROM (programkód, konstans adat (pl. kártya kódja))
  - RAM (változók, stack)
- Perifériák:
  - Kártyaolvasó
  - Billentyűzet
  - Ajtónyitó
  - LED-ek
- Dekóder/demultiplexer (memóriaillesztés)
- D tároló, 2 ellenállás (LED-ek működtetése)
- 2 komparátor (perifériák címének kiválasztása)

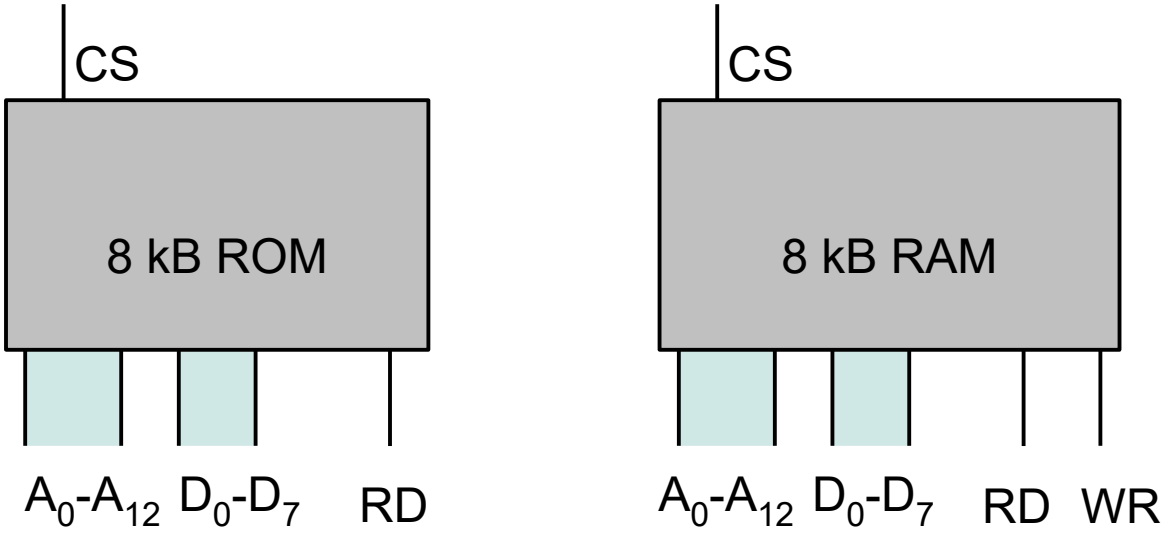
Alkatrészek



- 8 bites CPU
  - Adatbusz: 8 bit (1 byte) széles
  - Címbusz: 16 bit széles
  - Multiplexált memória és perifériabusz
  - Kezdőcím: **0000h**
  - Megszakításkezelő címe: **1000h**
  - RISC utasításkészlet
  - 3 operandusú utasítások

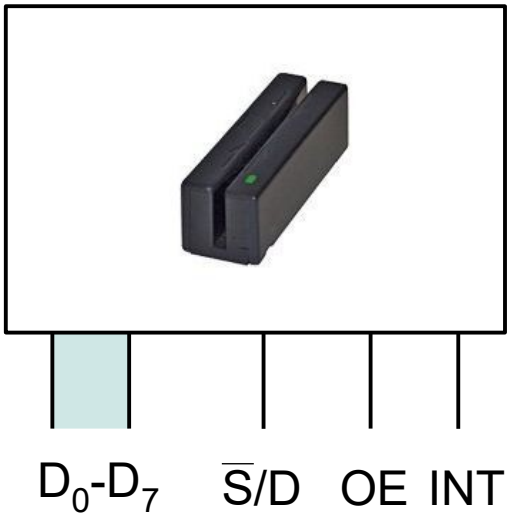
Alkatrészek

- 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



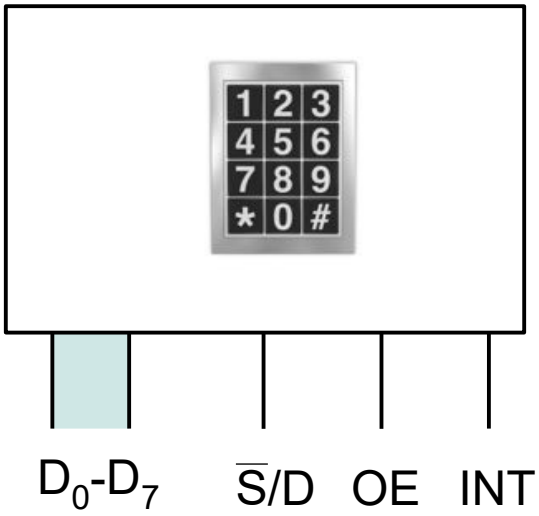
Alkatrészek

- Kártyaolvasó
  - OE: output enable
  - Van egy 8 bites 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



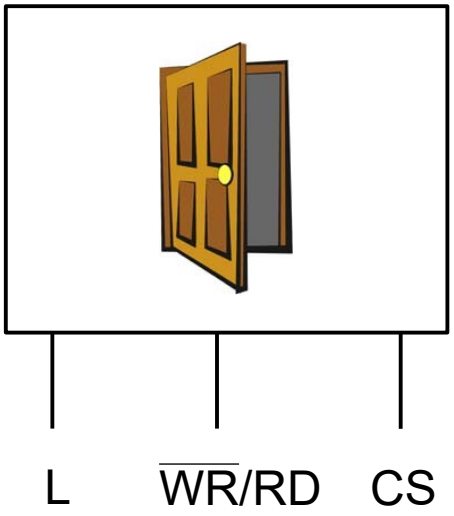
Alkatrészek

- 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



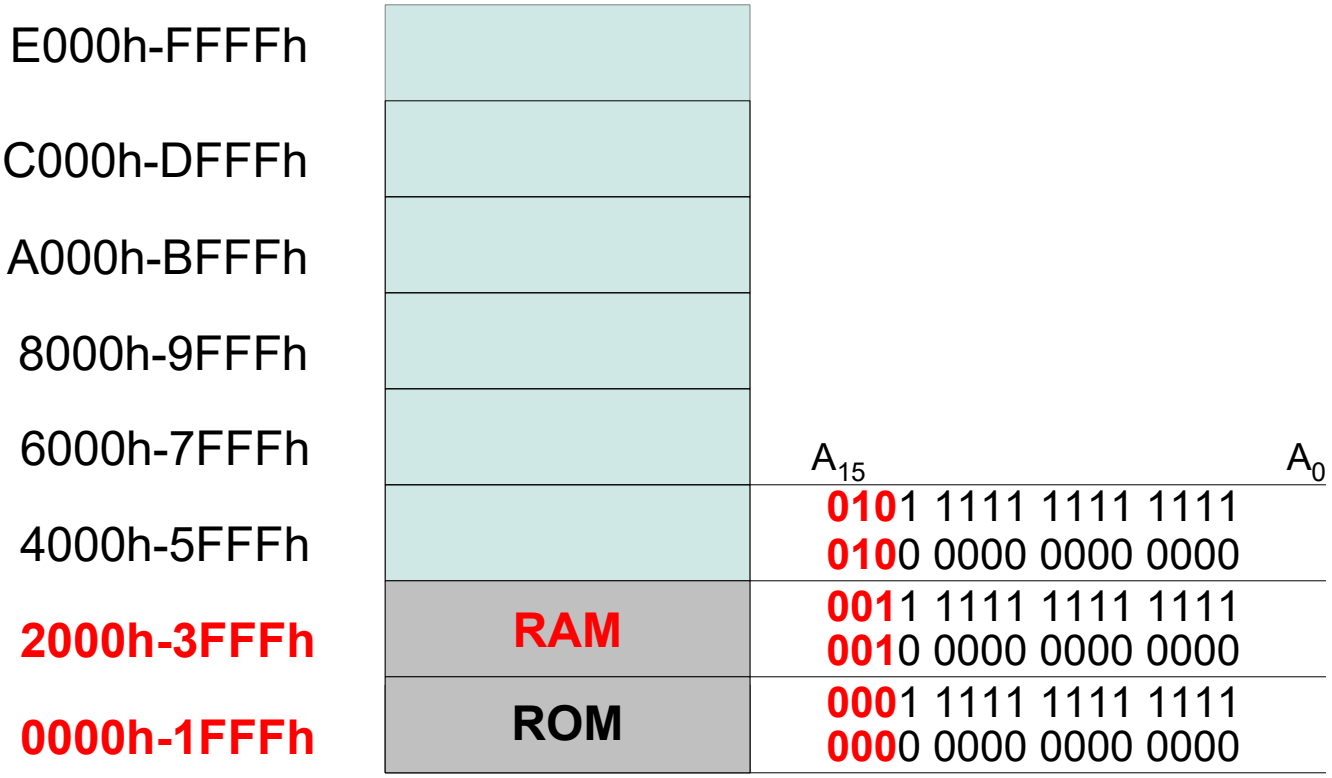
Alkatrészek

- 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 lejárta után automatikusan újra bezár.



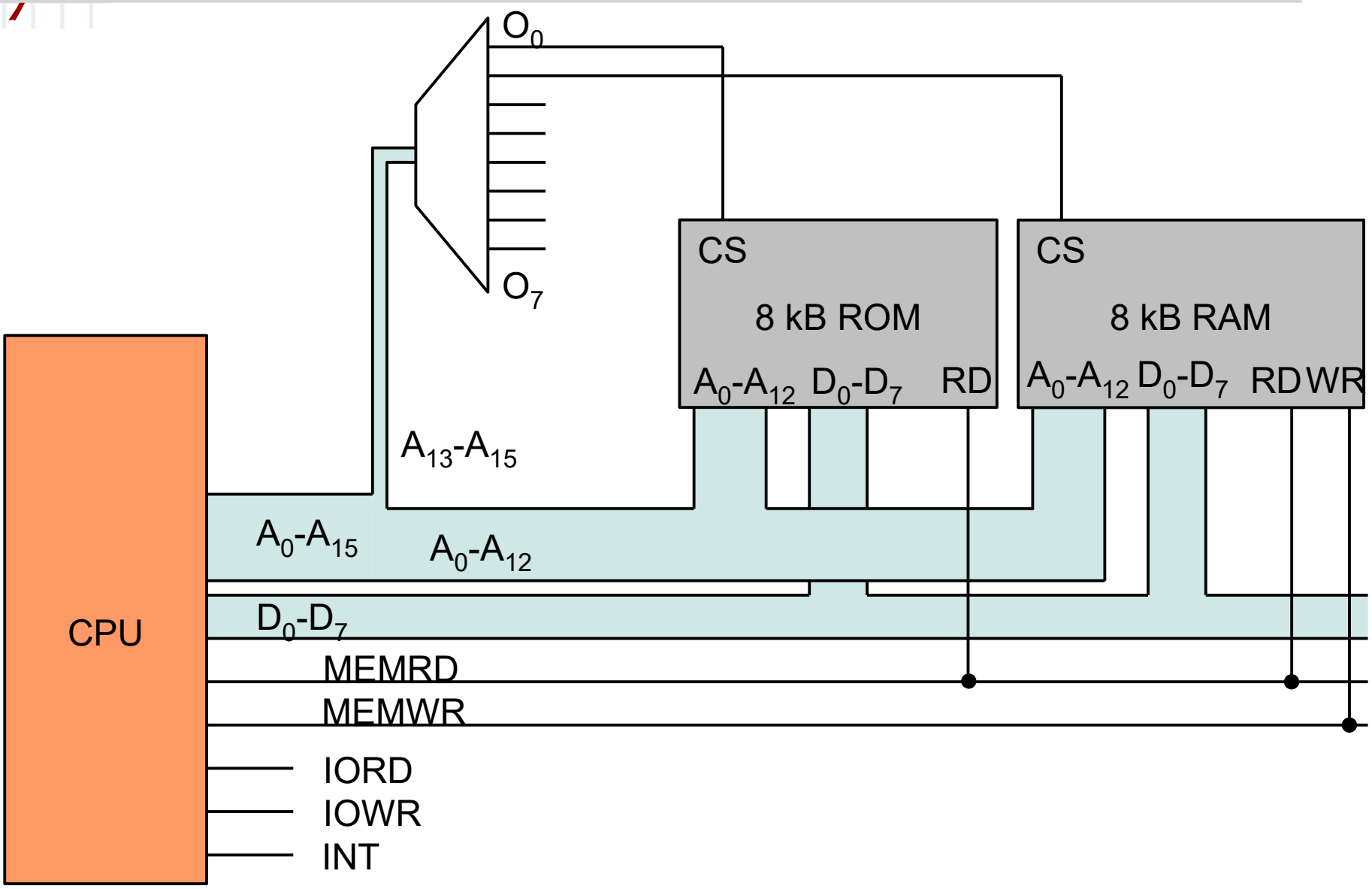
Memóriaillesztés

- 64 kB-nyi cím - 8 kB ROM + 8 kB RAM
- Memóriatérkép:

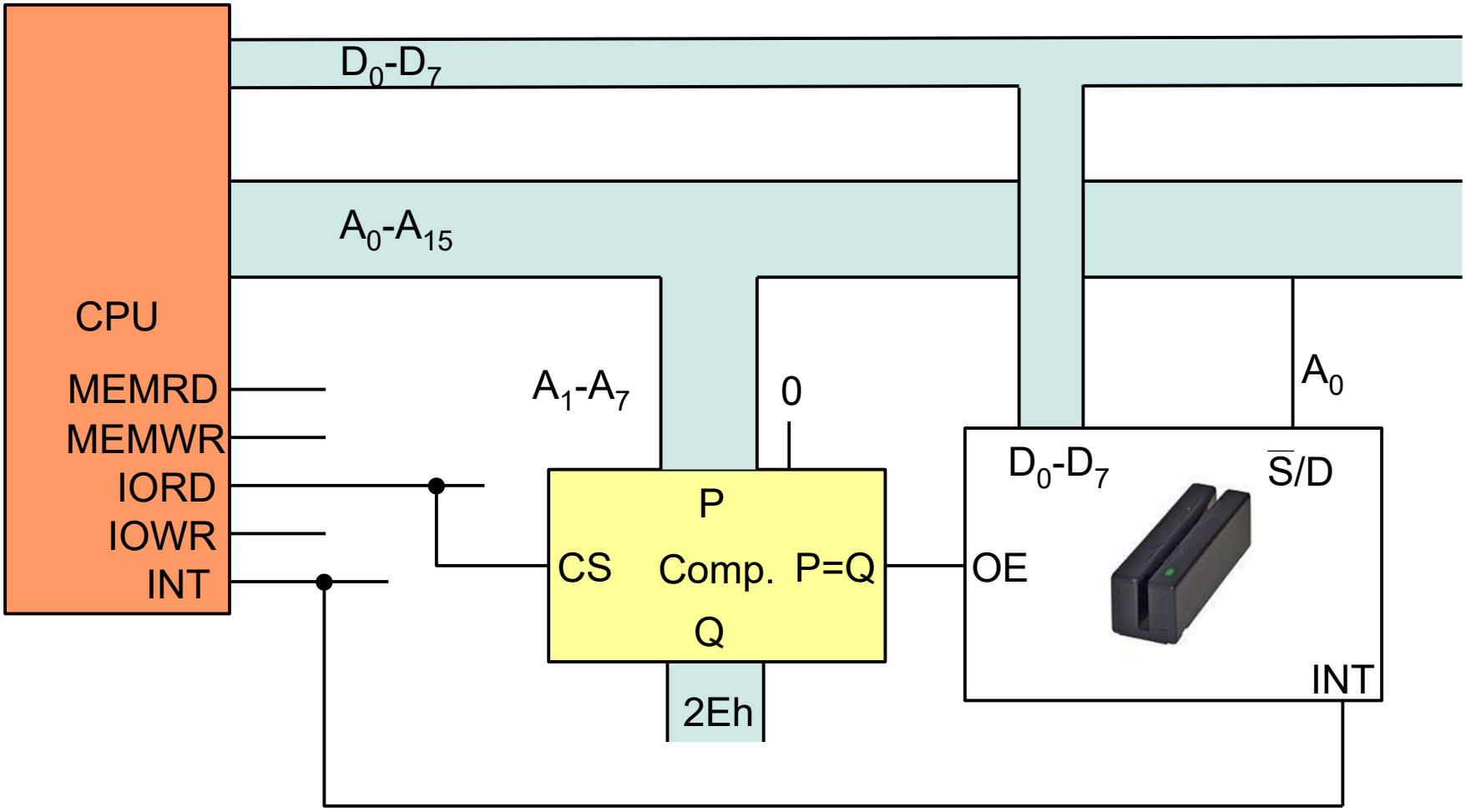


- A_{13}-A_{15} meghatározza, hogy melyik modulra vonatkozik a cím
- A_0-A_{12} a modulon belüli pozíció

Memóriaillesztés

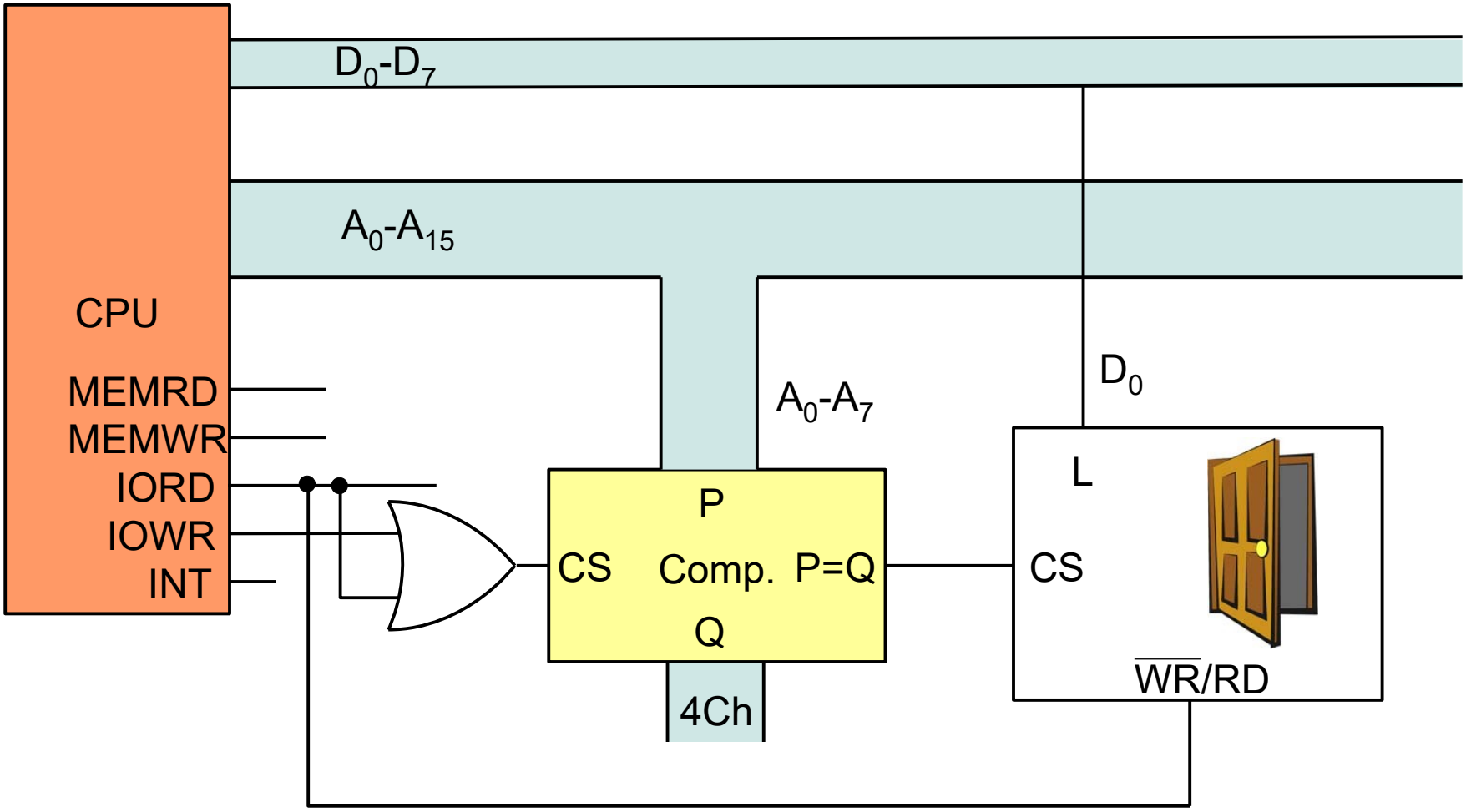


A kártyaolvasó illesztése

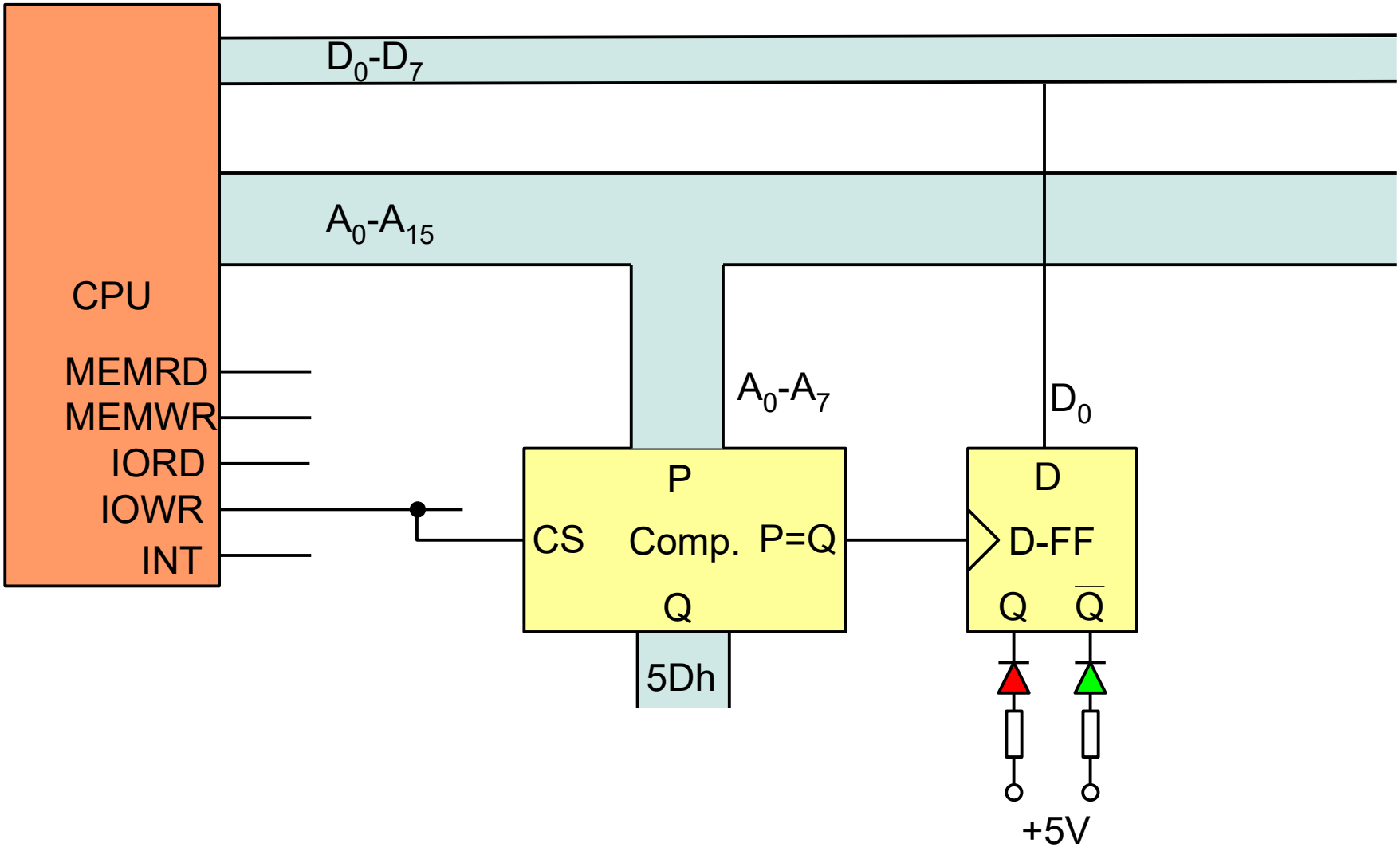


- A billentyűzetet ugyanígy illesztjük, de azt a 3Eh báziscímre

Az ajtónyitó illesztése



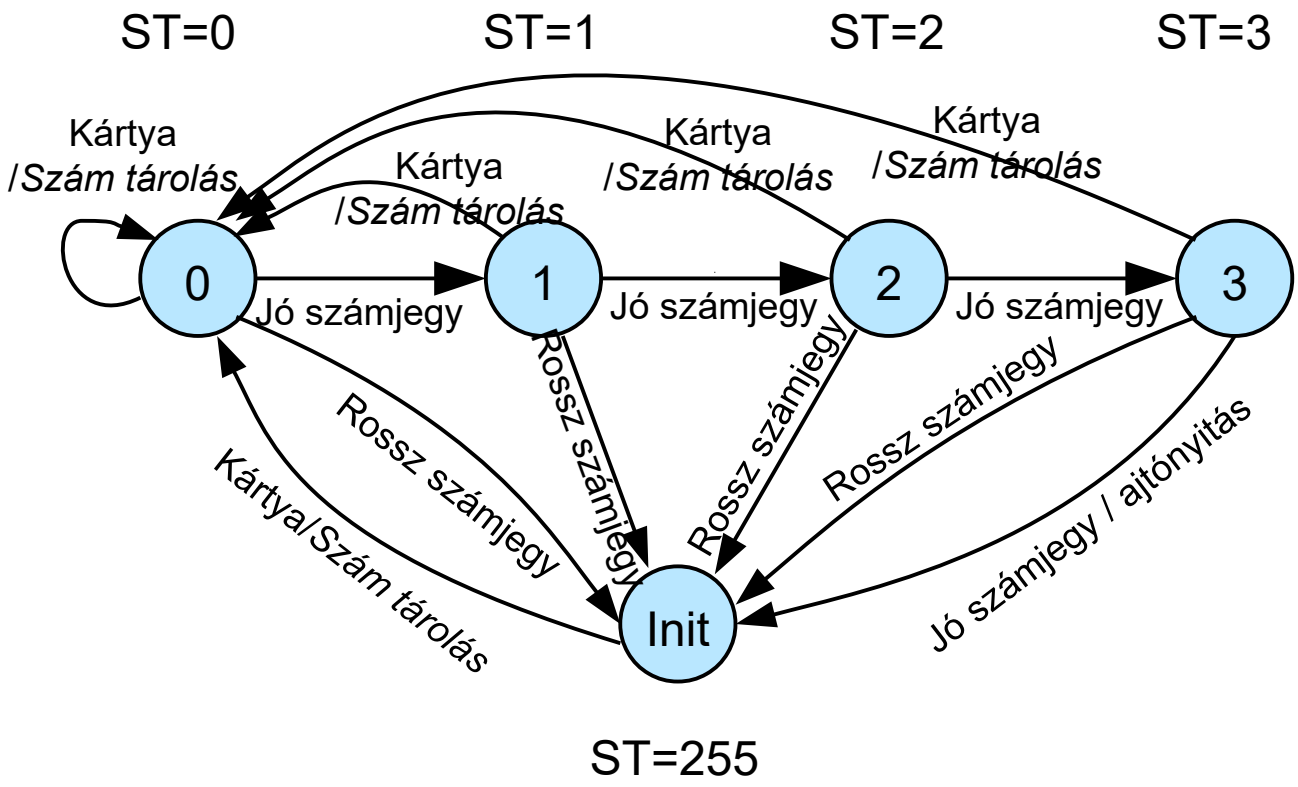
A LED lámpák illesztése



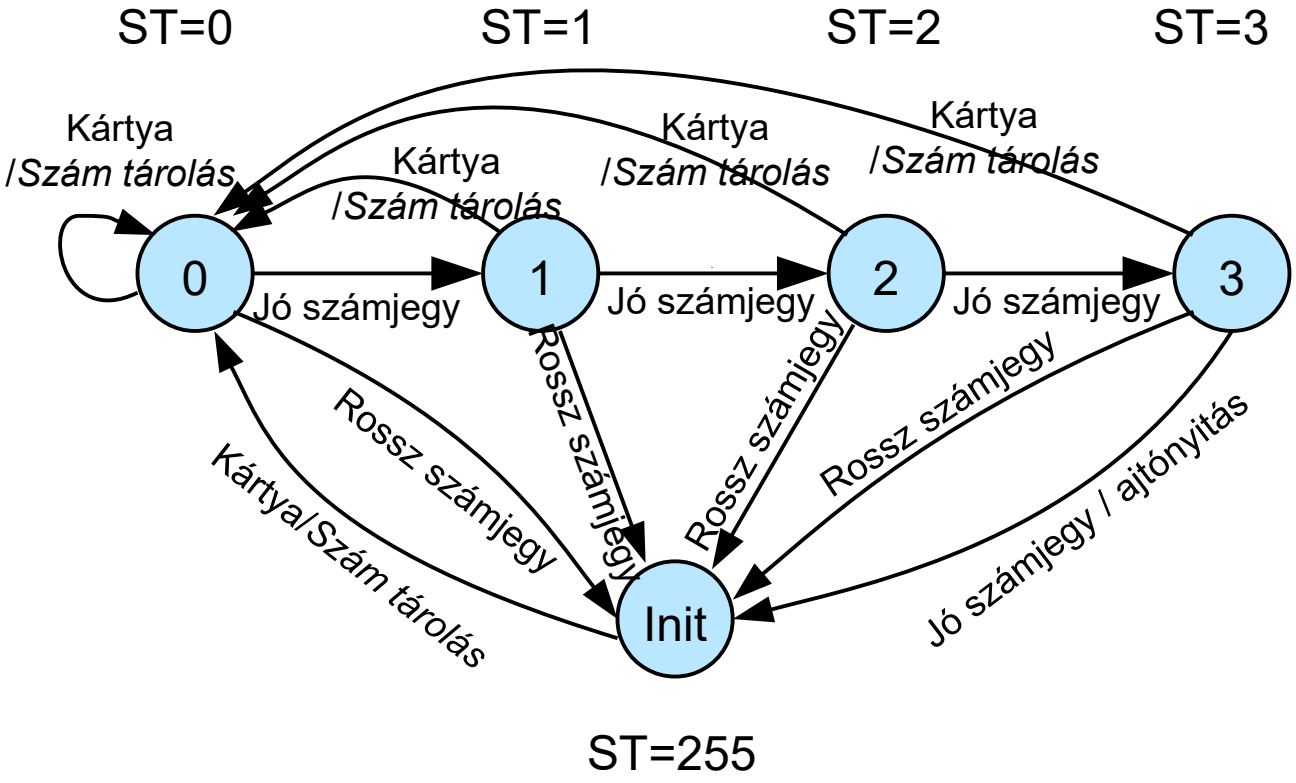
A szoftver

Állapotgép modell

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



Állapotátmenetek



- Kártyalehúzáskor: ST=0, és kártyaszám tárolása 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*

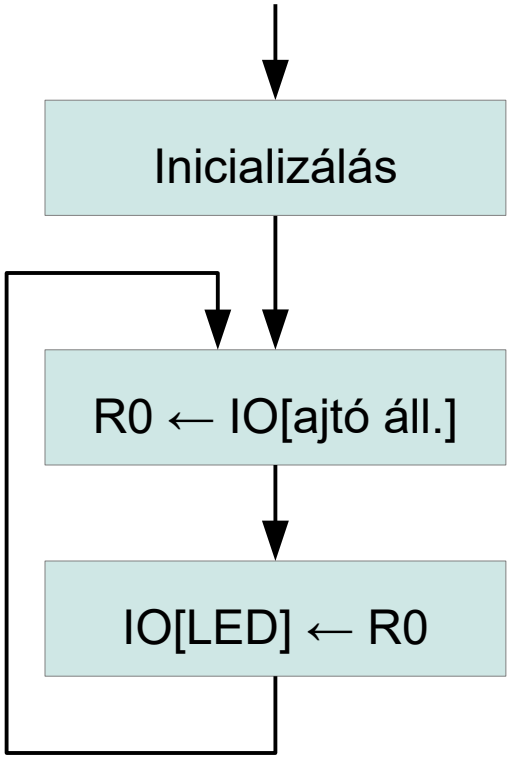
- Lokális változók:
  - ST: 1 bájtos egész szám
  - CARDID: 1 bájtos egész szám

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

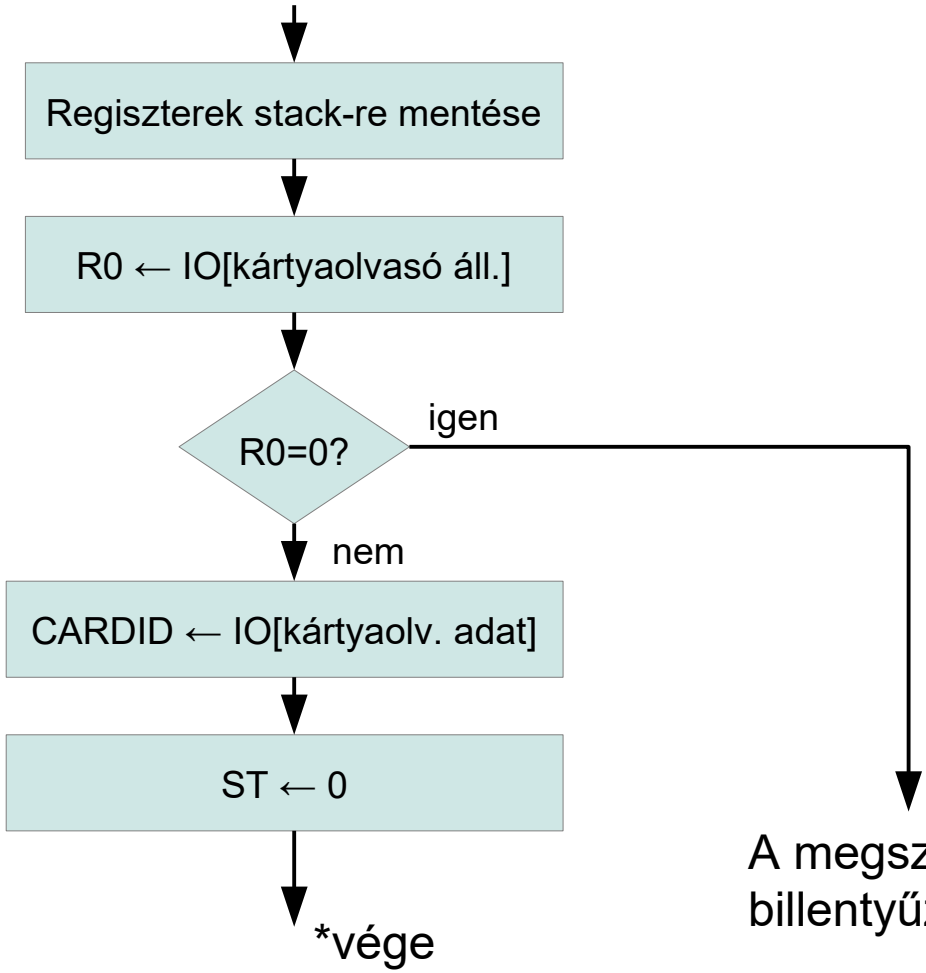
A főprogram

- 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

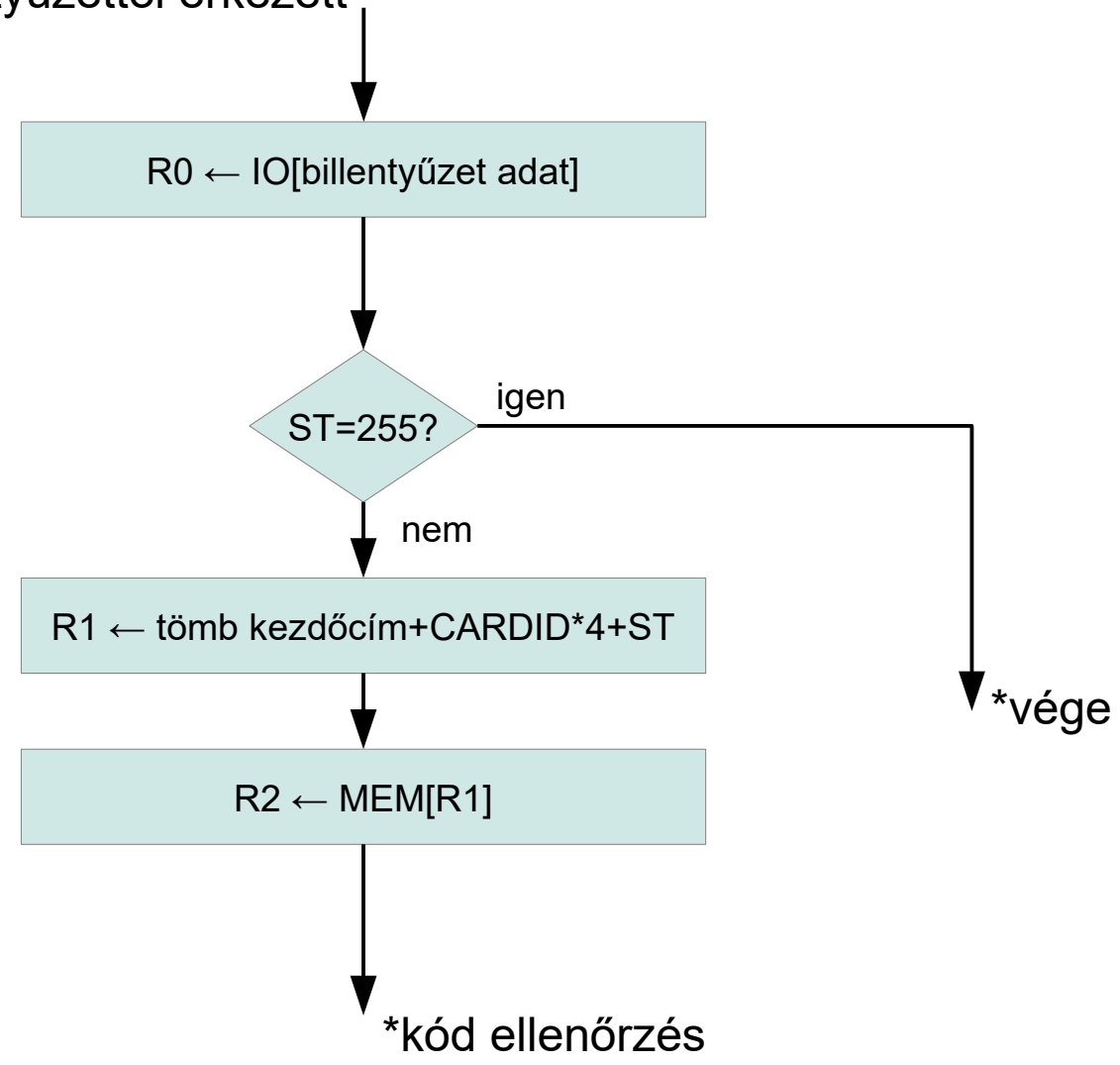
A megszakításkezelő

A megszakítás a billentyűzettől érkezett

Beolvassuk az új számjegyet

Biztos, hogy számjegyet várunk?

A memóriából kiolvassuk a várt számjegyet

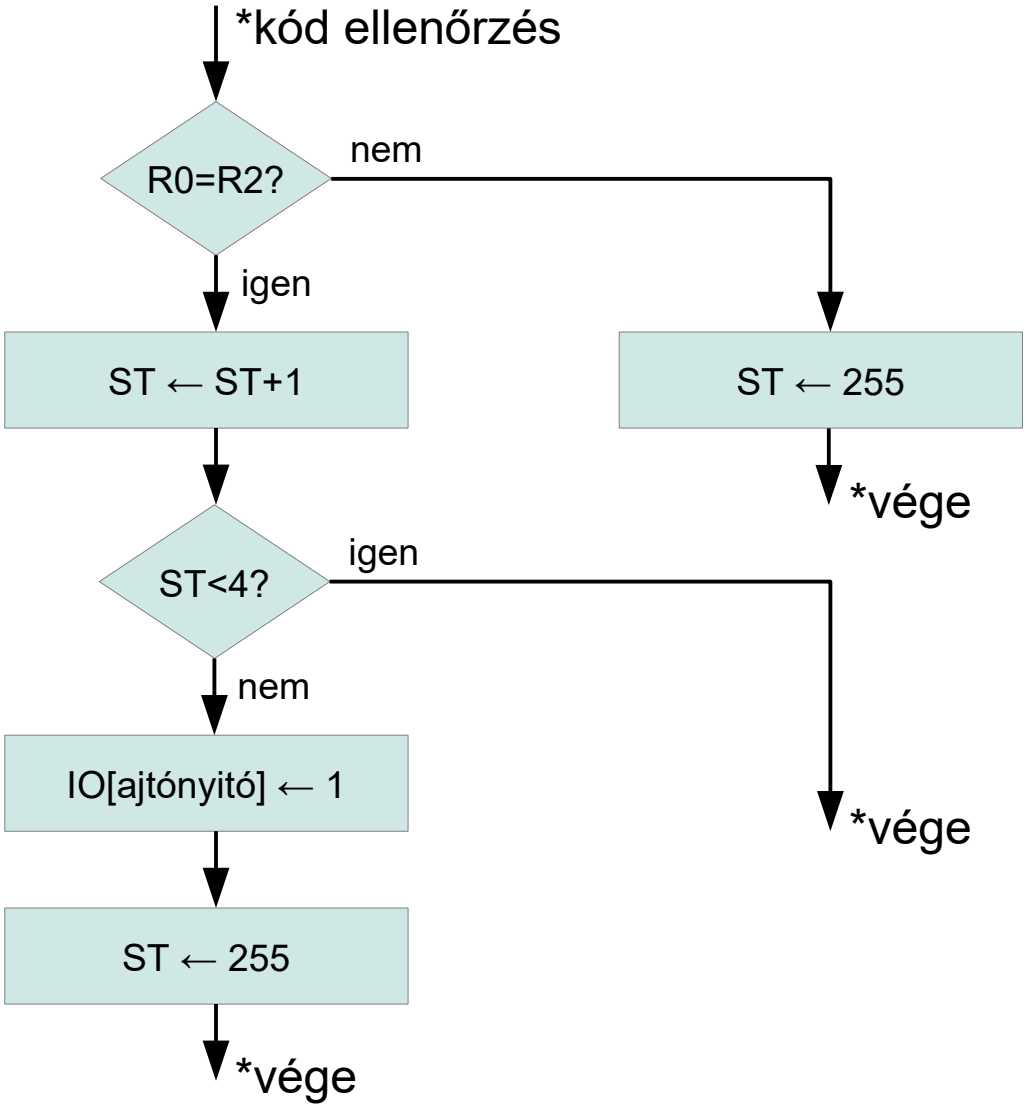


A megszakításkezelő

A várt számjegyet kaptuk?

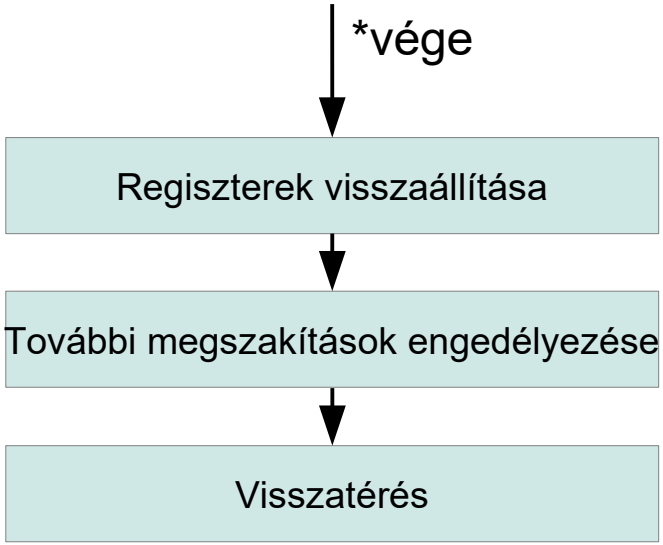
Ha igen, ST-t frissítjük.

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 stacket
    - 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