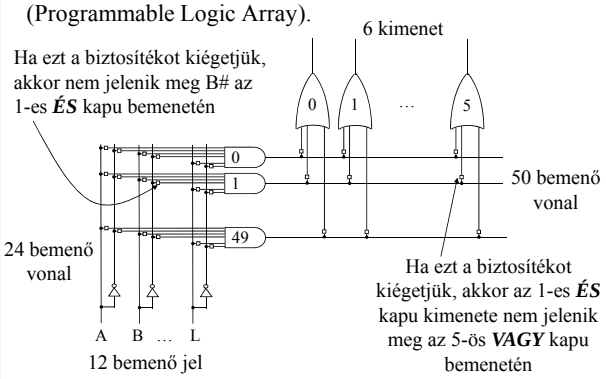
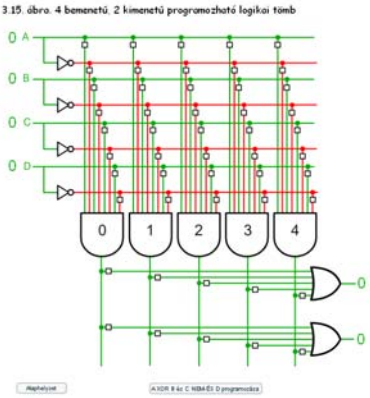
- **Programozható logikai tömbök: PLA (3.15. ábra)** (Programmable Logic Array).



Máté: Architektúrák 3. előadás 1



Máté: Architektúrák 3. előadás 2

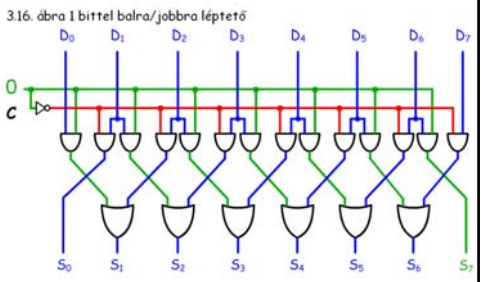
Aritmetikai áramkörök

A kombinációs áramkörökön belül külön csoportot alkotnak.

Léptető:

C=1: jobbra,
C=0: balra léptet.

3.16.swf

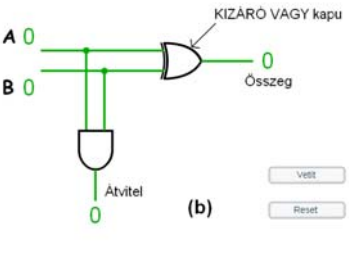


Máté: Architektúrák 3. előadás 3

Összeadók:

3.17. ábra (a) 1 bites összeadás igazságtáblája. (b) Fél összeadó áramkör

A	B	Összeg	Átvitel
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



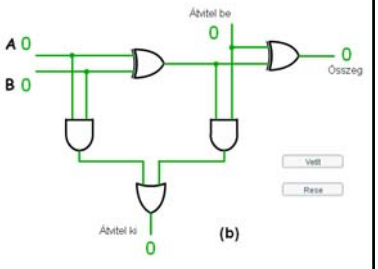
Fél-összeadó (half adder, **3.17. ábra,** 3.17.swf)

Máté: Architektúrák 3. előadás 4

Összeadók:

3.18. ábra (a) A teljes összeadó igazságtáblázata (b) Teljes összeadó áramkör

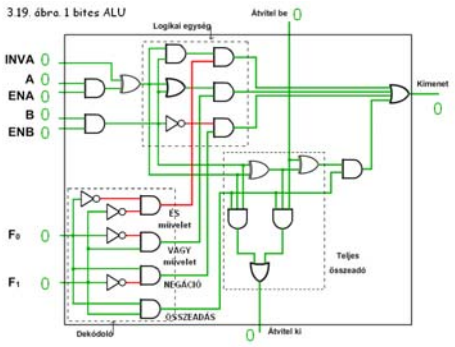
A	B	Átvitel be	Összeg	Átvitel ki
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Teljes-összeadó (full adder, **3.18. ábra,** 3.18.swf)

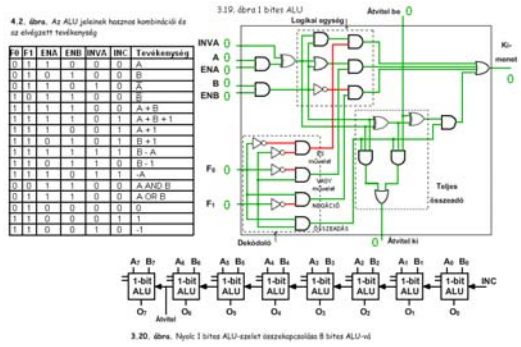
Máté: Architektúrák 3. előadás 5

Aritmetikai-logikai egység: bitszelet (bit slice, **3.19. ábra**), **F0**, **F1** -től függően ***ÉS***, ***VAGY***, ***NEGÁCIÓ*** vagy +



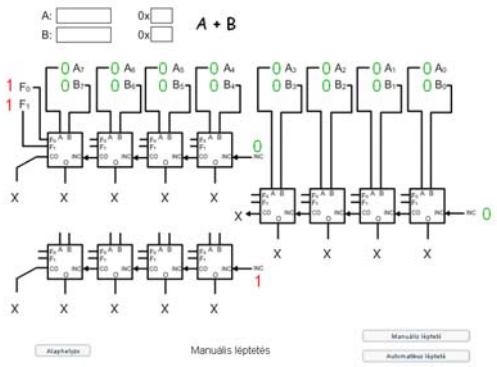
Máté: Architektúrák 3. előadás 6

- átvitel továbbterjesztő összeadó (ripple carry adder):



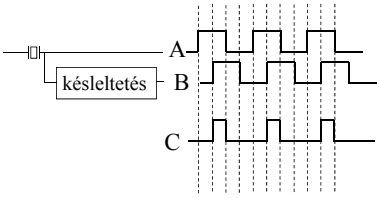
3.19b.swf

- átvitel kiválasztó összeadó (carry select adder) eljárás:



Nem kombinációs áramkörök

Óra (clock, 3.21. ábra): ciklusidő (cycle time).
Pl.: 500 MHz - 2 nsec.
Finomabb felbontás késleltetéssel.
Aszimmetrikus óra.

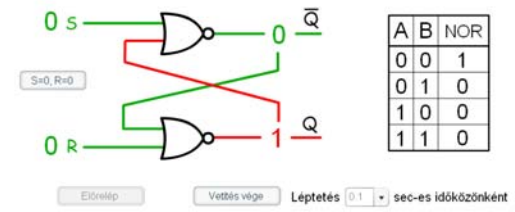


Memória: „Emlékszik” az utolsó beállításra.

Tároló: Szint vezérelt (level triggered).

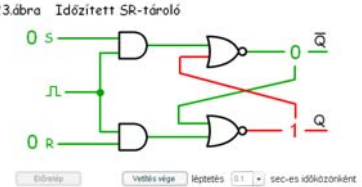
SR tároló (Set Reset latch, **3.22. ábra,** 3.22.swf).
Stabil állapot: a két kimenet **0**, **1** vagy **1**, **0**.
S (set), **R** (reset) bemenet. (Q# ≡ $\overline{Q}$)

3.22.ábra NEM-VAGY tároló



A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

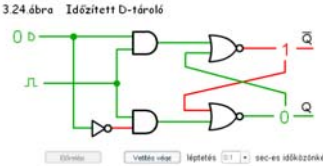
3.23.ábra Időzített SR-tároló



A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

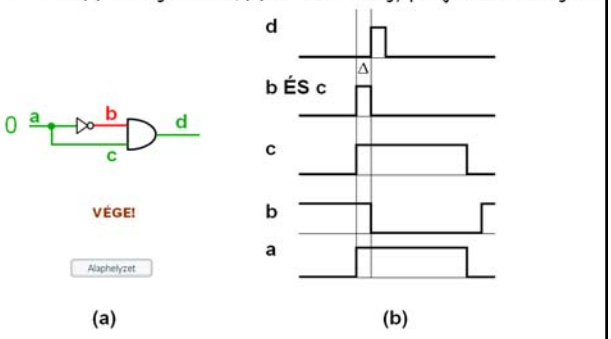
Mindkét **SR** tároló indeterminisztikussá válna, ha **S = R = 1** egyszerre fordulna elő.

3.24.ábra Időzített D-tároló



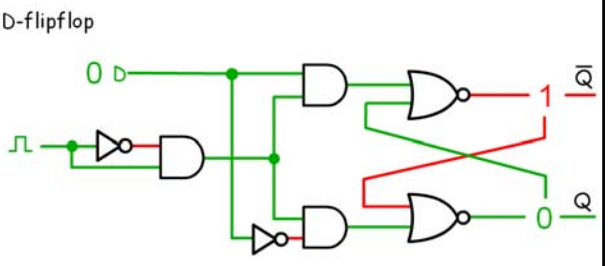
A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

3.25.ábra (a) Pulzusgenerátor, (b) Az áramkör négy pontjának az időd

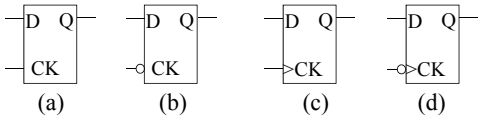


Az inverternek van egy pici (1-10 ns) késleltetése (Δ).

Flip-flop: élvezérelt (edge triggered), **D** flip-flop: **3.26. ábra**.



3.27. ábra: Tárolók és flip-flopok



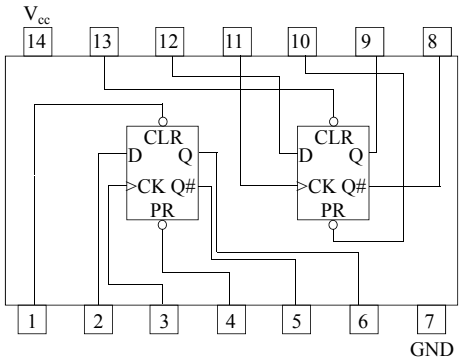
CK: órajel

(a) **CK=1**, (b) **CK=0** szint esetén írja be **D**-t,

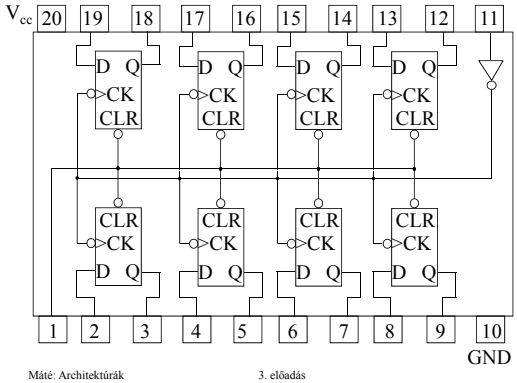
(c) **CK** emelkedő, (d) **CK** lefelé menő élénél.

Sokszor **S** (set, **PR** preset), **R** (reset,**CLR** clear) be- és **Q#** kimenet is van.

3.28. ábra: (a) 2 független D flip-flop,



3.28. ábra: (b) közös **CK**-val és **CLR**-rel vezérelt 8 bites D flip-flop: **regiszter**



Alapvető digitális logikai áramkörök

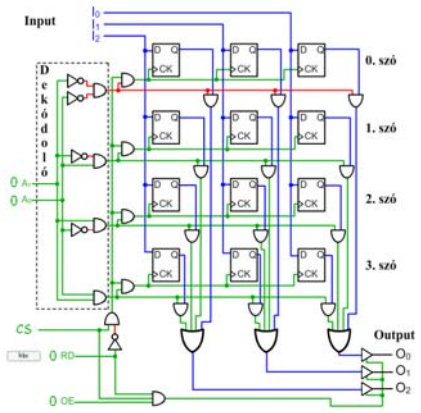
Integrált áramkör (**IC**, Integrated Circuit, chip, lapka) 5x5 mm^2 szilícium darab kerámia vagy műanyag lapon (tokban), lábakkal (pins). Négy alaptípus:

- **SSI** (Small Scale Integrated 1-10 kapu),
- **MSI** (Medium Scale ..., 10-100 kapu),
- **LSI** (Large Scale..., 100-100 000 kapu),
- **VLSI** (Very Large Scale ..., > 100 000 kapu).

Memória szervezése

Elvárás: szavak címezhetősége.

3.29. ábra: Négy db három bites szó. Bemenetek:
- három a vezérléshez,
  - **CS** (Chip Select): lapka választás,
  - **RD** (ReaD): 1: olvasás, 0: írás választása,
  - **OE** (Output Enable): kimenet engedélyezése.
- kettő a címzéshez (dekódoló),
- három a bemenő adatoknak,
- három adat kimenet.



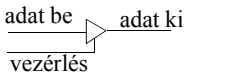
3.29. ábra.

4 × 3-as memória

3.29.swf

Memória szervezése

Az igazi memóriáknál a bemenet és kimenet közös (kevesebb lábra van szükség): Nem invertáló és invertáló pufferek (ezek három állapotú eszközök, **tri-state device**, **3.30. ábra**).

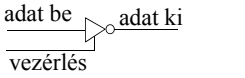


nem invertáló puffer

Ha a vezérlő jel

magas

alacsony



invertáló puffer

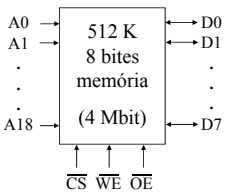
Ha a vezérlő jel

magas

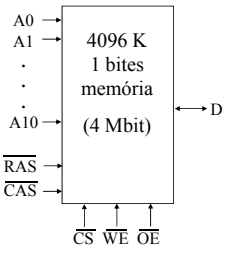
alacsony

Memórialapkák

Előnyös, ha a szavak száma **2 hatvány**.
4 Mbit-es memória kétféle szervezése: **3.31. ábra.**



19 cím, 8 adat vonal



11 cím, 1 adat vonal

Row **A**ddress **S**trobe
Column **A**ddress **S**trobe

Memórialapkák

A jel (bemenet) **beállított** (asserted) vagy **negált**.
CS beállított: **1**, de **CS#** beállított: **0**

a) **512 K** bájtos elrendezés: **19** cím, **8** adat vonal.

b) **2048*2048** bites elrendezés: **11** cím, **1** adat vonal: Bit kiválasztás sor- (**RAS**: Row Address Strobe) és oszlopindex **CAS** (Column ...) segítségével. Gyakran alkalmazzák nagyobb memóriáknál, bár a két cím megadása lassíthat.

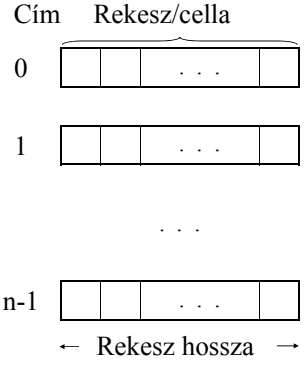
Nagyobb memóriáknál **1**, **4**, **8**, **16** bites kimeneteket is használnak.

Központi memória (2.9. ábra)

A programok és adatok tárolására szolgál.

Bit: a memória alapegysége, egy 0-t vagy 1-et tartalmazhat.

Memória rekesz (cella): több bit együttese. Minden rekesz ugyanannyi bitből áll. Minden rekeszhez hozzá van rendelve egy szám, a **rekesz címe**. Egy rekeszre a címével hivatkozhatunk. A rekesz a legkisebb címezhető egység.



A rekesz hossza manapság legtöbbször 8 bit (byte ~ bájt).

n a memória cellák száma

Központi memória (2.9. ábra)

A bitek száma rekeszenként néhány számítógép-történetileg érdekes, kereskedelmi forgalomba került gépen (**2.10. ábra**)

Számítógép	Bit
Burroughs B1700	1
IBM PC	8
DEC PDP-8	12
IBM 1130	16
DEC PDP-15	18
XDS 940	24
Electrologica X8	27
XDS Sigma 9	32
Honeywell 6180	36
CDC 3600	48
CDC Cyber	60

Máté: Architektúrák 4. előadás 25

Bájtsorrend

A legtöbb processzor több egymás utáni bájttal is tud dolgozni (**szó – word, …**).

A legmagasabb helyértékű bájt a szóban a

legalacsonyabb címen:	legmagasabb címen:
nagy (big) endian	**kis (little) endian**
MSBfirst (SPARC)	**LSBfirst (Pentium)**

Most/**L**east **S**ignificant **B**yte **first**

Ha egy 32 bites szó bájtjainak értéke rendre: a, b, c, d, akkor a szó értéke:

$a*256^3+b*256^2+c*256+d$ $a+b*256+c*256^2+d*256^3$

Máté: Architektúrák 3. előadás 26

Bájtsorrend (2.11. ábra)

A memória címek úgy vannak fölírva, hogy a legmagasabb helyértékű bájt van bal oldalon.

Cim	Nagy endian				Kis endian				Cím
0	0	1	2	3	3	2	1	0	0
4	4	5	6	7	7	6	5	4	4
8	8	9	10	11	11	10	9	8	8
12	12	13	14	15	15	14	13	12	12

← 32 bites szó → ← 32 bites szó →

Máté: Architektúrák 3. előadás 27

Bájtsorrend (12. ábra)

A szövegek karaktereit mindkét esetben növekvő bájt sorrendben helyezik el

Cím	nagy endian			
0	*0* **T**	*1* **E**	*2* **X**	*3* **T**
4	*4* **12**	*5* **34**	*6* **56**	*7* **78**

	kis endian				Cím
	3 **T**	*2* **X**	*1* **E**	*0* **T**	0
	7 **12**	*6* **34**	*5* **56**	*4* **78**	4

Cím				
0	*0* **T**	*1* **E**	*2* **X**	*3* **T**
4	*4* **78**	*5* **56**	*6* **34**	*7* **12**

A **TEXT** szöveg és az **12345678** hexadecimális szám elhelyezése a két géptípuson

Problémák a gépek közötti kommunikációban!

Máté: Architektúrák 3. előadás 28

Kódolás: adat + ellenőrző bitek = **kódszó**.

Két kódszó Hamming távolsága: az eltérő bitek száma.
Pl.: 11001 és 11011 (Hamming) távolsága = 1.

Hibaérzékelő kód: bármely két kódszó távolsága > 1: paritás bit.

d hibás bit javítása: a kódszavak távolsága > 2*d*.

Egy hibát javító kód (**2.13. ábra**):
m adat, *r* ellenőrző bit, összesen $n = m + r$.
2^m „jó” szó, + minden „jó” szónak *n* db „egyhibás” szomszédja van, ezért $(1+n)2^m \le 2^n = 2^{m+r}$,
2^m -mel egyszerűsítve: $m + r + 1 \le 2^r$,
vagy másképp: $m + r < 2^r$ szükséges.

Máté: Architektúrák 3. előadás 29

RAM (Random Access Memory)

- **Statikus RAM** (**SRAM**). **D** flip-flop elemekből épül fel. Amíg áram alatt van, tartja a tartalmát. Elérési idő: néhány nsec (cache-nek jók).
- **Dinamikus RAM** (**DRAM**): minden bit egy tranzisztor és egy kondenzátor: néhány msec-onként frissíteni kell, de nagyobb adatsűrűség érhető el. Elérési idő: néhány tíz nsec (főmemóriák).
  - régi: **FPM** (Fast Page Mode) sor-, oszlopcím.
  - újabb: **EDO** (Extended Data Output) lehet új memóriahivatkozás, mielőtt az előző befejeződik.
- **SDRAM** (Synchronous **DRAM**). A központi óra vezérli. Blokkos átvitel.
  Újabban: **DDR** (Double Data Rate). Az órajel föl- és lefutó élénél is van adatátvitel.

Máté: Architektúrák 3. előadás 30

ROM (Read-Only Memory)

ROM: gyárilag kialakított tartalom.

PROM (Programmable **ROM**): a tartalom biztosítékok kiégetésével alakul ki (a **PLA**-khoz hasonlóan, **3.15. ábra**).

EPROM (**Erasable PROM**): a biztosítékok speciális fénnyel kiolvaszthatók és „kijavíthatók”.

EEPROM: elektromos impulzusokkal.

Flash memória: törlés és újraírás csak blokkonként. Kb. 100 000 használat után „elkopnak”. Ilyen van a legtöbb MP3 lejátszóban, digitális fényképezőgépben …

512 MB-os flash memória (2006)

1 GB-os flash memória (2007)

Memória hierarchia (2.18. ábra)

Gyorsító tár (cache – 2.16. ábra)

A processzorok mindig gyorsabbak a memóriáknál.

A **CPU** lapkára integrálható memória gyors, de kicsi.

Feloldási lehetőség: a központi memória egy kis részét (gyorsító tár) a **CPU** lapkára helyezni: Amikor egy utasításnak adatra van szüksége, akkor először itt keresi, ha nincs itt, akkor a központi memóriában.

Lokalitási elv: Ha egy hivatkozás a memória ***A*** címére történik, akkor a következő valószínűleg valahol ***A*** közelében lesz (ciklus, mátrix manipulálás, …).

Ha ***A*** nincs a gyorsító tárban, akkor az ***A***-t tartalmazó (adott méretű) blokk (gyorsító sor - cache line) kerül beolvasásra a memóriából a gyorsító tárba.

Találati arány (***h***): az összes hivatkozás mekkora hányada szolgálható ki a gyorsító tárból.

Hiba arány: *1-h*.

Ha a gyorsító tár elérési ideje: ***c***,
a memória elérési ideje: ***m***, akkor az
átlagos elérési idő = ***c + (1- h) m***.

A gyorsító tár mérete: nagyobb tár – drágább.

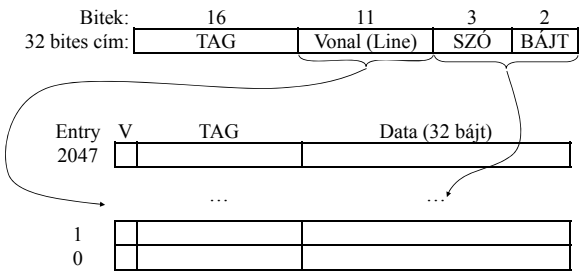
A gyorsító sor mérete: nagyobb sor, a hivatkozott cím nagyobb környezete lesz a gyorsító tárban – nagyobb a sor betöltési ideje is. Ugyanakkora tárban kevesebb gyorsító sor fér el.

Osztott (külön utasítás és adat) **gyorsító tár előnyei:**
- Egyik szállítószalag végzi az utasítás, másik az operandus előolvasást.
- Az utasítás gyorsító tárat sohasem kell visszaírni (az utasítások nem módosulnak).

Egyesített gyorsító tár: nem lehetséges párhuzamosítás.

Hierarchia:
- elsődleges, a **CPU** lapkán,
- másodlagos, a **CPU**-val egy tokban,
- külön tokban.

Direkt leképezésű gyorsító tár működése: (4.38. ábra)



Ha a gyorsító tár **Vonal** által mutatott sorában **V=1** (valid), és a **TAG** megegyezik a címben lévő **TAG**-gel, akkor az adat bent van a gyorsító tárban (ebben a sorban).

Halmazkezelésű (csoportasszociatív) gyorsító tár

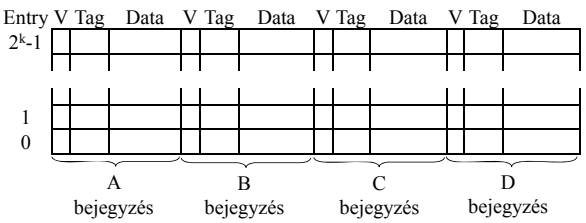
Ha egy program gyakran használ olyan szavakat, amelyek távol vannak egymástól, de ugyanoda képződnek le a gyorsító tárban, akkor sűrűn kell cserélni a gyorsító sort.

Ha minden címhez ***n*** bejegyzés van, akkor ***n*** utas halmazkezelésű gyorsító tárról beszélünk.

Gyakori a 2 és 4, újabban a 8 utas kezelés.

LRU (Least Recently Used) algoritmus: gyorsító sor betöltése előtt a legrégebben használt bejegyzés kerül ki a gyorsító tárból.

Halmaz kezelésű gyorsító tár (4.39. ábra)



Ha a gyorsító tár **Vonal** által mutatott sorában az **A**, **B**, **C** és **D** bejegyzések egyikében **TAG** megegyezik a címben lévő **TAG**-gel, és a hozzá tartozó **V=1** (valid), akkor az adat bent van a gyorsító tárban (ebben a bejegyzésben).

Memóriába írás

Stratégiák:

Írás áteresztés (**write through**): az írás a memóriába történik. Ha a cím a gyorsító tárban van, oda is be kell írni, különben el kellene dobni a gyorsító sort.

Késleltetett írás (**write deferred**, **write back**): ha a cím bent van a gyorsító tárban, akkor csak a gyorsító tárba írunk, a memóriába csak gyorsító sor cserénél.

Ha a cím nincs a gyorsító tárban, akkor előtte betölthetjük: **írás allokálás** (**write allocation**) – többnyire ezt alkalmazzák késleltetett írás esetén.

Feladatok

Mit nevezünk kombinációs áramkörnek?

Milyen kombinációs áramköröket ismer?

Milyen be- és kimenetei vannak a multiplexernek, a demultiplexernek, a dekódolónak?

Mire használható a multiplexer, és hogyan?

Mire használható a **PLA**, és hogyan?

Milyen aritmetikai áramköröket ismer?

Hogy működik a léptető?

Hogy működik a „fél összeadó”?

Mi indokolja a „fél összeadó” elnevezést?

Feladatok

Hogy épül fel a teljes összeadó?

Milyen részei vannak az **ALU**-nak?

Milyen be- és kimenetei vannak az 1 bites **ALU**-nak?

Milyen műveletek végezhetők el az **ALU**-val?

Milyen vezérlő bemenetek esetén lesz **1** az eredmény?

Milyen eredményt szolgáltat az $\mathbf{F_0=0}$, $\mathbf{F_1=1}$, **ENA=0**, **ENB=0**, **INVA=1**, **INC=1** vezérlő bemenet?

Hogy működik az átvitel továbbterjesztő/kiválasztó összeadó?

Feladatok

Hogy érhetünk el az órajelnél finomabb időzítést?

Milyen nem kombinációs áramköröket ismer?

Kombinációs áramkör-e az **ALU**?

Hogyan csökkenthető az összeadásnál az átvitelekből származó idő?

Hány stabil állapota van az **SR** tárolónak?

Mi a különbség az **SR** és az időzített **SR** tároló között?

Mi a különbség az **SR** és **D** tároló között?

Mi a pulzusgenerátor, és mi a működési elve?

Mi a különbség a tároló és a flip-flop között?

Feladatok

Hogy működik az invertáló és a nem invertáló puffer?

Miért használnak a memóriáknál invertáló vagy nem invertáló puffert?

Hogy címezhető meg ***n*** címlábon $\mathbf{2^n}$ -nél nagyobb memória?

Mit jelent, hogy a **CS/CS#** bemenet beállított/negált?

Mi a **RAM**?

Milyen elemekből épül fel a **SRAM**?

Milyen elemekből épül fel a **DRAM**?

Hogy működik a **DRAM**?

Feladatok

Mi a memória cella/rekesz?

Mit jelent a big endian kifejezés?

Milyen problémát okoz az eltérő bájtsorrend?

Mi a Hamming távolság?

Mekkora a hexadecimális E6 és C7 Hamming távolsága?

Hány ellenőrző bit szükséges 256 kódszó 1 hibát javító kódolásához?

Feladatok

Az alábbi memóriák közül melyik lehetséges, melyik ésszerű? Indokolja meg!

10 bites címek	1024 db	8 bites rekesz
10	1024	12
9	1024	10
11	1024	10
10	10	1024
1024	10	10

Egy régi gépnek 8192 szavas memóriája volt. Miért nem 8000?

Feladatok

A memória 100-adik bájtjától a 01234567H 4 bájtos számot és – folytatólagosan – az abcd szöveget helyeztük el. Mi az egyes bájtok tartalma, ha a memória big/little endian szervezésű?

Feladatok

Hogy működik az **SDRAM**?

Mit jelent az **FPM** rövidítés?

Mit jelent az **EDO** rövidítés?

Hogy működik a **DDR**?

Mit jelent a **ROM** rövidítés?

Hogy működik az **EPROM**?

Hogy működik az **EEPROM**?

Milyen memória van a legtöbb fényképezőgépben?

Feladatok

Hol helyezkedhet el a gyorsító tár?

Mi a lokalitási elv?

Mit nevezünk találati aránynak?

Mi a szerepe a találati aránynak?

Mi a hiba arány?

Hogy határozható meg az átlagos keresési idő?

Mi a gyorsító sor?

Mit nevezünk osztott gyorsító tárnak?

Mit nevezünk egyesített gyorsító tárnak?

Mik az osztott gyorsító tár előnyei?

Feladatok

Mit tartalmaz a direkt leképezésű gyorsító tár egy bejegyzése?

Mi a TAG?

Mire szolgál a valid (érvényes) jelzés?

Hogy működik a direkt leképezésű gyorsító tár?

Egy memória cella hány helyen lehet egy direkt leképezésű gyorsító tárban?

Hogy dönthető el, hogy egy memória cella bent van-e egy direkt leképezésű gyorsító tárban?

Milyen esetben nem hatékony egy direkt leképezésű gyorsító tár?

Feladatok

Milyen a halmazkezelésű gyorsító tár felépítése?

Hogy működik a halmazkezelésű gyorsító tár?

Mi a halmazkezelésű gyorsító tár előnye a direkt leképezésűvel szemben?

Mi az **LRU** algoritmus?

Milyen memóriába írási stratégiákat ismer gyorsító tár esetén?

Mit nevezünk írás áteresztésnek (write through)?

Mit nevezünk késleltetett írásnak (write deferred, write back)?

Mit nevezünk írás allokálásnak (write allocation)?

Az előadáshoz kapcsolódó

Fontosabb tételek

Programozható logikai tömbök

Aritmetikai áramkörök. Léptető, fél és teljes összeadó, ALU, az ALU-val végezhető műveletek, átvitel továbbterjesztő és kiválasztó összeadás

Nem kombinációs áramkörök. Óra, tárolók, flip-flop-ok

Központi memória, bájtsorrend

Hamming távolság. Hibaészlelő, hibajavító kódok

Az előadáshoz kapcsolódó

Fontosabb tételek

Gyorsító tár (cache). Találati és hiba arány. Egyesített és osztott gyorsító tár. Direkt leképezésű és halmaz kezelésű gyorsító tár. Memóriába írás