4. Fejezet : Az egész számok (integer) ábrázolása

The Architecture of Computer Hardware and Systems Software:
An Information Technology Approach

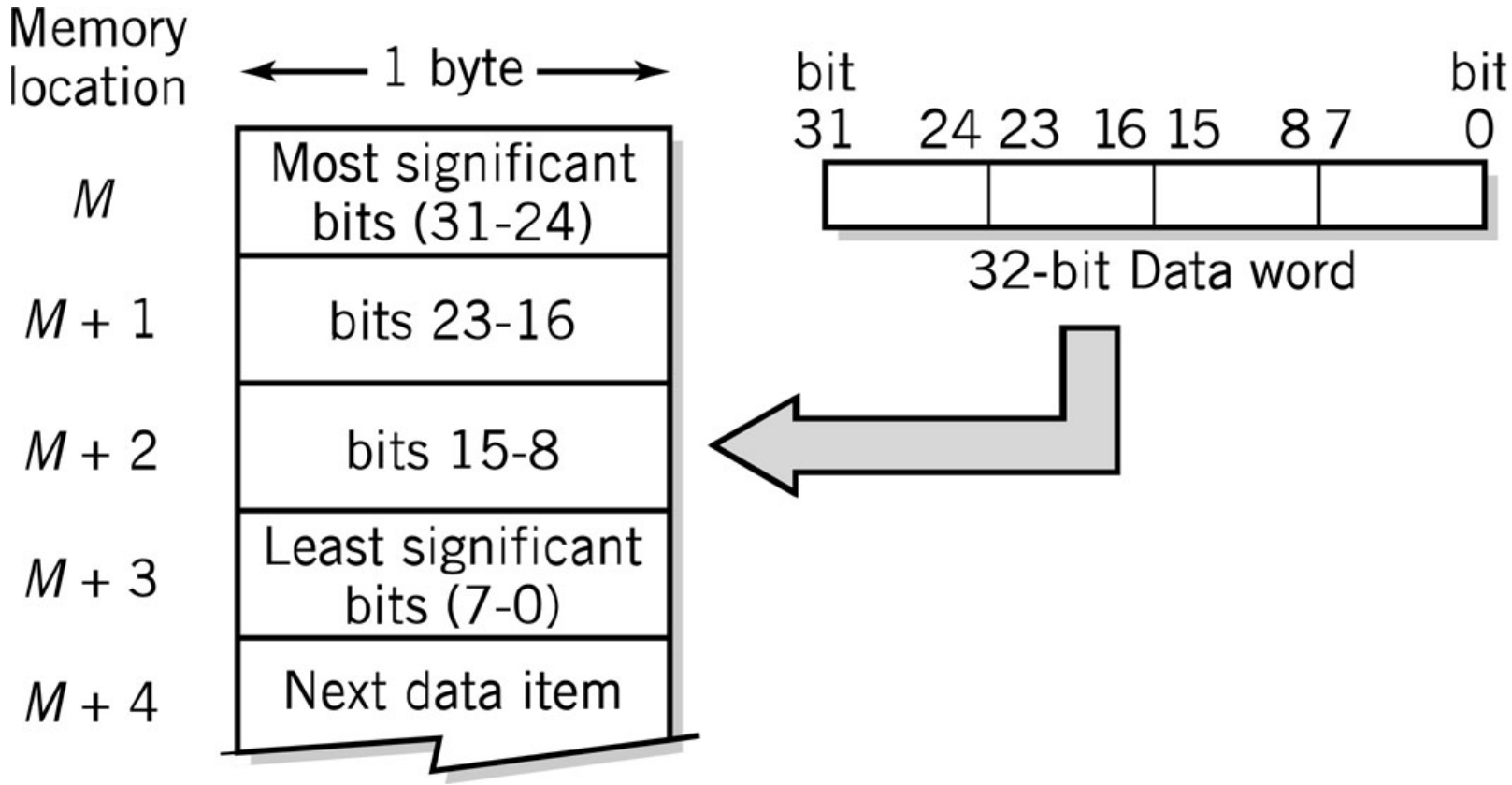
3. kiadás, Irv Englander

Wilson Wong, Bentley College
Linda Senne, Bentley College

Számábrázolás

- Egészszámok ábrázolásához a következő információt kell tárolnunk:
  - Egészszám abszolút értéke (value, magnitude)
  - Előjel (plusz vagy mínusz)

32 bit-es adat szó



Előjel nélküli egész számok: Integerek

- Bináris ábrázolás: Előjel nélküli egész szám vagy *integer*
- BCD ábrázolás: Decimális egész közvetlen *bináris* megfelelője (Binary Coded Decimal)
  - 4 bit: 0-tól 9-ig
  - 8 bit: 0-tól 99-ig
  - 16 bit: 0-tól 9.999-ig
  - 32 bit: 0-tól 99.999.999-ig

Decimális érték	Bináris ábrázolás	BCD ábrázolás
68	= 0100 0100 = $2^6 + 2^2 = 64 + 4 =$ ***68***	= 0110 1000 = $2^2 + 2^1 =$ ***6*** $2^3 =$ ***8***
99 (legnagyobb 8 bit-en ábrázolható szám BCD kódolással)	= 0110 0011 = $2^6 + 2^5 + 2^1 + 2^0 =$ = 64 + 32 + 2 + 1 = ***99***	= 1001 1001 = $2^3 + 2^0$ $2^3 + 2^0$ = ***9*** ***9***
255 (legnagyobb 8 bit-en ábrázolható szám bináris kódolással)	= 1111 1111 = $2^8 - 1 =$ ***255***	= 0010 0101 0101 = 2^1 $2^2 + 2^0$ $2^2 + 2^0$ = ***2*** ***5*** ***5***

Ábrázolható értékek összehasonlítása: Bináris vs. BCD

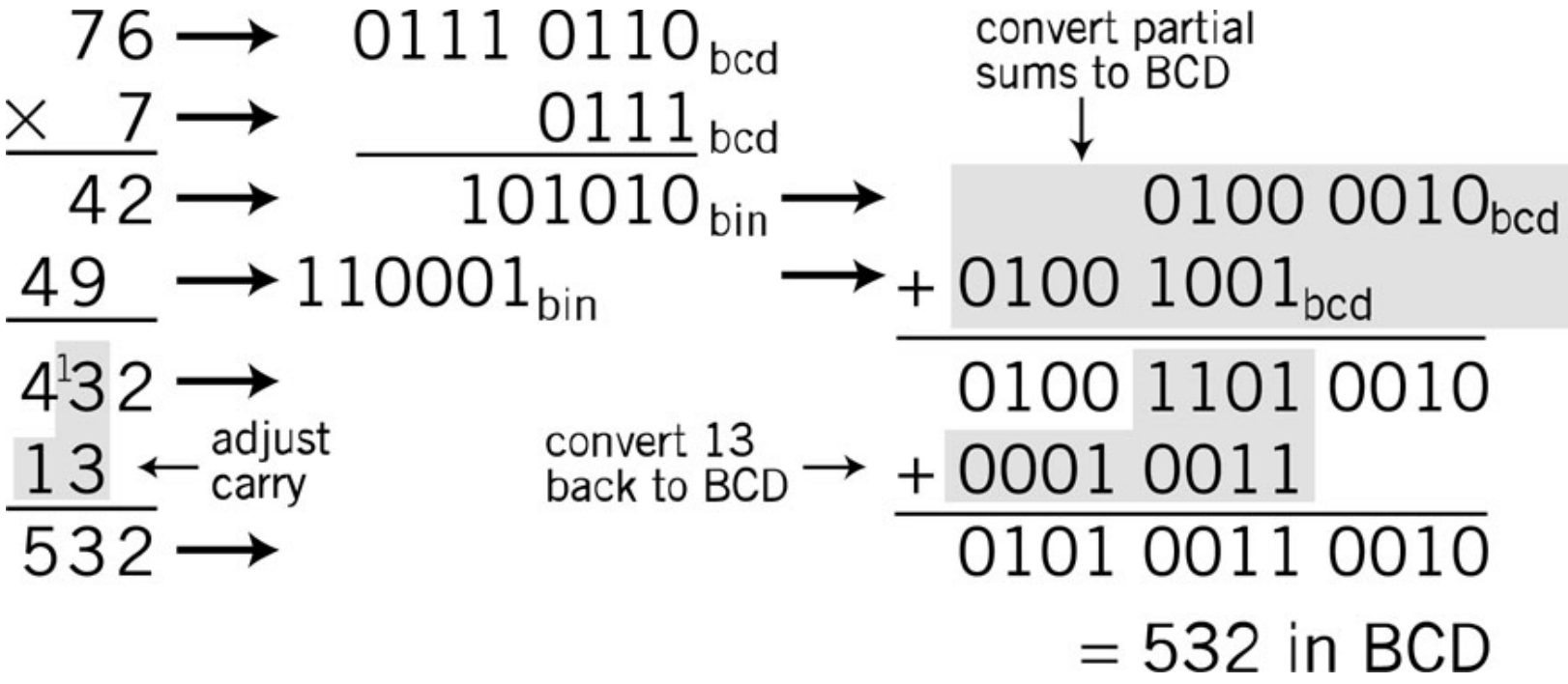
- Ábrázolható értékek BCD kódolással **<** ábrázolható értékek hagyományos bináris ábrázolás
  - Bináris: 4 bit 16 különböző értéket képes tárolni (0-tól 15-ig)
  - BCD: 4 bit csak 10 különböző értéket tud tárolni (0-tól 9-ig)

Bit-ek száma	BCD határ		Bináris határ	
4	0-9	1 számjegy	0-15	1+ számjegy
8	0-99	2 számjegy	0-255	2+ számjegy
12	0-999	3 számjegy	0-4.095	3+ számjegy
16	0-9.999	4 számjegy	0-65.535	4+ számjegy
20	0-99.999	5 számjegy	0-1 million	6+ számjegy
24	0-999.999	6 számjegy	0-16 million	7+ számjegy
32	0-99.999.999	8 számjegy	0-4 billion	9+ számjegy
64	$0-(10^{16}-1)$	16 számjegy	0-16 quintillion	19+ számjegy

Hagyományos bináris vs. BCD

- Bináris ábrázolás kedveltebb
  - Nagyobb értéket képes tárolni adott számú bit-en (helyiértéken)
  - Számolási műveletek elvégzése egyszerűbb
- BCD gyakran használatos vállalati rendszereknél, főként a tizedes kerekítés és a tizedes pontosság miatt

Egyszerű BCD szorzás



Előjeles egész számok ábrázolása

- Nincs nyilvánvaló, közvetlen ábrázolása az előjelnek bináris ábrázolásban
- Lehetőségek:
  - Előjel-és-érték ábrázolás
  - 1-es komplemens
  - 2-es komplemens (gyakoribb)

Előjel-és-érték

- A baloldali legszélső bit használata előjelként
  - 0 = pozitív; 1 = negatív
- Az ábrázolható értékek száma nem változik
  - Számok fele pozitív, fele negatív
  - A legnagyobb ábrázolt érték feleakkora lesz, mint előjel nélkül
- Példa 8 bit-en:
  - Előjel nélküli: 1111 1111 = (+)255
  - Előjeles: 0111 1111 = +127
1111 1111 = -127
  - Megjegyzés: kettő érték a 0-ra:
+0 = 0000 0000 és
-0 = 1000 0000

Bonyolult számítási algoritmusok

- Előjel-és-érték algoritmusok összetettek és bonyolultak ahhoz, hogy hardver hajtsa végre
  - Vizsgálni kell a 0 mindkét értékét
  - BCD kódolás esetén használható
  - Az előjeles szám és a carry/átvitel sorrendje hibát okoz
- Példa: Decimális összeadó algoritmus

Összeadás: 2 pozitív szám	Összeadás: 1 előjeles szám		
4 +2 6	4 - 2 2	2 - 4 -2	12 - 4 8

Komplemens ábrázolás

- A szám előjelét nem kell elkülönítve kezelni
- Megegyezik minden különböző előjelű bemeneti számkombinációra
- Két módszer
  - Alapszám: alap (abszolút) érték
  - Csökkentett alap: alap értéke mínusz 1
    - 1-es komplemens: a 2-es alap csökkentett alapja

Modulus aritmetika (számolás)

- A műveletek eredménye megadja a (számolás alapjaként használt értékkel) modulus történő osztás maradékát
- Példa:
  - 4 mod 4 = 0
  - 5 mod 4 = 1
  - 1000 mod 999 = 1
- A modulusos aritmetika legfontosabb jellemzője
  - A számolást nullától újrakezdjük, ha a számolás során túllépnénk a *moduluson*

Számábrázolás választásának indokai

- Legyen konzisztens (hasonló logikájú) a megszokott aritmetikával
  - - (-érték) = érték
- Ha egy értéknek kétszer kiszámoljuk a komplemensét, akkor az eredeti értéket kapjuk vissza:
  - komplemens = alap – érték
  - Egy komplemens értékre még egyszer alkalmazzuk a komplemens számolást:
    - alap – komplemens = alap – (alap – érték) = érték (!!!)

Túlcsordulás

- Rögzített szó (word) méretnek rögzített a terjedelme
- Túlcsordulás: számok azon kombinációja, amelyek összege a határon kívül esik
- Túlcsordulás átvitellel, ezt a problémát kerüli ki
- Komplemenses aritmetika: *határon kívüli* számoknak ellentétes előjelük van
  - Teszt: Ha összeadásnál mindkét bemeneti szám azonos előjelű és az eredmény előjele más, akkor túlcsordulás lépett fel

1-es bináris komplemens

- *Komplemens számítás*: az érték kivonása az alap eredeti értékéből
  - Bináris (2-es alap) csökkentett alapú komplemense
  - Alap mínusz 1 = 2 – 1 ➡ 1 mint az alap
- *Invertálás: egyesek és nullák felcserélése (1->0, 0->1)*
  - 0-val kezdődő számok pozitívak
  - 1-el kezdődő számok negatívak
  - Két értéke van a nullának (-0, +0)
- Példa 8 bit-es bináris számokkal

Számok	Negatív		Pozitív	
Megjelenítési mód	Komplemens		Szám önmaga	
Decimális szám terjedelme	-127_{10}	-0_{10}	$+0_{10}$	127_{10}
Számolás	Invertálás		Nincs	
Ábrázolási példa	10000000	11111111	00000000	01111111

1-es bináris komplemens

- 1. Példa: $0101\ 1111_{1kB} = +95_D$
- 2. Példa: $1010\ 0000_{1kB}$
  - negatív
  - a negáltja amely egyben az abszolút értéke is(!!!): $0101\ 1111_B = |95_D|$
  - tehát az eredmény: -95_D

Összeadás

- Adjunk össze 2 db 8 bit-es pozitív számot
- Adjunk össze 2db 8 bit-es különböző előjelű számot
  - Vegyük az 58 egyes komplemensét (invertáljuk)
    0011 1010
    1100 0101

0010 1101 = +45

0011 1010 = +58

0110 0111 = + 103

0010 1101 = +45

1100 0101 = –58

1111 0010 = –13

Invertáljuk, hogy megkapjuk az értéket

0000 1101

8 + 4 + 1 = |13|

Összeadás átvitellel (carry-vel)

0010 1101 = +45

1110 0001 = –30

1|0000 1110 = +

+1 = 15

+15

1110 0001 = –30

0001 1110 = |30|

Összeadás átvitellel (carry-vel)

- 8 bit-es szám
- Negatív érték hozzáadása:
  - Invertálás
    0000 0010 (2_{10})
    1111 1101
  - Összeadás
  - 9 bit

Túlcsordulás átvitellel

0110 1010	=	+106
1111 1101	=	–2
10110 0111		
+1		
0110 1000	=	+104

Kivonás átvitellel (carry-vel)

- 8 bit-es számábrázolás
- szám kivonása = szám -1 szeresének hozzáadása (negatív alak!)
  - invertálás
    0101 1010 (90_{10})
    1010 0101
  - Összeadás
  - 9 bit

    Túlcsordulás átvitellel!

```
 0110 1010  =   +106
-0101 1010  = - (+90)
-----------   -------

 0110 1010  =   +106
+1010 0101  =    -90
-----------   -------
 10000 1111
      +1
-----------   -------
 0001 0000  =   +16
```

Kivonás

- 8 bit-es számábrázolás
- szám kivonása = szám -1 szeresének hozzáadása (negatív alak!)
  - invertálás
    1111 0110 (-9_{10})
    0000 1001
  - összeadás

0110 1010	=	+106
- 1111 0110	=	- (-9)

0110 1010	=	+106
+ 0000 1001	=	+9
0111 0011	=	+115

Kivonás

- 8 bit-es szám
  - invertálás
    1010 0101(90_{10})
    0101 1010
  - Összeadás

0110 1010	=	+106
- 1010 0101	=	-90
0110 1010	=	+106
+ 0101 1010	=	+90
1100 0100	=	-
0011 1011	=	|59|

Túlcsordulás! Rossz eredmény!!!

Túlcsordulás

- 8 bit-es szám
  - 256 különböző szám
  - Pozitív számok: 0 to 127
- Összeadás
  - *Túlcsordulás* tesztelése
  - 2 pozitív bemenetre negatív eredményt adott
  - → *túlcsordulás!*
  - Rossz eredmény!!!

0100 0000 = +64

0100 0001 = +65

1000 0001 -126

0111 1110

Invertáljuk, hogy megkapjuk az értéket

126_{10}

- Programozók óvakodjatok: néhány magas-szintű nyelv pl.: a BASIC néhány verziója nem ellenőrzi a túlcsordulást megfelelően!

2-es komplemens

- Modulus = 2-es alapú „1”-es után nullák
  - 8 bit-en a modulus = 1000 0000
- Két módszer van megtalálni a komplemenst
  - Kivonjuk a szám értékét a modulusból vagy invertálunk

Számok	Negatív		Pozitív	
Megjelenítési mód	Komplemens		Szám önmaga	
Decimális szám értéke	-128_{10}	-1_{10}	$+0_{10}$	127_{10}
Számolás	Invertálás		Nincs	
Megjelelenítési példa	10000000	11111111	00000000	01111111

1-es vs. 2-es komplemens

- A választás a számítógép tervezőitől függ
- 1-es komplemens
  - Egyszerű előjelet váltani
  - Összeadásnak szüksége van egy extra túlcsordulás átvitelre
  - Algoritmusnak tesztelnie és konvertálnia kell a ”-0” -át
- 2-es komplemens egyszerűbb
  - Negálás után egy |1| hozzáadás szükséges

2-es bináris komplemens

- 1. Példa: $0101\ 1111_{2kB}$ = $+95_D$
- 2. Példa: $1010\ 0000_{2kB}$
  - negatív
  - a negáltja amely **nem** az abszolút értéke(!!!):
    $0101\ 1111_B$
  - |1| hozzáadása a negálthoz
  - így: $0101\ 1111_B$
    $+\quad 1_B$
    $0110\ 0000_B = |96_D|$
  - tehát az eredmény: -96_D

Egy összetett példa

- Határozza meg a **decimális** és **hexadecimális** értékét a következő 8 bit-es bináris számoknak, ha **közönséges binárisként**, vagy **1-es komplemensként** illetve **2-es komplemensként** értelmezük őket. A hexadecimális számoknál az előjeleket kezelje a decimális számoknál megszokott módon!
- $0111\ 0100_B$: közönséges: 116_D és 74_H
  1-es: $+116_D$ és $+74_H$
  2-es: $+116_D$ és $+74_H$
- $1101\ 0101_B$: közönséges: 213_D és $D5_H$
  $|0010\ 1010_B|$ 1-es: -42_D és $-2A_H$
  $|0010\ 1011_B|$ 2-es: -43_D és $-2B_H$

Egy még összetettebb példa

- Végezze el a következő **kivonást** 8 bit-es bináris számokkal úgy, hogy a számpár első tagját **1-es komplemensként** a második tagját **2-es komplemensként** értelmezük. Az eredményt adja meg bináris 1-es komplemensként, majd váltsa át azt decimális számrendszerbe:
- $1010\ 0101_{B1K}$ **-** $1001\ 1010_{B2K}$ **=** $1010\ 0101_{B1K}$ **-** *(*$0110\ 0101_B$ **+** $|1|_B$ = $0110\ 0110_B$ **=>** *)* $1001\ 1001_{B1K}$
- $1010\ 0101_{B1K}$
  + $0110\ 0110_B$
  $1\ 0000\ 1011_{B1K}$
  $\rightarrow 1_B$
  $0000\ 1100_{B1K}$ **=** $+12_D$

Integer mérete

- Pozitív számok 0-val kezdődnek
- Kicsi, negatív számok (közel a 0-hoz) több 0-val kezdődnek
  - $1111\ 1110_{B2K}$ = -2 8 bit-es 2-es komplemens
  - $1000\ 0000_{B2K}$ = -128, nagyobb negatív szám 2-es komplemense
  - Cseréljünk fel minden 1-est és 0-ást és határozzuk meg az értéket

Túlcsordulás és Átvitel közötti különbség

- *Átvitel (carry) bit*:
  - Jelzi, ha egy ***adott*** helyiértéken elvégzett művelet eredménye meghaladja az ott ábrázolható értéket.
- *Túlcsordulás (Overflow)*:
  - ha egy összeadás vagy kivonás művelet eredménye kívül esik az aktuális számábrázolási tartományon.

Túlcsordulás/Átvitel példák

- Példa 1:
  - Helyes eredmény
  - Se túlcsordulás, se átvitel

0100	=	(+ 4)
0010	=	+ (+ 2)
0110	=	(+ 6)

- Példa 2:
  - Helytelen eredményt kapunk
  - Van túlcsordulás, és átvitel is

0100	=	(+ 4)
0110	=	+ (+ 6)
1010	=	(– 6)

Négy bit-es 2-es komplemens számábrázolás!

Túlcsordulás/Átvitel példák

- Példa 3:
  - Az átvitelt elhagyva az eredmény helyes
  - Van átvitel, de nincs túlcsordulás

1100	=	(– 4)
1110	=	+ (– 2)
1010	=	(– 6)

- Példa 4:
  - Helytelen eredmény
  - Van túlcsordulás, átvitel mellőzve

1100	=	(– 4)
1010	=	+ (– 6)
0110	=	(+ 6)

Négy bit-es 2-es komplemens számábrázolás!