Introducere în organizarea calculatoarelor și limbaje de asamblare (IOCLA)

Reprezentarea datelor în sistemele de calcul – Complement față de 2

Modificat: 22-Oct-23

Cuprins

- Reprezentarea numerelor cu semn
- Complementul față de 2
- Reprezentarea tipurilor de date de nivel înalt

Suport

- Introduction to Assembly Language Programming
  - Anexa A, secțiunea A4.4

REPREZENTAREA NUMERELOR CU SEMN

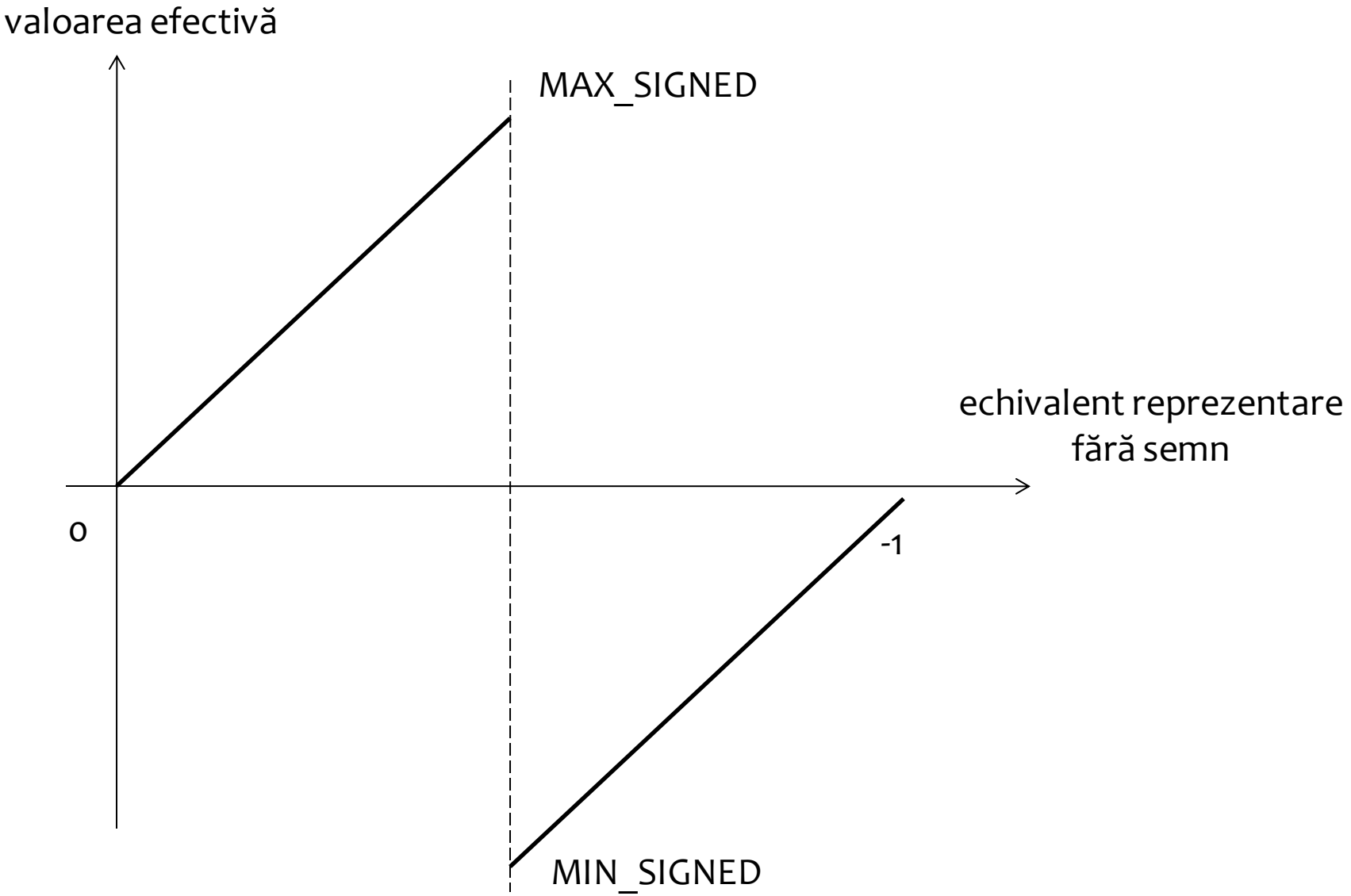
Cât înseamnă oxFF?

- Depinde
  - F*16+F = 255, pentru număr fără semn
  - -1 pentru reprezentare cu semn
- Într-o reprezentare cu semn toți biții sunt parte a numărului (fiecare bit are o pondere valorică)

Ideea de reprezentare a numerelor cu semn

- Se alocă primul bit pentru semn
  - 0: număr pozitiv
  - 1: număr negativ
- Restul biților dau valoarea

Intuiția reprezentării numerelor cu semn



Exemple de reprezentare cu semn

- 8 biți
  - **00000000** (0x00): 0
  - **01111111** (0x7F): 127 (SCHAR_MAX)
  - **10000000** (0x80): -128 (SCHAR_MIN)
  - **11111111** (0xFF): -1
- 16 biți
  - **00000000 00000000** (0x0000): 0
  - **01111111 11111111** (0x7FFF): 32767 (SHRT_MAX)
  - **10000000 00000000** (0x8000): -32768 (SHRT_MIN)
  - **11111111 11111111** (0xFFFF): -1

Complement față de 2

- Îl vom numi C2
- Numerele pozitive rămân neschimbate
- Numerele negative
  - Au ca prim bit 1 (bitul de semn)
  - Algoritm
    - 1. Se neagă tot șirul de biți reprezentând numărul în modul
    - 2. Se adună 1 la valoarea respectivă, ca la adunările între numere întregi fără semn
    - 3. Se trunchiază șirul de biți la dimensiunea inițială

Caracteristici C2

- Pentru o reprezentare pe N biți, acoperă plaja

$$[-2^{N-1}, 2^{N-1}-1]$$

  - Pentru 8 biți: [-128, 127]
  - Pentru 16 biți: [-32768, 32767]
- Dacă se incrementează fără semn MAX (reprezentarea maximă) se ajunge în MIN (reprezentarea minimă)
  - Demo
- Reprezentarea lui -1 e echivalentul celui mai mare număr dacă ar fi reprezentarea fără semn (numai biți de 1)

Exemple de reprezentare de numere în C2

- Pe 8 biți
- -20 (not 20 + 1)
  - 20: 00010100
  - not 20: 11101011
  - -20 în C2 (not 20 + 1): 11101011 + 1: 11101100
- Ce reprezinta 11010101 (0xD5), dacă știm că numărul e cu semn?
  - Primul bit: 1, negativ
  - R-1: 11010101 – 1 = 11010100
  - -N = not (R-1) = 00101011 = 43
  - N = -43

Incrementări interesante în C2

- SCHAR_MAX + 1 = SCHAR_MIN
  - 01111111 + 1 = 10000000
  - signed_value(10000000) = - (NOT 1000000 + 1) = - (01111111 + 1) = - 10000000 = -0x80 = -128 (SCHAR_MIN)
- -1 + 1 = 0 (Duh!)
  - 11111111 + 1 = 00000000

Operații aritmetice în C2

- SCHAR_MIN + SCHAR_MAX
  - 10000000 + 01111111 = 11111111 (-1)
- -1 + 10
  - 11111111 + 00001010 = 00001001
- 100 + 100 (ambele numere cu semn)
  - 01100100 + 01100100 = 11001000 -> -56
  - se schimbă bitul de semn de la doi operanzi de semn opus
    - » se activează "overflow flag"
    - » informație că rezultatul este "incorect"

Ce înseamnă?

- 0xFE
- (unsigned char) -1
- (int) 0xff
- char c = 0xff; int d = c;
- Curiozate: INT_MIN % -1

C2 este MAGIC!

- Un singur circuit pentru adunare și scădere
  - Adunare $a + $b
  - Scădere $a - $b = $a +~$b + 1

- Rezultatul adunării e corect pentru operanzi
  - Signed / Unsigned
  - programatorul știe care este semnificația datelor
  - compilatorul știe care este semnificația datelor

Folosit de toate* procesoarele de azi

Seminar în gdb

- gdb = calculator/convertor hex/binar/dec/C2
- Comanda set
  - set $a = (char)100
  - set $b = (3*$a) & 0xff
  - whatis $b
- Comanda p
  - p/x - afișează în hexa
  - p/d - zecimal
  - p/u - unsigned
  - p/t - binar

Seminar în gdb

a> 127

* set $a = (char)150
* p/t $a

  10010110
* p/x $a

  0x96
* p/d $a

  -106
* p/u $a

  150

a <= 127

* set $b = (char)50
* p/t $b

  110010
* p/x $b

  0x32
* p/d $b

  50
* p/u $b

  50
* whatis $a + $b

  int

Seminar în gdb

* set $c = (char) ($a + $b)
* p/t $c

  10010110
* p/x $c

  0xc8
* p/d $c

  -56
* p/u $c

  200

$c conține **ambele** rezultate:

-106 + 50 = -56

150 + 50 = 200

Magic!

Interpretările lui $c

* $c fără semn? 0xc8

* $c cu semn?

p/t $c & 0B1000000 =

= 10000000 => bit de semn

* Așadar negativ:
* p/t ~$c + 1

  » 56

Seminar în gdb

* set $d = (char) ($a - $b)
* p/t $t

  1100100
* p/x $c

  0x64
* p/d $c

  100
* p/u $c

  100

signed: -106 - 50 = -156

nu se reprezintă în C2 (byte)

unsigned: 150 - 50 = 100

* Implementarea scăderii C2 în hardware:
* $a - $b = $a + ~$b + 1
* $b - $a = $b + ~$a + 1

Seminar în gdb

* Să se efectueze cu și fără semn:
* 10 + 10
* 100 + 100
* 10 – 11
* 100 – 200
* -100 + 120
* 100 - 120
* 150 + 160
* 120 + 160
* 150 - 160

- Demo în SASM/gdb:
  - **`demo/curs-02`** se încearcă diverse valori AH +- AL
  - atenție la .gdbinit
  - add, sub pe byte
  - activare CF, OF
- Utilitar **`demo/curs-04`**
- **`print_flags`** – adunare/scădere pe octeți, cu flag-uri

Extensia bitului de semn

- numere negative pe mai mulți octeți -

```
gdb> set $c = (char) -100
gdb> set $s = (short) -100
gdb> set $i = (int) -100
gdb>
gdb> p/x $c
$1 = 0x9c
gdb> p/x $s
$2 = 0xff9c
gdb> p/x $i
$3 = 0xffffff9c
```

movsx AX, BL

AL:=BL și AH se umple cu bitul de semn din BL

REPREZENTAREA TIPURILOR DE DATE DE NIVEL ÎNALT

Structuri

- Date contigue
- Date de diferite tipuri (dimensiuni, cu semn/fără semn)
- O dată este la un anumit deplasament (offset) față de adresa de început a structurii

Vectori (arrays)

- O înșiruire (de obicei compactă) de date
- Vector de întregi (32 de biți)
  - La fiecare 4 octeți avem o nouă valoare întreagă
  - A, A+1, A+2, A+3 (little sau big endian)
- sizeof(a): dimensiunea ocupată de toate elementele vectorului

Șiruri

- Vector de caractere
- Ultimul caracter este NUL-terminatorul (‘\0’, valoarea 0x0)
- char s[] = “ana”; /* 4 bytes; NUL-terminated */
- Char *s = "ana"; /* rodata */
- char s[] = {‘a’, ‘n’, ‘a’}; /* 3 bytes */
- char s[] = “\xb0\xcd\xd5”; /* string (non ASCII) */

Cuvinte cheie

- Numere cu semn
- Numere fără semn
- Complement față de 2
- Structuri, vectori, șiruri