# Introducere în organizarea calculatoarelor și limbaje de asamblare (IOCLA)

## Reprezentarea datelor în sistemele de calcul

Modificat: 22-Oct-23

# Cuprins

- Date numerice (întregi)
- Reprezentarea numerelor în baza 2
- Operații cu numere în baza 2
- Reprezentarea numerelor în baza 16
- Reprezentarea numerelor cu semn
- Reprezentarea tipurilor de date de nivel înalt

# Suport

- Introduction to Assembly Language Programming
  - Anexa A, fără 4.1-A4.3

# DATE NUMERICE (ÎNTREGI)

# Ce reprezentăm cu “date”?

- instrucțiuni - prin coduri de instrucțiuni
- date
  - logice: Adevarat/Fals, Închis/Deschis, Pornit/Oprit
  - numerice: întregi, fracționare, pozitive/negative
  - alfanumerice: caractere, text
  - multimedia: sunet, imagine (audio/video)
- structuri de date

# Date numerice

- Tipurile de bază
  - Caractere, întregi, **unele** numere fracționare (floating point)
  - Toate celelalte tipuri existente sunt derivate din tipurile de bază
- Tipurile întregi
  - Numere cu semn sau fără semn
    - » Cele cu semn sunt considerate pozitive
  - Au reprezentare pe un număr de biți (8, 16, 32, 64 de biți)
- Tipurile fracționare
  - Virgulă fixă sau virgulă mobilă (floating point)
  - Uzual reprezentare în virgulă mobilă
  - Vor fi detaliate în capitolul 9

# Lungime tipuri de date întregi

- signed/unsigned char
  - lungime 1 octet (8 biti) (arh. pe 32 și 64 de biți)
- signed/unsigned short int
  - 2 octeți (16 biți) (arh. pe 32 și 64 de biți)
- signed/unsigned int
  - lungime 4 octeți (32 biți) (arh. pe 32 și 64 de biți)
- signed/unsigned long int
  - lungime 4 octeți (32 biți) (arh. pe 32 de biți)
  - Lungime 8 octeți 64 de biți) (arh. pe 64 de biți)
- signed/unsigned long long int
  - lungime 8 octeți (64 biți) (arh. pe 32 și 64 de biți)

# Numere întregi fără semn

- Reprezentare prin simboluri
  - Modelul de scriere romană a numerelor

    **LIX = 59**

    **XXXXXIX = 59**
  - Dezavantaj: nu neapărat o reprezentare unică, spațiu mare ocupat
- Reprezentare pozițională
  - Reprezentare unică
  - Alegerea unei baze de numeratie
    - » În funcție de criterii subiective sau obiective
    - » Ex: în prezent baza ***10***, fenicienii aveau baza ***60***, calculatoarele utilizează baza ***2***

# Baze de numerație

- Reprezentarea pozitionala utilizeaza baze de numeratie si cifre care au valoare diferita in functie de pozitie
  - Ex: fie baza de numeratie ***B***
    - ***B > 1*** si **0** $\leq x_{n-1}, x_{n-2}, x_{n-3}, \dots, x_1, x_0 <$ ***B***

avem reprezentarea unica in baza B : $x_{n-1}x_{n-2}x_{n-3} \dots x_1x_{0\,(B)}$

unde: $x_{n-1}x_{n-2}x_{n-3} \dots x_1x_{0\,(B)} = x_{n-1} * B^{n-1} + x_{n-2} * B^{n-2} + \dots + x_1 * B^1 + x_0 * B^0$

# Reprezentarea numerelor în baza 10

- 2891 = 2*10^3 + 8*10^2 + 9*10^1+1
- Preferată pentru specificul uman (numărat pe degete)

# REPREZENTAREA NUMERELOR ÎN BAZA 2

# Reprezentarea numerelor în baza 2

- Reprezentare binară
- Ușor de implementat pe calculator
- Două niveluri: închis/deschis, sus/jos
  - circuite digitale (vezi curs de electronică)
  - bit = 0 -> bitul ***nu* este** sub tensiune
  - bit = 1 -> **este** sub tensiune
- Bit = Binary Digit

# Formate binare de reprezentare

- Bit
  - binary digit
  - unitatea elementară de informație
  - starea unui bistabil, sau a unei celule elementare de memorie
- Octet (byte)
  - grup de 8 biți
  - unitatea elementara de adresare la cele mai multe calculatoare actuale (inclusiv Intel x86)
  - poate reprezenta o valoare numerică, un caracter (ASCII)

# Conversii din baza 2 în baza 10

- 10111010: 1*2^7 + 0*2^6 + 1*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 186
- Rapide:
  - 10000000: ?
  - 00100000: ?
  - 00011111: ?
  - 00001111: ?
  - 00001110: ?

# Conversii din baza 10 in baza 2

- Clasic: se împarte la doi, se reține bitul de rest; apoi se ia câtul, se împarte la 2, etc.
- Mai ingineresc: se adună succesiv puteri ale lui 2
  - 200 = 128 + 64 + 8 = 11001000
  - 163 = 128 + 32 + 2 + 1 = 10100011
- Rapid:
  - 32: ?
  - 65: ?
  - 15: ?
  - 14: ?

# OPERAȚII CU NUMERE ÎN BAZA 2

# Operații aritmetice în baza 2

- Adunare, scădere, împărțire, înmulțire
- Aceleași principii ca la baza 10
- Făcute de calculator
- Improbabil să fie nevoie să fie făcute de noi
  - Dacă este cazul, facem conversia înainte și înapoi în/din baza 10

# Operatii pe biți

- Bitwise operations
- Bitwise OR vs. Logical OR
- AND
- NOT
- XOR (Exclusive OR)

# Deplasări (shift)

- Shift right (SHR)
- Shift left (SHL)
- shift logic vs. shift aritmetic
  - clarificăm la numere cu semn
- Ce înseamnă SHR N, 2 (shift la dreapta cu 2)?
- Ce înseamnă SHL N, 2 (shift la dreapta cu 2)?

# Măști de biți

- Valori predefinite
- Folosite pentru verificare
- Fie masca 11110000
  - Ce înseamnă N & masca? (bitwise AND)
  - Ce înseamnă N | masca? (bitwise OR)
- Ce înseamnă N | 0, N & 0, N & MAX, N | MAX?
- Ce înseamnă N ^ N? (eXclusive OR)
- Ce înseamnă N ^ 0, N ^ MAX?

# Exemple practice de operatii pe biți

- Rețelistică (mască de subrețea, adresa de subrețea)
- Hărți de biți (bitmaps): pentru spațiu ocupat puțin: gestiunea spațiului liber pe disc
- Opțiuni active sau nu: permisiuni Unix pe fișiere
- Configurarea perifericelor pentru micro-controllere

# REPREZENTAREA NUMERELOR ÎN BAZA 16

# Reprezentarea numerelor în baza 16

- Numită reprezentare în hexazecimal
  - Informal: "în hexa"
- De ce este utilă?
  - Formă compactă a reprezentării binare
  - 4 biți formează o cifră hexa (nibble)
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Reprezentare în ingineria calculatoarelor
  - 0x89AB
  - 89ABh

# Scenarii de utilizare reprezentare hexa

- Dump hexa pentru date binare
  - Utilitare precum xxd, hexdump, od
- Vizualizare adrese de memorie
  - Dezasamblare
  - Debugging
  - Hardware
- Generare date binare
  - Demo echo, python, perl (pe Linux)

# Conversii baza 2 <-> baza 16

- 2 -> 16
  - Se iau câte 4 biți și se transformă în nibble
  - 1101: B, 1110: E, 11000101: C5
- 16 -> 2
  - Se transformă fiecare nibble în șir de 4 biți
  - A: 1010, C: 1100, 87: 10000111
- Exerciții rapide
  - **11000000 10101000**: ?
  - **10010011 00111110**: ?
  - ABCD: ?
  - 9876: ?

# Conversii baza 10 <-> baza 16

- De obicei are sens pentru numere până în 256
- 10 -> 16
  - Se împarte la 16 și se folosesc câtul și restul
  - 190 -> 16*11 + 14 -> 16*B + E -> 0xBE
  - 95 -> 16*5 + 15 -> 16*5 + F -> 0x5F
- 16-> 10
  - Înmulțire cu 16 a primei cifre și adunarea ultimeia
  - AA -> 16*A + A -> 160 + 10 -> 170
  - CC -> 16*C + C -> 192 + 12 -> 204

- Demo în gdb
  - set $i = 26
  - p/d $i*2
  - p/x $i
  - p/t $i
  - p/d 0x22
  - help x
  - set $j = -11
  - p/d ($j & 0xff)
  - În sasm: add pe byte 11 + 245

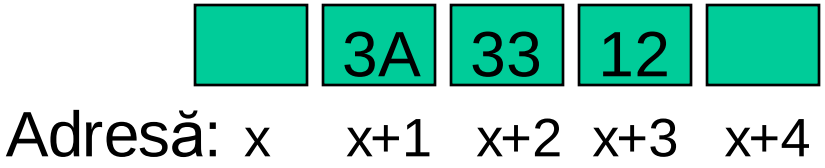
# Împachetarea datelor (endianess)

- LSB: Least Significant Byte
- MSB: Most Significant Byte
- Little endian: LSB la adresa de memorie cea mai mică
- Big endian : MSB la adresa de memorie cea mai mică
- Important la transferul de date între sisteme diferite; de ales un format comun

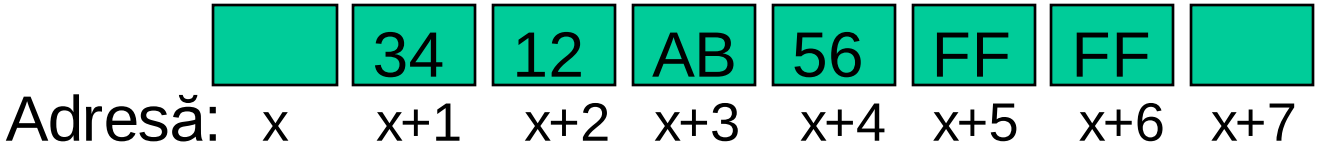
- x86: little endian
- Internet: big endian

# Reprezentare Little Endian

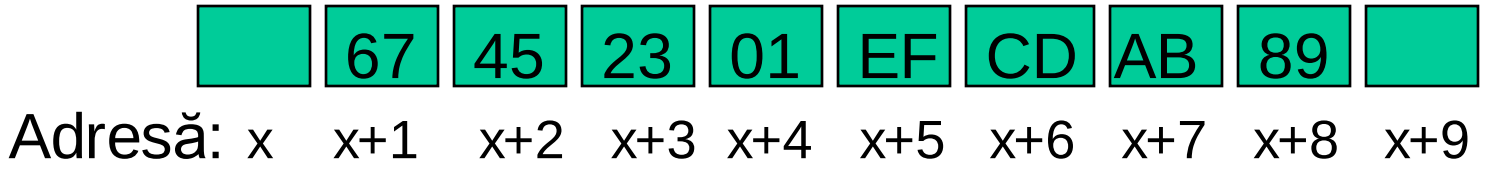
* Octeti: 3AH, 33H, 12H




* Cuvinte: 1234H, 56ABH, FFFFH




* Dublu-cuvinte: 01234567H, 89ABCDEFH



- `git clone http://github.com/iocla/demo.git`
- `cd ./demo/curs-04; make`
- Laboratorul 1, exercițiul 5
  - Printați pentru fiecare octet și adresa la care se află
- Demo în SASM
  - `a db 1, 2, 3, 4, 0xa, 0xb, 0xc, 0xd, 'a', 'b', 'c', 'd'`
  - `b dw 0x0102, 0x0304, 0x0a0b, 0x0c0d, 4127, -27714`
  - `c dd 0x01020304, 0xabcd6789, 100, -100`
  - Examinare cu `x/20xb &a`, `x/20xh &b`, `x/20xw &c`, etc

# Cuvinte cheie

- Date întregi
- Baze de numerație
- Baza 10
- Baza 2
- Baza 16
- Conversii

- Endianess
- Bit, octet
- Operații logice (pe biți)
- Mască de biți
- Nibble