Seminare ASC - Rezolvări

Matei-Iulian Cocu
matei-iulian.cocu@s.unibuc.ro

Ștefan Chiper
stefan.chiper@s.unibuc.ro

Cuprins

Seminar 0x00

1. Completați următorul tabel cu reprezentările lipsă ale unor numere naturale:

Baza: $B = 2$	$B = 4$	$B = 8$	$B = 10$	$B = 16$
1011 1110 1110 1111	*2332 3233*	*137 357*	*48879*	0xBEEF
0000 0000 0010 1010	*222*	*52*	42	*0x002A*
1001 1100 1111 0011	*2130 3303*	*116 363*	*40179*	*0x9CF3*
0000 0010 1011 1101	22331	*1275*	*701*	*0x02BD*
0000 0001 1111 1111	*13333*	777	*511*	*0x01FF*
1101 1110 1010 1111	*3132 2233*	*157 257*	*57007*	0xDEAF
1100 1010 0101 0011	*3022 1103*	*145 123*	*51795*	*0xCA53*
0000 0000 1110 0100	3210	*344*	*228*	*0x00E4*
0000 1110 1001 1101	*322131*	7235	*3741*	*0x0E9D*
0000 0111 1110 0100	*133210*	*3744*	2020	*0x07E4*
0001 0001 0001 0001	*101 0101*	*10 421*	*4,369*	0x1111
1111 1111 1111 1111	*3333 3333*	*177 777*	*65535*	0xFFFF
1111 1111 0000 0000	*3333 0000*	*177 400*	*65280*	*0xFF00*
0000 0000 1111 1111	*3333*	*377*	*255*	*0x00FF*

2. Completați următorul tabel cu reprezentările lipsă ale unor numere întregi (16 biți):

Baza: $B = 2$	$B = 4$	$B = 8$	$B = 10$	$B = 16$
1111 1110 1110 1101	*3332 3231*	*177 355*	*-275*	0xFEED
1111 1111 1101 0110	*3333 0332*	*377 326*	-42	*0xFFD6*
0101 1100 1111 0011	*1130 3303*	*057363*	*23795*	*0x5CF3*
0000 0010 1011 1101	22331	*2535*	*701*	*0x2BD*
111111111	*3333*	777	*511*	*0x1FF*
1010 0001 1111 0001	*2020 3121*	*24361*	*41457*	0xA1F1
1100 1010 0101 0011	*3102 1123*	*145123*	*51891*	*0xCA53*
1110 0100	3210	*344*	*228*	*0xE4*
1110 0000 0111	*320013*	7007	*3591*	*0xE07*
1111 1010 1011 0111	*3322 2313*	*3667*	-1337	*0xFAB7*
1111 0001 0001 0001	*3302 2211*	*170421*	*61713*	0xF111
1111 1111 1111 1111	*3333 3333*	*177777*	*65535*	0xFFFF
1111 1111 0000 0000	*3333 0000*	*177400*	*65280*	*0xFF00*
1111 0000 1111 0000	*3300 3300*	*170360*	*61680*	*0xF0F0*

3. Realizați următoarele operații aritmetice cu numere pe biți reprezentate în complement față de doi (verificați-vă calculele folosind sistemul zecimal):

```
  0101 1100 1111 0011
  1111 1111 0000 0000   +
  ---------------------
  overflow - 0101 1011 1111 0011

  0111 1111 1111 1111
  0000 0000 0000 0001   +
  ---------------------
  1000 0000 0000 0000

  1111 1111 1111 1111
  0000 0000 0000 0000   +
  ---------------------
  1111 1111 1111 1111

  1111 1111 1111 1111
  0000 0000 0000 0001   +
  ---------------------
  overflow - 0000 0000 0000 0000

  0101 1100 1111 0011
  0111 0000 1111 0000   +
  ---------------------
  1100 1101 1110 0011

  1000 0000 0000 0000
  0000 0000 0000 0001   +
  ---------------------
  1000 0000 0000 0001
```

În următoarele cazuri, numărul de biți scriși reprezintă dimensiunea totală alocată pentru reprezentarea numerelor (14 și 8 biți în stânga, 12 și 8 biți în dreapta):

11 1000 0010 0101	
1011 1100	+
11 1000 1110 0001	

1100 1001 0010	
0100 1010	+
1100 1101 1100	

4. Realizați următoarele operații logice pe biți:

0101 1100 1111 0011	
0101 1100 1111 0011	*AND*
0101 1100 1111 0011	

0111 1111 1111 1111	
0000 0000 0000 0001	*AND*
0000 0000 0000 0001	

1101 1001 0110 0001	
1111 1111 0000 0000	*AND*
1101 1001 0000 0000	

0000 0000 1111 1111	
0000 0001 0000 0000	*AND*
0000 0000 0000 0000	

1101 1100 1111 0011	
1101 1100 1111 0011	*XOR*
0000 0000 0000 0000	

0101 1100 1111 0011	
0111 0000 1111 0000	*OR*
0111 1100 1111 0011	

0101 1100 1111 0011	
0000 0000 1111 1111	*OR*
0101 1100 1111 1111	

1100 0110 1001 1110	
1001 1111 0110 1100	*XOR*
1100 0110 1001 1110	*XOR*
1001 1111 0110 1100	

5. Folosing cunoștințele de curs, răspundeți la următoarele întrebări:

(a) care este cel mai mare număr zecimal (natural) care se poate reprezenta pe N biți? ***R:*** $2^N - 1$

(b) care este cel mai mare număr zecimal (întreg) care se poate reprezenta (în complement față de doi) pe N biți? dar cel mai mic? ***R:*** $2^{N-1} - 1$*(mare)*; -2^{N-1}*(mic)*

(c) fie x un număr natural, de câți biți este nevoie pentru a-l reprezenta în binar? ***R:*** $\lfloor \log_2(x) \rfloor + 1$

(d) dacă un număr x este reprezentat cu k cifre în sistemul hexazecimal, de câți biți avem nevoie pentru a-l reprezenta în binar? ***R:*** $4k$

(e) dacă un număr x este reprezentat cu k biți în sistemul binar, de câte cifre avem nevoie pentru a reprezenta x în sistemul hexazecimal? ***R:*** $\frac{k}{4}$

(f) dacă un număr x este reprezentat cu k cifre în sistemul zecimal, de câți biți avem nevoie pentru a-l reprezenta în binar? ***R:*** $\lceil k \cdot \log_2(10) \rceil$

6. Reprezentarea binară a unui număr poate fi extinsă la dreapta, cu numere sub-unitare în felul următor *(binary fixed-point)*:

...	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	...

- calculați valorile în sistemul zecimal pentru următoarele numere reprezentate în formatul de mai sus (punctul stă între 2^0 și 2^{-1}):
  - (a) 101.101 ***R:*** *5.625*
  - (b) 111.001 ***R:*** *7.125*
  - (c) 1110.00111 ***R:*** *14.21875*
  - (d) 1010.0000011 ***R:*** *10.0234375*
  - (e) 1111.0010011 ***R:*** *15.148438*
- calculați valorile în sistemul binar fixed-point pentru următoarele numere reprezentate în formatul zecimal:
  - (a) 3.75 ***R:*** *11.11*
  - (b) 12.3125 ***R:*** *1100.0101*
  - (c) 3.078125 ***R:*** *11.000101*
  - (d) 17.671875 ***R:*** *10001.101011*
  - (e) $\frac{2}{3} \approx 0.6667$ ***R:*** *0.101010101010... (periodic)*
- când au numere reprezentate în acest sistem o reprezentare finită (fără repetiția zecimalelor)? ***R:*** *Atunci când partea fracționară poate fi exprimată ca o sumă finită de puteri negative ale lui 2, adică atunci când numărul fracționar este un multiplu finit al unei puteri negative a lui 2. Altfel, numerele care nu pot fi exprimate astfel vor avea o reprezentare periodică în binar.*
- cum arată un număr negativ reprezentat într-un astfel de sistem de reprezentare? Reprezentați $\frac{5}{8}$ și $-\frac{5}{8}$ în sistemul binary fixed-point (folosiți un sigur bit pentru reprezentarea părții întregi, gândiți-vă și cum folosiți un bit de semn și la reprezentarea în complement față de doi).
  R: *Vom aloca 8 biți (1 byte) pentru reprezentarea unui număr întreg. Similar cu reprezentarea în complement față de 2 a numerelor întregi, aceeași idee se aplică și pentru numerele din sistemul fixed-point: se calculează complementul față de 2 a numărului non-negativ și se adaugă 1 la acesta (un LSB în reprezentarea respectivă). Astfel,* $\frac{5}{8} = 0.625 \rightarrow 0b0000.1010 \implies -\frac{5}{8} = -0.625 \rightarrow 0b1111.0110 = 0b1111.0101 + 0b0000.0001$.

7. Demonstrați cum reprezentarea în complement față de doi a numărului $-x$ poate fi obținută prin inversarea tuturor biților lui x și adunarea lui 1 la rezultat.

R: Fie x un număr întreg pozitiv reprezentat pe N biți. Reprezentarea sa binară este dată de suma:

$$x = \sum_{i=0}^{N-1} b_i \cdot 2^i, \quad b_i \in \{0, 1\}$$

Atunci, reprezentarea în complement față de doi a numărului $-x$ este dată de:

$$-x = 2^N - x = 2^N - \sum_{i=0}^{N-1} b_i \cdot 2^i = \sum_{i=0}^{N-1} (1 - b_i) \cdot 2^i + 1$$

unde $\sum_{i=0}^{N-1} (1 - b_i) \cdot 2^i$ reprezintă inversarea tuturor biților lui x, iar adunarea lui 1 finalizează procesul de obținere a reprezentării în complement față de doi.

8. Demonstrați că metoda asocierii a p termeni consecutivi este soluția corectă pentru problema transformării unui număr din baza B în baza B^p, $p > 1, p \in \mathbb{N}$.

R: Fie un număr N reprezentat în baza B ca o secvență de cifre $d_k d_{k-1} \ldots d_1 d_0$, unde fiecare cifră d_i satisface $0 \leq d_i < B$. Atunci, valoarea acestui număr în baza 10 este dată de:

$$N = \sum_{i=0}^{k} d_i \cdot B^i$$

Pentru a transforma acest număr în baza B^p, împărțim secvența de cifre în grupuri de câte p cifre, începând de la dreapta spre stânga. Fiecare grup de p cifre reprezintă o cifră în baza B^p. Astfel, fiecare grup G_j format din cifrele $d_{jp+p-1} d_{jp+p-2} \ldots d_{jp}$ poate fi convertit în baza B^p prin calculul:

$$C_j = \sum_{i=0}^{p-1} d_{jp+i} \cdot B^i$$

unde C_j este cifra corespunzătoare în baza B^p. Prin această metodă, fiecare grup de p cifre din baza B este transformat într-o singură cifră în baza B^p, păstrând astfel valoarea numerică a numărului original. Aceasta demonstrează că metoda asocierii a p termeni consecutivi este corectă pentru transformarea unui număr din baza B în baza B^p.

9. Demonstrați că $\lfloor log_2 x \rfloor = i_{max}$ unde x este un număr dat pe N biți iar $i_{max} = max\{i | b_i = 1, \forall i = 0, \ldots, N-1\}$ unde b_i reprezintă al i-lea bit din reprezentarea binară a numărului x. De exemplu, dacă $x = 00101110$ (46 zecimal) atunci $\lfloor log_2 x \rfloor = 5$.

R: Fie x un număr reprezentat pe N biți, astfel încât:

$$x = \sum_{i=0}^{N-1} b_i \cdot 2^i, \quad b_i \in \{0, 1\}$$

Fie $i_{max} = \max\{i \mid b_i = 1, \forall i = 0, \ldots, N-1\}$. Atunci, putem observa că: $2^{i_{max}} \leq x < 2^{i_{max}+1}$ Aplicând logaritmul în baza 2 pe toate părțile inegalității, obținem:$i_{max} \leq \log_2(x) < i_{max} + 1$ Din această relație rezultă că:$\lfloor \log_2(x) \rfloor = i_{max}$.

10. În 2014, videoclipul melodiei "Gangnam Style" se apropia de *2.147.483.647* vizualizări (adică $2^{31} - 1$). Cei de la Youtube au trebuit să facă niște modificări pentru a acomoda acest număr mare de vizualizări. Explicați ce s-a întâmplat: care este semnificația numărului de mai sus, care era riscul, și care e soluția pentru rezolvarea potențialei probleme.
R: *Numărul de vizualizări era stocat într-o variabilă de tip întreg pe 32 de biți, folosind reprezentarea în complement față de doi. Astfel, valoarea maximă care putea fi stocată era $2^{31} - 1 = 2.147.483.647$. Când numărul de vizualizări a atins această valoare, orice incrementare ar fi cauzat un overflow, resetând contorul la o valoare negativă sau zero, ceea ce ar fi fost incorect. Soluția a fost trecerea la o reprezentare pe 64 de biți pentru stocarea numărului de vizualizări, permițând astfel un număr mult mai mare de vizualizări fără riscul de overflow. De ce nu* ***unsigned****? regulile interne Youtube, pe scurt.*

11. În Anexa 1 aveți două implementări ale algoritmului de căutare binară; care este cea optimă?

```c
int binarySearch1(int arr[], int start, int end, int x)
{
    if (end >= start)
    {
        int mid = start + (end - start) / 2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] > x)
            return binarySearch1(arr, start, mid - 1, x);
        return binarySearch1(arr, mid + 1, end, x);
    }
    return -1;
}
```

```c
int binarySearch2(int arr[], int start, int end, int x)
{
    if (end >= start)
    {
        int mid = (start + end) / 2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] > x)
            return binarySearch2(arr, start, mid - 1, x);
        return binarySearch2(arr, mid + 1, end, x);
    }
    return -1;
}
```

Seminar 0x01

1. Avem un pachet de cărți de joc (52 de cărți). Luăm cărțile pentru prima dată afară din pachet. Câtă informație avem în acest moment despre cărți? Amestecăm cărțile aleator *(apropo, cum facem asta, algoritmic, eficient?)*. Câtă informație avem acum în pachetul de cărți? *(folosiți și aproximarea lui Stirling pentru calcularea rezultatului)* ***R:*** *Informația inițială este de* $log_2(52!) \approx 225.58$ *biți, iar după amestecare rămâne aceeași, deoarece amestecarea nu schimbă entropia sistemului. Amestecarea se poate face folosind algoritmul Fisher-Yates, care are complexitate O(n).* Aproximarea lui Stirling

2. Se dă o urnă în care avem 5 bile roșii și 3 bile albastre. Ni se spune că cineva extrage o bilă din urnă și aceasta este albastră. Se cere:

(a) câtă informație primim în urma acestei observații?
(b) care a fost entropia urnei înainte de extragere și care este entropia urnei după extragere?
(c) continuați să calculați entropia presupunân că extragem pas cu pas toate bilele albastre;
(d) similar cu cerința anterioară pentru bilele roșii (începând cu urna inițială presupunem că extrageți rând pe rând fiecare bilă roșie și calculați entropia la fiecare pas).

3. Se dau 12 bile. 11 dintre ele au aceeași greutate, iar una este mai ușoară. Folosind o balanță, care este numărul minim de cântăriri necesare pentru a identifica bila mai ușoară?

4.

5.

6.

7. Considerăm următorul mesaj bloc:

D_{00}	D_{01}	D_{02}	P_{0l}
D_{10}	D_{11}	D_{12}	P_{1l}
D_{20}	D_{21}	D_{22}	P_{2l}
P_{c1}	P_{c2}	P_{c3}	P_{cl}

Elementele D_{ij} sunt date (deci avem 9 biți) iar elementele P_{ij} **8.**

9.

10.

Seminar 0x02

Seminar 0x03

Seminar 0x04

Seminar 0x05