



# ARHITECTURA SISTEMELOR DE CALCUL

## Temă

Termen de predare: 26-Mai-2023 23:00

### Context



Povestea îl are în centru pe eroul nostru, Menumorut, care pentru a reuși să țină piept armatei de invadatori, are nevoie de stabilirea unui mecanism de protecție a veștilor pe care le primește de la iscoadele sale. Scopul mecanismului este de a păstra confidențialitatea mesajelor în cazul interceptării acestora de către ostașii inamici.

În urma unei reuniuni a Sfatului Bătrânilor, s-a ajuns la concluzia că singura variantă pentru implementarea mecanismului respectiv este să folosească ceva nemaîntâlnit de oameni până atunci și anume limbajul de asamblare x86.

Menumorut ar dori ca fiecare mesaj să fie mai întâi criptat pe baza unei chei obținute pe baza momentului de timp, apoi codificat pe baza unui alfabet cunoscut. Cheia are un factor de aleatorism iar același mesaj criptat de două ori, la momente diferite de timp, generează două ieșiri total diferite.

### Obiective

Scopul acestei teme este de a implementa în limbaj de asamblare, limbaj înțeles de procesoarele x86 pe 16 biți, mecanismul de criptare-codificare descris în secțiunea anterioară și vizează punctarea următoarelor concepte:

- Utilizarea subfuncțiilor din cadrul vectorului de întrerupere **21h**;
- Utilizarea datelor reprezentate pe diferite dimensiuni, e.g., **BYTE**, **WORD** etc.;
- Stăpânirea elementelor de sintaxă, i.e., utilizarea operațiilor aritmetice, logice și de comparație pentru definirea unor relații complexe, indexarea memoriei, definirea și utilizarea subrutinelor;
- Implementarea unui algoritm simplist de criptare a unui șir de octeți;
- Studiarea și implementarea unui algoritm de codificare a datelor, utilizat în aplicațiile moderne pentru stocarea și partajarea fișierelor media.

### Bibliografie

- *Tabela ASCII*
- *Vectorul de întreruperi MS-DOS 21H*
- *Codificarea Base64 - RFC<sup>1</sup> 4648*

---

<sup>1</sup> **Request for Comments** - document formal ce conține specificații, studii, inovații aplicabile asupra modalității de funcționare a Internetului și apare sub forma unei publicații

## Date introductive

Programul ce trebuie implementat presupune existența unei singure intrări sub forma unui **vector de caractere ASCII** de **dimensiune**  $\leq 48$ , fiecare element fiind selectat din submulțimea de **caractere alfanumerice**<sup>1</sup>. Un șir de intrare va fi citit dintr-un fișier text de intrare, e.g., in.txt, pentru a fi stocat în segmentul de date al programului respectiv. Această funcție de citire va fi pusă la dispoziție în arhiva asociată temei (Vezi secțiunea **Resurse disponibile**).

De exemplu, dacă în fișierul de intrare există textul ‘**Atentie!**’ (Figure 1), atunci vom avea un șir de 8 octeți utili, plasat în memorie și adresat prin intermediul unei etichete.

A	t	e	n	t	i	e	!
41	74	65	6E	74	69	65	21
P <sub>0</sub>	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>	P <sub>5</sub>	P <sub>6</sub>	P <sub>7</sub>

Fie  $n$  lungimea mesajului,  
așadar pe parcursul temei  
textul de intrare va fi  
reprezentat ca un șir de  
caractere  
 $(P_i)_{0 \leq i < n}$

Figure 1: Conținut fișier de intrare

Tema poate fi separată în două funcții principale, prezentate în detaliu în continuare.

- **Criptare**

Criptarea se va realiza sub forma unei operații de **XOR** între fiecare octet din șirul de intrare și câte un octet generat pe baza următoarei relații matematice a **termenului general**

$$x_n = (a * x_{n-1} + b) \mod 255,$$

unde

**a** = Suma literelor prenumelui studentului în reprezentare ASCII modulo 255 (Figure 2),

e.g., pentru **Dragos**, **a** = **D**(68) + **r**(114) + **a**(97) + **g**(103) + **o**(111) + **s**(115) mod 255;

**b** = Suma literelor numelui studentului în reprezentare ASCII modulo 255 (Figure 2),

e.g., pentru **Ioana**, **b** = **I**(73) + **o**(111) + **a**(97) + **n**(110) + **a**(97) mod 255;

**x<sub>n-1</sub>** = Termenul de rang  $n-1$  (termenul anterior) din șirul  $(x_n)_{n \geq 0}$ ;

---

<sup>1</sup> Setul de caractere alfanumerice poate fi regăsit în intervalul de valori [32-126] din tabela ASCII.

**D r a g o s**

$a = (68 + 114 + 97 + 103 + 111 + 115) \bmod 255$   
 $a = 608 \bmod 255$   
 $a = 98$

**I o a n a**

$b = (73 + 111 + 97 + 110 + 97) \bmod 255$   
 $b = 488 \bmod 255$   
 $b = 233$

Termenul inițial al șirului  $x_0$  va fi calculat pe baza timpului actual al sistemului obținut prin intermediul vectorului de întrerupere **21h**, subfuncția **2Ch - GET SYSTEM TIME**, astfel

$$x_0 = ((CH * 3600 + CL * 60 + DH) * 100 + DL) \bmod 255.$$

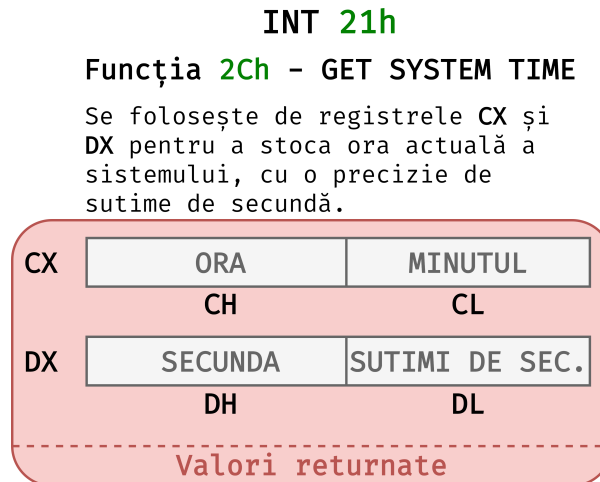


Figure 2: Calcularea coeficienților pentru funcția de generare a cheii de criptare

**Punctarea implementării** urmărește o ordine clară de rezolvare a cerințelor după cum urmează.

1. Calculul valorii termenului inițial  $x_0$  din cheia de criptare  $(x_n)_{n \geq 0}$ .
2. Realizarea unei subrutine **RAND** unde va fi calculat termenul de rang  $n$  pe baza coeficienților  $a, b$  și a termenului anterior de rang  $n - 1$ .
3. Implementarea unei subrutine **ENCRYPT** cu scopul de a realiza operația de criptare dintre șirul dat la intrare și termenii cheii de criptare  $(x_n)_{n \geq 0}$  (Figure 3).

**HINT:** Pe măsură ce sunt generați termenii șirului  $(x_n)_{n \geq 0}$ , se realizează și operația de **XOR** cu octetul curent din șirul de intrare.

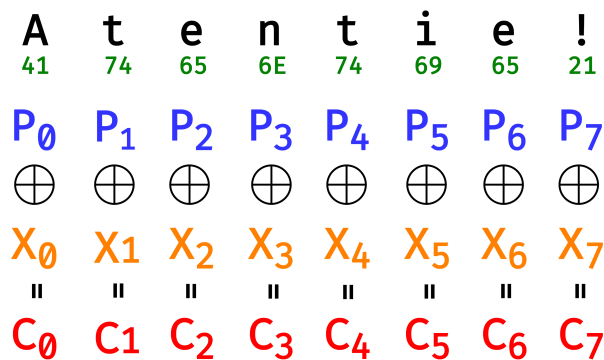


Figure 3: Procesul de criptare

- **Codificare**

Operația de codificare se va realiza asupra șirului criptat și poartă denumirea de **COD64**, o derivare de la codificarea **Base64**. Această transformare lucrează pe grupuri de câte 24 de biți, pe care le separă în 4 secvențe de câte 6 biți, urmând ca fiecare din aceste secvențe să indiceze (Figure 4) un anumit caracter dintr-un alfabet (Tabel 3) de 64 de caractere. În contextul codificării, un bloc reprezintă un grup de 24 de biți iar o secvență reprezintă un grup de 6 biți.

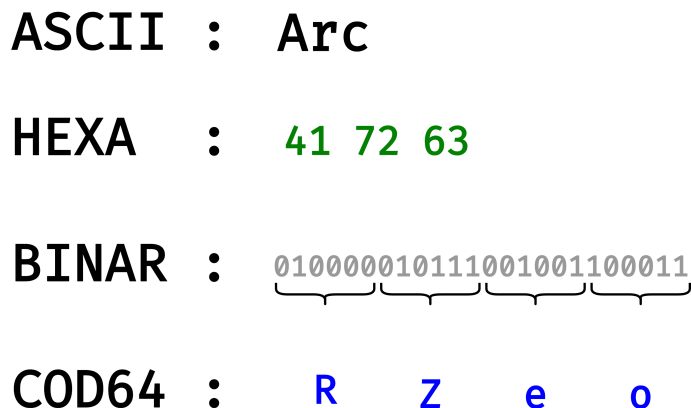



Figure 4: Procesul de codificare, cazul implicit

Dacă numărul de biți din șirul de intrare nu este divizibil cu 24, intervin următoarele cazuri speciale:

- **Cazul I:** (Număr de biți din șirul de intrare) mod 24 = 8  
În cazul acesta se va realiza completarea șirului cu 16 biți (2 octeți) cu valoarea 0, iar din ultimul bloc ce urmează a fi codificat vor rezulta 2 caractere COD64 și două caractere de padding '+',
- **Cazul II:** (Număr de biți din șirul de intrare) mod 24 = 16  
În cazul acesta se va realiza completarea șirului cu 8 biți (1 octet) cu valoarea 0, iar din ultimul bloc ce urmează a fi codificat vor rezulta 3 caractere COD64 și un caracter de padding '+'.  
Se poate observa că pentru fiecare grup de 3 octeți, vor rezulta 4 caractere în urma codificării, deci

**ASCII : Arca**

**HEXA : 41 72 63 61**

**BINAR :** 

**COD64 : R Z e o K R + +**

Figure 5: Procesul de codificare, cazul I

**ASCII : Arcas**

**HEXA : 41 72 63 61 73**

**BINAR :** 

**COD64 : R Z e o K Z f +**

Figure 6: Procesul de codificare, cazul II

și dimensiunea finală a ieșirii poate fi exprimată prin

$$\begin{cases} 4 * \lfloor \frac{n}{3} \rfloor & n \% 3 = 0 \\ 4 * (\lfloor \frac{n}{3} \rfloor + 1) & n \% 3 \neq 0, \end{cases}$$

unde  $n$  este dimensiunea inițială a șirului de intrare. Pentru a vizualiza modul cum evoluează atât dimensiunea, cât și caracterele existente în codificare, se poate porni de la un șir inițial a cărui lungime este incrementată (Tabel 1). Dacă o secvență nulă, i.e., 000000, este întâlnită în interiorul mesajului aceasta se va transforma în caracterul ‘B’, iar dacă este întâlnită la finalul mesajului aceasta se va transforma în caracterul de padding ‘+’.

Șirul rezultat va fi salvat într-un fișier text de ieșire, e.g., out.txt, folosind o funcție dedicată deschiderii și populării fișierelor text. Această funcție de scriere în fișier va fi pusă la dispoziție în arhiva asociată temei (Vezi secțiunea **Resurse disponibile**).

### *Exemplu de rezolvare*

Pentru o mai buna deprindere a modului de implementare, în continuare, vor fi realizate calculele pentru un exemplu particular.

Intrare		Ieșire		Padding
Text	Lungime	Text	Lungime	
Hello W	7	w6I5Y6=S Iq++	12	2
Hello Wo	8	w6I5Y6=S Ib=+	12	1
Hello Wor	9	w6I5Y6=S IbLn	12	0
Hello Wor l	10	w6I5Y6=S IbLn YB++	16	2
Hello Wor ld	11	w6I5Y6=S IbLn Y6R+	16	1
Hello Wor ld!	12	w6I5Y6=S IbLn Y6RU	16	0

Table 1: Exemple de codificare COD64

Presupunem că la execuția programului și tratarea funcției **2Ch** de către vectorul de întreruperi **21h**, momentul de timp este **14:23:38.76**, conform formatului **hh:mm:ss.[SS]**. Valorile salvate în registre sunt

**CH** 0Eh Ora  
**CL** 17h Minut  
**DH** 26h Secundă  
**DL** 4Ch Sutimi de secundă.

Primul pas va consta în calculul valorii termenului inițial  $x_0$  al șirului  $(x_n)_{n \geq 0}$ , obținând:

$$x_0 = (3600 * CH + 60 * CL + DH) * 100 + DL \mod 255$$

$$x_0 = (3600 * 0Eh + 60 * 17h + 26h) * 100 + 4Ch \mod 255$$

$$x_0 = (60 * (60 * 0Eh + 17h) + 26h) * 100 + 4Ch \mod 255$$

$$x_0 = (50.400 + 1380 + 38) * 100 + 76 \mod 255$$

$$x_0 = 51.818 * 100 + 76 \mod 255$$

$$x_0 = 5.181.876 \mod 255$$

$$x_0 = 21 = \mathbf{15h}$$

Coeficienții  $a$  și  $b$  din ecuația termenului general  $x_n = a * x_{n-1} + b \mod 255$ , calculați pentru perechea nume-prenume **Dragos Ioana**, sunt afișați în Figure 2, rezultând formula

$$x_n = 98 * x_{n-1} + 233 \mod 255.$$

În continuare, pe baza formulei anterioare, ar trebui definită o subrutină care să calculeze următorul termen din șir și cu ajutorul căreia se va realiza operația de criptare. Textul de intrare pe care îl vom lua ca exemplu va fi **Scut**.

$$x_1 = 98 * x_0 + 233 \mod 255 = \mathbf{FBh}(251)$$

$$x_2 = 98 * x_1 + 233 \mod 255 = \mathbf{60h}(96)$$

$$x_3 = 98 * x_2 + 233 \mod 255 = \mathbf{CEh}(206)$$

Fișier in1.txt	Fișier out1.txt	
Conținut	Conținut	Explicație
Scut	0x15	$x_0$ (termen inițial)
	0x62	$a$ (primul coeficient)
	0xE9	$b$ (al doilea coeficient)
	0xCE	$x_{n-1}$ ( $x_3$ în cazul actual)
	0x469815BA	mesaj criptat
	14SIkS++	mesaj codificat

Table 2: Exemplu fișiere intrare-ieșire

<b>S</b>	<b>c</b>	<b>u</b>	<b>t</b>
<b>53h</b>	<b>63h</b>	<b>75h</b>	<b>74h</b>
$\oplus$	$\oplus$	$\oplus$	$\oplus$
<b>15h</b>	<b>FBh</b>	<b>60h</b>	<b>CEh</b>
<b>x<sub>0</sub></b>	<b>x<sub>1</sub></b>	<b>x<sub>2</sub></b>	<b>x<sub>3</sub></b>
<b>  </b>	<b>  </b>	<b>  </b>	<b>  </b>
<b>46h</b>	<b>98h</b>	<b>15h</b>	<b>BAh</b>

După realizarea operației de criptare, va fi nevoie de realizarea unor operații suplimentare și anume, numărul de secvențe de 24 de biți (3 octeți) și dimensiunea padding-ului pentru ultimul bloc din text. Pentru exemplul prezentat, avem un singur grup de 3 octeți (**0x46 0x98 0x15**), iar pentru ultimul bloc, format dintr-un singur octet (**0xBA**), va fi nevoie de un padding de 2 octeți.

**46 98 15 BA || 00 00**

**46 98 15 BA 00 00 → 14SI kS++**

Este foarte important ca fișierul de ieșire să respecte cu strictețe formatul de populare, precum în exemplul din Tabelul 2. Verificarea temei va fi automată, iar nerespectarea formatului va genera un *fail* pentru testul curent, i.e., 0 puncte. Unul dintre fișierele de test va fi chiar cel din Tabelul 2.

## Resurse disponibile

Pentru facilitarea implementării temei, în cadrul scheletului asociat specificației de proiect, au fost adăugate o serie de subrutine cu scopul de a se ocupa cu citirea din, respectiv scrierea în fișierele text de intrare/ieșire, astfel:

- **FILE\_INPUT** - realizează citirea șirului din fișierul de intrare specificat de către variabila **filename** și stochează atât caracterele șirului în cadrul variabilei **message**, cât și lungimea acestuia în cadrul variabilei **msglen**.
- **WRITE\_HEX** - realizează scrierea octeților din reprezentarea unui număr în memorie în caractere hexazecimale, e.g., coeficientul  $a$  este reprezentat în memorie sub forma **05 00 → 30 35 30 30**
- **WRITE** - realizează formatarea fișierului de ieșire prin apeluri consecutive ale funcției **WRITE\_HEX** (având ca parametrii coeficienții  $x_0$ ,  $a$ ,  $b$ , respectiv șirul criptat), cât și prin scrierea formei codificate a șirului.

## Observații

Punctajul acordat temei variază între 0 și 10 puncte și va reprezenta o pondere de 40% din nota finală a probei practice. Evaluarea temei va depinde de fiecare punct descris mai jos, în ordine.

1. Tema este individuală. După predarea temelor, acestea vor fi verificate atât pentru plagiat cât și pentru meta-informațiile asociate. În cazul depășirii unui anumit prag de similaritate, ambele teme vor primi 0 puncte;
2. Este interzisă utilizarea platformelor de tip ChatBot, precum ChatGPT. Vor fi efectuate teste de similaritate asupra surselor generate de astfel de platforme. În cazul depășirii unui anumit prag de similaritate, tema va primi 0 puncte;
3. Tema va fi încărcată prin intermediul platformei GitHub. Repository-ul în cadrul căreia va fi încărcată tema, va fi **privat**, pentru a preveni utilizarea sa ca sursă de inspirație și posibilitatea de clasificare ca fiind plagiată. Pe platformă va fi necesară încărcarea unei arhive cu extensia **.zip** care să conțină un fișier **main.asm** cu implementarea temei și un document cu extensia **.docx**. Fișierul **.docx** va avea denumirea **[Nume Prenume][Grupa].docx**, e.g., [Ioana Dragoș][C112F].docx, și va cuprinde o detaliere privind modul de implementare și raționamentul avut în rezolvarea temei. Această documentație va conține și text și imagini (printscreen-uri) care să ajute în explicarea modului de rezolvare. Lipsa fișierului **.docx** va asigura automat 0 puncte pentru temă;
4. În cazul unui *fail* în faza de asamblare/link-editare, tema va primi 0 puncte;
5. Punctajul maxim de 10 puncte va fi distribuit egal între toate fișierele de test. Dintr-un punctaj acordat per fișier de test, ponderile pentru calcularea corectă a valorilor  $x_0, a, b, x_{n-1}$ , mesaj criptat și mesaj codificat vor fi de 10%, 5%, 5%, 20%, 20%, respectiv 40%.



Index	Binar	Simbol	Index	Binar	Simbol
0	000000	B	32	100000	S
1	000001	q	33	100001	U
2	000010	m	34	100010	2
3	000011	g	35	100011	o
4	000100	p	36	100100	v
5	000101	8	37	100101	J
6	000110	6	38	100110	O
7	000111	C	39	100111	h
8	001000	P	40	101000	W
9	001001	e	41	101001	4
10	001010	9	42	101010	l
11	001011	D	43	101011	y
12	001100	f	44	101100	5
13	001101	N	45	101101	E
14	001110	z	46	101110	k
15	001111	7	47	101111	r
16	010000	R	48	110000	q
17	010001	l	49	110001	s
18	010010	w	50	110010	n
19	010011	j	51	110011	A
20	010100	H	52	110100	x
21	010101	I	53	110101	u
22	010110	M	54	110110	b
23	010111	Z	55	110111	T
24	011000	K	56	111000	V
25	011001	G	57	111001	0
26	011010	c	58	111010	3
27	011011	Y	59	111011	a
28	011100	X	60	111100	=
29	011101	i	61	111101	L
30	011110	F	62	111110	/
31	011111	t	63	111111	d
Padding			+		

Table 3: Alfabetul COD64