



Tehnici de programare

Operații pe biți

ovidiu.banias@aut.upt.ro

Operații pe biți

Reprezentarea întregilor

Operanzi și operații pe biți în C

Sistemul binar de numerație

- Ce este un bit?

O cifră în sistemul binar

- Ce valori poate stoca un bit?

$\{0,1\}$

- De ce calculatoarele folosesc sistemul binar? De ce nu cel zecimal?

Mai ușor de implementat hardware

Bit-ul

➤ Bit : $\{0,1\}$



➤ 4 biti : 0000 \rightarrow 1111 – $2^4=16$ valori distincte

➤ 8 biti : 00000000 \rightarrow 11111111 – $2^8=256$ valori distincte

➤ 16 biti : $2^{16}=65.535$

➤ 24 biti : $2^{24} \approx 16.7$ milioane

➤ 32 biti : $2^{32} \approx 4$ Miliarde

Octet-ul

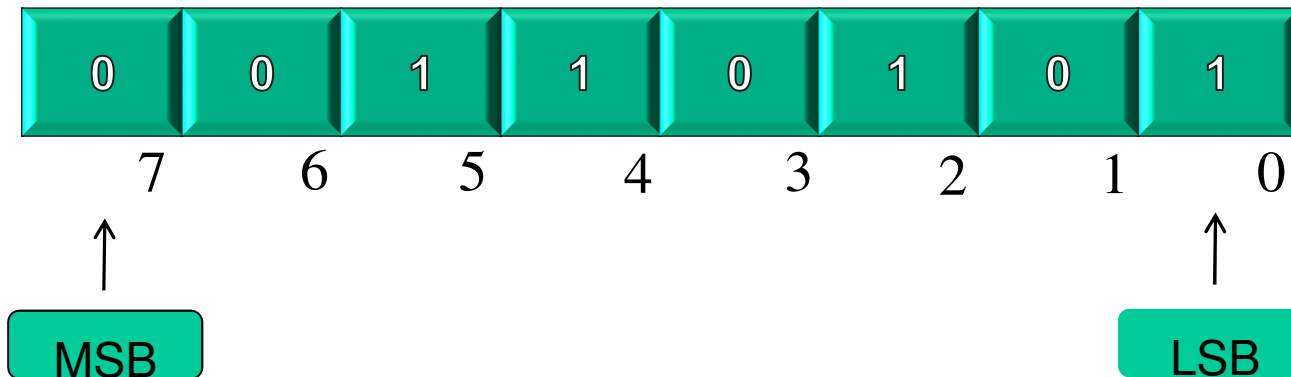
- Ce este un octet (byte)?



1 octet = 8 biti

$$00000000 \rightarrow 11111111$$

$$0*2^7 + 0*2^6 + 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 53$$



Octet-ul

$X_{(10)}$	$X_{(2)}$
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
13	1101
21	00010101
253	11111101

➤ $X_{(10)} \rightarrow X_{(2)}$?

➤ $X_{(2)} \rightarrow X_{(10)}$?

➤ Cum se reprezintă în binar numerele negative?

Complement de 1

Complement de 2



Tipul de dată char / unsigned char (ASCII)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	:	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Exemple

$X_{(10)}$	$X_{(2)}$	$X_{(8)}$	$X_{(16)}$
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B

$X_{(10)}$	$X_{(2)}$	$X_{(8)}$	$X_{(16)}$
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
36	100100	44	24
61	111101	75	3D
79	1001111	117	4F
100	1100100	144	64
140	10001100	214	8C
254	11111110	376	FE
255	11111111	377	FF

Operatori pe biți



Operator	Denumire	Exemplu	Descriere
<code>~</code>	Not	<code>~ v</code>	Inversează biții var. v
<code>&</code>	And	<code>v₁ & v₂</code>	<code>0&0=0, 0&1=0, 1&1=1</code>
<code> </code>	Or	<code>v₁ v₂</code>	<code>0 0=0, 0 1=1, 1 1=1</code>
<code>^</code>	Xor (Sau exclusiv)	<code>v₁ ^ v₂</code>	<code>0^0=0, 0^1=1, 1^1=0</code>
<code><<</code>	Shift left	<code>v << i</code>	Deplasează var. v la stânga cu i poziții
<code>>></code>	Shift right	<code>v >> i</code>	Deplasare la dreapta cu i poziții

Operații pe biți



v_1	v_2	$v_1 \& v_2$
0	0	0
0	1	0
1	0	0
1	1	1

v_1	v_2	$v_1 v_2$
0	0	0
0	1	1
1	0	1
1	1	1

v_1	v_2	$v_1 \wedge v_2$
0	0	0
0	1	1
1	0	1
1	1	0

v	$\sim v$
0	1
1	0

v	i	$v \ll i$
0001	1	000000 10
0101	2	000 10100
0001	3	0000 1000
1001	4	10010000
0011	5	0 1100000

v	i	$v \gg i$
00110110	1	000 11011
01010011	2	000 10100
00000011	3	00000000
00011001	4	0000000 1
11111111	5	00000 111

Măști/Șabloane (Mask)

Măștile se folosesc pentru accesarea, setarea anumitor biți dintr-un octet/cuvânt. Fie **n** – octet și **k={0,...,7}** – poziția unui bit în octet

SetFlag – setează bitul de pe poziția k la valoarea 1

$$n = n | (1 \ll k)$$

UnsetFlag – setează bitul de pe poziția k la valoarea 0

$$n = n \& \sim (1 \ll k)$$

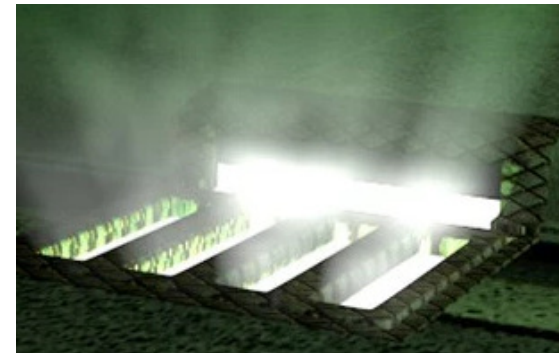
GetFlag – returnează valoarea bitului de pe poziția k

$$(n \& (1 \ll k) == (1 \ll k))$$

$$(1 \& (n \gg k))$$

ChangeFlag – schimbă valoarea bitului de pe poziția k

$$n = n ^ (1 \ll k)$$



Probleme

1. Se dă o mulțime de numere naturale cu valori de la 0 la 1000. Se citesc de la intrare diferite valori în intervalul $[0,1000]$, să se utilizeze un vector de octeți de dimensiune cât mai mică pentru memorarea elementelor mulțimii.
2. Fiind necesară setarea/verificarea permisiunilor pentru utilizatorii unei aplicații, și presupunând că informațiile legate de permisiune (per utilizator) pot fi salvate doar într-o variabilă de dimensiunea unui octet (limitări de memorie), să se implementeze funcții pentru setarea și verificarea permisiunilor unui anumit utilizator. Există 5 tipuri de permisiuni: Read/Write/Delete/Rename/Copy.