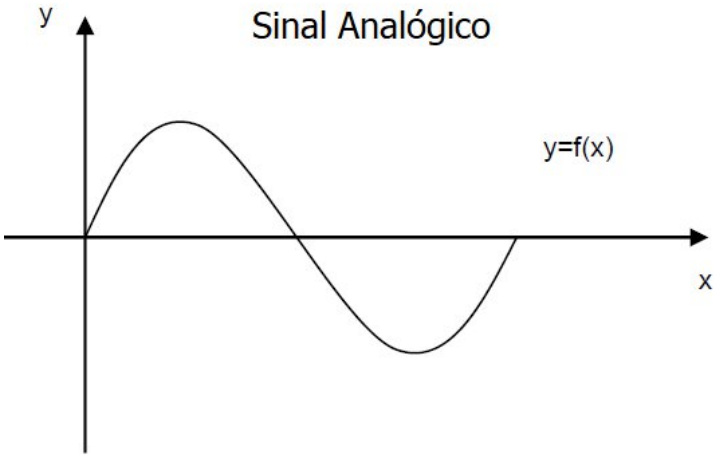
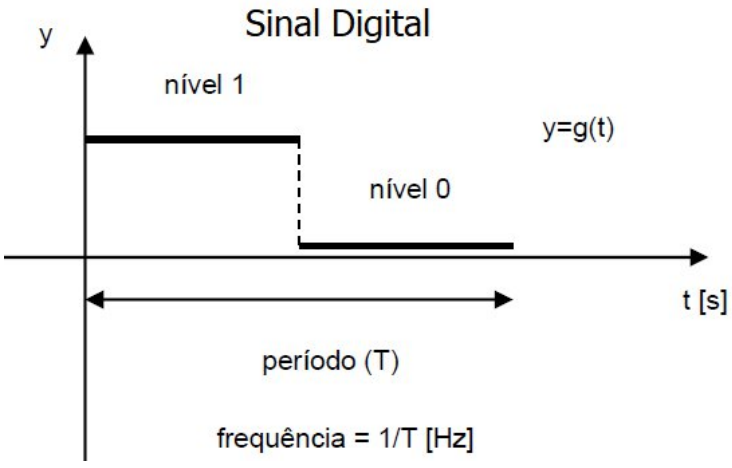
PUC Minas	SISTEMAS DE NUMERAÇÃO E REPRESENTAÇÕES DE DADOS		
	CAMPUS PRAÇA DA LIBERDADE		
	Turno: M	Período:2	Data: 01/08/2021
	GUIA 2	Valor: 5	Nota:
	Disciplina: Arquitetura de Computadores I		
	Professor: JÚLIO CONWAY, DSc		

Representação de dados

Função contínua em um intervalo.



Função discreta em um intervalo.



Computadores Analógicos x Digitais

- com estados contínuos (corrente, tensão, pressão, vazão etc.)
- com estados discretos (ou valores distintos distribuídos

Sistemas de Numeração
Exemplo:

Sistema decimal
$1x2^7+0x2^6+1x2^5+0x2^4+0x2^3+0x2^2+1x2^1+1x2^0$ - forma canônica
$128 + 0 + 32 + 0 + 0 + 0 + 2 + 1 = 163_{(10)}$

Sistema binário
$1010\ 0011_{(2)}$ - número na base 2
representado apenas com algarismos {0,1}

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	potências da base 2
128	64	32	16	8	4	2	1	valor equivalente da potência
1	0	1	0	0	0	1	1	coeficientes

Equivalentes em sistemas com potências de 2
$1010\ 0011_{(2)} = [1010]\ [0011]_{(16)} = A3_{(16)}$ e $A_{(16)}=10$ em hexadecimal (grupos de 4)
$= 10x16^1+ 3x16^0 = 163_{(10)}$ com algarismos
{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

OBS: Caso necessário, completar com zeros (0) para formar grupos de mesmo tamanho.

1.) Converter decimal para binário

Sistema decimal
$163_{(10)} = 1x10^2+6x10^1+3x10^0$ - na forma canônica

Para converter um valor decimal (base=10) para binário (base=2),
usar divisões sucessivas por 2 e tomar os restos na **ordem inversa**
em que forem calculados:

operação		quociente		resto	
163 / 2	=	81	+	1	(último)
81 / 2	=	40	+	1	
40 / 2	=	20	+	0	
20 / 2	=	10	+	0	
10 / 2	=	5	+	0	
5 / 2	=	2	+	1	
2 / 2	=	1	+	0	
1 / 2	=	0	+	1	(primeiro)

Sistema binário
$1010\ 0011_{(2)}$ - número na base 2

ou

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	potências da base 2
128	64	32	16	8	4	2	1	valor equivalente da potência
1	0	1	0	0	0	1	1	coeficientes

2.) Converter binário para decimal

Para converter um valor binário (base=2) para decimal (base=10),
usar a soma dos produtos de cada algarismo pela potência da base
equivalente à posição:

Sistema binário
$1010\ 0011_{(2)}$ - número na base 2

Sistema decimal
$1x2^7+0x2^6+1x2^5+0x2^4+0x2^3+0x2^2+1x2^1+1x2^0$ - forma canônica
$128 + 0 + 32 + 0 + 0 + 0 + 2 + 1 = 163_{(10)}$

3.) Converter decimal para base 16 (hexadecimal)

Para converter um valor decimal para a base 16 (hexadecimal):

operação	quociente	resto	
163 / 16 =	10	+ 3	(último)
10 / 16 =	0	+ 10	(primeiro, substituindo pelo algarismo A=10)

Sistema hexadecimal
$A3_{(16)}$ - número na base 16

4.) Converter da base 16 para decimal

Sistema hexadecimal
$A3_{(16)} = (A=10)x16^1+3x16^0$ - número na base 16 forma canônica
$= 160 + 3 = 163_{(10)}$

5.) Sistema binário (base=2) para quaternário (base=16=2^4):
$1010\ 0011_{(2)} = [1010]\ [0011]_{(16)} = A3_{(16)}$ e $A_{(16)}=10$ agrupar de 4 em 4
e substituir pelos dígitos equivalentes

OBS: Caso necessário, completar com zeros para formar os grupos.

Tabela de Conversão Decimal-Binário-Hexadecimal

$X_{(10)}$ decimal	$X_{(2)}$ binário	$X_{(16)}$ hexadecimal
00	0000 0000	00
01	0000 0001	01
02	0000 0010	02
03	0000 0011	03
04	0000 0100	04
05	0000 0101	05
06	0000 0110	06
07	0000 0111	07
08	0000 0000	08
09	0000 0001	09
10	0000 0010	0A
11	0000 0011	0B
12	0000 0100	0C
13	0000 0101	0D
14	0000 0110	0E
15	0000 0111	0F

Representações de potências de 2.

x	2^x	$X_{(10)}$	$X_{(2)}$	$X_{(16)}$
0	2^0	1	1	1
1	2^1	2	10	2
2	2^2	4	100	4
3	2^3	8	1000	8
4	2^4	16	1 0000	10
5	2^5	32	10 0000	20
6	2^6	64	100 0000	40
7	2^7	128	1000 0000	80
8	2^8	256	1 0000 0000	100
9	2^9	512	10 0000 0000	200
10	2^{10}	1024	100 0000 0000	400

Termos associados à representação de dados em binário

Termo	Quantidade	Observação
bit	1	"***b****inary dig****it***" – dígito binário (0 ou 1)
nibble	4 bits	dígito hexadecimal equivalente (semiocteto)
byte	8 bits	octeto (Werner Buchholz, 1956) – unidade de armazenamento
word	xx bits	dependente do sistema (ex.: 14, 16, 32, 54, 64 etc.)
kiloBytes (kB)	1024 Bytes	(ex.: arquivo texto)
MegaBytes (MB)	1024 kiloBytes (kB)	1 048 576 bytes (ex.: arquivo mp3)
GigaBytes (GB)	1024 MegaBytes (MB)	1 073 741 824 bytes (ex.: filme)
TeraBytes (TB)	1024 GigaBytes (GB)	1 099 511 627 776 bytes (ex.: 800 filmes)
PetaBytes (PB)	1024 TeraBytes (TB)	1 125 899 906 842 624 bytes (ex.: acervo do Google)
ExaBytes (EB)	1024 PetaBytes (PB)	(ex.: acervo da Internet)
ZetaBytes (ZB)	1024 ExaBytes (EB)	
YottaByes (YB)	1024 ZetaByes (ZB)	

Representação de símbolos por códigos equivalentes (Tabela ASCII - 8 bits)

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	&#127;	DEL

Source: www.asciitable.com

Tabela ASCII Estendida

128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	│	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	169	_	185	╣	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	171	½	187	╗	203	╦	219	▉	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	_
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	▒	192	└	208	╨	224	α	240	≡		

Source: www.asciitable.com

Exemplos:

'0' = $0011\ 0000_{(2)} = 30_{(16)} = 48_{(10)}$

'A' = $0100\ 0001_{(2)} = 41_{(16)} = 65_{(10)}$

'a' = $0110\ 0001_{(2)} = 61_{(16)} = 97_{(10)}$

"Computador" = $43\ 6F\ 6D\ 70\ 75\ 74\ 61\ 64\ 7F\ 72_{(16)}$

Sistemas de Numeração – Operações aritméticas

Exemplos:

1.) Adição

Sistema binário

Relações fundamentais:

$0_{(2)} + 0_{(2)} = 0_{(2)}$
$0_{(2)} + 1_{(2)} = 1_{(2)}$
$1_{(2)} + 0_{(2)} = 1_{(2)}$
$1_{(2)} + 1_{(2)} = 10_{(2)}$ (zero e "vai-um" para a próxima potência)

Aplicação:

$$\begin{array}{rl} {\scriptstyle 1\,1\,1\,1} & \leftarrow \text{"vai-um"} \\ 101101_{(2)} & \leftarrow \text{operando 1} \\ +\quad 111_{(2)} & \leftarrow \text{operando 2} \\ \hline 110100_{(2)} & \leftarrow \text{resultado} \end{array}$$

Sistema hexadecimal

Aplicação:

$$\begin{array}{rlrl} {\scriptstyle 1\,1\,1\,1} & & {\scriptstyle 1} & \leftarrow \text{"vai-um" (excessos de 16)} \\ 101101_{(2)} & & 2D_{(16)} & \leftarrow \text{operando 1} \\ +\quad 111_{(2)} & & +\ 7_{(16)} & \leftarrow \text{operando 2} \\ \hline 110100_{(2)} & & 34_{(16)} & \leftarrow \text{resultado} \end{array}$$

Subtração

Relações fundamentais:

$0_{(2)} - 0_{(2)} = 0_{(2)}$
$0_{(2)} - 1_{(2)} = ???$
$1_{(2)} - 0_{(2)} = 1_{(2)}$
$1_{(2)} - 1_{(2)} = 0_{(2)}$
$10_{(2)} - 1_{(2)} = 01_{(2)}$ (zero e "vem-um" para a potência considerada)
$100_{(2)} - 1_{(2)} = 011_{(2)}$ (zero e "vem-um" para as potências necessitadas)

Aplicação:

	1	(10)		1	← "vem-um"
$101101_{(2)}$	$10100\underline{1}_{(2)}$	$1010\ (0)1_{(2)}$	$100\ (10)\ 01_{(2)}$	$101101_{(2)}$	← operando 1
- $111_{(2)}$	- $111_{(2)}$	- $1\ 1\ 1_{(2)}$	- $1\ 11_{(2)}$	- $111_{(2)}$	← operando 2
$0_{(2)}$	$0_{(2)}$	$1\ 0_{(2)}$	$100\ 1\ 10_{(2)}$	$100110_{(2)}$	← resultado

OBS:
Quando se "toma emprestado" na potência seguinte,
um valor unitário é debitado na potência que "empresta",
e "creditado" na potência que o recebe, compensada a diferença entre essas potências.

Multiplicação

Sistema binário

Relações fundamentais:

$0_{(2)} * 0_{(2)} = 0_{(2)}$
$0_{(2)} * 1_{(2)} = 0_{(2)}$
$1_{(2)} * 0_{(2)} = 0_{(2)}$
$1_{(2)} * 1_{(2)} = 1_{(2)}$

Aplicação:

$101101_{(2)}$ ← operando 1
* $101_{(2)}$ ← operando 2

```
  1111
   101101
+  000000-
  101101--
```

$11100001_{(2)}$ ← resultado

Divisão

Sistema binário

Aplicação:

$11100001_{(2)} \mid 101_{(2)}$
$-\ 101 \qquad 1_{(2)}$

010

$11100001_{(2)} \mid 101_{(2)}$
$-\ 101 \qquad 101_{(2)}$

01000
$-\ \ 101$

00011

$11100001_{(2)} \mid 101_{(2)}$
$-\ 101 \qquad 10110_{(2)}$

01000
$-\ \ 101$

000110
$-\ \ 101$

00000101

$11100001_{(2)} \mid 101_{(2)}$
$-\ 101 \qquad 10_{(2)}$

0100

$11100001_{(2)} \mid 101_{(2)}$
$-\ 101 \qquad 1011_{(2)}$

01000
$-\ \ 101$

000110
$-\ \ 101$

0000010

$11100001_{(2)} \mid 101_{(2)}$
$-\ 101 \qquad 101101_{(2)}$

01000
$-\ \ 101$

000110
$-\ \ 101$

00000101
$-\ \ \ 101$

00000000

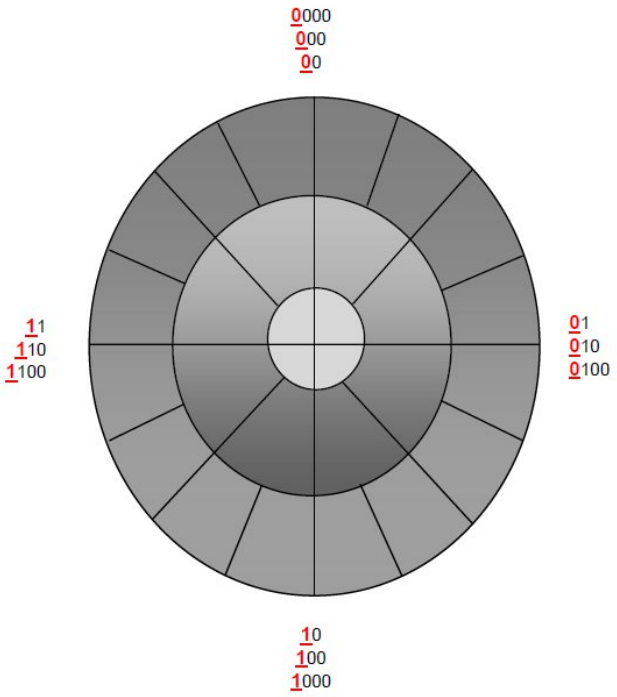
Sistemas de Numeração – Representações de dados

A representação de dados numéricos necessita, por vezes, utilizar uma indicação especial para sinal (positivo e negativo). Para isso, é comum reservar o primeiro bit (o mais a direita para isso), em valores inteiros ou reais. Entretanto, a representação de valores negativos necessitará de ajustes a fim de que as operações aritméticas produzam resultados coerentes.

Representações para tipos de dados comuns (em Java)

Tipos		Intervalo	Tamanho
boolean ***false, true***		[false:true]	1 byte
byte 0, 0x00		[-128 : 127] [0 : 255] (sem sinal)	1 byte
char '0','\u0000'		[0 : 65535]	2 bytes (Unicode)
short 0	± a	[-32768 : 32767] (sinal+amplitude)	2 bytes
int 0	± a	[-2^{31} : 2^{31}-1] (sinal+amplitude)	4 bytes
long 0L	± a	[-2^{63} : 2^{63}-1] (sinal+amplitude)	8 bytes
float 0.0f	± e 1.m IEEE754	[$-3.4e^{-38}$: $3.4e^{38}$] (sinal+amplitude+1	4 bytes .mantissa)
double 0.0, 0.0e0	± e 1.m IEEE754	[$-1.7e^{-308}$: $1.7e^{308}$] (sinal+amplitude+1.	8 bytes .mantissa)
String "", "0", ***null***	. . .		n bytes

Representação binária dependente do número de bits.



A representação binária depende da quantidade de bits disponíveis e dos formatos escolhidos.

Para os valores inteiros, por exemplo, pode-se utilizar o formato em que o primeiro bit, à esquerda, para o sinal e o restante para a amplitude, responsável pela magnitude (grandeza) do valor representado.

Exemplo:

$5_{(10)} = 101_{(2)}$

$+5_{(10)} = \underline{0}101_{(2)}$

$- 5_{(10)} = \underline{1}101_{(2)}$

Essa represesentação, contudo, não é conveniente para realizar operações, pois ao adicionar ambos, obtém-se:

$+5_{(10)} = \underline{0}101_{(2)}$
$- 5_{(10)} = \underline{1}101_{(2)}$

$0_{(10)} = (\underline{1})\ 0010_{(2)}$

o que ultrapassa a quantidade de bits originalmente escolhida e, obviamente, não é igual a zero em sua amplitude.

Complemento de 1

Uma das possíveis representações para valores negativos pode ser aquela onde se invertem os valores individuais de cada bit.

Exemplo:

$5_{(10)} = 101_{(2)}$

$+5_{(10)} = \underline{0}101_{(2)}$

$- 5_{(10)} = \underline{1}010_{(2)}$ (complemento de 1)

Essa represesentação, contudo, também não é conveniente para realizar operações, pois ao adicionar ambos, obtém-se:

$+5_{(10)} = \underline{0}101_{(2)}$
$- 5_{(10)} = \underline{1}010_{(2)}$

$- 0_{(10)} = \underline{1}111_{(2)} \rightarrow +0_{(10)} = \underline{0}000_{(2)}$

o que mantém a quantidade de bits originalmente escolhida, mas gera duas representações para zero (-0) e (+0), o que requer ajustes adicionais nas operações.

Complemento de 2

Outra das possíveis representações para valores negativos pode ser aquela onde se invertem os valores individuais de cada bit, e acrescenta-se mais uma unidade ao valor encontrado, buscando completar o que falta para atingir a próxima potência da base.

Exemplo:

$5_{(10)} = 101_{(2)}$

$+5_{(10)} = \underline{0}101_{(2)}$

$- 5_{(10)} = \underline{1}010_{(2)}$ (complemento de 1, ou $C_1(5)$)

$- 5_{(10)} = \underline{1}011_{(2)}$ (complemento de 2, ou $C_2(5)$)

Essa represesentação é bem mais conveniente para realizar operações, pois ao adicionar ambos, obtém-se:

$+5_{(10)} = \underline{0}101_{(2)}$
$- 5_{(10)} = \underline{1}011_{(2)}$

$0_{(10)} = (\underline{1})\ \underline{0}000_{(2)}$

com uma única representação para zero, mas com um excesso (1) que não é comportado pela quantidade de bits originalmente escolhida. Porém, se desprezado esse excesso, o valor poderá ser considerado correto, com a ressalva de que a quantidade de bits deverá ser rigorosamente observada (ou haverá risco de transbordamento – OVERFLOW).

Para efeitos práticos, o tamanho da representação deverá ser sempre indicado, e as operações deverão ajustar os operandos para a mesma quantidade de bits (de preferência, a maior possível).

Exemplo:

$5_{(10)} = 101_{(2)}$

$+5_{(10)} = \underline{0}101_{(2)}$

$- 5_{(10)} = \underline{1}010_{(2)}$ (complemento de 1, com 4 bits ou C_{14} (5))

$- 5_{(10)} = \underline{1}011_{(2)}$ (complemento de 2, com 4 bits ou C_{24} (5))

logo,

$C_{15}(5) = C_1(\underline{0}0101_{(2)}) = \underline{1}1010_{(2)}$

$C_{25}(5) = C_2(\underline{0}0101_{(2)}) = \underline{1}1011_{(2)}$

$C_{18}(5) = C_1(\underline{0}0000101_{(2)}) = \underline{1}1111010_{(2)}$

$C_{28}(5) = C_2(\underline{0}0000101_{(2)}) = \underline{1}1111011_{(2)}$

De modo inverso, dado um valor em complemento de 2, se desejado conhecer o equivalente positivo, basta retirar uma unidade e substituir os valores individuais de cada dígito binário.

Exemplo:

$\underline{1}011_{(2)}$ (complemento de 2, com 4 bits)

$\underline{1}011_{(2)} - 1 = \underline{1}010_{(2)}$ e invertendo $\underline{0}101_{(2)} = +5_{(10)}$

logo, $\underline{1}011_{(2)} = -5_{(10)}$

Portanto, para diferentes quantidades de bits:

$\underline{1}1011_{(2)} = \underline{1}1010_{(2)} = \underline{0}0101_{(2)} = 5_{(10)}$

$\underline{1}1111011_{(2)} = \underline{1}1111010_{(2)} = \underline{0}0000101_{(2)} = 5_{(10)}$

Resumindo:

- correspondente, aplica-se o complemento de 2 (inverte todos os bits e soma 1)
- correspondente, aplica-se o complemento de 2 (inverte todos os bits e soma 1)

Subtração mediante uso de complemento

Operar a subtração mediante uso de complemento pode ser mais simples do que realizar a operação diretamente, como visto anteriormente.

Aplicação:

		1		(10)				1	← "vem-um"
	$101101_{(2)}$		$10100\,1_{(2)}$		$1010\,(0)\,1_{(2)}$		$100\,(10)\,01_{(2)}$	$101101_{(2)}$	← operando 1
-	$111_{(2)}$	-	$111_{(2)}$	-	$1\ 1\ 1_{(2)}$	-	$1\ 11_{(2)}$	- $111_{(2)}$	← operando 2
	$0_{(2)}$		$0_{(2)}$		$1\ 0_{(2)}$		$100\ 1\ 10_{(2)}$	$100110_{(2)}$	← resultado

OBS:
Quando se "toma emprestado" na potência seguinte,
um valor unitário é debitado na potência que "empresta",
e "creditado" na potência que o recebe, compensada a diferença entre essas potências.

Aplicação do complemento:

Para aplicar o complemento, a primeira providência é normalizar os operandos na mesma quantidade de bits, reservado o bit de sinal.

	$101101_{(2)}$	→	$0\ 101001_{(2)}$
-	$111_{(2)}$	→ -	$0\ 000111_{(2)}$

Em seguida, calcular e substituir o subtraendo pelo complemento:

$C2\ (\ 0\ 000111_{(2)}\ ) = C1\ (\ 0\ 000111_{(2)}\ ) + 1_{(2)} = 1\ 111000_{(2)} + 1_{(2)} = 1\ 111001_{(2)}$

	$101101_{(2)}$	→	$0\ 101001_{(2)}$
-	$111_{(2)}$	→ -	$1\ 111001_{(2)}$

Para finalizar, operar a **soma** dos operandos, respeitando a quantidade de bits:

		(1) 1 1 1 1		← "vai-um"
	$101101_{(2)}$	→	$0\ 101001_{(2)}$	
-	$111_{(2)}$	+	$1\ 111001_{(2)}$	
			$0\ 100110$	

Observar que o bit que exceder a representação deverá ser desconsiderado, por não haver onde acomodá-lo. Ainda poderá haver erro por transbordamento (OVERFLOW).

ATIVIDADE A SER ENTREGUE – REGRAS GERAIS

1. Este GUIA 2 deverá ser entregue em um único arquivo PDF (**outras formas não serão aceitas, tornando nula a atividade)**. **Dica**: Vá confeccionando o relatório em Word (.doc), acrescentando as respostas aos exercícios, listagens de programas e figuras, se for o caso, e ao final, converta para PDF.

2. Como nosso curso é de **Ciência da Computação**, vamos dar também um enfoque de programação nas atividades deste GUIA. Assim, a atividade inclui a confecção de pequenos programas de teste. Neste relatório o aluno poderá escrever o programa na linguagem que está mais familiarizado, preferencialmente, a linguagem adotada no curso no semestre atual.

   Poderá confeccionar o programa para ser executado em modo texto (resultado na tela do DOS), ou modo gráfico, se tiver conhecimento desta técnica.

   Deverá ser incluída no relatório a listagem do (s) programa (s), bem como a exibição do(s) resultado(s) na tela (tela de saída do DOS em modo texto ou modo gráfico).

3. Para facilitar o estudo da parte teórica do GUIA, o aluno deverá assistir aos seguintes vídeos (somente sistemas decimal, binário e hexadecimal):

   http://www.youtube.com/user/henriquencunha/videos
   http://www.youtube.com/watch?v=Ojd770C2GTk

4. Para as atividades abaixo, mostrar as etapas para a resposta aos itens de '*a'* até '*e'*. **Sugestão**: Para a representação dos tipos binário e hexadecimal, pode ser utilizado o tipo String.

   4.1. Fazer as conversões de decimal para binário:
   a.) 27(10) = X(2)
   b.) 51(10) = X(2)
   c.) 713(10) = X(2)
   d.) 312(10) = X(2)
   e.) 360(10) = X(2)

   4.2. Escrever uma função **`dec2bin(x)`**. Esta função recebe um número inteiro decimal e devolve o binário correspondente. Faça um programa **`main`** que passa para a função os números decimais dos itens de '*a'* até '*e'* e que imprima os binários correspondentes na tela. Mostre o código e os resultados exibidos na tela, no relatório.

4.3. Fazer as conversões de binário para decimal:
a.) 10101(2) = X(10)
b.) 11010(2) = X(10)
c.) 101001(2) = X(10)
d.) 111001(2) = X(10)
e.) 100011(2) = X(10)

4.4. Escrever uma função **`bin2dec(x)`**. Esta função recebe um número binário e devolve o decimal correspondente. Faça um programa **`main`** que passa para a função os números binários dos itens de '*a'* até '*e'* e que imprima os binários correspondentes na tela. Mostre o código e os resultados exibidos na tela, no relatório.

4.5. Fazer as conversões de decimal para hexadecimal:
a.) 73(10) = X(16)
b.) 47(10) = X16)
c.) 61(10) = X(16)
d.) 157(10) = X(16)
e.) 171(10) = X(16)

4.6. Escrever uma função **`dec2hex(x)`**. Esta função recebe um número inteiro decimal e devolve o hexadecimal correspondente. Faça um programa **`main`** que passa para a função os números decimais dos itens de '*a'* até '*e'* e que imprima os hexadecimais correspondentes na tela. Mostre o código e os resultados exibidos na tela, no relatório.

4.7. Fazer as conversões de hexadecimal para decimal:
a.) 73(16) = X(10)
b.) ABC(16) = X10)
c.) 100(16) = X(10)
d.) 9A8(16) = X(10)
e.) 100016) = X(10)

4.8. Escrever uma função **`hex2dec(x)`**. Esta função recebe um número hexadecimal e devolve o decimal correspondente. Faça um programa **`main`** que passa para a função os números hexadecimais dos itens de '*a'* até '*e'* e que imprima os decimais correspondentes na tela. Mostre o código e os resultados exibidos na tela, no relatório.

4.9. Converter entre símbolos e códigos de representação alfanumérico (ASCII). **Sugestão**: veja a codificação da tabela ASCII acima.
a.) "PUC-Minas" = X(16_ASCII) (converter para os nove algarismos hexa corresp.)
b.) "2021-1" = X(16_ASCII)
c.) "Brasil" = X(16_ASCII)
d.) 124 101 122 104 105(16) = X(ASCII) (converter para o texto ASCII corresp.)
e.) 62 2E 68 2E 2D 6D 67(16) = X(ASCII)

4.10.
Escrever a função **`ASCII2hex(x).`** Esta função recebe um texto ASCII e devolve os caracteres hexadecimais correspondentes.
Escrever a função **`hex2ASCII(xx)`**. Esta função recebem caracteres hexadecimais e devolve o texto ASCII correspondentes.
Faça um programa **`main`** que teste as funções acima. Mostre o código e os resultados exibidos na tela, no relatório

Bibliografia

Guias Práticos – Arquitetura de Computadores I – Professor Teldo Cruz Franqueira

Apostila Introdução aos Sistemas de Numeração – Professor Júlio C. D. Conway