데이터의 종류

컴퓨터 구조

김동완

INDEX

Alpha Numeric : 수치, 문자, 특수문자 등을 입력하여 메모리에 저장하기 위해 '0'과 '1로 코드화 한 것

영문자, 숫자 등을 표현할 수 있는 ASCII 코드
다국어 표현이 가능한 유니코드 등

넓은 의미

좁은 의미

Numeric(수치코드)

산술 연산용 숫자

수치 만을 표현할 수 있는 코드

계산을 위해 만들어진 컴퓨터의
본 목적을 만족시킴

0, 1, 2, ... 8,9 (10개)

Alpha (영문자 코드)

데이터 처리용 영문자

대문자 A,B, ... Y, Z (26자)

소문자 a,b, ... y, z (26자)

* 영어를 제외한 외국어는 영문자의 확장판

특수 목적용 기호 (Special)

특수 문자, 제어문자 등을 표현

!,@,#,$, ... *,(,+,-,}

진수와 진법

$10 = 1 \times 2^{1} + 0 \times 2^{0} = 2$

진법 - 0부터 n개의 숫자를 사용하여 수를 표현하는 방법, 0~(n-1)만큼 표현

진수 - 진법으로 나타내어진 수로 n진법으로 나타낸 수

기수(radix) - 진법(숫자 표현)에 기준이 되는 수 ex)radix-2 : 2진 {0,1}, radix-10 : 10진 {0,1,...9}

41.5 변환해보기

10진수 : $4 \times 10^{1} + 1 \times 10^{0} + 5 \times 10^{-1} = 41.5$

2진수 : $1 \times 2^{5} + 0 \times 2^{4} + 1 \times 2^{3} + 0 \times 2^{2} + 0 \times 2^{1} + 1 \times 2^{0} + 1 \times 2^{-1} = 101001.1$

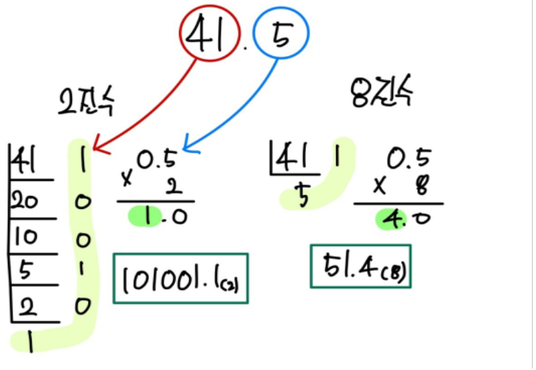
8진수 : $5 \times 8^{1} + 1 \times 8^{0} + 4 \times 8^{-1} = 51.4$

16진수 : $2 \times 16^{1} + 9 \times 16^{0} + 8 \times 16^{-1} = 29.8$

소수가 포함되어 있는 경우에는
정수부와 소수부를 나눠 계산한다.

Tip : 소수가 있는 10진수를 n진수로 바꾸기

1. 소수가 있는 10진수에 'x n'를 한다
2. 소수 부분이 0이 될때까지 계속 곱한다.
3. 위에서부터 읽어내린다.



2진화 8진수 Octal

Octal number	Binary-coded octal	Decimal equivalent
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7
10	001 000	8
11	001 001	9
12	001 010	10
13	001 011	11
36	011 110	30
62	110 010	50
143	001 100 011	99
310	011 001 000	200
620	110 010 000	400

2진화 16진수 Hexadecimal

Hexadecimal number	Binary-coded hexadecimal	Decimal equivalent
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1101	14
F	1111	15
F8	1111 1000	248

2진화 10진수 BCD(Binary Code Decimal)

Decimal number	Binary-coded decimal number
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	0001 0000
20	0010 0000
50	0101 0000
99	1001 1001
247	0010 0100 0111
330	0011 0011 0000
1000	0001 0000 0000 0000

컴퓨터는 0과 1의 순서 만 이해
화면에 정확히 표시되어야하는 것을 알기 위해서는
각각의 고유 번호를 각 기호에 지정

BCD 코드

존 비트		숫자 비트			
A	B	8	4	2	1
x	x	x	x	x	x

존 비트 AB의 값
- 00 : 0, 19(1010, 00011001)
- 01 : 문자 AI(00011001)
- 10 : 문자 R(00011001)
- 11 : 문자 S(00101001)

존 비트		숫자 비트				표현 문자	존 비트		숫자 비트				표현 문자	존 비트		숫자 비트				표현 문자	존 비트		숫자 비트				표현 문자
0	0	0	0	0	1	1	0	1	0	0	0	1	A	1	0	0	0	0	1	J							
0	0	0	0	1	0	2	0	1	0	0	1	0	B	1	0	0	0	1	0	K	1	1	0	0	1	0	S
0	0	0	0	1	1	3	0	1	0	0	1	1	C	1	0	0	0	1	1	L	1	1	0	0	1	1	T
0	0	0	1	0	0	4	0	1	0	1	0	0	D	1	0	0	1	0	0	M	1	1	0	1	0	0	U
0	0	0	1	0	1	5	0	1	0	1	0	1	E	1	0	0	1	0	1	N	1	1	0	1	0	1	V
0	0	0	1	1	0	6	0	1	0	1	1	0	F	1	0	0	1	1	0	O	1	1	0	1	1	0	W
0	0	0	1	1	1	7	0	1	0	1	1	1	G	1	0	0	1	1	1	P	1	1	0	1	1	1	X
0	0	1	0	0	0	8	0	1	1	0	0	0	H	1	0	1	0	0	0	Q	1	1	1	0	0	0	Y
0	0	1	0	0	1	9	0	1	1	0	0	1	I	1	0	1	0	0	1	R	1	1	1	0	0	1	Z
0	0	1	0	1	0	0																					

6비트를 사용하며,
상위 2비트의 존비트와 하위 4비트의 숫자 비트로 구성

EBCDIC 코드

존 비트				숫자 비트			
A	B	C	D	8	4	2	1
x	x	x	x	x	x	x	x

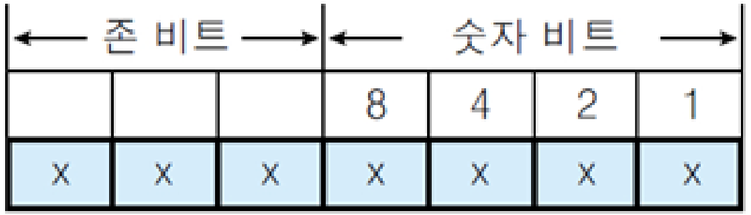
존 비트 AB의 값
- 00 : 여분
- 01 : 특수 문자
- 10 : 영어 소문자
- 11 : 영어 대문자

존 비트 CD의 값
- 00 : 문자 AI(00011001)
- 01 : 문자 R(00011001)
- 10 : 문자 S(00101001)
- 11 : 09(00001001)

상위 / 하위	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	NUL	DLE	DS		SP	&	-						{	}	₩(\)	0
0001	SOH	DC1	SOS				/		a	j	~		A	J		1
0010	STX	DC2	FS	SYN					b	k	s		B	K	S	2
0011	ETX	TM							c	l	t		C	L	T	3
0100	PF	RES	BYP	PN					d	m	u		D	M	U	4
0101	HT	NL	LF	RS					e	n	v		E	N	V	5
0110	LC	BS	ETB	UC					f	o	w		F	O	W	6
0111	DEL	IL	ESC	EOT					g	p	x		G	P	X	7
1000	GE	CAN							h	q	y		H	Q	Y	8
1001	RLF	EM							i	r	z		I	R	Z	9
1010	SMM	CC	SM		¢	!		:								
1011	VT	CU1	CU2	CU3	.	$	,	#								
1100	FF	IFS		DC4	<	*	%	@								
1101	CR	IGS	ENQ	NAK	(	)	_	'								
1110	SO	IRS	ACK		+	;	>	=								
1111	SI	IUS	BEL	SUB		¬	?	"								

8비트를 사용하며,
상위 4비트의 존비트와 하위 4비트의 숫자 비트로 구성

ASCII 코드의 구성



미국 표준 부호 체계로 영문자, 숫자, 특수문자에 고유한 숫자를 부여한 것

7비트를 사용하며 상위 3비트의 존비트와 하위 4비트의 숫자 비트로 구성

행은 숫자비트, 열은 존비트

장점 : 모든 기호들이 할당되어 있는 부호 체계이며, 매우 단순

단점 : 2바이트 이상의 코드를 표현할 수 없음

하위 \ 상위	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	END	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	W (\)	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

UNICODE

16비트(2바이트)를 사용하여 세계 모든 언어의 문자와 그 밖의 기호에까지 코드값을 부여한 것

언어와 상관없이 모든 문자를 16비트로 표현

최대 65,536자를 표한 할 수 있음
38,885자는 주요 국가 언어
6400은 사용자 정의 영역
2만여자는 새로 추가될 언어 영역

장점 : 멀티바이트라는 가변적 공간을 이용하여 모든 언어 표현 가능

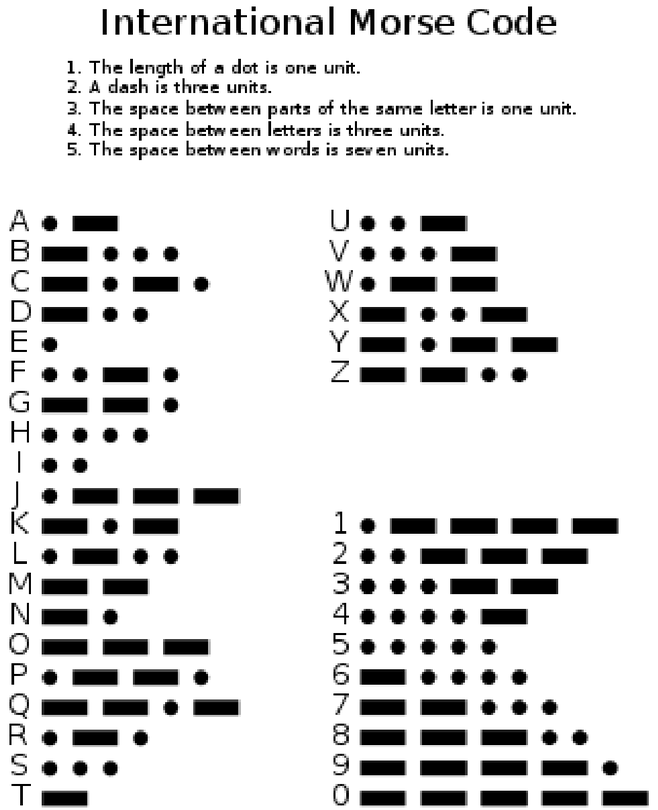
단점 : 인코딩 종류에 따라 멀티바이트 공간이 달라지는데, 언어에 적합하지 않은 인코딩을 사용하면 공간이 낭비될 수 있다.

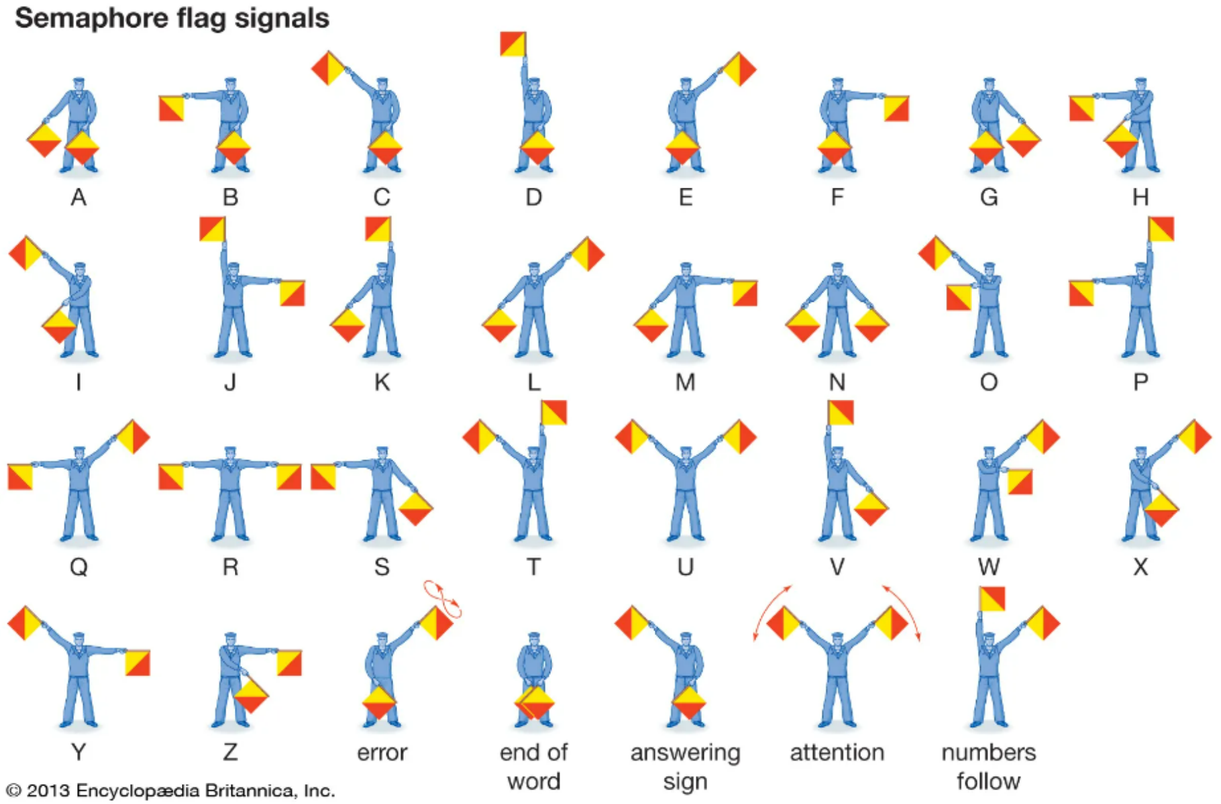
멀티바이트 : ASCII 코드에 포함되지 않는 문자들을 2byte로 표현하는 것

AC00 Hangul Syllables

	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7	AC8	AC9
0	가 AC00	감 AC10	갠 AC20	갰 AC30	걀 AC40	걬 AC50	걠 AC60	거 AC70	검 AC80	겐 AC90
1	각 AC01	갑 AC11	갡 AC21	갱 AC31	걁 AC41	걑 AC51	걡 AC61	걱 AC71	겁 AC81	겑 AC91
2	갂 AC02	값 AC12	갢 AC22	갲 AC32	걂 AC42	걒 AC52	걢 AC62	걲 AC72	겂 AC82	겒 AC92
3	갃 AC03	갓 AC13	갣 AC23	갳 AC33	걃 AC43	걓 AC53	걣 AC63	걳 AC73	것 AC83	겓 AC93
4	간 AC04	갔 AC14	갤 AC24	갴 AC34	걄 AC44	개 AC54	갬 AC64	건 AC74	겄 AC84	겔 AC94
5	갅 AC05	강 AC15	갥 AC25	갵 AC35	걅 AC45	객 AC55	갭 AC65	걵 AC75	경 AC85	겕 AC95
6	갆 AC06	갖 AC16	갦 AC26	갶 AC36	걆 AC46	갞 AC56	갮 AC66	걶 AC76	겆 AC86	겖 AC96
7	갇	갗	갧	갷	걇	갟	갯	걷	겇	겗

UTF-8은 매우 일반적인 인코딩 방식이지만 3bytes 이상의 문자를 사용할 경우에는 비효율적일 수 있다.
UTF-16은 16bit 기반으로 저장하는 UTF-8의 변형이라고 보면 된다.
UTF-32는 모든 문자를 4bytes로 인코딩한다. 문자 변환 알고리즘이나 가변길이 인코딩 방식에 대한 고민을 하고 싶지 않을 때 유용할 수 있다. 그러나 매우 비효율적으로 메모리를 사용하므로 자주 사용되지는 않는다.


International Morse Code
1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
1
2
3
4
5
6
7
8
9
0


Semaphore flag signals
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
error
end of word
answering sign
attention
numbers follow
© 2013 Encyclopædia Britannica, Inc.

컴퓨터는 연산을 진행할 때 두자기 특징을 가진다.

1. 2진수 사용
2. 가산기 사용(더한다)

곱하기는 더하기를 여러 번 반복
그렇다면 빼기는?

컴퓨터에서는 보수를 이용해 뺄셈을 덧셈으로 해결한다!
*** 나누기는 빼기의 연속**

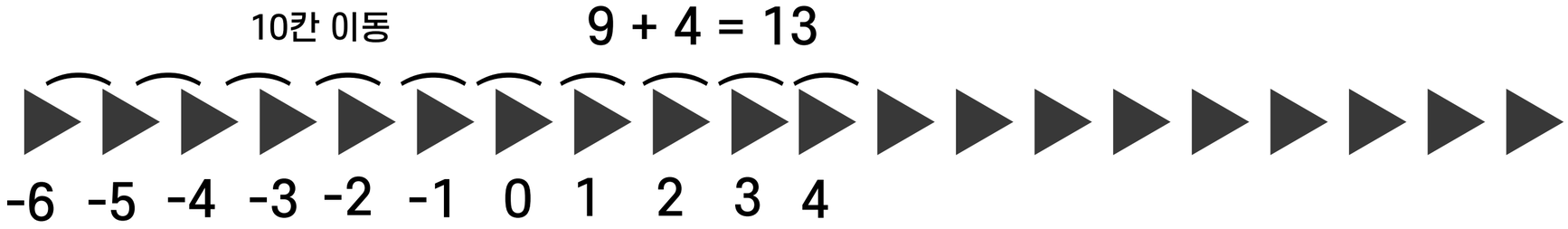
Ex) 9에서 6을 빼보자

암산 : 9 - 6 =3

가산기

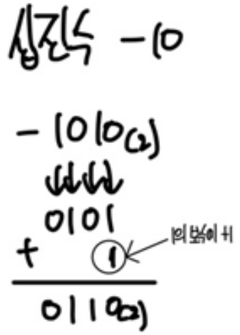
x = -6에 대한 보수

9 + 4 = 13

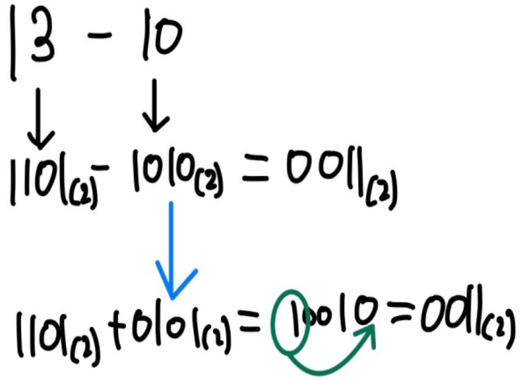


보수의 정의

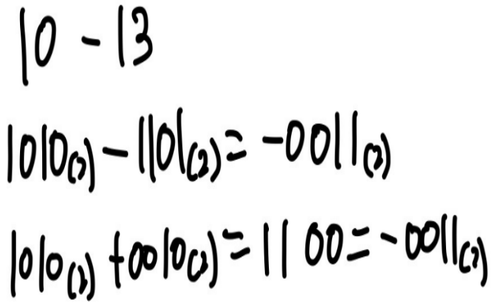
보충을 해주는 수
진법의 기수(radix)에 대응하는 역(reverse) 값
가산기를 이용해 음수 연산을 하기 위한 장치
즉, 뺄셈을 덧셈으로 하기 위해서 만들어 놓은 장치
ex) 1에 대한 10의 보수는 9, 4에 대한 15의 보수는 11, 2에 대한 1의 보수는 1



1의보수 (4비트)

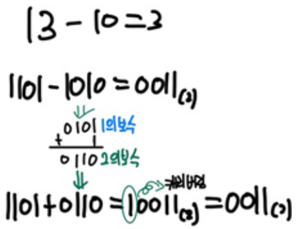


마지막에 생긴 최상위 비트(캐리비트)를
최하위 비트에 더해줌

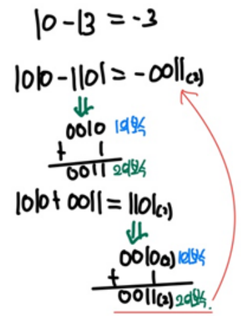


캐리비트가 발생하지 않았을 때는
1의 보수를 한 번 더 취하고 -부호를 더해줌

2의보수 (4비트)



마지막에 생긴 최상위 비트(캐리비트) 버림



캐리비트가 발생하지 않았을 때는
2의 보수를 한 번 더 취하고 -부호를 더해줌

Q & A