4 Data Representation

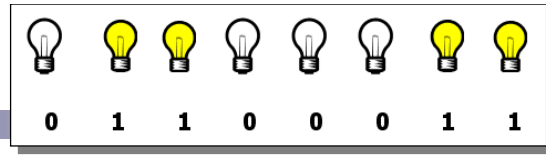
Fundamental of Computer Science

CS-ICT-GIS
Faculty of Informatics
MSU

Outline

- หน่วยทางคอมพิวเตอร์
- ระบบเลขฐานที่ใช้ในคอมพิวเตอร์
- การแทนค่าข้อมูลตัวอักษร
- การแทนค่าข้อมูลตัวเลขจำนวนเต็ม
- การแทนค่าข้อมูลตัวเลขจำนวนจริง

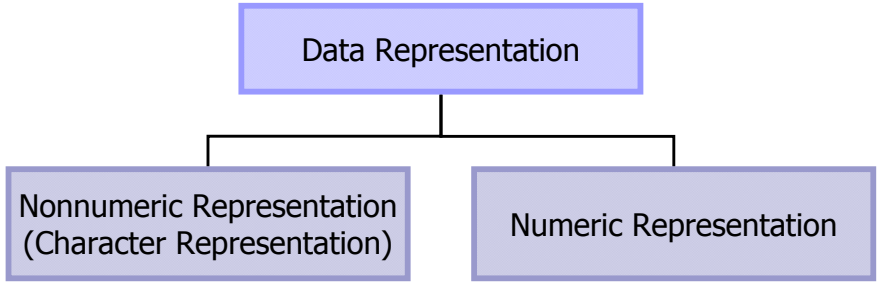
Bit & Byte



- Bit (**BI**nary Digi**T**)
  - หน่วยที่เล็กที่สุดที่ใช้ในการเก็บข้อมูลในคอมพิวเตอร์
  - แทนสถานะอย่างใดอย่างหนึ่งของสวิตช์ ("เปิด" หรือ "ปิด")
- ไบต์ (Byte)
  - กลุ่มของบิตที่มีความหมายเฉพาะ
  - 1 ไบต์ = 8 บิต
  - แสดงสัญลักษณ์ที่แตกต่างกันได้ถึง 256 รหัส
  - 1 KiloByte(KB) = 1024 Bytes = 2^{10} Bytes = ? Bits
  - 1 MegaByte(MB) = ? KB = ? Bytes = ? Bits
  - 1 GigaByte(GB) = ? MB = ? KB = ? Bytes = ? Bits

การแทนค่าข้อมูล (Data Representation)

- รหัสที่ใช้แทนตัวเลข ตัวอักษร สัญลักษณ์ต่างๆ ที่ประกอบอยู่ในคำสั่ง และข้อมูล เพื่อใช้ในการประมวลผล



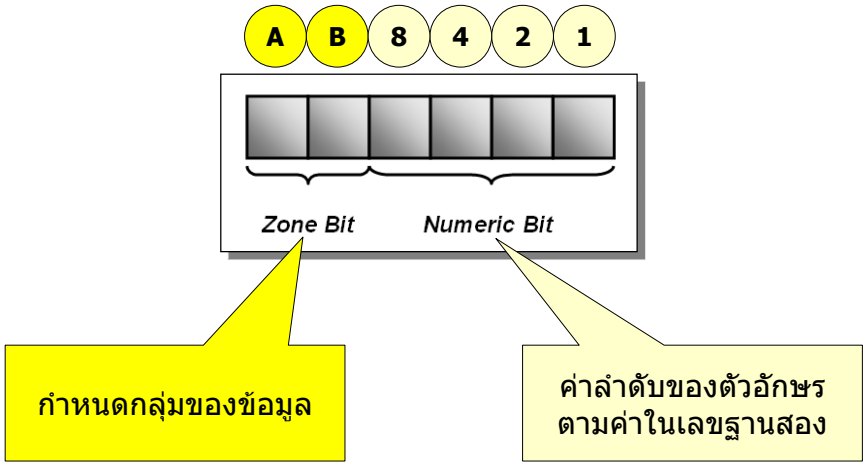
Character Representation

- BCD
- EBCDIC
- ASCII
- Unicode

BCD - Binary Code Decimal

- กำหนดขึ้นโดย IBM เพื่อใช้ในคอมพิวเตอร์ IBM ในยุคแรก ๆ (1959-1965)
- 6 บิต แทนข้อมูล 1 อักขระ (Character)
- สร้างรหัสได้ $2^6 = 64$ รหัส
  - อักขระตัวพิมพ์ใหญ่ 26 ตัว
  - ตัวเลข 10 ตัว
  - สัญลักษณ์พิเศษ 28 ตัว

Standard BCD



BCD – Number

Zone Bit
Zone A จะเป็น 0
Zone B จะเป็น 0

Numeric Bit
ค่าของตัวเลขนั้น ในระบบเลขฐานสอง

EX

0 = 00 **0000**
1 = 00 **0001**
2 = 00 **0010**
3 = 00 **0011**
4 = 00 **0100**
5 = 00 **0101**
6 = 00 **0110**
7 = 00 **0111**
8 = 00 **1000**
9 = 00 **1001**

http://www.nrru.ac.th/learning/science/sc_007/03/unit3/data2.html

BCD – Character

Zone Bit

A - I Zone A จะเป็น 1
Zone B จะเป็น 1

J - R Zone A จะเป็น 1
Zone B จะเป็น 0

S - Z Zone A จะเป็น 0
Zone B จะเป็น 1

Numeric Bit

ค่าลำดับของตัวอักษรนั้น ในระบบเลขฐานสอง

B = 11 0010

(B อยู่ในข้อมูลชุดที่ 1 ลำดับที่ 2)

M = 10 0100

(M อยู่ในข้อมูลชุดที่ 2 ลำดับที่ 4)

X = 01 0110

(X อยู่ในข้อมูลชุดที่ 3 ลำดับที่ 6)

BCD – Special Character

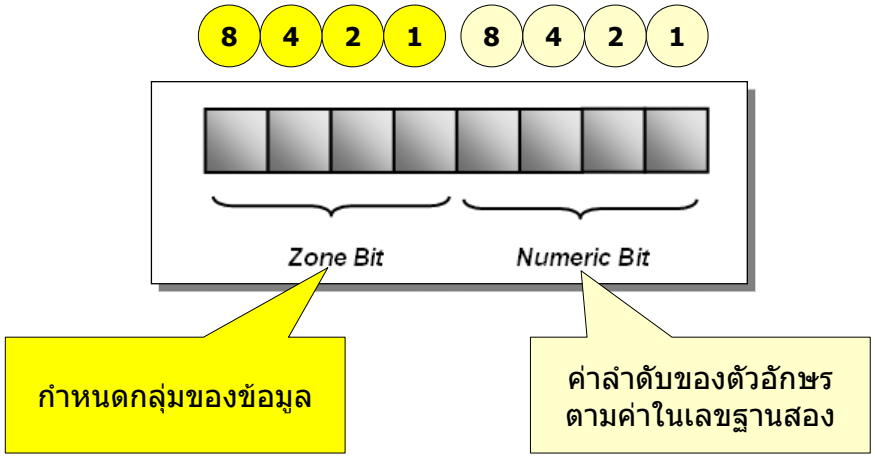
บริษัทผู้ผลิตเครื่องคอมพิวเตอร์จะเป็นผู้กำหนดขึ้นมาเอง

(= 11 1101
) = 10 1101
*** = 10 1100**
/ = 01 0001
+ = 01 1010
- = 10 0000

EBCDIC : Extended Binary Coded Decimal Interchange Code

- **IBM พัฒนารหัส EBCDIC ขึ้นใช้กับเครื่อง IBM System/360 (1965)**
- **8 บิต แทนข้อมูล 1 อักขระ (Character)**
- **สร้างรหัสได้ 2^8 = 256 รหัส (ใช้เลขฐาน 16 แสดงรหัสข้อมูล)**

EBCDIC



EBCDIC – Number

Zone Bit

- จะเป็น $(1111)_2$

Numeric Bit

- ค่าของตัวเลขนั้น ในระบบเลขฐานสอง

EX

0 = 1111 0000
1 = 1111 0001
2 = 1111 0010
3 = 1111 0011
4 = 1111 0100

5 = 1111 0101
6 = 1111 0110
7 = 1111 0111
8 = 1111 1000
9 = 1111 1001

http://www.nrru.ac.th/learning/science/sc_007/03/unit3/data2.html

EBCDIC – Character

Zone Bit

A - I จะเป็น $(1100)_2$ หรือ $(C)_{16}$
J - R จะเป็น $(1101)_2$ หรือ $(D)_{16}$
S - Z จะเป็น $(1110)_2$ หรือ $(E)_{16}$
a - i จะเป็น $(1000)_2$ หรือ $(8)_{16}$
j - r จะเป็น $(1001)_2$ หรือ $(9)_{16}$
s - z จะเป็น $(1010)_2$ หรือ $(A)_{16}$

Numeric Bit

- ค่าลำดับของตัวอักษรนั้นในระบบเลขฐานสอง

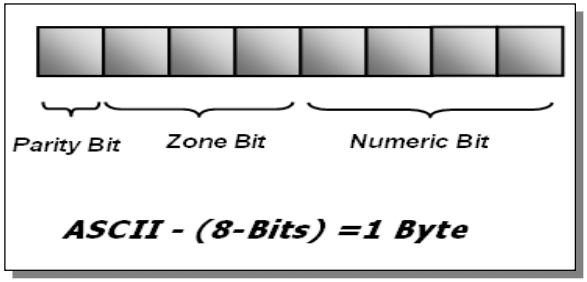
Ex.

B = $(1100\ 0010)_2$ หรือ $(C2)_{16}$
M = $(1101\ 0100)_2$ หรือ $(D4)_{16}$
X = $(1110\ 0110)_2$ หรือ $(E6)_{16}$

http://www.nrru.ac.th/learning/science/sc_007/03/unit3/data2.html

ASCII

- American Standard Code for Information Interchange
- International Alphabet Number 5
- In Trend Code
- พัฒนาโดย American National Standard Institute: ANSI
- ประกอบด้วยเลขฐานสอง 7 บิต (ปัจจุบันใช้ 8 บิต) แทนอักขระ 1 ตัว



ASCII TABLE

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	[illegible]	[illegible]	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	[illegible]	[illegible]	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

ASCII TABLE

"Higher ASCII" : แทนอักขระภาษาต่าง ๆ

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

www.AsciiTable.com

Unicode : Unicode World Wide Character Standard

- ใช้เนื้อที่ 2 ไบต์ หรือ 16 บิต ในการแทนสัญลักษณ์ต่าง ๆ
- สามารถแทนสัญลักษณ์ได้ถึง 2^{16} = 65,536 สัญลักษณ์
- เพียงพอสำหรับภาษาญี่ปุ่น (50,000 ตัวอักษร)

Adobe, hp invent, Apple, SYBASE, Microsoft, ORACLE, IBM, SAP, WWW.SUN.COM Sun microsystems

Error Detection and Error Correction Codes

- ข้อมูลในระบบดิจิตอลที่ผิดพลาดคือ ข้อมูลที่มีตั้งแต่ 1 บิตขึ้นไปที่ไม่ถูกต้อง
  - *"Single Errors"* : ผิด 1 บิต
  - *"Multiple Errors"* : ผิดมากกว่า 1 บิต
- "Simple Parity Code"
  - ใช้สำหรับการตรวจสอบข้อผิดพลาดแบบ "Single Errors"
  - เพิ่มบิตพิเศษเข้าไปในข้อมูล 1 บิต โดยจะเรียกบิตที่เพิ่มเข้าไปนี้ว่า *"Parity Bit"*

Parity Bit	Information Bits

Simple Parity Code

- **Odd-Parity Codes**

  เติม Parity Bit ที่เป็น "1" หรือ "0" เพื่อให้จำนวนบิตของ "1" ของข้อมูลนั้น ๆ เป็นจำนวนคี่ (Odd)
- **Even-Parity Codes**

  จะเติม Parity Bit ที่เป็น "1" หรือ "0" เพื่อให้จำนวนบิตของ "1" ของข้อมูลนั้น ๆ เป็นจำนวนคู่ (Even)

Character	ASCII Code	จำนวนบิต "1"	Odd-Parity Code	Even-Parity Code
A	100 0001	2	1100 0001	0100 0001
B	100 0010	2	1100 0010	0100 0010
C	100 0011	3	0100 0011	1100 0011
0	011 0000	2	1011 0000	0011 0000
9	011 1001	4	1011 1001	0011 1001

Odd-Parity Codes : จำนวนคี่ (Odd)
Even-Parity Codes : จำนวนคู่ (Even)

Example 1

- ให้ส่งคำว่า "MSU" ด้วย Odd Parity
- จากตาราง ASCII ข้างต้นจะได้ว่า

Character	Hex	Bin	จำนวนบิต "1"	Odd-Parity Code
M	4D	100 1101	4	$1100\ 1101_2$ หรือ $(CD)_{16}$
S				
U				

- การส่งข้อความ "MSU" ด้วย Odd Parity คือ $(CD\ _\ _\ _\ _)_{16}$

Example 2

- ข้อมูลชุดหนึ่งถูกส่งด้วย Odd Parity ดังนี้
- $C8\ E5\ EC\ EC\ EF_{16}$
- ให้หาว่าผู้ส่งข้อมูล ส่งข้อความใดถึงผู้รับ

$C8_{16}$	$= 1100\ \ 1000_2$	ตัดบิตซ้ายสุดที่เป็น Parity Bit ออก จะได้	$100\ \ 1000_2 = 48_{16}$	→ H
$E5_{16}$	$= 1110\ \ 0101_2$	ตัดบิตซ้ายสุดที่เป็น Parity Bit ออก จะได้		
EC_{16} } EC_{16}				
EF_{16}				

Hello

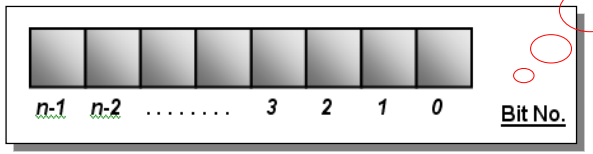
Numeric Representation

- Fixed Point Representation (Integer Representation)
  - Unsigned Integer
  - Sign-Magnitude
  - Complement
    - One's Complement
    - Two's Complement
- Floating Point Representation
  - IEEE 754 Standard

Unsigned Integer Representation

- แทนค่าตัวเลขได้โดยใช้เลขฐานสองโดยตรง

0	0	0	0	0	0	0	0	=	0
0	0	0	0	0	0	0	1	=	1
0	0	1	0	1	0	0	1	=	41
1	0	0	0	0	0	0	0	=	128
1	1	1	1	1	1	1	1	=	255



เลขฐานสองตัวเลขเยอะ ให้เว้นวรรคทุกๆ 4 ตัวด้วยจะได้ดูง่าย ทำข้อสอบก็ไม่ผิด

ถ้าจำนวนบิตข้อมูลที่ใช้ในการแทนค่าเป็น n บิต ค่าที่แทนได้จะอยู่ในช่วงตั้งแต่ 0, 1, 2, ..., (2^n - 1)

ตัวอย่างการแทนค่า Unsigned Integer ขนาด 4 บิต

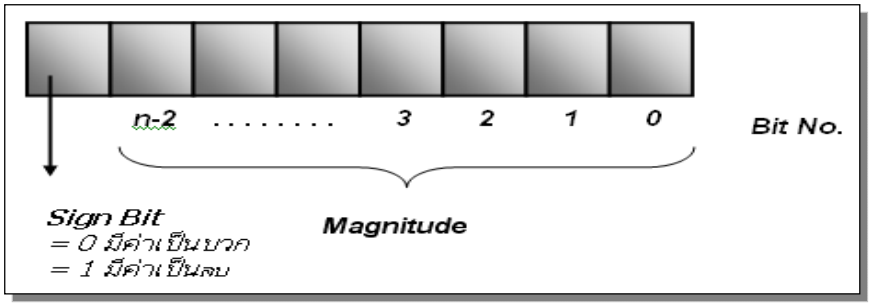
Dec	Hex	Unsigned Integer
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

- ข้อมูลขนาด n = 4 บิต
- จำนวนข้อมูล $2^n = 2^4 = 16$ จำนวน
- ช่วงข้อมูล 0, 1, 2, ..., 15

ตารางนี้ทุกคนควรต้องจำให้ได้นะ จะทำให้แปลงเลขฐาน 2 กับ 16 ได้คล่อง

Sign-Magnitude Representation

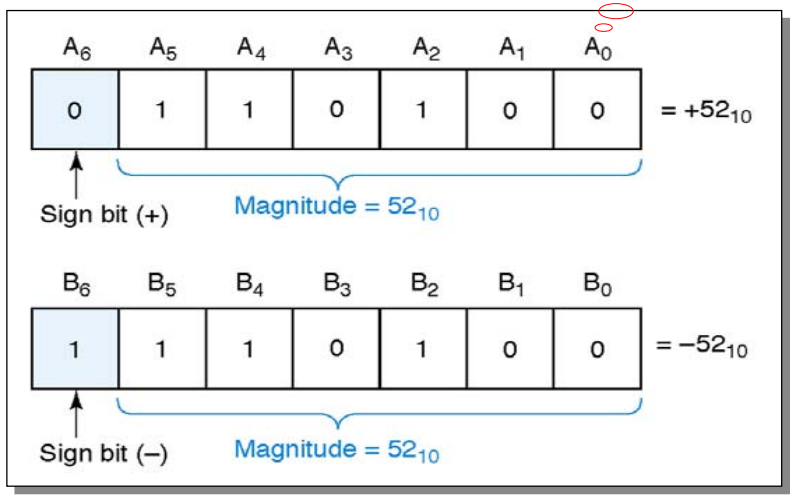
- แทนค่าเลขจำนวนเต็มแบบมีเครื่องหมาย (Sign Integers)



- จำนวนบิตที่ใช้ในการแทนค่าเป็น n บิต
- จำนวนค่ารหัสตัวแทนทั้งหมด 2^n
- แบ่งเป็นรหัสตัวแทนจำนวนเลขบวกและเลขลบอย่างละ 2^{n-1}
- ค่าที่สามารถแทนได้จะอยู่ในช่วง ตั้งแต่

– (2^{n-1} – 1), ..., -1, -0, 0, 1, ..., + (2^{n-1} – 1)

เลขคู่ LSB ต้องเป็น 0 เสมอ



ปัญหา Sign Magnitude

- **มีตัวแทนของ 0 สองค่า**

$+0_{10}$	= 0000 0000	
-0_{10}	= 1000 0000	(Sign Magnitude)

- **บวก-ลบเลขผิดพลาด (กรณีตัวตั้งเป็นเลขลบ)**
  - ตัวอย่าง ให้ X = 2 และ Y = 5 และมีจำนวนบิต 4 บิต

+X+Y	+X-Y	-X+Y	-X-Y
↓	↓	↓	↓
(+X)+(+Y)	(+X)+(-Y)	(-X)+(+Y)	(-X)+(-Y)
$+2 \rightarrow 0010_2$	$+2 \rightarrow 0010_2$	$-2 \rightarrow 1010_2$	$-2 \rightarrow 1010_2$
$+5 \rightarrow 0101_2$	$-5 \rightarrow 1101_2$	$+5 \rightarrow 0101_2$	$-5 \rightarrow 1101_2$
$+7 \rightarrow 0111_2$	$-3 \rightarrow 1111_2$ ✗	$+3 \rightarrow 1111_2$ ✗	$-7 \rightarrow 0111_2$ ✗
	1011_2	0011_2	1111_2

Complement Representation

- One's Complement Representation
- Two's Complement Representation

One's Complement Representation

- ตัวเลขบวก : ใช้หลักการเดียวกับ Sign Magnitude
- ตัวเลขลบ : ใช้วิธีการแทนค่าแต่ละบิต $(N)_2$ ด้วยค่าตรงข้าม (Complement) ของบิตนั้น คือ
  กลับบิต 1 เป็น 0 และจาก 0 เป็น 1

- จำนวนบิตที่ใช้ในการแทนค่าเป็น n บิต
- จำนวนค่ารหัสตัวแทนทั้งหมด 2^n
- แบ่งเป็นรหัสตัวแทนจำนวนเลขบวกและเลขลบอย่างละ 2^{n-1}
- ค่าที่สามารถแทนได้จะอยู่ในช่วง ตั้งแต่
  $-(2^{n-1}-1), \ldots, -1, -0, 0, 1, \ldots, 2^{n-1}-1$

One's Complement Representation

0	0	0	1	0	0	1	0	= +18
0	1	0	0	0	0	0	1	= +65

↑ Sign Bit

0 0 1 0 0 1 0	= 18	
$(0\ 0\ 1\ 0\ 0\ 1\ 0)_{1cns}$	One's Complement	
1 1 1 0 1 1 0 1	= -18	
1 0 0 0 0 0 1	= 65	
$(1\ 0\ 0\ 0\ 0\ 0\ 1)_{1cns}$	One's Complement	
1 0 1 1 1 1 1 0	= -65	

Sign Bit

หาค่าตัวแทนจากรหัสฐานสองเป็นรหัสฐานสิบ

- หาค่าตัวแทนของ $(0100\ 1010)_2$
  - บิตเครื่องหมายเป็น +
  - ให้นำบิตที่เหลือมาหาค่าด้วยวิธีการแปลงเลขฐานสอง
  - $+(100\ 1010)_2 = +(2^6 + 2^3 + 2^1) = +(74)_{10}$
- หาค่าตัวแทนของเลข $(1100\ 1010)_2$
  - บิตเครื่องหมายเป็น –
  - ให้นำบิตที่เหลือมาหาค่าด้วยวิธี One's Complement
  - $-(100\ 1010)_2 = -(011\ 0101)_2$
  - นำผลที่ได้มาหาค่าเลขลบด้วยการแปลงเลขฐานสอง
  - $-(011\ 0101)_2 = -(2^5+2^4+2^2+2^0) = -(53)_{10}$

การบวกและลบของเลข One's Complement

- แทนค่าตัวเลขที่ต้องการคำนวณด้วย 1's Complement
- หาผลบวกของ 1's Complement ที่ได้จากข้อ 1
- ถ้ามีตัวทดสุดท้าย (End Round Carry) ให้นำตัวทดนั้นไปบวกกับหลักขวาสุด เพื่อชดเชยผลลัพธ์ความผิดพลาดของผลลัพธ์ขั้นต้น
- พิจารณาผลลัพธ์ขั้นสุดท้าย โดยพิจารณาบิตซ้ายสุด (Sign Bit)
  - ถ้าบิตซ้ายสุดเป็น ⓪ แสดงว่า ผลลัพธ์เป็นบวก
  - ถ้าบิตซ้ายสุดเป็น ① แสดงว่า ผลลัพธ์เป็นลบ ให้ทำ One's Complement กับบิตที่เหลือ

ตัวอย่างที่ 6 ให้หาผลลัพธ์ของ 7 + 5 กำหนดให้แทนรหัสข้อมูลแบบ One's Complement ขนาด 8 บิต

(+7)	แทนด้วย	0000 0111	+
(+5)	แทนด้วย	0000 0101	
(+7) + (+5)	ได้ผลลัพธ์คือ	0000 1100	ไม่มีตัวทด และเป็นค่าบวก

ดังนั้น ผลลัพธ์ของเลขจำนวนข้างต้นคือ $+(000\ 1100)_2$ = $+(12)_{10}$

ตัวอย่างที่ 7 ให้หาผลลัพธ์ของ 7 – 5 กำหนดให้แทนรหัสข้อมูลแบบ One's Complement ขนาด 8 บิต

7 – 5	= (+7) + (-5)		
(+7)	แทนด้วย	0000 0111	+
(-5)	แทนด้วย	1111 1010	
(+7) + (-5)	ได้ผลลัพธ์คือ	❶ 0000 0001	มีตัวทด
นำตัวทดที่ได้ไปบวกกับผลลัพธ์ได้		0000 0010	เป็นค่าบวก

ดังนั้น ผลลัพธ์ของเลขจำนวนข้างต้นคือ $+(000\ 0010)_2$ = $+(2)_{10}$

ตัวอย่างที่ 8 ให้หาผลลัพธ์ของ -7 + 5 กำหนดให้แทนรหัสข้อมูลแบบ One's Complement ขนาด 8 บิต

-7 + 5	= (-7) + (+5)		
(-7)	แทนด้วย	1111 1000	+
(+5)	แทนด้วย	0000 0101	
(-7)+(5)	ได้ผลลัพธ์คือ	1111 1101	เป็นค่าลบ

One's Complement ของ $(111\ 1101)_2$ คือ $(000\ 0010)_2$

ดังนั้น ผลลัพธ์ของเลขจำนวนข้างต้นคือ $-(000\ 0010)_2$ = $-(2)_{10}$

ตัวอย่างที่ 9 ให้หาผลลัพธ์ของ -7 – 5 กำหนดให้แทนรหัสข้อมูลแบบ One's Complement ขนาด 8 บิต

-7 – 5	= (-7) + (-5)		
(-7)	แทนด้วย	1111 1000	+
(-5)	แทนด้วย	1111 1010	
(-7) + (-5)	ได้ผลลัพธ์คือ	❶ 1111 0010	มีตัวทด
นำตัวทดที่ได้ไปบวกกับผลลัพธ์ได้		1111 0011	เป็นค่าลบ

One's Complement ของ $(111\ 0011)_2$ คือ $(000\ 1100)_2$

ดังนั้น ผลลัพธ์ของเลขจำนวนข้างต้นคือ $-(000\ 1100)_2$ = $-(12)_{10}$

ปัญหา One's Complement

- แก้ปัญหาการบวก-ลบเลขผิดพลาด กรณีตัวตั้งเป็นลบได้
- ยังคงมีตัวแทน 0 สองค่า
  - $+0_{10} = 0000\ 0000_2$
  - $-0_{10} = 1111\ 1111_2$

ดังนั้นช่วงข้อมูลจะอยู่ในช่วงตั้งแต่
$-(2^{n-1} - 1), \ldots, 1, -0, 0, 1, \ldots, 2^{n-1} - 1$

Two's Complement Representation

- จัดเก็บเหมือน One's Complement
- ตัวเลขบวก : ใช้หลักการเดียวกับ Sign Magnitude
- ตัวเลขลบ : ให้ดำเนินการดังนี้
  - ทำ One's Complement
  - นำ 1 มาบวกกับค่า One's Complement ที่ได้

- จำนวนบิตที่ใช้ในการแทนค่าเป็น n บิต
- จำนวนค่ารหัสตัวแทนทั้งหมด 2^n
- แบ่งเป็นรหัสตัวแทนจำนวนเลขบวกและเลขลบอย่างละ 2^{n-1}
- ค่าที่สามารถแทนได้จะอยู่ในช่วง ตั้งแต่
$-(2^{n-1}), \ldots, -1, -0, 0, 1, \ldots, 2^{n-1} - 1$

Two's Complement Representation

```
0 0 0 1   0 0 1 0  =  +18
0 1 0 0   0 0 0 1  =  +65
Sign Bit
```

```
 0 0 1    0 0 1 0    = 18
(0 0 1    0 0 1 0)1cns
 1 1 0    1 1 0 1    One's complement
(0 0 0    0 0 0 1)   (+1)2
(1 1 0    1 1 1 0)   Two's Complement
1 1 1 0   1 1 1 0    = -18

 1 0 0    0 0 0 1    = 65
(1 0 0    0 0 0 1)1cns
 0 1 1    1 1 1 0    One's complement
(0 0 0    0 0 0 1)   (+1)2
(0 1 1    1 1 1 1)   Two's Complement
1 0 1 1   1 1 1 1    = -65
```


การบวกและลบของเลข Two's Complement

- แทนค่าตัวเลขที่ต้องการคำนวณด้วย 2's Complement
- หาผลบวกของ 2's Complement ที่ได้จากข้อ 1
- ถ้ามีตัวทดสุดท้าย (End Round Carry) ให้ตัดตัวทดนั้นทิ้งไป
- พิจารณาผลลัพธ์ขั้นสุดท้าย โดยพิจารณาบิตซ้ายสุด (Sign Bit)
- ถ้าบิตซ้ายสุดเป็น ⓪ แสดงว่า ผลลัพธ์เป็นบวก
- ถ้าบิตซ้ายสุดเป็น ① แสดงว่า ผลลัพธ์เป็นลบ ให้ทำ 2's Complement บิตที่เหลือ

A+B

ตัวอย่างที่ 12 ให้หาผลลัพธ์ของ 7 + 5 กำหนดให้แทนรหัสข้อมูลแบบ Two's Complement ขนาด 8 บิต

(+7)	แทนด้วย	0000 0111	+
(+5)	แทนด้วย	0000 0101	
(+7) + (+5)	ได้ผลลัพธ์คือ	0000 1100	ไม่มีตัวทด และเป็นค่าบวก

ดังนั้น ผลลัพธ์ของเลขจำนวนข้างต้นคือ $+(000\ 1100)_2$ = $+(12)_{10}$

A-B=A+(-B)

ตัวอย่างที่ 13 ให้หาผลลัพธ์ของ 7 – 5 กำหนดให้แทนรหัสข้อมูลแบบ Two's Complement ขนาด 8 บิต

7 – 5	= (+7) + (-5)		
(+7)	แทนด้วย	0000 0111	+
(-5)	แทนด้วย	1111 1010	
(+7) + (-5)	ได้ผลลัพธ์คือ	❶ 0000 0001	ตัดตัวทดทิ้ง
นำตัวทดที่ได้ไปบวกกับผลลัพธ์ได้		0000 0010	เป็นค่าบวก

ดังนั้น ผลลัพธ์ของเลขจำนวนข้างต้นคือ $+(000\ 0010)_2$ = $+(2)_{10}$

-A+B=(-A)+B

ตัวอย่างที่ 14 ให้หาผลลัพธ์ของ -7 + 5 กำหนดให้แทนรหัสข้อมูลแบบ Two's Complement ขนาด 8 บิต

-7 + 5	= (-7) + (+5)		
(-7)	แทนด้วย	1111 1001	+
(+5)	แทนด้วย	0000 0101	
(-7)+(5)	ได้ผลลัพธ์คือ	1111 1110	ไม่มีตัวทด และเป็นค่าลบ

Two's Complement ของ $(111\ 1110)_2$ คือ $(000\ 0010)_2$

ดังนั้น ผลลัพธ์ของเลขจำนวนข้างต้นคือ $-(000\ 0010)_2$ = $-(2)_{10}$

-A-B=(-A)+(-B)

ตัวอย่างที่ 15 ให้หาผลลัพธ์ของ -7 – 5 กำหนดให้แทนรหัสข้อมูลแบบ Two's Complement ขนาด 8 บิต

-7 – 5	= (-7) + (-5)		
(-7)	แทนด้วย	1111 1001	+
(-5)	แทนด้วย	1111 1011	
(-7) + (-5)	ได้ผลลัพธ์คือ	❶ 1111 0100	ตัดตัวทดทิ้ง และเป็นค่าลบ

Two's Complement ของ $(111\ 0100)_2$ คือ $(000\ 1100)_2$

ดังนั้น ผลลัพธ์ของเลขจำนวนข้างต้นคือ $-(000\ 1100)_2$ = $-(12)_{10}$

ปัญหา Two's Complement

- สามารถแก้ปัญหาการบวก-ลบเลขได้
- แก้ปัญหาการแทนค่า 0
  - $+0_{10}$ = $0000\ 0000_2$
  - -0_{10} = $1111\ 1111$ One's complement

    $0000\ 0001_2$

    $\cancel{1}\ 0000\ 0000_2$

Example : การหาช่วงข้อมูลสำหรับการแทนค่าตัวเลขจำนวนเต็มแบบมีเครื่องหมาย

- ข้อมูลขนาด 4 บิต
- จำนวนข้อมูล = $2^n = 2^4 = 16$ จำนวน
- จำนวนเลขบวก = จำนวนเลขลบ = $2^{n-1} = 2^3$
- Sign Magnitude และ One's Complement มีช่วงข้อมูลคือ
  -7, -6, …, -1, -0, 0, 1, …, 6, 7
- Two's Complement มีช่วงข้อมูลคือ
  -8, -7, …, -1, 0, 1, …, 6, 7

Example : การแทนค่าจำนวนเต็มแบบมีเครื่องหมาย ขนาด 4 บิต

Decimal	Sign Magnitude	1's Complement	Decimal	2's Complement
-7	1111	1000	-8	1000
-6	1110	1001	-7	1001
-5	1101	1010	-6	1010
-4	1100	1011	-5	1011
-3	1011	1100	-4	1100
-2	1010	1101	-3	1101
-1	1001	1110	-2	1110
-0	1000	1111	-1	1111
0	0000	0000	0	0000
1	0001	0001	1	0001
2	0010	0010	2	0010
3	0011	0011	3	0011
4	0100	0100	4	0100
5	0101	0101	5	0101
6	0110	0110	6	0110
7	0111	0111	7	0111

Exercise 1

- ถ้ากำหนดขนาดข้อมูลมีขนาด 8 บิต จงหา
  - จำนวนค่ารหัสตัวแทนของข้อมูลขนาด 8 บิต เมื่อมีการแทนค่าสำหรับตัวเลขจำนวนเต็ม
  - จำนวนค่ารหัสตัวแทนของข้อมูลตัวเลขจำนวนเต็มแบบมีเครื่องหมาย
    - เลขลบ
    - เลขบวก
  - ช่วงของข้อมูล
    - Unsigned Integer
    - Sign Magnitude
    - One's Complement
    - Two's Complement
- จงหาค่าตัวเลขเมื่อรหัสที่ถูกแทนค่าเป็น

  $(1111\ 0101)_2 = (?)_{10}$

  - เมื่อทำการแทนค่าด้วย
    - Unsigned Integer
    - Sign Magnitude
    - One's Complement
    - Two's Complement

Exercise 2

- จงหาค่ารหัสตัวแทน โดยจัดเก็บข้อมูลขนาด 8 บิต เมื่อมีการจัดเก็บข้อมูลตัวเลขต่างๆ ดังนี้
  - $(250)_{10} = (?)_2$

    เมื่อจัดเก็บด้วย Unsigned Integer
  - $(120)_{10} = (?)_2$

    เมื่อจัดเก็บด้วย
    - Sign Magnitude
    - One's Complement
    - Two's Complement
  - $(-120)_{10} = (?)_2$

    เมื่อจัดเก็บด้วย
    - Sign Magnitude
    - One's Complement
    - Two's Complement

Exercise 3

- ถ้ากำหนดขนาดข้อมูลให้มีขนาด 5 บิต จงหาว่าค่าผลลัพธ์ที่ได้ต่อไปนี้เกิด Overflow หรือไม่ ถ้าเกิดตัวเลขที่ได้ตรงกับเลขอะไรในฐานสิบ
  - $(01101)_2+(10100)_2$
    - เมื่อจัดเก็บแบบ Unsigned Integer
  - $(10111)_2+(01001)_2$
    - เมื่อจัดเก็บแบบ Sign Magnitude
  - $(10111)_2+(10110)_2$
    - เมื่อจัดเก็บแบบ One's Complement
    - เมื่อจัดเก็บแบบ Two's Complement

Overflow

64+65=129

ตัวอย่างที่ 16 ให้หาผลลัพธ์ของ 64+65 กำหนดให้แทนรหัสข้อมูลแบบ Two's Complement ขนาด 8 บิต

(+64)	แทนด้วย	0100 0000	+
(+65)	แทนด้วย	0100 0001	
(+64) + (+65)	ได้ผลลัพธ์คือ	1000 0001	ไม่มีตัวทด และเป็นค่าลบ

Two's Complement ของ $(000\ 0001)_2$ คือ $(111\ 1111)_2$

ผลลัพธ์ของเลขจำนวนข้างต้นคือ $-(111\ 1111)_2$ เป็น**คำตอบที่ผิด**

-64-65=-129

ตัวอย่างที่ 17 ให้หาผลลัพธ์ของ -64-65 กำหนดให้แทนรหัสข้อมูลแบบ Two's Complement ขนาด 8 บิต

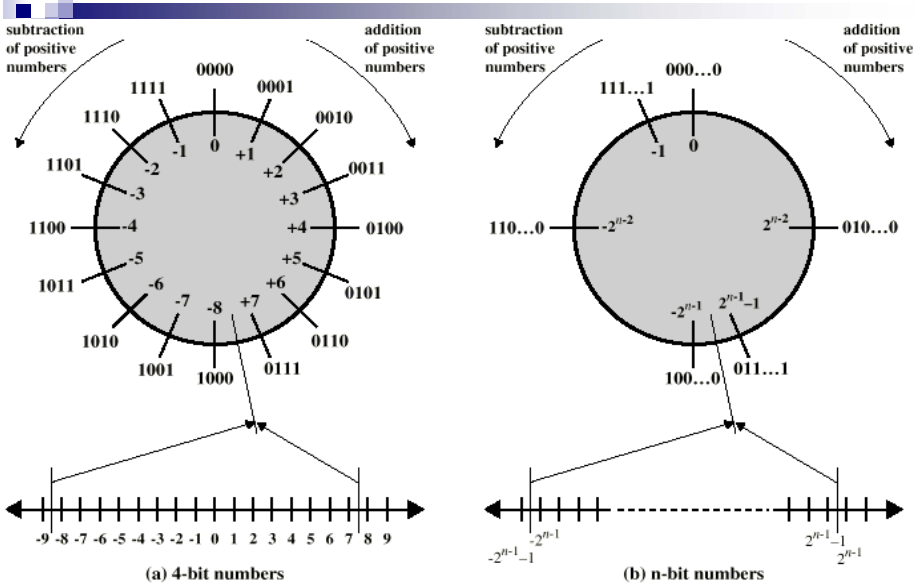
-64-65	= (-64) + (-65)		
(-64)	แทนด้วย	1100 0000	+
(-65)	แทนด้วย	1011 1111	
(-64) + (-65)	ได้ผลลัพธ์คือ	❶ 0111 1111	มีตัวทด และเป็นค่าบวก

ผลลัพธ์ของเลขจำนวนข้างต้นคือ $+(111\ 1111)_2$ เป็น**คำตอบที่ผิด**

Overflow

- เกิดจากผลลัพธ์จากการคำนวณมีขนาดใหญ่เกินกว่าที่คอมพิวเตอร์สามารถจัดเก็บได้
- การพัฒนาโปรแกรมจะต้องมีการตรวจสอบ Overflow เพื่อแจ้งให้ user ทราบถึงความผิดพลาดที่เกิดขึ้น

ภาพแสดงการแทนค่า Two's Complement และการเกิด Overflow



ตัวอย่างช่วงข้อมูล

- เลข n บิต แบบไม่คิดเครื่องหมาย สามารถเก็บข้อมูลได้ตั้งแต่ 0 ถึง $(2^n - 1)$
  - 8 บิต เก็บข้อมูลได้ตั้งแต่ 0 ถึง 255
    - เกิด Overflow ถ้าข้อมูลมีค่าน้อยกว่า 0 และมากกว่า 255
  - 16 บิต เก็บข้อมูลได้ตั้งแต่ 0 ถึง 65,535
    - เกิด Overflow ถ้าข้อมูลมีค่าน้อยกว่า 0 และมากกว่า 255
- เลข n บิต แบบคิดเครื่องหมายสำหรับ Two's Complement สามารถเก็บข้อมูลได้ตั้งแต่ $-(2^{n-1})$ ถึง $+(2^{n-1} - 1)$
  - 8 บิต เก็บข้อมูลได้ตั้งแต่ -128 ถึง +127
    - เกิด Overflow ถ้าข้อมูลมีค่าน้อยกว่า -128 และมากกว่า 127
  - 16 บิต เก็บข้อมูลได้ตั้งแต่ -32,768 ถึง +32,767
    - เกิด Overflow ถ้าข้อมูลมีค่าน้อยกว่า -32,768 และมากกว่า +32,767

Exercise 4

- จงหาผลลัพธ์ต่อไปนี้ว่าเกิด Overflow หรือไม่
  - ถ้าแทนค่าด้วย One's Complement ขนาด 8 บิต
    - -1-127
    - $(0000\ 0010)_2 + (0111\ 1111)_2$
  - ถ้าแทนค่าด้วย Sign Magnitude ขนาด 8 บิต
    - +1+127
    - $(1000\ 0010)_2 + (1111\ 1111)_2$
  - ถ้าแทนค่าด้วย Unsigned Integer ขนาด 8 บิต
    - +2+254
    - $(0000\ 0010)_2 + (1111\ 1110)_2$
  - ถ้าแทนค่าด้วย Unsigned Integer ขนาด 8 บิต
    - -3-126
    - $(0000\ 0111)_2 + (0111\ 1110)_2$

Floating Point Representation

1.234×10^{85}

M → +1.234
B → 10
e → +85

1.234×10^{-85}

M → +1.234
B → 10
e → -85

$$R = \pm M \times B^{\pm e}$$

องค์ประกอบของเลขจำนวนจริง

1. Mantissa หรือ Fraction Part (M)
2. Base Exponent (B) และ
3. Exponent (e)

IEEE 754 Standard

IEEE Short Real: 32 bits	1 bit for the sign, 8 bits for the exponent, and 23 bits for the mantissa. Also called *single precision*.
IEEE Long Real: 64 bits	1 bit for the sign, 11 bits for the exponent, and 52 bits for the mantissa. Also called *double precision*.

Single – precision Floating Point

Double – precision Floating Point

IEEE 754 Standard

Sign → จำนวนบวก : 0
→ จำนวนลบ : 1

Exponent → Short Real : Bias ของ 2^7-1 = 127
→ Long Real : Bias ของ 2^{10}-1 = 1023

Mantissa → ค่านัยสำคัญในรูปแบบ "Normalized Form"

$$\frac{1}{B} \le |M| < 1$$

การแทนค่า Floating Point ด้วย IEEE 754

- แปลงเลขฐานสิบเป็นฐานสอง
  - $10.625 = (1010.101)_2$
- จัด Normalize Form กับเลขฐานสอง โดยให้อยู่ในรูป
  - $\pm(1.ddddd...)_2 \times 2^{\pm e}$
  - เช่น $(1010.101)_2 \times 2^0 \rightarrow (1.010101)_2 \times 2^3$
    $-(0.0001101)_2 \times 2^0 \rightarrow -(1.101)_2 \times 2^{-4}$
- นำค่า e (exponent) มาหาค่า Bias Exponent ในรูปฐานสอง เช่น
  - Short real : Bias = 127 ดังนั้น
    - $(1.010101)_2 \times 2^3$ ซึ่ง e = 3 แล้ว
      Bias Exponent = 127 + 3 = 130 = $(0100\ 0010)_2$
    - $-(1.101)_2 \times 2^{-4}$ ซึ่ง e = - 4 แล้ว
      Bias Exponent = 127 + (-4) = 123 = $(0111\ 1011)_2$

MSB 31	30 … 23	22 … LSB 0
Sign	Exponent	Mantissa

Single – precision Floating Point

Binary Value	Normalize As	Exponent	Biased Exponent (127 + Exponent)
-1.11	-1.11	0	127
1 01111111 11000000000000000000000			
+1101.101	+1.101101	+3	130
0 10000010 10110100000000000000000			
-.00101	-1.01	-3	124
1 01111100 01000000000000000000000			
+100111.0	+1.001110	+5	132
0 10000100 00111000000000000000000			
+.0000001101011	+1.101011	-7	120
0 01111000 10101100000000000000000			

Exercise 4 : Floating Point

ให้แสดงการแทนค่าข้อมูลจำนวนจริงต่อไปนี้
ตามมาตรฐาน **IEEE 754** แบบ **Single precision**

- +1.101
- -11.11
- +1.1
- -101.001
- +1101.0101
- +1110.00111
- -10000.101011
- +111.0000011
- -11.000101

Exercise 5 : Floating Point

ให้แสดงการแทนค่าข้อมูลจำนวนจริงต่อไปนี้
ตามมาตรฐาน **IEEE 754** แบบ **Double precision**

- +1.101
- -11.11
- +1.1
- -101.001
- +1101.0101
- +1110.00111
- -10000.101011
- +111.0000011
- -11.000101

Exercise 6 : Integer

+15	**+100**	**-15**	**-100**
+35	**+120**	**-35**	**-120**
+65	**+130**	**-65**	**-130**
+68	**+255**	**-68**	**-255**
+96	**+256**	**-96**	**-256**

กำหนดให้ใช้ข้อมูลขนาด 8 บิต ในการจัดเก็บข้อมูล ให้แสดงการแทนค่าข้อมูลตัวเลขจำนวนเต็มข้างต้น ตามรูปแบบต่าง ๆ ดังต่อไปนี้

- Unsigned Integer
- Sign-Magnitude
- One's Complement
- Two's Complement