LECTURE 2: DIGITAL LOGIC CIRCUIT DESIGN

By the end of this class you should be able to:

- **Obtain the r's and (r-1)'s complements in decimal, binary, and any other number system**
- **Perform subtraction by addition of the complements**

Decimal number complements:

9's complement of the decimal number N = $(10^n - 1) - N$
$= n\ (9's) - N$
i.e. {subtract each digit from 9}

Example → 9's complement of 134795 is 865204

Similarly

1's complement of the binary number N = $(2^n - 1) - N = n\ (1's) - N$

Example → 1's complement of 110100101 is 001011010 which can be obtained by replacing each one by a zero and each zero by one.

r's complement:

10's complement of the decimal number N = $10^n - N$ = (r-1)'s complement + 1

Example → 10's complement of 134795 is 865205

Example → find the 9's and 10's complements of 314700.

Answer → 9's complement = 685299
10's complement=685300

Rule: To find the 10's complement of a decimal number leave all leading zeros unchanged. Then subtract the first non-zero digit from 10 and all the remaining digits from 9's.

The 2's complement of a binary number is defined in a similar way.

Example: Find the 1's and 2's complements of the binary number 1101001101

Answer → 1's complement is 0010110010
2's complement is 0010110011

Example: Find the 1's and 2's complements of 100010100

Answer → 1's complement is 011101011
2's complement is 011101100

Subtraction using r's complement:

To find M-N in base r, we add M + r's complement of N

Result is $M + (r^n - N)$

1) If M > N then result is M – N + rn (rn is an end carry and can be neglected.

2) If M < N then result is r^n –(N-M) which is the r's complement of the result.

Example: Subtract (76425 – 28321) using 10's complements.

Answer → 10's complement of 28321 is 71679

Then add →
$$\begin{array}{r} 76425 \\ +\ 71679 \\ \hline 148104 \end{array}$$

Discard

Therefore the difference is 48104 after discarding the end carry.

Example: subtract (28531 – 345920)

Answer → It is obvious that the difference is negative. We also have to work with the same number of digits, when dealing with complements.

10's complement of 345920 is 654080

Then add →

```
  0 2 8 5 3 1
+ 6 5 4 0 8 0
-------------
  6 8 2 6 1 1
```

No end carry

Therefore the difference is negative and is equal to the 10's complement of the answer.

Difference is → - 317389

The same rules apply to binary.

Example: subtract (11010011 – 10001100)

Example: Subtract (76425 – 28321) using 9's complements.

Answer → 9's complement of 28321 is 71678

Then add →

```
  7 6 4 2 5
+ 7 1 6 7 8
-----------
1 4 8 1 0 3
      └→ 1
-----------
  4 8 1 0 4
```

Difference

Answer → 2's complement of 10001100 is 01110100

Then add →

```
  1 1 0 1 0 0 1 1
+ 0 1 1 1 0 1 0 0
-----------------
1 0 1 0 0 0 1 1 1
```

Discard

The difference is positive and is equal to 01000111

The same rules apply to subtraction using the (r-1)'s complements. The only difference is that when an end carry is generated, it is not discarded but added to the least significant digit of the result. Also, if no end carry is generated, then the answer is negative and in the (r-1)'s complement form.

LECTURE 2: DIGITAL LOGIC CIRCUIT DESIGN

Example: subtract (11010011 – 10001100) using 1's complement.

Answer → 1's complement of 10001100 is 01110011

Then add →
```
  1 1 0 1 0 0 1 1
+ 0 1 1 1 0 0 1 1
-------------------
1 0 1 0 0 0 1 1 0
              1
-------------------
1 0 1 0 0 0 1 1 1
```

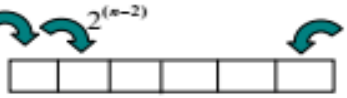
Difference

Signed Binary Numbers:

Unsigned binary numbers → $2^{(n-1)}$ ... 2^0

n bit binary number →

Signed binary numbers → Sign bit $2^{(n-2)}$ ... 2^0



The sign bit is 0 for " + " and 1 for " – ".

Three types of signed numbers are used

			Examples using 8 bits
1. Signed magnitude representation	+9	→	00001001
	-9	→	10001001
2. Signed 1's complement representation			
	+9	→	00001001
	-9	→	11110110
3. Signed 2's complement representation			
	+9	→	00001001
	-9	→	11110111

LECTURE 2: DIGITAL LOGIC CIRCUIT DESIGN

Addition of signed binary numbers:

1. In signed magnitude representation follow the rules of ordinary arithmetic. If the signs are the same, add the magnitudes and give the sum the same sign. If different signs, subtract and give the result the sign of the big number.
2. In complement representation, add the two numbers including the sign bit. Any carry out from the sign bit is ignored. No comparison or subtraction is needed.

Examples:

Add (-6) + (+13) using signed 2's complement form with 8 bits. Repeat for (+6) + (-13)

Answer → $(+6) = 00000110$ **and** $(+13) = 00001101$

$\therefore (-6) = 11111010$ **and** $(-13) = 11110011$.

-6 → 11111010
+13 → 00001101

100000111 → Answer is +7

Carry out

Sign bit

Also, +6 → 00000110
-13 → 11110011

11111001 → Answer is – 7

Sign bit

Arithmetic Subtraction:

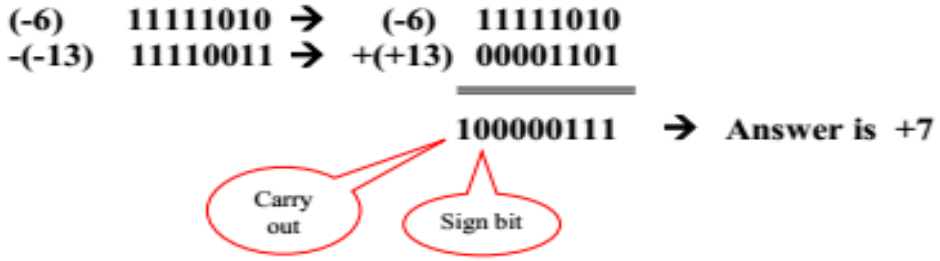
Take the 2's complement of the subtrahend, including the sign bit, and add it to the minuend. A carry out is discarded.

$$(\pm A) - (\pm B) = (\pm A) + (\mp B)$$

LECTURE 2: DIGITAL LOGIC CIRCUIT DESIGN

Example:

Perform the subtraction (-6) - (-13) using signed 2's complement representation with 8 bits.



Binary codes:

- Digital systems and circuits work with signals that have only one of two states corresponding to digital 1 and 0.
- Any discrete element of information among a group of quantities (elements) can be represented by a binary code.
- One bit can represent up to two elements (1 or 0).
- A group of 2^n distinct elements requires a minimum of n bits.
- ∴ to code a group of m elements, we need to use n bits such that: $2^n \geq m$.
- Examples:
  - A group of four elements can be represented by two bit code [00, 01, 10, and 11].
  - A binary code to represent the decimal digits [0-9], must contain at least four bits because

    $(2^4=16) \geq 10 \geq (2^3=8)$.

LECTURE 2: DIGITAL LOGIC CIRCUIT DESIGN

Decimal Codes:

Decimal Digit	BCD 8421	Excess-3	84-2-1	2421
0	0000	0011	0000	0000
1	0001	0100	0111	0001
2	0010	0101	0110	0010
3	0011	0110	0101	0011
4	0100	0111	0100	0100
5	0101	1000	1011	1011
6	0110	1001	1010	1100
7	0111	1010	1001	1101
8	1000	1011	1000	1110
9	1001	1100	1111	1111

BCD Addition:

The addition of two BCD digits with a possible carry from the previous less significant pair of digits results in a sum in the range 0 to a maximum of (9+9+1=19). There is a difference in the representation of the sum in binary and in BCD code.

1. If $0 \leq \text{Sum} \leq 9$ then, sum in BCD = sum in binary,
2. If $10 \leq \text{Sum} \leq 19$ then, sum in BCD consists of 8 bits which is not equal to the sum in binary. The correction needed in the sum to represent it in BCD is the addition of 6 to the binary sum.

4	0100	4	0100	8	1000
+5	0101	+8	1000	+9	1001
9	1001	12	1100	+17	10001
			+0110		+0110
			10010		10111

LECTURE 2: DIGITAL LOGIC CIRCUIT DESIGN

Error Detection Code:

A common method to achieve error detection in binary information transmission from one location to another is by means of a parity bit. A parity bit is defined as an extra bit included with a message to make the total number of 1's transmitted either odd or even.

	Odd Parity		Even Parity
message	P	message	P
0000	1	0000	0
0001	0	0001	1
0010	0	0010	1
0011	1	0011	0
0100	0	0100	1
0101	1	0101	0
0110	1	0110	0
0111	0	0111	1
1000	0	1000	1
1001	1	1001	0
1010	1	1010	0
1011	0	1011	1
1100	1	1100	0
1101	0	1101	1
1110	0	1110	1
1111	1	1111	0

Gray Code:

Decimal equivalent	Binary	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

LECTURE 2: DIGITAL LOGIC CIRCUIT DESIGN

ASCII Code [American Standard Code for Information Interchange]:

- Used for handling numbers, letters, characters…etc.
- This type of code is called alpha-numeric code.
- To handle 10 decimal digits, 26 small letters, 26 capital letters, and at least 20 other characters, the code must cater for at least 82 elements. $(128=2^7) \geq 82 \geq (64=2^6)$.
- Therefore we need seven bits for the code.

The ASCII code is given in the following table.

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	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	ENQ	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	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	-	o	DEL