وزارة التعليم العالي والبحث العلمي
الجامعة التقنية الشمالية
المعهد التقني كركوك

الحقيبة التعليمية

القسم العلمي: التقنيات الالكترونية والاتصالات

اسم المقرر: الدوائر الرقمية ١

المرحلة / المستوى: الاولى

الفصل الدراسي: الاول

السنة الدراسية: ٢٠٢٤ ـ ٢٠٢٥

معلومات عامة

اسم المقرر:	الدوائر الرقمية ١					
القسم:	التقنيات الالكترونية والاتصالات					
الكلية:	المعهد التقني كركوك					
المرحلة / المستوى	الاولى					
الفصل الدراسي:	الاول					
عدد الساعات الاسبوعية:	نظري	٢	عملي	٢		
عدد الوحدات الدراسية:	٤					
الرمز:	ECT103					
نوع المادة	نظري		عملي		كلهما	نعم
هل يتوفر نظير للمقرر في الاقسام الاخرى						
اسم المقرر النظير						
القسم						
رمز المقرر النظير						
معلومات تدريسي المادة						
اسم مدرس (مدرسي) المقرر:	فيصل غازي صابر					
اللقب العلمي:	مدرس مساعد					
سنة الحصول على اللقب	٢٠١٣					
الشهادة :	ماجستير					
سنة الحصول على الشهادة	٢٠١٢					
عدد سنوات الخبرة (تدريس)	٩					

الوصف العام للمقرر

تعريف الطالب على:

تعليم الطالب أسس الدوائر المنطقية في الحاسبات الالكترونية وكيفية عملها.

الاهداف العامة

- انواع انظمة الارقام
- طريقة التحويل بين انظمة الارقام
- البوابات المنطقية
- الجبر البوليوني وقاعدة دي موركن
- تبسيط المعادلات المنطقية باستخدام الجبر البوليوني وقاعدة دي موركن
- جدول كارنوف
- تبسيط المعادلة النمطقية باستخدام جدول كارنوف
- الحساب باستخدام النظام الثنائي (الجمع والطرح زالقسمة والضرب)
- دائرة المقارنة الرقمي
- دائرة فك التشفير

الأهداف الخاصة

- التعرف على الانظمة الرقمي وطرق التحويل بينها
- التعرف على النظام الرقمي الثنائي المستعملة في التقنيات الرقمية
- التعرف على البوابات الرقمية وتشكيل دوائر منها
- طرق الاختصار لكتابة المعادلات المنطقية
- طرق الحساب باستخدام النظام الرقمي الثنائي

الأهداف السلوكية او نواتج التعلم

- التعرف على اسماء القطع الالكترونية التي تتكون منها البوابات المنطقية
- التعرف على المجالات الممكنه استخدام البوابات المنطقية
- الظروف القياسية لتشغل كل قطعة من البوابات المنطقية

- **أمثلة أهداف تدريسية:**
- استخدام البوابات المنطقية في تشكيل دوائر التشفير
- استخدام البوابات المنطقية في تشكيل دوائر فك التشفير
- استخدام البوابات المنطقية في تشكيل دوائر التضمين
- استخدام البوابات المنطقية في تشكيل دوائر فك التضمين

المتطلبات السابقة

- مادة الفيزياء والرياضيات لطلبة الاعدادية
- مادة الطبيعيات لطلبة الدراسات المهنية

الفصل الاول اساسيات الارقام والدوال الرقمية

عنوان الفصل / التوزيع الزمني	الوقت: النظري	الوقت: العملي	العنوان الفرعي		طريقة التدريس	التقنيات	طرق القياس
الأسبوع الأول	٢		A general idea of numerical systems (types and details)		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
الاسبوع الثاني	٢		Transfers between the numerical systems		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
الاسبوع الثالث	٢		Logic gates (types, working principle, truth tables, logical symbol)		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
الاسبوع الرابع	٢		How to connect the logic gates to form logic circuits.		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
الاسبوع الخامس	٢		Boolean algebra and the rule of de-Morgan		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
الاسبوع السادس	٢		Simplification of logical equations using Boolean algebra and the laws of De Morgan's laws		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
الاسبوع السابع	٢		The design of the logical gates using NOR and NAND circuits,		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
الاسبوع الثامن	٢		Ways of writing the equation from truth table (POS, SOP).		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
الاسبوع التاسع	٢		Karnaugh Map (for two variables, the three variables, the four variables)		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
الاسبوع العاشر	٢		Simplification of logical equations using Karnaugh Map		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	

عنوان الفصل	الوقت						
الفصل الثاني الدايود							
التوزيع الزمني	النظري	العملي	العنوان الفرعي		طريقة التدريس	التقنيات	طرق القياس
الأسبوع الحادي عشر	٢		Calculations in the binary system (addition, subtraction, subtraction using complements).		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
الأسبوع الثاني عشر	٢		Logic circuit applications(half adder, full adder, parallel adder circuits)		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
الاسبوع الثالث عشر	٢		Binarysubtractorcircuits (half subtractor,full subtractorparallel subtractor) circuit using the adder circuit by method of 1s complements.		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
الاسبوع الرابع عشر	٢		The circuit of digital comparator (one stage and two stages)		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
الاسبوع الخامس عشر	٢		The circuit of decoder size of 2:4 ,3:8 and 4:10		محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	
					محاضرة	عرض تقديمي، شرح، أسئلة وأجوبة, مناقشة	

Outline

- Base (radix)
- Conversion of any decimal number to the base R
- Conversion of any base R to the decimal number
- Conversion of Binary
- Mathematical operations

1- Base (Radix)

In the number system the base or radix tells the number of symbols used in the system. In the earlier days, different civilizations were using different radixes. The Egyptian used the radix 2, the Babylonians used the radix 60 and Mayans used 18 and 20.

The base of a number system is indicated by a subscript (decimal number) and this will be followed by the value of the number. For example $(952)_{10}$, $(456)_8$, $(314)_{16}$.

Number System that are used by the computers-

- Decimal System
- Binary System
- Octal System
- Hexadecimal System

1.1. Decimal System

The decimal system is the system which we use in everyday counting. The number system includes the ten digits from 0 through 9. These digits are recognized as the symbols of the decimal system. Each digit in a base ten number represents units ten times the units of the digit to its right.

For example

$9542 = 9000 + 500 + 40 + 2 = (9 \times 10^3) + (5 \times 10^2) + (4 \times 10^1) + (2 \times 10^0)$

1.2. Binary System

Computers do not use the decimal system for counting and arithmetic. Their CPU and memory are made up of millions of tiny switches that can be either in ON and OFF states. **0** represents OFF and **1** represents ON. In this way we use binary system.

Binary system has two numbers 0 and 1. Binary system has base 2 therefore the weight of n^{th} bit of the number from Right Hand Side is n^{th} bit $\times 2^{n-1}$.

1.3. Octal System

The octal system is commonly used with computers. The octal number system with its 8 digit 0,1,2,3,4,5,6, and 7 has base 8. The octal system uses a power of 8 to determine the digit of a number's position.

1.4. Hexadecimal System

Hexadecimal is another number system that works exactly like the decimal, binary and octal number systems, except that the base is 16. Each hexadecimal represents a power of 16. The system uses 0 to 9 numbers and A to F characters to represent 10 to 15 respectively.

Table 1 Special base

Decimal	Binary	Octal	Hexadecimal
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
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Table 2 Deferent base

	2	3	4	5	...	8	...	10	11	12	...	16
	0001	001	01	01		01		01	01	01		1
	0010	002	02	02		02		02	02	02		2
	0011	010	03	03		03		03	03	03		3
	0100	011	10	04		04		04	04	04		4
	0101	012	11	10		05		05	05	05		5
	0110	020	12	11		06		06	06	06		6
$(N)_b$	0111	021	13	12		07		07	07	07		7
	1000	022	20	13		10		08	08	08		8
	1001	100	21	14		11		09	09	09		9
	1010	101	22	20		12		10	0A	0A		A
	1011	102	23	21		13		11	10	0B		B
	1100	110	30	22		14		12	11	10		C
	1101	111	31	23		15		13	12	11		D
	1110	112	32	24		16		14	13	12		E
	1111	120	33	30		17		15	14	13		F

2- Conversion of any decimal number to the base R

Any number in one number system can be converted into any other number system. There are the various methods that are used in converting numbers from one base to another.

2.1. Conversion of Decimal to Binary

2.1.1 Integers

The method of converting a decimal integer number to a binary integer number entails repeatedly dividing the decimal number by 2, keeping track of the remainder at each step. To convert the decimal number *x* to binary:

- Divide *x* by 2 to obtain a quotient and remainder. The remainder will be 0 or 1.
- If the quotient is zero, you are finished: Proceed to Step 3. Otherwise, go back to Step 1, assigning *x* to be the value of the most recent quotient from Step 1.
- The sequence of remainders forms the binary representation of the number.

Example 1:

$(23)_{10} \longrightarrow (10111)_2$

23/2 = 11	Remainder	↑
11/2 = 5	1	LSB
5/2 = 2	1	
2/2 = 1	1	
1/2 = 0	0	MSB
	1	

(LSB) → **Least Significant Bit**

(MSB) → **Most Significant Bit**

Example 2:

$(53)_{10} \longrightarrow (110101)_2$

53/2 = 26	Remainder	↑
26/2 = 13	1	LSB
13/2 = 6	0	
6/2 = 3	1	
3/2 = 1	0	
1/2 = 0	1	MSB
	1	

2.1.2 Fractions

To convert a decimal fraction to its binary fraction, multiplication by 2 is carried out repetitively and the integer part of the result is saved and placed after the decimal point . The fractional part is taken and multiplied by 2. The process can be stopped any time after the desired accuracy has been achieved.

Example 3:

$(0.375)_{10} \longrightarrow (0.011)_2$

$0.375 \times 2 = 0.75$	Integer	↓ MSB
$0.75 \times 2 = 1.5$	0	
$0.5 \times 2 = 1$	1	
	1	LSB

Example 4:

$(0.6875)_{10} \longrightarrow (0.1011)_2$

$0.6875 \times 2 = 1.375$	Integer	↓ MSB
$0.375 \times 2 = 0.75$	1	
$0.75 \times 2 = 1.5$	0	
$0.5 \times 2 = 1$	1	
	1	LSB

Example 5:

$(53.6875)_{10} \longrightarrow (110101.1011)_2$

2.2 Conversion of Decimal to Octal

2.2.1 Integers

We follow the same process of converting decimal to binary. Instead of dividing the number by 2, we divide the number by 8.

Example 6:

$(45)_{10} \longrightarrow (55)_8$

45/8 = 5	Remainder	↑ LSB
5/8 = 0	5	
	5	MSB

Example 7:

$(514)_{10} \longrightarrow (1002)_8$

514/8 = 64	Remainder	↑
64/8 = 8	2	LSB
8/8 = 1	0	
1/8 = 0	0	
	1	MSB

2.2.2 Fractions

We follow the same steps of conversions of decimal fractions to binary fractions. Here we multiply the fraction by 8 instead of 2.

Example 8:

$(0.75)_{10} \longrightarrow (0.6)_8$

0.75 × 8 = 6.00	Integer	↓ MSB
	6	LSB

Example 9:

$(0.1875)_{10} \longrightarrow (0.14)_8$

0.1875 × 8 = 1.5	Integer	↓ MSB
0. 5 × 8 = 4.0	1	
	4	LSB

Example 10:

$(45.1875)_{10} \longrightarrow (55.14)_8$

2.3 Conversion of Decimal to Hexadecimal

2.3.1 Integer

We divide by 16 instead of 2 or 8. If the remainder is in between 10 to 16, then the number is represented by A to F respectively.

Example 11:

$(45)_{10}$ ⟶ $(2D)_{16}$

45/16 = 2	Remainder	↑ LSB
2/16 = 0	(13) = D	MSB
	2	

Example 12:

$(295)_{10}$ ⟶ $(127)_{16}$

295/16 = 18	Integer	↑ MSB
18/16 = 1	7	
1/16 = 0	2	LSB
	1	

2.3.2 Fractions

Here we multiply the fraction by 16 instead of 2 or 8. If the non-zero integer is in between 10 to 16, then the number is represented by A to F respectively.

Example 13:

$(0.75)_{10}$ ⟶ $(0.C)_{16}$

0.75 × 16 = 6.00	Integer	↓ MSB
	(12) = C	LSB

Example 14:

$(0.9875)_{10}$ ⟶ $(0.FCC)_{16}$

0.9875 × 16 = 15.8	**Integer**	↓ **MSB**
0. 8 × 16 = 12.8	**(15) = F**	
0.8 × 16 = 12.8	**(12) = C**	
	(12) = C	**LSB**

Example 14:

$(295.9875)_{10}$ ⟶ $(127.FCC)_{16}$

3- Conversion of any base R to the decimal number

3.1- Conversion of Binary to Decimal

3.1.1 Integer

Each position of binary digit can be replaced by an equivalent power of 2 as shown below

2^{n-1}	2^{n-2}	……	……	2^3	2^2	2^1	2^0

Thus to convert any binary number replace each binary digit (bit) with its power and add up.

Example15: convert $(1011)_2$ to its decimal equivalent

2^{n-1}	2^{n-2}	……	……	2^3	2^2	2^1	2^0
				1	0	1	1

$(1011)_2 = (1×1 + 1×2 + 0×4 + 1×8)_{10} = (1+2+0+8)_{10} = (11)_{10}$

3.1.2 Fraction

In a binary fraction, the position of each digit (bit) indicates its relative weight as was the case with the integer part, except the weights to in the reverse direction. Thus after the decimal point, the first digit (bit) has a weight of 1/2, the next one has a weight of 1/4, followed by 1/8 and so on.

.	2^{-1}	2^{-2}	2^{-3}	2^{-4}	….	….	2^{-n}

Example16: convert $(0.1011)_2$ to its decimal equivalent

.	2^{-1}	2^{-2}	2^{-3}	2^{-4}	….	….	2^{-n}
.	1	0	1	1			

$(0.1011)_2 = (1\times1/2 + 0\times1/4 + 1\times1/8 + 1\times1/16)_{10}$

$= (0.5+0+0.125+0.0625)_{10} = (0.6875)_{10}$

3.2- Conversion of Octal to Decimal

3.2.1 Integer

Each position of octal digit can be replaced by an equivalent power of 8 as shown below

8^{n-1}	8^{n-2}	……	……	8^3	8^2	8^1	8^0

Thus to convert any octal number replace each octal digit (bit) with its power and add up.

Example17: convert $(743)_8$ to its decimal equivalent

8^{n-1}	8^{n-2}	……	……	8^3	8^2	8^1	8^0
					7	4	3

$(743)_8= (3\times1 + 4\times8 + 7\times64)_{10} = (3+32+448)_{10} = (483)_{10}$

3.2.2 Fraction

In a Octal fraction, the position of each digit (bit) indicates its relative weight as was the case with the integer part, except the weights to in the reverse direction. Thus after the decimal point, the first digit (bit) has a weight of 1/8, the next one has a weight of 1/64, followed by 1/512 and so on.

.	8^{-1}	8^{-2}	8^{-3}	8^{-4}	….	….	8^{-n}

Example18: convert $(0.413)_8$ to its decimal equivalent

.	8^{-1}	8^{-2}	8^{-3}	8^{-4}	....	....	8^{-n}
.	4	1	3				

$(0.413)_8 = (4/8 + 1/64 + 3/512)_{10}$
$= (0.5+0.015625+0.005859375)_{10} = (0.5221)_{10}$

3.3- Conversion of Hexadecimal to Decimal

3.3.1 Integer

Each position of Hexadecimal digit can be replaced by an equivalent power of 16 as shown below

16^{n-1}	16^{n-2}	...	...	16^3	16^2	16^1	16^0

Thus to convert any hexadecimal number replace each hexadecimal digit (bit) with its power and add up.

Example19: convert $(F4C)_{16}$ to its decimal equivalent

16^{n-1}	16^{n-2}	...	...	16^3	16^2	16^1	16^0
					F=(15)	4	C=(12)

$(F4C)_{16} = (12\times1 + 4\times16 + 15\times256)_{10} = (12 + 64 + 3840)_{10} = (3916)_{10}$

3.3.2 Fraction

In a Hexadecimal fraction, the position of each digit (bit) indicates its relative weight as was the case with the integer part, except the weights to in the reverse direction. Thus after the decimal point, the first digit (bit) has a weight of 1/16, the next one has a weight of 1/256, followed by 1/4096 and so on.

.	16^{-1}	16^{-2}	16^{-3}	16^{-4}	....	....	16^{-n}

Example20: convert $(0.\ B1EB)_{16}$ to its decimal equivalent

.	16^{-1}	16^{-2}	16^{-3}	16^{-4}	...	...	16^{-n}
.	B=(11)	1	E=(13)	B=(11)			

$(0.\ B1EB)_{16} = (11/16 + 1/256 + 13/4096 + 11/65536)_{10}$
$= (0.6875 + 0.0039 + 0.0031 + 0.0001)_{10} = (0.6946)_{10}$

4- Conversion of Binary

4.1 Conversions of Binary to Octal

4.1.1 Integer

We use the following steps in converting binary to octal

- Break the number into 3-bit sections starting from LSB to MSB . If we do not have sufficient bits in grouping of 3-bits, we add zeros to the
- Left of **MSB** so that all the groups have proper 3-bit number .
- Write the 3-bit binary number to its octal equivalent.

Example21: Convert $(1101101)_2$ into octal.

Binary Number	001	101	101
Octal Number	1	5	5

Thus $(1101101)_2 = (155)_8$.

4.1.2 Fraction

We use the following steps in converting binary fractions to octal fractions

- Break the fraction into 3-bit sections starting from MSB to LSB .
- In order to get a complete grouping of 3 bits, we add trailing zeros in LSB .
- Write the 3-bit binary number to its octal equivalent.

Example22: Convert $(101101.11)_2$ into octal.

Binary Number	001	101.	110
Octal Number	5	5.	6

Thus $(101101.11)_2 = (55.6)_8$.

4.2 Conversions of Binary to Hexadecimal

4.2.1 Integer

We convert binary to hexadecimal in the similar manner as we have converted binary to octal. The only difference is that here, we form the group of 4 –bits.

Example23: Convert $(101101)_2$ into hexadecimal.

Binary Number	0010	1101
Hexadecimal Number	2	D

Thus $(101101)_2 = (2D)_{16}$.

4.2.2 Fraction

We convert binary fractions to hexadecimal fractions in the similar manner as we have converted binary fractions to octal fractions. The only difference is that here we form the group of 4 bits.

Example24: Convert $(101101.11)_2$into hexadecimal.

Binary Number	0010	1011.	1100
Hexadecimal Number	2	D.	C

Thus $(101101.11)_2 = (2D.C)_{16}$.

5. Logic Gates

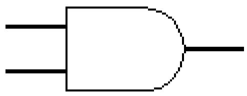
A logic gate is an elementary building block of a digital circuit. Most logic gates have two inputs and one output. At any given moment, every terminal is in one of the two binary conditions low (0) or high (1), represented by different voltage levels. The logic state of a terminal can, and generally does, change often, as the circuit processes data. In most logic gates, the low state is approximately zero volts (0 V), while the high state is approximately five volts positive (+5 V).

There are seven basic logic gates: AND, OR, XOR, NOT, NAND, NOR, and XNOR.

5.1 AND Gate

The AND gate is so named because, if 0 is called "false" and 1 is called "true," the gate acts in the same way as the logical "and" operator. The following illustration and table show the circuit symbol and logic combinations for an AND gate. (In the symbol, the input terminals are at left and the output terminal is at right.) The output is "true" when both inputs are "true." Otherwise, the output is "false."

Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

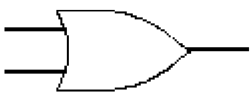


AND gate

5.2 OR Gate

The OR gate gets its name from the fact that it behaves after the fashion of the logical inclusive "or." The output is "true" if either or both of the inputs are "true." If both inputs are "false," then the output is "false."

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

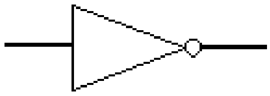


OR gate

5.3 NOT Gate

A logical inverter, sometimes called a NOT gate to differentiate it from other types of electronic inverter devices, has only one input. It reverses the logic state.

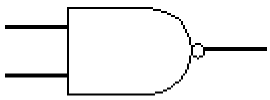
Input	Output
1	0
0	1

NOT gate  Inverter or

5.4 NAND Gate

The NAND gate operates as an AND gate followed by a NOT gate. It acts in the manner of the logical operation "and" followed by negation. The output is "false" if both inputs are "true." Otherwise, the output is "true."

Input 1	Input 2	Output
0	0	1
0	1	1
1	0	1
1	1	0

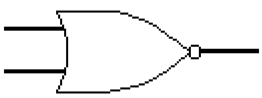


NAND gate

5.5 NOR Gate

The NOR gate is a combination OR gate followed by an inverter. Its output is "true" if both inputs are "false." Otherwise, the output is "false".

Input 1	Input 2	Output
0	0	1
0	1	0
1	0	0
1	1	0

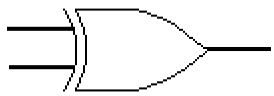


NOR gate

5.6 XOR Gate

The XOR (exclusive-OR) gate acts in the same way as the logical "either/or." The output is "true" if either, but not both, of the inputs are "true." The output is "false" if both inputs are "false" or if both inputs are "true." Another way of looking at this circuit is to observe that the output is 1 if the inputs are different, but 0 if the inputs are the same.

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

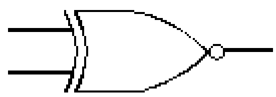


XOR gate

5.7 XNOR Gate

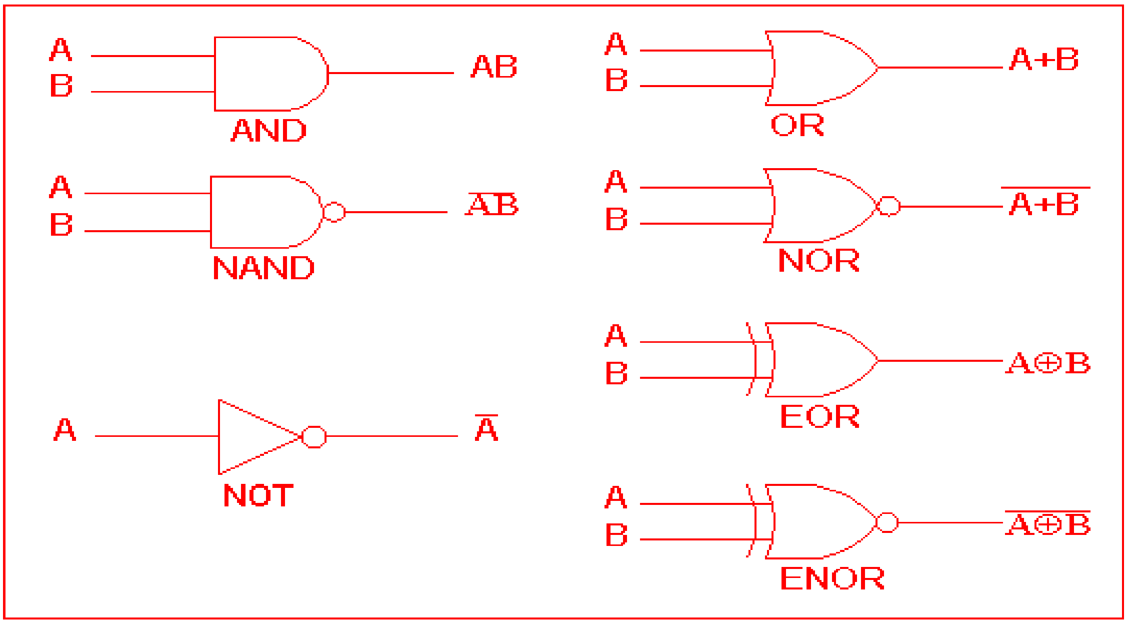
The XNOR (exclusive-NOR) gate is a combination XOR gate followed by an inverter. Its output is "true" if the inputs are the same, and "false" if the inputs are different.

Input 1	Input 2	Output
0	0	1
0	1	0
1	0	0
1	1	1



XNOR gate

Using combinations of logic gates, complex operations can be performed. In theory, there is no limit to the number of gates that can be arrayed together in a single device. But in practice, there is a limit to the number of gates that can be packed into a given physical space. Arrays of logic gates are found in digital integrated circuits (ICs). As IC technology advances, the required physical volume for each individual logic gate decreases and digital devices of the same or smaller size become capable of performing ever-more-complicated operations at ever-increasing speeds.



INPUTS		OUTPUTS					
A	B	AND	NAND	OR	NOR	EXOR	EXNOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

6- Boolean Algebra

Like all algebras, there are rules to manipulate Boolean expressions. The simplest are the rules that concern the unary operator NOT

$$\overline{(\bar{A})} = A$$

$$A \cdot \bar{A} = 0$$

$$A + \bar{A} = 1$$

General rules like the distributive(توزيعي), commutative(تبادلي), and associative(ترابطي) rules hold for the AND and OR binary operators as follows.

Associative $(A\cdot B)\cdot C = A\cdot(B\cdot C)$

$(A+B)+C = A+(B+C)$

Commutative $A\cdot B = B\cdot A$

$A+B = B+A$

Distributive $A\cdot(B+C) = A\cdot B+A\cdot C$

$A+(B\cdot C) = (A+B)\cdot(A+C)$

In addition, there are simplification rules for Boolean equations. There are three important groups of simplification rules. The first one uses just one variable:

$$A \cdot A = A$$

$$A + A = A$$

The second group uses Boolean constants 0 and 1:

$$A \cdot 0 = 0$$

$$A \cdot 1 = A$$

$$A + 0 = A$$

$$A + 1 = 1$$

The third group involves two or more variables and contains a large number of possible simplification rules (or theorems) such as:

$$A + A \cdot B = A$$

proof $\quad A+A \cdot B=A \cdot (1+B)+A \cdot 1=A$

There are two important rules which constitute de Morgan's theorem:

$$\overline{(A + B)} = \overline{A} \cdot \overline{B}$$

$$\overline{(A \cdot B)} = \overline{A} + \overline{B}$$

de Morgan's theorem

SUMMARY

1− $A \cdot B = B \cdot A$

$A + B = B + A$

2- $A \cdot (B \cdot C) = (A \cdot B) \cdot C$

$A + (B + C) = (A + B) + C$

3− $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$

$A + (B \cdot C) = (A + B) \cdot (A + C)$

4- $A \cdot A = A$

$A + A = A$

5- $A \cdot (A + B) = A$

$A + (A \cdot B) = A$

6 – $A \cdot \overline{A} = 0$

$A + \overline{A} = 1$

7 - $\overline{\overline{A}} = A$

8 – $\overline{(A \cdot B)} = \overline{A} + \overline{B}$

$\overline{(A + B)} = \overline{A} \cdot \overline{B}$

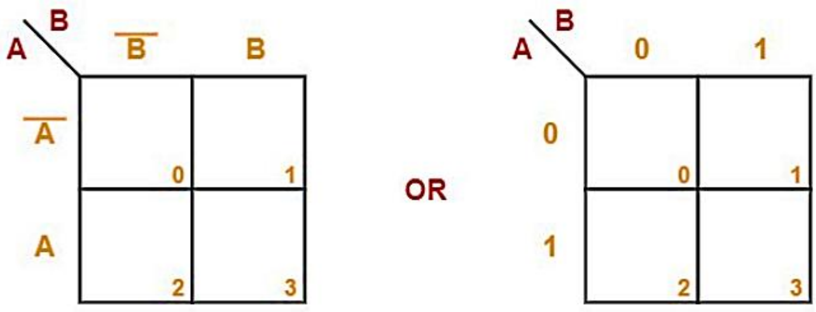
7. Karnaugh Map

The Karnaugh Map also called as K Map is a graphical representation that provides a systematic method for simplifying the boolean expressions For a boolean expression consisting of n-variables, number of cells required in K Map = 2^n cells.

7.1. Two Variable K Map

1. Two variable K Map is drawn for a boolean expression consisting of two variables.
2. The number of cells present in two variable K Map = 22 = 4 cells.
3. So, for a boolean function consisting of two variables, we draw a 2 x 2 K Map.

Two variable K Map may be represented as



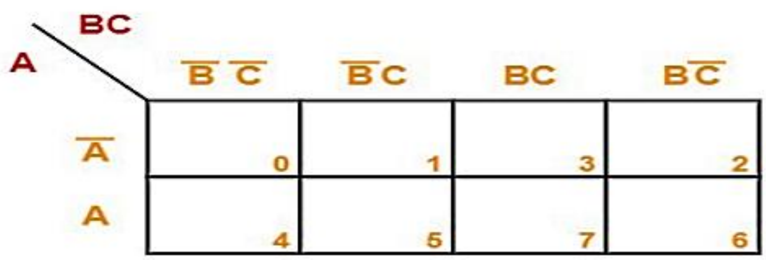
Two variable K Map

Here, A and B are the two variables of the given boolean function.

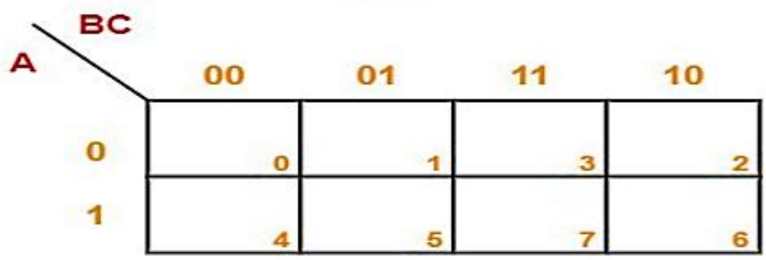
7.2. Three Variable K Map

1. Three variable K Map is drawn for a boolean expression consisting of three variables.
2. The number of cells present in three variable K Map = 23 = 8 cells.
3. So, for a boolean function consisting of three variables, we draw a 2 x 4 K Map.

Three variable K Map may be represented as



OR



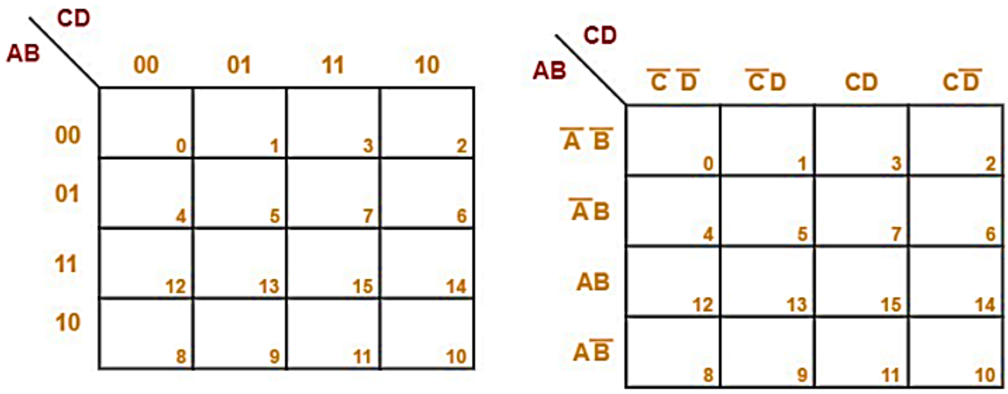
Three variable K Map

Here, A, B and C are the three variables of the given boolean function.

7.3. Four Variable K Map

1. Four variable K Map is drawn for a boolean expression consisting of four variables.
2. The number of cells present in four variable K Map = 24 = 16 cells.
3. So, for a boolean function consisting of four variables, we draw a 4 x 4 K Map.

Four variable K Map may be represented as



Four variable K Map

Here, A, B, C and D are the four variables of the given boolean function.

7.4. Karnaugh Map Simplification Rules

To minimize the given boolean function,

1. We draw a K Map according to the number of variables it contains.
2. We fill the K Map with 0’s and 1’s according to its function.
3. Then, we minimize the function in accordance with the following rules.

Rule-01:

1. We can either group 0’s with 0’s or 1’s with 1’s but we cannot group 0’s and 1’s together.
2. X representing don’t care can be grouped with 0’s as well as 1’s.

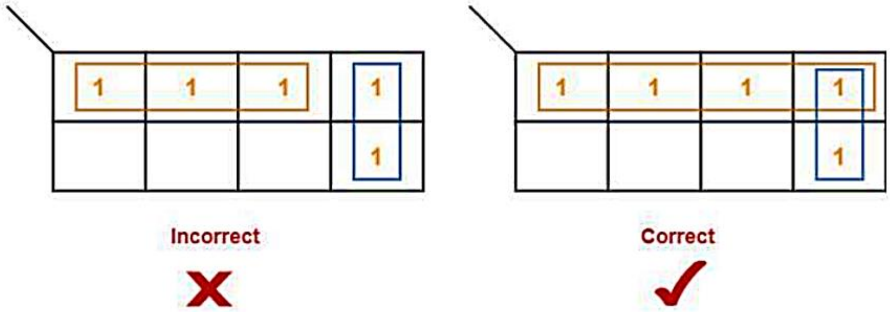
NOTE There is no need of separately grouping X’s i.e. they can be ignored if all 0’s and 1’s are already grouped.

Rule-02:
Groups may overlap each other.

Rule-03:

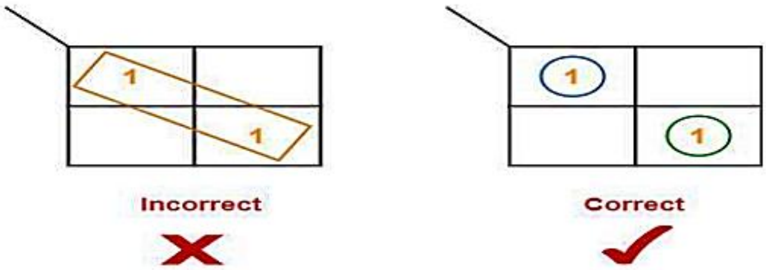
1. We can only create a group whose number of cells can be represented in the power of 2.
2. In other words, a group can only contain 2n i.e. 1, 2, 4, 8, 16 and so on number of cells.

Example 25

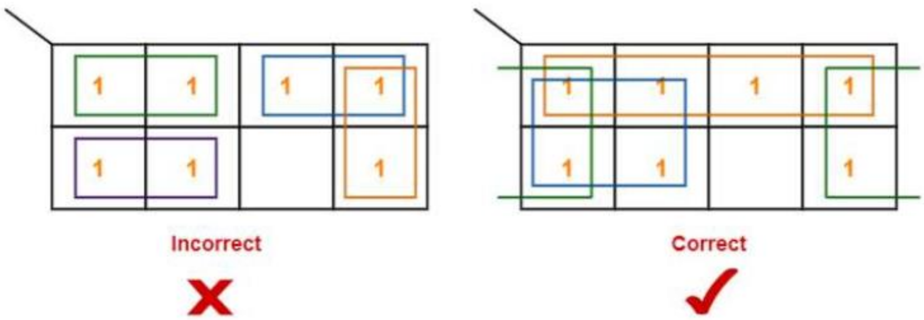


Rule-04:

1. Groups can be only either horizontal or vertical.
2. We cannot create groups of diagonal or any other shape.



Example 26



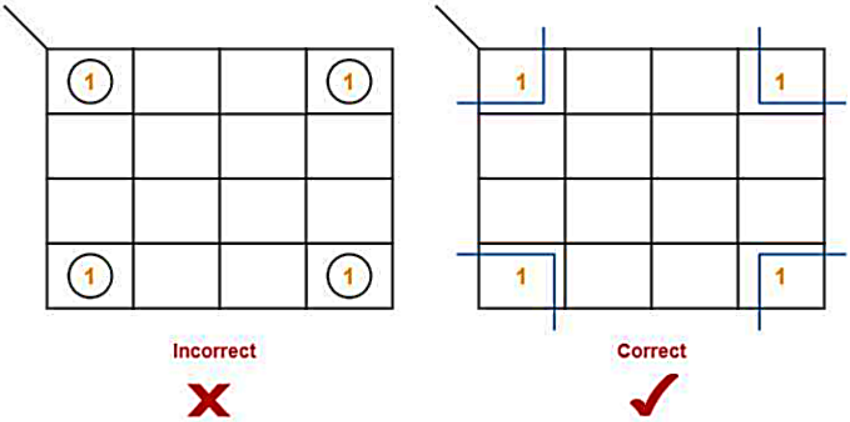
Rule-05:

Each group should be as large as possible.

Rule-06:

1. Opposite grouping and corner grouping are allowed.
2. The example of opposite grouping is shown illustrated in Rule-05.
3. The example of corner grouping is shown below.

Example 27



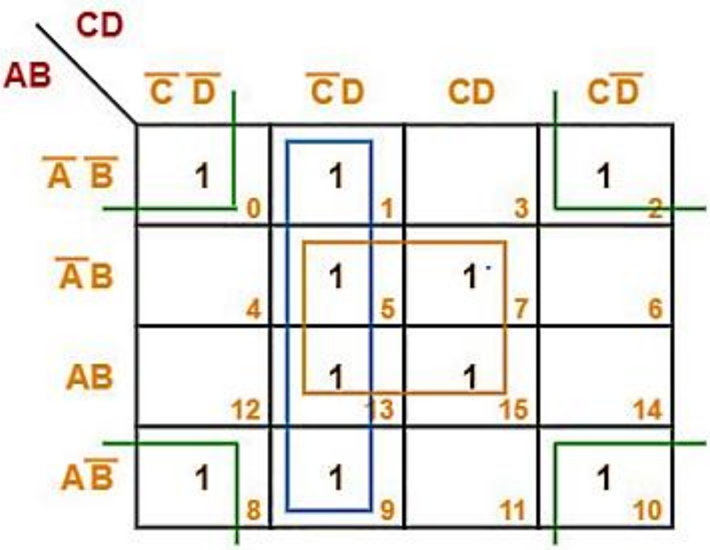
Rule-07:
There should be as few groups as possible

Example 28

Minimize the following boolean function

F(A, B, C, D) = Σm(0, 1, 2, 5, 7, 8, 9, 10, 13, 15)

Solution-
1. Since the given boolean expression has 4 variables, so we draw a 4 x 4 K Map.
2. We fill the cells of K Map in accordance with the given boolean function.
3. Then, we form the groups in accordance with the above rules.
Then, we have



Now, F(A, B, C, D) = (A'B + AB)(C'D + CD) + (A'B' + A'B + AB + AB')C'D + (A'B' + AB')(C'D' + CD') = BD + C'D + B'D'
Thus, minimized boolean expression is

F(A, B, C, D) = BD + C'D + B'D'

Example 29

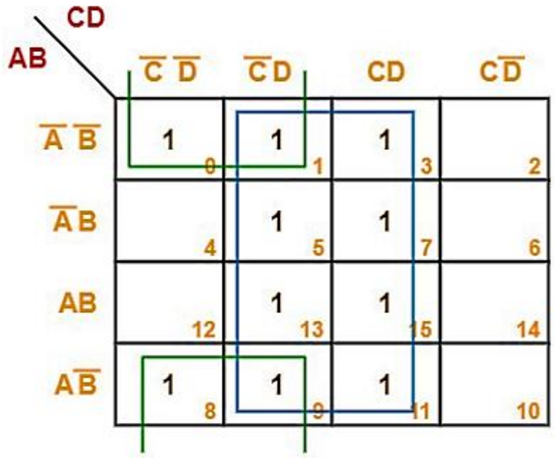
Minimize the following boolean function

F(A, B, C, D) = Σm(0, 1, 3, 5, 7, 8, 9, 11, 13, 15)

Solution-

1. Since the given boolean expression has 4 variables, so we draw a 4 x 4 K Map.
2. We fill the cells of K Map in accordance with the given boolean function.
3. Then, we form the groups in accordance with the above rules.

Then, we have



Now,

F(A, B, C, D) = (A’B’ + A’B + AB + AB’)(C’D + CD) + (A’B’ + AB’)(C’D’ + C’D) = D + B’C’

Thus, minimized boolean expression is

F(A, B, C, D) = B’C’ + D

Example 30

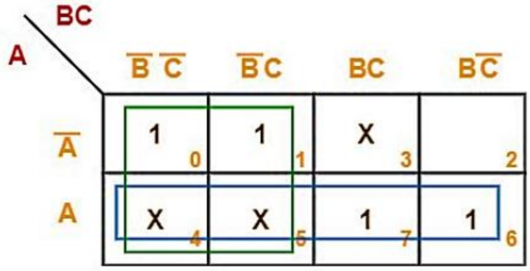
Minimize the following boolean function

F(A, B, C) = Σm(0, 1, 6, 7) + Σd(3, 4, 5)

Solution-

1. Since the given boolean expression has 3 variables, so we draw a 2 x 4 K Map.
2. We fill the cells of K Map in accordance with the given boolean function.
3. Then, we form the groups in accordance with the above rules.

Then, we have



Now, F(A, B, C) = (A + A’)(B’C’ + B’C) + A(B’C’ + B’C + BC + BC’) = B’ + A
Thus, minimized boolean expression is
F(A, B, C) = A + B’

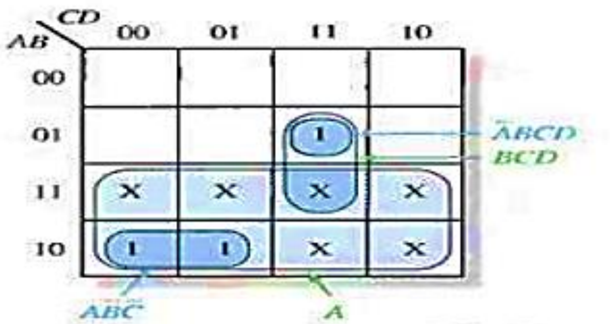
"Don't Care" Conditions

Sometimes a situation arises in which some input variable combinations are not allowed. For example, recall that in the BCD code there are six invalid combinations: 1010, 1011, 1100, 1101, 1110, and 1111. Since these unallowed states will never occur in an application involving the BCD code, they can be treated as "don't care" terms with respect to their effect on the output. That is, for these "don't care" terms either a 1 or a 0 may be assigned to the output: it really does not matter since they will never occur. The "don't care" terms can be used to advantage on the Karnaugh map. Fig. shows that for each "don't care" term, an X is placed in the cell. When grouping the 1 s, the Xs can be treated as 1s to make a larger grouping or as 0s if they cannot be used to advantage. The larger a group, the simpler the resulting term will be.

Inputs A B C D	Output Y
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	0
0 1 0 0	0
0 1 0 1	0
0 1 1 0	0
0 1 1 1	1
1 0 0 0	1
1 0 0 1	1
1 0 1 0	X
1 0 1 1	X
1 1 0 0	X
1 1 0 1	X
1 1 1 0	X
1 1 1 1	X

Don't cares

(a) Truth table



(b) Without "don't cares" $Y = A\bar{B}\bar{C} + \bar{A}BCD$
With "don't cares" $Y = A + BCD$

The truth table in above: (a) describes a logic function that has a 1 output only when the BCD code for 7,8, or 9 is present on the inputs. If the "don't cares" are used as 1s, the resulting expression for the function is A + BCD, as indicated in part (b). If the "don't cares" are not used as 1s, the resulting expression is ABC + ABCD: so you can see the advantage of using "don't care" terms to get the simplest expression.

1- Arithmetic Operations

1.1 Additional Operations

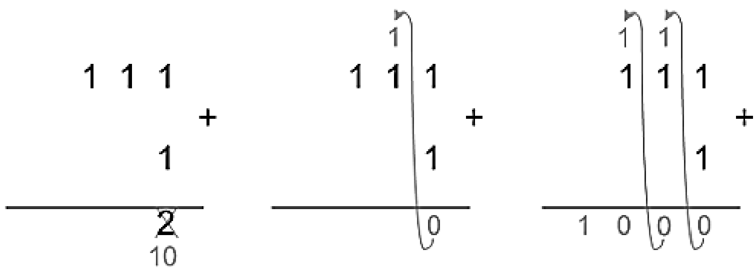
1.1.1 Addition of Binary Numbers

Addition of binary numbers is basically the same as addition of decimal numbers. Each system has a sum, and carries.

Since only two symbols, 0 and 1, are used with the binary system, only four combinations of addition are possible.

Sum	Carry
0 + 0 = 0	0
0 + 1 = 1	0
1 + 0 = 1	0
1 + 1 = 0	1

Example 1: Add $(1)_2 + (111)_2$



Example 2: Add $(1001.011)_2 + (1101.101)_2$

Carries:	*10011 11*	
	1001.011	$= (9.375)_{10}$
	1101.101	$=(13.625)_{10}$
	10111.000	$= (23)_{10}$ = Sum

1.1.2 Addition of Octal Numbers

The addition of octal numbers is not difficult provided you remember that anytime the sum of two digits exceeds 7, a carry is produced. Compare the two examples shown below:

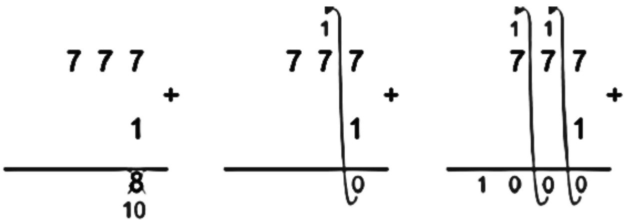
$$\begin{array}{r} 4_8 \\ +2_8 \\ \hline 6_8 \end{array} \qquad \begin{array}{r} 4_8 \\ +4_8 \\ \hline 10_8 \end{array}$$

The octal addition table in table 3 will be of benefit to you until you are accustomed to adding octal numbers.

Table 3 Octal Addition Table

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

Example 3: For the octal system, $(1)_8 + (777)_8 = (1000)_8$



Example 4: Add 6_8 and 5_8

$$\begin{array}{r} 6_8 \\ +\ 5_8 \\ \hline 13_8 \end{array}$$

1.1.3 Addition of Hexadecimal Numbers

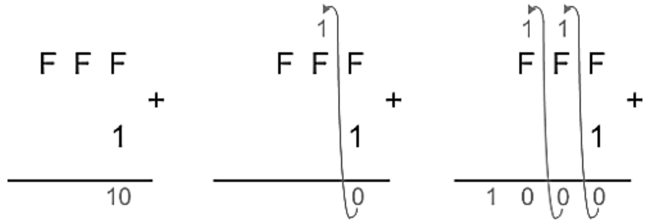
The addition of Hexadecimal numbers may seem intimidating at first glance, but it is no different than addition in any other number system. The same rules apply. Certain combinations of symbols produce a carry while others do not. Some numerals combine to produce a sum represented by a letter. After a little practice you will be as confident adding hexadecimal numbers as you are adding decimal numbers.

Study the hexadecimal addition table in table 4. Using the table, add 7 and 7. In this case 7 + 7= E. As long as the sum of two numbers is 15_{10}or less, only one symbol is used for the sum. A carry will be produced when the sum of two numbers is 16_{10} or greater.

Table 4 Hexadecimal Addition Table

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Example 5: For the hex system, $(1)_{16} + (FFF)_{16} = (1000)_{16}$



Example 6: Add 456_{16} and 784_{16}

$$
\begin{array}{r}
456_{16} \\
+\ 784_{16} \\
\hline
BDA_{16}
\end{array}
$$

1.2 Subtraction Operations

1.2.1 Subtraction of Binary Numbers

Now that you are familiar with the addition of binary numbers, subtraction will be easy. The following are the four rules that you must observe when subtracting:

Difference	Borrow
0 – 0 = 0	0
0 – 1 = 1	1
1 – 0 = 1	0
1 – 1 = 0	0

Example 7: Subtract $(10001)_2$ from $(11000)_2$

```
   1
  011
 11000        24
-10001      - 17
 00111         7
```

1.2.2 Subtraction of Octal Numbers

The subtraction of octal numbers follows the same rules as the subtraction of numbers in any other number system. The only variation is in the quantity of the borrow. In the decimal system, you had to borrow a group of 10_{10}. In the binary system, you borrowed a group of 2_{10}. In the octal system you will borrow a group of 8_{10}.

Consider the subtraction of 1 from 10 in decimal, binary, and octal number systems:

DECIMAL	BINARY	OCTAL
10_{10}	10_2	10_8
$-\ 1_{10}$	$-\ 1_2$	$-\ 1_8$
9_{10}	1_2	7_8

Example 8: Subtract $(7)_8$ from $(46)_8$

```
  10
  3
  46₈
-  7₈
  37₈
```

2.2.3 Subtraction of Hexadecimal Numbers

The subtraction of Hexadecimal numbers looks more difficult than it really is. In the preceding sections you learned all the rules for subtraction. Now you need only to apply those rules to a new number system. The symbols may be different and the amount of the borrow is different, but the rules remain the same.

Example 9: Subtract $(642)_{16}$ from $(ABC)_{16}$

$$\begin{array}{r} ABC_{16} \\ -\ 642_{16} \\ \hline 47A_{16} \end{array}$$

1.3 Multiplication Operation (Binary Multiplication)

The binary multiplication table is as follows:

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

The process of binary multiplication is illustrated by the following example:

```
   110.10
     10.1
   ------
    11010
   00000
  11010
  -------
10000.010
```

For every digit of the multiplier which is equal to 1, a partial product is formed consisting of the multiplicand shifted so that its least significant digit is aligned with the 1of the multiplier. An all-zero partial product is formed for each 0 multiplier digit. Of course, the all-zero partial products can be omitted. The final product is formed by summing all the partial products. The binary point is placed in the product by using the same rule as for decimal multiplication: the number of digits to the right of the binary point of the product is equal to the sum of the numbers of digits to the right of the binary points of the multiplier and the multiplicand.

Example 10: find $(53)_{10} \times (7)_{10}$

```
   110101
   000111
   ------
   110101
  1101010
 11010100
 ---------
101110011     (371)10
```

1.4 Division Operation (Binary Division)

Division is the most complex of the four basic arithmetic operations.

Example 11: find $(72)_{10} \times (6)_{10}$

```
           1100
      ___________
110  |  1001000
         110
         ----
         0110
          110
          ---
          000
```

$(1100)_2 = (12)_{10}$