東北大學
Northeastern University

汇编语言程序设计

主讲： 刘松冉 （讲师）

单位： 东北大学计算机学院

智慧系统国际联合实验室

联系方式： liusongran@cse.neu.edu.cn

个人主页： http://faculty.neu.edu.cn/liusongran

https://liusongran.github.io/

第二章 计算机运算基础

一. 进位计数制

二. 数制之间的转换

三. 二进制编码

四. 带符号数的机内表示

五. 二进制运算

一. 进位计数制

进位计数制：按进位的方法进行计数

- **十进制**（Decimal）数： 数据尾部加一后缀D，如2355D
- **二进制**（Binary）数：数据尾部加一后缀B，如1011B
- **八进制**（Octal）数：数据尾部加一后缀Q，如62Q
- **十六进制**（Hexadecimal）数：数据尾部加一后缀H，如13ABH
  - 十六进制数由0~9的10个数码和A~F的6个字母组成，基数为16，按“逢十六进一”的原则进行计数。

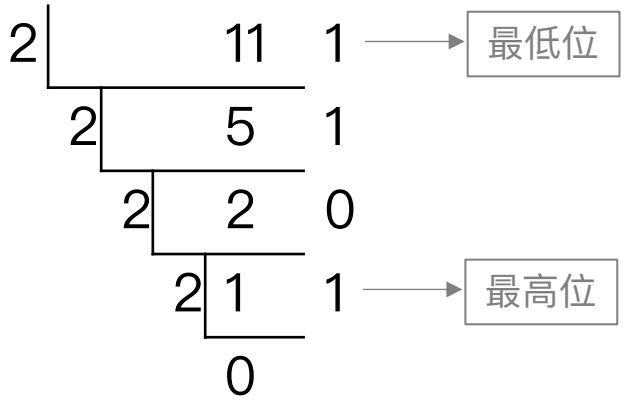
第二章 计算机运算基础

二. 数制之间的转换

1. 二进制与十进制之间的转换
2. 二进制与十六进制之间的转换
3. 十六进制与十进制之间的转换

1. 二进制与十进制之间的转换

- 十进制 → 二进制 （除2取余法）
  - 例：11D = 1011B



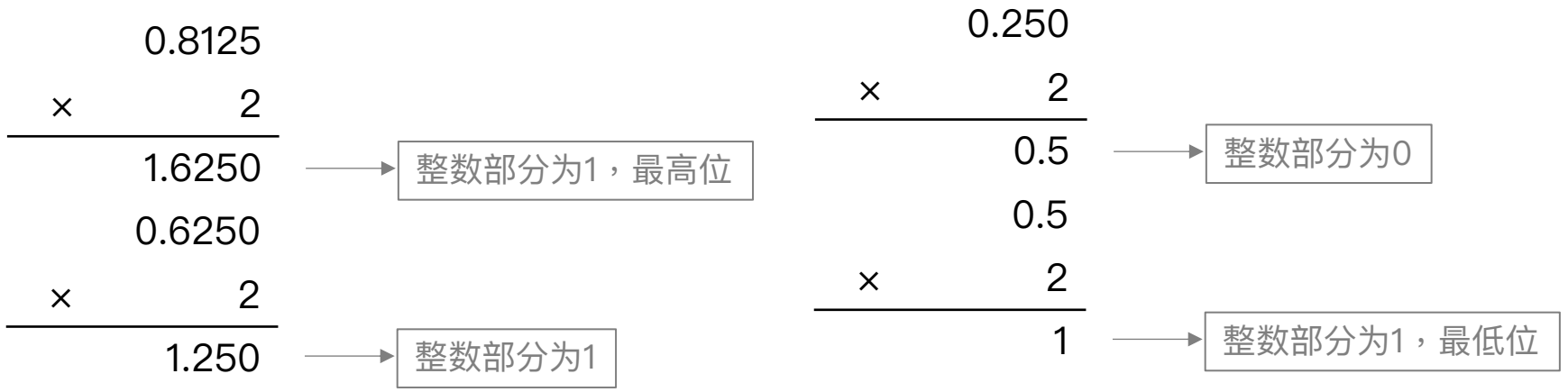
- 二进制 → 十进制 （将二进制数按“权”展开相加即可）
  - 例：10110.101B

$$= 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^2 + 0\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$$

$$= 22.625D$$

1. 二进制与十进制之间的转换

- 十进制小数 → 二进制小数 （乘2取整法）
  - 例：0.8125D = 0.1101B



2. 二进制与十六进制之间的转换

- 二进制 → 十六进制
  - 方法：将二进制数从小数点开始，分别向左向右4位分成一组，不足4位补0，然后写出对应的十六进制数即可。
  - 例： 10110.11B → 16.CH

- 十六进制 → 二进制
  - 方法：将每位十六进制数写出对应的4位 二进制数，然后去掉前导0和尾数0即可。
  - 例：3A.6H → 0011 1010.0110 = 111010.011B

3. 十进制与十六进制之间的转换

- 十进制 → 十六进制
  - 方法：采用“除16取余法”。
  - 例： 43D = 2BH

  - 十进制小数 → 十六进制小数 （采用“乘16取整法”）
  - 例：0.75D = 0.CH

- 十六进制 → 十进制
  - 方法：将十六进制数按“权”展开相加即可。
  - 例：2B.CH → 43.75D

第二章 计算机运算基础

三. 二进制编码

1. 二进制编码的十进制数（BCD，Binary Coded Decimal）
2. 字符编码

三. 二进制编码

1. 二进制编码的十进制数（BCD，Binary Coded Decimal）

- 例子：35.8的BCD码为：00110101.1000

BCD码表

BCD码	十进制数	BCD码	十进制数
0000	0	1000	8
0001	1	1001	9
0010	2	1010	这6种情况在BCD码中不允许出现
0011	3	1011	
0100	4	1100	
0101	5	1101	
0110	6	1110	
0111	7	1111	

2. 字符编码

- ASCII码（American Standard Code for Information Interchange）
- 国标码(区位码、异形国标码)
- Unicode 编码（UCS2）

2. 字符编码

- ASCII码（American Standard Code for Information Interchange）
  - 例子：“F”的ASCII码为46H。

十进制	➡	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
⬇	十六进制	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	空	►	空格	0	@	P	‘	p	Ç	É	á	▒	└	╨	α	≡
1	1	☺	◄	!	1	A	Q	a	q	ü	æ	í	▓	┴	╤	β	±
2	2	☻	↕	“	2	B	R	b	r	é	Æ	ó	▓	┬	╥	Γ	≥
3	3	♥	‼	#	3	C	S	c	s	â	ô	ú	│	├	╙	π	≤
4	4	♦	¶	$	4	D	T	d	t	ä	ö	ñ	┤	─	╘	Σ	⌠
5	5	♣	§	%	5	E	U	e	u	à	ò	Ñ	╡	┼	╒	σ	⌡
6	6	♠	▬	&	6	F	V	f	v	å	û	ª	╢	╞	╓	μ	÷
7	7	•	↨	’	7	G	W	g	w	ç	ù	º	╖	╟	╫	τ	≈
8	8	◘	↑	(	8	H	X	h	x	ê	ÿ	¿	╕	╚	╪	Φ	°
9	9	○	↓	)	9	I	Y	i	y	ë	Ö	⌐	╣	╔	┘	θ	∙
10	A	◙	→	*	:	J	Z	j	z	è	Ü	¬	║	╩	┌	Ω	·
11	B	♂	←	+	;	K	[	k	{	Ï	¢	1/2	╗	╦	■	δ	√
12	C	♀	∟	,	<	L	\	l	|	Î	£	1/4	╝	╠	∞	n
13	D	♪	↔	—	=	M	]	m	}	Ì	¥	i	╜	═	▐	φ	2
14	E	♫	▲	.	>	N	^	n	~	Ä	₨	<<	╛	╬	▌	∈	■
15	F	☼	▼	/	?	O	_	o	⌂	Å	ƒ	>>	┐	╧	▄	∩	BLANK

2. 字符编码

- **国标码（中华人民共和国国家标准信息交换用汉字编码字符集（基本集）GB 2312–80）**
  - 双字节表示，分为区码和位码
  - 包括202个一般符号：60个序号，22个数字，52个拉丁字母，169个日文假名，48个希腊字母，56个俄文字母，26个汉语拼音符号；37个汉语注音字母，6763个汉字
  - 6763个汉字分为两级。第一级汉字3755个按汉语拼音顺序排列；第二级汉字3008个按笔画顺序排列

2. 字符编码

- 国标码（中华人民共和国国家标准信息交换用汉字编码字符集（基本集）GB 2312–80）

GB 2312–80（一般符号）

1 区	00	01	02	03	04	05	06	07	08	09
00		(SP)	、	。	•	ˉ	ˇ	¨	〃	々
10	—	～	‖	…	‘	’	“	”	〔	〕
20	〈	〉	《	》	「	」	『	』	〖	〗
30	【	】	±	×	÷	∶	∧	∨	∑	∏
40	∪	∩	∈	∷	√	⊥	//	∠	⌒	⊙
50	∫	∮	≡	≌	≈	∽	∝	≠	≮	≯
60	≤	≥	∞	∵	∴	♂	♀	°	′	″
70	℃	$	¤	¢	£	‰	§	№	☆	★
80	○	●	◎	◇	◆	□	■	△	▲	※
90	→	←	↑	↓	=					

2 区	00	01	02	03	04	05	06	07	08	09
00	i	ii	iii	iv	v	vi	vii	viii	ix	x
10								1.	2.	3.
20	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.
30	14.	15.	16.	17.	18.	19.	20.	(1)	(2)	(3)
40	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)
50	(14)	(15)	(16)	(17)	(18)	(19)	(20)	①	②	③
60	④	⑤	⑥	⑦	⑧	⑨	⑩	€		㈠
70	㈡	㈢	㈣	㈤	㈥	㈦	㈧	㈨	㈩	
80		Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅴ	Ⅵ	Ⅶ	Ⅷ	Ⅸ
90	Ⅹ	Ⅺ	Ⅻ							

2. 字符编码

- **国标码（中华人民共和国国家标准信息交换用汉字编码字符集（基本集）GB 2312–80）**
  - 例子：汉字“爱”
    - 区位码：1614
    - 国标码：100EH+2020H
    - 异形国标码：B0AE=国标码+8080H

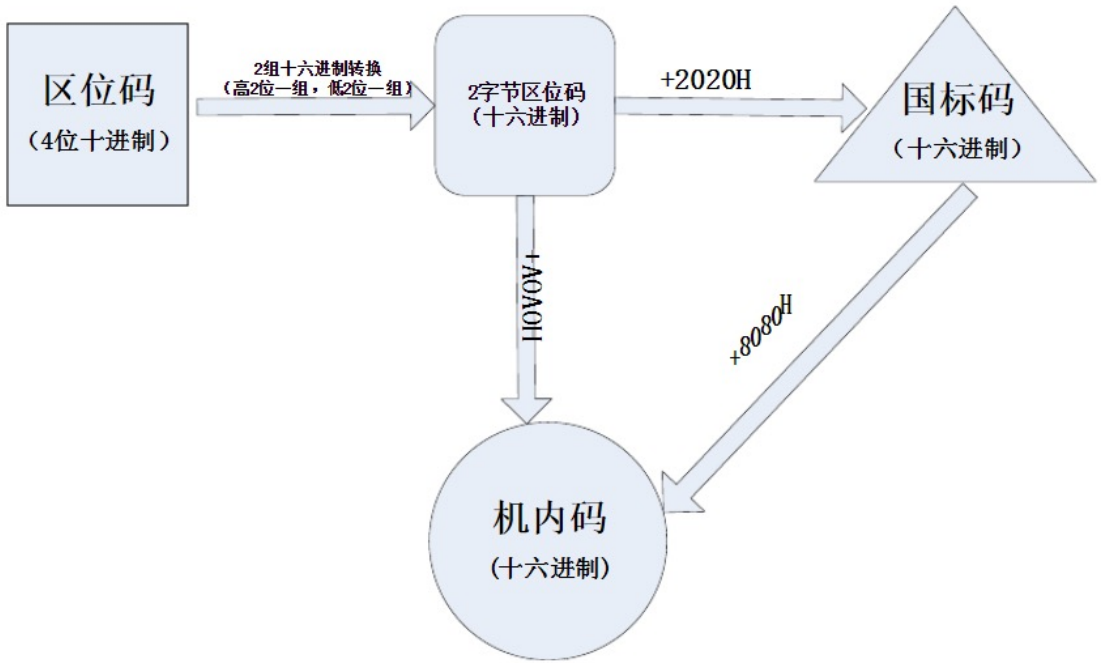
GB 2312–80（汉字）

16区	00	01	02	03	04	05	06	07	08	09
00		啊	阿	埃	挨	哎	唉	哀	皑	癌
10	蔼	矮	艾	碍	爱	隘	鞍	氨	安	俺
20	按	暗	岸	胺	案	肮	昂	盎	凹	敖
30	熬	翱	袄	傲	奥	懊	澳	芭	捌	扒
40	叭	吧	笆	八	疤	巴	拔	跋	靶	把
50	耙	坝	霸	罢	爸	白	柏	百	摆	佰
60	败	拜	稗	斑	班	搬	扳	般	颁	板
70	版	扮	拌	伴	瓣	半	办	绊	邦	帮
80	梆	榜	膀	绑	棒	磅	蚌	镑	傍	谤
90	苞	胞	包	褒	剥					

56区	00	01	02	03	04	05	06	07	08	09
00		亍	丌	兀	丐	廿	卅	丕	亘	丞
10	鬲	孬	噩	丨	禺	丿	匕	乜	夭	爻

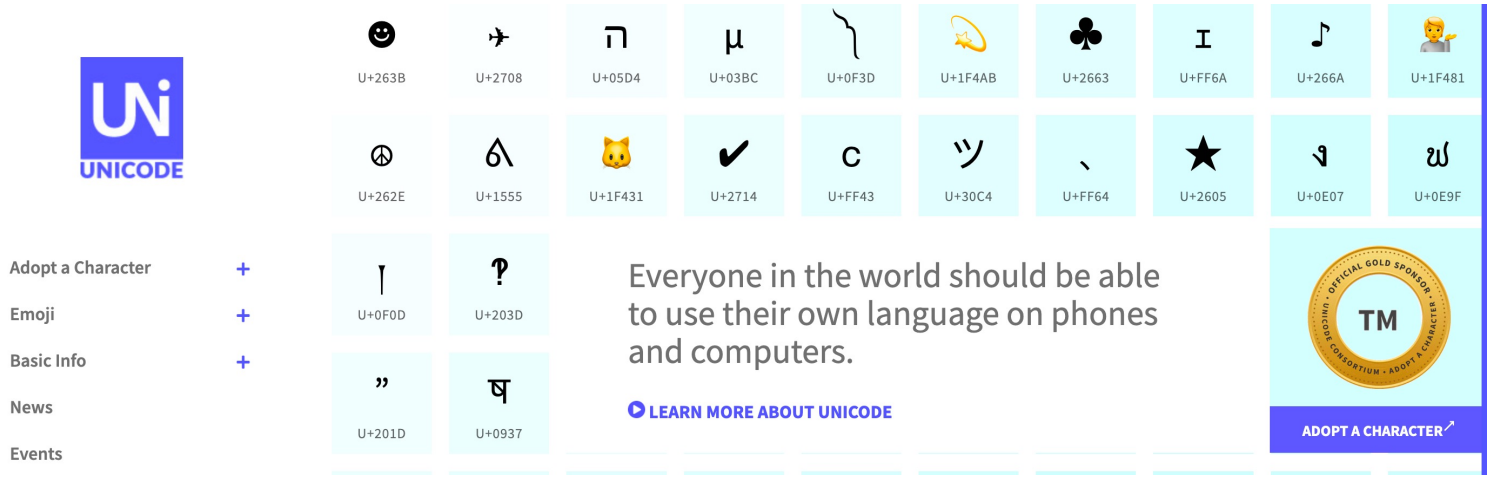
2. 字符编码

- 国标码（中华人民共和国国家标准信息交换用汉字编码字符集（基本集）GB 2312–80）
  - 例子：汉字“爱”
  - 汉字区位码、国标码（交换码）及机内码转换关系图



2. 字符编码

- Unicode 编码(UCS2)
  - Unicode 是一个16位的字符集，它包括了几乎所有常见的信息交换用的字符（英、法、德、中（简、繁）、朝鲜、日等），其64K的编码空间有大约1/3尚未分配。
  - 例：“A” 的 Unicode 是4100
  - 例：“爱”的 Unicode 是3172
  - 例：“愛”的 Unicode 是1B6



[1] https://home.unicode.org/

第二章 计算机运算基础

四. 带符号数的机内表示

1. 机器数与真值
2. 原码表示法
3. 反码表示法
4. 补码表示法
5. 过余码表示法

四. 带符号数的机内表示

1. 机器数与真值

- 计算机在处理实际问题时遇到的带符号数，数据的“+”号和“–”号在计算机内也是用二进制位表示，“0”表示正，“1”表示负。
- 例：数字N_1 = +1011011；数字N_2 = –1011011

  机内表示（机器数）：N_1 = 01011011；N_2 = 11011011
- 定义：将已经数值化了的带符号数称为**机器数**，而把原来的数称为机器数的**真值**。

2. 原码表示法

- 有符号数，数据的最高位用来表示符号,称为**符号位**，符号位为0表示正数，符号位为1表示负数，其余位为**数值位**，用数据的绝对值表示。
- 例：　　$X = +85$，$X_{原} = 01010101$

  　　$X = -85$，$X_{原} = 11010101$
- 对于0，有两种表示形式：$+0_{原} = 00000000$，$-0_{原} = 10000000$
- 8位二进制原码，能表示的有符号数的数据范围： $-127 \sim 127$

3. 反码表示法

- 在反码表示中，仍用0表示正数，1表示负数。对于正数，其反码表示与其原码表示完全相同；**对于负数**，符号位为1，**其余用数值的反码表示**。
- 例：　　$X = +85$，$X_{反} = 01010101$

  　　$X = -85$，$X_{反} = 10101010$
- 对于0，有两种表示形式：$+0_{反} = 00000000$，$-0_{反} = 11111111$
- 8位二进制反码，能表示的有符号数的数据范围： $-127 \sim 127$

4. 补码表示法

- 在补码表示中，仍用0表示正数，1表示负数。对于正数，其补码表示与其原码表示完全相同；**对于负数**，符号位为1，**其余各位按位取反加1**。
- 例：　　$X = +85$，$X_{补} = 01010101$

  　　$X = -85$，$X_{补} = 10101011$
- 对于0，只有一种表示形式：$0_{补} = 00000000$
- 8位二进制反码，能表示的有符号数的数据范围： –128~127
  - 在8位二进制补码表示法中，把符号位为1，数值位为0的编码10000000**规定为–128的补码**。

5. 过余码表示法

- 在过余码表示中，是将数据的真值直接与一个过余量相加，结果就是其过余码表示。过余量通常为64、128、1024等。
- 例：$X = +85$，$X_{过余128} = 128+85 = 10000000+01010101 = 11010101$

  $X = -85$，$X_{过余128} = 128-85 = 10000000-01010101 = 00101011$
- 对于0，只有一种表示形式。上例中，$0_{过余128} = 10000000$

6. 小节（原码、反码、补码间的相互关系）

- 对于正数X：$X_{原} = X_{反} = X_{补}$
- 对于负数X：三种编码规则各不相同

第二章 计算机运算基础

五. 二进制运算

1. 补码加减运算
2. 逻辑运算

1. 补码加减运算

- 补码的**加法运算**有如下的公式：$[X]_{补} + [Y]_{补} = [X+Y]_{补}$
  - **例1：两个正数相加**

		补码	原码
	45	00101101	00101101
+	22	00010110	00010110
	67	01000011	

1. 补码加减运算

- 补码的**加法运算**有如下的公式：$[X]_{补} + [Y]_{补} = [X+Y]_{补}$
  - 例1：两个正数相加
  - **例2：正数 + 负数**

	补码	原码
45	0010 1101	00101101
+ (−22)	1110 1010	10010110
23	00010111	

1. 补码加减运算

- 补码的**加法运算**有如下的公式：$[X]_{补} + [Y]_{补} = [X+Y]_{补}$
  - 例1：两个正数相加
  - 例2：正数 + 负数
  - **例3：负数 + 正数**

		补码	原码
	(–45)	1101 0011	1010 1101
+	22	0001 0110	0001 0110
	–23	1110 1001	

1. 补码加减运算

- 补码的**加法运算**有如下的公式：$[X]_{补} + [Y]_{补} = [X+Y]_{补}$
  - 例1：两个正数相加
  - 例2：正数 + 负数
  - 例3：负数 + 正数
  - **例4：负数 + 负数**

	补码	原码
(–45)	1101 0011	1010 1101
+ (–22)	1110 1010	1001 0110
–67	1011 1101	

1. 补码加减运算

- 补码的**加法运算**有如下的公式：$[X]_{补} + [Y]_{补} = [X+Y]_{补}$
  - 例1：两个正数相加
  - 例2：正数 + 负数
  - 例3：负数 + 正数
  - 例4：负数 + 负数
  - **结论**：用补码表示的数据进行加法运算时可以不考虑符号位，直接运算， 即与不带符号的数据的运算完全相同。

1. 补码加减运算

- 补码的**加法运算**有如下的公式：$[X]_{补} + [Y]_{补} = [X+Y]_{补}$
  - 例1：两个正数相加
  - 例2：正数 + 负数
  - 例3：负数 + 正数
  - 例4：负数 + 负数
  - **结论**：用补码表示的数据进行加法运算时可以不考虑符号位，直接运算， 即与不带符号的数据的运算完全相同。
    - **示例**

		补码	原码
	126	0111 1110	0111 1110
+	4	0000 0100	0000 0100
	130	1000 0010	

1000 0010 → –126的补码 ?

1. 补码加减运算

- 补码的**加法运算**有如下的公式：$[X]_{补} + [Y]_{补} = [X+Y]_{补}$
  - 例1：两个正数相加
  - 例2：正数 + 负数
  - 例3：负数 + 正数
  - 例4：负数 + 负数
  - **结论**：用补码表示的数据进行加法运算时可以不考虑符号位，直接运算， 即与不带符号的数据的运算完全相同。
    - **示例 – 加法溢出**
      - 运算结果超出了目标所能容纳的范围，称发生了溢出。
      - 例如：8位所能表示的补码数据的范围是: –128~+127。
      - 同号相加，异号相减时才可能发生溢出。即：两个同号数相加，结果的符号位和运算数的符号位不同，则发生了溢出。

		补码	原码
	126	0111 1110	0111 1110
+	4	0000 0100	0000 0100
	130	1000 0010	→ –126的补码

1. 补码加减运算

- 补码的**减法运算**有如下的公式：$[X]_{补} - [Y]_{补} = [X-Y]_{补} = [X]_{补} + [-Y]_{补}$
  - **例：**

		补码	原码
	56	0011 1000	0011 1000
−	34	0010 0010	0010 0010
	22	0001 0110	

		补码	原码
	56	0011 1000	0011 1000
+	(−34)	1101 1110	1010 0010
	22	0001 0110	

1. 补码加减运算

- 补码的**减法运算**有如下的公式：$[X]_{补} - [Y]_{补} = [X-Y]_{补} = [X]_{补} + [-Y]_{补}$
  - **减法溢出**
    - 运算结果超出了目标所能容纳的范围，称为溢出。
    - 只有两个异号数相减时，才有可能发生溢出。即：两个异号数相减，结果的符号位和被减数的符号位不同，则发生了溢出。
    - 对–128取补，会发生溢出。

2. 逻辑运算（按位运算）

- 与(∧)：对应位同时为“1”
- 或(∨)：对应位，至少有一个为“1”
- 非(¬)：取反
- 异或(⊕)：对应位相同为“0”，不同则为“1”

运算规则表

第一操作数	第二操作数	与运算	或运算	异或运算
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

2. 逻辑运算（按位运算）

- 例1：56H ∧ 3FH
- 例2：56H ∨ 3FH
- 例3：56H ⊕ 3FH
- 例4：¬ 56H

2. 逻辑运算（按位运算）

- 例1：56H ∧ 3FH = 16H
- 例2：56H ∨ 3FH = 7FH
- 例3：56H ⊕ 3FH = 69H
- 例4：¬ 90H = 6FH