WEIJI YUANLI
YU JIEKOU JISHU
FUXI ZHIDAO
HE XITI JIEDA

微机原理与接口技术复习指导和习题解答

王玉良　吴晓非　张琳　禹可　编著

北京邮电大学出版社
www.buptpress.com

微机原理与接口技术复习指导和习题解答

王玉良　吴晓非　张琳　禹可　编著

北京邮电大学出版社
·北京·

内容简介

本书是《微机原理与接口技术(第2版)》(王玉良、吴晓非、张琳、禹可编著)一书的辅助教材。全书共分8章,每章内容由三部分组成,分别是知识要点、例题详解、习题及参考解答。知识要点给出了该章的复习要点、难点及必须掌握的内容;例题详解对各类题型作了详尽的解答;习题及参考解答是对上面一书各章后面的所有习题都给出了参考答案。书后附录提供了几套期末考试的模拟试题及答案。

本书可作为相关专业本科生该课程学习和复习的指导书,也可作为报考硕士研究生的复习辅导书,还可作为讲授《微机原理与接口技术》的有关教师的教学参考书。

图书在版编目(CIP)数据

微机原理与接口技术复习指导和习题解答/王玉良等编著. —北京:北京邮电大学出版社,2006(2007.7重印)
ISBN 978-7-5635-1353-6

Ⅰ.微... Ⅱ.王... Ⅲ.①微型计算机—理论—高等学校—教学参考资料②微型计算机—接口—高等学校—教学参考资料 Ⅳ.TP36

中国版本图书馆CIP数据核字(2006)第131078号

书　　名:微机原理与接口技术复习指导和习题解答
作　　者:王玉良　吴晓非　张琳　禹可
责任编辑:王晓丹
出版发行:北京邮电大学出版社
社　　址:北京市海淀区西土城路10号(100876)
北方营销中心:电话:010-62282185　传真:010-62283578
南方营销中心:电话:010-62282902　传真:010-62282735
E-mail:publish@bupt.edu.cn
经　　销:各地新华书店
印　　刷:北京市梦宇印务有限公司
开　　本:787 mm×1 092 mm　1/16
印　　张:12.75
字　　数:316千字
印　　数:3 001—6 000册
版　　次:2006年12月第1版　2007年7月第2次印刷

ISBN 978-7-5635-1353-6/TP·261　　定价:19.00元

前　言

本书是我们编著的《微机原理与接口技术(第2版)》(王玉良、吴晓非、张琳、禹可编著)一书的配套辅助教材。微机原理与接口技术课程是信息工程、电子工程、通信工程等专业的一门专业基础课,是上述专业本科生的必修课和考试课。该课程对本科学生在微型计算机方面的基础知识、基本原理、接口技术以及综合运用软硬件能力都有较高的要求。此书旨在帮助学生在期末复习时,能在较短时间内理解本课程的主要内容,了解各章的重点,熟悉各种题型,掌握解题技巧。

全书共分8章,每章内容由三部分组成,分别是知识精要,例题详解,习题及参考解答。知识精要给出了该章的复习要点、难点及必须掌握的内容;例题详解对各类题型作了详尽的解答,有些题型还根据知识点进行了必要的讨论;习题及参考解答是《微机原理与接口技术(第2版)》各章后面的习题的参考答案。书后附录提供了两套前几年期末考试的试题及答案。根据教学大纲要求,有些内容和习题不是必须掌握的,用*标识,只作为参考与了解。

本书第1、2章由张琳编写,第3、5章由禹可编写,第4、6章由吴晓非编写,第7、8章由王玉良编写。王玉良负责全书的统稿。编者是在教学一线长期从事微机原理与接口技术课程教学的教师,在编写时力求基本概念、基本原理清晰,知识重点突出,例题有代表性,文字通俗易懂。

本书可作为相关专业本科生该课程学习和复习的指导书,也可作为报考硕士研究生的复习辅导书,还可作为讲授微机原理与接口技术课程的有关教师的教学参考书。

由于编者水平有限,错误和不妥之处,敬请读者与专家批评指正。

编　者

2006年10日

目　录

第1章　微型计算机的基础知识

1.1　知识精要

计算机是一种高速信息处理和信息传递的工具，其操作对象是信息（或称数据），处理（或称加工）的结果，也是信息（或数据）。

计算机之所以获得广泛的应用，是因为其运算速度快，精度高，可以不间断地自动连续工作。有了计算机的介入，从根本上改变了人们生产、科学研究、生活、学习和事物管理的水平和质量。

实践证明，学好本课程并不是轻而易举的，应从总体上掌握各部分内容在本课程中的地位和目标，即，为什么要讲这一部分？它的原理是什么？和其他部分的关系是什么？总之，在复习的过程中，更强调融会贯通。

1.1.1　计算机的基本构成

计算机包括硬件和软件两部分。硬件的各个部分的工作互相关联。

1. 硬件

硬件包括主机和外围设备。

- CPU：计算机的控制和加工中心。
- 存储器：记忆信息，包括命令信息和被加工的信息与加工结果。前者表示加工的方法，后者是被加工的对象和结果。前者能被计算机理解和执行（存放在“程序区”或称“码区”），后者供人或其他机器（设备）所理解和接受（存放在存储器的“数据区”）。
- I/O 接口：总线和相应外设之间的适配电路的总称。
- 总线：一组公用信息的传送通路。总线结构不仅使系统的连线数大大减少，便于系统配置和扩充，而且使接口设计规范化。
- 外围设备：显示器、键盘、鼠标、打印机、扫描仪及其他设备。

2. 软件

软件包括系统软件和应用软件。

- 系统软件：管理系统的硬件和软件资源，如 CPU、存储器、I/O 设备、软件的调度和运行、编译程序、汇编程序等等。
- 应用软件：在给定应用场合下，管理相应的硬件和软件以及用户数据的有关软件包。

硬件和软件的结合，构成了完整的计算机系统。硬件是计算机的物质基础，而软件是计算机的灵魂，这两者缺一不可。计算机硬件，只有在相应软件的控制下，才能“活”起来，才可完成给定的任务，成为对人们真正有用的工具。

1.1.2 计算机的工作原理

下面，通过一个简单的例子来说明计算机的工作过程和原理。

1. 计算机的基本工作过程

上电后，CPU 做如下工作：

- 取指令(从 PC 复位值的地址处)；
- 分析指令(译码)；
- 执行指令；
- 再取下条指令，重复上述操作。

2. 以执行 s=a+b，说明计算机的工作过程

为完成这个计算任务，必须先做如下工作：

- 编写完成该任务的程序(源程序)；
- 输入计算机(通过适当的外围设备和 I/O 接口，经总线到存储器)；
- 编译、检错、纠错，编译通过后，变成“机器码”，放在内存的程序区；
- 运行程序；
- 输出结果。

能完成 s=a+b 的一种汇编程序如下：

```
;以下是数据区
a       DB      56H     ;被加数单元
b       DB      24H     ;加数单元
s       DB      0       ;和单元
;以下是程序区
START:  MOV     A,[a]   ;取被加数
        ADD     A,[b]   ;加加数
        MOV     [s],A   ;存和
        HLT             ;停机
        END START       ;编译结束
```

程序经输入设备输入计算机，再经编译程序编译后，若没有错误，就把它翻译成计算机能识别并执行的相应二进编码指令(机器码)。运行“START”程序时，机器将自动从标号 START 开头的语句(由程序计数器 PC 指向它)开始运行。由于指令执行过程的重要性，这里重复说明如下：

(1) 取第一条指令

把 PC 指向的地址送“地址总线”，并发出“存储器读”命令，内存收到该命令，就把相应单元的内容(指令的机器码)读出送到数据总线，CPU 接收它，并把它存入指令寄存器 IR。在取指令后，PC 自动增加并指向下条指令。

该指令经译码后，识别出是从存储器读取数据，并写到累加器 A，就自动启动下一条总

线操作以实现该指令的功能。

控制器把指令中 a 的地址送地址总线，同时发出“读存储器”命令；存储器收到命令后，就读出相应地址单元的内容（操作数 a）送到数据总线；CPU 从数据总线接收该数据，再送到累加器 A，就完成了第一条指令的操作。

这里，取指令操作和读数据操作，对存储器来说，没有什么差别，但对 CPU 来说，它们是完全不同的操作：取指令取出的是能指导计算机工作的“指令”，它要被送到 IR 去执行；而取数据操作，取出的只是操作对象而已。这种差别，CPU 的控制器完全清楚，并能控制它到达该去的地方，这是由指令译码和“微命令发生逻辑”生成的微命令控制的结果。

（2）下一周期的取指令

与前面一样，把 PC 指示的指令（第二条指令）取出来，送到 IR，PC 自动指向再下一条指令。

“指令译码”阶段，使控制器知道它是一条加法指令，且一个加数在 A 内，一个加数在存储器内，它自动生成相应命令，完成该指令的操作。

首先，它把指令中 b 的地址送地址总线，并发出读存储器命令；存储器接到该命令，就把操作数 b 送到数据总线；CPU 接收该数据，并把它暂存在 TEMP 寄存器；其次，CPU 把 A 和 TEMP 的内容在“运算器”相加，并把“和”送到累加器 A，同时改变状态寄存器的标志，如 C、N、O、Z、A、P 等（这些标志的含义和作用，将在本章后面的“运算基础”中介绍）。

（3）第三条指令的取指令

译码与前面类似，执行过程中，把 A 的内容写到 s 单元：CPU 把 s 的地址送地址总线，把 A 内容送到数据总线，并发出“存储器写”命令，存储器接收到该命令后，就把数据总线上的数据写到相应地址单元中去。

（4）再下条指令是“停机”指令

表示程序到此结束，以免程序继续向下执行，把不是指令的其他数据也当作指令执行，这将会造成系统的错误动作，产生难以预料的结果。当然，除了用 HLT 指令使程序停止执行外，还有其他方法使用户程序运行终止，在学习本课程后，就会知道。

程序运行的结果是否正确，可以通过输出设备，把单元 a，b，s 的内容输出到输出设备（比如显示器）来检查和验证。若结果不对，可用软件工具来检测调试，必要时，可以逐条执行指令，检查每条指令的执行情况，直到找出并纠正存在的错误，得出正确结果为止。

上述操作是 CPU 经常性的基本操作。这些操作都是在 CPU 控制器的控制下完成的。

3. CPU 对其他外部事件作出响应

CPU 除了上述运行程序之外，还有对以下外部事件作出相应响应的能力。

- RESET：系统复位（从错误中恢复的手段）；
- 内部或外部意外事件的处理请求（中断或陷阱）；
- 总线请求：其他主设备请求使用系统总线；
- 联络控制：保证信息在机器内高效、可靠地传送。

此外，现代计算机的 CPU 内还加入了某些大中型计算机才使用的技术，如浮点运算部件、存储器管理部件、高速缓存、指令和数据队列等。这些都是为了提高计算机的性能（如提高运行速度）和处理复杂事物的能力（如管理更大容量的存储器）而加入的。

1.1.3 计算机的特点和应用

1. 特点

- 速度快；
- 处理能力强大，几乎无所不能；
- 能自动地连续工作。

2. 应用

- 信息处理
- 传输控制
- 科学计算
- 实时控制（过程控制）
- 决策帮助
- 辅助设计

1.1.4 计算机运算基础

1. 进位计数制及二进制计算

数值 N 可表示为

$$N=\sum_{i=-m}^{i=k} d_i R^i$$

式中，数字 N 有 m 位小数，$k+1$ 位整数，d_i 是每位数字的取值，R 是该数的基数（Radix），是大于或等于 2 的整数，R^i 称为 d_i 位的权，或者位值。基数是每位数字的取值范围〔0～$(R-1)$〕。$R=10$，称为十进制数；$R=2$，称为二进制数。

计算机内主要用二进制表示和运算。而四进制、八进制或十六进制等 2 的整数幂进位计数制，只是作为二进制的缩略表示而经常使用。不同进位计数制间的转换是经常进行的变换，应熟练掌握它们的变换原理和方法。

2. 无符号数的运算

加、减、乘、除是最基本的运算，应熟练掌握。这里，应该知道如何判断运算的结果：正确还是不正确？是否为零？

3. 符号数的表示和运算

符号数也是用二进制数表示和运算的。通常用一位二进制数表示数的符号（+ 或 −）。

（1）原码表示（若用 k 位表示）

$$[n]_{原}=\begin{cases} n & n\geqslant 0 \\ 2^{k-1}+|n| & n\leqslant 0 \end{cases}$$

表示范围：$(-2^{k-1}+1)\sim(+2^{k-1}-1)$。

原码表示可进行数值的乘、除运算，即，先进行数值计算，再把结果的符号加上。

（2）反码表示和运算（k 位表示）

$$[n]_{反}=\begin{cases} n & n\geqslant 0 \\ 2^{k}-1+n & n\leqslant 0 \end{cases}$$

表示范围：$(-2^{k-1}+1)\sim(+2^{k-1}-1)$。

数的反码可用于加法和减法运算。

- 加法：$Z=X+Y+$循环进位
- 减法：$Z=X+(-Y)+$循环进位

其中，X、Y、Z 是反码表示的数，循环进位是 $X+Y$ 产生的进位，再加到结果的最低位。

(3) 补码表示和运算

$$[n]_{补}=\begin{cases} n & n\geqslant 0 \\ 2^k+n & n\leqslant 0 \end{cases}$$

若引入模数的概念，则有

$$n=2^k+n \mod 2^k$$

表示范围：$-2^{k-1}\sim(2^{k-1}-1)$。

利用补码可进行加、减、乘、除运算，且

$$Z=X+Y \mod 2^k$$
$$W=X+(-Y)=X-Y \mod 2^k$$

其中，X、Y 是补码表示的数，Z、W 是结果的补码。

符号数的运算和无符号数的运算方法基本是一样的。只有如下差别：

无符号数运算的进位表示结果溢出，而符号数运算产生的进位应该舍弃，其结果溢出有不同的判别方法：

若正数＋正数(或正数－负数)得负数；或者负数＋负数(或负数－正数)得正数，则表明结果溢出。

因此，若运算结果超出了结果单元能表示的范围，则产生溢出。这必须用其他方法来指示。这就是“标志位”的用途。通常，

- 用 Z 标志表示结果是否为 0；
- 用 O 标志表示符号数溢出(算术溢出)；
- 用 S 标志表示结果是正或负数；
- 用 C 表示结果的最高位是否产生了进位(无符号数溢出)。

显然，它们都是逻辑变量，且“1”表示真，“0”表示假。

4. 十进制数的表示和运算

(1) 要表示一位十进制数，至少要用四位二进制数，常用两种方法表示。

① 压缩的 BCD 数，一字节表示两位 BCD 数。

② 非压缩的 BCD 数，这又有两种表示方法：

- ASCII 码表示，一字节存放一位十进制数；
- 非 ASCII 码表示，一字节存放一位数字，高位是 0。

(2) 十进制运算采用和二进制完全相同的办法，但计算结果要修正为十进制数。修正原则如下：

对压缩的十进制数，若相加结果：

- 低位是非十进制数或产生进位，则在低位加 6 修正；
- 若高位产生进位或是非十进制数字，或者高位结果是 9，但低位作＋6 修正，则在高位加 6 修正。

所以，在十进制运算中，低 4 位向上是否有进位或者借位，应有指示，这就是设置半进位

H 或辅助进位 A，或(十六进制或十进制)数字进位 DC 的作用。标志 A 和 C 是用来作十进制运算的。

对于非压缩的十进制数(包括 ASCII 数字)运算，高位进位没有意义，所以，其低位进位 A 应传递到 C 标志，以便向高位结果传送(用连加指令 ADC 或连减指令 SBB)。

5. 逻辑变量的表示和运算

1 比特可完全表示 1 个逻辑变量，因此，逻辑运算都是比特运算。

逻辑运算中，C、O、A、S 标志均无意义。但 8086 和其他的 CPU 在逻辑运算中都改变 S 标志，这时，它只表示最高位逻辑运算的结果，没有其他含义。这种情况下，可利用逻辑运算指令把 O、C、A 置成某种状态："0"或"1"。

逻辑运算改变 P 标志，指示运算结果"1"的个数的奇偶性。

6. 文字在计算机内的表示和操作

计算机中的文字也用二进制编码表示(外文或汉字)。除了文字外，如下内容也用二进制编码表示。

- 数字：0，1，…，9；
- 字母：A～Z，a～z；
- 标点和符号：！，。，？，：，"，"，'，'，{}，[]，()，$，¥，#，& 等。

以上都是打印字符，而设备和文件(记录)控制符、传输控制符等是不可打印字符，它们起到某种控制功能。

为了提高传输的可靠性，经常在被传信息中加入"多余(冗余)"信息，以便收端能验证信息的正确或错误。最简单、最常用的方法是奇偶校验。因此，标志中加入了 P 标志。

***7. 浮点数的表示和运算**

(1) 浮点数的表示

前面的例子都是用定点整数表示法，即，数的小数点位置是固定的，在数的最低位后面。此外，还有定点小数表示法，就是小数点在符号位的后面。

而浮点数的小数点位置是不固定的，它随每个数的大小而变化。浮点数的一般表示如下：

$$N = M \cdot R^P$$

其中，M 是数 N 的尾数部分，而 R 是数的基数，P 是阶码。

计算机内浮点数的基数通常是 2，但也有以四位二进制数为一组，而用 $R=16$ 为基数的。在基数确定后，浮点数就完全由尾数和阶码决定了。

M 和 P 都包括数值和符号。尾数的符号就是该数的符号；而阶码的符号表示小数点在尾数部分的位置，如下所示：

S_P	P	S_M	M

其中，S_P 是阶码 P 的符号位，S_M 是尾数 M 的符号位。

各计算机厂家生产的机器的浮点表示法各不相同，尤其是阶码的表示方法和"规格化"浮点数的定义。以下是关于浮点数的一些概念。

- 规格化：移动小数点的位置，使其尾数变成其标准格式的过程。

• 对阶：移动一个操作小数点的位置，使两个数的小数点对齐（阶码相同）的过程。

下面以 IEEE 浮点数表示法为例，说明浮点数的表示和运算规则，以便大家对它有一个基本了解。

IEEE 浮点数分单精度和双精度两种，单精度为 32 位，双精度是 64 位。单精度的阶码用 8 位表示（连符号位），尾数连符号 24 位。双精度的阶码占 11 位，尾数连符号位共 53 位。

单精度格式如下所示：

b_{31}	b_{30} ~ b_{23}	b_{22} ~ b_0
S_M	阶码(8 位)	尾数(23 位)

双精度格式

b_{63}	b_{62} ~ b_{52}	b_{51} ~ b_0
S_M	阶码(11 位)	尾数(52 位)

IEEE 浮点数规定：规格化浮点数的小数点在符号位 S_M 的后面，且小数点前有一个“1”。即，其尾数可表示 1～2 之间的数。其阶码连同阶码的符号统一编码，对单精度数，阶码＝0 表示 2^{-127}，7FH 表示 2^0，FFH 表示 2^{128}。它所能表示的绝对值最小的数是

$$\pm 1\times 2^{-127}=\pm 5.877\times 10^{-39}$$

若小于该数，则表示为机器零，用全 0 表示。它能表示的绝对值最大的数是

$$\pm 2\times 2^{128}=\pm 6.8\times 10^{38}$$

若超出这一范围，则产生上溢出。

下面是一些单精度数值表示的举例：

① $+1.0=1.0\times 2^0$，表示为

0011,1111,1000,0…0B＝3F800000H

② $-3.0=-1.5\times 2^1$，表示为

1100,0000,0100,…0B＝C0400000H

③ $-128.625=-1.00000000001\times 2^7$，规格化表示为

1100,0011,0000,0000,0001,00…0B＝C3001000H

对于双精度数，其阶码有 11 位，当阶码是 3FFH 时，表示 2^0，当阶码为 0 时，表示 $2^{-1\,023}$，阶码是 7FFH 时，表示 $2^{+1\,024}$。

它所能表示的绝对值最小的数是

$$\pm 1\times 2^{-1\,023}=\pm 8.99\times 10^{-307}$$

它所能表示的绝对值最大的数是

$$\pm 2\times 2^{1\,024}=\pm 3.595\times 10^{303}$$

(2) 浮点数的运算

① 浮点加/减法的运算步骤是：

• 对阶，使两个操作数的小数点对齐（阶码相同）；

• 尾数相加减；

• 结果规格化（注意小数点前面有一位隐含的 1）。

② 浮点乘/除运算：

• 尾数相乘/除，得出积/商及余数的尾数；

• 阶码相加/减，得出积/商的阶码；
• 对积/商和余数规格化；
• 将它们的尾数加上符号位。

③ 浮点数的特点

与定点数相比，浮点数有如下特点：
• 浮点数表示的数值范围大；
• 一般来说，浮点数运算的精度高；
• 浮点运算比定点运算复杂，它所需的硬件设备也多。

1.2 例题详解

1.2.1 数制变换

例 1.1 求 352.34=_____B=_____Q=_____H。

解 人工变换时，可先变换成八进制，再变为其他进制数。用计算器计算时，用十六进制可减少变换次数。

整数部分

352/8=44 余 $0=d_0$

44/8=5 余 $4=d_1$

5/8=0 余 $5=d_2$

小数部分(精确到小数点后 3 位)

.34×8=2.72 取整 $d_{-1}=2$

.72×8=5.76 取整 $d_{-2}=5$

.76×8=6.08 取整 $d_{-3}=6$

所以 352.34=540.256Q=101100000.010101110B=160.57H。

例 1.2 求 702.57Q=_____B=_____D=_____H。

解 二进制和十六进制可直接变换，而十进制可用常规方法或者根据定义变换。

$$702.57Q=111000010.101111B=1C2.BCH$$

根据定义变换：

$$702.57Q=7\times8^2+2\times8^0+5\times8^{-1}+7\times8^{-2}=450.734$$

也可以根据变换方法，按八进制作乘 12Q 或除 12Q 运算。

整数部分

702/12Q=55 余数 $d_0=0$

55/12Q=4 余数 $d_1=5$

$d_2=4$

小数部分(精确到小数点后三位)

.57×12Q=7.26Q 整数 $d_{-1}=7$

.26×12Q=3.34Q 整数 $d_{-2}=3$

.34×12Q=4.3Q 整数 $d_{-3}=4$

所以，702.57Q=450.734。

注意：这里的乘除都是八进制运算。

1.2.2 二进制运算及结果分析

计算机对数据运算(加工)时，除了要关心运算结果外，还要注意标志位的变化，这也是运算结果的一部分。

例 1.3 按 6 位运算器计算 2DH+19H=?

解

2DH	101101B
+.19H	+.011001B
06H	000110B

最高位有进位，C=1(结果溢出)；

结果不等于 0，Z=0；

低位数字没有产生向上进位，A=0。

例 1.4 按 8 位计算器计算 7CH−D5H=?

解

7CH	01111100B
−.D5H	−.11010101B
A7H	10100111B

最高位有借位，C=1(无符号数结果超出表示范围)；

结果不等于 0，Z=0(非 0 结果)；

低位数字没有产生向上借位，A=0。

若用被减数的补码按加法运算，则有如下过程：

被减数 D5H 的补码是 2BH，所以有

7 CH	0111 1100B
+ 2.BH	+ 0010.1011B
A 7H	1010 0111B

最高位无进位(有借位)，C=1(结果超出表示范围)；

结果不等于 0，Z=0(非 0 结果)；

低位数字产生向上进位(无借位)，A=0。

1.2.3 符号数运算

若把符号数中的符号位也当作一位数字，则可把整个数字按无符号数运算，结果是一样的。但要注意：因为是“模数运算”，所以进位/借位标志没有意义。同时，增加了“算术溢出”标志 O(或 V)和负数标志 N(或 S)。

例 1.5 用 8 位运算器计算 49H+D8H=?

解

49H	0100 1001B
+ D8H	+ 1101 1000B
21H	0010 0001B

因为正数加负数得正数，没有溢出，O=0；

结果为正，N=0；

结果不为 0，Z=0；

最高位有进位，C=1(只在连加/减运算中才有用)；

低位有进位，　　A=1。

所以，结果 = 21H = 00100001B(+33 的补码)。

标志 NZOCA=00011。

例 1.6　用 8 位运算器计算 49H−D8H=？

解　直接用减法计算：

49H	0100 1001B
− D8H	− 1101 1000B
71H	0111 0001B

结果为正，　　N=0；

正数减负数，结果为正，没有溢出，　　O=0；

结果不为 0，　　Z=0；

最高位有借位，　　C=1；

低位无借位，　　A=0。

所以，结果= 71H = 01110001B (+133 的补码)。

标志 NZOCA=00010。

例 1.7　有 12 位补码数 X=3FCH 和 8 位补码数 Y=9AH，计算 X+Y 和 X−Y=？

解　这里要注意，加减运算时，两个数的符号位要对齐。所以，对 8 位数要进行“符号扩展”，变为 Y=F9AH。

3FCH (1020)	0011 1111 1100B
+ F9AH(−102)	+ 1111 1001 1010B
396H	0011 1001 0110B

结果为正，　　N=0；

正+负，没有溢出，　　O=0；

结果不为 0，　　Z=0；

最高位有进位，　　C=1；

低位有进位，　　A=1。

所以，结果 = 396H(+918 的补码)，

标志 NZOCA=00011。

3FCH (1020)	0011 1111 1100B
− F9AH(−102)	− 1111 1001 1010B
462H	0100 0110 0010B

结果为正，　　N=0；

正−负=正，没有溢出，　　O=0；

结果不为 0，　　Z=0；

最高位有进位，　　C=1；

低位没有进位，　　A=0。

(若忘了对符号数进行符号扩展，结果肯定出错。而无符号数的运算是无需符号扩展的。)