微计算机丛书

〔日〕中村和夫 井出裕巳 著
陆玉库译

INTEL 8086 微处理器应用入门

電子工業出版社

INTEL 8086

微处理器应用入门

〔日〕中村和夫 井出裕巳 著
陆玉库 译

電子工業出版社

内 容 简 介

本书主要译自〔日〕《インターフェース》杂志，1982年第8期所刊载的INTEL 8086特集；部分内容选译自〔日〕オーム社出版的《8086の使い方》。

全书包括：8086的结构、指令系统、外围芯片、硬件设计、汇编语言程序设计、多总线及开发系统等内容。书后附录有INTEL 8086的指令详解。

本书内容充实、文字简明扼要、图表丰富。对于已了解8位微计算机的读者来说，该书是学习8086微计算机的一种入门读物；并可作为8086指令参考手册。

本书可供从事16位微计算机设计和使用的科技人员及大专院校有关专业师生阅读和参考。

INTEL 8086
微处理器应用入门
〔日〕中村和夫　井出裕巳著
陆　玉　库　译
※
电子工业出版社出版
中国科技情报所印刷厂印刷
新华书店北京发行所发行
各地新华书店经售
※
1983年6月第一版　开本：850×1168 32开
1983年7月第一次印刷　印张：11 1/8
印数：1—25,000册　字数：294千字
统一书号：15290·3　本社书号：P31·03
定价：1.60元

译 者 的 话

自从美国INTEL公司于1971年制造出了单片的4位微处理器4004以来，微处理器已经历了十多年的发展历程。在这十多年的过程中，微处理器不仅在各个领域里得到了广泛的应用，而且微处理器本身不断更新换代，表现了它的强大生命力。继4位、8位微处理器之后，INTEL公司于1978年又首先发表了16位微处理器8086，此后ZILOG公司、MOTOROLA公司又相继发表了16位微处理器Z8000（1979）、MC68000(1980)，使微处理器应用进入了一个新阶段。由于16位微处理器不仅具备以前微处理器所有优点，还吸取了不少小型计算机结构设计上的特点，从而使它的处理能力进一步加强，使微型机和小型机差别日趋减小，并有取代小型机的趋势。

目前，我国在广泛应用4位和8位微计算机的基础上已开始重视16位微机的研制开发和应用。但是在实际工作中深感资料不足。译者根据当前国内微机发展形势的需要，编译了《Intel8086微处理器应用入门》一书。如果本书能对于从事微机研制和应用的技术人员有所启发和帮助，为我国微机的发展起到点滴作用的话，将是译者的最大荣幸。

本书1～6章译自日文杂志《インターフェース》1982年第8期；7、8、9章选译自〔日〕井出裕巳著《8086の使い方》一书。

北京工业大学计算中心李礼贤副教授对译文作了校订，并提供了附录资料，在此谨表谢意。

由于译者水平有限，缺乏实践经验，不妥之处，谨請读者批评指正。

1983.4.

前　言

本书是以打算采用8086微处理器的人为对象，以8086的结构、指令系统、硬件设计、汇编语言程序设计及多路总线等在系统设计中应注意的事项为主要内容，并加以详细的说明。

书中所列举的CP/M操作系统，多总线为 DIGITAL RESEACH公司和INTEL公司的专利。

中村和夫

目　录

附录 1　8086 指令详解

（按英文字母顺序排列）

AAA

(ASCII adjust for addition)——加法的ASCII修正指令

操作： 若AL的低 4 位大于 9 或辅助进位标志AF被置为“1”，则往AL中加 6，往AH中加 1，AF与CF标志均被置位。结果AL中新的数值的高 4 位应为“0”，而低 4 位是上述加法所得出的数（0～9 之间的数）。

```
if ((AL) & 0FH) > 9 or (AF) = 1 then
  (AL) ← (AL) + 6
  (AH) ← (AH) + 1
  (AF) ← 1
  (CF) ← (AF)
  (AL) ← (AL) & 0FH
```

编码：

0 0 1 1 0 1 1 1

定时(时钟)： 4

实例：　AAA　；在加法指令的后面使用

标志： 影响　AF, CF.
不确定　OF, PF, SF, ZF

说明： AAA 指令用来对两个非压缩型的 BCD (ASCII) 数相加的结果（在 AL 中）进行修正，以获得一个非压缩型的十进制“和”。

AAD

(ASCII adjust for division)——除法的ASCII修正指令

操作： 累加器的高位字节 (AH) 乘10，然后与低位字节 (AL) 相加，结果存于AL中，再把AH清 0 。

(AL) ← (AH)*OAH + (AL)
(AH) ← 0

编码：

11010101	00001010

定时(时钟)： 60

实例： AAD ；在除法指令的前面使用

标志： 影响 PF, SF, ZF.
不确定 AF, CF, OF

说明： 在两个非压缩型十进制数相除指令之前，用AAD指令对AL中的被除数进行修正，使除法指令执行后获得的商是一个非压缩型的十进制数。

AAM

(ASCII adjust for multiply) ——乘法的ASCII修正指令

操作： 用AL除10的结果去置换AH中的内容，然后再用这次除法所产生的余数去代替AL中的内容，也就是对AL取模10的余数去代替AL的内容。

```
(AH) ← (AL) / 0AH
(AL) ← (AL) % 0AH
```

编码：

11010100	00001010

定时(时钟)： 83

实例：
```
AAM ;在乘法指令后面使用
```

标志： 影响 PF, SF, ZF.
不确定 AF, CF, OF

说明： AAM指令的功能是对两个非压缩型十进制数相乘的结果(在AX中）进行修正，以获得非压缩型的十进制积。

AAS

(ASCII adjust for subtraction)——减法的ASCII修正指令

操作： 若AL的低4位大于9，或辅助进位标志AF置位，则从AL中减去6，从AH中减去1，AF和CF标志被置位。AL中老

的数值被指令所建立的新数值代替，这个新数值的高4位为0，低4位是0～9中的一个数。

```
if ((AL) & 0FH) > 9 or (AF) = 1 then
  (AL) ← (AL)−6
  (AH) ← (AH)−1
  (AF) ← 1
  (CF) ← (AF)
  (AL) ← (AL) & 0FH
```

编码：

0 0 1 1 1 1 1 1

定时(时钟)： 4

实例： AAS ；在减法指令后面使用

标志： 影响 AF, CF.
不确定 OF, PF, SF, ZF

说明： AAS 指令对两个非压缩型十进制数相减的结果（在AL中）进行修正，以获得非压缩型十进制数的差值。

ADC
(Add with carry)—— 带进位的加法指令

操作： 如果进位标志CF已经置位，那么ADC指令在往目的操作数（最左边）存入结果之前，要先将结果加1。若进位标志CF没有置位（即为0），则不往结果中加1。

if (CF) = 1 then (DEST) ← (LSRC) + (RSRC) + 1
else (DEST) ← (LSRC) + (RSRC)

编码：有三种格式

存储器或寄存器操作数与寄存器操作数相加：

0 0 0 1 0 0 d w	mod reg r/m

if d = 1 then LSRC = REG, RSRC = EA, DEST = REG
else LSRC = EA, RSRC = REG, DEST = EA

定时(时钟)：

(a) 寄存器到寄存器	3
(b) 存储器到寄存器	9 + EA
(c) 寄存器到存储器	16 + EA

实例：

```
(a) ADC   AX,   SI
    ADC   ,SI   ;和上面的功能一样
    ADC   DI,   BX
    ADC   CH,   BL

(b) ADC   DX,   MEM_WORD
    ADC   AX,   BETA [SI]
    ADC   ,BETA [SI]  ;与上面一样
    ADC   CX,   ALPHA [BX] [SI]

(c) ADC   BETA [DI],   BX
    ADC   ALPHA [BX] [SI],   DI
    ADC   MEM_WORD,   AX
```

立即操作数到累加器：

0 0 0 1 0 1 0 w	data	data if w=1

if w = 0 then LSRC = AL, RSRC = data, DEST = AL
else LSRC = AX, RSRC = data, DEST = AX

定时(时钟)： 4

实例：

```
ADC   AL,   3
ADC   AL,   VALUE_13_IMM
ADC   AX,   333
ADC   AX, IMM_VAL_777
ADC   ,IMM_VAL_777 ;与上面的指令一样
```

立即操作数到存储器/寄存器操作数：

1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s:w=01

LSRC = EA, RSRC = data, DEST = EA

定时(时钟)： (a) 立即数到存储器 17 + EA
(b) 立即数到寄存器 4

实例：

```
(a) ADC   BETA [SI],   4
    ADC   ALPHA [BX] [DI],   IMM4
    ADC   MEM_LOC,   7396

(b) ADC   BX,   IMM_VAL_987
    ADC   DH,   65
    ADC   CX,   432
```

如果寄存器或存储器的字要与立即数字节相加，那么在加之前，要先进行符号扩展，把符号扩展到最高位（第16位），即把字节立即数扩展为字立即数。

标志：影响　AF, CF, OF, PF, SF, ZF

说明：ADC 指令实现两个操作数的相加，若CF置位，则将结果加1后，送回目的操作数中去。

ADD

(Addition) ——加法指令

操作：将两个操作数的和存入目的（左边的）操作数中去。

(DEST) ← (LSRC) + (RSRC)

编码：有三种格式

存储器/寄存器操作数与寄存器操作数：

0 0 0 0 0 0 d w	mod reg r/m

if d = 0 then LSRC = REG, RSRC = EA, DEST = REG
else LSRC = EA, RSRC = REG, DEST = EA

定时(时钟)：

(a) 寄存器到寄存器	3
(b) 存储器到寄存器	9+EA
(c) 寄存器到存储器	16+EA