微處理基本原理

(硬體與軟體)

Microprocessing Fundamentals

(Hardware and Software)

原著者：E V. Ramirez
M. Weiss
譯述者：陳　友　武

科技圖書股份有限公司

微處理基本原理

硬體與軟體)

Microprocessing Fundamentals

(Hardware and Software)

原著者：E V. Ramirez
M. Weiss

譯述者：陳　友　武

科技圖書股份有限公司

原　序

本書旨在介紹微處理器。所涵蓋的主題是；微處理器的基本原件、程式的基本理論、微處理器的界面以及微處理器的應用等。在此領域中，微處理器的種類日增。本書僅介紹一般微處理而非針對某一特殊微處理器。也因爲本書不含任何特定的微處理器的分析及資料，故能提供一個健全的微處理基本原理。在研究書中各課題時，並不一定需要具備數位技術方面的背景；若已有此背景，更好。

第一章簡介基本數位技術，對未了解數位技術的讀者有所助益。第二章複習數位電算器系統的各種基本元件，並強調那些在功能等級上產生的基本作業。在第三章，爲介紹微處理器概念。第四章討論固態記憶的技術與功能。

第五章說明程式的基本原理，介紹指令概念與選址法。並說明30個指令的基本“一般指令組”，用範例協助瞭解選址法與指令概念。在瞭解一般指令組後，就具備在任何微處理器上寫程式的基本常識。

第六章專論軟體，並說明機器、組合以及高階語言。另介紹軟體如何開發及其使用的工具。第七章介紹流程圖，並用步進方式說明各種流程範例所需的程式指令。

在第八章中，深入探討微處理器與數元分割微處理器單元（bit-slice microprocessor unit）。第九章討論界面問題。這是了解微處理器的一個重要環節。此外，傳輸技術與輸入-輸出裝置等均爲本章的主要課題。第十章專門討論微處理系統的應用與選擇。

本書可供大專學生，欲瞭解微處理器內部作業以及那些想閱讀微處理器初階的工程師及經理人員們的使用。

若欲將本書作爲授課教材，則書中資料可供電機或電腦工程系一學期的教材。本書亦適用於個人自修，因爲先介紹微處理器基本概念，然後再用各章末附問題來加强這些概念。各章內容均力求硬體與軟

體資料間的平衡。

僅此向審閱手稿的 Jim Becker, Bart Collins 及 Grumman Aerospace 的 Lenny Mc Donough 及 Sperry Gyroscope 的 Walter Flattau 先生致謝。並以此書獻給我們的家人，感謝他們這段時間給我們的鼓勵與協助。

Edward V. Ramirez 拉米蘭芝

Melvyn Weiss 維 斯

目　錄

第四章 技 術

第五章 程式寫作基本原理

第六章 軟 體

第七章 程式應用

第八章 超微處理器矽晶片

第九章 微處理器界面

第十章 應用與選擇

導論—微處理器的演進

1970 年代中，微處理器（ microprocessor）的發展，代表了電子工業的重要進步。微處理器之所以大行其道，是有其導因的。諸如型體小，所需的電源低，較配線邏輯（ hard-wired logic ）用的配件數少。此外，可能是最重要的因素，爲成本低廉。由於微處理器的體型小與成本低，消費者及小型企業，目前可擁有足夠的矽晶片（ chip ），其計算能力，可與 1970 年代中僅能由大用戶負擔的資料處理系統相比擬。

在 1960 年代中，一個用於數位邏輯電路的電晶體，其售價約爲五塊美金。但在 1970 年代末期，同樣的五塊錢可買到一個微處理器矽晶片。此一晶片含有 10,000 個電晶體，並含有必要的電阻與元件連接，的確是了不起的技術成就。

由於軍事太空計畫的需要，我們也享受了微處理器大幅度發展的利益。此種需要，使廠商發展出小型的電子電路，稱爲微電腦（ microelectronics），軍事與太空計畫需要性能可靠、功率消耗低、量輕、體小的系統，這些目標需要小型電路。半導體廠商可利用較簡單的處理及減小遮型面積（ pattern mask size）以增加基本矽晶片的電路密度（ circuit density ）。此種要求較高密度包裝的趨勢，自 1960 年代開始一直不斷地發展。但，微處理器的成功，實應歸因於商用市場，由於他們的需要，創出高水準產品與微處理器售價的不斷降低。

我們可將微處理器定義爲：一個含有成仟的數位閘（ digital gate），能執行通用電算器（ general-purpose computer ）所能執行的算術、邏輯以及控制功能的積體電路。它是大比例積體電路（ large-scale integrated circuit ）族中的一員。這反映出自 1940 年代末期電晶體發展以來的現在趨勢。

由於半導體工業不斷製造多功能、低成本的矽晶片，所以很自然地要求較高級的電路積體（circuit integration）。微處理器需要外部的輸入、輸出電路以及外部記憶，使在功能上作爲電算器用。其趨勢是將這些電路納入同一微處理器晶片中，而在單一晶片上組成一個電算器。此種結構，稱爲微電算器（microcomputer）。在小於40,000 平方米爾（mil²）範圍內，集合有微處理器的記憶，輸入、輸出界面電路，及算術、邏輯與控制功能。只要這些新品上市，微處理器與微電算器的使用亦就日益廣大。

在1960 年代中期，「中型比例積體（medium-scale integration，MSI），每一晶片含有 50 至400個電晶體」技術出現時，就被廣泛用來生產比當時已有的電算器體積更小。誠如其名，小型電算器（minicomputer）僅企圖供給一定的，有限的市場。小型電算器設計上是用來處理低層（low end）作業的，諸如以前由大型電算器所處理的管制器功能，資料搜集以及顯示等。

小型電算器的優點計有，處理速度快，字長（word length）較短，具多種輸入、輸出結構。另外在"小型"（mini）方面的成功，是由於採用MSI而獲得較小體型。大型電算器，稱爲主機（main frame units），亦接着採用MSI。小型電算器的所以成功，成本也許是最主要因素。在60 年代中期，電算器售價約爲25,000 美元；但到1970 末期，售價約爲10,000 美元。此一低廉價格，加上最佳性能，使小型電算器在市場上佔有一席之地。

資料處理（data processing）的範圍甚廣，所以，微處理器與微電算器也都佔有部份市場，甚至創出新的市場。微處理器，將佔有目前由小型電算器所執行的低階層（low end）應用。

另一趨勢是，在更新的微處理器中所見到的複雜等級日增。諸如計時器／計數器（timer／counter）以及傳動器等電路，在較老的微處理器中，都是置在矽晶片以外的，但現在都已納入晶片中了。第一部微處理器發展成功時，它是一個 4 數元（4-bit）機，具有一般密度。此後發展的其他機種，都增加了數元與密度。到了1970 年代末期，大多數微處理機都採用每字（word）8 個數元，更新的每字有

16 個數元。吾人可以很合理的預期，以目前强調 8 數元微電算器與 16 數元微處理器來看，單片 16 數元微電算器即將成爲事實。

第一章　基本數位技巧

微處理器，是大比例積體（ large scale integration, LSI ）技術的產品。採用數位技巧作爲輸入、輸出以及內部結構。本章旨在複習數系與數位積體電路（ digital integrated circuit, IC ）的組成方塊（ building block ）。

我們所用的十進位數系（ decimal number system ），其基數爲 10 ，每一數位可爲 0 至 9 中的任一數字，一共有 10 個不同的數字。早期人們企圖製造十進位電算器時，效果並不太好，原因是要想可靠地表示十種狀態實在不易。結果採用較爲滿意的雙態（ two levels ）方法，用邏輯（ logic ）1 表示一態，而以邏輯 0 來代表另一態。爲求了解二進位數系（ binary system），必需先徹底了解原已習用的十進位（基數10）數系。

1.1 十進位數系

4385 一數，讀爲“四仟三百八十五”，由最左的最大有效數字讀起。將 4385 一數寫出的方式爲：

$$4385_{10} = 4 \times 10^3 + 3 \times 10^2 + 8 \times 10^1 + 5 \times 10^0$$

最右邊的數字位（ digit position）數值最小。每向左一位，其值以 10 的冪次增加。

在十進位數系中，將兩數相加的規則是先由最低值位數加起，然後進行次高值位，餘此類推。若兩個數字相加的值超過 10 ，即產生一個進位（ carry ）進入次一較高數字位，例如：

$$\begin{array}{r} 086 \\ +057 \\ \hline \end{array}$$

其加算法如下：

$$6 + 7 = 3 + \text{“進位 1”}$$
$$8 + 5 + 1\text{（前一進位）} = 4 + \text{“進位 1”}$$

0 + 0 + 1（前一進位）= 1 + "進位0"
答案 = 143

在減法中，減數由被減數中減去，減法規則規定，若被減數值小於減數值，則被減數加上 10，左邊一位的減數即有一"借位 1"（borrow 1）加在其上。例如：

86 被減數
−57 減數

其減算法如下：

6 − 7 = 10 (1 借位) + 6 − 7 = 16 − 7 = 9
8 − 5 = 8 − (5 + "借位 1") = 8 − 6 = 2
答案 = 29

另一種比較特殊的減法，稱爲9 補或 10 補法（9's 或 10's complement system）。

1.1.1　9補或10補法

9 補與 10 補法，可用來轉換減法運算爲加法運算。

一數若轉換爲9 補，是將其每一個十進位數字均用 9 減之。例如，試求 37, 45, 及 29 的9 補。

99	99	99
−37	−45	−29
62	54	70

37 的 9 補 = 62
45 的 9 補 = 54
29 的 9 補 = 70

在9 補減法（9's complement subtraction）中，先將減數求補，再與被減數相加。由最高數字位所產生的進位並不用作一個新的數字位。相反，我們將它加在個位數中。

例題 1

由 84 中減去 17　　由 75 中減去 3
17 的 9 補 = 82　　03 的 9 補 = 96

$$\begin{array}{r} 84 \\ +\ 82 \\ \hline 166 \end{array} \qquad \begin{array}{r} 75 \\ +\ 96 \\ \hline 171 \end{array}$$

進位 1 67　　進位 1 72

欲將一數由其 9 補轉換爲 10 補，僅需將其 9 補值加 1 即得。

例題 2

試求 12, 44 與 73 的 10 補

12 的 9 補 = 87,　12 的 10 補 = 88

44 的 9 補 = 55,　44 的 10 補 = 56

73 的 9 補 = 26,　73 的 10 補 = 27

在 10 補減法（10's complement subtraction）中，減數先轉換爲其 10 補形式，然後再與被減數相加。最高數字位所產生的進位，並不形成一個新的數字位。相反的，我們不計其值。

例題 3

用 10 補算術學（arithmetic）作減法

$$\begin{array}{r} 86 \\ -57 \end{array}$$

減數 = 57

9 補 = 42

10 補 = 42 + 1 = 43

$$\begin{array}{r} 86 \\ +\ 43 \\ \hline 129 \end{array}$$

進位不計

答案 = 29

例題 4

用 10 補算術計算

$$\begin{array}{r} 456 \\ -308 \end{array}$$

減數 = 308

9 補 = 691

10 補= 691 + 1 = 692

$$\begin{array}{r} 456 \\ +\ 692 \\ \hline 1148 \end{array}$$

進位不計

答案= 148

1.2　二進位數系

數位系統（ digital system ），與類比系統（analog system ）不同，它只有兩態（ state ），ON 與 OFF 。這兩個狀態，可用開關的關（ opening 與合（ closing ）來代表，或用一般電晶體-電晶體邏輯（ transistor-transistor logic ） TTL ）中的 1 態（ 4V ± 1V ）與 0 態（ 0.2 V ± 0.2 V ）來表示。通常邏輯 1 態最小定爲 2.4 V ，而邏輯 0 態最大爲 0.8 V 。由於一個數位系統僅有兩態，故可用二進位數系來表示。二進位數系只有兩個記法（ notation ），0 與 1 。二進位數系是基數爲 2 的系統，與基數爲 10 的十進位數系不同。用二進位來表示任何十進位數的原則，是位置記法（positional notation ）。也就是說，任何在小數點（二進位數系中稱爲二進位小數點（ binary point)左邊的數字位，均爲 2 的升冪次。在小數點左邊的第一個，或最小有效數位，其值爲 2^0 。再次一位則有 2^1 值，或稱爲該位置的權值（ weight value ） 。左邊的每一數位，均有其用 2 的冪次每一位增加一次的權值。一個 4 位數，其數字位權值爲 2^3 2^2 2^1 2^0 ，任何數字位上的數字爲 1 時，即獲有該位的權值。二進位數 1010110 可轉換成等值的十進位數如下：

$$\begin{aligned} 1010110 &= 1\times 2^6 + 0\times 2^5 + 1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 1\times 2^1 + 0\times 2^0 \\ &= 1\times 64 + 0 + 1\times 16 + 0 + 1\times 4 + 1\times 2 + 0 \\ &= 86_{10} \end{aligned}$$

表 1-1 所示，爲十進位數 0 至 15 的等值二進位數。2的冪次表可參考附錄 1 。注意， 15 一數的各數字位均爲 1 ，四數位可表示的最大值爲 15 。可用數式來表示爲：

最大十進位值= $2^n - 1$

表 1-1

十進位數	二進位等值	十進位數	二進位等值
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

式中的n爲二進位數字位數值。若爲8個數字位，即可表示的最大十進位數爲255；亦即，$2^8 - 1 = 256 - 1 = 255$。

1.2.1 數元與其他名詞

一個二進位數字（binary digit）（1或0），我們稱爲一個數元（bit）。在計算系統中，連續4個數元稱爲一個（nibble）。連續8個數元組成一個byte（數元租）。大部份的微處理系統，均有一個8數元的內部結構，其中各個組成件間的資料傳送都用8數元爲單位作並聯實施。通常用來說明一組共同工作的8數元單位的名詞爲“字句”（word）。在典型微處理系統中，一個字句，即爲一個byte（8個數元）；但在較大系統中，一個字句，可能爲16個數元。大型資料處理系統，可用到32或64數元。這代表4至8個byte，或兩倍此數的nibble。

在任何資料字句結構（data word organization）中，離最小有效數元（least significant bit LSB）最遠的數元，稱爲最大有效數元（most significant bit, MSB）。在電算器術語中，LSB及MSB數位，通常都是如此定義的。

1.2.2 二進位碼的應用

假定要用二進位碼（binary code）來表示數字0到9（10個字）及大小寫的英文字母（52個字）；一共需用62個字。由於$2^5 <$

表 1-2 字碼的二進位表示

B5 (MSB)	B4	B3	B2	B1	B0 (LSB)	字號	字
0	0	0	0	0	0	0	A
0	0	0	0	0	1	1	B
0	0	0	0	1	0	2	C
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
0	1	1	0	0	1	25	Z
0	1	1	0	1	0	26	a
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	1	0	0	1	1	51	z
1	1	0	1	0	0	52	0
1	1	0	1	0	1	53	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

$62 < 2^6$，故最少需要 6 個數元。這 6 個數元組成一個字句，每一字句均可代表一個字。則這 62 個字的二進位格式（binary format）如表 1-2 所示。用 B0 與 B5 代表 6 個二進位數元。當這二進位格式傳送到一個 8 數元微處理器時，每一個 byte 代表一個字加上兩個備份數元（spare bit），數元 B6 及 B7 。這些備份數元，可用來驗證在傳送給微處理器時來漏失數元，並可示知微處理器所傳送的字均爲正確。

一種自我檢查方法，可加強對傳送資料的驗證，可用同位（parity）法來實施。同位，是一種方法，它不論所傳送的資料爲何，都永遠使資料中的二進位 1（或 0）的總數加起來爲偶數個或奇數個。若字句中二進位 1 的和爲奇數，就稱爲奇同位（odd-parity）。若和爲偶數，則稱爲偶同位（even-parity）。表 1.3 所示，即爲使用全部 8 個數元的校對字句結構（revised word structure）。微處理器利用此一字句結構，可測試數元 B6 來決定的字是否合理（validity），在決定是否正確後，再檢查傳送是否正常。

1.2.3 正邏輯與負邏輯

常見的一個不正確觀念是，將負邏輯（negative logic）視爲負電壓，這是不對的。在 TTL 中，正邏輯（positive logic）僅