

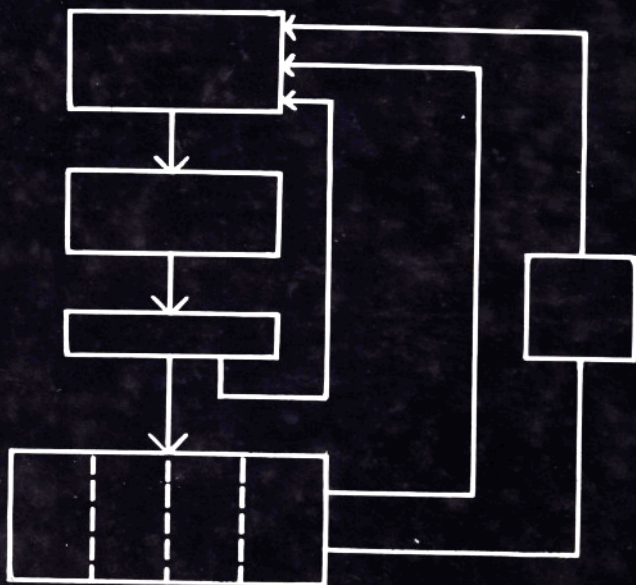
第二版

計算機系統結構

黃文增 ■ 郭惟讓 譯

SECOND EDITION

M. MORRIS MANO



計算機系統結構

黃文增 譯
郭惟讓

SECOND EDITION

COMPUTER SYSTEM ARCHITECTURE

M. MORRIS MANO

雲陽出版社印行

版權所有 翻印必究

計算機系統結構

S 0001A, C 005A

131

作 者 黃文增・郭惟讓
出版者 臺灣出版社
台北市光復南路17巷46號
台北郵政信箱36-60
7629705 7610482
登記證警版台警字第0908號
發行人 陳文慶
台北市光復南路17巷46號
7629705 7610482
印刷者 遠東印刷廠
台北市東區街260巷25號
3071088
基 價 平裝 伍圓、精裝 陸圓
版 次 中華民國72年8月版

學校及團體用書請向本社直接洽購

緒 言

計算機結構是關於計算機工程其討論計算機系統之硬體設計，和計算機科學家涉及設計硬體之相關軟體系統。一個計算機系統為一個系統其包含硬體和軟體。本書大部分關於計算系統硬體之觀點，但影響在計算機結構之軟體沒有被省略。

計算機結構有時候被定義只包含那些有興趣為程式師之計算機性質。在這裏，我們定義計算機結構為考慮那些專業從事計算機設計必須知道之事物。計算機結構必須熟悉其計算機被結構時之基本硬體建築方塊。它們必須有計算機結構和行為性質以及它們被設計之方法等的知識。如此，計算機結構在本書之定義為關於結構的構造和數位計算機系統之硬體設計。

在一個單獨處理機中之實質結構包含它的暫存器，資料流，微運算和控制函數，其最好以暫存器轉換語言符號方法來敘述。如此一個語言為在本書中被發展而其相對於硬體結構和數位計算機之設計全都被解釋。此暫存器轉換語言為被在許多機會下使用來以簡短和精確方法來指定不同計算機運算。

本書之計劃為首先出現較簡單事件而稍後介面更高級之科目。此首先六章節涵蓋基本了解計算機結構，設計，和程式規劃一個簡單數位計算機所須之資料。此最後六章出現在數位計算機中之個別函數單元和強調在較早部分沒涵蓋更高級之論點。

第二版之修正為在後六章。第七章為中央處理機單元，第十一章為輸入-輸出結構已被完全重寫。第八章為微程式控制和第十二章為記憶體結構已經被修改並加上新的資料。其餘章節保持和第一版基本上相同除了少許之重新組織。

第一章介紹設計數位系統時所須之基本知識此時它們為用個別之閘和正反器來結構。其包括布林代數，組合電路，和循序電路。為便於保持本書有合理之界限，它必須限制此科目之討論為一個介紹性章節。為辨明這個方法為用不同之範圍來設計數位計算機此時使用積體電路功能來代替個別之閘和正反器。包含在第一章之原料為提供將出現數位計算機必須了解之知識背景。

第二章開始枚舉積體電路之通常性質。它涵蓋一些基本數位函數大部分的一些細節諸如暫存器，計數器，解碼器，多工器，隨機存取記憶體，和唯讀記憶體。這些數位函數為被用來在跟隨本章之較大單元的建設方塊。

第三章出現在數位計算機可發現之不同資料型式並指出它們在計算機暫存器中之二進制形式如何表示。在數字表示之重要處為在算術計算使用之數字和符號之二進制碼，諸如在資料處理使用之字母文字，和其它在特殊應用之分立符號。

第四章定義暫存器轉換語言並指出如何用在數位計算機暫存器之間微運算的符號表示形式之中。符號被定義為算術，邏輯，和移位微運算與起動微運算之控制函數。此表示盡一切所能來指出聯合不同符號和暫存器轉換陳述之硬體內容。

第五章出現在一個小型基礎數位計算機之結構和設計。在計算機之暫存器被定義和一組計算機指令被指定。此計算機陳述為暫存器轉換陳述之正規化，其指定在暫存器間之微運算和起動微運算之控制函數。此時指出用來設計計算機資料處理機部分之微運算組合。以暫存器轉換陳述所列之控制函數提供設計控制單元之資訊。

第六章使用在第五章定義基本計算機中之二十五條指令來舉例，說明許多程式一個計算機共同使用之技術。在符號碼之程式例子為表示許多基本之資料處理工作。在二進制程式，符號碼程式，和高階語言程式之間的關係為用例子解釋。此必須載入翻譯程式諸如組合器和編輯器。在一個組合器之基本運算和其它系統程式一起表示。本章之目的為介紹計算機軟體之基本概念而不深入其細節。軟體

原則之知識來和硬體連接時的表示給予讀者全覽整個計算機系統，其包含硬體和軟體。

第七章討論數位計算機之中央處理機單元（*CPU*）。一個匯流排處理機被表示和一個特定之算術邏輯單元（*ALU*）為被設計。記憶體疊存器之結構為用一些它的應用來示範說明解釋。不同指令規格為和其選址型式一起來說明。在一個典型計算機可找到之最通用指令為以其函數來枚舉說明。微處理機為包含 *CPU* 在一個積體電路包裝之中在此時被介紹，而其內部和外界性質也被分析。本章以並行和導管處理之一節來結束。

第八章介紹微程式之觀念。一個特定之控制單元被發展用例子來指出在計算機指令中如何產生給典型組合之微程式。一個微程式循序器被發展，來示範說明在數元-晶片種類用 *LSI* 成員設計之程序。最後一節為討論微程式應用之好處。

第九章專心從事一個算術處理機之設計。它出現在二進制整數記號-大小表示之加法、減法、乘法和除法之演算法。此算術處理機為用暫存器轉換語言來設計。此算術處理機之結構為連繫第五章設計的計算機。一個二進制計算器被定義並用來示範說明算術運算可被微程式之方法。

第十章出現其它算術演算法。演算法是為記號-2 的補數二進制資料，為浮點資料，和十進制資料所發展。此演算法為用流程圖來表示其使用暫存器轉換語言來指定微運算之序列和在演算法完成中所須之控制決定。

第十一章解釋在一些通用之輸入和輸出裝置的功能。在處理機和 *I/O* 裝置之間的一個介面設備為被解釋而在 *I/O* 轉換之不同結構被枚舉。此包含非同步轉換，直接記憶體存取，和優先權間斷。其它話題為涵蓋輸入-輸出處理機，資料通訊處理機和多重處理機系統結構。

第十二章介紹記憶體階級組織，高速緩衝記憶體之構成、主記憶體，和輔助記憶體諸如磁碟和磁帶之觀念。在結合型記憶體之內

4 緒 言

部結構和外界運算被詳細解釋。記憶體管理之觀念為經由一個高速緩衝記憶體和一個虛構記憶體系統之硬體設備的表示來介紹。

每一章包含一組問題和一個參考書目表列。一些問題保留為在此章中包含事件之練習之用。其它者為增加更多之性質和試圖在數位計算機硬體設計方面聯合之問題解決提供一些練習。一個解答手冊 (*solutions manual*) 為可由出版者之指導書中找到。

這本書適合電子工程，計算機工程，或計算機科學科系之計算機結構課程。在畫中之部分可在不同方法下使用：(1)如為在計算機硬體結構之第一個課程時涵蓋第一章至第五章加上來自第七、八或九章為指導介質即為大致足夠。(2)如為計算機設計之課程有先前之數位邏輯知識即設計為複習第五章而後涵蓋第七章至第十二章。(3)如為計算機硬體系統之課程其涵蓋在數位計算機之五個函數單元：處理機（第七章）、控制（第八章）、算術（第十章）輸入 - 輸出（第十一章），和記憶體（第十二章）。此書也適合自己研讀計算機工程和須要獲得計算機硬體結構基本知識之科學家所使用。

M. Morris Mano

目 錄

第一章 數位邏輯電路

1—1	邏輯閘.....	1
1—2	布林代數.....	6
1—3	圖示簡化法.....	9
1—4	組合電路.....	18
1—5	正反器.....	25
1—6	循序電路.....	31
1—7	結論備註.....	41
	問題.....	42

第二章 積體電路和數位函數

2—1	數位積體電路.....	45
2—2	IC 正反器及暫存器.....	53
2—3	解碼器與多工器.....	58
2—4	二進制計數器.....	63
2—5	移位暫存器.....	70
2—6	隨意進出記憶體.....	74
2—7	僅讀記憶體.....	81
	問題.....	86

第三章 資料表示法

3—1	資料種類.....	91
-----	-----------	----

2 目 錄

3—2	定點表示法	100
3—3	浮點表示法	109
3—4	其他的二進碼	112
3—5	錯誤偵測碼	116
	問題	119

第四章 暫存器轉換及微運算

4—1	暫存器轉換語言	125
4—2	內部暫存器轉換	127
4—3	算術微運算	139
4—4	邏輯微運算	144
4—5	移位微運算	153
4—6	控制函數	155
	問題	162

第五章 基本電腦結構及設計

5—1	指令碼	167
5—2	計算機指令	171
5—3	節時序和控制	176
5—4	指令之執行	182
5—5	輸入-輸出和間斷	190
5—6	計算機設計	196
5—7	總結	201
	問題	203

第六章 電腦軟體

6—1	導論	209
6—2	程式計碼語言	211
6—3	組合語言	215

6—4	編輯機組合程式	220
6—5	程式環	226
6—6	程式計劃算術和邏輯運算	230
6—7	副程式	237
6—8	輸入-輸出程式計劃	243
6—9	系統軟體	249
	問題	258

第七章 中央處理結構

7—1	處理器匯流排結構	263
7—2	算術邏輯單元 (ALU)	266
7—3	疊存器結構	276
7—4	指令形式	284
7—5	選址型式	290
7—6	資料轉換和處理	298
7—7	程式控制	305
7—8	微處理機結構	316
7—9	並行處理	328
	問題	340

第八章 微程式控制結構

8—1	控制記憶體	349
8—2	地址循序	352
8—3	巨式程式的例子	358
8—4	微程式循序	369
8—5	微指令格式	375
8—6	軟體輔助	381
8—7	優點與應用	384
	問題	388

第九章 算術程序設計

9-1	概論	393
9-2	無記號二進制數字之比較和減法	395
9-3	加法和減法演算法	400
9-4	乘法演算法	404
9-5	除去演算法	408
9-6	處理機結構	414
9-7	設計控制部分	419
9-8	微程式計算器	422
	問題	432

第十章 算術演算法

10-1	概論	435
10-2	用記號-2的補數數字的算術	437
10-3	乘法和除法	441
10-4	浮點算術運算	449
10-5	十進制算術單元	461
10-6	十進制算術運算	466
	問題	474

第十一章 輸入-輸出結構

11-1	周邊裝置	481
11-2	I/O 介面	485
11-3	非同步資料轉換	496
11-4	直接記憶體存取	511
11-5	優先權間斷	517
11-6	輸入-輸出處理機(IOP)	528
11-7	多重處理機系統結構	540

11—8 資料通訊處理機.....	550
問題.....	563

第十二章 記憶體結構

12—1 輔助記憶體.....	571
12—2 微型計算機之記憶體.....	576
12—3 記憶體階級組織.....	582
12—4 結合型記憶體.....	584
12—5 虛構記憶體.....	591
12—6 高速緩衝記憶體.....	599
12—7 記憶體管理硬體.....	609
問題.....	619

第一章

數位邏輯電路

1—1 邏輯閘

數位計算機，顧名思義，是一個可以處理各種計算工作的數位系統。數位的（*digital*）這個名詞是指計算機的資訊是以離散性（*discrete*）或分離化（*quantized*）的數值來說明。這些數值由能維持一個分離性狀態之限定數目的構成零件做內部處理十進制數位0, 1, 2, …… 9，如此例，提供10個分離性的數值。實際上，如果只有使用兩種狀態，數位計算機的特性將更可靠。因為構成零件物質上的限制，乃因人類的邏輯更趨近於二進制（也就是說，真或假，是或否的格式），被強制要採用分離數值的數位的構成零件，更進一步的，強制去採用只有兩個數值也就稱為二進制。

數位計算機採用二進制數目系統，它有2個數位：0和1。一個二進制的數字，稱為數元（*bit*）。在數位計算機中，資訊是以數元群來代表的。藉著使用不同的密碼技術，數元群可以代表二進制數外，還有任何其他分離性的符號，就如十進制的數目，或字母的文字也都可代表之。藉由二進制安排的賢明使用，及藉由各種密碼技術，二進制數或數元群可用發展完整的指令以執行各類型的計算機工作。

普通採用底數為 10 的十進制數與採用底數為 2 的二進制數有很大的差異。例如，二進制數 101101 乃說明一個數量，由每個數元乘以底數為 2，如下皆整數展開式而成了十進制數：

$$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 45$$

101101 這 6 個數元說明十進制等於 45 的二進制數。無論如何，這 6 個數元說明一個二進制的密碼，爲了字母的文字或是指定的數位系統中特定的決定性的邏輯。換句話說，數位計算機中的數元群是用以說明許多不同的事情。這個概念乃類似於相同的字母文字被構成不同的語言，如英語和法語。

二進制資訊在數位系統中以物理數量我們稱之爲信號的 (signals) 來說明。電子信號如電壓，存在遍及數位系統代表兩個可識別值的表示狀態，並且說明一個爲 1 或 0 的二進制變數，例如，一個指定的數位系統中採用 3V 的信號表示二進制的 1，0.5V 的信號表示二進制的 0，如圖 1-1 所示，每個二進制數值都有名義上可被接受的偏移，在兩個可允許區域的中間區域，只在兩個狀態變遷時才經過。數位電路的輸入端可接受在可允許的容量的二進制訊號，並且有二進制訊號在特定的容量充滿的輸出端的電路響應。

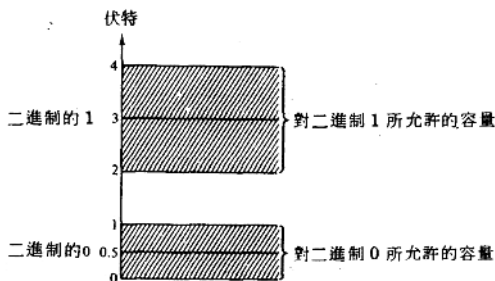


圖 1-1 二進制信號的例子

名稱	符號 圖形	代數 函數	真值表														
AND	 $x = A \cdot B$ or $x = AB$	<table> <tr> <th>A</th> <th>B</th> <th>x</th> </tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	A	B	x	0	0	0	0	1	0	1	0	0	1	1	1
A	B	x															
0	0	0															
0	1	0															
1	0	0															
1	1	1															
OR	 $x = A + B$	<table> <tr> <th>A</th> <th>B</th> <th>x</th> </tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	A	B	x	0	0	0	0	1	1	1	0	1	1	1	1
A	B	x															
0	0	0															
0	1	1															
1	0	1															
1	1	1															
inverter	 $x = A'$	<table> <tr> <th>A</th> <th>x</th> </tr> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </table>	A	x	0	1	1	0									
A	x																
0	1																
1	0																
buffer	 $x = A$	<table> <tr> <th>A</th> <th>x</th> </tr> <tr><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td></tr> </table>	A	x	0	0	1	1									
A	x																
0	0																
1	1																
NAND	 $x = (AB)'$	<table> <tr> <th>A</th> <th>B</th> <th>x</th> </tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	A	B	x	0	0	1	0	1	1	1	0	1	1	1	0
A	B	x															
0	0	1															
0	1	1															
1	0	1															
1	1	0															
NOR	 $x = (A + B)'$	<table> <tr> <th>A</th> <th>B</th> <th>x</th> </tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	A	B	x	0	0	1	0	1	0	1	0	0	1	1	0
A	B	x															
0	0	1															
0	1	0															
1	0	0															
1	1	0															
互斥 OR (XOR)	 $x = A \oplus B$ or $x = A'B + AB'$	<table> <tr> <th>A</th> <th>B</th> <th>x</th> </tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	A	B	x	0	0	0	0	1	1	1	0	1	1	1	0
A	B	x															
0	0	0															
0	1	1															
1	0	1															
1	1	0															
互斥 NOR 或 等值	 $x = A \odot B$ or $x = A'B' + AB$	<table> <tr> <th>A</th> <th>B</th> <th>x</th> </tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	A	B	x	0	0	1	0	1	0	1	0	0	1	1	1
A	B	x															
0	0	1															
0	1	0															
1	0	0															
1	1	1															

圖 1-2 數位邏輯閘

二進制邏輯 (*binary logic*) 處理二進制變數和具有邏輯意義的運算元。它是用於被以代數或表的型式描述，二進制資訊的操作和處理二進制資訊的操作是由邏輯電路稱閘 (*gates*) 做的。閘是一塊硬體，當輸入邏輯的須求滿足時就產生二進制 1 或 0 的信號，不同的邏輯閘普遍地可在數位計算機系統發現。每個閘有一個不同的圖解的符號而它的操作元可用代數函數的方法來描述。對每個閘二進制變數的輸入—輸出關係可用表的方式說明就是真值表 (*truth table*)。

名字，圖解的符號，代數函數，及八個邏輯閘的真值表，被列在圖 1-2，每個閘有一或二個輸入變數命名 A 和 B ，和一個輸出端標為 x ，及 (AND) 閘產生及 (AND) 閘的邏輯連接；那就是，如果輸入端 A 和 B 都是 1 那輸出就是 1，否則輸出就是 0，這些狀態也可定在及 (AND) 閘的真值表上，這表顯示只有當輸入 A 和 B 是 1 時，輸出 x 才是 1。及 (AND) 函數的代數操作符號，和一個算術的操作符號是一樣的，我們不僅使用一個點在變數之間，而且連接彼此間沒有運算符號的變數，及 (AND) 閘可能有 2 個以上的輸入端，而且由定義上，輸出為 1 是若且唯若，所有輸入為 1。

或 (OR) 閘產生相容或 (*inclusive-OR*) 函數，那就是，如果輸入 A 或 B 或兩者都是 1，則輸出為 1，否則輸出為 0。或 (OR) 閘的代數函數就是“+”，類似於算術加號。或 (OR) 閘可能有 2 個以上的輸入端而且由定義上，輸出為 1，若是任何 (*any*) 輸入為 1。

反向器 (*inverter*) 電路就是反二進制信號的邏輯意義，它產生了否 (NOT) 或補數 (*complement*) 的函數。用於邏輯補數的代數符號是一個質數或在這變數符號上有一桿。這本書用一個二進制變數之邏輯補數的質數，當這字上的一桿被反向以分辨一個補數的微—操作 (*micro-operation*) 如第四章定義的。

一個反向器 (*inverter*) 的圖解符號之輸出端有一個小圓圈叫做 1 個邏輯補數，本身的三角符號叫做緩衝電路，一個緩衝器並沒有產生任何指定的邏輯函數，既然輸出的二進制數和輸入的二進制

數是相同的，這個電路只用於做信號放大，例如，一個以 3V 表示二進制 1 的緩衝器，若輸入為 3V 則將產生 3V 的輸出，無論如何，產生在輸入端的電流比產生在輸出端的電流小很多，這個方法，一個緩衝器可以驅動許多閘，須要大量的電流，並非可從適合給緩衝器輸入的小電流而得到的。

反及 (NAND) 閘函數是及 (AND) 的補數函數，如表示的圖解符號，包括及 (AND) 的圖解符號跟著一個小圓圈。反及 (NAND) 的命名是源自 NOT-AND 的縮寫。一個更適合的命名該為及 - 反向 (AND-inverter)，既然它是 1 個被反向的及 (AND) 函數。反或 (NOR) 閘是或 (OR) 閘的補數而且用或 (OR) 圖解符號跟著一個小圓圈。反及 (NAND) 和反或 (NOR) 個別的可能有兩個以上的輸入端，而且輸出總是及 (AND) 和，或 (OR) 函數的補數。

不相容或 (exclusive-OR) 閘有圖解符號，類似於或 (OR) 閘，除了在輸入面之額外的彎曲，如果任何輸入為 1，但不包括兩者輸入同時為 1，輸出才為 1，不相容或 (exclusive-OR) 函數有它自己的代數函數或能用及 (AND) 或 (OR)，和補數操作元如圖 1-2 來表示。不相容反或 (exclusive-NOR)，是不相容或 (exclusive-OR) 的補數，而以一一個小圓圈在圖解符號來表示，只要兩者輸入端有相同的二進制值時，輸出就是 1，我們視不相容反或 (exclusive-NOR) 為相等 (equivalence) 函數，如此，不相容或 (exclusive-OR) 和相等 (equivalence) 函數不都是另一者的補數。一個更合適於不相容或 (exclusive-OR) 的操作稱為一個奇 (odd) 函數，也就是說，如果輸入的 1 為奇數個則輸出為 1。如此，在一個有三輸入端的不相容或 (exclusive-OR) 奇 (odd) 函數，輸出要為 1，只有當一個輸入是 1 或者是所有三個輸入都是 1。相等函數是一個偶 (even) 函數；那就是，若輸入的 0 為偶數個則輸出為 1。對一個有三輸入端的相等函數，輸出為 1，如果輸入沒有 0 (所有輸入都是 1) 或是輸入兩者是 0 (一個輸入是 1)。仔細審查將顯示不相容或 (exclusive-OR) 和相等函數兩者互為補數，當閘有偶數