

計算機結構與組織

譯者 曲濟清

徐氏基金會出版

科學圖書大庫

計算機結構與組織

譯者 曲濟清

徐氏基金會出版

徐氏基金會科學圖書編譯委員會

監修人 徐銘信

發行人 陳俊安

科學圖書大庫

版權所有



不許翻印

一九八二年十月廿二日初版

計算機結構與組織

基本定價 4.40

譯者 曲濟清 國立交通大學電子研究所碩士

本書如發現裝訂錯誤或缺頁情形時，敬請「刷掛」寄回調換。 謝謝惠顧

局版臺業字第1810號

出版者 財團法人 臺北市徐氏基金會 臺北市郵政信箱 13-306 號

發行者 財團法人 臺北市徐氏基金會 郵政劃撥帳戶第 15795 號

承印者 大原彩色印製有限公司 台北市武成街三五巷九號

9221763
9271575
電話 9271576
9286842

電話 3017427

原 序

計算機結構 (architecture) 可概略定義為計算機構造 (structure)、行為及設計的研究。近年來由於計算機的廣泛使用，以及計算機技術的進展，此已成為一單獨的學問。70 年代前後所發展之 MSI 及 LSI，已對計算機設計提供一廣泛有力且價廉的元件。或許這些發展中最有意義者為微處理器及微算機的崛起，這些元件於相當短的時間，即將計算機系統融入於原先特殊目的邏輯電路的應用中。因而，計算機結構的觀念對許多工程師及科學家的關係愈形密切。

此書主要作為電機及計算機系所大四及研一的計算機結構及組織的教材。此處強調的是計算機硬體，但相關的軟體面亦論及。本書假定讀者對計算機程式有相當的瞭解，並熟知至少一種簡單的計算機操作，且假定有邏輯設計的基礎。此處並無需具備特殊的數學先修科目，但偶然需一些簡單的微積分及機率理論。

本書的目的為提供對計算機結構深入及完整的探討。所包含的論題與 IEEE 所推薦之計算機結構一致¹。所強調的是設計原理及性能評估，此處並使用相同的專有名詞及符號。本書且包含了約一百個問題，使學生熟悉計算機的分析及設計。

本書劃分為六章。第一章為回顧計算機的發展，並介紹本書所用的專有名詞及符號。次章為設計方法概論，並探討記錄器及處理器設計詳情。此外，亦介紹使用基本排隊理論及模擬性能的評估。隨後兩章為有關指令集處理器的設計。第三章為指令集、算術單元設計及並行處理。

控制單元設計為第四章的主題。簡短的探討定線的控制；隨後對微程式有一完整的討論。第五章則探討記憶器的技術及組織，包含虛記憶、相屬記憶及機架記憶。末章為有關系統結構及通訊。輸入—輸出系統，多處理器，錯誤容忍計算機及計算機網路均被討論及。許多實際計算機系統均用為本書的實例，包括Von Neumann的IAS計算機，IBM的360/370系列，Intel的8080及8748，系帝(CDC)公司的STAR-100，惠普公司(HP)的21MX，ILLIAC IV，貝爾ESS第一號，及ARPANET計算機網路。

本書的材料已用於南加州大學研究所一年級的計算機系統結構課程教材多年。本書所包含的教材略超過一學期所需，授課的教授可依其所需強調者而選擇各個論題。第一及二章許多背景材料，例如若學生有適當的準備，第2-1節之邏輯設計的回顧可予以省去。某些較特殊的論題，例4-3節之微程式數最小化，第5-3節之相屬記憶器或第6-3節之錯誤容忍計算機亦可略去而不影響教材的連續性。

作者希望感謝南加州大學的同事及學生，尤其P. Vahdat及A. Zygielbaum的建議。此外，加大柏克萊分校之G. Miller先生的評論亦深致謝忱。對吾妻Terre於本書編寫時仔細閱讀各個章節亦一併致謝。

John P. Hayes

譯 序

數位計算機為有限狀態的處理機器，其工作基本原理為將欲處理之資訊分為許多基本運算可解決的問題，而後儲存至記憶器內，需要時依序或分至更精細的運算處理之。由於其處理方式完全可程式，過去、現在及將來均是最重要的數位系統。

近三十年來計算機所用之元件自真空管、電晶體、SSI、MSI、LSI以至VLSI進展何止千萬里，然基本計算機結構仍大致相同。唯近年來受數位信號處理及本身計算需求的衝擊，欲要求既快又好，其結構已走向多處理器方式。

需要感謝的人實在太多，無法一一述及，謹依中國的傳統——天代表衆人——將本書獻與天！

曲濟清

目 錄

原 序

譯 序

第一章 計算機的發展

1-1	計算機的本質	1
1-1-1	計算的需求	2
1-1-2	計算機抽象模型	3
1-1-3	計算機的極限	6
1-2	機械計算機時代：1623 至 1945 年	10
1-2-1	第一部計算器	10
1-2-2	Babbage 的計算機	11
1-2-3	隨後的發展	15
1-3	第一代計算機：1946 至 1954 年	17
1-3-1	電子計算機	17
1-3-2	儲存程式計算機	19
1-3-3	第一代計算機實例	22
1-4	第二代計算機：1955 至 1964 年	31
1-4-1	一般特性	31
1-4-2	第二代計算機實例	32
1-4-3	輸入—輸出處理	36
1-4-4	軟體及其他發展	38

1-5	最近的發展	41
1-5-1	第三代計算機	41
1-5-2	第三代計算機族	47
1-5-3	微處理器及微算機	54
1-6	摘 要	61

第二章 設計方法概論

2-1	概 論	67
2-1-1	系統模型	67
2-1-2	設計層次	72
2-1-3	組合邏輯電路設計	75
2-1-4	順序邏輯電路設計	80
2-2	記錄器層次	85
2-2-1	一般特性	85
2-2-2	敘述語言	89
2-2-3	組合邏輯元件	94
2-2-4	序向邏輯元件	102
2-2-5	設計方法	108
2-3	處理器層次	117
2-3-1	引 言	117
2-3-2	元 件	120
2-3-3	設計技術	125
2-3-4	排隊模型	130
2-3-5	模 擬	134
2-4	摘 要	139

第三章 處理器設計

3-1	概 論	147
3-1-1	處理器組織	147
3-1-2	資訊代表方式	152

3-1-3	數字格式	157
3-2	指令集	165
3-2-1	指令格式	165
3-2-2	指令型態	173
3-2-3	實 施	178
3-3	算術運作	179
3-3-1	固定點加法及減法	179
3-3-2	固定點乘法	185
3-3-3	固定點除法	197
3-3-4	算術邏輯單元 (ALU) 設計	206
3-3-5	浮點算術	209
3-4	並行處理	215
3-4-1	引 言	215
3-4-2	性能考慮	219
3-4-3	管線處理器	228
3-4-4	多單元處理器	235
3-5	摘 要	241

第四章 控制器設計

4-1	概 論	250
4-1-1	指令序列	250
4-1-2	指令翻譯	255
4-2	定線控制	257
4-2-1	設計方法	257
4-2-2	乘法器控制單元	266
4-2-3	中央處理器 (CPU) 控制單元	272
4-3	微程式控制	277
4-3-1	基本概念	277
4-3-2	微指令數最小化	288
4-3-3	乘法器控制單元	295

4-4	微程式計算機.....	304
4-4-1	中央處理器 (CPU) 控制單元.....	304
4-4-2	傳統計算機.....	311
4-4-3	非傳統計算機.....	318
4-5	摘 要.....	323

第五章 記憶器組織

5-1	記憶器技術.....	330
5-1-1	記憶器元件特性.....	331
5-1-2	隨意取材記憶器.....	338
5-1-3	串聯取材記憶器.....	350
5-2	層次憶器.....	361
5-2-1	記憶器結構.....	362
5-2-2	主記憶器指派.....	371
5-2-3	取代策略.....	375
5-2-4	分段及分頁.....	381
5-2-5	檔案組織.....	386
5-3	高速記憶器.....	393
5-3-1	交越記憶器.....	393
5-3-2	快取記憶器.....	396
5-3-3	相屬記憶器.....	400
5-4	摘 要.....	408

第六章 系統組織

6-1	通 訊.....	415
6-1-1	本地通訊.....	415
6-1-2	長距通訊.....	417
6-1-3	互聯結構.....	419
6-1-4	匯流線控制.....	423
6-2	輸入—輸出系統.....	432

6-2-1	程式輸入 / 輸出	433
6-2-2	直接記憶體取材 (DMA) 及中斷	440
6-2-3	輸入 / 輸出處理器	448
6-2-4	中央處理器及輸入 / 輸出間交互作用	456
6-3	多重中央處理單元系統	464
6-3-1	多處理器	464
6-3-2	錯誤容忍計算機	473
6-3-3	計算機網路	484
6-4	摘 要	491

第一章 計算機的發展

本章回顧數位計算機的發展歷史。此提供計算機結構廣泛的巡禮，並引入隨後數章深入探討的大多數觀念。對具代表性的亦提出詳盡敘述。

1-1 計算機的本質

在歷史上人類主要依賴其頭腦作計算，換言之彼等即為計算機。雖發明一些計算協助物如算盤及計算尺；其可簡化，但仍無法取代人腦計算工作。而當計算規模及複雜性大為增加，人腦計算有兩項嚴重的限制。

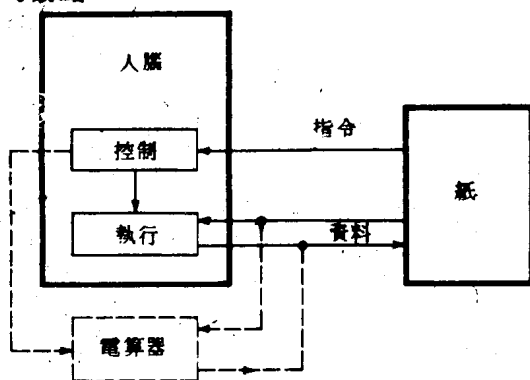
1. 人腦計算速率有限制。典型的基礎運作如加法或乘法需數秒或數分，需數十億此類運算現已交給計算機。此類問題從無法以合理時間或成本以紙筆運算而成。

2. 人類經常犯錯，因而紙筆所作的複雜運算通常並不可靠，除非有極精密的預防以消除錯誤。由於計算機從不被人類錯誤來源（心神渙散、不小心、疲勞等）所影響，彼可提供於極寬廣範圍內無誤差。

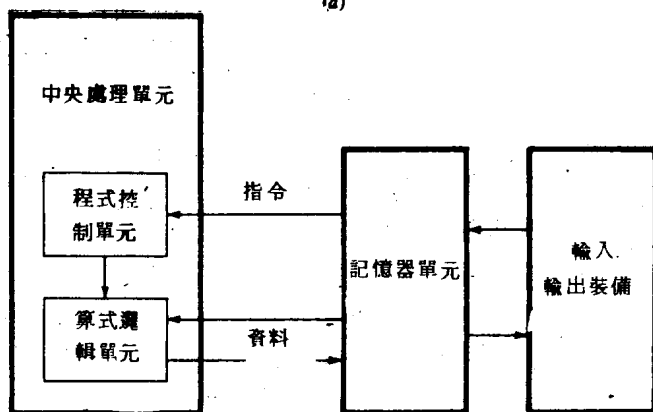
下列實例常被Charler Babbage (1792-1871) 所引用，以評斷彼所建立之第一架計算機，差分引擎〔20〕。於1794年由法國政府G.F.Prony 所領導之計劃為由紙筆計算巨大的數學函數表。在此函數表中，包含自1至200,000整數的對數計算至19位精確度。對正弦及正切函數，及其對數，以及正弦及正切函數至其反函數比值的對數均已求出。整個計劃僱用了70至100人，耗了兩年時間方完成。所僱用工作者的數學能力多僅限於加法及減法。一小組數學家則提供應執行的指令。為減少誤差，每一數均由兩人分別計算再比較之。最後所獲之函數表佔了17個對開面積（然而，從未出版）。自然數的對數表即估計有8百萬個數字。

1-1-1 計算的需求

分析人腦用紙筆計算作一開端甚為有用，紙的主要目的為資訊儲存。儲存於紙上之資訊可包含一序列指令，即算式或程式；隨後於計算中資料則被使用。於計算時，中間結果及最後結果被記錄於紙上。所需的計算過程發生於腦中，因而可稱為處理器。由人腦進行兩種主要功能；控制功能，解釋指令並確保其正確順序；執行功能，執行某特定計算，如加、減、乘及除。於執行功能中，人腦現由電算器所協助。圖 1-1 a 描述人腦計算的觀點。



(a)



(b)

■ 1-1 a(a)人腦計算；(b)機器計算之主要元件。

計算機主要的元件亦類似；彼等描述於圖 1-1b 中。記憶單元對應於人腦計算機所用的紙；其目的為儲存指令及資料。程式控制單元翻譯及序向指令。算術邏輯單元執行指令。彼稱為此是因指令係作數值運作（算術）或非數值運作，如程式分支及符號處理。對後者為邏輯運作。程式控制及算術邏輯單元合併而稱中央處理單元（CPU），可粗略對應紙筆運算的人腦。人腦及計算機主要的差異在其代表資訊的方法，即指令及資料，人腦使用自然語言其具有廣泛的符號，數學常以 10 進位表之，而於現代計算機是以二進位形式儲存及處理。為提供對計算機及人腦的溝通，因而需機器語言與人類語言轉換的方法，此為圖 1-1b 輸入—輸出裝置方塊的主要功能。

因而，每一計算機不論其為人腦或人造者，均需有下列單元。

1. 可解釋及執行程式的處理器。
2. 儲存程式及資料的記憶器。
3. 記憶器與處理器，以及計算機及外界資訊轉換方式。

1-1-2 計算機抽象模型

計算可被認為計算一函數 $f(X)$ ， X 為所給予的輸入資料， $Z = f(X)$ 為所需的輸出資料。 X ， Z ，及 f 可為非常廣泛的意義。 X 及 Z 可代表數字、文字敘述，資訊檔案等等。 f 可為數值計算，定理的證明，檔案更改的步驟等等。為使用某特定計算機以計算 $f(X)$ ，吾人需將 f 表示為一序列函數 $f_1, f_2, \dots, f_n$ ，其可用計算機指令集，即計算機可進行的基本運算，表之。 $f_1, f_2, \dots, f_n$ 可認為計算 $f(X)$ 的程式，及基本運算的序列。

$$Y_1 = f_1(X)$$

$$Y_2 = f_2(Y_1)$$

$$\dots\dots\dots$$

$$Y_{n-1} = f_{n-1}(Y_{n-2})$$

$$Z = f_n(Y_{n-1})$$

可被認為計算的正規定義。

在任何計算機設計前，一很自然產生的問題為：這些計算是否沒有

“合理的”計算機可擔任？若有，則應熟知彼等，而使吾人可試圖建一計算機以實施不可能的工作。通常接受的兩個極廣泛的合理性為：

1. 計算機不應儲存所有可能問題的解答。
2. 彼應可以有限時間處理資訊（執行指令）。

此一計算機僅當可於有限步驟內產生答案，方具有進行特定運算的能力。

滿足先前合理性的計算機的抽象模型由英國數學家 Alan Turing (1912-1954) 於 1936 年提出〔26〕。此模型現被稱為 Turing 機器。圖 1-2 顯示 Turing 機器的組成。如前節所示，任何計算機兩個必需元件為記憶器及處理器。Turing 機器的記憶器為無限長之帶 M 劃分為連續小方塊。每一小方塊可為空白或一有限組符號之一。處理器為具有有限狀態數之數位機器。其具有可讀出帶上任一方塊內含，改變小方塊內含，並移動帶上目前位置向左向右移動的讀寫頭。

Turing 機器可被認為有一具後列格式之有限指令組： s_k, t_i, o_j 。 s_k 意表若控制處理器 P 於狀態 s_k ，符號 t_i 目前正在讀寫頭，而後進行運作 o_j ，並改變 P 的狀態 s_k 。運作 o_j 可為下列四種運算之一。

1. $o_j = t_i$ ，意表寫入符號 t_i 至帶上（取代先前符號 t_i ）。
2. $o_j = R$ 意表移動讀寫頭位置至右方，即將帶移至左方。
3. $o_j = L$ 意表移動讀寫頭位置至左方，即將帶移至右方。
4. $o_j = H$ 意暫停計算。

此指令集基本上為所有可能的 P 的狀態轉移。指令的資訊亦可自 P 的狀態表敘述而獲得。

由 Turing 機器計算 $Z = f(X)$ 是以下列方式進行。首先輸入資料 X 以適當編碼形式置於其他空白帶上。而後開動 Turing 機器，使得運作依 $f_1, f_2, \dots, f_n$ 序列進行。在第 n 個運作結尾，機器應暫停。此時帶上即包含結果 Z 。

·例 1-1：可加兩自然數的 Turing 機器

任何自然數 n 可用 1 進位方式，即 n 個 1 代表。帶符號字母僅需包含兩符號，1 及 b ， b 代表空白。欲相加之兩數 n_1 及 n_2 以下列格式先寫入帶上。

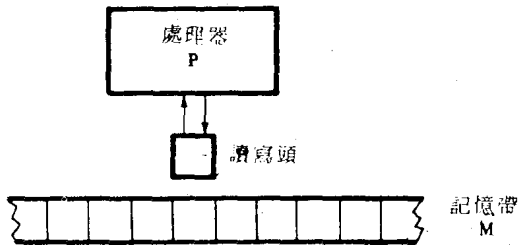


圖 1-2 Turing 機器。

$$\begin{array}{c}
 b \quad \underbrace{1 \quad 1 \quad \dots \quad 1}_{n_1} \quad b \quad \underbrace{1 \quad 1 \quad \dots \quad 1}_{n_2} \quad b
 \end{array}$$

讀寫頭位置於最左方 1 之空白，機器由 1 取代分離 n_1 及 n_2 間之空白，而後除去最左方之 1 即得此計算。最後格式為

$$\begin{array}{c}
 b \quad b \quad \underbrace{1 \quad 1 \quad \dots \quad 1 \quad 1 \quad 1 \quad \dots \quad 1}_{n_1 + n_2} \quad b
 \end{array}$$

出現於帶上即為所需之答案，其與輸入資料同為一進位格式。— 6 狀態 Turing 機器可執行此計算之行爲（程式）示於圖 1-3。

指令	註 解
$s_0 \quad b \quad R \quad s_0$	
$s_0 \quad 1 \quad R \quad s_1$	移讀寫頭經過 n_1 至右方。
$s_1 \quad 1 \quad R \quad s_1$	
$s_1 \quad b \quad 1 \quad s_2$	於 n_1 及 n_2 間以 1 代替空格。
$s_2 \quad 1 \quad L \quad s_3$	移動讀寫頭至左方。
$s_3 \quad 1 \quad L \quad s_3$	
$s_3 \quad b \quad R \quad s_4$	到達 1 的終點時，退格。
$s_4 \quad 1 \quad b \quad s_5$	刪除最左方之 1。
$s_5 \quad 1 \quad H \quad s_0$	$n_1 + n_2$ 現在帶上。

圖 1-3 加兩單進位數之 Turing 機器程式。

Turing 最引人注意的結果之一，為證明存在有一“通用” Turing 機器，其可進行任何 Turing 機器所作的每一計算。通用 Turing 機器本質上為 Turing 機器的模擬機器。若供給另一機器 M 之敘述 dM ，例列出指令集，通用機器即可模擬 M 的運作。其證實通用 Turing 機器僅需 t 個帶符號及 s 狀態處理器，而 $ts < 30$ 。此一機器有一極小指令集，然而確信可執行最有力“合理的”計算機所有的計算。（此宣告由美國數學家 Alonzo Church 發現故稱 Church 論證，由於合理的計算機符號為主觀，故無法精密論證。）

1-1-3 計算機的極限

不可解的問題 由於 Turing 機器具有進行所有可能的計算，吾人可以 Turing 機器定義一可計算函數的觀念。若 $f(X)$ 可對任何特定的 X 以 Turing 機器用有限步驟計算出，則此函數 f 為（實際上）可計算。令人驚訝的是有許多合理的函數以此方式却不可計算，此一事實首先由 Kurt Godel 於 1931 年之文獻提出〔10〕。

令 (M, X) 表 Turing 機器 M 及輸入帶 X ，即 X 為帶上最初資訊。吾人將說若 M 以輸入帶 X 開始。經有限步驟 M 暫停，即為 (M, X) 暫停。令 $f_H(M, X)$ 為對所有 Turing 機器及輸入帶定義為下。

$$f_H(M, X) = 1 \quad \text{若 } (M, X) \text{ 暫停}$$

$$f_H(M, X) = 0 \quad \text{若 } (M, X) \text{ 從不暫停}$$

其可證實函數 f_H 為不可計算者〔19〕。此暗示產生一通用方法以決定是否 (M, X) 暫停的問題為不可計算者（亦稱不可決定者）。此特定問題稱為 Turing 機器暫停問題。許多相關的問題亦認為不可解者。

不可解問題的存在為具有某些重要的暗示。通常由一無經驗寫程式者造成的錯誤為寫一具無窮環路的程式。因而在某些輸入條件無法暫停。如有一通用除錯程式及機器可決定是否此程式應暫停則甚為有用。Turing 機器暫停問題不可解性立即暗示不可能設計此類除錯工具。應注意的是此推斷僅當施於所有可能的程式方為真。對任何特定程式或程式組，其可能對暫停問題產生一解。因而存在不可解問題所引入的限制，