第二部份 SAP(SIMPLE-AS-POSSIBLE) 計算機

第 十 章

SAP-1

SAP (Simple-As-Possible) 計算機是針對初學者而設計之計算機。SAP 計算機的主要功能在使讀者明白計算機操作的基本原理，而不用做複雜的討論。但是，像 SAP 如此簡單的計算機也含有許多高深的觀念。爲了避免立即探討到繁瑣的原理，我們將逐步地說明三種不同代的 SAP 計算機。

SAP-1 是進入近代計算機的第一種型式。雖然它非常簡單，但是 SAP-1 對初學者而言是很重要的關鍵。要認真地探討本章，及瞭解 SAP-1 的結構和它的程式設計，明白其內部的電路操作，則你便能開始學習 SAP-2 計算機。

10-1 結 構

圖 10-1 爲 SAP-1 計算機之結構 (architecture)。所有接到 W 匯流排之暫存器輸出均爲三態；此可允許一般的資料傳送。所有不和 W 匯流排連接的暫存器輸出均爲雙態；這些輸出持續地推動所連接的方塊。

圖 10-1 僅強調 SAP-1 中的暫存器。就是因爲如此，所以我們並未單獨列出一被稱爲控制單元的方塊來包容所有的控制電路，也沒有另一被稱爲 I/O 單元的方塊來包含所有的輸入及輸出電路。

圖 10-1 中的若干暫存器我們早已在前面的例題與討論中說明過，想必已相當熟悉。下面我們將對每一方塊做一簡短的敘述，詳細的說明則留待後頭。

程式計數器

程式的第一個指令存在記憶體的開始位置，也就是位於二進位址 0000，第二個指令則位於位址 0001，第三個位於位址 0010，以下依此類推。程式計數器

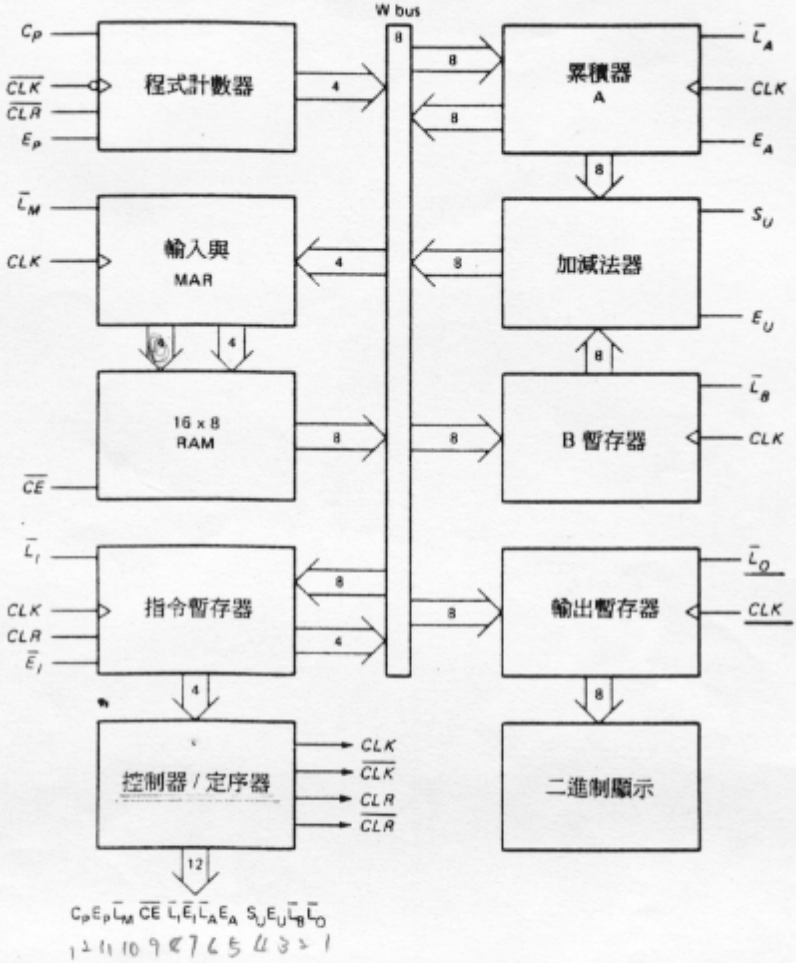


圖 10-1　SAP-1 計算機之構造

(program counter) 是控制單元的一部份，能由 0000 計數到 1111。其任務是將下一個要執行的指令位址送到記憶體內。它的工作情形如下：

在每次計算機運作之前，程式計數器都被重設爲 0000。當計算機開始運轉時，程式計數器送出 0000 位址到記憶體，然後程式計數器加 1 得到 0001。當第一個指令取出且執行後，程式計數器就送出 0001 位址到記憶體，然後再次地加 1。待第二個指令取出且執行後，程式計數器又送 0010 位址到記憶體。以此方式，程式計數器一直保持著下一個即將取出執行的指令所在位址。

程式計數器就好像一個人用手指頭指在一列指令上，命令先做這個，次做那個……。這也就是爲什麼程式計數器有時被稱爲指標器 (pointer) 的道理；它指向記憶體中存有某些重要程式資料的位址之上。

輸入與 MAR

程式計數器的下方是一稱爲輸入及 MAR 的方塊，它包含了 9-4 節中所討論的位址與資料開關暫存器。這些開關暫存器是輸入單元的一部份，能使我們將 4 位元的位址與 8 位元的資料送到 RAM 中。如果回想一下即可知在計算機運轉前，指令和資料字均已寫入記憶體內了。

記憶體位址暫存器 (memory address register，簡稱爲 MAR) 是 SAP-1 記憶體的一部份。在計算機運轉的時候，程式計數器中的位址被鬥到 MAR 之中。之後，MAR 將此 4 位元的位址送到正在執行讀取動作的 RAM 上去。

隨機讀取記憶體 (RAM)

這邊所提之 RAM 是一 16×8 的靜態 TTL RAM。如 9-4 節所討論的，我們可以經由位址及資料開關暫存器將程式存入 RAM。如此即可在計算機運轉前將程式與資料存入記憶體內。

在計算機運轉的時候，RAM 由 MAR 抓取 4 位元位址，且執行一讀取的動作。依此方式，儲存於 RAM 中的指令或資料字即出現於 W 匯流排之上，能供計算機的其他部份使用。

指令暫存器

指令暫存器 (instruction register) 屬於控制單元的一部份。從記憶體中讀出一指令，計算機必須執行記憶體讀取的動作。此動作將被定址的記憶體位置中的內容放於 W 匯流排。同時，指令暫存器也預備好在下一正向時脈邊緣載入指令。指令暫存器的內容分成兩個半位元組 (nibble)。上半部的位元組是屬雙態輸出，直接接到標有控制器 一 定序器的方塊上。下半部的位元組則是屬於三態輸出，當需要時，可由 W 匯流排讀取。

控制器 — 定序器

下一個方塊是控制器 — 定序器 (controller-sequencer)。每一次計算機運轉前，會送一 $\overline{CLR}$ 信號到程式計數器及一 CLR 信號到指令暫存器。這動作將程式計數器重設爲 0000，而且將指令暫存器的上一指令清除掉。

所有的緩衝暫存器都連接到 CLK 時脈訊號，其作用在使計算機的操作能同步，以保證一些該發生的事情能發生。也就是說，所有暫存器的資料轉移動作都發生在一相同的 CLK 信號正邊緣來臨時。注意到 $\overline{CLK}$ 信號同樣的也接到程式計數器上。

由控制器 — 定序器輸出的 12 位元，組成了一個控制計算機其他部份的字元 (如同一位告知其他人該做何事的監督者)。傳送控制字的 12 條線稱爲控制匯流排 (control bus)。

控制字的格式如下：

$$\mathbf{CON} = C_P E_P \overline{L}_M \overline{CE} \quad \overline{L}_I \overline{E}_I \overline{L}_A E_A \quad S_U E_U \overline{L}_B \overline{L}_O$$

這控制字決定了暫存器如何對下一個正 CLK 邊緣反應。舉例來說，一個高電壓 E_p 及低電壓 $\overline{L}_M$ 代表下一正時脈邊緣須將程式計數器的內容鬥到 MAR。再舉一例，一低電壓 $\overline{CE}$ 及一低電壓 $\overline{L}_A$ 代表被定址的 RAM 字元在下一正時脈邊緣時須送到累積器。稍後，我們會參考時序圖來說明這些資料是在何時及如何轉移的。

累積器

累積器 (accumulator，簡寫爲 A) 是由一組正反器構成，能在計算機運轉期間暫時儲存運算的結果。在圖 10-1 中，累積器有二個輸出，雙態輸出直接送到加減法器內，三態輸出連接到 W 匯流排。因此，累積器內 8 位元的字連續地供給 ALU；同樣的字在 E_A 爲高電壓時，出現在 W 匯流排上。

加減法器

SAP-1 用了一個 2 的補數加減法器。當圖 10-1 中的 S_U 爲低電壓，則加減法器便會執行加法運算。即

$$\mathbf{S = A + B}$$

當 S_U 爲高態，則執行 2 的補數減法。即

$$\mathbf{A} = \mathbf{A} + \mathbf{B}'$$

（應當還記得 2 的補數就等於十進位中的符號變換）。

需知加減法器是非同步的（未加時脈）；這表示只要輸入資料改變，其輸出的內容亦會立刻的改變。當 E_U 爲高電壓時，其內容會出現在 W 匯流排上。

B 暫存器

B 暫存器爲另一個緩衝暫存器，它會在算術運算時用到。一低態 $\overline{L}_B$ 及正時脈邊緣將 W 匯流排上的資料載入 B 暫存器內。B 暫存器的雙態輸出會驅動加減法器，提供累積器所要加入或減去的數字。

輸出暫存器

例 8-1 討論了輸出暫存器。在計算機結果運作時，累積器內包含了處理後的結果。此時我們必需將此結果送到外界。當 E_A 爲高電壓而 $\overline{L}_O$ 爲低電壓時，下一個正時脈邊緣會將累積器的內容送到輸出暫存器。

輸出暫存器通常又稱之爲輸出埠 (output port)，因爲處理完的資料可經此暫存器輸出至外界。在微算機中，輸出埠是與介面電路 (interface circuit) 相接的。介面電路是用來推動印表機、陰極射線管 (CRT)，電傳打字機等等的週邊裝置（一個介面電路能推動一種裝置）。

二進位顯示幕

二進位顯示幕是由一行八個發光二極體 (LED) 所構成的。因爲每一個 LED 連接至輸出埠的一個正反器上，此二進位顯示幕可示出輸出暫存器之內容。因此，在將答案由累積器傳送到輸出埠之後，可看到二進位形式的答案。

摘　要

SAP-1 之控制單元包括有程式計數器、指令暫存器，及產生控制字、清除信號、時脈信號的控制器 — 定序器。SAP-1 的 ALU 包含有一累積器、一加減法器，一 B 暫存器。SAP-1 的記憶體包含了 MAR 及一個 16×8 的 RAM。I/O 單元

包含有可規劃的輸入開關、輸出埠及二進位顯示幕。

10-2 指令集

計算機要等到有人寫入程式後才能執行工作，否則只是一堆沒用的硬體電路而已。在計算機運轉之前，要先送入可以一步一步操作的指令至記憶體內。但是，在作程式設計之前，要先瞭解指令集 (instruction set)，以明白此計算機所能操作的基本運算。以下說明 SAP-1 之指令集。

LDA

在第九章提到記憶體中的資料字能以符號 $\mathbf{R}_0$, $\mathbf{R}_1$, $\mathbf{R}_2$ 等來代表。$\mathbf{R}_0$ 代表存於位址 00H，$\mathbf{R}_1$ 存於位址 01H，$\mathbf{R}_2$ 存於位址 02H 等等。

LDA 代表「將資料載入累積器內」(load the accumulator)。一完整的 LDA 指令應包含有資料所存之十六進制位址。例如 LDA 8H 表示「將記憶體位置 8H 中的內容載入累積器中」。於是，如果

$$\mathbf{R}_8 = 1111\ \ 0000$$

執行完 LDA 8H 的結果爲

$$\mathbf{A} = 1111\ \ 0000$$

同樣的，LDA AH 表示「將位址爲 AH 的記憶體內容載入至累積器內」；LDA FH 表示「將記憶體位址爲 FH 的內容送至累積器。」

ADD

ADD 是另一個 SAP-1 內之指令。一完整的 ADD 指令應包含有欲相加資料的儲存位址。舉例而言，ADD 9H 表示「將記憶體位置爲 9H 的內容加到累積器內」。相加的結果取代了累積器原來的內容。

再舉一例，假設累積器中有十進制數字 2，而記憶體位置 9H 有十進制數字 3。則

$$\mathbf{A} = 0000\ \ 0010$$
$$\mathbf{R}_9 = 0000\ 0011$$

執行 ADD 9H 時，會發生下列情況。首先 R_9 載入 B 暫存器，使

B = 0000 0011

而且幾乎在同時加減法器的輸出產生了 **A** 與 **B** 的和

SUM = 0000 0101

接下來，此和才被載入累積器而得

A = 0000 0101

以上的方式是整個 ADD 指令的執行過程。被指定的 RAM 之內容送到 B 暫存器，而加減法器的輸出則送到累積器。這就是爲什麼執行了 ADD 9H，能將 R_9 的內容加至累積器內，執行 ADD FH 會將 R_F 加到累積器的原因。

SUB

SUB 是 SAP-1 的另一指令。一完整 SUB 指令應包括有欲減去之數值的所在位址。舉例而言，SUB CH 表示「將累積器中的內容減去記憶體位置 CH 的內容」。加減法器所得之差會取代累積器中原先的內容。

再舉一例，假設累積器中有十進制數字 7，而記憶體位置 CH 的內容爲十進制數字 3。則

A= 0000 0111
R_C = 0000 0011

執行了 SUB CH 後，會發生下列動作。首先 R_C 載入 B 暫存器而得

B = 0000 0011

且幾乎同時在加減法器得到 **A** 和 **B** 的差爲

DIFF = 0000 0100

之後，這個差值被載入累積器，使得

A = 0000 0100

上面的方式說明了 SUB 指令的所有執行過程；被指定 RAM 的內容送到 B 暫存器，而加減法器的輸出送到累積器。這就是爲什麼執行了SUB CH 後，會將累

積器的內容減去 R_C 的道理，同理執行了 SUB EH 時，會從累積器減去 R_E。

OUT

OUT 指令告訴 SAP-1 將累積器內之內容傳送至輸出埠上。在 OUT 指令操作完畢後，可看到所得的結果。

OUT 本身是一個完整的指令；也就是在使用 OUT 指令時，不須要包括一個位址在內，因爲此指令，並不牽涉到記憶裝置。

HLT

HLT 表示停止，此指令告訴計算機停止操作，不必處理資料。HLT 表示一個程式的結束。類似於句號代表一個句子的結尾。在每一個SAP-1 程式的後面必須使用 HLT 指令；否則，計算機會送出沒有用的資料 (由於不適當的處理過程產生的無意義結果)。

HLT 本身亦是完整的指令，使用時不用包括一個 RAM 字的位址，因爲此指令並不牽涉到記憶裝置。

記憶體相關指令

LDA , ADD 和 SUB 均稱爲記憶體相關指令 (memory-reference instruction) 因爲它們均用到了存於記憶體內部的資料。相反的，OUT 及 HLT 就不是記憶體相關指令，因爲他們並未包括存於記憶體內的資料。

助憶符號

LDA , ADD , SUB , OUT 與 HLT 爲 SAP-1 之指令集，像這種簡寫的指令，稱爲助憶符號 (mnemonics，有助於記憶)。在計算機作業上，助憶符號是非常普遍的，因爲它們能提示我們當這些指令執行時會產生什麼樣的動作。表 10-1 摘錄了 SAP-1 的指令集。

表10-1 指令集

助憶符號	操 作
LDA	將 RAM 資料送至累積器內
ADD	將 RAM 資料加至累積器內
SUB	將累積器內資料減去 RAM 內資料
OUT	將累積器之字送至輸出暫存器上
HLT	停止

8080 與 8085

8080 是第一個 被廣泛使用的微處理機。它共有 72 個指令。8085 是 8080 的改良型，具有所有的 8080 指令。爲了使 SAP 更實用一點， SAP 的指令必須和 8080/8085 的指令相容。換言之，SAP-1 的指令 LDA , ADD , SUB , OUT 及 HLT 均是 8080/8085 的指令。同樣的，SAP-2 及 SAP-3 的指令也都是 8080/8085 指令集的一部份。學習 SAP 的指令能夠幫助我們更進一步學習 8080 和 8085 的指令。一旦學會了 8080/8085 的指令集，若想再學習其他的微處理機時就簡單多了。

例題 10-1

以下是一以助憶符號出現的 SAP-1 程式

位址	助憶符號
0H	LDA 9H
1H	ADD AH
2H	ADD BH
3H	SUB CH
4H	OUT
5H	HLT

較高位址的資料爲

位址	助憶符號
6H	FFH
7H	FFH
8H	FFH
9H	01H
AH	02H
BH	03H

CH	04H
DH	FFH
EH	FFH
FH	FFH

試說明每一指令的操作情形如何？

解：此程式存於記憶體中的較低位址，位於 0H 到 5H 之間。第一個指令將記憶體位置 9H 的內容載入到累積器內，所以累積器的內容變爲：

$$A = 01H$$

第二個指令將記憶體位置 AH 的內容加到累積器而得到之累積器新值爲

$$A = 01H + 02H = 03H$$

同樣的，第三個指令將記憶體 BH 位置的內容加到累積器：

$$A = 03H + 03H = 06H$$

SUB 指令將累積器內容減去記憶體 CH 位置的內容而得到：

$$A = 06H - 04H = 02H$$

OUT 顯示幕累積器的內容送到輸出埠，而二進位顯示幕顯示：

0000 0010

HLT 指令停止了資料處理的動作。 ▮

10-3 設計 SAP-1 的程式

要將指令及資料字送到 SAP-1 記憶體內，我們必須使用一些計算機能認識的碼。表 10-2 示出了用在 SAP-1 的碼。數字 0000 代表 LDA，0001 代表 ADD，0010 代表 SUB，1110 代表 OUT，1111 代表 HLT。因爲這些碼告知了計算機應該做那些動作，所以稱它爲操作碼 (operation code)，簡稱「OP 碼」。

就像前面所討論的，圖 9-7 所表示的位址和資料開關允許我們將程式送到 SAP-1 記憶體。這些開關之設計使得當它位於上端 (U) 時產生一個 1，在下端的位置 (D) 產生 0。當由開關輸入指令時，OP 碼會進到上半位元組 (nibble)，而且運算元 (operand，指令的其餘部份) 會進到下半位元組。

表 10-2 SAP-I 的操作碼

助憶符號	操作碼
LDA	0000
ADD	0001
SUB	0010
OUT	1110
HLT	1111

舉個例子來說，如果我們要存入下面的指令：

位址	指令
0H	LDA FH
1H	ADD EH
2H	HLT

首先，將每一個指令變成二進位，如下面所表示：

LDA FH = 0000 1111
ADD EH = 0001 1110
HLT = 1111 XXXX

在第一個指令當中，0000 是 LDA 的 OP 碼，而 1111 是 FH 的二進位形式。第二個指令當中，0001 是 ADD 的 OP 碼，1110 是 EH 的二進位形式。在第三個指令當中，1111 是 HLT 的 OP 碼，XXXX 表示任意值，因爲 HLT 並不是記憶體相關指令。

其次是設定位址和資料開關，如下面所表示：

位址	資料
下下下下	下下下下 上上上上
下下下上	下下下上 上上上下
下下上下	上上上上 XXXX

然後每個位址和資料都被設定好的時候，就可以按下寫入 (write) 按鈕。因爲「下」存入二進數 0，而「上」存入二進數 1，所以前面三個記憶體位置的內容是：

位址	內容
0000	0000 1111
0001	0001 1110
0010	1111 XXXX

最後一點，就是利用組合語言 (Assembly language) 撰寫程式時，可以直接使用助憶符號。而機器語言 (machine language) 就需要用一連串的 0 和 1 來表示。下面的例子指出這兩種語言的不同處。

例題 10-2

將例 10-1 中的程式翻譯成 SAP-1 的機器語言。

解：將例 10-1 的程式重列於下：

位址	指令
0H	LDA 9H
1H	ADD AH
2H	ADD BH
3H	SUB CH
4H	OUT
5H	HLT

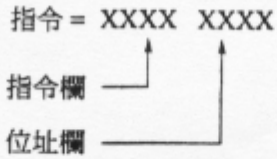
這個程式是以組合語言撰寫，爲了將其轉換成機器語言，我們將它譯爲 0 及 1，如下所示：

位址	指令
0000	0000 1001
0001	0001 1010
0010	0001 1011
0011	0010 1100
0100	1110 XXXX
0101	1111 XXXX

現在，程式是以機器語言的形式出現。

任一程式是以機器語言形式撰寫的程式，就稱爲「目的程式」(object program)。而以助憶符號寫成的程式就稱爲「原始程式」(source program；根源程式)。在 SAP-1 中，如想利用位址和資料開關將程式送進計算機，則操作員首先須將原始程式譯爲目的程式。

最後要說明的一點爲在 SAP-1 中，機器語言指令之 4 位元 MSB 是在說明運算的動作，4 位元 LSB 是表示位址；今後，我們稱此種 MSB 爲指令欄 (instruction field)，LSB 稱爲位址欄 (address field)。用符號表示爲：



例題 10-3

試撰寫一 SAP-1 的程式以求出下面的算術問題之解

$$16 + 20 + 24 - 32$$

其中數字爲十進制形式。

解：方法之一是使用前面例題中的程式，將資料 (16 , 20 , 24 , 32) 存在記憶體位置 9H 到 CH。參考附錄 2，你能將十進制數變換成十六進制數，而得到下面的組合語言形式：

位址	內容
0H	LDA 9H
1H	ADD AH
2H	ADD BH
3H	SUB CH
4H	OUT
5H	HLT
6H	XX
7H	XX
8H	XX
9H	10H
AH	14H
BH	18H
CH	20H

其機器語言之形式爲：

位址	內容
0000	0000 1001

0001	0001 1010
0010	0001 1011
0011	0010 1100
0100	1110 XXXX
0101	1111 XXXX
0110	XXXX XXXX
0111	XXXX XXXX
1000	XXXX XXXX
1001	0001 0000
1010	0001 0100
1011	0001 1000
1100	0010 0000

請注意，程式存在資料的前面。也就是說程式位於低記憶位址，而資料位於高記憶位址。這在 SAP-1 中是必要的，因爲程式暫存器將位址 0000 當做第一指令，而 0001 當做第二指令。 ▮

例題 10-4

將上一例題中的程式及資料變換成十六進制的形式。

解：

位址	內容
0H	09H
1H	1AH
2H	1BH
3H	2CH
4H	EXH
5H	FXH
6H	XXH
7H	XXH
8H	XXH
9H	10H

AH	14H
BH	18H
CH	20H

這種形式的程式及資料仍算是機器語言。

附帶提到，負數是以 2 的補數形式載入。例如 −03H 是以 FDH的形式輸入。

10-4 提取週期

控制單元是計算機能自動操作的主要關鍵之處。控制單元產生控制字以提取及執行各個指令。當提取及執行每個指令的時候，計算機須經過不同的時序狀態 (timing state，也稱 T 狀態)，在時序之期間內，暫存器的內容被改變。以下說明各種不同的 T 狀態。

環式計數器

我們在前面已有提到 SAP-1 的環式計數器 (參閱圖 8-16 的電路)。圖 10-2(a) 是環式計數器的符號，其輸出有

$$\mathbf{T} = T_6\,T_5\,T_4\,T_3\,T_2\,T_1$$

在計算機運轉的初期，環式計數器之內容爲：

$$\mathbf{T} = 000001$$

以後依時脈的順序所產生的環字是

$$\mathbf{T} = 000010$$
$$\mathbf{T} = 000100$$
$$\mathbf{T} = 001000$$
$$\mathbf{T} = 010000$$
$$\mathbf{T} = 100000$$

然後，環式計數器重設回到 000001，重複循環。每一個環式代表一個機器狀態。

圖 10-2(b) 示出環式計數器輸出的時序脈波。原先的 T_1 狀態是由一個負向時脈邊緣開始，到次一個負向時脈邊緣結束。在此 T_1 狀態期間，環式計數器的輸出

訊號 T_1 爲高電壓。

在下一個狀態期間，T_2 爲高電壓；以下的狀態是 T_3 爲高電壓；然後 T_4 爲高電壓，依此類推。可看出，此環式計數器共產生六個 T 狀態。每一個指令在此六個 T 狀態期間被提取與執行完畢。

注意到，CLK 信號的正向邊緣正好在每個 T 狀態的中央發生。此種安排的重要性在後面會提到。

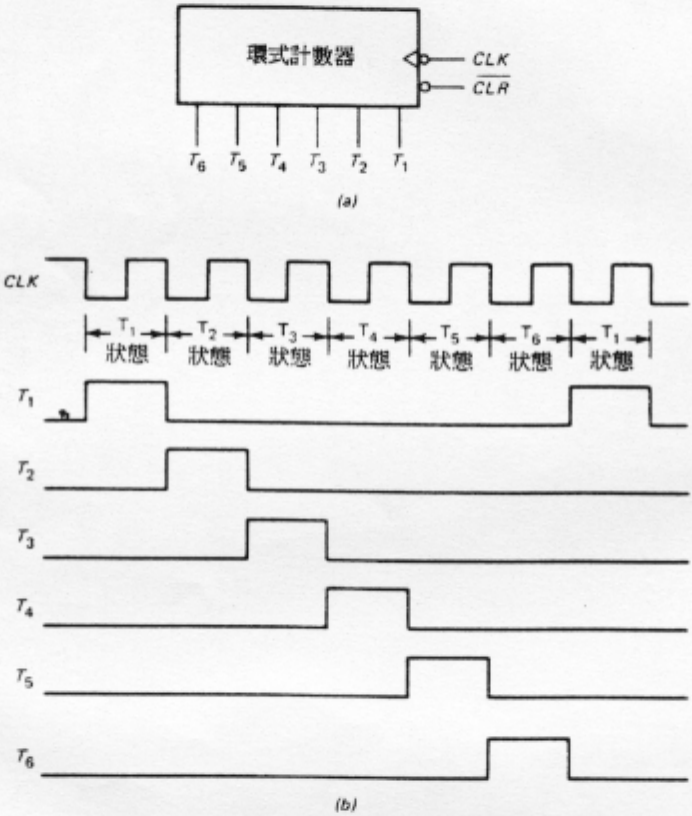


圖 10-2　環式計數器：(a) 符號；(b) 時脈及時序信號

位址狀態

T_1 狀態又稱作位址狀態 (address state) 因爲程式計數器內的位址在此狀態時傳到記憶體位址暫存器 (MAR) 內。圖 10-3(a) 示出了在此狀態下發生作用的計算機部份。(發生作用的部份較明亮，有陰影的部份是不發生作用的)。

在位址狀態期間，E_p 和 $\overline{L}_M$ 有作用；其他的控制位元均無作用。這代表控制器—定序器正送出一控制字爲

$$\begin{aligned} CON &= C_P\,E_P\,\overline{L}_M\,\overline{CE}\quad \overline{L}_I\,\overline{E}_I\,\overline{L}_A\,E_A\quad S_U\,E_U\,\overline{L}_B\,\overline{L}_O \\ &= 0\ \ 1\ \ 0\ \ 1\quad 1\ \ 1\ \ 1\ \ 0\quad 0\ \ 0\ \ 1\ \ 1 \end{aligned}$$

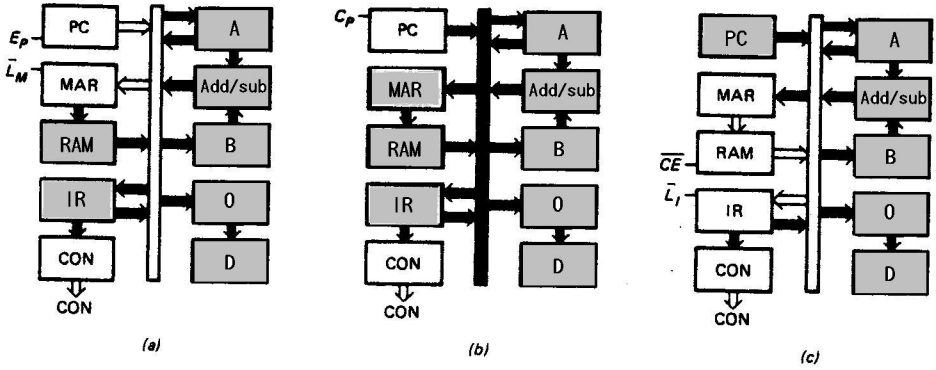


圖 10-3 提取週期：(a) T_1 狀態；(b) T_2 狀態；(c) T_3 狀態

遞增狀態

圖 10-3(b) 示出了在 T_2 狀態時產生動作的 SAP-1 部份。此一狀態稱爲遞增狀態 (increment state)，因爲，程式計數器在此狀態時被加 1。於此狀態時，控制器—定序器產生的控制字爲：

$$\begin{aligned} CON &= C_P\,E_P\,\overline{L}_M\quad \overline{CE}\quad \overline{L}_I\,\overline{E}_I\,\overline{L}_A\,E_A\quad S_U\,E_U\,\overline{L}_B\,\overline{L}_O \\ &= 1\ \ 0\ \ 1\quad 1\quad 1\ \ 1\ \ 1\ \ 0\quad 0\ \ 0\ \ 1\ \ 1 \end{aligned}$$

此時，C_p 有作用

記憶狀態

T_3 狀態稱作記憶狀態，因爲被指定的 RAM 位置中的指令，由記憶裝置送到指令暫存器。圖 10-3(c) 示出了在此狀態時 SAP-1 產生動作的部份。在此狀態中有作用的控制位元是 $\overline{CE}$ 和 $\overline{L}_I$，而控制器—定序器的輸出字如下：

$$\mathbf{CON} = C_P\,E_P\,\overline{L}_M\,\overline{CE}\quad \overline{L}_I\,\overline{E}_I\,\overline{L}_A\,E_A\quad S_U\,E_U\,\overline{L}_B\,\overline{L}_O$$
$$= 0\ 0\ 1\ 0\quad 0\ 1\ 1\ 0\quad 0\ 0\ 1\ 1$$

提取週期

位址、遞增及記憶狀態統稱爲 SAP-1 的提取週期 (fetch cycle)。在位址狀態期間，E_p 及 $\overline{L}_M$ 有作用，這代表程式計數器經由 W 匯流排設定 RAM。如圖 10-2(b) 所示，一正時脈邊緣發生在位址狀態的中央，這使得 PC 的內容載入 RAM 之內。

在遞增狀態的時候，只有 C_p 有作用，它使得程式計數器在正向時脈邊緣來臨時能增加 1。

在記憶狀態的時候，$\overline{CE}$ 及 $\overline{L}_I$ 有作用。於是被定址的 RAM 內容就經由 W匯流排載入指令暫存器內。在記憶狀態的中央，一正時脈的邊緣就將被定址的 RAM 內容送到指令暫存器內。

10-5 執行週期

接著的三個狀態 (T_4, T_5 和 T_6) 是 SAP-1 的執行週期 (execution cycle)。在執行週期時，暫存器會隨著不同的指令產生不一樣的動作，舉例來說，LDA 9H 和 ADD BH 所需的暫存器轉移動作便不相同。所依據的就是各種不同的指令的控制常式 (control routine)。

LDA 常式

爲了能有一較具體的討論，我們假設指令暫存器已存入指令 LDA 9H：

IR = 0000 1001

在 T_4 週期間，指令欄 0000 送到控制器 — 定序器加以解碼，位址欄 1001 被送到 MAR。圖 10-4(a) 示出了在 T_4 狀態期間，SAP-1 發生作用的部份。注意到 $\overline{E}_I$ 及 $\overline{L}_M$ 有作用；而其他的控制位元均無作用。

在 T_5 狀態期間，$\overline{CE}$ 和 $\overline{L}_A$ 變爲低電壓。這代表被定址的 RAM 資料字在下一正時脈邊緣時會送到累積器。看圖 10-4(b)。

T_6 是一無動作狀態。在這第三個執行狀態間，所有暫存器均無作用 (圖 10-4(c))。這表示控制器 — 定序器會送出一個控制字，其中的所有位元均無作用。「NOP」(讀作 no op)；無運算代表「無動作」。LDA 常式中的 T_6 狀態是一個 NOP。

圖 10-5 示出了提取及 LDA 常式的時序圖。在 T_1 狀態期間，E_p 及 $\overline{L}_M$ 有作用；這狀態中間的正時脈邊緣會將程式計數器內的位址送到MAR。在 T_2 狀態之間，C_p 有作用，使得程式計數器在正時脈邊緣時增加 1。在 T_3 狀態間，$\overline{CE}$ 及 $\overline{L}_I$ 有作用，當正時脈邊緣出現時，被定址的 RAM 的內容會送到指令暫存器。LDA 的執行從 T_4 狀態開始，這時候 $\overline{L}_M$ 及 $\overline{E}_I$ 有作用；而在正時脈邊緣，指令暫存器的位址欄就送到 MAR。在 T_5 狀態間，$\overline{CE}$ 和 $\overline{L}_A$ 有作用；表示被定址的 RAM 資料會在正時脈邊緣時送到累積器。而 LDA 的 T_6 狀態則是 NOP。

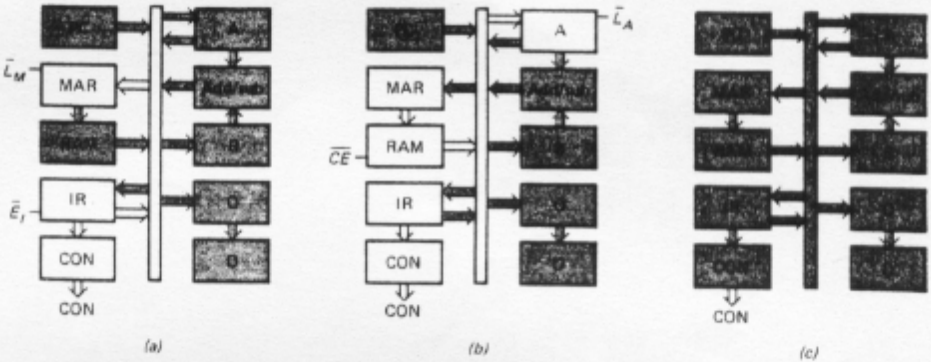


圖 10-4 LDA 常式：(a) T_4 狀態；(b) T_5 狀態；(c) T_6 狀態

ADD 常式

假如在提取週期結束的時候，指令暫存器內的指令是 ADD BH：

IR = 0001 1011

在 T_4 狀態時指令欄會被送到控制器 — 定序器，而位址欄則送到 MAR(見圖 10-6(a))。這種狀態間 $\overline{E}_I$ 及 $\overline{L}_M$ 有作用。

在 T_5 狀態時，控制位元 $\overline{CE}$ 及 $\overline{L}_B$ 有作用，這種允許了被定址的 RAM 資料字能設定 B 暫存器 (圖 10-5(b))。如前所述，載入的動作發生於狀態的中央，也就

是，B 暫存器 CLK 輸入端的時脈信號處於正邊緣時。

在 T_6 狀態期間， E_U 及 $\overline{L}_A$ 有作用；於是加減法器的輸出接到累積器（圖 10-6(c)）。這種狀態的中央處，正時脈邊緣會將總和載入累積器內。

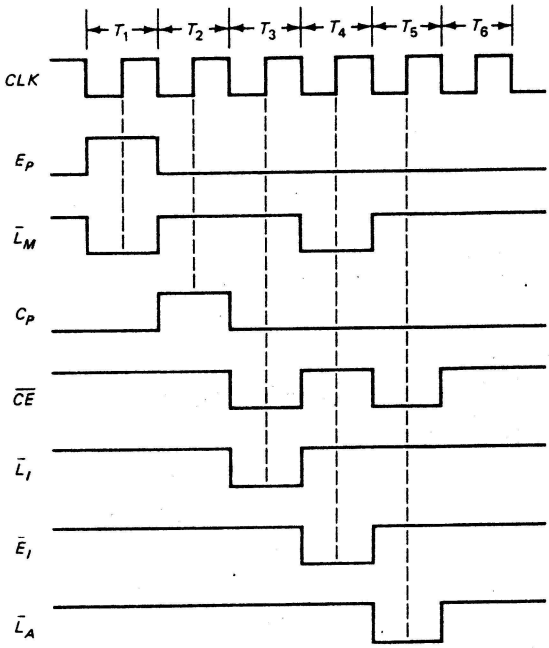


圖 10-5　提取及 LDA 時序圖

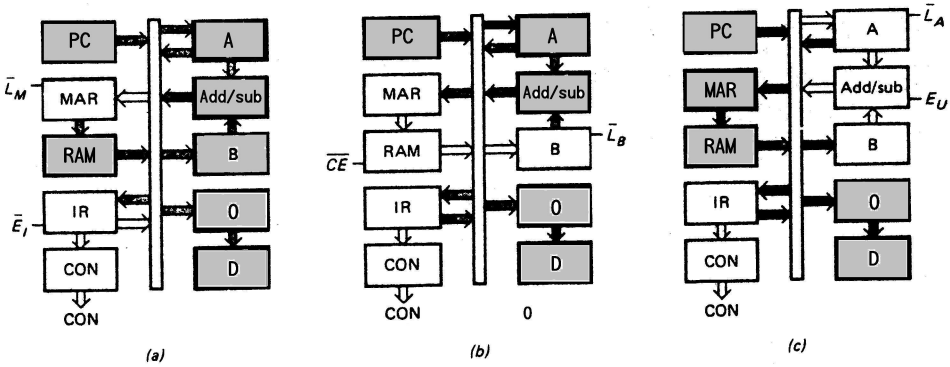


圖 10-6　ADD 及 SUB 常式：(a) T_4 狀態；(b) T_5 狀態；(c) T_6狀態。

附帶的，預備時間 (setup time) 與傳輸延遲時間的特性，在這執行狀態期間避免了累積器的競走現象。當 (圖 10-6(c)) 中的正時脈邊緣到達後，累積器內容才改變，連帶的迫使加減法器的內容改變。這新的內容會返回累積器的輸入端，但是這種新內容返回時已經過了兩個傳輸延遲時間 (一個由累積器，一個由加減法器引起。這個時候要設定累積器已經太遲了。這就可以防止累積器競走 (racing) 的現象 (在同一正時脈邊緣載入多次的現象)。

圖 10-7 示出了提取及 ADD 常式的時序圖。提取常式和以前相同：T_1 狀態將 PC 位址載進 MAR；T_2 狀態將程式計數器加 1；T_3 狀態將指定的指令送到指令暫存器。

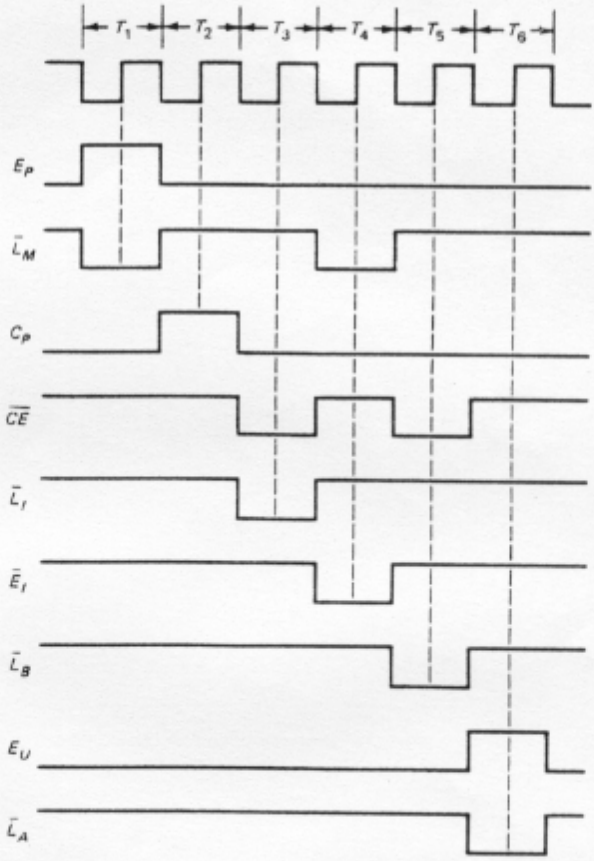


圖 10-7 提取和 ADD 常式時序圖

在 T_4 狀態間，$\overline{E}_I$ 及 $\overline{L}_M$ 有作用；在下一正時脈邊緣，指令暫存器中的位址傳送到 MAR。在 T_5 狀態間，$\overline{CE}$ 和 $\overline{L}_B$ 有作用；於是被定址的 RAM 的資料在這狀態的中央時送到 B 暫存器。而在 T_6 狀態間 E_U 和 $\overline{L}_A$ 有作用；當正時脈邊緣來到的時候，加減法器輸出的和即存進累積器內。

SUB 常式

SUB 常式和 ADD 常式很類似。圖 10-6(a) 及 (b) 示出了在 T_4和 T_5 狀態間 SAP-1 中有作用的部份。在 T_6 狀態間，一個高電壓 S_U 信號送到圖 10-6(c) 的加減法器。時序圖差不多和圖 10-7 一樣。注意到在 T_1 到 T_5 之間 S_U 爲低電壓，而在 T_6 狀態時 S_U 爲高電壓。

OUT 常式

假設提取週期結束時，指令暫存器內容是 OUT 指令。即

$$\mathbf{IR} = 1110\ \ XXXX$$

指令欄送到控制器—定序器加以解碼。而控制器—定序器則送出控制字，將累積器的內容送到輸出暫存器內。

圖 10-8 示出了執行 OUT 指令時 SAP-1 發生作用的部份。因爲 E_A 和 $\overline{L}_O$ 有作用，所以下一正時脈邊緣時，會在 T_4 狀態間將累積器內容送到輸出暫存器。而 T_5 及 T_6 均爲 nop。

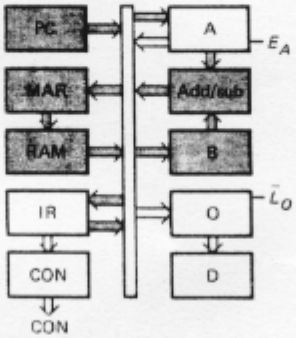


圖 10-8　OUT 指令的 T_4 狀態

圖 10-9 是提取及 OUT 常式的時序圖。提取週期跟前面一樣都是：位址狀態、遞增狀態及記憶狀態。在 T_4 狀態間，E_A 及 $\overline{L_O}$ 有作用；這使得累積器內容會在正時脈邊緣時送到輸出暫存器內。

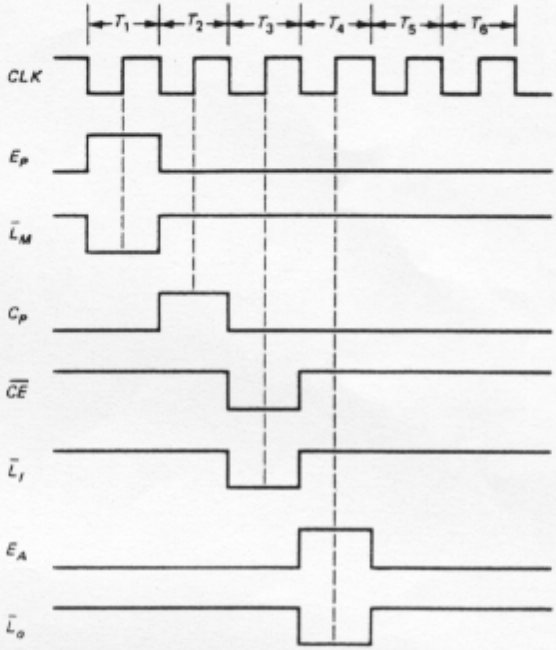


圖 10-9 提取及 OUT 時序圖

HLT

HLT 不需控制常式，因爲在執行 HLT 指令時不需要用到暫存器。當 IR 內容是

IR = 1111 XXXX

時指令欄 1111 通知控制器—定序器停止處理資料。控制器—定序器藉著關閉時脈來停止計算機的動作（線路在以後討論）。

機器週期和指令週期

SAP-1 有六個 T 狀態（三個提取及三個執行狀態）。這六個狀態合稱作機器週

期 (machine cycle)(圖 10-10(a))。提取及執行任一個指令都需要花費一個機器週期。SAP-1 的時脈頻率是 1 KHz，相當於一個週期 (period) 爲 1 ms。所以一個 SAP-1 機器週期要花 6 ms。

SAP-2 有些不同，因爲它有一部份指令在提取及執行時會超過一個以上的機器週期。圖 10-10(b) 示出了一個需要兩個機器週期的指令。前三個 T 狀態是提取週期；而後九個 T 狀態爲執行週期。這是因爲雙機器週期的指令比較複雜，因此需要額外的 T 狀態來完成。

提取及執行一指令所需的 T 狀態數稱之爲指令週期 (instruction cycle)。在 SAP-1 中指令週期和機器週期一樣。而 SAP-2 及其他計算機的指令週期可能等於或大於兩個以上的機器週期，如圖 10-10(b) 所示。

8080 和 8085 的指令週期所佔用的機器週期在一個到五個之間。

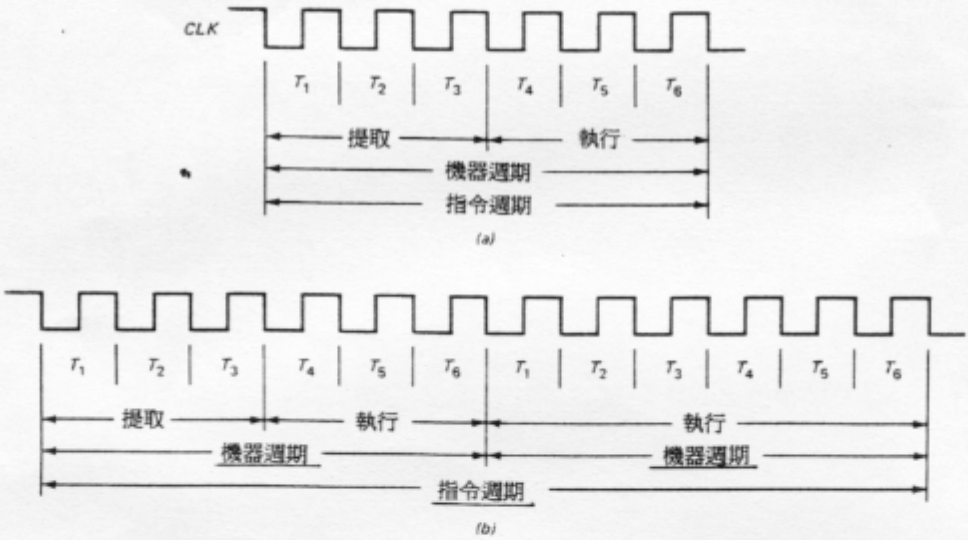


圖 10-10 (a) SAP-1 的指令週期；(b) 佔有兩個機器週期的指令週期

例題 10-5

8080/8085 程式手冊上說明了提取及執行 — LDA 指令要花掉 13 個 T 狀態。如果系統時脈的頻率是 2.5 MHz，那麼這指令週期需要多少時間？

解：時脈的週期是

$$T = \frac{1}{f} = \frac{1}{2.5\ \text{MHz}} = 400\ \text{ns}$$

所以每一 T 狀態維持 400 ns。因爲要花掉十三個 T 狀態來提取及執行 LDA 指令，所以其指令週期佔有

$$13 \times 400\ \text{ns} = 5{,}200\ \text{ns} = 5.2\ \mu s$$ ▮

例題 10-6

圖 10-11 示出了 SAP-1 的六個 T 狀態，在每個 T 狀態的中央，正時脈的邊緣才來到。這一點有什麼重要性？

解：SAP-1 是一個匯流排結構的計算機（目前最常用的型式）。可以使每個暫存器間經由匯流排來溝通。但是，必須預備時間與保持時間都符合的時候，才能正確的傳輸於各暫存器間。在輸進到暫存器內之前先等待半個週期，這滿足了預備時間；在輸進暫存器之後等待半個週期，這滿足了保持時間。這是在每一 T 狀態中央，才有正向時脈邊緣到達每一暫存器的道理。（圖 10-11）。

在輸進暫存器之前，等待半個週期，還有另外一個原因。當想輸出資料的暫存器 *ENABLE* 輸入端變成有作用的時候，這暫存器內的內容立即輸到 W 匯流排上。電路間的雜散電容和電感抑制匯流排上的線，使其不能立刻到達正確的電壓位準。換句話說，在 W 匯流排上有暫態現象產生。如想有可靠的資料，必須等到這些資料在匯流排線上穩定後才可以。在時脈邊緣到達之前有半個週期的延遲，足夠讓所有的暫態現象消除。 ▮

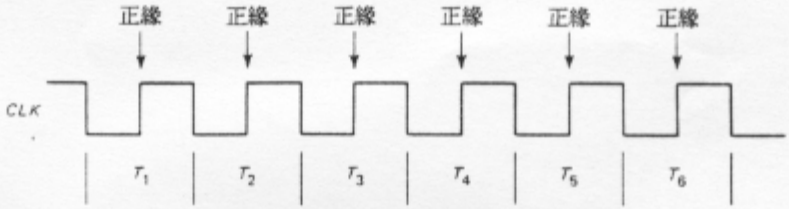


圖 10-11 在 T 狀態的中央正時脈邊緣才發生

10-6 微程式

我們立刻要分析 SAP-1 計算機的線路圖，但是我們首先要將每個 SAP-1 的指令執行動作摘要地列在一稱作微程式 (microprogram) 的表中。

微指令

控制器 — 定序器在每個 T 狀態或時脈週期間會送出一個控制字。這些字告知了計算機的其他部門該做些什麼事。因爲一個控制字只是整理資料處理步驟裡的一小部份，所以一控制字又稱爲一微指令(microinstruction)。觀察 SAP-1 的方塊圖 (圖 10-1)，我們可以看出有一連串穩定的微指令由控制器 — 定序器輸出到其他的 SAP-1 電路。

巨指令

我們曾用來撰寫程式的指令 (LDA , ADD , SUB , ……等) 有時稱作「巨指令」(macroinstruction) 以和微指令有所區別。SAP-1 每一巨指令是由三個微指令構成。舉例來說，LDA 指令包含表 10-3 中的微指令。爲了簡化這些微指令的外觀，我們用了示於表 10-4 的十六進制的形式。

表 10-3

巨指令	狀態	$C_P E_P \overline{L}_M \overline{CE}$	$\overline{L}_I \overline{E}_I \overline{L}_A E_A$	$S_U E_U \overline{L}_B \overline{L}_O$	動作訊號
LDA	T_4	0 0 0 1	1 0 1 0	0 0 1 1	$\overline{L}_M$, $\overline{E}_I$
	T_5	0 0 1 0	1 1 0 0	0 0 1 1	$\overline{CE}$, $\overline{L}_A$
	T_6	0 0 1 1	1 1 1 0	0 0 1 1	None

表 10-4

巨指令	狀態	CON	動作訊號
LDA	T_4	1A3H	$\overline{L}_M$, $\overline{E}_I$
	T_5	2C3H	$\overline{CE}$, $\overline{L}_A$
	T_6	3E3H	None

表 10-5 示出了 SAP-1 的微程式，即列出所有的巨指令及執行巨指令時所需的微指令。這個表總摘了 SAP-1 指令的執行常式。

表 10-5 SAP-1 微程式

巨指令	狀態	CON	動作訊號
LDA	T_4	1A3H	$\overline{L}_M$, $\overline{E}_I$
	T_5	2C3H	$\overline{CE}$, $\overline{L}_A$
	T_6	3E3H	None
ADD	T_4	1A3H	$\overline{L}_M$, $\overline{E}_I$
	T_5	2E1H	$\overline{CE}$, $\overline{L}_B$
	T_6	3C7H	$\overline{L}_A$, E_U
SUB	T_4	1A3H	$\overline{L}_M$, $\overline{E}_I$
	T_5	2E1H	$\overline{CE}$, $\overline{L}_B$
	T_6	3CFH	$\overline{L}_A$, S_U, E_U
OUT	T_4	3F2H	E_A, $\overline{L}_O$
	T_5	3E3H	None
	T_6	3E3H	None

† CON = $C_pE_p\overline{L}_M\overline{CE}$ $\overline{L}_I\overline{E}_I\overline{L}_AE_A$ $S_UE_U\overline{L}_B\overline{L}_O$.

10-7 SAP-1 的線路圖

在本節中我們要研究 SAP-1 的整個線路圖。圖 10-12 到 10-15 示出了 SAP-1 的所有晶片、接線及信號。在以後的討論中，我們均會利用這些圖。附錄 4 列出一些比較複雜晶片的詳細說明。

程式計數器

圖 10-12 中的 C1 , C2 , C3 晶片構成了 SAP-1 的程式計數器。晶片 C1 是編號爲 74LS107 的二合一 *JK* 主僕式正反器，它產生了比較高的兩位址位元。晶片 C2 也是一個 74LS107，能產生比較低的兩位址位元。晶片 C3 是編號 74LS126 四合一的三態開關；這晶片提供了程式計數器三態輸出的能力。

計算機開始運轉時，一低電壓的 $\overline{CLR}$ 信號將程式計數器重設作 0000。在 T_1 狀態間，一高電壓的 E_p 將位址放在 W 匯流排。而在 T_2 狀態間，高電壓 C_p 加在程式計數器；在這狀態中央，負的 $\overline{CLK}$ 邊緣 (相當於正的 *CLK* 邊緣) 將程式計數器加 1。

在 T_3 到 T_6 狀態間，程式計數器不作用。

MAR

晶片 C4 是編號 74LS173 的 4 位元緩衝暫存器；它充當 MAR。注意到其腳 1 及 2 接地；這使得三態輸出變成雙態輸出。換句話說，MAR 的輸出不接到 W 匯流排，所以不需要用三態輸出。

2 對 1 多工器

晶片 C5 是編號 74LS157 的 2 對 1 的 4 位元多工器。左半位元組 (腳 14 , 11 , 5, 2) 來自位址開關暫存器 (S_1)。右半位元組 (腳 13 , 10 , 6 , 3) 則來自 MAR。RUN 及 PROG 開關 (S_2) 用來選擇接到 C5 的半位元組。當 S_2 位在 PROG 位置，就選上了位址開關暫存器輸出的半位元組。相反的，當 S_2 位在 RUN 位置，就選擇 MAR 的輸出。

16×8 RAM

晶片 C6 及 C7 是編號爲 74189 的 16×4 靜態 RAM。合在一起我們可以得到一 16×8 的讀寫記憶體 (read-write memory)。S_3 是資料開關暫存器 (8 位元)，而 S_4 是讀 — 寫開關 (read-write switch)(一按鈕開關)。如果要將程式寫進記憶體時，首先須將 S_2 放在 PROG 位置。這使得 $\overline{CE}$ 輸入 (腳 2) 爲低電壓。然後再扳動位址和資料開關，設定出正確的位址和資料字。短暫的按下讀 — 寫開關使得 $\overline{WE}$ 爲低電壓，將資料載進記憶體內。

等程式和資料進入記憶體後，須將 RUN - PROG 開關 (S_2) 放在 RUN 位置來準備讓計算機運轉。

指令暫存器

晶片 C8 和 C9 是編號爲 74LS173 的 4 位元三態緩衝暫存器。將 C8 的腳 1 及 2 接地能將三態的輸出更改成雙態的輸出 $I_7 I_6 I_5 I_4$。這半位元組接到控制器 — 定序器的指令解碼器。信號 $\overline{E}_I$ 控制 C9 的輸出，即指令暫存器的低半位元組。當 $\overline{E}_I$ 爲低電壓時，這半位元組即被置於 W 匯流排上。

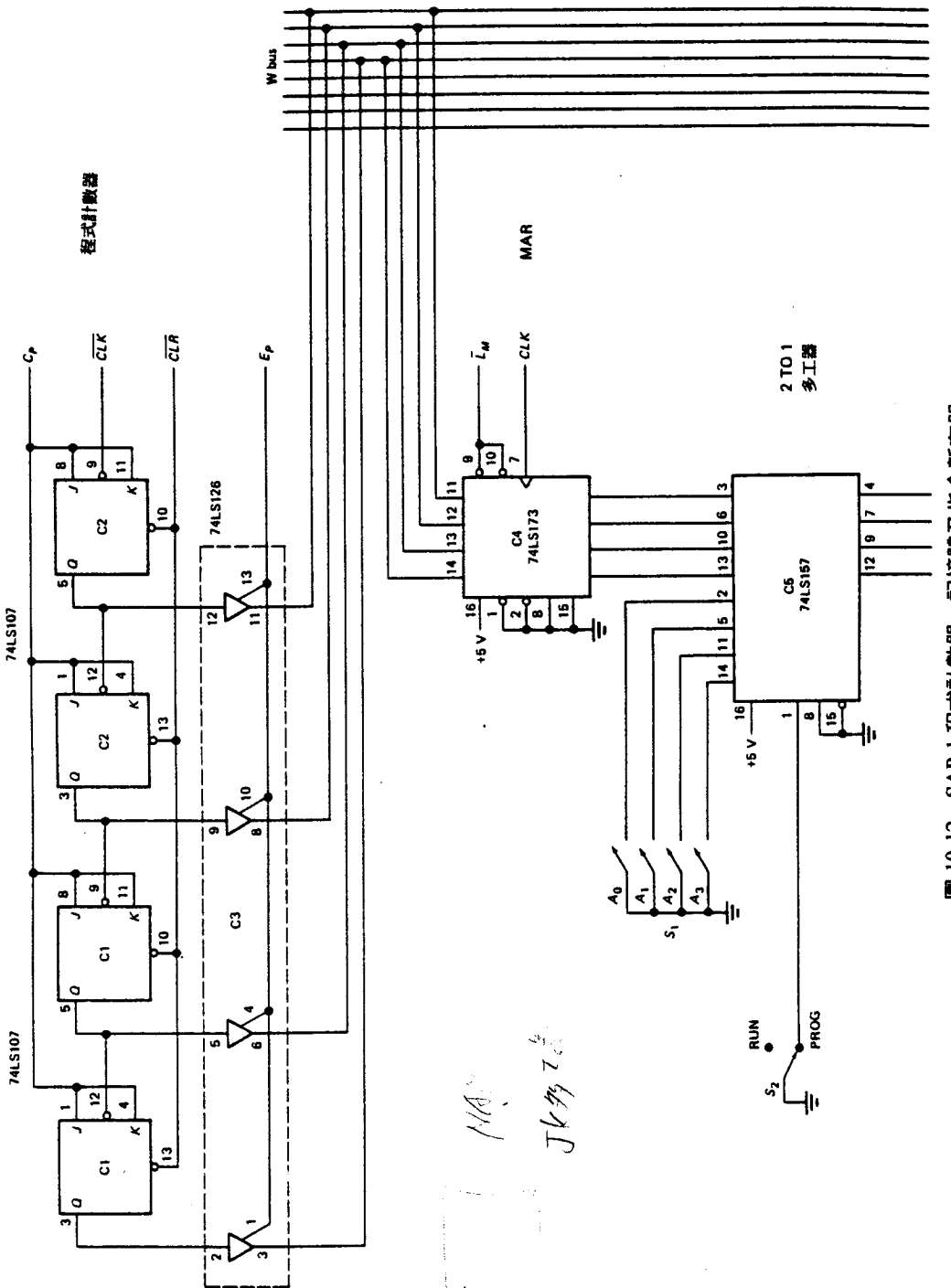


圖 10-12 SAP-1 程式計數器，記憶體及指令暫存器

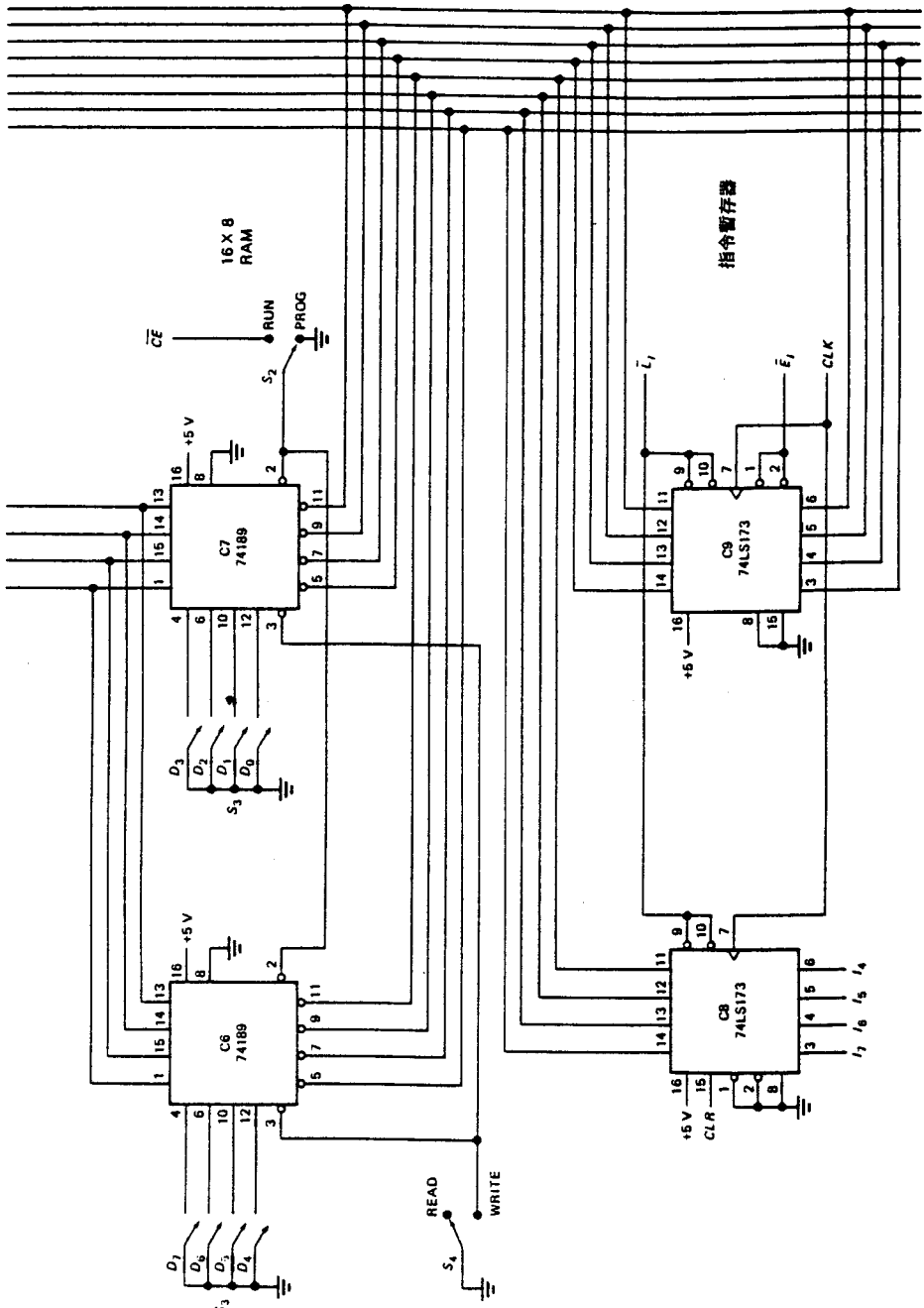


圖 10-12 SAP-1 程式計數器、記憶體及指令暫存器（續）

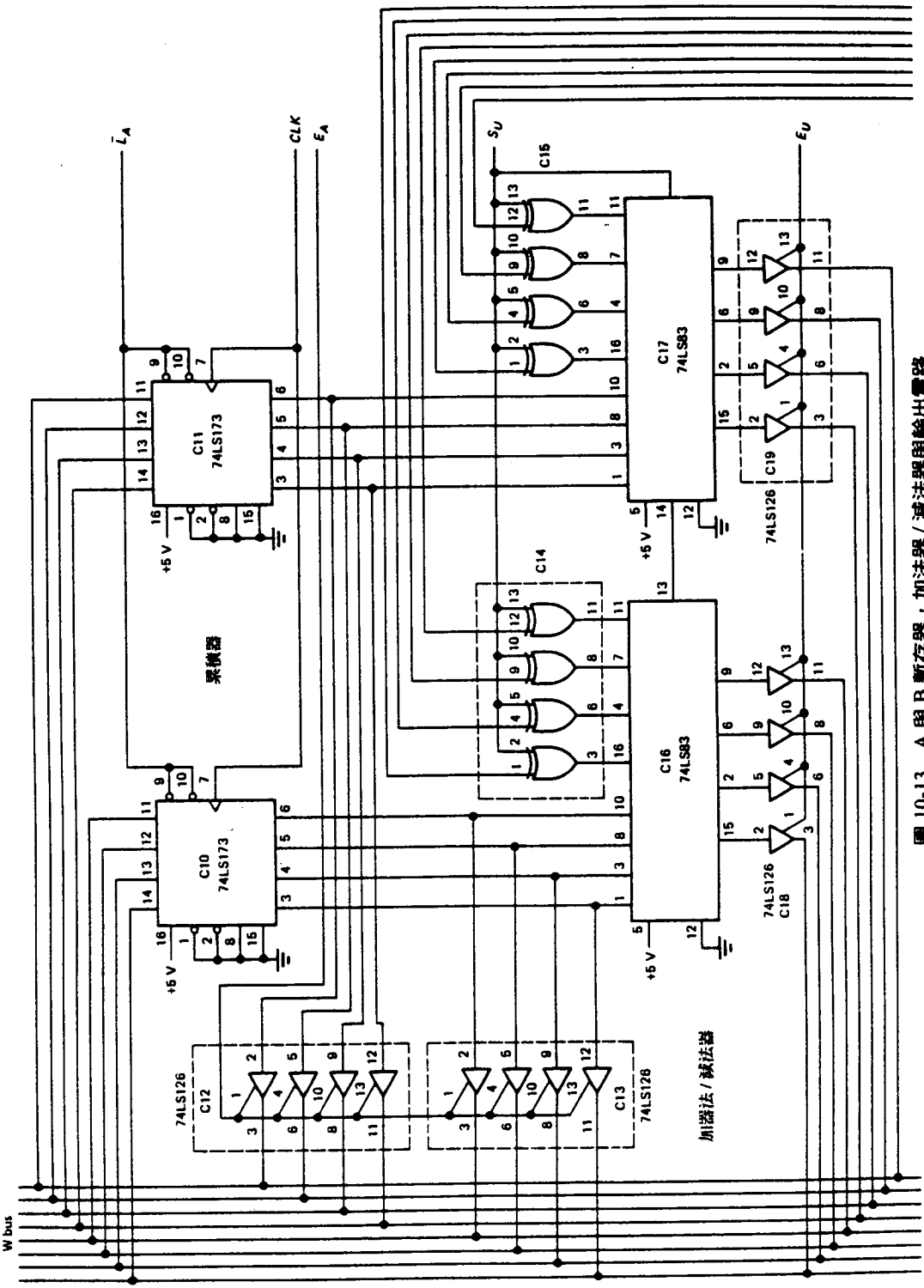


圖 10-13 A 與 B 暫存器・加法器/減法器與輸出電路

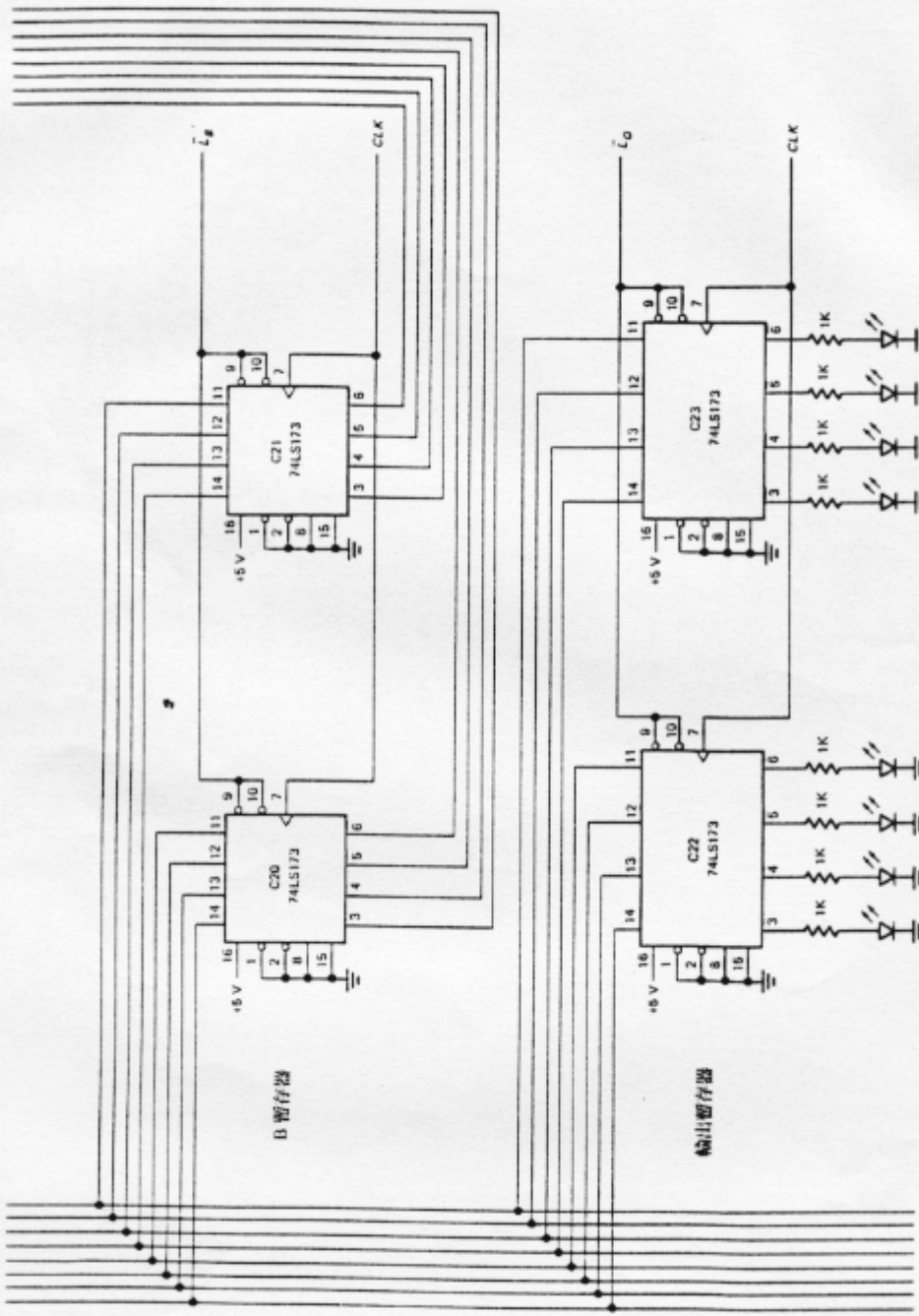


圖 10-13　A 與 B 暫存器，加法器／減法器與輸出電路（續）

累積器

晶片 C10 和 C11 是編號爲 74LS173 的累積器 (accumulator)(參閱圖10-13)。腳 1 和腳 2 都接地以產生一雙態輸出到加減法器。晶片 C12 和 C13 是 74LS126，這三態開關當 E_A 爲高電壓的時候將累積器的內容放進 W 匯流排。

加減法器

晶片 C14 和 C15 是編號爲 74LS86 的互斥或閘，能用來作爲可控制的反相器。當 S_U 爲低電壓，B 暫存器的內容直接輸出。當 S_U 爲高電壓時會將 B 暫存器內容以 1 的補數送出，而且加 1 到 LSB 來形成 2 的補數。

晶片 C16 和 C17 是 74LS83。這些 4 位元的全加器合起來產生一 8 位元的和或差。晶片 C18 和 C19 都是 74LS126，主要用來將這 8 位元的答案轉換成三態的輸出來推動 W 匯流排。

B 暫存器及輸出暫存器

晶片 C20 和 C21 是 74LS173，形成 B 暫存器。它存有要加入或從累積器減去的資料。兩晶片的腳 1 和腳 2 都接地，因此會產生雙態的輸出送到加減法器。

晶片 C22 和 C23 是 74LS173，形成了輸出暫存器。它推動了二進位顯示幕，使我們可以看出資料處理完的情形。

清除 — 起始反跳彈器

圖 10-14 中的清除 — 起始反跳彈器 (clear-start debouncer) 會產生兩個輸出：用在指令暫存器的 *CLR* 及用在程式計數器和環式計數器的 $\overline{CLR}$ 信號。$\overline{CLR}$ 也接到 C29，即時脈 — 起始正反器 (clock-start flip-flop)。S_5 是一按鈕開關。當按下時，會接到 CLEAR 位置，產生一高態 *CLR* 和一低態 $\overline{CLR}$。當釋放 S_5 時，會返回 START 位置，產生高電壓 $\overline{CLR}$ 和低電壓的 *CLR*。

注意到一半的 C24 用在清除 — 起始反跳彈器而另一半用在單步反跳彈器 (single-step debouncer)。晶片 C24 是編號爲 7400 的四合一二輸入反及閘。

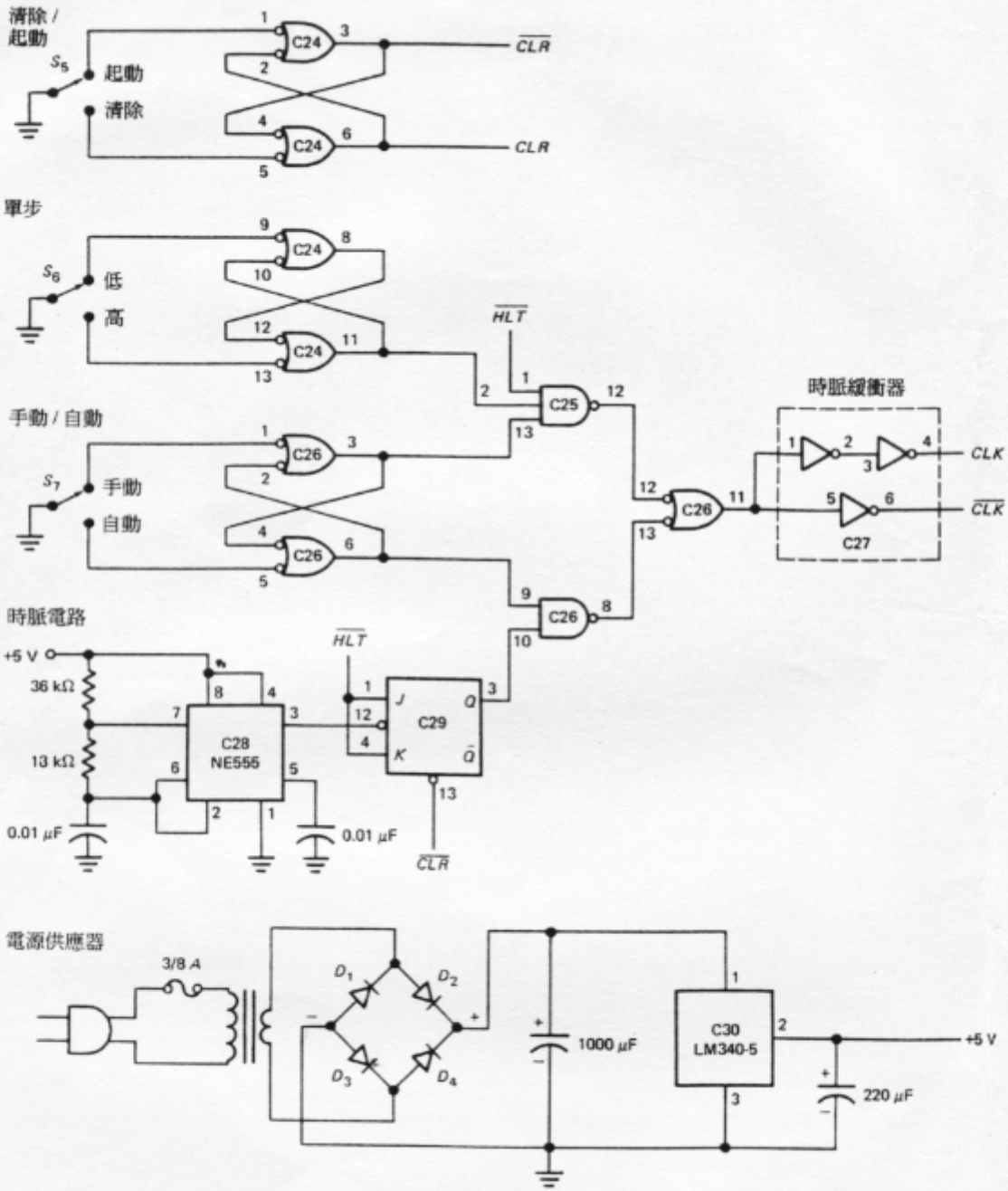


圖 10-14　電源供給器、時脈和消除電路

單步反跳彈器

SAP-1 擁有兩種運轉的模式，一是手動，一是自動。在手動模式中，按下及釋放一次 S_6 即產生一時脈信號。當按下時 *CLK* 爲高電壓，釋放時 *CLK* 爲低電壓。換句話說，圖 10-14 中的單步反跳彈器當你按放按鈕一次，就產生一個 *T* 狀態。這就允許在除錯 (debugging) 時能夠一步一步的執行指令 (除錯表示在程式中找出錯誤)。

手動 — 自動反跳彈器

開關 S_7 是一單刀雙擲 (SPDT) 開關能夠在 MANUAL (手動) 或AUTO (自動) 兩位置中的任何一個位置。當在 MANUAL 時，單步按鈕開關才有作用。當在 AUTO 位置，計算機就是自動運轉。C26 中的兩個 NAND 閘可以防止手動 — 自動開關的跳彈現象。C26 的另兩個反及閘是 NAND-NAND 電路的一部份，用來引導單步時脈或自動時脈信號到最後的 *CLK* 及 $\overline{CLK}$ 輸出端。

時脈緩衝器

C26 的腳 11 輸出用來推動時脈緩衝器。在圖 10-14 中有兩個反相器用來產生最後的 *CLK* 輸出，另一反相器則產生 $\overline{CLK}$ 。和其他晶片不同的是，C27 是一個標準 TTL 而不是一低功率蕭特基 (看附錄 5 的 SAP-1 零件表)。用標準 TTL 的原因是它能驅動 20 個低功率蕭特基 TTL 負載。如表 4-5 所示。

如果對照 74LS107 和 74LS173 的資料手冊查出它輸入電流，就可以算出低功率蕭特基 TTL 在下面時脈和清除信號上的電流負載：

CLK = 19 LS 負載

$\overline{CLK}$ = 2 LS 負載　　(*LS*：低功率蕭特基)

CLR = 1 LS 負載

$\overline{CLR}$ = 20 LS 負載

這表示 C27 (標準 TTL) 輸出的 *CLK* 和 $\overline{CLK}$ 信號適用在驅動低功率蕭特基 TTL 負載。同時 C24 (標準 TTL) 輸出的 *CLR* 和 $\overline{CLR}$ 信號也能驅動其負載。

時脈電路和電源供給器

晶片 C28 是一 555 定時器 (timer)。這 IC 可以產生具 75% 工作週期的 2 KHz 方波輸出。如前面所談，一起動時脈正反器 (C29) 能將信號除爲具有 50% 工作週期的 1 kHz 方波信號。

電源供給器 (power supply) 包含有一全波橋式整流器和一濾波器電容器。跨越 1000 μF 電容的直流電壓差不多是 20 V。晶片 C30 是編號爲 LM340T-5 的電壓調整器，能夠產生穩定的 + 5 V 電源。

指令解碼器

晶片 C31 是六合一的反相器，用來產生 OP 碼 $I_7 I_6 I_5 I_4$ 的補數(看圖 10-15)。晶片 C32 , C33 和 C34 將 OP 碼解碼爲 5 個輸出信號：LDA , ADD , SUB , OUT 和 $\overline{HLT}$。記住一次解碼，其中只能有一信號產生作用 ($\overline{HLT}$ 是低電壓時有作用；而其他信號在高電壓時才有作用)。

當指令暫存器內是 *HLT* 指令，其位元 $I_7 I_6 I_5 I_4$ 是 1111，那麼 $\overline{HLT}$ 是低電壓。這信號返回到 C25 (單步時脈) 和 C29 (自動時脈)，不論在自動或手動模式中，時脈都會停止，使得計算機停止運轉。

環式計數器

環式計數器有時又稱作狀態計數器 (state counter)，包含有三個晶片，C36, C37 和 C38。這些晶片都是編號 74LS107 的 *JK* 主僕式正反器。當清除 — 起始按鈕 (S_5) 按下後，會將此計數器重設。爲了使 Q_0 正反器的輸出反 (C38 的腳 6) 推動 Q_1 正反器的 *J* 輸入 (C38 的腳 1)，Q_0 正反器被顛倒來使用。所以，T_1 的輸出開始時是高電壓。

CLI' 信號推動一爲低電壓動作的輸入端。這表示 *CLK* 信號的負緣是在每一 *T* 狀態的開始，半週期以後，暫存器的載入發生在 *CLK* 信號的正緣。

控制矩陣

指令解碼器將指令解碼後，就從 *LDA* , *ADD* , *SUB* 及 *OUT* 端輸出信號給由 C39 到 C48 所組成的控制矩陣 (control matrix)。同時，環式計數器的 T_1 到 T_6 信號也推動這控制矩陣。

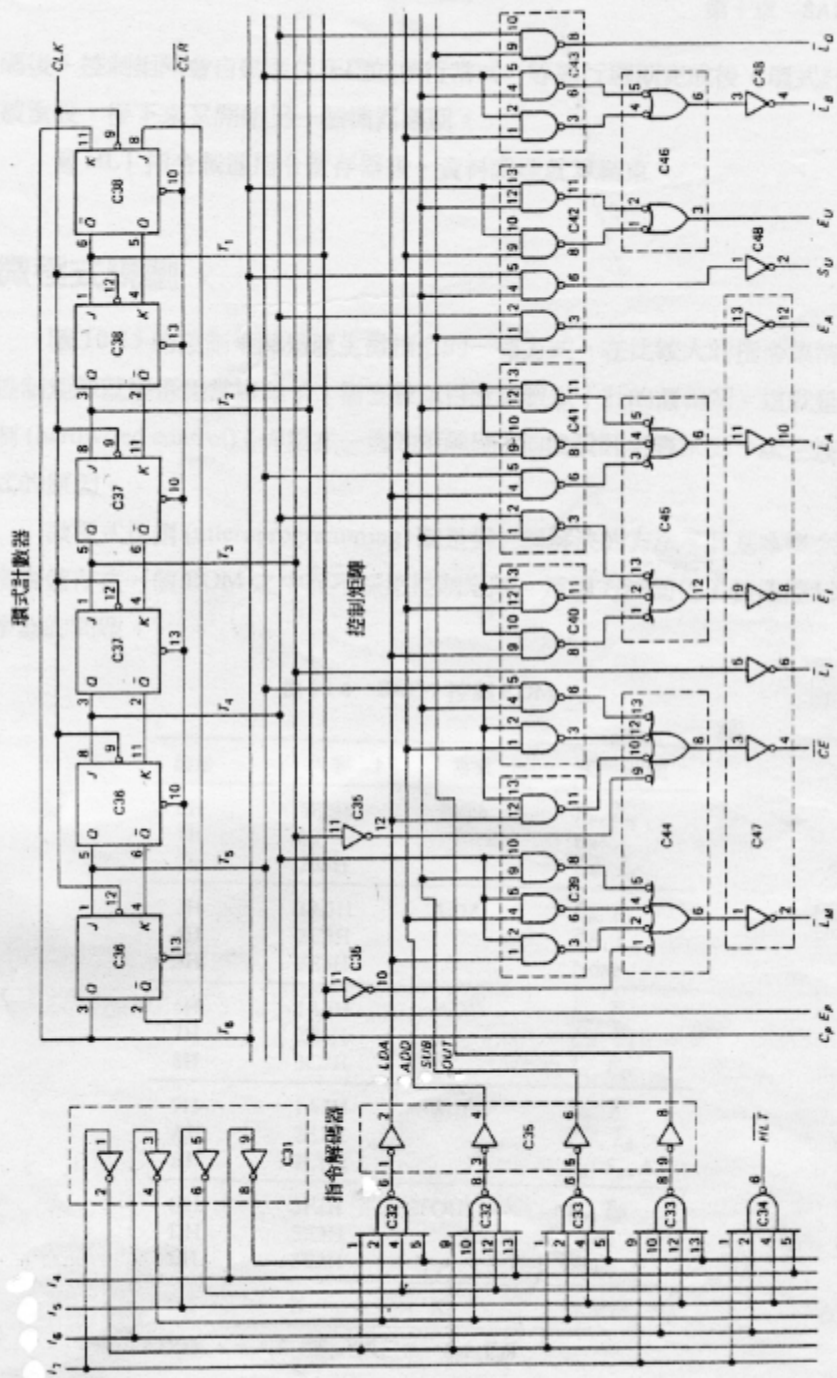


圖 10-15 指令解碼器、環式計數器、和控制矩陣

這矩陣產生了一個具有 12 位元的 CON 字，據此指揮計算機其他部份的動作。

在圖 10-15 中，T_1 先變爲高電壓，接下來是 T_2，再下來時 T_3，以此類推。分析了控制矩陣就可以發現，一高電壓的 T_1 會導致一高態的 E_p 及一低態的 $\overline{L}_M$（位址狀態）；高態的 T_2 會產生一高態的 C_p（遞增狀態）；一高態 T_3 產生一低態的 $\overline{CE}$ 與一低態的 $\overline{L}_I$（記憶狀態）。前面三個 T 狀態是 SAP-1 中的提取週期。提取週期中的 CON 字表列如下：

狀態	CON	有作用的控制數元
$T1$	5E3H	Ep，$\overline{L}_M$
$T2$	BE3H	Cp
$T3$	263H	$\overline{CE}$，$\overline{L}_I$

在執行狀態中，T_4 到 T_6 依序成爲高電壓。而在這解碼輸出的信號 (LDA 到 OUT) 中只有一信號爲高電壓。因此，此矩陣自動引導有作用位元到正確的輸出控制線上。

舉個例子來說，當 LDA 爲高電壓，只第一，四，七，十個二輸入反及閘被致能 (enable)。當 T_4 爲高電壓，就啓動了第一及第七個反及閘，產生低態的 $\overline{L}_M$ 及 $\overline{E}_I$ 低態的（將位址欄載進 MAR)。當 T_5 爲高電壓，就啓動第四及第十個反及閘，而產生低態 $\overline{CE}$ 及 $\overline{L}_A$（將 RAM 資料載進累積器）。當 T_6 變成高電壓後，所有的控制位元都沒有作用 (nop)。

如果我們繼續將剩餘的指令一一加以分析的話，將會發現這控制矩陣所產生 ADD , SUB 與 OUT 指令的微指令和表 10-5（SAP-1 微程式）所列的相符。

操　作

在每次計算機運轉前，操作員須分別將程式及資料輸進到 SAP-1 記憶體的低、高位址部份。程式和資料輸進完成後，操作員就可按下清除按鈕。當釋放按鈕時 CLK 與 $\overline{CLK}$ 信號就開始驅動暫存器與計數器；而控制器 — 定序器輸出的微指令就在每一正 CLK 邊緣處決定做些什麼事。

每一 SAP-1 機器週期都是以提取週期爲開頭。T_1 爲位址狀態，T_2爲遞增狀態，T_3 爲記憶狀態。提取週期完成後，指令就已存入指令暫存器內。等指令欄解

碼後，控制矩陣會自動產生正確的執行常式。等執行週期完成後，環式計數器會被重設，接下來又開始另一個機器週期。

當 HLT 指令載進指令暫存器後，資料處理就算結束。

10-8 微程式規劃

圖 10-15 的控制矩陣是產生微指令的一種方式。在比較大的指令集情況下，控制矩陣就變得相當複雜了，需要數以百計或數以千計的邏輯閘。這就是硬線控制 (hardwired control) (焊接在一塊的矩陣閘) 迫使設計者尋求另一產生控制字方式的原因。

微程式規劃 (microprogramming) 就是另一種解決的方法。其基本概念是將微指令儲存在一個 ROM 之中而不採用控制矩陣。這種方法簡化了製造控制器一定序器的問題。

表 10-6 SAP-1 控制 ROM

位址	內容 †	常式	動作訊號
0H	5E3H	Fetch	E_P, $\overline{L}_M$
1H	BE3H		C_P
2H	263H		$\overline{CE}$, $\overline{L}_I$
3H	1A3H	LDA	$\overline{L}_M$, $\overline{E}_I$
4H	2C3H		$\overline{CE}$, $\overline{L}_A$
5H	3E3H		None
6H	1A3H	ADD	$\overline{L}_M$, $\overline{E}_I$
7H	2E1H		$\overline{CE}$, $\overline{L}_B$
8H	3C7H		$\overline{L}_A$, E_U
9H	1A3H	SUB	$\overline{L}_M$, $\overline{E}_I$
AH	2E1H		$\overline{CE}$, $\overline{L}_B$
BH	3CFH		$\overline{L}_A$, S_U, E_U
CH	3F2H	OUT	E_A, $\overline{L}_O$
DH	3E3H		None
EH	3E3H		None
FH	X	X	未使用

† CON = $C_PE_P\overline{L}_M\overline{CE}$ $\overline{L}_I\overline{E}_I\overline{L}_AE_A$ $S_UE_U\overline{L}_B\overline{L}_O$.

微程式之儲存

加上一提取常式，並且分配各常式的位址後，我們可以得到示於表 10-6 的 SAP-1 微指令。這些微指令能夠儲存在一控制 *ROM* 內，其中的提取常式位於位址 0H 到 2H，LDA 常式位於位址 3H 到 5H，ADD 常式位於 6H 到 8H，SUB 常式位於 9H 到 BH，OUT 常式位於 CH 到EH。

要執行任一常式，我們必需提供正確的位址。舉個例子來說，要執行 ADD 常式時，我們需要提供位址 6H , 7H 與 8H。要執行 OUT 常式，我們得提供位址 CH , DH 和 EH。所以執行任何一個常式需要三個步驟：

1. 取得常式的起始位址。
2. 經由常式位址一步一步執行下去。
3. 提供位址給控制 ROM (控制 ROM 的位址)。

位址 ROM

圖 10-16 示出了如何規劃 SAP-1 的微程式。其中包括了一位址 ROM，一可預設的計數器及一控制 ROM。位址 ROM 儲存了每一常式的起始位址。如表 10-7 所表示。LDA 常式的起始位址爲 0011，而 ADD 常式的起始位址爲 0110。

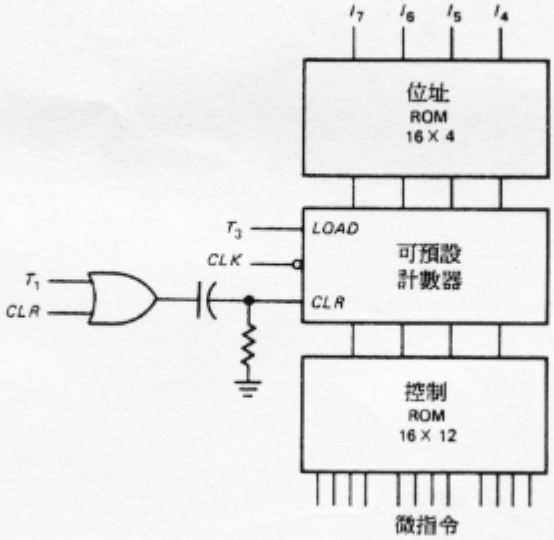


圖 10-16　SAP-1 的微程式控制

表 10-7 位址 ROM

位址	內容	常式
0000	0011	LDA
0001	0110	ADD
0010	1001	SUB
0011	XXXX	None
0100	XXXX	None
0101	XXXX	None
0110	XXXX	None
0111	XXXX	None
1000	XXXX	None
1001	XXXX	None
1010	XXXX	None
1011	XXXX	None
1100	XXXX	None
1101	XXXX	None
1110	1100	OUT
1111	XXXX	None

當 OP 碼位元 $I_7 I_6 I_5 I_4$ 推動位址 ROM 時，對應在這指令的起始位址就產生了。舉個例子來說，如果正在執行 ADD 指令，那麼 $I_7 I_6 I_5 I_4$ 爲 0001。這就是位址 ROM 的輸入，而其輸出爲 0110。

可預設的計數器

當 T_3 爲高電壓，那麼可以預設計數器的 *LOAD* 輸入爲高電壓，因此位址 ROM 的起始位址就被載進計數器。在其他狀態時，計數器則執行計數的動作。

開始的時候，從清除一起始反跳彈器來的 *CLR* 信號經微分而得一正的狹窄波尖，將計數器重設 (reset)。當計算機開始運轉後，在 T_1時計數器輸出爲 0000，在 T_2 狀態時爲 0001，而在 T_3 狀態時爲0010。每一提取週期均和這個相同，因爲在狀態 T_1 , T_2 和 T_3 間，計數器輸出都是 0000 , 0001 和 0010。

指令暫存器中的 OP 碼控制了執行週期。如果提取到 ADD 指令，那麼 $I_7 I_6 I_5 I_4$ 位元爲 0001。這個 OP 碼位元推動位址 ROM，產生的輸出是 0110 (表 10-7)。這起始位址被送到可預設計數器的輸入。當 T_3 爲高電壓時，下一負時脈邊緣就將 0110 載入可預設計數器。這時候計數器已被預設，而計數器由此 ADD 常式的起始位址開始計數。在 T_4 狀態計數器輸出爲 0110，在 T_5 狀態時輸出爲 0111，在 T_6 狀態時輸出爲 1000。

當 T_1 狀態開始的時候，T_1 信號的前緣被微分而產生一正波尖，將計數器重設爲 0000，也就是提取常式的起始位址。於是重新開始另一個機器週期。

控制 ROM

控制 ROM 儲存了 SAP-1 的微指令。在提取週期中，它接收到位址0000, 0001 及 0010。於是其輸出爲

5E3H
BE3H
263H

這些微指令列在表 10-6，分別產生了位址狀態，遞增狀態和記憶狀態。

假如執行 ADD 指令，那麼控制 ROM 在執行週期中接收到位址 0110 , 0111 和 1000。其輸出爲

1A3H
2E1H
3C7H

這些微指令前面已經討論過，它們能夠執行加法。

再舉另外一個例子，假設執行 OUT 指令，這時 OP 碼爲 1110，再起始位址爲 1100 (表 10-7)。在執行週期之中，計數器輸出爲 1100 , 1101 和 1110，這時控制 ROM 的輸出爲 3F2H , 3E3H 和 3E3H (表 10-6)。這常式將累積器內容送到輸出埠。

可變的機器週期

表 10-6 中的微指令 3E3H 是一 nop。它在 LDA 常式中出現一次，在OUT 常式中出現了兩次。SAP-1 利用這些 nop，使每一個指令都具有相等長度的機器週期。換句話說，不論是何指令，其機器週期一定佔有六個 T 狀態。在某些計算機中，具有固定長度的機器週期是一優點。但是考慮到執行速度時，nop 則顯得非常笨拙，必需加以修改。

加快 SAP-1 動作速度的一種方法就是遇到 nop 時就跳開這 T 狀態。藉著重新設計圖 10-16 的線路可以省略 nop 狀態。利用這種方法可將 LDA 指令的機器週期縮減爲 5 個狀態 (T_1 , T_2 , T_3 , T_4 和 T_5)。同樣也能將 OUT 指令的機器週期減短爲四

個 T 狀態 (T_1, T_2, $T3$ 和 T_4)。

圖 10-17 示出了一種得到可變機器週期 (variable machine cycle) 的方法。在 LDA 指令中，T_1 到 T_5 狀態的動作同前。當 T_6 狀態開始時，控制 ROM 會產生 3E3H (nop 微指令) 的輸出。一些反及閘會立即查驗出，而且產生一低態輸出信號 $\overline{NOP}$。$\overline{NOP}$ 信號經由一及閘回饋到環式計數器。如圖 10-18 所表示。這信號將環式計數器重設到 T_1 狀態，而令一新的機器週期開始。如此就可以使 LDA 的機器週期由 6 個狀態減短到 5 個狀態。

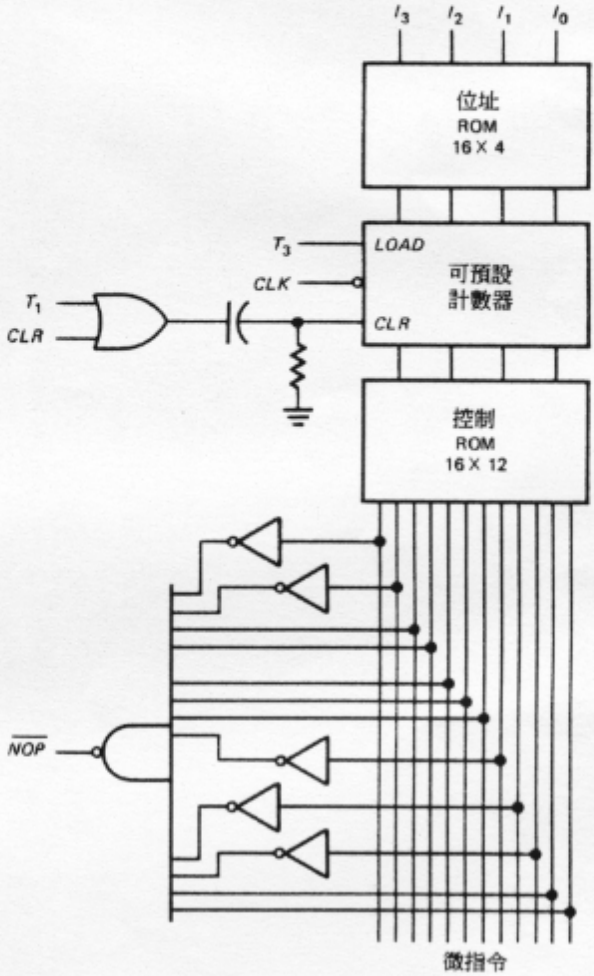


圖 10-17 可變的機器週期

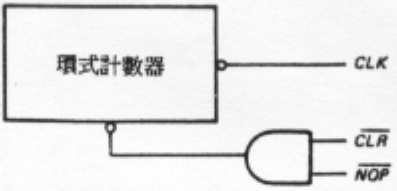


圖 10-18

在 OUT 指令時，第一個 nop 出現在 T_5 狀態。在這情形下，T_5狀態一開始，控制 ROM 就產生一 3E3H 輸出，同樣的反及閘也偵測到這輸出。所以產生了低電壓 $\overline{NOP}$ 信號將環式計數器重設到 T_1 狀態。就這樣，我們可以將 OUT 指令的機器週期由 6 個狀態減少成 4 個。

可變的機器週期常用在微處理機中，舉個例子來說，在 8085 中，機器週期擁有的 T 狀態數可以從 2 個到 6 個間變化，因爲所有不需要的 nop 狀態都省略了。

優　點

微程式的優點之一是省略了指令解碼器及控制矩陣；在比較大的指令集情況下，指令解碼器與控制矩陣就變得相當複雜了。也就是說將微指令儲存在 ROM 裡較製作一個指令解碼器和控制矩陣要來得簡單多了。

除了這個以外，一旦裝好了一指令解碼器和控制矩陣後，如果要更改指令這需要一種方法，就是將線路拆掉然後重裝。在微程式控制中就不需要這麼麻煩，所要做的只是改變控制 ROM，和位址 ROM 的內容。如果你要改善過時裝備的性能，這就是一項非常好的優點。

摘　要

總而言之，目前所製造出來的計算機，絕大部份都採用微程式控制而不是硬線控制。其微程式表及電路都比較 SAP-1 複雜多了，但是觀念仍是一樣的 — 微指令存在於控制 ROM，藉著提供所需要的微指令的位址來取得微指令。

名詞解釋

位址狀態 (address state)　即 T_1 狀態。在這種狀態間，程式計數器的位址得送到 MAR。

累積器 (accumulator)　存放算術及邏輯運算的答案之處稱爲累積器。有時稱爲 A 暫存器。

組合語言 (assembly language)　採用助憶符號來撰寫程式的一種語言。

B 暫存器 (B register)　一種輔助暫存器，可用來儲存欲加入或減去累積器內容的資料。

提取週期 (fetch cycle)　指令週期的第一部份。在此提取週期內，位址送到記憶體，程式計數器加 1，指令由記憶體送到指令暫存器。

遞增狀態 (increment state)　亦即 T_2 狀態。在此狀態間，程式計數器加 1。

指令週期 (instruction cycle)　提取及執行指令所需的全部狀態。

指令暫存器 (instruction register)　由記憶體接收指令的暫存器。

指令集 (instruction set)　計算機認識的所有指令。

LDA　載入累積器的動作之助憶符號。

機器週期 (machine cycle)　由環式計數器產生的所有狀態。

機器語言 (machine language)　用於程式的一連串 0 或 1。

巨指令 (macroinstruction)　指令集中的一個指令。

MAR　記憶體位址暫存器。此暫存器接受要從記憶體取出資料的位址。MAR 提供此位址給記憶體。

記憶體相關指令 (memory-reference instruction)　須要將資料讀出或寫入記憶體的指令。

記憶體狀態 (memory state)　即 T_3 狀態。在此狀態間，記憶體內的指令送到指令暫存器。

微指令 (microinstruction)　由控制器 — 定序器輸出的控制字。是資料處理中最基本的動作單位。

NOP　無動作。至此狀態間，無動作發生。

輸出暫存器 (output register)　這種暫存器接受由累積器送來的處理過的資料，用來推動 SAP-1 的輸出顯示幕。也稱爲輸出埠。

目的程式 (object program)　用機器語言寫成的程式。

OP 碼　即運算碼。指令中告知計算機執行什麼動作的部份。

程式計數器 (program counter)　一以二進位計數的暫存器。它的內容指出了要提取及執行的下一指令的所在位址。

RAM　隨機接達記憶體。一較恰當的名稱爲讀寫記憶體。RAM 儲存有計算機運轉時所需的程式及資料。

原始程式 (source program)　以助憶符號寫成的程式。

自我測驗複習

閱讀下面問題，並且將空格處寫入正確的答案。每一題的答案可在下一題的括弧內找到。

1. _____ 計數器是控制單元的一部份，能從 0000 計數到 1111。它將下一指令的 _____ 送到記憶體。
2. (程式、位址) MAR 又稱 _____ 暫存器，能將由程式計數器傳來的位址閂住。稍後，MAR 將此位址提供給 _____，以執行一讀取的動作。
3. (記憶體位址、RAM) 指令暫存器是控制單元的一部份。_____ 暫存器的內容被分爲兩個半位元組。上半位元組送到 _____。
4. (指令、控制器 — 定序器) 控制器 — 定序器會產生一 12 位元的字，以控制計算機其他的部份。傳送此 12 位元的 _____ 字的電線稱爲控制 _____。
5. (控制、匯流排) _____ 是一緩衝暫存器能儲存和或差。其雙態的輸出接到加減法器。當 S_U 爲低電壓時 _____ 產生和，當 S_U 爲高電壓則產生差。輸出暫存器有時稱爲輸出 _____。
6. (累積器、加法器 — 減法器、埠) SAP-1 的 _____ 集內有 LDA , ADD , SUB , OUT 及 HLT 等指令。LDA , ADD 及 SUB 稱之爲 _____ 指令，因爲它們會用到存於記憶體中的資料。
7. (指令、記憶體相關) 8080 是第一個廣泛使用的微處理器。_____ 是具有 8080 相同指令的改良型。
8. (8085) LDA , ADD , SUB , OUT 及 HLT 是由 4 位元 0 與 1 組成的碼。這樣

的碼稱爲 _____ 碼。_____ 語言是使用助憶符號來撰寫程式。_____ 語言是用一連串的 0 與 1 來寫程式。

9. (OP、組合、機器) SAP-1 有 _____ 個 T 狀態，在每一狀態間暫存器內容會改變。環式計數器又稱爲 _____ 計數器，會產生這些 T 狀態。這六個 T 狀態代表一個機器週期。SAP-1 中指令週期僅佔有一機器週期。在 8080 及 8085 的微處理器中，_____ 週期所佔的機器週期可由一到五個。

10. (六、狀態、指令) 控制器一定序器在一個 T 狀態中送出一個控制字。每一控制字稱爲一個 _____。而像 LDA，ADD，SUB 等的指令稱爲 _____。每一個 SAP-1 巨指令是由三個 _____ 組成。

11. (微指令、巨指令、微指令) 在較大指令集的情況下，控制矩陣變得相當複雜。這就是硬線控制被 _____ 取代的原因。其基本觀念就是將 _____ 儲入控制 ROM。

12. (微程式、微指令) SAP-1 的每一指令均具有相等長度的機器週期。也就是說，每一機器週期正好佔有六個 T 狀態。8085 微處理器具有可變的機器週期，因爲所有不需要的 nop 均被取消了。

習 題

10-1 用助憶符號寫一 SAP-1 的程式 (像例 10-1)，這種程式能顯示出下式的運算結果：

$$5+4-6$$

使用位址 DH, EH 和 FH 來儲存資料。

10-2 將問題 10-1 的組合語言轉換成 SAP-1 的機器語言。用二進位及十六進位來表示此答案。

10-3 寫一能執行下式的組合語言程式：

$$8+4-3+5-2$$

10-4 將習題 10-3 的程式及資料轉換爲機器語言，將這個結果用二進位及十六進位來表示。

10-5 圖 10-19 示出了 ADD 指令的時序圖。試畫出 SUB 指令的時序圖。

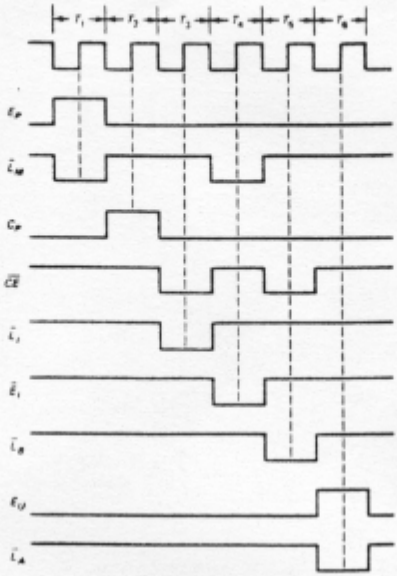


圖 10-19

10-6 假定 8085 的時脈頻率爲 3 MHz。並且執行 8085 的 ADD 指令共需花費 4 個 T 狀態。請求出所需的執行時間。

10-7 請利用二進位與十六進位的形式，分別列出 SAP-1 的 LDA 與 SUB 常式所需的微指令。

10-8 假如我們想將累積器的內容轉移給 B 暫存器，那麼則需另外增加一個新的微指令。請利用十六進位與二進位的形式，列出此微指令。

10-9 參閱圖 10-20，回答下面的問題：

a. 在 $\overline{CLK}$ 信號的正或負邊緣出現時，程式計數器的內容會改變嗎？如果是的話，那麼在這時候 CLK 信號究係在上昇邊緣還是下降邊緣？

b. 程式計數器加 1 時，C_p 信號須爲高態或低態？

c. 欲將程式計數器消除時，$\overline{CLR}$ 信號須爲高態或低態？

d. 欲將程式計數器的內容置於 W 匯流排時，E_p 信號須爲高態或低態？

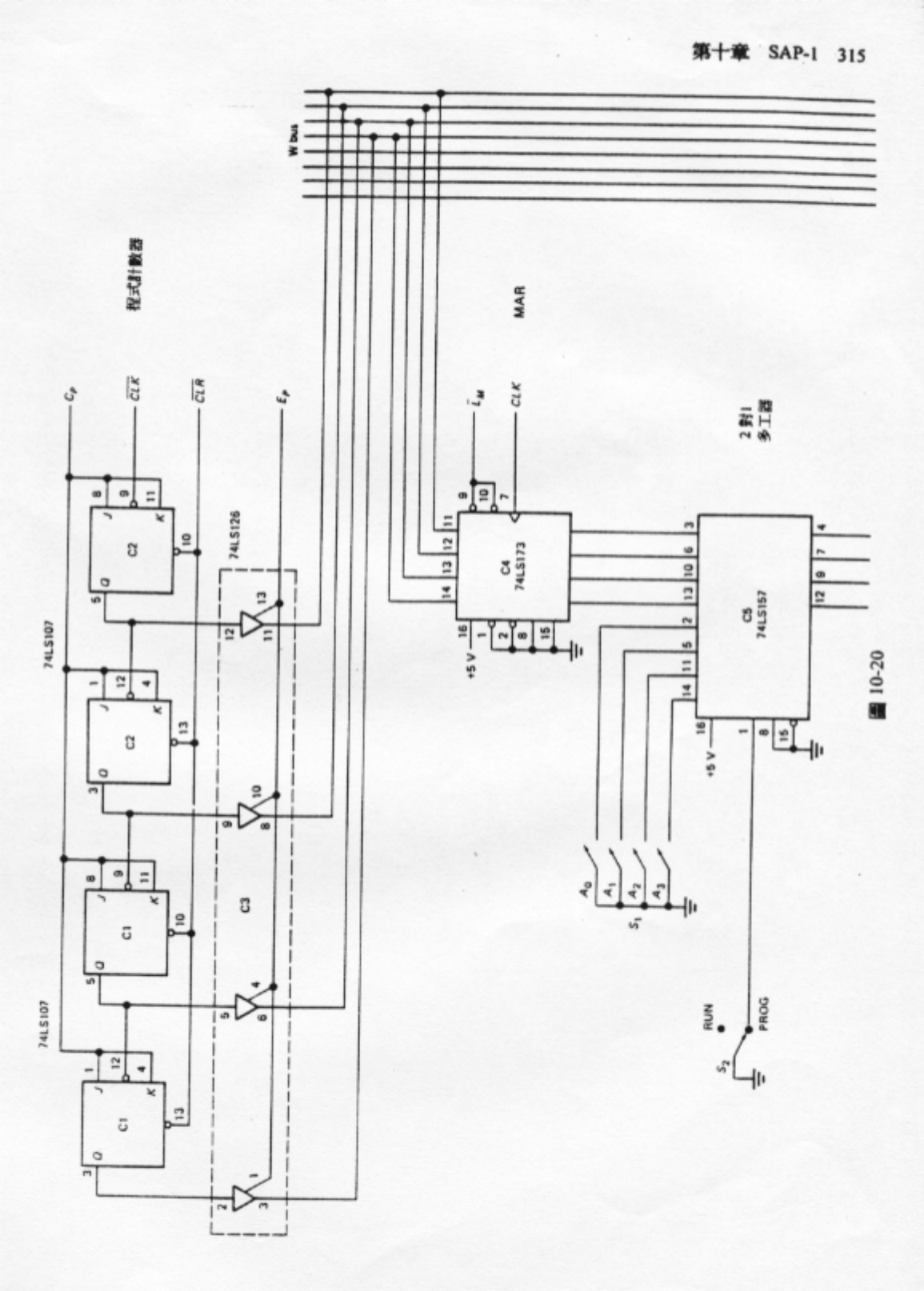

W bus
程式計數器
MAR
2對1
多工器
74LS107
74LS126
C1
C2
C3
C4
74LS173
C5
74LS157
+5 V
RUN
PROG

圖 10-20

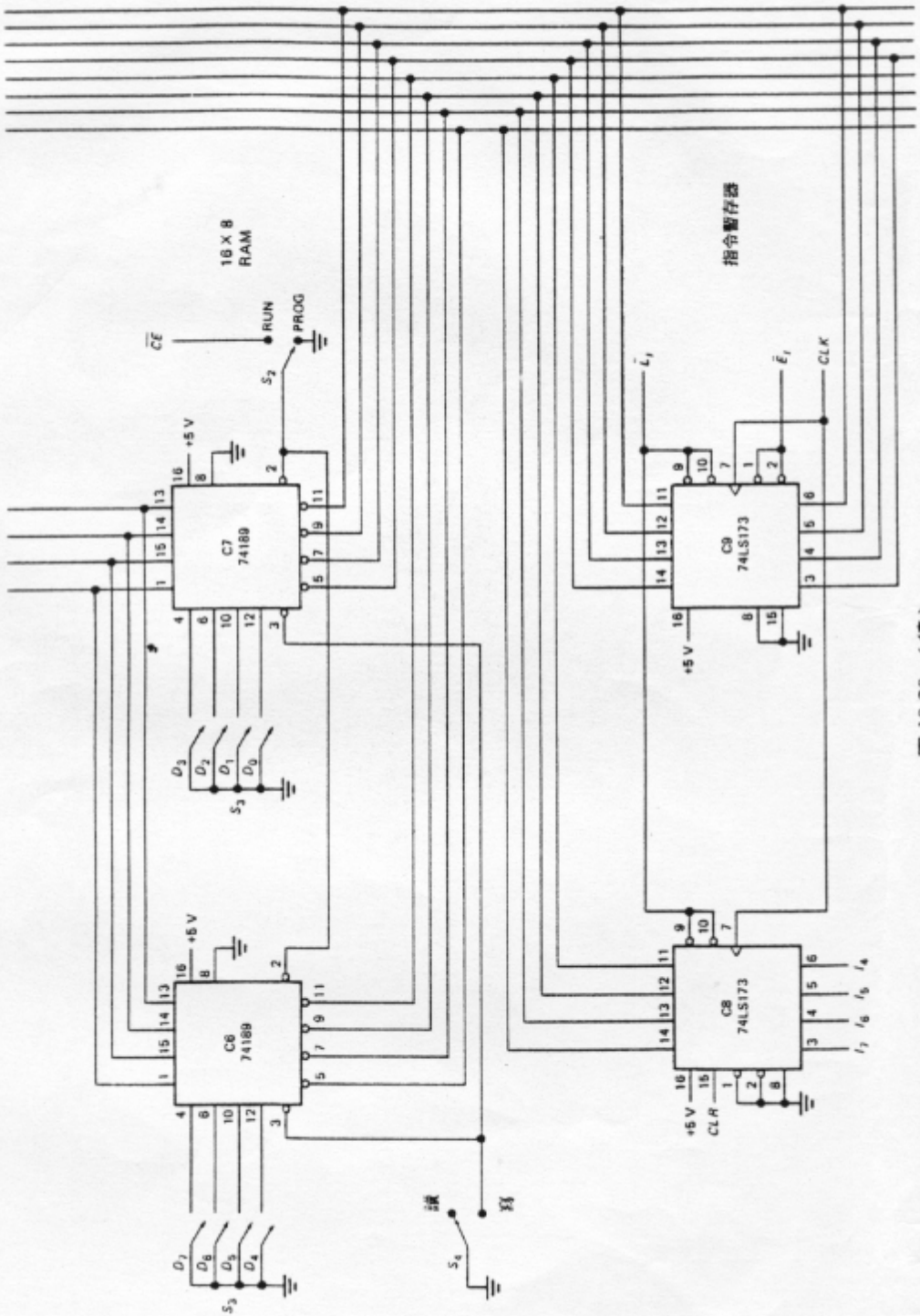

16 × 8
RAM
指令暫存器
C7
74189
C8
74189
C9
74LS173
C8
74LS173
+5 V
RUN
PROG
CLR
CLK
讀
寫

圖 10-20（續）

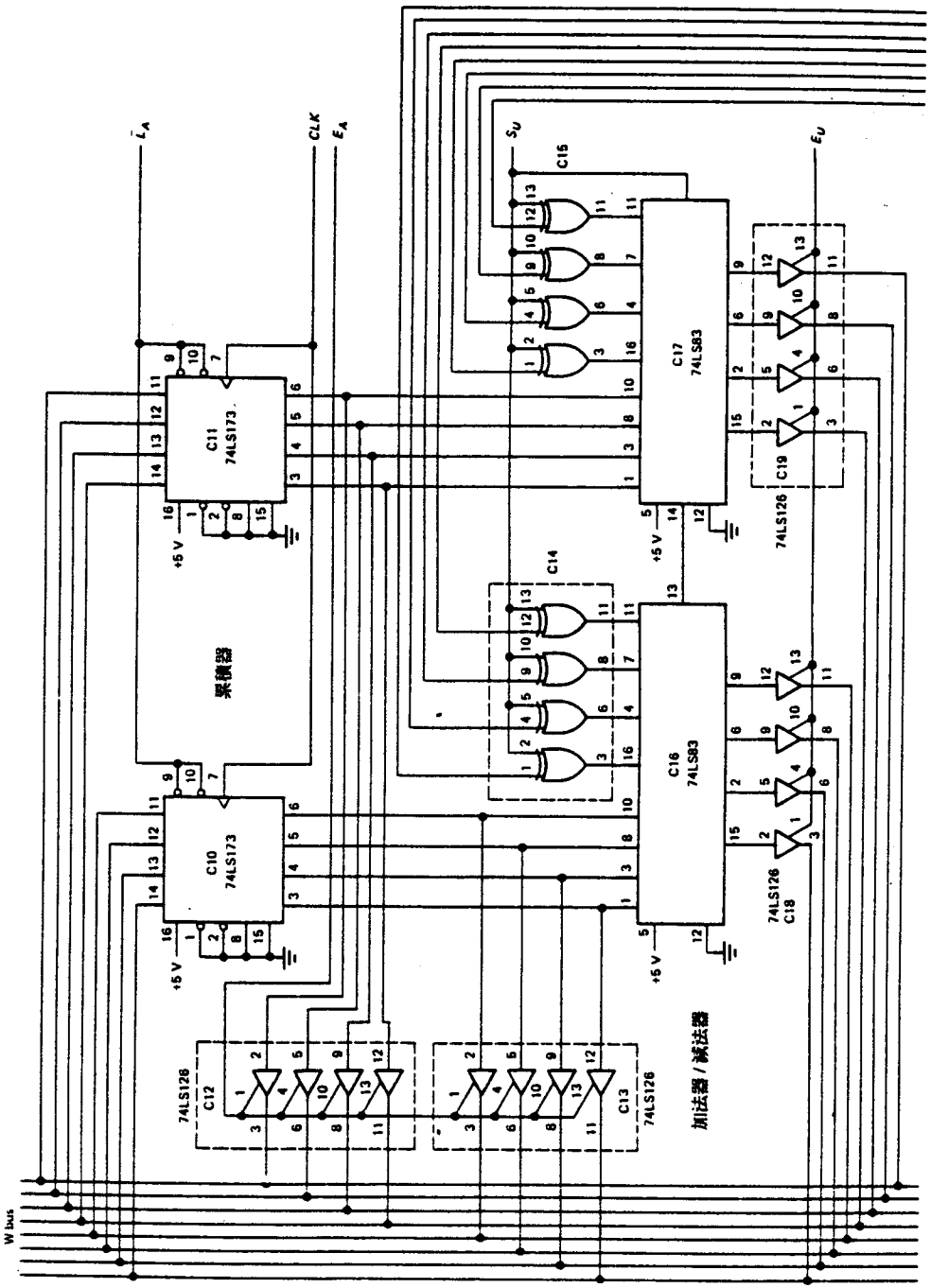

$\overline{L_A}$
CLK
E_A
S_U
E_U
C15
C17
74LS83
C19
74LS126
C14
C16
74LS83
C18
74LS126
C11
74LS173
C10
74LS173
累積器
+5 V
C12
74LS126
C13
74LS126
加法器/減法器
W bus

圖 10-21

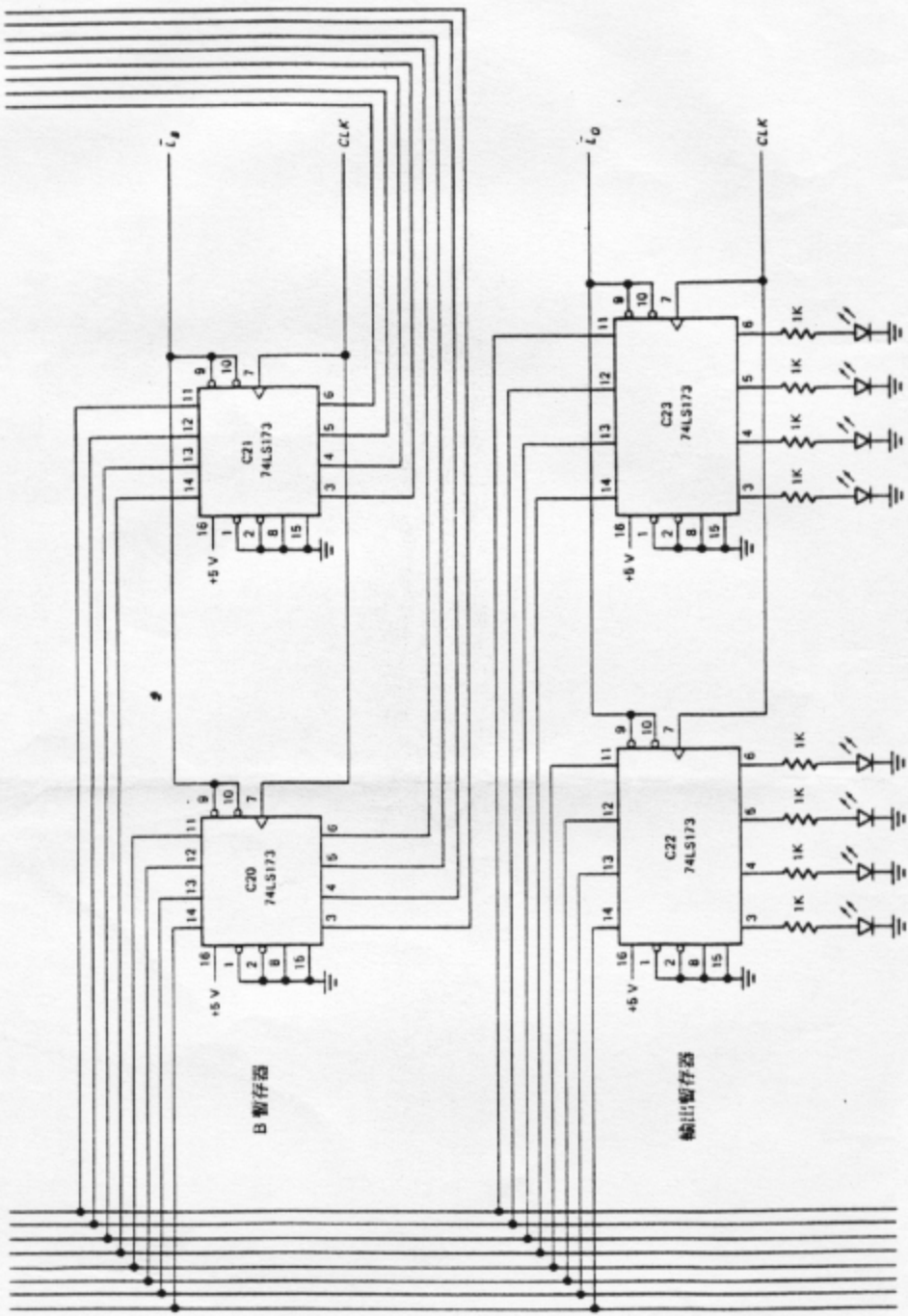

$\bar{L}_B$
CLK
C21
74LS173
C20
74LS173
+5 V
B 暫存器
$\bar{L}_O$
CLK
C23
74LS173
C22
74LS173
1K
+5 V
輸出暫存器

圖 10-21 （續）

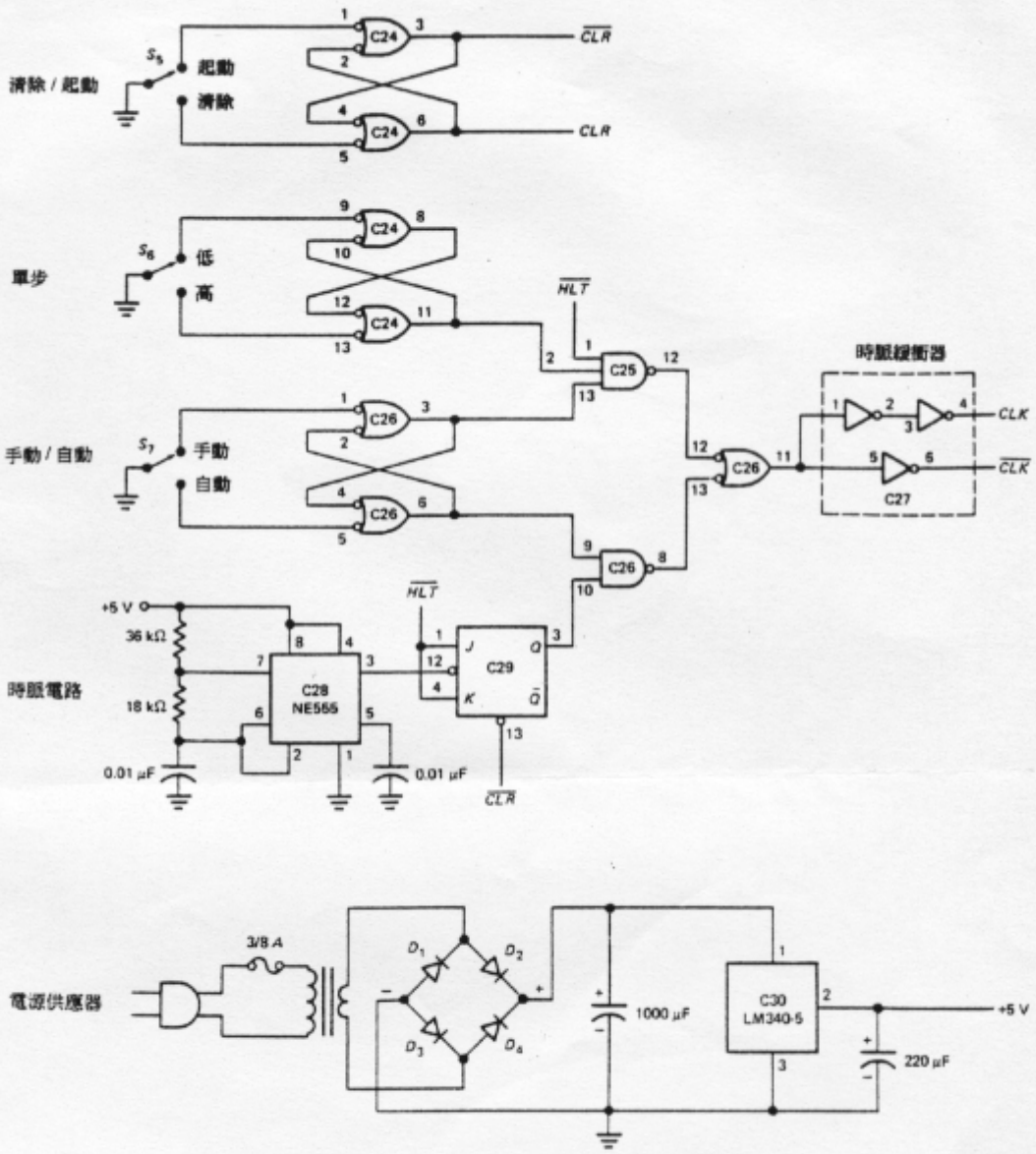

清除 / 起動
S_5
起動
清除
C24
$\overline{CLR}$
CLR
單步
S_6
低
高
$\overline{HLT}$
C25
手動 / 自動
S_7
手動
自動
C26
時脈緩衝器
CLK
$\overline{CLK}$
C27
時脈電路
+5 V
36 kΩ
18 kΩ
0.01 μF
C28
NE555
C29
J
K
Q
$\overline{Q}$
電源供應器
3/8 A
D_1
D_2
D_3
D_4
1000 μF
C30
LM340-5
220 μF
+5 V

圖 10-22

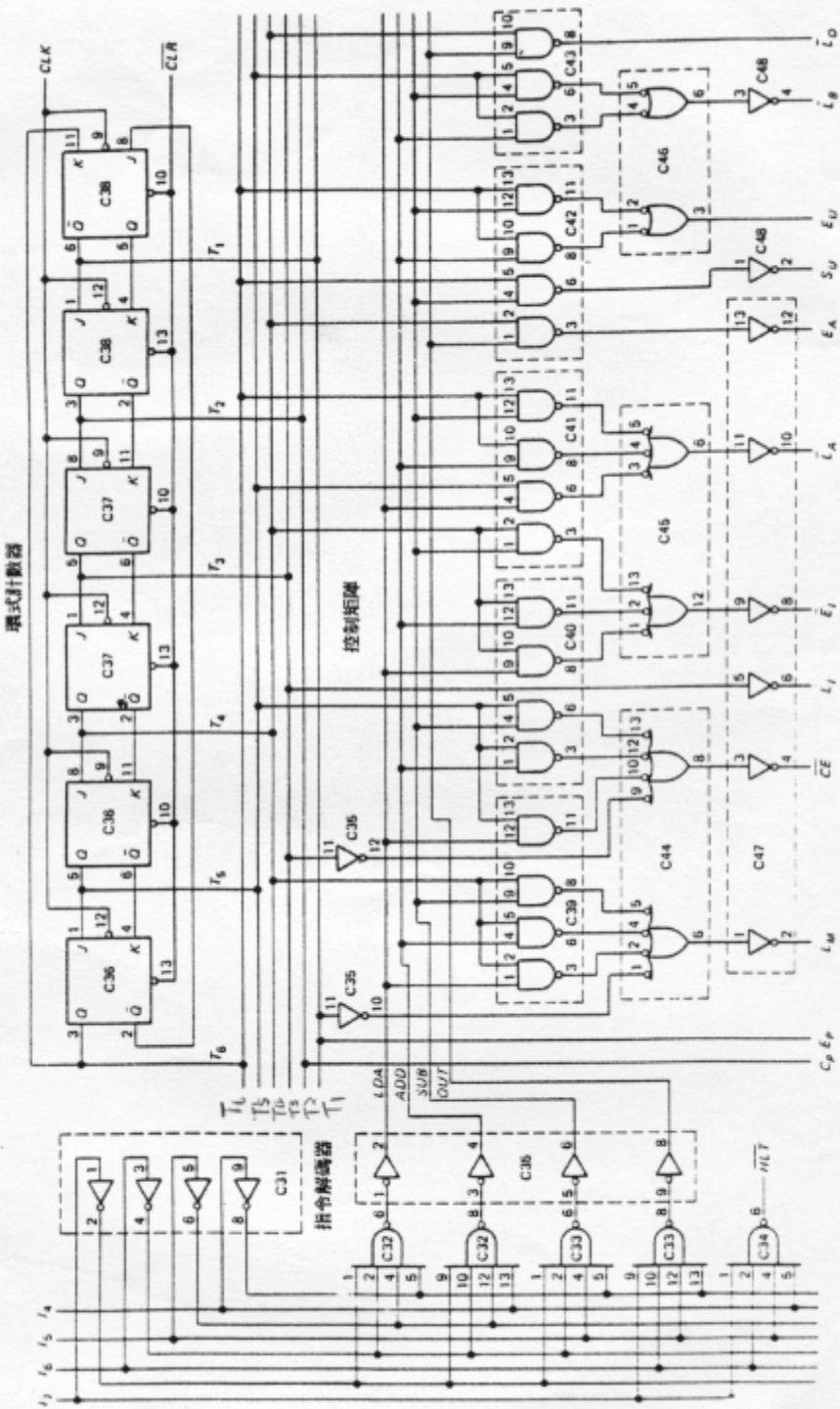

環式計數器
控制矩陣
指令解碼器
CLK
CLR
LDA
ADD
SUB
OUT
HLT
C31
C32
C33
C34
C35
C36
C37
C38
C39
C40
C41
C42
C43
C44
C45
C46
C47
C48

圖 10-23

10-10 參閱圖 10-21：

a. 如果 $\overline{L}_A = 1$，那麼下一個正向時脈邊緣來臨時，累積器的內容會發生什麼樣的變化？

b. 假定 A = 0010 1100 , B = 1100 1110，當 $E_A = 1$ 時 W 匯流排的內容爲何？

c. 假定 A = 0000 1111 , B = 0000 0001，當 $S_U = 1$，當 $E_U = 1$ 時，W 匯流排的內容爲何？

10-11 參閱圖 10-22 回答下面的問題：

a. 當 S_5 在 CLEAR 位置時，$\overline{CLR}$ 爲高態或低態？

b. 當 S_6 在 LOW 位置時，C24 的第 11 腳爲高態或低態？

c. 若欲於 C29 的接腳 3 輸出時脈信號，$\overline{HLT}$ 須爲高或低態？

10-12 參閱圖 10-23 回答下面的問題：

a. $I_7 I_6 I_5 I_4 = 1110$ 時，C35 的那一接腳爲高態輸出？

b. $\overline{CLR} = 0$ 時那一個計時信號 (T_1 到 T_6) 爲高態？

c. *LDA* 與 T_5 都是高態時，C45 的第 6 腳爲高態或低態？

d. ADD 與 T_4 都是高態時，C45 的第 12 腳爲高態或低態？