技術講座

電 子 計 算 機

이 連載講座는 지난 4月 24~26日에 있었던 電子, 通信技術세미나에서 日本의 林一郎 博士가 講演한 것을 研究調査部에서 간추린 것이다.

研究調査部

第1部 情報處理技術

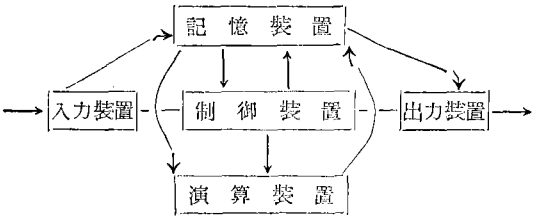
第1章 電子計算機의 原理

1. 電子計算機와 普通機械와의 差異

〔普 通 機 械〕: 原料→生産工程→製品
(固定目的)

〔電子計算機〕: 入力(Data)→
電子計算機→出力(結果)(可變目的 即多目的)
↑
프로그램
(處理順序)

2. 電子計算機의 基本構造



處理過程

1). 프로그램을 記憶裝置에 記憶시킨다.
2). 프로그램 하나 하나를 制御裝置에서 解讀한다.
3. 制御裝置는 다른 各裝置를 動作시킨다.
4). 入力裝置에서 데이터를 읽어 記憶裝置에 記憶시킨다.
5) 記憶裝置와 演算裝置와의 사이에서 計算處理가 遂行된다.
6). 結果는 記憶裝置에서 出力裝置로 옮겨지고 여기서 解答이 나온다.

3. 電子計算機의 機能

電子計算機는 莫大한 量의 데이터를 記憶하고 迅速, 正確히 處理할 수 있을 뿐만 아니라 프로그램에 依하여 人間으로부터 賦與된 範圍內에서의 判斷能力도 갖는다.

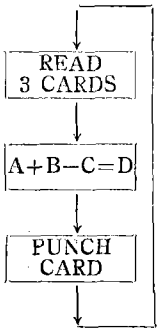
電子計算機가 갖는 機能의 特徵은 다음과 같다.

1). 處理順序(프로그램)를 機械自身이 記憶할 수있다.
2). 處理할 데이터나 結果를 記憶할 수 있다.
3). 프로그램에 의하여 自動的으로 데이터를 處理한다.
4). 處理速度가 相當히 빠르고 또 正確하다.
5). 프로그램에 의하여 주어진 範圍內에서의 判斷機能을 갖는.
6). 다프로그램을 變更하므로써 多樣性있는 處理能力을 갖는다.

4. 프로그램이란 어떠한 것인가?

프로그램의 原理를 理解하기 쉽게 하기위하여 다음과 같은 實際의 例題를 들고 說明하기로 하자

(가) 機械語로 된 프로그램의 例



命令의 場所	操 作	데이터	다음命令
0500	70	0101	0501
0501	70	0102	0502
0502	70	0103	0503
0503	01	0101	0504
0504	11	0102	0505
0505	12	0103	0506
0506	20	0110	0507
0507	71	0110	0500

說 明

① 이 프로그램에서는 70이라는 操作記號는 카아드로부터 읽으라는 命令이며 3枚의 카아드에 적혀있는 數字들을 各各 0101, 0102, 0103의 아드레스番號에 記憶시키라는 操作을 指示한다.

② 操作記號(Operation Code) 01은 0101番地에서 數字를 꺼내어 演算裝置內에 이미 들어 있는 것을 지우고 여기에 A의 값으로서 넣도록 한다.

③ 操作記號 11은 앞에 나온 값에 加하라는 指示로서 앞에 나온 A의 값에 0102番地의 數字를 B의 값으로하여 加한다.

④ 記號 12는 앞에 나온 값으로부터 減하라는 指示로서 앞의 A+B의 값으로부터 0103番地에 있는

값을 減한다.
⑤ 記號 20은 記憶裝置에 記憶시키라고 하는 指示로서 A+B−C=D의 값을 0110番地에 넣는다.
⑥ 記號 71은 穿孔하라는 指示로써, 0110 番地에 들어 있는 D의 값을 꺼내어 카아드에 펀치한다.

위와 같은 演算을 記號語로 쓴 프로그램의 例를 보이면 다음과 같다.

(나) 記號語로 된 프로그램

命令의 場所	操 作	데 이 터
BEGIN	C·READ	ALPHA
	C·READ	BETA
	C·READ	GAMMA
	CLA	ALPHA
	ADD	BETA
	SUB	GAMMA
	STORE	DELTA
	PUNCH	DELTA
	JUMP	BEGIN

說 明
① 첫번째 카아드를 읽어서 (C·READ), Alpha番地에 넣어라
② 2번째 카아드를 읽어서, Beta番地에 넣어라
③ 3번째 카아드를 읽어서, Gamma番地에 넣어라
④ 演算裝置에 加何한 값이 들어 있더라도 지우고 거기에 Alpha番地안에 들어 있는 값을 넣어라
⑤ 앞의 것에 Beta番地에 있는 값을 加하라 (Add Beta)
⑥ 그 結果로부터 Gamma番地안에 들어 있는 값을 減하라 (Sub Gamma)
⑦ 그 結果를 Delta番地에 넣어라 (Store Delta)
⑧ Delta番地에 있는 값을 펀치하라(Punch Delta)

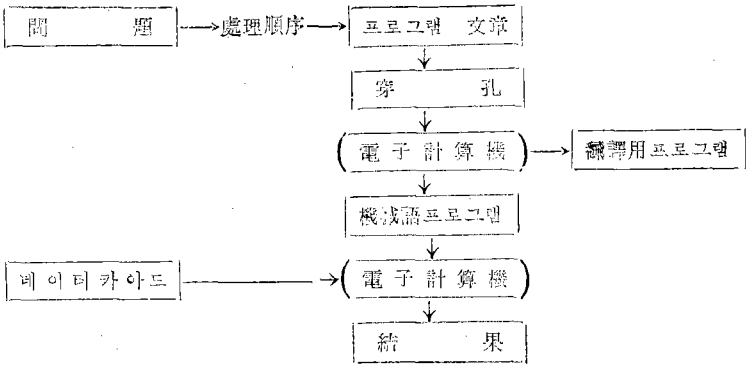
다음엔 같은 內容을 文章으로 쓴 프로그램을 쓰면 다음과 같다.

(다) 文章으로 된 프로그램

BEGIN	READ 3 CARDS (A·B·C)
	D=A+B−C
	PUNCH CARD(D)
	IF (CARD) NEXT, BEGIN
NEXT	STOP
	END

說 明
① 3枚의 카아드를 읽어서 A·B·C番地들에 넣어라
② A+B−C를 計算하여 그 結果를 D로 하라
③ 위의 結果를 펀치하라
④ 만일(IF) 카아드가 있으면 다음(NEXT) 演算은 Begin으로 가라.
⑤ 카아드가 없으면 Next로가서 스톱(Stop)하라

5. 데이터處理의 順序



說 明

問題를 풀기위한 處理順序가 프로그램 文章으로 되어 穿孔된후 電子計算機에 넣어진다. 即 프로그램은 計算機가 읽을 수 있는 機械語로 바꾸기 위해서 飜譯用프로그램에 의하여 機械語프로그램으로 바꾸지 않으면 안된다. 이 械機語프로그램과 데이터가 計算機의 記憶裝置에 들어간다. 이렇게 하므로써 프로그램의 處理順序에 따라서 데이터가 處理된다.

6. 프로그래밍言語

프로그래밍言語에는 다음과 같은 것들이 있다

ALGOL, FORTRAN (以上科學技術計算用言語)
COBOL (事務計算用言語)
PL/1 (여러가지 言語의 長點만을 取한 汎用言語)

7. 2進法

10進法의 數字는 0부터 9까지로 되어 있으나 2進法의 數字는 0과 1만으로 구성된다. 例컨대 10進法의 6은 2進法으로 表示하면 0110으로된다 即 $1\times2^3+1\times2^1+0=6$ 또 2進法 101111은 10進法 數字로는 47이다.

即 $1\times2^5+0\times2^4+1\times2^3+1\times2^2+1\times2^1+1=47$

2進法과 10進法의 對應表

2 進 法	10進法	2 進 法	10進法	2 進 法	10進法	2 進 法	10 進 法
0	0	10000	16	100000	32	110000	48
1	1	10001	17	100001	33	110001	49
10	2	10010	18	100010	34	110010	50
11	3	10011	19	100011	35	110011	51
100	4	10100	20	100100	36	110100	52
101	5	10101	21	100101	37	110101	53
110	6	10110	22	100110	38	110110	54
111	7	10111	23	100111	39	110111	55
1000	8	11000	24	101000	40	111000	56
1001	9	11001	25	101001	41	111001	57
1010	10	11010	26	101010	42	111010	58
1011	11	11011	27	101011	43	111011	59
1100	12	11100	28	101100	44	111100	60
1101	13	11101	29	101101	45	111101	61
1110	14	11110	30	101110	46	111110	62
1111	15	11111	31	101111	47	111111	63

8. Bit란 무엇인가?

2進法의 數字 1個를 "bit" (binary digit)라고 한다 Bit는 0과 1로서 表示되며 이것은 情報의 最小單位로써 使用된다. 電子計算機의 記憶裝置는 數萬 Bit로 構成된다. 이 Bit는 靜止된 狀態로는 電位나 磁化의 方向으로 0이나 1을 表示하여 움직이는 狀態로는 電氣的·펄스의 有無로서 나타낸다. 電子計算機가 理解할 수 있는 言語(機械語)들은 모두 이 같은 Bit로 만들어 진다.

9. 8進法

數値나 情報(데이터)를 Bit로 表示함에 있어 2進法의 한 例를 들면

11010100111001010

와 같은 모양이 된다. 이 같이 連續的으로 表示하면 複雜하므로 3Bit식 묶어서

110, 101, 000, 111, 001, 010

로 表示하고 各各 獨立的으로 3Bit식 읽으면 650712로 되어 便利하다. 이것은 3Bit식 묶여서 8進法의 數字 1個를 表示하는 方法이다.

8進法 1桁(3Bit)은 $2^3=8$ 即 8가지의 變化를 나타낼 수 있으며, 8進法 2桁(6Bit)로는 $2^6=64$ 가지의 變化를 나타낼 수 있고, 8進法 3桁(12Bit)는 $2^{12}=4096$ 가지의 變化를 나타낼 수 있다.

10. 命令語

記憶裝置는 많은 數의 記憶場所로 區分되어 있으며 그 하나하나에 番地(Adress)가 붙여져 있다. 그리고 하나의 番地에는 電子計算機에 따라서 다르나 대개 16~36 Bit의 情報를 記憶할 수 있다.

21Bit로 구성된 例를 보면 다음과 같다.

000 000 000 000	000	000 000
番地部	修正部	演算部

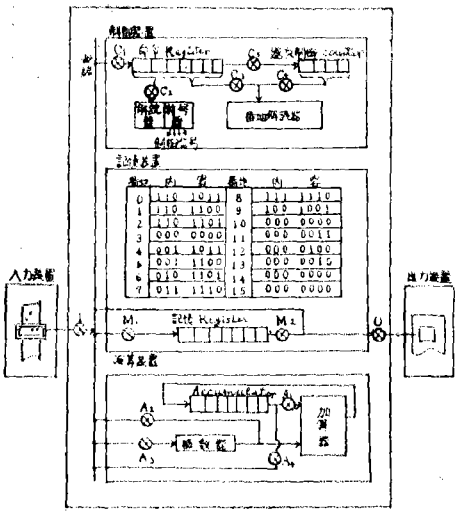
演算部는 演算을 指定하는 것으로써 「加算," "減算", "乘算", "除算", "記憶"等의 操作을 指定한다.

番地部는 演算될 對象의 所在를 가르친다. 修正部는 番地部에 있는 番地를 그대로 使用하지 않고 修正한 番地를 使用할 때 쓴다.

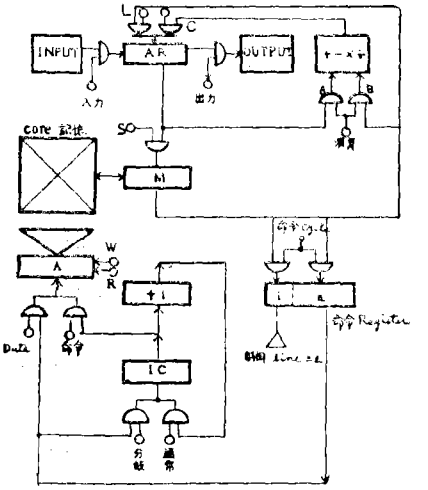
11. 電子計算機의 構成

(가) 電子計算機의 構成

(나) 電子計算機의 系統圖



가. 電子計算機의 構成



나. 電子計算機의 系統圖

Input: 카아드 리이더, 종이테이프 리이더등
OUtput:라인프린타, 타이프라이타등
AR: Arithmetic Register
M: Memory Register
A: Address Register
IC: Instruction Counter
端子L: Load
〃 C: Calculate
〃 S: Store
〃 W: Write
〃 R: Read

12. 論理回路

(가) AND 回路(論理積)

論	理	積
A · B = C		
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

(나) OR回路(論理和)

論	理	和
A + B = C		
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

(다) NOT回路

(라) Flip-Flop回路

13. 演算裝置

演算裝置는 制御裝置로부터의 指令에 따라서 動作한다. 가장 基本的인 演算裝置는 加算器와 Register이다.

가). 半加算器　　나). 全加算器

다). Register

1個의 Flip-Flop은 1個의 Bit를 記憶할 수 있다. 따라서 4個의 Flip-Flop은 4Bit의 情報를 記憶할 수 있다. Register는 Flip-Flop을 여러개 連結한 것으로써 데이터를 一時記憶시키기 위하여 使用된다.

14. 記憶裝置

入力裝置로 읽은 情報를 記憶하고 2情報中에서 指令이 될 프로그램을 制御裝置에 供給하고 必要한 데이터를 演算裝置에 傳達하며 그 演算結果를 다음 計算을 爲해서 또는 出力裝置로 보내기 위해서 一時的으로 貯藏하는 裝置로써 이에는 다음과 같은 것들이 있다.

(1) 磁氣코아
(2) 磁氣드럼
(3) 磁氣디스크
(4) 磁氣테이프
(5) 磁氣薄膜
(6) 磁氣카아드

15. 制御裝置

制御裝置는 處理順序에따라 命令을 차례차례로 꺼내어 이를 解讀하고 電子計算機內部에 必要한 信號를 보내어 그 命令을 實行시킨다.

制御裝置는 普通 命令 Register, 逐次制御카운터, 解讀器, 符號器, 番地 解讀器등으로 구성된다.

16. 入力裝置

(1) 穿孔카아드　리이더
(2) 穿孔테이프　리이더
(3) 磁氣테이프　리이더
(4) 磁氣잉크文字 리이더

17. 出力裝置

(1) 穿孔카아드　펀치
(2) 穿孔테이프　펀치
(3) 磁氣테이프　裝置
(4) 磁氣잉크印刷機
(5) Line Printer
(6) 圖表表示器
(7) 音聲應答裝置
(8) TV應用裝置등