連載講座 1

電子計算機의 프로그래밍과 그 應用

宋 吉 永*
(Kil-yeong Song)

第1部 電子計算機概說

1. 緒 言

韓國電力에서는 第1次電源開發計劃의 成功的인 達成에 이어 第2次電源開發計劃(1967~1971年)에서는 이 計劃期間內의 總需要電力(GWH) 및 尖頭需要의 平均增加率을 世界에서도 그 類例를 볼 수 없을 정도로 높은 29.6% 및 27.2%로 想定하고 現在 이에 對한 意欲的인 電源開發事業을 推進하고 있다.

(註) 長期電力需要想定(1967~1976) 1967. 5 韓國電力株式會社

이와같은 情勢에 對應하여 앞으로의 電力系統도 超高壓送電線의 新設, 送變電設備의 建設等으로서 더욱더 膨大될 것이 豫想되고 있으며, 또 이에 따른 系統運用도 經濟性의 追求와 質的 向上이라는 兩面을 結付시킨 合理的인 運用을 指向하여 果敢하게 展開되어야 할 것이다.

이러한 觀點에서 우리가 하루빨리 解決하지 않으면 안될 問題가 山積되고 있을 것이나, 이 中에서 于先 重要한 것만을 추려 보면

(1) 經濟的, 合理的인 長期電源開發問題
(2) 合理的인 送電線 및 變電所의 建設問題
(3) 直接接地問題 및 通信線誘導障害問題
(4) 保護繼電方式 整備에 따른 安定度問題
(5) 異常電壓對策 및 絕緣協調에 關한 問題
(6) 系統電壓, 電力 및 周波數의 合理的制御方式의 問題
(7) 系統의 經濟運用問題

等을 들 수 있다.

上記한 諸問題를 다루어 나가기 爲하여서는 이제까지의 經驗, 筆算만 가지고는 그 解決이 到底히 不可能할 것이며 最近 널리 開發利用되고 있는 電子計算機, 其中 digital 計算機의 利用없이는 充分한 電力系統의 合理的 運用을 期할 수 없는 實情에 있다.

이러한 意味에서 最近 締結된 送變電施設 第1次 AID 借款協定에도 뚜렷하게 上記 digital 計算機의 導入이 包含되고 있어 우리들의 關心을 크게 끌고 있다. 따라서 本文에서는 不遠 韓國電力에도 이와같은 電子計算機가 導入될 것이라는 展望 아래에서 電力系統問題에 있어서의 電子計算機의 利用狀況을 몇차례에 걸쳐 連載紹介하고저 하는 바이다.

먼저 第1部에서는 電子計算機의 槪要를 其中 digital 計算機의 Soft ware를 中心으로 說明하고, 第2部에 實際로 計算機를 움직여 나가는데 必要한 Programming問題를 Machine Language와 Fortran을 中心으로 解說하고 마지막에 第3部에 있어서 몇가지 電力系統問題에 關한 digital 計算機의 適用例를 具體的인 計算手法과 計算例를 들어 說明하겠다.

2. 各種電子計算機의 特徵과 그 運用

먼저 電子計算機의 種類와 그 運用例를 적어 보겠다. 電子計算機는 一般的으로 Analog型과 digital型으로 大別된다. 前者에는 所謂 말하는 Analog 計算機 微分解析器, 直流計算盤 및 交流計算盤이 있으며, 넓은 意味로서는 模擬送電線裝置도 여기에 包含된다.

이 Analog型의 計算原理는 數를 直接計數하는 것이 아니고 相似的으로 測定하여 計算한다는 것이다. 이것이 各自의 用途에 따라 專門化되어 있음으로 目的하는 바 計算을 빨리, 또 簡單하게 實行할 수 있지만, 物理量으로 數를 相似케 하는 以上 그 計算程度는 이들 物理量의 測定精度에 支配되어 그 精度를 올리기가 퍽 힘들게 되고 있다.

따라서 이 Analog型 電子計算機는 그 利點으로서 特別히 精度를 必要로 하지 않는 경우에 簡便하게 또 迅速하게 計算을 行할 수 있다고 하겠으나, 目的에 따라서 專門化하지 않으면 안되기 때문에 이것이 應用面에 있어서 큰 缺點이 되고 있다.

한편 後者의 digital型은 所謂 말하는 digital 計算機로 代表되는 것인데 計算原理가 數를 直接計數함에 있고, 이것을 實로 놀라울 程度의 超高速(現在 主要機種의 演算速度는 10^{-6}秒 order이며 이미 10^{-9}秒 order의 것이 開發되고 있다)으로 行할 수 있을 뿐만 아니라 特히 그 큰 特徵이 計算機自身이 實行하고 있는 計算을 스스로가 制御할 수 있다는 것 곧 內部記憶能力을 所有하고 있다는 것이다. 또 最近의 數學的 手法의 눈부신 發達이 그 能力의 偉大性을 더욱더 倍加시키고 있다.

* 韓國電力株式會社 技術部 正會員

參考的으로 **表 1**에 digital 計算機와 Analog 計算機의 性能을 比較整理 하였다. 여기에서 볼 수 있는 바와 같이 問題에 따라서는 各自의 特點을 살려 問題에 알맞는 機種을 選擇, 利用하도록 하여야 할 것이다.

表 1. Analog 및 digital 計算機의 比較

性 能	Analog	digital
計算準備	block diagram, Analog의 結線을 함	Flow chart Programming 함
計算結果	Oscilogram 表示	數字로表示(最近에는 圖示도 可能)
精 度	誤差 1.0~0.1%	10^{-3}~10^{-6}%
速 度	一般으로 Analog 計算機로 풀 수 있는 digital 問題는 計算機보다 빠르다.	問題에 따라 다르겠으나 最近에는거의 問題가안 된다.
應用範圍	比較的 좁다 (特殊用途에 便利)	거의 制限없이 모든 問題를 풀수 있으며 計算規模도極히 크다
保守의難易	相當히 어렵다.	쉽다
價 格	比較的 싸다	比較的 비싸다

다음에 代表的인 몇가지 計算機의 利用狀況을 簡單히 적어 보겠다.

2~1 交流計算機의 利用

交流計算盤은 **그림 2~1**에 보인 바와 같이 電力系統을 直接模擬하는 것이지만, 모두가 靜止機器로 이루어져 있기 때문에 模擬送電線처럼 回轉機의 過渡的인 特性을 그대로 模擬할 수는 없다. 따라서 多少 解析上의 限界는 있으나 Analog 計算機와의 併用 또는 自動化裝置等을 부쳐서 過渡現象問題에도 相當히 應用되고 있다.

交流計算盤의 取扱하는 問題는 主로

(1) 電力潮流問題, 送電損失, 電壓分布計算
(2) 定態 및 過渡安定度(段段法 採用) 計算
(3) 故障(不平衡故障包含)時의 電壓, 電流分布計算
(4) 損失方程式의 Bmm 係數의 基礎 DATA 計算(系統임피단스計算)

等이 있으나 이中 美國等의 使用實績에 比하면 (1)의 計算이 全體의 80%를 占하고 있다. 이와같은 計算에있어서는 마치 實系統의 움직임을 監視하듯이 計測机의 測定器를 通하여 直接系統의 電壓 電流分布를 監視 把握할 수 있음으로 電力會社에서 많이 쓰이고 있다. 韓國電力에서도 지난 3~4月에 日本東京電力會社의 交流

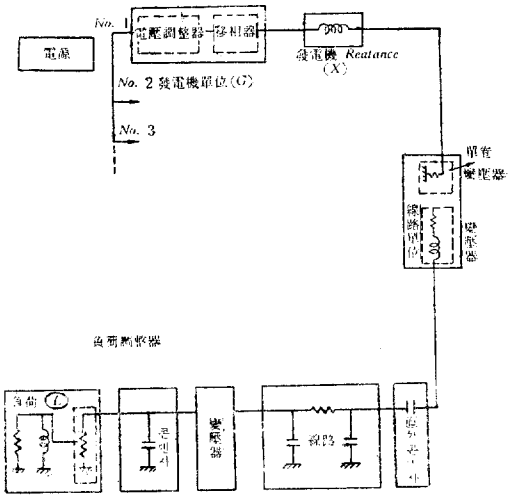
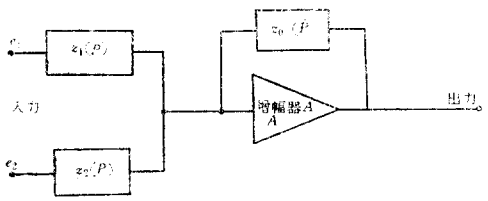


그림 2—1 交流計算盤의 構成

計算盤을 빌려 第2次電源開發計劃最終年度(1971年)의 潮流計算을 實施하였다.

2~2 Analog Computer의 利用

Analog Computer는 主로 電子式것이 널리 普及되고 있는데 그 原理는 **그림 2~3**에 보이는 바와 같이 演算增幅器가 主體로 되어 電子管의 增幅率을 아주 높으게 해서 加減算 및 微積分(但 微分回路는 雜音을 줏기쉽고, 또 計算精度가 낮음으로 普通 使用치 않음)을 行하는 것이며, 이 以外에 特殊回路로서 除算, 非線型要素의 導入을 可能케 하고 있다.



e_1, e_2, e_0 : 電壓
A : 增幅率
$p : \frac{d}{ad}$

$$e=\left(\frac{e_1}{Z_1(p)}+\frac{e_2}{Z_2(p)}\right)\Big/\left\{-\frac{1}{Z_0(p)}+\frac{1}{A}\left(\frac{1}{Z_0(p)}+\frac{1}{Z_1(p)}+\frac{1}{Z_2(p)}\right)\right\}\fallingdotseq-\left(\frac{Z_0(p)}{Z_1(p)}e_1+\frac{Z_0(p)}{Z_2(p)}e_2\right)$$

그림 2~3 Analog 計算機의 計算原理圖

電力系統運用上 Analog 計算機가 主로 많이 利用되고 있는 問題는

(1) 系統周波數, 連絡線負荷의 自動制御

(2) 同期機의 解析
(3) 自動電壓調整器를 包含한 動態安定度解析
(4) 火力發電所의 自動燃燒裝置의 解析
(5) 原子力發電所의 自動制御 問題 等이 있다.

2~3 digital 計算機의 利用

Digital 計算機는 大別하여 事務用(主로 料金調定 統計用)과 技術用으로 나누어진다.

이中 事務用 것은 取扱하는 DATA 數가 많기 때문에 一般으로는 小數點固定方式이 많고(10進法으로 使用) 또 카—드 穿孔機 分類機 印刷機等의 附屬設備를 많이 必要로 하기 때문에 技術用과 比較하여 計算速度가 若干 느리고 取扱도 複雜하게 되고 있다.

以下 技術用 digital 計算機에 對하여 살펴보겠다.

表 2~2는 電力統系에 있어서의 digital 計算機의 利用狀況을 項目別로 分類整理한 것이다.

表 2~2. 電力系統에 있어서의 digital 計算機利用狀況

項 目	內 容
電力系統	系統運用, 電壓, 周波數制御, 潮流計算, 故障計算, 損失計算, 系統諸特性, 可能電力量(電源開發) 經濟運用
水力發電	機器特性 水理計算 構造物計算 運轉管理 河川流量 其他
火力發電	機器特性 構造物計算 運轉管理
原子力發電	原子爐의 設計 및 最適化問題 〃 의 安定과 事故, 原子 物理
送變電	電氣計算, 構造計算
配電	配電設計 設備管理 電壓管理
通信	通信 一般

以上 列記된 諸問題解析을 爲하여서는 마치 交流計算盤의 回路結線과 같은 意味를 가지는 計算順序命令 곧 programming을 미리 作成하여 두지 않으면 안된다. 이 programming을 잘하느냐 못하느냐에 따라 計算速度에도 相當히 影響이 있음으로 計算機를 充分하게 驅使하기 爲하여 이 programming에 熟練해 둘 必要가 있을 것이다.

다음에 이 digital 計算機를 좀 더 具體的으로 살펴 보기로 하겠다.

3. Digital 電子計算機

一般的으로 digital 計算機의 科學技術問題에 關한 應用의 process를 分析하여 본다면

(i) 問題의 理解, 定式化(problem formulation)
(ii) 解答의 計劃, 數學的解析(Mathematical Analysis)
(여기에 數値計算法(Numerical Analysis)도 包含됨)
(iii) 計劃의 實行, programming & Operating
(iv) 얻어진 解答의 檢討

의 네가지 段階로 大別될 것이다.

이 네가지 段階——理解, 計劃, 實行, 檢討——는 單純히 科學技術上의 問題에만 限定되는 것이 아니고 널리 一般問題의 解決에 있어서도 必要되는 것이며 또 問題解決을 爲하여 適用될 手段이나 道具에 關係없는 것이라고 말할 수 있다.

따라서 電子計算機를 利用하여 問題를 풀 경우에도 例外가 아닐 것이지만, 一般으로 電子計算機가 直接으로 適用되는 것은 위에서 곧 알 수 있는 바와 같이 (iii)의 段階, 即 實行의 段階이다.

이러한 意味에서 자칫하면 電子計算機의 使用問題는 (iii)의 段階에 限定시켜 生覺하기 쉽지만, 어디까지나 (i)~(iv)의 諸段階를 綜合的으로 (有機的)으로 結付시켜 對應하지 않으면 안될 것이다.

아무리 優秀한 計算機라 하더라도 問題를 正確하게 把握하고 꼭 問題에 알맞는 計劃을 세워서, 이것을 充分히 驅使하지 못한다면 決코 完成된 計算機 System이라고 말할 수 없을 것이다.

따라서 電子計算機에서는 이것을 어떻게 利用하는가 하는 方法의 科學이며 重要한 것이며 이 때문에 專門家 또는 道具가 必要하게 되는 것이다.

다음에 이와같은 計算機의 利用을 북돋는 機能으로서 다음의 세가지를 들 수 있다.

곧 計算機 自身의 演算機能 自體를 말하는 Hardware와 이 Hardware를 效果的으로 利用하기 爲하여 要求되는 모든 programming system을 말하는 software와 그리고 마즈막에 問題를 效果있게, 多角的으로 取扱할 수 있는가 하는 適用에 關한 Application ware이다.

이들 關係는 **그림 3~1**과 같이 어디까지나 有機的으로 運用하는 System으로서 把握할 必要가 있을 것이다.

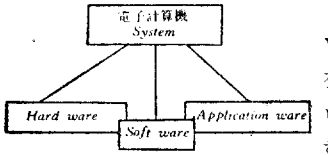


그림 3~1 digital 計算機 system의 槪念圖

다음에 이 세가지 ware를 簡單히 說明하겠다. 但 앞서 說明한 바와같이 本文이 意圖하는 바는 電力系統問題에 있어서의 digital 計算機의 利用狀況을 紹介코져 하는 것임으로 어디까지나 問題의 Analysis 및 System planner(Engineer)的인 立場(따라서 Application)에 重點을 두고 또 이를 實行하기 爲한 programming手法(soft ware)을 中心으로 살펴 보기로 하고 있다(Hard-ware에 對하여서는 다음 機會를 얻어 說明하겠다).

3—1 digital 計算機의 裝置와 構成(Hard ware)

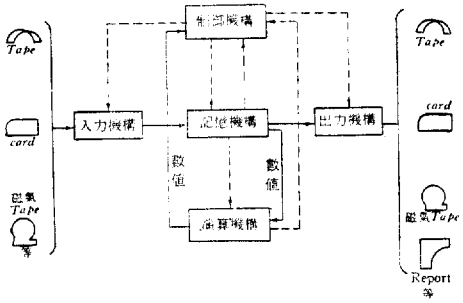
이것은 電子計算機의 骨格을 이루는 것으로 다음과 같은 다섯가지 機構로 構成되고 있다.

Hard ware

(1) 演算機構 (Arithmetic Unit)
(2) 記憶機構 (Memory Unit)
(3) 入力機構 (Input Unit)
(4) 出力機構 (Output Unit)
(5) 制御機構 (Conrol Unit)

그림 3—2는 計算機의 構成을 보인 Block diagram 이다.

以上과 같이 電子計算機의 Hard ware는 演算機構, 記憶機構, 入力機構, 出力機構 및 制御機構의 다섯개의 機構를 有機的으로 結合하여 構成되는 것이며 嚴格하게 表現한다면 오히려 電子計算組織(Electronic Computing System)이라고 불러야 될 것이다.



——DATA의 흐름, ……制御情報의 흐름

그림 3—2 電子計算機의 基本構成

3—2 digital 計算機使用의 技術 (Soft ware)

前節에서 電子計算機의 Hard ware의 概略만을 說明하였으나 이것을 살리거나 죽이는 것은 Soft ware에 달렸다고 하겠다. 一般으로 問題 또는 system이 解析되고 이것을 電子計算機를 使用하여 풀 경우에는 먼저 Flow chart의 step에 있어서의 四則演算이나 判斷等을 計算機가 實行할 수 있는 基本的인 演算으로 表現하고 이것을 그 順序에 따라 列記하지 않으면 안된다. 이와같이 列記된 것이 프로그램(program)이며 이 手續을 programming, 이것을 하는 사람을 programmer라고 부르고 있다.

電子計算機에서는 前述한 바와같이 몇개의 計算順序가 符號化되어 基本的인 演算에 들어가는 것이며 이것을 命令(Instruction)이라고 하는데 이것들은 우리가 日常使用하고 있는 言語와 다른 機械特有한 言語이기 때문에 이것을 機械語(Machine Language)라고 부른다. 따라서 가장 素朴한 programming은 問題의 Flow diagram으로부터 그 四則演算이나 判斷을 順序에 따라 一連의 機械語로 表現하는 것이라고 할 수 있을 것이다.

그러나 이러한 programming은 大端한 努力을 要求하고 또 그 內容에 困難을 띈 複雜한 作業이 된다.

그 理由는 前述한 바와 같이 機械語(命令)가 우리들의 日常쓰고 있는 言語와 無關한 것으로 記憶하기가 퍽 어렵고 또 命令이나 數値가 記憶되고 있는 番地를 항상 programmer가 記憶해 두지 않으면 안된다는 繁雜性이 있기 때문이다.

이러한 理由로 부터 機械語에 依한 programming은 programmer에 過大한 負擔을 지울 뿐만 아니라 이에 要하는 時間도 길어진다.

또 program 가운데 Error가 생길 可能性도 크기 때문에 그 Error를 補正한다는 것, 即 Debugging이 programmer의 큰 作業으로 될 것이다.

이것을 除去하기 爲하여서는 한가지 方法으로서 機械語 代身에 우리들이 記憶하기 쉬운 言語를 設定하여 programmer는 그 設定된 言語에 따라 program을 만들면 남어지는 計算機가 自動的으로 이것을 機械語로 번역하여 計算을 遂行케 하는 길이 있다. 이것이 곧 自動 programming라고 불려지는 것으로 오늘날의 soft-ware를 代表하는 것으로 되고 있다.

또 이것을 實現하기 爲하여서는 設定된 言語를 機械語로 번역하는 辭典에 相當하는 system program이 必要하게 된다.

새로 設定하는 言語에 對하여서는 우리가 日常使用하고 있는 言語에 가까우면 가까울수록 그만큼 program이 容易하게 되는 것이나, 그 反面 言語가 機械語로 부터 멀어질 것이므로 이것을 機械語로 번역하는 辭典도 膨大한 것으로 될 것이다.

現在 이루워지고 있는 自動 program은 크게 나누어 아래와 같이 分類되고 있다.

自動 Program
- 解釋 Routine
- 飜譯 Routine
  - Assembler(組立 program)
  - Compiler(編集 program)

解釋 Routine은 **그림 3—3**과 같이 먼저 問題를 機械語보다 記憶에 便利한 命令, 即 擬似命令(pseudo Instruction)으로 programming 한다.

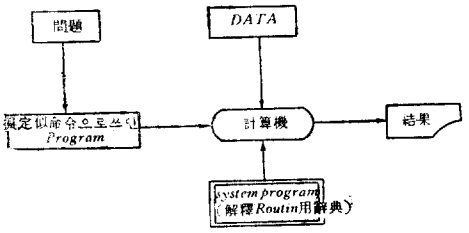


그림 3—3 解釋 Routine의 槪念圖

다음에 이 program을 計算機에 넣으면 計算機는 이

擬似命令을 앞서 記憶해둔 解釋 Routine用의 System program으로 解讀하고 이것을 機械語로 고쳐 計算을 實行하는 것이다.

곧 이것은 命令을 하나씩 하나씩 機械語로 고쳐 마치 通譯을 中介하여 이야기 하는 것과 같은 役割을 하는 이것다.

이에 對하여 번역 Routine에서는 그림 3—4와 같이 우리가 記憶하기 쉬운 言語로 쓰인 program(이것을 여기서는 Source program이라고 함)을 먼저 計算機에 넣어 미리 記憶해둔 번역 Routine用의 System program으로 번역하여 program 全體를 機械語의 program(이것을 Object program이라함)에 再編集한다. 다음에 이 Object program을 計算機에 새로 넣음으로서 所要의 計算을 實行하는 것이다.

따라서 System program은 Object program이 만들어지면 必要가 없어져 System program이 占有하고 있던 記憶場所도 使用할 수 있게 된다.

바꾸어 말하자면 번역 Routine에서는 作品全部를 譯者(Translator)에 번역시켜 가지고 그 譯本을 읽는 것과 같은 것이다.

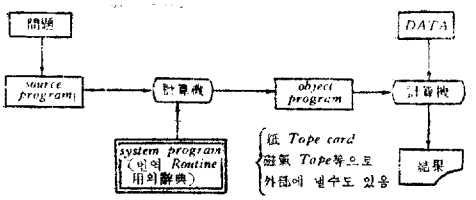


그림 3—4 번역 Routine의 槪念圖

다음에 Compiler는 program statement라고도 말하는 自動 program으로서는 가장 本格的인 것이다.

그 言語는 Assembler보다도 더욱 더 日常語에 가까운 것을 採用하여 全然 英文 그대로 또 數字, 數式을 併用하여 program을 쓸 수 있는 것이다.

따라서 機械語와 一對一의 對應이 아니고 또 이 言語에 따를 경우에는 programmer는 거의 機械自身에 對한 知識없이 機械로부터 상당히 떨어진 言語 단의 知識으로써 計算機와 對話할 수 있게 된다. 그結果로써 programming의 技術을 習得하는데 要하는 時間은 極히 短縮될 것이며 누구나 쉽게 program을 쓸 수 있게 될 것이다.

또 實際의 programming에 要하는 時間도 極度도 短縮되어 機械語, Assembler 및 Compiler의 三者를 比較한 結果에 依하면

100 : 50 : 10이 되고 있다고 한다.

(註) Programming이 容易하게 된다고 해서 問題가 쉽게 풀린다고 하는 것과는 意味가 다르다. Compiler(Fortan)를 使用하면 數學의 知識없이 問題가 풀려질 것이라고 錯覺하는 사람이 많다. 이것은 어디까지나 計算機의 知識이 거의 없어도 Program이 된다는 것으로써 數學의 知識은 如前히 必要되는 것이다.

Compiler를 使用할 때의 言語로서 科學用의 것에 Fortran(Fornuler Translator—數式번역의 意味)과 1960年1月에 國際的인 機構에 의하여 設定된 ALGOL(Argorithmic Language—算法言語의 意味)이 有名하다. 前者는 主로 美國에서 後者는 主로 구라파 諸國에서 많이 쓰여지고 있다.

이 以外에도 여러 種類의 것이 開發되고 있으나 根本的으로는 上記 兩者의 어느것에 屬한다고 하겠다.

또 事務用으로서는 1960年에 美國國防省이 國內規格으로서 制定한 Cobol (Common Business Orientel Language)이 有名하다.

이들 言語를 使用할 때의 큰 利點으로서는 이들 言語가 個個의 計算機에 依存하여 決定된 것이 아님으로 만들어진 program은 어떤 種類의 計算機에 의해서도 一部의 制約이 있더라도 거의 無難히 處理되는데 있다고 하겠다.

다음에 위에서 說明한 槪念을 明確하게 하기 위하여 具體的인 例를 하나 들어 보겠다.

問 題

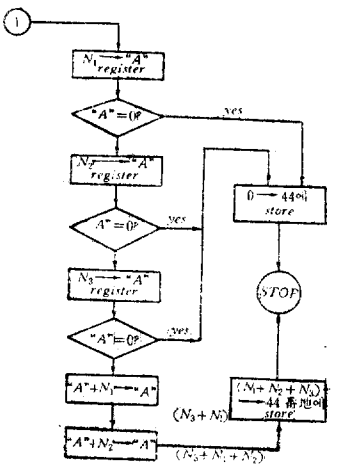
지금 番地 41, 42, 43, 에 各各 N_1, N_2, N_3이라는 數値가 들어 있다.

萬若 $N_1 \sim N_3$ 가운데 0이 있으면 이 0을 番地 44에 Store하고 計算을 停止할, 萬若 $N_1 \sim N_3$이 0이 아니면 이 셋의 和$(N_1+N_2+N_3)$를 44番地에 Store하고 計算을 停止하는 programming를 하라.

但 使用計算機의 機械語는 다음과 같다.

01××：××番地의 內容을 A-register에 가져온다.

02××：A-register의 內容을××番地에 Store 함.



03××：A-register의 內容에 ××番地의 內容을 加하여 그 結果를 A-register에 保存함.

04××：A-register의 內容이 0이면 ××番地 Jump

6666：計算을 停止함

上記 問題를 整理한 Flow chart는 아래와 같다.

機械語 Program		Assembly code program		Fortran program	
番地	命令	Assembly	命令		Statement
11	0141	LDA	N_1		
12	0421	↓ZJP	STORE↓		IF(N_1×N_2×N_3)8, 10, 8
13	0142	LDA	N_2	10	SUM=0
14	0421	ZJP	STORE		GO TO 12
15	0143	LDA	N_3	8	SUM=N_1+N_2+N_3
16	0421	ZJP	STORE	12	다음 命令을 實施
17	0341	ADD	N_1		
20	0342	ADD	N_2		
21	0244	STORE STA RESULT			
22	6666	STOP			

3—3 digitae 計算機의 多角的인 利用(Application ware)

電子計算機의 Hard ware를 效果的으로 使用하기 爲하여 Soft ware가 必要하며 이 Soft ware를 效果的으로 使用하기 爲하여서는 3의 ware로서 Application ware가 必要하게 된다는 것은 이미 說明하였다. Application ware라고 하는 것은 要컨데 電子計算機의 應用의 問題이며 이 세가지를 完全히 가춘 計算機 System만이 참다운 힘을 發揮하게 될 것이다.

다음에 最近 問題되고 있는 Application ware의 몇가지 問題點(動向)을 적어보겠다.

(1) 電子計算機와 commuication의 結合問題

現在는 電子計算機에 依한 自動制御(computing control), 遠距離間의 Data傳送, 情報의 集中處理(Computing communication)의 三者가 有機的으로 結合하여 人間과 process, 人間과 Data를 結付하는 媒體로서의 重要한 役割을 하고 있다.

(2) 電子計算機의 多重化問題

이것은 하나의 計算機에 複數個의 program을 並行的으로 處理시키고저 하는 것이다.

元來 電子計算機는 program을 直列的으로 實行해 나가는 것이므로 結局 並行的이라 하는 것은 時分割(Time sharing)이라는 形態를 取할 것이다.

이것은 主로 計算機의 入出力機構의 實行速度가 느리기 때문에 計算時間의 虛費를 적게 하자는 目的에서 生覺되는 것이다.

最近에 利用되고 있는 內容에는 第1의 program이 入出力機構等의 周邊裝置에 옮겨가 있는 사이에 다른 第2의 program을 計算機本體를 써서 計算하는데, 第1의 program의 動作이 끝나면 周邊機器로부터 그 信號가 本體에 傳해서 兩 program의 優先順位(priority)에 따라, 例를 들면 第2의 program을 中斷하고 第1의 program을 處理케하는 優先順位處理(priority processing)가 있다.

이와같은 機能의 完備에 依하여 앞으로는 보다 多角的인 電子計算機利用의 길이 열려 있다.

具體的인 한가지 例로서 電力系統의 經濟運用裝置에 있어서의 計算優先順位를 들어보겠다.

經濟運用裝置의 計算優先順位例

1. 系統內事故檢出 (系統異常時)
2. 豫測制御計算 (3分마다)
3. 融通電力制御計算 (15分마다)
4. 汐流制御計算 (60分마다)
5. DATA Logging (15分마다)
6. 翌日豫想計算 (要求있을 때마다)
7. 一般技術計算 (〃)

(3) Hybrid 計算機

最近의 Analog 計算機는 自動 program 裝置와 Analog 記憶裝置의 開發에 依하여 從來 Digital 計算機만이 取扱해온 自動反復演算이나 論理演算이 可能하게 되어 그 應用分野가 擴大되고 있다.

그러나 本質的으로 救하기 어려운 問題로서는 精度 DATA 處理性의 限界가 남겨져 있다.

한편 digital 計算機에 있어서도 高速論理素子 記憶素子演算方式이 開發되고 있으나 問題에 따라서는 Analog 計算機만 못한 것도 있다.

따라서 이들 計算機의 長點을 살려 計算을 分擔시키면서 各自의 短點을 補完하도록 하는 것이 Hybrid 計算 System이다.

이것이 實用化되고 있는 例로서 電力經濟負荷配分裝置(Economical Load Dispatching : ELD)가 있다. 이것은 水力發電所, 火力發電所, 負荷 및 이것을 連結하는 送電線으로부터 構成되는 電力系統에 있어서 하루를 通한 負荷變動과 各發電所의 容量이 주어졌을 때에 火力發電所의 하루의 燃料消費量을 最小로 하는 各發電所의

發電計劃을 만드는 것이다.

그림 3~5는 그 構成을 보인 것이다.

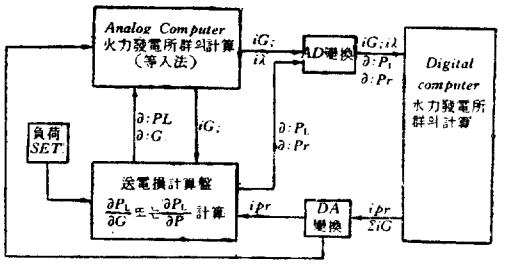


그림 3—5 Hybrid 計算裝置의 構成圖

從來 實用化되고은 ELD는 火力發電所를 中心으로 한 것이며, 火力의 發電費用이 比較的 簡單하게 計算되었으므로 Analog 計算機로 充分하였다.

그러나 水力發電所에 있어서의 물의 價値를 하루의 使用量等의 條件으로 부터 計算한다는 것이 極히 번잡하므로 水力發電所를 包含한 ELD는 Analog 計算機만으로는 精度, 裝置의 크기 等으로 限界에 達하였다. 한편 digital 計算機만으로도 可能하지만 演算時間, 聯時制御라는 面에서 問題가 있다. 따라서 計算의 分擔을 火力發電所에 關한 部分이나 送電損失計算은 Analog 計算機로 水力發電所에 關한 計算과 計算結果에 對한 DATA 處理를 digital 計算機에서 할 수 있도록 하는 Hybrid System이 實現된 것이다.

4. 電子計算機에 依한 演算의 概要

다음에 簡單한 計算機를 例를 들어 計算機를 使用할 때 어떠한 順序를 밟아 演算해 나가는가 하는 演算段階를 說明해 두겠다.

그림 4—1은 이때의 演算段階를 보인 것이다.

먼저 問題가 주어지면 要求되는 結果를 함께 考慮하여 Analog 計算機를 使用할 것인가 digital 計算機를 使用할 것인가를 決定하고, digital 計算機를 使用하게 된다면 이에 必要한 手法이 이미 開發되었거나 또는 Sub-Routine으로서 利用할 수 있는가 어떤가를 調査해 볼 必要가 있다.

未開發인 問題라면 計算機에 그 計算順序를 指示하는 一連의 手法 곧 Program을 만들게 된다. 計算機에 依한 計算手法은 손으로 直接하는 筆算과는 相當히 틀림으로 計算機에 適合되는 問題의 解法—數値計算法—에 따라 풀지 않으면 안될 것이다. 例를 들면 平方根을 求할 경우 Newton의 近似法을 使用하여

$$X_{n+1}=\frac{1}{2}\left(X_n+\frac{a}{X_n}\right)$$

但 a는 平方根을 求하고자하는 數

X_n는 初期値($\neq 0$)

에 따라 풀게된다.

其外 技術計算에서는 여러가지 方程式을 取扱하게 되는데 一般으로 方程式에 對한 解法은 여러가지가 있을 터임으로 이 中에서 가장 計算機에 알맞는 方法을 擇하여야 할 것이다.

一般的으로는 筆算때와 反對로 計算量의 多少보다도 計算過程이 單純하느냐 어떠냐에 重點을 두는 것이 좋을 것이며, 더 나가서는 機械內部에 있어서의 DATA 處理方式을 計算速度, 所要記憶容量數 및 Program의 難易度等을 考慮하여 擇하여야 할 것이다.

이와같이 計算順序가 決定되면 다음에는 Programming에 들어가게 되는데 Programming에는 보통 Flow charting와 Coding이 包含된다. Flow chart라는 것은 여러가지 約束이 있겠으나 槪略을 말하자면 여기서 使用하여 說明하고 있는 **그림 4—1**과 같이 計算順序를 Block마다 區分, 整理해 나가는 것으로서 한편 Block diagram이라고도 불려지고 있다.

複雜한 問題가 되면 될수록 이 Flow chart의 役割이 重要할 것이며 이것을 完全히 쓴다는 것이 Programming의 第一步가 될 것이다.

다만 初步的인 段階에 있어서는 아직 計算機의 機能을 確實히 把握못하기 때문에 오히려 이것을 번잡하게 느끼게 될지 모르나 充分히 時間을 들여 習得해둘 必要가 있으며 한번 習得해 놓고 나면 다음부터는 이것이 極히 便利하다는 것을 알게 될 것이다.

다음 段階는 이 Flow chart에 따라 Coding(計算機의 命令配列)을 하지 않으면 안된다. 電子計算機는 各機種에 固有한 特定의 命令밖에 實行못함으로 —이 言語를 機械語라함— 所要되는 일을 할 수 있겠금 모든 順序를 機械語로 指定하여야 하는 것이다.

最近에는 特히 이 Coding의 簡便化를 의하여 여러가지로 自動 Programming이 開發되고 있으나 이 部分이 소위 말하는 Programming으로서 問題의 計算內容을 忠實히 履行하도록 注意해야 할 것이다. 이 Coding은 Coding sheet에 쓰여지고 이것을 Tape에 Punch하거나 Card에 穿孔하게 된다. 使用者로서의 準備段階는 여기까지 이지만 한번 Coding해서 이것이 그대로 完全하게 움직인다는 것은 極히 드문일이다.

몇차례에 걸쳐 Error를 較正하면서 漸次完全한 Program으로 이루워진다는 것이 普通이다.

따라서 優秀한 Programmer라는 것은 어디까지나 처음부터 全然 Error도 없는 Program을 만든다는 사람이 아니고 오히려 Error를 迅速的確하게 發見하여 適切한 處置를 할줄 아는 사람이라고 하겠다.

Tape 또는 Card가 만들어지면 Operator의 손으로 이것을 計算機에 넣게 된다. 計算機記憶裝置 속에 들어

간 Program은 Console의 Switch(button)에 依하여 그 順序에 따른 計算이 開始되고 DATA는 記憶裝置와 演算回路間을 往復하면서 結果가 Store되고 必要에 따라 Output 裝置를 通하여 印刷되어 演算이 끝나는 것이다.

(以下 次號 繼續)

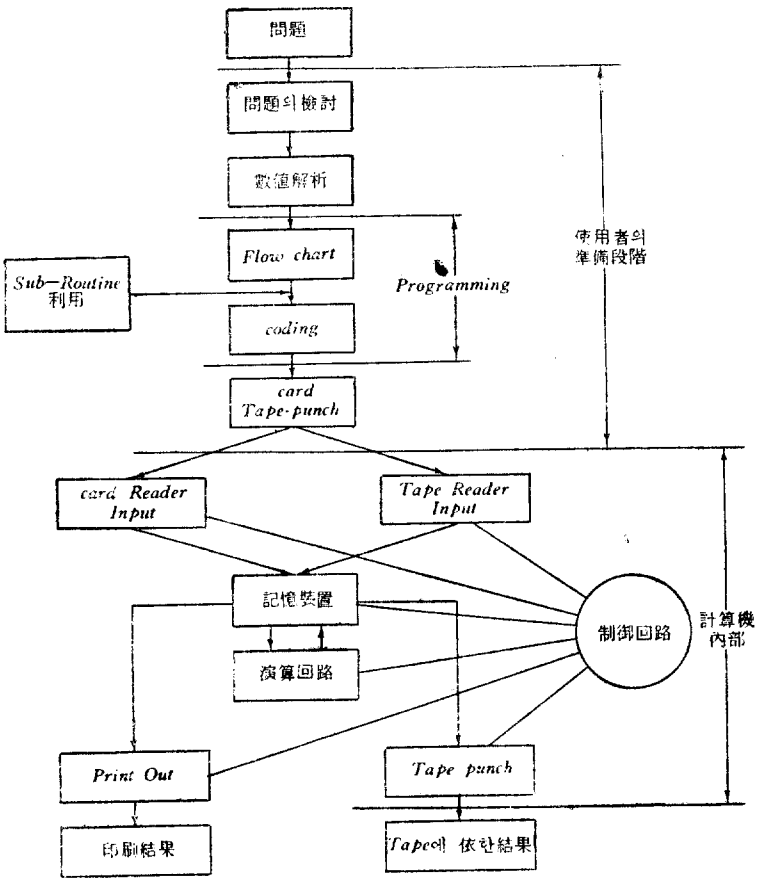


그림 4—1 演算段階說明圖