技術講座

電 子 計 算 機 (三)

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.

2.5 入出力裝置

電子計算機의 入出力裝置端末裝置를 포함한 것으로 한다)에 대해서는 최근 技術上의 劃期的인 進展은 보이지 않으나 將來의 開發의 重點은 入出力의 機械的部分을 可能한 限 電子化해 나가는데 있다고 생각한다. 그리고 入出力裝置로서 直接 視覺 또는 聽覺에 전달되는 形式이 續續登場할 것으로 期待된다. 光學的文字 Reading 裝置(OCR나 OMR)이 더 開發됨에 따라 印刷된 또는 손으로 쓴 文字를 그대로 入力으로서 Reading 할 수 있고 그 利用이 普及될 것이다. 出力裝置端末裝置로서 CRT(陰極線管)을 使用한 디스프레이(display) 裝置가 많이 使用되는 傾向에 있고 이에 鍵盤을 이어 入出力의 機能을 가진 端末裝置가 普及될 것이고 또한 表示된 情報를 複寫할 수 있는 電子裝置의 需要量도 擴大될 것으로 생각된다.

音聲應答裝置로서는 풋슈버튼式電話機(텃취·토ー니電話機가 그대로 情報處理의 端末器로서 利用되기 때문에 데이타 通信의 發展에 따라서 텃취·토ー니電話機가 端末裝置로서 大活躍을 하게 될 것이다.

音聲을 直接 電子計算機에 入力시키는 것은 아직은 困難한 問題로 되어 있어 當分間 實現의 可望性이 적다.

入出力裝置에 關해서 一般的으로 말할 수 있는 것은 裝置의 動作의 確實性·信賴性의 向上에 加一層의 努力이 必要하다는 것이다. 그렇게 하면 電子計算機의 하아드웨어·시스템 全體의 信賴性을 올릴 수가 있다.

以上電子計算機 하아드웨어를 構成하는 各裝置中에서 基本的이 에레맨트를 골라서 그 技術的動向을 전망한 것인데 그 움직임의 主流를 이루고 原動力이 되어 있는 것은 實로 IC인 것이며 電子計算機의 하아두웨어의 構成에 짐작하기 어려울程度 큰 影響을 줄려고 하고 있고 今後도 그 影響이 擴大될것으로 보인다. 더군다나 IC는 單只 電子計算機의 하아드웨어에 對한 영향 뿐이 아니고 그 쏘프트웨어面에 있어서도 顯著한 進步改善을 갖어올려고 하고 있는 것이다.

3. 쏘프트웨어技術의 動向

3. 槪 說

電子計算機의 쏘프트웨어라함은 電子計算機의 하아드웨어를 能率좋게 움직이고 所望된바 情報處理를 하게 하기 爲해서 作成되는 프로그램等을 말한다. 이는 電子計算機의 利用技術이라고 말할 수 있다. 高性能의 電子計算機도 强力하고 融通性있는 쏘프트웨어가 없으면 充分한 機能을 發揮할 수가 없다.

十餘年前에는 電子計算機의 프로그램의 스텝(命令)數는 1,000程度것이 많았다.

오늘날에는 數十萬스텝數에 도달하는 프로그램으로서 움직이고 있는 시스템이 흔하고 100萬單位 스텝도 나타나기 시작하고 있다. 나아가서는 이것이 千萬單位 스텝을 向할것으로 생각된다.

電子計算機의 코스트·퍼ー퍼맨스라고 하면 前에는 오로지 하아드웨어의 코스트·퍼퍼맨스

로 評價되었지만 쏘프트웨어가 하아드웨어에 不可缺한것이 된 다음으로 부터는 兩者 續合한 코스트퍼—퍼맨스를 생각하게 되었다. 將次는 도리혀 쏘프트웨어의 코스트·퍼—퍼맨스가 하아드웨어보다 重要한 要素가 될 것이다.

오늘날 하아드웨어와 쏘프트웨어 關係를 보면 쏘프트웨어가 過重한 感이 있어 그 不均衡을 어떻게 調整하느냐 하는데에 問題가 介在함을 알 수 있다. 또한 프로그램 生產의 合理化問題 하아드웨어技術과 쏘프트웨어 技術을 어떻게 綜合 또는 融合시키는가의 問題, 쏘프트웨어 標準化의 問題等 現在와 未來에 풀려야 할 問題가 많다. 그러나 全世界 사람들이 이 問題에 關心을 이미 갖기 始作하고 있으니, 今後十年間 하아드웨어 技術과 쏘프트웨어 技術에는 놀라운 進展이 있으리라는 것이 豫想된다. 그것은 電子計算機에 있어서 十年前의 技術과 오늘의 技術을 比較해보면 알수 있는 있이다.

3.2 쏘프트웨—어의 分類

오늘날의 쏘프트웨어는 다음 다섯가지로 分類할 수 가 있다.

(1) 制御프로그램

制御 프로그램은 씨스템管理, 타스크管理, 데이타管理, 잡管理(Job)等의 各 프로그램로서 成立되어 있다.

(2) 處理프로그램

(ㄱ) 言語프로셋써

言語프로셋써는 프로그램言語로 쓰인프로그램을 機械語로 變換하는 것으로서 아셈블로—·Algol 콤파일러—, Cobol 콤파일러—, Fortran 콤파일러—. PL/I 콤파일러—, 自然言語處理用 프로그램等이 있다.

(ㄴ) 써어비스·프로그램

써어비스·프로그램에는 쏘—트·마—즈(파일의 組合을 가지고 分類를 行하는 方法), 유—틸리티—, 디벅·에이드(Dedug·aid), 링케이즈·애디터—(LinkageEditor)等이 있다.

(ㄷ) 汎用어프리케이션·프로그램

汎用어프리케이션·프로그램에는 씨뮤레이터—, 프로세쓰制御, 情報檢索, 스케듈링 線形計劃等에 對한 프로그램이 있다.

(ㄹ) 各種 問題向 프로그램

이들中 制御프로그램, 言語프로셋써—, 써어비스·프로그램은 基本프로그램로서 電子計算機 메이커어가 유—서—에 提供한다. 汎用어프리케이션·프로그램은 메이커어가 만들 경우와 유—서—가 만들 경우가 있으며, 各種 問題向 프로그램은 보통 유—서—가 만들게 되어 있다.

1969年 3月 調査에 依하면 유—서—가 保有하는 (유—서—가 責任져서 만든) 프로그램數가 約 18萬個 메이커어가 保有하는 프로그램數가 約2萬3千個, 쏘프트어웨어會社가 保有하는 프로그램이 約2千5百個였다.

이들 쏘프트웨어의 內容을 調査해보면 電子計算機의 하아드웨어의 進步와 마찬가지 쏘프트웨어 機能과 性能이 高度化로 繼續했다는 것을 알수 있다.

3.3 오퍼래이팅·씨스템

오퍼레이팅·씨스템(OS)란 電子計算機씨스템 全體의 操作效率을 올리기 위해서 만들은 全쏘프트웨어 體系를 말한다.

狹義로는, 이 쏘프트웨어의 中心이 되는 制御 프로그램을 OS라고 한다. 또 制御프로그램을 모니터—라고 불으는 時代도 되었다.

오퍼래이팅·씨스템을 프로그램의 形式으로 말하자면 當初의 프로그램言語時代로 부터 종파일러—·모니타—·씨스템時代로 發達하여 그 다음에 오늘날의 타임셰어·멀티·프로그래밍時代로 進化하여 왔다. 電子計算機의 利用面으로 부터 보면 當初의 科學計算時代로부터 科學事務計算時代로 옮겨가고 그다음 오늘날의 電子計算機와 人間의 對話를 通하여 상당히 큰일을 하는 맨·머신時代의 初期를 맞았다고 말할 수 있다. 將次는 더욱더 쏘프트웨어 體系가 高度化하여 좀더 人間의 頭腦의 機能에 接近한 일을 하게 될 것이다. 電子計算機利用의領域은 點으로부터 線으로, 線으로부터 面으로, 그리고 今後 立體

로 擴大한다. 오퍼레이팅·씨스템은 그러한 進化의 內容을 나타내는 하나의 標本이라고 볼 수 있다. 오퍼레이팅·씨스템은 制御프로그램과 處理프로그램로서 構成되어 있고 그 內譯은 前項(2)에 記載되어 있다. 要는 各種프로그람에 따라서 各種 쟙(Job)이 自動的·連續的으로 處理되고 또 멀티·프로그래밍手法에 依해서 여러개의 獨立프로그램이 同時에 處理되는 것이다. 그 結果 一定時間內에 處理될 수 있는 일의 量이 커지고 일의 着手로 부터 完了까지의 時間이 減少하기 때문에 電子計算機의 하아드웨어가 가지고 있는 高速情報處理能力을 最大限으로 살리게된다는 것이다.

타임쉐링·씨스템은 汎用오퍼이팅·씨스템의 一種으로서 制御프로그램, 處理프로그램을 능란하게 驅使하지 안으면 그 機能을 發揮시킬 수가 없는 것이다. 그리고 또한 現在 아직 開發途上에 있는 것인데 開發努力의 殆半은 그 쏘프트웨어에 集中되어 있다. 또한 御御프로그램을 어떠한것으로 하는것이 最適인가 하는 것이 研究의 中心이되어 있는것 같다.

特定業務를 對象으로 하는 온라인·리얼타임·씨스템에 있어서도 電子計算機의 오퍼래이팅·씨스템이 그 活躍을 보여준다.

3.4 現在프로그래밍手法

現在의 프로그래밍手法은 當初와 比較하면 그 進步가 놀라웁다. 當初 機械語 즉 計算機가 直接理解하고 作動하는 命令語로 써야 했던 言語가 요사히는 우리가 普通 使用하는 言語로 쓸수가 있고, 이것이 自動的으로 機械語로 번역이 되어서 電子計算機가 움직인다. 作成된 프로그램의 잘못도 電子計算機를 使用해서 수정된다. Algol, Cobol, Fortran PL/Ⅰ等은 가장 便利한 電子計算機 言語로서 알려있고, 이들을 機械語로 變換하는 프로셋써 即 프로그램도 完備되어 있다.

이러한 實情에도 不拘하고 現在 프로그래밍手法은 아직 확실한 學問, 技術이 되어있지 않다. 그것은 같은 目的을 爲해서 作成된 프로그램이 作成하는 사람에 따라 多少 틀리다는 것이며 또한 作成된 프로그램은 完全히 他人이 흉내낼 수 없는 藝術品과 같다 하는 얘기이다. 이는 프로그램의 더큐맨태이션 (文書化)가 아직 完備되어 있지 않는 것이 하나의 原因이 되어 있다.

普通의 物品의 生產이라면 仕樣書에 依해서 設計하여 製作하고 檢査할 수가 있고 生產의 工程도 管理할 수 있지만 電子計算機의 프로그램 生產은 이러한 과정의 細目에 있어서 아직 決定못하는 것들이 있다. 이때문에 프로그래머어 A가 무슨 事情에 依해서 프로그래밍作業을 中止했을때 이것을 B가 引繼받아도 B는 그때까지 A가 作成한 프로그램을 살려서 다음을 이어가지 못하고 結局 그 프로그램을 처음부터 고쳐써야 한다는 實情인 것이다.

따라서 作成된 프로그램을 電子計算機에 걸어보면 같은 目的을 위해서 作成된 프로그램일지라도 쓴 사람에 따라 電子計算機의 使用時間이 다르다. 숙달한 사람이 作成한 프로그램은 命令數가 적어서 電子計算機를 必要以上 動作시키지 안는다. 프로그램 生產의 機械化는 他項에 서술한 바와 같이 部分的으로 되기 始作하였다고는 하지만 아직 당분間은 人海戰術로서 프로그램生產이 이루어지고 있는 것을 피할 수 없고 電子計算機의 爆發的인 利用增大에 對해서 프로그래머어의 不足은 今後 加一層 深刻한 事態가 되리라고 豫想된다.

3.5 1人當 쏘프트왜—어生產高

우리나라(日本)어느 會社에서 어느 機種의 電子計算機에 관해서 쏘프트웨어 生產의 실태를 調査했더니 다음과 같은 結果가 얻어 졌다.

制御프로그램	14,000 스탭
處理프로그램	160,000 스탭
其他	20,000 스탭
合計	約 200,000 스탭
所要延人員	40人年

즉 1人年間 平均 5,000스탭의 命令을 쓴것으로 된다. 土曜日을 完全히 쉬고 超過勤務도 안한다면 한사람 年間平均 3,000~4,000스탭 程度일 것이다.

美國 어느會社에서는 쏘프트웨어 生產의 能

率로서 다음과 같은 數字를 發表하였다.

① 뉴우·넥널러지— 500~600 스탭/人年

② 에스타브 릿쉬드·넥널러지— 1,000~3,000 스탭/人年

③ 프로덕트·넥널러지— 6,000~10,000 스탭/人年

①은 新技術開發段階에 있어서의 能率일 것이다. 이 段階에서는 한사람이 하루 平均 2스탭밖에 못쓰는 꼴이 된다.

②는 新技術이 開發되어서 이것이 市場에 나올때의 能率일 것이다.

③은 프로그램 作成이 容易하게 되고 大量生産에 돌입했을 때의 能率일것이다. 前記 日本경우는 美國경우의 ②와③ 中間程度에 該當하게 되지 않을까? 日本사람의 能率이 美國사람能率보다 相當히 높은 것으로 되어 있는데 그것이 믿어지지 않는다 손치더라도 日本사람의 프로그램作成能率이 美國사람能率보다 떨어지는 것으로 생각되지는 않는다.

3.6 쏘프트웨어의 디벅(Debug)

쏘프트웨어 生産效率을 나쁘게하는 要因의 하나는 프로그램이란 반드시 디벅해야 完成된다는 것이다. 디벅이란 벌레를 除去한다는 뜻이며 프그로램上 잘못을 發見하고 訂正함을 말한다. 쏘프트웨어 코스트中 디벅이 차지하는 비중은 相當히 크고 때로는 全코스트의 數 10% 以上을 차지 할때가 있다. 試行錯誤形式으로 프로그램을 作成해야 할때가 많기때문이며 프로그램의 코—딩(Coding)같은 것은 이것에 比하면 훨씬 적은 比重을 차지한다. 이 디벅은 電子計算機를 써서 行해지기때문에 디벅을 爲해서 電子計算機의 使用時間을 어떻게 줄이는가 하는 것이 쏘프트웨어의 코스트를 줄이는데 窮理해야할 重要한 事項이다. 디벅의 能率을 올림으로서 프로그램이 半의 코스트로 될수있다고 한다.

메이커어는 쏘프트웨어를 生産하는데 프로그램의 디벅에 電子計算機使用時間의 5~10% 또는 그以上을 쓰는 것으로 간주하고 있다. 유—서—를 위해서도 랜털料 月400萬원 내지 800萬원(韓貨)이나 되는 計子計算機를 使用할때 디벅을 위한 費用은 無視 못한다. 따라서 디벅을 高能率로 行하는 方法을 考案하고 있는데 多數 유—서—의 共同利用을 爲한 타임쉬어링·씨스템을 利用할수 있다면 그것이 가장 有效한 手段이 될것이다.

또한 디벅의 難度는 프로그램 總스탭數의 自乘에 比例한다하니 긴 프로그램을 한꺼번에 디벅한다는 것은 다시 할때마다의 損失이 대단해서 混亂을 초래하는 原因이된다. 따라서 다음과 같은方式이 取해진다.

하나의 프로그램은 몇개의 모듈(Module)로서 構成되어 있다. 즉 하나의 프로그램은 同質의 部分과 異質의 部分으로서 成立되어 있다. 이 各部分을 모듈 또는 썹·프로그램(Sub-program)라고 한다. 따라서 먼저 프로그램을 적당히 分割해서 모듈化하고 이 하나하나의 모듈을 디벅한다. 다음에 이 디벅된 모듈을 結合해서 다시 디벅한다. 이와같이 하여 作成된 프로그램을 짜넣어서 目的의 프로그램을 作成하면 된다.

3.7 쏘프트웨어의 모듈化

프로그램의 모듈化에 關해서는 이미 서술한 바 있으나 하나의 모듈이 大槪 어느정도의 스탭數로서 成立되어 있는가 하면 100~1,000 스탭 平均해서 300스탭 程度이다. I B M 社의 오퍼래이팅·씨스템의 一例를 보면 100萬스탭이 34個의 콤포넨트(Component)로서 되어있고 모듈數가 3,300이니까 하나의 모듈은 平均 300스탭로서 構成되고 하나의 콤포넨트는 100個의 모듈로 되어 있다. 이 콤포넨트는 例를 들어 콤파일러(Compiler)를 構成하고 있는 그러한 것이다.

또 쏘프트웨어로서 構成되는 씨스템에 關해서도 모듈이라는 것이 있다. 例를 들어 타임쉬어링·씨스템에서는 時間的인 管理(프로세스의 스케듈, 入出力機器의 動作스케듈), 空間的인 管理(主記憶裝置의 割當管理, 外部記憶裝置의 割當管理), 機能의 管理(入出力의 特殊動作의 制御, 유—서—로부터의 制御要求의 變換, 誤動作 잘못된 프로그램의 處理等)等이 必要 되는데 이들 諸機能은 다시 對象마다 分類整理되어서 프로그램全體는 몇개의 모듈 即 썹씨스템으로서

構成되어 있다.

즉 타임쉐어링·씨스템에서는 다음과 같은 모듈(썹씨스템)로서 構成할 수가 있다.

①프로세스·씨스템(프로세스를 바꾸기 爲한)
②메모리—·씨스템(메모리—의 割當·保護用)
③콘솜·씨스템(코맨드의 處理, 入出力管理用)
④파일·씨스템(카타로그管理, 入出力制御·保護用)

이와 같이 해서 아무리 複雜하고 膨大한 씨스템도 機能的으로 모듈化해서 다시 構成한다는 方式으로 씨스템이 만들어진다.

3.8 쏘프트웨어의 팩케즈化 (Package)

一般的으로 電子計算機의 유—서—는 自己의 業務에 알맞은 프로그램을 作成해야 된다. 그 프로그램에는 極히 簡單한 것으로 부터 대단히 複雜한 것까지 범위가 넓다. 이러한 프로그램은 프로그래머어가 만드는 것이니 電子計算機設置數의 激增은 프로그래머어의 需要를 뜻하는 데 이 需要供給의 언바란스는 쉽게 解消될것 같지 않다.

例를 들어 1968年 3月 現在에 있어서 우리 나라(日本)의 電子計算機要員은 17,740名, 細分하면 씨스템·아날리스트 7500名, 프로그래머어 1,0240名이다. 이것이 1972年에는 그 約 3.5倍에 該當하는 65,000名 細分하면 씨스템·아날리스트 20,000名 프로그래머어 30,000名이 必要하게된다.

이러한 事態에 對備하기 위해서는 프로그래머어 充足에 努力을 해야함은 勿論이지만 그것만으로는 問題가 解決되리라고 생각되지 않는다. 그러면 어떠한 딴方法이 있는가? 하나의 方法은 프로그램作成의 自動化를 推進하고 同時에 從前과 같이 맞침프로그램을 되도록이면 기성품 프로그램作成 方向으로 이끌어 가는 것이다. 이것은 또한 프로그램의 코스트 低減을 위해서도 좋은 方案이다.

쏘프트웨어의 팩키이즈(Package)는 간단히 말해 "쏘프트웨어통조림"이며 어느 特定의 일을 하기 위해 만들어진 프로그램旣成品이다. 이는 맞춤것보다 훨씬 싸다. 유—서—는 이 팩케이즈에 없는 프로그램만 만들면 되고 나머지는 쏘프트웨어·팩케이즈中에서 必要한 部分만 가지고와서 電子計算機에 맞추어 넣기만 하면 된다.

쏘프트웨어·팩케이즈는 이미 美國에서는 多數發表되어 實用化되고 있다. 우리나라(日本)에서도, 最近 이것이 많이 製作되는 경향에 있다. 現在 IOCS(인풋·아웃풋·콘트로—르·씨스템)라고 하는 周邊機器를 制御하기 위해 使用되는 적은 팩케이즈로부터 給與計算, 在庫管理, 需要豫測과 같은 事務計算, 나아가서는 回歸分析, 微分方程式과 같은 科學技術計算, 等의 어프리캐이션·프로그램에 이루기까지 여러 팩케이즈가 돌고 있다. 其他 最近話題가 되어 있는 MIS(멘니즈맨트·인포메이션·씨스템)의썹씨스템으로서의 팩케이즈나 타임 쉐어링用으로서 유—서—가 技術的인 知識이 없어도 遠方의 端末裝置로 부터 容易하게 中央의 電子計算機를 使用할 수 있게한 쏘프트웨어·팩케이즈가 있다. 이러한 팩케이즈에는 十餘萬원으로부터 數百萬 또는 그 以上가는 것 까지 있다. 第1表에 美國에서 發表되어 있는 쏘프트웨어·팩케이즈의 一例를 수록하였다. 今後 科學技術計算 및 事務計算兩域에 걸쳐 汎用팩케이즈 및 專用팩케이즈가 繼續해서 市場에 出現할 것으로 보인다.

또한 쏘프트웨어·팩케이즈같은 기성품 프로그램 또는 注文品이 아니고 이지오더어(Easy Order)式의 쏘프트웨어가 美國에 발달하여 이 方式으로 장사를 시작한 사람들이 생겼다. 이것은 쏘프트웨어의 코스트를 줄이는 方法으로서 어느 假想의 電子計算機에 對해서 쏘프트웨어의 原型을 만들어 놓고 電子計算機의 機種이 定해지면 그것을 고쳐서 最適의 쏘프트웨어로 하는 方法이다.

3.9 쏘프트웨어의 標準化

今後情報處理產業의 發展에 따라 電子計算機의 하아드웨어와 쏘프트웨어 사이의 인터어페이스

(Interface)(情報信號의 授受가 行해지게 서로 接續된 두개以上의 裝置에 있어서, 그 接續의 境界로 생각되는 假想的인 面)을 規定하고 統一한 다음에 標準化를 進行시킬 必要가 더욱더 있게 된다. 즉 生産된 쏘프트웨어가 情報處理網內에서 充分한 互換性을 갖는 것이 대단히 重要한 事項이 된다는 뜻이다.

그러나 하아드웨어 技術이나 쏘프트웨어技術의 發達이 큰데 그 進步를 방해하는 일이 없이 쏘프트웨어의 標準化를 實現시킨다는 것은 대단한 難事業으로서 이 標準化의 問題는 國家의 政策으로서 强力하게 對處할 必要가 있다. 이 경우 쏘프트웨어의 標準化와 하아드웨어의 標準化를 分離해서 생각할 수 없다.

쏘프트웨어 標準化의 對象은 具體的으로 프로그램作成에 關한 事項. 一例를 들어 文字의 種類, 쓰는법, 코오딩포—ㅁ, 플로우차아트·씸볼, 플로우차아트그리는 法의 標準化 및 프로그램言語의 統一, 프로그램使用手續의 標準化, 프로그램의 說明書쓰는 方法 및 그 用語의 統一等이다. 各種用語等에 關해서는 이미 檢討가 끝나고 JIS化된것 및 1970~71年度를 期해서 JIS化될 豫定인것이 많다.

쏘프트웨어의 標準化의 效果로서 例를 들면 이미 記述한바와 같이 쏘프트웨어의 팩케이즈가 어떠한 매이커어가 만든 電子計算機에도 互換性을 갖게되고 쏘프트웨어의 生産性도 현저히 向上하고 프로그램의 共同開發 및 프로그래머어 敎育도 容易하게 되는 點을 지적해 둔다.

第1表 旣發表쏘프트웨어·팩케이즈의 例

쏘프트웨—어·팩케이즈名	開發機關名	使用目的
GIS(General Information System	IBM(International Business Machines)	一般的인 用途를 위해서 開發된파일·프로셋싱씨스템
Manage	SDS(Scientific Data Systems)	在庫管理나 人工衛星에 對한 地上으로부터의 指令等에 關한 파일·메내이즈맨트씨스템
INFOL(Information Oriented Language)	CDC(Control Data Corporation)	IR(情報收集과 檢索)用씨스템
INRAD(Information Management Retrieval and Dissemination System)	VNIVAC	IR用
BEST (Business EDP Systems Technique)	NCR(National Cash Resister)	베이타프로셋싱을 爲해서 集約해서 하나의 System로 한것
TDMS(Time-Shared Data Management System)	SDC(System Developement Corporation	User가 使用하는 計算機言語 Data를 Base로 한 System. Time Shared System를 위해서 使用된다.
ASI-SΓ	Application Software	第3世代의 計算機用 System. MIS(Management Information System)에 使用
SCERT	COMRESS	Data Processing에 必要된 計算機의 機能도 포함 MIS에 使用된다.
PROMPT	Aries Corporation	Program의 修正과 計劃을 포함 새로운 Management Control Package
AUTOFLOW	AppliedData Research Inc	Program의 Source Deck로 부터 自動的으로 Flow Chart를 作成
CFSS(Combined File Search System)	S3C(Service Bureau Corporation)	IR에 使用 File Management. Program가 Package가 된것
FFS(Formatted File Sysfem)	IBM	
ICS(Information File System)	North American Rochwell	
DM/I	Auerbach Corporation	
Mark Ⅳ	Information Inc	

現在 우리나라에서는 美國과 같은 모양으로 쏘프트웨어 技術을 보호하는 特許權이나 著作權等은 認定되어 있지 않다. 따라서 좋은 着想은 極力 外部에 새지않게 注意해가면서 各會社마다 各樣의 생각으로서 쏘프트웨어의 生産이 行해지고 있는 實情이다. 이에 對해서 쏘프트웨어의 創作에 대한 權利를 認定하는 特許權 또는 그에

준한 權利를 주어 쏘프트웨어 技術의 交流를 꾀해야 된다는 意見도 나와 있다.

美國에서는 이 쏘프트웨어의 特許權問題가 議會에서도 手扱되어 討議에 붙인바 있었으나 時期尙早라고 아직 結論이 안나와있다. 그런데 最近 와싱톤에서 "Law of Software"라는 議題로 會議가 열려 關係者에 依해서 쏘프트웨어의 特許權問題가 論議되었다. 議會에서도 이 問題를 또다시 審議해서 어떠한 結末을 지어야 할듯하다.

今後 우리나라(日本)에서도 이 問題에 대해서 關係分野에서 檢討를 되풀이하게 되겠지만 여태껏 特許權이 問題視되지 않은 것은 비록 지금 쏘프트웨어의 技術이 급격히 進展되고 있다고는 하지만 生産 프로셋쓰의 技術이 아직 幼稚해서 今後 進步에 크게 期待를 걸고 있음이 實情이며 그다지 쏘프트웨어 着想의 盜用등을 問題로 할 必要가 없었기 때문이다. 事實上 쏘프트웨어팩케이즈等이 盜作되었다는 말은 못들었다.

1968年 6月, IBM副社長의 J.W·버킨스톡氏가 來日하여 "우리들의 未來를 形成하는 爆發的變化"라는 題目으로 日本情報處理學會主催의 講演會에서 演說했는데 거기에서 氏는 쏘프트웨어의 特許權問題에 대해서 다음과 같은 要旨를 말하고 있다. "프로그래밍의 技術은 今來十年사이에 長足의 進步를 나타내어 쏘프트웨어 關係에 數十萬名의 專門家가 종사하고 있다. 이 進步發展은 새로운 프로그래밍의 槪念, 技術및 細目을 世界어디에서나 關心이 있는 사람들 사이에 어느정도 自由롭게 交換할 수가 있었다는 点에 起因한 것으로 보인다.

그러나 電子計算機 프로그래밍 發展을 위해서 이를 特許制度에 依해서 保護해야 한다는 主張도 있다. 그 까닭은 特許의 保護가 없으면 프로그래밍의 技術이 秘密取扱되어 技術水準向上을 防害한다는 뜻에서 이다. 이 主張에 나는 贊成할 수가 없다. 그까닭은 電子計算機 프로그래밍에 特許를 주고 保護한다는 것은 今後에도 지금까지와 같이 發展을 繼續하는데 큰 支障을 주기 때문이다.

왜냐하면 첫째 特許가 될만한 새로운 着想이 特許에 依해서 保護된다는 措置가 確立될때까지 사람들이 이것을 利用할 수가 없다는 点이다. 또한 電子計算機 유—서—는 自己 프로그래밍 活動을 그 法的인 立場이 詳細하게 檢討가 完了할때 까지 發表하지 않고 두어야 하기 때문이다 다시 또한 유—서—는 他유—서—의 特許權에 依해서 自己의 自由가 制限되어 있는 것을 알았을때 防禦措置로서 自己의 프로그램에 대해서도 特許申請을 아니할 수가 없게 된다. 그렇게 되면 特許청에서는 價値나 獨創性이 疑心스러운 프로그래밍의 特許申請沙汰에 시달리게 될 것이다.

結論으로서는 프로그래밍에 特許를 주면 電子計算機의 利用은 制限되고 國家經濟에 미치는 惡영향이 클 것이다. 나는 立法府와 特許청이 이를 認識하여 프로그래밍에 特許를 안주기로 決定을 내리기를 念願하고 있다.

版權이 붙어있는 情報에는 印刷된 情報資料와 印刷되어 있지않은 머쉰·베이스의 情報資料와의 二種數가 있다. 그 어느쪽에 대해서도 版權所有者에게 自己의 版權對象物을 販賣하고 또는 制限하는 能力에 關한 모든 傳統的 權利는 維持되어야 할 것으로 생각한다. 또 한편 이 情報를 축적하고 販賣하는 業者는 이 情報의 利用者에 대해서는 版權所有者에게 金錢的報酬를 지불한 後에 自由롭게 그 內容을 선택하고 또는 複寫·印刷를 할 수 있게 하면 하는것이 나의 所見이다.

3.11 쏘포트·웨어 偏重의 경향

電子計算機應用分野의 擴大에 따라서 쏘프트웨어가 情報處理技術分野의 主役을 맡게되었다. 이것은 電子計算機 利用의 高度化가 電子計算機 하아드웨어를 쏘프트웨어로서 管理함으로서 얻어진다는 데 그 原因이 있다. 電子計算機가 쓰이기 시작한 무렵에는 電子計算機의 總코스트中 쏘프트웨어가 차지하는 比率은 不過 10餘퍼센트이었는데 오늘날에는 하아드웨어와 쏘프트웨어의 比率이 50對50이 되어 있다. 이 狀態로 進展하면 멀지않아 쏘프트웨어의 比重을 總 코스트의 80퍼센트에 도달하리라 豫測된다. 美國에서는 1967年에 쏘프트웨어의 開發費로서 책정된

金額은 一兆圓을 넘었고 每年 20 乃至 30퍼—센트의 率로 增加하고 있기 때문에 1970年에는 3兆圓이 되리라는 豫想이다.

쏘프트웨어의 開發이 이와 같이 强力하게 推進되고 있다고 하는 것은 電子計算機의 利用擴大上 必要한 일이기는 하지만 이와같은 하드대 쏘프트比의 傾向에 대해서 反省하기 始作했다. 即 最近의 타임·쉐어링씨스템에서는 쏘프트웨어 開發에 莫大한 비용이 걸리고 또한 쏘프트웨어 介在에 依한 電子計算機의 오우버어헤드·로스가 數10퍼—센트 以上이 되는 点으로 보아 하드와 쏘프트의 코스트 比率의 合理的 반란스를 어떻게 잡어야 하느냐하는 問題가 고려되기 始作하였다.

이 問題는 지금까지 하아드웨어와 쏘프트웨어 技術사이에 境界線이 그어져 있어서 쏘프트웨어 사람들이 하아드웨어와 관계없이 어떤 問題이건 쏘프트웨어로서만 解決하려고 하는 傾向이 있어서 하아드웨어로서 解決될 問題까지도 쏘프트웨어가 움켜쥐고 있었다하는데도 關聯된다. 그러면 쏘프트웨어에의 依存度가 過重하다는 (一種의) 非正常상태를 시정하기위해서는 어떻게 하면 되는가 그 方法의 하나로서 最近 하아드웨어의 현저한 發達을 活用하여 하드와 쏘프트의 機能의 分擔을 어떻게 하느냐 하는 課題가 고려되어 있다. 다음에 서술하는 펌웨어의 提案도 그 中하나이다.

3.12 펌웨어의 役割

美國의 A·오플러氏는 現在의 마이크로 프로그래밍이 LSI等의 利用에 依해서 쏘프트웨어가 하아드웨어와 結合하여 더욱더 發達한 쏘프트웨어가 出現하리라는 見解로부터 쏘프트와 하드中間에 펌웨어라고 이름을 붙일 것을 提案하고 있어 쏘프트웨어와 하드웨어를 이어 第三의 웨어가 出現하려고 하고있다.

펌웨어는 LSI等의 高速 메모리를 使用한 固定記憶裝置와 마이크로·프로그래밍으로서 構成된 것으로 볼 수 있다. 그러면 固定記憶裝置와 마이크로·프로그래밍과의 關係는 어떠한 것인가 이는 다음과 같이 說明할수가 있다.

電子計算機의 어떠한 命令도 基本的인 몇개의 마이크로 操作의 合成으로 表現할 수 있다. 例를 들어 게이트 操作等은 많은 命令에 共通의 마이크로 操作으로서 간주할 수 있다. 이러한 마이크로 操作을 並列로 行할 수 있게 合成한것을 마이크로 命令이라 하고 다시 그 組合을 마이크로·프로그램이라고 부르고 있다. 모든 命令을 마이크로 프로그램으로서 實行한다면 그 콘트롤 씨스템은 월등히 簡單하게 된다. 이는 마이크로 프로그램을 貯蓄한 固定記憶裝置로서 쉽게 具體化 된다.

이러한 方法의 利点은 計算機의 構成이 簡單하고 論理設計 및 保守가 容易하게 되고 機能에 融通性이 生긴다는 点에 있다. 缺点은 計算機의 動作이 固定記憶裝置速度로서 限定되어 버린点에 있다. 그러나 固定記憶裝置에 LSI가 使用된다면 이 缺点은 除去된다.

이로서 알 수 있는 바와 같이 今後期待되는 펌웨어는 LSI를 使用한 固定記憶裝置와 마이크로·프로그래밍으로서 形成되는 情報處理方式으로서 이 方式은 하아드웨어편에서 보면 쏘프트웨어이고 쏘프트웨어편에서 보면 하아드웨어 인것같이 하아드웨어와 쏘프트웨어의 境界에 位置하는 웨어이다. 펌웨어는 電子計算機의 코스트·퍼—포—맨스를 向上시키는 것뿐이 아니고 萬能制御프로그램의 實現에 可望性을 주는 것이다. 또한 現在의 쏘프트웨어의 많은 部分을 하아드웨어의 分野로 置換함으로서 具體的으로는 固定記憶裝置內部에 프로그램·썹루—티—느을 넣어둠으로서 쏘프트웨어의 生產性을 높이고 우리가 直面하고 있는 쏘프트웨어 人員의 不足을 緩和하는데에도 貢獻할 것이다. 前項에서 서술한 하아드웨어와 쏘프트웨어 코스트의 不均衡도 이로서 放善될 것으로 期待할 수 있다.

現在 하아드웨어, 쏘프트웨어 및 펌웨어의 生產에 要하는 人力을 比較해보면 이들 比率은 100對 100對 25程度이지만 第四代가 되면 이것이 100對 40對 60程度의 比率이 될 것으로 오포러—氏는 보고 있다.

4. 씨스템 構成의 動向

電子計算機의 씨스템 構成은 하아드웨어 技術과 쏘프트웨어 技術을 綜合하여 코스트·퍼ー프멘스를 最善의 것으로 만들기 위해 設計되어 있다. 따라서 이 構成은 電子計算機의 發達과 아울러 그 最適設計의 內容이 變해가는 것은 당연하다.

오늘날까지 電子計算機技術의 發達에 依해서 論理素子의 動作速度는 10年에 10倍 向上되었고 코스트는 5年마다에 10倍 다시 말해 10年에 100倍 싸게 되었다. 퍼ー프멘스 (이는 一秒間에 處理할 수 있는 프로그램 命令數로로서 表示되는데)는 5年마다에 10倍, 즉 10年마다 100倍씩 向上하였다. 이러한 進步의 傾向이 아직도 繼續되리라는 것은 아무도 疑心할 바가 아니다. 以下 現在의 씨스템構成과 將來의 씨스템構成에 關해서 考察하기로 하자.

4.1 現在의 씨스템構成의 特性

(1) 互技性을 갖는 씨스템構成으로 한것.

第三世代의 電子計算機와 第二世代의 그것과 比較해서 다른 가장 큰 差異點은 第三世代의 電子計算機는 그들 相互間의 互換性에 重點을 둔 點일 것이다. 다시 말해 One man machine system family series system의 實現이다. 그러나 하아드웨어 쏘프트웨어의 標準化가 아직 實施段階에 到達하지 않고 있는 것이 많기 때문에 메이커어가 다른 機種사이의 直接的인 互換性은 아직 바라지 못하고 있다.

(2) 多重處理技術이 進步하고 있다는 點

多重處理技術의 開發이 進捗되고 同時 多數프로그램並行處理가 어느정도 可能하게 되었다. 이 技術은 今後 더욱 높은 레벨까지 向上할 것이다.

多重處理를 爲해서는 멀티·프로셋써가 使用되고 이들이 相互間에 結合되게 되는 것이지만 前에는 마스터·슬레이브·씨스템이라 하여 主프로셋써와 從屬 프로셋써로 나눠서 일을 分擔하는 씨스템이 었는데 現在는 프로셋써를 主從으로 區別함이 없이 平等하게 取扱하여 어느 프로셋써가 故障을 일으켜도 그分量의 處理能力만이 상실되고 또 하나의 프로셋써는 運轉을 繼續할 수 있는 씨스템 即 Faile soft system이 많이 採用되게 되었다.

(3) 大規模化 大形化의 경향으로 움직이고 있는 點

情報處理量·프로그램의 스텝數의 增加等으로 因해 電計計算機의 大規模化, 大形化의 傾向이 보인다. 이는 處理코스트를 내리는 데에도 必要하다. 特히 記憶容量의 大形化가 뚜렷하다.

(4) 信賴性이 向上하고 있는 點

電子計算機의 動作의 確定性·信賴性은 씨스템을 構成하는 各要素의 發達·特히 IC의 使用으로 因해 向上하고 있다. 이는 電子計算機의 本體 뿐만이 아니고 入出入裝置를 包含한 하아드웨어씨스템의 信賴性의 向上에 努力이 기울려지고 쏘품트웨어 씨스템을 포함한 情報處理씨스템 全體의 信賴性을 높이기 위하여 故障에 對한 自動診斷 自動修復用 쏘프트웨어 하아드웨어를 設置하는 方式등의 開發이 推進되고 있다. 이미 管理프로그램에 依한 씨스템 診斷은 行하여 지고 있다.

(5) 高速化가 繼續추진되고 있는 點

電子計算機의 情報處理機能의 高速化는 그 演算部·制御部·記憶部等 電子計算機本體의 高速化外에 씨스템構成上의 配慮에 依해서 高速化할 수 있다. 例를 들어 演算에 使用되는 命令이나 데이타의 出入을 容易하게 하기위해서 래지스터를 豊富하게 準備해주거나 加減算專用, 乘算專用, 除算專用의 演等專用의 演算래지스터를 設定함으로서 加一層의 高速化가 이루어지기 때문에 이러한 種類의 씨스템 構成이 생각된다.

4.2 處理方式으로 본 씨스템構成의 例

情報處理方式으로 본 씨스템構成의 例로서는

① Batch system
② Remote Batch system
③ On-line real time ststem
④ Time sharing system等이 있다.

最近 온·라인 리얼타임 씨스템의 需要가 增

加하고 있고 이미 實施中인 씨스템도 많아졌다고는 하지만 情報處理의 主流는 Batch system로서 美國에서도 現在情報處理의 90퍼—센트 以上이 Batch sʎstem로서 行하여지고 있다. Remote Batch system은 通信回線으로서 데이타를 보내어 이를 電子計算機로서 一括處理하는 方式으로서 Batch system의 高處理能率과 온·라인·리얼타임 씨스템의 便利性을 兼備한 씨스템이다.

온·라인·리얼·타임씨스템과 타임·쉬어링 씨스템의 共通性은 온·라인 (File 및 端末裝置를 包含해서)인것·實時間性을 갖는 點 時分割技術을 利用하고 있는 것 等이며 그 相違點은 前者는 時定業務를 對象으로 하고 後者는 汎用 프로그램으로서 多數의 業務處理가 可能한 點일 것이다·

타임·쉬어링에 關해서는 이미 記述한바와 같이 아직 技術的으로는 幼年期에 있고 成年期에 들어가기 까지에는 이제부터 相當한 時日이 必要하다고 생각된다. 問題點은 타임·쉬어링 씨스템의 標準 씨스템이란 무엇인가가 아직 알려져 있지 않고 있다는 點이다. 또한 이 씨스템은 將次 特別業務를 對象으로한 온·라인 씨스템을 吸收할 수 있느냐 없느냐 하는 點이다. 理想的으로는 兩者를 하나로한 汎用 씨스템의 誕生이 所望되는 것이다. 그러나 그것은 아마도 아직은 바랄 수 없을 것이다. 즉 온·라인·리얼 타임·씨스템의 多樣化가 將來의 方向이고 타임·쉬어링 씨스템은 그 範疇內의 하나의 씨스템으로서 생각해야 한다는 것이 現時의 常識인듯 하다

타임·쉬어링·씨스템의 쏘프트웨어의 複雜性과 製作의 困難性 때문에 大規模 타임·쉬어링 씨스템의 制御 프로그램의 開發方法이 아직 確立되지 않았다고 한다. 例를 들어 制御 프로그래밍 言語가 아직 決定되어 있지 않다.

以上과 같이 지금으로서는 타임·쉬어링·씨스템이 여러 問題를 짊어지고 苦難의 길을 걷고 있다고는 하지만 今後 많은 經驗을 쌓서 또한편 하아드웨어의 進步와 쏘프트웨어의 發達, 例를 들어 펌웨어의 活用等에 依해서 將次에는 理想的인 타임·쉬어링·씨스템이 出現하게 될 것이다.

4.3 將來의 씨스템 構成

將來의 電子計算機의 씨스템構成은 LSI와 磁性薄膜 메모리 等의 利用에 依해서 現在의 씨스템構成과 比較해서 劃期的인 것이 될것이다. 그것은 從前의 씨스템 設計에 구애되지 않고 完全히 새로운 思想으로서 그 構成과 맞설수 있게 해줄 몇개의 要素가 나타났기 때문이다.

例를 들어 지금까지는 電子計算機의 씨스템設計에 있어 論理回路를 얼마나 使用하느냐가 코스트 관계때문에 큰 問題였다. 今後 LSI 製作技術이 發達하면 이것은 큰 問題가 아니게 된다 왜냐하면 하나의 基板上에 組立하는 論理回路數를 2倍 3倍로 증가시켜도 LSI의 코스트, 體積 및 消費電力은 조금밖에 안올라가기 때문이며 論理回路로서 만들어지는 게이트를 豊足하게 使用한 씨스템設計가 可能하게 되기 때문이다. 또한 信賴度의 確保는 씨스템 設計者에게는 最大의 難問題라고 해도 좋으리 만큼 重要한 것이었는데 이것도 LSI를 쓰면 物理的으로 信賴性이 올라가는 것 뿐이 아니고 故障의 自動診斷, 自動修復等의 씨스템 設計도 考慮할 수 있는 전망이 섰다. 以下 이들 問題에 關聯해서 詳細히 叙述하기로 한다.

(1) 씨스템構成에 依한 LSI의 役割

LSI는 電子計算機의 高速化·小形化·코스트의 低下, 低電力化 및 信賴度의 向上等에 貢獻하리라 함은 이미 서술한 바이다. 다시 이에 關해서 敷衍하자면 LSI는 씨스템 構成上 다음과 같은 일에 所用된다.

① 低廉한 스크랫츠·팸·메모리나 連想 메모리의 權成

② 汎用 씨스템에 적합한 特殊論理回路의 作成

③小容量의 高速 밧퍼·메모리(二個의 裝置사이에서 動作步調가 다를 때 그 中間에 設置하여 兩者의 速度, 時間등의 調整을 하고 兩者를 獨立的으로 動作시키는데 必要한 記憶裝置)의 作成

④ 프로그램을 하아드웨어에 組立시켜버리는 것 즉 펌웨어의 作成

⑤ 故障診斷回路의 作成과 切換用 豫備素子의 提供

⑥ 超小形의 電子計算機의 製作

이것은 自家用의 電子計算機로서 使用하는 것뿐이 아니고 remote Batch用이나 타임·쉬어링用의 端末機器로서 利用될 수가 있다.

(2) Module에 依한 씨스템構成

將來의 電子計算機의 씨스템은 機能 모듈의 連結에 依하여 構成되는 것이 理想이다. 프로셋써어, 記憶裝置, 入出力裝置 等의 機能을 制御하는 모듈이 생각될 수 있다. 씨스템의 擴大(縮小)나 交換도 簡單하게 이들 모듈의 操作으로서 行해진다. 機能 모듈은 LSI基板上에서 組立할 수 있다.

(3) 프로그램·제네레이터어의 活用

效率이 좋은 프로그램·제네레이터어를 多數 使用하여 씨스템 構成을 能率化할 수 있다. 이 제네레이터어는 프로그램의 뼈대와 그를 作成하기 위한 諸條件을 파라미터어로서 주기만 하면 目的 프로그램 루우틴을 만들어 주는 프로그램을 말한다. 이러한 種類의 제네레이터어는 이미 作成되어 利用되고 있다.

(4) 安全性 確保의 新 씨스템

現在의 電子計算機도 하아드웨어의 오차는 診斷 루—틴으로서 檢出되어 오차處理 루—틴에 依해서 修理시키는 機能을 갖이고 있다. 그러나 아직 그 機能程度는 初步의 段階에 있다.

將來 씨스템에서는 지금보다 더욱 高度한 自動診斷, 自動修復機能을 갖인 것이 設計될 것이다. 즉 이 診斷機能은 오차나 故障을 檢出하여 故障場所를 發見하고 그를 分離시키고 豫備部分에 切換하는 等 오차·故障을 制御하여 自動保全을 可能케 하는 것이다. 또한 이를 위해서는 오차 檢出의 論理回路의 作成, 오차 制御 프로그램의 作成, 自動修復用論理回路와 그 目的을 위한 프로그램의 作成, 故障의 경우의 切換用豫備部品의 準備, 故障狀態를 規定한 細目의 作成이 必要하게 된다.

이러한 目的을 爲해서는 將來에는 LSI를 마음것 써서 Built in coutrol用 셉루우틴을 多數 作成하게 되리라 생각된다. 電子計算機에는 從前에도 計算處理의 正確性을 保存하기 위해서 機械部에 있어서 Data의 移轉오차, 計算오차 等을 防止하거나 오차가 發見되었을 때 即時로 對策을 講求하도록 하는 回路나 裝置에 配慮가 되어 있다. 이것을 Built in Control이라고 한다. 패리티 點檢도 그 하나의 例이다.

(5) 펌웨어의 씨스템構成에 주는 影響

現在 쏘프트웨어로서 取扱되어 있는 各種 어프리케이션·프로그램이 하아드웨어化 하는 즉 펌웨어가 됨으로서 오늘날의 씨스템의 모듈化의 槪念은 더욱더 細部의 構成까지 침투하여 펌웨어에 依한 갈아 꽂일수 있는 機能의 모듈化가 이루어질 것이다. 따라서 理論上 유우저의 適用業務에 適合한 어떠한 씨스템構成도 모듈에 依해서 合成할 수 있다. 이와 같이해서 씨스템構成의 多樣化가 進行된다면 高度의 씨스템 아날리스트, 씨스템·엔지니어 等의 需要가 커지는 代身 基本的인 어플리케이션·프로그램이 펌웨어 하기 때문에 從前의 어플리 케이션 程度의 프로그래머어의 必要性은 相對的으로 減少한다.

(6) 同時並行 處理씨스템의 高度化

電子計算機 씨스템의 使用技術의 發展은 그 Parallelism 즉 同時 並行處理 씨스템의 發展과 호흡을 같이 하고 있다. 同時 並行處理 씨스템 즉 multi programming system의 高度化한 것으로는 multi processor 方式을 들 수 있다.

이는 多數의 프로셋써어와 多數의 記憶裝置 및 多數의 入出力裝置의 유닐各各이 全部 相互間 結合되어 相異한 프로그램을 同時에 또한 獨立的으로 處理할 수 있는 씨스템이다. 이 方式은 씨스템의 增減이 各 유닐의 增減으로 實現할 수가 있다. 各 유닐은 負荷의 過重에 對해서 서로 助力할 수 있고 또한 그 中 하나가 故障이 되도 全部의 運轉이 멈추는 일이 없고 安全性을 올리는데에 도움이 된다. 今後 LSI의 採用等에 依하여 프로셋써어나 記憶裝置가 지금보다 훨씬 高速하게 할 수 있기 때문에 이 씨스템의 效用은 더욱더 클 것이다.

(7) 電子計算機와 通信網을 連結한 씨스템의 最適化

今後 온·라인·씨스템의 發展은 활목할 것이 될 것은 분명하지만 通信網과 電子計算機를 結合한 總合씨스템을 어떻게 設計하면 最適이 되는가에 關해서는 未知의 點이 많고 차후의 硏究에 期待해야 할 것이 적지않다. 通信과 電子計算機의 發達된 技術을 統合하고 이들을 融合해야만 처음으로 最適의 씨스템 構成이 얻어질 것으로 생각되는 故로 將次 이 問題의 硏究가 추진 될 것이다.

(8) 씨스템構成의 自動設計

LSI의 構成을 設計하기 爲해서 電子計算機를 利用한다는 것은 이미 言及했으나 電子計算機의 機能, 信賴度 價格, 사이즈等을 綜合한 最適의 設計를 얻을려면 電子計算機를 利用하는 수 밖에 없다. 씨스템構成의 最適設計를 이루기 위해서는 電子計算機에 依한 自動設計의 開發이 必要하다.

5. 結 論

將來의 電子計算機技術의 發達은 프로그램作成의 번거러움으로 부터 우리를 解放하고 또한 電子計算機의 高度利用을 可能케 해줄 것이다. "電子計算機는 무엇을 할줄 아느냐?"가 아니고 "電子計算機로 하여금 무엇을 시킬 수 있는가?"가 今後의 課題이다. 모든 것은 우리人間들의 能力에 달려있다.

이 글은 將來의 電子計算機가 어떠한 方向으로 그 技術의 開發이 進行되는가에 關해서 叙述한 것이지만 그中 몇개 항목을 再記하고 이글을 끝맺고저 한다.

① IC, LSI, 薄膜等의 利用擴大로 電子計算機의 速度가 더욱더 커진다.

② 코스트·퍼—포먼스가 더욱더 좋아진다.

③ 安全性이 더욱더 커진다. 自動檢診, 自動修復을 指向한다.

④ 互換性, 標準化의 重要性이 높아지고 關係者의 協力이 强調된다.

⑤ 固定記憶裝置·마이크로 프로그램 方式이 高度化한다. 다시말해 펌웨어 開發이 進展되고 汎用 셈루우틴의, 펌웨어化가 發達한다.

⑥ 쓰프트·웨어의 팩게이즈化가 더욱더 많어진다.

⑦ 쓰프트·웨어의 모듈化가 진척된다.

⑧ 모듈에 依한 씨스템構成方式이 普遍化한다.

⑨ 멀티 프로그램, 멀티 프로세씨어 方式의 高度化가 이루어 진다.

⑩ 오퍼레이팅·씨스템이 汎用씨스템化를 지향한다.

⑪ 타임·쉐어링·씨스템이 改良되고, 進化한다.

⑫ Build-in 機能이 增大한다.

⑬ 메모리의 階層化가 繼續된다.

⑭ 값싼 大容量 메모리가 開發된다.

⑮ 音聲에 依한 入出力裝置·패턴 認識入出力裝置의 開發이 進行된다.

⑯ manomachine inter face改善의 努力이 繼續된다.

⑰ 電子計算機의 共同利用이 發展하는 한편 自家用電子計算機가 더욱더 增加한다.

⑱ 超小形電子計算機의 普及, 이를端末器로서 利用하는 傾向이 많어진다.

⑲通信電子計算機의 最適씨스템 構成의 實現을 指向해서의 檢討가 繼續된다.

⑳ 씨스템構成의 自動設計가 可能하게 된다.

(第2章 끝)