電子計算機를 利用한 自動 프로우 챠아팅

金 淇 龍*

Abstract:

A flowchart is a most useful tool for program debugging and documentation. But it is often a tedious and cumbersome job to draw a flowchart. Thus, automatic flowcharting is very useful. The features of this automatic flowcharting program are as followings.

First, instruction sequence is divided into blocks and groups. Each block stands for one processing, and groups are used for determining the location. The location of each group is determined one after another as the drawing of the flowchart advances. The flow line consists of one vertical line which takes one of the fixed positions and two horizontal lines which can take any position.

序 論

情報 處理의 順序를 表示하는 手段으로써 프로우 챠아트는 가장 간결하고 明確한 것이다. 이 프로우 챠아트의 目的하는 바는 코오딩을 하는 경우의 最終 資料, 디버그(debbug)의 資料, 또는 프로그램의 더큐멘테이숀(decumentation) 등이며, 프로그램 作成에 있어서는 不可缺의 것이다. 그런데 프로우 챠아트를 그린다는 것은 번잡하고 상당한 勞力이 필요하다. 特히 프로그램을 詳細히 表示하는 디테일 프로우 챠아트와 같이 프로그램에 直接 對應되는 것은 프로우 챠아트의 量이 많고 교정을 빈번히 하여야 함으로 복잡함과 勞力은 매우 크다. 따라서 이와 같은 프로우 챠아트가 프로그램의 리스트를 만드는 경우와 같이 용이하게 할 수 있다면 프로그램 作成에 소요되는 勞力을 크게 輕減할 수 있을 것이다. 이러한 理由로 10餘年前부터 自動 프로우 챠아팅에 관한 硏究가 있었다. 여기에는 프로그램의 解析을 主로 하는 것으로부터 실지 프로우 챠아트를 얻는 프로그램에 관한 것까지 있다. 또한 이 가운데에는 실지로 一般의 使用者의 利用이 되고 있는 것도 있다.

* 電子計算學科 敎授

本 論

1. 우선 쏘오스(source) 프로그램을 두개의 관점, 즉 각 命令의 역할 기능 및 콘트롤의 흐름이라는 두가지 觀點에서 解析하고 블록크와 그룹을 만든다. 블록크는 하나의 機能을 갖는 命令列을 종합한 것으로 하나의 프로우 챠아팅에 對應하는 것이다. 한편 그룹은 프로그램을 프로우 챠아트로서 圖面上에 配置하여 나갈 때의 單位가 되는 것이다. 配置方法은 遂次方法을 취한다. 이것은 하나의 그룹을 圖化할 때마다, 다음에 圖化하는 그룹과 그의 시작하는 位置를 決定해 가는 方法이며, 配置에 관한 메모리를 크게 節約할 수 있는 方法이다. 프로우 라인은 모든 그룹을 그린 다음, 그때까지 만들어진 分岐命令에 關한 데이터를 참조하며 그어 간다. 또한 프로우 라인은 任意의 位置를 취하는 橫線 두개와 미리 정해진 위치 중에서 선택하여 긋는 縱線 하나로 構成하고 프로그램의 簡單化를 企한다.

이 方法에서의 特徵은 알아보기 쉬운 프로우 챠아트를 간단한 方法으로 얻을 수 있다는 것이다. 이것을 위한 出力機器는 原則으로써 XY 프롯터(plotter)를 사용한다. 그런데 從來의 方法에서는 各 블록크의 配置를 미리 定해서 出力을 내는 것이므로 配置에 관한 데이터가 상당히 방대해지고 프로그램도 복잡해진다. 또한 프로우 챠아트 심볼의 크기를 變化하는 것도 용이한 것이 아니다. 이러한 方式은 라인 프린터와 같이 반대 방향으로 갈 수 없는 경우는 좋으나 XY 프롯더와 같이 반대 방향으로도 움직일 수 있는 出力機器를 사용하는 경우는 별로 적합한 方法이 못 된다. 이러한 理由로 앞서 말한 遂次方式을 취한다.

遂次方式에서는 하나의 블록크가 끝날 때마다 이것을 圖示하고 하나의 그룹을 圖化한 후에 다음에 圖化할 그룹과 그 位置를 定해 나가는 것이다. 따라서 配置에 관한 데이터가 적어도 충분하고 프로우 챠아트 심볼의 크기도 內部에 들어가는 文字의 크고 작은 것에 따라 간단히 변화할 수 있다. 그리고 라인 프린터를 出力機器로서 사용하는 경우는 이 遂次方式을 사용하여도 메모리 節約은 안된다.

이 프로그램의 대상이 되는 言語는 어셈불러로 한다. 어셈불러 言語는 機械에 左右되고 利用者가 限定된다는 缺點이 있으나 앞서 말한바와 같이 프로우 챠아트의 効果가 가장 크고, 블록크化 등의 問題로 보아 적합한 것이다. 그리고 콤파일러 言語를 對象으로 하는 경우는 프로그램 解析方法을 조금 變化시키므로서 基本的인 것은 그대로 利用된다.

2. 프로우 챠아트化의 方式

디지털 콤퓨터의 프로그램을 프로우 챠아트로 하는 경우 重要한 것은 處理內容을 적당한 圖形을 사용하여 알기 쉽게 表現하는 것, 一次元으로 配列되어 있는 프로그램을 2次元의 圖面으로 再配置하여 各 處理間의 關係를 보기 쉽도록 할 것, 그리고 여기에 프로그램 콘트롤의 흐름을 알기 쉽게 하는 것이다. 프로우 챠아트의 좋고 나쁜 것은 이상의 것을 어떻게 하는가

에 따라 決定된다고 하여도 過言이 아니다. 여기에서는 이러한 處理方法에 重點을 두고 프로우 챠아트化의 方式을 설명한다.

2·1. 블록크化

블록크는 프로그램 處理內容을 프로우 챠아트로 表示하는 경우의 單位가 되는 것인데 表示하는 內容은 프로우 챠아트의 레벨(디테일 프로우 챠아트, 제네럴 프로우 챠아트, 프로세스 프로우 챠아트)에 따라 달라진다.

어셈불러 言語의 프로그램을 프로우 챠아트로 하는 경우, 한 命令마다 圖示한다면 이것이 디테일 프로우 챠아트일지라도 너무 상세하게 되어 블록크化할 必要, 즉 몇개의 命令을 종합하여 블록크를 만들 必要가 있다. 命令列을 종합하는 方法으로서는 블록크가 하나의 處理를 表示함으로, 어느 하나의 機能을 갖는 命令列을 종합하는 것이 된다. 命令 自身은 그 命令이 目的하는 바는 明記하지 않으므로 命令列의 어디서 어디까지가 하나의 機能을 수행하고 있는지, 이것을 콤퓨터로 하여금 判別시키는 것은 一般的으로 困難하다. 여기서는 다음과 같은 原則을 定하고 이에 따라 블록크化 하기로 한다. 이는 하나의 機能을 수행하고 있다고 明白히 알 수 있는 것만을 종합하고 불확실한 것은 종합하여 表示하는 것으로 프로우 챠아트의 레벨로서는 디테일드 프로우 챠아트에 해당한다.

2·2. 블록크化의 原則

(1) 라벨이 붙어 있는 命令, 또는 여기에 分岐되어 오는 命令이 있을 때는 그 바로 앞의 命令까지를 一區分으로써 블록크化한다.

(2) 시프트 命令, 스테이터스·프립·후롭의 셋트 또는 그 상태의 스토어 命令, 콘트롤 命令 등은 1 命令 1 블록크로 한다.

(3) 演算命令은 累算器으로의 置數命令 (Load)에서 스토어 命令(Store)까지를 하나의 기능을 수행하고 있는 命令列로써 종합한다.

(4) 인덱스 레지스터로의 置數 또는 그 內容을 스토어하는 命令은 같은 命令이 계속되는 限 종합해 나간다.

(5) 條件付 分岐命令은 그 다음에 계속되는 分岐命令을 3方向으로 나누어지지 않는 限 종합해 나간다. 分岐條件의 종류가 다른 경우는 종합하지 않는다.

(6) 이상의 原則이 서로 모순되는 경우는 上位의 原則을 優先으로 한다.

以上의 原則에 따라 블로크化를 시행하는 方法中 演算命令이 블록크化를 中心으로 하는 部分을 그림 1에 表示한다. 그림 1에서 사용된 記號 가운데서 BRD는 브랜치 데이터의 略號다. 여기에는 各 分岐命令에 關한 데이터가 저장되어 있고 1 分岐 命令當 16語가 割當되어 있다. 그 構成은 그림 12~14에 表示하였다. 또 BLD는 블록크 데이터의 略號이며 여기에는 블록크化 될 경우 作成되는 데이터가 저장된다. 이러한 데이터에는 使用하는 프로우 챠아트 심볼의

종류, 크기, 기록 시작점, 現在의 펜 位置, 分岐의 條件, 심볼 內部에 넣는 文字 등이다. 또한 이 메모리는 各 블록크에 共通으로 사용되고, 심볼 내에 넣는 文字를 作成하기 위한 作業領域을 포함하여 約 300語이다.

이외에도 이 方式에서 特記할 것은 코멘트의 利用이 있다는 것이다. 즉 引數 또는 複數個의 出口가 있는 서브루우틴 呼出命令에서는 이것이 標準的인 形을 갖지 않고서는 그 다음에 계속되는 命令이 引數 또는 그 도중의 出口가 있는지 없는지 분간을 못한다. 이러한 경우 코멘트를 넣으므로서 이것이 引數 또는 出口라는 것을 表示하고 정확한 프로우 챠아트가 얻어지도록 되어 있다.

또한 分岐되는 곳이 實行前에 定해 있지 않은 分岐命令의 경우 코멘트가 있으면 規格에 준한 프로우 챠아트 심볼로 그려지고, 코멘트가 없으면 分岐되는 곳의 어드레스만을 表示하게 되어 있다.

2·3. 그룹化

디지털 콤퓨터의 프로그램은 그 性質上, 一次元으로 配列하여야 한다. 이를 그대로 圖示하면 그림 2와 같이 된다. 이것도 프로우 챠아트라고 할 수 있으나 圖面이 갖는 2次元的 性質을 利用하지 않고 있어서 보기 쉬운 것은 아니다.

이것을 2次元으로 바꾸어 본다. 그림 2를 보아도 알 수 있는데, 이 중에는 앞서 命令에서 콘트롤이 넘어오지 않는 命令이 있고 이것은 반드시 그 位置에 없어도 된다. 따라서 2次元으로 配置를 고치는 경우는 다음 命令에 콘트롤이 넘어가지 않는 命令으로 區分되는 一連의 命令을 하나로 추려서 적당히 配置를 바꾸면 된다. 이렇게 바꾼 것이 그림 3이다.

本 方式에서는 以上에서 表示한 命令列의 집합을 그룹이라 하고 配置 決定의 경우 단위로 한다. 이 그룹을 만드는데는 分岐命令에 주목하여 프로그램을 解折하는 것인데 그 알고리즘은 다음과 같은 것이다.

즉 無條件 分岐命令은 이것이 스킵命令의 直後에 있는 경우, 또는 서브루우틴의 途中 出口가 되는 경우를 除外하고 그룹을 구분하는 것으로 한다. 단 條件付 分岐命令은 그룹을 區分하지 않는다. 또한 條件付 分岐命令에서 그룹을 구분하지 않는 것은 프로그램과 프로우 챠아트의 對應을 쉽게 하기 위한 것이다. 以上의 알고리즘을 圖示하면 그림 4와 같다.

2·4. 配置方法

프로그램을 圖面에 配置하는데는 그룹을 單位로 한다는 것은 앞서 말한바와 같다. 各 그룹을 圖面上에 어떻게 配置하는가는 프로우 챠아트를 알기 쉽도록 整然하게 定하는데 重要한 要素의 하나다. 그러나 프로우 챠아트를 알아보기 쉽게 整然하게 그려놓는 것과 이것을 實現하는 프로그램의 단순화와는 一船的으로 一致하지 않는다. 여기서는 가능한 限 양쪽을 만족하도록 逐次方式과 같은 다음 方式을 취한다.

(1) 圖面은 그림 5에 표시한 바와 같이 4개의 領域으로 分割하고 쓰기 시작하는 位置는 左로 부터 두번째, 즉 領域 3으로부터 시작한다.

(2) 챠아트의 흐르는 方向은 下方向만으로 한다(단 프로우 라인은 윗 方向으로 간다).

(3) 圖示할 그룹과 시발 위치는 하나의 그룹을 그릴 때마다 逐次 定해간다.

(4) 다음에 그릴 그룹을 定하는 것은 그룹 데이터에 주어진 領域番號 또는 分岐番號에 따른다.

(5) 領域番號는 어떤 領域에 그릴 것인가 하는 것이 決定된 그룹에 대해서 주어지고 分岐領域番號는 그룹의 途中에서 分岐되는 그룹에 주어진다.

(6) 그리는 그룹을 선정할 때는 우선 領域番號가 주어진 그룹을 찾고, 없는 경우는 分岐領域番號를 받을 그룹을 선정한다.

(7) 分岐領域番號가 複數個의 그룹에 주어졌을 때는 그 중에서 가장 높은 位置에서 分岐된 것을 선정한다.

(8) 分岐領域番號는 주어져 있어도 左右 어느 領域에 分岐하는가는 決定되어 있지 않다. 그것을 決定하는데는 그 그룹 도중에서 다른 그룹으로 分岐하는지 아니하는지 이미 그려진 그룹과의 位置關係 및 바로 그 앞에 있는 條件付 分岐命令에서 分岐된 그룹은 어떤 쪽에 있는가 하는 것 등을 감안하여 定한다.

(9) 各 그룹을 그리기 시작하는 位置는 이것이 이미 그려진 그룹과 중첩되지만 않는다면 그 옆 領域에 있는 그룹과 같은 높이로 한다. 複數個의 位置로 부터 分岐하여 올 때는 이미 그려진 것과 중첩되지 않는 것 중, 가장 높은 위치에서 分岐되어 오는 것의 位置로 한다. 각각 중첩이 될 것 같으면 중첩되지 않는 位置까지 내린다.

위의 方法에 따라 各 그룹의 配置를 決定하는 알고리즘을 그림 6에 표시한다. 이것을 說明하면 대략 다음과 같다. 우선 最初로 그리는 것은 프로그램의 선두에 있는 그룹으로서 그리기 시작 점은 領域 3이다. 이 그룹을 그리고 나면 다음에 그리는 그룹과 그리기 시작하는 位置를 定하기 위한 處理를 한다.

아직 그리지 않은 그룹 속에 現在의 領域番號의 레지스터(DMN라고 略함)의 領域番號를 갖는 그룹이 있는지 없는지 조사하고 그러한 그룹이 있으면 이것을 이어서 그려간다. 그리고 이것과 같은 處理를 領域番號가 주어진 그룹이 없어질 때까지 계속한다. 이것이 끝나면 現在分岐領域番號의 레지스터(BDN라고 略함)의 값을 한개 더해가서 그 分岐領域番號를 갖는 그룹 즉 이웃 領域에 分岐하는 그룹을 찾아 낸다. 해당하는 그룹이 두개 以上 있는 경우는 (7)에 表示한 方法에 따라 하나의 그룹을 찾아내서 그것을 (8), (9)에 表示한 方法에 따라 그린다. 이 그룹의 그림이 끝나면 앞서와 똑같이 現在의 DMN 領域番號가 주어진 그룹을 찾아내서 그러한 그룹이 있으면 계속 그려나가고 없으면 現在 BDN의 分岐領域番號를 주어진

그룹 속에서부터 (7)에 表示한 方法으로 그룹을 하나 選出하여 (8), (9)에서 表示한 方法으로 圖化한다.

이렇게 한 다음 現在의 BDN의 分岐領域番號를 주어진 그룹이 없어지면 다시 BDN을 하나 進展시켜 그 分岐番號를 주어진 그룹의 圖化를 시작한다. 以上의 處理를 반복한 다음 콘트롤의 흐름이 直接 연결된 그룹을 전부 그린 다음에는 나머지 그룹을 다시 위에서 말한 方法에 따라 圖化해 나간다. 이것을 반복하면 모든 그룹이 圖化된다.

2•5. 프로우 라인

프로우 라인을 그는 方法은 配置方法과 같이 프로우 챠아트를 이해하기 쉽게 하는데 必要한 重要한 要素의 하나다. 여기에서 取한 方法은 프로그램을 간단히 하는데 중점을 두고 利用하는 데이터의 形을 생각하여 다음과 같이 한다.

(1) 프로우 라인은 모든 그룹을 圖化한 다음에, 그때까지 만들어진 브랜치 데이터에 따라 그어 나간다.

(2) 各 領域의 양쪽에 그림 7과 表示한 세개의 프로우 라인을 그릴 수 있는 場所를 確保해 두고 프로우 라인을 그을 때는 그중 하나를 선택하여 준다. 따라서 프로우 라인은 세로 한개와 가로 두개의 선으로 구성하게 된다. 이 프로우 라인을 그는 장소에는 6~31의 번호가 주어지고 프로우 라인의 종선을 긋는 장소의 지정은 그 번호 즉 라인 번호를 사용하며 실행한다.

(3) 프로우 라인을 긋는 것은 分岐되어 나가는 곳이 同一領域 또는 이웃 領域內에 있는 경우로 한다. 더 먼곳으로 分岐하는 경우는 콘넥터로 結合시킨다.

(4) 라인 번호를 주는 순서는 이웃 領域으로 分岐하는 것을 우선적으로 하고 同一領域內에 分岐하는 것을 후로 한다. 또한 그 중에서는 그룹이 적은 것을 우선적으로 한다.

(5) 라인 번호를 주는 方法은 分岐되어 가는 곳의 領域에 따라 다음 두가지 경우가 있다.

(가) 이웃 領域에 分岐되어 가는 곳이 있는 경우 두개의 領域으로 포위선 라인 番號 중에서 양단의 두개를 제외한 4개를 취하고 끝에서부터 차례로 이미 라인 번호를 지정해 준 프로우 라인과의 중첩여부를 조사하고 중첩되지 않으면 그 번호를 그 프로우 라인에 정해준다.

(나) 同一領域內에 分岐되어 가는 곳이 있는 경우 그 領域의 양쪽 및 바깥쪽 한개씩, 합계 8개를 취하고 左右 교대로 안쪽에서 차례로 그것이 이미 라인 번호를 주어진 프로우 라인과 중첩되지 않고 또한 같은 位置로부터 나오는 프로우 라인과 같은 方向으로 안될 것인가를 조사하고 그러한 라인番號가 있으면 그것을 그 프로우 라인에 지정한다.

또한 (가)(나)에서 최후까지 條件을 만족하는 번호가 없으면 콘넥터로 결합시킨다.

3. 自動 프로우 챠아팅 프로그램

다음에 이상에서 설명한 方式을 實現할 수 있는 프로그램에 대해서 설명한다. 이 自動 프로우

챠아팅 프로그램은, 쏘오스 프로그램을 3回 읽는 方式을 취한다. 그 구성은 그림 8에 表示하였다.

어셈불러 處理는 Pass 1, Pass 2에서 실행하고 프로우 챠아트를 얻기 위한 處理는 主로 Pass 2와 Pass 3으로 한다.

Pass 1은 그림 9에 표시한 바와 같다. 여기서는 우선 카아드에서 쏘오스 프로그램을 읽고 라벨이 있으면 이를 라벨 테이블에 登錄한다. 다음으로는 그 命令의 解讀을 하고 그 命令에 對應하는 命令番號를 주고 그것을 現在의 ILC(Instruction Location Caunter), 그 命令의 豫測길이(SL)의 값과 같이 쏘오스 프로그램 텍스트에 기입한다. 또한 解讀한 命令에 따라 ILC 및 SL의 값을 增加시킨다. 이러한 處理를 카아드 一枚 읽을 때마다 실행하고 END가 되면 Pass 1의 處理를 끝낸다.

Pass 2는 그림 10에 표시한 바와 같다. 여기서는 우선 쏘오스 프로그램 1 命令分을 읽고, 이것이 分岐命令인지 조사한다. 分岐命令이 아니면 그대로 다음 處理로 넘어간다. 分岐命令이면 그곳에서 그룹을 자를 수 있는지 조사하고 자를 수 있으면 그룹化를 하고 할 수 없으면 그대로 브랜치 데이터 作成에 들어가고 그 分岐命令의 어드레스 등 데이터를 기입한다. 그 사이에 分岐되어 가는 곳의 어드레스, 分岐의 條件, 이 命令의 브랜치 데이터의 어드레스 등을 쏘오스 프로그램 텍스트에 記入한다. 이렇게 기입하다가 END가 나오면 Pass 2의 處理를 끝낸다.

Pass 3은 그림 11에 表示되어 있다. 여기서는 우선 첫 그룹을 읽으면서 블록크化를 진행한다. 블록크는 하나가 完成될 때마다 圖化한다. 이렇게 하여 最初의 그룹을 圖化했으면 다음에 圖化하는 그룹과 시발점을 定하고 그 그룹을 圖化한다. 이것을 계속하다가 모든 그룹을 圖化하고 나면 다음은 프로우 라인을 긋는 處理를 한다.

프로우 라인을 긋는데 必要한 位置 데이터는 블록크化의 過程에서 브랜치 데이터에 기입되어 있어서 이것을 사용하여 다시 라인番號, 分岐方向을 定하고 프로우 라인(콘넥터에 의한 결합도 포함)을 긋는다. 프로우 라인이 전부 끝나면 Pass 3의 處理를 끝내고 프로우 챠아트의 作成이 끝난다.

結　論

以上에서 論述한 自動 프로우 챠아팅 프로그램을 사용하여 實際로 프로우 챠아트를 그리면 그림 15와 같다. 이외에도 여러 프로그램으로 시험하였으나 대략 期待한 바 結果를 얻을 수 있었다. 경우에 따라서는 세로 긴 프로우 챠아가 되는 경우도 있다.

이것은 프로그램 간단化를 위하여 取하던 逐次方式, 또는 프로그램의 對應을 좋게 하기 위하여 條件附 分岐命令으로 그룹을 나누지 않는 方法을 取했기 때문이다. 그리고 지금까지 問

題가 된바 있는 複數個의 出口가 있는 서브루우틴 呼出命令 또는 分岐되어 가는 곳이 實行中에 定해지는 命令 등의 블록크化는 코멘트를 利用하여 解決하였는데 實行中에 命令이 바뀌는 것에 대해서는 아직 해결되지 않고 있다.

참고로 이 自動 프로우 챠아팅 프로그램에 要하는 메모리는 쏘오스 프로그램 텍스트를 제외하고 24k語이다. 또한 50 스탭의 프로그램을 프로우 챠아트로 하는데 要하는 時間은 出力機器의 X-Y 레코더를 콤퓨터와 On Line 制御하면 7~10分은 걸린다. 이것을 Off Line으로 하면 콤퓨터 사용시간은 一分 정도가 되고 충분히 실용가치가 있다.

參 考 文 獻

1. Richard M. Karp: A Note on the Application of Graph Theory to Digital Computer progframming, Information and Control, Vol. 3, No. 2 (1960), pp. 179~190.
2. Edward A. Voorhees: Algebric Formulation of Flow diagrams, Communication of the Association for Computing Machinery, Vol. 1, No. 6 (1958), pp. 4~8.
3. Hee Krider: Flow Analysis Algorithm, Journal of the Association for Computing Machinery.
4. Lois M. Haibt: A Program to draw Multi level Flow chart, Proceedings of the western Joint Computer Conference, 1959, pp. 131~137.
5. IBM Application Program, system/360 Flow chart (360 A-SE-22 S) Usex's Manual.

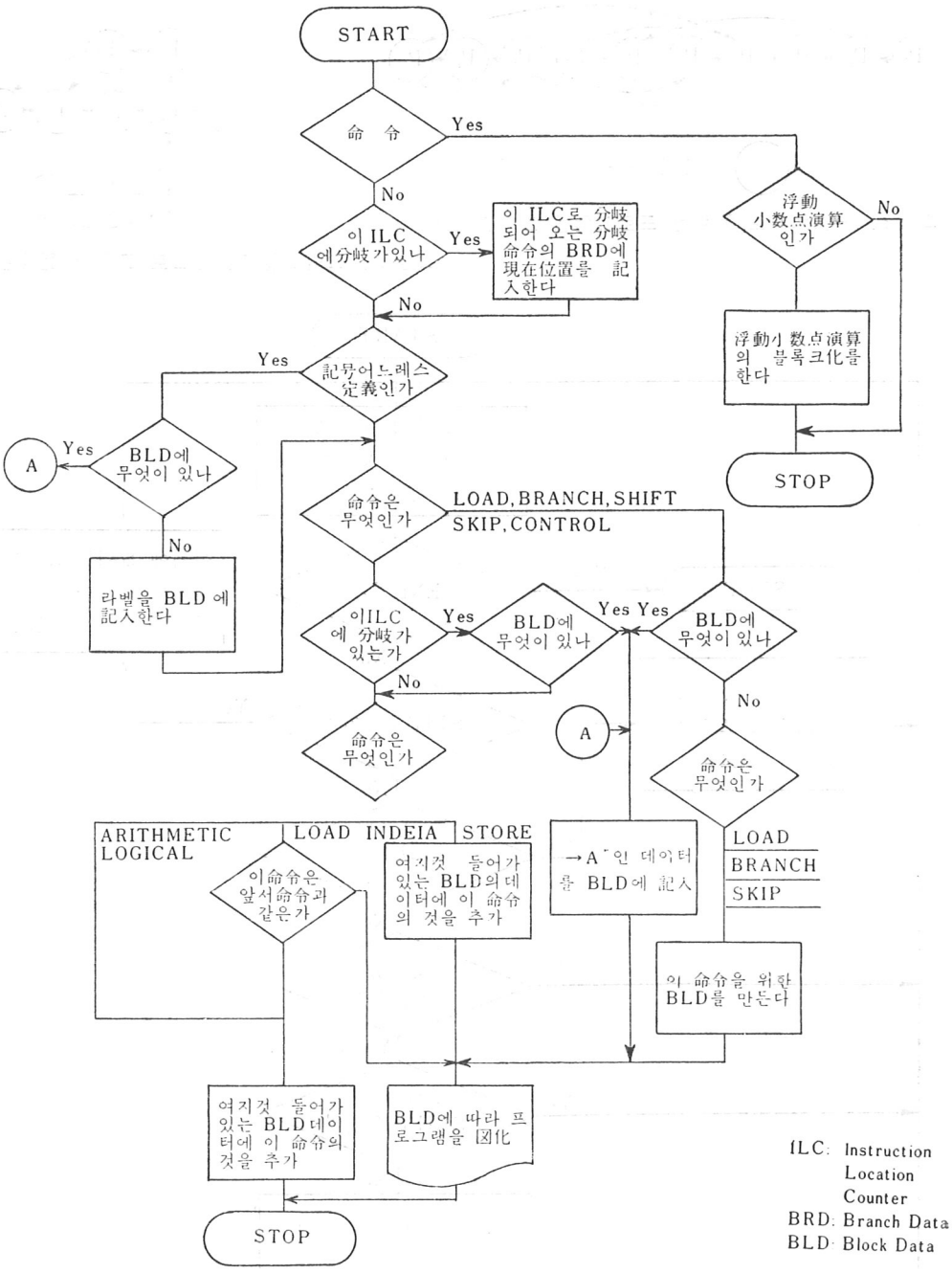

그림 1. 블록크化 (演算命令을 主로하는 部分)

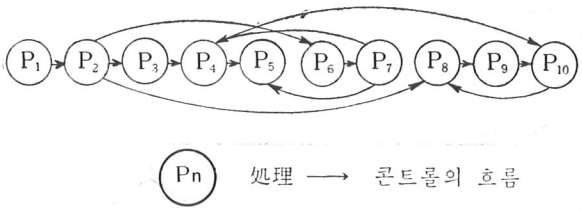

그림 2. 一次元으로 配列된 프로그램을 그대로 그린 것

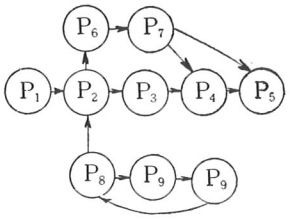

그림 3. 그림 2 를 2 次元으로 고쳐서 配置한 것

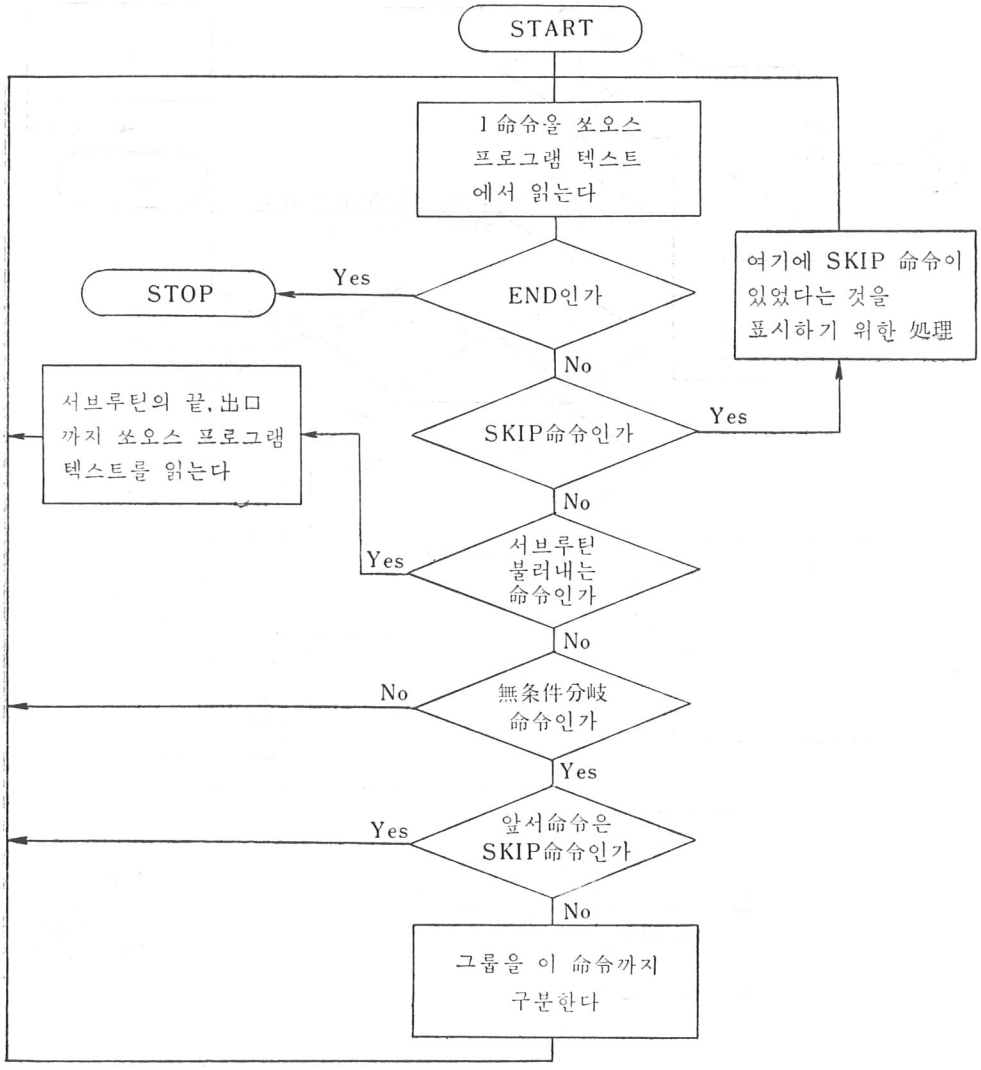

그림 4. 그룹化의 알고리즘

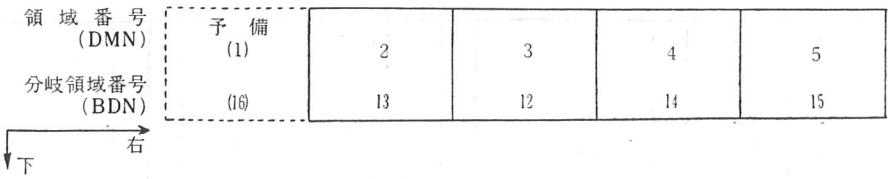

그림 5. 図面의 分割과 그의 領域番号 및 分岐領域番号

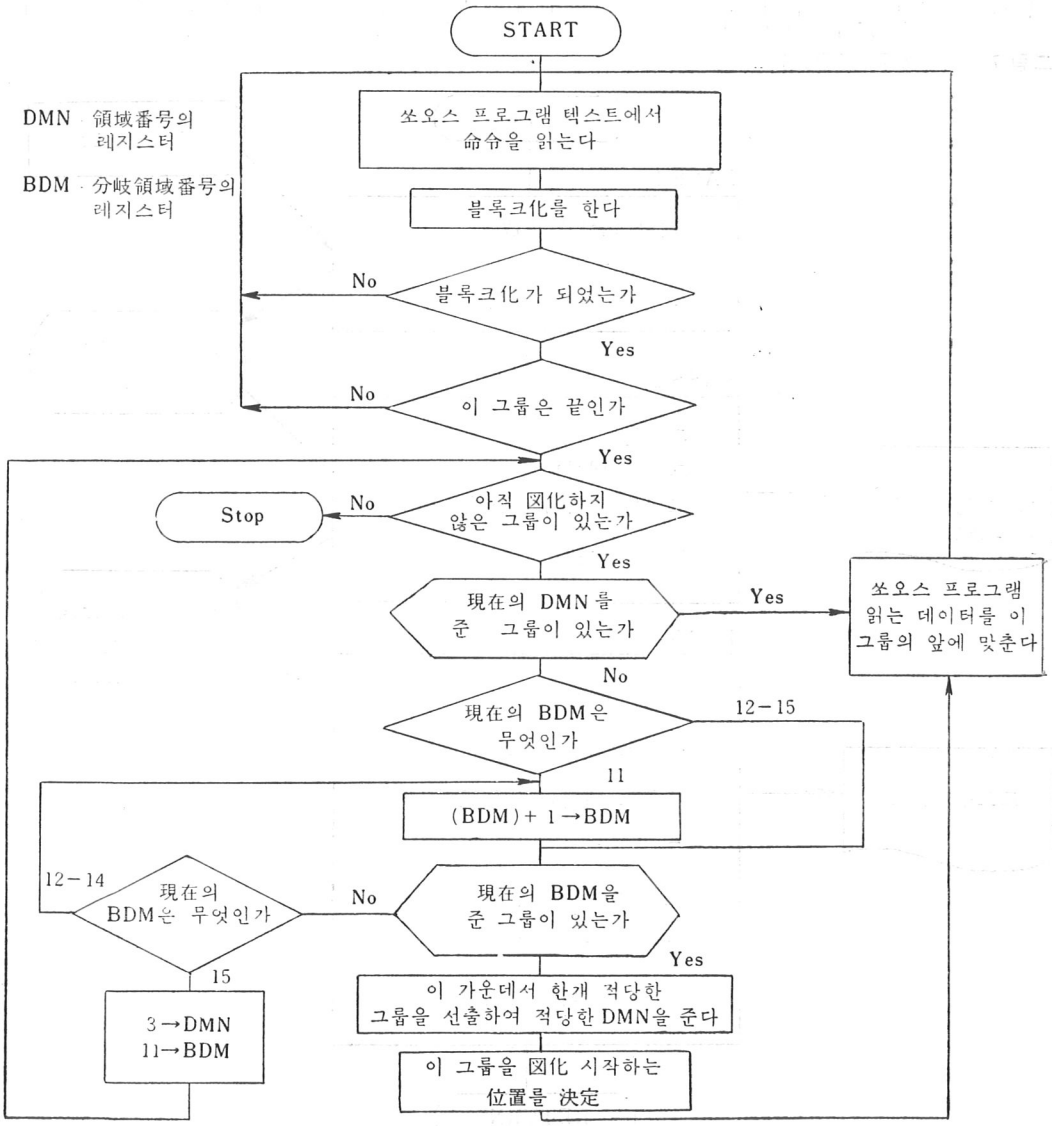

그림 6. 配置決定의 알고리즘

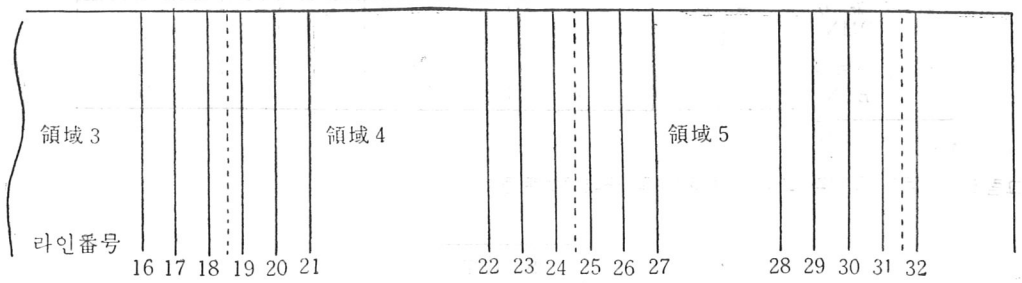

그림 7. 라인番号(部分図)

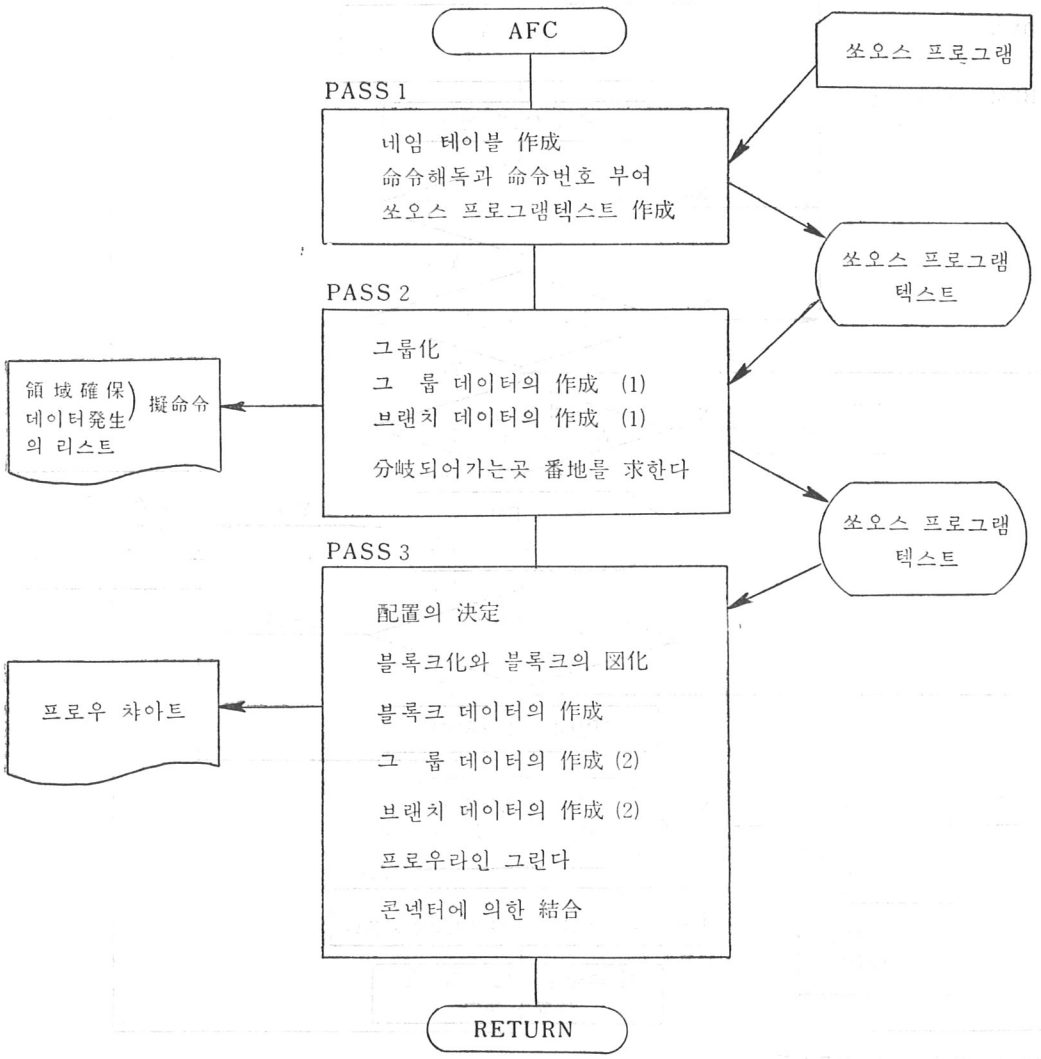

그림 8. 프로우 차아트 自動作成 프로그램

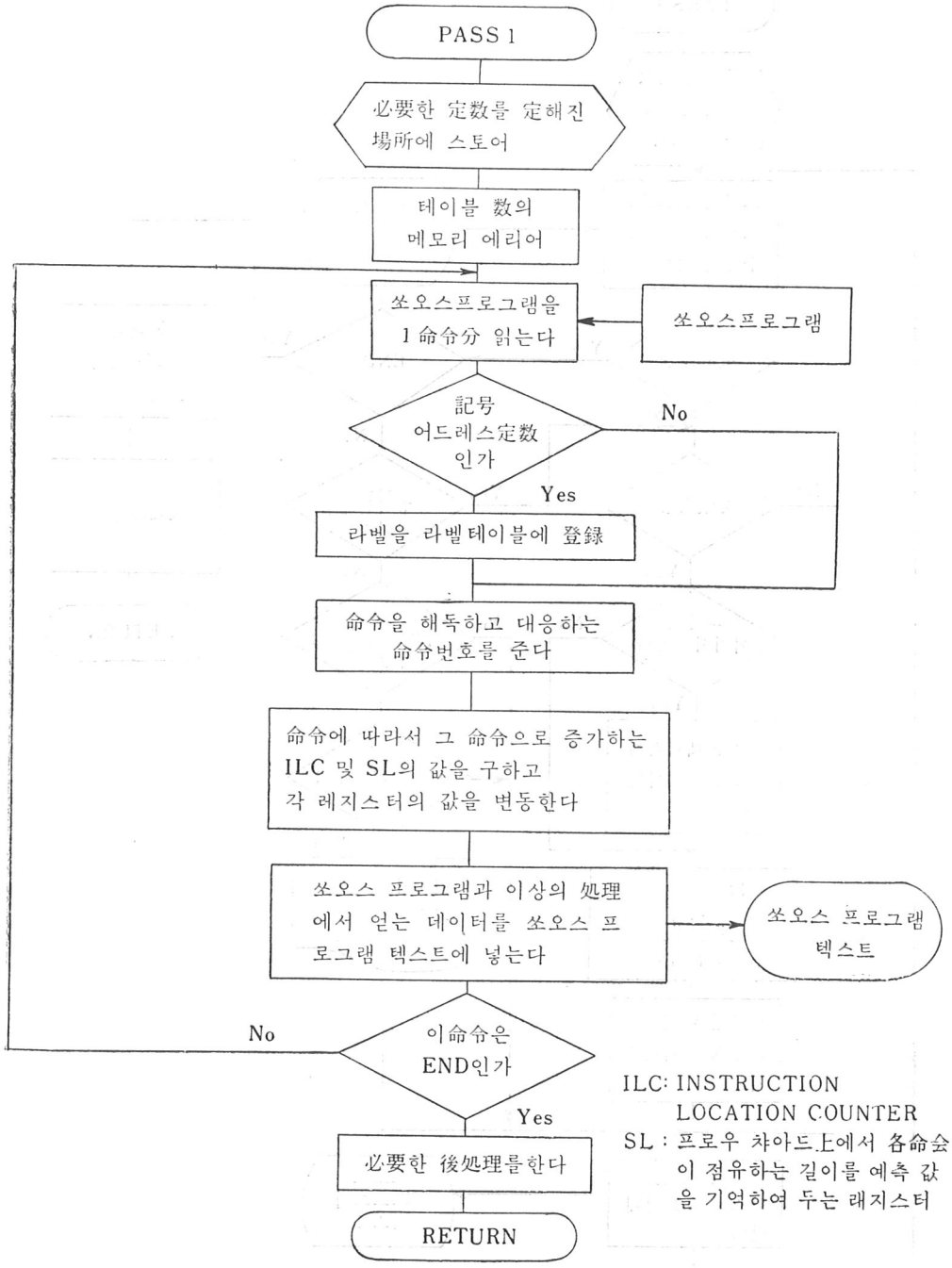

그림 9. PASS 1 의 프로그램

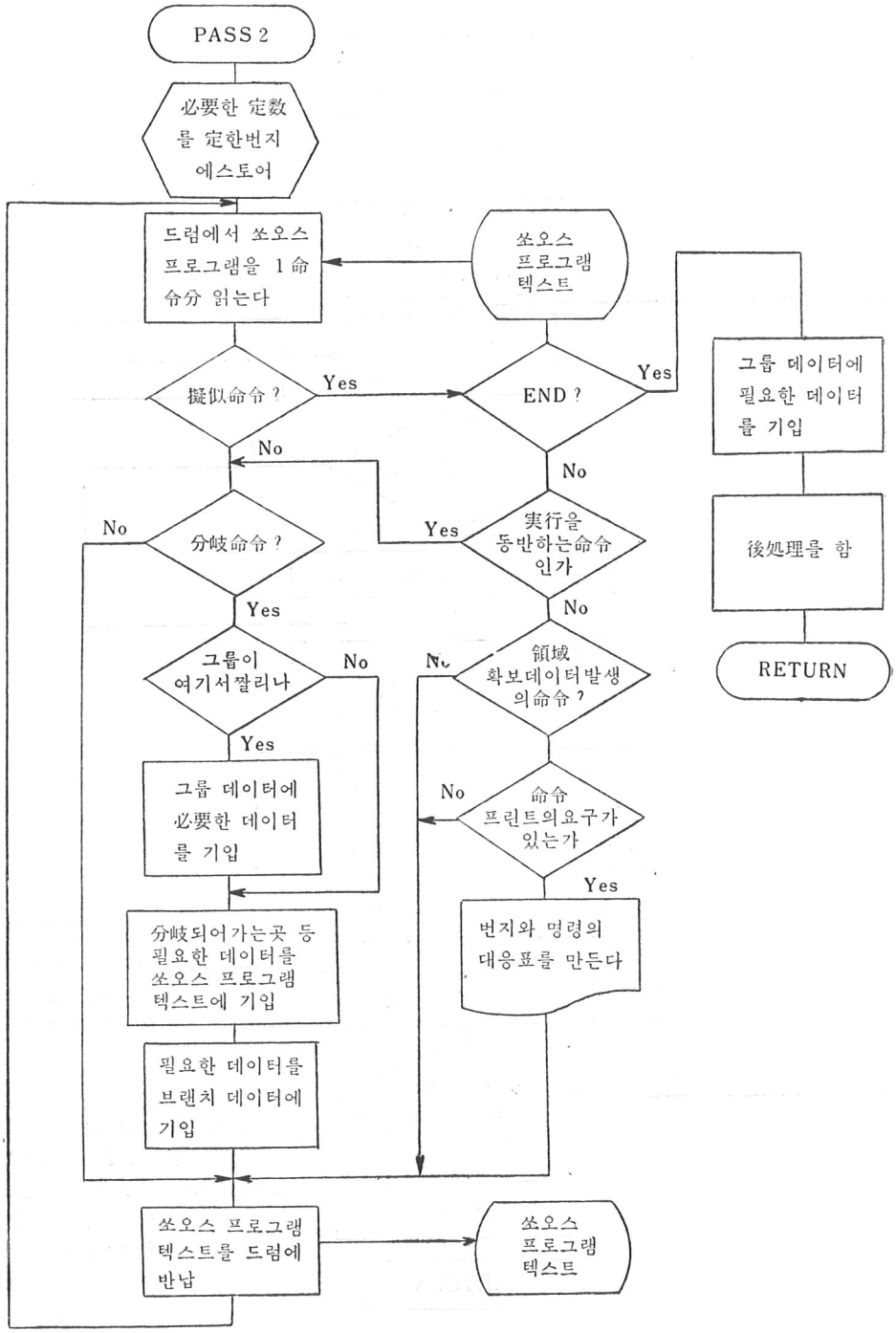

그림10. PASS 2의 프로그램

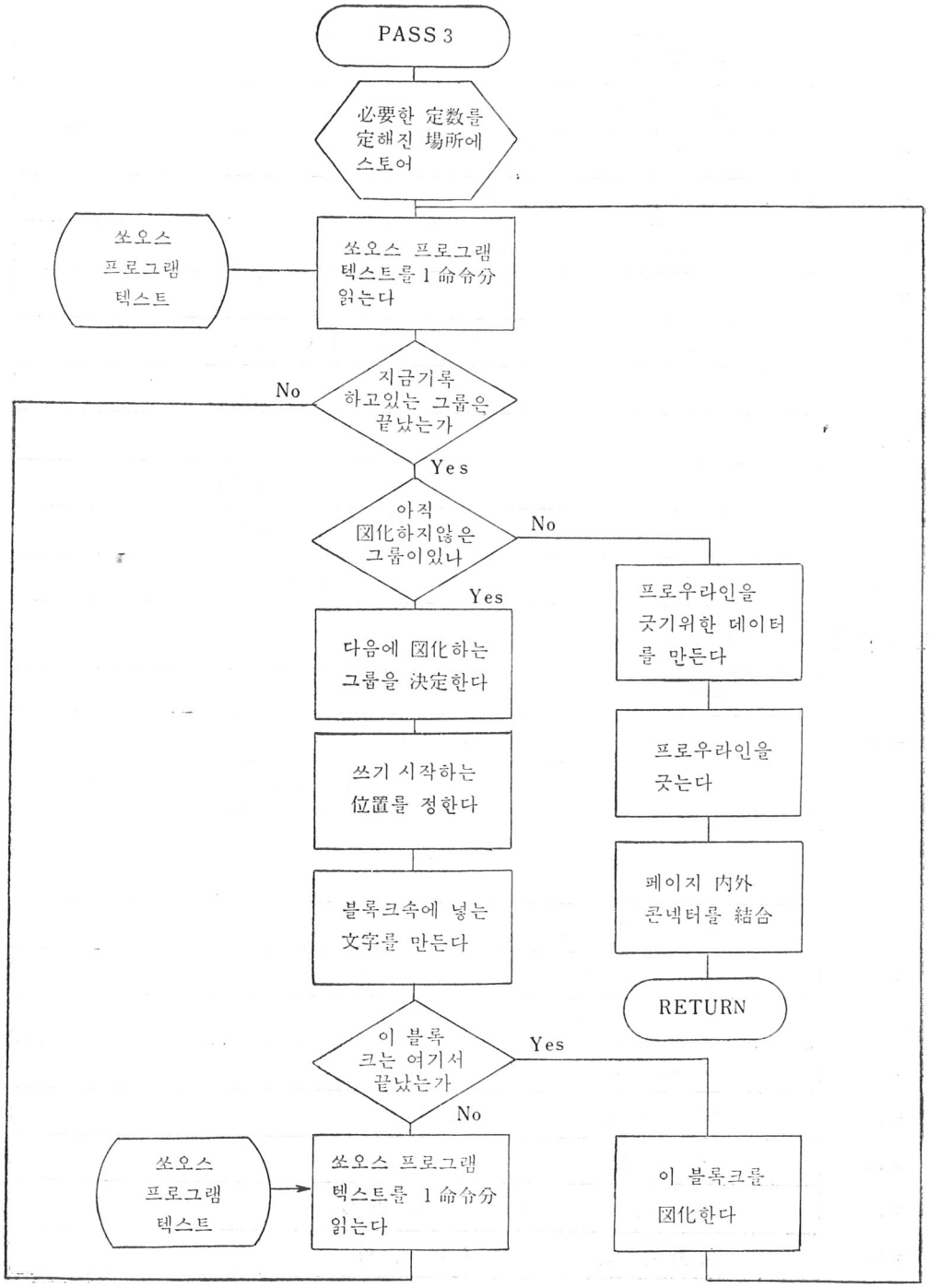

그림11. PASS 3의 프로그램

	그 룹 번 호
+ 1	領域番号/分岐領域番号
+ 2	이 그 룹 의 길 이 予 測 値
+ 3	이 그 룹 선 두 의 ILC
+ 4	이 그 룹 최 후 의 ILC
+ 5	予 備
+ 6	이 그룹의 선두에 分岐하여오는 命令의 ILC (No 1)
+ 7	〃 (No 2)
+ 8	〃 (No 3)
+ 9	〃 (No 4)
+10	〃 (No 5)
+11	이 그룹의 선두命令 쏘오스 프로그램 텍스트의 어드레스

그림12. 그룹 데이터 (GPD)

	브랜치 番号
+ 1	이 分岐命令이 소속하는 그룹의 번호
+ 2	分岐되어가는곳의 命令이 소속하는 그룹의 번호
+ 3	이 分岐命令의 ILC
+ 4	分岐되어가는 곳의 ILC
+ 5	그룹이 여기서 끝나는지 여부/分岐의 方向
+ 6	라인번호/페이지內콘넥터 번호/페이지 外 콘넥터번호
+ 7	分 岐 의 条 件
+ 8	分 岐 点 의 X 座 標
+10	分 岐 되 어 가 는 곳 의 X 座 標
+12	分 岐 点 의 領 域 番 号
+13	分 岐 되 어 가 는 곳 의 領 域 番 号
+14	프 로 우 라 인 의 縱 線 의 길 이

그림13. 브랜치 데이터 (BRD)

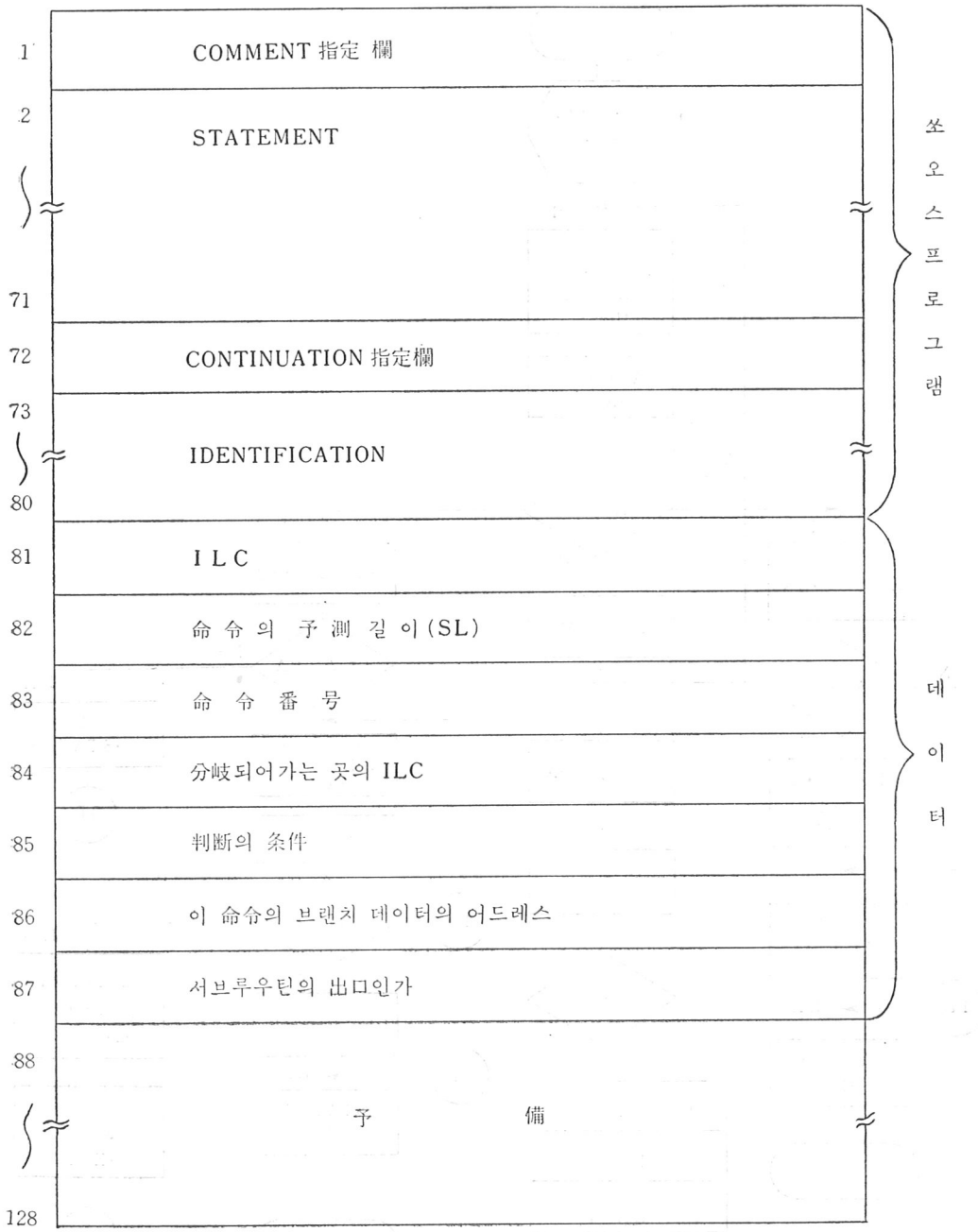

그림14. 쏘오스 프로그램 텍스트

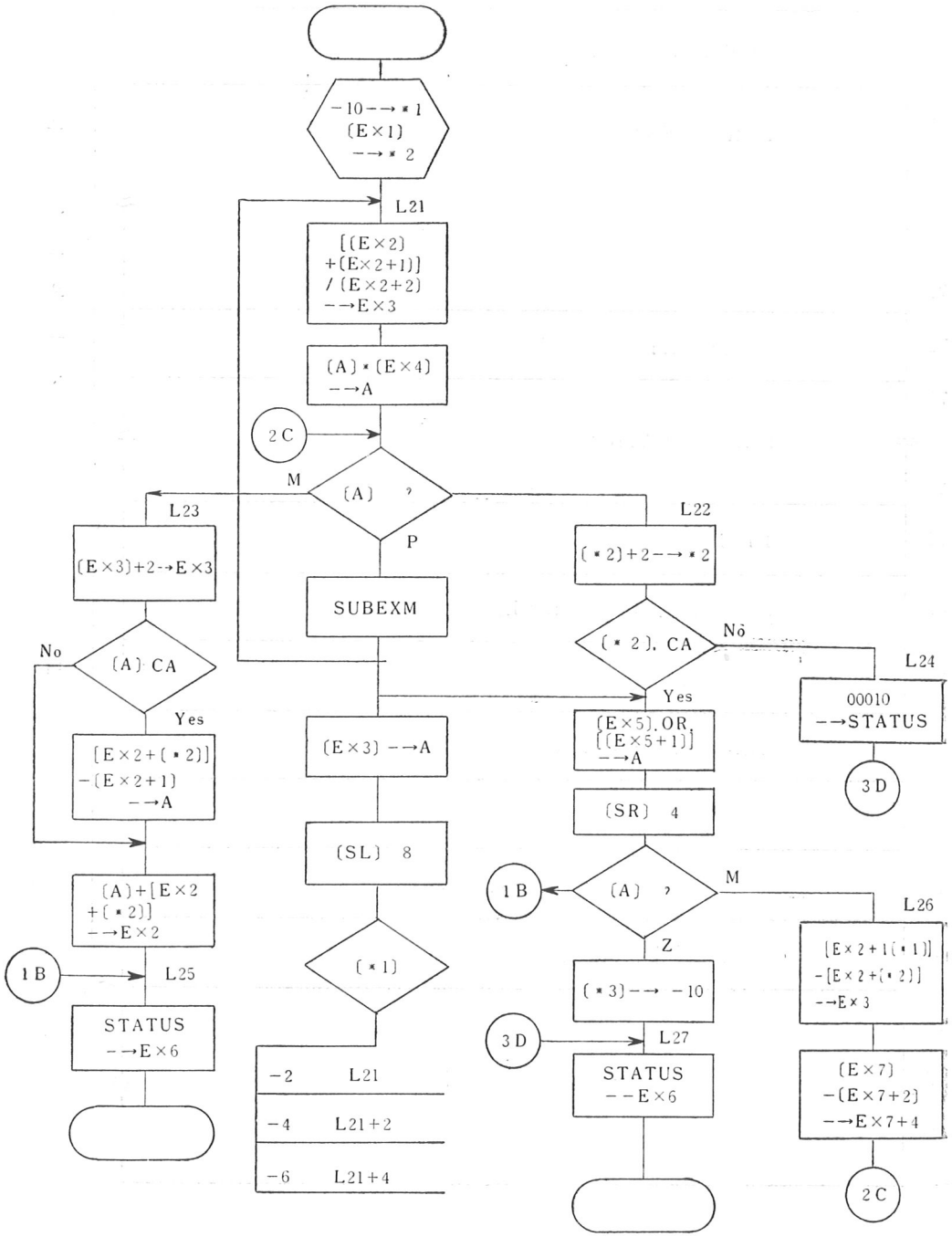

그림15. 自動 프로우 챠아팅 프로그래밍에 의하여 그려진 프로우 챠아트