教育的視点から考察したOperating Systemにおけるソフトウェアの最新開発手法 その1:分割・連結

渋井 二三男（城西短期大学）

プログラムはその設計上の都合などにより、幾つかのまとまりのある部分に分割してプログラム開発する場合が多々ある。これらの分割されたここの部分を別々にCompiler（翻訳）した後で、Jobの目的に応じて結合させることができる。この結合はNTT、KDDIに代表される第1種電気通信事業者、NEC、日立、東芝、沖電気・・・などの大手製造業者も含め、ユーティリティプログラムの一つによって行われるが一般的である。ここではユーティリティプログラムの代表例である分割・連結を教育的側面から説明する。

Keyword：ノットページ構造、セグメンテーション、リュウザブル、ライブラリ組み込み、エントリベクタ、トランスファベクタ、リエントラント

1. プログラムの構造

計算機システムを効率的に稼動するためには、プログラムが実行される時、充分な程度にプログラムを最適な大きさに設定することである。即ち、ユーザプログラムを実行する時、他のプログラムを機能別に細分化し、プログラム・メンテナンスが簡易化に出来るようなサブルーチンとして、そのプログラムを設計することである。その主たる方法は、セグメンテーション、オーバレイ、リンケージなどがある。

一般にコントロール・プログラムにロードモジュール（コントロール・プルグラムによって、プログラムとして主記憶にロード出来る形式のモジュールである）が与えられ、これが主記憶装置にロードされると新しいタスクが作られる。そのタスクを構成するプログラムの構造には、ノットページ構造、セグメント構造、動的構造などがある。

大きなプログラムを更に小さなプログラムに再分割し、一つのまとまった論理的な単位として主記憶装置に読み込むことの出来る最小単位がセグメント（Segment）である。又、セグメントに分けられたプログラムをセグメント・プログラムという。セグメント・プログラムには分割修正などが出来るので、プログラムの作成が便利になる。このような方法がセグメンテーション（Segmentation）である。

(1) ノットページ構造

ノットページ構造のプログラムは、プログラム全体が一度にローディングされ、実行されるものをいう(単純構造ともいう)。これはプログラムを実行しているとき、ほかのプログラムを呼び込む必要がない場合である。したがって結合後のノットページ構造のプログラムは、プログラム全体が一度にコアにローディングできる大きさでなければならず、それが仮にコアをオーバーするような大きさである場合は実行出来ない。このような場合にはセグメント構造のプログラムとしてプログラミングしなおす必要がある（図1)。

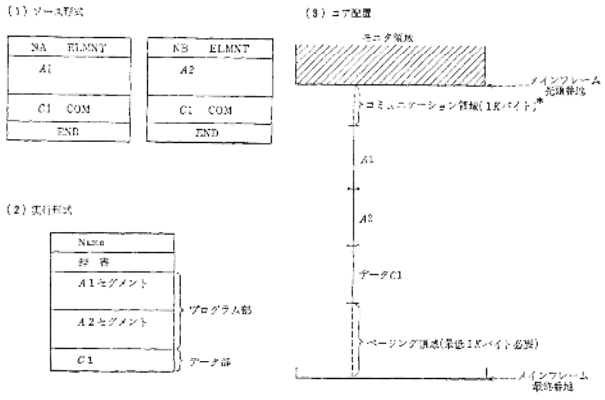

図１ノットページ構造

(2) セグメント構造

セグメント構造のプログラムは、一般的にはコアに常駐するセグメント(メイン・セグメント又はルート・セグメント)とコアに常駐せずオーバレイで動作するセグメントに分け、必要に応じて常駐セグメントからオーバレイ・セグメントを呼び出して実行させるというような構造(ツリー構造)をとる（図2)。

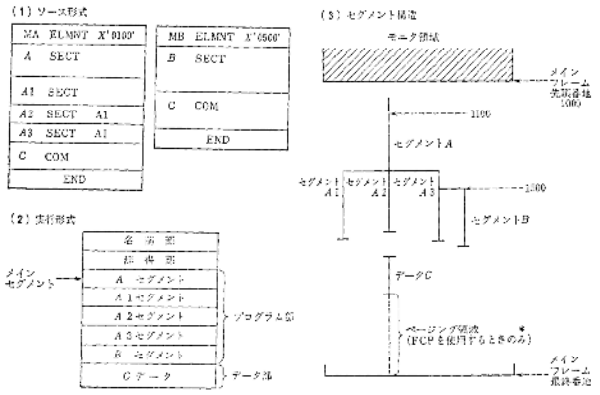

図 2 セグメント構造

ユーザプログラムが大きくて一度に主記憶に格納できない時、それをいくつかのブロック(セグメントフェイズ)に分割し、プログラムの進行に応じて必要なフェーズを補助記憶装置から呼び込みながら処理を進めていくのがオーバレイ構造である。オーバレイ構造にプログラムを進めるためには、ユーザがプログラムの中の制御の流れや各コントロール・セクションの関係を詳細に知り、同時に主記憶装置に存在する必要のない部分をモジュールに分割し、それらの各部分の関係をツリー構造に表現する必要がある。例えば図3の場合、A1はつねに主記憶装置上に存在しなければならないが、A2と、A3、A4及びA2とA3、A5は同時に主記憶装置上に存在する必要はない。又、A4もA5も同時に主記憶装置上に存在する必要はない。

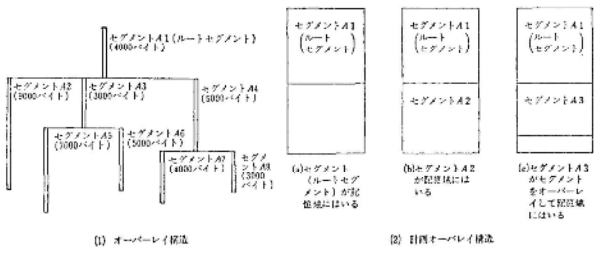

図 3 オーバレイ構造

(3) 動的構造

あるジョブを行うための一のプログラムの大きさが設置されている主記憶装置の容量を越えることがある。又、マルチプログラミングの場合、一つのプログラムに割り当てられる主記憶装置の大きさには制限がある。プログラムの大きさがこの制限を越えることが多い。このようなときにはオーバレイを行うか、ダイナミックリンケージ行わなければならないが、このような工夫を実施できるようにプログラムの構造がなっている時、それぞれを計画オーバレイ構造及びダイナミックリンケージ構造という。ダイナミックリンケージ構造のことをダイナミック構造と略すこともある。又、計画オーバレイ構造はよくオーバレイ構造と略される。これらのような複雑な構造でないものをシンプル構造、そしてこれらの構造を総称してプログラム構造という。

2. プログラムの性質

ライブラリなどに組み込むプログラムは、一般に、ノンリューザブル、シリアリ・リューザブル、リエントラントなどいずれかの性質をもたせることが出来る。

(1) ノン・リューザブル

これは一度にプログラムを実行した後、プログラムの状態が変更されたままで終了するようなプログラムの機能である。同じ仕事をもう一度行いたい時は、再ロードを行わなければならない。

(2) シリアリ・リューザブル

これはプログラムの入口又は出口で実行に必要な定数、作業エリア、センス・スイッチなどを初期の状態にリセットするようなプログラムの機能である。即ち、1度実行が完全に終了するまでは新たな実行は出来ないため、このプログラムの実行要求が多数発生した時は、実行待ち行列が作成され、実行されるようになる。

(3) リエントラント

これはいくつかの実行要求を同時に処理出来るようなプログラムの機能である。これは、実行中にプログラムの流れや定数を変更せず、作業エリアを持たないようなプログラムである。

即ち、プログラムを不変な状態に書き、処理に必要な定数や作業エリアなどは、それを利用するプログラムの方で用意し参照させるようにしたものである。

(4) リューザブル

これはプログラムを何度もロードしなくてもいくつかのタスクによって使用できる機能をいう。即ち、あるプログラムを実行し終了したあとで、そのままの状態で再び使用することが出来るプログラムの機能である。

3. プログラム設計とプログラム言語

前述のように、プログラムの設計を効率的に行うための機能に関して、Simple Structure、Dynamic Structure、Overlay Structure、Serially Reusable、Reentrantなどについてみてきたが、これらのすべてを実現出来るプログラム言語はアセンブラ言語である。その他の高水準言語は全部というわけにはいかないが、ユーザーが利用可能なものは次のようなものがある。

表 1 プログラムのデザインとプログラム言語

プログラムデザイン上の諸機能	PL/1	RPG	COBOL	ALGOL	FORTRAN
simple struture	できる	できる	できる	できる	できる
overlay structure	できる	できない	できる	できない	できる
dynamic structure	できない	できない	できる	できる	できない
serially reusable progran	できる	できない	できる	できる	できる
reentrant program	できる	できない	できない	できない	できない

4. プログラムの分割

プログラムには、色々な理由でいくつかのまとまりのある部分に分割されている場合がある。分割された個々の部分をセクションという。セクションには実行可能制御セクションと参照制御セクションに分類される。原始プログラムは、それ自体あるジョブをするために作成されたもので、まとまった論理的単位をまつものである。例えば、アセンブラ言語で在庫管理をするプログラムを作成すれば、このプログラムのジョブは、「在庫管理をする」ということである。このような目的を持った原始プログラムは、数ステップになるときもあるし、膨大なステップになるときもある。これを管理することは大変面倒である。アセンブルする時間も無駄になる。そのために、一つのジョブをする原始プログラムを複数個のプログラムに分割してアセンブルをし、ロードモジュールを作成するときにこれらの3つのプログラムを結合する。図4はプログラムの分割によるロードモジュールの作成概念である。例えば、在庫管理とするプログラムは、処理の過程を考えれば、①データの入力、②入力データの処理、そうして③結果の出力という3つのまとまったブロックに分割することができる。

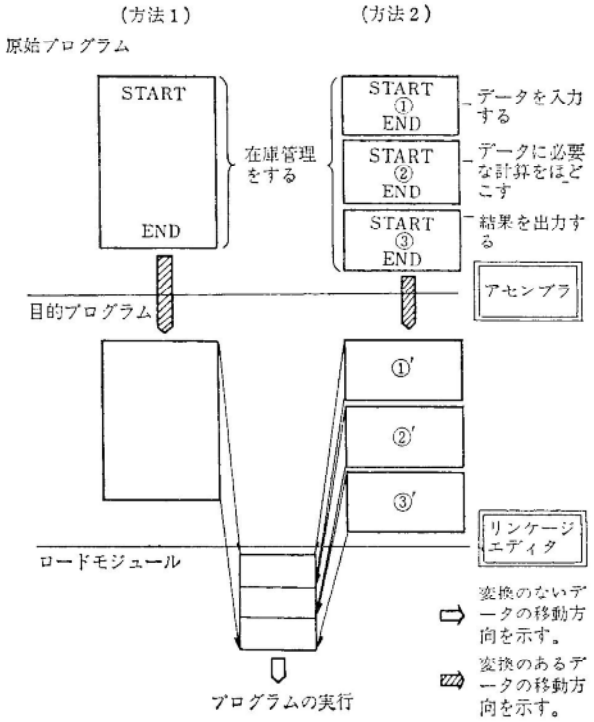

図 4 プログラムの分割によるロードモジュールの作成

実行プログラムは、実際には実行する命令のみから構成されるものではなく、命令が実行するために必要な定数と、結論を一時的にストアするようなエリアも含まれている。実行部分とそうでない部分は、原始プログラム上では、実行可能制御セクション、及び参照制御セクションに分かれて定義をする。

図5は制御セクションの分割である。前述の「在庫管理」の原始プログラムについて考えてみる。

データを入力する部分のプログラムは、データエリアを確保しなければならない。データを

入力するプログラムは、入力したデータを分類して次の処理へ渡さなくてはならないので、このプログラムも、複数個の制御セクションに分類する。ロードモジュールを作成するリンケージエディタは、他の目的プログラムから、ある実行可能制御セクションを結合することが出来る。

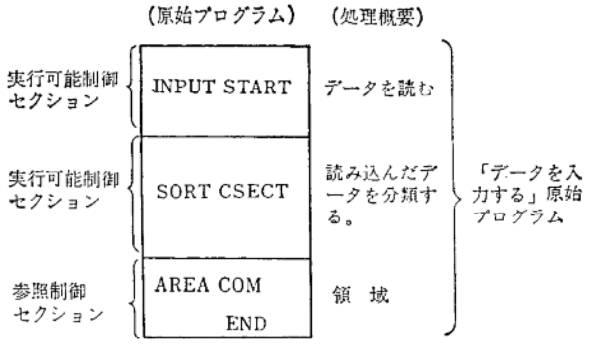

図５制御セクションの分割

PFUアセンブラは、セクション毎に番地割当カウンタを設定する。各セクションに対する番地割当カウンタは、そのセクションの最初の位置で0にセットされ、それ以後そのセクション内に命令もしくはデータが出てくるたびにそれらの語長だけ加算されていく。つまり、番地割当カウンタの値はそのセクションに属する命令やデータに対し、セクションの先頭番地から相対的距離をきめていく。 アセンブラは、1エレメント内の各セクションに対し、先頭番地を割り当てる。したがって、セクションはローディングの最小単位ということが出来る。又、論理的な流れを持つプログラムをセクションに分割する場合、そのプログラムの流れから見て区切りのつく部分を1つのセクションとすべきである。

1つのエレメントは、すくなくとも1個のセクションより構成される。したがって、1つのエレメント中には、少なくとも1個のセクションに関する命令がなくてはならない。ノットページ構造をとるプログラムでは、エレメントがローディングの単位となるので、セクションに関する命令はなくてもよい。もし仮想セクションがあれば、それを無視して翻訳する。コモンセクションは、他の言語処理プログラムのオブジェクト・プログラムの結合で用いる必要があるので、コモンセクションは正規の意味をもつ。リンケージ・エディタからみたセクションは、ローディングの最小単位すなわちセグメントとして扱われる（表2)。

表2アセンブラ命令一覧

機能分類	命令コード	機　能　概　要
プログラム制御命令	PROG	プログラムの開始を宣言すると共にプログラム名及びプログラム番号を定義する。プログラムの先頭に一つ必要である。
	VECT	プログラムベクタ付きプログラムであることを宣言すると共に，アセンブル開始アドレス及びプログラムの属性を宣言する。PROG命令の直後に VECT／NOVECT いずれかが一つ必要。
	NOVECT	プログラムベクタなしプログラムであることを宣言すると共に，アセンブル開始アドレス及びプログラムの属性を宣言する。
	ENTRY	プログラムのエントリポイントを指定する。ENTRY命令は一つのプログラムで最大16個使用できる。NOVECT 命令が指定されたプログラムでは本命令は使用できない。
	END	プログラムの終了を宣言する命令である。プログラムの最後に一つ必要である。
サブプログラム結合命令	DSUB	プログラム中で使用する外部サブプログラムのプログラムエントリ番号を指定する。NOVECT 命令が指定されたプログラムでは本命令は使用できない。
	CALL	DSUB 命令で指定された外部サブプログラムに制御を移す。
セグメント制御命令	DSEG	プログラム中で使用するセグメントのセグメント名とそのプログラムエントリ番号を指定する。NOVECT 命令が指定されたプログラムでは本命令は使用できない。
	LOAD	DSEG命令で指定されたセグメントを主記憶にローディングする。ローディングされたプログラムに制御を移すには CALL 命令を用いる。
	DELETE	LOAD 命令によって主記憶にローディングされた プログラムを削除する。
プログラム結合命令	GLOBAL	はん用記号（GLOBAL SYMBOL）を宣言する。
	EXTRN	外部記号（EXTERNAL SYMBOL）を宣言する。
共通領域定義命令	COM	タスク内共通領域の大きさと名前を定義する。
	SCOM	タスク間共通領域の大きさと名前を定義する。

一方、プログラム単位間の結合は次のようなものがある。

・実行時にプログラム間の結合を行うもの。

・記号によるプログラム間の結合などがある。

プログラムから他のプログラムを呼び出す機能は、DSUB命令とCALL命令がある。図6は実行時におけるプログラム間の結合過程である。DSUB命令によってサブプログラムXYZへの制御移行の情報が作られる。この分岐命令のアドレス部はサブプログラムXYZが主記憶上に配置されたとき、そのエントリアドレスがEBローダの機能によって設定される。このDSUB命令で定義された制御移行の情報をトランスファベクタという。図7はその結合例である。

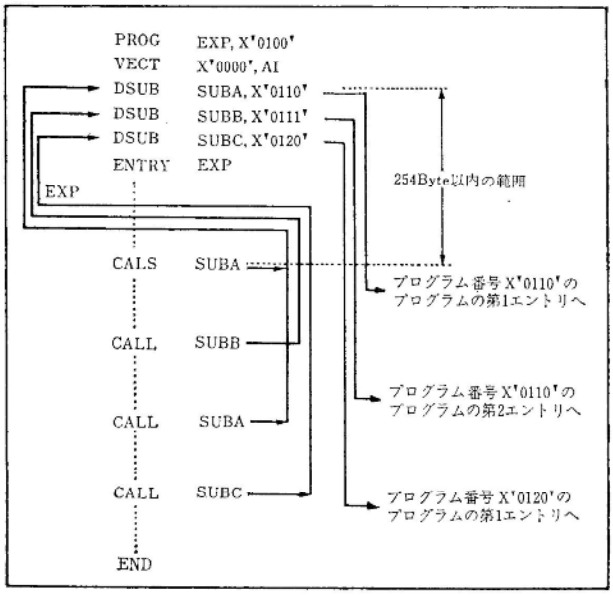

図7　プログラム間の結合例

プログラム間の結合を行う方法は、①呼出し主プログラムが主記憶にローディングされた時、呼出されるサブプログラムも同時に主記憶上に配置しリンケージをとるものである。②主プログラムが実行時において必要とする時にサブプログラムを補助記憶から主記憶上にローディングして来て、リンケージをとるものである。このように実行時において動的にサブプログラムを補助記憶から主記憶上にローディングし、そのリンケージをとる方法をセグメンテーションという。セグメンテーションを制御するものとして、DSEG命令とLOAD命令及びDELETE命令かある。

一方、記号によるプログラム間の結合は、汎用記号と外部記号を用いて実行する以前に別々に翻訳されたプログラム間の結合をいう。図8は記号によるプログラムの結合例である。図において、プログラム1、2を別々に翻訳し、実行形式に変換する時、同時にEBREGの入力になるようにすれば、プログラム2の外部記号GXA、GBXは結合される。

図8の例では、記号GXA、GBXは番号①、②ではは ん用記号として宣言され、番号⑤、⑥では外部記号として宣言されている。番号③でGXAが定義され、番号④でGXBが定義されている。番号⑦の命令文で使用されている記号GXBは外部記号である。従って翻訳時には値が決まらないが、EBREGで結合する時、番号④の文の先頭アドレスが割り当てられる。番号⑦の命令を実行するとはん用レジスタ4にF'3'が入る。同様に番号⑧の命令を実行すると、制御は番号③に移る。

以上の様に、プログラム結合の特徴は、プログラムの分割作成を簡易にし生産性を向上させる点と、数種のプログラム言語を一つのプログラムの為に用いることが出来る点である。

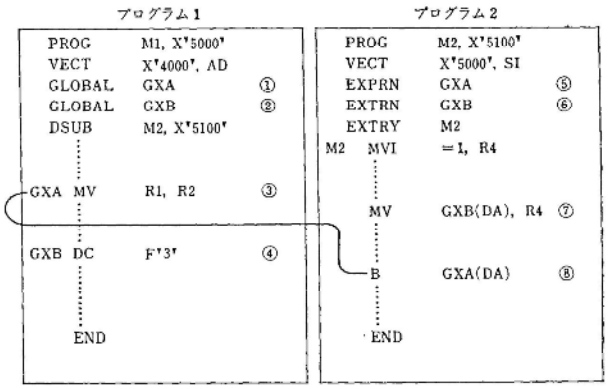

図8 記号によるプログラムの結合例

プログラムは、その設計上の都合により数百ステップ程度の小さいものから数万ステップ以上の大きいものまである。とくにオペレーティング・システムやタイムシェアリング・システムの傾向が一層進展するために、システム全体のステップ数が益々増大の方向にある。一方、主記憶装置の容量に限度があるので、プログラム作成上の問題からも、プログラムを分割して作成し、あとで結合することが不可欠要素となっている。

この段階の目的は、分割されてアセンブルまたはコンパイルされたプログラムを結合することである。結合されたプログラムは全部主記憶装置内に格納できる場合と、格納しきれない場合がある。前者は次の方式にもとづいて行われる。即ち、ⓐ指定どおりの各セクションと接続を行う。ⓑコモンセクションがある場合には、その部分を同一主記憶装置に割当するようにする。Ⓒ「ライブラリ」から呼び出すプログラムがある場合には、該当なものを呼び出して接続をする。後者の場合はツリー構造の原理を適用して処理をする。即ち、プログラムを接続する際に、主記憶装置の同一領域へ複数個のセグメントがはいるようにすればよい。図9はその一例であるプログラムの作成から実行までの状態である。

即ち、図9において、開発した該当プログラムに対して、ソフト・ハードライブラリを参照（インタープリタ方式で・・・）しながら、object fileを生成（製造）する。いくつかのobject fileをリンケイジエディタを経由してload fileを生成（製造）する。いわゆるこのload fileがコンピュータ＆NETに対して初めて実行可能形式となる。

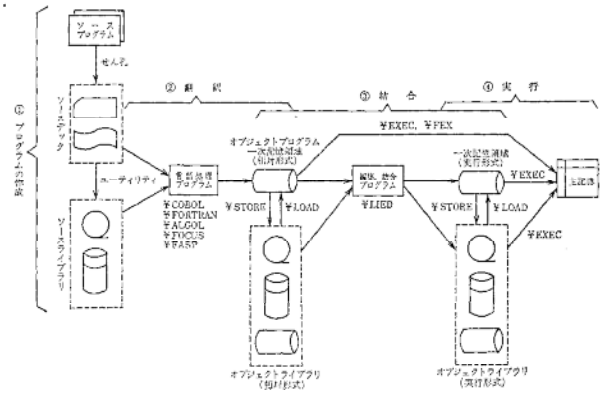

図 9 プログラムの作成から実行までの手順

(1)作成段階

ここではコーディングシート上にプログラムを作成する。こるがソース・プログラムに該当する。ソース・プログラムは各種のプログラム言語そのままの形で作られている。

(2)翻訳段階

ここでは言語処理プログラムを用いて、ソース・プログラムに翻訳する。翻訳ずみのプログラムをオブジェクトプログラムという。翻訳直後のプログラムは、相対形式になっている。（図10）

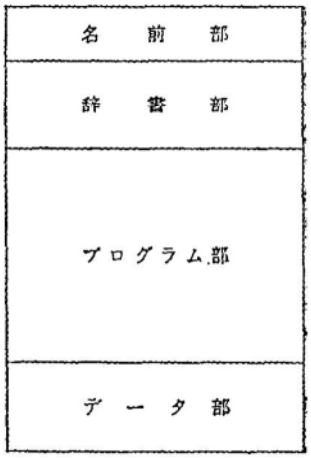

図 10 プログラムの構成

(3)結合段階

ここではいくつかの相対形式のプログラムを結合して一つの実行出来プログラムを作成する。プログラムの結合を行うのは、結合編集プログラムのことである。結合編集プログラムを用いることによって、相対形式のプログラムを実行形式プログラムにすることが出来る。

(4)実行段階

相対形式のオブジェクト・プログラムを実行形式に変換し、又はすでに実行形式になっているオブジェクト・プログラムをシステム・ドラムへロードして実行に入る。

5. プログラムの共通領域

プログラム・モジュール内における主記憶領域の定義命令U-1400モデルにはDSの命令があるが、別々にアセンブルされるプログラム・モジュール間におけるデータの受け渡しや作業領域を目的として共通領域がある。

これらのDS命令との相違は、確保される領域がプログラム・モジュール内に確保されるか、共通領域としてプログラム・モジュール外に確保されるかであり、定義の記述はDS命令と全く同様である。

ここで、プログラム・モジュールとは、プログラム作成時における1回のアセンブルの単位である。又プログラムとは、1つのまとまった仕事の行単位であり、1個またはいくつかプログラム・モジュールが、実行段階における1つのまとまりとして結合されたものである。

プログラムはジョブの手続きを記述したものであり、プログラムの実行の優先レベルが割り当てられて、タスクと呼ばれる制御単位で行われる。

COM命令は、プログラム・モジュール間の共通領域、即ちプログラム内の共通領域を定義し、SCOM命令は、プログラム間の共通領域を定義するために使用される。

コモン・セクションをCOM領域、システム・コモン・セクションをSCOM領域と呼ぶ。従って、プログラムの構成において、2つの共通領域が存在するものとして図11の様に表現することが出来る。

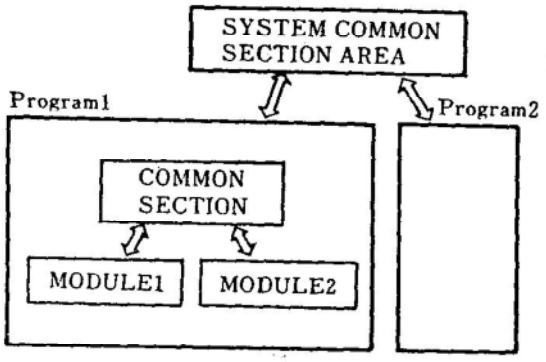

図 11 共通領域の概念

ソース・プログラムの中で、COM又はSCOM命令が現われるごとに、そのオペランド・フィールドで定義された大きさの領域が共通領域としてリザーブされてゆく。これをコモン・サブフィールドという。そして名前フィールドに記述された記号は、そのコモン・サブフィールドの領域の先頭のアドレスを示す。その後に続くCOM又はSCOM命令で定義されるコモン・サブフィールドは、その直前のコモン・サブフィールドの続きとしてリザーブされてゆく。

このようにしてリザーブされたコモン・サブフィールドのトータルが、そのプログラム・モジュールにおける共通領域の大きさを示すことになる。図12はそのモジュールの結合を示し、又、このときのCOM領域及びSCOM領域は図13のようになる。

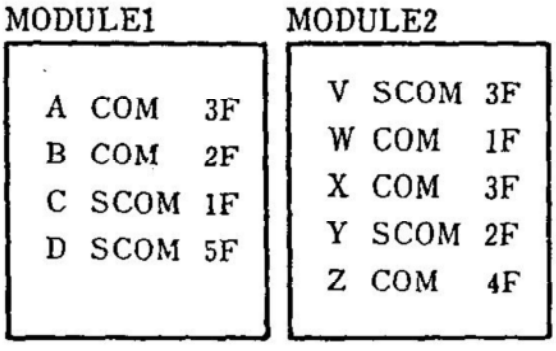

図 12 共通領域を含むモジュール

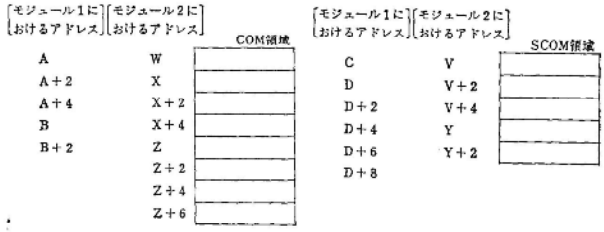

図 13 共通領域の配置

即ち、共通領域の大きさは、いくつかのプログラム・モジュールが結合された時、これらの定義しているCOM領域とSCOM領域の最も大きな領域が、リザーブされる必要がある。

謝辞

本論文を執筆する際、富山鐵朗先生のご指導いただいた。ここに深く感謝申し上げます。

参考文献

富山鐵朗,渋井二三男,電子計算機のシステムプログラム、法学書院

渋井 二三男

明治大学大学院電気工学研究科後期博士課程了。沖電気工業㈱制御方式部入社を経て、コンピュータ、ネットワーク、D10・D20局用電子交換機・DPBXNETWORKの共同開発・研究のためNTT武蔵野研究所に研究員として出向。
豪州、ホンジェラス、ヴェネゼエラ国営郵政NTEWORK、中華民国民需ネットワーク応札開発事業に参加。
1987年より城西短期大学教授、同現代政策学部、御茶ノ水女子大学にて教鞭とる。
文部科学省・放送大学・メディア教育開発センター共同研究員。現在、ネットワーク・情報処理の教育・研究に従事。電子情報通信学会、情報処理学会、教育システム情報学会会員・評議員及び本学会賞受賞。日本ユニシス㈱より努力論文賞受賞。工学博士、アナログ第一種電気通信技術者（総務省）。