MIMD 型並列計算機 HYPHEN C-16 における性能評価用プログラミングシステム†

末吉敏則†† 有田五次郎†††
最所圭三†† 牛島和夫††

MIMD 型高多重並列計算機システムを構成する場合の基本問題のうち，我々は既に同期のオーバヘッド問題に対して FIFO キューを同期手段とする待ちなし並列プログラム (SPP) の概念と，メモリアクセス競合問題に対しては階層ルーチングバス (H-R バス) とを提案した．そこで，上記提案のシステムアーキテクチャの評価を行い，並列処理のためのソフトウェアを開発することを目的として並列計算機 HYPHEN C-16 を製作した．本論文では，HYPHEN C-16 の性能評価を行うために開発したプログラミングシステムについて述べ，実際にこのシステムの上で応用並列プログラムを作成した経験から明らかになった幾つかの問題点を挙げる．このプログラミングシステムは，SPP の構造をそのまま記述することを念頭に開発したもので，強くマシンに依存している．しかし，ここで挙げる問題点のうち，並列プログラムの巨大化およびそのデバッグの問題は MIMD 型高多重並列計算機におけるプログラミングに共通であり，またメモリ割当て (プロセッサ割当て) 問題も構造を持った並列計算機に共通するものである．そこで最後に，これらの問題を解決するために，あるいは解決を支援するために並列処理用プログラミング言語に要求される機能について考察する．

1. はじめに

現在までに数多くの並列処理システムが提案されているが，実際に稼動している並列処理システムのほとんどは応用分野の限られた計算機であり，その多くは SIMD 型の並列処理システムである．このため，最も柔軟性がある MIMD 型の並列処理システムのためのソフトウェアについて，実際に並列処理環境下で使用して評価を行った例はわずかである．MIMD 型並列処理システムのためのプログラミング言語は，並列アルゴリズムの記述能力や使い勝手という意味で応用に密接に関連し，同期手段・データ保護・プロセス間の通信等の機能を通じてオペレーティングシステム (OS) さらにはアーキテクチャやハードウェアにまで関係する．したがって，実際の並列処理環境において並列処理用プログラミング言語の実装実験を行ってその問題点を研究し，これを解決するために並列処理用プログラミング言語あるいはその実行支援環境に要求される機能を明確にすることは，実用性の高い汎用性のある並列処理システムを開発する上で重要である．

MIMD 型高多重並列計算機システムを構成する場合の基本問題のうち，我々は既に同期を高速化する方法として FIFO キューを同期手段とする待ちなし並列プログラム (Self-synchronizing Parallel Program, SPP)[1)~3)] と，アクセス競合問題を解決するための階層ルーチングバス (Hierarchical Routing bus, H-R バス)[4)] とを提案した．SPP を実行する並列計算機のモデルは，プロセッサユニット (pu) とメモリユニット (mu) から成るプロセッサモジュール (pm) をアクセス機構によって結合したモジュラ型のメモリ共有型複合計算機である．

回線結合の複合計算機システムにおいても，システム内にある他計算機のメモリ内容をアドレスを指定して直接参照することが可能であれば，データ参照が遅いことを除いてバス結合のメモリ共有型システムと等価であるとみなすことができ[3)]，したがってこの上で SPP が実行可能である．そこで我々は，16 台の pm を回線型 H-R バスで結合した高多重並列計算機システム HYPHEN C-16 を製作した[5)]．

HYPHEN C-16 は，H-R バスの性能を評価するとともに並列処理のためのソフトウェアを開発することを目的としている．本論文は，HYPHEN C-16 の性能評価を行うために開発したプログラミングシステムについて述べ，実際にこのシステムの上で応用並列プログラムを作成した経験から明らかになった本プログラミングシステムの問題点を挙げ，その解決法を考察

† Programming System for Performance Evaluation of MIMD Type Parallel Computer HYPHEN C-16 by TOSHINORI SUEYOSHI (Department of Computer Science and Communication Engineering, Faculty of Engineering, Kyushu University), ITSUJIRO ARITA (Department of Computer Science, Faculty of Engineering, Kyushu Institute of Technology), KEIZO SAISHO and KAZUO USHIJIMA (Department of Computer Science and Communication Engineering, Faculty of Engineering, Kyushu University).

†† 九州大学工学部情報工学科
††† 九州工業大学工学部情報工学科

する.

2. HYPHEN C-16 の概要

2.1 システム構成

HYPHEN C-16 は 16 台の pm を2進木構造の回線型 H-R バスによって結合した複合計算機システムであり，その構成を**図1**(a)に示す．各 pm は，バスに接続される物理的位置によって決まるシステムアドレス (SAD) と呼ぶモジュール識別番号を持っており，システム内ではこれによって一意に識別できる．HYPHEN C-16 は，回線結合の複合計算機であると同時に，メモリ共有型並列処理システムのモデルである．よって，システム内のアドレス空間は，図1(b)に示すように SAD とモジュール内のロケーション (LOC) との2次元アドレスで表され，単一のアドレス空間を構成する．

システムの諸元は以下のとおりである．なお，pm #F (SAD=F) は入出力装置を備えた,マイクロコンピュータシステムであり，OS として Digital Research 社の CP/M が動作する．したがって，これを HYPHEN C-16 の制御システムとして使用する．

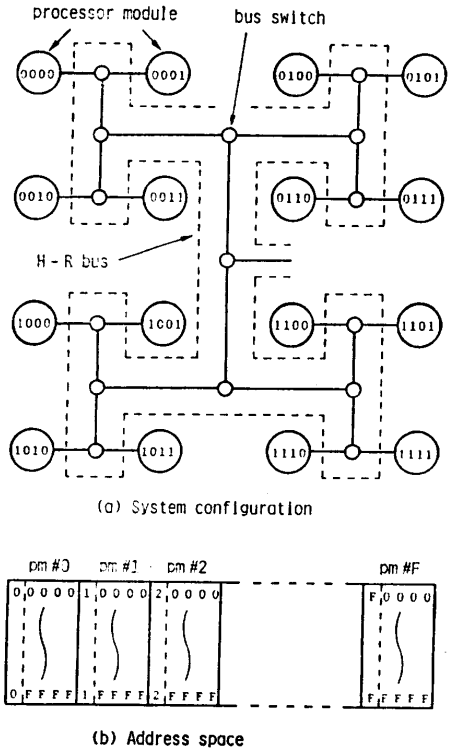

図 1 HYPHEN C-16 の構成とアドレス空間
Fig. 1 System configuration and address space of HYPHEN C-16.

pm #0～E　CPU : Z-80 (4.0 MHz), RAM 12 KB, ROM 4 KB

pm #F　CPU : Z-80 (2.5 MHz), RAM 64 KB, FDD 2台, CRT, PRINTER, CP/M システム

アクセス機構　回線型 H-R バス (19,200 bps 直列転送)

2.2 プログラムの構造と制御の移動

図2(a)の逐次的プログラムはデータの依存関係を考慮して(b)のようなグラフで表現でき，適当な変換操作によって(b)から(c)を得ることができる[2]．ここで点はオペレーションを，枝は制御の移動を表し，同一列のオペレーションは同一プロセッサで実行されることを意味する．したがって，異なる列の点に向かう枝は異なるプロセッサで遂行されるオペレーションの起動，すなわち並列分岐を表す．異なるプロセッサ上のオペレーションは並列実行が可能であるから(b)，(c)は並列プログラムを表現している．

図2(c)のように表現されたプログラムの各列が異なるプロセッサのメモリに分散格納され，各プロセッサが FCFS (first come first served) でそのプロセッサ上のオペレーションを実行すると，(c)の実行結果は(a)の実行結果と等しくなる．このような木構造グラフで表現される並列プログラムを待ちなし並列プロ

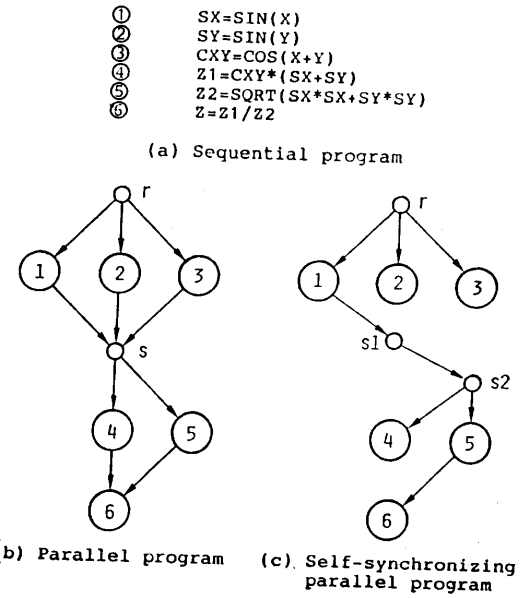

図 2 待ちなし並列プログラムの例
Fig. 2 Example of self-synchronizing parallel program (SPP).

グラム (SPP) と呼ぶ. SPP は同期操作を持っておらず，同期は FCFS という逐次操作に還元されている.

SPP において，一度制御が渡ると中断されずに実行される同一プロセッサ上のオペレーションの集合をタスクと呼ぶ. タスクは SPP の実行の単位であり，各 pm の mu 上に分散配置される. 各 pm は pu を持っているので，異なる pm に割当てられたタスク，すなわち SAD を異にするメモリ領域にあるタスク同士は並列実行が可能である.

各 pm はその pm 内のタスクの実行順序を制御するために実行すべきタスクの入口番地を保持する FIFO キューを持ち，各 pu はそのキューに対する二つの特別な命令を持つ.

(1) タスク切換え (exchange task, EXTASK)

EXTASK 命令は，自分のキューの先頭データをプログラムカウンタに入れる. これにより新しいタスクに制御が渡る. もしキューが空であれば pu はアイドル状態となり，次に実行すべきタスクがキューに登録されるのを待つ.

(2) 並列分岐 (parallel branch PARBR)

PARBR 命令は，タスクの入口番号 (SAD, LOC) をオペランドとし，SAD で指定された pm のキューに LOC を登録する. この命令を実行した pu は引続き実行を続けることができるので，この操作は並列分岐となる.

すべてのタスクにおいて最後に実行される命令は EXTASK 命令である. タスクへの物理的な制御の移動は，前に実行中であったタスクが EXTASK 命令を実行したときに起こるが，論理的には他のタスクが PARBR 命令を実行したときに起こると考えてよい. すなわち，タスクは他のタスクの発行した PARBR 命令で起動され，自分の発行する EXTASK 命令によって終了する.

2.3 基本マクロ

回線結合の複合計算機において SPP を実行するために必要な基本機能は以下のとおりである. これらは括弧内に示すようにメモリ共有型並列処理システムのハードウェアに要求される機能と同じである.

(1) データ共有 (共有メモリ)
　　他 pm のメモリの読み書き

(2) 制御の移動 (割込み)
　　他 pm 上のプログラムの起動

(3) 排他制御 (テスト・アンド・セット)
　　メモリ領域のロック／アンロック

HYPHEN C-16 ではこれらの機能を実現するために，それぞれに対して次の基本マクロを用意している.

(1) READ／WRITE

(2) PARBR／EXTASK

(3) LOCK／UNLOCK

これらの基本マクロを含めて，OS に必要な機能を実現する基本マクロ[3]の処理ルーチンを分散型 OS の核(カーネル)と呼ぶ. カーネルは，各 pm 上の ROM にファームウェアとして実装している.

3. プログラミングシステム

3.1 プログラミングシステムの構成

HYPHEN C-16 上で動作する並列プログラムを記述するためのプログラミングシステムの構成を図 3 に示す. このプログラミングシステムは，CP/M の下に動作する.

プログラミングシステムにおいては，並列アセンブラと並列リンカがその中心的役割を果たす. この並列アセンブラは単に基本マクロの使用を許して並列プログラムの作成を容易にするだけでなく，並列プログラムの構造を決定している. また，並列リンカはプログラム間の結合を行うことによりプログラムライブラリの使用を許すとともに，オブジェクトプログラムに対するプロセッサ割当ての変更やメモリ内での再配置を自由に行えるようにしている.

並列アセンブラだけでも並列プログラムの作成は可能であるが，複雑な処理を行うプログラムになるとそ

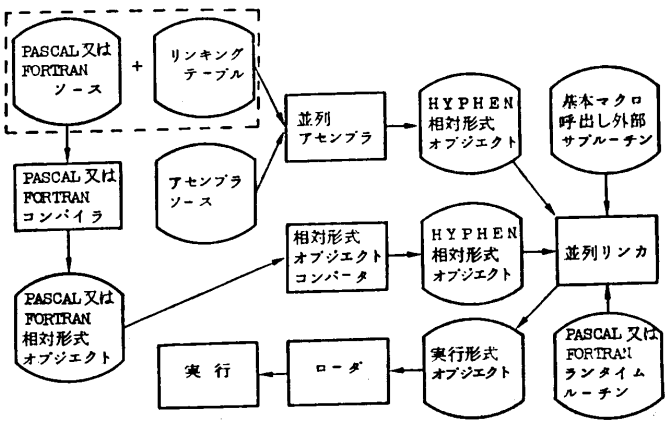

図 3 プログラミングシステムの構成
Fig. 3 Configuration of programming system.

の記述が困難になり，並列プログラム記述用の高水準言語が望まれる．FORTRAN や PASCAL は本来逐次プログラムを記述するための高水準言語であり，そのままでは並列プログラムを記述できない．しかし，並列プログラムは各 pm 上で並列実行可能なように複数のタスクに分割されるので，並列実行される各々のタスクは，基本マクロを外部サブルーチンの形で組込むことにより，FORTRAN のサブルーチン副プログラムや PASCAL の手続きとして記述することができる．この場合問題となるのは，基本マクロのパラメータとなる他 pm 上に配置されたタスクの入口や変数のアドレス情報を FORTRAN や PASCAL で書かれたプログラムだけで知ることはできないことである．そこで本システムでは，後述するように他 pm 上の変数のアドレス情報をリンク時に外部から与える方法によってこの問題を解決している．

3.2 並列プログラムの概念

本システムにおける並列プログラムは，論理的に一つの処理を行うプロセスが複数の pm に分散配置され並列に実行される．そこで，実際に実行される機械語の並列プログラムの構造をそのまま記述するために，並列アセンブラとその出力である相対形式オブジェクトに以下の概念を導入した．

（1） エレメント

並列アセンブラに対する入力である一つのソースファイルをエレメントと呼ぶ．エレメントは一度にアセンブルされるソースプログラムの単位である．並列アセンブラが生成する複数の相対形式オブジェクトファイルは，汎用記号と外部参照記号により結合される．この結合された結果のプログラムは，単一のプロセス ID を持つ一つのプロセスとしてシステム上にロードされ，実行される．

（2） セグメント

一つのエレメントは，一つまたは複数のセグメントから成る．セグメントは，同じ pm 上にロードされるプログラムの最小単位であり，タスクやデータの集合である．並列リンカは，このセグメントを単位として結合・メモリ割当て・再配置を行う．

並列プログラムでは，効率を上げるために各プロセッサが全く同一の演算ルーチン等をそれぞれのローカルメモリ上に持つ場合がある．そこで，このような共通あるいは汎用の処理を記述したセグメントに対してフローティングセグメント (FS) という概念を導入している．図4に示すように，FS を宣言したセグメントはそれを参照する各セグメントと同じ pm 上に常にロードされる．このため，図4の label 3 のように FS 内で定義されたラベルは，同時に複数のプロセッサ上の記憶場所を表していることになる．

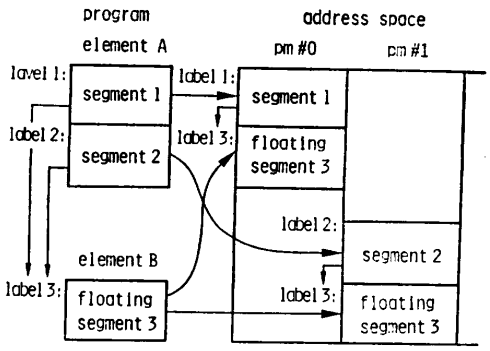

図 4 プログラムの再配置
Fig. 4 Relocation of program.

表 1 並列アセンブラの擬似命令
Table 1 Pseudo instructions of parallel assembler.

擬似命令	機能
ELEMNT	エレメントの始まりを宣言する．
END	エレメントの終わりを宣言する．
SEGMNT	セグメントの始まりを宣言する．
GLOBAL	汎用記号を宣言する．
EXTERN	外部参照記号を宣言する．
EQU	記号を定義する．
DB	1バイトデータを定義する．
DW	2バイトデータを定義する．
DX	式のシステムアドレスとロケーションをそれぞれ2バイトで展開する．
DS	作業領域を定義する．
REFER	記号の参照を定義する．

（3） アドレス

HYPHEN C-16 のアドレス空間における特定のアドレスを表すためには，各プロセッサに付けられた1バイトのシステムアドレス (SAD) とローカルメモリ内での2バイトのロケーション (LOC) とを必要とする．これを再配置可能にするために相対形式オブジェクト内では，前者をセグメント ID，後者をセグメントの先頭からの相対アドレスで表す．これにより，複数の pm のメモリにまたがった並列プログラムにおける任意のアドレスを表すことができる．

3.3 並列アセンブラ，並列リンカ，ローダ

（1） 並列アセンブラ

並列アセンブラは，一つのエレメントから成るソースファイルを読込み，相対形式オブジェクトを生成する．このとき，そのエレメントに含まれる各セグメン

表 2 並列リンカの制御文
Table 2 Control statements of parallel linker.

制御文	機能
SELECT	選択するエレメントとセグメントを指定する.
ALLOC	セグメントにメモリを割当てる.
PROCESS	プロセス ID を指定する (省略 : ID=1).
START	実行開始アドレスを指定する.
OBJECT	実行形式ファイルの名前を指定する.

トは，それぞれ自由に再配置可能な状態にある．並列アセンブラに入力されるソースファイルは，通常の機械語命令の他に，基本マクロ命令や表 1 に示す擬似命令を含んでいる．

(2) 並列リンカ

並列リンカは複数のエレメントの相対形式オブジェクトファイルを読込み，それらの結合・メモリ割当て・再配置を行い，実行形式オブジェクトファイルを出力する．並列リンカに対して，これらの指示を行うのがリンカ制御ファイルである．リンカ制御ファイルには，前記の指示の他に，実行開始アドレスやプロセス ID の指示に含む．これらの指示は表 2 に示すリンカ制御文を用いて記述する．

ただし，SELECT 文により選択されながら ALLOC 文でメモリ割当てが指定されていないセグメントに対しては，並列リンカが自動的にメモリ割当てを行う．この場合，まだ何も割当てられていない pm のメモリ領域を優先的に割当てる．また，フローティングセグメントは，参照されたときにいもづる式に自動的に組込まれ，ALLOC 文で陽に指定する必要はない．

(3) ロ ー ダ

ローダは絶対ローダであり，並列リンカが出力した一つのプロセスの実行形式オブジェクトファイルを入力し，プログラムを各 pm の指定されたロケーションにロードする．起動時にプログラムの実行を指定すれば，ロード後直ちにその実行開始アドレスへ並列分岐を行う．入力となる実行形式オブジェクトファイルは，プロセス ID，実行開始アドレスおよび複数個のブロックから成っている．ブロックはロードの単位であり，それぞれブロック内に含まれるオブジェクトのバイト数，ロード開始アドレスおよびオブジェクトから成る．

3.4 既存高水準言語を利用する方法

(1) 相対形式オブジェクトコンバータ

CP/M 上で利用可能な多くのコンパイラは，Microsoft 社標準相対形式のオブジェクト (REL) を生成する．相対形式オブジェクトコンバータは，これを本システムにおける相対形式オブジェクト (PRL) に変換する．これを用いることで CP/M 上の各種プログラミング言語を HYPHEN C-16 のために用いることができる．

REL ファイルは，モジュールと呼ばれる再配置の単位の集まりである．各モジュールは，絶対形式，プログラム相対形式，データ相対形式，コモン相対形式のセグメントを持つ．ここでのコモンは，FORTRAN 等のコモンブロックに対応している．相対形式オブジェクトコンバータは，REL ファイルの各セグメントを PRL ファイルのセグメントに変換する．ただし，絶対形式のセグメントは，PRL ファイルのセグメントがすべて相対形式であるため変換できない．また，コモン相対形式のセグメントは，変換後の実体を PRL ファイルの内部に作るか外部に作るかを指定できる．

(2) FORTRAN の場合

FORTRAN にはコモン文がある．相対形式オブジェクトコンバータは，コモン領域の実体を外部参照の形で外部に作成することを許す．このことを利用して，プロセッサ間の相互参照に必要なアドレス情報を FORTRAN プログラムに与える．

実際に FORTRAN プログラムを書くときには，相手に見せる変数および相手のアドレス情報を示す変数のコモンブロック内における割付けをコモン文により指定する．そこで，並列アセンブラでそのコモンブロックの実体 (これを，リンキングテーブルと呼ぶ) を作り，その中でアドレス情報の相互参照を行う．これらを並列リンカで結合することにより，FORTRAN プログラムの実行に必要な情報を与える．

(3) PASCAL の場合

ここで使用する PASCAL コンパイラは分割コンパイル可能であるので，最も外側の大域変数と手続きを外部に見せることができ，また外部の大域変数と手続きを参照することができる．この外部参照機能を利用して，FORTRAN の場合と同様に，相互参照に必要なリンキングテーブルを並列アセンブラで作成する．

4. 並列プログラム例

ここでは，3 章の理解を助けるために本プログラミングシステムを利用して並列プログラムを作成する例を示す．プログラム例は，S : =1+2+3+4 を X : =1+2 と Y :=3+4 とに分割し，二つのプロセッサ上で並列に処理した後 S : =X+Y とするものである．こ

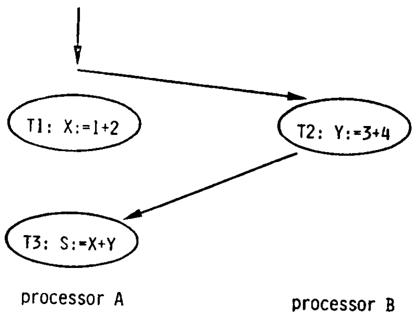

図 5　S:=1+2+3+4 を求める待ちなし並列プログラム
Fig. 5　SPP which calculates S:=1+2+3+4.

```
 ←セグメントID
 |      ←セグメント内相対アドレス
                                    ELEMNT
                                    GLOBAL   ADD12
                          ;
00                        SUMSEG    SEGMNT
00   0000   F7 06 0000    ADD12:    PARBR    ADD34      ①
00   0004   0000
00   0006   3E 01                   LD       A,1         タスク T1
00   0008   C6 02                   ADD      A,2         X:=1+2
00   000A   32 001C                 LD       (X),A
00   000D   F7 07                   EXTASK
                          ;
00   000F   3A 001D       ADDXY:    LD       A,(Y)
00   0012   47                      LD       B,A         タスク T3
00   0013   3A 001C                 LD       A,(X)       S:=X+Y
00   0016   80                      ADD      A,B
00   0017   32 001E                 LD       (S),A
00   001A   F7 07                   EXTASK
                          ;
00   001C                 X:        DS       1
00   001D                 Y:        DS       1
00   001E                 S:        DS       1
                          ;
                          ;
01                        SUBSEG    SEGMNT
01   0000   3E 03         ADD34:    LD       A,3         タスク T2
01   0002   C6 04                   ADD      A,4         Y:=3+4
01   0004   32 0019                 LD       (TEMP),A
01   0007   F7 01 0000              WRITE    Y,TEMP,1   ②
01   000B   001D 0019
01   000F   0001
01   0011   F7 06 0000              PARBR    ADDXY      ③
01   0015   000F
01   0017   F7 07                   EXTASK
                          ;
01   0019                 TEMP:     DS       1
                          ;
                                    END
                                    WRITE マクロ命令
                                    (変数Yのアドレスへ TEMPの
                                     内容（1バイト)を書込む。)
No Fatal Error(s)

Segment(s) :              Referree(s):
 00   SUMSEG:
 01   SUBSEG:

Symbol(s) :
 ADD12    A 00 0000       ADD34    A 01 0000       ADDXY    A 00 000F
 S        A 00 001E       SUBSEG   S 01 0000       SUMSEG   S 00 0000
 TEMP     A 01 0019       X        A 00 001C       Y        A 00 001D
```

図 6　並列アセンブラのみによるプログラム (DEMO 1)
Fig. 6　Parallel assembly language program (DEMO 1) for SPP in Fig. 5.

のプログラムは，**図 5** に示す木構造グラフで表される SPP である．図中の円で囲まれた T 1，T 2，T 3 はタスクであり，並列分岐命令 (PARBR) により起動され，タスク切換え命令 (EXTASK) によって終了する．この SPP が実行される手順を以下に説明する．

(1)　プロセッサAが並列分岐命令によって起動される．

(2)　プロセッサ A は，プロセッサ B へ並列分岐命令により T2 の起動要求を送る．

(3a)　プロセッサAは，T 1 の X：=1+2 を実行する．

(3b)　プロセッサBは，T 2 の Y：=3+4 を実行する．

(4a)　プロセッサAは，タスク切換え命令を実行する．

(4b)　プロセッサ B は，プロセッサ A へ並列分岐命令によりT3 の起動要求を送る．

(5)　プロセッサAは，T3 の S：=X+Y を実行する．

最初に，並列アセンブラのみによるプログラム (DEMO 1) を**図 6** に示す．このプログラムは，二つのセグメント (SUMSEG, SUBSEG) から成っている．プログラム中の ADD 12 や ADDXY などのラベルは，システム内のあるアドレスを指すアドレス型の記号であり，その値としてセグメント ID とセグメントの先頭からの相対アドレスを持つ．よって，異なるセグメントを参照する基本マクロは，ラベルを使用することによりセグメント ID を特に指定する必要はない (図 6 の①，②，③)．

次に，FORTRAN を利用したプログラムを**図 7** に示す．図 7 (a) の FORTRAN プログラム (DEMO 2) は，通常の FORTRAN プログラム (逐次プログラム) と同じ形式である．これは，並列実行可能なように分割されたタスクにおける処理が逐次的に行われるためである．図 7 (b) のリンキングテーブル (DEMO 2 SUB) は，FORTRAN プログラムにおいて定義したコモンブロックの実体を記述したものである．FORTRAN プログラムでは，メインおよびサブルーチンプログラムとそこで定義された各コモンブロックとがそれぞれセグメントに対応する．それらのセグメント名は，メインプログラムの場合 $ MAIN に，サブルーチンの場合そのサブルーチン名になる．また，コモンブロックの場合には，リンキングテーブルにおいて定義されたセグメント名である．すなわち，このプログラムは，図 7 (a) で定義した三つのセグメント ($ MAIN, ADDXY, ADD 34) と図 7 (b) で定義した二つのセグメント (COMM 1, COMM 2) から成っている．

上記プログラムを実行するためのリンカ制御文を**図 8** に示す．図 8 (b) の SELECT 文における FORLB 1, FORLB 2 は相対形式オブジェクトコンバータで変換した FORTRAN コンパイラ付属のランタイムルー

```
      INTEGER X,Y,S,SAD                          タスク T1
      COMMON /COMAIN/SAD,LOC                       X:=1+2
      COMMON /VAR/X,Y,S
      CALL PARBR(SAD,LOC)
      X=1+2
      CALL EXTASK
      END
C
      SUBROUTINE ADDXY                           タスク T3
      INTEGER X,Y,S                                S:=X+Y
      COMMON /VAR/X,Y,S
      S=X+Y
      WRITE (3,10) X,Y,S
      STOP
   10 FORMAT (1H ,3I5/)
      END
C
      SUBROUTINE ADD34                           タスク T2
      INTEGER Y,SAD                                Y:=3+4
      COMMON /COMSUB/SAD,LOC,LOCY
      Y=3+4
      CALL WRITE(SAD,LOCY,Y,2)    WRITEマクロを呼出す外部サブルーチン
      CALL PARBR(SAD,LOC)         (SAD, LOCYで指定するアドレスへYの内容(2バイト)を書込む)
      CALL EXTASK
      END
```

(a) FORTRAN source program (DEMO2)

```
                                ELEMNT
                                GLOBAL  COMAIN,VAR,COMSUB
                                EXTERN  ADDXY,ADD34
                          ;
00                        COMM1   SEGMNT
00  0000   0000 0000     COMAIN: DX      ADD34   ;SAD,LOC
                          ;
00  0004                  VAR:    DS      2       ;X
00  0006                  Y:      DS      2       ;Y
00  0008                          DS      2       ;S
                          ;
                          ;
01                        COMM2   SEGMNT
01  0000   0000 0000     COMSUB: DX      ADDXY   ;SAD,LOC
01  0004   0006                  DW      Y       ;LOCY
                          ;
                                END

 No Fatal Error(s)

Segment(s) :          Referree(s):
 00  COMM1 : E01
 01  COMM2 : S00  E00

Symbol(s) :
 ADD34  E 01 0000     ADDXY  E 00 0000     COMAIN A 00 0000
 COMM1  S 00 0000     COMM2  S 01 0000     COMSUB A 01 0000
 VAR    A 00 0004     Y      A 00 0006
```

(b) Linking table (DEMO2LT)

図 7 FORTRAN を利用したプログラム (DEMO 2, DEMO 2 LT)

Fig. 7 FORTRAN program (DEMO 2) and its linking table (DEMO 2 LT) for SPP in Fig. 5.

チンのライブラリであり，PFORLB 2 は基本マクロを呼出す手続きのライブラリである．これらのライブラリに含まれるルーチンは，すべてフローティングセグメントである．

5. 並列プログラム作成における問題点

我々は現在までに，本プログラミングシステムを利用して各種の並列プログラムの作成・実行を行った．これらの並列プログラムの作成過程において次のような問題点が明らかになった．

(1) MIMD 型並列処理システムにおける実際の並列プログラムは，データの参照先や並列分岐先が異なるだけというように，非常に類似した処理を行うセグメントの集まりであることが多い．しかし，上記プログラミングシステムでは，それらの類似したセグメントを個々に記述する必要があり，非常に多くの労力を要する．また，そのようなセグメントから成る並列プログラムのソーステキストは非常に大きくなり，全体の読みやすさを損なってしまう．

```
SELECT  DEMO1     相対形式オブジェクトファイル名(DEMO1.PRL)の指定
ALLOC   DEMO1(SUMSEG)=0F,8000
ALLOC   DEMO1(SUBSEG)=08,2000
PROCESS 2          セグメント名   SAD, LOCの指定
START   ADD12
OBJECT  DEMO1     実行形式オブジェクトファイル名(DEMO1.PEX)の指定
```

(a) Linker control statements for DEMO1 program

```
SELECT  DEMO2
SELECT  DEMO2LT
SELECT  FORLB1
SELECT  FORLB2     ライブラリファイルの指定
SELECT  PFORLB2
ALLOC   DEMO2($MAIN,ADDXY),DEMO2LT(COMM1)=0F,1000
ALLOC   DEMO2(ADD34),DEMO2LT(COMM2)=02,2000
START   $MAIN
OBJECT  DEMO2
```

(b) Linker control statements for DEMO2 program

図 8 DEMO 1, DEMO 2 プログラムのためのリンカ制御文

Fig. 8 Linker control statements for DEMO 1 and DEMO 2 programs.

(2) 既存の高水準言語 (FORTRAN, PASCAL) を利用したプログラミングでは，並列アセンブラを用いてリンキングテーブルを作成する必要があり，記述性が悪く手間がかかる．また，他セグメントの変数や手続きの参照はリンキングテーブルを介して，そのアドレスを自セグメントの変数に取込む方法を採っているので，高水準言語で記述したプログラムだけを見てもセグメント間の関係（制御の流れ）は全く分からない．

(3) HYPHEN C-16 は MIMD 型並列処理システムであるので，実行時のプロセスは自律的に動作しかつ同期（実行の制御，データの排他制御）をとるタスクから成っており，それらの振る舞いを動的に理解することは困難である．よって，並列プログラムのデバッグには大変な時間を費やすことがある．

(4) HYPHEN C-16 は，そのアクセス機構が階層構造になっているので，問題の解法におけるデータ参照の局所性を利用することにより高い効率が期待できる．このため，問題解法に当たっては局所性を高めることに留意し，メモリ割当てに際してはデータ参照ができるだけ近い距離にある pm のメモリに対して行われるように工夫することが重要になる．現在の並列リンカでは，できるだけ高い効率を実現するためにはメモリ割当てをユーザが指定しなければならない．

このメモリ割当てを決定する作業は煩わしく困難である.

これらの問題の解決法について以下に考察する.

まず,(1)の問題を解決する一つの方法は,1個のソースプログラムから最終的に類似した各々のオブジェクトを生成することを可能とする,以下のようなコンパイラ制御機能を実現することである.

(Ⅰ) 繰り返しコンパイル

(Ⅱ) 条件付きコンパイル

(Ⅲ) マクロ展開

これらの機能は,マクロアセンブラや汎用マクロ言語にあるマクロ機能の並列プログラミングへの応用であり,MIMD 型高多重並列計算機システムにおけるプログラム記述量の巨大化を防ぐ有効な手段となる.したがって,コンパイラ制御機能は,並列プログラミングにおける労力の節約と理解のしやすさを図るために有用な汎用のツールになると期待できる.

次に,(2)の問題を解決するには,並列プログラムを各プロセッサに割当てる際の最小単位であるセグメントやフローティングセグメントの概念を高水準言語に導入して,各セグメント間で参照される変数や手続き名をその言語で直接取扱えるようにすることである.

(3)のデバッグ問題は,並列プログラムに限らずプログラミング全般に共通する問題である.プログラムのデバッグは,逐次プログラム,並行プログラム,並列プログラムの順に困難になる.並列処理環境下で実行される並列プログラムに,単一プロセッサで実行される逐次プログラムにおけるデバッグ手法をそのまま適用することには無理がある.

しかし,3.2節で述べた並列プログラムの概念から分かるように,本システムでは並列プログラムを全く変更することなく,これを単一プロセッサでそのまま実行できる.すなわち,並列リンカは,プログラムに含まれるすべてのセグメントを同一のプロセッサに割当てるように ALLOC 文で指定されていると,単一プロセッサ上で動作する実行形式オブジェクトを作成する.また,HYPHEN C-16 に用意している SPP のための実行管理機構は,正しい並列プログラム (SPP) であるならば単一プロセッサに割当てられても正常に実行することを保証している.

このため,並列プログラムのデバッグを,逐次プログラムのための各種ソフトウェアツールを利用して効果的に行える.例えば,行単位のトレース,手続きや関数の入口・出口の表示,およびブレイクポイントの設定などの機能を持つシンボリックデバッガを利用すれば,並列プログラムをソースプログラムイメージで会話的にデバッグすることも可能である.しかし,この方法でも並列処理環境における同期のタイミングのずれを完全にデバッグすることはできない.

そこで,このような同期の虫が発生しにくい並列プログラムをあらかじめ記述できるような高水準言語を設計することが重要になってくる.同期の虫は,HYPHEN C-16 がメモリ共有型システムのモデルであるのでデータをどのプロセッサからでも随時アクセスできること,および並列プログラムを記述するために現在使用できる同期プリミティブが SPP の構造をそのまま記述するための基本マクロ (PARBR, EXTASK) そのものであることなどに原因がある.このため,信頼性の高い並列プログラムを作成できる高水準言語は,データの排他制御のためにデータ抽象化の機能を陽に持ち,高水準の同期プリミティブを持つ新たな制御構造を導入する必要がある.さらに,その処理系は,型,名前の有効範囲,アクセサビリティなどについて非常に厳密にチェックすることも必要である.

最後の(4)の問題に関しては,並列リンカが並列プログラムのオブジェクトコードを調べるだけでこれを解決するのは難しい.この問題を解決するには,並列プログラムを構成するセグメントに対する並列リンカのメモリ割当てを助ける情報を言語処理系が出力する必要がある.この機能を実現するには,データ抽象化を行ってそのモジュール間の参照関係を明確に宣言させること,および参照する頻度を明示するためにセグメント間の'近さ'を指示する制御命令を用意することが必要である.例えば,言語 TASK[6] にある近接指示命令 (proximity directive) は,これに近い目的を有する制御命令である.また,データ抽象化および参照関係の明確な宣言は,(3)の問題を解決するためにも役立つものである.なお,H-R バスの性能を評価するためにメモリ割当てを種々変更して局所性と効率との関係を調べる実験を行うには,メモリ割当てをユーザが指定する機能も必要である.

6. おわりに

本プログラミングシステムは,並列アセンブリ言語や既存の高水準言語 (FORTRAN, PASCAL) を用いて並列プログラムを作成して各種の性能評価テストを

行うことができ，当初の目的を果たすことができた．しかし，それと同時に並列プログラミングにおける種種の問題点も明らかになり，その解決法について考察した．

我々は既に（1），（2）の問題に対する考察に基づき，FORTRAN をベースとした PARFOR および PASCAL をベースとした P-PASCAL[7] という二つの言語の開発を行った．これらは，先述のコンパイラ制御機能をプリプロセッサによって実現し，かつセグメントの概念や基本マクロ命令文を導入してリンキングテーブルを必要としない並列処理用プログラミング言語である．しかし，ここで説明したプログラミングシステムや PARFOR および P-PASCAL は，SPP の構造をそのまま記述することを念頭に開発したもので，強くマシンに依存している．

そこで現在，（3），（4）の問題に対する考察も考慮して高水準プログラミング言語 Nano-2 を設計し，その開発を行っている[8]．Nano-2 は高水準の同期プリミティブを持ち，マシン依存部分(SPP)の抽象化を行った HYPHEN C-16 のための高水準言語を目指したものであるが，言語のマシン依存度と抽象化のトレードオフやその制御構造などについて現在再検討を行っている．

参 考 文 献

1) 有田：FIFOキューを同期手段とする並列プログラムについて(I)ー待ちなし並列プログラムー，情報処理学会論文誌，Vol. 24, No. 2, pp. 221-229 (1983).

2) 有田，荒木：FIFO キューを同期手段とする並列プログラムについて(II)ー並列プログラムの待ちなし変換ー，情報処理学会論文誌，Vol. 24, No. 2, pp. 230-237 (1983).

3) 有田，末吉：FIFO キューを同期手段とする並列プログラムについて(III)ー実行管理機構ー，情報処理学会論文誌，Vol. 24, No. 6, pp. 838-846 (1983).

4) 末吉，有田：階層ルーチングバスについて，電子通信学会論文誌（D），Vol. J 67-D, No. 11, pp. 1309-1316 (1984).

5) 末吉，最所，有田：階層構造高多重並列計算機実験システム HYPHEN C-16 について，情報処理学会論文誌，Vol. 25, No. 5, pp. 813-822 (1984).

6) Jones, A. K. and Schwans, K.: TASK Forces: Distributed Software for Solving Problems of Substantial Size, Proc. 4th Int. Conf. Softw. Eng., pp. 315-330 (1979).

7) 平原，荒木，有田：並列処理用プログラミング言語 P-PASCAL の処理系の実現，情報処理学会ソフトウェア基礎論研究会資料，9-3 (1984).

8) Araki, K., Arita, I. and Hirabaru, M.: NANO-2: High-level Parallel Programming Language for Multiprocessor System HYPHEN, Proc. COMPCON FALL '84, pp. 449-456 (1984).

(昭和 60 年 3 月 27 日受付)
(昭和 60 年 5 月 9 日採録)