Vol. 27 No. 3 情報処理学会論文誌 Mar. 1986

マイクロプログラム制御計算機 QA-2 のシステム管理プロセッサ†

中田 登志之††* 北村 俊明††** 柴山 潔††
富田 眞治†† 萩原 宏††

本論文はユニバーサル・ホスト計算機 QA-2 のシステム管理用プロセッサ SVP のハードウェア構成方式について述べたものである．SVP は，(a)システム記述専用高級言語の処理に適した高速機械命令の実現，(b)マルチ・プロセスの効率の良い実行環境の実現，(c)ソフトウェアのモジュール化の支援，などを達成できるよう構成されている．このような高機能機械命令（マクロ命令）の高速実行を実現するために，SVP では，(a)簡潔な水平型マイクロ命令を用いた制御方式によって得られる柔軟性と高速性の利用，(b)システム管理に重要な役割を果たす主記憶管理機構・アクセス管理機構の高速化手法の導入，などを図っている．本論文では SVP のマイクロ/マクロ・アーキテクチャについて述べた後，OS の一部であるファイル管理プログラムを例にとって本システムで採用した方式の妥当性についても言及する．

1. まえがき

我々は，低レベル並列処理機能（算術論理演算・レジスタ転送レベルでの並列処理）をマイクロプログラム方式で制御するというシステム構成法のもとに，汎用エミュレーション[1]を指向したユニバーサル・ホスト計算機 QA-2 を開発し[2),3)]，図形・画像・信号処理などのリアル・タイム処理や高級言語処理などを始めとする多様な応用に利用している[4)~7)]．

QA-2 では，**図1** に示すようにレジスタ・ALU 部 (RALU) と順序制御部 (SCU) から成る CPU と，主記憶管理プロセッサ (MMP)，システム管理プロセッサ (SVP) の三つの機能部分を独立に構成している．CPU 部では，4台の均質な ALU が水平型マイクロ命令（1ワード＝256 ビット）の相異なるフィールドによって独立に制御され，6k バイトのレジスタを共有して並列演算を実行できる．また MMP は 16 ウェイ・インタリーブ構成によって四つの主記憶アクセスを同時に行うことができ，さらにデータ探索など高機能操作を CPU と独立に実行できる構造になっている．

CPU を構成する RALU と SCU および MMP のハードウェア構成については，他の稿[2),3)]で述べているので，本論文ではシステム管理プロセッサ (SVP) のハードウェア構成，特に，機械命令レベルの (マクロ) アーキテクチャとそれをサポートするマイクロ命令レベルの(マイクロ)アーキテクチャについて詳述する．

2. SVP のマイクロ・アーキテクチャ

2.1 マイクロ・アーキテクチャの設計指針

SVP の基本仕様は，(a)マイクロプログラム・デバッガやエバリュエータおよび実行モニタなど，QA-2 のマイクロプログラミング支援を行う強力な計算機となり得ること，(b)ファイル装置や研究室内ネットワークも含めた外部入出力装置の管理などの通常の OS 機能を高速に実現できること，(c)QA-2 上で種種の仮想計算機のエミュレーションを並行して行うために制御記憶 (CS) や制御テーブルのダイナミックな切り換えを行うマルチ・マイクロプログラミング環境を実現できること，などである．これらの機能を実現するシステム・プログラムは高速性が要求され，また大規模なものとなるので，プログラムの生産性や処理効率を向上し得るよう，(a)システム記述専用高級言語の処理に適した機械命令の装備，(b)マルチプロセスの効率の良い実行環境の提供，(c)ソフトウェアのモジュール化の支援など，SVP 機械命令の機能をシステム管理向きに強化することが必要である．

これらの高機能機械命令をサポートする SVP のマイクロ・アーキテクチャは，(a)極度なハードウェア化は抑えて，他の高級言語処理への対応やマイクロプログラムの生産性向上も容易に達成できるよう柔軟性

† System Supervisory Processor of the Microprogrammable Computer QA-2 by TOSHIYUKI NAKATA, TOSHIAKI KITAMURA, KIYOSHI SHIBAYAMA, SHINJI TOMITA and HIROSHI HAGIWARA (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 京都大学工学部情報工学教室
* 現在 日本電気(株) C & C システム研究所
C & C Systems Research Laboratories, NEC Corporation
** 現在 富士通(株)本体事業部
Main Frame Division of Computer Systems, Fujitsu, Ltd.

に富んだクリーンな構成方式とすること，(b)システム管理に重要な役割を果たす主記憶管理機構や主記憶アクセスの高速化を図ること，(c)可変長機械命令のフェッチ・デコードのオーバヘッドを極力減らすこと，などを念頭に設計されている．

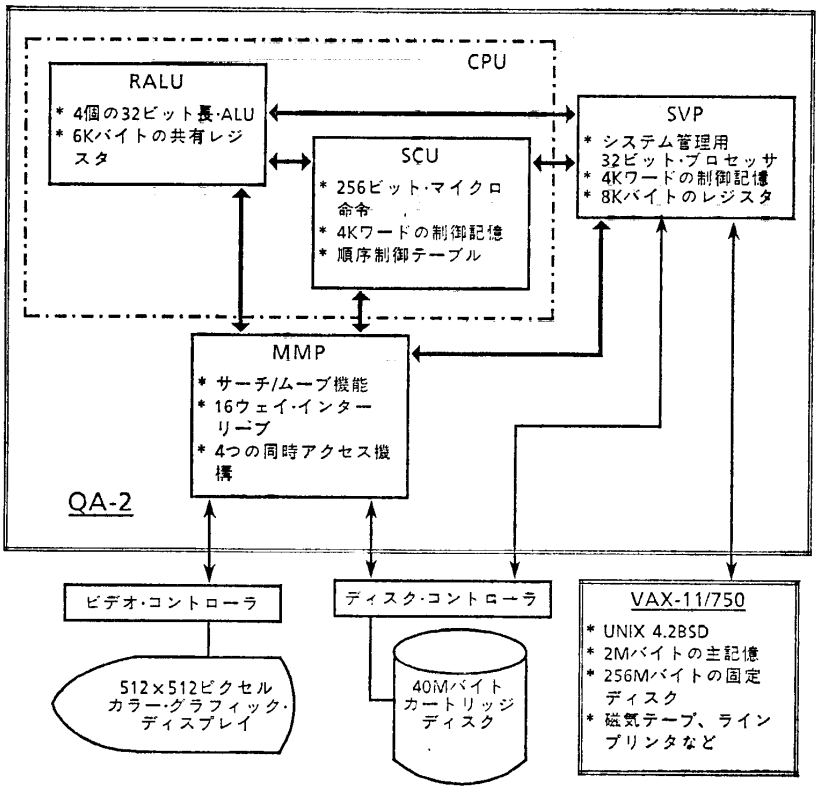

図 1　QA-2 のシステム構成
Fig. 1　System organization of QA-2.

2.2　マイクロ命令形式とバス構成

SVP のマイクロ命令は図 2 に示すように 1 語 114 ビットからなる簡潔な水平型となっている．マイクロ命令の各フィールドは，図 3 に示すハードウェア・モジュールを制御している．タイミング制御部以外のモジュールは，L-BUS, R-BUS, D-BUS の 3 種のバスにより結合されている．各バスの幅は 32 ビットであり，レジスタ部 (WR, IRF など)，および各モジュールの特殊レジスタの内容が，L-BUS, R-BUS を通って演算部へ転送され，演算結果が D-BUS を通ってレジスタ部，および特殊レジスタに転送される．

2.3　演算部の構成

演算部では図 2 のマイクロ命令の OP-フィールドで指定する演算の種類と，LG-フィールドで指定する演算長とに従って，入力データに演算を施す．

演算長は，8, 16, および 32 ビットであり，LG-フィールドの最上位ビットが“0”の場合は，LG-フィールドの下位 2 ビットにより指定される．最上位ビットが“1”の場合は，特殊レジスタ LGR (operation LenGth Register) の内容により指定される．

演算結果はマイクロ命令の MK-フィールドで指定されるマスク値との論理積を施された後，D-BUS に出力される．

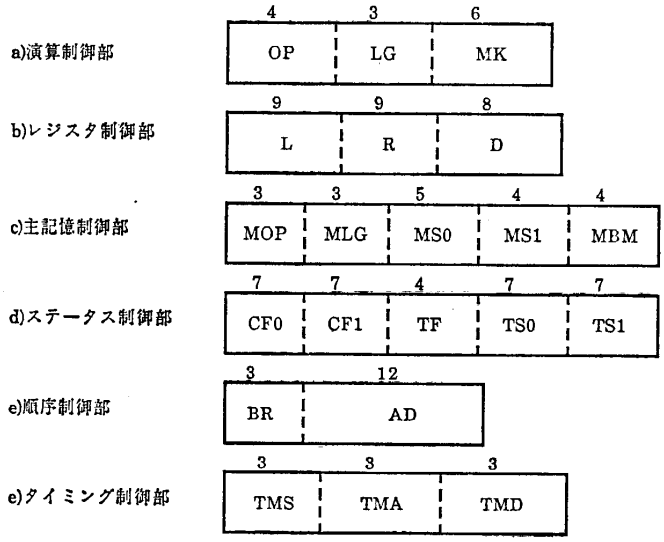

図 2　SVP のマイクロ命令
Fig. 2　Micro-instruction format of the SVP.

2.4　レジスタ部の構成

レジスタ部には，2 ポートの汎用レジスタ (WR: Work Register) と，特殊レジスタ (SR: Special Register) および間接レジスタ・ファイル (IRF: Indirect Register File) が装備されている．WR と SR は，マイクロ命令の L-フィールド，R-フィールドの最上位ビットが“1”のときに下位 8 ビットによって直接アクセスされ，その内容がそれぞれ L-BUS, R-BUS に出力される (最上位ビットが“0”の場合には 8 ビッ

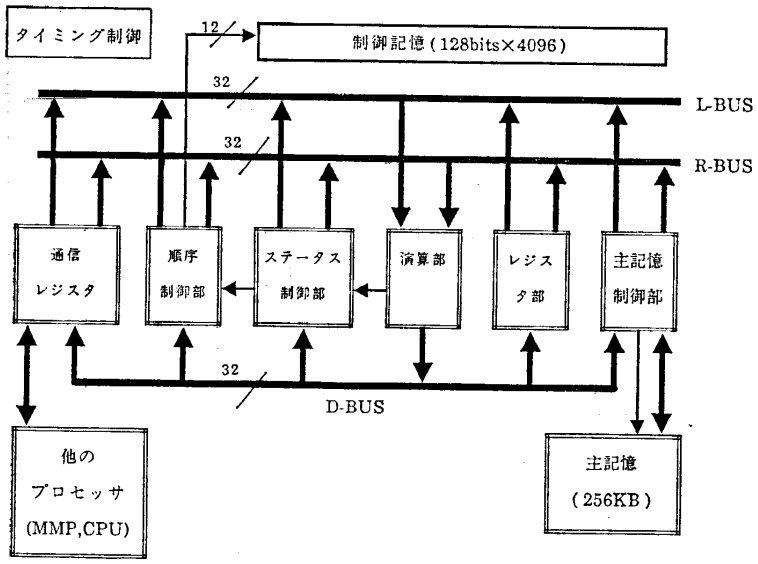

図 3 SVP の概要
Fig. 3 Configuration of the SVP.

トの定数値となる). D-フィールドでは D-BUS 上のデータを格納するレジスタを指定する.

WR はバイト単位でアクセスされ, 容量は 64B である. ハードウェア的には4バンク構成 (8ビット×4) となっているため, 演算長に合わせて, 連続したバイト・アドレスへのアクセスが可能である.

一方, IRF は, SR の一つである IAR (Indirect Address Register) の内容をアドレスとして, 間接アクセスのみが可能なレジスタである. SVP は1kバイト (256 エントリ×32 ビット) の IRF を8個装備している. IAR には自動増減機能が装備されており, IRF は高速なスタックやキューなどを容易に構成できる.

2.5 ステータス制御部の構成

システム内のステータス・レジスタは 98 ビット長である. マイクロ命令の CF0, および CF1 の二つのフィールドにより陽にステータス・レジスタのアドレスと操作を指定することによって, 1マイクロ命令で二つのステータスの設定が可能である.

マイクロ命令の TS0 および TS1 の二つのフィールドにより, 1マイクロ命令で同時に二つのステータス (論理変数 (A_1, A_0)) を読み出すことができる. TF-フィールドは4ビット用意されており, TS0 と TS1-フィールドで指示された二つの論理変数(A_1, A_0)に対する任意の論理関数 $f(A_1, A_0)$ (恒等的に 1, 0 の場合を含む) を指定できるため, 2方向分岐などで複雑な条件設定ができる.

2.6 順序制御部の構成

分岐形式はマイクロ命令のBR-フィールドで指定され, 次のようなものが用意されている.

(i) 単純分岐: マイクロ命令の AD-フィールドの値をマイクロプログラム・カウンタに設定する.

形式は if $f(A_1, A_0)$ then GOTO AD else NEXT

(ii) サブルーチン・コール: 復帰番地をアドレス・スタック (深さ 16) にプッシュし, AD-フィールドの値を MPC に設定する.

形式は if $f(A_1, A_0)$ then CALL AD else NEXT

(iii) 4方向分岐: 分岐条件として選択された二つのステータスの値を2ビットの整数とみなし, AD-フィールドの値との和をMPC に設定する.

形式は case (A_1, A_0) of /0/ GOTO AD,
/1/ GOTO AD+1, /2/ GOTO AD+2,
/3/ GOTO AD+3

(iv) テーブルを用いた間接分岐: 特殊レジスタである JTPR (Jump Table Pointer Register) の内容で JTB (Jump TaBle: 4k エントリ) を索引し, その値を MPC に設定する. 一般的には, 機械命令コードを JTPR に設定し, 各実行マイクロ・ルーチンへの分岐を行う.

(v) IJR (Indirect Jump Register) を用いた多方向分岐: IJR の内容と AD-フィールドの値との和を MPC に設定する.

形式は if $f(A_1, A_0)$ then case (IJR) of
/0/ GOTO AD, /1/ GOTO AD+1, ⋯,
/n/ GOTO AD+n, else NEXT

(vi) サブルーチンからの復帰: アドレス・スタックをポップ・アップしてその値を MPC に設定する.

形式は if $f(A_1, A_0)$ then RETURN else NEXT

2.7 主記憶制御部の構成

主記憶制御部は, 次章に示す機械命令コードやオペランドの高速フェッチを可能とするよう特殊なハードウェア設計がなされている.

(i) 高速アドレス生成機構

SVP には8個の MAR(Memory Address Register) が装備されており, この中から任意の2個のマイクロ

命令の MS0, MS1-フィールドにより選択する．主記憶アドレスはこれら二つのMARをハードウェアで直接加算して得られる．MS0-フィールドにより選択された MAR に対しては，自動増減を行うことができる．その際，加算または減算する値は，MLG-フィールドで指定されるアクセス長に等しい．アクセス長は，8, 16, 24, および 32 ビットのいずれかであり，MLGR (Memory access LenGth Register) による間接指定も可能である．

また，SVPには領域チェックのために，4組の UBR と LBR (Upper/Lower Boundary Register) が装備されている．これらのレジスタで，上限，下限を指定することにより，四つの領域に対するチェックを高速に行うことができる (MBM-フィールドで指定)．

（ii） 高速データ・アクセス機構

主記憶のアクセス動作は，MOP-フィールドにより次のように指定される．

（a） 読み出し：読み出したデータを MBR (Memory Buffer Register) に格納する．読み出し動作の間，他のハードウェアは動作しない (CPU フリーズ)．

（b） 連続読み出しの開始：指定されたアドレスで読み出したデータを MBR に格納した後，そのアドレスにアクセス長を加えた値をアドレスとして再度読み出し SRB (Sequential Read Buffer) に格納する．CPU フリーズは1回目の読み出し時にのみ起こり，2回目は，主記憶制御と他の演算・制御が並列に行われる．

（c） 連続読み出し：直前に実行した連続読み出しで格納された SRB の内容を MBR に格納するとともに，指定されたアドレスにアクセス長を加えた値をアドレスとして読み出し SRB に格納する．CPU フリーズは起こらない．

（d） 書き込み：MBR の内容を書き込む．CPU フリーズは起こらない．

3. SVP のマクロ・アーキテクチャ

3.1 システム記述用高級言語向き機械命令セットの設計指針

システム記述用高級言語のための機械命令を設計するにあたっては，効率の良いシステム・プログラムの実現と，多重プログラミング環境における信頼性の向上を目的として，以下に述べるような方針をとった．

（i） 機械命令の機能の強化

機械命令の機能を強化することによって，機械命令と高級言語との間のセマンティック・ギャップを狭め，多様なアドレッシング・モード（データ・アクセス用20種，分岐用6種）を装備し配列やレコードに対するアクセスにおけるオーバヘッドの軽減を図った．

これらのデータ・アクセスを実現するにあたってはSVP の8個の MAR を効率良く用いている (3.4 節参照)．また SVP の機械命令レベルにはレジスタという概念を導入せず，演算をすべて主記憶間での演算とすることにより，不必要な load-store 命令を省くようにした．なおレジスタを導入しなかったことによる主記憶アクセスのオーバヘッドを防ぐために

① 主記憶のローカル・データを一部 SVP の IRF にコピーし (3.4 節参照)

② SVP の連続読み出し機能および非同期書き込み機能 (2.7 節参照)

を多用している．

（ii） ソフトウェアのモジュール化支援

大規模なプログラムを作成するにあたっては，プログラムをモジュール化することが必要不可欠である．SVP ではモジュール化の概念を機械命令レベルで積極的にサポートしている．

SVP の機械語のプログラムは，複数個のモジュールからなる．各モジュールは一連のまとまったデータおよびそのデータを操作するサブルーチン群によって構成される．モジュール間のデータの受け渡しはサブルーチン間のパラメータの受け渡しとしてのみ可能である．この受け渡しを効率良く行うために値呼び，番地呼びの形式によるパラメータを用いたサブルーチン・コールを機械命令レベルで実現している．

図4に主記憶の構成を示す．主記憶を，(a)命令語

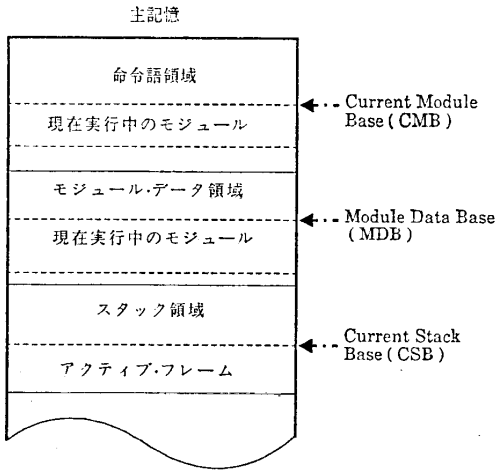

図 4　主記憶の構成
Fig. 4　Schema of the main memory.

領域，(b)個々のモジュールに固有のモジュール・データ領域，および(c)パラメータの受け渡しやサブルーチンの局所変数を保持するスタック領域，の三つに分割した．さらに各領域外への不当なデータ・アクセスの防止やプログラムの暴走の回避を図っている．

また，処理の高速化のために，実行中のプロセスのスタック領域の上の部分を SVP の間接レジスタ・ファイル (IRF) にコピーしている．

(iii) マルチプロセスの実行環境の整備

マルチプロセスの実行を効率良く行うためには，(a)コンテクスト・スイッチング時におけるオーバヘッドを軽減するとともに，(b)臨界領域(複数個のプロセスによって共有されるデータ)に対する操作を効率良く行わなければならない．

(a)に対しては，プロセスの生成/消滅を行う fork 命令/join 命令やプロセス間の同期を行うセマフォアの P/V 操作に対応する wait 命令/signal 命令を機械命令として装備した．

(b)に対しては，Hoare の提唱したモニタ[8]の概念を機械語に導入し，モジュールの一種として，モニタ・モジュールという特殊なモジュールを用意した．各モニタ・モジュールには複数個のプロセスに共有されるデータ，そのデータに対して施される操作，およびモニタ・ロックという名のセマフォア変数を割り当てている．このモニタ・ロックに対して前述した wait 命令や signal 命令を施すことにより，臨界領域に対するプロセスの相互排斥が実現できる．

3.2 機械命令の形式

図5に SVP の機械命令の形式を示す．機械命令の固定長部分は命令コード (OP) とオペランド長の指定フィールド (L) および分岐条件指定フィールド (MOD) からなる．

機械命令の可変長部分はいくつかのオペランド指定部からなる．オペランド指定部の個数はほとんどの命令では定まっているが，call 命令や fork 命令ではパラメータの数によって可変である．オペランド指定部はアドレッシング・モードを指定するフィールド (M) とアドレス指定フィールド (N) から成っており，図5に示すように8種類の形式をとる．

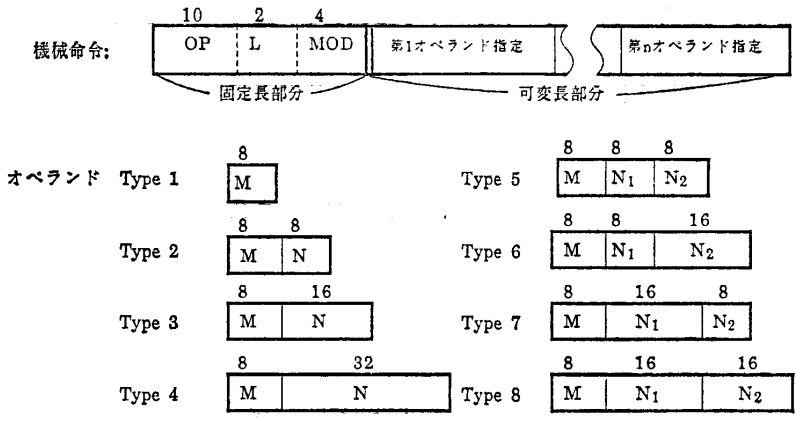

図5 機械命令の形式
Fig. 5 Format of the macro-instructions.

表1 データ・アクセス用アドレッシング・モード
Table 1 Addressing mode for accessing data.

モード番号	モード	注
0	N	i)
1	N	ii)
2	N	iii)
3	mod(N)	
4	mod(mod(N))	iv)
5	mod(mod(N))+	iv), v)
6	mod(mod(N))−	iv), vi)
7	mod(mod(mod(N)))	vii)
8	mod(mod(mod(N)))+	v), vii)
9	mod(mod(N_1)+N_2)	viii)
10	mod(mod(N_1)+mod(N_2))	viii)
11	stk(CSB+N)	
12	mod/ext/stk(stk(CSB+N))	iv)
13	mod/ext/stk(stk(CSB+N))+	iv), v)
14	mod/ext/stk(stk(CSB+N))−	iv), vi)
15	mod/ext/stk(stk(CSB+N_1)+N_2)	viii)
16	mod/ext/stk(stk(CSB+N_1)+stk(CSB+N_2))	viii)
17	mod/ext/stk(stk(CSB+N_1)+mod(CSB+N_2))	viii)
18	mod/ext/stk(mod/ext/stk(stk(CSB+N)))	vii)
19	mod/ext/stk(mod/ext/stk(stk(CSB+N)))+	v), vii)

注) mod：モジュール・データ
stk：スタック
ext：他のモジュールのモジュール・データ
CSB：Current Stack Base
i) 8-bit 定数
ii) 16-bit 定数
iii) 32-bit 定数
iv) 間接アクセス
v) ポインタの自動増加
vi) ポインタの自動減少
vii) 二重間接アクセス
viii) 二つのポインタの和によるアクセス

3.3 アドレッシング・モード

（i） データ・アクセス用アドレッシング・モード

このモードは図4の主記憶の構成を反映して，以下のように大別される（**表1**参照）．

（a） 定数アクセス：1バイト，2バイト，または4バイトの定数が指定できる（表1のモード番号 0～2）．

（b） モジュール・データ・アクセス：実行中のモジュールのモジュール・データのアドレスを直接，または間接に指定する．間接指定の場合はポインタ・データの自動増減（表1のモード5, 6），2個のポインタの和による指定（モード9, 10），2重間接指定（モード7, 8）など豊富なアクセス方法がある．

（c） スタック・アクセス：スタック領域に割り当てられるデータとしては，①サブルーチンへ渡されるパラメータ，②サブルーチンで動的に割り当てられる変数（ローカル・データ）の2種類がある．各サブルーチンには最大1kバイトの SF (Stack Frame) が割り当てられる．

①サブルーチンへのパラメータの渡し方としては，値呼びと番地呼びの2種類がある．番地呼びの場合，表1のモード番号12～19に示すように，そのパラメータが同じモジュールのモジュール・データ (mod) か，他のモジュールのモジュール・データ (ext) か，あるいはスタック上のデータ (stk) かによってアクセス方法が異なる．mod, ext, stk のいずれのアクセス方法を採るかについての情報はパラメータ自身にタグとして付加されている．

②ローカル・データへのアクセスは，サブルーチンが呼ばれた時点で自動的に割り当てられるもの（C言語における自動変数に対応する）に対するアクセスに相当する．アクセス・モードは表1のモード番号11～19に対応するが，パラメータへのアクセスと異なり，他のモジュール内データへのアクセスは行われない．

（ii） 分岐用アドレッシング・モード

このモードにはモジュール内分岐とモジュール外分岐の二つがある（**表2**参照）．モジュール内分岐/モジュール外分岐いずれの場合にも，直接指定（モード番号 0/モード番号3），スタック・データによる間接指定（モード番号1/モード番号4），モジュール・データによる間接指定（モード番号2/モード番号5）がある．モジュール外分岐の場合（モード番号 3～5），モジュール番号がロード時にしか定まらないため，モジュール・データ内のモジュール番号辞書を参照する必要がある．モジュール番号から実際のモジュールのアドレスを決定する方

表 2 分岐用アドレッシング・モード
Table 2 Addressing mode for sequence control.

モード番号	モ　ー　ド	注
0	CMB+N	i)
1	CMB+stk(CSB+N)	i)
2	CMB+mod(N)	i)
3	MBT(mod(N_1))+N_2	ii)
4	MBT (mod (N_1))+stk (CSB+N_2)	ii)
5	MBT (mod(N_1))+mod(N_2)	ii)

注) mod：モジュール・データ
stk：スタック
MBT：各モジュールの命令語領域の先頭番地を保持
CMB：現在アクティブなモジュールの命令語領域の先頭番地
i) モジュール内分岐
ii) モジュール外分岐

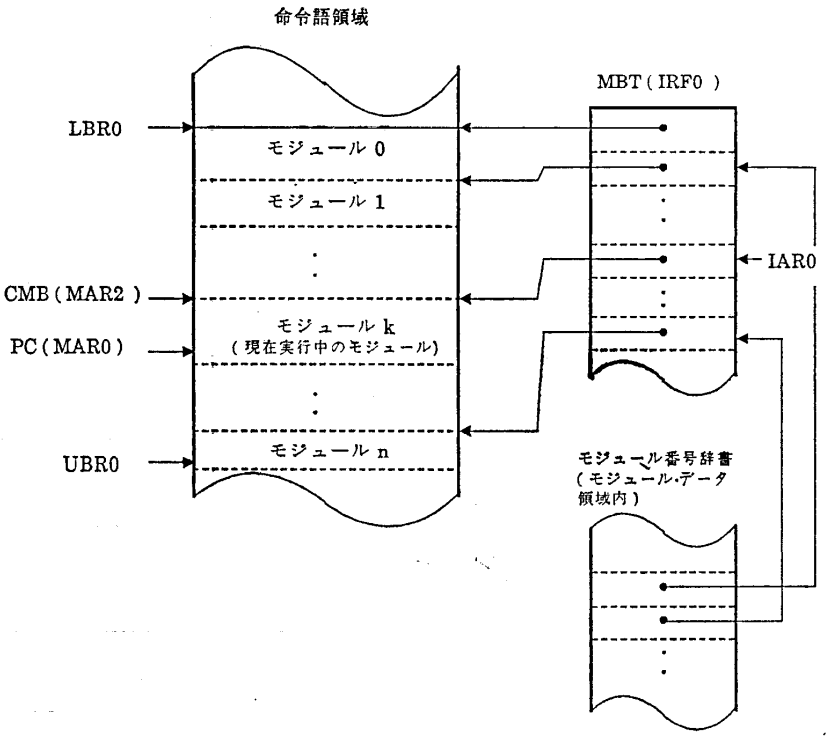

図 6 命令語領域の概要
Fig. 6 Schema of the instruction area.

法については次節3.4の(ⅰ)で述べる.

3.4 機械命令実行のための資源割り当て

本節では機械命令の処理に2章で述べた SVP のマイクロ・アーキテクチャがどのように活用されているかを記憶領域の各部に対応させて説明する.

(ⅰ) 命令語領域

図6に示すように, 命令語領域はモジュールごとに分割されており, 各モジュールの先頭アドレスは MBT (Module Base Table, IRF0 に割り付け) に格納されている. 実行中のモジュールの先頭アドレスは CMB (Current Module Base, MAR2 に割り付け) に格納され, 実行中の命令の CMB からの相対アドレスは PC (Program Counter, MAR0 に割り付け) に格納されている. 命令語領域の上, 下限は UBR0, LBR0 に設定されている. 表2のモジュール外分岐では割り当てられたモジュール番号を保持するモジュール番号辞書を参照して実際のモジュール番号を得る, また MBT を用いて主記憶上の番地を得る.

(ⅱ) モジュール・データ領域

モジュール・データ領域はモジュールごとに分割されており, 一つのモジュールに対して最大 64kB まで設けることができる. 各モジュール・データの先頭アドレスは MDBT (Module Data Base Table, IRF2 に割り付け) に格納されている. 実行中のモジュールに対応するモジュール・データ領域の先頭アドレスは MDB(Module Data Base, MAR4 に割り付け) に格納される. モジュール・データ領域の上, 下限は UBR1, LBR1 に設定される.

(ⅲ) スタック領域 (**図7**参照)

スタック領域は各プロセスに対して割り付けられる. 各プロセスに対応するスタック領域の先頭アドレスは SBT (Stack Base Table, IRF1 に割り付け) 格納されている. 実行中のプロセスに対応するスタック領域の先頭アドレスは SB (Stack Base, MAR3 に割り付け)に格納されている. 実行中のサブルーチンの SF の先頭アドレスは CSB (Current Stack Base MAR5 に割り付け) に格納されている. また, スタック領域の上, 下限は UBR2, LBR2 に設定されている.

サブルーチンが呼ばれた時点では, 図7のアクティブ・フレームに示すように SF には直前に起動されたサブルーチンの CSB, 復帰アドレス, およびパラメータが格納されている. サブルーチンの最初の命令でサ

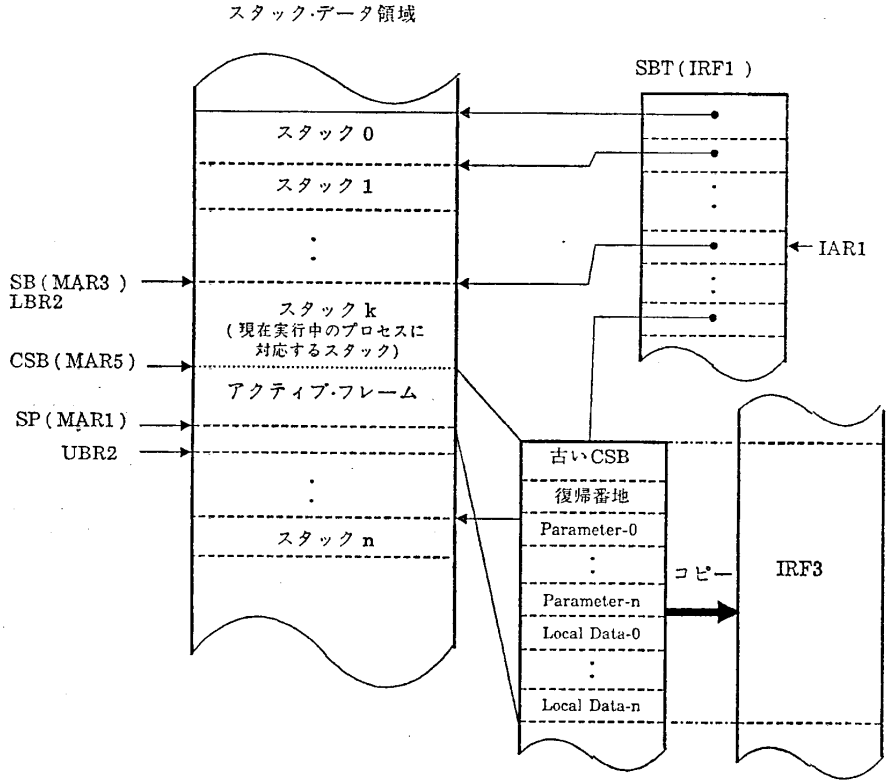

図7 スタック・データ領域の概要
Fig. 7 Schema of the stack area.

ブルーチンで使用されるローカル・データ用の領域が確保される．3.3 節でも述べたように，スタックの上部のデータは IRF3 にコピーされている．

3.5 マイクロプログラムによる機械命令の解釈実行過程

機械命令の解釈実行は以下のマイクロルーチンによってなされる．

（i） 命令フェッチ・ルーチンの処理

命令フェッチ・ルーチンでは，図5の機械命令の固定長部分の読み出し，割り込みの検出，および命令コードに対応する実行ルーチンへの分岐を行う．機械命令の読み出しは連続読み出し機能（2.7 節参照）を用いて先読みによる高速化を図っている．

フェッチ・ルーチンで読み出された固定長部分のうち，命令コード（OP）は JTPR（2.6 節参照）に，長さ指定フィールド（L）は LGR（2.3 節参照）に格納される．命令コードに対応する実行ルーチンのマイクロプログラムのアドレスは JTB に格納されており，テーブルを用いた間接分岐（2.6 節参照）により各実行ルーチンに分岐する．

（ii） オペランド参照ルーチン

オペランド参照ルーチンでの処理は，①連続読み出しにより，アドレッシング・モード(M)を読み出して IJR に格納する；②IJR による多方向分岐（2.6 節参照）を用いてアドレッシング・モードに対応するルーチンに分岐する；③連続読み出しにより，アドレス指定フィールド（N）を読み出す；④アドレス計算を行い，オペランドへのアクセスを行う；からなる．

スタック上のデータへアクセスする場合，現在実行中のサブルーチンの SF のデータを読み出すときには IRF3 上のデータを用いる．逆に実行中のサブルーチンの SF にデータを書き込むときは IRF3 および主記憶の両方にデータを書き込む．

命令フェッチ・ルーチンとオペランド参照ルーチンにおいては2個の MAR の和による主記憶アクセス機能を多用している．

（iii） 実行ルーチンの処理

例えば CALL 命令に対する処理は次のように行われる．①分岐条件を調べ，条件が偽ならば PC の値を調整してフェッチ・ルーチンへ分岐する．条件が真ならばオペランド参照ルーチンを呼んでパラメータの個数を得る；②パラメータの個数を用いて IRF3 にデータが納まるかどうか判断し，IRF3 にデータが納まらない場合は IRF3 の後半の 512 バイトのデータを IRF3 の前半に移動させる；③CSB をスタックおよび IRF3 に積む；④復帰番地を保存するためにスタック・トップのアドレスを退避する；⑤パラメータを番地呼び，あるいは値呼びに応じてスタックおよび IRF3 に積む；⑥分岐先アドレスをフェッチする．モジュール外分岐の場合は MDB を更新する；⑦復帰番地をスタックの④で退避していた番地に格納する；⑧PC にモジュール内アドレスを設定する．モジュール外分岐の場合は CMB を更新する；⑨フェッチ・ルーチンへ分岐する．

3.6 マクロ・アーキテクチャの考察

SVP ファイル・システムのプログラムの一部分を Pascal-like な高級言語で記述した例では，Pascal-like の高級言語8ステップが SVP の機械命令15ステップに相当していた．また SVP の機械命令の機能レベルと汎用マシンの代表的なアーキテクチャである IBM 370 アーキテクチャの機能レベルを比較するため，IBM370 の機械語で同じプログラムを記述したところ，SVP の1機械命令は IBM 370 の機械命令約3命令に対応しており，高級言語とのセマンティック・ギャップ[9]は大幅に狭められているといえる．これは，

（i） 機械命令レベル・アーキテクチャにレジスタという概念を導入せずにすべて主記憶間での演算としたため，不必要なデータ転送が省けたこと，

（ii） 複合分岐（演算を行って，その結果によって分岐する）命令を活用できたこと，

（iii） 豊富なアドレッシング・モードを活用すると，配列やレコードに対する処理が容易となったこと，

（iv） サブルーチンコールにおけるパラメータの受け渡しをすべて機械命令レベルで吸収したこと，

などに起因する．また（i）の特長は，SVP のマイクロ・アーキテクチャが2.7節で述べたように，連続主記憶読み出し機能や複数個の MAR（メモリ・アドレス・レジスタ）を装備していたために主記憶アクセスのオーバヘッドを吸収できたことによって初めて可能になったものである．

4. むすび

QA-2 のシステム管理プロセッサ SVP のマイクロ・アーキテクチャ並びにマクロ・アーキテクチャについて概説した．汎用計算機の機械命令と比較して SVP で採用した機械命令のレベルは高く（総命令ステップ数が約 1/3 になっている），モジュール構造支援やマル

チプロセスの実行環境を効率良く実現している．現在，図1に示したVAX-11/750で開発してきたQA-2のシステム管理機能をSVP上に移植中である．

謝辞 我々と共に，QA-2・SVPの設計・開発を行った中島浩氏（三菱電機(株)）に感謝いたします．

参考文献

1) 飯塚 肇：ユニバーサル・ホスト・プロセッサ，ダイナミック・アーキテクチャ，相磯，飯塚，坂村（編），*bit*・臨時増刊，Vol. 12, No. 10, pp. 1329-1350 (1980).
2) 北村俊明，中田登志之，柴山 潔，富田眞治，萩原 宏：ユニバーサル・ホスト計算機QA-2の低レベル並列処理方式，情報処理学会論文誌，Vol. 27, No. 4 (1986年4月掲載決定).
3) 柴山 潔，北村俊明，中田登志之，富田眞治，萩原 宏：ユニバーサル・ホスト計算機QA-2の高機能順序制御方式，情報処理学会論文誌（投稿中）.
4) 柴山 潔，富田眞治，萩原 宏：ユニバーサル・ホスト計算機QA-2による逐次型Prologマシンのエミュレーション，電子通信学会・技術研究報告，EC 85-52 (1985).
5) 中田登志之，柴山 潔，富田眞治，萩原 宏：QA-2を用いたLISPシステムの作成，情報処理学会・第30回全国大会講演論文集，7C-3 (1985).
6) 湯浅真治，玄 光均，中田登志之，富田眞治，萩原 宏：マイクロプログラム制御計算機QA-2による実時間色彩動画システムの開発-I・II，情報処理学会・第30回全国大会講演論文集，4J-9・10 (1985).
7) 金井達徳，中田登志之，富田眞治，萩原 宏：プログラムの解釈・実行とデータの操作・管理を分離したAPL計算機の構成，情報処理学会・第30回全国大会講演論文集，3C-1 (1985).
8) Hoare, C. A. R.: Monitors: An Operating System Structuring Concept, *Comm. ACM*, Vol. 17, No. 10, pp. 549-557 (1974).
9) Myers, G. J.: *Advances in Computer Architecture*, John Wiley & Sons, Massachusetts (1978).

（昭和60年5月16日受付）
（昭和60年10月17日採録）

中田登志之（正会員）

昭和32年生．昭和55年京都大学工学部情報工学科卒業．昭和60年同大学院博士課程修了．同年日本電気(株)入社．現在，同社C&Cシステム研究所に勤務．在学中は，Lispマシンや QA-2の開発に従事．電子通信学会会員．

北村 俊明（正会員）

昭和30年生．昭和53年京都大学工学部情報工学科卒業．昭和58年同大学院博士課程修了．同年富士通(株)入社，現在に至る．在学中，高級言語計算機の研究およびQA-2の開発に従事．電子通信学会，ACM, IEEE 各会員．

柴山 潔（正会員）

昭和26年生．昭和49年京都大学工学部情報工学科卒業．昭和54年同大学院博士課程単位修得退学．同年同大学工学部情報工学教室助手，現在に至る．京都大学工学博士．計算機システム，計算機アーキテクチャの研究に従事．電子通信学会，IEEE, ACM 各会員，ICOT・WG 委員．

富田 眞治（正会員）

昭和20年生．昭和43年京都大学工学部電子工学科卒業．昭和48年同大学院博士課程修了．この間，零交さ波による音声合成の研究に従事．京都大学工学博士．同年京都大学工学部情報工学教室助手．昭和53年同助教授，現在に至る．計算機アーキテクチャ，並列処理システムなどに興味をもつ．著書（分担執筆）「計算機ハードウェア実験」「VLSI コンピュータI」．電子通信学会，ACM, IEEE 各会員．ICOT・WG 委員．

萩原 宏（正会員）

大正15年生．昭和25年京都大学工学部電気工学科卒業．NHKを経て，昭和32年京都大学工学部助教授，昭和36年同教授，現在に至る．工学博士．情報理論，パルス通信，電子計算機などの研究に従事．昭和31年度稲田賞受賞．昭和50年本学会論文賞受賞．昭和56～58年度本学会副会長．著書「電子計算機通論1～3」「マイクロプログラミング」など．電子通信学会，ACM, IEEE 各会員．