ユニバーサル・ホスト計算機 QA-2 の高機能順序制御方式†

柴山 潔†† 北村俊明†*† 中田登志之†**
富田眞治†† 萩原宏††

QA-2は，低レベル並列処理機能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方式により，高速性と柔軟性を兼備したユニバーサル・ホスト計算機である．QA-2のマイクロ・アーキテクチャは，処理速度，システムの柔軟性，およびマイクロプログラムの生産性の向上を図るために，高機能かつ一様な並列演算機構，メモリ・アクセス機構および順序制御機構を採用し，図形・画像・信号データのリアルタイム処理や高級言語処理などの多様な応用に対して，柔軟に適応可能となっている．特に，QA-2の低レベル並列処理機構の高速性を生かし，しかもユニバーサル・ホスト計算機としての広範な問題適応能力を得るために，ハードウェア機構として様々な工夫を施した高機能順序制御方式を開発した．本論文では，QA-2の順序制御部 (SCU) のハードウェア構成について詳述し，その方式について簡単に評価を加える．

1. まえがき

我々は，低レベル並列処理機能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，高速性と柔軟性を兼備した実験用計算機：QA-2を開発した．QA-2は，図形・画像・信号データのリアルタイム処理や高級言語処理を主要な応用対象としたユニバーサル・ホスト計算機である[1]．

QA-2の設計思想は，「低レベル並列処理方式とマイクロプログラム制御方式の有機的結合」であり，QA-2には処理速度，システムの柔軟性，およびマイクロプログラムの生産性を飛躍的に高め得るように，高機能かつ一様な構造の並列演算機構，メモリ・アクセス機構，順序制御機構が採用されている．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とMMPは，図2に示した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制御される．SVPは，独立した32ビット・プロセッサである[2]．

同一構造で高機能な複数ALUによる低レベル並列処理機能とは，4個の可変長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で独立に制御され，それらが均一構造を持ちかつ大容量（6Kバイト）のレジスタ・ファイルを共有しながら動作する方式のことである．4個のALUは互いに独立な四つのオペランド群に対して並列演算を実行できるだけでなく，1個の演算結果を他のALU演算の入力オペランドとして使用するALU連鎖演算を行うことも可能である．

QA-2の低レベル並列処理機構（レジスタ・ALU部）については別論文[10]で述べているので，本論文ではユニバーサル・ホスト計算機として実現したQA-2の順序制御方式に焦点を絞り，詳述する．

2. ユニバーサル・ホスト計算機の順序制御方式

ユニバーサル・ホスト計算機におけるマイクロプログラムの順序制御機構に対して一般的に要求される機能は，高速性と柔軟性（問題適応性）につきる．特にQA-2では，低レベル並列処理方式を採用していることにより，これらの要求は次のような特殊な状況として具体化してきている．

(1) 大規模なユーザ・マイクロプログラミングにおける生産性を向上させるために採用したALU演算や順序制御機構を，ユーザが構造化マイクロプログラミングにおいて活用できなければならない．

(2) 問題適応性を高めるために，できるだけ自由度のある（ハードウェア構成による制限が少ない）分岐形式の選択，条件ステータスの抽出，および条件判定用論理関数の指定などが可能でなければならない．

† Highly Functional Sequence Control Mechanism of a Universal Host Computer QA-2 by KIYOSHI SHIBAYAMA, TOSHIAKI KITAMURA, TOSHIYUKI NAKATA, 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 富士通(株)本体事業部
(Main Frame Division of Computer Systems, Fujitsu, Ltd.)
** 現在 日本電気(株) C&C システム研究所
(C & C Systems Research Laboratories, NEC Corporation)

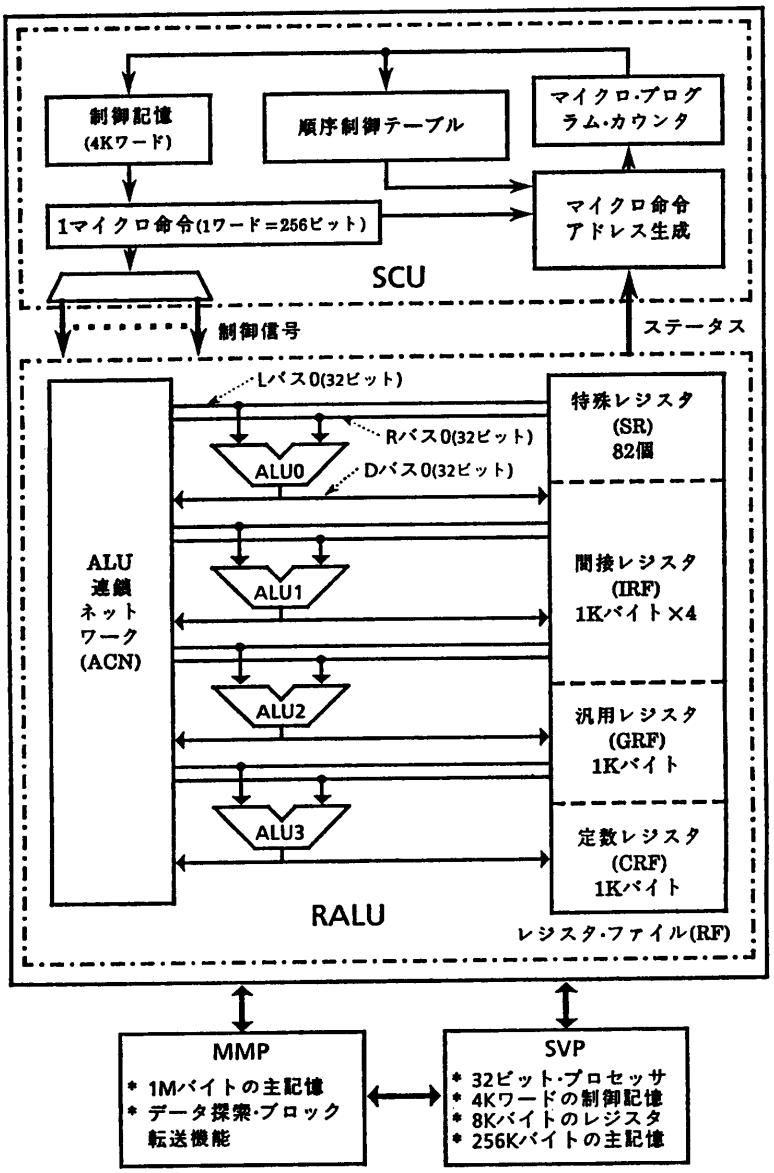

図 1 QA-2 のシステム構成
Fig. 1 System organization of the QA-2.

(3) 並列・連鎖 ALU 演算により，1マイクロ命令の実行で多数のステータスが生成されるので，これらを十分に活用できる条件分岐機能が実現されなければならない．

(4) レジスタ・ ALU 部と順序制御部の機能バランスを考えて，4個の ALU 演算フィールドができるだけ遊ばないようなマイクロプログラミングが可能でなければならない．

2.1 低レベル並列処理方式と順序制御機能

マイクロ命令内の並列処理度が多い QA-2 の場合には，各種の論理変数の組み合わせによる複雑な条件テストの出現頻度が増大するので，組織的な条件テスト部を構成しなければならない．

2方向分岐 (IF-THEN-ELSE 分岐) 方式としては，その制御に必要なマイクロ命令のフィールドが冗長になるのを避けて，ELSE 節を明示しない形式や ELSE 節がリテラル・フィールド (分岐先アドレス) を必要としない形式などが採用されることが多い．しかし，THEN 節と ELSE節 の分岐形式を同一構造にすることは，マイクロプログラムの生産性を向上させる効果もあり，QA-2 ではこの形式を採用している．

例えば，**図 3** に示したような2方向分岐とサブルーチン・コールとが混在する順序制御においては，これを QA-2 では1マイクロ・ステップで実現できるのに対して，“IF (X) THEN GOTO A ELSE NEXT ;” 形式という単純な2方向分岐機能しか持っていない計算機では，サブルーチン・コールが起動されるまでに3マイクロ・ステップを要する．

また，低レベル並列処理計算機の場合には，ステータス・レジスタの内容に応じた多方向分岐(CASE-OF 分岐) が効率の良いマイクロプログラムの実行に本質的な役割を果たす．**図 4** に示したように，QA-2 の場合には1マイクロ・ステップで終了するようなマイクロプログラムでも，単純な2方向分岐機能しか持たない計算機では4マイクロ・ステップが必要である．多方向分岐機能としては，ステータスの値をそのままマイクロ命令のリテラル・フィールドの値と連結したり加算したりしてマイクロプログラム・カウンタの値とする方式や，ステータスの値を用いた ALU 演算により直接マイクロプログラム・カウンタの内容を変更する方式などがあるが，これらの方式では分岐形式や分岐先の指定に対する制限が強くならざるを得ない．QA-2 では，ステータス・レジスタ内の任意ビットを切り出してマイクロプログラム・

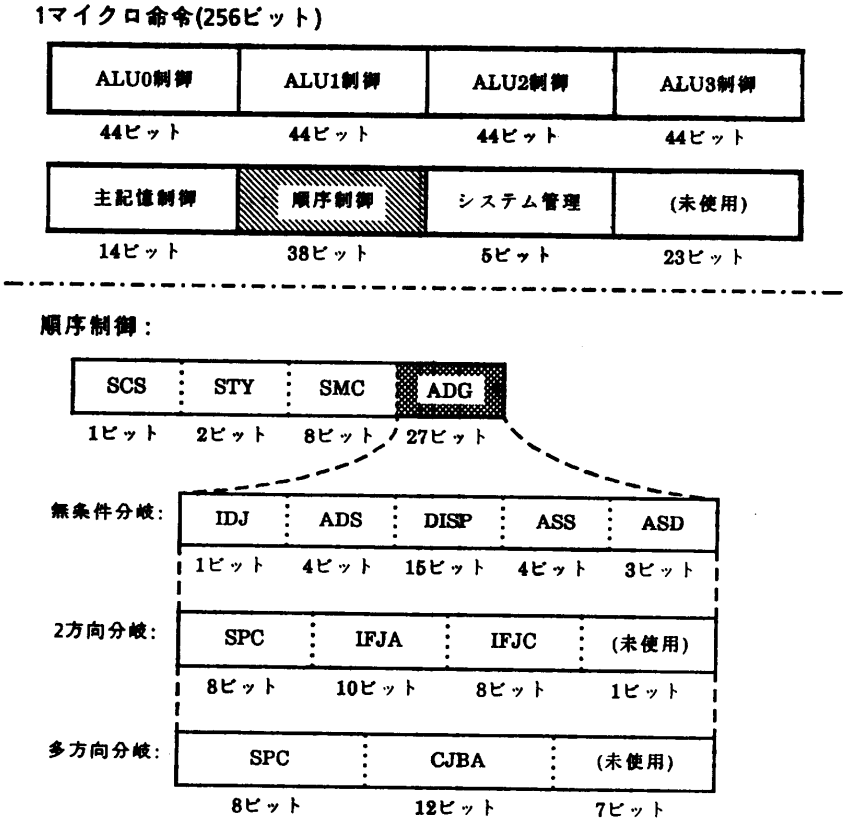

図 2 マイクロ命令形式
Fig. 2 Micro-instruction format.

カウンタの値の設定に使用できる機構を装備し，さらにディスパッチ・テーブルに格納する情報を豊富にして，各条件値に対する多様な分岐形式の指定を柔軟に行うことができるようになっている．

2.2 構造化マイクロプログラミング

ユニバーサル・ホスト計算機上で大規模マイクロプログラムを作成するためには，ソフトウェア工学で提唱された構造化手法を適用する必要がある[6),7)]．しかし，マイクロプログラミングの場合には，ハードウェアの直接的な高速制御機能を実現しなければならないので，構造化手法の適用がシステムの性能低下を招かないように注意する必要がある．したがって，構造化マイクロプログラミングのためには，相応の順序制御機構をハードウェアとして用意しなければならない[8)]．

また，マイクロプログラムはハードウェアをきめ細かく制御するものであり，通常のマシン命令プログラムと比較して，分岐命令の出現頻度が極めて高い．特に，水平形マイクロ命令形式で多数の機能装置を並列に制御する方式を採っている QA-2 の場合には，この傾向はより顕著になっている．高水準マイクロプログラム記述言語の利用は，マイクロプログラミングにおける生産性を高める一方式であるが，マイクロ・アーキテクチャが複雑で非均質な構造を持つ場合には，高水準言語を使用しても実行効率の良いマイクロプログラムの生成が困難になる．低レベル並列処理機

単純な順序制御方式の場合：

```
IF (A0) THEN GOTO L ELSE NEXT;
IF (A1) THEN GOTO L ELSE NEXT;
CALL S;
```

QA-2の場合：

```
IF f(A0,A1)=(A0|A1) THEN GOTO L
                    ELSE CALL S;
```

図 3 複雑な順序制御の指定例
Fig. 3 An example of composite sequence control.

単純な順序制御方式の場合：

```
IF (P) THEN GOTO L1 ELSE NEXT;
IF (Q) THEN GOTO L2 ELSE NEXT;
IF (R) THEN GOTO L3 ELSE NEXT;
GOTO A;
L1:
IF (Q) THEN GOTO L4 ELSE NEXT;
.....
L3:
CALL B;
.....
L4:
IF (R) THEN GOTO L5 ELSE NEXT;
.....
L5:
RETURN;
```

QA-2の場合：

```
CASE (P,Q,R) OF
        /000/ GOTO A
        /001/ CALL B
        .....
        /111/ RETURN;
```

図 4 多方向分岐の効果
Fig. 4 Effect of the multi-way branch.

表 1 順序制御部の仕様
Table 1 Specification of the sequence control unit.

項目	QA-2
①2方向分岐形式	IF $f(A_0, \cdots, A_7)$ THEN {GOTO A/CALL B/RETURN} ELSE {GOTO C/CALL D/RETURN};
②多方向分岐形式	最大256方向分岐が可能 CASE($A_0, \cdots, A_7$)OF /0/ {GOTO A/CALL B/RETURN} ⋮ /255/ {GOTO C/CALL D/RETURN};
③2方向分岐条件ステータス	最大8論理変数の自由な組み合わせ ($A_0, \cdots, A_7$)
④2方向分岐条件判定用論理関数: f	ユーザが定義可能
⑤多方向分岐条件ステータス	最大8論理変数の自由な組み合わせ ($A_0, \cdots, A_7$)
間接分岐	"GOTO" あるいは "CALL" が可能
順序制御テーブル	2方向分岐用 (1Kエントリ), 多方向分岐用 (4Kエントリ), 間接分岐用 (2Kエントリ)
制御記憶容量	4Kワード/256ビット
仮想制御記憶	デマンド・ページング方式 (1ページ=512ワード)
マイクロ・アドレス・スタック	512レベル×2本 (SVP による管理)
順序制御用マイクロ命令フィールド長	43ビット

構を実現している QA-2 の場合，レジスタ・ALU 部と同様に順序制御部のハードウェア構造も簡潔で分かり易くすることが，ユーザや高水準マイクロプログラム記述言語用コンパイラの負担を軽減するために重要である．

QA-2 の順序制御部 (SCU) では，その分岐形式として無条件分岐のほかに表1の①②に示すような IF-THEN-ELSE 形式の2方向分岐および CASE-OF 形式の多方向分岐をハードウェア機能として装備している．また，表1の③～⑤に示したように，条件分岐判定用論理関数はユーザが定義可能であり，その論理変数として最大8個のステータスを自由に組み合わせることができる．

これらの高度で複雑な条件判定や順序制御の機能を実現する制御論理は，高速アクセス可能でかつ書き換え可能な ECL・RAM チップによって実装されたテーブルを使用して実現している．この順序制御テーブルには，マイクロプログラム・アセンブラが翻訳したオブジェクト・データが実行に先立って格納される．ユーザは，制御記憶だけでなく，これらのテーブルを応用ごとに書き換えることができるので，条件分岐方式やステータスの制御方式をユーザ自身の応用に適した順序制御論理にすることができる．

QA-2 用マイクロプログラム・アセンブラを利用してマイクロプログラミングを行う際には，ユーザは各種順序制御テーブルの存在や容量をまったく意識する必要はなく，順序制御に関する煩雑なマイクロプログラミングから解放されている．高級言語風構文としての IF-THEN-ELSE 文や CASE-OF 文で記述されたマイクロプログラムの制御構造は，アセンブラが各種順序制御テーブルや制御記憶へのオブジェクト・プログラムとして翻訳し，編集する．これらの順序制御用ファシリティは主記憶をバックアップとして仮想化されているので，もし物理的容量を超えたオブジェクト・データであっても，SVP の管理下で動的にオブジェクト・データの入れ換えを行うことができる．この機能は，マイクロプログラムの制御記憶の書き換えのみが可能な従来のユニバーサル・ホスト計算機には装備されていない，大きな特長となっている．

3. 順序制御部のハードウェア構成

SCU は 4K ワードの制御記憶 (CS) と，各種の制御テーブルを実装したマイクロプログラムの順序制御部とから構成されている．

3.1 マイクロ命令のフェッチ機構

マイクロプログラムの高速化を図るために，マイクロ命令のフェッチと実行をパイプライン化する手法がある．しかし通常，制御記憶へのアクセス時間は ALU 演算時間（レジスタ参照時間を含む）と同程度のことが多いこと，マイクロプログラムには条件分岐が極めて多いこと，後続する命令で先行命令の結果を使うことが多い（データ依存性が高い）ことなどにより，マイクロプログラム制御計算機ではパイプラインのステージ数をあまり増やしても効果があがらない[8]．したがって QA-2 では，セーブド・モードとしてマイクロ命令のフェッチと実行という2ステージをパイプライン化した順序制御機構を採用した．

マイクロ命令（図2参照）の SCS フィールドによ

って，SCU におけるマイクロ命令のフェッチ・サイクルと RALU におけるマイクロ命令の実行サイクルとをオーバラップさせるか否かの選択を行う．セーブド・モードでは，両サイクルがオーバラップし，順序制御に使用されるステータス情報は以前に実行されたマイクロ命令によるものが使用される．このモードでは，現在実行中の ALU 演算の結果を使用することはできないが，SCU と RALU との間でのステータス転送に伴うオーバヘッドは生じない．カレント・モードでは，両サイクルはオーバラップせずに，RALU による演算実行を待って SCU が次に実行すべきマイクロ命令アドレスの計算を始める．このモードでは，そのマイクロ命令による ALU 演算結果を直ちにそのマイクロ命令による順序制御に使用することができる．

両モードを使い分けることにより，RALU と SCU 間で複数ステータスの授受を同期させる際に生じる順序制御のオーバヘッドを，極力避けることが可能である．

3.2 ステータス・セーブの制御方式（図5参照）

QA-2 が1マイクロ命令の実行のたびに更新するステータス情報は 96 種類にものぼり，主なものとして ALU 演算結果によるものや MMP の操作結果によるものなどがある．これらのステータス情報は 256 ビットに展開され，さらにその各ビットごとにマイクロ命令の SMC フィールドによって指定された多様な選択操作（save/set/reset/trigger/nop などの操作）を経てステータス・セーブ・レジスタ（SSR；256 ビット）に設定される．SMC フィールドはステータス・マスク・テーブル（SMT；(256×2) ビット×256 エントリ）の1エントリを選択するための SMT アドレスを指定する．1マイクロ命令の実行ごとに SMC フィールドに従って読み出される SMT の1エントリ(256×2 ビット）は，SSR の各1ビット情報に対する選択操作を2ビットで（4種類のうちから）指定する．また，SMT エントリの一部分は，ユーザが定義できるフリップ・フロップの操作，割り込みベクトルやマイクロ・アドレス・スタックのバンク選択，およびカウンタの増減操作などの指定にも使用しており，問題ごとに書き換え可能である．

3.3 マイクロプログラムの分岐形式

マイクロプログラムの分岐形式は，(A)無条件分岐と，(B)条件分岐（2方向分岐と多方向分岐）とに大別され，マイクロ命令（図2参照）の STY フィールドによって指定される．分岐形式のさらに細かい指定を行う ADG フィールドは，STY フィールドによる選択に応じて3形式のサブ・フィールド構成となる．ADG フィールドを使用して生成される順序制御情報としては，次に実行すべきマイクロ命令のアドレスの生成方法やアドレス情報の管理方法などがある．**図6** に各種制御テーブルによる順序制御の流れを示した．

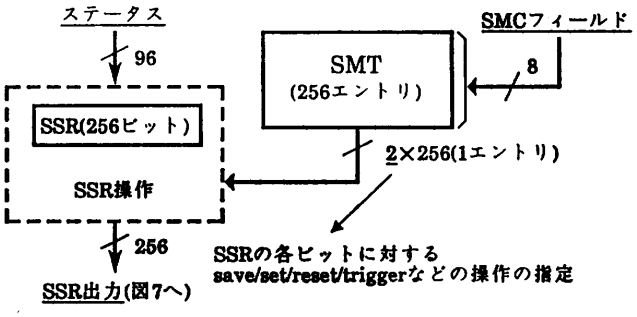

図 5 ステータス・セーブの制御方式
Fig. 5 Control mechanism for saving status.

（A） 無条件分岐

無条件分岐が指定されている場合は，図2に示したように，ADG フィールドは，さらに IDJ, ADS, DISP, ASS, ASD の各サブ・フィールドに分けられ，これらが直接順序制御情報として使用される（図6の①）．次に実行すべきマイクロ命令のアドレスは，基本的には DISP サブ・フィールドであるリテラル値と，ADS サブ・フィールドによって選択されたアドレス・セーブ・ファシリティ（マイクロプログラム・カウンタ：MPCR，割り込みアドレス・セーブ・レジスタ：ISVR，セーブ・レジスタ：SVR，アドレス・スタック：ASTK のいずれか）の内容とが加えられることにより生成される（図6の②）．例えば，ASTK が選択されるとマイクロ・サブルーチンからのリターン指定となる．ASS および ASD サブ・フィールドは，一度生成されたアドレス・データを退避したり，退避されていたアドレス・データを読み出したりする際に使用するアドレス・セーブ・ファシリティを選択する（図6の③）．例えば，ASD フィールドで ASTK が指定されると，マイクロ・サブルーチン・コールが実行されることになる．また，IDJ サブ・フィールドが“1”であれば，生成されていたデータをアドレスとしてさらに間接分岐テーブル（IJT；2K エントリ）から読み出されたものが最終的なアドレス・データとなる間接分岐が行われる（図6の④）．

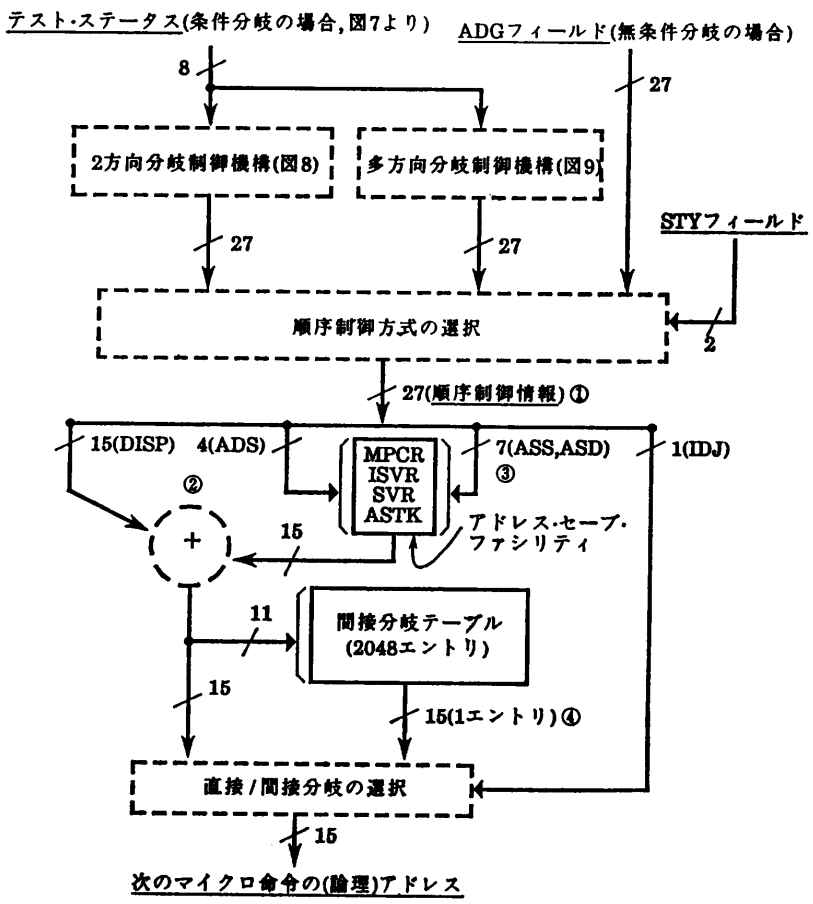

図 6　順序制御の処理フロー
Fig. 6　Flowchart of the sequence control.

(B) 条 件 分 岐

2方向分岐や多方向分岐が指定された場合，図2の ADG フィールドはテスト・ステータスの選択やテスト条件の設定を行うための各種順序制御テーブルのアドレス指定などに使用され，無条件分岐時のように ADG フィールドが直接順序制御情報の生成に使用されるわけではない．条件分岐の場合は，いくつかの順序制御テーブルが駆動されて初めて，無条件分岐時の ADG フィールドと同じサブ・フィールド構成の順序制御情報が得られる（図6の①）．この順序制御情報を用いて最終的なアドレス・データが生成されるまでの過程は，無条件分岐の場合と同じである．次に，条件分岐における順序制御情報の生成過程について述べる．

3.4 テスト・ステータスの選択方式（図7参照）

条件分岐の場合の条件テストに使うテスト・ステータスは一度に最大8個であり，SSR の 256 ビットのうちから自由に選択し，組み合わせることができる．特に，低レベル並列処理機構（複数個の ALU 演算）から発生する多種・多数のステータスを拾い上げ，順序制御用に使用できることが必要である．これらのテスト・ステータスはマイクロプログラムの条件文における論理変数：A_i $(i=0,1,\cdots,7)$ とみなすことができる．SSR の1ビットと A_i との対応付けは，ステータス・パック・テーブル（SPT；256 エントリ）からマイクロ命令の SPC フィールドによって選ばれた1エントリによって行う．1エントリには，8個の8ビット・データが格納され，各々が SSR のアドレスを示している．

3.5 条件分岐の制御機構

（1） 2方向分岐（図8参照）

マイクロプログラムの順序制御における2方向分岐は，高級マイクロプログラミング言語による記述形式である次のような "IF-THEN-ELSE" 文に対応するものである．

IF $f((A_0),(A_1),\cdots,(A_i))$ THEN (分岐操作指定)
ELSE (分岐操作指定)；
(ただし，f はユーザが定義した論理関数，(A_i) は論理変数 A_i の値である．$0 \leqq i \leqq 7$)

この文の THEN 節で指定する分岐操作と ELSE 節で指定する分岐操作にはいずれも無条件分岐と同様に GOTO 文，CALL 文，RETURN 文が何の制約もなく指定できるので，ユーザは構造化された制御構造を持つマイクロプログラムを記述することが可能である．

テスト条件である論理関数 f は，実際には真偽テーブル（TFT；256 エントリ）のうち1エントリがマイクロ命令の IFJC サブ・フィールドによって選択される．1エントリは 256 ビットであり，テスト・ステータス (A_0)〜(A_7) の値に対する真理値表となっている．

決定された f の値によって，THEN 節あるいは ELSE 節のいずれかに対応する IF 分岐テーブル

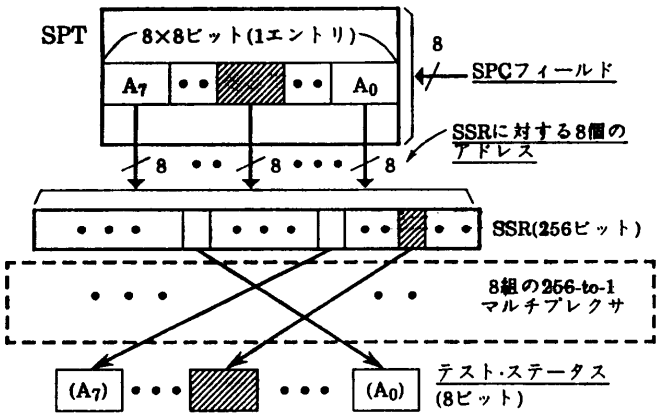

図 7 テスト・ステータスの選択方式
Fig. 7 Control mechanism for selecting status information.

(IFJT；1024 エントリ) が駆動され，マイクロ命令の IFJA サブ・フィールドの値をそのアドレスとして，1エントリ (27 ビット) が順序制御情報として読み出される．この順序制御情報は無条件分岐の場合のマイクロ命令の ADG フィールドとまったく同じサブ・フィールド構成となっている．したがって，例えば IF 文に CALL 文や RETURN 文を混ぜたような高度な条件分岐の指定が可能となっている．

(2) 多方向分岐 (図9参照)

QA-2 における多方向分岐の高級マイクロプログラミング言語による記述形式は次のような "CASE-OF" 文に対応するものである．

CASE($A_0, A_1, \cdots, A_i$)OF
/0/ (分岐操作指定)…
…/$2^{i+1}-1$/ (分岐操作指定)；
(ただし，A_i は論理変数 A_i の値である．$0 \leqq i \leqq 7$)

テスト条件である A_0〜A_i は最大8ビットの論理値列であり，マイクロ命令の CJBA サブ・フィールドのリテラル値に加算されて，CASE 分岐テーブル (CJT；4096 エントリ) の1エントリ (27ビット) を選択する．したがって，CJBA サブ・フィールドの値は CJT のベース・アドレスであり，A_i の論理値列はそのディスプレースメントである．CJT から読み出されるものは無条件分岐の場合の ADG フィールドと同じ順序制御情報であり，2方向分岐と同様に多方向分岐の場合の分岐操作として CALL 文や RETURN 文も書くことができる．

3.6 仮想制御記憶方式

制御記憶 (CS) は 4K ビットの MOS スタティック RAM チップから成り，現在の実装容量は 4K ワードである．また，CS は1ページ=512 ワードでページ化され，15ビットの論理アドレスによる仮想空間を構成している．CS のバックアップ記憶は主記憶 (MM) であり，QA-2 の機能部分の一つであるシステム管理プロセッサ SVP が，この仮想制御記憶方式を管理する．制御記憶や制御テーブルの仮想化により，大規模ユーザ・マイクロプログラミングが可能となっている．SVP は，仮想制御方式のページ入れ換えアルゴリズムを個々の問題に応じてダイナミックに変えることができる．CS や SCU 内の他の制御テーブルのダイナミックな切り換えの制御も SVP に任されている．

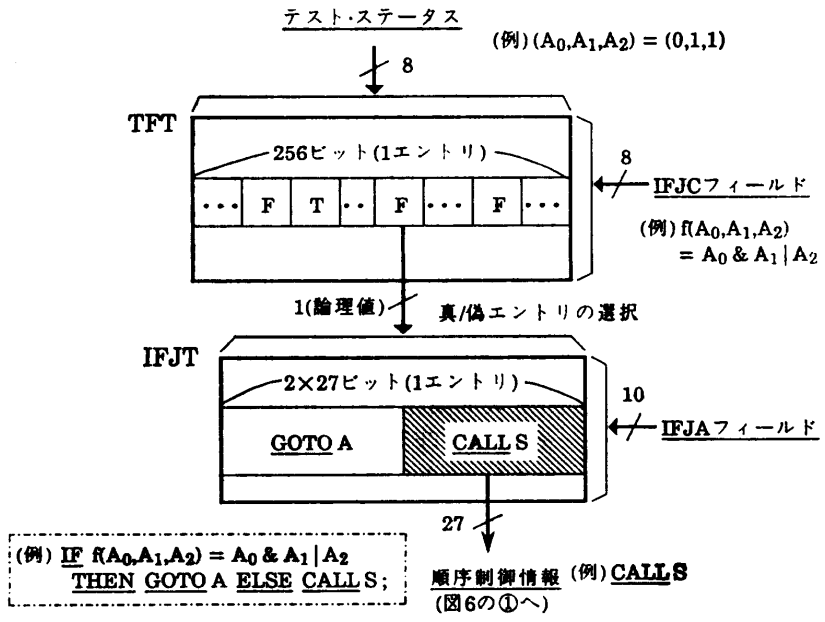

図 8 2方向分岐の制御機構
Fig. 8 Control mechanism for the two-way branch.

4. QA-2 の順序制御方式の評価

QA-2 の順序制御方式の有効性を確認するために，QA-2 を

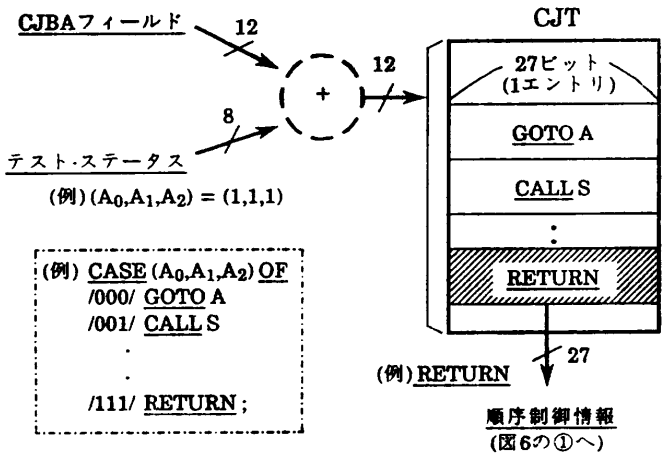

図 9 多方向分岐の制御機構
Fig. 9 Control mechanism for the multi-way branch.

実際の問題に応用し，動的な性能評価を行った．例題としては，処理対象に明示的な並列性のある応用として，(1)スキャンライン・アルゴリズムによる3次元グラフィックス，また明示的に並列性を有しない応用として，(2)逐次型 Prolog のインタプリタ，(3) Lisp のインタプリタ，を選んだ．いずれの応用においても，QA-2 のマイクロ・アーキテクチャについて様々な観点から定量的な評価を加えたが，本章では QA-2 の順序制御機能の効果の概要を述べるにとどめ，詳細については別稿[3)~5)]などに譲る．

各応用における QA-2 の代表的な特性を抽出するために，(1)では30個の立方体をランダムに表示する場合の可視部分の決定ルーチン，(2)では30個の要素から成るリストを反転するプログラム，(3)では「tarai-4」のベンチマーク・プログラムを，それぞれ具体的な問題として選んだ．

まず，現状の順序制御部を "IF (X) THEN GOTO A ELSE NEXT;" の単純な分岐形式（1個のステータスの値による2方向分岐）しか持たない構成にした場合に，各例題プログラムでどれだけの性能低下が生じるのかを調べた．表2に示したように，3次元グラフィックスや Prolog インタプリタでは，約40%も性能が低下する．これらの応用に比べて Lisp インタプリタでは，Lisp の基本関数に対応するマイクロ・ルーチンをプログラム中に展開して埋めこむ方式によって高速化を図ったために，マイクロプログラムのモジュール化がなされず，24%の性能低下にとどまっている．しかし，1マイクロ命令当たりの平均 ALU 使用個数が多い Prolog インタプリタの場合には，それがおよそ1ALU分も低下する（平均 ALU 使用個数が約3個から約2個になる）ことが判明した．これは，ステータスを生成するレジスタ・ALU 部とそのステータスを使用する順序制御部との機能バランスが崩れて，低レベル並列処理機能が殺されてしまっていることを示している．

Prolog や Lisp のインタプリタは，個々のデータにタグ・フィールドを付加したデータ構造を基本とする直接実行型高級言語計算機のエミュレータとみなすことができる．これらの応用では，表2からも分かるように，データそのものに明示的な並列性を持っている3次元グラフィックスよりも，条件分岐命令が出現する割合が多い．さらにこれらの応用においては，カレント・モード（そのマイクロ命令実行によって生成されたステータスを直ちに使用して順序制御を行うモード）による条件分岐の割合が多い（表2参照）．このモードの分岐では，レジスタ・ALU 部による並列演算と SCU による順序制御機能がオーバラップされない．しかし，このモードによる分岐が多い理由は，生成されたステータスをセーブする必要がなく直ちに順序制御に使うからであり，QA-2 の並列演算機能と順序制御機能のいずれもが処理のあい路になることなく，機能バランスはとれていると言える．

表 2 順序制御部の動的評価
Table 2 Run-time statistics of the sequence control unit.

応用問題	現状の順序制御方式			現状の順序制御方式での総実行命令数に対する，単純な順序制御方式にした場合† の総実行命令数の相対比
	総実行命令数に占める条件分岐命令の割合 (%)	総実行条件分岐命令数に占める2方向分岐命令の割合 (%)	総実行条件分岐命令数に占めるカレント・モード分岐命令の割合 (%)	
(1) 3次元グラフィックス	34	72	68	1.43
(2) Prolog インタプリタ	41	62	84	1.40
(3) Lisp インタプリタ	43	69	94	1.24

† "IF (X) THEN GOTO A ELSE NEXT;" 形式の2方向分岐機能しかない場合

タグ・アーキテクチャを採っている Prolog/Lisp マシンのエミュレーションの場合には，実行時に行われるタグの検査において，多数の（最大8個）ステータスによる多方向分岐機構が有効に機能している．またタグ・フィールドの各ビットは，既にそれぞれ独立した論理値としての意味を持っているので，マイクロプログラミング時にそれを論理変数として組み合わせ，条件判定用論理関数式とする操作は冗長で煩わしい．したがって，これらの応用において2方向分岐の場合に使用された条件判定用論理関数にはあまり複雑なものは無い．

2方向分岐機能のハードウェア規模が割合大きいことを考慮すると，あまり利用されることのない2方向分岐機能のハードウェア化は損である．したがって，マイクロプログラム・コンパイラに論理関数の展開（場合分け）を分担させて，ハードウェアとしては CASE-OF 分岐だけをサポートする構成方式も考えられる．（例えば，2方向分岐命令 "IF X & Y THEN GOTO A ELSE NEXT ;" を，多方向分岐命令 "CASE (X, Y) OF/00/NEXT/01/NEXT/10/NEXT/11/GOTO A ;" に変換してしまう．）しかし，この方式ではディスパッチ・テーブルがすぐにあふれるおそれがある．ハードウェアとソフトウェアのトレードオフに関連する評価については，さらに考察を加える必要がある．

実測によると，QA-2 のマイクロ命令サイクルは，無条件分岐命令やセーブド・モードの条件分岐命令の場合には最小で約 600 ナノ秒である．この時，RALU における演算実行サイクルと SCU における順序制御サイクルはほぼ同じ時間を要し，完全にオーバラップしている．また，カレント・モードの条件分岐命令の場合には，RALU の演算実行サイクルが終了してから（ステータスが確定してから），SCU の順序制御サイクルが始まるので，マイクロ命令サイクルは最小約 1,200 ナノ秒となっている．

RALU と SCU とは1マイクロ命令サイクル中に2回，同期をとる必要がある．RALU から SCU へステータスを渡す場合と，SCU から RALU へマイクロ命令を渡す場合である．これらの同期はシェーク・ハンド方式で行っており，大規模ハードウェア構成による信号遅延とが合わさって，かなりのオーバヘッド（実測によると約 200 ナノ秒）となっている．ハードウェア実装方式を改良し，RALU と SCU を同一のクロックで制御する完全な同期制御方式を採用すれば，このオーバヘッドをなくすことができる．この場合にも RALU による低レベル並列処理機能と，SCU による順序制御機能との機能バランスはとれ，現在の QA-2 の方式をほとんど変更することなく適用できるものと考えている．

5. むすび

本論文では，QA-2 の順序制御方式の特徴と実現方式について述べた．QA-2 の応用としては，逐次型 Prolog マシンや LISP マシンのエミュレーション[3),4)]，および3次元図形表示システムへの応用[5)]などが既に行われている．種々の応用に対して，QA-2 の高機能順序制御方式が良く適合することを確かめている．

また我々は，高級言語Cで記述したプログラムを直接 QA-2 のマイクロプログラムへ変換するコンパイラを開発しており，その場合にも QA-2 で実現している構造化マイクロプログラミング方式が有効に機能している．例えば，条件分岐とマイクロ・サブルーチンとを機能的に組み合わせることが可能である点などは，コンパイル時における最適化戦略を構成するための有効な道具として利用されている．

現在では，QA-2 のハードウェア実装時には入手できなかった種々の高機能ビルディング・ブロック（例えば，Weitek 社の浮動小数点演算器や AMD 社の Am 293XX シリーズなど）が利用できるようになってきた．これらの LSI を有効に利用することによって，QA-2 は多数のユーザの要求を満たし得る次世代の高性能ワークステーション[9)]となるであろう．

しかし，現状の QA-2 のハードウェア規模はかなり大きい（使用 IC 数は 22,000 個）ので，多様な応用実験を行って，ソフトウェア／ファームウェア／ハードウェアのトレードオフの見直しを図る必要もあろう．

謝辞 我々とともに，QA-2・SCU の設計・開発を行った河村武司氏（松下電器(株)）に感謝いたします．

参考文献

1) 中田登志之，北村俊明，柴山 潔，富田眞治，萩原 宏：低レベル並列処理機能を有するマイクロプログラム制御計算機 QA-2 の開発，情報処理学会「アーキテクチャワークショップインジャパン '84」シンポジウム講演論文集 (1984).
2) 中田登志之，北村俊明，柴山 潔，富田眞治，萩原 宏：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，情報処理学会論文誌，Vol. 27, No. 3, pp. 356-364 (1986).
3) 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，電子通信学会電子計算機研究会研究報告，EC 85-52, pp. 1-12 (1985).
4) 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，第 30 回情報処理学会全国大会論文集，7C-3 (1985).
5) 湯浅真治，中田登志之，新實治男，富田眞治，萩原 宏：低レベル並列処理計算機による 3 次元色彩図形表示処理，情報処理学会論文誌，Vol. 27, No. 6, pp. 630-638 (1986).
6) Jones, L. H.: Instruction Sequencing in Microprogrammed Computers, *AFIPS Conf. Proc. of NCC*, Vol. 44, pp. 91-98 (1975).
7) Davidson, S. and Shriver, B. D.: An Overview of Firmware Engineering, *Computer*, Vol. 11, No. 5, pp. 21-34 (1978).
8) 富田眞治：処理装置の構成，VLSI コンピュータ I，元岡 達(編)，岩波書店，pp. 12-172 (1984).
9) Tomita, S., Shibayama, K., Kitamura, T., Nakata, T. and Hagiwara, H.: A User-Microprogrammable Local Host Computer with Low-Level Parallelism, *IEEE Conf. Proc. of the 10th Annual Int. Symp. on Comput. Architecture*, pp. 151-157 (1983).
10) 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, pp. 445-453 (1986).

(昭和 61 年 1 月 8 日受付)
(昭和 61 年 7 月 16 日採録)

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．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．

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，QA-2 の開発に従事．電子通信学会，ACM, IEEE 各会員．

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，QA-2 の開発に従事．電子通信学会会員．

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．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．