低レベル並列処理計算機のためのマイクロプログラム最適化方式†

齊藤雅彦††* 新實治男††
柴山潔†† 萩原宏††

低レベル並列処理（レジスタ-ALU レベルの並列処理）機能を有する計算機においては，複数個の ALU を効率良く動作させるというマイクロプログラムの最適化処理が重要である．本論文では，低レベル並列処理計算機のマイクロプログラムに最適化処理を施すシステムについて実験的に考察した．このシステムでは，「トレース・スケジューリング法」と呼ばれる広域的最適化手法を中心とし，これを局所的最適化処理と組み合わせることにより，マイクロプログラム全体に渡って最適化処理を適用する．本論文では，我々自身が開発した低レベル並列処理計算機 QA-2 を中心とする種々の低レベル並列処理計算機に対して，(a)マイクロプログラムの動作・構成を変化させた場合；(b)計算機のマイクロ・アーキテクチャを変化させた場合；の2種類について最適化処理の効果を調べた．結果として，(a)プログラム中のマイクロ命令の実行回数に偏りが大きい場合；(b)計算機のマイクロ・アーキテクチャが豊富でセグメント（分岐点／合流点から次の分岐点／合流点まで）長を短くできる場合；において，広域的最適化処理（トレース・スケジューリング法）の効果が高いことが判明した．また，このマイクロプログラム最適化方式の適用によって，3～10% 程度の動的マイクロ命令数の減少が得られた．

1. はじめに

低レベル並列処理とは，レジスタ-ALU（演算装置）レベルの並列処理であり，マイクロプログラム制御方式と組み合わせることにより柔軟性と高速性を兼備した計算機アーキテクチャを提供できる技術である．しかし，低レベル並列処理方式による計算機は並列に動作可能な機能部分を数多く装備するため，それを制御するマイクロプログラムの作成が特に困難な作業となる．また，マイクロプログラムは計算機ハードウェアと直接対応するものであるため，その実行効率に対する要求もかなり厳しい．この意味で，マイクロプログラムの効率的な作成とその最適化は低レベル並列処理計算機において特に重要な課題である．

我々は，「低レベル並列処理方式とダイナミック・マイクロプログラム制御方式との有機的結合」を基本とする低レベル並列処理計算機 QA-2[1),2)] を開発している．QA-2 は柔軟性を有した高速な計算機システムをユーザに提供しており，図形処理[3)]や高級言語処理[4)]など多様な応用分野で活用されている．また，QA-2 におけるマイクロプログラム作成システムとして，C言語プログラムを直接マイクロプログラムへ翻訳するマイクロプログラム・コンパイラ[5)]を開発している．

これらの実験結果から低レベル並列処理方式においては，

（1） 複数個の ALU の同時制御が可能となることによる，ALU 演算の同一マイクロ命令への集約の可能性；

（2） ALU が複数個存在することによるステータス制御の増加；

という2つの要因によって，セグメント（マイクロプログラムの分岐点／合流点から次の分岐点／合流点まで）の長さが極端に短くなる（マイクロ命令数にして1～3程度）ことが分かっている．このため，セグメント間に渡る最適化（広域的最適化）とセグメント内部の最適化（局所的最適化）とを繰り返して実行することによって，マイクロプログラム全体に渡る最適化を行う必要が生じた．

我々はこの問題を解決するため，「トレース・スケジューリング法[8),9)]」と呼ばれる広域的最適化手法を改良し，QA-2 マイクロプログラムに最適化処理を施すシステムを開発した．さらに，この最適化方式を適用できる計算機を QA-2 から，より一般的な低レベル並列処理計算機に拡張し，これらに対するこの最適化方式の効果を調べている．

† A Microprogram Optimization Technique for Low-Level Parallel Computers by MASAHIKO SAITOH, HARUO NIIMI, KIYOSHI SHIBAYAMA 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 (株)日立製作所 Hitachi, Ltd.

2. トレース・スケジューリング法の改良

効率の良い目的マイクロプログラムを生成するための最適化技術としての広域的最適化法には，(i)セグメント間のマイクロ命令移動の可能性を静的に調べる方式（例えば，文献6)）と，(ii)マイクロプログラムの実行トレースによってセグメントの実行頻度を求め，それが最も高いセグメントを優先的に最適化する方式，とがある[7]．トレース・スケジューリング法は，(ii)の方式を低レベル並列処理計算機に適用した実例として注目されている．

トレース・スケジューリング法は，J. A. Fisher らによって開発された低レベル並列処理計算機向きの最適化アルゴリズムであり，演算の省略・共有ではなく，各演算を複数個の ALU に埋め込むという最適化処理を実行する．トレース・スケジューリング法は図1のように，(a)フロー・グラフ生成；(b)最も実行頻度の高いマイクロ命令系列（メイン・トレース）の選択；(c)メイン・トレース内での最適化処理；(d)最適化処理に伴うメイン・トレース外への修正；の4ステップから構成される．

しかし，この手法をそのまま使用して最適化処理を行うと，第4ステップにおいて，メイン・トレース外への修正処理だけではなく，メイン・トレース内部をも修正しなければならないことがある．例えば，図2(a)のように，マイクロ操作集合 MO_4 が MO_2 と同時に実行可能であり，MO_4 を上方に移動することを考える．この場合，MO_3 中に存在した分岐点は修正され，MO_1 の直後に移動されなければならない（図2(b)参照）．また，分岐点移動では，条件分岐を移動するだけでなく，条件分岐のステータスを設定する分岐条件計算を転記しておかなければならない．すなわち，本来の分岐条件計算の演算（MO_3）は元の位置に置いたままにし，その演算の出力をレジスタに書き込まないように修正した演算（$MO_{3'}$）を上方に転記する必要がある．このような場合，メイン・トレース上の演算回数の増加が生じる．トレース・スケジューリング法による最適化処理においては，このようなメイン・トレース上における演算回数の増加を，次のような理由によって，できる限り避ける必要がある．(i)実行頻度の高いメイン・トレース上での演算の増加であるため，実行マイクロ命令数の増加を引き起こす可能性が高い．(ii)メイン・トレース上での広域的最適化処理（マイクロ操作のコンパクション）を実行した後の演算処理フローによって，メイン・トレースに変更を加えるのであれば，最初からフロー・グラフを対象として，後の修正はメイン・トレース外だけで済ませるようなコンパクションを行ったほうが，最適化されたプログラムの実行効率を高め，さらに最適化処理の実行時間を短縮できる．

なお，複数個のマイクロ操作を1マイクロ命令にコンパクションすることのできる低レベル並列処理計算

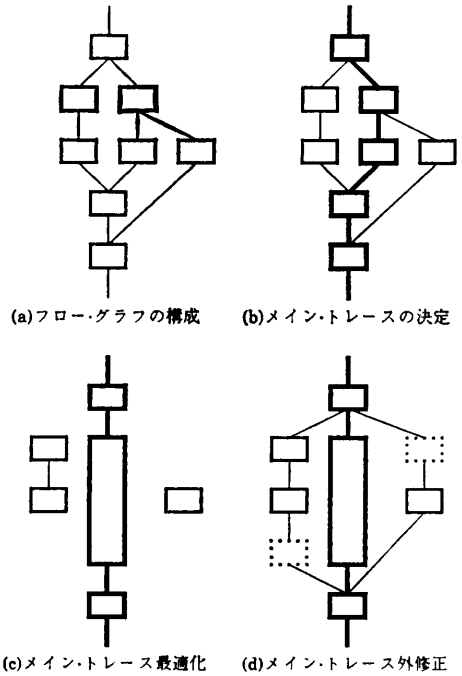

図1 トレース・スケジューリング法
Fig. 1 Schematic flow of Trace-Scheduling method.

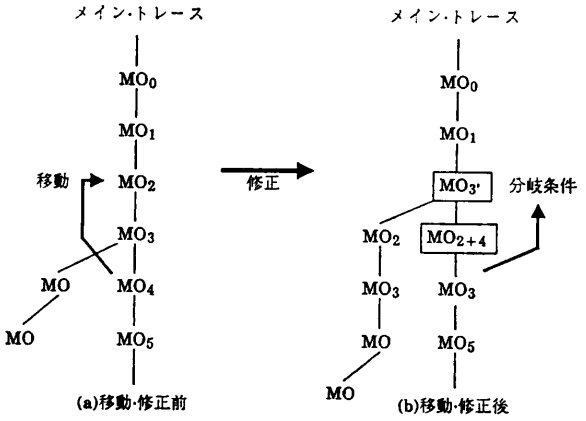

図2 マイクロ操作の移動によるメイン・トレース内の修正例(1)
Fig. 2 Modification in main-trace by moving up micro-operations (1).

機では，図2(b)において，$MO_{1+3'}$ というマイクロ命令が合成できる可能性が指摘できる．しかし，これが可能である場合は，図2(a)において既に，MO_1 と MO_3 の合成（局所的最適化）が可能となっているはずであり，この最適化が適用済みであれば，これは図2の例とはならない．

上述した演算回数の増加につながる修正を避けるため，本論文で提案する最適化方式では，トレース・スケジューリング法の第3ステップ（メイン・トレースにおける最適化処理）を以下のように変更し，メイン・トレース上での修正が必要でないように最適化処理を行う．

（1） メイン・トレースをセグメントに分割する．

（2） 各々のセグメントに対して，セグメントを含み分岐点から合流点までを形成する最も小さいマイクロ命令系列（「サブ・トレース」と呼ぶ）を決定する（**図3**(a)参照）．

（3） 各セグメント内の ALU 演算の移動をサブ・トレース内で行う（図3(b)参照）．

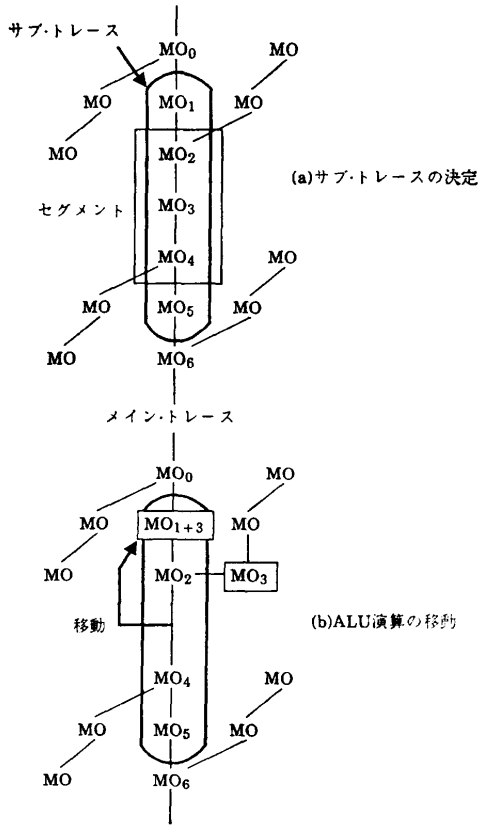

図3 セグメントとサブ・トレース
Fig. 3 Segment and sub-trace in flow graph of microprograms.

（4） 現在選択しているメイン・トレースのすべてのセグメントに対して，上記の処理を繰り返す．

メイン・トレース上の最適化処理をこの方法で実行することにより，先に述べたメイン・トレース（またはサブ・トレース）上での演算回数の増加は起こらないことを以下に示す．あるセグメントを含むサブ・トレース内での最適化処理では，セグメント内からセグメント外（ただし，サブ・トレース内）への ALU 演算マイクロ操作の移動を基本としている．この移動は，合流点を越えて上方に移動させる（図3(b)）か，分岐点を越えて下方に移動させるかの，いずれかとなる．この時，**図4**(a)において，演算 MO_c が合流点 MO_b を越えて上方に移動することによって，ALU 演算の集約（コンパクション）が行われるとする．これに上記の最適化処理を適用し，メイン・トレース外への修正処理を施した結果として，図4(b)のフロー・グラフが得られる．したがって，この処理では，演算の移動のみで，メイン・トレース内での演算回数が増加することはない．メイン・トレース外では，演算回数が増加することになる．しかし，メイン・トレース上でマイクロ命令がコンパクションによって減少していれば，メイン・トレースは実行頻度が高いので，メイン・トレース外で演算回数の増加でマイクロ命令が増加してもオーバヘッドとはならない．演算を下方に移動させる場合も同様である．

さらに，隣り合うサブ・トレースは相互に重なり合っており，サブ・トレースに分割することによる最適化効果の減少という問題をかなり防ぐことができている．

3. QA-2 のためのマイクロプログラム最適化方式

3.1 低レベル並列処理計算機 QA-2

QA-2 は，低レベル並列処理方式とダイナミック・

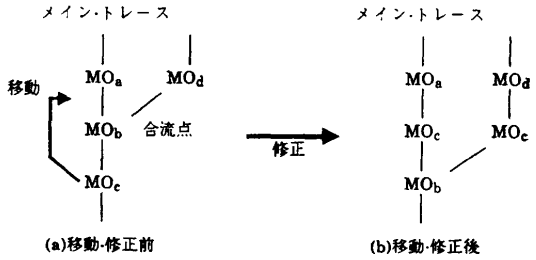

図4 マイクロ操作の移動によるメイン・トレース内の修正例(2)
Fig. 4 Modification in main-trace by moving up micro-operations (2).

マイクロプログラム制御方式との有機的結合を設計思想の基本とする計算機である[1]．QA-2 は，**図5**に示すように，同一機能をもつ4個の ALU を有し，各 ALU は水平型マイクロ命令の異なるフィールドで互いに独立に制御される．4個の ALU は一様で大容量のレジスタ・ファイルを共有して動作する．ALU は並列に動作できるだけでなく，ALU 連鎖演算機能を有している．ALU 連鎖演算とは，1つの ALU で得られた演算結果を同一マイクロ命令実行中直ちに他の ALU 演算の入力オペランドとして利用できる機能であり，並列演算時よりも若干マイクロ命令実行サイクルは延びるが，複数個のマイクロ命令を用いて演算するよりは早く演算を完了することが可能となる．この機能の活用によって，逐次的な演算系列に対しても QA-2 マイクロ命令の制御フィールドを有効に利用できる．

また，強力な順序制御機構[2]の装備によって，無条件分岐として GOTO/CALL/RETURN 分岐，条件分岐として IF/CASE 分岐を実行できるほか，マイクロ命令中で発生したステータスをその順序制御で使用しない「セーブド・モード」，マイクロ命令中で発生したステータスをその順序制御で使用する「カレント・モード」の指定も可能である．

3.2 最適化処理システムの構成

QA-2 のためのマイクロプログラム最適化処理システムは，**図6**の構成をとる．トレース・スケジューラは，QA-2 シミュレータのインタフェース部分に接続され，フロー・グラフ生成，広域的最適化処理，局所的最適化処理，ALU 演算移動可能性計算の4つの処理を行う各プログラムから構成されている．

フロー・グラフ生成プログラムは QA-2 マイクロプログラムの順序制御部を解析して，2重リンクのグラフを構成する．残り3つのプログラムはトレース・スケジューラ内部のインタフェースを通して，フロー・グラフを参照しながら動作する．

広域的最適化処理プログラムは，トレース・スケジューリング法を基本とし，それに対してメイン・トレース内における修正を行わないように変更した手法（2章で詳述）を用いている．局所的最適化処理プログラムは，広域的最適化処理プログラムから呼び出される形で動作し，フロー・グラフを参照してセグメントを計算し，その内部での最適化処理を実行する．また，広域的・局所的最適化両プログラムから呼び出される ALU 演算移動可能性計算プログラムは最適化の

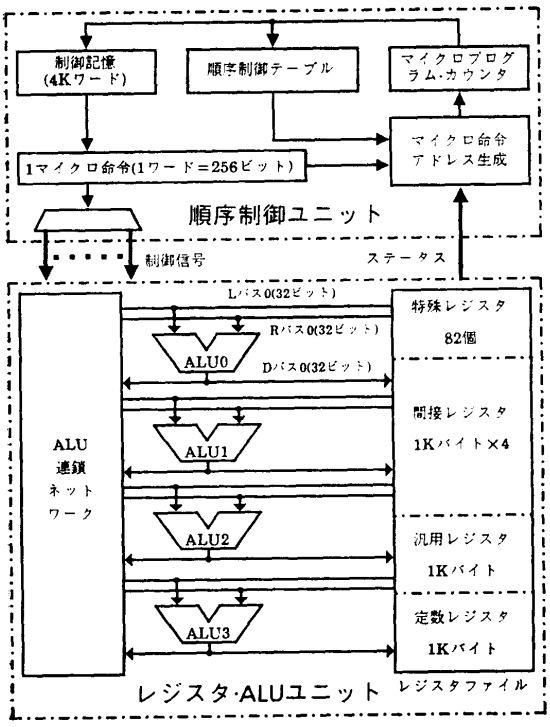

図5 QA-2 の CPU バスの構成
Fig. 5 Organization of QA-2's CPU.

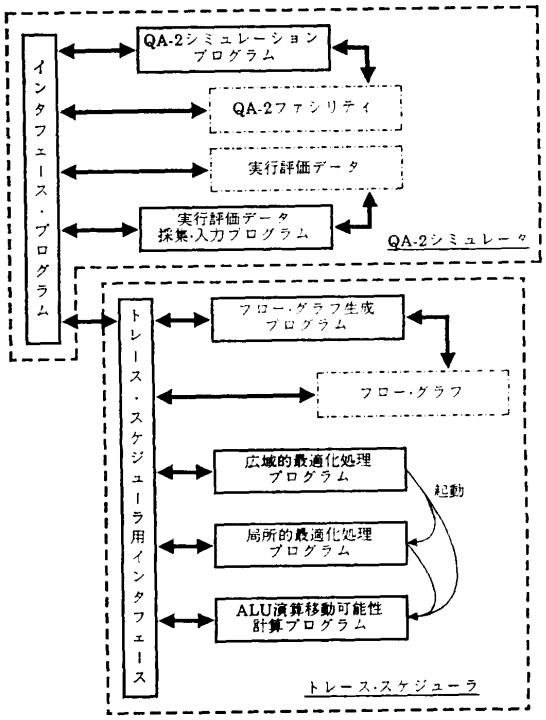

図6 QA-2 のための最適化処理システム
Fig. 6 Microprogram optimization system for QA-2.

中心となる ALU 演算の移動において，上方可動範囲・下方可動範囲を計算する．

以下，トレース・スケジューリング法を用いて QA-2 マイクロプログラムに最適化処理を施す手順を示す．

（a） QA-2 シミュレータによって，マイクロプログラムの実行評価データを採集する．

（b） マイクロプログラムのフロー・グラフを構成し，メイン・トレースを次のように決定する．

- マイクロ・サブルーチンの開始点（プログラムの最初，または，CALL 分岐の合流点）から解析を開始する．
- 現在解析している点が分岐点であれば，次に実行可能なマイクロ命令中，最も使用頻度の高いマイクロ命令を選択する．
- 現在解析している点が合流点であれば，以前そのマイクロ命令を訪れたか否かを検査する．訪れたことがあれば，ループであるから，ループ内部をメイン・トレースとして解析を終了する．
- HALT 条件，RETURN 分岐により，解析を終了．

（c） メイン・トレース上で，対象とする ALU 演算に対して，上方可動範囲と下方可動範囲を計算する．

（d） 対象 ALU 演算を含むセグメントとサブ・トレースを計算する．

（e） 対象 ALU 演算の上方可動範囲・下方可動範囲のいずれかがセグメント内部から外部への移動が可能となる値であり，かつ，移動によってマイクロ命令が増加しない場合，ALU 演算の移動をサブ・トレース内で行う．対象 ALU 演算の上方可動範囲・下方可動範囲のいずれもセグメント内部から外部への移動が不可能な値である時，局所的最適化処理を行うプログラムを呼び出す．この判定では，対象となる演算の上方可動範囲・下方可動範囲が計算されているので，これがセグメントの外部まで達しているか否かのみを判定すれば十分であり，他の条件は不必要である．なお，上方可動範囲・下方可動範囲の計算では，ソースおよびデスティネーション・レジスタの競合関係を調べる方法をとっている．また，同一マイクロ命令内で既に競合が生じている場合や，ALU 連鎖演算機能を利用している場合には，複数個の演算をまとめて移動させる時の可動範囲を計算する．

（f） ALU 演算の移動によって必要となる修正を行う．

（g） マイクロプログラム中で，まだ最適化処理を施していない部分から次のメイン・トレースを選択し，最適化処理を繰り返す．

3.3 QA-2 のための最適化処理の効果

QA-2 マイクロプログラムに対して最適化処理を施した効果について調べた．次の 2 種類のマイクロプログラムに対して最適化を行った結果を**表 1** に示す．（a）sin：三角関数計算プログラム（sin (100.0) を計算）；（b）primes：素数計算プログラム（1000 から 1160 の間の素数を出力）．（a）は，プログラム中の各マイクロ命令の実行回数に偏りが小さいもの，（b）は，逆に偏りが大きいものの代表として選択した．トレース・スケジューリング法を用いる場合，マイクロプログラムのフロー・グラフ中でマイクロ命令の実行回数の偏りが大きければ，実行頻度の大きいマイクロ命令系列，すなわち，メイン・トレースにおける最適化処理のプログラム全体に及ぼす影響が大きいのは当然である．表 1 の評価においても同様の結果が現れ，動的マイクロ命令数の減少は（a）では 0.43% 程度しか得られないのに対して，（b）では 4.26% の効果が現れた．このように，トレース・スケジューリング法による効果はマイクロ命令の実行回数の偏りが大きい場合に限られている．特に，QA-2 の応用として実際に開発されたマイクロプログラム（図形処理[3]や高級言語処理[4]など）ではこの偏りが大きいので，この最適化方式の適用が有効である．

表 1 QA-2 のための最適化処理の効果
Table 1 Effects of microprogram optimization system for QA-2.

	静的マイクロ命令数	静的平均 ALU 使用個数（個/4 個）	動的マイクロ命令数	動的平均 ALU 使用個数（個/4 個）
sin 最適化前	302	2.26	939	2.60
sin 最適化後	301 (0.33% 減)	2.37	935 (0.43% 減)	2.62
primes 最適化前	241	2.03	13740	2.18
primes 最適化後	233 (3.32% 減)	2.23	13154 (4.26% 減)	2.28

4. 低レベル並列処理計算機のための最適化処理システム

4.1 最適化処理システムの構成

本節では，QA-2 に対して構成した最適化処理システムを拡張し，より一般的な低レベル並列処理計算機に適用できる最適化処理システムについて述べる．本システムは，**図7**に示すように，種々の低レベル並列処理計算機のシミュレータを生成するシステム「低レベル並列処理計算機シミュレータ・ジェネレータ」を基に構成される．図7中のトレース・スケジューラは QA-2 のための最適化処理システムと同様の構成であるので，説明は省略する．

低レベル並列処理計算機シミュレータ・ジェネレータが生成できる目的計算機は，1個の順序制御装置と複数個の ALU が各種記憶ファシリティを共有して動作する構成のものである．これらの要素を定義した後，各マイクロ操作の実行順序の定義を行う．各定義要素の概略を以下に示す．

（1） 記憶ファシリティの定義：記憶ファシリティは実在する記憶場所を決定する基本ファシリティと，他のファシリティに対する参照を定義する合成ファシリティとに分類される．これらを組み合わせて種々の記憶ファシリティの依存関係を定義できる．

（2） ALU の定義：ALU 演算は記憶ファシリティを読み出し／書き込みモードで参照して動作するものであるため，記憶ファシリティへの参照を特別に定義する．この定義により ALU 演算間の入出力依存関係を計算でき，トレース・スケジューラにおける ALU 演算移動可能性計算の実行を可能にしている．ほかに，ALU の個数，ALU 演算の種類・演算長，ALU 連鎖演算の可能性などを定義する．

（3） 順序制御装置の定義：低レベル並列処理計算機では，マイクロプログラム制御方式によってハードウェアを直接制御するために，ハードウェア内の各種ステータスを設定するステータス制御機能，およびそれに伴う強力な順序制御機能が必要であり，これらをまとめて順序制御装置として定義する．

（4） 実行順序の定義：上記の定義のうちマイクロ操作として実行に関係する ALU 演算，ステータス制御，順序制御の3機能の実行順序を，並列実行を表す演算子 “,” と逐次実行を表す演算子 “;” とを用いて定義する．条件文を使用してこの定義を記述すれば，QA-2 などのように，多様な実行モードをもつ計算機の定義が可能となる．

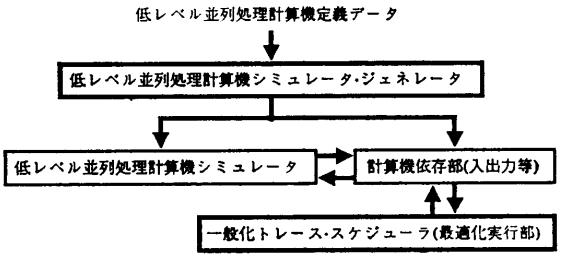

図7 マイクロプログラム最適化処理システムの構成
Fig. 7 Microprogram optimization system for low-level parallel computers.

4.2 種々の低レベル並列処理計算機のための最適化処理の効果

前節のシステムを利用して種々の低レベル並列処理計算機における最適化処理システムを構成し，その効果を測定した．この評価では，プログラムの構成・動作を一定とし，低レベル並列処理計算機の構成を変化させて資料を収集している．評価の対象とした低レベル並列処理計算機は次の4種である．

（a） QA-2 の CPU（図5）アーキテクチャと同一構成の計算機．

（b） QA-2 の CPU よりカレント・モード分岐機能（3.1 節参照）を削除した計算機．

（c） QA-2 の CPU より ALU 連鎖演算機能（3.1 節参照）を削除した計算機．

（d） QA-2 の CPU よりカレント・モード分岐機能と ALU 連鎖演算機能を削除した計算機．

なお，評価に使用したすべてのプログラムは，C言語で記述されたプログラムを文献5)のマイクロプログラム・コンパイラによってマイクロプログラムに機械的に変換したものである．このマイクロプログラム・コンパイラでは，コンパイル時の最適化処理オプションとして「ALU 連鎖演算機能の利用」や「カレント・モード分岐機能の利用」などをパラメータによって指定することが可能である．したがって，同一のプログラムからこの最適化パラメータのみを変更して得られる上記の(a)～(d)の各対象計算機用マイクロプログラムのコーディング・レベルは同一であると見なすことができる．

これらの計算機に対して，**表2**のような結果が得られた．プログラムは，3.3節の素数計算プログラムに対して各計算機構成に応じた変更を加えたものである．表2からそれぞれの構成における動的マイクロ命令数の減少率を測定すると，(a)9.15%；(b)7.61%；

(c)7.05%；(d)6.60%；となっている．この結果をそれぞれのプログラムにおけるセグメント長と比較して考察する．

通常，セグメント長が長くなれば局所的最適化処理の効果は向上するはずであり，また，メイン・トレースの長さも延びるために広域的最適化処理の効果も増大する．したがって，セグメント長の長いほうから，構成(d)＞(c)＞(b)＞(a)の順に，最適化処理の効果が得られる，あるいは少なくとも同程度の効果となる，という結果を予想していた．しかし，上述したように，実際にはこの予測と逆の結果が得られた．この理由については，次の2点が考えられる．

(1) 構成(d)＜(c)＜(b)＜(a)の順にマイクロ・アーキテクチャの機能が豊富になっており，最適化の際にその機能を利用して，最適化処理の効果を向上させることができる．

しかし，**表3**に示すように，最適化前後のカレント・モード分岐使用数は同じであり，ALU 連鎖演算を使用できる計算機構成でも，最適化後に ALU 連鎖演算回数は 10 程度増加したのみである．これらすべての ALU 連鎖演算の増加が，マイクロ命令数の減少に結び付くとは限らない．また，たとえマイクロ命令数の減少につながったとしても，高々 10 個の動的なマイクロ命令数の減少にすぎず，実際にはこの効果による影響は少なかったと判断できる．

(2) 広域的最適化処理の効果が局所的最適化処理の効果以上に影響を及ぼしているため，全体としての動的マイクロ命令数の減少に大きく反映するのは，セグメントとセグメントの接続点におけるマイクロ命令数の減少である．

例えば，マイクロ命令の減少率ではなく，減少した命令数は，(a) 356；(b) 356；(c) 424；(d) 446；となっており，対象計算機構成の変化に伴うセグメント長の増加傾向に比べると，それほど顕著な増加傾向とは言えない．マイクロプログラムは対象となる計算機の構成によって変更を加えてあるが，そのプログラムの構成や動作自体が変更するわけではないので，セグメントとセグメントの接続点におけるマイクロ命令数の減少個数は，ALU 連鎖演算の使用増加が少ないこととも相まって，各計算機構成においてほとんど変化しなかったと判断できる．

本節では，対象とする低レベル並列処理計算機の構成を変化させ，最適化処理システムの効果を評価した．その結果，各計算機に対するマイクロプログラムの最適化を行うと，広域的最適化処理の効果が大きい場合には，マイクロ命令の減少個数がほぼ同じ程度と

表 2 計算機の構成の変更による広域的最適化の評価
Table 2 Effects of microprogram optimization system for several low-level parallel computers.

	静的マイクロ命令数	静的 ALU 使用個数 (個/4 個)	動的マイクロ命令数	動的 ALU 使用個数 (個/4 個)	平均セグメント長
構成(a) 最適化前	167	2.33	3889	2.21	1.67
構成(a) 最適化後	164	2.45	3533 (9.15% 減)	2.43	1.66
構成(b) 最適化前	196	1.98	4680	1.83	1.96
構成(b) 最適化後	193	2.09	4324 (7.61% 減)	1.98	1.95
構成(c) 最適化前	227	1.72	6011	1.43	2.27
構成(c) 最適化後	223	1.85	5587 (7.05% 減)	1.54	2.19
構成(d) 最適化前	253	1.54	6766	1.27	2.53
構成(d) 最適化後	248	1.67	6320 (6.60% 減)	1.37	2.43

表 3 ALU 連鎖演算使用数とカレント・モード分岐使用数の変化
Table 3 Numbers of ALU-chaining and current-mode-branch.

	動的マイクロ命令数	動的 ALU 連鎖演算使用数(回)	カレント・モード分岐使用数(回)
構成(a) 最適化前	3889	3277	791
構成(a) 最適化後	3533	3287	791
構成(b) 最適化前	4680	3277	0
構成(b) 最適化後	4324	3287	0
構成(c) 最適化前	6011	0	791
構成(c) 最適化後	5587	0	791
構成(d) 最適化前	6766	0	0
構成(d) 最適化後	6320	0	0

なることが分かった．すなわち，マイクロ命令の減少率としては，セグメント長の短くなるような計算機構成が有利になるという結果が導かれた．QA-2 は ALU 連鎖演算機能やカレント・モード分岐，あるいは，豊富な ALU 演算命令・演算長など，多彩な機能を有し，相対的にマイクロプログラムのセグメント長を短くできる．これに加えて，さらに広域的最適化処理を実行すれば，なお一層の速度的向上・マイクロプログラムの最小化を図ることができよう．

なお，本論文における評価の尺度とした動的マイクロ命令数のほかにも，各マイクロ命令の実行時間とマイクロ命令の実行頻度をミックスしたものを尺度とする評価方法が考えられる．しかし，本論文では，次のような理由によって，動的マイクロ命令数による評価方法を採用した．(i)低レベル並列処理計算機のマイクロ操作は多彩であり，その実行時間も同一でないのが普通である．これらの並列動作可能なマイクロ操作を組み合わせて構成される低レベル並列処理計算機のマイクロ命令は，さらに多種多様な組み合わせとなり，その命令実行時間と実行頻度のミックス値を計算するのには膨大な時間がかかる．したがって，ミックス値による評価では，動的な最適化処理というトレース・スケジューリング法の利点が失われてしまう．(ii)今回の評価では，表3に示すように，マイクロ命令実行時間が長くなる ALU 連鎖演算やカレント・モード分岐の使用数は，最適化前後でほとんど変化していないため，動的マイクロ命令数による評価で十分である．

5. おわりに

本論文で考察した低レベル並列処理計算機は，レジスタ-ALU レベルでの並列処理を行うため，あらゆる応用に対してかなりの程度の並列性を抽出できるという利点がある．しかし，その反面，マイクロプログラム中の平均セグメント長が1～3マイクロ命令という，極端に短い値となる．このため，セグメント内のみに注目する局所的最適化だけでは十分な効果を得ることができず，広域的最適化処理と局所的最適化処理を繰り返して実行することが必要不可欠であることが分かった．

そこで，広域的最適化手法としてトレース・スケジューリング法を用いる最適化処理システムを作成した．本最適化方式の適用効果として，3～10% 程度の動的マイクロ命令数の減少が見られたほか，一般的な低レベル並列処理計算機に対してもセグメント長の短くなるような構成，すなわち，機能が豊富で多数のマイクロ操作を1マイクロ命令に納めることが可能な構成である場合，広域的最適化による効果が非常に高いという興味深い結果が得られた．言い換えれば，低レベル並列処理計算機においては広域的最適化処理機能が必須ということになる．

今後は，このような広域的最適化処理の重要性を考慮した上で，低レベル並列処理計算機のためのマイクロプログラム作成システムを構築する必要があろう．

謝辞 本学・萩原研究室において，QA-2 の開発に携わった諸氏，特に，QA-2 の開発・応用にあたり懇切丁寧にご指導してくださいました現在，九州大学大学院総合理工学研究科 教授 富田眞治先生に深謝いたします．

参考文献

1) 北村ほか：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).
2) 柴山ほか：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，Vol. 27, No. 10, pp. 960-969 (1986).
3) 平池ほか：低レベル並列処理計算機による3次元図形表示処理—視線探索法の場合—，第 33 回情報処理学会全国大会論文集，No. 3C-6, pp. 201-202 (1986).
4) 柴山ほか：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，情報処理学会論文誌，Vol. 27, No. 8, pp. 754-767 (1986).
5) 斉藤ほか：低レベル並列処理計算機 QA-2 のためのC言語マイクロプログラム・コンパイラ，第 35 回情報処理学会全国大会論文集，No. 1C-6, pp. 103-104 (1987).
6) 山崎ほか：マイクロプログラム制御計算機 QA-1 のマイクロプログラムオプティマイザ，電子通信学会論文誌，Vol. J64-D, No. 8, pp. 742-749 (1981).
7) 馬場：マイクロプログラム記述言語とその処理系，情報処理，Vol. 28, No. 12, pp. 1573-1584 (1987).
8) Fisher, J. A.: Trace Scheduling: A Technique for Global Microcode Compaction, *IEEE Trans. Comput.*, Vol. C-30, No. 7, pp. 478-490 (1981).
9) Su, B. et al.: An Improvement of Trace Scheduling for Global Microcode Compaction, *Proc. of the 17th Microprogramming Workshop*, ACM, pp. 78-85 (1984).

(平成元年 5 月 29 日受付)
(平成元年 11 月 14 日採録)

齊藤 雅彦（正会員）

昭和39年生．昭和61年京都大学工学部情報工学科卒業．昭和63年同大学院修士課程情報工学専攻修了．同年(株)日立製作所に入社．現在同社日立研究所第八部に勤務．在学中本研究を行った．電子情報通信学会会員．

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．昭和61年同助教授，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などの教育・研究に従事．電子情報通信学会，人工知能学会，IEEE, ACM 各会員．ICOT・WG 委員．昭和61年度本学会論文賞受賞．

新實 治男（正会員）

昭和31年生．昭和54年京都大学工学部情報工学科卒業．昭和56年同大学院修士課程修了．同年同大学工学部情報工学教室助手，現在に至る．計算機アーキテクチャ，図形処理システムなどに興味を持つ．電子情報通信学会会員．

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．電子情報通信学会，ACM, IEEE 各会員．