複合バンク機構を考慮した系統的レジスタ割当て方式とその一般化†

森 教安†† 渡邊 坦†† 神野俊昭††

近年，半導体技術の大幅な進歩により多数のレジスタを単一の LSI チップに実装することが可能となってきた．実装レジスタ数の増加に伴い，手続き呼出しやタスク切り換えの際のレジスタの退避回復のコストも増大する．そのようなオーバヘッドの削減を主目的として，レジスタ組を円環状に配置したリングバンクと，タスクごとに1組ずつ割り当てるグローバルバンクを併設する複合バンク方式を提案し，16ビットマイクロプロセッサ H16 に実現した．一方，コンパイラにおけるレジスタ割当ての有力な方法であるグラフ彩色法は，特性や使用条件の異なる複数のレジスタ群を持つ計算機に適用するにあたってはいくつかの問題がある．そこで，それらの問題点を明確にし，リングバンクとグローバルバンクのような複合バンク機構を有する実行モデルに対し，グラフ彩色に基づくレジスタ割当て方法を適用した複合バンク割当て方式を提示する．この方式は，彩色の優先度を算出するプロフィット関数を，複数のレジスタクラス上で非固定的に定義することにより，複合バンク実行方式に対して，効率的な割当てを可能にする．また，この方式は，従来のグラフ彩色に基づくレジスタ割当ての概念を完全に包含し，アクセスコストや用途の異なる複数種類のレジスタ群を持つ複合レジスタクラスモデルに対しても，プロフィット関数の定義を与えれば，アルゴリズムは変えずに適用できる応用範囲の広い方式である．

1. はじめに

実装されているレジスタ数の少ない計算機では，アクセスコストの高い主メモリへのアクセスが頻繁に行われる．レジスタ数を多くすれば主メモリへのアクセス回数を削減できる[7]．半導体技術の大幅な進歩によって数百個のレジスタを単一の LSI チップに実装することも可能となっているが，レジスタの数が増えるに従って，手続き呼出しやタスク切り換えに伴うレジスタの退避回復のオーバヘッドが増大する．

手続き呼出しオーバヘッドを削減する手段として，レジスタウィンドウ[1),3)]が提案され，どのようなインストラクション・セットにおいてもかなりの性能向上を図ることができると報告されている[2)]．しかし，これではタスク切り換えのオーバヘッドが大きく，高速なタスク切り換えを要する仕事にはあまり適していない．そこで，16 個のレジスタからなるレジスタバンクを 16 組設け，そのうち8組をタスクごとに割り付けるグローバルバンクとし，他の8組をリング状に構成して手続きの呼出し/復帰時にバンク使用位置を前進/後退させるリングバンクとする複合バンク機構を開発し，16 ビットマイクロプロセッサ H16 に実現した[10),11)]．これは，手続き呼出しやタスク切り換えに伴うレジスタの退避回復オーバヘッドを両方ともほとんどなくせる方式である．

一方，コンパイラにおけるレジスタ割当ての方法としてグラフ彩色に基づく技法の有効性が認められ広く研究されている[4)~6),8),9)]．しかしながら，このグラフ彩色法に基づき，複合バンク実行方式に適するレジスタ割当てを行うにあたっては役割とアクセスコストの異なるレジスタ群が混在するため，いくつかの問題が生ずる．

本稿では，それらの問題点を明確にし，解決策として複合バンク割当て方式を提示する．この方式は，彩色時に優先度の尺度となるプロフィット関数を複数個設けることにより，複合バンク実行方式においてもグラフ彩色法に基づいた効率的な割当てを可能とする．また，この方式は，従来行われてきたグラフ彩色法に基づく方法を，複合バンク実行方式に限らず，レジスタの役割とアクセスコストが不均一な計算機に対する一般的な手法（複合レジスタクラス割当て方式）として利用できることも併せて述べる．

以下，2章ではグラフ彩色法のリングバンク実行方式への適用上の問題点を，3章では解決策としての複合バンク割当て方式をそれぞれ述べ，4章では一般化した複合レジスタクラス割当て方式，および，その応用例を示す．

† A Register Allocation Method Applicable to Compound-Bank Organization and Its Generalization by NORIYASU MORI, TAN WATANABE and TOSHIAKI KOHNO (Systems Development Laboratory, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所

2. グラフ彩色法と複合バンク実行方式

2.1 グラフ彩色法

グラフ彩色とは，辺で結ばれている隣接ノードは同一の色で塗らないでグラフ上のすべてのノードを彩色することであり，コンパイラのレジスタ割当ての有力な方法として広く研究されている．このグラフ彩色法では，レジスタ割当て対象（データフロー解析等により抽出されるもので，仮想レジスタ，論理レジスタなどとも呼ぶが，以降では単に割当て対象と呼ぶ）をノードで表し，実際に存在する個々のレジスタ（実レジスタ，物理レジスタと呼ばれる）がそれぞれ固有の色を持つと考える．2個の割当て対象をそれぞれ別々のレジスタに割り当てる必要がある場合，対応するグラフ上のノードを辺で結ぶ．これを干渉辺といい，これらのノードと干渉辺により干渉グラフを作成する．レジスタの数を r とし，この干渉グラフを r 色の異なる色で塗り分けることができるとき，r 彩色可能であるといい，彩色結果が求める割当て結果となる．r 彩色不能であるときには，いくつかのノードをレジスタに割り当てることを断念し，干渉辺とともにグラフ上から削除して，r 彩色可能となるようにグラフを変形した後，再度彩色を行う．このとき，彩色を断念したノードは，主メモリに割り当てられる．（これをスピルアウトと呼ぶ．）

グラフ彩色法に基づくレジスタ割当て方式の代表的なものとしては，Chaitin[5] や Chow[6] の方法が挙げられる．しかしながら，両者とも複合バンク実行方式の下で実現するには問題がある．次節では，複合バンク実行モデルを示し，それらの適用上の問題点について述べる．

2.2 複合バンク実行モデル

H 16 マイクロプロセッサで開発したリングバンク方式は，手続き呼出しとタスク切り換えに伴うレジスタ退避回復コストの削減および，パラメータ渡しの効率化を目的とする．そのリングバンクとグローバルバンクを併設する，複合バンク方式のアーキテクチャモデルを**図1**に示す．

リングバンク RB i (i=0, 1, …, 7) は，それぞれ 16 個のレジスタを含むレジスタ組であり，現在のバンク番号 CBNR (Current Bank Number Register) が指すカレントバンク CB と，1つ前のバンク番号 PBNR (Previous Bank Number Register) が指すプレビアスバンク PB が各時点で使用可能である．CBNR および PBNR は，手続き呼出し/回復のたびに，ハードウェア的にインクリメント/デクリメントされる．また，このほかに 16 個のレジスタからなるグローバルバンク G が 8 組と，主メモリ M が存在する．

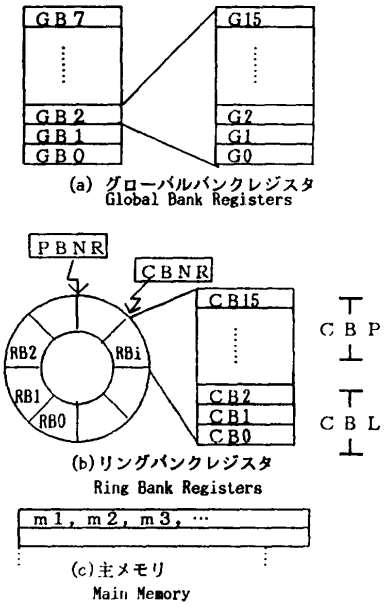


図1 複合バンク実行モデル
Fig. 1 The model for compound bank execution.

この複合バンク方式の効果的活用のためのコンパイラの実行方式として，複合バンク実行方式を考案した．すなわち，CB/PB をパラメータ領域 CBP/PBP と局所変数領域 CBL/PBL に分け，

（1） パラメータは CBP に割り当て，仮パラメータとして PBP を用いることで，パラメータ渡しはバンク切り換え時に自動的に行う（CBP と PBP は物理的には同一レジスタ組）．

（2） 局所変数は CBL に割り当て，呼出し時に退避回復しない．

（3） グローバルバンクはタスクごとに別々に割り当てる．各タスクの大域変数は，そのタスクのグローバルバンクに割り当て，手続き呼出し時には，呼出し側が必要なレジスタのみを退避回復する．

とした．これにより，パラメータの受け渡し，および，手続き呼出しに伴う局所変数の退避回復は，不要となる．上記のうち，(1)と(3)には，特に問題はないが，(2)の扱いに関しては，割当て性能上のトレードオフが存在する．すなわち，CBL には，退避回復コストがないという利点があるが，1回当りの平均的アクセスコストは，G のほうが小さい．（これは，命令

語のオペランドのサイズが異なるため，命令フェッチ等に差が生じるためである.）したがって，手続き呼出しの前後で使用する（これを手続き呼出しを横断すると呼ぶ）ことのない局所変数は，G に割り当てた方が利益が大きい．そこで，問題点の明確化のため，複合バンク実行方式の限定された環境をモデル化した複合バンク実行モデルを示す．

割当て対象の集合 T として局所変数のみを考慮し，リングバンクレジスタは CBL のみに限定し，単に R と表記する．また，グローバルバンク G は，コンパイラにおいてはシングルタスクの場合を考えればよいので，普通の汎用レジスタ群と考える．

したがって，割当て先の記憶域として有効なレジスタクラスの集合 RC は，以下の3種類が存在する．(以後，メモリも広義のレジスタと考えて，これらの3種類をすべてレジスタクラスと呼ぶことにする.)

R：リングバンクレジスタクラス (CBL)

G：汎用レジスタクラス

M：主メモリ（レジスタ）クラス

また，それぞれのレジスタクラス $C\in RC$ に対し，

$n(C)$…C に属するレジスタの個数

$a(C)$…C に属するレジスタの1回当りの平均アクセスコスト

$s(C)$…C に属するレジスタ1回当りの平均手続き呼出し横断コスト（退避回復コスト）.

とする．このとき，前記のアーキテクチャにおいては，

ⅰ) $n(G)=16$, $n(R)=10$, $n(M)\approx\infty$.

ⅱ) $a(G)<a(R)<a(M)$.

ⅲ) $s(G)>s(R)=s(M)=0$.

の関係が成り立っている．

すなわち，G と R の個数は少数であるが，M は事実上無制限である．G は最もアクセスコストが小さいが，手続き呼出し横断時には $s(G)$ だけ余分なコストがかかり，手続き呼出しの横断頻度が高い場合には，R に割り当てた方がよい．

2.3 適用上の問題点

前節で述べたグラフ彩色アルゴリズムを用いた割当て方法を，複合バンク実行方式の環境の下に適用する場合の問題点を考察する．

r 彩色不能な場合のスピルレジスタの選択にあたっては，レジスタに割り当てる効果の小さいものを選ぶ必要がある．このレジスタに割り当てる効果を（コンパイル時に予測し）定量表現した指数を，ここではプロフィット値と呼ぶことにする．プロフィット値の小さいものから順にスピルアウトするが，このとき，プロフィット値はレジスタ割当て効果の尺度ともなる．同様にして，プロフィット値を決定する各種の要因を，プロフィット要因と呼ぶ．一般的なプロフィット要因として考えられるものには，以下のものがある．

（1） メモリに割り当てた場合の総アクセスコスト.

（2） 生存区間（レジスタの占有度）.

複合バンク実行モデルにおいては，さらに，

（3） 手続き呼出し横断頻度

を考慮する必要がある．

このとき，単一のレジスタクラスとメモリから構成されるアーキテクチャモデルを想定し，できる限り汎用レジスタへ割り当てようとする従来のレジスタ割当て方式では，次のような問題点が発生する．

問題点Ⅰ レジスタ割当て対象 t は，使用形態によって適するクラスが異なる．すなわち，アクセスコストの点では G が望ましく，手続き呼出し横断コストの点では R がよい．この2つのプロフィット要因はそれぞれ独立したものであり，単純な比較はできないが，t に関する最適なレジスタクラス，および，あるレジスタクラスで最も割当て効果の大きい t を表現できる必要がある．

問題点Ⅱ スピルアウト先はメモリとは限らない．例えば，彩色中に G が不足したとする．もし，R にまだ余裕があればスピルアウト先としては M よりも R とする方が，有利である．（手続き呼出し横断コストは R と M の間に差がないので，アクセスコストの差により R の方がよい.）

問題点Ⅲ 与えられるプログラムの割当て対象集合 T の特性により，レジスタクラスの優先度を変える必要がある．例えば，固定的に R, G, M の順に彩色を行うとプログラム特性が反映できない．

そこで，これらの問題点に対する対応策として考案した複合バンク割当て方式について次章で述べる．

3. 複合バンク割当て方式

この章では，前章で述べた適用上の問題点を解決する複合バンク割当て方式を提示する．本方式の骨子は，多重プロフィット関数の導入，およびそれを評価尺度として最大限の割当て効果をあげるアルゴリズムにある．以下，両者を説明した後，複合バンク実行モデルに対する具体的な彩色例で，その問題点の解決策を示す．

3.1 多重プロフィット関数の導入

個々の割当て対象 t に対し，プロフィット値を与える関数をプロフィット関数と呼ぶ．一般的には，それをアクセスコストの差やアクセス頻度から算出する．複合バンク実行モデルでは，複数のレジスタクラス上で割当て可能であり，レジスタクラスの種類によって好適度が異なるため，プロフィット関数はどのレジスタクラス上で割り当てるかにより，定義を変える必要がある．アクセスコストや手続き呼出し横断コストは，レジスタクラスに固有な尺度であり，アクセス頻度や手続き呼出し横断頻度は t に固有である．

そこで，レジスタクラスごとの独立したプロフィット値を，共通のプロフィット要因関数の和として与える多重プロフィット関数を導入する．ここに，プロフィット要因関数とは，プロフィット要因ごとに割当て効果を定量表現するものである．

割当て対象の集合を T，レジスタクラスの集合を RC としたとき，ある割当て対象 $t\in T$ のレジスタクラス $C\in RC$ におけるプロフィット値は，プロフィット関数

$$P(C,t)=\begin{cases}\perp & \text{[割当て不可能なとき]}\\ \sum_{\alpha\in A} W\alpha\cdot P\alpha(C,t) & \text{[その他の場合]}\end{cases}$$

で与えられる．ここに，

$P\alpha(C,t)$：プロフィット要因関数
α：プロフィット要因
A：プロフィット要因の集合
$W\alpha$：プロフィット要因 α に対する重み付け定数

である．この定義によれば，すべての割当て対象 t は個々のレジスタクラス C に割り当てた場合の効果 $P(C,t)$ を別々に持ち，それぞれの割当て効果は共通のプロフィット要因関数 $P\alpha(C,t)$ およびその要因に対する重み付け定数 $W\alpha$ により計算される．プロフィット関数 $P(C,t)$ は，任意の C と t に対して定義しているが，ある特定のレジスタクラス X の $P(X,t)$ をレジスタクラス X のクラスプロフィット関数と呼ぶ．このとき，t のクラスプロフィット値は，レジスタクラスの数だけ存在するが，実際に割り当てるクラスは1つである．したがって，すべての t に関して割り当てた結果のプロフィット値の総計が，プログラム全体に対する割当て効果の定量表現である．また，一般に，割当て対象 t は任意のレジスタクラスに割当て可能ではない．例えば，整数型のデータは浮動小数点レジスタに割り当てられないように，各種の制約条件が存在するのが普通である．そこで，ある割当て対象 t があるレジスタクラス C 上で割当て不可能なとき，特定の意味を持つプロフィット値として $\perp$ を与える．このとき任意の値と $\perp$ の和は $\perp$ である．

次に，この多重プロフィット関数を用いて，複合バンク実行モデルに適用する際の関数構成例を示す．プロフィット要因 α としては，2.3 節で述べたとおり，アクセス頻度 u，手続き呼出し横断頻度 c，生存区間長 l を考える．それぞれのプロフィット要因関数を，以下のように定義できる．

$$Pu(C,t)=u(t)/a(C)\quad[\geqq 0]$$
$$Pc(C,t)=-c(t)\cdot s(C)\quad[\leqq 0]$$
$$Pl(C,t)=-l(t)/n(C)\quad[\leqq 0]$$

ここに，

$u(t)$：割当て対象 t のアクセス頻度
$c(t)$：割当て対象 t の手続き呼出し横断頻度
$l(t)$：割当て対象 t の生存区間長

であり，割当て不能な場合($\perp$)は存在しない．また，$RC=\{G,R,M\}$ に対し以下の条件が成り立っている．

$$Pu(G,t)>Pu(R,t)>Pu(M,t)>0$$
$$Pc(G,t)<Pc(R,t)=Pc(M,t)=0$$
$$Pl(G,t)=Pl(R,t)<Pl(M,t)=0$$

したがって，各レジスタクラスのクラスプロフィット関数は

$$P(G,t)=Wu\cdot u(t)/a(G)-Wc\cdot c(t)\cdot s(G)-Wl\cdot l(t)/n(G)$$
$$P(R,t)=Wu\cdot u(t)/a(R)-Wl\cdot l(t)/n(R)$$
$$P(M,t)=Wu\cdot u(t)/a(M)$$

と簡略化できる．

このような多重プロフィット関数を構成することにより，おのおののプロフィット要因を独立して反映できるので，適用上の問題点Iを解決している．すなわち，特定の t から見た場合は $c(t), l(t), u(t)$ の値により G, R, M のどのクラスが一番有利かを表現すると同時に，特定のレジスタクラスにおいて，割当て対象 t_1, t_2 ($\in T$) のどちらが割当て効果が大きいかも表す．

3.2 アルゴリズム

多重プロフィット関数に基づくグラフ彩色アルゴリズム（C風に記述）を図2に示す．本アルゴリズムは，与えられた割当て対象集合 T に関するレジスタクラスの優先順位を決定する前半部と，そのクラス優先順位に従ってグラフ彩色を行う後半部から成る．

前半部では，割当て対象集合 T のすべての要素 t

```
Compound_Register_Class_Coloring() {
  for (i=1; i<=n(RC); ++i) {
    SP(Ci)=0;
    for (j=1; j<=n(T); ++j) {
      SP(Ci)=P(Ci, tj)+SP(Ci);
    }
  }
  UT←T; UC←RC; CC←φ;
  while (UC≠φ) {
    Cmi ← max(UC|SP(UC));
    CC ← CC+Cmi;
    UC ← UC−Cmi;
    tm ← max(UT|P(Cmi, UT));
    mayC ← φ;
    for (i=1; i<=n(RC); ++i) {
      if (#nb_colors(t, Ci)<n(Ci)) mayC←mayC+Ci;
    }
    Cm ← max(mayC|P(mayC, tm);
    if (Cm∈CC) {
      color(t, Cm); UT ← UT−tm;
    }
  }
}
```

[記号説明]

RC: すべてのレジスタクラスの集合 (∋C_i, 個数を n(RC) とする)

CC: 彩色クラス集合 (⊆RC)

UC: 未彩色なレジスタクラスの集合 (⊆RC)

T : すべての割当て対象の集合 (∋t_j, 個数を n(T) とする)

UT: 未彩色な割当て対象の集合 (⊆T)

mayC: t に対し彩色可能なレジスタクラスの集合 (⊆RC)

CC ← CC±Cmi: クラス Cmi をクラス集合 CC [に加える/から除く]

#nb_colors(t, C): t の隣接ノードのクラスCの彩色数

max(UC|P(UC, t)): P(C∈UC, t) が，最大となるクラス C_i(∈UC)

max(UT|P(C, UT)): P(C, t∈UT) が，最大となる割当て対象 t(∈UT)

color(t, Cm): 割当て対象 t をクラス Cm 中の使用可能なレジスタで彩色する.

図 2 複合レジスタクラス割当てアルゴリズム
Fig. 2 The compound register class allocation algorithm.

に対しプロフィット値を調べ，クラス別のプロフィット値総計

$$SP(C_i)=\sum_{t\in T} P(C_i, t).$$

を求める．このとき，$SP(C_i)$ は，すべての t を C_i 上で割り当てた場合の割当て効果を表現している．この $SP(C_i)$ により，与えられた割当て対象集合に関したクラス優先順位を決定するのが目的であり，ここでは具体的な割当ては行わない．

後半部では，彩色クラス集合 CC 上でのグラフ彩色を逐次行う．CC は，最初は空集合であり，クラス優先度の高いものから1つずつ加えられる．CC が更新されるごとに，すべての未彩色なノード t に対し以下の彩色条件を満たすか否かを判定する．

[彩色条件]

(i) 彩色クラス C は，CC に含まれている．

(ii) C は彩色可能なクラスのなかでは，t に関するプロフィット値が最大である．

(iii) t は，C 上で彩色可能．すなわち，t に隣接するノード中 C に属する相異なる色で塗られた数は $n(C)$ より少ない．

あるノードに対し上記の彩色条件を満たすレジスタクラス C 上のレジスタが存在するときのみ割り当てる．そうでないときには未彩色のまま残し，CC が更新された後に処理する．グラフ上のすべてのノードを塗り終えた時点でこのアルゴリズムは実質的に終了する．

上記のアルゴリズムにおいて，与えられた特定の割当て対象の集合 T に関する動的優先順位（前半部と彩色条件(i)）により問題点Ⅲを，彩色可能なクラスの内の最適クラスへの割当て（彩色条件(ii)）により問題点Ⅱを解決する．

次節では，上記のアルゴリズムによる具体的な彩色例を用いて，その解決例を示す．

3.3 彩色例と割当て効果

説明を簡潔に行うために以下のような状況を仮定する．

$G=\{g1, g2\}$

$R=\{r1, r2\}$

$M=\{m1, m2, m3, \cdots\}$

$T=\{t1, t2, t3, t4, t5, t6, t7, t8\}$

また，実際の値（命令語のマシンサイクル数）からの概算により，各レジスタクラスの特性は，

$a(G)=1$ $a(R)=2$ $a(M)=10$

$s(G)=20$ $s(R)=0$ $s(M)=0$

$n(G)=16$ $n(R)=10$ $n(M)\approx\infty$

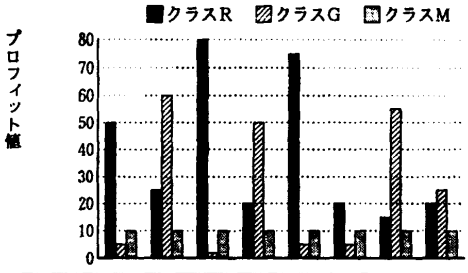
とする．重み付け定数は，実現上の調整値であるが，

$Wu : Wc : Wl = 1 : 1 : 1/4096$

とする．また，$u(t), c(t)$ は，それぞれの発生点のループのネストの深さの総計で近似し，$l(t)$ は命令語レベルの命令数で近似する．

以上の状況において，T のおのおののレジスタクラス G, R, M 上でのクラスプロフィット値は**図3**，干渉グラフは**図4**の(a)のとおりであったとする．

この設定のもとで，先のアルゴリズムに従って彩色



	t1	t2	t3	t4	t5	t6	t7	t8	総計
クラスR	50	25	80	20	75	20	15	20	305
クラスG	5	60	2	50	5	5	55	25	207
クラスM	10	10	10	10	10	10	10	10	80

図 3 プロフィット関数例
Fig. 3 An example of profit functions.

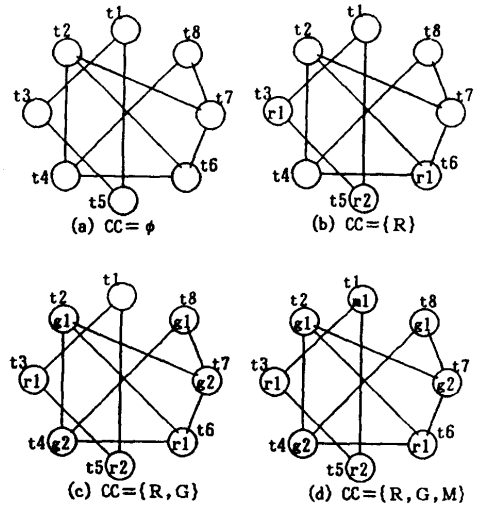


図 4 複合レジスタクラス彩色例
Fig. 4 An example of compound register class coloring.

が行われていく過程を干渉グラフ上で示したのが図4の(b)～(d)である．前半部でのクラス別のプロフィット値総計の比較においては

$$SP(M)<SP(G)<SP(R).$$

なので，この例におけるクラス優先順位は R, G, M と決定される．(一般的には，全体的に手続き横断頻度が高いときにこの順となる．)

後半部で，まず彩色クラス集合 $CC=\{R\}$ を仮定して $t1$～$t8$ に対し彩色条件を判定し，彩色を施した結果が(b)である．次に，$CC=\{R,G\}$ と更新し，上記と同様に処理した結果が(c)である．最後に，$CC=\{R,G,M\}$ となった後（すなわち，すべてのレジスタクラスを使用して），最終的な彩色結果となる(d)を得る．

この例では，T に関して R,G,M とクラス優先順位を決めており，この順にレジスタ資源を割り当てる．一方で，図4の(d)において，$t1$ は G ではなく M にスピルアウトしている．これは，

$$P(R,t1)>P(M,t1)>P(G,t1)$$

に従い，彩色条件(ii)が最適なスピルアウト先を選択したためである．

次に，割当て効果の優位性に関して述べる．グラフ彩色法による方式として

方式A：複合バンク割当て方式．

方式B：クラス優先順位を G, R, M に固定した方式．

方式C：クラス優先順位を R, G, M に固定した方式．

とし，その割当て効果を比較する．方式B，Cでは，最初からクラス優先順位を決めてあり，この順に従ってレジスタ資源を割り当てる．方式Cは，この例では方式Aでの動的クラス優先順位と一致している．方式B，Cによる彩色結果を**図 5, 6** にそれぞれ示す．また，3方式それぞれの割当て結果のプロフィット値総計を**図 7** に示した．プロフィット値は，割当て結果のプロフィット値の総計が割当て効果を示すように定めたものなので，優先順位を固定する方式に対するA方式の優位性はこの例において明らかである．(もし，この結果が実情に合わないならば，その要因を分析し，プロフィット関数にその要因を反映する必要がある．)

ここで，割当て対象 T を，この節で示した具体例に限定しない一般の場合に拡張して考える．方式B，Cはプロフィット要因（手続き呼出し横断頻度とアク

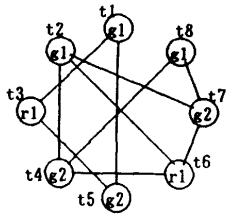


図 5 方式Bによる彩色結果
Fig. 5 The result using coloring method B.

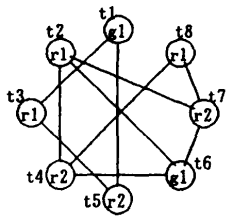
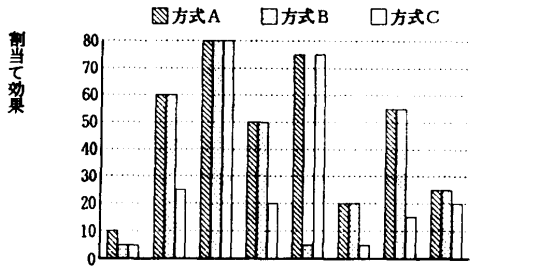


図 6 方式Cによる彩色結果
Fig. 6 The result using coloring method C.



	t1	t2	t3	t4	t5	t6	t7	t8	総計
方式A	10	60	80	50	75	20	55	25	375
方式B	5	60	80	50	5	20	55	25	300
方式C	5	25	80	20	75	5	15	20	245

図 7 割当て方式の効果比較
Fig. 7 Comparison of allocation method's effect.

セス頻度）が，どちらか一方に極端に偏った場合に適する方式である．ここで，方式Aは方式BまたはCのいずれかの結果に近づくが，両方式を下回ることはない（∵彩色条件(ii)）．以上の点から，方式AがB，Cを含めた各種方式のなかで，割当て結果のプロフィットの総計を最大にすることがわかる．

4. 汎用性と適用範囲

前章では，複合バンク実行モデルへの適用のため複合バンク割当て方式を提示した．しかし，この割当て方式は，複合バンク実行モデルに限定する必要はなく，レジスタクラスとプロフィット関数を定義すれば，様々なケースに適用できる．そこで，以降では一般的なレジスタ割当てを考慮した方式であることを明示するため，複合レジスタクラス割当て方式と呼ぶ．実際，本方式は，従来のグラフ彩色法による方式を完全に包含した拡張となっている．この章では，本方式の適用範囲である複合レジスタクラスモデルを提示し，その拡張された部分を明確化するとともに，若干の適用例を示す．

4.1 複合レジスタクラスモデル

以下，いくつかの定義を述べた後，適用性の議論にはいる．

定義1 レジスタクラス C_1 と C_2 に以下の関係が成り立つとき，C_1 と C_2 は排他的彩色可能集合を持つという．（以後，単に排他的と呼ぶ）

$^{\forall}t\in T$ に対し

$(P(C_1,t)=\perp)\vee(P(C_2,t)=\perp)$.

C_1 と C_2 が排他的であるとき，両者は独立して彩色可能であり，彩色結果が互いに影響を及ぼすことはない．よって，両者は別々の干渉グラフを持つものとして彩色すれば，レジスタクラスの彩色順序とは無関係に，彩色結果は一意に定まる．

一方，C_1 と C_2 が非排他的であるとき，

$^{\exists}t\in T$ に対し

$(P(C_1,t)\neq\perp)\wedge(P(C_2,t)\neq\perp)$.

であり，ある t に関しては双方のレジスタクラス上で割当て可能である．このことは，C_1 と C_2 は，独立して彩色できず，C_1 の彩色結果が C_2 での彩色（あるいは，その逆）に影響を及ぼし，どちらを優先するかで全体的な彩色結果が異なることを意味する．

定義2 非排他的レジスタクラス C_1 と C_2 に以下の関係が成り立つとき，C_1 は C_2 に対しプロフィット関数 P 上で割当て対象集合 T に関して優位なクラスであるという．

$SP(C_1)\geqq SP(C_2)$.

定義3 非排他的レジスタクラス C_1 と C_2 のプロフィット関数 P に以下の関係が成り立つとき，C_1 は C_2 に対し P に関して絶対優位なクラスであるという．

$^{\forall}t\in T$ に対し

$P(C_1,t)\geqq P(C_2,t)$.

優位性は，あらかじめ定めてあるプロフィット関数 P を用いて，与えられた T をレジスタクラスごとに評価した結果により定まる．個々の割当て集合 T（言い換えれば，具体的なプログラム例）に対して定義されるので，任意の T に対する一意な順序付けではない．さらに，T に関するレジスタクラスの優先度を与えるものであって，個々の割当て対象 t $(\in T)$ に関するレジスタクラス優位度とは異なる．したがって，C_1 が C_2 に対し優位であっても，

$P(C_1,t')<P(C_2,t')$.

となる t' $(\in T)$ が存在してもよい．これは，この t' に関しては C_2 が望ましいが，T 全体としては C_1 が望ましいことを意味する．

それに対して，絶対優位性はプロフィット関数 P を定めた時点で決定されており，個々の割当て対象集合とは無関係に定まる．また，この場合は，T に関するレジスタクラスの優先順位と，個々の割当て対象 t $(\in T)$ に関するレジスタクラス優位順位が一致する．個々の割当て対象 t がすべて

$P(C_1,t)\geqq P(C_2,t)$.

となるようにクラスプロフィット関数が定義されていることから，

「絶対優位ならば，優位である．」

のは明らかであるが，逆は真ではない．

一般に，m 個のレジスタクラス C_i $(i=1,2,\cdots,m)$

が存在するとして，任意の2個のクラス C_i, C_j $(i \neq j)$ の間に絶対優位関係があるならば，その絶対優位な順に彩色クラス集合 CC に加えていくことにより，最適な割当て結果が得られる．この場合，上位のクラス上で彩色不能になるに従って，下位のクラスへと逐次スピルアウトが行われていく．汎用レジスタクラス G とメモリクラス M のみから成る，いわゆる“従来のレジスタ割当て問題のモデル”のほとんどがこの範疇に属する．すなわち，G が M に対し絶対優位であると仮定して，できる限り G へと割り当てようとする．

これに対し，複合レジスタクラスモデルは，任意のレジスタクラス間に絶対優位関係を仮定しない範囲に拡張している．G と M のみの従来のレジスタ割当て法は，本モデルのプロフィット関数設定にあたり，G が M に対し絶対優位とする制限を与えた場合にほかならない．この G と M のみの場合でも，両者を非絶対優位の関係として扱えば，より効果的な割当て結果を得ることができる（呼出し側で退避回復するとき，手続き呼出し横断頻度によっては，M の方が有利になる場合が存在する）．

次節では，このモデルが複合バンク実行モデルや，その他の実行環境モデルを包含した汎用性を持つことを，若干の例を交えて述べる．

4.2 適 用 例

まず，複合バンク実行モデルは，複合レジスタクラスモデルの一例である．すなわち，レジスタクラスとして R, G, M の3者が存在し，R と G の間には非排他的で，絶対優位でない関係がある．この関係を適切に表現したのが，多重プロフィット関数である．

さらに，2.2 節でモデルとしては割愛したパラメータ変数に関しても，実パラメータクラス CBP を設け，プロフィット要因に実パラメータ要因 ac を与える．すなわち，実パラメータとしての出現頻度を $ac(t)$，レジスタクラス C の実パラメータに対するアクセスコストを $aac(C)$ として，

$$Pac(C, t) = -ac(t) \cdot aac(C)$$

を付け加えるのみでよい．（このとき，$aac(CBP)=0$）

割当て上の制約のある場合の例として16ビットマイクロプロセッサ68000のアドレスレジスタ AR とデータレジスタ DR を挙げる．AR, DR は「乗除算のオペランドと結果は DR」「インデクスレジスタは AR」等の制約があるほかは，大部分の場合両者ともに割当て可能であり非絶対優位な関係にある．したがって，レジスタクラスとして，AR, DR および，M を考えて

$$P(AR, t) = \begin{cases} \perp & \text{[乗除算のオペランドと結果]} \\ Wu \cdot u(t)/a(AR) - Wl \cdot l(t)/n(AR) & \text{[その他].} \end{cases}$$

$$P(DR, t) = \begin{cases} \perp & \text{[配列等のインデクス]} \\ Wu \cdot u(t)/a(DR) - Wl \cdot l(t)/n(DR) & \text{[その他].} \end{cases}$$

とすれば複合レジスタクラスモデルを適用できる．

また，いくつかのアーキテクチャに存在するペアレジスタも，本方式の枠組で扱える．例えば，HITAC-M シリーズにおいて除算に使うレジスタは，偶数番レジスタから始まる連続領域でなければならない．このときペアレジスタクラス PR および，単独で用いた場合のクラス SR を設け，

$$P(PR, t) = \begin{cases} Wu \cdot u(t)/a(PR) - Wl \cdot l(t)/n(PR) & \text{[除算の結果]} \\ \perp & \text{[その他].} \end{cases}$$

$$P(SR, t) = Wu \cdot u(t)/a(SR) - Wl \cdot l(t)/n(SR).$$

として，PR の彩色時には連続したレジスタを割り当てればよい．この場合，$n(PR)$ は，$n(SR)$ の半分であり，ペアレジスタとして扱う対象の多いときには自然にそれが優先される．そのほか，関数の返す値の格納場所が規定されている場合にも，関数値クラスを設けることで容易に解決できる．

以上述べたとおり，本稿の方式の適用範囲は，絶対優位性を仮定しない複合レジスタクラス実行モデルにまで拡張されており，さまざまな応用が可能である．

5. おわりに

複合バンク機構を有する複合バンク実行モデルに対し，グラフ彩色に基づくレジスタ割当て方法を適用する際の問題点と解決策を提示した．それらの問題点は，従来のグラフ彩色法において，複数のレジスタクラス（特に，プロフィット要因の異なるレジスタクラス）に対する配慮がないことに起因していた．そこで，主メモリをも含む広義のレジスタクラスに対しそれぞれプロフィット値を与える多重プロフィット関数を導入し，それに忠実に従って最適な割当て結果を得るアルゴリズムを示した．また，本方式は，従来のグラフ彩色に基づくレジスタ割当ての概念を完全に包含した複合レジスタクラスモデルに対して，一般性を失うことなく適用でき，プロフィット関数の定義を与えれば，アルゴリズムは変えないで適用できることも併せて示した．

本方式は，各種状況に対する適用性が高く，今後の

計算機アーキテクチャの発展に伴い，新しい概念に基づくレジスタクラスが出現した際に，柔軟に対応する一手段となりえる．プロフィット関数の定義方法やレジスタクラスの設定等の手法に関しては，さらに理論的定式化が必要であるが，多重プロフィット関数とアルゴリズムの枠組は，広範囲にわたって活用できる．

謝辞 本方式の適用仕方等について，有意義な御意見を多数賜りました立教大学の島内剛一教授，ならびに，本研究の機会を与えていただいた日立製作所システム開発研究所川崎淳前所長，同武蔵工場マイコンソフト設計部伊藤紀彦部長に感謝いたします．

また，査読者の方には，アルゴリズムや彩色例の不備等，きめ細かい御指摘を受けた．あわせて感謝いたします．

参考文献

1) Patterson, D. A. and Sequin, C. H.: A VLSI RISC, *Computer*, Vol. 15, No. 9, pp. 8-21 (Sep. 1982).
2) Hitchcock Ⅲ, C. Y. and Brinkley Sprunt, H. M.: Analyzing Multiple Register Sets, *Proc. of the 12th Annual International Symp. on Computer Architecture*, pp. 55-63 (Jan. 1985).
3) Patterson, D. A.: Reduced Instruction Set Computers, *Comm. ACM*, Vol. 28, No. 1, pp. 8-21 (Jan. 1985).
4) Chaitin, G. J. et al.: Register Allocation via Coloring, *Comput. Lang.*, Vol. 6, pp. 47-57 (1981).
5) Chaitin, G. J.: Register Allocation & Spilling via Graph Coloring, *Proc. of the ACM SIGPLAN '82 Symp. on Compiler Construction*, pp. 98-105 (1982).
6) Chow, F. and Hennessy, J.: Register Allocation by Priority-based Coloring, *Proc. of the ACM SIGPLAN '84 Symp. on Compiler Construction*, pp. 222-232 (1984).
7) Radin, G.: The 801 Minicomputer, Proc. Symp. on Architechtual Support for Programming Languages and Operating Systems, *ACM Sigarch*, Vol. 10, No. 2, pp. 39-47 (Mar. 1982).
8) Ditzel, D. R. and McLellan, H. R.: Register Allocation for Free: The C Machine Stack Cash, Proc. Symp. on Architectural Support for Programming Languages and Operating Systems, *ACM Sigarch*, Vol. 10, No. 2, pp. 48-56 (Mar. 1982).
9) Larus, J. R. and Hilfinger, P. N.: Register Allocation in the SPUR Lisp Compiler, *Proc. of the ACM SIGPLAN 1986 Symp. on Compiler Construction*, pp. 255-263 (1986).
10) 馬場志朗，渡邊 坦ほか：システム集積型オリジナルマイクロプロセッサ H 16，日立評論，Vol. 69, No. 7, pp. 25-32 (Jul. 1987).
11) Maejima, H., Kida, H., Watanabe, T. and Baba, S.: A 16-bit Microprocessor with Multi-register Bank Architecture, *Proc. FJCC 1986*, pp. 1014-1019 (1986).

(昭和63年2月 5 日受付)
(平成元年4月11日採録)

森 教安（正会員）

昭和34年生．昭和57年早稲田大学理工学部数学科卒業．同年(株)日立製作所に入社．以来，ソフトウェア設計・開発支援ツール，プログラミング言語，コンパイラの研究開発に従事．現在，同社システム開発研究所第2部に勤務．

渡邊 坦（正会員）

1939年生．1962年京都大学理学部数学科卒業．同年日本アイ・ビー・エム(株)入社．1967年(株)日立製作所に入社後，同社中央研究所，システム開発研究所にて，設計計算システム等のプログラムの研究から，プログラム開発を容易にするための計算機言語とコンパイラ，プログラミングツールの研究，ならびにソフトウェア面からの計算機アーキテクチャの検討へと進んできた．現在も計算機の使い方を簡易化するための言語と言語処理系に対する関心が高い．1973年度情報処理学会論文賞受賞．工学博士．ACM, IEEE Computer Society，日本ソフトウェア科学会各会員．

神野 俊昭（正会員）

昭和26年生．昭和48年早稲田大学理工学部電気工学科卒業．昭和50年同大学院修士課程修了．同年(株)日立製作所入社．以来，同社システム開発研究所にて，プログラミング言語，コンパイラ，コンパイラ生成系の研究開発に従事．昭和58～59年スタンフォード大学客員研究員．現在，同所第2部主任研究員．日本ソフトウェア科学会，ACM 各会員．