

新マイクロプロセッサH32とその開発環境

加藤 肇彦

日立製作所 宇宙技術推進本部

茶木 英明

日立製作所 半導体事業部

TRON仕様に準拠した32ビットマイクロプロセッサH32のアーキテクチャを簡単に紹介し、そのソフトウェア開発環境について述べる。日立製作所では、H32のサポートソフトウェアを自社開発するだけでなく、国内外のサードパーティーと連携して、サポートソフトウェアの充実をはかっている。

サポートソフトウェア群を構成するものは、アセンブラ、Cコンパイラ、リンカージェネリタ、シミュレータ、デバッガ、ITRON仕様OS、FORTRANコンパイラ、PASCALコンパイラ、UNIX、VRTXリアルタイムOS等である。

最後に、H32のアーキテクチャを利用したCコンパイラの最適化技法について詳述する。

H32 MICROPROCESSOR AND ITS SOFTWARE DEVELOPMENT ENVIRONMENTS

Hatsuhiko Kato*

Hideaki Chaki **

* Space Systems Division, Hitachi Ltd.

216 Totsukamachi, Totsukaku Yokohama 244, Japan

** Semiconductor Division, Hitachi Ltd.

5-20-1 Josui Honmachi, Kodaira city, Tokyo 187, Japan

After briefly introducing the features of TRON architecture microprocessor H32, this paper discusses its software development environments.

In addition to its own inhouse activities to develop H32 oriented softwares, Hitachi encourages third party software houses to develop supporting software.

The supporting softwares consist of an assembler, compiler, linker and related OS support.

The presentation will conclude by discussing the details of compiler optimization used for the implementation of the C compiler.

1 H32の概要

1.1 H32の位置づけ

TRONプロジェクトでは32ビットマイクロプロセッサの仕様を提示しており、これに準拠したプロセッサを、東芝、松下電器、富士通、日立製作所、三菱電機、沖電気などの各社が開発中または製品化中である。さらにこれら各社の内、富士通、日立製作所、三菱電機、沖電気の4社は、G_{MICRO}ファミリと称するTRON仕様準拠のCPUと周辺VLSIならびに開発環境などについて開発を分担し、製品化を進めている。³⁾ G_{MICRO}/200はG_{MICRO}ファミリの中で中堅の位置を占め、日立製作所ではH32/200（以下H32）と呼んでいる。³⁾

日立製作所ではH32を含めた独自アーキテクチャのマイクロプロセッサをHシリーズとして系列化し、8ビットのH8、16ビットのH16に対して、H32をHシリーズの最上位に位置づけている。H32の応用対象としては、パソコン、ワークステーション、制御・通信用コンピュータ、グラフィックエンジンなどを想定している。⁴⁾

1.2 H32の仕様

H32は図1-1に示す内部構成を持ち、構成要素である5つのユニットは、同時並行動作を実行することにより、6段のパイプライン制御を実現している。

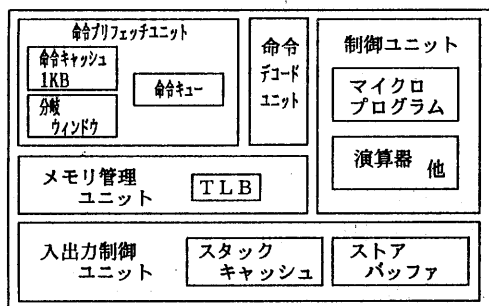


図1-1 H32の構成

基本命令は99種類と少ないが、16本の32ビット汎用レジスタを利用した豊富なアドレスモードとの直交性を持ち、各命令の機能は強力である。高速処理のために短縮命令フォーマット、異種サイズ間演算、分散キ

ャッシュ等の技術を採用し、最大性能10MIPS、EDNベンチマーク7MIPS、演算プロセッサ使用時4MWIPS（いずれも20MHz動作時）を達成している。状態制御には例外、割込み、トラップの3種類の機能があり、プロセッサ状態語（PSW）の退避／回復によってコンテキストを切り替えている。

4ギガバイトのアドレス空間を持ち、4キロバイト単位で2レベルページングによる仮想記憶を、内蔵メモリ管理ユニット（MMU）によってサポートしている。PSW中に2ビットのリングフィールドを持ち、上位のリングレベルへのメモリアクセスを禁止している。最上位のリングレベル0へは、特権モードでのみアクセスが可能である。

FPU（浮動小数点演算ユニット）の管理と制御は専用命令により実行される。

64ビットオペランド指定やPSW中の拡張ビットにより、将来の64ビットアーキテクチャへの拡張が考慮されている。

2 H32の開発環境

2.1 開発環境整備の基本方針

日立ではG_{MICRO}グループならびにTRONプロジェクトを背景に、H32のソフトウェア開発環境の効率的な整備を進めている。日立の基本方針はつぎのとおりである。

- (1) 最も基本的であり、どのユーザにも使われる共通のサポートソフトウェアを、自社開発する。
- (2) サポートソフトウェアの仕様については、Hシリーズ間で共通の仕様とし、また、極力標準規格の仕様を取り入れる。
- (3) ソフトウェア専門メーカーが得意とする分野では、サードパーティに開発段階から積極的に働きかけ、TRONプロジェクトを活用したH32用サポートソフトウェア環境の整備を充実させる。

2.2 自社開発ツール

日立が自社で開発している基本サポートソフトウェアは、アセンブラ、Cコンパイラ、リンケージエディタ、シミュレータ・デバッガ、ITRON仕様OS(HI32)である。

これらの内HI32以外はいずれも当面VAX/VMSで稼働す

るが、将来はSUNワークステーション等のUNIXエンジンやPC環境へも搭載する。

アセンブリモニックはTRON仕様に準拠し、構造化アセンブリ機能、マクロ機能を有する。CコンパイラはANSI規格案に準拠し、その詳細は次章に述べる。リンケージエディタは(財)日本規格協会のSYSROF (SYmbol-information Standard Relocatable Object Format)に準拠した。

2.3 他社開発ツール

日立ではG_{MICRO}グループ各社とともに国内外のサードパーティーとの協調により、表2-1に示すようにサポートソフトウェアの充実をはかっている。

表2-1 他社開発サポートソフトウェア

品 種	機 能
アセンブラ, リンカ, デバッガ, コンパイラ, FORTRANコンパイラ, PASCALコンパイラ (MRI社など)	VAX/VMS, VAX/ULTRIX, SUNワークステーションなどの上で稼働。アセンブラ仕様は日立のH32モニックに準拠。デバッガはCPUとFPUの命令をシミュレート。Cの言語仕様はANSI規格案に準拠。FORTRANとPASCALの言語仕様はANSI規格に準拠。
U N I X (ISC社)	System V Release 3に準拠。日立のシングルボードコンピュータH32SBC上で稼働。
VRTX (Versatile Real Time Executive) (RSC社)	コンパクトなリアルタイムオペレーティングシステム。H32SBC上で稼働。

日立製作所では、今後も国内外のサードパーティとの連携を進め、ソフトウェアの充実をはかって行く。

3 H32用Cコンパイラの開発

3.1 コンパイラの特長

日立製作所では、H32のCコンパイラを自社開発した。開発に当たっては、下記の点に留意した。⁵⁾

- (1) 言語仕様のANSI規格案への準拠。⁶⁾
- (2) 機械語オブジェクト・フォーマットの(財)日本規格協会が標準化を進めているSYSROFへの準拠。

(3) H32のアーキテクチャに適合した、高性能オブジェクト・プログラムの生成。

(4) シンボリック・デバッグのためのシンボル情報の出力。

(5) ROM化可能なオブジェクト・プログラムの出力。

3.2 コンパイラの最適化技法

H32用Cコンパイラの開発においては、下記の方針で最適化技法を採り入れることにした。

(1) 既存最適化技法の採用

構文解析の結果をコード生成に送る前の中流工程において、構文自身の持つ冗長性を分析してこれを除去するためには、一般的に知られている最適化技法を最大限に活用する。

結果が常に一定となる値をコンパイル段階で評価してしまう定数たたみこみ、共通部分式を一回だけ評価して以降その結果を使用する共通部分式の消去、乗除算をシフト命令で代用する演算強度の緩和、0の加減算や1の乗除算を省略する不要命令の除去、起こり得ない分岐条件による不要分岐命令の除去、重複して発生する共通コードシーケンスの除去等がその技法である。

(2) H32のアーキテクチャを考慮したコードの最適化

H32のアーキテクチャには、高水準アドレスモードや高水準命令のような、コンパイラ開発時の便宜を考慮した機能がある。また、命令パイプライン制御機能を考慮すれば、コードを最適化することができる。以下に、H32のアーキテクチャを考慮した最適化について述べる。

3.3 高水準アドレスモードの発生

H32では間接アドレスやインデクスアドレスを組み合わせることによって、最大4段階までのチェインドアドレスモードが可能である。例えば、H32のアセンブリ言語では

```
MOV @(@ (R0,R1*4,10),R2,R3*2,10),R4
```

という表現が可能である。これは”レジスタR0の内容をスケーリングファクタ4によってスケーリングした値と、ディスプレースメント10を加えて得たアドレスによって間接指定したアドレスの内容 @ (R0,R1*4,10)

と、レジスタR2の内容と、レジスタR3の内容を2によってスケールした値に、ディスプレースメント10を加えて得たアドレスによって間接指定したアドレスの内容を、レジスタR4へフェッチする。”という操作を表す。

例：Cのプログラム例

```
char*a;
f()
{
    register int b, c;
    char d[80];
    a=&d[b+c+1]+1;
}
```

の中の代入文 `a = &d[b+c+1]+1;` は、もしチェインドアドレスモードを使用しなければ

```
MOV    R14,R0
ADD    R13,R0
ADD    R12,R0
SUB    #78,R0
MOV    R0,@_a
```

と翻訳され、その結果の実行には10サイクルかかるのに対し、チェインドアドレスモードを使用すると、

```
MOVA   @(R12,R13,-78,R14),@_a
```

と翻訳されて8サイクルで実行できる。

例：同様に、Cのプログラム

```
inta;
f()
{
    register int b;
    a=b *12;
}
```

の中の代入文 `a=b *12;` は、チェインドアドレスモードを使用しなければ、

```
MOV    R13,R0
MUL    #12,R0
MOV    R0,@_a
```

と翻訳されて、13サイクルを要する。チェインドアドレスモードを使用すると、

```
MOVA   @(R0*4,R0*8),@_a
```

と翻訳されて6サイクルで完了する。

3.4 高水準命令の発生

Cの言語仕様に規定されていないが、H32に装備されているストリング操作用ならびにFPU用高水準命令は、Cのソースコード上では標準ライブラリ関数として記述され、コンパイルの結果、命令列はオブジェクトコード中にインライン展開される。

例：アドレスaからはじまる文字列の長さを求めるプログラム

```
char*a;
f()
{
    register int x;
    x=strlen(a);
}
```

はインライン展開の結果、

```
MOV    @_a,R0
MOV:Z   #0,R2
MOV:Z   #0,R3
SSCH.B/EQ
ADD     #1,R2
NEG     R2
MOV     R2,R13
```

というオブジェクトになる。ここに命令SSCH.B/EQはレジスタR0（この場合@_a）で示されるアドレスから、レジスタR3の内容（この場合0）と一致するデータが見つかるまで探索をつづけ、その回数をレジスタR2に反映する。

3.5 命令パイプライン制御の最適化

H32の命令はパイプライン制御によって実行されているため、命令列はなるべくパイプライン・フローを乱さないように構成されていることが望ましい。また、CPUとFPUは極力並列動作することが望ましい、これらの観点から、

- (1) あるレジスタに値が設定された直後には、そのレジスタをアドレス計算に使用しない。
- (2) FPUで実行される命令と、その直後のCPU命令は並列処理させる。

という条件を極力実現するように、命令を配列する。そのためには命令の実行順序に対する制約を有向グラフで表現し、その制約に違反しない範囲で上記の条件を満たすよう、命令を再配列する。

例：Cのソースプログラム

```
f()
{
    register int b,c,*d;
    int a;
    ...
    c=a+b;
    d=&a;
    b=*d;
    c++;
    ...
}
```

の4つの代入文はコンパイルの結果、

- ① MOVA @(@ (R14,-4),R13),R12
- ② MOVA @(-4,R14),R11
- ③ MOV @R11,R13
- ④ ADD #1,R12

という命令列に変わる。しかし、ここでレジスタR11

が命令②で代入操作を受けた直後に命令③によってアドレス計算に使用されているため、パイプライン制御に乱れが生じる。そこで各命令の実行順序に対する制約を有向グラフによって示すと、図4-1ようになる。

この図から命令①と命令②は、命令③に先んじるという制約さえ満たすならば、どちらを先に実行しても同じ結果を与えることがわかる。そこで、レジスタR11を使用する命令②と命令③の間に時間を置くために、命令①と命令②を転置し、

命令列

- ② MOVA @(-4,R14),R11
- ① MOVA @(@ (R14,-4),R13),R12
- ③ MOV @R11,R13
- ④ ADD #1,R12

を得る。この結果命令②によって決定したレジスタR11の内容を使用して、命令①の実行中に命令③のオペランドを先取りし、パイプライン制御が円滑に実行できる。この操作によってプログラムの実行時間は、1マシンサイクル短縮される。

4 結論と今後の課題

H32のソフトウェア開発環境、ならびにH32のアーキテクチャを利用したCコンパイラのコード最適化について述べた。

日立製作所ではG_{MICRO}グループ各社と連携して、H32/300 (G_{MICRO}/300) およびH32/100 (G_{MICRO}/100) のアセンブラとシミュレータ・デバッガの開発を進めている。さらにUNIXワークステーションやPC上への搭載を進め、国内外サードパーティとの連携により、サポートソフトウェア、デバッグ用ユーティリティ、OS等のソフトウェアのラインアップを充実させる。

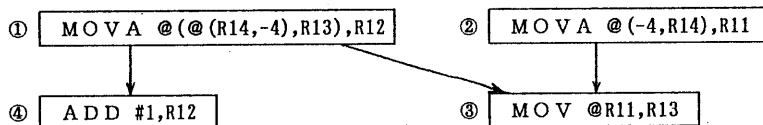


図4-1 命令実行順序の制約

Cコンパイラの最適化については、現在以下の改良を推進中である。

(1) 大域的レジスタ割り付けの強化

Cの言語仕様では、変数割り付けのレジスタ宣言の機能を持つ。

H32Cコンパイラにおいても、レジスタ宣言された変数をレジスタに割り付けるとともに、レジスタ宣言されなかった自動変数やパラメタ、演算時テンポラリ変数等をレジスタに割り付ける機能を持つ。今後は、配列名や関数名のアドレスやグローバルなstatic変数、extern変数に対して、関数内での参照頻度解析情報から大域的にレジスタが割り付くよう改良していく。

(2) ループ内の最適化

for文やwhile文等Cソースプログラムでループ実行する文に対しては、実行効率を向上させるために特別な考慮が必要である。ループ制御変数へのレジスタ割り付け、ループ内不変式のループ外移動、同一のループ制御を持つ文の融合、ループ帰納変数の削除等を実現し高速化を図っていく。

参 考 文 献

- 1) 小特集「TRON」, 情報処理, Vol.30, No5(1989).
- 2) 稲吉, 西向井, 海永, 長谷川: 32ビットマイクロプロセッサ" H32" ファミリー, 日立評論, Vol.70, No12, pp.25-30 (1988).
- 3) Inayoshi, H., Kawasaki, I., Nishimukai, T. and Sakamura, K.: Realization of G_{MICRO}/200, IEEE micro, Vol.8, No2, pp.12-21 (1988).
- 4) 富永: 情報産業市場ニーズとHシリーズマイクロコンピュータ, 日立評論, Vol.70, No12, pp.1-8 (1988).
- 5) Kashiwagi, Y., Chaki, H. and Narushima, M.: Development of a C Compiler for G_{MICRO} Micro-processor Based on TRON Architecture, TRON Project 1988, pp.341-350(1988).
- 6) ANSI, Draft Proposed Standard-Programming Language C, ANSI X3J11 Language Subcommittee, X3J11/86-0074 (1986).