解　説

6. 命令セットアーキテクチャの具体例

6.3 ミニコン・ワークステーション HPPA†

高　橋　伸　明††

1. HPPA 命令セットアーキテクチャの設計方針

HP プレシジョン・アーキテクチャ（以下 HPPA と略す）は，縮小命令セット・コンピュータ（RISC）アーキテクチャの一例として紹介されてきた．事実このアーキテクチャの実行モデルは RISC ベースであり，単一サイクル処理，レジスタ中心処理，LOAD/STORE（ロード/ストア）設計といった特徴がある．

一般的に RISC マシンは，命令セット中の命令数を減らしインプリメンテーションの簡略化や実行時間の改善を目指しているものといえる．しかし，HP コンピュータの新世代アーキテクチャとなる HPPA を設計するにあたっては，まずさまざまな処理環境（データベース・アクセス中心の処理環境，高速技術演算が主となる処理環境，リアルタイム処理環境，ネットワーク処理環境，プログラム開発環境，AI 環境など）を決め，次にここで必要とされるイントリンシック機能を決めている．そして，このイントリンシック群を効率良くサポートできることを目的として HPPA アーキテクチャが設計された．実際に，これまでに開発され使用されているいくつものアーキテクチャ上で，処理環境ごとに命令の使用頻度・実行時間占有率を調査・解析し，HPPA 命令セット決定の基礎としている．ここでは命令の数を最少化することよりも，使用頻度の高い命令を効率良く実行することに重点が置かれ，これまでよりもシンプルなハードウェア上で，かつ，単一サイクルで処理される命令を選択することが重要と判断された．

このアーキテクチャは，十分に検討・吟味された命令群をもつことにより，使用目的やシステム規模，パフォーマンス・クラスが異なる製品群をカバーすることも目指している．汎用コンピュータ・クラスのビジネス・システムやスーパ・ミニコン，エンジニアリング・ワークステーション（EWS），そして，デスクトップの単一ユーザ向け製品までが HPPA により商品化され，アーキテクチャは統一されることになる．

以下に，RISC をベースとして開発された HPPA の特徴をまとめた．

2. 命令の機能と形式

HPPA に定義されている 140 種類の命令セットを表-1 に示す．命令セットは，大きく分けて次の 7 種類の命令群からなる．

① メモリ参照命令（24 個）
② 条件分岐命令（12 個）
③ 無条件分岐命令（6 個）
④ 算術・論理演算命令（54 個）
⑤ システム制御命令（31 個）
⑥ SFU☆ オペレーション命令（4 個）
⑦ COP☆☆ 用命令（9 個）

2.1 命令形式とアドレッシング

HPPA では，図-1 のように，すべての命令がメモリの 1 語に相当する 32 ビットの固定長となっている．この中でも時間にクリティカルな機能は特定フィールド（位置）に配置されており，最少の負荷もしくはまったくデコードの負荷なしで処理が行える．すべての命令はワード単位で作られており，命令がページ境界をまたぐことはない．

二つある汎用レジスタのソース・オペランドのアドレスは特定フィールド（ビット 6 から 10 とビット 11 から 15）に配置されており，レジスタを命令のデコード段階もしくはそれ以前に読むことができる．汎用レジスタのオペランドではなく即値（イミーディエット）オペランドが必要な場合には，ALU もしくは

† An Example of Instruction Set Processor Architecture HPPA by Nobuaki TAKAHASHI (Produc Manager Business Systems Sector Marketing Y・H・P).

†† 横河・ヒューレット・パッカード(株)　BSS マーケティング部

☆ 特殊機能ユニット
☆☆ 補助プロセッサ

表-1 HPPA の命令セット

命令セット	算術/論理演算命令	システム制御命令
メモリ参照命令	Add	**Break**
Load Word	Add Immediate	Return From Interrupt
Load Halfword	Add Immediate Left	Set System Mask
Load Byte	Load Immediate Left	Reset System Mask
Load Word Indexed	Add Logical	Load Space ID
Load Halfword Indexed	Add and Trap on Overflow	Move to Space Register
Load Byte Indexed	Shift One and Add	Move to Control Register
Load Word Short	Shift Two and Add	Move from Space Register
Load Halfword Short	Shift Three and Add	Move from Control Register
Load Byte Short	Shift One and Add Logical	Move to System Mask
Load Word and Modify	Shift Three and Add Logical	Synchronize Caches
Load Word Absolute	Shift One, Add, and Trap on Overflow	Probe Read Access
Load Word Absolute Short	Shift Two, Add, and Trap on Overflow	Probe Read Access Immediate
Load Offset	Shift Three, Add, and Trap on Overflow	Probe Write Access
Load and Clear Word Indexed	Add with Carry	Probe Write Access Immediate
Load and Clear Word Short	Add with Carry and Trap on Overflow	Load Physical Address
Store Word	Subtract	Load Hash Address
Store Halfword	Subtract from Immediate	Purge Instruction TLB
Store Byte	Subtract and Trap on Overflow	Purge Instruction TLB Entry
Store Word Short	Subtract Immediate and Trap on Overflow	Purge Data TLB
Store Halfword Short	Subtract with Borrow	Purge Data TLB Entry
Store Byte Short	Subtract with Borrow and Trap on Overflow	Insert Data TLB Address
Store Word and Modify	Subtract and Trap on Condition	Insert Data TLB Protection
Store Word Absolute Short	Subtract and Trap on Condition or Overflow	Insert Instruction TLB Address
Store Bytes Short	Divide Step	Insert Instruction TLB Protection
無条件分岐	Compare and Clear	Purge Data Cache
Branch and Link	Inclusive OR	Flush Data Cache
Gateway	Exclusive OR	Flush Instruction Cache
Branch and Link Register	AND	Flush Data Cache Entry
Branch Vectored	AND Complement	Flush Instruction Cache Entry
Branch External	Unit XOR	Diagnose
Branch and Link External	Unit Add Complement	**特殊機能ユニット用オペレーション**
条件分岐	Unit Add Complement and Trap on Condition	Special Operation Zero
Move and Branch	Decimal Correct	Special Operation One
Move Immediate and Branch	Intermediate Decimal Correct	Special Operation Two
Compare and Branch if True	Add Immediate and Trap on Overflow	Special Operation Three
Compare and Branch if False	Add Immediate and Trap on Condition	**補助プロセッサ用カード/ストア命令**
Compare Immediate and Branch if True	Add Immediate, Trap on Condition or Overflow	Coprocessor Load Word Short
Compare Immediate and Branch if False	Compare Immediate and Clear	Coprocessor Load Word Indexed
Add and Branch if True	Variable Shift Double	Coprocessor Load Doubleword Short
Add and Branch if False	Shift Double	Coprocessor Load Doubleword Indexed
Add Immediate and Branch if True	Variable Extract Signed	Coprocessor Store Word
Add Immediate and Branch if False	Variable Extract Unsigned	Coprocessor Store Indexed
Branch on Variable Bit	Extract Signed	Coprocessor Store Doubleword
Branch on Bit	Extract Unsigned	Coprocessor Store Doubleword Indexed
	Variable Deposit	Coprocessor Operation*
	Variable Deposit Immediate	
	Deposit	
	Deposit Immediate	
	Zero and Variable Deposit	
	Zero and Variable Deposit Immediate	
	Zero and Deposit	
	Zero and Deposit Immediate	

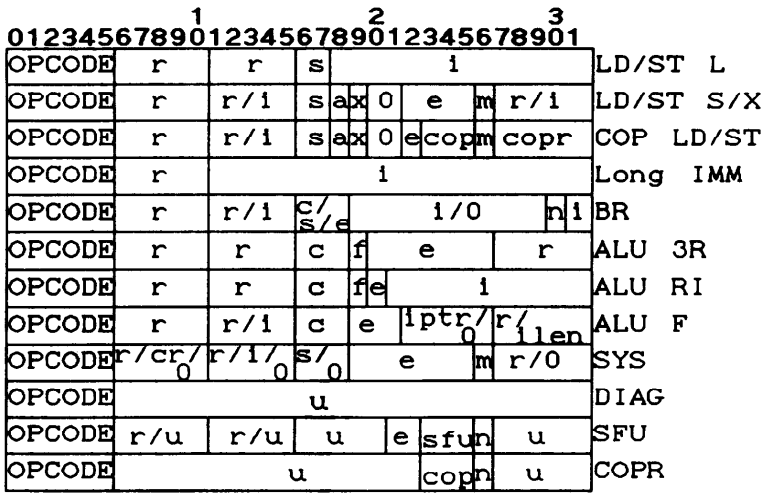
* 浮動小数点演算を含む

SMU（シフト・マージ・ユニット）の前でマルチプレクサにより，その選択が行われる．また，3レジスタ命令では，3番目のレジスタ指定は命令の最終5ビット（ビット 27 から 31）に配置されるが，ソース・オペランドとなるレジスタは必ず最初の2レジスタ・フィールドを使用することになっている．データ変換およびデータ移動オペレーションのターゲット・レジスタは，三つのどの位置にも配置できる．これは，結果の書き込みがオペランドの読み取り後に発生するものであり，ターゲット・レジスタのアドレス・デコードが時間にクリティカルではないためである．

空間レジスタも特定フィールドに配置される．これは，仮想記憶アドレスとしてオペランドが使われるからである．

2.2 基本データ形式

基本データ形式は，ビット，バイト，整数，浮動小数点数，10進数である．ビットを直接アドレスすることは許していないが，汎用レジスタ内の個々のビットやビット・フィールドの操作，テストがサポートされている．整数は8，16，32ビット長で，符号付き整数は2の補数として表される．浮動小数点数は，単精度（32ビット），倍精度（64ビット），4倍精度（128ビ



フィールド名として使われている略号の説明

r	汎用レジスタ指定	f	条件否定指定
s	空間レジスタ指定	iptr	即値ポインタ
i	即値（イミーディエット)*	ilen	即値長
a	修飾前／後指定	cr	制御レジスタ
x	インデックス・レジスタ指定	0	未使用（0にセット）
e	subop (拡張 opcode)	u	未定義
m	修飾指定	sfu	SFU 指定
n	無効化指定	cop	コ・プロセッサ指定
c	条件指定	copr	コ・プロセッサ・レジスタ

* ディスプレイスメントもしくはオフセットも可

図-1 HPPA の命令形式

ット）2進形式がソフトウェアもしくは特殊ハードウェア（オプション，モデルによる）によりサポートされる．この浮動小数点数表現は ANSI/IEEE 754-1985 標準に準拠したものである．パック型およびアンパック型十進数は7，15，23，31の BCD 数を表す．

2.3 アドレッシング

HPPA はメモリをバイト・アドレスでアクセスするが，単語長，一語，二語単位でアドレスすることも許されている．アドレスは物理または仮想アドレスのどちらかであり，ロード／ストア命令はどれも，物理モードまたは仮想モードを使用できる．また，仮想モードは，プロセッサのステータス・ワード中にある二つのフラグにより命令フェッチとデータ・アクセスとで別々に使用可能となる．物理メモリのポインタは32ビット長の符号無し整数であり，オペランドの第一バイトのアドレス値をもつ．

仮想記憶はグローバルに定義されており，同一アドレスを異なるプロセスの異なるオブジェクトに使うことは許していない．仮想記憶は 2^{32} バイトのアドレス空間群から構成されている．空間数はインプリメンテーションのレベルにより異なるが，現在リリースされている製品では 2^{32} となっている．この仮想アドレスは32ビットの空間識別子と32ビットの空間内オフセットの結合で表される．

命令アドレスは，命令フェッチ時および命令キャッシュ・フラッシュ，命令 TLB の各命令，そして分岐ターゲット演算で計算される．空間レジスタを参照する命令の場合には，命令フォーマットの中の3ビット長Sフィールドを使い，8個ある空間レジスタの一つを特定している．

データ・アドレスは，ロード，ストア，セマフォ，プローブ，データ・キャッシュ，それにデータ TLB の各命令で計算される．

2.4 即　値

このアーキテクチャはレジスタをベースとしているが，32個の汎用レジスタのほかに，特に命令レジスタをオペランドのソースとして多用している．HPPA の命令の多くは32ビットの固定長フォーマットの中に即値フィールドをもっているが，このことは，定数は各命令中に即値として保有できる可能性が高いこと

を意味している．命令の種類により異なった長さの即値がきても，その符号ビットは常に同じ位置に置かれている．この即値オペランドは，汎用レジスタにロードする必要がないため主記憶のアクセスと汎用レジスタの使用をセーブすることができる特徴ともなる．

命令中フィールドの最長即値がほとんどの定数を表現できたとしても，32ビット長の即値を扱えるようにしておく必要がある．HPPA では，これを命令をペアにする方法で実現している．まず，長即値命令により，即値の最上位21ビットを汎用レジスタにロードもしくは加算する．これに続く次の命令は，このレジスタをベース・レジスタとして残りの下位ビットを埋め32ビット即値を完成させる．こうすることにより，汎用レジスタに32ビット定数を使うことができ，フル32ビット長のディスプレイスメントをロード・ストア命令は行えることになる．

3. HPPA 命令の実現技術

プロセッサ・モジュールは図-2 のようになっているが，プロセッサは高速キャッシュ・システムで密に結ばれた「命令フェッチ・ユニット」と「実行ユニット」から構成されている．キャッシュはオプションとされているが，おそらくほとんどのプロセッサは標準装備することになると思われる．また，TLB（トランスレーション・ルックアサイド・バッファ）と呼ばれるハードウェアによるアドレス変換テーブルや浮動小数点演算などの特定機能を高速で実行するためのアシスト・ハードウェアを装備する機種も多い．実行ユニットはローカル・レジスタのデータ変換を行い，キャッシュや主記憶が参照するアドレスを生成する．

フェッチ・ユニットは，命令アドレスの計算，命令のフェッチ，デコードを行い，実行ユニットに情報を送り出している．このフェッチ・ユニットは，固定長命令によるデコードおよび次命令アドレス演算の簡略化という，RISC をベースとした HPPA 命令セットの利点をおおいに活用している．

3.1 結合命令

命令セットは基本的に3種類のオペレーション（データ変換，データ移動，コントロール）に分類されるが，HPPA ではほとんどの命令が1サイクルで2種類のオペレーションを実行できる結合命令 (Combined Instruction) として設計されている．データ変換とコントロール・オペレーションの組合わせで例をあげると，これには二つのタイプがある．

特定のコントロールを備え一般的な変換動作を行う命令と，この逆の組合わせの命令になるが，最初の例が ADD 命令であり，条件により次命令をスキップする．

LOAD と STORE 命令は変換動作（アドレス生成と修飾をともなう）とデータ移動（汎用レジスタと主記憶間）を組み合わせている．HPPA の結合命令では，その変換部分も図-3 のような単純な実行エンジンで処理できるため，実行エンジンを効率良く使用できる．

3.2 アシスト命令

このアーキテクチャでは，「アシスト命令」により命令セットの拡張を可能にしている．アシスト命令とは，プロセッサもしくは主記憶とアシスト・ハードウェア間で決められるデータ移動機能をもつ命令であるが，データ変換機能は特定されていない．アシスト命令を指定することにより拡張命令を定義すると，データ変換はアシスト・ハードウェア上で実行される．アシスト・ハードウェアは，アシスト・命令群を高速処

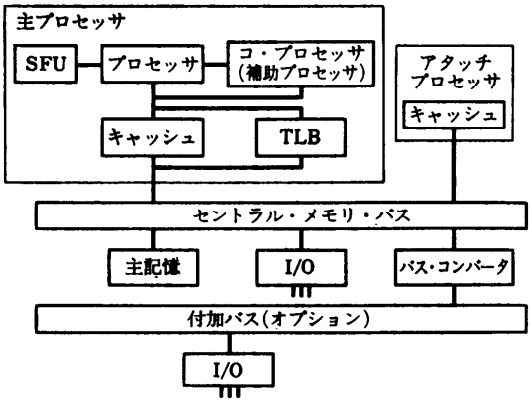


図-2　HPPA のプロセッサ・モジュール

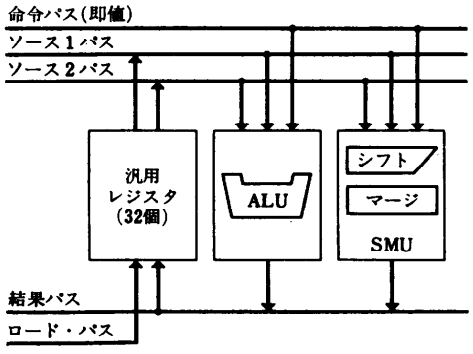


図-3　HPPA の実行データ経路

理するためのオプションのハードウェアである．これが実装されていない場合には，ソフトウェアでのエミュレーションが行われる．

アシスト・ハードウェアは，SFU（特殊機能ユニット）と補助プロセッサに大別される．SFU は，汎用レジスタをオペレーションのソースおよびターゲットとして使い，基本プロセッサやレジスタ・バスと密接な関係を維持する．一方，補助プロセッサの場合は，主記憶または補助プロセッサ独自のレジスタをオペレーションのオペランドやターゲットとして使用する．汎用レジスタで直接扱うのには大きな値を操作するのに適しているのが補助プロセッサである．この適応例が，浮動小数点コ・プロセッサ用の命令である．

3.3 無効化（NULLIFICATION）

HPPA では直後の命令の「無効化」と呼ばれる特徴がある．命令が無効化されると NOP (no-operation) として実行され，その命令は存在しなかったかのように扱われる．つまり，命令の無効化があったとしても，アーキテクチャ上では，汎用レジスタにも，主記憶にも，制御レジスタ，空間レジスタにも影響を与えないことになる．また，この無効化によりトラップを起こすこともないし，以下に続く命令を無効化することもない．分岐命令とデータ変換命令のすべては，次に実行する命令を無効化することができるようになっている．

分岐命令の場合は1ビットの無効化フィールドをもっている．無条件分岐ならば「常時無効化」か「無効化禁止」を，条件付分岐ならば遅延命令実行の「条件付無効化」か「無効化禁止」を，このビットにより指定する．無効化禁止の設定は，分岐命令の有無にかかわらず遅延命令が常時実行される状態になるときに行われる．以下に ALU 命令での無効化の例を示す．

高級言語で if (a<b) then b=b+1 と表現される場合，HPPA のアセンブリ言語では次のようになる．

```
SUB,>=a,b,r 0 ; 汎用レジスタ a の値から汎用
                レジスタ b の値を引き，結果
                は捨てる（常に値0のレジス
                タに書き込む）．そして，レ
                ジスタ a の値>=レジスタ b
                の値ならば次にくる命令を
                「無効化」する．
ADDI  1,b,b   ; 汎用レジスタ b に即値1を加
                え，結果を汎用レジスタ b に
                書き換える．
```

3.4 ミリコード

HPPA 上で使用するコンパイラと関連するが，コード・サイズをコンパクトに保つため，繰り返し使用されるような機能は特殊な共有可能な形式のライブラリ・ルーチンとして登録されている．このライブラリ・ルーチンはミリ・コードと名付けられており，ちょうどプロシージャと同じような一連の命令のパッケージとなっている．ミリ・コードはコールされて使われるが，状態の保存などの情報の受渡しは原則的に行われない．このライブラリはシステム上に唯一のコピーしかなく，全プロセスが共有できるものである．たとえば，「22バイト分の空白を埋める」という場合には，一語に及ぶ空白の操作と（一語の中の）部分的な空白の操作との二種類に大別される．すなわち，5語の空白操作と2バイトの空白操作になるが，5語の空白操作は繰り返し使用されるステップであり，これがミリ・コードで実行される．こうすることにより，この操作を行う先頭アドレスをロードする，（空白を埋める）文字のロード，ミリ・コードへの分岐，ミリ・コードからの復帰という4ステップのみがオーバヘッドとなるだけで済ませることが可能となる．

4. むすび

パイプライン処理などの他の関連技術については，ここでは説明を省く．他の HPPA に関する文献・資料を参考としていただきたい．また，現在までに製品化されている HPPA プロセッサ（HP 9000 シリーズ 800 モデル 855/850/840/835/825，HP 3000 シリーズ 955/950/940/935/925 など）の個々のインプリメンテーション・レベルなどに関しては，それぞれの資料を参照していただきたい．

（昭和63年7月27日受付）