IA-64プロセッサ

小巻　千洋
深堀　孔明

IA―64プロセッサの特徴

・明示的並列計算（EPIC）
　どの命令を並列処理するかを明示的に指定できる。

・分岐予測性能の向上
　プレディケーションなど

・メモリ・レイテンシの軽減
　投機実行（Advanced Loadなど）

・IA−32との互換性
　IA-32の命令を実行することが可能。

レジスタサイズ

汎用レジスタ(gr0 ～ gr127)	64 bit + NAT bit
浮動小数点レジスタ(fr0 ～ fr127)	82 bit
プレディケートレジスタ(pr0 ～ pr63)	1 bit
分岐レジスタ(br0 ～ br7)	64 bit
アプリケーションレジスタ(ar0 ～ ar127)	64 bit
制御レジスタ(cr0 ～ cr127)	64 bit

命令フォーマット

[(px)] op[hints] dest = src1[, src2]

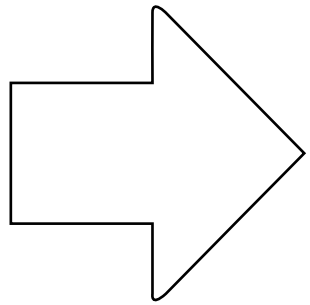
・命令は固定長(41bit)、3命令128bitを一単位として処理
(5bitは各命令の型を決めるためのもの)
・[ ]の部分は命令によってあったりなかったり。

・px　　:　修飾プレディケート
・op　　:　命令名
・hints　:　命令の補足情報
　　(例) ld4 r5 = [r33] の赤字部分
　　(r33が表すアドレスから4byteだけr5にロードする)
・dest　:　結果の代入先
・src1, src2　:　命令の入力値

プレディケートレジスタと命令

IA64のプレディケートレジスタにはtrueとfalseが保存されている。
また命令はいずれかのプレディケートレジスタを指定している。
各命令はそのレジスタの真偽によってコードが必要かを判定される。

```
if ( a<0 ){
    b = c + d;
}else{
    b = e + f;
}
```



```
cmp4.lt pr0,pr1 = 0 ,a

(pr0) add b = c , d
(pr1) add b = e , f
```

CPUにおいて条件分岐の失敗は大きな時間の損失になる。
IA64はその損失の元である分岐自体をなくし、
隙間なくコードの実行を行い並列性をあげ、高速化をしている。

Advanced Load

IA64にはポインタを使用したプログラムを高速に動かす命令がある。
ポインタを使用した場合ポインタの先が変更された場合に備えて、
何度も読み込みなおす必要がある。
しかしIA64は読み込みが成功するたびにテーブルに記録でき、
値を変更するたびにテーブルから削除される。
テーブルに記録があれば読み込む必要はない。

例: ld4.a a = [b]

テーブルに記録しつつaにbの指すアドレスの先の値を読み込む。

例: ld4.c a = [b]

テーブルの記録を調べて必要があれば読み込む。

N個の要素のベクトル同士の内積

実機でテスト。(Intel SR870BH2)
10万個の要素を持つベクトル2つの内積の計算

(1.0/100000, 2.0/100000, ・・・, 100000.0/100000)
* (1.0/100000, 2.0/100000, ・・・, 100000.0/100000)

を1000回実行するのにかかった時間。
inner_c.sにinner_c(int n, double* a, double* b)という関数を定義

[実行結果]

```
fukahori@season:~$ gcc inner_c.s test.c -o inner_c
fukahori@season:~$ ./inner_c
answer = 33332.833335
time   = 1.323241 sec
```