# ＡＩワークステーション（ＷＩＮＥ）の改良

3N-3

ハードウェアによるＬＩＳＰの高速化

五十嵐　強　　河込　和作　　瀬川　清
株式会社　東芝

## 1．はじめに

近年注目を浴びている人工知能の実用化に向け，我々は応用システムの１つとしてＡＩワークステーション（ＷＩＮＥ）の開発を行なっている。ＷＩＮＥはＲＩＳＣ風ハードウェアとマイクロプログラム制御を組み合わせることにより高速なＡＩ言語処理と充実した開発環境をユーザに提供する。６２／下に本学会で発表したときはｐｒｏｌｏｇをメインに開発したが，一方ソフトウェアの蓄積の多いＬＩＳＰも重要視されている。そのため今回ＬＩＳＰの高速化を主なる目的としてＡＩプロセッサ（ＡＩＰ）のハードウェアにいくつかの改良を行った。本稿では従来のものと改良版とを比較しながらその改良点の説明とベンチマークテストの結果についてのべる。

## 2．ハードウェアの改良点

ハードウェアの立場から見ると，汎用言語と比べＬＩＳＰが特徴としているのは次の点である。

(1) タグによるデータ処理の多様化
(2) ｂｉｇｎｕｍ処理
(3) ガーベジコレクション（ＧＣ）
(4) 特殊シンボル　ＮＩＬ

### (1) タグによるデータ処理の多様化

タグの値による処理はタグのチェックとデータの処理を同時に行ない，タグがマッチしなかった場合にはデータの処理をアボートするのが効率的であるが，従来はメモリアクセス動作をアボートする機能がなかったため余分なステップが必要であった。例えば CAR　命令

```
CAR   Rd,Rs,0      ( car Rs --> Rd )
```

は従来では次のようにマイクロ命令で実行する。

```
add   ,w0,ms1,,nl,0,ml3,rw,d,hep        ①
lau   md,,ms1,w0,ncons,cxr-next,abdn    ②
```

① Rs の指すメモリの内容をワークレジスタ w0 に read する。
②もし Rs のタグが cons であれば w0 の内容をRd にしまい終了する。 cons でなければ w0 の内容は Rd にはしまわず（アボート） cxr-next へいく。

X が cons ならば２ステップ（３クロック）かかる。しかし改良版ではメモリアクセスをアボートする機能が追加されたため従来のように一度 w0 レジスタにもってくる必要がなくなり，１ステップ（１クロック）で終了する。

```
add   ,md,ms1,,ncons,cxr-next,ml3,
                              rw,d,hep,abfm,dn   ①
```

①もし X が cons ならば X の指すメモリの内容を Rd に read して終了。 cons でなければ read 動作をアボートして cxr-next にとぶ。

### (2) ｂｉｇｎｕｍ処理

数値演算時の多倍長拡張処理も(1) と同様に演算と overflow の判定を同時に行なうようにした。例えばジェネリックな加算命令

```
GADD  Rd,Rs1,Rs2    ( Rs1 + Rs2 --> Rd )
```

において，従来では overflow フラグは他のフラグと同様に条件分岐とアボートにしか影響しなかったが，改良版では overflow 発生時にハードウェアトラップをかけるとい

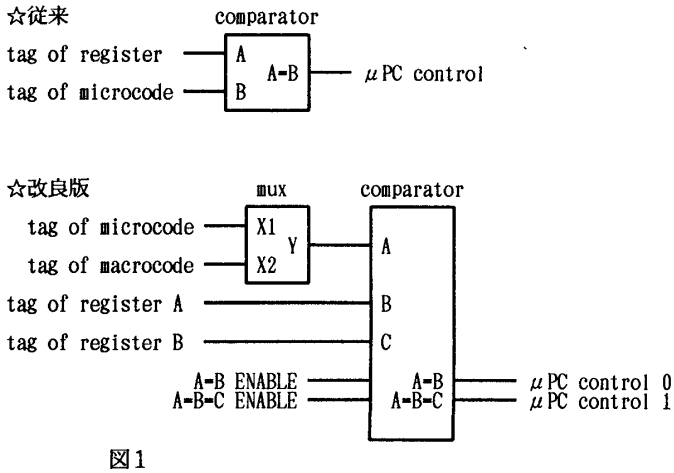


図1

---

AI Workstation(WINE) LISP hardware
Tsuyoshi IGARASHI, Kazusaku KAWAGOME, Kiyoshi SEGAWA
TOSHIBA Corporation

う指定を可能とした。また(1) にも関係するが，タグの比較は従来では１つのレジスタとマイクロ命令のタグ指定フィールドとの比較しかできなかったが，改良版では２つのレジスタとマクロまたはマイクロ命令の３つのタグの同時比較を可能とした（図１）。
従来では，Rs1,Rs2 とも fixnum の場合，

```
      lau  ,,ms1,,a1,mwj1,mw                    ①
fix1: padd  w0,,ms1,ms2,a1,fix2,sx              ②
fix2: lau   ,ms2,,nfix,tag-unmatch              ③
      lau   md,,,w0,v,overflow-exception,abdn   ④
```

① Rs1 のタグの内容でマルチウェイジャンプする。
② Rs1 + Rs2 の結果を w0 にしまう。
③もしRs2 が fixnum 以外なら tag-unmatchにとぶ。
④ w0 の内容を Rd にしまい終了。もし overflow ならばアボートして overflow-exception にとぶ。

改良版では，

```
      padd  md,,ms1,ms2,ndfix,gadd-next,
                                bot,abf,sxdn   ①
```

①Rs1 + Rs2 の結果を Rd にしまう。 overflow が発生したらトラップをかける。もしRs1 とRs2 のうち少なくとも一方が fixnum でなかったらアボートし，gadd-next へとぶ。

となり，この場合は４ステップが１ステップとなる。

(3) ガーベジコレクション（ＧＣ）
論理空間はアドレスの上位２ビットにより４つの空間に分けられる（図２）。そのうち２つは新空間と旧空間とよばれヒープ領域である。ＧＣ時には旧空間に存在するオブジェクトは新空間に移動され，それが終了すると新／旧が入れ換わる。改良版ではリアルタイムＧＣを高速に行なうためのハードウェアをサポートしており，ユーザはＧＣを行なう必要があるかどうかの判定は不要である。 cons 等

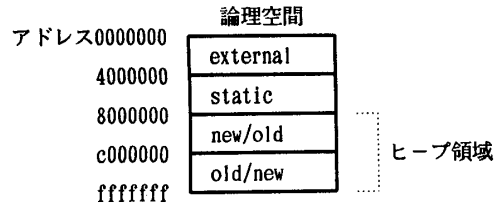

図2

のオブジェクト生成命令はＰＳＷ中のＧＣモード（ＧＣすべき）であるというフラグ（ＧＣＭ）によってＧＣ処理を含むマイクロ命令にハードウェア的にすりかわる。どちらが新か旧かというフラグ（ＬＳＩ）もＰＳＷレジスタにおかれ，レジスタの指すオブジェクトがどちらの空間に属しているかはコンディションとして判別できる。

(4) 特殊シンボル　ＮＩＬ
ＮＩＬはシンボルであるが，ほかのシンボルとの区別を高速にするため，そのポインタ部はＬＳＢを１とした（図３）。 改良版ではこれを検出するコンディションフラグ（ＯＤＤ）も追加したため，マイクロ命令中であるレジスタの内容がＮＩＬであるかどうかは，タグがシンボルでかつＯＤＤであるという条件で済む。

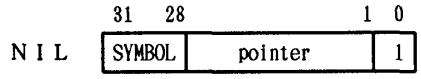

図3

## 3. 結果
以上のような改良を評価するため，ベンチマークテストを行なったところ，従来に比べ２～３倍性能の向上が実現できた。ただしこの値はハードウェアのみでなくコンパイラの性能も大きく関わっているため，純粋にハードウェアのみの数字ではないが，ハードウェアの改良によりシステム全体でこのくらいよくなることがわかった。

| | 従来 | 改良版 | 性能比 |
|---|---|---|---|
| Destructive (sec) | 0．9 (0．4) | 0．3 (0．2) | 3．0 (2．0) |
| Browse (sec) | 9．0 (5．0) | 4．0 ( 不明) | 2．3 ( － ) |

(括弧内は型宣言を行なった場合)

## 4. おわりに
今回はハードウェアの改良によって性能がどの位向上したかをベンチマークテストの数値で示したが，今後コンパイラ等の改良によりさらに性能を上げるよう研究を続けるつもりである。

## 参考文献
1）相川，他：ＡＩワークステーション（ＷＩＮＥ）の開発Ⅱ　ＩＰ７０４のアーキテクチャ，　情報処理学会第３５回（昭和６２年後期）全国大会