たかがメモリされどメモリ

中村　宏
東京大学 先端科学技術研究センター

最近，記憶を辿るのに手間取ることが増えた私のこだわりは，メモリである．記憶を辿るのが速い人間は仕事も速い，というのは真理であろう．計算機の高速化にも，メモリシステムの高速化は必須なのである．この問題に対する万能薬としてキャッシュが広く用いられている．キャッシュは自動制御の便利な機構であるが，自動制御ゆえに性能上の犠牲が生じる場合がある．そこで，さらなる高速化のためには，ソフトウェアが制御可能な小容量高速メモリを用いることを検討したい．

こだわりは研究の母

こだわり，というのはある種独善的なものであり，他人のこだわりを理解するのはなかなか難しい．まして他人のこだわりに共鳴・共感できることはそうそうない．ちなみに私の居室は，机の上は種々の書類が微妙なバランスの上に(要するに積めるだけ)積み上げられているが，床の上はきれいに掃除してある．これも私のこだわりであるが，こだわり以前の問題だという指摘もある．

まあ，そんなことはどうでもよいのだが，アーキテクトも皆さまざまなこだわりを持っている．処理能力を向上するためにはいろいろな検討点があるはずだが，どこを突いていくか，は多分に個人のこだわりによるところが多いし，各人のそのこだわりこそが，研究の大きな推進力となっているのは間違いないと思われる．あんまり変なこだわりを持っているとだんだん周囲から相手にされなくなるらしいが，私は違う，と自分では思っている．で，私のこだわりはメモリである．

メモリは永遠のテーマ

計算機アーキテクチャの歴史はメモリとの戦いである，と言われる．現在の計算機が命令とデータをメモリに格納するstored program方式を採用しているがゆえの宿命といえばそれまでだが，メモリとプロセッサの間のスループット不足が性能上の隘路(von Neumann Bottleneckとか Memory Wallとか呼ばれる)となるのである．私も最近仕事の能率が悪いのだが，仕事の処理自体が遅いというよりは，「なにするんだったっけ？」と記憶を辿るのに時間をとられていることが多いようで，若くしてボケたといえばそれまでだが，記憶を辿る，というのは，計算機の本質的な処理なのである．この喩えは個体差もあるので，別の喩えを用いるなら，図書館から文献を借りて調査するとき，どれだけ頭が良くても(どれだけ調査能力があっても)図書館から本を借りてくるのに時間がかかるので(一度にそれほど運べないし，図書館は物理的に遠い)本を借りる処理こそが全体の処理速度を決めてしまう，ということである．

この問題は，メモリに階層を導入する，ということで解決が図られてきた．すなわちプロセッサに近いところにキャッシュと呼ばれる小容量高速メモリを導入するのである．どうせ小容量であるから全部は入りきらないのであるが，たいていの処理においては命令やデータの参照には局所性(locality)があるからそれで事足りる，というのがその根拠である．局所性には時間的局所性と空間的局所性があり，前者は，一度参照されたものは近未来において再び参照されるという性質，後者は，あるものが参照されるときにはその近辺のものも参照されるという性質である．この性質があるので，新しい命令・データを参照するときには，近所にある命令・データもひっくるめて，遠くて遅い主記憶から近くて速いキャッシュへ持ってきておく．そうすれば，空間的局所性により近所の命令・データが，そして時間的局所性により近未来に再びそれらの命令・データは参照されるはずである．その時は速いキャッシュメモリを参照すればよいので，ぐっと処理性能は向上する，という仕掛けである．確かに，図書館から本を借りるとき，必要な部分だけ借りるよりは，その前後も読むかもしれないので適当にまとめて借りておきたいし，借りた本は一度用が終わっても，またすぐに見るかもしれないので，とりあえず机の上に数冊でも置ければ，図書館に行く回数が減り仕事の効率はぐっと上がるであろう．

私の場合，それでも机の上に置いた本を探すのが大変だったり，本や書類がたまる一方で机の上で溢れかえったりするのであるが，キャッシュはその点での心配はない．まず，置く場所に制約を加え，本の種類・番号に応じて机の上で置けるところを決めておく．後から探すとき，ここあそこだけを

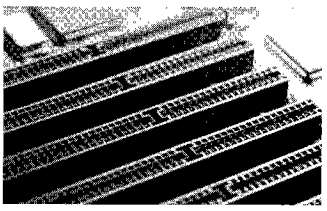
探せばよいように整理しておくのである．それから，溢れ問題に対しては，最近参照しなかったものを自動的に選び出し，それを主記憶に自動的に戻してくれる．これはなかなか便利で，私の机もしばらく見なかった書類を自動的に捨ててくれたり，本を自動的に本棚に戻してくれれば，私の机の上もきれいになるであろうし，机からはみ出て落ちて初めて目を通してもらえる不幸な(大抵締め切りを過ぎている)書類も減ろうというものである．

万能薬にも限界はある

驚くべきことに，一見単純に思えるこのキャッシュが思いのほか効くのである．種々の命令セットを持ついろんなプロセッサがあるにもかかわらず，省コスト指向から高性能指向のマイクロプロセッサ，さらにはメインフレームも含めてそのどれもがほぼ同じような制御をするキャッシュを採用していることから，「局所性があるから」の一言で理由づけされるその効用が大変広いことが分かっていただけるであろう．キャッシュはハズレのない万能薬なのである．

しかし，ベクトル型スーパーコンピュータはデータに関してキャッシュを用いない．スーパーコンピュータが対象とする大規模な科学技術計算においては，データ参照に時間的局所性がない，あるいは，あっても次に再利用されるまでの時間が非常に長いため，せっかくデータをキャッシュに置いても次回の利用時にはキャッシュから勝手に追い出されているからである．

スーパーコンピュータがベクトル型から超並列マイクロプロセッサ型へ移行した際，このメモリ階層の違いに多くのユーザが悩み，キャッシュがうまく働くように，すなわちデータ参照の局所性が高くなるように従来のプログラムに変更が加えられた．しかし，万能薬であるはずのキャッシュに自らを合わせないといけない，というのはどうも本末転倒である．どういうメモリ階層を採用しようと局所性が高いプログラムの方が性能が出るであろうから，たとえ本末転倒でもその変更をすること自体は良しとしよう．しかし，その



変更方法が非常にアドホックであり，たとえばキャッシュの大きさが変わるとプログラムを変更する必要が生じたりする．これは，先の図書館の喩えに戻れば，机の大きさを計算に入れて本を借りる順番を調整するようなものである．そこまで考えるのであれば机の上の本の置き場所をもう少し自由に考えたり，机の上からどの本を返却するかを自由に考えた方がよっぽど健全かつ効果的であろう．しかし，キャッシュの場合，どのデータをどこに置くか，あるいは，どのデータをメモリに戻すかは基本的にはユーザからは制御不能であり，考えることが許されていない．

こだわるべきは

かくして，私はもう少し痒いところに手の届くメモリ階層をユーザに提示するべきではないか，つまり，置き場所と返却物を自由に選べる・指定できる小容量高速メモリ(たとえばプロセッサチップ上に実装されるアドレス指定可能なメモリ)を用意するべきではないか，と思うのである．これは従来のキャッシュの代わりに，ではなく，キャッシュとは別に用意する，ということである．聞くところによれば，Intel Pentium IIIでは，データ参照に再利用性がないマルチメディアデータ用にキャッシュをバイパスするデータ転送命令を用意している(メモリストリーミングアーキテクチャと呼ばれている)．この手法の是非にはいろいろな意見があろうと思うが，ユーザにメモリ階層(の一部)の制御を許す，という方針自体は，これからさらに広まるのではないか，と思う．個人的には，プロセッサチップ上のメモリでデータの再利用性をより積極的に活用できるようなハードウェア機構が必要ではないか，と思っている．

本質的には，ソフトウェアとハードウェアの境界(これを計算機アーキテクチャの定義とすることもある)をどこに設定すべきか，という議論が必要であろう．ずっと以前は命令セットこそがその意味での計算機アーキテクチャであったが，近年はパイプライン／スーパースカラといった命令実行方式もこの意味でのアーキテクチャに含まれており(最適化コンパイラはその情報をフルに活用する)，この境界は変化している．そして，次はメモリ階層がこの境界を超えてくると私は主張したい．メモリに対する私のこのこだわりが受け入れられるか否かは，市場原理に従い，その効果と適用範囲が決めるところであろう．適用範囲は，大規模科学技術計算のようにアクセスパターンが規則的な処理である．規則的という意味では手堅い世界であり，今年の2月号の本コラムにあった中島氏の「投機」の余地がさほどない処理である．正確には，処理が規則的なためしょぼい予測が結構効果的であり，投機がおいしいところはすでに食べ尽くされている，というところであろう．しかし，大規模科学技術計算では，性能に対する要求が非常に厳しく，メモリ性能こそが処理性能を決めている現状では，この私のこだわりは，効果抜群で結構受け入れられるのではないかと期待している．

おわりに：カスタムメイドの机はいかが？

半導体技術の進展によりプロセッサ内部の処理は高速化されてもメモリの速度はさほど向上しないので，冒頭に述べたMemory Wallの問題は今後より深刻になると言われている．我々アーキテクトが使える武器は「集積度の向上」しかないので，キャッシュを大容量化する，という解決は，まったく理にかなった解決方法であり，実際これまでのプロセッサ技術のトレンドを見てもそうであった．しかし，それだけでは単に机を大きくしていっぱい本が置けるから仕事が進むだろうという発想と何ら変わりがなく，芸がない気もする．そろそろ机の大きさだけではなく，机の使い方を考えたいものである．お仕着せの机ではなく，あなたの使い勝手の良い机をカスタムメイドなさいませんか？

(1999.8.1)

予測は難しい

中田 登志之
NEC C&C メディア研究所

あれだけ，世の中を騒がした1999年7の月も無事に過ぎて，暑い8月になりました．さて，中村氏が，「メモリは永遠のテーマと」おっしゃっていますが，これだけプロセッサとの速度差が広がると，メモリ階層なんとかならないかというのは計算機アーキテクトには切実な問題です．実際，私はマイクロプロセッサベースの並列計算機の研究を行ってきましたが，キャッシュにヒットすれば50%の効率が得られても，キャッシュから出たとたんに効率がピーク性能の5%から10%に落ちてしまうことはしょっちゅうあります．

さて，議論の整理のために，メモリの速度に関する問題を少し整理してみます．メモリの速度を議論する場合は，遅延(レイテンシ)とバンド幅(スループット)の2種類の観点から議論しないといけません．遅延は最初のメモリアクセスを要求してから実際にデータが到達するまでの時間です．またバンド幅は1秒間にいくつのデータが送られるかというものです．

それで，ベクトルスーパーコンピュータと比べた場合，遅延の克服はなんとかなるでしょうが，バンド幅の克服は非常に難しいものがあります．たとえば，A(I)=B(I)+C(I) という評価式(倍精度)を1000万要素に関して行おうとすると，1回の演算に，2回のデータロードと1回のデータストアが必要です．500MFLOPSのPCの演算器をフルに作動させようとすると，500M*(2+1)*8(バイト(倍精度))=12GB/secのデータバンド幅が必要です．市販の100MHZ FSBのマシン(データバンド幅800MB/sec)では到底追い付かないことになります．

A，B，Cが再利用されない場合は，バンド幅を物理的に広げる以外に手はありません．しかし，A, B, Cのデータが再利用される場合は，再利用されるまで，どこかにデータを保持しておくことで，疑似的にバンド幅をあげるように見せることができます．これがキャッシュメモリの効果というものです．

ところが，中村氏も言われている通り，キャッシュメモリには容量的な限界があり，大規模な科学技術計算ではキャッシュメモリが役に立ちません．この点に関して，中村氏は，ユーザが制御できる小容量高速メモリを提案しておられます．これはRISC的な思想をメモリ階層にも適用しようと解釈することができましょう．この文章を見た時，15年程前に，「アーキテクチャ屋は何も凝ったことをせずに，ソフト屋(コンパイラ屋)にすべてを触れるようにしろ」と，とある人に言われたことを思い出しました．若干のコメントをさしあげます．

a) 小量のメモリで，レイテンシの削減を測ることは可能でしょうが，大規模科学技術計算では，アクセスすべきデータが膨大なので，小容量のメモリで本当に有効に扱えるものはあるのでしょうか．机の上に大きなメモ用紙を貼っても，図書館から100冊も本が送られてきて全部読まないといけないのだとしたら，メモ用紙はやはり，真っ黒になるだけなのでは，ということです．

b) MIPS社のR10000などでは，Prefetch命令が存在して，これでデータを先にとってきてキャッシュに置くようなことはもう可能です．本当にスループットネックの場合は，これでも結構大変ですが，レイテンシを稼ぐことはある程度できます．

c) また，バンド幅の克服への提案としては，DRAMロジック混在で，メモリへのスループット自体をあげるという議論がなされています．これに関しては1チップに入らない場合はどうするかということをしっかり見極めないといけません．

c)に関しては，いろいろな人から議論がでるとして，もう1つ議論のために必要なのは，a)をとるにせよ，b)をとるにせよ，「賢い」コンパイラが，うまくデータの参照パタンを予測して，良い命令を発行しないと宝の持ち腐れになるのではないか，ということです．いらないデータを間違ってとってきたら，ますますバンド幅を苦しくしてしまいます．コンパイラがアクセスパタンをうまく読み切って，データを必要とする前にとってくることが必要です．

「ノストラダムスは失敗しましたけど，私はあなたが次に読みたいという本をずばり当ててさしあげます．」という方おられないでしょうか？(そういう人はもう株で大儲けしてるって？)

(1999.8.16)

いい机よりいい秘書を

上原 哲太郎
和歌山大学システム情報学センター

私は，自称「アカデミックの世界ではベクトル型スーパコンピュータを日本一愛しているコンパイラ屋」である．

大きな平机とポストイット

中村氏の論旨は，私なりにまとめると，「プロセサチップ上にアドレス指定

可能な小容量高速メモリを設けて Memory Wallを克服しよう」である．コンパイラ屋の立場から，そういうメモリが与えられた場合どう使うかといろいろ考えたが，結論からいうと，そういう高速小容量メモリを与えられても持て余す場面も多く，もったいないと感じた．大雑把にいうとプログラムの大半を占めるキャッシュが効く場面では，結局，最適解を得るためにはキャッシュと同じことをその高速メモリにもやらなくてはならず，これは静的解析しかできないコンパイラにはとても面倒な仕事である，ということだ．よって多くの場合は，せっかくの高速メモリ領域もほとんど無駄な領域として放っておかれそうである．それなら，そのようなメモリを持つのをあきらめて，そのチップ面積をキャッシュに返しておいた方が大半の場面では役に立つであろう．

また，もしコンパイラに，キャッシュだけでは知り得ない情報が有り得るとしても，キャッシュに適当な制御命令があるだけでそれなりに頑張れるだろう．キャッシュのありがたいところは，何も考えずとも勝手にデータで埋まってくれて，適当に効果が現れることである．コンパイラ屋は，データ参照パターンの解析がうまくいったときには，レジスタ，キャッシュ，主記憶の3階層分の最適化を考えて，参照順序の工夫とキャッシュ制御命令を活用して性能向上を引き出すことができるだろう．もし解析がうまくいかないときはキャッシュにベッタリと甘えることもできる．

要するに，中村氏は「机は半分に仕切って，右には1週間以内に決済する書類，左にはその他の書類を置きましょうね」とおっしゃってるのではないかと思う．残念ながらズボラな私は，そのご期待に沿えず，きっと左半分だけで作業してしまうだろう．近々使うなぁと思った書類にポストイットを貼るだけで済ませるから，わざわざ右半分に整頓し直したりしない．そのうち左半分にはうずたかく書類が積まれてゆき，右半分は空しくガランとあいて，いつしかカタストロフィーが....

ということで，もし机の面積が決まっているのなら，あまりそこを用途別に仕切らない方が結局は柔軟に有効利用できるのではないか．凝ったデザインのシステムデスクより単なる平机の方が，少なくとも私は使い良い．まぁ時々机から勝手にこぼれ落ちる書類があるかもしれないが，大事なものはポストイットでも貼ってそれを落とさないようにする(キャッシュ制御命令を使う)くらいで，そこそこやっていけるだろう．ちょっとした要望があるとしたら，書類の用紙は小さめがいい(つまり，キャッシュラインを短めにして欲しい)．その方が整理整頓時に融通が効く．

いずれにせよ私は，ある特別な用途のために机の一部をあけておくのはもったいないのではと思う．それより何より，机にデータが載りきらない，つまり参照局所性がまったくなくてキャッシュがしょせん効かないときには，性能低下があまりにも激しくて，机の上の特別な領域も大した助けにならないのではないかと思う．もちろんその元凶は主記憶である．

だから，メモリに対する私のこだわりは机そのものではなく，書棚にある．その書棚から机の上に，だれがどうやって書類を運んできてくれるかにある．

書類は秘書に持ってきてほしい

現在の多くの計算機アーキテクチャでは，主記憶とキャッシュ間のデータ転送はキャッシュの管理単位であるキャッシュラインを単位として行われている．つまり，あるアドレスへの読み出しが発生すると，アドレス空間上でそのデータの近辺にあるデータもついでにキャッシュに持ってきてしまう．この方が，アドレス指定のオーバヘッドが省けるため，見かけ上のバンド幅が向上する．

机の例に言い換えれば，書類(データ)は全部バインダー(キャッシュライン)に束ねて整理してあって，書棚(主記憶)から机(キャッシュ)に書類を運ぶときはバインダーごと運んできてから必要な書類を読む，といえるだろう．実際には，ある書類を読むときに，そのバインダーに閉じてある他の書類も読むことが多い(参照の空間的局所性がある)ため，バインダーごと持ってきても無駄にならないことが多い．ところが，中村氏も述べているが，空間的局所性も時間的局所性もない，キャッシュにとって一番厳しい条件のメモリ参照というのも世の中にはあって，キャッシュが役に立たない．

私が愛してやまないベクトル型スーパーコンピュータなら，この問題が起きない．ベクトルスパコンは，主記憶を多数のバンクに分割することにより，非常に高いバンド幅でのデータ転送を実現している．ベクトルスパコンは演算性能の高さがよくいわれるが，これを本当に高性能たらしめているのは主記憶性能の高さといってよい．

主記憶のバンド幅を上げることは一般にレイテンシの低下を招くが，ベクトルスパコンはデータをベクトルという大きなデータの塊を単位として処理するので，パイプライン化によってレイテンシが隠しやすい．しかもベクトルスパコンはいくつかのアドレッシングモードを持ち，連続アドレス，等間隔アドレス，ベクトル間接アドレスといったいくつかの形式で主記憶上のベクトルを表現できる．つまり，必要な書類だけをまとめてバインダー(ベクトル)に綴じてすばやく届けてくれる有能な秘書がついており，また，上司(プロセッサ)も，必要な書類をかきあつめる方法(アドレッシングモード)を的確に秘書に伝えることができるのである．

行列計算のような大規模数値演算では，キャッシュは効かないが，参照パターンはコンパイラにもかなり正確に予測可能である．これは中村氏もご指摘の通りである．ただ，その予測できた参照パターンを生かして，必要なデータを無駄なくすばやくプロセッサへ持ってくるための手段が，今のパソコンやワークステーションのプロセッサやメモリには提供されていない．これではどうしようもない．

主記憶よもっと頑張って

急速に性能が向上するプロセッサに追い付くため，主記憶とプロセッサとの間のバンド幅向上にいろいろな技術が投入されている．これらはいずれも，レイテンシを犠牲にしてもバースト転

送時のバンド幅向上を目指している．これは技術的に見て仕方ないだろうと思う．問題は，絶対性能が全然足りないことである．プロセッサ性能に対して10倍以上遅いという状況は要するに主記憶を使った計算はするなといわれているようなものだ．さらに，これらのメモリがいずれも連続アドレスに並んだデータのバースト転送しかできず，プロセッサのキャッシュのラインを埋めるという動作に特化しているのも問題だ．こういう方向だけにメモリ技術が成長してゆくと，キャッシュが効かないアプリケーションを抱えるユーザはどんどん見捨てられていってしまう．

キャッシュに限界があることが分かっている以上，主記憶も，それ以外の場合に対応した「芸」を身につけて欲しいと思う．まず，これはプロセッサ側の問題だが，外部バスの絶対性能がもっと欲しい．レイテンシはある程度あきらめるから，せめてバンド幅だけでもプロセッサのピーク演算性能に比べ，1/4ぐらいのペースでデータ供給できるようにして欲しい．その際，システムが連続アドレスのバースト転送に特化していては前述のような問題が起きるので，ベクトルスパコンにならってもう少し柔軟な参照パターンを指定できるようにして欲しい．メモリモジュールも，そういうさまざまなパターンのメモリ要求に応じて，データを送出する能力が欲しい．

とまぁ，ここまで書くと結局マイクロプロセッサでベクトルマシンを作れ，という結論になりそうだが，私自身はプロセッサの実装はベクトル型でなくてもよいと思っている．ただ必要なのは，「これからどのようなパターンのメモリ参照が発生するか」をプロセッサからメモリへと伝える工夫である．ベクトルスパコンではベクトル参照命令がまさにそれだが，別にキャッシュあるいは中村氏が述べている高速メモリへのpreload命令のような実装でもかまわないと思う．要するに，キャッシュが効かない場面への対応のためメモリ回りに何か工夫をするなら，まずベクトルプロセッサで培われたノウハウを生かすべきではないかと思っている．

コンパイラ屋は，ベクトルプロセッサ部屋では，秘書を使って仕事をするのに慣れてきたのである．だが，新しく送り込まれたRISCプロセッサの部屋では，秘書にどうも話が通じない．ふとよくみると，みんなキャッシュという机の上で，自分だけで仕事をしているらしい．無能な秘書に文句を言う人が少ないのは，皆あきらめてしまったからだろうか？　それとも有能な秘書を雇うと高くつくからか？

私は，SDRAMやRDRAMのような，連続アドレス参照だけが速いメモリでは不満である．多少お金をかけても，もっと有能な秘書が欲しい．稲妻のように必要な書類を揃え，飛ぶように運んできて，鬼のようにどっさり机に積み上げてくれる有能なRAMがいないと，ナマケモノの私は仕事がはかどらないのである．

(1999.8.17)

それでも机の使い方は大事

中村　宏
東京大学 先端科学技術研究センター

反論をせねばならないのだが，両氏の主張に私はほぼ同意なのである．「レイテンシとバンド幅の2種類から論じる必要がある」という中田氏の主張はまったく同感であるし，上原氏の「バンド幅がまず重要である」という主張にも強く賛同するのである．バンド幅が確保されていれば，レイテンシ隠蔽手法は，たとえば中田氏の言うprefetch命令などいろいろあろう．また，私は以前，擬似ベクトルプロセッサなるRISCプロセッサを提案したこともあり，上原氏の「ベクトルプロセッサで培われたノウハウを活かすべき」という主張はまったくその通り，と声を大にして言いたい．

ただ，バンド幅確保の手段に関する見解が違うようである．中田氏は「賢いコンパイラがデータの参照パターンを予測すればよい」と言っておられるが，私はあまり賛成しない．上原氏も言っておられるように，科学技術計算ではデータの参照パターンはかなり予測できているのである．分かっているのにデータをとってこれない，その理由はバンド幅不足なのである．そこで上原氏は「良い秘書を」と主張されておられるが，その意味は「多用な書類の揃え方を指定でき，飛ぶように運んで来る秘書」ということであり，その後半部の意味は主記憶のバンド幅をあげろ，ということである．それが実現できればMemory Wallに対する本質的な解決になるであろう．しかし現在の半導体技術・実装技術ではそれは不可能，つまり秘書は飛べないと私は思うのである．

机は確実に大きくなるが

現実に目を向けると，集積度は向上する．つまり机は大きくなるのである．ではどうするべきか，ここに目を向けたい．私の考えは，現状より2桁程度以上容量の大きいキャッシュが実装できるようになったら，その一部(全部ではない)をアドレス指定可能なメモリにしよう，というものである．中田氏の「データの再利用性がないと全データが載らない限り結局だめ」という主張はまったく正しいが，実際計算物理学の分野の実アプリケーションを見ていて思うのは，アルゴリズムの変更までもが許されるとしたらデータの再利用性はそれなりにかなりある，ということである．ただ，データにより再利用のされ方が違うので画一的なキャッシュの制御ではうまくいかない．たとえば，毎日見る本と月に1度見る本を同じ机の上で一緒に扱い，後者を置くために前者を机から片付けるのは無駄なので

ある．そこまで参照の性質が分かっているのであれば，もう少し使いやすいメモリをプロセッサチップ上に用意すべきであろう．コンパイラにそこまで参照の性質が分かるのか？　という質問が両氏からあったかと思うが，私はそれをすべてコンパイラで面倒見ろとは主張しない．プログラマがプログラム上で陽に書いてもよいし，コンパイラに分かるようにdirectiveを与えてもよいと思っている．プログラマが何も分からなければ別に用意してある一般向けのキャッシュを用いればよい．

机の上にいつも必要な書類・本が置いてある状況が作れれば，仕事の半分以上は終わったようなものである．これからは今まで以上に主記憶が遠い(＝本棚が遠い)のであるから，この状況を作り出すためには，秘書の運搬業務を必要最低限に抑えることが重要である．そのために，机の上の整理方法を秘書とよく打ち合わせることがますます重要になるのである．この本戻しておいて，と秘書に言ったすぐ後で，やっぱりさっきの本をとってきて，というのは，時間のロスだし秘書も疲れる．実は私はまだ秘書を雇ったことがないのであるが，もし秘書と一緒に仕事をする機会が生じたなら，机の使い方の相談をまず最初にしよう，と強く思うのである．

(1999.8.19)

～　議論の続きは，次のURLをご覧ください．http://www.ipsj.or.jp/magazine/interessay.html　～

平成12年度会誌「情報処理」表紙デザイン募集

会誌「情報処理」(IPSJ Magagine) は内容，表紙とも一新しましてご好評をいただいております．本会誌は会員のみならず，社会の幅広い層の読者のため最新の情報科学技術を分かりやすく解説しています．

つきましては，本年度に引き続き平成12年度版（41巻1号～12号）の表紙デザインを広く募集いたします．奮ってご応募ください．採用された方には賞金を贈呈いたします．

応募条件

・表紙のデザイン（裏表紙，背表紙は含まず）．
・A4判（天地297mm×左右210mm）4色フルカラー．
・デザイン画はアート紙またはコート紙に描画してください．
・描画手法は問いません．
・「情報処理」の題字，巻号，法定文字，記事タイトル等必要項目は現会誌をご覧ください．
・応募資格は問いません．
・詳細は会誌担当までお問い合わせください．

注意事項

・応募作品は返却しません．
・採用作品の掲載にあたって，学会側で多少の変更をさせていただく場合があります．
・採用作品の著作権は（社）情報処理学会に帰属します．
・応募作品は未発表のものに限ります（応募作品の知的財産権について，第三者との間に紛争が生じた場合は，作者がその責を負う）．
・応募される場合は，上記注意事項に同意されたものとみなします．

応募締切　**平成11年9月30日（必着）**
結果通知　平成11年10月中旬
賞　　金　採用者1名に20万円

送付先／照会先　情報処理学会　会誌担当
E-mail:editj@ipsj.or.jp　Tel(03)5484-3535　Fax(03)5484-3534
〒108-0023 東京都港区芝浦3-16-20　芝浦前川ビル7F