

NS32532のアーキテクチャ

栗原 敏 雄

ナショナルセミコンダクタージャパン株式会社

1987年11月に発表されたNS32532アーキテクチャの基本コンセプトを説明する。その基本コンセプトは高集積、高性能、柔軟性、互換性である。特に、NS32532のキャッシュアーキテクチャはマルチプロセッサ、モードにおいても高性能維持が可能をうに設計されている。従って、インストラクション・キャッシュとデータ・キャッシュの構成を本報告で概説した。今後、増大高性能が要求される32ビット・マイクロプロセッサ分野において、NS32532のアーキテクチャは重要な方向性を示している。

NS32532

ARCHITECTURE

TOSHIO KURIHARA

NATIONAL SEMICONDUCTOR JAPAN LTD.

4-403 Ikebukuro, Toshima-ku, Tokyo 171

National Semiconductor Corporation announced NS32532, high performance 32 bit microprocessor in Nov. '87. The paper described the basic concepts of NS32532 architecture, which are high integration, high performance, versatility and compatibility. The cache architectures are specially designed to be high performance under the multiprocessing mode. Therefore, both architectures of instruction cache and data cache are discussed in this paper. As the demand of higher performances is rapidly increasing for 32 bit microprocessors nowadays, NS32532 architecture indicates the key direction in the 32 bit microprocessors development activities.

インテリジェント・セミコンダクター社が1987年11月よりサンプル出荷を開始したNS32532のアーキテクチャをここに要約して説明した。Series 32000, 32ビットCPUファミリーの歴史は長く、下記のように1982年に最初のCPU, NS32016が開発されている。

表1. Series 32000 CPUの歴史

発表年	CPU	データバス	アドレス空間	特徴
1982	32016	16ビット	16Mバイト	高速制御システムに最適
1983	32032	32ビット	16Mバイト	低価格UNIXシステム用
1985	32332	8, 16, 32ビット	4Gバイト	高性能システム制御及びUNIXシステム用
1986	32008	8ビット	16Mバイト	低価格32ビットシステム
1987	32532	8, 16, 32ビット	4Gバイト	超高性能システム用

I. NS32532アーキテクチャの基本コンセプト

NS32532の基本コンセプトは(1)高集積(2)高性能(3)柔軟性(4)互換性である。

(1) 高集積

図1のごとく、NS32532は現存する32ビットマイクロプロセッサとして最高位の集積度を保有する。

- 完全なデマンド・ページ仮想メモリ管理ユニットの内蔵。しかも、64エントリーを保有するトランスレーショナルソフトウェアサイド・バッファ(TLB)を持っている。
- 512バイトのインストラクション・キャッシュ内蔵。
- 1024バイトのデータ・キャッシュ内蔵。
- 4段のハイプライレを用いた命令の実行。

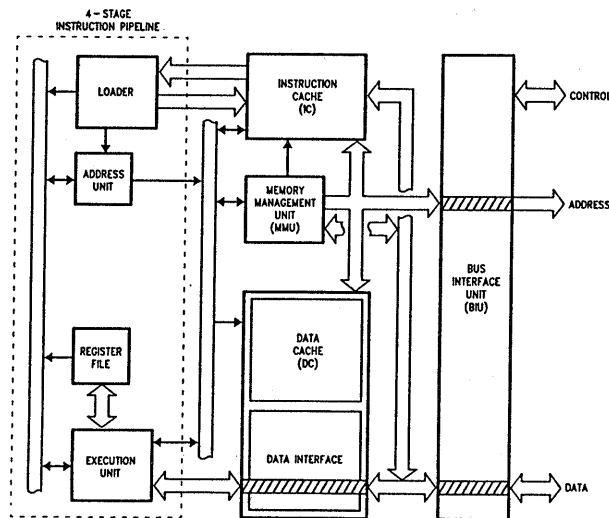


図1. NS32532のブロック・ダイアグラム

この64エントリ内蔵のTLBを所有することにより、仮想アドレスを物理アドレスに変換するのに、99%以上の可能性で、メイン・メモリ中のページ・テーブル参照なしに“HIT”した状態で高速に物理アドレスを得ることが出来る。

このインストラクション・キャッシュ及びデータ・キャッシュの内蔵により、外部メモリへのアクセス回数が著しく減少され、その結果システム性能を向上させる。新しいインストラクションもブランチする場合、キャッシュへのアクセスとメインメモリへのバスアクセスを同時に開始する。“キャッシュ・ヒット”時はメインメモリへのアクセスを中止し、キャッシュからインストラクションを取り出す。ただし“キャッシュ・ヒット”しなかつた場合、メインメモリに対するバスリイクルを宛先させて、ブランチを宛了させる。このようにして、システム性能は最大に維持される。また、512バイトのインストラクション・キャッシュはチップコストと“キャッシュ・ヒット”のバランズによって決定した。Servis 32000のインストラクションは平均3.2バイト長で、512バイトが“キャッシュ・ヒット”も効率的かつる為に必要なサイズである。一方、データ・キャッシュはより重要である。すなわち、タイムシェアリングを必要とするシステムではスタック・オペレーションが頻繁であり、データ・キャッシュのサイズはより大きい必要がある。そのため、1024バイトを準備し、しかもユエイ・セプト・アソシエイティブに設計してあるため、性能的には2048バイトのデータキャッシュに相当する。

キャッシュ・コーヒレンシーはキャッシュの有効性を維持するため、必ず確立されなければならない。DMAの機能を保有しないうシステムではキャッシュ・コーヒレンシーの維持は容易である。すなわち、NS32532はライト・スルーの機構を採用しており、書き込み時には必ず、メインメモリとデータ・キャッシュを同時に更新している。従って、キャッシュ・コーヒレンシーは維持される。しかしながら、DMAや他のCPUが存在するマルチ・プロセッシング・システムではキャッシュ・コーヒレンシーの維持は容易でなくなる。この問題を解決するため、NS32532はソフトウェアとハードウェアによるキャッシュ・インバリデーション機構を保有する。また、リアルタイムのアプリケーション時に有効な、インストラクション・キャッシュとデータ・キャッシュの“ロック”も可能である。この機能により、高速なコンテキスト・スイッチが割り込み処理が可能になってくる。

NS32532は4段のパイプラインでインストラクションの実行を処理するアーキテクチャを採用している。

- ・ローダーはインストラクション・プリフェッチとデコードを行う。
- ・アドレス・ユニットはアドレス計算とソース・オペランドのプリフェッチを行う。
- ・エグジキューション・ユニットはインストラクションの実行と例外処理を行う。
- ・レジスタ・ファイルはプログラム・カウンタ、他の重要なレジスタを保有し、アドレス・ユニットとエグジキューション・ユニットに用いられる。

これらの4機能ブロックは同時に実行を継続しており、CPUの処理能力向上に貢献している。各機能ブロックはバッファを保有しており、最大クイック

シヨンを同時に実行させることが可能である。

パイプライン・アーキテクチャを採用した場合、I/O処理のサポートが重要になってくる。同時に数インストラクションを処理している場合、メモリ・マップドI/Oの処理は特別に注意を払う必要がある。

- (a) ある周辺回路からデータを読み込む場合、リードを行なったためにその内容を変化させてしまう。例として、FIFOのバッファやUARTのデバイス、NS16450からライン・ステータス・レジスタがあげられる。一つ以前のインストラクションが実行完了していない時、この様なデイストラクティブ・リードは必ず避けなければならない。
- (b) ある周辺回路にデータを書き込む場合、その回路中の他のレジスタの内容を変化させてしまうことがある。例として、インターラクト・コントロール・ユニット、NS32202のカウンターの値をリードする場合は、必ずコントロール・レジスタにライトしてカウンターをストップさせてからリードする手順になっている。この場合、NS32202と云う周辺回路にライト、リードする順番が著しく重要になってくるわけである。従ってNS32202にライトするインストラクションが一つ前にあって、しかも実行完了していない場合、ソース・オペランドのリードは禁止される。

NS32532は次の二種類の方法でこれらの問題を回避している。

- (i) NS32532はI/OEMH(出カピン)とIOPEC(入カピン)を保有する。リードバスサイクル中に、実行されないインストラクションがフェッチされたり、以前のインストラクションのライト・オペレーションが完了していない状態でのリード・オペレーションが実行されると、I/OEMH信号を出力する。また、デイストラクティブ・リードが可能な周辺回路を採用した場合、図2に示されたようにIOPEC入力によって、NS32532はデイストラクティブ・リードを実行したことをセンスできる。この時、そのリード・データを無視し、インストラクションの実行を以前のインストラクションからシリアルに実行する。従ってデイストラクティブ・リードが避けられる。
- (ii) NS32532は図3に示された仮想アドレス空間を設定している。周辺回路をメモリ・マップド・I/Oエリアに設定した場合、CPUはライトオペレーションが待期状態の場合には、メモリ・マップドI/Oエリアに対するリードを実行しない。但し、通常のアドレス・エリアに対するリードやライトの実行は他のインストラクションの状態に関係なくリードやライトのオペレーションを同時に実行する。

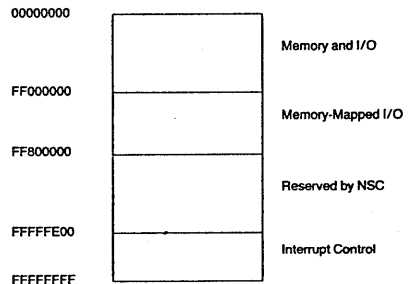
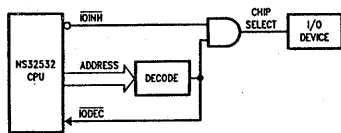


図2、メモリ・マップドI/Oの処理 図3、仮想アドレス空間の設定

(2) 高性能

NS32532は平均8~10Mエプスの高性能を保有し、最高処理速度として15Mエプスが可能である。

- 高速リアルタイム処理：インターラプト遅延時間は1.3μsでコンテキストスイッチ時間は3.6μsである。
- 高速バスサイクル：リード/ライトサイクルはクロックサイクルでバースト、モード、サイクルは1クロックである。
- 外部メモリに対するアクセスの容易性：メモリ、アクセス時に、NS32532は最大30MHzのクロック使用時に、アドレス、バリッドから外部レディまでの期間が49ns可能である。
- パラレルに数イストラクション実行時のエ/ロオハレシヨシモオシチンフ、ハードウェアエラーが起るないように実行順序を自動的に処理する。
- 外部バスバンド幅：96MB。
- イストラクション・キャッシュバンド幅：120MB。
- データ・キャッシュバンド幅：120MB。
- MMUに内蔵された64エントリーのトランスレーション・バッファは99%以上のヒット・レートを保証している。
- 512バイトのイストラクション・キャッシュと1024バイトのデータ・キャッシュは高率なキャッシュ・ヒットを保証する。

(3) 柔軟性

NS32532は従来のマイクロプロセッサでは不可能であった応用分野にもその高性能及び柔軟性ゆえに利用可能である。NS32532のアーキテクチャはシングル・プロセッサ・モードにもマルチプロセッサ・モードにも適している。

シングル・プロセッサ・モードでは、NS32532は高速クロック、高速メモリ・バンド幅、イストラクション・キャッシュ、データ・キャッシュ、MMU、パイプライン・アーキテクチャ、対称性の良いアーキテクチャ等を利用することにより広範囲の応用分野で高性能を發揮する。

マルチプロセッサ・モードでは、バス・ウツチヤ回路を用いて高速にキャッシュコーヒレシエーを維持する。バスマスターがいくつが同一システムにある場合キャッシュ・インバリテート・アドレス・ラインを用いて、直ちにキャッシュの一部をインバリッドにできる。今後システムの高性能かは不可避であり、マルチプロセッサ・モードに対する適応性はCPUにとって重要なファクターである。

(4) 互換性

Series 32000は"トップ・ダウン"のアプローチによってアーキテクチャを決定したため、各CPU間のソフトウェア互換性が維持されている。そのため、NS32008、NS32016、NS32032、NS32332のCPUに開発されたソフトウェアをNS32532にも容易に利用できる。また従来からあるスレーブ・プロセッサ、NS32081、NS32012、NS32382ともプロトコールの互換性をNS32532は維持している。

II. インストラクション・キャッシュ・アーキテクチャ

図4に示されているように、インストラクション・キャッシュはダイレクト、マップ可能な32ラインで構成されている。そして、各ラインは23ビットのタグ、4個のダブル・ワード及び4ビットのバリディティビットで構成される。タグはインストラクション・アドレスの上位23ビットを保有する。また、各バリディティビットは各ダブル・ワードに対応している。バリディティ・ビットがハイでタグ比較が一致すると、キャッシュ・ヒットで、そのダブル・ワードが選択され、ローダーに内蔵されているPバイト・キューにそのダブル・ワードが転送される。インストラクション・キャッシュはCFGレジスタの中にECビットによって禁止したり、有効にすることができ、ECビットが"0"であると、ECは無視され、直接外部メモリからダブル・ワードをフェッチする。ECの内容は全く変化なしになる。

インストラクション・キャッシュが利用可能時、存在あり、ECビットが"1"の時、アドレスビット4, 5, 6, 7, 8が32セットの4個のダブル・ワードから1セットを**選択**出し、アドレスビット2, 3を用いて1個のダブル・ワードを**選択**する。

インストラクション・キャッシュが存在してはいるが、しかもキャッシュがロックされている場合、次の動作を実行する。

- ・最初にタグの内蔵を新しい23ビットのアドレスに更新する。
- ・4ビットのバリディティビットをクリアする。
- ・外部メモリから4個のダブル・ワードをインストラクション・バッファにロードする。
- ・この間に、キャッシュが無効に存在する限り、インストラクション・キャッシュをアップデートする。
- ・最後にバリディティビットをセットする。

インストラクション・キャッシュはソフトウェアとハードウェアによって無効にすることができる。ソフトウェアの方法は：

- ・全体のインストラクション・キャッシュを無効にする為には、CFGレジスタのECビットを"0"にする。
- ・CENVインストラクションを用いて、全体のインストラクション・キャッシュを無効にする。
- ・CENVインストラクションを用いて、1セットだけ無効にする。

ハードウェアによる方法：

- ・ENVIC入力信号により、全体のインストラクション・キャッシュを無効にする。但し、ENVSET入力信号は"1"にしておく。
- ・CEAバスで1セット**選択**して、INVICとENVSETで、そのセットだけ無効にする。

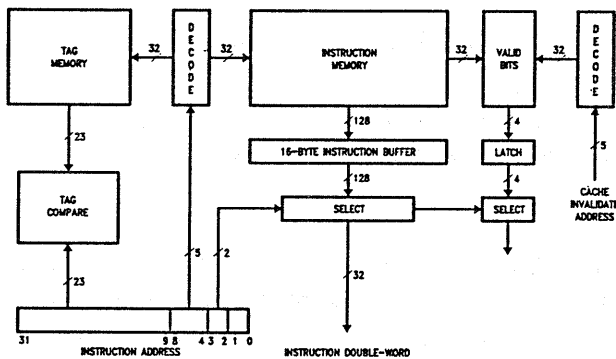


図4、イレストラクシヨン・キャッシュのアーキテクチャ

Ⅲ、データ・キャッシュ・アーキテクチャ

図5に示されているように、データ・キャッシュは2ウェイ・セット・アソシエイティブ構成で、64ラインを保有する。1セットが2ラインで構成される。1ラインは23ビット・タグ、4個のダブル・ワード、4ビットのバリディティビットで構成される。

データ・リード

データ・キャッシュが無効の場合、データ・キャッシュを無視して、外部メモリから直接リードして、データラインを行なっているイレストラクシヨンの実行に利用する。

データ・キャッシュが有効な場合、2ラインがリードされ、2個のタグが23ビットの物理アドレスと比較される。どしどしのタグが一致して、バリディティビットが有効の場合、アドレス・ビット2、3で1個のダブル・ワードを選択する。ビット0、1でこのダブル・ワードをアライメントして、イレストラクシヨンの実行に用いる。

キャッシュ・ヒットしなかった場合で、キャッシュがロックされている時は、次の動作を実行する。

- ・タグがマッチした場合、そのラインをアップデートする。
- ・タグがマッチしない場合、最も古く使われたラインを選択し、そのタグを23ビットのアドレスで更新する。
- ・そのラインのバリディティビットをクリアする。
- ・外部メモリから4個のダブル・ワードをリードする。
- ・キャッシュが無効に在る条件が在る限りにおいて、4個のダブル・ワードでそのラインを更新する。
- ・バリディティビットをセットする。

データ・キャッシュはソフトウェアアヒールドウェアによって無効にできる。ソフトウェアの方法は:

- ・ C F G L エキスタの D C ビットを " 0 " にする。
- ・ C E N V インストラクシヨんでデータ・キャッシュ全体を無効にする。
- ・ C E N V インストラクシヨんで、16 バイトブロックを無効にする。

ハードウェアによる方法:

- ・ ENVSET 入力が " 1 " で、INDDC が " 0 " のとき、データ・キャッシュ全体を無効にする。
- ・ CIA バスによって選択されたセットを INDDC 入力を " 0 " にして無効にする。但し、ENVSET 入力は " 0 " にしておく。

データ・ライト

データ・キャッシュが使用可能状態の場合、アドレスビット 4, 5, 6, 7, 8 で、1 セット選択し、2 個のタグが 23 ビットの物理アドレスと比較される。アドレスビット 2, 3 で 1 個のダブル・ワードとバリディティビットを選択する。もし、タグが一致して、バリディティビットが有効の時、そのダブル・ワードを更新する。但し、そのデータがデータ・キャッシュに存在しない時、データ・キャッシュは一切変更なしになる。そのデータは外部メモリに、ライト・スルーされるだけである。

データ・キャッシュの内容は CPU 外部に常にコピーしておくことが可能である。すなわち、CPU のシステム・インテーフエースを観察することにより、バス・ウオッチ回路がキャッシュ・コーヒレンシーを維持することになる。アドレスビット 4, 5, 6, 7, 8 でセットを検出し、CAS E C でラインを検出できる。タグ値はアドレスビット 9, 10, ---, 31 で更新する。

IV. まとめ

NS32532 は Series 32000 とソフトウェアの互換性を維持しながら、最高時で 15 M E P S という高性能を發揮した。シングル・プロセッサ・モードでの高性能だけでなく、マルチプロセッサ・モードにおいても高速処理を可能にする、ハードウェアによるキャッシュ・コーヒレンシーの維持が可能になっている。

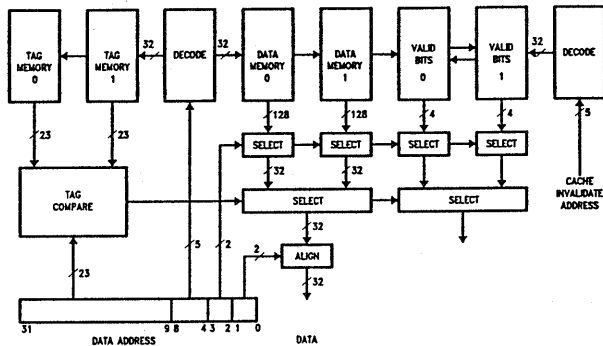


図 5. データ・キャッシュのアーキテクチャ