

## キャッシュについて

セガ・ソフトウェア技術開発部

1998 年 9 月 18 日

この文書では、キャッシュの仕組みについて簡単に説明し、SEGA ライブラリに提供しているキャッシュ関数の使用方法を理解して頂くものです。

### キャッシュとは

通常 CPU は非常に高速な動作をしています(SH7091 の場合 200MHz)。しかしながら、それと比べメモリへのアクセスは、非常に低速な為、折角 CPU 本体が高速で動作していても、遅いメモリアクセスのため、処理の足を引っ張ってしまう形になります。キャッシュとは、遅いメインメモリと、速い CPU との橋渡しをして、CPU の性能を落とさないためのモジュールで、非常に高速にアクセスできる RAM で構成されています。

もし、キャッシュの中に、メインメモリのコピーが保存されていれば、わざわざ遅いメインメモリを介さずに、高速にメモリの情報を引き出すことが可能になるのです。

こういった高速アクセスの可能な RAM はコストがかさむため、メインメモリとしては使用できませんが、キャッシュなどのわずかな RAM であれば、CPU のパフォーマンスを向上させることが出来ます。

### キャッシュの仕組み

一般にキャッシュとして積まれているメモリは、メインメモリから比べると、非常にわずかなものです。(SH7091 の場合、命令キャッシュ 8KB、データキャッシュ 16KB)、当然、メインメモリ上のデータがすべてキャッシュの中に乗るわけではありませんので、少ないキャッシュ RAM を効率良く使用する必要があります。

キャッシュは構成上、アドレスアレイとデータアレイに別れ、アドレスアレイは、メインメモリ上のアドレスを示し、データアレイはその中のデータを示します。

一つのアドレスアレイに対してデータアレイは 32 バイトあります。

この組をエントリと呼び、命令キャッシュで 256 エントリ、データキャッシュで 512 エントリあります。

CPU は、メモリアクセスする場合、まずキャッシュに該当するメモリの情報があるかを調べます。この時、キャッシュ上に求めるアドレスデータがあった場合(キャッシュヒット)、そこから、データを拾ってきます。

一方、キャッシュ内にデータが存在しない場合(キャッシュミスヒット)、メインメモリ上からデータを取得します。(キャッシュミスヒット時のペナルティ)

キャッシュミスヒット時には、一つのエントリをすべて埋めるため、目的のアドレスの前

後 32 バイト分をいっぺんに取得し、キャッシュエントリを埋めます。ミスヒット時には、ヒット時に比べて多くの時間がかかってしまいます。

これは、リード時であっても、ライト時であっても同様に、キャッシュにミスヒットがあった場合、必ずそのアドレス付近の 32 バイトを取得して、キャッシュエントリを埋めます。

キャッシュの資源に限りがありますので、いつでも利用可能なキャッシュエントリが見つかるというわけではありません。

その場合、キャッシュコントローラは、一番あとに使われたキャッシュエントリを選択し、そのエントリの不整合を修正した後、別のエントリとしてメモリ上から 32 バイトのデータを取り込みます。この、キャッシュラインを選択するアルゴリズムを LRU といいます。詳細については、一般的な CPU の解説書をご覧ください。(Pentium、PowerPC など)

## ライトバック

CPU がメモリに対してアクセスを行う場合、キャッシュにエントリがあれば、そこから読み込むだけでなく、書き込むことも行うことができます。書き込んだ場合、そのデータアレイにはダーティビットというビットが立てられ、メインメモリとキャッシュの間では不整合が発生していることが示されます。

当然キャッシュに書き込むだけでは、メインメモリへのデータ更新は行われませんが、メインメモリに対して読み書きするのに比べて、はるかに高速にアクセスすることが出来ます。

極力、メインメモリへのアクセスを減らしキャッシュの中だけで完結させることで処理の高速化が図れます。

しかし、この状態ではキャッシュの内容と、メインメモリの実体の間ではデータに不整合が起こります。

この状態を回避するために、あるタイミングでキャッシュの内容をメインメモリに書き戻し、キャッシュとメインメモリの間の不整合を解消させる必要があります。

これをライトバックといいます。

## キャッシュのコヒーレンシ

CPU がメモリにアクセスする場合には、必ずキャッシュを通して行われます。

ところが、CPU モジュールの中には、DMA などのキャッシュを介さないで直接メモリにアクセスできるような場合も存在します。

DMA がキャッシュのエントリに含まれるアドレスに対して、データライト等の処理を行った場合には、キャッシュで管理しているデータと、実際のメモリの間には不整合が発生してしまいます。

このような、キャッシュとメインメモリの関係をコヒーレンシといい、上記のような場合

には、コヒーレンシがあっていない状態となります。

### キャッシュのインバリデート

DMA などのアクセスによって、メインメモリとキャッシュの間に不整合が生じ、それがライトバックなどの手段でも、整合性を保つことが出来なくなった場合、そのエントリのキャッシュを無効化して、再度キャッシュラインにデータを取り込む必要が出てきます。

この操作をキャッシュのインバリデート(無効化)といい、エントリのインバリデートと前キャッシュのインバリデートに別れます。

この操作を行った場合、キャッシュのデータが破棄されインバリデートされたキャッシュエントリはアドレスアレイの有効ビットがおろされ、次のキャッシュ登録の候補となります。

SEGA ライブラリにおいて DMA が発生する典型的なものは GD へのファイルアクセスですが、GDFS ライブラリにおいては、データ転送後データキャッシュのインバリデートを行います。そのため通常、アプリケーションにおいてはインバリデートを意識する必要はありませんが、オーバレイなどの目的でプログラムを読み込んだ場合には、意識する必要があります。GDFS ライブラリは命令キャッシュをインバリデートすることはなく、そのままオーバレイされたプログラムを実行した場合、プログラムエリアの中にキャッシュにエントリがあるアドレスが含まれた場合、誤動作を起こします。

GD よりプログラムモジュールを読み込んだ場合は、読み込んだエリアに対して必ず、命令キャッシュのインバリデートを行う必要があります。

### キャッシュのパージ

キャッシュをライトバックした場合、そのキャッシュエントリは有効で、またそのエントリで示すアドレスに対して CPU がアクセスしようとした場合は、そのキャッシュエントリがそのまま利用されます。

しかしながら、もしライトバックしたエントリのアドレスにはアクセスする必要が無い場合、いつまでも必要の無いキャッシュエントリ持っていることは、キャッシュの効率上、あまり好ましいものではありません。

この場合、そのキャッシュエントリを書き戻した上、無効化してしまうほうが都合がよくなります。そういった操作はパージと呼ばれ、パージしたキャッシュエントリは有効ビットがおろされ、次のキャッシュエントリの登録候補となります。

### キャッシュのプリフェッチ

これまでに説明してきましたように、キャッシュに対してアクセスする場合、キャッシュヒットするか、キャッシュミスヒットするのかわでパフォーマンスは大きく変わってきます。キャッシュミスヒット時のペナルティは出来るだけ避けるためには、出来る限り効率のよ

いキャッシュの使い方をすると、前もって、キャッシュミスヒットすることが分かっている場合、キャッシュエントリを予め埋めておくという方法があります。

この後者の方法を実現するのがキャッシュのプリフェッチとなります。

プリフェッチは、データキャッシュに対してのみ行うことが出来ます。

プリフェッチを有効に使用するためには、出来上がったアセンブラコードを追いかけて、何処でキャッシュミスヒットが発生するのかを調べるとか、パイプラインシミュレータでデータのロード・ストアのパイプライン遅延が発生しているところを調べて挿入するなどの高度なテクニックを必要とします。

### キャッシュエントリの確保

メモリアクセス時に、そのアドレスの中身に何が書かれていても、内容は更新される場合、例えば、バッファとして確保している領域で、その領域には一方的に計算結果が格納されるような場合、そこに何が書かれていようが、新しいデータで更新されることが分かっている場合に、普通にそのアドレスをアクセスし、そのアドレスがキャッシュエントリに含まれていない場合、通常のキャッシュ動作では、まず、そのアドレス付近 32 バイトでキャッシュエントリを埋め、その後該当するアドレスを示すキャッシュのデータアレイに対して書き込みます。

しかし、この操作は、一旦 32 バイトのキャッシュエントリを埋めるためにメインメモリから 32 バイトのデータを取得するという作業が発生するため、効率がよくありません。

このような、特別な場合にメモリからデータを取得することなくキャッシュエントリを確保するアセンブラ命令が用意されています。

```
MOVCA.L R0 @Rn
```

SEGA ライブラリには、関数としてこの命令を実行するものが備わっています。

### キャッシュの RAM モード

SH7091 には、データキャッシュの半分(8KB)を RAM として利用するモードが備わっています。

キャッシュの仕組み上、キャッシュ RAM を普通の RAM として用いた場合、非常に高速なアクセスとなる事が分かっています。

しかしながら、SEGA ライブラリでは、アプリケーションがキャッシュを RAM モードにすることは禁止します。アプリケーションが RAM モードで正常動作する事を保証できないからです。