組み込みマイコンの仕組みを理解しよう

第1回

FRマイコンのアーキテクチャ

平石 郁雄

組み込みマイコンは，組み込みシステムの中核をなす部品と言える．本連載では，富士通が出荷している32ビット・マイコン「FRファミリ」を例に，マイコンの仕組みや使い方について解説していく．第1回は，マイコン・ハードウェアのアーキテクチャを中心に説明する．（編集部）

本誌の読者の中には，週に1度は電気店へと足を運ぶ方も多いのではないでしょうか．筆者も同じで，ぶらぶらと電気店へ行き，足の赴くままに店内をうろついています．行くたびに機能の異なる新製品がどんどん入荷されており，好奇心が刺激されます．これら多くの新製品を陰で支えているのが，マイクロコントローラ（マイコン）です．

マイコンは，プログラムによって任意の機能動作に変更できることを大きな特徴としています．少々の機能の違いであれば，簡単にプログラムの変更で対処できます．極端な話，回路図を変更することなく，プログラムの変更のみで新製品を開発することも可能なのです．また，開発途中の製品企画の変更にも柔軟に対応できます．「融通がきく」ということを大きな特徴としているので，新製品開発には必要不可欠の部品といえます．

このように便利であるが故に，マイコンが組み込まれている電子機器は多岐にわたります（**図1**）．例えば，常に進化を求められるテレビやオーディオ機器といった据え置き型の家電製品にも，携帯電話やデジタル・カメラ（デジカメ）といった携帯型機器にも，自動車や飛行機といった運輸機器にも組み込まれています．機器の筐体（きょうたい）に囲まれているので，通常は目に触れることはありませんが，私たちの生活になくてはならない部品といえます．あなたの目の前の電化製品を分解すると，きっと，たくさんのマイコンを発見することでしょう．

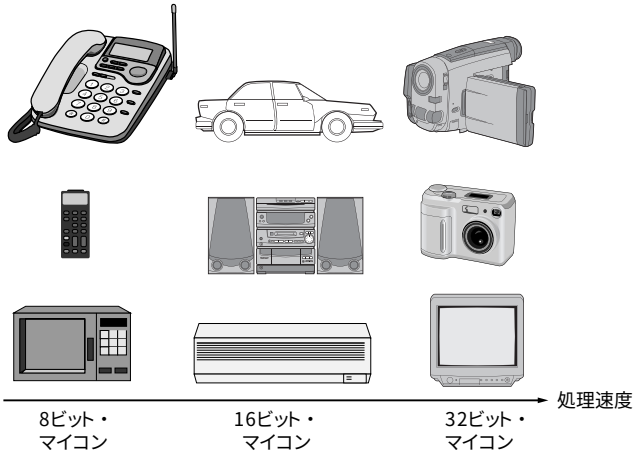


図1　マイコンが使用される電子機器

ビット数が大きくなるほど，処理速度は上がる．これに伴って搭載される機器が変わる．

本連載では，組み込みシステムの開発に利用されているマイコンの仕組みや使い方について解説していきます．実際のマイコンの例として，筆者ら（富士通）が開発・販売している32ビット・マイコン「FRファミリ」の内部構造や開発環境などを示しながら説明していきます．第1回は，マイコンのアーキテクチャと基本レジスタ・セット，パイプライン動作について述べます．

● 低消費電力と高速処理の微妙な関係

上述の内容と矛盾すると思われるかもしれませんが，異なる機器の中で同じプロダクト・ナンバ（型名）が付いたマイコンを探し出すことは困難だと思います．それは，それぞれの機器が，プログラムでは吸収できない仕様を求めているためです．例えば，デジカメのような携帯型機器では低消費電力が重要になりますが，据え置き型のオーディオ機器ではデジカメほど低消費電力にこだわる必要がないことは，直感的に分かると思います．

処理速度を重視する据え置き型機器には高速動作のマイコンが使用されますが，消費電力を重視する携帯型機器には低消費電力のマイコンが使用されます．例えば筆者らが提供しているマイコンの例では，高速動作が必要な機器にはFR60が使用され，低消費電力が求められる機器にはFR60Liteが使用される傾向にあります（右掲の**コラム1**を参照）．FR60は133MHzの高速動作に耐えうる回路構成になっていますが，消費電流はどうしても大きくなってしま

KEYWORD——マイコン，FRファミリ，CISC，RISC，アーキテクチャ，レジスタ・ファイル，プログラム・カウンタ，パイプライン，レジスタ・ハザード，レジスタ・バイパス，プリフェッチ・キャンセル

コラム1　FRマイコンへの道のり

筆者らは，1985年に8ビット・マイコン「F²MC-8」を世に送り出しました．これは，F²MC-8L，F²MC-8FXと世代を経て，20年以上にもわたって使用され続けています(**図A**)．

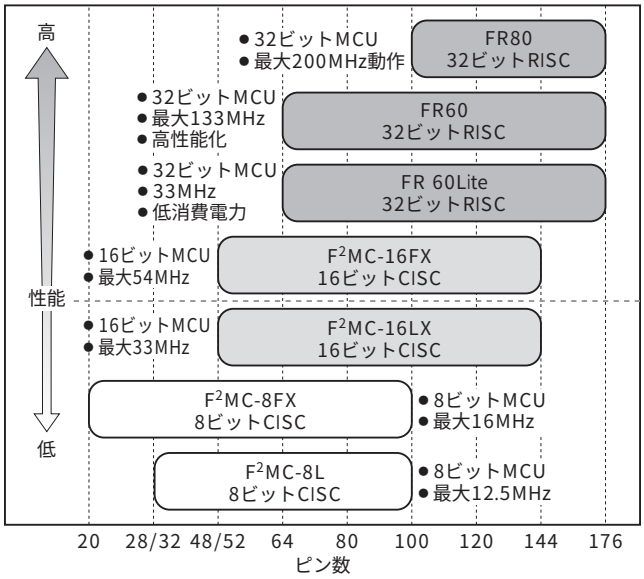
1985年当時は，16ビット・マイコンがエンジニアリング・ワークステーションに使用されていた時代で，8ビットのF²MC-8は高性能マイコンと位置付けられていました．ちょうど多くの周辺機能がマイコンに内蔵され始めたころです．4ビット・マイコンでは処理できないアプリケーションが増え，8ビット・マイコンの需要が高まっていました．国内の半導体メーカの多くは海外製マイコンのセカンド・ソース品を開発していたのですが，F²MC-8は国産の独自アーキテクチャを持つ珍しいマイコンでした．

F²MC-8の発売から遅れること4年，筆者らは1989年に16ビット・マイコンのF²MC-16Hを開発しました．そしてさらに5年後の1994年には，ロング・セラーとなるF²MC-16LXシリーズを開発しました．F²MC-16LXはその完成度の高さから，今日でも安全性が求められる車載システムなどに使用され続けています．

1991年には32ビット・マイコンFRファミリの第1号であるFR20を開発しました．当時，ほかの多くの半導体メーカは既に抱えている顧客の移行のしやすさを重視してマイコンの新製品を開発していたのですが，筆者らはまったく新しいアーキテクチャを持ったRISCベースの32ビット・マイコンを開発し，組み込み制御向けの高性能マイコンとして発売したのです．ちなみに，FR20のFとRは，それぞれFujitsuとRISCの頭文字に由来しています．また，FRファミリの初代がFR20なのは，社内の開発コード・ネームとして「20H」と呼んでいたことに起因します(32ビット・マイコンなので，ヘキサで“20”という意味)．

FR20，FR30からFR60への進化は，マイコン開発用のCAD(Computer Aided Design)ツールの進歩に大きく依存しています．信号線のクリティカル・パス(信号の伝播遅延時間が大きい配線経路．集積回路の動作速度低下の原因になる)を最適化することで，133MHzの高速動作が可能となりました．

FR80ではアーキテクチャの大幅な変更を行っており，パイプライン動作と命令セットを変えました(上位互換は維持)．例えば，パイプラインでは，コード・サイズが48ビットの「32ビットの即値付き命令」が取り扱えるようになりました．それまでのFR60では，16ビットごとにデコードし，3サイクルの実行時間が必要でしたが，FR80では，一度に48ビットをデコードできます．



図A　富士通製マイコンのラインナップ

います．一方のFR60Liteは動作周波数を33MHzに制限し，回路や信号線を最適化して消費電流を抑えています．

マイコンというと“機能”にのみ注目が集まりがちなので，ここではあえて処理速度と消費電力について述べました．このほかにも，動作温度やノイズ(EMI，EMS)，静電耐圧，信頼性，量産性，価格といったさまざまな要素を把握する必要があります．

● マイコンには頭脳と手足がある

マイコンの基本構造は，人間の頭脳に相当する「CPU(演算処理機能)」と「メモリ(記憶機能)」，手足となる「I/O(入

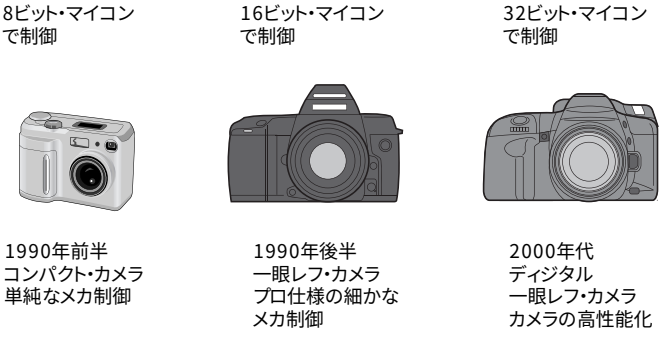
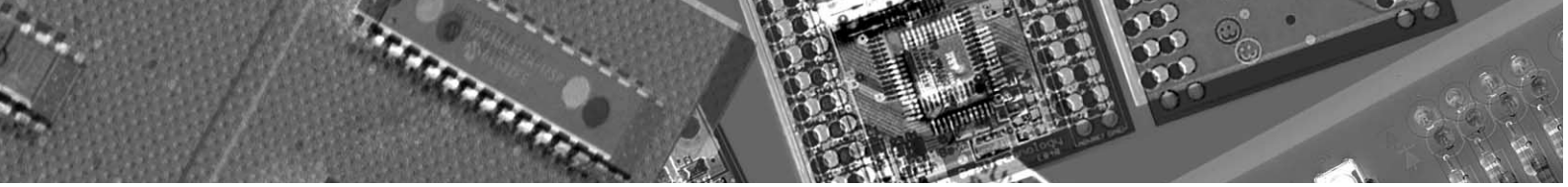


図2　カメラの進化

電子回路で実現する機能が増えるにつれて，マイコンに求められる処理速度も増えている．



出力機能)」の三つから構成されています.

CPUはプログラムを処理する部分で一度に処理できるデータのサイズにより,8ビット・マイコン,16ビット・マイコン,32ビット・マイコンに分類されます.一般にビット数が大きくなれば処理能力が上がりますが,処理回路やデータ線の増加により,チップ面積(価格に影響)やリーク電力(消費電力に影響)も大きくなる傾向があります.そのため,価格や消費電力を重視するシステムの電源制御などには8ビット・マイコンが使用され,処理速度が重視されるシステムには32ビット・マイコンが使用される傾向にあります(**図2**).

メモリはプログラム(コード)とデータを記憶する部分で,マスクROM,フラッシュROM,SRAMなどがあります.マスクROMやフラッシュROMは電源供給を遮断してもデータが保持されるので,プログラム格納用によく使用されます.またフラッシュROMについては,データの書き換えに消去コマンドや書き込みコマンドといった手続きが必要となるため,アクセスが遅くても問題のないデータの格納に使用されます.SRAMは,電源供給を遮断するとデータが消えるので,一時的なデータの保存に使用されます.

最近では,頻繁にプログラムを書き換える用途ではフラッシュROMとSRAMを内蔵したマイコンが利用されます.一方,成熟してプログラムを変更する必要がなく,1円でもコストを下げたい用途ではマスクROMとSRAMを内蔵したマイコンが使用される傾向にあります(下掲の**コラム2**を参照).

I/Oには汎用ポートと周辺機能があり,その入出力にはアナログまたはディジタルの電気信号を使用します.プログラムでCPUから直接ポートを制御する場合には,汎用ポートを使用します.間接的に制御する場合は,周辺機能を使用します.周辺機能には,タイマ系,カウンタ系,通信系,アナログ系などがあります.ほとんどの周辺機能はCPUとは独立して並行動作し,マイコンの処理能力向上や低消費電力,コスト低減などに一役買っています.

ただし,周辺機能としてハードウェアを搭載してしまうと,その機能を必要としないシステムにとっては,ただのお荷物になってしまいます.そのため,より多く使用される周辺機能に限定して,内蔵されることになります.また,仕様が頻繁に変わるような機能についても,ハードウェアとして内蔵するのではなく,CPUとプログラムで処理する傾向にあります.例えばUARTですが,プログラムを用いて汎用ポートを制御することで同じような動作を実現できますが,一般にはハードウェアの周辺機能として組み込まれることが多いようです(p.121の**コラム3**を参照).

コラム2　次世代のマイコン内蔵メモリはFRAM

マイコンというとCPUの種類に注目が集まりがちですが,マイコンを構成する重要な要素として,プログラムやデータを保存するためのメモリがあります.あまり知られていませんが,筆者らは,1996年の段階で,現在では主流となっているフラッシュROM内蔵マイコンを出荷しています.

当時は,マスクROM品やOTP(One Time Programmable)ROM品が主流でした.プログラムの変更は開発時にしか行わないという考えが定着しており,フラッシュROMの「書き換え可能」という特徴は,一部のユーザにしか受け入れられませんでした.

時代が変わり,プログラム容量が肥大化したことにより,今では開発時だけでなくフィールドでもプログラムの変更が行われるようになっています.2001年には,1Mバイトのフラッシュ ROM内蔵品や10万回の書き換えに対応したフラッシュROM内蔵品を開発しました.

そして,再び筆者らは新しい技術への挑戦を始めています.2007年春に,FRAM(Ferroelectric Random Access Memory ;強誘電体メモリ.FeRAMとも呼ぶ)を内蔵したマイコンを展示会で披露したのです.FRAMは,CPUの処理速度と同じ速度でアクセスできるため,CPU処理に使用するテンポラリ・データ(スタック・データなど)を保存する領域として使用できます.

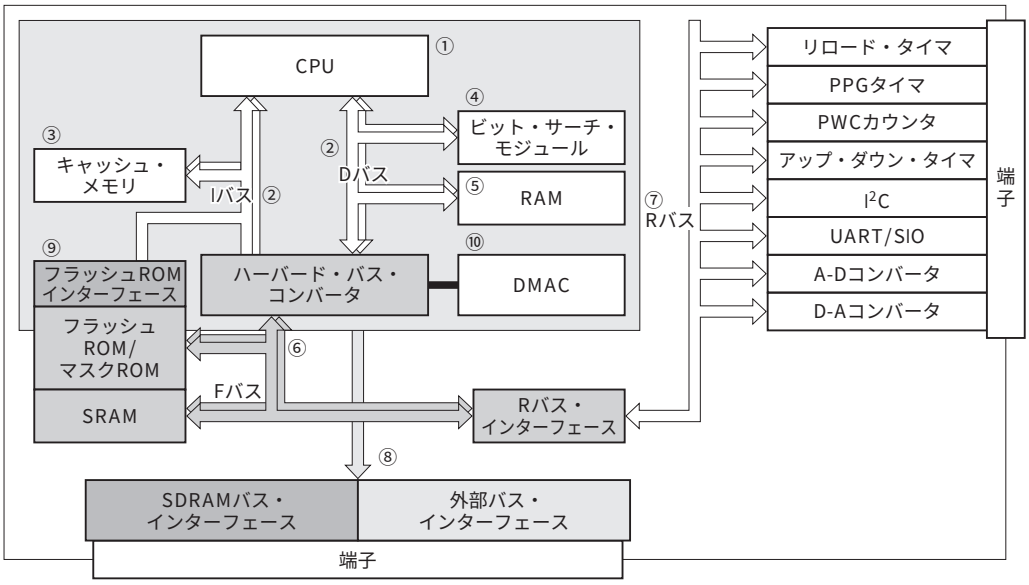
マイコンに内蔵されたすべてのメモリをFRAMにすれば,プログラム用にもデータ用にも使用できて良いのではないかと思われるかもしれません.しかし,FRAMはまだ新しいメモリ技術なので,マスクROMやSRAMほどの読み出し速度には到達していません.将来的には,ROMやSRAMの代わりとなって活躍すると筆者らは考えています.

FRAMの特徴や構造については,各種の文献が出ていますので,そちらを参照してください.

● メモリ重視なら CISC，速度重視なら RISC

実際のマイコン（FR）の内部構成は**図3**のように複雑です．しかし，マイコンの基本機能である CPU，メモリ，I/O の三つに焦点を絞って整理すれば，構成要素を単純化することができます．

CPU はプログラムに従って処理を行う機能で，大きく CISC（Complex Instruction Set Computer ；複合命令セット・コンピュータ）系と RISC（Reduced Instruction Set Computer ；縮小命令セット・コンピュータ）系に分けることができます（「CISC，RISC を議論するのは古い！」という声が聞こえてきそうだが，CPU の基本なのでしばしお付き合いを…）．

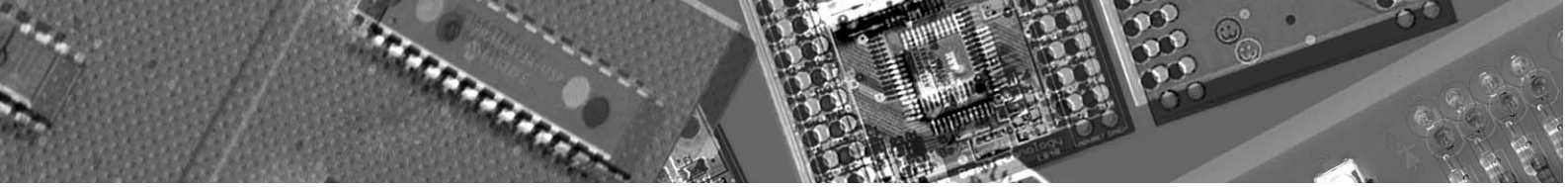


（**a**）構成例

構成要素	機　能
① CPU	マイコンにとって重要な命令処理（演算）用の CPU が搭載されている
②Ｉバス，Ｄバス	CPU バスは，命令フェッチ専用バス（Ｉバス）とデータ専用バス（Ｄバス）が接続されたハーバード・アーキテクチャのバス構成になっている．命令フェッチとデータ・アクセス用のバスが一つになったプリンストン・アーキテクチャのバス構成の場合と異なり，命令フェッチ・アクセスとデータ・アクセスのタイミングが重なることはなく，並列処理が行える．命令とデータ領域（特にスタック）をこの領域のメモリに配置することで，最高性能を引き出せる
③キャッシュ・メモリ	命令コードやデータの一時記憶用にキャッシュ・メモリを内蔵している．この例では，高速にアクセスできる命令バスに命令キャッシュを接続している．外部バスなどを使って低速なメモリから命令コードを読み出す場合は，1回目のアクセスで命令コードとアドレス情報を命令キャッシュへ格納する．2回目以降のアクセスでは命令キャッシュから命令コードを読み出すため，実行速度が向上する
④ビット・サーチ・モジュール	リアルタイム OS のタスク管理に使用することを想定したモジュール．1ワード中の MSB（Most Significant Bit）から最初の‘1’，‘0’の変化ビット位置をサーチする
⑤（データ）RAM	データやスタックなどを格納するメモリ
⑥ F バス	バス幅は 32 ビット．内部のフラッシュ ROM やマスク ROM などのメモリ接続用バスで，命令/データ・アクセス共用
⑦ R バス	バス幅は 16 ビット．アドレスとデータのマルチプレクス（時分割多重）が可能なバスとなっている．内蔵周辺機能が接続される
⑧外部バス・インターフェース	バス幅は 32 ビット．外部バスへのインターフェース
⑨フラッシュ ROM	マイコン内蔵のフラッシュ ROM．命令コードのアクセスでは，命令バス経由で直接アクセスする．データ・アクセスは F バス経由でアクセスする
⑩ DMAC	DMA（Direct Memory Access）転送を行うモジュール．DMA 転送を使用することで，CPU を経由することなく，各種のデータ転送を高速に行うことができる

（**b**）構成要素の説明

図3
FR マイコンの構成例
各モジュールとバスを記載した．Ｉバス，Ｄバス，Ｆバスには，アドレス・バスとデータ・バスがある．



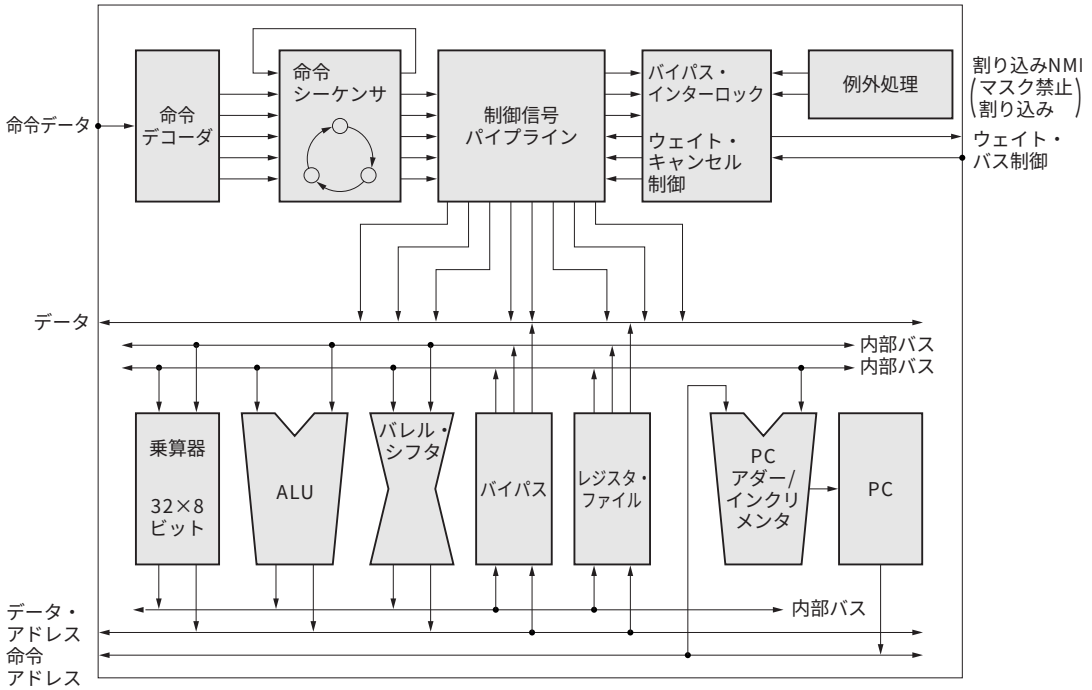


図4
CPU の構成例

矢印は，命令，アドレス，データの流れを示している．外側の四角い枠は CPU を示しており，四角い枠の外に記述してある「命令データ」，「データ」などは，CPU の外に対するバスであることを示している．四角い枠の中の「内部バス」は，CPU 内のみで使用していることを示している．

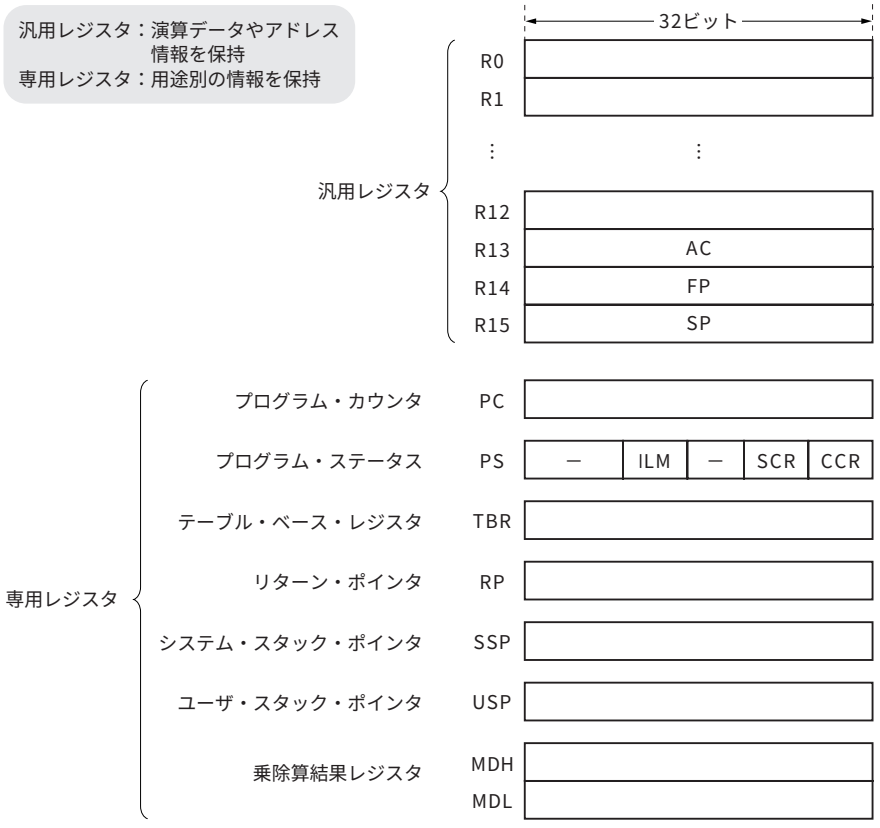


図5　基本レジスタ・セット

汎用レジスタは R0 ～ R15 である．専用レジスタには，PC，PS，TBR，RP，SSP，USP，MDH，MDL がある．すべてのレジスタは 32 ビット幅になっている．乗算結果は MDH，MDL レジスタを合わせた 64 ビット・レジスタに格納される．PS の中に記載した ILM，SCR，CCR はフィールド名である．

CISC アーキテクチャは，高級言語レベルの複雑な処理を一つの命令で実行できる優れモノですが，対応する命令数が多く，マイコン内部の回路が大規模で複雑になるというデメリットがあります．これに対して RISC アーキテクチャは，対応する命令数が少なく，回路が小規模で単純ではあるものの，高級言語レベルの処理を実行するために複数の命令が必要になります．また，この複数の命令はコンパイラによって生成されるので，コンパイラの出来不出来によって，同じ CPU を使用していても処理速度に差が出てきます．このように，RISC マイコンについては，CPU の開発者とコンパイラの開発者が協力して開発することが大変重要です[注1]．

筆者らが提供しているマイコンについて言えば，8 ビット・マイコン（F^{2}MC-8FX）と 16 ビット・マイコン

注1：このため，筆者らはサード・パーティ製のコンパイラもサポートしているが，自社でも独自のコンパイラを開発している．

(F²MC-16LX)のCPUはCISCアーキテクチャを採用しています．これは，コード・サイズの低減を重視したためです．一方，32ビットのFRにはRISCアーキテクチャを採用しています．CISCにすれば使用できる命令の種類が増え，コード・サイズが小さくなるというメリットはあるのですが，FRは処理速度と制御系命令を重視するマイコンと位置付けて，主要な命令を制御系命令に限定しています(命令セットの詳細については，本連載の第2回で解説予定)．これにより，CPU内部の回路を単純化し，実行速度を引き上げています．

● 乗算やバレル・シフトは専用回路で高速処理

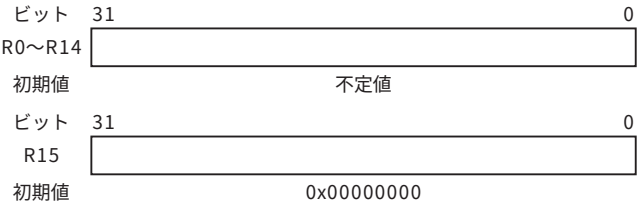
CPUの内部は，**図4**のような構成になっています．演算用としてALU(Arithmetic Logical Unit)や乗算器，バレル・シフタを搭載しています．ALUは算術演算や論理演算を行う演算回路ブロックで，CPUには不可欠の機能です．

ALUがあれば，乗算やビット・シフトも実現できるのですが，多くの計算時間(サイクル数)が必要となり，処理が圧迫されてしまいます．そこで，組み込みシステムで多用される乗算命令やビット・シフト命令を数サイクルで実行するために，乗算器やバレル・シフタの専用ハードウェア(演算回路ブロック)を搭載しています．

なお，一般に除算回路ブロックは搭載しない場合が多いようです．これは，CPUの回路規模が大きくなってしまうため，また，ALUとバレル・シフタの組み合わせで除算を計算できるためです．このように，CPU構成は，処理速度と回路規模(コスト)のバランスを考慮して決められています．

● 目的別の多彩なレジスタ群を装備

図4のレジスタ・ファイルは，データを保持するレジスタ(メモリ回路ブロック)です．レジスタには，各種演算結果やアドレス情報などを保持する汎用レジスタと，特殊な用途に使用する専用レジスタの2種類があります．RISCアーキテクチャでは，処理で使用・発生するデータを高速にアクセスできるレジスタへ格納することにより，「1サイクル1命令」の実行を実現しています．つまり，多くのレジスタを持つことが処理速度の向上につながります．しかし，多くのレジスタを内蔵すると回路規模が大きくなるというデメリットがあるため，最適なレジスタ数にすることが求められます．



(**a**) レジスタ構成

	呼び出し時	復帰時
R0	規定なし	規定なし
R1	規定なし	規定なし
R2	規定なし	規定なし
R3	規定なし	規定なし
R4	引き数/戻り値領域アドレス	戻り値
R5	引き数	戻り値
R6	引き数	規定なし
R7	引き数	規定なし
R8	規定なし	呼び出し時の値を保証
R9	規定なし	呼び出し時の値を保証
R10	規定なし	呼び出し時の値を保証
R11	規定なし	呼び出し時の値を保証
R12	規定なし	規定なし
R13	規定なし	規定なし
R14	フレーム・ポインタ	呼び出し時の値を保証
R15	スタック・ポインタ	呼び出し時の値を保証

(**b**) Cコンパイラのレジスタ規約

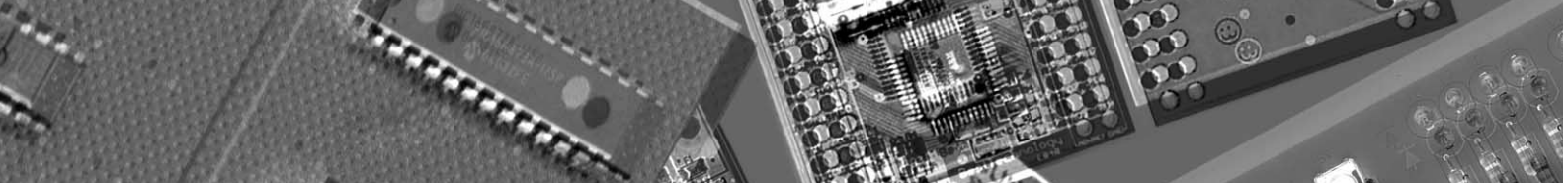
図6　汎用レジスタ

(a)のR0～R14はリセットで初期化されないので，不定値になる．R15は，リセットにより0x00000000へ初期化される．(b)では，汎用レジスタがC言語のコンパイル結果であるアセンブリ言語においてどのように使用されるかを記述している．「呼び出し時」とは，関数を呼び出すときに入っているデータがどのように使用されるかを示している．「復帰時」とは，関数からリターンするときのデータの取り扱いを記述している．

FRが備える基本レジスタ・セットを**図5**に示します．以下ではそれぞれのレジスタの役割について説明します．

1) 汎用レジスタ(R0～R15)

FRは，高級言語(C言語)のコンパイル効率を考慮して，32ビット長の汎用レジスタ〔**図6**(**a**)〕を16本内蔵しています．R0～R3，R8～R12の9本をデータの保持に使用することで，演算の高速性を確保しています．また，R4～R7を引き数や戻り値に，R14をフレーム・ポインタに，R15をスタック・ポインタに使用することで関数処理を効率化しています．さらに，R13をアキュムレータとして使用することで，メモリ・アクセスが効率的になります．どうして効率的になるのかは，命令セットを見ないと分からないと思います．これについては，本連載の第2回で説明する予定です．



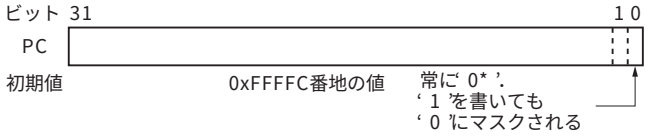


図7　プログラム・カウンタ

プログラム・カウンタは32ビットのデータ長があり，ビット0のみが‘ 0 ’に固定で，ほかのビットは任意の値を書き込める．

2）プログラム・カウンタ（PC）

専用レジスタには，各種の用途別の32ビット長専用レジスタが6本，乗除算用の64ビット長専用レジスタが1本あります．

プログラム・カウンタ（**図7**）は，メモリに格納されたプログラムを順次実行するため，実行中のプログラムのアドレスを格納する目的で使用します．分岐系以外の命令を実行した場合，この値がインクリメントされ，次の命令の実行に移ります．

命令コードは基本長の16ビットごとに処理されるので，命令の配置を偶数アドレスに限定しています．奇数アドレスのミス・アラインに対応しないことで，アーキテクチャを簡素化しています．

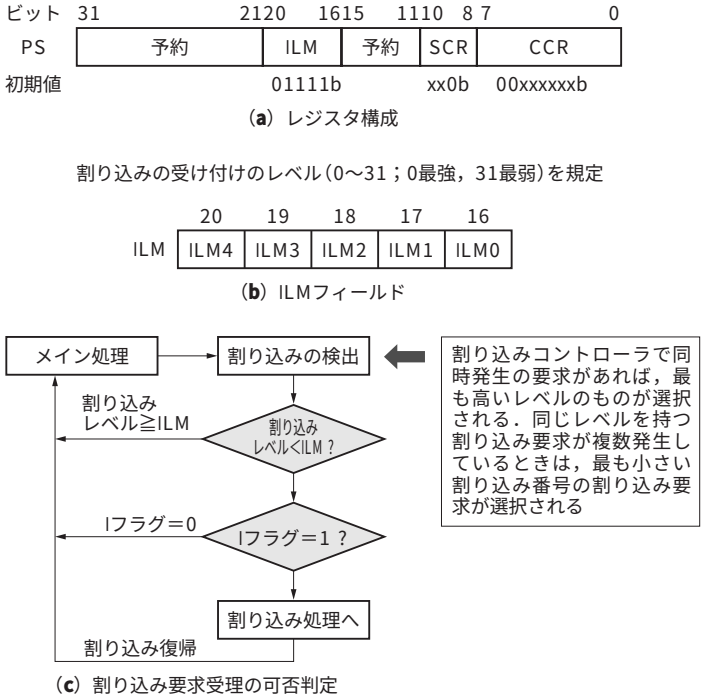
3）プログラム・ステータス（PS）

プログラム・ステータス〔**図8**（**a**）〕は，プログラム実行の状態を示すレジスタで，ILMフィールド，SCRフィールド，CCRフィールドに分かれています．

ILMフィールドは，割り込みの許可レベルを設定するフィールドで，32段階のレベルがあります〔**図8**（**b**）〕．そのうち，上位レベルの0～15はシステムが使用しており，下位レベルの16～31をユーザ・プログラムで使用できます．CCRフィールドのIフラグが‘ 1 ’のときに，ILMと割り込みレベルを比較して，ILMの値が大きいときに割り込み処理を行います〔**図8**（**c**）〕．割り込みレベルを設けることで，多重割り込みが発生したときの，処理のプライオリティ（優先度）を決定しないようにしています．

SCRフィールド〔**図8**（**d**）〕は，除算計算で符号処理に使用するフィールドです．除算は，ビット・シフトしながら演算する方式なので，符号をSCRへ保存して計算しています．

CCRフィールド〔**図8**（**e**）〕は，演算結果の状態や割り込み許可，スタック・ポインタの選択に使用するフィールドです．条件分岐のために，演算結果の正負を示すNフラグ



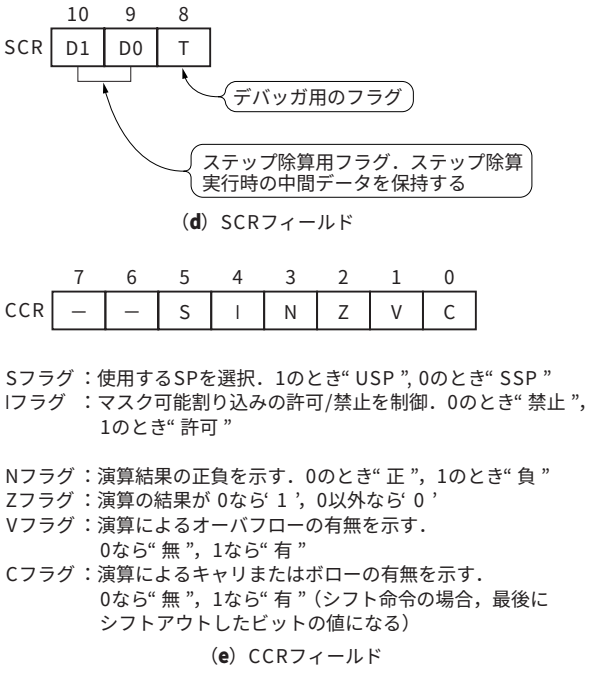


図8　プログラム・ステータス

ILMフィールド，SCRフィールド，CCRフィールドに分かれている．

コラム3　通信インターフェース機能を飲み込み続けるマイコン

通信方式は規格化されることが多く，大幅な仕様の変更があまり行われないという特徴があります．そのため，比較的マイコンに取り込まれやすい周辺機能です．

例えば，通信系の周辺機能としてUART/SIO（シリアル同期/非同期通信）やI/O拡張シリアル（シリアル同期通信）がありますが，筆者らは1985年ごろのマイコン（F2MC-8）にこの機能を内蔵しました．2000年ごろには，I^2C機能を内蔵したマイコンを開発しました．UARTとI^2Cの使用ポート数はシステムごとに異なることもあり，2006年ごろにはUART，SIO，I^2Cの回路を共用化したマルチファンクション・シリアルへと移行しています．同じような目的で，2002年ごろにUARTとしても車載用のLIN（Local Interconnect Network）としても使用できるLIN-UART機能を開発しました．

これ以外の通信系としては，CAN（Controller Area Network）を2000年ごろ，USBを2001年ごろ，Flex-Rayを2006年ごろマイコンに組み込んでいます．

このほか，タイマ系（リロード・タイマ，PWM，ウオッチドッグ・タイマなど）やカウンタ系，アナログ系（A-Dコンバータなど）の周辺機能も，20年以上前からマイコンに搭載されています．

と演算結果が0であることを示すZフラグなどを設けています．また，演算に使用するレジスタの大きさが有限なので，オーバフローなどが発生する場合があります．そのため，演算結果の確かさを示すVフラグやCフラグを設けています．

4）テーブル・ベース・レジスタ（TBR）

テーブル・ベース・レジスタ〔**図9**（**a**）〕には，割り込み発生時のエントリ・アドレスを格納します．TBRの内容と発生した割り込みに対応したベクタ・オフセットの加算値が，参照するベクタのアドレスになります〔**図9**（**b**）〕．初期値は固定ですが，プログラムで割り込みベクタ領域を変更できるようになっています．ブート領域の割り込みベクタと別にすることで，プログラムのバージョンアップにも使用できます．

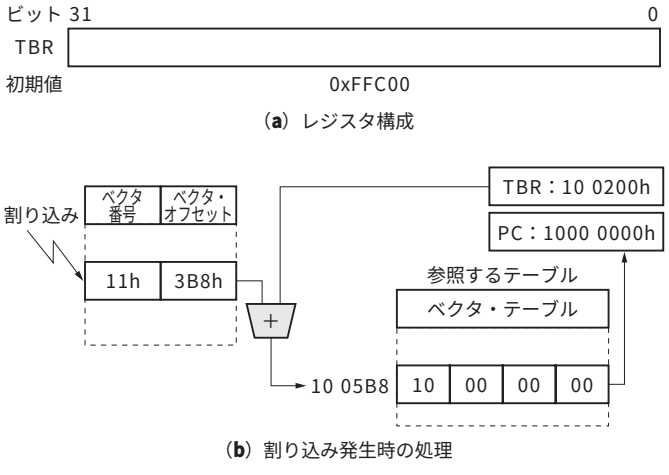


図9　テーブル・ベース・レジスタ
テーブル・ベース・レジスタは32ビット長で，リセットにより0xFFC00へ初期化される．

5）リターン・ポインタ（RP）

リターン・ポインタ〔**図10**（**a**）〕は，call命令の実行後の戻りアドレスを保証するために，call命令実行時のプログラム・カウンタ（PC）の値を保持するレジスタです．call命令の実行により，PC＋2の値がRPへ格納され，ret命令で復帰したときにcall命令の次の命令から実行されることになります〔**図10**（**b**）〕．

6）システム・スタック・ポインタ（SSP），ユーザ・スタック・ポインタ（USP）

システム・スタック・ポインタとユーザ・スタック・ポインタ〔**図11**（**a**）〕は，スタック領域を示すレジスタです．リアルタイムOSなどのスタック領域とユーザ・プログラムのスタック領域を分けて管理することで，プログラムの信頼性を高めています．

7）乗除算レジスタ（MDH，MDL）

乗除算レジスタ〔**図12**（**a**）〕は，乗算の結果を格納した

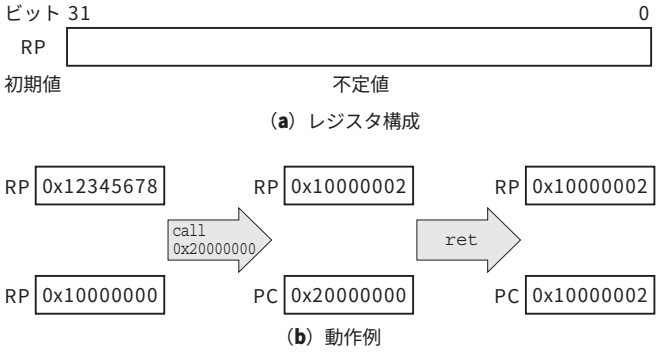


図10　リターン・ポインタ
リターン・ポインタは32ビット長で，リセット後は不定値である．

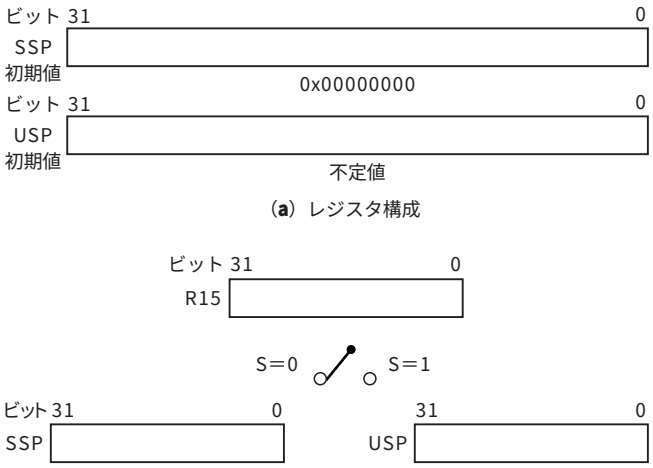


図11　スタック・ポインタ

スタック・ポインタには，SSP（システム・スタック・ポインタ）とUSP（ユーザ・スタック・ポインタ）の2種類がある．

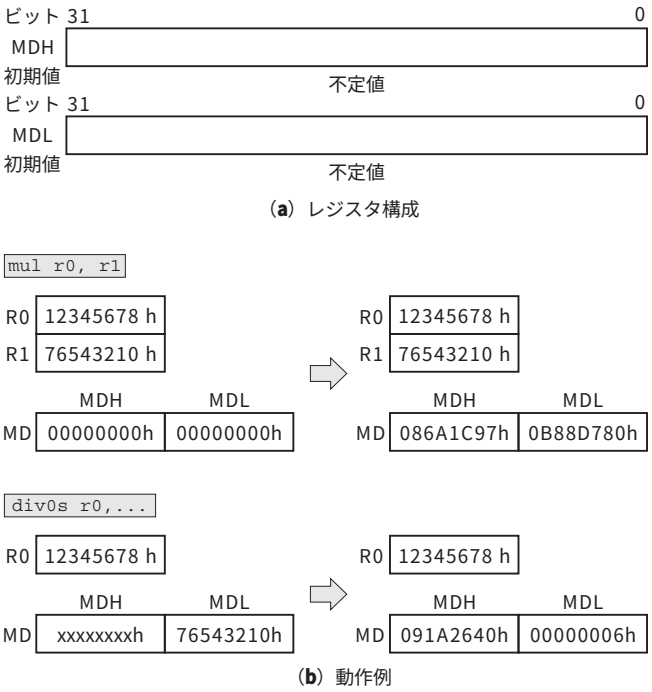


図12　乗除算結果レジスタ

MDHとMDLは共に32ビット長で，リセット後は不定値である．32ビット×32ビットの演算結果を格納するため，合計64ビット長として使用する．

り，除算の除数設定と結果を格納したりする64ビット・レジスタです．32ビットと32ビットの乗算結果を丸め込みなしで得るために，64ビットのレジスタを用意しています．また，除算の場合，除算開始前にMDLに被除数を設定して演算します．最終的にMDHに余りが，MDLに商が格納されます〔**図12**（**b**）〕．

● パイプラインは一糸乱れず，空きを作らず

CPUの処理は，一つの命令の実行を5段階（5ステージ）に分けて処理するパイプライン方式を採用しています〔**図13**（**a**），（**b**）〕．パイプライン方式には，命令実行を並列化することにより，実行速度を引き上げられるというメリットがあります．これにより，通常，複数サイクルかかる命令実行を，見かけ上1サイクルで処理します．

パイプライン動作を行っているCPUでは，連続する命令の前の命令で書き込んだレジスタ値を次の命令で参照しようとすると，参照時に書き込みが完了していないことがあります．これをレジスタ・ハザードと呼びます．レジスタ・ハザードが発生すると，パイプラインに無駄な待ち時間が発生し，CPUの処理速度が低下します．いわゆるインターロックと呼ばれる現象です．

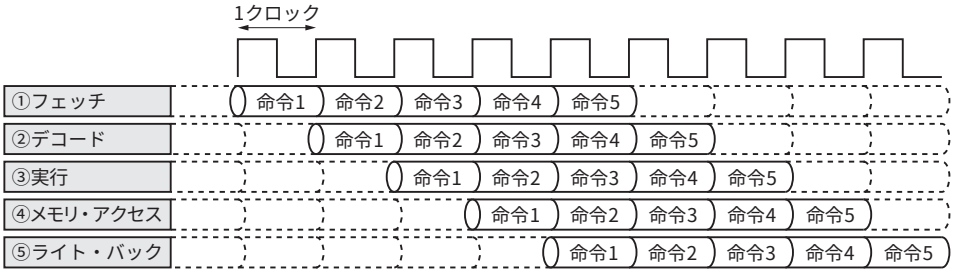
しかし，レジスタ・ハザードが発生しても，命令が参照するレジスタを，前の命令の処理の途中で取り出すことができれば，命令動作を遅らせることなく実行できます．こうした機能はレジスタ・バイパス機能と呼ばれています（**図14**）．FRにはレジスタ・バイパス機能があるので，前の命令の処理の途中から演算結果を取り出して，命令動作を遅らせることなく実行できます．

データがCPUに取り込まれるタイミングが遅い命令では，レジスタ・バイパス機能では対応できないレジスタ・ハザードが発生します．このような場合は，次の動作を待たせることになります．ただしこの場合も，ハードウェアでインターロック処理を行うので，プログラマはレジスタ・ハザードを意識してプログラミングを行う必要はありません．

条件分岐命令の処理では，分岐直後の命令をキャンセル（プリフェッチ・キャンセル）するため，一般にパイプラインに空きが生じてしまいます．FRでは遅延スロット付きの分岐命令が用意されています．この命令を使えば，分岐命令に伴うパイプラインの空きは生じません．

次回は，組み込みマイコンが備える命令セットについて解説する予定です．

ひらいし・いくお
富士通（株）電子デバイス事業本部 システムマイクロ事業部



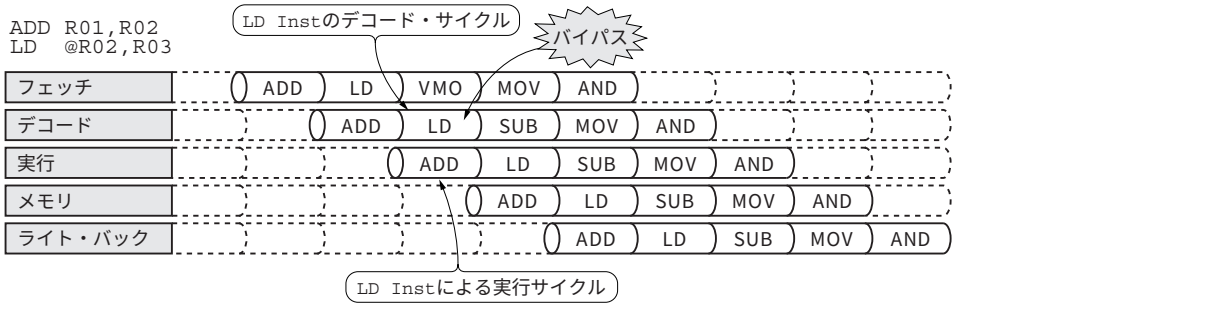
（**a**）パイプライン処理

ステージ	処　理
①フェッチ	命令のフェッチ（読み出し）動作を行う．外部バスに接続したROMからプログラムをフェッチするような場合，フェッチ効率向上のためにプリフェッチを行う
②デコード	命令のデコード（解釈）動作を行う
③実行	命令を実行する
④メモリ・アクセス	メモリへのアクセス，および内部リソースへのアクセスを行う． 低速のメモリへアクセスするような場合は，書き込み効率を向上するために，ライト・バッファへ書き込む
⑤ライト・バック	演算結果をメモリに格納する

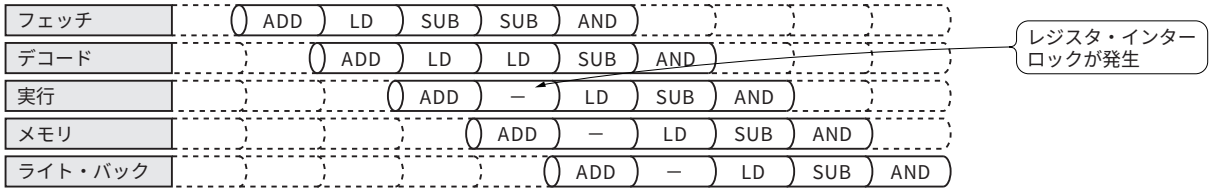
（**b**）各ステージの処理内容

図13
パイプライン

一番上の方形波はクロックを表している．クロックの立ち上がりに合わせて1クロックで1命令を実行する．



（**a**）レジスタ・バイパス機能



（**b**）レジスタ・バイパス機能がない場合

図14　レジスタ・バイパス

この図では，ADD命令で実行した結果がR02に格納され，そのR02を次の命令で使用している．(a)ではレジスタ・バイパス機能があるので，前の命令のADD処理の途中から演算結果を取り出して，遅らせることなくLD命令を実行している．