1.1.1 MPUの構成

コンピュータの中心部MPUを覗く

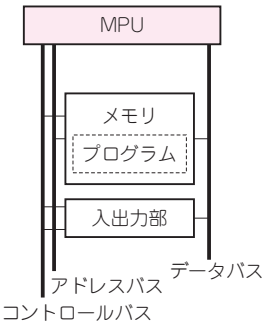
数ある製品の中から開発商品に適したMPUの選択をするには，内部のおおよその構成と動きを知っていることがカギになる．MPUのアーキテクチャは多様であるが，標準的な構成を紹介する．

MPUは，処理の経過が反映されるデータパスと，それを動かす制御部から構成される．

Key Word

CPU，MPU，データバス，アドレスバス，コントロールバス，制御部，データパス，レジスタファイル，ALU，命令レジスタ，プログラムカウンタ

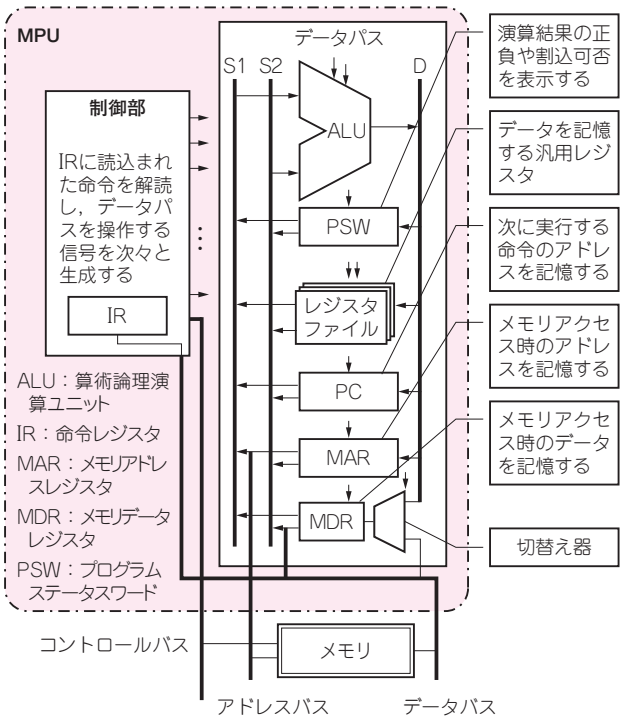
コンピュータの中心部分はかつて**CPU**（**Central Processing Unit**）と呼ばれ，大きな回路ブロックだった．しかし，今日では一つのIC素子で実現され，その機能も以前の大型計算機をはるかにしのぐ．このため，CPUの代わりに**MPU**（**Micro-Processing Unit**）と呼ぶことが多くなった．



コンピュータはMPUとメモリ，そして入出力部から構成される．われわれはMPUがもつ，いくつかの命令語を使ってプログラムを作成し，目的とする仕事をコンピュータに行わせる．プログラムはメモリに記録されており，MPUがプログラム中の命令語を逐次取り出して，命令語に応じた処理を行うことで，プログラムが実行される．

MPUがメモリから命令語を読み出したり，メモリに演算結果を書き込んだりするには，メモリの書き込み場所や読み出し場所を指定する信号線（**アドレスバス：Address Bus**）と，実際にデータが行きかうライン（**データバス：Data Bus**）が必要である．アドレスバスには，MPUからメモリなどに片方向の信号が乗るが，データバスは双方向の信号線で，一般に，16ビットのマイコンなら16本，32ビットマイコンの場合には32本の信号線になる．じつは，これらのバスのほかに，MPU，メモリ，入出力デバイス間でデータの転送を行う際に必要な指示や状態，タイミングなどを伝達する信号線が必要で，**コントロールバス**（Control Bus）と呼ばれる．コントロールバスは，コンピュータのアーキテクチャ（設計思想）によって構成に違いがある．

MPUは，実行中のプログラムにおいて次に実行する命令語のアドレスを記憶する**プログラムカウンタ**（Program Counter：PC）や，読み込んだ命令を格納する**命令レジスタ**（Instruction Register：IR）を備え，アドレスを指定して読み込んだ命令をIRに取り込んで処理を行う．その命令の実行のためにMPUは，PCやIRに加え，演算結果を一時記憶するためのレジスタファイル，そして算術演算や論理演算を行う**算術論理演算ユニット**（Arithmetic and Logical operation Unit：**ALU**）なども備えている．



MPUの内部では，命令語の実行にともない，ALUやレジスタファイル上をデータが移動することになるが，その処理を指示する信号を次々と発生させる必要がある．読み込んだ命令語を解読し，次々と信号を生成する回路部を**制御部**と呼ぶ．命令レジスタIRは通常，制御部にある．これに対し，信号線のラインとレジスタファイルやALUさらにPCなどの回路要素からなる部分を**データパス**と呼ぶ．データパスには，このほかメモリアクセス時のアドレスを一時記憶する**メモリアドレスレジスタ**(Memory Address Register：MAR)，メモリとのデータ転送時に用いられる**メモリデータレジスタ**(Memory Data Register：MDR)，さらに演算結果が正か負かゼロかオーバフローが発生したかといった情報や割込みの可否を示すフラグなどの情報を示す**プログラムステータスワード**(Program Status Word：PSW)なども含まれる．

要点のチェック

□ コンピュータはMPUとメモリ，そして入出力部から構成される．MPUとメモリの間には，メモリ上の場所を指示する**アドレスバス**とデータが乗る**データバス**，さらに**コントロールバス**が必要になる．

□ MPUは演算を行う**ALU**や各種レジスタとバスからなる**データパス**と，データパス上の各素子に命令語の処理に必要な**制御信号**を次々と与える制御部に分けられる．

□ データパスには，レジスタファイルのほか，メモリアクセス時のアドレスを一時記憶する**メモリアドレスレジスタ**，メモリとのデータ転送時に用いられる**メモリデータレジスタ**，次に実行する命令語のアドレスを記憶する**プログラムカウンタ**，そして算術演算や論理演算を行う算術論理演算ユニット(**ALU**)などを備えている．

□ 制御部は読み込んだ命令を記憶する**命令レジスタ**をもち，命令語を解読し，その処理に必要な信号を次々と生成して，**データパス**上の素子に与える．

1.1.2 MPUの動作

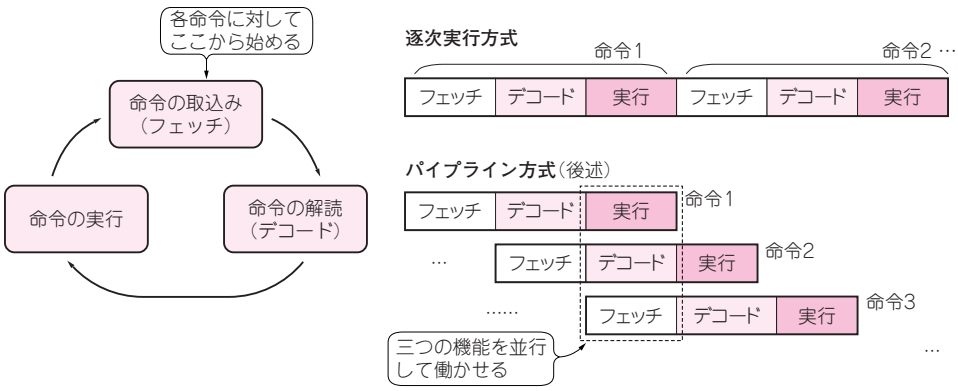
MPUは命令を次々と実行していく

MPUがプログラムを実行するとは，そのMPU固有の命令を順次実行していくことである．MPUの基本動作は，命令の読込み(フェッチ)，命令の解釈(デコード)，命令の実行などの作業を行い，「一つの命令の全体の実行」を完了する．

Key Word

命令，命令レジスタ，命令の読込み(フェッチ)，命令の解釈(デコード)，命令の実行，プログラムカウンタ(PC)，命令コード，クロック

MPUがプログラムを実行するということは，メモリに記憶されたMPU固有の**命令**(Instruction)を順次実行していくことである．MPUの基本動作はメーカや機種によって異なるが，ほぼ，**命令の読込み**(**フェッチ**：Fetch)，**命令の解釈**(**デコード**：Decode)，**命令の実行**(Execute)という作業を行って，一つの命令の全体の実行を完了する．そして，次々とこれを繰り返すことによって，プログラムの処理が進むことになる．

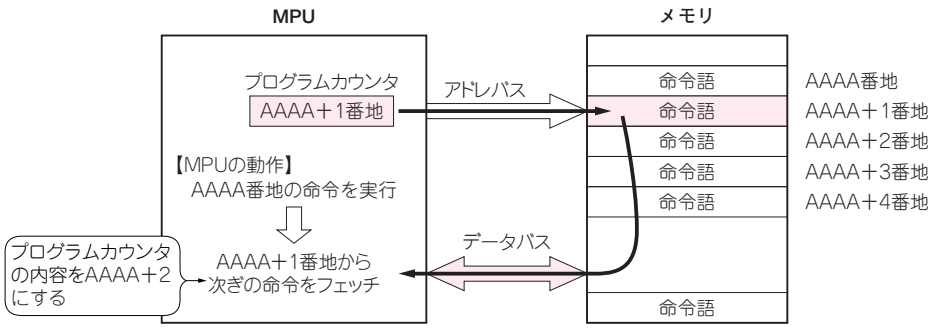


① 命令の取込み

MPUは，メモリ上の**プログラムカウンタ**(PC)が指すアドレスに格納されている**命令語**を読込み，**命令レジスタ**に入れる．このときPCの内容は，次の命令のアドレスに更新される．つまり，分岐命令の実行や後述の割込み発生の場合を除いて，現在の命令の次に実行されるのは，次の位置の(アドレスの高い)命令となる．

② 命令の解釈

MPUは，命令レジスタの命令を解釈し，どのような動作を行うかを決める．**命令語**は**命令コード**(オペレーションコード：**OPコード**などとも呼ばれる)とオペランドなどから成る．オペランドには，演



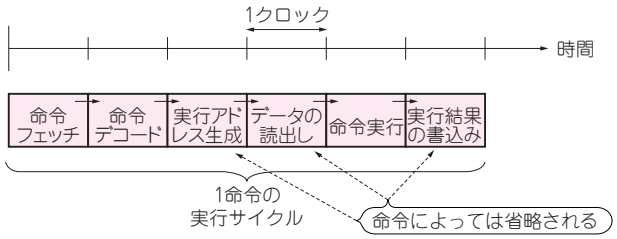
算の対象となるデータのアドレスまたは分岐先のアドレスなどが含まれる．命令語は命令コードに応じて解釈されるが，それはMPUの種類によって異なり，MPUのメーカ独自のものである．

③ 命令の実行

MPUは，解釈された内容によってその命令を実行する．命令には，演算命令，データ転送命令，分岐命令などがあるので，この段階では各命令に応じた動作をすることになる．たとえば，転送命令であれば，レジスタまたはメモリ間でデータの読出しと書込みを行う．なお，分岐命令の場合，分岐条件を満足すれば，PCの内容を変更する．その結果として，プログラムは分岐することになる．

一つの命令を実行するために必要な時間は，データの読出しや書込みがメモリ領域であるか，レジスタであるかによって違ってくる．レジスタであれば，MPUの内部にあるので余分な時間はいらないが，メモリであれば，後述のバスを通してデータのやり取りをするので，余分な実行クロック数を必要とする．

クロックとは，MPUが動作する時にタイミングをとるための基本となる周期的な信号である．通常，水晶発振子を利用して発生させる．毎秒の発振回数をクロック周波数と呼び，単位はHz(ヘルツ)が使われる．



要点のチェック

□ MPUがメモリに記憶されたMPU固有の**命令**を順次実行していくことでプログラムが実行される．

□ MPUが一つの命令を実行する基本動作を3段階に分けると，①命令の読込み(**フェッチ**)，②命令の解釈(**デコード**)，③命令の実行という作業になる．

□ 命令取込みステップでは，MPUはプログラムカウンタ(**PC**)が指すアドレスのメモリに格納されている**命令語**を取り出し，**命令レジスタ**に格納する．

□ MPUは，**クロック**信号によって各ステップの処理を順次実施する．

1.1.3 MPUの種類

組込み系に用いられるMPUは多彩だ

コストパフォーマンスを向上させるには，開発しようとする装置や機器に適したMPUの選択が求められる．MPUにはメモリや周辺機能ユニットを内蔵したシングルチップマイコン(1チップマイコン)から，システムLSI設計時にソフトウェア資産として提供されるものまで多彩である．

Key Word

マイクロプロセッサ，シングルチップマイコン，IPコア，システムLSI，低消費電力，スリープモード，スタンバイモード

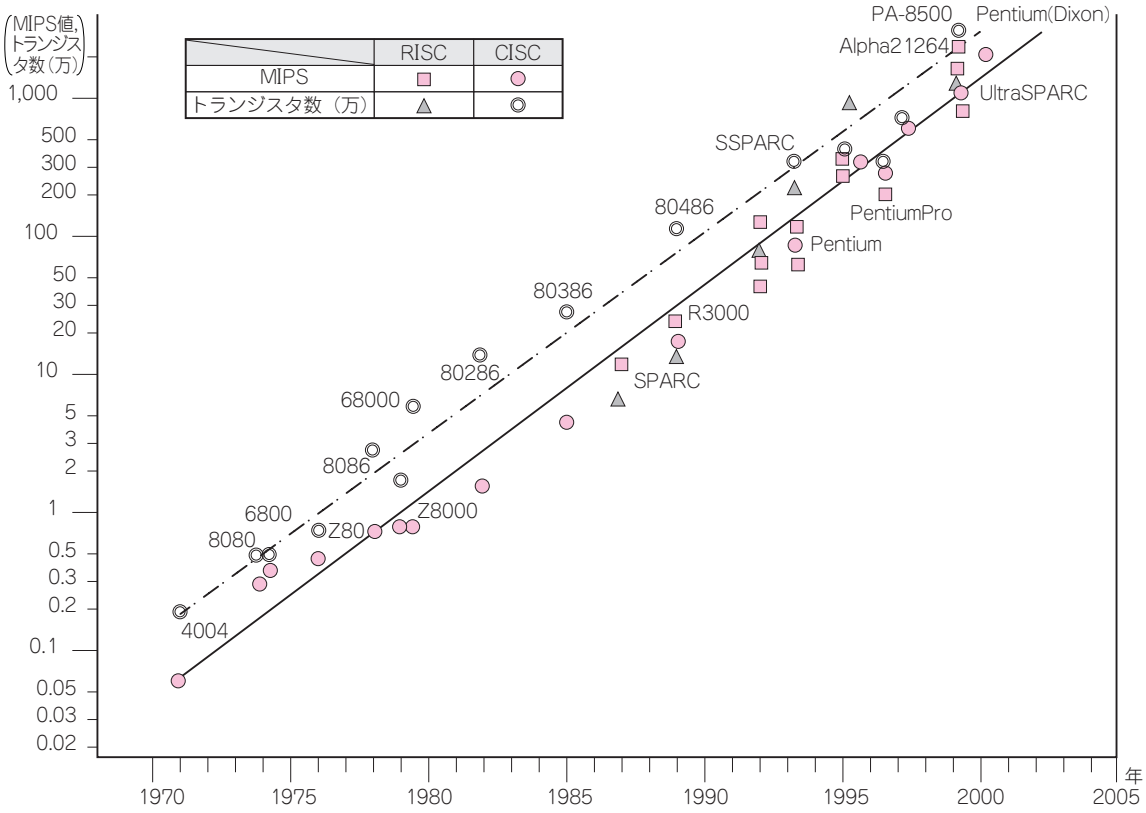
1971年にインテル社(米国)がi4004を開発した．マイクロプロセッサの登場である．当時，電卓の使用が急増していたが，その機能変更を容易にする必要性から，プログラマブルな算術計算用LSIとして産み出されたものがi4004であった．1973年には同一アーキテクチャのもと8ビットバージョンの8008が発売されたが，やがてミニコンピュータやメインフレームコンピュータのアーキテクチャを踏襲した8080，8086，6800というマイクロプロセッサが開発され，主流となった．その後16ビットや32ビットのMPUへとバス幅が拡大し，ワードプロセッサやパソコンなどに搭載された．

一方，MPUのほかに，タイマや通信制御といった周辺機能LSI(**ペリフェラルIC**)の開発も進み，8080用に開発したインテル社の各種周辺機能LSIはデファクトスタンダードとなり，その後の仕様の標準となっていった．

また，MPUのほかに，この周辺機能LSIやメモリを備えた**シングルチップマイコン**が開発され，小型化や低コストが優先される組込みシステムで多く用いられるようになった．これらの多くがLSI製品として商品化され市場に出されたのに対し，ARM社の各種製品は，チップとしてではなく，回路ライブラリの一つである**IP**(Intellectual Property：知的財産)**コア**として製造メーカに供給される．

携帯電話やデジタルカメラでは，**システムLSI**と呼ばれる「ユーザが開発したコントローラ」を実装することで大幅な小型化を図ろうとする傾向が強く，IPコアとして入手したMPUに多様な周辺機能が組み合わされて独自のマイコンを製造して利用するケースが多い．IPとしてのMPUコアの役割は，ますます拡大する．

さらに，組込みシステム用のマイコン，とくに携帯用の機器に使われるものに対しては，**低消費電力**が要求されることが多い．そのような要求に対応するために，MPUが**スリープモード**(Sleep Mode)や**スタンバイモード**(Stand-by Mode)と呼ばれるような低消費電力状態をもっているものが多い．低消費電力は，MPUの動作を停止したり，クロック周波数を極端に低くしたりして実現している．MPUだけの停止では省電力の効果が小さく，さらにクロックの停止をすると，効果がより大きくなる．このため，この面での技術開発も積極的に行われている．



要点のチェック

□ 1971年にインテル社(米国)がマイクロプロセッサi4004を開発した.

□ 8080，8086，6800はミニコンピュータやメインフレームコンピュータのアーキテクチャを踏襲して開発されたマイクロプロセッサである.

□ マイクロプロセッサやメモリ以外に，タイマや入出力機能を実現するために実装される機能素子を，ペリフェラルLSIという.

□ デジタルカメラや携帯電話などに用いられるコントローラには，IPコアとして入手したMPUに多様な周辺機能を組み合わせて開発したシステムLSIが用いられ，高機能化，小型化に寄与している.

□ 携帯用の機器は低消費電力が要求されるため，スリープモード(Sleep Mode)やスタンバイモード(Stand-by Mode)と呼ばれるような低消費電力状態のMPUが使われる.

消費電力を抑えるには，MPUを動作停止したり，クロック周波数を低くすると効果がある.

1.1.4 CISCとRISC

処理速度の向上に，相反する二つの戦略があった

処理速度向上は，MPU設計者の最大目標である．豊富な機械語を用意し，ソフトウェアの負担を軽減することで高速化を意図した時代から，単純な命令語に限定することで高速化をはかるRISCの概念がMPUの高速化に貢献した．

Key Word

CISC，RISC，逐次実行，パイプライン，レジスタマシン，ベクトルプロセッサ，スーパスカラ方式，VLIW

コンピュータの処理速度向上の手段として，たくさんの命令を用意し，プログラムのステップ数を少なくしようとする考え方があった．この考え方によるプロセッサを**CISC**（**Complex Instruction Set Computer**）**マシン**と呼ぶ．CISCマシンでは命令種が豊富になるが，MPU内部の構成は複雑になる．

また，命令の処理が複雑になるほど，実行に要するクロック数は多くなる．その結果，命令によって処理にかかるクロック数もさまざまになる．CISCマシンの中には，メインメモリ上のデータをそのまま演算対象とする演算命令語を有するものもあった．

しかしCISCマシンでは，豊富な命令語を有する代償として，命令語の解読に時間を要する上，命令語によって処理クロック数が異なるために，パイプライン処理（1.1.5項参照）に適さない問題もあった．しかも，実際のプログラムで用いられる命令語はそれほど多くはない．

その反省から，命令の種類を大幅に減らし，しかも単純な命令とすることによってMPU内部の構成を簡素化し，処理効率を高めようとする方式が注目された．この方針で設計されたMPUが**RISC**（**Reduced Instruction Set Computer**）**マシン**である．RISCマシンは命令が単純なため，個々の命令実行に要するクロック数は少なくて済み，高速なクロック周波数へも対応しやすくなる．

さらに，各命令語実行に要するクロック数（**CPI：Clock Cycles Per Instruction**）を同一にすることで，パイプライン処理にも有利になる．

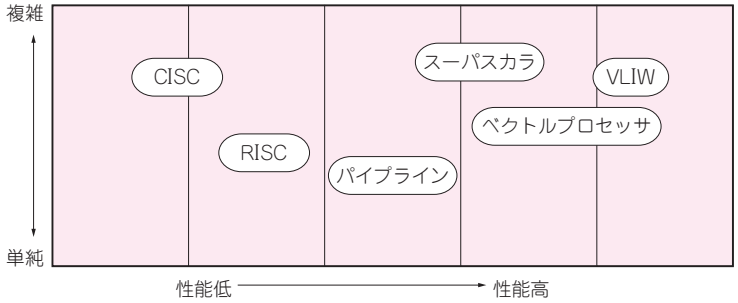
RISCマシンでは，基本的に演算処理はレジスタに対して行われるレジスタマシンが多く，処理対象となるデータは，あらかじめデータ移動命令を用いてメモリからレジスタへロードしておかなければならない．

一方，より高速化を求める技術開発も積極的に行われた．配列相互の演算は同一操作の繰り返しとなる．このような処理を同時並列的に実行させることで高速化を意図した**ベクトルプロセッサ**はその代表である．

さらに，パイプラインを複数用意し同時に実行させようという**スーパスカラ方式**も検討されたが，命令語間に依存関係があるものを同時に実行することはできないため，限界があった．

この問題の解決のために，あらかじめ命令フィールドに相互に独立な複数の命令を記載できる**VLIW**

(**Very Long Instruction Word**)と呼ばれるアーキテクチャが注目されている.



年　代	トピックと傾向
1970 年代前半	マイクロプロセッサが開発される
1970 年代半ば～後半	ターミナルへの搭載，パソコン用 MPU の登場
1980 年代前半	32 ビット CISC MPU が登場する
1980 年代後半	RISC MPU が登場する
1990 年代前半	64 ビット RISC MPU の登場
1990 年代後半	RISC MPU(MIPS R10000 など)，CISC MPU(Pentium Pro など)で高速化を競う
2000 年代前半	さまざまな用途(モバイル，ゲーム，PC，サーバ)用 MPU に分化
2000 年代後半	電力あたりの性能向上，マルチコア MPU 登場

要点のチェック

□ 豊富な命令群をもってプログラムのステップ数を少なくし高速化を実現しようとする方法があり，この概念で開発されたMPUを**CISCマシン**と呼ぶ.

□ 単純な命令のみに限定し，命令数を少なくすることで高速化をはかろうとして開発されたMPUを**RISCマシン**という.

□ RISCマシンは命令の実行サイクル数を**均一**にできるため，**パイプライン**方式に適している.

□ **レジスタマシン**は，レジスタにあるデータ同士の演算を行うMPUで，演算に必要なデータは，いったんメモリからレジスタに転送する必要がある.

□ RISCマシンの高速化を意図して,配列演算の同時実行に着目した**ベクトルプロセッサ**が開発された.

□ パイプラインエンジンを複数個用意して同時に実行させる**スーパスカラ**方式が提案されたが，命令語の独立性の確保が問題とされた.

□ これに対し，あらかじめ独立性を有する複数の命令からなる長形式の命令語を用いて複数のパイプラインエンジンを同時起動させる**VLIW**アーキテクチャなど，高速化を意図した技術が見られる.

1.1.5 パイプライン方式

パイプライン方式はMPUの処理をベルトコンベア方式で行うものだ

命令語の実行を逐次的に行うと，1命令に数クロックを要する．1クロックずつ時間をずらし，並列に処理するパイプライン方式では，1命令の平均実行時間を2クロック以下にまで縮小化できる．

Key Word

パイプライン方式，逐次処理，CPI，セグメント，パイプラインハザード，構造ハザード，制御ハザード，データハザード

1949年Wilkesがケンブリッジ大学でプログラム内蔵型の計算機EDSACの完成に成功して以降，ユーザプログラムはメモリに格納され，それが逐次読出されて実行される形態が確立した．

プログラムがメモリに内蔵されている以上，処理の遂行にはプログラムの命令を読込む作業が必要になる．この作業を**命令読込み**(Instruction Fetch：**IF**)という．

次にプロセッサは，その命令が何であるのか**解読**(Instruction Decode：**ID**)する．そして，演算を**実行**(Execution：**EX**)し，結果を指定されたレジスタに**格納**(Write Register Buffer：**WB**)する．命令にはデータの転送命令や分岐命令(プログラムの流れを変えるので制御命令ともいわれる)もある．このときには，演算の代わりにアドレスの生成が行われ，レジスタへの格納の前に，データの**移動**(Memory Access：**MEM**)が発生する．

このように，命令の実行にはIF，ID，EX，(MEM)，WBという，四つから五つのステップが遂行され，逐次処理型のMPUでは，これが終わるとまた次の命令が読み込まれる(IFステップ)．

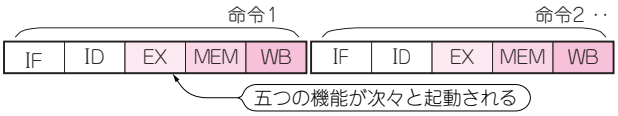
一方，**パイプライン方式**のMPUでは，命令の実行が終わらないうちに次の命令が読み込まれ，並列的に処理が行われる．ここで，各命令の実行ステップ(ステージともいう)がIF，ID，EX，MEM，WBの5セグメントからなる場合のパイプライン処理を考える．命令jのIFセグメントが終わると次のIDステージに処理が移されるが，IFを行う機構は終了しているため，次の命令$j+1$のIFも同時に遂行できる．したがって，M行(ライン)の命令が実行されるのに要するクロックサイクル数は，$5+(M-1)$となる．1命令の実行にnセグメントを要する場合には，$n+(M-1)$クロックだが，1命令あたりに換算すると，$n \ll M$であるから$(n+(M-1))/M \fallingdotseq 1$となる．

この1命令あたりの平均クロックサイクル数を**CPI**(Clock Cycle Per Instruction)といい，理想的なパイプライン処理の場合CPI=1になる．

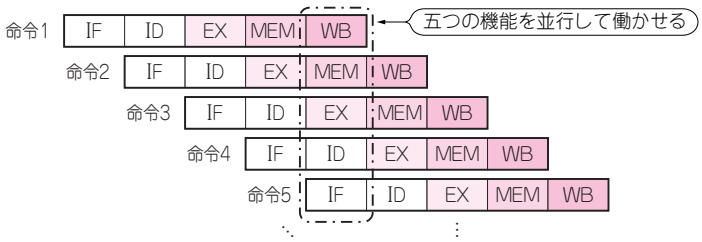
ただし，実際には並列処理を行えなくなる状態が生じることもあり，CPIは1より大きくなる．この処理の乱れを引き起こすことを，**パイプラインハザード**という．

パイプラインハザードには，ハードウェア資源が競合するために同時並列実行ができない**構造ハザード**や，分岐命令が成立したために別な命令を再度読み込まなくてはならなくなる**制御ハザード**，さらに

直前の命令の実行結果のデータを次命令で使用する場合などに発生する**データハザード**などがある．



（**a**）逐次処理方式



（**b**）パイプライン方式

ハザード	内　容
構造ハザード	それぞれのステージを実行するハードウェア資源が競合し同時には実行できなくなるハザード
データハザード	先行命令の処理結果を用いるため，並行処理が行えなくなるハザード ロード命令 IF ID EX MEM WB　ステージの最後でメモリからの読込みを終了する 次命令 IF ID 待ち EX MEM WB　そのデータを演算するには待たねばならない
制御ハザード	分岐命令が成立したために，分岐先の命令をフェッチしなければならなくなるハザード 分岐命令 IF ID EX MEM WB　分岐先アドレス計算　分岐成立ならPCを書き換える 次命令 IF ID EX IF ID EX MEM WB　廃棄　分岐先命令再フェッチ

要点のチェック

□ MPUがプログラムの命令を実行するときには，命令読み込み，**解読**，演算，レジスタへの書込みなど，いくつかのステップを経て行われる．

□ **逐次処理型**のMPUでは，一つの命令実行が終わってから次の命令が読み込まれ，実行される．

□ **パイプライン**方式では，命令の実行が終わらないうちに次の命令が読み込まれ，**並列的**に処理が行われる．

□ パイプライン方式において並列処理が行えなくなる乱れを引き起こすことを，**パイプラインハザード**という．

□ パイプラインハザードには，ハードウェア資源の競合に起因する**構造ハザード**や，分岐命令が成立したために発生する**制御ハザード**，さらに読み込んだデータを次命令で使用する場合などに発生する**データハザード**などがある．

□ 1命令あたりの平均クロックサイクル数を**CPI**という．

□ 理想的なパイプライン処理の場合CPI＝1であるが，パイプラインハザードのために**1より大**になる．

1.1.6 マイクロプログラミング

MPUの制御部の設計法に革命をもたらした技術

命令語の追加や変更のみならずデータパスの回路素子の仕様変更は，制御部の設計変更を余儀なくする難問だった．マイクロプログラミングは，ここに組織的な設計法を持ち込み，設計の柔軟化をもたらした．CISCを産み出した遠因でもある．

Key Word

マイクロプログラム，水平型マイクロ命令，垂直型マイクロ命令，制御部，データパス

MPUを構成する制御部とデータパスのうち，データパスはALUやいくつかのレジスタを規則的に配置すればいいが，制御部は，複雑な処理が求められるにもかかわらず，命令語の仕様が決定しデータパスの構造が決まらないと，設計できない．このことがコンピュータ設計のネックになることを懸念したWilkesは，データパスに次々と送るべき信号をあらかじめデータセットとして用意する方法を考えた．命令実行時には，命令語に対応したデータセットを選べばいい．しかも，このデータセットは命令語やデータパスに応じて，いかようにでも変更でき，命令語を実行するためのプログラムともいえる．このデータセットを制御部のメモリに書き込むだけで，任意のコンピュータができることになる．Wilkesはこの構造を，**マイクロプログラミング**(Microprogramming)と命名し，データセットをマイクロプログラムと呼んだ．

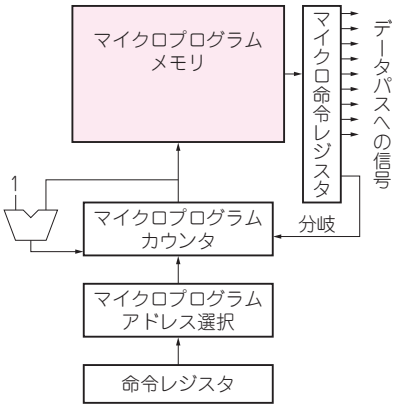
マイクロプログラミング方式のコンピュータは，同一のハードウェアで異なった機能にすることもできるし，任意の命令語の追加も可能となる．プログラムの上位互換性の保証も容易にできる．

しかし，命令語の実行に用いるプログラムなので，そのアクセス速度は高速でなければならない．いかにして制御部の高速メモリの容量を少なくするかがコスト面からは重視された．マイクロプログラムを記述するマイクロ命令は，データパスに与える信号の並びであるが，信号ビットはALUやレジスタ単位にフィールドという単位にまとめられる．同時に用いられることのないフィールドがあれば，そのフィールドを共用したり，フィールドの信号線が同時にONすることがない場合にはコード化したりして短縮化がはかられた．

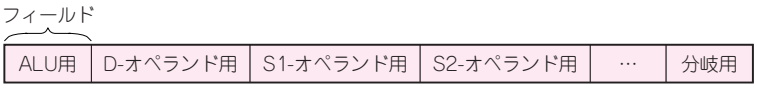
また，マイクロ命令の各フィールドは常時アクティブになるわけではないので，マイクロ命令を分割してアクティブになっている箇所のみをプログラムするといった方法が，記憶部削減のために考えられた．

このようにしてできたマイクロプログラムは縦長となるため，一つ一つの命令を**垂直型マイクロ命令**と呼ぶ．垂直型マイクロ命令からなる垂直型のマイクロプログラムは，メモリ効率は良いが実行時間がかかる．これに対し，処理速度を重視し短縮化をはからない命令を**水平型マイクロ命令**と呼ぶ．

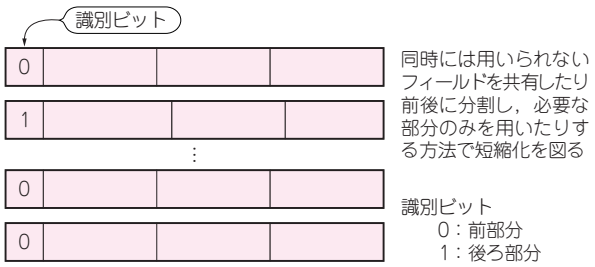
マイクロプログラミング方式



水平型マイクロ命令の例



垂直型マイクロ命令によるマイクロプログラムの例



要点のチェック

□ MPUの制御部の設計法の一つに**マイクロプログラミング**方式がある．

□ MPUのハードウェアが同一でも，**マイクロプログラム**を書き換えれば違った機能のMPUにできる．

□ マイクロプログラムは高価な高速メモリを必要とするため，メモリの節約をはかる**垂直型**マイクロプログラムが考案された．

□ **水平型**マイクロ命令は，**垂直型**マイクロ命令より高速化に適している．

1.2.1 ROMとRAM

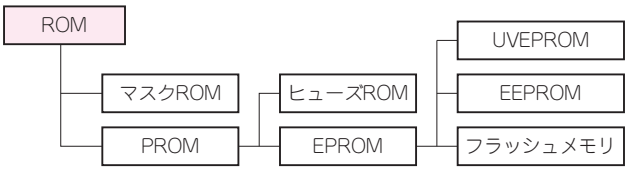
組込みシステムのプログラムはROMに書かれることが多い

開発対象機器やシステムにふさわしいメモリの選択がカギになる．とくにプログラムやデータを格納するROMの適切な選択は，組込み技術者の腕の見せ所でもある．

Key Word

ROM，RAM，マスクROM，EPROM，EEPROM，SRAM，DRAM，RAS，CAS，リフレッシュ

主記憶装置を構成する記憶素子は，電源が切れても書き込まれた情報が消失しない**ROM**（Read Only Memory）と，内容を自由に変更できるが電源を切ると内容が消えてしまう**RAM**（Random Access Memory）に分けられる．多くの組込みシステムでは，プログラムや固定値をROMに記憶させ，書換えや消去ができないようにする．一方，演算結果の記憶や画像メモリには，随時書換え可能なRAMを用いる．



ROMは読出し専用のメモリデバイスで，電源を切っても記憶されたデータは消えない．ROMをさらに分類すると，デバイスの製造段階でデータを作り込んでしまう**マスクROM**と，製造後にデータの書込み/消去ができる**EPROM**（Erasable Programmable ROM）とに分けられる．マスクROMは製品段階でデータが決まってしまうので，デジタルカメラのように大量に販売される組込み機器のプログラムや，固定的なデータの格納に使われる．一方，EPROMは試作段階や少量生産時のプログラム格納用に使用される．過去にはデータの消去に紫外線を使用する**UV-PROM**（Ultra Violet Programmable ROM）が用いられた．最近では，電気的に消去や書込みができる**EEPROM**（Electrically Erasable Programmable ROM）が主流となった．

RAMは，**SRAM**（Static RAM）と**DRAM**（Dynamic RAM）の2種類に分けられる．SRAMは，6個または4個のトランジスタで構成されるフリップフロップに1ビットの情報を記憶させるもので，電源が供給されている間は記憶内容を保持できる．SRAMは低消費電力と高速という利点があるが，DRAMと比較すると高価になる．

DRAMは，**SRAM**に比較して低速ではあるが，回路が単純で，集積度も簡単に上げることができ，価格も安いため，コンピュータのメインメモリとして用いられる．

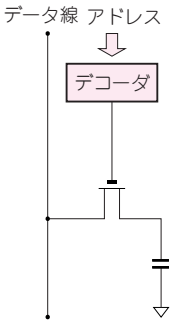
DRAMでは，1個のトランジスタと1個のコンデンサで構成される記憶セルが碁盤の目のように配置

され，横方向の行アドレスと縦方向の列アドレスの指定により行う．MPUから送られてくるアドレス信号は，**RAS**（Row Address Strobe）と**CAS**（Column Address Strobe）と呼ぶ信号線によって行と列に振り分けられる．

そのため，メモリへのアクセスは「行アドレス指定→列アドレス指定→データ読出し（書込み）」といった順番で行われる．このアクセスを行わないと，DRAMの記憶内容が消滅するため，DRAMはあるインターバル内でリフレッシュ動作を行わなければならない．

DRAMはEDO DRAMなどのような非同期方式のものが使用されていたが，現在は一般的にシステムクロックと同期して動作する**SDRAM**（Synchronous DRAM）が主流となっている．SDRAMはメモリバスクロックに完全に同期してデータ転送を行うことで高速化がはかられる．

さらに，メモリバスクロックの2倍でデータを転送できる**DDR SDRAM**（Double Data Rate SDRAM）が後継として規格化された．現在はDDR SDRAMを低電力化してより高いメモリバスクロックへ対応できるように改善した，第2世代の**DDR-2 SDRAM**も登場した．



要点のチェック

□ 記憶素子には，電源が切れても情報が消失しない**ROM**と，書換え可能であるが電源を切ると内容が消える**RAM**に分けられる．

□ LSIの製品段階でデータが書き込まれるROMを**マスクROM**という．これに対し，製造後にデータの書込み/消去ができるものが**EPROM**で，データの消去に紫外線を使う**UV-PROM**と電気的に消去や書込みができる**EEPROM**に分けられる．

□ RAMは，SRAMとDRAMの2種類に分けられる．**SRAM**は電源が供給されている間は記憶内容を保持できるが，**DRAM**は一定時間ごとに**リフレッシュ**をすることで記憶内容が維持できる．このため，DRAMは**処理時間**が遅いが廉価な**大容量**メモリとして用いられる．

□ DRAMでは，MPUから送られてくるアドレス信号を，**RAS**と**CAS**によって行と列に振り分ける．

□ DRAMの記憶内容を維持するには，限られた時間内での**リフレッシュ**が必要である．

□ DRAMはバスクロック信号と非同期方式のものが使用されていたが，現在では同期式の**SDRAM**が主流となっている．

□ SDRAMでは，バスクロックに完全に同期してデータ転送を行うことで**高速化**をはかっている．

1.2.2 EEPROMとフラッシュメモリ

電気的に消去できるROMは便利だ

紫外線消去式ROMから電気的消去可能ROMに移行し，ROMを実装したままで，データの書き換えが可能になった．そしてそれは，フラッシュメモリへと発展する．

Key Word

SDカード，コンパクトフラッシュ，メモリースティック，EEPROM，フラッシュメモリ

データの一時記憶に，**SDカード**や**コンパクトフラッシュ**，**メモリースティック**などの記憶媒体が用いられる．これらは，EEPROMの一つである**フラッシュメモリ**を一般製品化したものである．

EEPROM（Electrically Erasable and Programmable ROM）のデータ消去の際には，電源電圧よりも高い電圧が必要だが，最近のEEPROMは内部に電圧昇圧回路を内蔵していて，デバイスを基板に実装したままでデータの消去や書込みが可能になっている．

ただ，1ビットのみの書き換えはできなくて，必ず一度すべてのデータを消去した後で書き換える必要がある．

このようなEEPROMの欠点を改良したのが，フラッシュメモリ（Flash Memory）である．フラッシュメモリは，半導体メモリの長所である**読み書きの速さ**と**低消費電力**を兼ね備えた理想的な半導体記憶装置といえる．また，記憶している情報のすべてを一度に消せるのもフラッシュメモリの大きな特徴で，ブロック単位で消去や書換えができるようになった．フラッシュメモリのメモリセルアレイを次の図に示す．

図において，ソースラインとワードラインで指定された各メモリセルの情報が，ビットラインに表れることになる．

フラッシュメモリには記憶セルの接続構造により**NOR型**と**NAND型**の2種類がある．NOR型フラッシュメモリは，セクタごとにデータを消去しビットごとに書換えを行うため，1バイト単位の書込みが可能で，アクセスも高速だが，書込み速度は遅く，大容量化には向かない．

一方，**NAND型フラッシュメモリ**は，ビットごとに書き換えはせず，セクタごとにデータを書き換える．**NOR型フラッシュメモリ**と比べ，回路規模を小さくすることが可能であり，大容量化に向いている．また書込み速度や消去速度も比較的高速だが，データへのランダムアクセス性は劣る．

NAND型のフラッシュメモリは大容量のデータ記録媒体として商品化され，デジタルカメラ，携帯電話，USBメモリなど幅広く普及している．