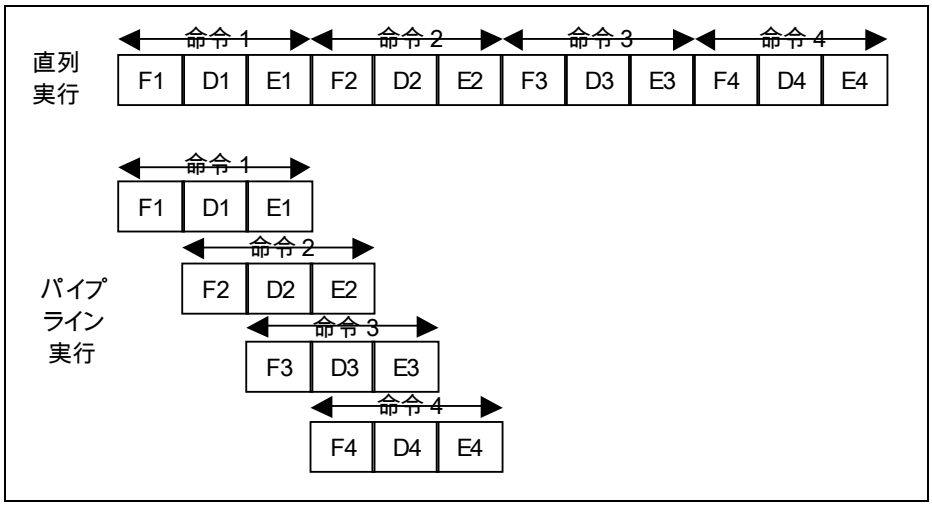
第１４回　　高速化の工夫・パイプラインアーキテクチャ

［１］　パイプラインとは



教 p.96 の図 10.2（右図）を見て、次の問に答えよ。

ａ．命令 1 の実行が 3 つの箱 F1, D1, E1 に分かれているのは、何を示しているのか

ｂ．　直列実行で、命令 1 ~ 4 の実行にかかる時間は、それぞれの箱を同じ時間Ｔとすると、全体でどれだけの時間か

ｃ．　パイプライン実行で、命令 1 の D1 と命令 2 の F2 が同じ時間帯にあるのは、どうしてか。なぜそれが同時に起こることができるのか

ｄ．　パイプライン実行で、命令 1 ~ 4 の実行にかかる時間は、それぞれの箱を同じ時間Ｔとすると、全体でどれだけの時間か

ｅ．　パイプライン実行の方が、直列実行に比較して、時間が短くなっているが、なぜそのようなことが可能なのか

ｆ．　上記の図の場合の加速率Ａ（直列実行に比較してパイプラインが何倍早く実行できるか）を求めよ。但し、

加速率Ａ　＝　１／（パイプライン実行にかかる時間 / 直列実行にかかる時間）

＝　直列実行にかかる時間 / パイプライン実行にかかる時間

ｇ．　パイプライン実行にかかる時間Ｐの一般式を求めよ。但し、全部の命令の数をＮ（上図では 4 命令）、1 つの

命令の中の箱（ステージ）の数をＳ（上図では 3 ステージ）と書くことにする。

ｈ．　直列実行にかかる時間Ｒの一般式を求めよ。

ｉ．　加速率Ａの一般式を求めよ

ｊ．　命令数が無限（無限に長い実行）になったとき、加速率Ａはどういう値に近づくか

（注）1 つの命令の中のそれぞれのステージ（F, D, E）の長さは、もともとは同一でないかもしれない（教 p.97 の図 10.3）が、パイプライン実行するためには揃えなければならない。短いステージを同じ長さに延ばしてやる必要があり、その分の時間は無駄（オーバーヘッド）になる。
（注）ステージの数は、設計のやり方で変わる。教 p.97 の図 10.3 や図 10.4 では 1 命令が 5 ステージになっている。

［２］　パイプラインのハザード

パイプラインが円滑に流れず停滞することを、ハザードと呼ぶ。教 10.2.1 節にあるように幾つかの原因がある。また、ハザードの発生を軽減するため、教 10.2.2 節にあるような、遅延分岐や分岐予測といった技術が使われる。

ａ．　教 10.2.1 を参照して空欄を埋めよ

ハザードとは：　何らかの理由で①（　　の　　）が乱れた時、②（　　）が低下することがあり、その要因をハザードと呼ぶ。

ｂ．　３種類のハザードについて説明せよ

①　（　　）ハザード：
②　（　　）ハザード：
③　（　　）ハザード：

ｃ．　次の現象は、ｂで整理したうちの、どのハザードの説明か？

①　命令実行パイプラインにおいて、条件分岐命令（COMET li であれば JPL や JMI など）の直後に発生する。たとえば JPL 命令の場合、次の命令のフェッチを、JPL の結果どちらへ分岐するかが定まらないうちに行うことはできない。だから、次の命令のフェッチは JPL 命令の結果が定まるまで（パイプラインを止めて）待たなければならない。

②　前の命令が実行結果をレジスタやメモリなどに書き込まないうちに、次の命令がそのレジスタやメモリなどを

参照（読み出し）するのは困る。具体的には、COMET II であれば ST 命令がメモリに書き込んだデータを、その直後に算術論理演算命令（たとえば ADDA）で読み出すプログラムの場合、前の ST 命令のメモリへの書き込みステージが完了するより先に、次の ADDA 命令のメモリ読み出しが始まることは困る。前の命令の書き込みステージが完了するまで、次の命令の読み出しステージは待たなければならない。
③　ハードウェアの構造上、同じ部分を同時に利用することができない場合がある。たとえばメモリに同時にアクセスしたり、レジスタに同時にアクセスしたり、また ALU を同時に使ったりすることはできない。（注：それぞれの機能が 1 つ分ずつしか置いていないと仮定するとき。）　たとえば前の命令が汎用レジスタ GR3 に書き込んでいるときに、次の命令が異なる汎用レジスタ GR5 を読み出す時、パイプラインで前の命令の書き込み動作と後ろの命令の読み出し動作が同時に起こる（絵を描いてみよ）ような場合、レジスタの読み書きの回路が 1 つしか無ければ、後ろの命令の読み出しステージは待たなければならない。

ｄ．教 10.2.2 の遅延分岐について整理しよう。　空欄を埋めよ。
遅延分岐は、①（　　）ハザードを軽減するために用いられる。具体的には、教科書図 10.6 の上段の（ａ）変更前では、条件分岐の A< 0 の判定結果が決まるより前に次の命令をフェッチ②（　　）。そこで下段の（ｂ）では、上段では A<0 の判定より前にあったロード命令 B←C を A<0 の判定より③（　　）に移した。ロード命令 B←C は、A<0 の判定には影響④（　　）ので、A<0 の判定の前でも後ろでも構わない。その結果、下段の（ｂ）ではどちらに分岐したとしても同じロード命令 B←C を実行することになるので、この口ード命令のフェッチは、条件判定の結果が出るのを待つ必要が無い。
（注：但し、（ｂ）では B←C を下へ移したため、その前の命令 A←A-1 の結果確定と条件判定 A<0 との間が短く、条件判定が A←A-1 の結果確定を待たなければならなくなる可能性がある。）

ｅ．教 10.2.2 の分岐予測について整理しよう。　空欄を埋めよ。
分岐予測は文字通り、分岐先を①（　　）しようとするもので、たとえばループの場合に脱出の条件判定はほとんどの場合は脱出②（する / しない）側にジャンプし、最後の回だけ脱出③（する / しない）側にジャンプする。もしループを１００回繰り返すとすれば、脱出する側に飛ぶ割合は④（　　）であり、脱出しない側の割合は⑤（　　）である。だから、脱出⑥（する / しない）側を先行フェッチしておけば、その「賭け」が当たる確率は高く、「大抵の場合に当たる」ようにすることができる。これが分岐予測の 1 例である。
実際は、教科書にあるように BTB や BHT によって今までの履歴（今までどちらにより多く分岐したか）を覚えておいて、それで賭けをすることが行われている。

［３］　高速化技術
下記の文中の空欄を埋めよ。
ａ．　スーパーパイプラインとは、従来のパイプラインの（　　①　　）を大きくしたものである。単純に大きくすると 1 命令あたりの実行時間が伸びてしまうが、その代わりに（　　②　　）を高める、つまり（　　③　　）を小さくすることにより、 1 命令あたりの時間を同等にした上で、並列性を高めて高速化している。
ｂ．　スーパースカラは、パイプラインを（　①　）用意して同時並列に動作させる。教 p.103 の図 10.10 の例では、（　②　）個のパイプラインで、命令（　③　）と（　④　）を同時にフェッチし、次のタイムスロットで命令（　⑤　）と（　⑥　）を同時にフェッチする。パイプラインの数を増やせば並列度が上がり、処理性能も（　⑦　）が、実際には並列処理できる命令の組合せ（個数）には限界があるため、並列度が使いきれないことが多くなる。
ｃ．　ＶＬＩＷは、命令の（　　①　　）を長くして、1 つの命令語の中に複数の命令（教 p.103 の図 10.11 ではスロット）を置く。これらのスロットは（　　②　　）に実行される。図 10.11 では、それぞれのスロットのできること（機能）は予め決まっており、たとえばスロット 1 は浮動小数加減算のみ、スロット 3 は整数演算のみ、のように、独立した回路に接続され、同時に実行される。これらのスロットにどのように命令を与えるかを（　　③　　）と

呼び、(　　④　　)が決める。(　　⑤　　)がいかにうまくスロットを埋めるか、つまりいかにうまく同時にそれぞれの機能を動かすかによって、実行の性能が変わる。

ｄ．　ベクトルコンピュータは、主に科学技術計算に多い(　　①　　)演算を並列高速に行う。具体的にはベクトルの(　　②　　)ごとに同じ計算をする場合、比較的単純な制御で高速に計算することができる。ＣＰＵ内にベクトルデータを保持する(　　③　　)を用意することによって、主記憶からベクトルデータをまとめて読み書きする、パイプライン式にベクトルデータを計算する、といった高速化が可能になった。

ｅ．　マルチプロセッサは、今まで見た１つのＣＰＵ内での並列化ではなく、(　　①　　)のＣＰＵを用いて並列計算を行う仕組である。ここで用いるＣＰＵは、パイプラインやＶＬＩＷなどの並列化をしたものを使って構わない。マルチプロセッサには、メインメモリ(主記憶)を共有する(　　②　　)システムと、共有しない(　　③　　)システムに大別される。

(②)は主記憶を共有するので、ＣＰＵ１が主記憶に書込んだ変数を、ＣＰＵ２が読み出すことが可能である。それに対して(③)は主記憶を共有しないので、ＣＰＵ１の計算結果をＣＰＵ２が知るためには転送(通信)を行う必要がある。一般に、(②)の主記憶は複数のＣＰＵによって共有されるのであるから、アクセスの頻度は単一ＣＰＵの場合に比較して(　　④　　)なり、Ｎ台のＣＰＵを結合するとアクセス頻度は(　　⑤　　)になると考えられる。であるから、Ｎが非常に大きいシステムでは主記憶が非常に(　　⑥　　)でなくてはならず、どうしてもＮの値には上限が出てくる(一般には数十と考えられている)。　それに対して(③)はその問題がないので、規模(Ｎ)の大きい並列システムを作ることができる。しかし一方、(②)は既存のプログラムを改良する程度で並列性を利用できるのに対し、(③)はプログラムの構造の改変を必要とする場合も多い。

≪解答≫

[１]

ａ　１つの命令の実行が３つの段階(ステージ)に分かれていることを示す。ステージは順番に実行される(同時に実行されることはない)。この例では、Ｆは命令読出し(フェッチ)、Ｄは命令解読(デコード)、Ｅは命令実行(エクゼキュート)を示している。

ｂ　直列実行にかかる時間＝(箱(ステージ)にかかる時間Ｔ)×(命令当りステージ数３)×(命令数４)

＝１２×Ｔ

ｃ　パイプライン実行の場合、Ｄ１とＦ２は同時に実行できる。なぜなら、命令１がＤステージ(Ｄ１)にあるとき、すでにＦステージ(Ｆ１)は完了しており、命令フェッチのためのハードウェアは空いている。だから次の命令２の命令フェッチＦ２を開始することができるのである。そのためＤ１とＦ２は同時に実行できる。

ｄ　パイプライン実行では、図の上で数えると、６×Ｔの時間で終わっている。

ｅ　直列実行のときは、Ｄ１とＦ２は別々の時間帯に実行しているので２Ｔの時間がかかるが、パイプライン実行のときは、同時に平行して実行しているのでＴの時間で済んでいる。同様にＥ１とＤ２とＦ３は同時に実行している。このように、同時並列実行されることにより、時間が短縮されている。

ｆ　図の場合の加速率Ａは、

Ａ　＝　直列実行にかかる時間 / パイプライン実行にかかる時間　＝　１２Ｔ / ６Ｔ　＝　２倍　である。

ｇ　パイプライン実行にかかる時間の一般式は、次のように考える。図を見るとすべての命令のＦステージは別々の時間に実行しなければならないので、１ステージ分の時間Ｔ×命令の個数Ｎだけかかる。これだと、最初の命令は先頭から数えているが、最後の命令はＦステージだけで、その後ＤステージとＥステージが終わっていない。この分は１命令当りのステージ数Ｓより１つ少ない(先頭の１ステージは既に数えてある)ので、(Ｓ－１)×Ｔ。この分を加えると、

Ｐ　＝　(Ｎ　×　Ｔ)　＋　((Ｓ－１)　×　Ｔ)

となる。念のため図の例の値を代入して、検算してみよう。Ｎ＝４、Ｓ＝３とすると、

Ｐ　＝　４Ｔ　＋　(３－１)Ｔ　＝　６Ｔ

となる。

h　直列実行にかかる時間Rは単純で、

　　R　＝　N　×　S　×　T

図の例の値を代入して検算すると、R　＝　4×3×T　＝　１２T　となる。

i　加速率Aの一般式は、

　　A　＝　直列実行にかかる時間 / パイプライン実行にかかる時間

　　＝　NST / （NT + （S - 1）T）

　　＝　NS / （N + S - 1）

図の例の値を代入して検算してみると、A　＝　（3×4）/（3 + 4 - 1）　＝　１２/ 6　＝　2

j　上の式で、Nを無限大にする。このままではわかりにくいので、

　　A　＝　NS / （N + S - 1）

　　＝　S　/　（1 +（（S - 1）/ N））　　分母･ 分子をいずれもNで割った

もしSが定数とすると（つまりN→∞に応じて増えることがないとすると）、　（S - 1）/ N　→　0　だから

　　A　＝　S　/　（1 + 0）　＝　S

つまり、ステージの個数倍だけ早くなる。3 ステージなら 3 倍、4 ステージなら 4 倍のようになる。 これは、上の図で並列に実行される数が、E1, D2, F3 や E2, D3, F4 のように 3 つ（ =ステージの数）であり、N→∞　とするとその状態がずっと続くことになるので、S倍早くなるのである。

[2]

a　①パイプラインの流れ　　②パイプラインの効率

b　①構造ハザード　　メモリやレジスタなどの機能（ハードウェア資源）を同時にアクセスしようとした際に発生するハザード

　　②データハザード　　前の処理で計算されるデータを後の処理が使う場合に（前の処理が間に合わない場合に）発生するハザード

　　③制御ハザード　　分岐命令で、次に実行する命令が決まらない場合に（決まるのを待たなければならないという）ハザード

c　①　制御ハザード　　②　データハザード　　③　構造ハザード

d　①　制御ハザード　　②　できない　　③　後ろ　　④　無い

e　①　予測　　②　しない　　③　する　　④　1回　　⑤　９９回　　⑥　しない

[3]

a　①　ステージの数　　②　動作周波数　　③　ステージ当りの時間

b　①　複数　　②　2　　③　1　　④　2　　⑤　3　　⑥　4　　⑦　上がる

c　①　語長（長さ）　　②　並列（同時並行）　　③　スケジューリング　　④　コンパイラ　　⑤　コンパイラ

d　①　ベクトル　　②　要素　　③　ベクトルレジスタ

e　①　複数個　　②　密結合　　③　疎結合　　④　大きく　　⑤　N倍　　⑥　高速

≪基本情報処理技術者試験問題から類似･ 関連問題≫

１）　シングルチップマイコンの特徴として，最も適切なものはどれか。　（基本２１春午前　問１０）

①　PC のメイン CPU に適している。

②　ROM は内蔵されているが，RAM は内蔵されていない。

③　高速処理システム又は大規模なシステムに適している。

④　入出力機能が内蔵されている。

２） プロセッサにおけるパイプライン処理方式を説明したものはどれか。 （基本２１春午前 問１１、基本１８春午前 問１７）

① 単一の命令を基に，複数のデータに対して複数のプロセッサが同期をとりながら並列にそれぞれのデータを処理する方式

② 一つのプロセッサにおいて，単一の命令に対する実行時間をできるだけ短くする方式

③ 一つのプロセッサにおいて，複数の命令を少しずつ段階をずらしながら同時実行する方式

④ 複数のプロセッサが，それぞれ独自の命令を基に複数のデータを処理する方式

３） スーパスカラの説明はどれか。 （基本２０春午前 問１７）

① 処理すべきベクトルの長さがベクトルレジスタより長い場合，ベクトルレジスタの長さの組に分割して処理を繰り返す方式である。

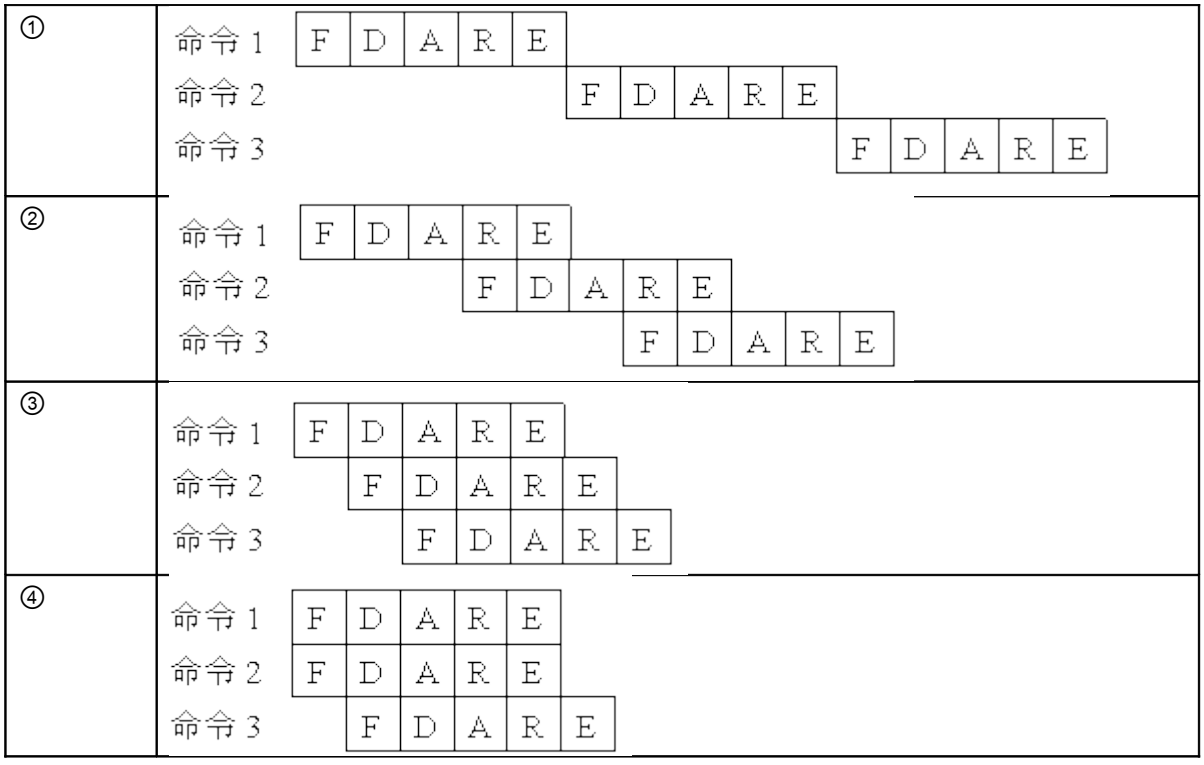
② パイプラインを更に細分化することによって，高速化を図る方式である。

③ 複数のパイプラインを用いて，同時に複数の命令を実行可能にすることによって，高速化を図る方式である。

④ 命令語を長く取り，一つの命令で複数の機能ユニットを同時に制御することによって，高速化を図る方式である。

４） 次の図のうち，パイプライン制御の説明として適切なものはどれか。ここで，図中の各記号の意味は次のとおりである。 （基本１８秋午前 問１８）

F：命令呼出し， D：解読， A：アドレス計算， R：オペランド呼出し， E：実行



≪解答≫

１） ④ シングルチップマイコンは、１つのＩＣチップの上に CPU のほかにメモリ（ROM、RAM）、周辺機器への接続機能、A/D・D/A 変換器などを搭載したもので、基本的に１つのチップで一定の機能を果たすように考えている。用途としては、家電製品や自動車のエンジン制御、携帯電話、ゲーム機などであり、限定された用途でかつそれほど大規模なデータ・プログラム・計算処理などをさせることのない場合に、安価なパーツとして搭載する。特定用途で大量に使われる時に、必要な機能だけを組み込んだチップを設計して用いる、というパターンが多いだ

ろう。
①はまちがいで、汎用の PC のメイン CPU には使われない。汎用 PC は大きなデータ･プログラム･計算処理を期待されるし、用途も予め特定できない。また周辺機器も様々である。 ②はまちがいで、RAM も少量ながら搭載される。大きな RAM を搭載することは普通は行われない。 ③はまちがいで、高速処理や大規模システムには使われない。

２） ③ ①はベクトルプロセッサ ②は特になし ④はマルチプロセッサ

３） ③ ①はベクトルプロセッサの説明 ②はスーパーパイプラインの説明 ④はＶＬＩＷの説明

４） ③ ①は直列実行 ②はなし ④はスーパースカラ (パイプラインが 2 本ある)