第2章　中央処理装置

学習目標

中央処理装置を構成する論理素子とその演算を理解させ、その後物理的に実現されている中央処理装置の構成を理解させる。また、中央処理装置で実行できる命令の形式とアドレッシングモード理解させ、実行の仕組みを理解させる。

内　　容

内容のあらまし

内　　容	説　　明	議　　論
論理素子と論理演算	・論理素子を用いた論理回路と、それと等価の論理式の対応を説明する ・論理式の基本演算を説明し、真理値表を作成できるようにする	
ＣＰＵの基本構造	・中央処理装置（ＣＰＵ）を構成する各レジスタの名称、役割について説明する。 ・各レジスタがどのように動きながらコンピュータの命令が実行されるかを説明する	
命令形式とアドレッシングモード	・コンピュータの機械語命令がどのような形式になっているかを説明する ・各種のアドレスの指定方式を説明し実際の主記憶装置内でどこを指すかを説明する	
命令の種類	・コンピュータが実行する命令の種類・機能を説明し、各々の命令が実行されると何がどのように変化するのかを説明する	

内　　容	説　　明	議　　論
ＣＯＭＥＴの例	・命令表記の例、オペランド表記の例、命令の種類の例としてＣＯＭＥＴが提供する命令の形式と、その機能を示す	
プログラム実行の仕組み	・演算処理装置が、命令を読み込んでから実行する動作の流れを説明する ・プログラムの実行中に発生する割り込みについてその種類、役割を説明し割込みの発生から、終了までのメカニズムを説明する	

２．１　論理素子と論理演算

論理素子を用いた論理回路と、それと等価の論理式の対応を理解させ、基本論理の組合せ回路及び論理式の基本演算を理解させ、真理値表を作成できるようにする。

２．１．１　論理和（ＯＲ）

入力Ａ，Ｂのうち、少なくとも一方が１のとき、出力Ｚが１となる演算を論理和といい、このような演算を実現する回路をＯＲ回路という（図２．１－１（ｂ）参照）。

論理和は、論理式の表現では

$$Z = A + B$$

または

$$Z = A \cup B$$

と表す。

論理変数Ａ，Ｂと、論理関数Ｚの関係を示した真理値は、図２．１－１（ａ）と表され、論理和の回路記号は、図２．１－１（ｂ）のように表す。

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

図２．１－１（ａ）　論理和　真理値表

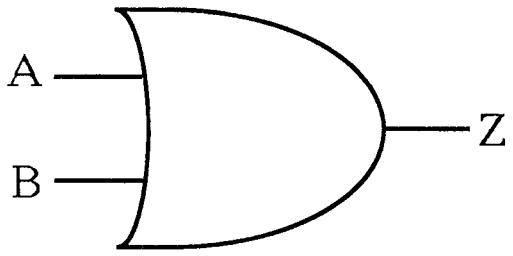


図２．１－１（ｂ）　論理和　回路記号

２．１．２　論理積（ＡＮＤ）

入力Ａ，Ｂが、ともに１のとき、出力Ｚが１となる演算を論理積といい、このような演算を実現する回路をＡＮＤ回路という（図２．１－２（ｂ）参照）。

論理積は、論理式の表現では

$$Z = A \cdot B$$

または

$$Z = A \cap B$$

と表す。

論理変数Ａ，Ｂと、論理関数Ｚの関係を示した真理値は、図２．１－２（ａ）と表され、論理積の回路記号は、図２．１－２（ｂ）のように表す。

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

図２．１－２（ａ）　論理積　真理値表

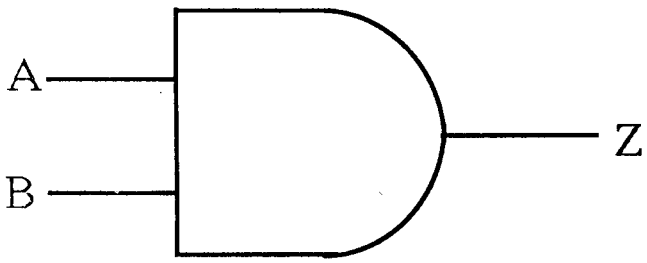


図２．１－２（ｂ）　論理積　回路記号

２．１．３　論理否定（ＮＯＴ）

入力Ａが、０のとき出力Ｚが１、入力Ａが、１のとき出力Ｚが０となる演算を論理否定といい、このような演算を実現する回路をＮＯＴ回路という。
（図２．１－３（ｂ）参照）

論理否定は、論理式の表現では

$$Z = \overline{A}$$

と表す。

論理変数Ａと、論理関数Ｚの関係を示した真理値は、図２．１－３（ａ）と表され、論理否定の回路記号は、図２．１－３（ｂ）のように表す。

A	Z
0	1
1	0

図２．１－３（ａ）　論理否定　真理値表

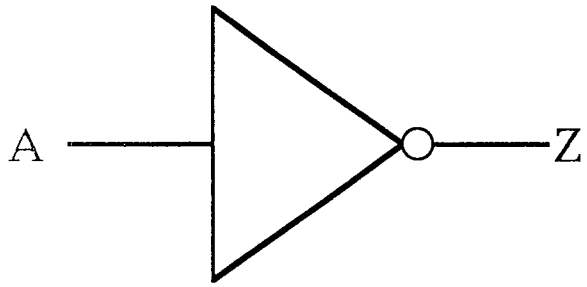


図２．１－３（ｂ）　論理否定　回路記号

２．１．４　論理和否定（ＮＯＲ）

入力Ａ，Ｂの論理和を否定し、少なくともいずれか一方が１のとき、出力Ｚが０となる演算を論理和否定といい、このような演算を実現する回路をＮＯＲ回路という（図２．１－４（ｂ）参照）。

論理和否定は、論理式の表現では

$$Z = \overline{A + B}$$

または

$$Z = \overline{A \cup B}$$

と表す。

論理変数Ａ，Ｂと、論理関数Ｚの関係を示した真理値は、図２．１－４（ａ）と表され、論理和否定の回路記号は、図２．１－４（ｂ）のように表す。

A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

図２．１－４（ａ）　論理和否定　真理値表

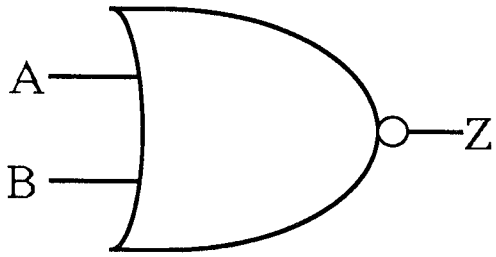


図２．１－４（ｂ）　論理和否定　回路記号

２．１．５　論理積否定（ＮＡＮＤ）

入力Ａ，Ｂの論理積を否定し、少なくとも一方が0のとき、出力Ｚが１となる演算を論理積否定といい、このような演算を実現する回路をＮＡＮＤ回路という（図２．１－５（ｂ）参照）。

論理積否定は、論理式の表現では

$$Z = \overline{A \cdot B}$$

または

$$Z = \overline{A \cap B}$$

と表す。

論理変数Ａ，Ｂと、論理関数Ｚの関係を示した真理値は、図２．１－５（ａ）と表され、論理積否定の回路記号は、図２．１－５（ｂ）のように表す。

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

図２．１－５（ａ）　論理積否定　真理値表

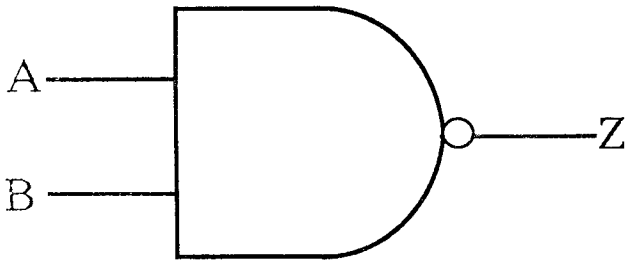


図２．１－５（ｂ）　論理積否定　回路記号

２．１．６　排他的論理和（ＥＯＲ）

入力Ａ，Ｂが、同じ値ではないとき、出力Ｚが１となり、同じ値のとき、出力Ｚが０となる演算を排他的論理和といい、このような演算を実現する回路をＥｘｃｌｕｓｉｖｅＯＲ回路という（図２．１－６（ｂ）参照）。

排他的論理和は、論理式の表現では

$A \neq B$ のとき $Z = 1$

$A = B$ のとき $Z = 0$

$Z = A \cdot \overline{B} + \overline{A} \cdot B$

または

$Z = (A \cap \overline{B}) \cup (\overline{A} \cap B)$

と表す。

論理変数Ａ，Ｂと、論理関数Ｚの関係を示した真理値は、図２．１－６（ａ）と表され、排他的論理和の回路記号は、図２．１－６（ｂ）のように表す。

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

図２．１－６（ａ）　排他的論理和　真理値表

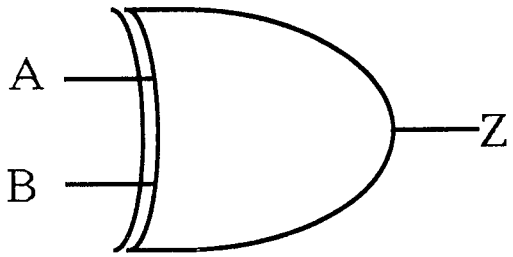


図２．１－６（ｂ）　排他的論理和　回路記号

２．１．７　組み合わせ回路の例

今までの基本回路組み合わせて、２進数の２入力加算回路を考える。この回路は、変数A，Bという２つの２進数の和をとり、結果を$Z_1 Z_0$に出力する。ここで、Z_0は和であり、Z_1は桁上がりである。

いま、A，B，Z_1，Z_0の真理値表は、図２．１－７（ａ）のように表す。

また、Z_1，Z_0の論理式の表現は、真理値表を参考にすると

$$Z_1 = A \cdot B$$

$$Z_0 = A \cdot \overline{B} + \overline{A} \cdot B$$

となる。　これらの情報をもとに、回路を作成すると図２．１－７（ｂ）となる。

A	B	Z_1	Z_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

図２．１－７（ａ）　２入力加算回路の真理値表

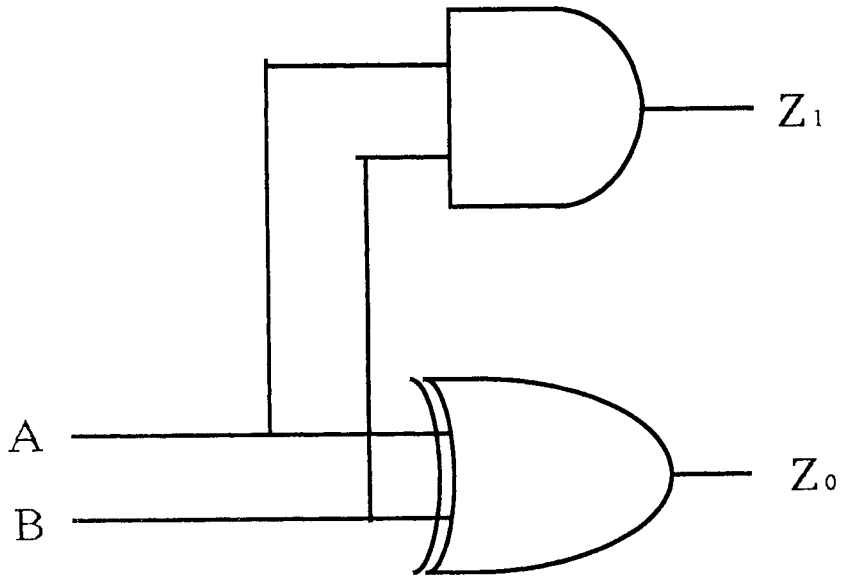


図２．１－７（ｂ）　２入力加算回路の回路図

２．１．８　フリップフロップ

フリップフロップ（flip flop）は、１ビットのデータを記憶できる電子回路で、２個のＮＯＲゲートを相互に繋ぐ事で実現されている（図２．１－８参照）。

この回路では、

① Ｒを１にするとＱが０になる（回路がリセットされる。リセット後はＲを０にする）。
② Ｓを１にするとＱが１になる（Ｒは０のまま）。
③ ②の後はＳが０になってもＱは１のままである（Ｒは０のまま）。

という動作をするので、Ｒを１にする（リセットをかける）までは、１個の数字が記憶されることになる。ＩＣメモリは、このフリップフロップが数多く並んだものである。

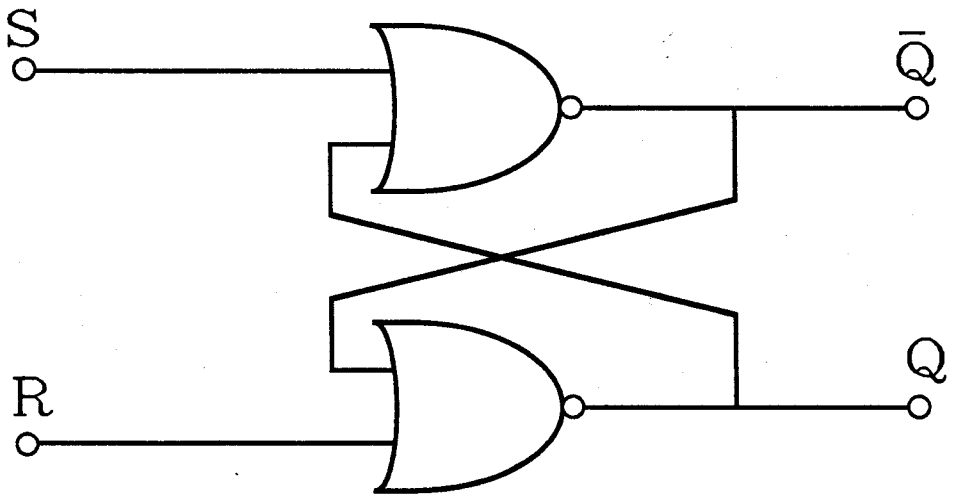


図２．１－８　フリップフロップ回路

指導上の留意点

ポイント

① 論理素子を用いた論理回路の入力と出力の関係を理解させる。

② 論理回路と等価な論理式を作成できるようにさせる。また、逆に論理式と等価な論理回路を作成できるようにさせる。

③ 論理式と、それに対応する真理値表が作成できるようにする。

用語

論理素子　論理回路　真理値　論理和（ＯＲ）　論理積（ＡＮＤ）　論理否定（ＮＯＴ）
論理和否定（ＮＯＲ）　論理積否定（ＮＡＮＤ）　排他的論理和（ＥＯＲ）

講師ノート

第２種情報処理技術者試験

・論理素子と論理回路については頻繁に出題される問題ではないが、得点のしやすい問題なので、各論理演算の意味を確実に理解し真理値表を書けるようにしておく事が重要である。

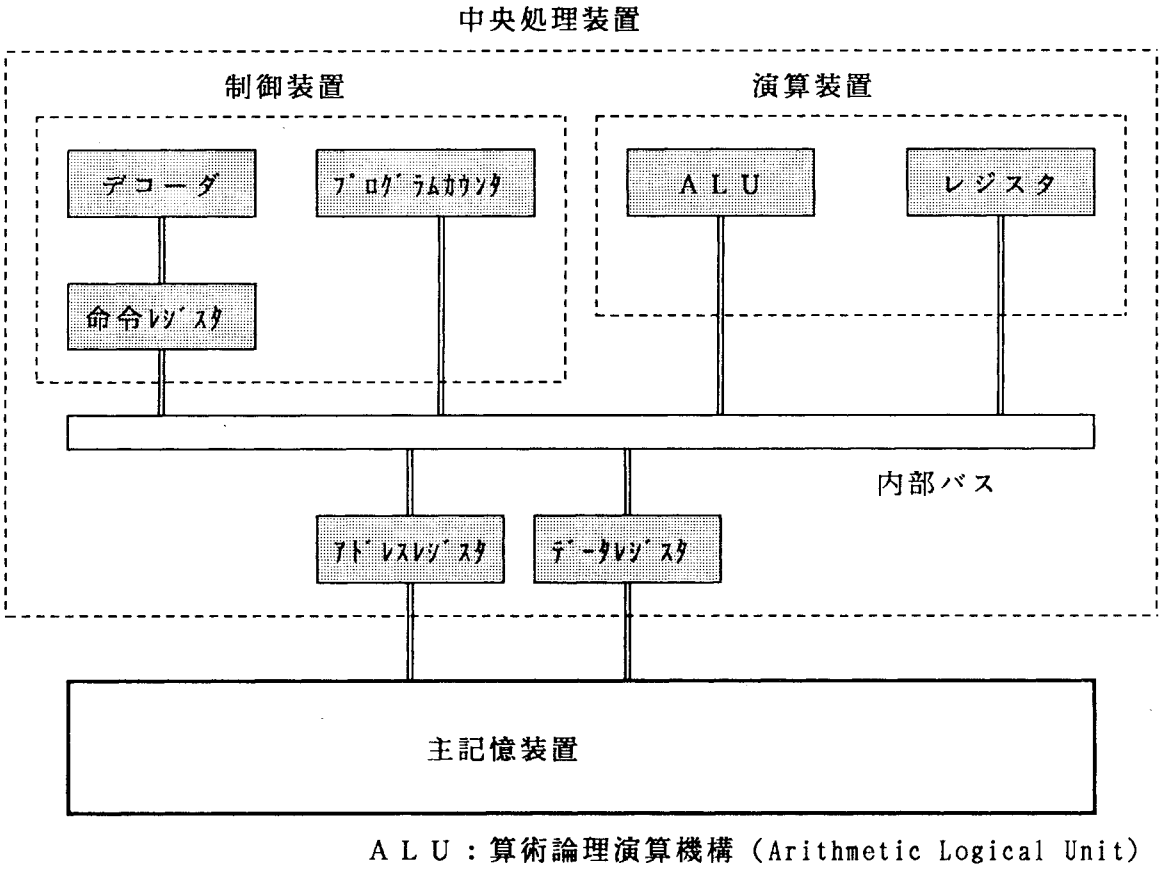
２．２　ＣＰＵの基本構造

中央処理装置（ＣＰＵ）を構成する各レジスタの名称、役割について理解させ、各レジスタがどのように動きながらコンピュータの命令が実行されるかを理解させる。

２．２．１　中央処理装置のハードウェア構成

中央処理装置は、制御装置、演算装置から構成され、互いに内部バスを介して接続されている。

中央処理装置のハードウェア構成を図２．２－１に示す。



ＡＬＵ：算術論理演算機構（Arithmetic Logical Unit）

図２．２－１　中央処理装置のハードウェア構成

２．２．２　プログラムカウンタ（program counter）

次に実行する命令の主記憶装置上のアドレスを指す。従って、通常の１命令を実行した後は、実行した命令のバイト長が加算され、次に実行する命令の先頭アドレスが設定される。
分岐（ジャンプ）命令の時は、分岐先のアドレスが入る。

２．２．３　命令レジスタ（instruction register）

主記憶装置から取り出した命令語を格納する。命令語は一般に命令の内容を示す部分（例えば、ロードや算術加算など）と、アドレスを指定する部分（例えばデータレジスタや主記憶装置の番地など）とに分けられ、命令の内容を指定する部分はデコーダへ送られ、アドレスを示す部分はアドレスレジスタへ送られる。

２．２．４　デコーダ（decoder）

解読器とも呼ばれ、命令レジスタから転送された命令の内容を解読し、命令に対応する回路に制御信号を出力する。

２．２．５　算術論理演算装置（ＡＬＵ：Arithmetic Logical Unit）

固定小数点算術演算、論理演算、シフトなどの演算を行う。計算機の規模に応じて乗除算装置や浮動小数点演算装置を持つものもある。演算の結果はアキュムレータ（accumulator）に格納する。

２．２．６　アキュムレータ（accumulator）

累算器とも呼ばれ、被演算数や演算結果を格納する。演算を行う際の中心となるレジスタである。

２．２．７　アドレスレジスタ（address register）

データの入出力を行う主記憶装置のアドレスを示す。

２．２．８　データレジスタ（data register）

主記憶装置へのデータの書き込み、読み込みはデータレジスタを介して行われる。

２．２．９　インデックスレジスタ（index register）

インデックス修飾アドレスのアドレッシングに使用する。このレジスタの内容と命令のアドレス部の加算により、アドレスの修飾を行う。連続する記憶領域に対して同一の処理を行いたい場合、繰り返しのたびごとにこのレジスタに増分を加える。

２．２．１０　ベースレジスタ（base register）

プログラムの再配置を行うとき、プログラムの先頭アドレスをこのレジスタに入れて使う。

２．２．１１　スタックポインタ（stack pointer）

スタックメモリとして使用する先頭のアドレスを入れて使うレジスタである。

2．2．12　汎用レジスタ（general register）

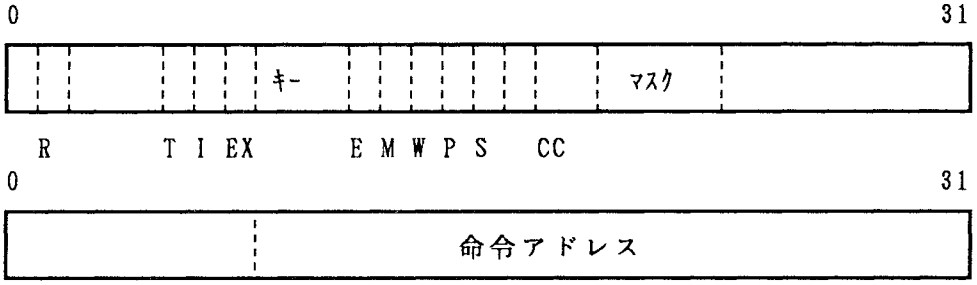
演算対象のデータの指定、アドレスの指定など自由に使用できるレジスタである。アキュムレータ、アドレスレジスタ、データレジスタ、インデックスレジスタ、ベースレジスタ、スタックポインタなどに使用できる汎用化されたレジスタである。

2．2．13　プログラム状況語（ＰＳＷ：Program Status Word）

プログラムの実行に必要な情報とプログラムの状況を示す情報を集約して保持するレジスタである。

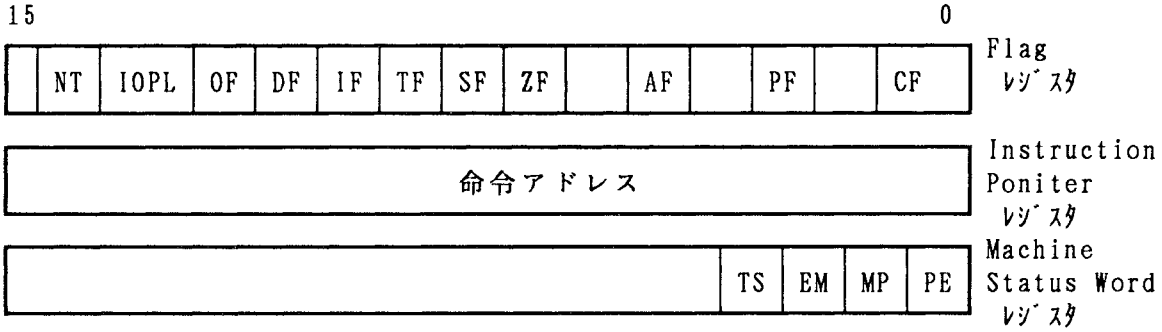
演算結果の状況を示す条件フラグ、割り込みを制御するためのマスク、実行モードを示すフラグなどを持つ。

例）　汎用コンピュータのプログラム状況語（ＩＢＭ）



R　プログラム事象記録に関連した割り込みを制御（許可）する。
T　動的アドレス変換を制御する。
I　ＣＰＵが入出力割り込みの実行を制御する。
EX　ＣＰＵが外部割り込みの実行を制御する。
E　ＣＰＵの制御モード（基本制御、拡張制御）を示す。
M　マシン・チェック割り込みを制御する。
W　ＣＰＵが待ち状態を示す。
P　ＣＰＵの状態（ユーザモード、スーパーバイザモード）を示す。
S　アドレス空間モードを制御する。
CC　命令実行で得られた結果の条件コードを示す。
ﾏｽｸ　プログラム例外（固定小数点桁あふれ等）の割り込みを制御する。
命令アドレス　次に実行される命令の先頭アドレスを示す。

例）　マイクロプロセッサのプログラム状況語（Intel 80286）



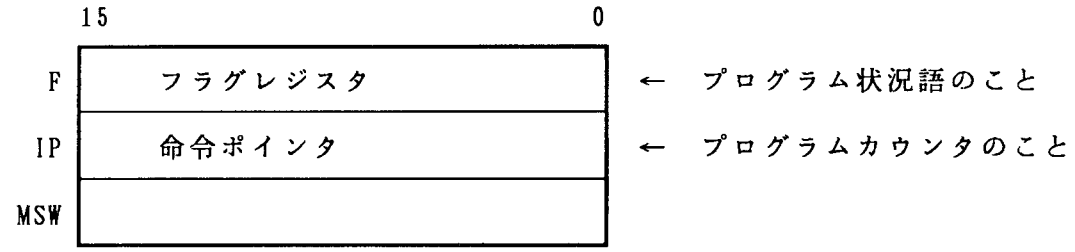
NT　ﾀｽｸのﾈｽﾃｨﾝｸﾞ状態
OF　ｵｰﾊﾞｰﾌﾛｰﾌﾗｸﾞ
IF　外部割り込みの許可
SF　ｻｲﾝﾌﾗｸﾞ
AF　補助ｷｬﾘｰﾌﾗｸﾞ
CF　ｷｬﾘｰﾌﾗｸﾞ
IOPL　I/O特権ﾚﾍﾞﾙ
DF　ｽﾄﾘﾝｸﾞ操作の方向の指定
TF　ｼﾝｸﾞﾙｽﾃｯﾌﾟ動作の指定
ZF　ｾﾞﾛﾌﾗｸﾞ
PF　ﾊﾟﾘﾃｨｰﾌﾗｸﾞ

２．２．１４　代表的なＣＰＵのレジスタ構成

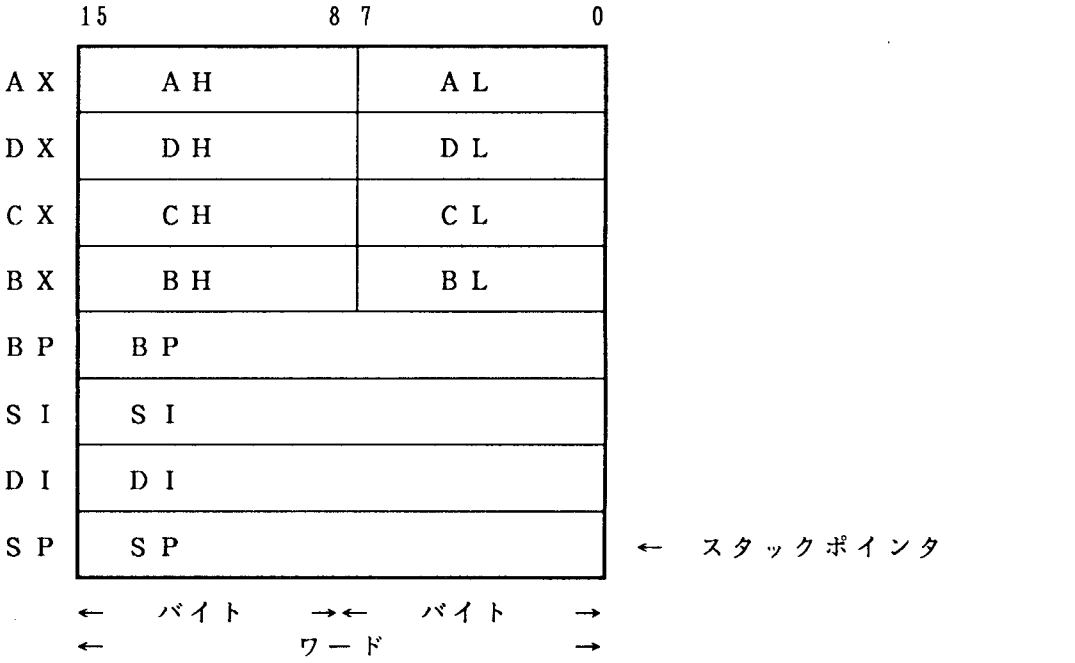
代表的なマイクロプロセッサのレジスタ構成と、汎用コンピュータのレジスタ構成を、また、情報処理技術者試験で出題されるハードウェアＣＯＭＥＴのレジスタ構成を示す。

（１）マイクロプロセッサのレジスタ構成

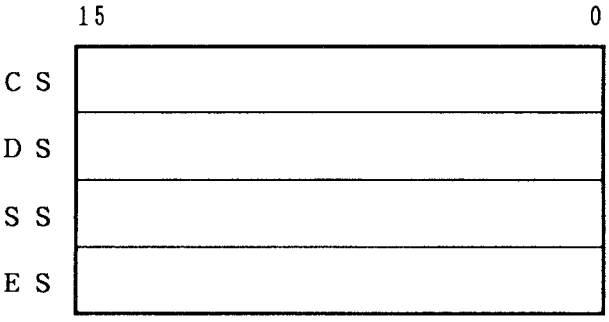
・ステータス／コントロール・レジスタ



・汎用レジスタ（１６ビット）

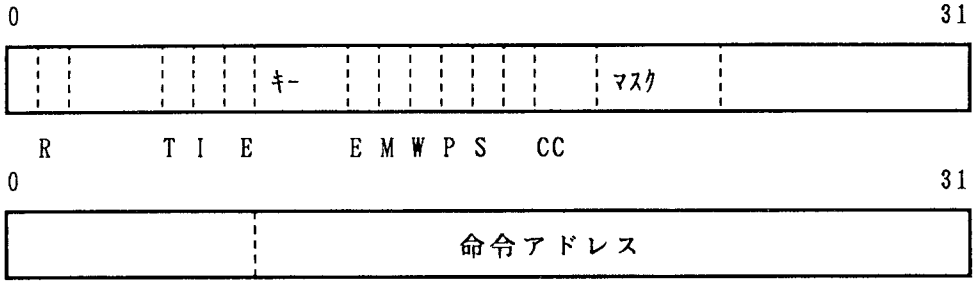


・セグメント・レジスタ

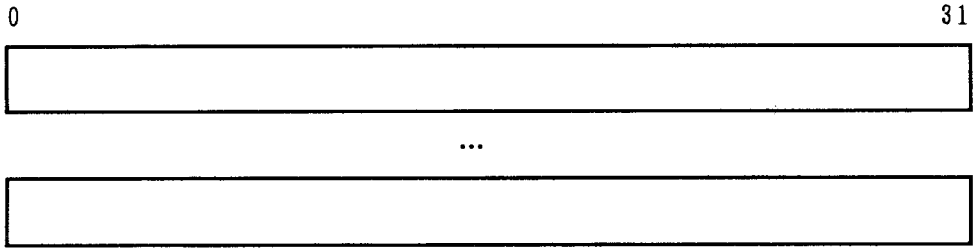


（２）汎用コンピュータのレジスタ構成

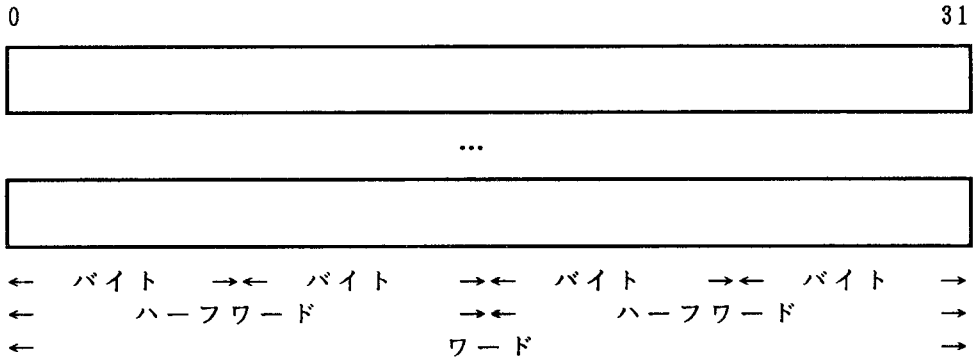
・プログラム状況ワード（ＰＳＷ）



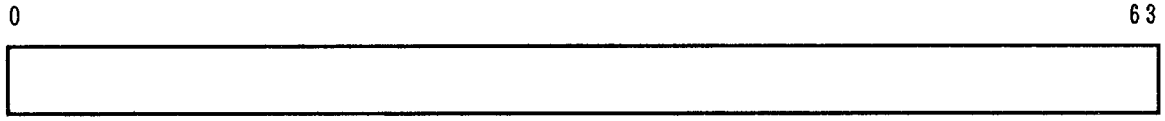
・制御レジスタ（３２ﾋﾞｯﾄ、１６個）



・汎用レジスタ（３２ﾋﾞｯﾄ、１６個）

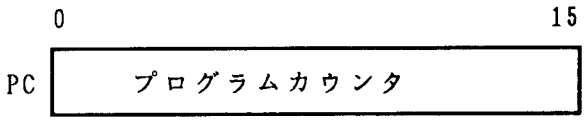


・浮動小数点レジスタ（６４ﾋﾞｯﾄ、４個）



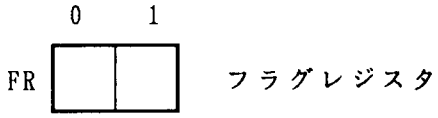
（３）ＣＯＭＥＴのレジスタ構成

①専用レジスタ



実行中の命令語の先頭アドレスを保持し、命令の実行が終わると、次に実行する命令語の先頭アドレスが設定される。一般に、命令の実行が終わるとPCに2がアドレス加算（注）され、分岐、コール、リターン命令の場合は、新たに分岐先のドレスが設定される。

（注）アドレス加算：被演算データを符号のない数値とみなし、その和を
６５５３６で割った剰余（和の下位１６ビット）を値とする。



FR（フラグレジスタ、flag register）は、ロードアドレス命令及び算術、論理、シフトの各命令の実行の結果、GRに設定されたデータが、負、零、正のいずれであるかの情報、または比較演算命令の実行により得られた、二数間の大小関係の情報を保持する。すなわち、実行結果により、FRは次の表（演算結果とFR）のとおり設定する。また、比較については、下の表（比較演算とFR）の通りに設定する。

演算結果とFRの表で、負はGRの符号ビットが1、零はGRの全ビットが0、正は符号ビットが0で、かつ零でないデータをいう。
FRの第1（左端の）ビットはGRの符号を示し、第2ビットはGRが零か否かを示す。
FRの値は、条件付き分岐命令で参照する。
その他の命令の実行によって、FRの値は変更されない。

演算結果とFR

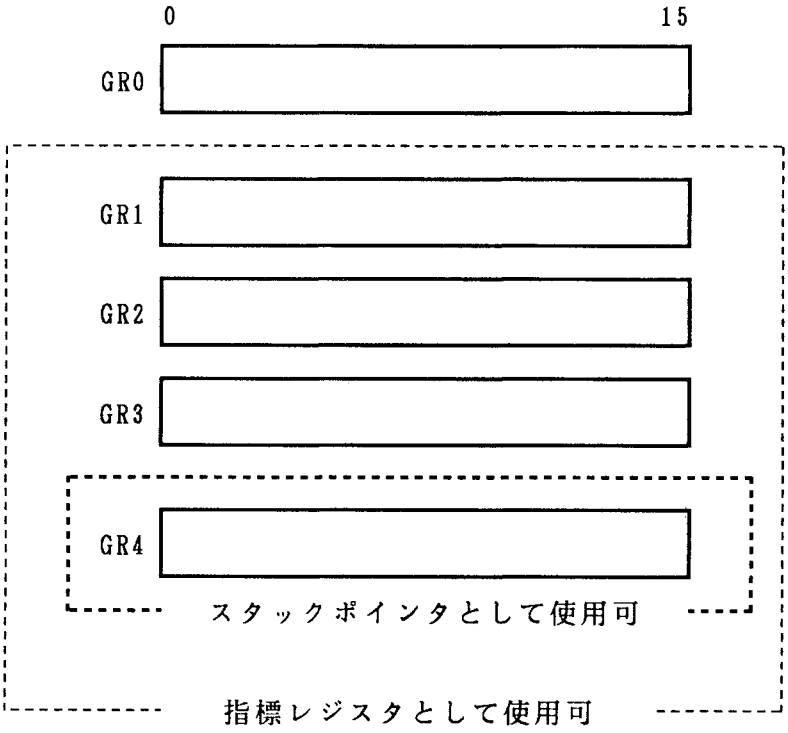
	GRに設定されたデータ		
	負	零	正
FRの値	10	01	00

比較演算とFR

(GR)と(有効アドレス)の算術比較、または論理比較を行い
その結果によりFRに次の値を設定する

比較結果	FRのビット値
(GR) ＞ (有効アドレス)	00
(GR) ＝ (有効アドレス)	01
(GR) ＜ (有効アドレス)	10

②汎用レジスタ



GR（汎用レジスタ、general register）は、5個あり、汎用レジスタGR0からGR4までとする。この5個のレジスタは、算術、論理、比較、シフト演算などに用いる。このうち、GR1からGR4までのレジスタは、指標レジスタ（index register）としても用いる。またGR4のレジスタは、さらにスタックポインタ（stack pointer）として用いる。

スタックポインタは、スタックの最上段（stack top）のアドレスを保持しているレジスタである。

指導上の留意点

ポイント

①ＣＰＵの基本的な構造を理解させる。

②各レジスタの名称、機能を理解させる。システムにより名称が異なる場合もあるので幾つか例を上げて説明するとよい。

③例を用い、実際のコンピュータの構造を理解させる。情報処理試験に出題されるＣＯＭＥＴのＣＰＵ構成については充分に理解させる。

④プログラム状況語（ＰＳＷ）の構成はメーカー、コンピュータの種類により異なる。

用語

プログラムカウンタ　命令レジスタ　デコーダ　算術論理演算装置（ＡＬＵ）
アキュムレータ　アドレスレジスタ　データレジスタ　インデックスレジスタ
ベースレジスタ　スタックポインタ　汎用レジスタ　プログラム状況語（ＰＳＷ）

講師ノート

第２種情報処理技術者試験

・中央処理装置の構成をしっかりと理解し、各レジスタがどのような機能を持つかを掴んでおく必要がある。

２．３　命令形式とアドレッシングモード

コンピュータの機械語命令がどのような形式になっているかを理解させる。また、各種のアドレスの指定方式を理解させ、実際の主記憶装置内でどこを指すかを理解させる。

２．３．１　命令コード

命令の機能を表す。

命令部　　オペランド部

命令ｺｰﾄﾞ	オペランド	オペランド

機能は次のように分類できる。詳細は「２．４命令の種類」を参照。

・データ転送命令
・算術演算命令
・論理演算命令
・比較演算命令
・シフト演算命令
・分岐命令
・入出力命令
・システム制御命令

各命令は、機械語では２進数の値で表現されるが、通常は人間に分かりやすいようにニーモニック表記が使われる。

例）ハードウェアＣＯＭＥＴのロード、ストア命令、算術演算命令、論理演算命令は、次のようなニーモニック表記で記述する。

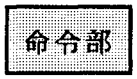
		ニーモニック表記
ﾛｰﾄﾞ	LoaD	LD
ｽﾄｱ	STore	ST
算術加算	ADD arithmetic	ADD
算術減算	SUBtract arithmetic	SUB
論理積	AND	AND
論理和	OR	OR
排他的論理和	Exclusive OR	EOR

２．３．２　オペランド

命令の操作対象を示す。

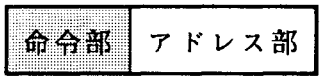
オペランドの数により分類する。

（１）０アドレス方式



スタック方式の演算で使用されるアドレス形式である。
スタック方式については、２．３．３演算実現方式を参照。

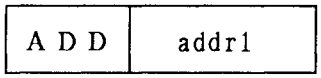
（２）１アドレス方式



（アキュムレータ方式）

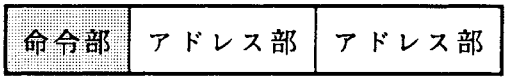
アキュムレータ（累算器）を用いる演算で使用されるアドレス形式である。
アキュムレータの内容と、アドレス部で指定されたアドレスの内容との演算を行い、結果をアキュムレータに格納する。

例）



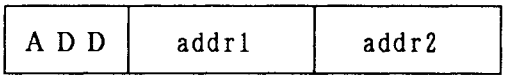
アキュムレータ　←　（アキュムレータ）＋（addr1）
アキュムレータの内容にaddr1の内容を加える。

（３）２アドレス方式



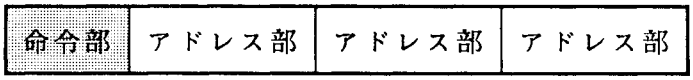
最初のアドレス部で指定されたアドレスの内容と、２番目のアドレス部で指定されたアドレスの内容を演算して、結果を最初のアドレス部で指定されたアドレスに格納する。

例）



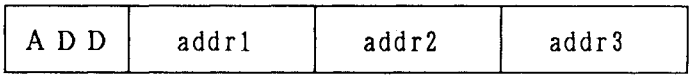
addr1　←　(addr1)　＋　(addr2)
addr1, addr2の内容を加えて、結果をaddr1に入れる。

（４）３アドレス方式



最初のアドレス部で指定されたアドレスの内容と、２番目のアドレス部で指定されたアドレスの内容を演算して、結果を３番目のアドレス部で指定されたアドレスに格納する。

例）



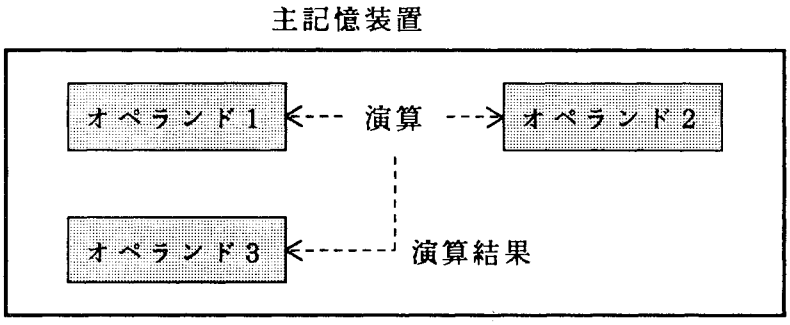
addr3　←　(addr1)　＋　(addr2)
addr1, addr2の内容を加えて、結果をaddr3に入れる。

２．３．３　演算実現方式

演算において、レジスタをどのように使用するかによりいくつかの方式がある。

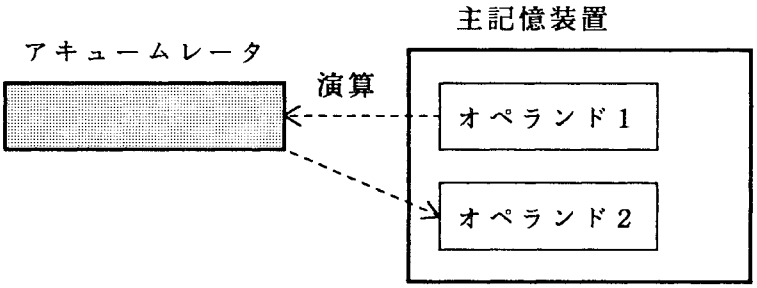
（１）メモリ間直接演算方式

２つの演算対象オペランド（オペランド１，２）と演算結果（オペランド３）の格納場所として、主記憶装置上のアドレスを直接指定する。



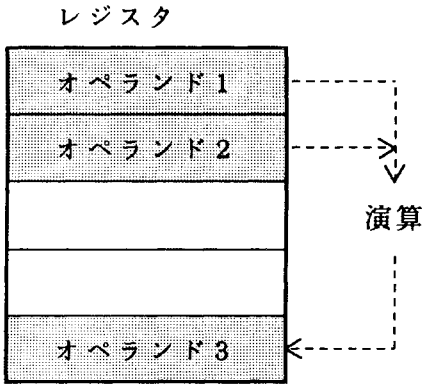
（２）アキュムレータ方式

演算対象のオペランド１をアキュムレータに固定し、演算結果の格納場所もアキュムレータに固定し、演算対象のオペランド２のメモリアドレスだけを指定する方式である。マイクロコンピュータなどの計算機で使用される方式である。



（３）多数レジスタ方式

演算対象として指定可能なレジスタを数多く用意し、命令のオペランドでどのレジスタを演算対象に使用するかを指定する。



（４）スタック方式

演算命令でオペランドを指定しない方式である。演算はハードウェアで用意してあるプッシュダウンスタックを用いて行う。演算の実行は、データが現れたならば、スタックにプッシュダウンし、演算子があらわれたならば、演算に必要な個数のデータをスタックの一番上からポップアップし、演算実行後、スタックの一番上にプッシュダウンする。この方式で演算実行をするためには数式を逆ポーランド記法に変換しておく必要がある。

例）(a-b*c)/d　をスタック方式で演算する

逆ポーランド記法に変換

(a-b*c)/d　→　a b c * - d /

① a をスタックにプッシュダウンする
push a

② b をスタックにプッシュダウンする
push b

③ c をスタックにプッシュダウンする
push c

④ 乗算演算を行う
multiply(*)
乗算（*）は、２つのデータを使用するので、b, c をスタックからポップアップし、multiply した結果をスタックの一番上にプッシュダウンする。

⑤ 減算演算を行う
subtract(-)
減算（-）は、２つのデータを使用するので、a,b*c をスタックからポップアップし、subtract した結果をスタックの一番上にプッシュダウンする。

⑥ d をスタックにプッシュダウンする
push d

⑦ 除算演算を行う
devide (/)
除算（/）は、２つのデータを使用するので、a-b*c, d をスタックからポップアップし、devide した結果をスタックの一番上にプッシュダウンする。

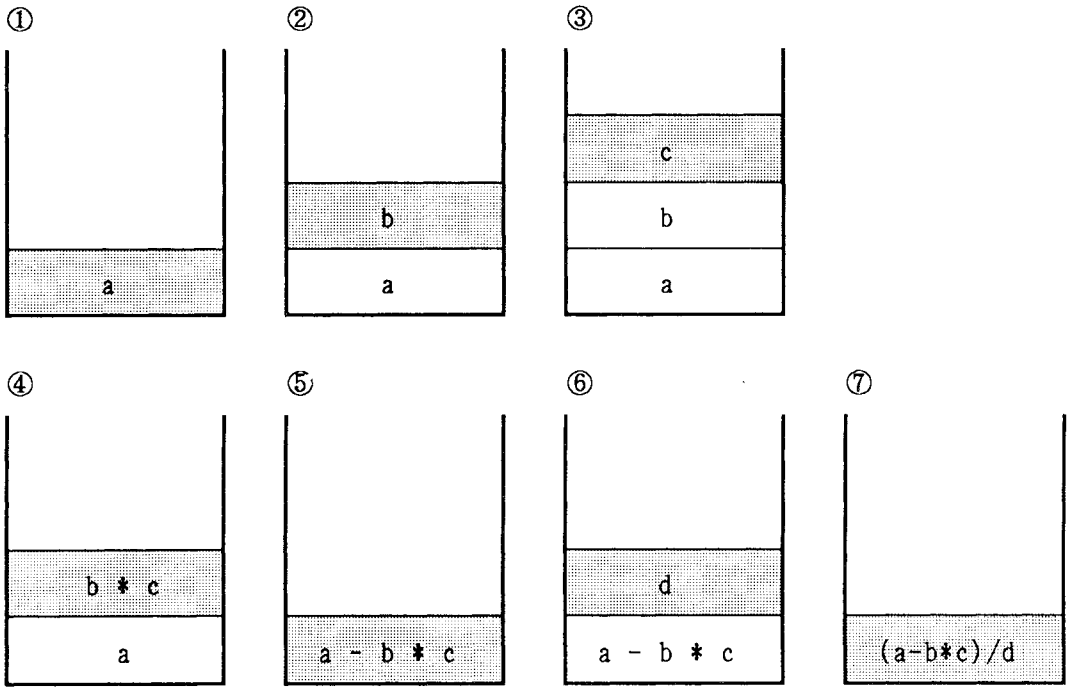


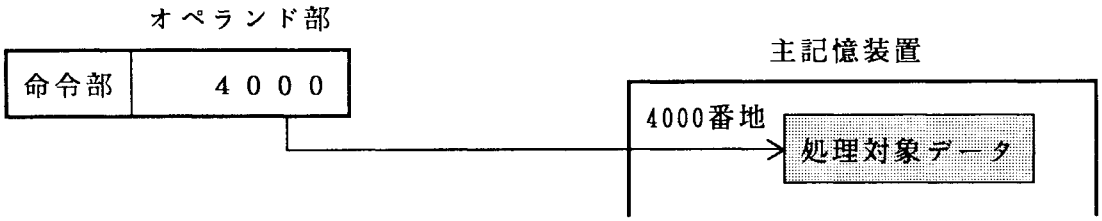
図２．３－１　スタック方式による演算

２．３．４　アドレッシングモード

オペランドのアドレス（有効アドレス）を指定する方法をアドレッシングモードと呼ぶ。

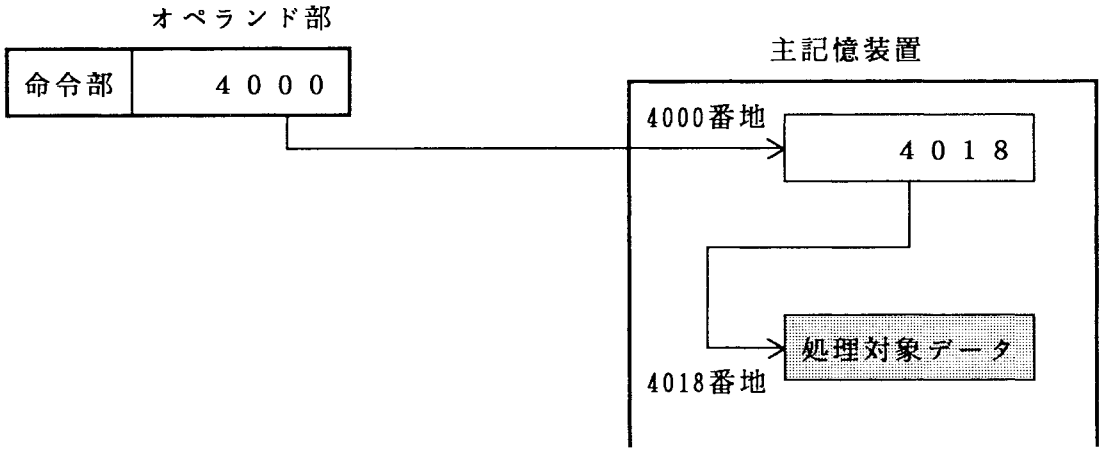
（１）直接アドレス（direct addressing）

オペランド部の値を有効アドレスとする。



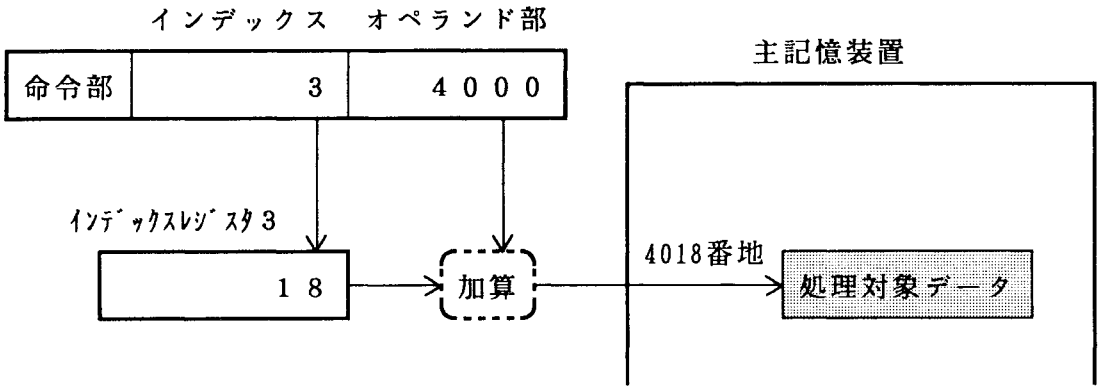
（２）間接アドレス（indirect addressing）

オペランド部の値が示すアドレスに格納されている値を有効アドレスとする。



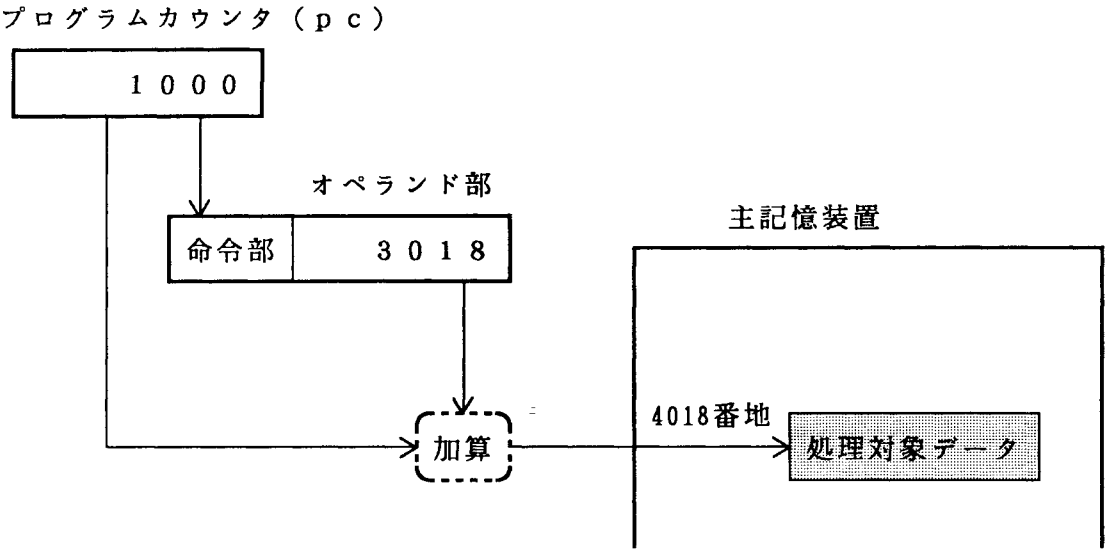
（３）インデックス修飾アドレス（indexed addressing）

オペランド部の値にインデックスレジスタの値を加算した値を有効アドレスとする。また、使用するインデックスレジスタも、オペランド部で指定する。



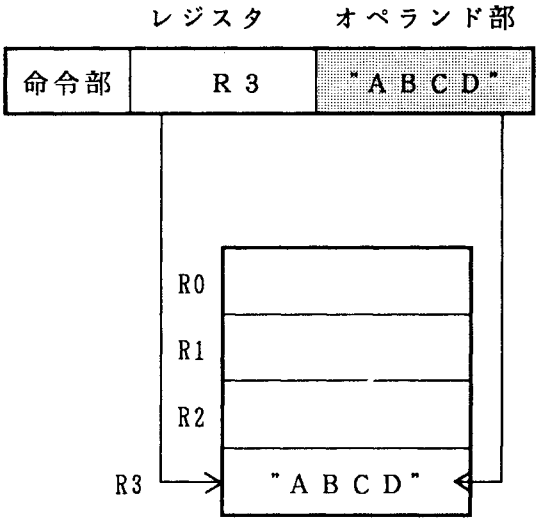
（４）相対アドレス（relative addressing）

オペランド部の値に基底アドレスの値を加算した値を有効アドレスとする。



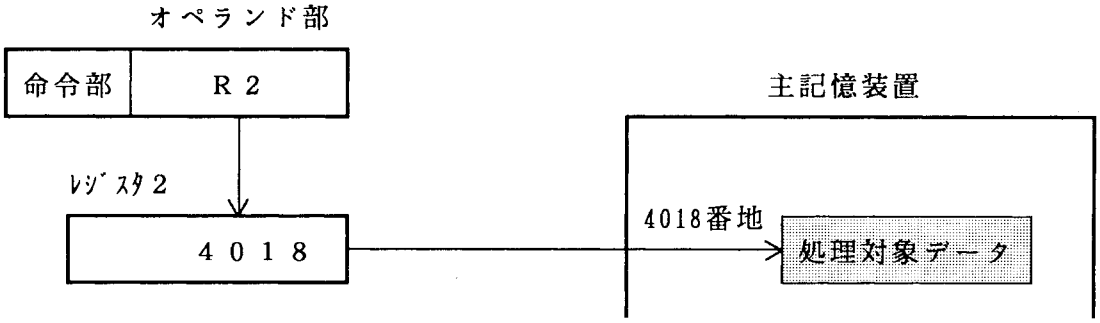
（５）即値アドレス（immediate address）

オペランド部の値がそのままデータとなる。



（６）レジスタアドレス（register address）

オペランド部のレジスタの内容を有効アドレスとする。
（単にレジスタを指す場合もある）

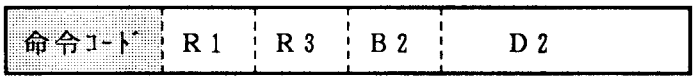


例）汎用コンピュータの命令形式

・ＲＲ形式　（レジスタとレジスタの操作）

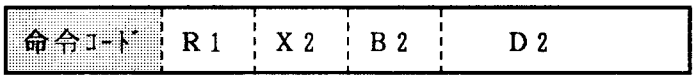
・ＲＳ形式　（レジスタと記憶域との操作）

8ﾋﾞｯﾄ　4ﾋﾞｯﾄ　4ﾋﾞｯﾄ　4ﾋﾞｯﾄ　12ﾋﾞｯﾄ



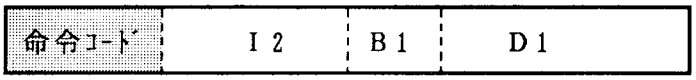
・ＲＸ形式　（レジスタとインデックス付き記憶域との操作）

8ﾋﾞｯﾄ　4ﾋﾞｯﾄ　4ﾋﾞｯﾄ　4ﾋﾞｯﾄ　12ﾋﾞｯﾄ



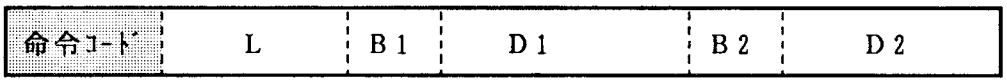
・ＳＩ形式　（即値と記憶域の操作）

8ﾋﾞｯﾄ　8ﾋﾞｯﾄ　4ﾋﾞｯﾄ　12ﾋﾞｯﾄ

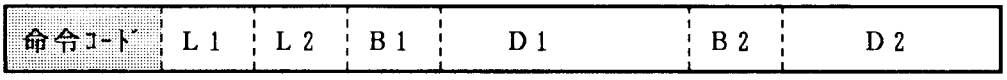


・ＳＳ形式　（記憶域と記憶域の操作）

8ﾋﾞｯﾄ　8ﾋﾞｯﾄ　4ﾋﾞｯﾄ　12ﾋﾞｯﾄ　4ﾋﾞｯﾄ　12ﾋﾞｯﾄ



8ﾋﾞｯﾄ　4ﾋﾞｯﾄ　4ﾋﾞｯﾄ　4ﾋﾞｯﾄ　12ﾋﾞｯﾄ　4ﾋﾞｯﾄ　12ﾋﾞｯﾄ



例）マイクロプロセッサのアドレッシングモード

・レジスタ・オペランドモード
・イミディエート・オペランドモード
・メモリ・アドレッシングモード
　ダイレクトモード
　レジスタ間接モード
　ベースドモード
　インデックスドモード
　ベースド・インデックスドモード
　ディスプレイスメント付きベースド・インデックスモード

指導上の留意点

ポイント

① 命令形式とオペランドの方式を正しく理解させる。

② レジスタの使用方式によりいくつかの演算の実現方式がある事を理解させ、各方式での演算のやり方を理解させる。

③ アドレッシングモードについて、その種類、方法を理解させ、有効アドレスを算出できるようにする。

④ 代表的なコンピュータで用いられているアドレッシングモードを理解させる。

用語

命令コード　オペランド　0アドレス方式　1アドレス方式　2アドレス方式
3アドレス方式　メモリ間直接演算方式　アキュムレータ方式　多数レジスタ方式
スタック方式　アドレッシングモード　直接アドレス　間接アドレス
インデックス修飾アドレス　相対アドレス　即値アドレス

講師ノート

第2種情報処理技術者試験

・アドレッシングモードとして、各レジスタの内容から有効アドレスを求める方法を充分に理解させる。

・COMETを例にとり、有効アドレスを求める演習を行うと良い。

２．４　命令の種類

コンピュータが実行する命令の種類・機能を理解させ、各々の命令が実行されると何がどのように変化するかを理解させる。

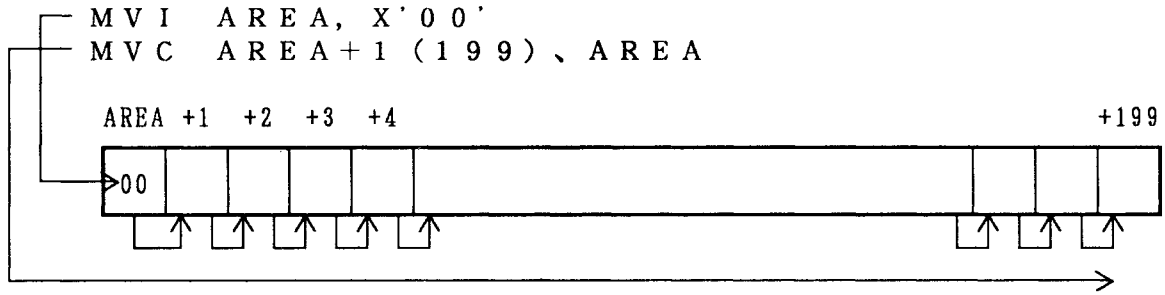
命令の種類には、一般命令と特権命令がある。一般命令は、単純命令（データ転送、算術命令、論理演算、比較命令、分岐）と、複合命令（コード変換など）がある。
また、特権命令は、一般のプログラムでは使用が許されておらず、ＯＳなど制御プログラムの中でしか使えない命令で、入出力命令、システム制御命令等がある。
命令の例としてＣＯＭＥＴを中心に挙げる。

２．４．１　データ転送命令

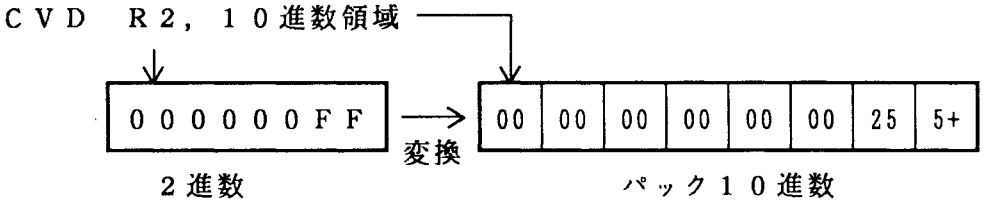
オペランドのデータを別のオペランドに移動、転送、変換する命令である。

例）
ＣＯＭＥＴでは、
ロード（LD）、ストア（ST）、ロードアドレス（LEA）の各命令がある。
ロードアドレス命令では、ロードされたデータの内容により、フラグレジスタの値（負、零、正）を設定する。

例）汎用コンピュータの領域クリアー



例）汎用コンピュータのデータ変換（２進数→パック１０進数）

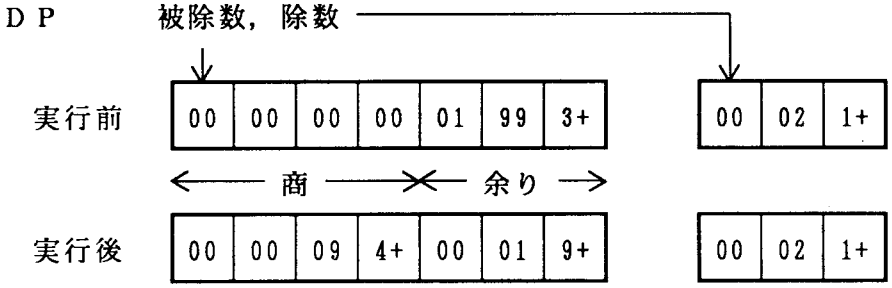


２．４．２　算術演算命令

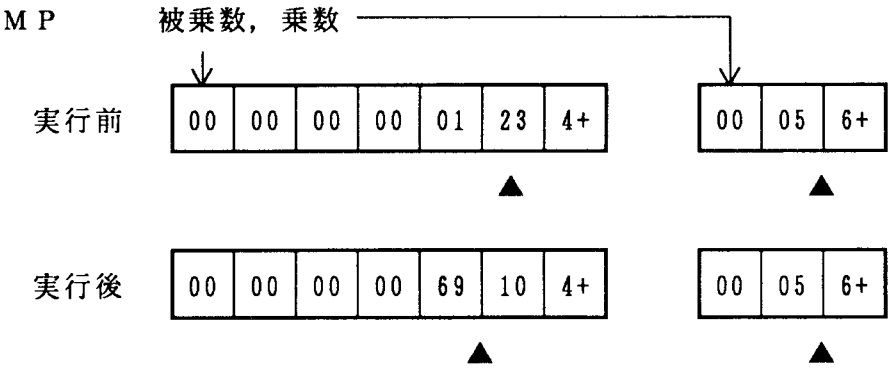
オペランドを符号付きの数値として取り扱う命令である。
命令の機能には、加算（＋）、減算（－）、乗算（＊）、除算（／）がある。
除算命令で除数が０であったり、演算結果がオーバフローすると命令の実行は中断され割り込みが発生する。

例）
ＣＯＭＥＴでは、
算術加算（ADD）、算術減算（SUB）の各命令がある。算術乗算、算術除算は用意されていない。
オペランドで指定された内容で演算を行い、指定されたオペランドに結果を格納する。演算結果によりフラグレジスタの値（負、零、正）を設定する。

例）汎用コンピュータの１０進数の除算



例）汎用コンピュータの１０進数の乗算（小数点の考え方）



２．４．３　論理演算命令

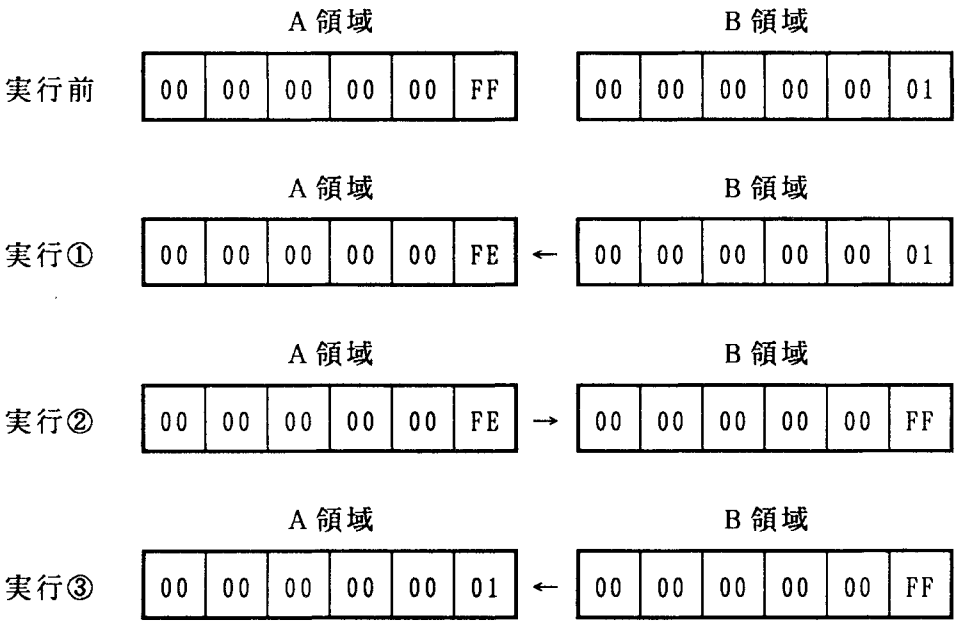
オペランドを無符号の２進数として取り扱う命令である。
命令の機能には、論理和（ＯＲ）、論理積（ＡＮＤ）、論理否定（ＮＯＴ）、排他的論理和（ＥＯＲ）などがある。

例）

ＣＯＭＥＴでは、
論理積（AND）、論理和（OR）、排他的論理和（EOR）の各命令がある。
オペランドで指定された内容で演算を行い、指定されたオペランドに結果を格納する。演算結果によりフラグレジスタの値（負、零、正）を設定する。

例）ワークエリアを使用しない領域の入れ替え（排他的論理和）

①　ＸＣ　Ａ領域，Ｂ領域
②　ＸＣ　Ｂ領域，Ａ領域
③　ＸＣ　Ａ領域，Ｂ領域

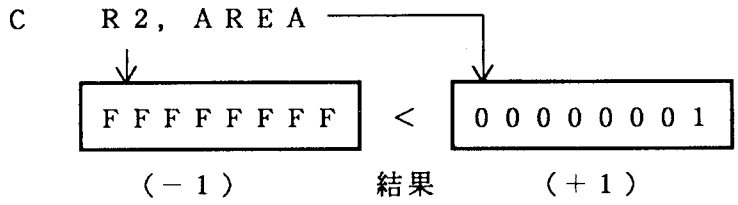


２．４．４　比較演算命令

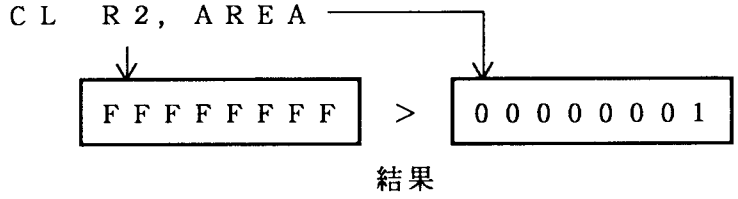
比較命令には、オペランドの符号を考慮した算術比較と、オペランドを無符号として比較する論理比較がある。
比較の実行結果は条件コードとしてプログラム状況語（ＰＳＷ）に保存される。条件コードは分岐命令で扱える。

例）
ＣＯＭＥＴでは、
算術比較（CPA）、論理比較（CPL）の各命令がある。
オペランドで指定された内容で算術比較または論理比較を行い、比較結果によりフラグレジスタに値（負、零、正）を設定する。

例）算術比較



例）論理比較

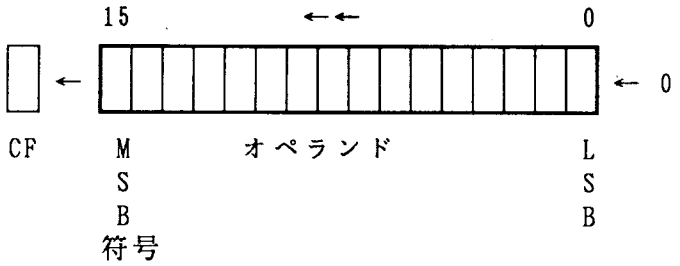


２．４．５　シフト演算命令

シフト命令は、データを指定ビット数シフトするときに使用する命令で、シフトの方法により算術左シフト、算術右シフト、論理左シフト、論理右シフトがある。算術左シフトと、論理左シフトは、マシンによっては全く同じ動作をする命令である。
ローテイト命令には、キャリー付き左ローテイト、キャリー付き右ローテイト、キャリー無し左ローテイト、キャリー無し右ローテイトがある。

（１）算術左シフト、論理左シフト

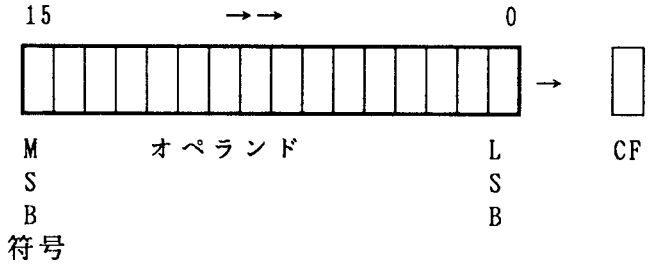
オペランドの内容を１ビットづつ左にシフトする。
最も左のビット（最上位ビット：符号ビット）は、キャリーフラグ（CF）に格納される。
最も右のビット（最下位ビット）には、０が入る。



注：　算術左シフトの場合、符号を除いた残りビットを左シフトするアーキテクチャ（汎用コンピュータ、ＣＯＭＥＴ）もある。

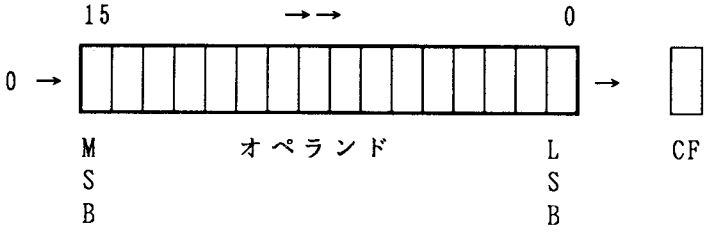
（２）算術右シフト

オペランドの内容を１ビットづつ右にシフトする。
最も左のビット（最上位ビット：符号ビット）は、シフト前と同じ値が入る。
最も右のビット（最下位ビット）は、キャリーフラグ（CF）に格納される。



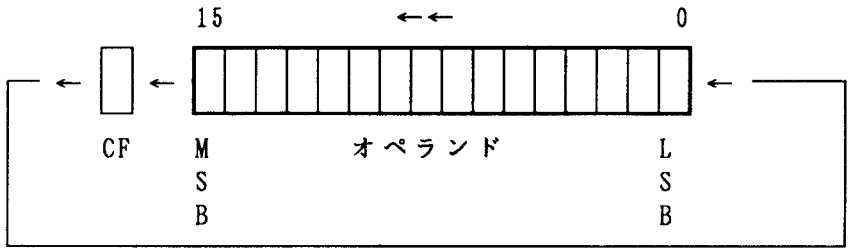
（３）論理右シフト

オペランドの内容を１ビットづつ右にシフトする。
最も左のビット（最上位ビット：符号ビット）は、０が入る。
最も右のビット（最下位ビット）は、キャリーフラグ（CF）に格納される。



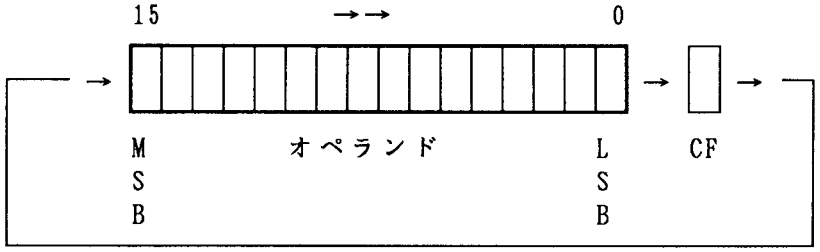
（４）キャリー付き左ローテイト

オペランドの内容を指定ビット数左に回転する。
このとき、キャリーフラグを拡張ビットとして回転するデータとともに用いる。
最も左のビット（最上位ビット）は、キャリーフラグ（CF）に格納され、キャリーフラグの内容が最も右のビット（最下位ビット）に入る。



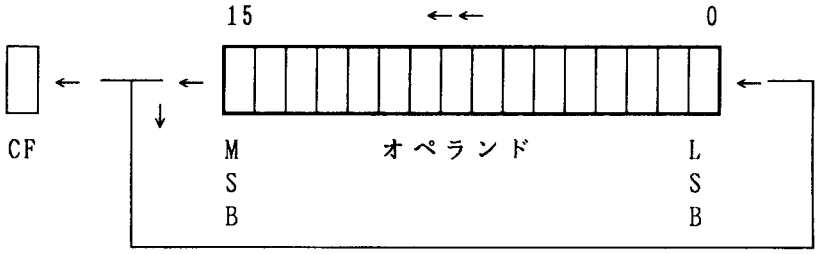
（５）キャリー付き右ローテイト

オペランドの内容を指定ビット数右に回転する。
このとき、キャリーフラグを拡張ビットとして回転するデータとともに用いる。
最も右のビット（最下位ビット）は、キャリーフラグ（CF）に格納され、キャリーフラグの内容が最も左のビット（最上位ビット）に入る。



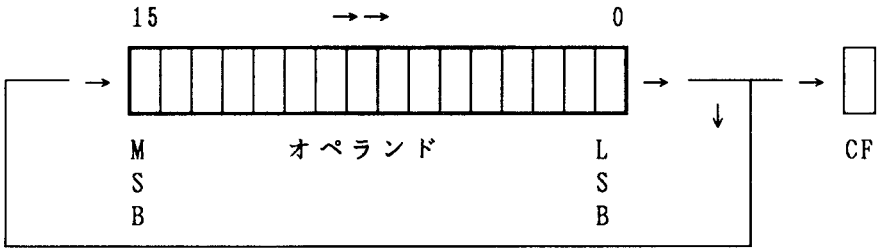
（６）キャリー無し左ローテイト

オペランドの内容を指定ビット数左に回転する。
最も左のビット（最上位ビット）は、キャリーフラグ（CF）と最も右のビット（最下位ビット）に格納される。



（７）キャリー無し右ローテイト

オペランドの内容を指定ビット数右に回転する。
最も右のビット（最下位ビット）は、キャリーフラグ（CF）と最も左のビット（最上位ビット）に格納される。



例）
ＣＯＭＥＴでは、
算術左シフト（SLA）、算術右シフト（SRA）、
論理左シフト（SLL）、論理右シフト（SRL）
の各命令があり、算術左シフトと論理左シフトでは、実行後の値が違う。
オペランドで指定された内容で算術左（右）シフト、または論理左（右）シフトを行い、シフト結果によりフラグレジスタの値を設定する。
ローテイト命令はない。

２．４．６　分岐命令

通常、主記憶上の命令は順次実行される。分岐命令は、比較命令、算術命令、論理演算命令、入出力命令を実行した結果の状況を判定して、命令の実行順序を変更するために使用する。

分岐命令は、無条件分岐命令、条件付き分岐命令、ループ分岐命令に分けることができる。無条件分岐命令には、コール、リターン命令を含む。

例）

ＣＯＭＥＴでは、

正分岐　　（JPZ : Jump on Plus or Zero）、
負分岐　　（JMI : Jump on MInu ）、
非零分岐　（JNZ : Jump on Non Zero）、
零分岐　　（JZE : Jump on ZEro）
無条件分岐（JMP : unconditional JuMP）

の各命令がある。

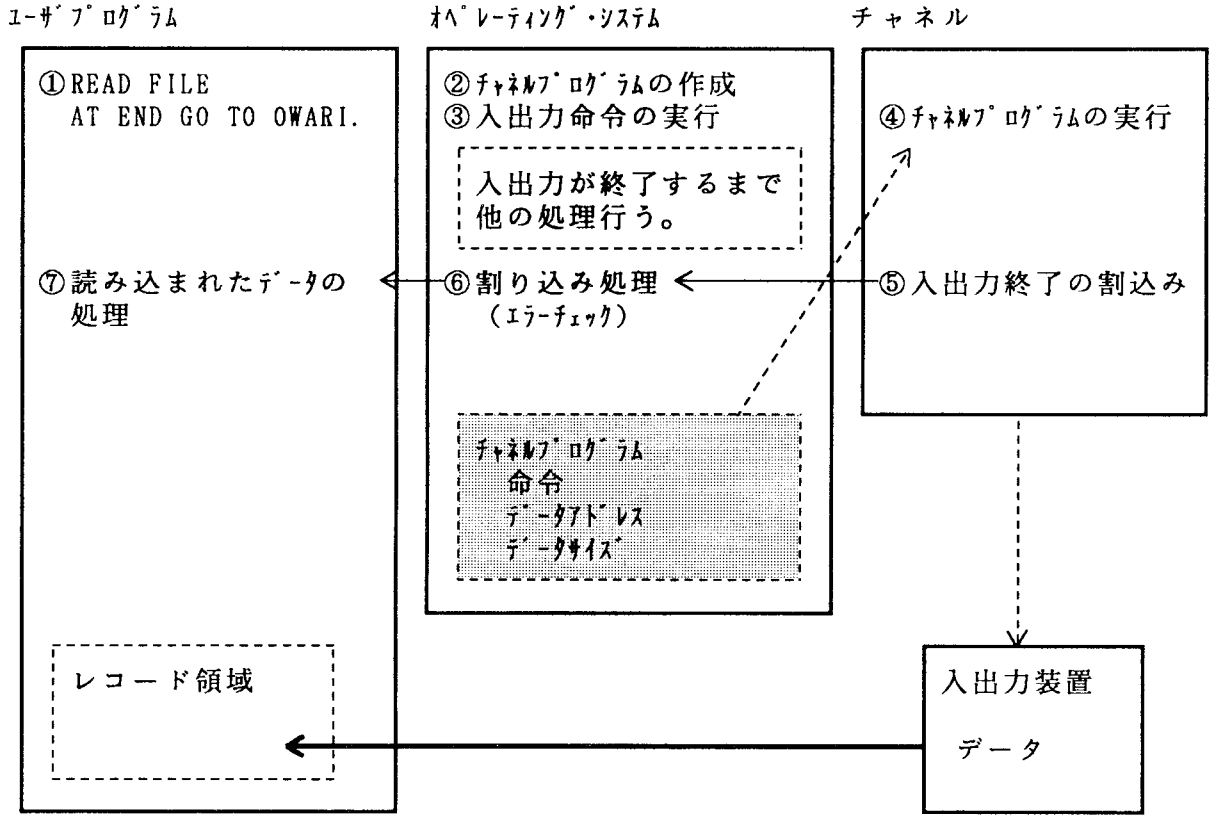
フラッグレジスタ（FR）の値（負：10、零：01、正：00）により有効アドレスに分岐する。分岐しない場合は次の命令に進む。

命　令	分岐するときのFRの値
JPZ （正分岐）	00, 01　（正、零）
JMI （負分岐）	10　　（負）
JNZ （非零分岐）	00, 10　（正、負）
JZE （零分岐）	01　　（零）
JMP （無条件分岐）	00, 01, 10, 11

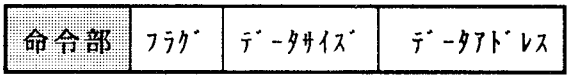
プログラミングのサブルーチン（関数）は、コール命令とリターン命令で実現している。コール命令では復帰情報を、スタック、または、レジスタに待避する。リターン命令では待避した情報を基にしてコール命令の次に戻る。

２．４．７　入出力命令

主記憶装置と入出力装置間のデータ転送を行う命令である。
汎用コンピュータでは、チャネルにチャネルプログラム（チャネルコマンド）の実行要求を送り、チャネルが入出力装置と主記憶装置の間で入出力操作を行う。入出力操作の終了はチャネルからの「割り込み」処理により知らされる。

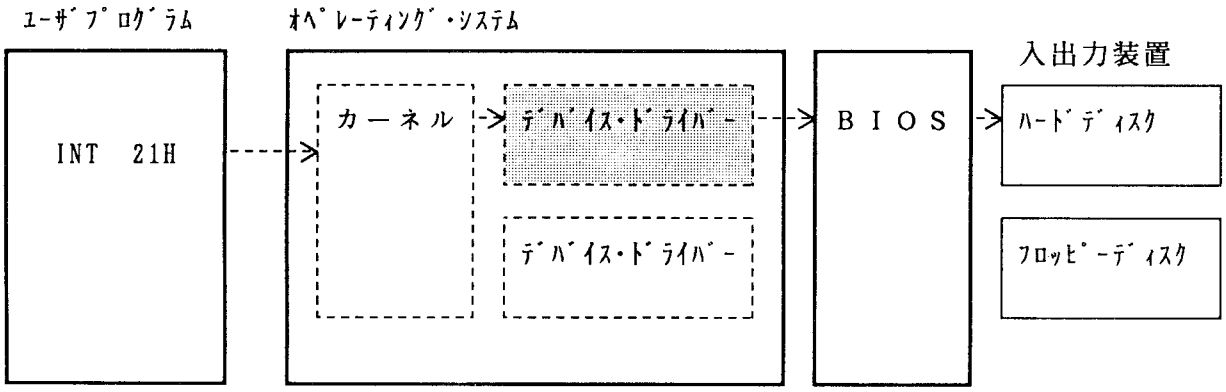


チャネルプログラムは、チャネルコマンド列（ＣＣＷ：Channel Command Word）で構成され、次の形式をしている。



命令部は、入出力動作（READ、SENSE、WRITE、CONTROL）を指定する。
データアドレスは、主記憶装置のアドレスを指す。または、入出力装置のデータアドレス（シリンダ、トラックなど）を指す。
データサイズは、転送するデータの長さを指定する。

マイクロプロセッサの入出力は次のように構成されている。



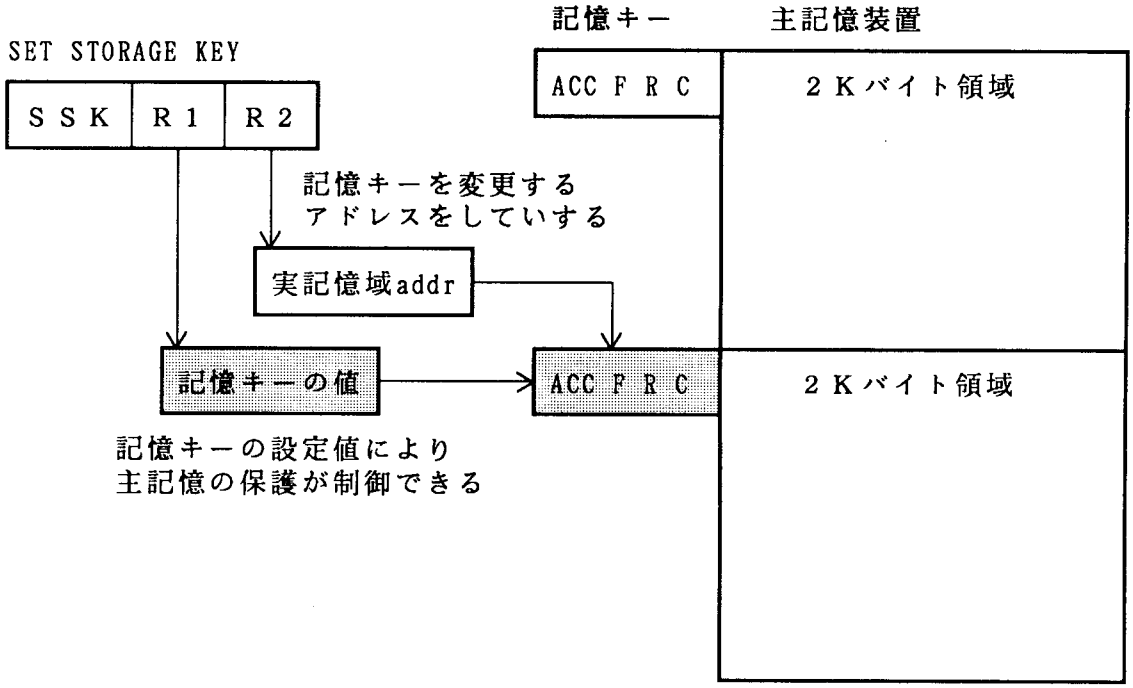
２．４．８　システム制御命令

割り込み処理、制御関連レジスタの更新など、システム状態の切り替えを行う命令であり、特権命令とも呼ばれる。

システム制御命令は、オペレーティング・システムがシステムの管理、制御を行うために使用され、一般のプログラミングでは直接使用することはない。使用すると割り込みが発生する。

オペレーティング・システムが使用する場合でも、ＣＰＵがスーパーバイザモードでないと実行できない。また、スーパーバイザモードとユーザモードの状態の切り替えもシステム制御命令を使用する。

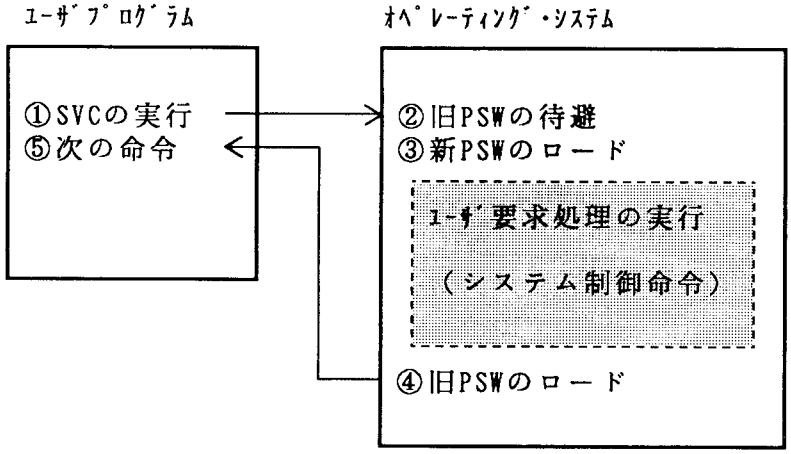
例）主記憶装置の記憶保護キーの設定の命令



２．４．９　スーパーバイザ・コール（システムコール）

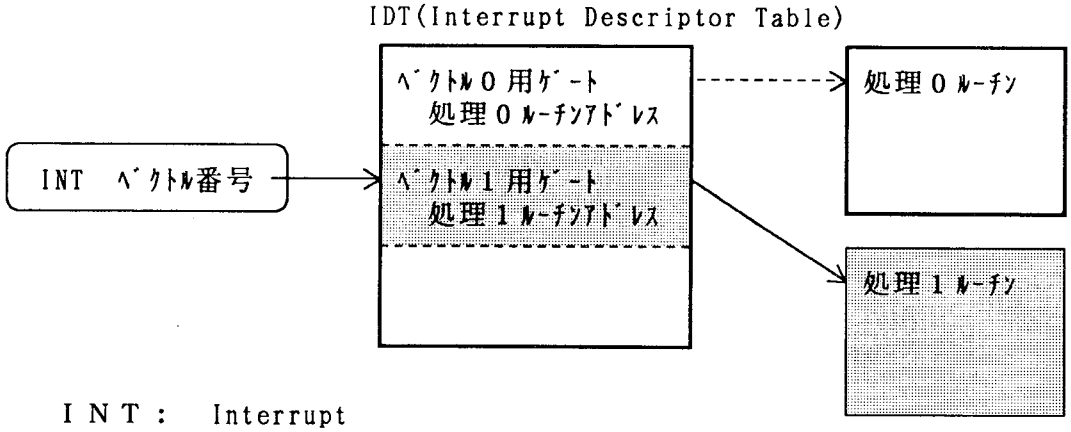
一般のプログラムが、入出力命令などでシステム制御命令（特権命令）の実行を依頼したいとき、スーパーバイザ・コール命令を実行して要求をオペレーティング・システムに伝える。

例）スーパーバイザ・コールの処理手順について（汎用コンピュータの例）



ＳＶＣ：　スーパーバイザ・コール
ＰＳＷ：　プログラム状況語

例）スーパーバイザ・コールのハードウェアの仕組みについて
（マイクロコンピュータの例）



ＩＮＴ：　Interrupt

指導上の留意点

ポイント

① コンピュータで用意されている命令の種類と内容を理解させる。

② 実行した命令により中央処理装置の状態（ＰＳＷの状態）や、コンディションコード（フラグ）がどのように変化するかを充分理解させる。

③ シフト命令のうち、算術左シフトの場合、符号を除いた残りビットを左シフトするアーキテクチャ（汎用コンピュータ、ＣＯＭＥＴ）と、符号を含んだビットを左にシフトするアーキテクチャもあるので注意を要する。

④ 入出力命令は、ユーザプログラムから要求が出された後どのように実行されるかのメカニズムを充分に理解させる。汎用機、マイクロプロセッサそれぞれについて例を用いながら説明する。

⑤ スーパーバイザ・コールの実行の仕組みを充分に理解させる。

用語

データ転送命令　算術命令　論理演算命令　比較命令　シフト命令　分岐命令
入出力命令　システム制御命令　スーパーバイザ・コール

講師ノート

第２種情報処理技術者試験

・アセンブラの試験対策も兼ねて、ＣＯＭＥＴの命令がどのように実行されるかを理解させると良い。

２．５　ＣＯＭＥＴの例

命令表記の例、オペランド表記の例、命令の種類の例としてＣＯＭＥＴが提供する命令の形式と、その機能を示す。

各命令の説明には、次の表記法を用いる。

GR	GRの値を番号とする汎用レジスタ（ただし、0≦GR≦4）
XR	XRの値を番号とする指標レジスタ（ただし、1≦XR≦4）
SP	スタックポインタ（汎用レジスタ4番）
adr	ラベル名（ラベル名に対応する番地を示す）または10進定数（ただし、-32768<=adr<=65535とする。adrはアドレスとして0～65535の値をもつが、32768～65535の値を負の10進定数で記述する事もできる。）
有効ｱﾄﾞﾚｽ	adrとXRの内容とのアドレス加算値またはその値が示す番地。
（x）	X番地の内容、Xがレジスタの場合はレジスタの内容。
[ ]	[ ]に囲まれた部分は、省略可能であることを示す。XRを省略した場合は、指標レジスタによる修飾を行わない。

命令及びその機能を示す。命令は、アセンブラの表記法で記述する。

命令	命令形式		命令の説明
	命令ｺｰﾄﾞ	ｵﾍﾟﾗﾝﾄﾞ	

(1)ロード、ストア命令

ﾛｰﾄﾞ LoaD	LD	GR,adr[,XR]	(有効ｱﾄﾞﾚｽ)をGRに設定する。
ｽﾄｱ STore	ST	GR,adr[,XR]	(GR)を有効ｱﾄﾞﾚｽが示す番地に格納する。

(2)ロードアドレス命令

ﾛｰﾄﾞｱﾄﾞﾚｽ Load Effective Address	LEA	GR,adr[,XR]	有効ｱﾄﾞﾚｽをGRに設定する GRの値によりFRを設定する

命令	命令形式		命令の説明
	命令コード	オペランド	

(3)算術、論理演算命令

算術加算 ADD arithmetic	ADD	GR,adr[,XR]	(GR)と(有効アドレス)に、指定した演算を施し、結果をGRに設定する。 なお、算術減算では、(GR)から(有効アドレス)を減算する。 算術結果によりFRを設定する。
算術減算 SUBtract arithmetic	SUB	GR,adr[,XR]	
論理積 AND	AND	GR,adr[,XR]	
論理和 OR	OR	GR,adr[,XR]	
排他的論理和 Exclusive OR	EOR	GR,adr[,XR]	

(4)比較演算命令

算術比較 ComPare Arithmetic	CPA	GR,adr[,XR]	(GR)と(有効アドレス)の算術比較又は論理比較を行い、比較結果によりFRに次の値を設定する。
論理比較 ComPare Logical	CPL	GR,adr[,XR]	

↓

比較結果	FRのビット値
(GR)>(有効アドレス)	00
(GR)=(有効アドレス)	01
(GR)<(有効アドレス)	10

(5)シフト命令

算術左シフト Shift Left Arithmetic	SLA	GR,adr[,XR]	(GR)を符号を除き有効アドレスで指定したビット数だけ左または右にシフトする。 シフトの結果、空いたビット位置には、左シフトのときは0、右シフトのときは符号と同じものが入る。 シフトの結果によりFRを設定する。
算術右シフト Shift Right Arithmetic	SRA	GR,adr[,XR]	
論理左シフト Shift Left Logical	SLL	GR,adr[,XR]	(GR)を符号を含み有効アドレスで指定したビット数だけ左または右にシフトする。 シフトの結果、空いたビット位置には、0が入る。 シフトの結果によりFRを設定する。
論理右シフト Shift Right Logical	SRL	GR,adr[,XR]	

命令	命令形式		命令の説明
	命令コード	オペランド	

(6)分岐命令

正分岐 Jump on Plus or Zero	JPZ	adr[,XR]	FRの値により、有効アドレスに分岐する。分岐しないときは次の命令に進む。 命令 / 分岐するときのFRの値 JPZ: 00,01 JMI: 10 JNZ: 00,10 JZE: 01
負分岐 Jump on MInus	JMI	adr[,XR]	
非零分岐 Jump on Non Zero	JNZ	adr[,XR]	
零分岐 Jump on ZEro	JZE	adr[,XR]	
無条件分岐 umconditional JuMP	JMP	adr[,XR]	無条件に有効アドレスに分岐する。

(7)スタック操作命令

プッシュ PUSH effective address	PUSH	adr[,XR]	SPから1をアドレス減算した後、有効アドレスを(SP)番地に格納する。
ポップ POP up	POP	GR	(SP)番地の内容をGRに設定した後、SPに1をアドレス加算する。

(8)コール、リターン命令

コール CALL subroutine	CALL	adr[,XR]	SPから1をアドレス減算した後、PCの現在地に2をアドレス加算した値を(SP)番地に格納し、有効アドレスに分岐する。
リターン RETurn from subroutine	RET		(SP)番地の内容を取り出した後、SPに1をアドレス加算し、先に取り出した内容(番地)に分岐する（取り出した内容をPCに設定する）

指導上の留意点

ポイント

①ＣＯＭＥＴを例にとり命令の実際の動きを理解させる。

用語

データ転送命令　算術命令　論理演算命令　比較命令　シフト命令　分岐命令
スタック操作命令

講師ノート

２．６　プログラム実行の仕組み

演算処理装置が、命令を読み込んでから実行する動作の流れを理解するとともに、プログラムの実行中に発生する割り込みについて、その種類・役割を理解し、割込みの発生から、終了までのメカニズムを理解させる。

２．６．１　処理装置の動作

処理装置では、命令の読み込みの動作と命令の実行動作の２つがある。この２つの動作（命令実行サイクル）を繰り返しながら、プログラムを実行する。

（１）読み込み動作

① プログラムカウンタが、読み込むべき命令が格納されている主記憶装置のアドレス（番地）を示している。
② プログラムカウンタが指している命令を、命令レジスタに読み込む。
③ プログラムカウンタを増加させ、次に読み込むべき命令を指すようにする。

（２）実行動作

④ 命令レジスタの内容のうち、命令部をデコーダに転送する。
⑤ 命令レジスタの内容のうち、読み込むべきデータのアドレスをアドレスレジスタが示すようにする。
⑥ アドレスレジスタに従って、データを読みに行く。
⑦ データをレジスタ１やレジスタ２に読み込む。
⑧ デコーダにより処理内容を演算回路に指示する。
⑨ 命令の内容に従って、レジスタ１，２の内容を演算回路により処理させる。
⑩ アドレスレジスタが、処理内容により格納すべきデータのアドレスを示す。
⑪ データを格納する
⑫ 一連の命令が終了したので、次の命令を実行する（①に戻る）。
この時、割り込み要求があれば、次の命令の状態を退避され、割り込み処理を実行する。

例）汎用コンピュータの「ＢＲＡＮＣＨ　ＡＮＤ　ＬＩＮＫ」の動作

ＢＡＬＲ　Ｒａ，Ｒｂ

動作説明　－　現ＰＳＷの命令アドレスがＲａにロードされ、その後、Ｒｂのブランチ・アドレスをＰＳＷの命令アドレスに置き換える。

```
L       R15, =A(SUBRTN)
BALR    R15, R15
```

上の例題のように、ＲａとＲｂが同じレジスタの場合でも、正しく動作（ＳＵＢＲＴＮへ分岐）することを説明する。

２．６．２　割り込み

特別な事象や条件が発生したときは、現在実行中のプログラムを一時中断させて、その事象や条件を処理するために予め用意されているプログラム（割り込み処理ルーチン）を強制的に実行する必要がある。

（１）システムの状態

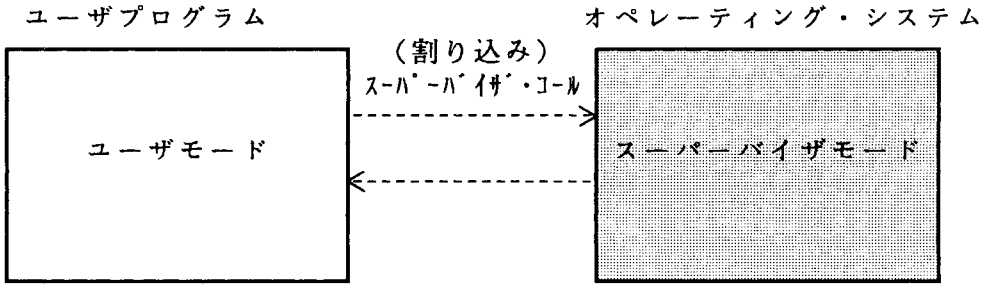
システムの状態の制御は、ユーザに直接行わせずオペレーティング・システムが管理する必要がある。この場合、オペレーティング・システムの働きを円滑にするために、計算機の状態を

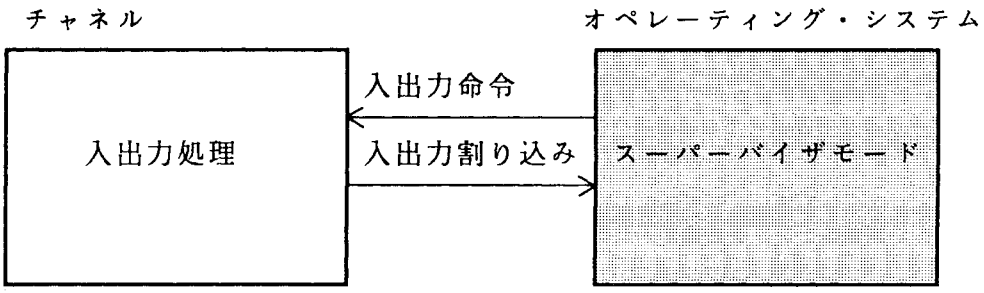
・スーパーバイザモード（オペレーティング・システム走行状態）
・ユーザモード（ユーザプログラム走行状態）

と区別する。

スーパーバイザモードでは、全ての命令が実行可能であるが、ユーザモードでは、特権命令（入出力命令、記憶保護関係など）を実行する事はできない。

ユーザモードで特権命令を実行しようとすると、割り込みが発生し、その命令は実行されない。





（２）割り込みの種類

割り込みには、以下のような種類がある。

① 機械チェック割り込み
ハードウェアの誤動作・故障や電源異常など重大な障害が発生した事を知らせる。

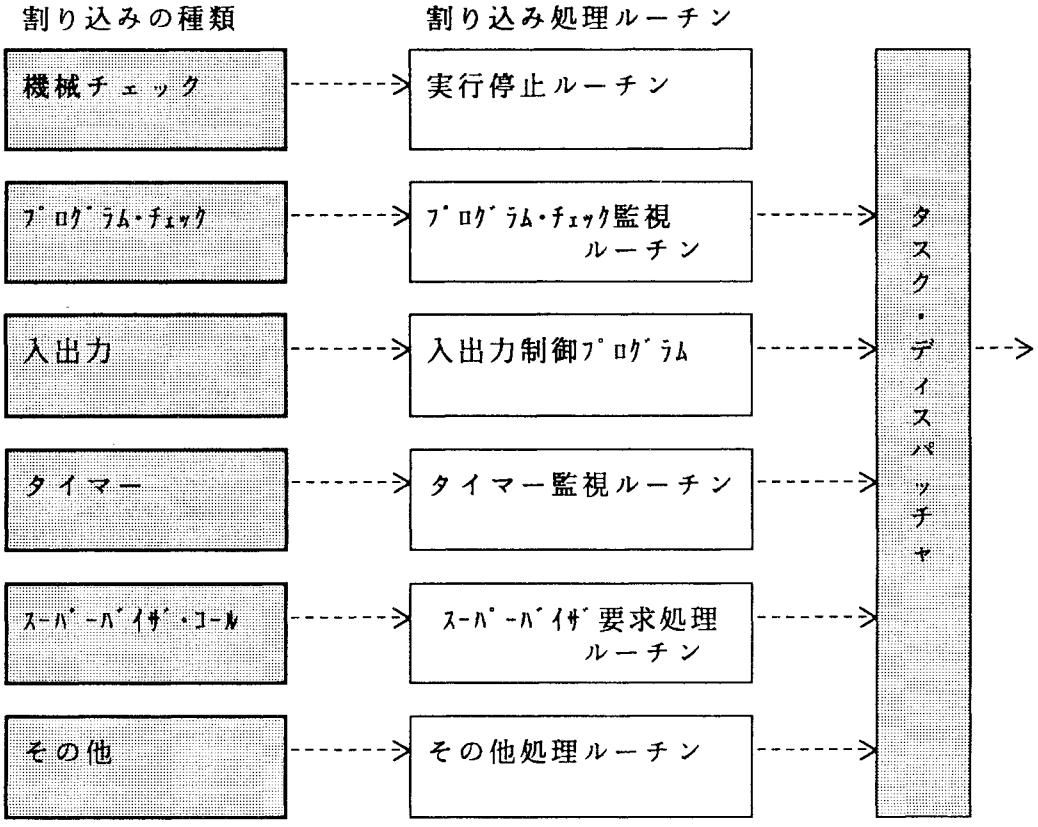
② 外部割り込み
オペレータによる要求などを知らせる。

③ プログラム割り込み（プログラム例外）
プログラム実行時の誤り（命令コードエラー、記憶保護違反、演算のオーバフロー、アンダフロー）などの発生を知らせる。

④ 入出力割り込み
入出力装置から、入出力動作の終了を知らせる。

⑤ スーパーバイザ・コール割り込み
一般ユーザのプログラムが、入出力命令などで特権命令（オペレーティング・システムでのみ実行可能）の実行を依頼するときに割り込みを発生させ、要求を知らせる。

⑥ タイマ割り込み
タイムシェアリングシステム（ＴＳＳ）などでタイムスライスの終了を知らせる。



（３）割り込み処理

割り込み処理は、以下の手順で行われる。

① 割り込み要求のチェック
プログラムの一命令実行サイクルの終了時に、割り込み要求が発生しているか否かをチェックする。割り込み要求がない場合は、実行中のプログラムの次の命令を実行する。割り込み要求が発生している場合には、②以下を実行する。

② 割り込みコードを設定し、現在のＰＳＷ（プログラム状況語）を旧ＰＳＷ領域に待避する。

③ 新ＰＳＷ領域の内容を現ＰＳＷにロードする。

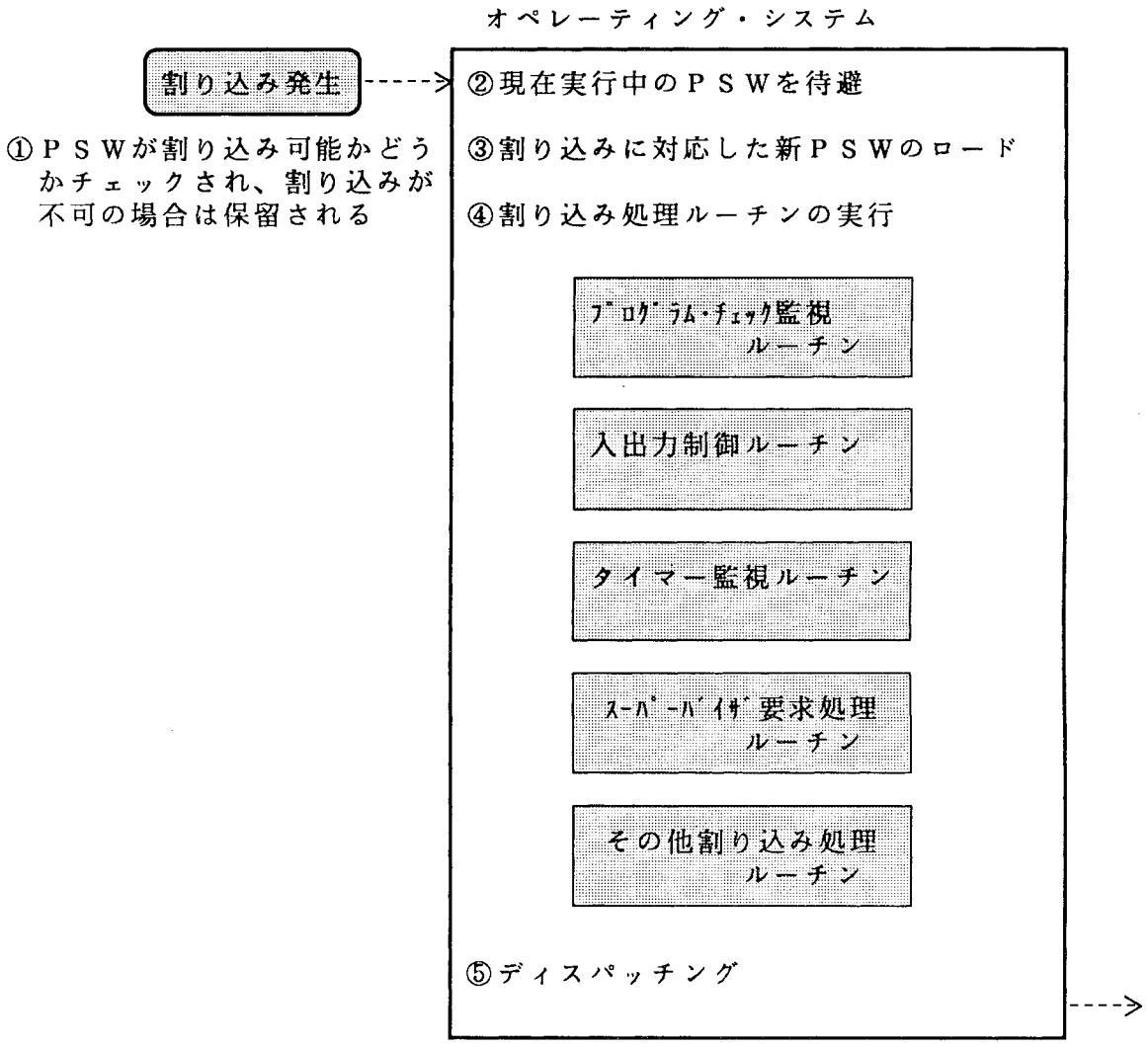
・②③は、ハードウェアの動作として行う

④ 割り込み処理ルーチンを実行する。

⑤ 実行可能状態のタスクから優先順位の最高のものを選択する（dispatcher）

⑥ 選択されたタスクの旧ＰＳＷを現ＰＳＷへロードし、選択されたタスクの実行を再開する。

・⑤⑥をディスパッチング（dispatching）という。

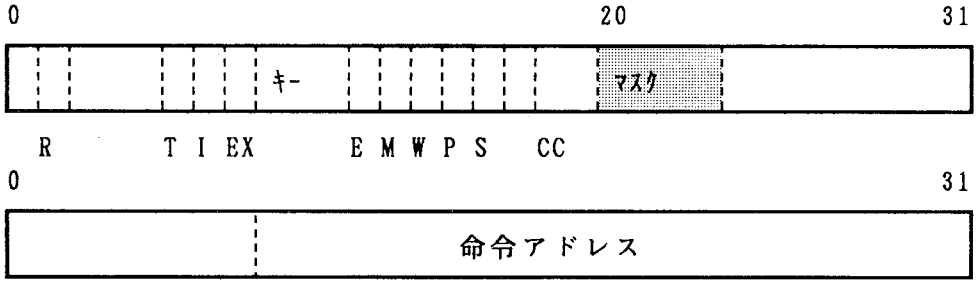


（４）割り込みの優先順位と割り込みマスク

同時に２種類以上の割り込みが発生したとき、それを受け付ける順序は次のように定められている。

優先順位	割り込み原因
1	マシンチェック
2	スーパーバイザ・コール／プログラムエラー／プログラム例外
3	外部割り込み
4	入出力処理の終了

割り込み要求が発生していても、割り込みを起こしたくないとき、これをプログラマが制御できるように、割り込み要因の１つ１つにマスク用のビットを用意し、これが１の時だけ実際の割り込みが受け付けられるようにして、割り込み処理に柔軟性を与えられるようにしている。これらのビットを、割り込みマスクレジスタとして一箇所に集め、命令で書き換えられるようにしている。



I　　ＣＰＵが入出力割り込みの実行を制御する。
　　このビットが０の場合は、チャネルは入出力割り込みができない。
P　　ＣＰＵの状態（ユーザモード、スーパーバイザモード）を示す。
ﾏｽｸ　プログラム例外の割り込みを制御（ビットが０で割り込み禁止）する。
　　ﾋﾞｯﾄ20　固定小数点桁あふれ
　　ﾋﾞｯﾄ21　１０進数桁あふれ
　　ﾋﾞｯﾄ22　指数下位桁あふれ
　　ﾋﾞｯﾄ23　有効数字

プログラミング言語（ＰＬ／１）でも、いくつかの割り込み（例外）をユーザが自由に制御できる。

```
(NOFIXEDOVERFLOW):  X = A * B;
```
①固定小数点桁あふれを禁止して演算を実行する。

```
ON ZERODIVIDE BEGIN;
                END;
```
②除算不正が発生した時、ユーザがその対応を行う。

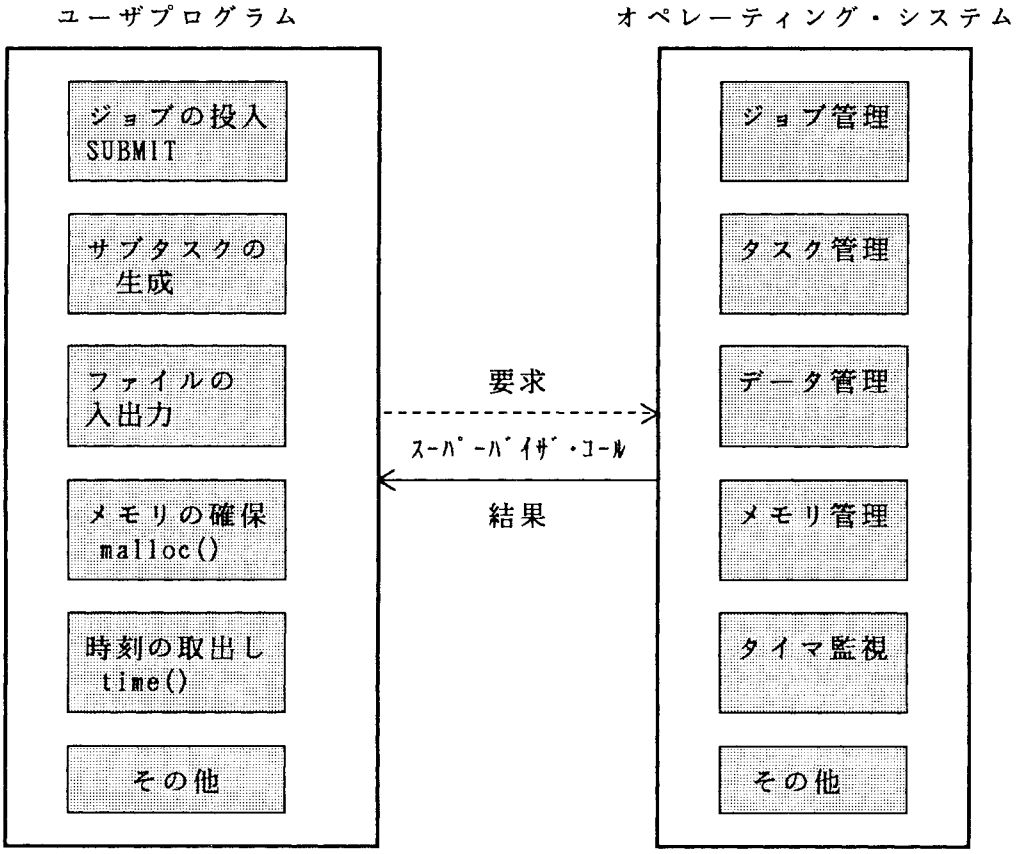
```
SIGNAL ATTENTION;
```
③ユーザが端末割り込みを発生させる。

（５）スーパーバイザ・コール（ＳＶＣ：Supervisor call）

ユーザプログラムは、直接は入出力命令を実行できないので、特別な命令によりオペレーティング・システムに処理を依頼する。
この特別な命令がスーパーバイザ・コールである。ＳＶＣ命令には、オペレーティング・システムに対してどのような処理を行うかを示すパラメータを持つ。

ＳＶＣ命令の実行は次の通りである。

① 割り込みと同様のイベントを発生させる。
② 現在のＰＳＷを旧ＰＳＷ領域に格納する。
③ 新ＰＳＷ領域の内容を現ＰＳＷにロードする。
④ ＳＶＣ命令のパラメータにより、対応するルーチンへ分岐する。
⑤ 割り込み処理と同様にディスパッチングを行い、選択されたタスクを実行する。



指導上の留意点

ポイント

① プログラムが、命令をフェッチしてからデコードし、実行するまでの内部の動きを確実に理解させる。

② 割り込みについて、その種類、処理手順を理解させる。

用語

ユーザモード　スーパーバイザモード
マシンチェック割り込み　外部割り込み　プログラム割り込み　入出力割り込み
タイマ割り込み

講師ノート

第２種情報処理技術者試験

・システムの動作状態、割り込みに関しては、正誤問題、穴埋め問題などで出題される事が多いので、割り込みの種類とその特徴を正確に覚えておくこと