数学・物理 回憶録

最終更新日: 2023 年 11 月 23 日

目次

【注意】

1. ベクトルは [　] で表す．また，原則列ベクトル表記 $\boldsymbol{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$ とする．紙面上の関係で行ベクトル表記で表すときは転置行列を表す $\top$ を付けて $\boldsymbol{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^\top$ と表す．
2. $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ はそれぞれ x 軸，y 軸，z 軸方向の基本ベクトルとする．

第 1 章　偏微分法

1.1　関数の極限

関数 $f(x,\ y)$ に於いて，点 $(x,\ y)$ が点 $(a,\ b)$ 以外の点を取りながら $(a,\ b)$ に限りなく近づくとき，関数の値が C に限りなく近づくならば，$f(x,\ y)$ は C に**収束する**といい，

$$\lim_{(x,\ y)\to(a,\ b)} f(x,\ y) = C \tag{1.1}$$

と表す．C を**極限値**という．

このとき，$(x,\ y)$ がどんな近づき方で $(a,\ b)$ に近づいても極限値がある一定の値 C になることが必要である．

例えば，$f(x,\ y) = \dfrac{xy}{x^2+y^2}$ について，$\lim\limits_{(x,\ y)\to(0,\ 0)} f(x,\ y)$ を考える．

(i)　点を直線 $y = x$ 上で近づけると $f(x,\ y) = \dfrac{x\cdot x}{x^2+x^2} = \dfrac{1}{2}$ であるから $\dfrac{1}{2}$ に収束する．

(ii)　点を直線 $y = 2x$ 上で近づけると $f(x,\ y) = \dfrac{x\cdot 2x}{x^2+(2x)^2} = \dfrac{2}{5}$ であるから $\dfrac{2}{5}$ に収束する．

よって，極限値はない．

関数 $f(x,\ y)$ の定義域内の点 $\mathrm{P}(a,\ b)$ について，

$$\lim_{(x,\ y)\to(a,\ b)} f(x,\ y) = f(a,\ b) \tag{1.2}$$

が成り立つとき，$f(x,\ y)$ は点 P で**連続である**という．

1.2　偏導関数

関数 $z = f(x,\ y)$ に於いて

$$\frac{\partial z}{\partial x} = \lim_{\Delta x\to 0} \frac{f(x+\Delta x,\ y) - f(x,\ y)}{\Delta x} \tag{1.3}$$

$f(x,\ y)$ の x **についての偏導関数**といい，f_x とも表す．また，

$$\frac{\partial z}{\partial y} = \lim_{\Delta y\to 0} \frac{f(x,\ y+\Delta y) - f(x,\ y)}{\Delta y} \tag{1.4}$$

を $f(x,\ y)$ の y **についての偏導関数**といい，f_y とも表す．

$x\ (y)$ **について偏微分可能**であるとは，点 $x = a\ (y = b)$ での偏微分係数が存在することである．また，偏微分係数 $f_x(a,\ b)$，$f_y(a,\ b)$ はそれぞれ点 $(a,\ b)$ の x 軸方向の傾き，y 軸方向の傾きを表す．

◆ A ◆　高階偏導関数

$z = f(x,\ y)$ で，2 階偏微分可能で全て連続のとき　$\dfrac{\partial^2 z}{\partial y\partial x} = \dfrac{\partial^2 z}{\partial x\partial y}$

一般に n 階偏微分可能で全て連続のとき，$n = k+l$ とすると $\dfrac{\partial^n z}{\partial x^k \partial y^l}$ は全て等しい．

◆ B ◆　全微分可能性

- $f(x,\ y)$ が点 $(a,\ b)$ で全微分可能　$\Longrightarrow$　$f(x,\ y)$ は $(a,\ b)$ で連続かつ $(a,\ b)$ で偏微分可能
- $f(x,\ y)$ の $\dfrac{\partial f}{\partial x}$，$\dfrac{\partial f}{\partial y}$ が $(a,\ b)$ で存在してそれらが連続である　$\Longrightarrow$　$f(x,\ y)$ は $(a,\ b)$ で全微分可能
- $f(x,\ y)$ が偏微分可能であっても全微分可能ではない（全微分の方が強い概念）．

$$df = \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y} \tag{1.5}$$

Tip　[導出]

関数 $f(x,\ y)$ の $\Delta x,\ \Delta y$ に対する増加量 Δf は

$$\Delta f = \underbrace{\underbrace{\frac{\partial f}{\partial x}\Delta x}_{\Delta x\text{ についての増加量}} + \underbrace{\frac{\partial f}{\partial y}\Delta y}_{\Delta y\text{ についての増加量}}}_{\text{接平面の公式}} + \varepsilon$$

更に，$\displaystyle\lim_{(\Delta x,\ \Delta y)\to(0,\ 0)} \frac{\varepsilon}{\sqrt{(\Delta x)^2+(\Delta y)^2}} = 0$ であれば，$f(x,\ y)$ は全微分可能という．

ここで，$\displaystyle\lim_{(\Delta x,\ \Delta y)\to(0,\ 0)} \frac{\varepsilon}{\sqrt{(\Delta x)^2+(\Delta y)^2}} = 0$ を，「ε は $\sqrt{(\Delta x)^2+(\Delta y)^2}$ より**高位の無限小**」という．これは，「ε は $\sqrt{(\Delta x)^2+(\Delta y)^2}$ とは比べ物にならないくらい速く 0 に近付く」という意味である．

◆ C ◆　接平面の方程式

曲面 $z = f(x,\ y)$ 上の点 $\big(a,\ b,\ f(a,\ b)\big)$ に於ける接平面の方程式

$$z - f(a,\ b) = \frac{\partial f(a,\ b)}{\partial x}(x-a) + \frac{\partial f(a,\ b)}{\partial y}(y-b) \tag{1.6}$$

曲面 $f(x,\ y,\ z) = 0$ 上の点 $(a,\ b,\ c)$ に於ける接平面の方程式

$$\begin{bmatrix} \dfrac{\partial f}{\partial x} & \dfrac{\partial f}{\partial y} & \dfrac{\partial f}{\partial z} \end{bmatrix} \begin{bmatrix} x-a \\ y-b \\ z-c \end{bmatrix} = 0 \tag{1.7}$$

これは，陰関数の微分法と式 (1.6) から分かる．また，これより曲面 $f(x,\ y,\ z) = 0$ の法線ベクトルは以下である．

$$\begin{bmatrix} \dfrac{\partial f}{\partial x} & \dfrac{\partial f}{\partial y} & \dfrac{\partial f}{\partial z} \end{bmatrix}^{\top} \tag{1.8}$$

◆ D ◆　チェーン・ルール（連鎖律）

チェーン・ルール (1)

$z = f(x,\ y)$ が全微分可能で，$x = x(t),\ y = y(t)$ が微分可能であるとき，z は t の関数である．

$$\frac{dz}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} \tag{1.9}$$

チェーン・ルール (2)

$z = f(x,\ y)$ が全微分可能で，$x = x(u,\ v),\ y = y(u,\ v)$ が偏微分可能であるとき

$$\frac{\partial z}{\partial u} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial u} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial u} \tag{1.10}$$

$$\frac{\partial z}{\partial v} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial v} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial v} \tag{1.11}$$

◆ E ◆　陰関数の微分法

$f(x,\ y) = 0$ によって表された x の関数 y の導関数

$$\frac{dy}{dx} = -\frac{\partial f}{\partial x} \Big/ \frac{\partial f}{\partial y} \tag{1.12}$$

Tip　[導出]

陰関数 $y(x)$ より，$f\big(x,\ y(x)\big) = 0$．両辺微分して

$$\frac{\partial f}{\partial x}\frac{dx}{dx} + \frac{\partial f}{\partial y}\frac{dy}{dx} = 0 \iff \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx} = 0 \quad \left(\because \frac{dx}{dx} = 1\right) \tag{1.13}$$

$f(x,\ y,\ z) = 0$ によって表された $x,\ y$ の関数 z の導関数

$$\frac{\partial z}{\partial x} = -\frac{\partial f}{\partial x} \Big/ \frac{\partial f}{\partial z} \qquad \frac{\partial z}{\partial y} = -\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial z} \tag{1.14}$$

Tip　[導出]

陰関数 $z(x,\ y)$ より，$f\big(x,\ y,\ z(x,\ y)\big)=0$．両辺 x で偏微分して

$$\frac{\partial f}{\partial x}\frac{\partial x}{\partial x}+\frac{\partial f}{\partial y}\frac{\partial y}{\partial x}+\frac{\partial f}{\partial z}\frac{\partial z}{\partial x}=0 \iff \frac{\partial f}{\partial x}+\frac{\partial f}{\partial z}\frac{\partial z}{\partial x}=0 \quad \left(\because \frac{\partial x}{\partial x}=1,\ \frac{\partial y}{\partial x}=0\right) \tag{1.15}$$

両辺 y で偏微分すると第 2 式が得られる．

◆F◆　2 変数のテイラーの定理

$f(x,\ y)$ が n 次までの連続な偏導関数を持つとき（C^n 級関数），

$$\frac{d^n f}{dt^n}=D^n f=\left(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\right)^n f$$

とおくと

$$f(a+h,\ b+k)=f(a,\ b)+\frac{1}{1!}Df(a,\ b)+\frac{1}{2!}D^2f(a,\ b)+\cdots+\frac{1}{(n-1)!}D^{n-1}f(a,\ b)+\underbrace{\frac{1}{n!}D^n f(a+\theta h,\ b+\theta k)}_{R_n} \tag{1.16}$$

を満たす θ $(0<\theta<1)$ が存在する（**テイラーの定理**）．

また，$R_n \xrightarrow{n\to\infty} 0$ であるとき

$$f(a+h,\ b+k)=f(a,\ b)+\frac{1}{1!}Df(a,\ b)+\frac{1}{2!}D^2f(a,\ b)+\cdots+\frac{1}{(n-1)!}D^{n-1}f(a,\ b)+\frac{1}{n!}D^n f(a,\ b)+\cdots \tag{1.17}$$

が成り立つ（**テイラー展開**）．

Tip　[補足]

$z=f(x,\ y)$ で，$x=a+ht,\ y=b+kt$ とすると，$z=f(a+ht,\ b+kt)$ より，1 変数 t の関数になる．微分して

$$\frac{df}{dt}=\frac{dx}{dt}\frac{\partial f}{\partial x}+\frac{dy}{dt}\frac{\partial f}{\partial y}=h\frac{\partial f}{\partial x}+k\frac{\partial f}{\partial y} \tag{1.18}$$

ここで，$D=\left(h\dfrac{\partial}{\partial x}+k\dfrac{\partial}{\partial y}\right)$ とおくと，$\dfrac{df}{dt}$ は $Df=\left(h\dfrac{\partial}{\partial x}+k\dfrac{\partial}{\partial y}\right)f$ と書ける．

次に，$\dfrac{d^2f}{dt^2}$ を考える．式 (1.18) より

$$\frac{d^2f}{dt^2}=h\frac{d}{dt}\left(\frac{\partial f}{\partial x}\right)+k\frac{d}{dt}\left(\frac{\partial f}{\partial y}\right)$$

今，$\dfrac{\partial f}{\partial x}$，$\dfrac{\partial f}{\partial y}$ は 2 変数 $x,\ y$ の関数なので，チェーン・ルールより

$$\frac{d}{dt}\left(\frac{\partial f}{\partial x}\right)=\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right)\frac{dx}{dt}+\frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right)\frac{dy}{dt}=h\frac{\partial^2 f}{\partial x^2}+k\frac{\partial^2 f}{\partial y\partial x}$$

$$\frac{d}{dt}\left(\frac{\partial f}{\partial y}\right)=\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right)\frac{dx}{dt}+\frac{\partial}{\partial y}\left(\frac{\partial f}{\partial y}\right)\frac{dy}{dt}=h\frac{\partial^2 f}{\partial x\partial y}+k\frac{\partial^2 f}{\partial y^2}$$

よって，代入して　$\dfrac{d^2f}{dt^2}=h\left(h\dfrac{\partial^2 f}{\partial x^2}+k\dfrac{\partial^2 f}{\partial y\partial x}\right)+k\left(h\dfrac{\partial^2 f}{\partial x\partial y}+k\dfrac{\partial^2 f}{\partial y^2}\right)$

第 2 次偏導関数が存在し，ともに連続とすると $\dfrac{\partial^2 f}{\partial y\partial x}=\dfrac{\partial^2 f}{\partial x\partial y}$ なので，

$$\frac{d^2f}{dt^2}=h^2\frac{\partial^2 f}{\partial x^2}+2hk\frac{\partial^2 f}{\partial x\partial y}+k^2\frac{\partial^2 f}{\partial y^2}=\left(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\right)^2 f=D^2f \tag{1.19}$$

このように，D を用いると

$$\frac{d^n f}{dt^n}=D^n f=\left(h\frac{\partial}{\partial x}+k\frac{\partial}{\partial y}\right)^n f \tag{1.20}$$

と略記することができる．

$a=b=0$ としたときのテイラーの定理を**マクローリンの定理**という．$h,\ k$ の代わりに $x,\ y$ でよく表す．

$$f(x,\ y)=f(0,\ 0)+\frac{1}{1!}Df(0,\ 0)+\frac{1}{2!}D^2f(0,\ 0)+\cdots+\frac{1}{(n-1)!}D^{n-1}f(0,\ 0)+\frac{1}{n!}D^n f(\theta x,\ \theta y) \tag{1.21}$$

を満たす θ $(0<\theta<1)$ が存在する．

◆G◆　包絡線

変数 $x,\ y$ の他に任意定数 α を含んでいる方程式

$$f(x,\ y,\ \alpha)=0 \tag{1.22}$$

は α を変化させて得られる全ての曲線の集合（曲線群）を表している．これを**曲線群の方程式**という．

曲線群の全ての曲線に接する曲線 or 直線を曲線群の**包絡線**という．包絡線上の点 $(x,\ y)$ は

$$f(x,\ y,\ \alpha)=0,\ \frac{\partial}{\partial\alpha}f(x,\ y,\ \alpha)=0 \tag{1.23}$$

を満たす．この 2 式を求めて，α を消去すると包絡線の方程式が求まる．

1.3　極値問題

◆ A ◆　2 変数関数の極値

ヘッシアン $H(a,\ b)=\begin{vmatrix} f_{xx}(a,\ b) & f_{xy}(a,\ b) \\ f_{xy}(a,\ b) & f_{yy}(a,\ b) \end{vmatrix}$ とおくと，点 $(a,\ b)$ に於いて

[1]　$H>0$ のとき

$$f_{xx}>0 \implies \text{点 } (a,\ b) \text{ で極小をとる} \qquad f_{xx}<0 \implies \text{点 } (a,\ b) \text{ で極大をとる} \tag{1.24}$$

[2]　$H<0$ のとき

点 $(a,\ b)$ では極値を取らない． (1.25)

[3]　$H=0$ のとき

極値の判定は出来ない．

Tip　[証明]

テイラーの定理より

$$f(a+h,\ b+k)-f(a,\ b)=Df(a,\ b)+\frac{1}{2}D^2f(a+\theta h,\ b+\theta k)\quad(0<\theta<1)$$

$(a,\ b)$ で極値を取るので，$Df(a,\ b)=h\cdot 0+k\cdot 0=0$ より

$$f(a+h,\ b+k)-f(a,\ b)=\frac{1}{2}D^2f(a+\theta h,\ b+\theta k)$$

となる．ここで，簡単のため

$$A=f_{xx}(a+\theta h,\ b+\theta k),\qquad B=f_{xy}(a+\theta h,\ b+\theta k),\qquad C=f_{yy}(a+\theta h,\ b+\theta k)$$

とおくと，$f(a+h,\ b+k)-f(a,\ b)=\frac{1}{2}(Ah^2+2Bhk+Ck^2)$

$h,\ k$ が十分 0 に近ければ，$AC-B^2$，A の符号はそれぞれ H，f_{xx} の符号に等しくなる．

変形して

$$f(a+h,\ b+k)-f(a,\ b)=\frac{A}{2}\left\{\left(h+\frac{B}{A}k\right)^2+\frac{AC-B^2}{A^2}k^2\right\}$$

[1]　$H>0$（即ち，$AC-B^2>0$）のとき

$f_{xx}>0$ ($A>0$) ならば，$f(a+h,\ b+k)>f(a,\ b)$（下に凸）より，極小

$f_{xx}<0$ ($A<0$) ならば，$f(a+h,\ b+k)<f(a,\ b)$（上に凸）より，極大

[2]　$H<0$（即ち，$AC-B^2<0$）のとき

（ア）　$A\neq 0$ または $C\neq 0$ のとき

$AC-B^2<0$ より，$Ah^2+2Bhk+Ck^2$ は $h,\ k$ によって正にも負にもなる．

（イ）　$A=C=0$ のとき

$AC-B^2<0$ より $B\neq 0$ なので，$Ah^2+2Bhk+Ck^2=2Bhk$ は $h,\ k$ によって正にも負にもなる．

◆ B ◆　条件付極値問題

xy 平面上の点 $(x,\ y)$ が条件 $\varphi(x,\ y)=0$ で表される曲線上を動くとき，平面 $x=f(x,\ y)$ が極値を取り得る点

$$\frac{f_x}{\varphi_x}=\frac{f_y}{\varphi_y}\quad(\varphi_x,\ \varphi_y\neq 0) \tag{1.26}$$

Tip　[導出]

方程式 $\varphi(x,\ y)=0$ の y が x の関数，即ち $\varphi\big(x,\ y(x)\big)=0$ とすると，陰関数の微分法より

$$\frac{dy}{dx}=-\frac{\partial\varphi}{\partial x}\Big/\frac{\partial\varphi}{\partial y} \tag{1.27}$$

このとき，関数 $z = f(x,\ y)$ は x の関数となるのでチェーン・ルールより，極値をとるとき

$$\frac{df}{dx} = \frac{\partial f}{\partial x}\frac{dx}{dx} + \frac{\partial f}{\partial y}\frac{dy}{dx} = 0 \tag{1.28}$$

式 (1.28) に式 (1.27) を代入して

$$\frac{\partial f}{\partial x} - \frac{\partial f}{\partial y} \cdot \frac{\dfrac{\partial \varphi}{\partial x}}{\dfrac{\partial \varphi}{\partial y}} = 0$$

整理すると得られる．

ラグランジュの未定乗数法

$f(x,\ y)$ は条件 $\varphi(x,\ y) = 0$ のもとで，$(a,\ b)$ で極値をとるとする．$\varphi_x(a,\ b) \neq 0$ または $\varphi_y(a,\ b) \neq 0$ であれば

$$\begin{cases} f_x(a,\ b) = \lambda \cdot \varphi_x(a,\ b) \\ f_y(a,\ b) = \lambda \cdot \varphi_y(a,\ b) \end{cases} \text{を満たす } \lambda \text{ が存在} \tag{1.29}$$

Tip　[導出]

式 (1.26) で，$\dfrac{f_x}{\varphi_x} = \dfrac{f_y}{\varphi_y} = \lambda$ とおくことによって得られる．

【条件付極値問題で，最大値・最小値を問われたとき】

$\varphi(x,\ y) = 0$ が端点をもたない場合，$z = f(x,\ y)$ が連続関数であれば，極値を取り得る点が最大値・最小値になる．

第 2 章　重積分法

2.1　2 重積分

◆ A ◆　定義

$$\iint_D f(x,\ y)\,dxdy = \lim_{|\Delta|\to 0} \sum_{i=1}^{l}\sum_{j=1}^{m} f(\xi_i,\ \eta_j)\,\Delta x_i \Delta y_j \tag{2.1}$$

◆ B ◆　性質

$$\left|\iint_D f(x,\ y)\,dxdy\right| \leq \iint_D |f(x,\ y)|\,dxdy \tag{2.2}$$

◆ C ◆　累次積分（逐次積分）

累次積分（1）

$D = \{(x,\ y) \mid a \leq x \leq b,\ c \leq y \leq d\}$ とする．

$$\iint_D f(x,\ y)\,dxdy = \int_a^b \left\{\int_c^d f(x,\ y)\,dy\right\}dx = \int_c^d \left\{\int_a^b f(x,\ y)\,dx\right\}dy \tag{2.3}$$

範囲に関数が含まれている場合は変数を含んでいる方を先に計算する．

累次積分（2）

[1]　$D = \{(x,\ y) \mid a \leq x \leq b,\ \varphi_1(x) \leq y \leq \varphi_2(x)\}$ とする．

$$\iint_D f(x,\ y)\,dxdy = \int_a^b \left\{\int_{\varphi_1(x)}^{\varphi_2(x)} f(x,\ y)\,dy\right\}dx \tag{2.4}$$

[2]　$D = \{(x,\ y) \mid \psi_1(y) \leq x \leq \psi_2(y),\ c \leq y \leq d\}$ とする．

$$\iint_D f(x,\ y)\,dxdy = \int_c^d \left\{\int_{\psi_1(y)}^{\psi_2(y)} f(x,\ y)\,dx\right\}dy \tag{2.5}$$

◆ D ◆　積分順序の交換

Tip　[例題 (1)]

$y = \sqrt{x}$ と $y = \dfrac{1}{2}x$ に囲まれた領域

$$\begin{aligned} D &= \left\{(x,\ y) \;\middle|\; 0 \leq x \leq 4,\ \frac{1}{2}x \leq y \leq \sqrt{x}\right\} \\ &= \{(x,\ y) \mid y^2 \leq x \leq 2y,\ 0 \leq y \leq 2\} \end{aligned}$$

Tip　[例題 (2)]

$y = \log x \iff x = e^y$ より

$$\int_1^e \left\{\int_0^{\log x} f(x,\ y)\,dy\right\}dx = \int_0^1 \left\{\int_{e^y}^e f(x,\ y)\,dx\right\}dy$$

◆ E ◆　変数変換

変数変換

$x = \varphi(u,\ v),\ y = \psi(u,\ v)$ について

$$\iint_D f(x,\ y)\,dxdy = \iint_{D'} f(\varphi,\ \psi)\left|\det\begin{bmatrix} x_u & x_v \\ y_u & y_v \end{bmatrix}\right| dudv \tag{2.6}$$

$\det\begin{bmatrix} x_u & x_v \\ y_u & y_v \end{bmatrix}$ を**ヤコビアン**といい，$\dfrac{\partial(x,\ y)}{\partial(u,\ v)}$ や $J(u,\ v)$ で表す．また，D' は D を $u,\ v$ で表しなおした領域である．

◆ F ◆　極座標変換

$x = r\cos\theta,\ y = r\sin\theta$ より

$$\begin{aligned}\iint_D f(x,\ y)\,dxdy &= \iint_{D'} f(r\cos\theta,\ r\sin\theta)\left|\det\begin{bmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{bmatrix}\right| drd\theta \\ &= \iint_{D'} f(r\cos\theta,\ r\sin\theta)\,rdrd\theta \end{aligned} \tag{2.7}$$

◆ G ◆　広義積分

領域に関数の定義されない部分が含まれている場合

Tip　[例題]

D が不等式 $x^2 + y^2 \leq 1$ の表す領域とするとき

$$\iint_D (x^2 + y^2)^{-\frac{3}{4}}\,dxdy$$

第 3 章　コンピュータアーキテクチャ

3.1　基本アーキテクチャ

◆ A ◆　基本ハードウェア構成

コンピュータは，

[1]　**プロセッサ（CPU）**
[2]　**メインメモリ**
[3]　**入出力装置**（外部装置で，キーボードとディスプレイ）

で構成させる．[1]，[2] を纏めて**内部装置**という．

3.2　内部装置のアーキテクチャ

◆ A ◆　内部装置のハードウェア構成

コンピュータの内部装置は，**プロセッサ**と**メインメモリ**の 2 つの基本ハードウェア装置を**内部バス**で接続して構成している．このうち，プロセッサは以下の 3 つの主要なハードウェア装置やハードウェア機構で構成する．

[1]　**制御機構**
[2]　**演算機構**
[3]　**レジスタ**

[2] 演算機構と [3] レジスタは**データバス**でデータを送受する．

◆ B ◆　プロセッサアーキテクチャ① 制御機構

[1]　制御機構

制御機構は，次の 2 つの機能を実現するハードウェア機構である．

[a]　**順序制御機構**　実行するマシン命令のメモリアドレスを決定する．即ちマシン命令の実行順序を決める．
[b]　**制御信号**の生成　プロセッサ更には内部装置全体の各所に，それぞれを制御するために必要なハードウェア信号を供給する．

命令コード（OP コード）　⟶　各装置に動作させる．

例　1001　⟶　{ 格納したい　⟶　レジスタに信号を送る．
　　加算したい　⟶　演算機構に信号を送る．

[2]　制御方式

命令コードから制御信号を生成することを（**命令**）**デコード**という．制御信号の生成方法は 2 通りある．

[1]　**配線論理制御方式（ワイヤードロジック）**　論理回路でデコーダをつくる．ハードウェア的制御方式なので，動作速度は速いが，変更はできない．
　単純な機能で簡潔な制御で済むマシン命令に対して適用される．
[2]　**マイクロプログラム制御方式**　命令デコーダをプログラムで表現してデコーダをつくる．ソフトウェア的制御方式なので，動作速度は遅いが，変更ができる．このマイクロプログラムは，**制御メモリ**という専用メモリに格納してある．
　高機能で複雑な制御が必要な場合に用いられる．

現代のコンピュータはこれらを併用している．

[3]　割り込み

不測の事態や事象（**割り込み要因**）が生じたときに割り込み機構が発動すると，実行中のプログラム（マシン命令列）を一時中断して，**割り込み処理**プログラムへ制御フローが分岐する．

[4] 割り込みの必要性

[1] 不測の事態や事象の対処
[2] 異常や例外の検知・対処
[3] 「ユーザプログラムから OS への通信」機能の実現
[4] 「ハードウェア装置から OS への通信」機能の実現
[5] ハードウェア同士の同期の実現
[6] 通信の競合の実現

[5] 割り込み要因

割り込みを引き起こす具体的な原因や事象を「割り込み要因の発生場所」で分類する.

[A] **内部割り込み**　要因の発生場所が内部装置特にプロセッサにある割り込み．マシン命令の実行するタイミングに合わせて発生し，マシン命令の実行というソフトウェア的要因に依るので，**ソフトウェア割り込み**ともいう．割り込みの必要性の [3] の手段.

(1) 命令実行例外

- **メモリアクセス例外**
  - **(i)** 指定したメモリアドレスにマシン命令やデータがないとき
  - **(ii)** **ページフォールト**…プログラムを仮想メモリ[1]に割り当てていてメインメモリにはまだ読み込んでいない状態で，プログラムにアクセスしようとしたとき
  - **(iii)** メモリ保護違反…アクセスする権限がないメインメモリへアクセスしようとしたとき
- **不正命令**
  - **(i)** 命令セットにない（＝定義されていない）命令コードであるとき
  - **(ii)** データなのに命令として実行しようとしたとき
- **不正オペランド**　オペランドで指定するアドレスにデータがないとき
- **演算例外**
  - **(i)** **オーバーフロー**[2]を起こしたとき
  - **(ii)** **0 除算**…0 を除数とする除算を行ったとき

(2) SVC（スーパーバイザコール）　OS を呼び出す SVC 命令を実行したとき．SVC 命令は以下の通り：

- **入出力命令**…ディスプレイやプリンタなど
- **ブレークポイント命令**…プログラム実行の中断点（ブレークポイント[3]）を OS に通知する.

[B] 外部割り込み　要因の発生場所が外部装置特に入出力装置にある割り込み．マシン命令の実行とは無関係に発生し，外部装置というハードウェア的要因に依るので，**ハードウェア割り込み**ともいう．割り込みの必要性の [4] の手段.

(1) 入出力割り込み　入出力装置・通信装置から OS に次の状態を知らせる割り込み
- **(i)** ユーザが入力装置によってデータを正常に入力した
- **(ii)** 出力装置の操作が正常に完了した
- **(iii)** 入出力装置に異常があった
- **(iv)** 通信装置を使って他のコンピュータが通信を要求した

(2) ハードウェア障害　ハードウェア装置から以下のような「障害発生」の通知
- **(i)** 電源異常
- **(ii)** メモリからの読み出しエラー
- **(iii)** 温度異常

(3) リセット　ユーザが電源ボタンを押したとき

複数の割り込みが発生した時，優先度を付与する.

【優先度】（高→低）

- ハードウェア障害
- リセット
- 命令実行例外
  - ページフォールト
  - メモリ保護違反
  - 演算例外
- 入出力割り込み

---

1) メインメモリ＋ファイル装置で作った仮想的なメモリ.
2) 演算結果が演算器そのものやレジスタの最上位ビットから溢れる場合.
3) デバグ（プログラムの誤りを発見・削除する操作）時にプログラム実行を一時中断する起点，プログラムをトレース（実行履歴を取る）する起点.

- ○ ファイル装置（高速）
- ○ キーボード・プリンタ

● SVC

● ブレークポイント命令

［6］ 割り込み処理

順序制御機構と OS 機能とが機能分担して処理する.

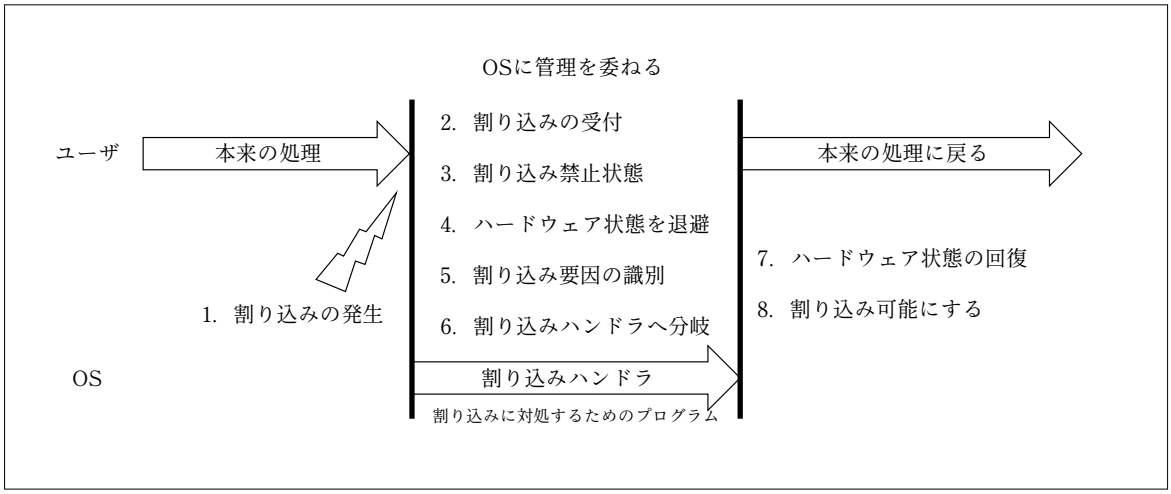

図 3.1: 割り込み処理の流れ

1. 割り込みの発生…3.2.B 節の［5］で紹介した要因で割り込みが発生.
2. 割り込みの受付…ハードウェア機構が割り込みを受付→割り込み処理開始.
3. 割り込み禁止状態…他の割り込みを受け付けないようにする.
4. ハードウェア状態を退避…ハードウェア機構によって今までのプログラムが生成していたハードウェア状態をプロセッサ内の退避領域へ退避.
5. 割り込み要因の識別…割り込み要因は何かを識別.
6. 割り込みハンドラへ分岐…割り込み要因ごとの割り込み処理プログラムへ分岐.
7. ハードウェア状態の回復…退避領域から今までのプログラムが生成していたハードウェア状態を回復.
8. 割り込み可能にする…他の割り込みを許可する.

［7］ 命令パイプライン処理

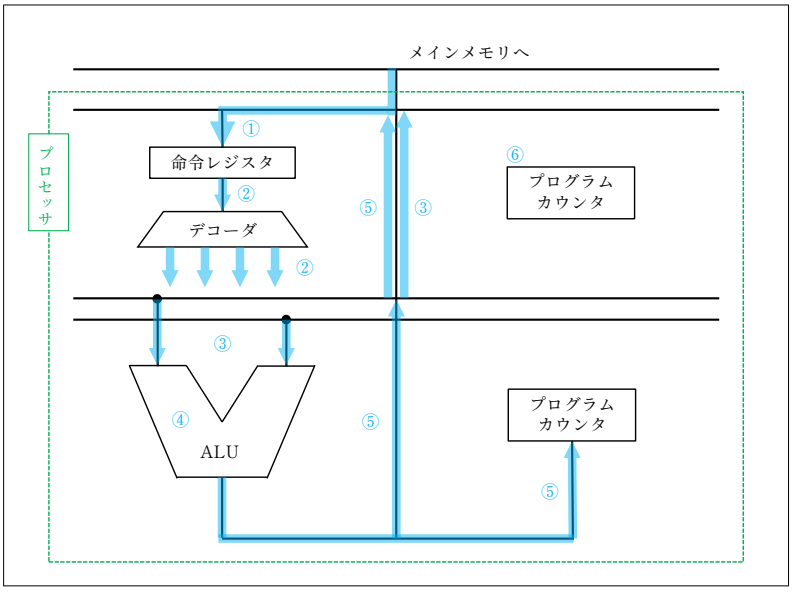

図 3.2: 命令実行サイクルの流れ

命令実行サイクルは以下の通り.

① 命令取り出し（I）
② 命令デコード（D）
③ オペランド取り出し（O）
④ 実行（E）
⑤ 結果格納（W）
⑥ 次アドレス決定

上の命令実行サイクルの 1 つ 1 つを**ステージ**という．この 1 連のマシン命令をプロセッサに投入して処理するとき，高速化するため**命令パイプライン処理**をする．命令パイプライン処理の具体的な流れは以下の通り．

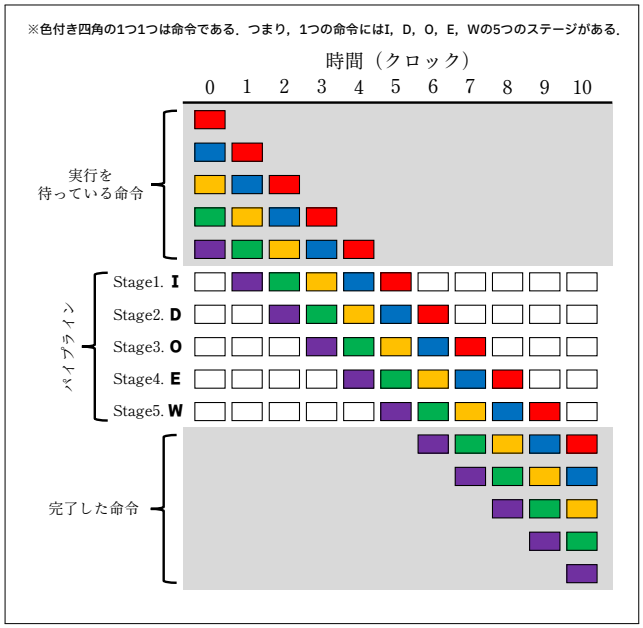

図 3.3: 命令パイプライン処理の流れ

表 3.1: 命令パイプライン処理の流れ

時間（クロック）	実行内容
0	5 つ（赤，青，黄，緑，紫）の命令が実行されるのを待っている．
1	紫色の命令を取り出す．
2	紫色の命令をデコードする． 緑色の命令を取り出す．
3	紫色の命令のオペランドを取り出す． 緑色の命令をデコードする． 黄色の命令を取り出す．
4	紫色の命令を実行する（実際の命令処理を行う）． 緑色の命令のオペランドを取り出す． 黄色の命令をデコードする． 青色の命令を取り出す．
5	紫色の命令の結果を（レジスタなどに）格納する． 緑色の命令を実行する． 黄色の命令のオペランドを取り出す． 青色の命令をデコードする． 赤色の命令を取り出す．
6	紫色の命令は完了した． 緑色の命令の結果を格納する． 黄色の命令を実行する． 青色の命令のオペランドを取り出す． 赤色の命令をデコードする．
7	緑色の命令は完了した． 黄色の命令の結果を格納する． 青色の命令を実行する． 赤色の命令のオペランドを取り出す．
8	黄色の命令は完了した． 青色の命令の結果を格納する． 赤色の命令を実行する．
9	青色の命令は完了した． 赤色の命令の結果を格納する．
10	赤色の命令は完了した＝**全命令を実行した．**

このように，5 つのマシン命令が完了するまでに 10 サイクルの時間が必要になる．

Tip　[例題]

上の図のように 5 ステージで構成する命令実行サイクルを持つ命令パイプライン処理機構がある．ただし，1 ステージ時間は 10 ナノ秒とする．即ち，1 命令実行サイクルは 50 ナノ秒要する．このパイプライン機構に 100,000 個のマシン命令を投入すると，全部のマシン命令が完了するまでにはどれだけ時間がかかるか求めなさい．また，命令パイプライン処理機構がない場合と比較しなさい．

パイプライン処理をしない場合，1 つの命令につき 50 ナノ秒なので，100,000 個の命令では $50 \times 100000 = 5000000$ ナノ秒，つまり，5 ミリ秒かかる．
パイプライン処理をする場合，まず，1 個目の命令が Stage1. に入るまでから 100,000 個目の命令が Stage1. に入るまで 10 ナノ秒ずつずれていくわけなので，10 ナノ秒 $\times 100000 = 1000000$ ナノ秒 掛かる．そのあと，100,000 個目の命令は残り 4 つのステージが残っているわけだから，10 ナノ秒 $\times 4 = 40$ ナノ秒．
つまり，合計 1000000 ナノ秒 $+40$ ナノ秒 $= 1000040$ ナノ秒 $= 1.00004$ ミリ秒 ≈ 1 ミリ秒 掛かることになる．つまり，しない場合の約 5 分の 1 で完了できる．

この例題は上の図で，「時間（クロック）」が「時間（ナノ秒）」となり，「0, 10, 20, $\cdots$, 1000040」と置き換え，100,000 色の四角（命令）があると考えた場合である．

このように，命令実行サイクルを s 個のステージで構成する命令パイプライン処理機構に I 個のマシン命令を投入すると，処理時間は I の値にかかわらず，パイプライン処理をしない場合の**約 s 分の 1** に短縮できる．
※例題の 5 ステージに対し 100,000 個の命令のように，ステージ数に対し命令数が十分大きくではならない（$I \gg s$）．でないと処理時間はそんなに変わらない．

◆ C ◆　プロセッサアーキテクチャ② 演算機構

[1]　数の表現

表 3.2: 数の表現

整数	**固定小数点数表現** 小数点は最下位ビットの右に固定．
実数	**浮動小数点数表現** 小数点の位置を変えて任意の精度で表現する．小数点も“0”か“1”で表現する．問題として，ハードウェアは有限なので**循環小数**や**無理数**の**丸め誤差**が生れる．

[2]　演算機構とデータバス

マシン命令の転送とデータの転送の両方で共用する内部バスに対し，データ専転送だけに使用する演算機構 – レジスタ間の転送路を**データバス**という．

演算器（ALU）は，データバスに並列接続する．

[3]　加算器

四則演算は**加算**を基にしてできる．

- **減算**　加える数を負の数にする．
- **乗算**　加算の繰り返し：$a \times b$ の場合，0 を最初の加えられる数にして，a を b 回繰り返し加算する．
- **除算**　減算の繰り返し：$a \div b$ の場合，a を最初の引かれる数にして，b を繰り返し引いたとき，b 未満になるまでに何回引いたかが商（計算結果）となる．

加算器で，1 個下のビットからの繰り上げも考慮したものを**全加算器**という．足される数を X，足す数を Y，1 個下のビットからの繰り上げを C_{in}，当該ビットの和を S[4]，1 個上のビットへの繰り上げを C_{out} とすると，真理値表は以下の通り．

1 個下のビットからの繰り上げ出力を当該ビットの C_{in} 入力として，32 個の全加算器を直列に繋げば「32 ビット加算器」が構成出来る．このとき，最下位ビットの繰り上げ C_{out_0} が 32 個の全加算器を最上位ビットからの繰り上げ $C_{\text{out}_{31}}$ まで，次々に伝播することによって累積する遅延が加算時間を決める．この加算器を**桁上げ伝播加算器**という．

よって，加算時間を減らすため以下の 2 つの加算器を現代のコンピュータは装備している．

[1]　**桁上げ先見加算器**　各ビットでの加算数・被加算数を使って予め和を出しておく．桁上げ先見回路という専用の装置を使う．
[2]　**桁上げ保存加算器**　部分加算を行い，出た部分和の桁上げ情報を保存する．

4)　例えば，2 進数の $(1)_2 + (1)_2$ をすると，$(10)_2$ となる．このとき，S は 0，C_{out} は 1 である．

表 3.3: 全加算器の真理値表

X	Y	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

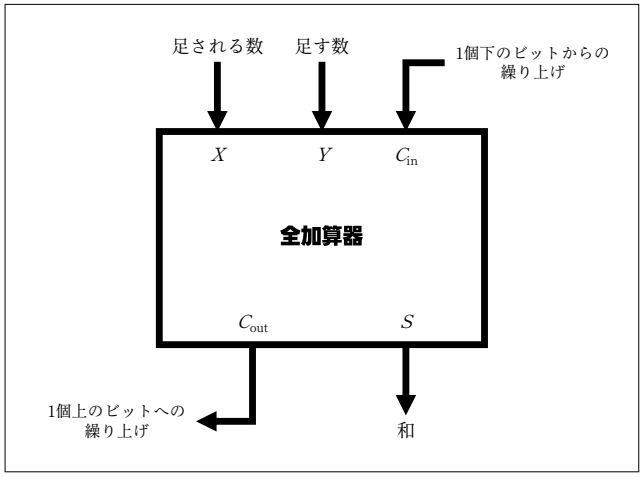

図 3.4: 全加算器のブロック図

[4]　負数の 2 進数表現 – 補数表現

負数を 2 進数で表すとき，“+”，“−”符号も “0”，“1” で表す必要がある．負整数の 2 進数表現として，以下の 2 通りがある．

[1]　1 の補数 $\overline{N}$

N に加えると和が $1\cdots11$ になる数.

1 の補数は 2 進数の正整数の各ビットを反転させる（“0” ⇔ “1”）と求められる．m ビットの 2 進数を 10 進数で表したとき，以下の式になる．

$$\overline{N} = 2^m - 1 - N \tag{3.1}$$

[2]　2 の補数 $\overline{\overline{N}}$

m ビットの N に加えると和が $1\underbrace{0\cdots00}_{m\ \text{ビット}}$ になる数.

2 の補数は 1 の補数に $+1$ すると求められる．m ビットの 2 進数を 10 進数で表したとき，以下の式になる．

$$\overline{\overline{N}} = 2^m - N = \overline{N} + 1 \tag{3.2}$$

Tip　[例題]

$N = (0111)_2$ の 1 の補数 $\overline{N}$ と 2 の補数 $\overline{\overline{N}}$ を求めなさい.

1 の補数：

機械的にする場合	$N = (0111)_2$	$\overline{N} = (1000)_2$
10 進数に直して考える場合	$N = (0111)_2 = (7)_{10}$	$\therefore \overline{N} = 2^4 - 1 - 7 = (8)_{10} = (1000)_2$

2 の補数：

機械的にする場合	$N = (0111)_2$	$\overline{\overline{N}} = (1001)_2$
10 進数に直して考える場合	$N = (0111)_2 = (7)_{10}$	$\therefore \overline{\overline{N}} = 2^4 - 7 = (9)_{10} = (1001)_2$

符号無しでは，4 ビット $(0000)_2$〜$(1111)_2$ の 2 進数が表現出来る範囲は $(0)_{10}$〜$(15)_{10}$ である．しかし，補数表現を使うと最上位ビットは符号ビット（“0”：正数，“1”：負数）となり，表現できる範囲は $(-8)_{10}$〜$(+7)_{10}$ となる．

表 3.4: 4 ビット 2 進数の 2 の補数表現に於ける 10 進数との対応表

対応する 10 進数	-8	-7	-6	-5	-4	-3	-2	-1	0	$+1$	$+2$	$+3$	$+4$	$+5$	$+6$	$+7$
2 進数	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

[5]　補数による加算

[1]　1 の補数を使うとき

1. 負数があるときは 1 の補数にする．
2. 2 つの値を加算する．
3. **2.** の結果，最上位ビットで繰り上がり（エンドキャリ）が発生したら，**2.** に $+1$ をした後，最上位ビットの繰り上がりは無視する．この補正を**エンドアラウンドキャリ（循環桁上げ）**という．
4. その結果，最上位ビットが“0”であれば正数，“1”であれば 1 の補数表現で示された負数である．

> **Tip**　[例題]
>
> $(-4)_{10}+(-2)_{10}$ を 1 の補数によって加算せよ．
>
> 1. どちらも負数なので，1 の補数にする．
>
> $$(-4)_{10}=\overline{(0100)_2}=(1011)_2$$
> $$(-2)_{10}=\overline{(0010)_2}=(1101)_2$$
>
> 2. $(1011)_2+(1101)_2=(1\,1000)_2$
> 3. エンドキャリが発生したので，$+1$ すると $(1\,1001)_2$．繰り上がりそのものは無視して結果は $(1001)_2$．
> 4. 最上位ビットが“1”なので，これは 1 の補数で示された負数である．よって，符号を反転させて $(0110)_2=(+6)_{10}$ なので，答えは $(-6)_{10}$ である．

[2]　2 の補数を使うとき

1. 負数があるときは 2 の補数にする．
2. 2 つの値を加算する．
3. **2.** の結果，最上位ビットで繰り上がり（エンドキャリ）が発生したら無視する．
4. その結果，最上位ビットが“0”であれば正数，“1”であれば 1 の補数表現で示された負数である．

> **Tip**　[例題]
>
> $(-4)_{10}+(-2)_{10}$ を 2 の補数によって加算せよ．
>
> 1. どちらも負数なので，2 の補数にする．
>
> $$(-4)_{10}=\overline{\overline{(0100)_2}}=(1100)_2$$
> $$(-2)_{10}=\overline{\overline{(0010)_2}}=(1110)_2$$
>
> 2. $(1100)_2+(1110)_2=(1\,1010)_2$
> 3. エンドキャリが発生したので無視して結果は $(1010)_2$．
> 4. 最上位ビットが“1”なので，これは 2 の補数で示された負数である．よって，符号を反転後 $+1$ して $(0110)_2=(+6)_{10}$ なので，答えは $(-6)_{10}$ である．

2 の補数による加算では 1 の補数と違って補正は必要無くエンドキャリは無視すればいいだけなので都合がよい．よって，現代のコンピュータでは減算・負数の加算は 2 の補数で行うのが一般的．

[6]　乗算器と除算器

- **乗算器**を実現させる仕組み
  - **(1)**　ブース法…負の数を 2 の補数で表現し，部分積を足して実現．
  - **(2)**　配列型乗算器（並列乗算器）…部分積を 2 次元配列状に並べ，全加算器で同時に加算．
  - **(3)**　ウォリス木…tree 構造（根付き木）に部分積を格納．
  - **(4)**　早見表…基本乗算の結果を表にして部分積を求める．
- **除算器**を実現させる仕組み

(1)　引き戻し法…部分商を求め，結果 0 なら減算[5]を無かったことにする．
(2)　引き放し法…部分商が 0 のとき，当該ビットでの減算はそのまま次のビットの操作に移行．
(3)　乗算収束型除算法
(4)　部分商の同時生成

[7]　実数の表現

実数 R は**浮動小数点数表現**を用いる（表 3.2）．小数点の左側 m を**仮数部**，右側 e を**指数部**といい，式では以下のように表す．

$$R = m \times 2^e \qquad (m \text{ も } e \text{ もコンピュータ内部では 2 進数}) \tag{3.3}$$

m や e が負数であれば補数表現で表される．このとき，m が負であれば R は負となる．

m は**有効数字**ともいい，e は小数点の位置決めをしている．よって，e は m が整数となるように調整している．言い換えれば，実数を 2 つの整数 m, e によって表現する．

> **Tip**　[例題]
> 2 進実数 $(10100)_2$, $(11.011)_2$, $(0.1111)_2$ を浮動小数点数表現せよ．
>
> $(10100)_2 = (101)_2 \times 2^{(10)_2}$
> $(11.011)_2 = (11011)_2 \times 2^{(-11)_2}$
> $(0.1111)_2 = (1111)_2 \times 2^{(-100)_2}$
>
> ※ここでは，負数を絶対値に“–”符号を付けることによって表しているが，実際のコンピュータ内部では補数表現されている．

$R_1 = m_1 \times 2^{e_1}$ と $R_2 = m_2 \times 2^{e_2}$ の積 $R_1 \times R_2$ は次のように計算できる．

$$R_1 \times R_2 = (m_1 \times m_2) \times 2^{(e_1+e_2)} \tag{3.4}$$

よって，浮動小数点数用の演算器では，乗算器と加算器を同時に行う工夫をしている．

[8]　論理演算器とシフト演算器

論理演算器では，ビットの 1 つ 1 つが論理値であるので並列で論理演算できればよい．対して，**シフト演算器**では，上下位のビットを隣り合うビットに移動する操作を行うので論理回路が必要である．

[9]　演算機構のアーキテクチャ

演算器（ALU）は

[1]　補数器
[2]　整数の加算器
[3]　整数の乗算器
[4]　整数の除算器
[5]　浮動小数点数の加算器
[6]　浮動小数点数の乗算器
[7]　浮動小数点数の除算器
[8]　論理演算器とシフト演算器シフト演算器
[9]　2 進数→ 10 進数変換機構
[10]　10 進数→ 2 進数変換機構

で構成される．

◆ D ◆　メモリアーキテクチャ

[1]　メインメモリ（内部メモリ）

メインメモリ（**内部メモリ**）は，プロセッサと共にコンピュータの内部装置を構成するものである．メインメモリとプロセッサで，「プロセッサで処理するマシン命令とデータを予めメインメモリに用意しておく」という**プログラム内蔵**の実装を実現している．

メインメモリは以下の機能を備え，その機能に求められる評価がある．

---

5)　除算は減算の繰り返しである．

表 3.5: メインメモリに求められる機能と評価

機能	評価
格納	容量が大きいほど嬉しい
アクセス	アクセスが速いほど嬉しい

現代のコンピュータのメインメモリは

[1] **ランダムアクセス**　アドレスと格納場所が 1 対 1 であり，一定時間でアクセスできる．謂わば，アドレスを指定すれば計算量 $O(1)$ で即アクセスできる（配列に近いイメージ）．
[2] **線形アドレス**　メモリ空間を決まった空間で区切り，連番のアドレスを付ける．つまり，アドレスを $+1$ ずつ増加させるだけで順番にアクセスできる．

の 2 つの要件を満たさなければならない．

[2] メインメモリへのアクセス

プロセッサがメインメモリにアクセスするときは，アドレスを指定し，読み出しと書き込みのどちらのアクセスかを指定する必要がある．その為に次の 2 つの機構をプロセッサ側に備える．

[1] **メモリアドレスレジスタ（MAR）**…メインメモリアドレスをアクセスする間保持しておく．
[2] **メモリデータレジスタ（MDR）**……アクセスするマシン命令やデータをアクセスする間保持しておく．

プロセッサ内の MAR，MDR を備える機構を**メインメモリ管理機構（MMU）**という．

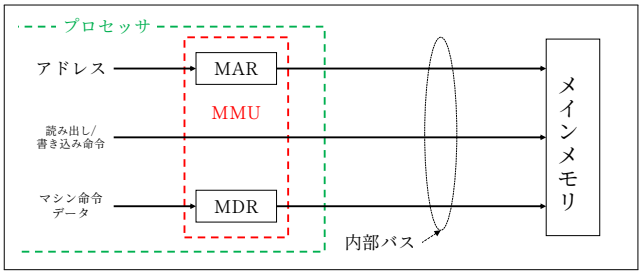

図 3.5: プロセッサによるメインメモリへのアクセス

[3] メモリの階層構造

メモリ素子とは，“0”か“1”を格納出来るメモリの最小単位（1 ビット）である．メモリ素子は以下のハードウェアで構成される．

[1] **半導体**　電流の ON/OFF を切り替える．
[2] **磁性体**　磁場の向きで“0”と“1”を記録する．
[3] **コンデンサ**　電荷を蓄える．

メモリ素子によって表 3.5 の評価が決まるが，容量が大きいこととアクセス時間が短いことは**両立しない**．つまり，どちらの性能も高いメモリは実在しない．結果として，適材適所を図ることが必要．

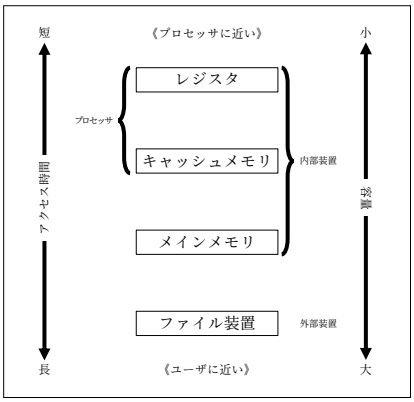

図 3.6: 主要なメモリ階層

[4] 参照局所性

実行中のプログラムがアクセス又は参照するマシン命令やデータの格納場所（アドレス）が一部或いは特定の場所に集中することを「**参照局所性**が高い」という．空間的と時間的の 2 種類がある．

[1] 「空間的参照局所性が高い」とは，一度アクセスしたアドレスの“格納場所に近いアドレス”が近いうちにアクセスする可能性が高いということ．

[2] 「時間的参照局所性が高い」とは，一度アクセスしたアドレス“そのもの”が近いうちにアクセスする可能性が高いということ．

従って，参照局所性が高いプログラムを上の階層におけば，よりプロセッサに近くなり時間的・空間的に効率よく処理出来るようになる．

[5] 仮想メモリ (1) – 原理

メインメモリには，プロセッサが実行しようとするプログラムやデータを格納しておく，つまりプログラム内蔵である必要がある．しかし，プログラムが巨大で一時的にメインメモリに入らないとき，取り敢えずそれ以外のメモリに格納しておいて必要なときに必要なプログラムをメインメモリに持ってこなくてはいけない．もしくは，複数のプログラムを同時に行うとき，個々のサイズは小さくても数が多過ぎてメインメモリに入りきらないときがある．

このとき，ユーザが作成したプログラムをファイルとして**ファイル装置**に格納しておき，使用する可能性があるときにメインメモリに転送，そしてプロセッサが実行するときにアクセスするようにする．

このように，使用する可能性が高い＝参照局所性が高いファイルだけを置いて残りはファイル装置に置くことで「プロセッサから見えるメインメモリの容量を見かけ上大きくする機能」を**仮想メモリ**方式という．

実装しているメインメモリに付いている物理的なアドレスを**実アドレス**[6]，マシン命令中のオペランドから生成する実効アドレスを**仮想アドレス**という．

仮想メモリは，次の 2 種類のメモリ＝アドレス空間を独立に構成し，相互に対応付ける（**マッピング**）ことによって実現する．

[1] **実メモリ** 実アドレスで指定するアドレス空間．即ち，メインメモリのアドレス空間[7]である．

[2] **仮想メモリ** 仮想アドレスで指定するアドレス空間．即ち，仮想の／論理的なメモリのアドレス空間である．

仮想メモリを実現すると以下のメリットを得られる．

- 実メモリであるメインメモリの利用効率が良くなる；
- 実メモリであるメインメモリの容量による種々の制約を事実上撤廃出来る；

[6] 仮想メモリ (2) – 仮想メモリ機構

仮想メモリ機構に於けるメインメモリへのアクセスは，MMU 内で以下のように実行する．

1. マシン命令のオペランドが示している仮想アドレス（実効アドレス）を含むブロック（マシン命令やデータを一定の範囲で区切った塊）がメインメモリ上にあるかチェックする．
2. (a) **ある場合** マッピングに従って，仮想アドレスから対応する実アドレスへ変換する．
   (b) **ない場合**（**ページフォールト**）
   (1) メインメモリ上にある不要なブロックをファイル装置へ追い出し，代わりに仮想アドレスで指定した命令やデータを含むブロックをファイル装置からメインメモリに入れる（**ブロック置換**）．
   (2) 置換後のマッピングに従って，仮想アドレスから対応する実アドレスへ変換する．

[7] 仮想メモリ (3) – マッピング

仮想アドレスと実アドレスのマッピングは**マッピングテーブル**（**アドレス変換テーブル**）に記述してある．

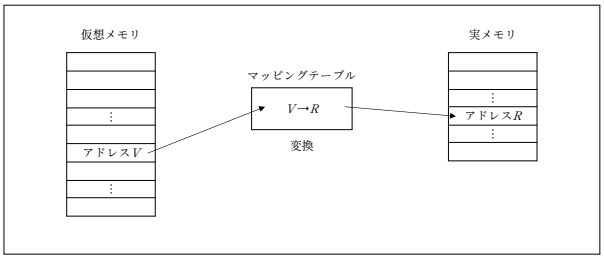

図 3.7: マッピング

6) メインメモリで格納されているマシン命令やデータの格納場所を指定するために，メインメモリはアドレス付けをしている．
7) アドレス付けしたメインメモリの格納領域をメインメモリのアドレス空間という．

マッピングには次の 3 つの方式がある．

(A)　ページング

ページという一定サイズで区切ったブロック単位でマッピングする．仮想／実アドレス空間を論理的な意味を無視して一定のサイズに区切る．→参照局所性が保存されにくい．

仮想アドレス空間にはページごとに「仮想ページ番号」を，実アドレス空間にはページごとに「実ページ番号」を割り振り，仮想ページと実ページのマッピングを**ページテーブル**に記述する．

アクセスを要求した仮想ページがメインメモリに無い場合は，**ページフォールト割り込み**が生じる．

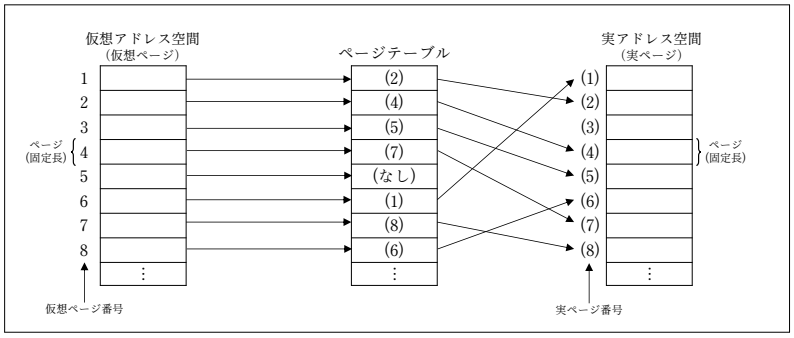

図 3.8: ページング

(B)　セグメンテーション

セグメントという論理的意味を持つブロックで区切って，仮想セグメントと実セグメントをマッピングする．その為，サイズは可変．

仮想アドレス空間にはセグメントごとに「仮想セグメント番号」を，実アドレス空間にはセグメントごとに「実セグメント番号」を割り振り，仮想セグメントと実セグメントのマッピングを**セグメントテーブル**に記述する．

アクセスを要求した仮想セグメントがメインメモリに無い場合は，**セグメントフォールト割り込み**が生じる．

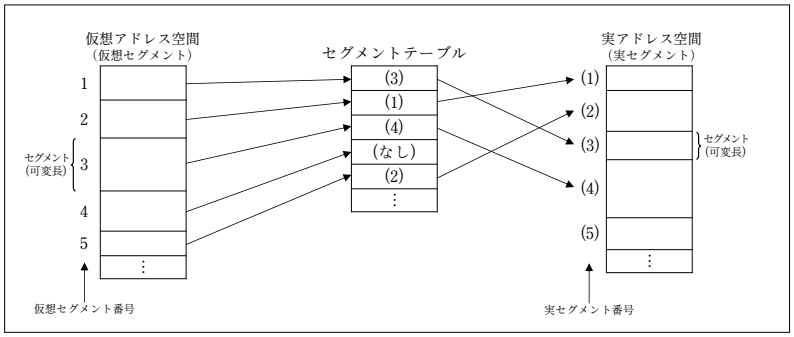

図 3.9: セグメンテーション

上の図 3.9 でブロック置換を行うとする．メインメモリ（実アドレス空間）の (3) のブロックを追い出したとき，(3) よりサイズが大きいブロックは格納出来ない．よって，セグメンテーションではメインメモリに無駄が発生しやすい．

(C)　ページセグメンテーション

ページングとセグメンテーションを併用するマッピング方式．次の順で適用する．

1. まず，全体では**セグメント**単位で，仮想セグメントと実セグメントを「セグメントテーブル」でマッピングする．
2. 次に，各セグメント内では**ページ**単位で，仮想ページと実ページを各セグメントごとに備える「ページテーブル」でマッピングする．

こうすることにより，外側ではセグメンテーションにより論理的意味を持つブロックに分かれているが，その中ではページングにより一定のサイズで分割されているので，

- セグメンテーションの長所である，論理的意味で区切るので参照局所性が保存される；
- マッピングの長所である，一定のサイズで区切るので無駄な空間が生じにくい；

というそれぞれの長所がページセグメンテーションでは引き継がれる．

ページセグメンテーションに於けるアドレス変換の手順は以下の通りになる．

1. 「セグメントテーブル」を引くことによって，仮想セグメント番号（図 3.10 では “9”）から「9 で備えられているページテーブル」を得る．
2. 「**1.** で得たページテーブル」を引くことによって，仮想ページ番号から実ページ番号を得る．

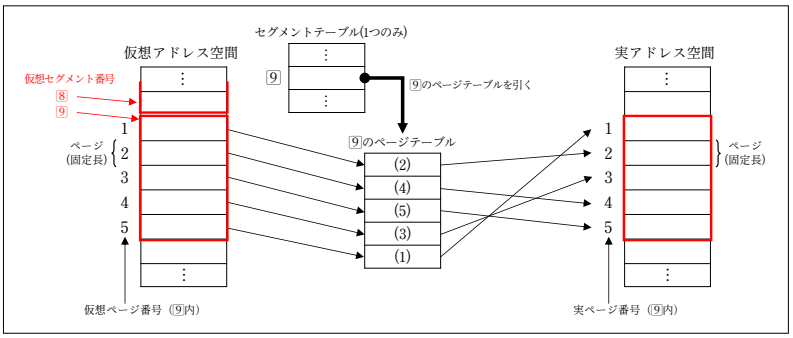

図 3.10: ページセグメンテーション

[8]　仮想メモリ (4) – ページ置換アルゴリズム

ページ置換では，使用頻度＝参照局所性が高いページをメインメモリで保持し，参照局所性が低い実ページをファイル装置へ追い出す．このとき，「メインメモリにあるどのページを追い出すか」を決める方法を**ページ置換アルゴリズム**という．

最も参照局所性が低いページを決定する方法に以下がある．

[1]　**LRU**（Least Recently Used）
　　最後のアクセス時刻が最も古いページを最優先で追い出す．

[2]　**FIFO**（First In First Out）
　　メインメモリに一番最初に読み込んだページを最優先で追い出す．

[3]　**ランダム**
　　任意或いは無作為に選んだページを追い出す．

[9]　キャッシュメモリ (1) – 原理

メインメモリがプロセッサの要求に応じてマシン命令やデータをアクセスするとき，単純に速いプロセッサと，容量とアクセス時間のバランスをつ必要があるメインメモリでは動作速度に差が出来てしまう．

→キャッシュメモリというメモリを備えている．メモリ階層では，キャッシュメモリ階層はレジスタ階層とメインメモリ階層の中間にある（図 3.6）．

メインメモリが保持するプログラムのうち，実際に使用中＝参照局所性が高いマシン命令やデータ乃至(ないし)その一部やコピーをキャッシュメモリに一時的に保持する．キャッシュメモリのデータのやり取りの単位を**ライン**という．

命令実行サイクル（3.2.B 節［7］を参照）の各ステージで，要求されるデータがキャッシュメモリに

(a)　あるとき（**ヒット**という）
　　キャッシュメモリから命令やデータを取り出して提供する．

(b)　ないとき（**ミスヒット**という）
　　→キャッシュメモリの比較的今後アクセスされないラインを追い出して目的の命令やデータが含まれるラインを空いたところに格納する（**ライン置換**）．

[10]　キャッシュメモリ (2) – 処理時間を求める

キャッシュメモリがあるときの実質的なメインメモリへのアクセス時間

- 実質的なメインメモリへのアクセス時間 T
- キャッシュメモリに命令やデータがある確率（**ヒット率**）R_c
- キャッシュメモリに命令やデータがない確率 $R_m = 1 - R_c$
- キャッシュメモリへのアクセス時間 T_c
- メインメモリへのアクセス時間 T_m

とすると

$$T \quad = \quad R_c T_c + R_m T_m \quad = \quad R_c T_c + (1 - R_c) T_m \tag{3.5}$$

［11］ キャッシュメモリ (3) – メインメモリとのマッピング

キャッシュメモリとメインメモリとのマッピングは，参照局所性を活用するために**ライン**単位で行う．マッピングは**キャッシュタグ**に記述しておく．マッピングには次の 3 つの方式がある．尚，キャッシュメモリのライン総数を L_C とする．

(A)　ダイレクト

「メインメモリの あるライン をキャッシュメモリの どのライン にマッピングするか」を予め固定しておく．

→探索の必要がない＆処理が速い．

メインメモリの第 i 番目のラインをキャッシュメモリの第 j 番目のラインにマッピングするには，例として $j \equiv i \pmod{L}_C$ がある[8)]．

(B)　フルアソシアティブ

メインメモリの 各ライン をキャッシュメモリの どのライン にも自由にマッピング出来る．

→探索の必要がある＆キャッシュタグ検索に時間がかかるがライン置換の頻度を抑えられる．

(C)　セットアソシアティブ

一定数個のキャッシュラインを**セット**にし，

- メインメモリのラインとセットとのマッピングはダイレクト；
- 各セット内のラインのマッピングはフルアソシアティブ；

を適用する．

［12］ キャッシュメモリ (4) – ライン置換アルゴリズム

ライン置換では，メインメモリへ追い出す或いは廃棄するキャッシュラインを決める必要がある（内容が更新されている場合は追い出し，されていない場合は廃棄する）．このとき，「どのラインをキャッシュメモリからメインメモリへ追い出したり廃棄したりするか」を決める方法を**ライン置換アルゴリズム**という．その方法は次の 5 つがある．

[1]　**LRU**（Least Recently Used）
最後のアクセス時刻が最も古いラインを最優先で追い出す．

[2]　**FIFO**（First In First Out）
キャッシュメモリに一番最初にコピーしたラインを最優先で追い出す．

[3]　**FINUFO**（First In Not Used First Out）
一定時間アクセスのないラインのうち，最初にキャッシュメモリにコピーしたラインを追い出す．

[4]　**LFU**（Least Frequently Used）
アクセス頻度がいちばん低いラインを追い出す．

[5]　**ランダム**
任意或いは無作為に選んだラインを追い出す．

※ **[1]**，**[2]**，**[5]** はページ置換（3.2.D 節［8］）と同じアルゴリズム．

---

8)　$j \equiv i \pmod{L}_C$ は，**モジュロ演算**といい，「i を L_C で割った余りと j を L_C で割った余りが等しい」ことを表している．例）$6 \equiv 2 \pmod{4}$