5.3　アセンブラ

コンピュータが直接理解できるのは**機械語**で、今でも組み込み式コンピュータなど限られた部分ではありますが使用されています。また、コンピュータが登場した当時はこの言語を使ってプログラムを作成していましたが、この言語は直接コンピュータの資源を利用するためプログラムの追加・削除・変更が容易ではありませんでした。その後、機械語と 1 対 1 に対応した**アセンブラ言語**が登場し、プログラムを実行する直前に機械語に変換したので、機械語の欠点が改善されました。しかしながら、この言語も中央処理装置の違いによってアセンブラ言語が異なっていて、本テキストで学習する C 言語のように汎用性に富んではいませんでした。その後、汎用性に富んだ言語が次々に現れることになります。

情報処理試験で用いられる仮想のコンピュータを例に挙げ、コンピュータの計算の仕組みを詳しく見ていきましょう。COMET II と呼ばれる 5 大要素を備えた仮想のコンピュータとその上で動く CASL II と呼ばれるアセンブラ言語を用います。ノイマン型のコンピュータの制御方式は**逐次制御**で「**フェッチ**(命令の取り出し) $\Longrightarrow$ **デコード**(命令の解読)$\Longrightarrow$ 実行」を順番に繰り返すことによって動作しています。これらを実現するためにコンピュータの基本構成は下図のようになっています。

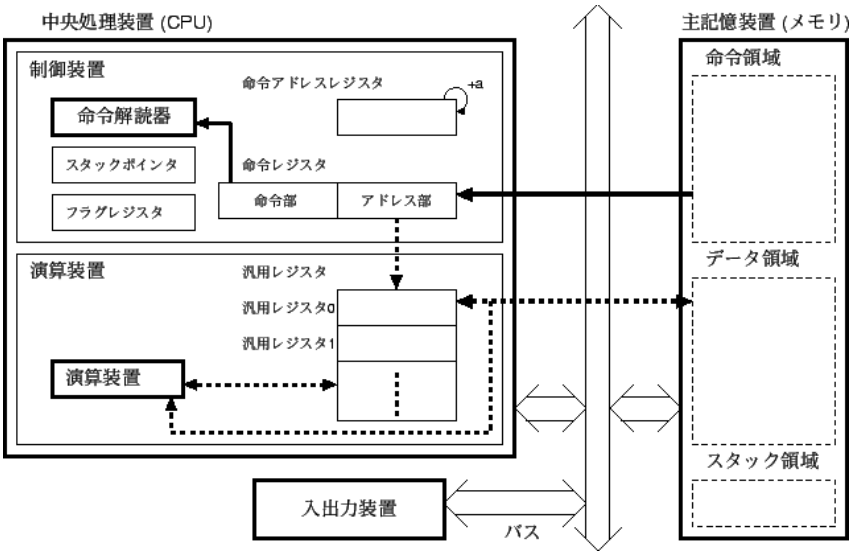

図 5.4: 計算機内部の模式図

主記憶装置　**主記憶装置**はノイマン型のコンピュータの特徴で、命令とデータを記憶するところです。主に命令領域とデータ領域に分れますが、その他にスタック領域として使用します。COMET II では 1 語は 16 ビット (2 進数で 16 桁) で表され、アドレスは 16 ビット (0000(16) から FFFF(16)) で表現できる 65536 語を記憶します。

命令解読器　**命令解読器**は制御装置の中心的存在で命令レジスタの命令部を解読し、各装置に命令を出します。命令デコーダとも呼ばれます。

演算装置　**演算装置**では記憶装置のデータや汎用レジスタの値を用いて計算を行います。また、命令レジスタのアドレスの計算なども行います。

命令アドレスレジスタ 命令アドレスレジスタは次にどこからプログラムを進めれば良いのか位置 (アドレス) を記憶しています。命令の語の長さによって加数は変化します。

命令レジスタ 命令レジスタは命令を記憶する場所で命令部とアドレス部に分れています。また、命令部は命令解読器に送られアドレス部は演算装置に送られます。

汎用レジスタ 汎用レジスタは演算装置に処理するデータや処理されたデータが一時的に記憶される場所です。

スタックポインタ 記憶装置にスタック領域が確保され、スタックポインタはその先頭アドレスを記憶しています。また、スタックには最後に入れたものを最初に取り出す (LIFO:Last In First Out) 機能があります。

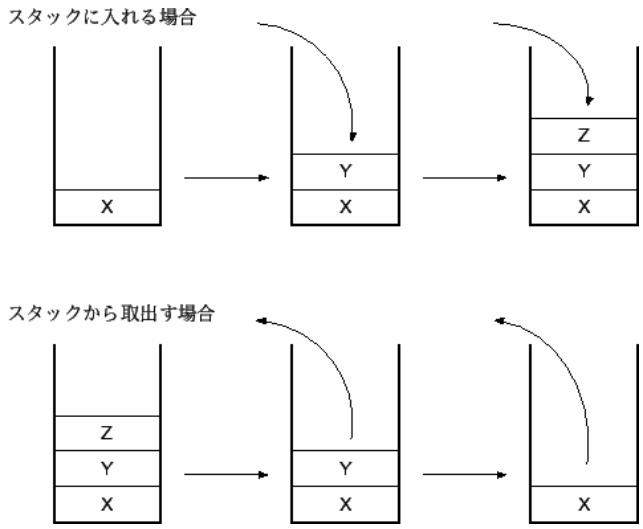

図 5.5: スタックの仕組み

フラグレジスタ フラグレジスタはオーバーフロー (OF), サイン (SF), ゼロ (ZF) フラグで構成され、演算装置による演算結果により値を変えます。

バス バスは CPU とメモリに引かれた道路で命令やデータが流れます。

コンピュータのプログラム実行は次の手順によって実行されていきます。

step 0 コンピュータでプログラムが実行されるとき、最初に命令アドレスレジスタの値が 0 に初期化されます。

step 1(フェチ) 0 番地の命令が命令レジスタに読み込まます。

step 2(デコード) 命令解読器に送られ命令の解読が行われます。 その際、命令レジスタの語長によって命令アドレスレジスタの値を次に実行する命令アドレスまで進めておきます。

step 3(実行) 命令によって各装置を制御します。

step 4 step 1 に戻り命令アドレスレジスタが示す命令をフェチします (命令アドレスレジスタは次に実行する命令のアドレスを示しています)。

それではもう少し詳しく知るために主な命令に対して各装置の動作を見ていきましょう。

ロード命令　もっとも基本的な命令でメモリ又は汎用レジスタの値を他の汎用レジスタに代入します。

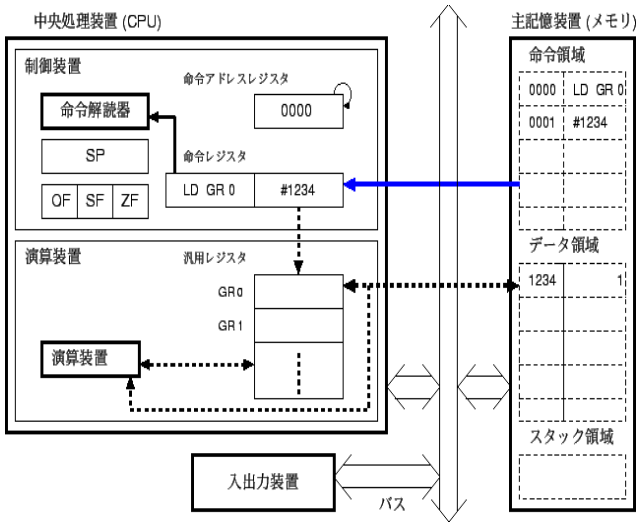

図 5.6: ロード命令:フェチ

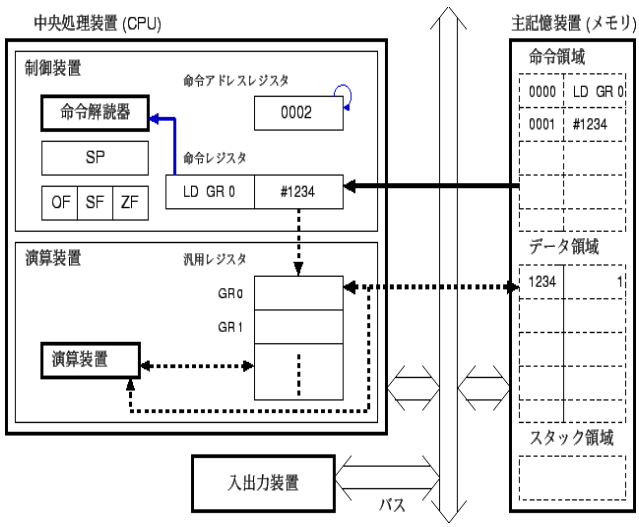

図 5.7: ロード命令:デコード

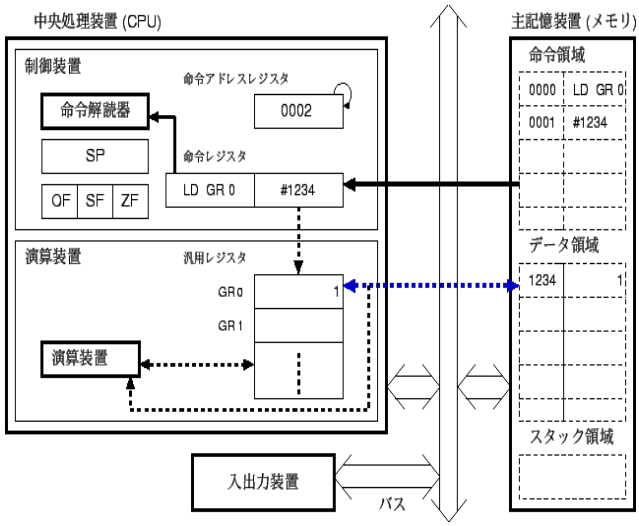

図 5.8: ロード命令:実行

算術加算命令　汎用レジスタにメモリ又は他の汎用レジスタの値を加えます。

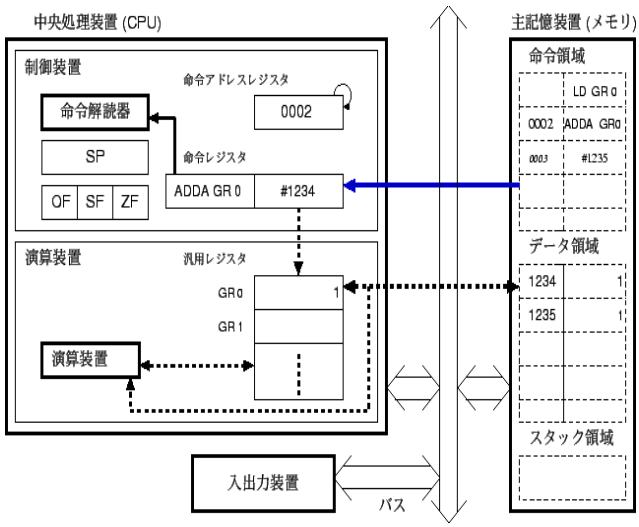

図 5.9: 算術加算命令:フェチ

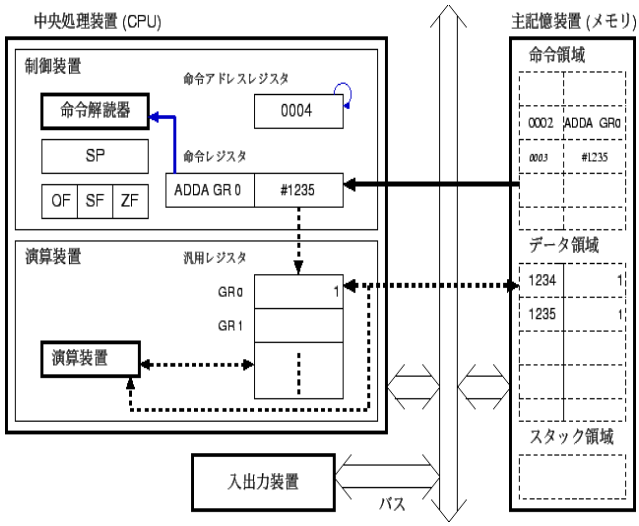

図 5.10: 算術加算命令:デコード

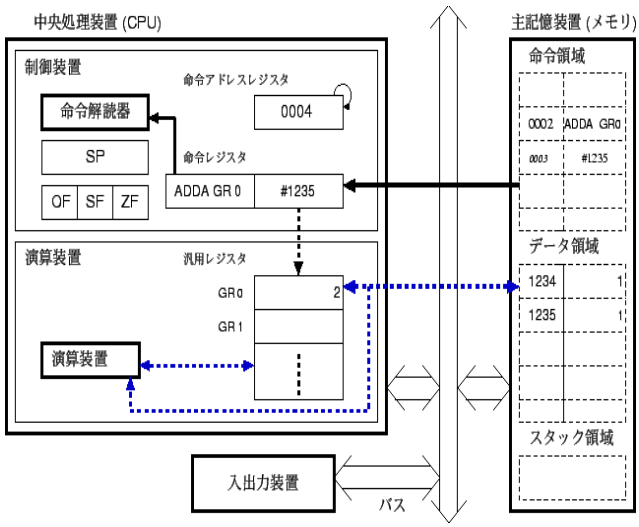

図 5.11: 算術加算命令:実行

比較演算命令と分岐命令 比較演算命令によってフラグレジスタの値が変ったとき続けて分岐命令を行うと条件判別のプログラムとなります。

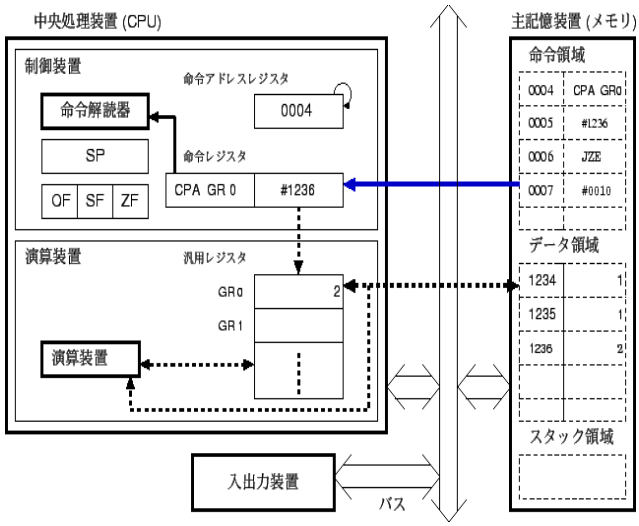

図 5.12: 比較演算命令:フェチ

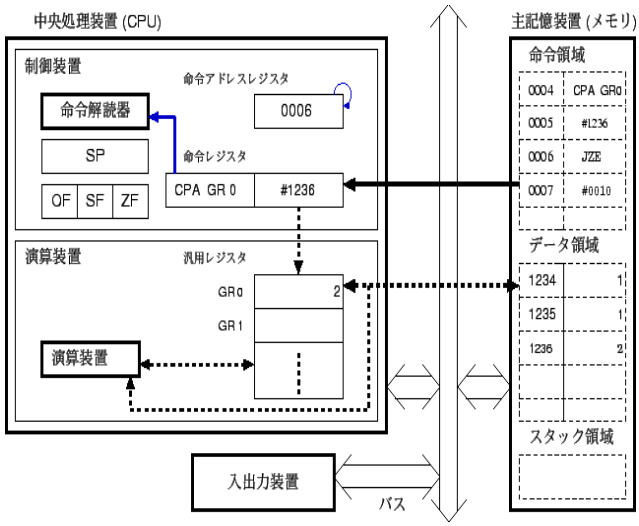

図 5.13: 比較演算命令:デコード

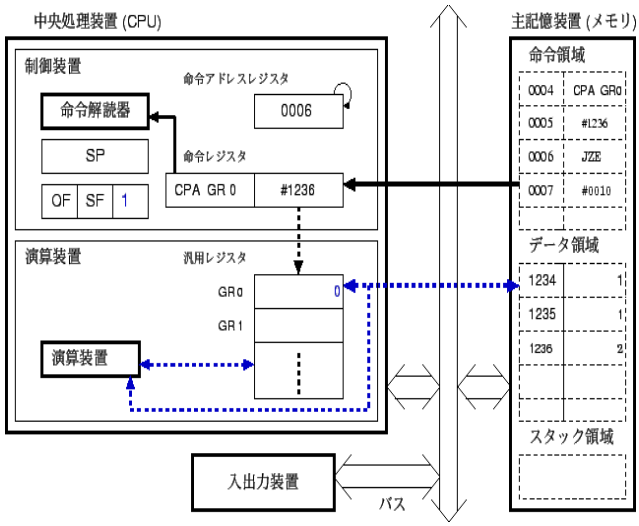

図 5.14: 比較演算命令:実行

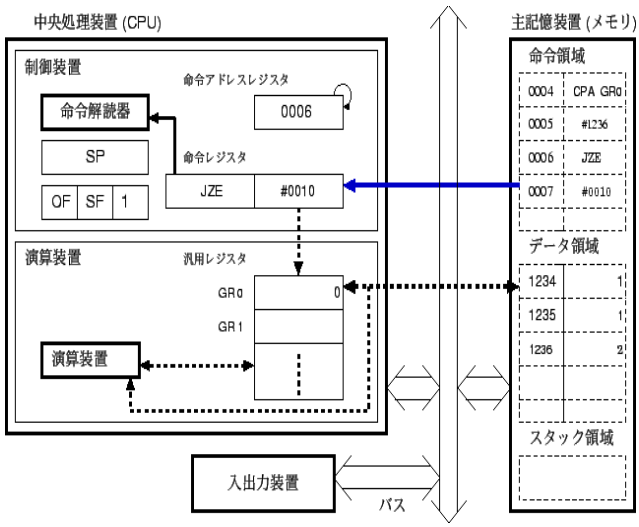

図 5.15: 分岐命令:フェチ

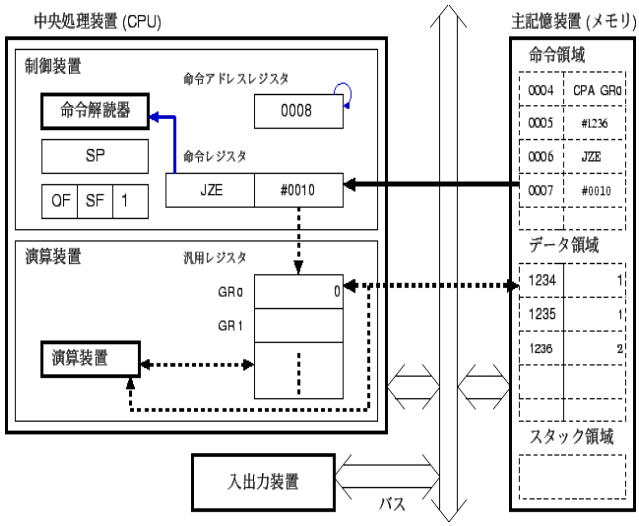

図 5.16: 分岐命令:デコード

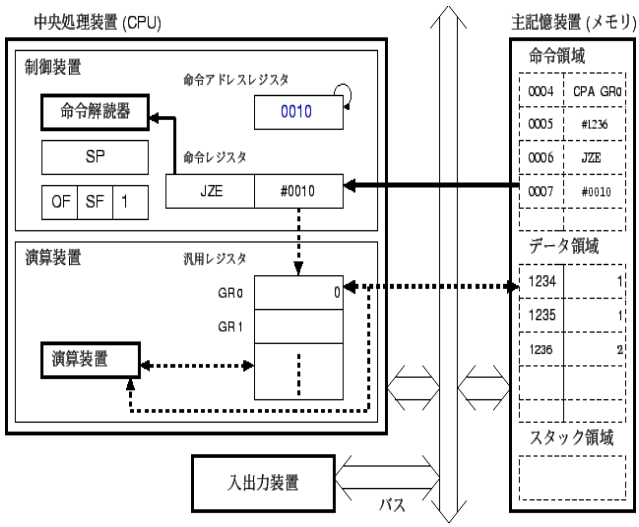

図 5.17: 分岐命令:実行