第 6 回(10/16) プログラムを処理する回路の構成

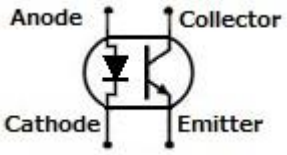
プログラムでは命令の順序に従って処理されるように、順序に従ってそれぞれの命令に番号を付けて、プログラムカウンタによって動作のタイミングを制御します。

6.1 アドレスを用いるデータの転送方法
6.2 中央処理装置で操作されるマシンサイクルの動作
6.3 アッセンブリ言語のプログラム
6.4 c-言語のプログラム
6.5 繰り返される動作のプログラム
6.6 サブルーチンの階層構造
6.7 まとめ

6.1.1 転送されるパルスにより他の回路を制御する方法

-中央処理装置(CPU)のマシンサイクルを駆動するトライジスタ回路―

単安定マルチバイブレータを結合コンデンサーで連結すると一定の時間幅のパルスを次々と転送することができます。このパルス転送回路で他の回路を確実に駆動する場合には LED の代わりに図 40 に示すフォトカップラーを使います。



フォトカップラーは内部で発光ダイオードの発光の[ON-OFF]によりフォトトランジスタを[ON-OFF]します。この素子は光を経由にして他の電子回路を制御するので、成業する回路の動作が制御される回路の動作の影響を受けません。

図 40 フォトカップラーの回路図

フォトカップラー[PC817]を用いて、転送されるパルスの回路から次々と出力を取り出す回路を図 41 に示します。

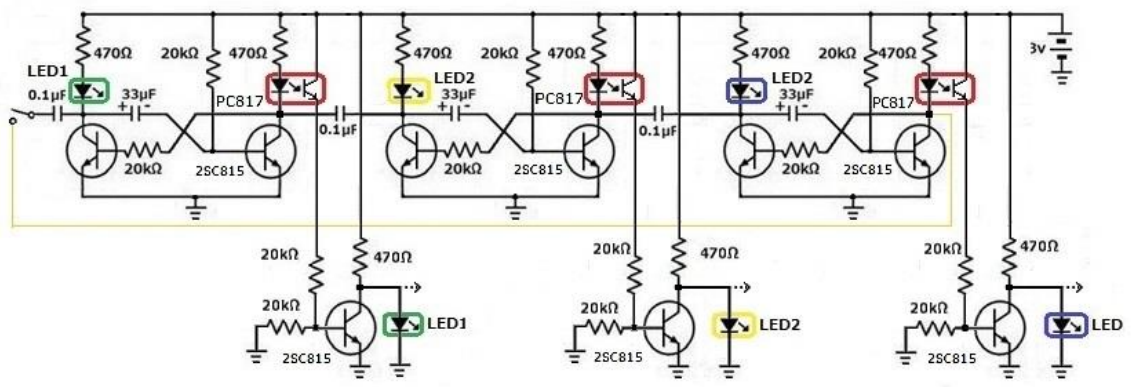


図 41 転送されるパルスから出力を取り出す回路

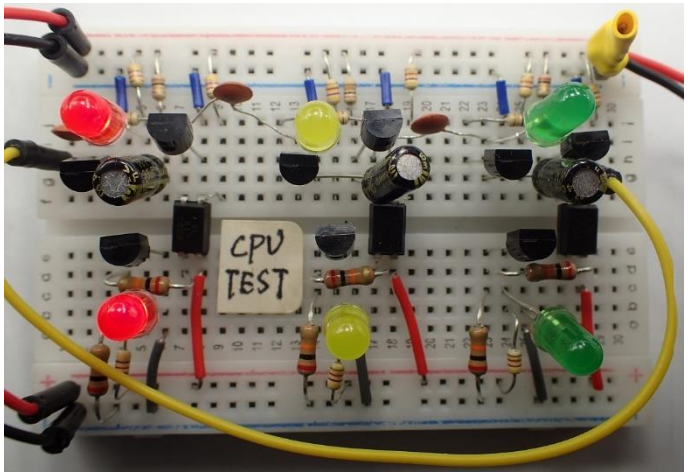


図 42 転送されるパルスから出力を取り出す回路

待機状態では単安定マルチバイブレータの前段のトランジスタが[OFF]で、後段のフォトカップラーが[ON]状態であり、出力の反転増幅器のトランジジスタが[ON]となり、出力のLED は発光していません。電位に下がる入力すると、一定の時間だけ待機状態が反転して入力側と出力側の発光ダイオードが点灯します。
コンピュータでは一区切りの処理が終わった時に、次の区切りのステップの動作をするようにしています。

6.1.2 アドレスを用いるデータの転送方法

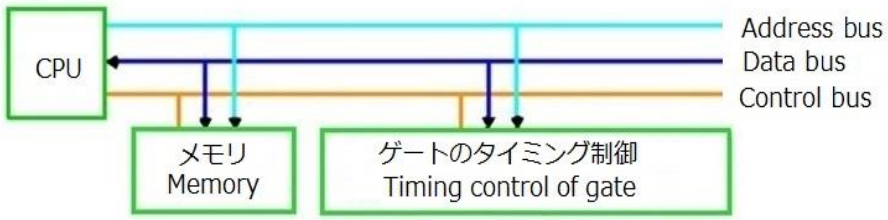


図 43 データ処理におけるバスの役割

プログラム内蔵システムのデータを転送するシステムを図 38 に示します。この装置では入出力装置やレジスタあるいはメモリが多数存在していて、それ等は共通のアドレスバスおよびデータバスに並列に接続されています。そこで、指示する宛先の装置を区別するために番地(Address) をつけます。そのアドレスの情報を送るためにアドレスバス (Address Bus) を使用します。プログラムによって転送先が指定されて、そこでデータバスのゲートを開けばデータが転送されます。

プログラムによって、一区切りの処理が終わった時に、次の区切りのステップの動作を指定できます。そこでは、アドレスを用いるデータの転送方法によってサブルーチンを呼び出

して情報処理してその結果を得て元に戻る処理（コール・アンド・リターン システム）をします。なお、同期方式ではゲートを開く操作のタイミング信号があります。その信号は、クロック（3.2 節無安定マルチバイブレータ）とカウンタによって得られます。

6.2 中央処理装置で操作されるマシンサイクル の動作

プログラム内蔵方式でプログラムを処理することを中央処理装置(CPU)で行います。そこではマシンサイクルと言って次に示す 4 つに分けられた一連の動作を繰り返しています。

1. ステップ 1 では メモリから命令を取得します。
2. ステップ 2 では命令を解読します。
3. ステップ 3 では命令を実行します。
4. ステップ 4 ではメモリに結果を保存します。

マシンサイクルを行う典型的 CPU の構成を図 41 に示します。

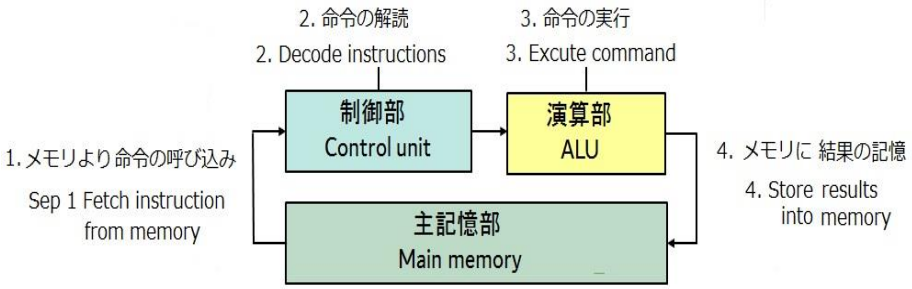


図 44 中央処理装置の構成

プログラムカウンタで CPU が実行すべき命令のタイミングを指定します。

6.3 アッセンブリ言語のプログラム

図 40 にアッセンブリ言語のプログラムの例を示します。このプログラムは 8bit データの場合でメモリの R0L に 100, R0H に 15, R1H に 20 を格納してから, R0L＋R0H－R1H を計算して答えを R0L に納めます。このようにアッセンブラのプログラムではデータを入出力するモリのアドレスを指定します。処理の命令には加算(ADD)や減算(SUB)や条件分岐(JMP)などがあります。

```
.ORG 0D000h          ; ①プログラムを格納する先頭アドレスを 0D000h とする
MOV.B   #100, R0L    ;    ② R0L ← 100
MOV.B   #15 ,R0H     ;    ③ R0H ← 15
MOV.B   #20, R1H     ;    ④ R1H ← 20
ADD.B   R1L, R0L     ;    ⑤ R0L ← R0L + R0H
SUB.B   R1H, R0L     ;    ⑥ R0L ← R0L - R1H
Stop:   JMP Stop     ;    ⑦ 実行停止
```

図45 アッセンブリ言語のプログラム

容量が少ないマイクロコンピュータで一時的にメモリとして用いるRAM (Random Access Memory)をレジスタのように書き換えて使う場合にメモリの番地を指定するアッセンブリ言語のプログラムが使われます。アッセンブル(assemble)というのは組み立てるという意味です。プログラミング言語で記述されたテキスタをソースコードといい、[0]と[１]を組み合わせた機械語の実行ファイル(オブジェクトコード)に変換(コンパイル)します。

6.4 c-言語のプログラム

ｃ-言語は異なるアーキテクチャのCPUでも移植をすることができるように開発されたプログラミング言語です。その結果、プログラムを準備して（include）してそのプログラムを呼び出して使う関係で、c-言語のプログラムは積み木構造に階層化しています。

c-言語のプログラムでは変数で宣言し、指定された変数の内容を定義します。アドレスを指定しません。

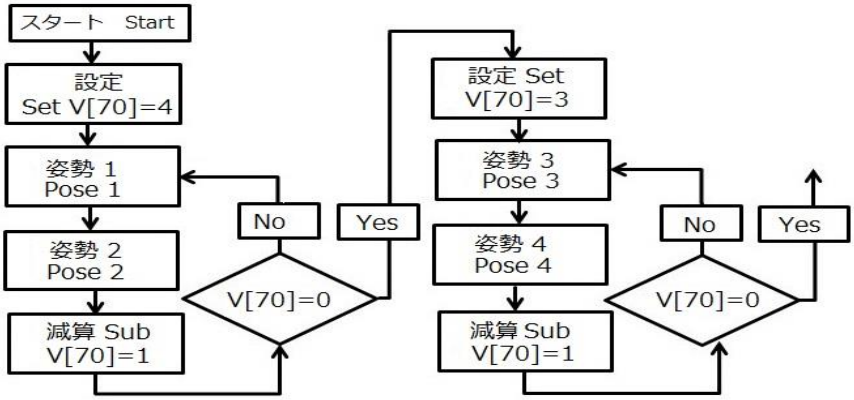
図46 にｃ-言語で記述された加算（A(10)+B(20)=C）のプログラムを示します。

c-言語はテキスト全体をまとめて実行ファイルに変換します。このタイプをコンパイル型のプログラミング言語と言います。前節のアッセンブリプログラム言語もコンパイル型です。

他方、1行呼び出されては実行するインタープリタ型（同時通訳型）のタイプにはロボットのプログラミングなどで使われるPythonがあります。

```
#include <stdio.h>
int add(int a, int b);  //function declaration
int main()
{
        int a=10,b=20;
        int c=add(10,20);  //function call
        printf("Addition:%d¥n",c);
        getch();
}
int add(int a,int b)   //function body
{
        int c;
        c=a+b;
        return c;
}
```

図46　c-言語のプログラムの例



6. 5 繰り返される動作のプログラム

図47 に、4回の繰り返される動作と3回の繰り返される別の動作を行うプログラムを示しています。

図47 繰り返される動作のプログラム

ここでは、繰り返しの数を記憶するメモリの場所を 70 番地 V[70]として初期を設定し、データを動作の終わりに 70 番地のメモリの数値を 1 減算する処理をして、その結果の数値が 0 になるまでループさせています。

ループするプログラムにおいては分岐やジャンプ命令があるので、メモリのアドレスを改めて指定する必要があります。そこで、アドレスを処理するためにメモリ（レジスタ）が必要となります。プログラム内蔵方式を実現する回路のしくみから、繰り返し動作のプログラムが図 36 示すようになることが理解できます。

6.6 サブルーチンの階層構造

メインのプログラムからサブルーチンを呼び出すしくみを図48 に示します。

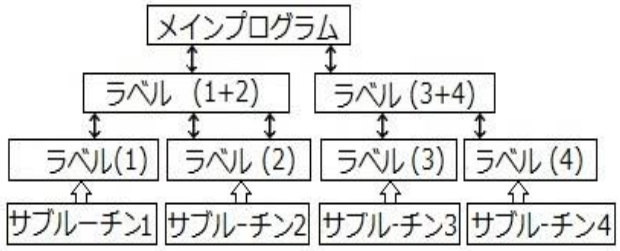


図 48 サブルーチンのファイル名の階層構造

サブルーチンを呼び出して情報を処理して元に戻る「コール アンド リターンシステム」によってプログラムを共用できます。すなわち、上位のプログラムで下位のプログラムをプログラムの名前（表象）をタグとして扱います。階層構造のファイルでは下層のファイルをファイル名で代表させて、上層の階層のファイルで構成要素のファイル名を示します。実行するプログラムは最も低い階層のプログラムです。階層構造では上位の階層で情報圧縮しています。

文章が単語を要素としています。言語の表現では情報圧縮が行われています。その人間の言語活動には大脳の連合野が関与しています。表象で情報を処理する方法は行動の制御も言語の表現も同じです。しかし、言語活動の時間の進行は行動の制御の活動の時間進行とは相違しています。

6.7 まとめ

A) パソコンでは予めプログラムをインストールしておいて、ファイル名にデータを入れますがその際に、拡張子でプログラムを指定しています。
B) 「最初にアドレス A 番地のメモリにデータ X を収め、アドレス B 番地のメモリにデータ Y を収めます。」としておいて、アドレス A 番地のメモリとアドレス B 番地のメモリと加え合わせてアドレス C 番地のメモリに収めるというようにプログラムを作っています。
C) プログラムを用いる装置では階層構造のサブルーチンを記憶しておいて、それを呼び出して使うのであらかじめ順序正しく準備しておくことが必要です。
D) パソコンの操作でうまくいかなくなった場合には最初から始める方が早道になることが多いです。