コンピュータ工学 講義プリント(9 月 11 日)

今回は、サブルーチンの動作や、サブルーチンを呼び出すのに必要なスタックについて説明する。

・スタック(教科書 P.31 参照)

頻繁に読み書きするデータは、通常 RAM に記憶する。RAM は、アドレスを指定する事で、任意の情報を任意の順番で読み書きできるので、便利であるが、これを裏返せば、全ての情報のアドレスをちゃんと管理しなければならないという事である。

一時的に使いたいデータのアドレスまできちんと管理すると、作業が煩雑になる。そこで、「任意の情報を任意の順番で読み書きできる」という柔軟さを捨てる代わりに、アドレスの管理を不要にした記憶方式がいくつか考えられている。スタックも、そういった記憶方式の一つである。

スタックは、複数の情報を記憶させた後に、情報を読み出すと、最後に記憶された情報から順に読み出せるという性質を持った、記憶方式である。

そもそもスタック(stack)とは、本などの物を積んでできた山の事である。スタックの記憶方式を理解するには、データを積んでできた山をイメージすると分かりやすい。

スタックでは、push と pop の 2 つの基本操作ができる。push とは、新しいデータを 1 つ記憶させる操作で、pop はデータを 1 つ取り出す操作である。以下、push と pop の操作について、図を使って説明する。

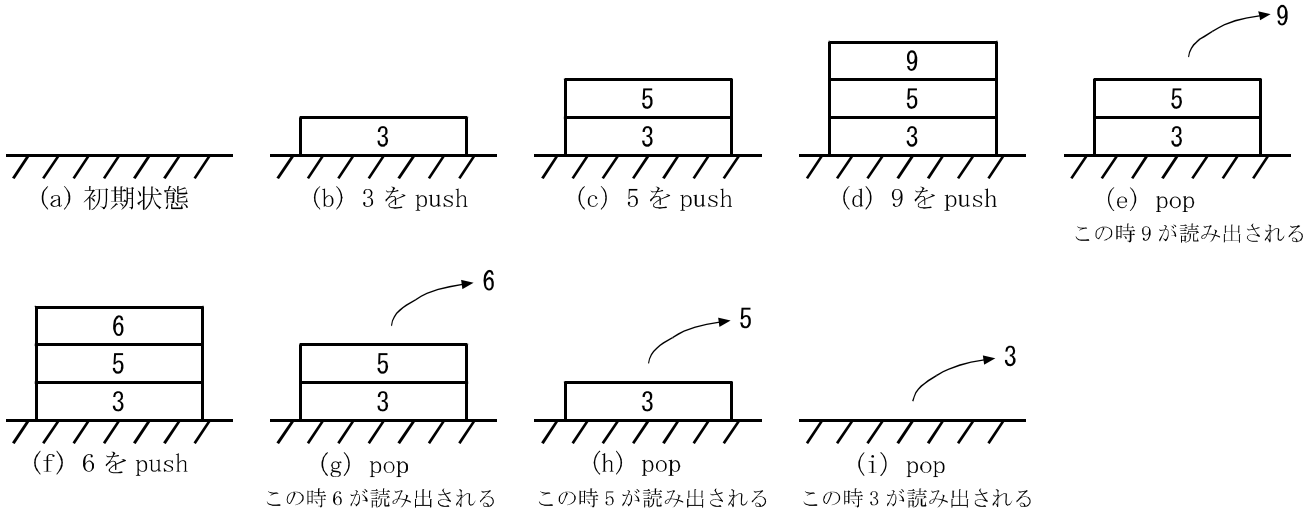


図 1、スタックの動作の説明

図 1(a)は、スタックの初期状態で、何も記憶していない様子を表している。上に何も置いていないテーブルをイメージして欲しい。

図 1(b)は、初期状態のスタックに、3 という数字を push した後の状態を表している。テーブルの上に、表紙に 3 と書いた本が載っている様子をイメージして欲しい。

図 1(c)は、さらに、5 という数字を push した後の状態を表している。3 の本の上に、5 の本が積まれた。

図 1(d)は、さらに、9 という数字を push した後の状態を表している。5 の本の上に、9 の本が積まれた。

図 1(e)は、図 1(d)の状態で、pop した様子を表している。pop すると、最後に push した情報(一番上の本)を読み出す事ができ、またその情報(本)は、スタックから取り除かれる。この場合は、9 が読み出され、かつその 9 は、スタックから取り除かれる。

図 1(f)は、さらに、6 という数字を push した後の状態を表している。

図 1(g)は、図 1(f)の状態で、pop した様子を表している。一番上の 6 が読み出され、かつその 6 はスタックから取り除かれる。

図 1(h)は、さらに pop した様子を表している。一番上の 5 が読み出され、かつその 5 はスタックから取り除かれる。

図 1(i)は、さらに pop した様子を表している。一番上の 3 が読み出され、かつその 3 はスタックから取り除かれる。これで、スタックに記憶したデータ(テーブルの上の本)はなくなり、初期状態にもどった。

以上の様に、スタックという記憶方式を使うと、メモリの何番地に情報を書き込むかを考える必要がなく、どの順で、データを書いたり読んだりするかで、読み出されるデータの順が決まる。

なお、push の操作は、山の上に物を積むことに似ているので、「データを push する」ことを「データをスタックに積む」とも言う。

・スタックの実装

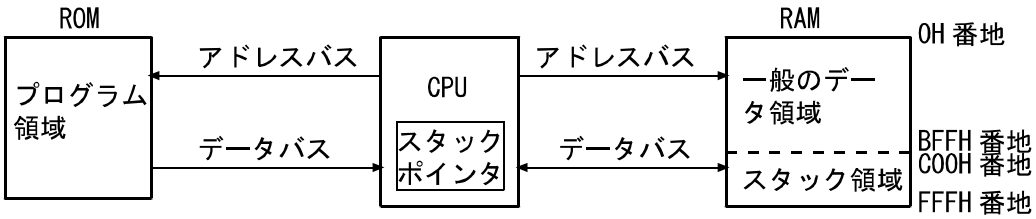


図 2、スタックの実装の例

スタックは通常、データ記憶用の RAM の一部をスタック用のメモリとして用いる事で、実現している。

図 2 は、ハーバード・アーキテクチャを採用したマイコンで、RAM 内にスタックを実装した例である。このマイコンでは、RAM に 0H 番地から FFFH 番地までのアドレスが振られている。このうち、先頭の 0H 番地から BFFH 番地までを、アドレスとデータの内容を紐付けて管理する一般のデータ領域に使い、C00H 番地から FFFH 番地までを、スタックの実現のために使う領域(スタック領域)として使っている。また通常、CPU にはスタックポインタと呼ばれる、未使用のスタック領域の最後のアドレスを記憶するレジスタがある。レジスタとは、CPU 内部の小容量の記憶装置のことである。

なお、スタックポインタが、使用済みのスタック領域の先頭のアドレスを記憶する方式の場合もあるが、この場合は、未使用のスタック領域の最後のアドレスを記憶する方式の場合より、1 だけ大きいアドレスを記憶する。

次のページの図 3 は、RAM 内のスタック領域と、スタックポインタの様子を表した図である。図 3(a)～(f)は、図 1(a)～(f)にそれぞれ対応している。

図 3(a)は初期状態を表す。スタック領域に何もデータが格納されていないので、スタックポインタはスタック領域の最後のアドレスである、FFFH 番地を指している。

図 3(b)は、3 を push した状態を表している。スタック領域最後の FFFH 番地に 3 が書き込まれ、スタックポインタの内容は、1 減って、FFEH となる。この様に、データを push した後でも、スタックポインタはちゃんと未使用のスタック領域の最後のアドレスを指している事がわかる。なお、通常 CPU にはデータをスタックに push する命令が備えられている。アセンブリ言語で

```
PUSH r0
```

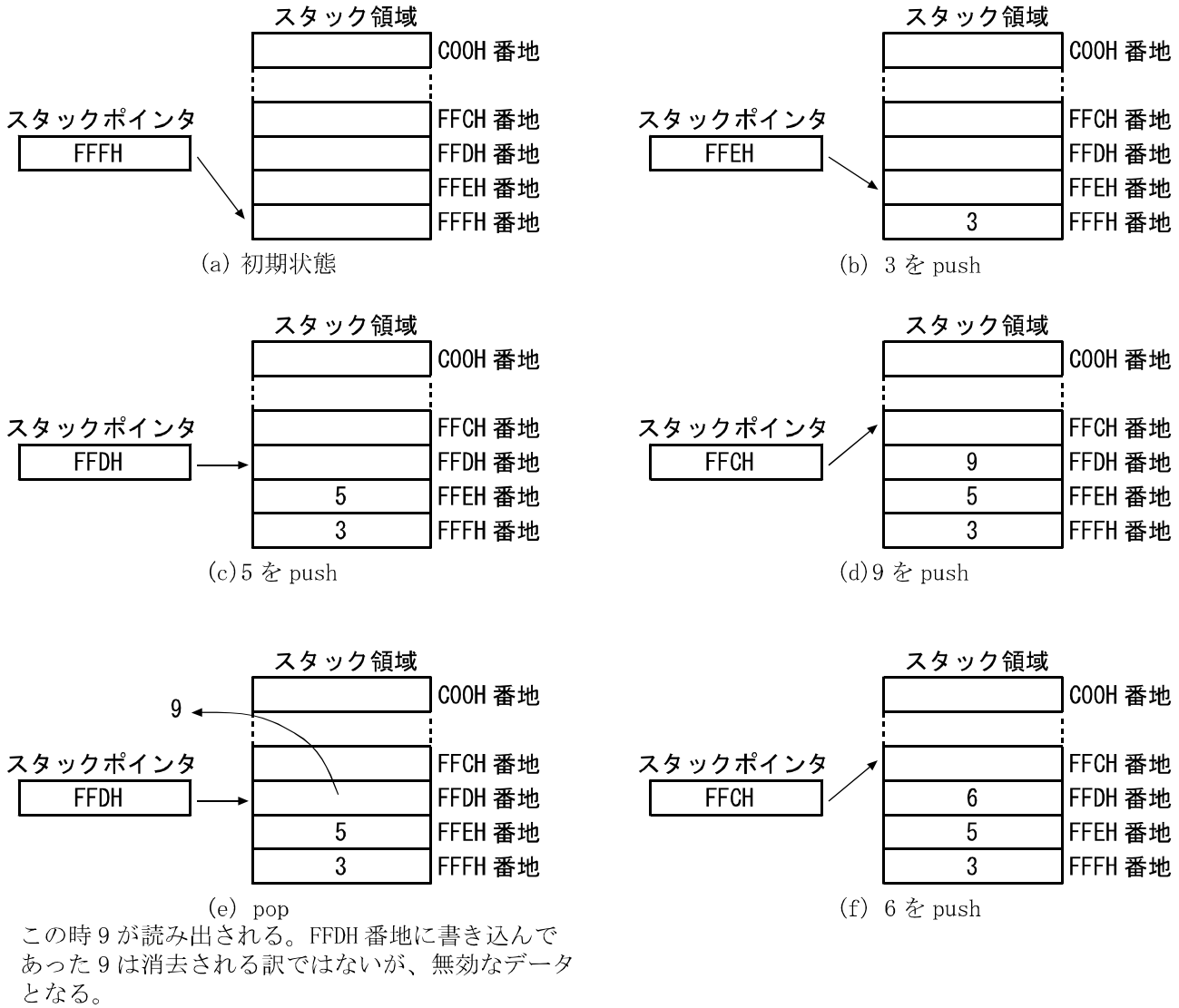


図 3、スタックの動作

などとすれば、スタック領域にレジスタ r0 のデータを書き込むことも、スタックポインタの内容を 1 減らす事も、CPU が 1 命令で実行してくれる。

図 3(c)は、さらに 5 を push した様子を表している。FFEH 番地に 5 が書き込まれ、またスタックポインタの内容が 1 減って、FFDH になっている。

図 3(d)は、さらに 9 を push した様子を表している。

図 3(e)は、図 3(d)の状態でデータを pop する様子を表している。まずスタックポインタの内容に 1 を加えて、スタックの使用領域の先頭アドレス(FFDH 番地)を求め、次にそのアドレスの内容(9)を読み出す。なお、通常 CPU にはデータを pop する命令が備えられている。アセンブリ言語で

POP r0

などとすれば、スタックポインタの内容に 1 を加える事も、スタックから読み出したデータをレジスタ r0 に書き込むことも、CPU が 1 命令で実行してくれる。

図 3(f)は、さらに 6 を push した様子を表している。

・サブルーチンの実行とスタック

サブルーチンとは、プログラム中の、ある機能単位をまとめたものである。機能単位にプログラムをグループ化することによって、プログラムを読みやすくしたり、あるいは、繰り返し使われる機能を 1 箇所にまとめ、それを再利用する事によって、プログラムのサイズを小さくしたりするために用いられる。

Pascal においては手続き(procedure)や関数(function)と呼ばれるものがサブルーチンである。C 言語においては、main 関数以外の関数がサブルーチンである。

ここで、リスト 1 の様な C 言語のプログラムを考える。

リスト 1、サブルーチン呼び出しの例

```
void func_A(void)
{
  ・・・
}

void func_B(void)
{
  ・・・
  func_A(); // line 3
}

void func_C(void)
{
  ・・・
  func_A(); // line 5
  ・・・
  func_B(); // line 6
  ・・・
}

void main(void)
{
  func_A(); // line 1
  func_B(); // line 2
  func_C(); // line 4
}
```

このプログラムには、main 関数以外に func_A、func_B、func_C の 3 つの関数(サブルーチン)がある。また、後の説明が分かりやすいように、要所要所にコメントで行番号を入れてある。

このプログラムをコンパイルして機械語にすると、ROM 内のプログラムの配置は、例えば次のページの図 4 の様になる。

機械語プログラムの先頭にあるのは、初期化ルーチン(0000H 番地～00070H 番地)である。初期化ルーチンは、例えばスタックポインタにスタック領域の最終アドレスを代入するといった、プログラムを開始する際に必要となる各種初期化を行う。

初期化ルーチンが終わると、main 関数(0080H 番地～0086H 番地)が実行される。Main 関数は、func_A の呼び出し(line 1)、func_B の呼び出し(line 2)、func_C の呼び出し(line 4)、CPU の停止命令の 4 つの命令からなる。

サブルーチンを呼び出すには、通常 CALL というニーモニックの命令を使う。例えば、func_A を呼び出すなら、func_A の先頭アドレス 0087H を使って、

CALL 0087H

とすればよい。そうすると CPU は次の 2 つの動作を行う。

(1) スタックに CALL 命令の次のアドレス(0082H)を積む
(2) プログラムカウンタに 0087H を代入する

プログラムカウンタについては、4 月 10 日の講義で説明したが、次に実行する命令のアドレスを保存しているレジスタである。

図 5 は、プログラムの特定の場所を実行した際の、プログラムカウンタとスタックの状態を表した図である。

初期化ルーチン最後の命令を実行し終わった状態でのプログラムカウンタとスタックの状態は図 5(a)の様になっている。

Line 1 の func_A の呼出し命令(CALL 00087H)の実行の後には、図 5(b)の様な状態となる。すなわちスタックには func_A の呼出し命令の次の命令のアドレス(0082H)が代入されており、プログラムカウンタには、func_A の先頭アドレス(0087H)が代入される。よって、この CALL 命令

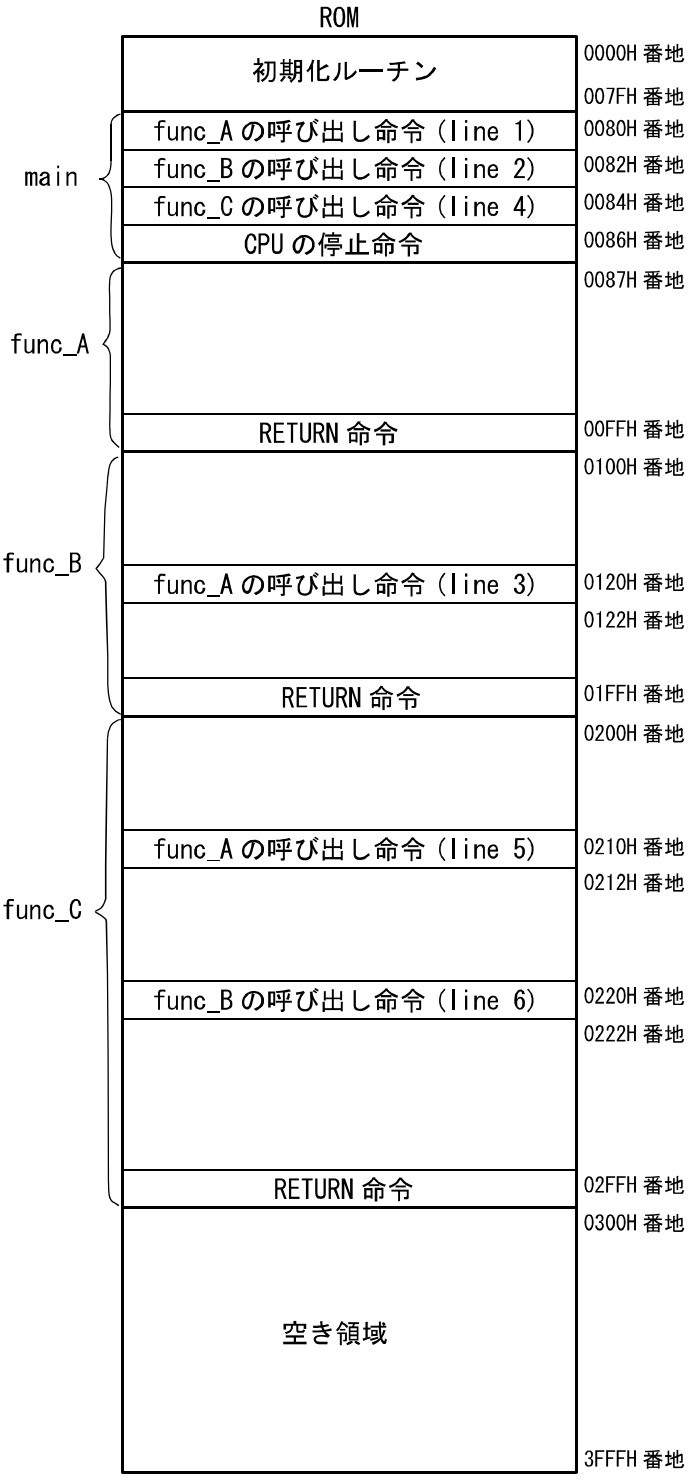


図 4、リスト 1 をコンパイルしてできた機械語プログラムのメモリマップ

サブルーチンを呼び出す命令は 2 ワードと仮定している。また、ハーバード・アーキテクチャのマイコンでは、ROM も RAM も 0 番地から始まることに注意。(ノイマン型コンピュータでは、ROM と RAM に同じアドレスを割り当てる事はない)

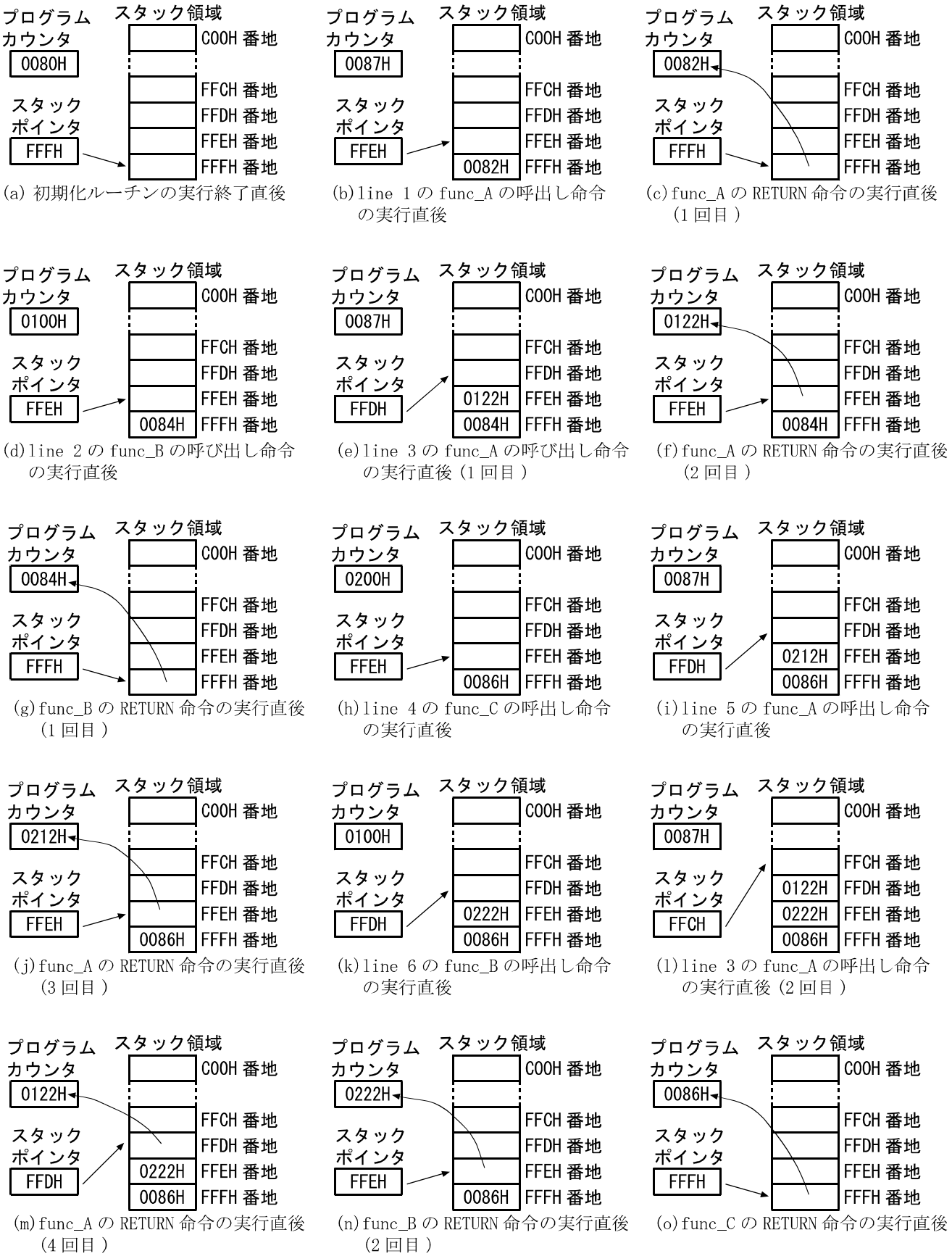


図 5、プログラム実行の各時点でのプログラムカウンタとスタックの状態

話を簡単にするために、RAM の 1 ワードのビット数は、ROM のアドレスを格納するのに十分な大きさがあるものとしている。

の次に実行されるのは、func_A の最初の命令である。なお、サブルーチンの呼出し命令の次の命令のアドレスのことを、リターンアドレスと呼ぶ。

func_A が実行されると、最後に RETURN 命令が実行される。RETURN というニーモニックの、この命令を実行すると、CPU は次の 2 つの動作を行う。

(1) スタックからリターンアドレスを pop する。
(2) 読み出したリターンアドレスをプログラムカウンタに代入する。

図 5 (c)は、func_A の最後の RETURN 命令の実行直後の状態である。スタックから、リターンアドレス(0082H)が pop され、プログラムカウンタに代入されているのが分かる。

次に実行される命令は、0082H 番地の func_B の呼出し命令である。この命令はソースリストの line 2 に対応している。この func_B の呼出し命令の実行直後の状態を図 5(d)に示す。

呼び出された func_B を実行していると、さらに func_A の呼出し命令(ソースリストの line 3 に対応)に行き着く。この func_A の呼出し命令の実行直後の状態を図 5(e)に示す。サブルーチンの中でさらにサブルーチンを呼んでいるので、スタックにリターンアドレスが 2 つ積まれているのが分かる。

func_A の最後の RETURN 命令を実行すると、スタックに積まれている上側のリターンアドレスが pop されて、プログラムカウンタに代入される。RETURN 命令が実行し終わった状態を図 5(f)に示す。

次に実行される命令のアドレスはプログラムカウンタの指す 0122H で、処理が func_B の内部に戻る。そのまま func_B を実行していると、最後に RETURN 命令に行き着く。この RETURN 命令の実行直後の状態が図 5(g)である。

以降の説明は、省略するが、図 5(h)以降の様に、プログラムカウンタとスタックの状態が変化していく。

この様な仕組みで、サブルーチンが正しい順序で実行される。

・PIC マイコンのスタックの特徴

教科書で扱っている PIC16F84A では、RAM の一部をスタック領域として使うのではなく、スタック専用のメモリを用意している。(教科書 P.21 の図 2.7 参照) この様な構造になっているマイコンは極めて特殊である。

通常のマイコンではスタックはサブルーチンのリターンアドレスを記憶するだけではなく、一般的なデータの一時的な記憶にも使われる。しかしながら PIC16F84A では、スタックがサブルーチンのリターンアドレスの記憶に特化しているため、データを記憶させる事はできない。

PIC16F84A には、サブルーチンを呼び出す CALL 命令(教科書 P.98 参照)や、サブルーチンから復帰する RETURN 命令(教科書 P.100 参照)は存在するが、データをスタックに積むための PUSH 命令や、データをスタックから取り出すための POP 命令はない。

レポートや試験で記述式の解答を書く際の注意点

今回のレポートや試験では、記述式の解答を求める問題が増えてくる。中間試験で、記述式の問題の点数が低い傾向にあったので、どのような点に注意して解答を書くべきか、簡単に説明する。

・記号・変数・信号名の説明をきちんとする事

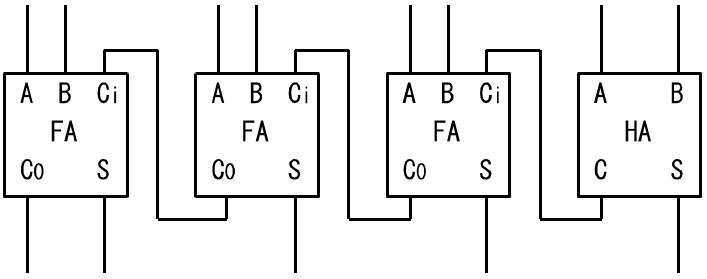
記号や変数を使う場合、一般的に認知されている d(微分記号)、≧(大なりイコール)、π(円周率)、e(自

然対数の底)などの記号・変数でなければ、必ず説明して使う事。また、回路図において、複数の信号を扱うときは、各信号に信号名を付け、区別する事。

例題を挙げて説明する。

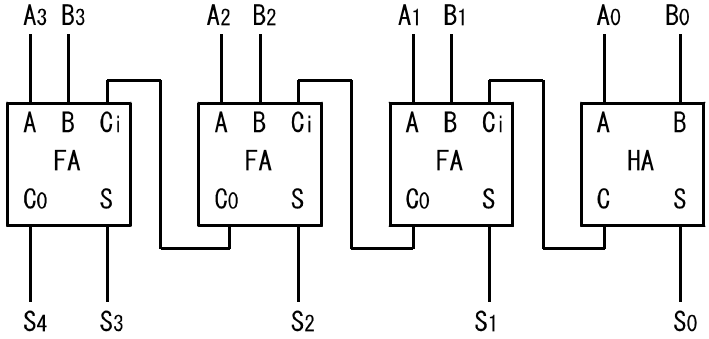
【例題】半加算器 1 つと全加算器 3 つを用いて、2 つの 4 ビットの入力を足した結果を 5 ビットで出力する回路を作りなさい。

【良くない解答例】



【解説】この解答では、どの信号が入力信号で、どの信号が出力信号かもはっきりしない。また、ビットの上位・下位の関係もわからない。

【良い解答例】2 つの 4 ビットの入力信号を A_3,A_2,A_1,A_0 および B_3,B_2,B_1,B_0 とする。ここで A_0 と B_0 は最下位ビット、A_3 と B_3 は最上位ビットである。また、5 ビットの出力信号を S_4,S_3,S_2,S_1,S_0 とする。ここで S_0 は最下位ビット、S_4 は最上位ビットである。この時、求められている加算器は、次の回路図で表わされる。



【解説】この解答では、先ほどの良くない解答例の問題点が解決しており、コンピュータ工学の定期試験の解答としては十分である。ただし、上の回路図で使った半加算器や全加算器の記号は、一般的に使われているとは言い切れないので、例えば大学院の入学試験なら、半加算器や全加算器の記号の説明も書かなければならない。

・図を使わなければ伝わりにくい概念は、必ず図を使って説明する事

技術用語の意味を解説させるような問題の場合、文字だけでは十分意味が伝わらず、図を使わなければならない場合が多い。この様な場合に文字だけで解答すると、大きな減点の原因となる。

例えば、広辞苑第五版で「トランジスター」を調べると、次のように書いてある。

ゲルマニウム・珪素などの半導体の接合を利用して、増幅・発振・スイッチングの作用を行う素子。transfaer of signals through varistor からの造語。エミッター・ベース・コレクターの 3 部からなり、極めて小型、加熱電極が不要で、必要電力は小さい。ショックレーらが発明。

この説明を読んで、十分に理解できる人は少ないだろう。それに「エミッター・ベース・コレクターの 3 部からなり」と書いてあっても、それらの位置関係は、この説明では把握できない。辞書は百科事典のように図が多用できないし、また、学術的に厳密な説明も求められているわけではない。上記の説明は、辞書に掲載するものとしては十分であるが、電子工学科の学生の解答としては不十分である。

この問題に関して、コンピュータ工学の授業で習った範囲で例題を挙げ、もう少し詳しく説明する。

【例題】プルアップ抵抗およびプルダウン抵抗の意味や利用方法について説明せよ。

【良くない解答例】プルアップ抵抗は入力端子の論理を 1 に固定するための抵抗である。プルダウン抵抗は入力端子の論理を 0 に固定するための抵抗である。

【解説】この解答では、どのように抵抗を使って入力端子の論理を 0 あるいは 1 に固定するのかが良く分からない。また、利用方法については、何も答えていない。

【良い解答例】プルアップ抵抗は、論理回路の入力端子の論理を、1 に固定するために、図 1 の様に、電源 VCC と入力端子を接続する抵抗の事である。

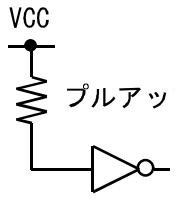


図 1、プルアップ抵抗

また、プルダウン抵抗は、論理回路の入力端子の論理を 0 に固定するために、図 2 の様に、GND と入力端子を接続する抵抗の事である。

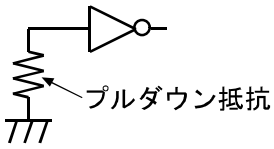


図 2、プルダウン抵抗

単に論理を 1 あるいは 0 に固定したいだけなら、抵抗を使わずに、入力端子を直接 VCC または GND に接続するだけでも可能である。

しかしながら、図 3 に示す様なスイッチ入力回路の場合、プルアップ抵抗を配線に置き換えると、スイッチを ON にした場合、VCC と GND がショートしてしまう。プルアップ抵抗を使う事で、電源をショートさせることなく、スイッチの操作で入力の論理を切り替える事ができるのである。

また、図 4 に示すように、プルダウン抵抗を用いれば、スイッチ入力回路の論理を図 3 の回路とは反対にする事ができる。

プルアップ抵抗やプルダウン抵抗は、これらのスイッチ入力回路の様に、入力端子の論理を 1 または 0 に仮に固定しておき、後から入力端子を電源などに接続する事で、

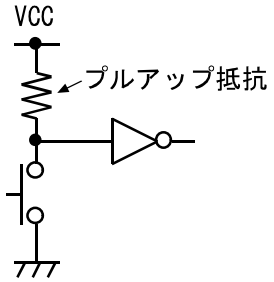


図 3、プルアップ抵抗を用いたスイッチ入力回路

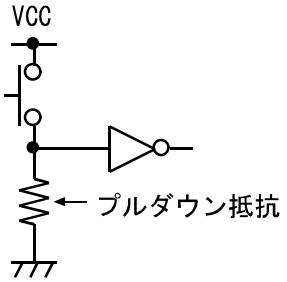


図 4、プルダウン抵抗を用いたスイッチ入力回路

論理を逆転させたい場合に利用できる、論理の固定方法である。

【解説】図 3 や図 4 の様なスイッチ入力回路を例としてあげ、プルアップ抵抗やプルダウン抵抗の使い方を説明する場合、図を用いなければむりである。仮に「プルアップ抵抗の入力端子側の一端を、スイッチを介して GND に接続し…」という具合に、図 3 の回路図を言葉で説明したら、とても理解しにくい文章になってしまう。この様な場合は、積極的に図を書くこと。