コンピュータ工学 講義プリント(9 月 4 日)

第 1 回目の講義において、マイクロコンピュータの構成の概要を説明したが、今回の講義では、マイクロコンピュータの構成について、さらに詳しく解説する。特に、RISC と CISC の違い、およびノイマン型コンピュータとハーバード・アーキテクチャを採用したコンピュータの違いについて詳しく述べる。

・高級言語と機械語

我々がコンピュータのプログラムを作る場合は、C 言語、Pascal などのいわゆる高級言語を使う事が多い。高級言語は、人間が分かりやすい記述方法でプログラムを記述するために開発されたプログラム言語であるが、コンピュータは高級言語を直接理解できない。

コンピュータが理解できるのは、機械語と呼ばれる言語のみである。機械語は、2 進数の数列であるので、人間にはとても理解しにくい。

そこで通常は、人間は高級言語でプログラムを作成し、コンパイラと呼ばれるソフトウェアでそのプログラムを機械語に翻訳してから実行する手順をふむことになる。なお、コンパイラで高級言語のプログラムを機械語に翻訳する作業の事をコンパイルという。

なお、高級言語を機械語に翻訳せずに、インタプリタと呼ばれる、高級言語を解釈しながら実行するソフトウェアを利用する方法もある。

高級言語で記述したプログラムの例として、リスト 1 に、Arduino Uno 用の Hello world!のスケッチ(プログラム)を示す。このプログラムを実行すると、シリアルポート(通信ポートの一種)に”Hello world!”のメッセージを出力する。(なお、通常は Hello world!といえば、画面に”Hello world!”と表示するプログラムの事であるが、Arduino Uno をはじめとするマイコンボードには通常画面がないため、シリアルポートに出力する仕様とした)

Arduino Uno のスケッチは、C++という高級言語に基づく言語で記述するので、リスト 1 を見れば、どのような動作をするか、諸君もなんとなく理解できるだろう。

一方で、リスト 1 をコンパイルし、機械語に翻訳した結果がリスト 2 である。機械語は、本来は 2 進数の数列であるが、長くなるので 16 進数で表記してある。このように、機械語は一見暗号の様に見え、人間にはとても理解しにくい。

リスト 1、Arduino Uno 用の Hello world!のスケッチ

```
void setup()
{
  Serial.begin(9600); // 9600bps でシリアルポートを初期化
  Serial.println("Hello world!"); // シリアルポートに"Hello world!"を送信
}

void loop()
{
}
```

リスト 2、Hello world!のスケッチをコンパイルして得た機械語プログラム(先頭部分を抜粋)

```
0C 94 35 00 0C 94 5D 00 0C 94 5D 00 0C 94 5D 00 0C 94 5D 00 0C 94 5D 00
0C 94 5D 00 0C 94 5D 00 0C 94 5D 00 0C 94 5D 00 0C 94 5D 00 0C 94 5D 00
0C 94 5D 00 0C 94 5D 00 0C 94 5D 00 0C 94 5D 00 0C 94 0C 03 0C 94 5D 00
0C 94 74 00 0C 94 C2 00 0C 94 5D 00 0C 94 5D 00 0C 94 5D 00 0C 94 5D 00
0C 94 5D 00 0C 94 5D 00 2E 02 11 24 1F BE CF EF D8 E0 DE BF CD BF 11 E0
A0 E0 B1 E0 E2 E7 F7 E0 02 C0 05 90 0D 92 AE 31 B1 07 D9 F7 11 E0 AE E1
B1 E0 01 C0 1D 92 A1 3D B1 07 E1 F7 10 E0 CA E6 D0 E0 04 C0 22 97 FE 01
0E 94 B3 03 C8 36 D1 07 C9 F7 0E 94 84 02 0C 94 B7 03 0C 94 00 00 08 95
```

高級言語では、例えば

```
a=b*2+c;
```

と記述すると、「変数 b の内容を 2 倍して、そこに変数 c の内容を足し、変数 a に代入する」という操作を記述できるが、機械語においては、2 倍するのに 1 命令、足し算をするのに 1 命令、代入するのに 1 命令といった具合に、1 つの命令でできる操作が非常に限られている。そのため、高級言語において 1 行で記述できる操作も、機械語に翻訳すれば、通常多数の命令の組み合わせになってしまう。

・RISC と CISC(教科書 P.22 参照)

初期のコンピュータにおいては、高級言語を使わずに、アセンブリ言語を使ってプログラムを開発する事が多かった。アセンブリ言語とは、機械語の命令と 1 対 1 に対応するニーモニックというアルファベット列を使って記述するプログラム言語である。

例えば、PIC16F84A というマイコンで、「12H 番地のデータを w レジスタに読み込む」という機械語の命令は、0010000010010 という 14 ビットの 2 進数になるが、アセンブリ言語で表すと、

```
MOVF 12H,0
```

となる。(教科書 P.70 参照) ニーモニックの MOVF は、MOVE と FILE という 2 つの英単語を組み合わせてできるため、意味を連想しやすい。

前の節で述べたとおり、機械語 1 命令で処理できる内容はわずかである。そのため、機械語やアセンブリ言語でプログラムを開発すると、簡単な処理をするにも、非常にたくさんの命令を組み合わせる必要があり、プログラムの手間がかかった。

そこで、なるべく少ない命令でプログラムを実行できるように、コンピュータの CPU は、複雑な操作をする命令を多数設けるように進化していった。

ところが、1970 年代後半になると、その様な複雑な命令の大多数が、実際のプログラムではほとんど使われていない実態が、IBM などの研究で判明した。この頃になると、高級言語でプログラムを開発し、それをコンパイルしてから実行する事が普通になっていたが、コンパイラが CPU の複雑な命令を活用仕切れていなかったのである。また、コンパイラを使う事が前提なら、アセンブリ言語で開発する場合と異なり、機械語のプログラムが長くなってしまっても、プログラムの開発に手間がかかる事はない。

そこで、複雑な処理をする命令をたくさん設けるよりも、簡単な処理をする命令を多数組み合わせてプ

ログラムを作るスタイルにする方が、CPU の回路規模が小さくなり、それで浮いた回路を、レジスタと呼ばれる高速メモリをたくさん内蔵するために使う方が、処理速度も向上するのではないかという考え方が出てきた。

この考えに基づいてできた、命令数の少ないコンピュータの事を、RISC(Reduced Instruction Set Computer)と呼ぶ。また、RISC が誕生する以前からある、複雑な命令を多数設けたコンピュータの事を、CISC(Complex Instruction Set Computer)と呼ぶ。

諸君が実習で使う PIC16F84A というマイコンも、Arduino Uno に使われている ATmega328P というマイコンも、RISC タイプのマイコンである。これらのマイコンは、アセンブリ言語でのプログラムの作りやすさよりも、コストを重視しているため、回路規模が小さくなる RISC アーキテクチャが向いているからである。なお、アーキテクチャとは、コンピュータの構造の事を指す。

・ノイマン型コンピュータ

私たちは、パソコンやタブレットコンピュータ、スマートフォンなど、色々な種類のコンピュータを使って暮らしている。これらのコンピュータの大多数はノイマン型というアーキテクチャを採用している。

コンピュータの扱う情報は、大きく分けてプログラムとデータに分けられる。例えば、スマートフォンで音楽を再生する場合を考えると、再生したい音楽のファイルはデータであるし、それを再生するためのいわゆる「アプリ」はプログラムである。

しかし、音楽のファイルにせよ、アプリにせよ、インターネット経由でダウンロードして入手できる点は同じである。それは、データもプログラムも、コンピュータ内部では 2 進数の数列として表現されている点で共通しているからである。

ノイマン型コンピュータというのは、CPU にバスを経由して、メモリ(RAM や ROM)が接続されており、そのメモリ内に、プログラムとデータを区別せずに記憶させる形式のコンピュータの事である。

図 1 に、ノイマン型コンピュータの概念図を示す。CPU(中央処理装置：命令を解釈し、プログラムを実行する装置)に、アドレスバスとデータバスを経由して、メモリがつながった構造になっている。なお、実際のコンピュータにはインターフェースを経由して周辺装置がつながっているが、この図は、話を簡単にするため、CPU とメモリの関係に焦点を絞って書いてある。

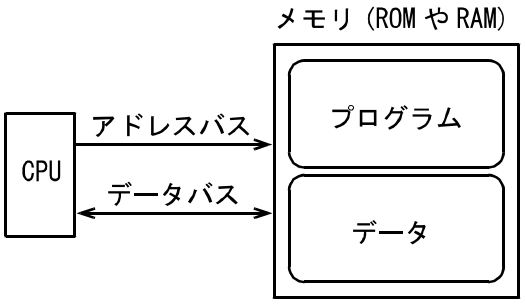
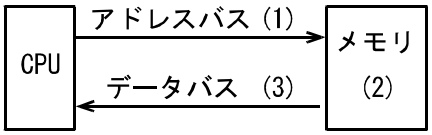


図 1、ノイマン型コンピュータの概念図
周辺装置やインターフェースについては省略している

図 1 のメモリは、具体的には 7 月 10 日の授業で説明した RAM や ROM の事である。RAM や ROM は、2 進数の情報を大量に記憶できるが、記憶した情報を特定するために、各情報に番号を振っている。この番号をアドレスという。

CPU とメモリの間は、アドレスバスとデータバスの 2 種類のバスでつながっている。バスとは、配線の束の事である。アドレスバスは、読み書きしたい情報のアドレスを伝達するためのバスで、常に CPU 側からメモリ側に情報が伝達される。一方のデータバスは、読み書きしたい情報を伝達するためのバスであるが、メモリに情報を書き込む際には、CPU 側からメモリ側に情報が伝えられる。またメモリから情報を読み出す際には、メモリ側から CPU 側に情報が伝達される。メモリに情報を読み書きする場合の手順を図 2 および図 3 に示す。

メモリが、複数のメモリ IC で構成される場合は、バスを分岐し、メモリ IC を並列に接続する。図 4 に、バスに ROM1 個と RAM1 個を接続した例を示す。

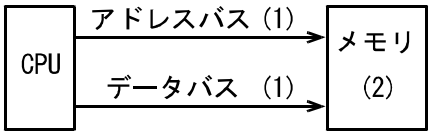


(1) 読み出したい情報のアドレスを、CPU がメモリに、アドレスバス経由で通知する。

(2) メモリが指定されたアドレスの情報を読み出す。

(3) メモリが CPU に、データバス経由で読み出した情報を伝達する。

図 2、メモリから情報を読み出す手順



(1) 情報を書き込むアドレスを、CPU がメモリに、アドレスバス経由で通知すると共に、書き込みたい情報を、データバス経由で通知する。

(2) メモリが、情報を指定のアドレスに書き込む。

図 3、メモリに情報を書き込む手順

この図の例では、ROM には 0H 番地～7FFF 番地、RAM には 8000H 番地～9FFF 番地のアドレスが割り当てられている。(アドレスの番号の後には、通常「番地」と書く。これは、住所になぞらえているからである)

CPU が ROM や RAM に読み書きする場合、対象となる情報のアドレスが、ROM と RAM の双方に通知されるが、通知されたアドレスが、自分に割り当てられているメモリ IC の方だけが反応する。例えば、1234H 番地から情報を読む場合は、そのアドレスが ROM と RAM の双方に通知されるが、ROM は、1234H 番地が自分に割り当てられているので、情報を CPU に送り返し、RAM は、1234H 番地が自分に割り当てられていないので、反応しない。

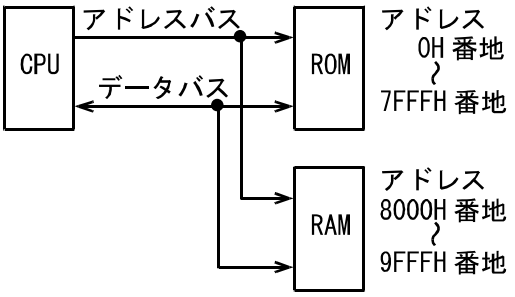


図 4、バスに ROM と RAM を接続した例

・ノイマン型コンピュータの利点

ノイマン型コンピュータは、アメリカの数学者、ジョン・フォン・ノイマンが 1946 年に考案したものであるが、この考え方は、当時としては画期的であった。

1 回目の講義で、ノイマン型コンピュータが考案されたのと同じ年の 1946 年に、ENIAC というコンピュータが発表され、これが世界初のコンピュータとされる事が多いと説明した。この ENIAC は、データはメモリに記憶させるものの、プログラムは、手作業で配線をつなぎかえる事で行っていた。そのため、プログラムを変えて、別の計算をさせるには、多数の配線をつなぎなおさなければならず、多大な手間がかかっていた。

一方でノイマン型コンピュータの場合、プログラムをデータと同様にメモリに記憶させておく事で、プログラムの変更は、メモリ内の情報を書き換えるだけで済むようになった。また、プログラムを CPU がメモリから読み取る機構は、データを読み取る機構をそのまま流用しているため、回路の規模が小さくなるという利点もある。これらの利点があるため、ノイマン型コンピュータは、21 世紀の今日でも、基本的に同じ原理のまま使われ続けている。

・ノイマン型コンピュータの問題点

ノイマン型コンピュータは、プログラムもデータも、同じバスを経由してメモリから読み出してくる。そのため、プログラムの読み出しと、データの読み書きを同時にできないという問題点がある。これが、計算速度を向上する上でネックになる事が多い。

・命令実行サイクル(教科書 P.38 参照)

CPU が記憶装置内のプログラム(機械語の命令の列)を実行する際には、一般に、フェッチ、デコード、実行の 3 段階に分けて行う。(ただし、PIC16F84A では、デコードと実行を同時に行う)

フェッチは、機械語の命令を 1 つ、記憶装置から読み出す作業の事である。

デコードは、読み出した機械語の命令が、どんな種類の処理をするための命令かを解読する作業のことである。

実行は、機械語の命令を、実際に実行する作業の事である。

以上の 3 つの段階の内、メモリにアクセスするのは、フェッチと実行である。フェッチにおいては、命令を必ず記憶装置から読み出す。また、実行においては、実行する命令に応じて、データをメモリから読み出したり、データをメモリに書き戻したり、あるいはメモリにアクセスしなかったりする。

・パイプライン(教科書 P.38 参照)

図 5、非パイプライン方式の CPU の命令の処理

初期のコンピュータにおいては、図 5 の様に、フェッチ→デコード→実行のサイクルを、命令ごとに繰り返していた。しかし、この方式では、フェッチしている時にはデコードを行う回路と実行を行う回路が停止しており、デコードを行う時にはフェッチを行う回路と実行を行う回路が停止しており、実行を行う時にはフェッチを行う回路とデコードを行う回路が停止している。この様に、回路の利用効率が悪いという問題点があった。

そこで、フェッチ、デコード、実行の回路を、同時に動作させる事のできる、パイプライン方式の処理方法が考案された。

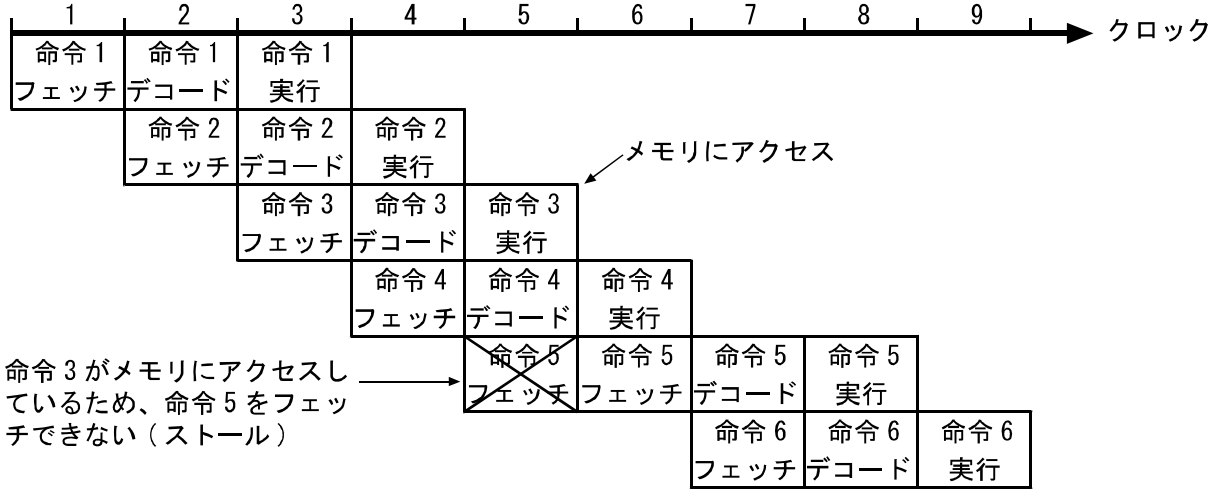


図 6、パイプライン方式の CPU の命令の処理

命令 1～命令 6 の内、命令 3 だけがメモリにアクセスする命令だと仮定している。

図 6 は、パイプライン方式の CPU が、命令 1～命令 6 の 6 つの命令を、順に処理している様子を表している。また、これらの命令の中で、命令 3 だけが、メモリにアクセスする命令だと仮定する。

3 クロック目を見ると、命令 1 を実行すると同時に、命令 2 をデコードし、命令 3 をフェッチしている事が分かる。この様に、パイプライン方式を使うと、フェッチ、デコード、実行の各処理を行う回路を、

同時に使う事ができ(これを処理の並列化という)、回路規模を大きくすることなく、CPU を高速化できることが分かる。

ところが、5 クロック目を見ると、命令 3 を実行する時にメモリにアクセスするため、既にバスが使用されており、命令 5 のフェッチができていない事がわかる。よって同時に処理できているのは命令 3 の実行と命令 4 のデコードだけであり、並列化の度合いが低下している。この様な現象をストールと呼ぶ。

6 クロック目を見ると、5 クロック目でストールが発生した影響で、命令 5 のフェッチがこのクロックから始まっており、命令 4 の実行と命令 5 のフェッチの 2 つしか処理できていない。やはり 6 クロック目でも並列化の度合いが低下している。

図 6 では、命令 3 だけがメモリにアクセスする命令だと仮定したが、実際にはメモリにアクセスする命令の割合はもっと多く、ストールは頻繁に生じると考えられる。

・ハーバード・アーキテクチャ(教科書 P.22 参照)

前節で、パイプライン方式を採用すれば処理を並列化でき、CPU を高速化できることを説明した。また、メモリにアクセスする命令があると、ストールが発生し、処理の並列度が低下してしまう事も説明した。

ところで、マイコンの場合に限定して考えると、プログラムは ROM に記憶させ、データは RAM に記憶させる事が多い。(パソコンの場合は、プログラムもデータも RAM に記憶させる。プログラムをハードディスクから RAM に転送してから実行する事で、ハードディスク内の色々なプログラムを実行できるようにしている)

プログラムとデータを別のメモリ IC に格納するなら、それぞれのメモリ IC に専用のバスを用意すれば、それぞれのメモリに同時にアクセスできるはずである。この考え方に基づいて考え出されたのが、ハーバード・アーキテクチャである。

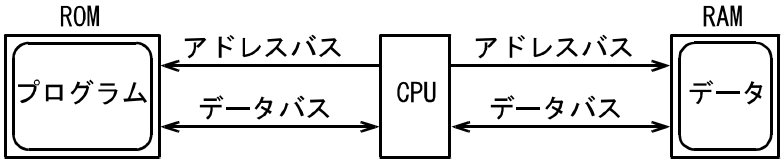


図 7、ハーバード・アーキテクチャの概念図
周辺装置やインターフェースについては省略している

図 7 は、ハーバード・アーキテクチャの概念図である。プログラムを記憶するメモリと、データを記憶するメモリのそれぞれに、専用のアドレスバスとデータバスがあることが分かる。なお、プログラムを記憶するメモリを ROM としているが、マイコンの場合は通常プログラムを ROM に記憶するからである。マイコン以外の場合は、プログラムを記憶するメモリが RAM のハーバード・アーキテクチャのコンピュータも存在する。

Arduino Uno に使われている ATmega328P というマイコンも、教科書で扱っている PIC16F84A というマイコンも、バーバード・アーキテクチャを採用する事で、パイプラインのストールを低減している。

前期定期試験について

前期定期試験の試験問題は、6 月 19 日から 9 月 11 日の講義で説明した範囲から出題する。試験時間は 90 分で持ち込みは不可である。一部選択問題で、一部必須問題となる。中間試験よりは難易度が上がるので、しっかり勉強し来ること。

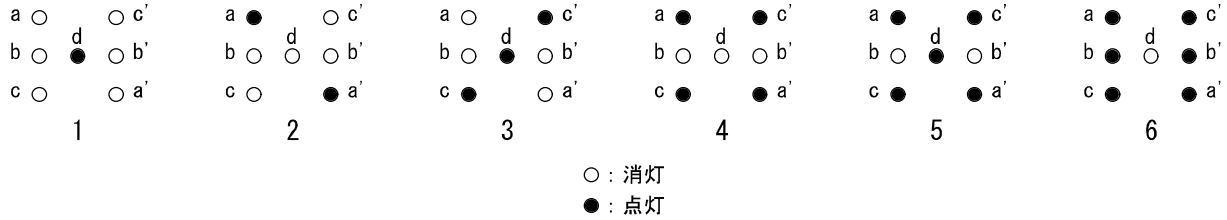
レポート課題

問題 1 を解くこと。また、問題 2～10 の中から好きな 2 問選び、解くこと。(解く必要があるのは合計 3 問)

レポートは、A4 のレポート用紙に書き、1 枚目の上部に名前と出席番号を書くこと。レポートの上部 2 箇所をホッチキスで止めること。また、解いた問題の番号は明記する事。

レポートは、10 月 9 日の講義の最後に集める。

【問題 1】(必須問題) 6 月 26 日に電子サイコロの設計法について説明したが、LED の点灯パターンを下の図の様に変更して、電子サイコロを再設計せよ。結果の回路図だけでなく、設計の過程も説明する事。また状態遷移図を描いて、フリップフロップがどのような初期状態であっても、一定クロック後には正常な電子サイコロとして機能することを確認する事。



LED の点灯パターン

6 月 26 日のプリントの図 2 とは、2 の目だけ点灯パターンが異なる

【問題 2】(選択問題) (1)～(3)の問いに答えよ。

(1) チャタリングの意味を説明せよ。

(2) チャタリングが発生すると、どういう不具合が発生するか説明せよ。

(3) チャタリングを除去する回路の例を挙げよ。

【問題 3】(選択問題) 6 月 19 日の講義プリントを参考に、回答者が 4 人の場合の早押し判定回路を設計せよ。

【問題 4】(選択問題) (1)および(2)の問いに答えよ。

(1) 電源に入れるバイパスコンデンサ(パスコン)の働きについて説明せよ。

(2) パスコンにはどのような種類のコンデンサを選ぶべきか、説明せよ。

【問題 5】(選択問題) ハザードの意味を説明せよ。説明には、ハザードの発生する回路の例や、そのタイミングチャートも書くこと。

【問題 6】(選択問題) SRAM と DRAM の違いを、動作原理、特徴(特性)の観点から説明せよ。

【問題 7】(選択問題) フラッシュメモリの動作原理、特徴、用途について説明せよ。

【問題 8】(選択問題) キーボードから入力した 10 個の整数の中で、一番小さい数を表示するプログラムの、フローチャートを書け。また、そのフローチャートに基づいて、C 言語または Pascal でプログラムを作れ。

【問題 9】(選択問題) (1)および(2)の問いに答えよ。

(1) ノイマン型コンピュータと、ハーバード・アーキテクチャを採用したコンピュータの構造の違いを、図を用いて説明せよ。

(2) パイプライン方式の CPU と非パイプライン方式の CPU とにおける命令の実行の仕方の違いを、図を用いて説明せよ。

【問題 10】(選択問題) (1)および(2)の問いに答えよ。

(1) スタックおよびスタックポインタの働きについて、図を用いて説明せよ。

(2) PIC16F84A のスタックは、一般的なコンピュータと構造が異なる。どのように異なるか説明せよ。