１．「２進数から始めよう」

いま、みんなが普通に使っている数字は１０進数だ。すなわち、０から出発して１、２、・・・、９と数えていって、その次の数字は「１０」の２桁となる。要するに「１０」を表す１桁の数字を決めてなく、「９」の次は１つ上の桁を「１」として「０」に戻している。もう少し詳しく書くと、「０」は「００」、「１」は「０１」であり、「９」は「０９」となる。たとえば、数字の最大桁数を１０桁と決めると、表せる数は

「０００００００００」～「９９９９９９９９９」

までとなる。当然、桁数が多ければ多いほど、表せる数字は大きくなる。

１０進数の他に、日常何気なく使っているものに時間がある。これは、６０秒が１分、６０分が１時間となるように、６０になると上の桁（この場合は桁ではないけれども）に上がる。たとえば、「２時」を「１時６０分」と言ったりすることはない（中にはいるかもしれないけれど）。このような桁の上がり方は６０進数である。

さて、話の方向を調整して、「２進数」に向けよう。いま、Ａ君とＢ君が川の両側にいて、Ａ君がＢ君に合図をしようとした場合、君だったらどうする？。大声を出す。聞こえなかったら？。石を川に投げる。投げた所をＢ君が見てなかったら？。色々やり方はあるけれど、手を上げ下げするなんてどうかな？。結構いい考えじゃないかな。そうだそうだ。でも、手を横にするなんてことを考えたら、ここでの話が進まないので、やめよう。それじゃ、手を上げ下げする方法でＡ君がＢ君に数字を送ろうって思って、二人で話し合って、「手を下げたとき」を「０」、「手を上げたとき」を「１」なんてことを決めてしまったのだ。それで、二人でそのゲーム（ゲームと言う程、遊べない）をしばらくしていたが、Ｂ君はほとんどおもしろくなくなってしまった。なんと言っても、「０」と「１」しかないのだ。そこで、Ａ君は二人より少し賢いＣ君を呼び出した。ちょうどいいことにＣ君はなんと「２進数」の勉強をしたばかりだった。Ｃ君は「二人並んで手を上げ下げしよう」と言い、やり始めたがＢ君は説明を聞いてないので、さっぱりわからなかった。そこで、Ｃ君は次のように説明した。「二人とも手を下げたときは０、Ａ君が上げて僕が下げた時は１、Ａ君が下げて僕が上げた時は２、二人とも手を上げたときは３たい、わかった？」。そして、次のような図を書いた。

Ｃ君	Ａ君	数字	
0	0	0	０は手を下げた時
0	1	1	１は手を上げた時
1	0	2	
1	1	3	

Ｂ君は「なんだ、簡単じゃないか」なんて言ったりした。その時だった。Ｂ君はおもしろいことに気が付いた。「じゃあ、二人の数字には２なんて数がないんだね。」Ｃ君は「そうだよ。これが２進数たい」と言って笑ったりした。納得したＢ君たちは夕方までそれで遊んでいたが少し飽きてきて、みんな家に帰ってしまった。

ここで、もう少し専門的に話をしよう。コンピュータはなんと言っても電気で動作する。電気的に「０」と「１」を表す時には、例えば、乾電池に豆電球を付けてスイッチを入り切りすれば電球がついたり消えたりする。Ａ君たちの遊びでは、豆電球を２個使って、それぞれを入り切りすれば同じことである。でも、２個では０から３までの数字しか表せないので、さらに大きい数字を表すには電球がたくさんいる。コンピュータに対応させれば、電球が１６個並んでいると考えればよい。し

かも、１６個の電球は８個づつ２組に分けて考えることもできるというわけである
そうすると、８個では、

０００００００００～１１１１１１１１

まで表せる。それぞれの電球を「ビット（Ｂｉｔ）」という。「ビット」とは「かけら、小片」などという意味である。「８ビット」をひとかたまりにしたものを「１バイト（Ｂｙｔｅ）」という。また、「１６ビット」は「１ワード（Ｗｏｒｄ）」という。Ｃ君の説明に従えば、８ビットでは

０～２５５　　すなわち　０～（２の８乗）－１
（表せる数値は２の８乗）

１６ビットでは

０～６５５３５　　すなわち　０～（２の１６乗）－１
（表せる数値は２の１６乗）

となる。
電気信号でこの「０」、「１」を表すために電圧の大きさで定義している。すなわち、ＴＴＬレベルと言う電圧の大きさを決めている。ＴＴＬとは「Transistor-Transistor Logic 」である。ＴＴＬでは「０」が０Ｖ、「１」が５Ｖである。

２．「慣れるまでに時間がかかる１６進数」

あまり勉強が好きでないＢ君が、どういう訳か昨日Ｃ君に習った「２進数」の事を思いだして紙に色々書き始めた。少し前に「１６ビットコンピュータ」とか言う言葉を聞いた事があったので、「１６ビット」についてお勉強をした。ところが、どうも面倒くさくて、やってられなくなってきた。なんと言っても、０と１を１６個もずらずら書かないといけないからだ。何かよい方法がないか本で調べるのがいやなＢ君は、さっそく少し賢いＣ君に電話した。
Ｂ君：「２進数の勉強をしてたけど、０と１がいっぱいありすぎて、なんかごちゃごちゃになって、ようわからんけど、どうしたらよかや？」
なんて言うと、Ｃ君は次のように答えた。
Ｃ君：「当り前たい。そんなもんいちいち書きよったら、俺だってようわからんようになるばい。そういう時は、４ビットづつ区切って表したらよかたい」
Ｂ君：「ほほー。でも、４ビットづつ区切っても０は０、１は１やろも」
Ｃ君：「だけん。４ビットを一つの数で表すったい。４ビットやったら、０から１５まで表せるき、１６ビットやったら４つの数で表せるやんか」
Ｂ君：「そしたら、１０から１５まではどうするん？」
Ｃ君：「１０をＡ、１１をＢ、１２をＣ、１３をＤ、１４をＥ、１５をＦときめるったい。わかった？。結局、４ビットで０からＦまで表せるやろが。Ｆち言うのは１５だから、１６になったら、次の４ビットに桁上がりするったい。紙に書いて、調べてん。これが１６進数って言うったい。よかや」
Ｂ君「わかった。わかった。やってみるけん」
と言って、Ｂ君は調べ始めた。Ｂ君は１０から１５までと、ＡからＦの対応がピンとこなくて、指を折りながら何回も数えていた。

ここで、少し専門的にまとめてみよう。１６ビットでは

０００００００００００００００　（１０進数で０）

から

１１１１１１１１１１１１１１１１　（１０進数で６５５３５）

まで表せる。これを４ビットずつに区切って

０００００００００００００００

とすると、それぞれの４ビットは

０００００　～　１１１１

まで表せるので、これを１桁の数で表すが、１桁の数は０から９までしかないので、新しく１０から１５に対応する１桁の数を決めないといけない。そこで

１０進数	１０	１１	１２	１３	１４	１５
１６進数	Ａ	Ｂ	Ｃ	Ｄ	Ｅ	Ｆ

とすると、１６ビットの数は、１６進数では

０００００　～　ＦＦＦＦ

として表すことができる。このような表現法が１６進数である。「０００００」から数えていって「０００Ｆ」の次は「００１０」、また「００ＦＦ」の次は「０１００」となる。
最初のうちはＡからＦがどの数に対応するのかピンとこないので、Ｂ君がやっているように、指を折りながら数えて、慣れないとしようがないでしょう。また、４ビットの２進数、１０進数、１６進数の対応表なんか作っておくのも良いと思います。
通常、これらの数を区別して表すのに、次のようにＤ、Ｂ、Ｈを付けたりします。

１０進数	４６６０Ｄ
２進数	０００１００１０００１１０１００Ｂ
１６進数	１２３４Ｈ

Ｄは「Decimal」、Ｂは「Binary」、Ｈは「Hexadecimal」の意味です。

３．「負の数なんてどうするの」

次の日、２進数を勉強したＡ君がＢ君に次のような質問をした。
「２進数で足し算やら引き算とかできるや？。たとえば、簡単なやつで、

	０００００	０００００	０００１	００１１
＋	０００００	０００００	０００１	０１００

なんかどう？」
Ｂ君は１６進数については少しマスターしてたが、足し算はやっていなかった。し

かし、もともと感の良いＢ君は次のように書いた。

「００００　００００　００１０　０１１１」

「すげー」なんて言ってＡ君はびっくりした。Ｂ君も「やったね。簡単やん」と言って自分で感心した。「そしたら、これはできんやろうね」と思って、Ａ君は次のような引き算の問題をだした。

「　００００　００００　００００　０１００
　－００００　００００　００００　１０００　」

Ｂ君は少し考えて「なんやこれ。計算できんやん」と言った。Ａ君は「難しいやろ」と言って自分のだした問題を見たが、まずい事に気が付いた。なんと答えが負の数になってしまうのだ。Ａ君は負の数の表し方までは勉強してなかった。「問題が間違っとった」なんて言えないので、Ｃ君に「できる？」とか質問した。Ｃ君はあっさり次にように答えを書いた。

「１１１１　１１１１　１１１１　１１００」

Ａ君はＣ君に「Ｂ君がわからんみたいやき、ちょっと説明してよ。」と言ってその場を逃れた。Ｃ君は次のように説明した。

「これは、２の補数表現という負の数の表し方たい。１６ビットやったらめんどくさいき、８ビットで説明するね。１６ビットでもおんなじたい。２進数で負の数を表すときは、その１番上のビットが０か１で正負を表すように決めてるんだよ。従って

　　０ＸＸＸＸＸＸＸ

の時は正の数

　　１ＸＸＸＸＸＸＸ

の時は負の数たい。
だから、正の数の時には

　　００００００００～０１１１１１１１（１０進数で１２７）

負の数の時には

　　１１１１１１１１～１００００００００（１０進数で－１２７）

となる。例えば、

　　０００００００１

は１０進で「１」

　　１１１１１１１１

は１０進で「－１」となる。この時「０００００００１」に対する２の補数が「１１１１１１１１」たい。当然

　　　０００００００１
　　＋１１１１１１１１
　　１０００００００００

で、下の８ビットは「０００００００００」で１０進で「０」となる。２の補数を作るには「２の８乗」を基準にして

　　「１０００００００００」

から引けばよい。すなわち、「００００００１０」の場合には

```
  １００００００００
－　０００００００１０　（１０進で２）
  １１１１１１１１０　（１０進で－２）
```

として求められる。あるいは、各ビットを反転して１を加えてもよかたい。要するに「０００００００１０」では

「１１１１１１０１＋１＝１１１１１１１０」

となる。負の数から２の補数を求めるときも同じなのだ」

Ｃ君の説明に対して、Ｂ君は当然わからないが、Ａ君もわからなかった。Ａ君はＣ君の説明に対して、「今の説明で、９ビットになっているところがあるけど、これは何？」と言って首をかしげた。Ｃ君はさらに説明を続けた。

「いま、０００００００１０（１０進で２）と１１１１１１１０（１０進で－２）を足し算するとき

```
  ０００００００１０
＋１１１１１１１１０
  １００００００００
```

となる。ビット数を８ビットで考えているので、答えは下の８ビットだけとなり、９ビット目の「１」は「桁上げフラグ」という所に記憶されるったい。だから答えは「０」でいいやない。でも、この計算は８ビットの正負表現に対して正しい答えが得られ、正の数だけの表現ではちょっとおかしい。どうしてかと言うと、答えが８ビットを越えているからだ。当然、９ビットでは正しい答えとなるけど。わかった？。わからんやったら、今日、家に帰ってもう一度勉強しなさい」なんて言ったので、Ａ君とＢ君は「そうだね」とか言って笑った。

Ｃ君の説明にもう少し付け加えよう。足し算や引き算すなわち、加算と減算を行う場合には、扱う数を「正の数のみ」なのか、あるいは「正負表現」なのかを考えておく必要がある。１６ビットでは、

正の数のみの場合は　０～６５５３５

正負表現の場合は　　－３２７６８～３２７６７

の数までしか扱えない。従って、この値を越えない範囲で計算を行った場合に正しい答えが１６ビットとして得られる。

コンピュータは扱う数字が「正の数のみ」でも「正負表現」の場合でも同じ演算を行い、その結果を求める。だから、プログラマがそれを決めてプログラムを作る必要がある。

```
  ０００００００１
－０００００００１０
  １１１１１１１１１
```

上のような演算を行った時には「正負表現」では８ビットに正しい結果が得られるが、「正の数のみ」では意味がない。この時には「桁借りフラグ」というビットが１（９ビット目に対応）となる。８ビットでは、当然

正の数のみの場合は　０～２５５

正負表現の場合は　　－１２８～１２７

までの数しか扱えない。

4．「マイクロコンピュータの歴史も少しくらい知っててもいいと思う」

８０系
１９７１年　インテル社　４００４－４ビットＣＰＵ
（Central Processing Unit - 中央処理装置）
８００８－４ビットＣＰＵ
１９７３年　８０８０－８ビットＣＰＵ
（現在の８０系の先祖）
１９７６年　８０８５Ａ（８０８０を改良）
ザイログ社　Ｚ８０（８０８０を元に機能を充実）
１９７８年　インテル社　８０８６，８０８８－１６ビットＣＰＵ

６８系
１９７４年　モトローラ社　８ビットＭＰＵ６８００
（Micro Processing Unit - マイクロプロセッサ）
１９７９年　ＭＰＵ６８０９－８ビット
１６ビットＣＰＵ－６８０００

卒論で使用するコンピュータ（ＮＥＣ－ＰＣ９８０１ＶＸ）は１６ビットコンピュータであり、ＣＰＵは８０８６とソフト互換性をもつＶ３０（μＰＤ７０１１６－１０）である。

5．「メモリってよく聞くけど」

Ａ君はＣ君に「メモリ」の事を聞いた。詳しい事まで知らないＣ君は簡単に次のように説明した。
「メモリ」というのは「記憶する」と言う意味だから、「０」か「１」かを記憶しておく所たい。「０」か「１」だけだったら１ビットの記憶箇所があればいいということたい。でも、「０」か「１」かを１０箇所の別々の所に記憶するときには１ビットの記憶箇所が１０個いる。その時に、どの箇所（場所）に記憶するかを決めるために、それぞれの場所に順番に番号を付けておく必要があるったい。この番号のことを「番地（アドレス）」と言うんだ。そして、記憶する「０」か「１」の値を「データ」と言うんだよ。普通、「番地」は「０番地」から始めるようになっているので、この場合は、０番地から９番地ということだね。いま、考えているのは「０」か「１」かの１ビットで表せるデータだから、１ビットを表せる信号線、すなわち１本の信号線があればいいでしょ。この信号線のことを「バス（ＢＵＳ）」といい、「データ」を送る（乗せる）信号線のことを「データバス」という。それから、「番地」も２進数の値で決めるわけだから、この場合は「９番地」まで表せないといけないので、４ビット必要だね。だから、４本の「番地」を送る（乗せる）信号線がいることになるやろ。この「番地」を決めるための信号線のことを「アドレスバス」と言うったい。でもね、４ビットだったら「００００（１０進数で０）」から「１１１１（１０進数で１５）」まで表せるので、本当は番地は１５番地まで決めれるのだ。こんな風に、「データバス」が１本、「アドレスバス」が４本で決まるような「メモリ」は全部で１６ビットのデータを記憶できるので、「１６ビットメモリ」あるいは、８ビットは１バイトだから「２バイトメモリ」とか言うんだよ。でも、こんな「メモリ」はなくて、普通はデータは８ビットの単位として構成されているので、「データバス」は８本ある。また、「アドレスバス」ももっと多いよ。例えば、８ビットデータバスでアドレスバスが１０本の場合には、記憶でき

る容量は「（２の８乗）×（２の１０乗）」だから「２５６×１０２４ビット」となるったい。２進数では（２の１０乗）＝１０２４を「１Ｋ」というふうに表すので、「２５６Ｋビット」なのだ。これは「３２Ｋバイト」だし、「１６Ｋワード」と言ってもいいったい。まあ、今の僕の知識ではこれくらいかな。
Ｂ君は少しわかったような気がして、気分がよかった。

普通、「メモリ」には大きく分けて２つのタイプがある。
ＲＯＭ（Read Only Memory）：読みだしのみができるメモリであり、電源を切っても記憶内容は消えない。
ＲＡＭ（Random Access Memory）：書き込み、読みだし共にできるメモリであるが、電源を切ると記憶内容も消える。

６．「ＰＣ９８０１のメモリ構成はどうなっているんだ」

メモリ・マップは次のようである。

アドレス	メモリ	メモリ内容
ＦＦＦＦＦ Ｆ０００００	バンク１５	ＢＡＳＩＣ・ＲＯＭ
Ｅ０００００	バンク１４	グラフィックＶＲＡＭ３
Ｄ０００００	バンク１３	システム予備
Ｃ０００００	バンク１２	システム予備（ＤＳＰボード）
Ｂ０００００	バンク１１	グラフィックＶＲＡＭ０～２
Ａ０００００	バンク１０	テキストＶＲＡＭ
９０００００	バンク９	ＲＡＭ６４０ＫＢ
８０００００	バンク８	
７０００００	バンク７	
６０００００	バンク６	
５０００００	バンク５	
４０００００	バンク４	
３０００００	バンク３	
２０００００	バンク２	
１０００００	バンク１	
００００００	バンク０	

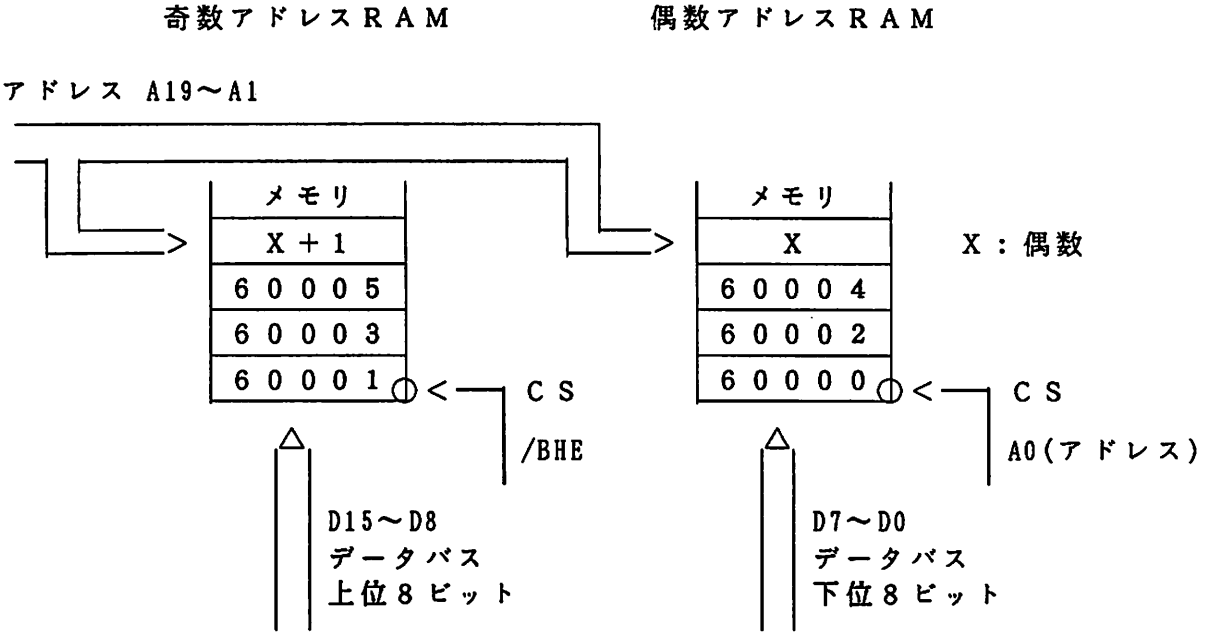
００００００番地から１ＫＢは割り込みベクトル領域

１バンクは６４ＫＢ（「Ｂ」はバイトを表す）なので、全メモリは６４×１６ＫＢ＝１ＭＢ（メガバイト）である。従って、ＣＰＵは１ＭＢのメモリを使用している。Ａ００００番地からＦＦＦＦＦ番地まではＶＲＡＭなどＰＣ９８０１のシステムで使用される領域である。ユーザーは０番地から９ＦＦＦＦ番地でプログラムできるが、下位番地の領域は「ＤＯＳ（Disk Operating System)」のシステムプログラムが占める。先の方で出てくるが、「Ｃ言語」によるプログラムで実行する場合

には、このユーザーが使用できる領域はあまり考えなくてもよい。しかし、「割り込みベクトル領域」と「バンク１２（ＤＳＰボード）の領域」は重要であるから、頭に入れておかなければならない。「ＶＲＡＭ」とは「Video ＲＡＭ」のことであり、ディスプレイに表示するテキスト画面とグラフィック画面の情報を記憶するメモリである。メモリ構成に示した番地は「物理アドレス」といい、１６進数で５桁すなわち、「２０ビット」でそのアドレスを表していることに注意しよう。

７．「メモリ配置も少し知っておこう」

８０８６（Ｖ３０）ＣＰＵは１６ビットコンピュータである。１６ビットと言うのは「１６ビットのデータを１度に転送あるいは処理できる」ことを意味しているまた、８ビット単位でも同じことができる。そうすると、メモリは８ビット単位で構成した方がよいことになる。実際、メモリは８ビット単位で構成されており、前のメモリマップの１つのアドレスは８ビットのデータを記憶する。マップでは表現上、連続してアドレスを書いているが、実際のメモリ（ＲＡＭ）の配列は違っている。



図のように、偶数アドレスと奇数アドレスに対応した８ビットＲＡＭが２０ビットアドレスの内のA19-A1に並列接続されており、偶数アドレスのＸを指しているときには、奇数アドレスは（Ｘ＋１）を指すようにする。ＲＡＭにはそれぞれＣＳ端子（ＣＳ：チップセレクト＝チップ選択）があり、この信号が「０」の時に、そのアドレスのデータをデータバスに出力する（これは読み出しの場合で、書き込みの時には、データバスのデータをそのアドレスに取り込む。図では、読み出しか書き込みかわからないが、実際は、どちらなのかを識別する別の端子がある。）。それぞれのＲＡＭは偶数アドレスＲＡＭがデータバスの下位８ビット、奇数アドレスＲＡＭがデータバスの上位８ビットに接続されている。／ＢＨＥはＣＰＵから出力される信号で、奇数アドレスのデータを処理するときに「０」となる。Ａ０はアドレス信号の最下位ビットなので、偶数の時には「０」、奇数の時には「１」となる。８ビットデータの時には上位または下位のみなので、特に問題はないが、１６ビッ

トデータを扱う場合には次のようなことが起こる。偶数アドレスＲＡＭに下位８ビットデータ、奇数アドレスＲＡＭには上位８ビットデータとする時には、１度に処理できる。しかし、奇数アドレスＲＡＭに下位８ビットデータ、偶数アドレスＲＡＭに上位８ビットデータとする時には、（Ｘ＋１）と（Ｘ＋２）となるので、１度に１６ビットデータは処理できず、２回にわけて処理しなければならなくなる。すなわち、（Ｘ＋１）と（Ｘ＋２）ではアドレスバスA19-A1の値が異なる。従って、１６ビットデータを扱うときには偶数アドレスが下位データとなるように定義しないと効率が悪くなり、結果として、プログラムの実行時間が長くなる。実際のプログラム作成では、この点は十分注意しておく必要がある。

８．「８０８６ＣＰＵの端子も見ておこう」

８０８６ＣＰＵの端子は次のようである。

ＧＮＤ ―――	1	40	―――ＶＣＣ
ＡＤ１４＜ ― ＞	2	39	＜ ― ＞ＡＤ１５
ＡＤ１３＜ ― ＞	3	38	―― ＞Ａ１６／Ｓ３
ＡＤ１２＜ ― ＞	4	37	―― ＞Ａ１７／Ｓ４
ＡＤ１１＜ ― ＞	5	36	―― ＞Ａ１８／Ｓ５
ＡＤ１０＜ ― ＞	6	35	―― ＞Ａ１９／Ｓ６
ＡＤ９＜ ― ＞	7	34	―― ＞／ＢＨＥ／Ｓ７
ＡＤ８＜ ― ＞	8	33	＜ ―― ＭＮ／／ＭＸ
ＡＤ７＜ ― ＞	9	32	―― ＞／ＲＤ
ＡＤ６＜ ― ＞	10	31	＜ ― ＞／ＲＱ／／ＧＴ０
ＡＤ５＜ ― ＞	11	30	＜ ― ＞／ＲＱ／／ＧＴ１
ＡＤ４＜ ― ＞	12	29	―― ＞／ＬＯＣＫ
ＡＤ３＜ ― ＞	13	28	―― ＞／Ｓ２
ＡＤ２＜ ― ＞	14	27	―― ＞／Ｓ１
ＡＤ１＜ ― ＞	15	26	―― ＞／Ｓ０
ＡＤ０＜ ― ＞	16	25	―― ＞ＱＳ０
ＮＭＩ ―― ＞	17	24	―― ＞ＱＳ１
ＩＮＴＲ ―― ＞	18	23	＜ ―― ／ＴＥＳＴ
ＣＬＫ ―― ＞	19	22	＜ ―― ＲＥＡＤＹ
ＧＮＤ ―――	20	21	＜ ―― ＲＥＳＥＴ

ＣＰＵは４０ピンをもつＤＩＰ（Dual In-line Package）のＬＳＩ（Large Scale Integrated Circuit）である。全てのピンの働きを理解する必要はないので、プログラムに関係したものだけを述べる。

８０８６は１６ビットＣＰＵなのでデータバスは１６本必要である。また、物理アドレスを出力するために２０本のアドレスバスが必要である。８０８６では、下位１６ビットのアドレスピン（Ａ１５～Ａ０）はデータピンと共用し、時分割によってその働きをバス・コントローラからの信号で制御している。このようなタイミングは普通、考えないでよいので、基本的にアドレスバス２０本、データバス１６本と考えておいてよい。アドレスピンＡ１６～Ａ１９はアドレスを出力しない時には、ステータス信号としても利用している。ＲＤピンは、この信号が「０」の時にＣＰＵがメモリに対して読み出し動作であることを示す。Ｓ０～Ｓ１はＩ／Ｏやメ

モリに対する処理のためのステータス信号を出力する。
　重要なのは、ＩＮＴＲピンである。これはＣＰＵに対する割り込み要求信号を受ける。要求信号は割り込みコントローラ（８２５９）というＬＳＩを通して送られＣＰＵはこの信号を受けると、８２５９で受けた割り込みタイプの割り込みベクトルを参照して割り込みルーチンへ移る。この割り込みについては、後で詳しく述べるので、ここでわからなくてもあまり気にしなくてよい。

９．「１６ビットＣＰＵなのにアドレスが２０ビットなんだ」

　メモリのことがだいぶわかったＢ君は、また、少し悩んでいる。というのは、１６ビットＣＰＵと言うのに、なぜ、アドレスは２０本あるのか、ということだ。そこで、またまたＣ君に質問した。Ｃ君は、「１６ビットＣＰＵというのは、１度に処理できるデータが１６ビットということで、アドレスのビット数とは関係ないったい。これは前に出てきたやんか」と言ったので、Ｂ君は笑って納得した。
　８０８６は２０ビットのアドレスをもっているので１ＭＢまでのメモリを指定できる。逆に言えば、１ＭＢのメモリを指定できるようにするために、２０本のアドレスを設けたと考えてもよい。

１０．「２０ビットアドレスの決め方が問題だ」

　しばらく顔を出さなかったＡ君は、実は秘かに８０８６ＣＰＵの勉強をしていたのだ。それで、優越感に浸ろうとＢ君を呼び出して次のような質問をした。
Ａ君：「メモリの番地を決めるには２０ビットいるけど、どうやって決めるか知っとうや？」
Ｂ君：「そんなもん、２０ビットで決めたらよかやん」と真剣に答えた。
Ａ君：「あほ。ＣＰＵの中はすべて１６ビットやけん、決められんったい」
Ｂ君：「そんなもん、俺が知る訳ないやろが。聞くな」と言って、
　　「どげんすると？」なんて聞き返した。
Ａ君はうきうきした。
Ａ君：「あのね。それはセグメントとオフセットと言う方法で決めるったい」
Ｂ君：「セグメントとオフセット。なんやそれ」
Ａ君は説明を続けた。
「２０ビットのアドレスを１６ビットで表すと４ビット、すなわち１６進数で１桁たらんやろが。それで２０ビット、１６進数で５桁を上４桁と下４桁に分けるったい。分けるち言っても、その分け方が問題たい。例えば、物理アドレス（Ａ君は結構勉強していて「物理アドレス」も知っていた）で１２３４５Ｈというのをこんなふうに表すわけたい。

```
  １０００      セグメント
＋　２３４５    オフセット
  １２３４５
```

セグメントの最下位桁は、書いてないけど「０」として計算するんだ。セグメントの値はＣＰＵ内部の「セグメント・レジスタ」という所に記憶され、プログラムで書き直さない限り、その値は変化せん。オフセットの値は演算で使用する汎用レジスタなんかに入っている。このセグメントやオフセットの値のことを「論理アドレス」って言うんだよ。同じ１２３４５番地でも色々な表し方があるったい。例えば、