アーキテクチャ　第４回　追加メモ　ＣＰＵ内部構造の追加メモ

初めにお願い

最初の授業で言ったことだが、質問は授業中に発言して欲しい。その場で共有するのが一番能率が良いと思う。「質問シートに書いたから」だと、その場で共有されないので、あとで１人１人に説明しなければならないし、説明をしない人もたくさん残る。だから、シートに書いてきたとしても、それを授業の場で質問して欲しい。
では、共通に出てきた質問について補足する。

デコーダとは

（組合せ）論理回路の一種で、N本の入力線と、2^N本の出力線から成る組合せ論理回路である。組合せ論理回路とは、要するに論理式で書ける論理（正確には、現在の入力のパターンだけで出力が決まるような論理）をもつ回路である。
（注： 出力が過去の入力パターンにも依存するときは、組合せ論理回路ではなく、順序論理回路と呼ばれる。 この場合、内部に過去の入力を記憶するような「状態」を持っている。）

I2	I1	O0	O1	O2	O3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

デコーダの真理値表は、入力のパターンを2進数 i だと見なしたとき，2^N本の出力線のうちの i 番目の１本だけが1となり、その他の出力線はすべて0となる。右の例は、入力線が2本、出力線が2^2＝４本の場合である。 入力線を I2 と I1（I2 の方が上位のビットと見なす）、出力線を O0～O3 とすると、入力のパターンが 00 の時は出力 O0 だけが1となり、入力パターンが 01 の時は出力 O1 だけが1となり、10 のときは O2 だけが、11 の時は O3 だけが1になる。 つまり、2本の入力線の組合せパターンを、出力側の「どれか１つが1になる」という形に変換する。イメージとしては、2ビットの2進数を入力して、その数が 0～3 のどれであるかを判定して、その番号のランプを付ける、というイメージである。このような回路をデコーダと呼ぶ。
逆方向の変換、つまり0～3の4本の入力線があってどれか１本だけが1になるという状況の時に、2本の出力線上に2進のパターン 00, 01, 10, 11 のどれかが出力される、という回路を、エンコーダと呼ぶ。

COMET-II とは何か、COMET-II と CASL-II の違いは何か

COMET-II の説明のページ http://www.jitec.jp/1_13download/shiken_yougo_ver2_0.pdf を参照のこと。
基本情報処理技術者試験でサンプル機として用いる仮想的なコンピュータ。 実体は売られていない（試作した人はいる）。必要最小限の構成にしてあり、初心者レベルでもわかりやすいことを狙っている。
COMET-II は CPU の種類の名前であり、 CASL-II は COMET-II という CPU の機械命令を（ニーモニックコードから機械命令に）翻訳するアセンブラである。 アセンブラとは、人間が読めるニーモニックコード（やその他もう少し要素がある）で書かれた「アセンブリプログラム」（Java のソースコードに対応）を、機械が読める機械命令（2進数で書かれた機械命令）に、翻訳・変換するプログラムである。 後の回でアセンブリプログラムをいろいろと作ってみる機会がある。

汎用レジスタとメインメモリ（主記憶）の違いは何か

後の回の「記憶装置」の項で詳しいことを学ぶことにしよう。 ここでは簡単に答えておく。
先に「メモリ」という言葉をきちんと理解しよう。「メモリ」という言葉は（文脈によって）いろいろな意味に使われるので、どの意味で使われるのかを正しく認識しなければならない。 具体的には
① メモリ＝記憶装置一般をさす つまり、記憶をする装置はすべてメモリと呼んでよい。 コンピュータの構成図で見た「メインメモリ（主記憶）」だけでなく、ハードディスクや CD-ROM、DVD、USB メモリなどの「補助記憶」もメモリである。 また、CPU の中に置かれている、汎用レジスタ、フラグレジスタ、命令レジスタなどのレジスタ類も、記憶装置でありメモリである。

② 計算機の構成図で「メインメモリ」または「主記憶」と書かれている箱のことを、「メイン」を略して「メモリ」と呼ぶことがある。 たとえば「この計算機のメモリは2ギガバイト」と言った時には、「メモリ」とは「メインメモリ」のことを指している。

次に、汎用レジスタとメモリの違いについて。 汎用レジスタは、CPU の中に置かれたメモリ（記憶装置）の一種である。COMET-II の例では、1つ1つの要素の大きさが16ビット（2進16桁）を記憶でき、その要素が全体で8個（GR の 0 番から 7 番まで）備えられている。つまり、8個のうち1つを選択するには GR3 のように番号で指定する。 汎用レジスタは（メインメモリに比較すると）一般に個数は少ない（8個とか16個とか、多いものでも32個とか）が、その代り高速に読み書きできる。
他方、メインメモリ（主記憶、時々単に「メモリ」と呼ぶ）は、CPU の外に置かれ、記憶できる場所（引き出し、箱）の数は非常に多いが、CPU から読書きする速度はやや遅い。 具体的には、そこらにある PC のメインメモリは、PC の用途や値段によって異なるが、だいたい1～8ギガバイト程度のメインメモリを持つ（ギガ＝10^9で、バイト＝8ビット）。 また COMET-II では 65535 語分（1語＝16ビット＝2バイト）に設定されている。 いずれにせよ8個程度と比べると桁違いに多い。 速度を比較すると、汎用レジスタは、1ナノ秒以下で（ナノ＝10^{-9}）、読み書きすることができ、CPU のクロック速度の程度に速い。 メインメモリの読み書きの時間は数ナノ秒～10ナノ秒ぐらい（実際はブロックで読み書きするので、読書き時間を論ずるのはかなりややこしい）であり、汎用レジスタに比較すると10倍以上遅い感じになる。1つの命令中でメモリから値を読み出す必要がある場合は、実行時間を伸ばしているようなプロセッサが多い。
使い分けを考えてみる。 汎用レジスタもメインメモリも、計算の入力や出力を置いておくことに使う。 たとえば次々に値を足していくときなど、最初の足し算の結果を一旦汎用レジスタに格納し、次の足し算でそのレジスタの内容と次に足す値を加えるように使う。つまり、計算の途中結果を一時的に置いておく場所として、汎用レジスタを使う。これはもちろん、メインメモリに一時的に置いても構わないのであるが、メインメモリより汎用レジスタの方が速いのであるから、汎用レジスタに置いた方が計算が速くなる。 他方、一時的に置くと言っても数が多いと、8個しかない汎用レジスタには置き切れなくなる。その時はやむを得ずメインメモリに置くことになる。 また、メインメモリは、プログラムの開始から終了までずっと確保することができるので、プログラム上で使われる変数や定数などを置く場所として確保しておく。
使い分けの考え方については、第9回で「記憶の階層」でもういちど細かく学んでほしい。

プログラムカウンタ（PC）とは何か

プログラムカウンタ（PC）は、CPU の制御部の中にあるレジスタの一種で、現在実行している命令のメインメモリ上での場所（アドレス）を保持している。
まず、命令（プログラム）はメインメモリに置かれていることを思い出してほしい（プログラム内蔵）。 プログラムは命令が並んだもの（＋データを含めることもある）である。 メインメモリは、1個1個の命令が入れられる箱がたくさん並んでいて、順番に番号（＝番地、アドレス）が付いている。 つまりアドレス（番地）は、箱の（メインメモリの中での）場所を表している。 それで、命令をメインメモリ上に置くと、それは先頭のアドレスから順番に1つずつ増えたアドレスが付いていることになる。たとえば、先頭が 123 番地から始まるとすると、最初の命令は 123 番地、次の命令は 124 番地、3番目の命令は 125 番地、のように置かれる。
それで、プログラムカウンタ（PC）は、今実行している命令のアドレス （厳密に言えば、今実行している命令が置かれている、メインメモリ上のアドレス） を保持している。
CPU の命令実行サイクルを思い出してほしい。 先頭の「命令の読出し」では、命令を1つだけ、メインメモリから CPU へ読み出してくる。 この時、CPU ではプログラムカウンタの内容をメインメモリにさし示して、「このアドレスの中身を読み出して、私に返してください」とお願いする。メインメモリは CPU にその中身を返す。もらった中身（何か1つの命令のはず）を CPU は「命令の解釈」で解釈し、「命令の実行」で実行する。 ということは、全体を見ると、プログラムカウンタ（PC）でさし示すメインメモリ内の命令を実行していることになる。 ここまでよいだろうか？
さて、命令実行サイクルの最後の「PC←PC+1」について考えよう。 プログラムカウンタは、今実行している命令のメインメモリ内のアドレスを保持しているのだから、その命令が実行し終わったところで1増やすと、メインメモリ上の次のアドレスの

場所に移ることになる。 たとえば、もし前が 123 番地の命令を実行し終わった（この時はまだ PC の値は 123）のであれば、最後の PC←PC+1 で、PC の保持する値は 124 になる。従って、命令実行サイクルが先頭へ戻って「命令の読出し」をすると、124 番地の命令を読み出すことになる。 つまり、順番に並んでいる命令の「次の命令」を読み出して、解釈、実行することになる。 これが「逐次実行」、つまりメインメモリ上の命令を（アドレスの）順番に実行する、ということの原理である。

メインメモリ上で、命令とデータはどうやって区別するのか？

この質問が何件かあったが、答は簡単である。メインメモリ上に置かれた状態では、どちらも 0/1 のパターン（2進数）であり、区別は出来ない。（普通は）特にマークも付いていない。
では、困ることは無いのか？ ある。 データのつもりでメインメモリに書き込んだ部分を、命令として読出して解釈しようとすると、とんでもない（命令としては解釈できない）パターンであることがある。 逆に、命令を置いた部分をデータとして読み出すと、命令のビットパターンが読める（特に実害はないが）。
2年生で学ぶOSの仕組の中で、「メモリの保護」があり、それを使っていれば（Windows や Linux 下では使っている）、実際に間違って読み出す前にエラーが検出されて処理をストップさせられる。

（余談） メモリのアドレス付けと「バイトアドレス」

ここの話は、かなり混乱すると思うので、今のところは忘れてもよいことにする。 ただ、実際にコンピュータを使う上では、この点が結構やっかいなので、この授業の全体が分かった後でもう一度自分なりに理解してみてほしい。
COMET-II では、整数のデータは 16 ビット幅（15 ビット＋符号ビット）であり、汎用レジスタの格納の幅も ALU の計算の幅も 16 ビットになっている。更にはメインメモリ上へのデータの格納も、16 ビットを単位（これを1語、1ワードという）としている。また COMET-II の命令は、短いものが1語（16 ビット）、長いものが2語（32 ビット）、の2パターンであるが、これも語（16 ビット）を単位にしている。
他方、最近のコンピュータでは、パソコンにせよ高性能のワークステーションにせよ、メインメモリのアドレスはバイト（8ビット）を単位にして付けられているものがほとんどである。いくつかの理由があるが、データの長さが8ビット（Java 等での char 型）、16ビット（short 型）、32ビット（int 型や浮動小数の float 型）、64ビット（浮動小数の double 型）などと様々なものがあること、命令の長さも Intel の i3/5/7 の系列（昔の 8086 からずっと続いている系列、通称 x86 系）では1バイトから8バイト（こちらは自信無し、もっと長い命令もあったかもしれない）まで様々なものがあること、が挙げられる。 なお、これらの様々な長さのものは、メインメモリ上に混在できる。

> 脱線であるが、そうなると、メインメモリの内容を見て、何バイト目がデータの始まりか、そのデータの長さが何バイトか、などは判断できない。プログラムがあらかじめ承知している長さで理解する。（324番地から4バイト分が int 型として入っている、などとプログラム・命令が知っていてアクセスする）

メインメモリのアドレスをバイト単位で付けることにする（＝メインメモリの引き出しの大きさを8ビットにする）と、命令の実行が終わって次の命令に進むときに PC←PC+1 とするのが、1を足すだけでは済まなくなる。1つの命令が複数のバイトを占めるからである。もし命令が3バイトの長さであれば、PC は1だけ進めるのではなくて3つ分進める必要がある。もし命令が4バイトの長さなら、4だけ進めなければならない。 こうなると、PC を複数バイト分進めるというだけではなく、その進め方も命令の長さによって、決めなければならない。 命令の長さが命令自体や命令のオペランドの個数、オペランドの形式（レジスタかメモリのアドレス指定かなど）によってさまざまに異なる場合（たとえば Intel の x86 系）、それぞれの命令に応じて PC をいくつ進めるか（PC←PC+〇の〇をいくつにするか）を CPU 制御部の中で判断することになる。