アーキテクチャ第６回予習資料　　第６回の授業までに予習。

第6回　命令の形式・アドレッシング・CISC と RISC・マイクロプログラミング・命令の実行性能

６－１．命令の形式

（用語・概念）

（復習） コンピュータを構成する3つの要素は （　　）と（　　）と（　　）。

（復習） 命令は、英語で （ Instruction ）

（復習） 命令の実行サイクルは、1つの命令を（　　から　　）し、それを（　　）し、（　　）します。最後のステップで（　　）。 このサイクルは、コンピュータの3要素のうちの（　　）で行われます。

命令は、コンピュータの3要素のうちの（　　）に置かれている。それぞれの命令が具体的にどうやって書かれているか（「足せ」と書いてあるのか「add」と書いてあるのか…）というと、（　　）の形で書かれています。

1つの命令は、いくつかの部分に分かれていて、その部分のことを（　　）と呼びます。 通常は、命令の種類（足し算とかロードとか条件分岐とか）を示す （　　）のフィールドと、処理・演算の対象を示す（　　） のフィールドがあります。 この対象を示す（　　）フィールドは1つとは限らず、命令の処理によっては2つ以上のこともあります。 また、そのほかにこまごまとした情報を表すフィールドが付いている場合もあります。

通常は、CPU の種類（命令の体系）によって、それぞれのフィールドの長さは決まっています。 たとえば、COMET－Ⅱの命令は、命令コードが8ビット（2進8桁）、オペランドは演算対象がレジスタの場合4ビット、メモリの場合16ビットです。

指定するオペランドの個数は、CPU の種類によっていくつかのパターンがあります。COMET－Ⅱでは2オペランド方式ですが、他の CPU では1オペランド方式が使われることもあります。 方式としては3オペランド方式も考えられますが、使われていません。 また0オペランド方式というのもあって、仮想的な CPU では使われることもありますが、実行の効率が良くないので実用にはされていないようです。

それぞれのオペランド方式において、演算命令 Z ← X ＋ Y に必要な演算対象 X、Y、Z を指定する方法を説明してください。

3オペランド方式　　3つのオペランドフィールドを使って、それぞれに X、Y、Z を指定する

2オペランド方式　　

1オペランド方式　　

命令の種類（足し算か引き算かロードか条件分岐かなど）は、それぞれの命令中の（　　）のフィールドに書かれますが、（　　）の形で書かれます。 これは人間にとっては読みにくいので、名前で呼ぶことにします。この名前（略した名前、略称）のことを、（　　コード　）と呼びます。

この（　　コード　）は、あくまで人間の為のもので、機械（＝コンピュータのハードウェア）は元の（　　数　）の形で保持します。 そこで、人間が（　　コード　）で書いた命令をコンピュータに掛ける（実行させる）ときには、（　　数　）の形に変換をしてから掛ける必要があります。

COMET－Ⅱの命令のニーモニックコードは、別途配布する機械命令の一覧表を見てください。

（確認問題）

命令はどのような形式で表されているのか、説明してください。

命令の主なフィールドの名称とその役割を説明してください。

2オペランド方式の場合、足し算・引き算のように入力が2つ、出力が1つある演算を命令にしようとすると、合計3つの演算対象を2つのフィールドで指定しなければなりません。それはどのような方法で指定するのか、説明してください。

（追加問題）「次の命令へ移る」のと、「ノイマン型」のうちの「逐次実行」との関係を、説明してください。

（追加問題）

それぞれのフィールドの長さ（何ビット使うか）がどうして決まるのか、説明してみてください。

(a) オペランドとして、メモリを指定するときを考えます。 メモリの中から1つのセル（COMET-Ⅱではワード）を指定するには、アドレス（番地）で指定します。 では、メモリが 65536（＝2^{16}）ワードの大きさの時、アドレスは何ビットあれば指定できるでしょうか？ 別の言い方をすれば、1番目（＝0番地）から2^{16}番目（＝2^{16}－1番地＝65535 番地）を区別するためには、2進数で何桁分を用意すればよいでしょうか？

(b) 同様に、オペランドとしてレジスタを指定するときは、どうでしょうか。 但し COMET-Ⅱでは GR0～GR7 の8個です。

(c) 命令コードはどうでしょうか？ COMET-Ⅱでは、命令の種類が28あります。

6－2. アドレッシング

（用語・概念）

オペランド（＝命令の操作対象）のうち、メモリにあるデータを指定するのには、メモリの（　　）を指定しますが、そのときに「修飾」（＝ちょっとした細工＝追加の計算）ができるようになっています。

ビデオで紹介した6種類のやり方が、代表的なものです。 但しどの種類が使えるかは、CPU によって異なります。

.....アドレッシング	命令のオペランドフィールドに与えられた値を、そのままアドレスとしてメモリにアクセスする
.....アドレッシング	命令に与えられたアドレスの場所を読んで、そこに書かれている値を（もう一度）アドレスだと思ってメモリをアクセスする
.....アドレッシング	命令に与えられたアドレスの値と、指標レジスタの中の値とを足して、それをアドレスとしてメモリをアクセスする
.....アドレッシング	命令に与えられた値と、命令が置かれているメモリ上の場所（アドレス）とを足して、それをアドレスとしてメモリをアクセスする
.....アドレッシング	命令に与えられたアドレスの値と、基底レジスタの中の値とを足して、それをアドレスとしてメモリをアクセスする
.....アドレッシング	命令に与えられたアドレスの値を、そのまま値として演算に使う （＝メモリをアクセスするのではなくそのまま値として使う。他の方式は値をアドレスと思ってメモリをアクセスし、その中身を返す）

上記のようないろいろなアドレッシングの方法のことを、「モード」と呼ぶことがあります。 また、指標＝「インデックス」、基底＝「ベース」と書くことがあります。

また、修飾によって計算された結果得られたアドレスのことを、日本語では（　　）アドレス、英語では effective address と言います。

それぞれの命令の中に、アドレッシングのモードを指定する仕組みがあります。 たとえば COMET-Ⅱでは、指標アドレッシングモードを示すのに、指標レジスタを指定するフィールドの値が0であれば指標を使わない（＝直接モード）し、1～7であればその1～7の値が指標レジスタの番号（GR1～GR7 を使う）を指定します。

直接アドレッシング:　LD　GR3,　2345

これは、メモリ上の（ ..... ）番地と汎用レジスタの（ ..... ）を対象としたロード命令で、特に指定が無いので（ ..... ）アドレッシングです。 具体的には、（ ..... ）から（ ..... ）へデータをコピーします。

間接アドレッシング:　LD　GR3,（間接）2345　　（COMET-Ⅱにはこのモードはない）

COMET-Ⅱには間接アドレッシングモードが無いので、命令に書けないのですが、仮にここでは（間接）と書きました。

この命令の動作は、（間接をうまく表現してください: ..... ）から（ ..... ）へデータをコピーします。

指標アドレッシング:　LD　GR3,　2345

COMET-Ⅱでは、指標アドレシングを書くときに、上のように最後に「 , GR5 」のように指標レジスタを指定します。これが指定されていないとき（つまりアドレスの値 2345 までで終わっているとき）は、直接アドレシングになります。

この例の場合は、命令上で指定した（ ..... ）番地と、汎用レジスタ GR5 を指標レジスタに指定しているので GR5 の（ ..... ）とを（　　）した値を、メモリに対するアドレスとして使います。 もし GR5 が 3 ならば、ロード命令は、（ ..... ）から（ ..... ）へデータをコピーします。

相対アドレッシング:　JUMP　（相対） 5　　（COMET-Ⅱにはこのモードはない）

相対モードも、COMET-Ⅱには無く、命令に書けないので、仮に「（相対）」と書いておきます。

この命令が置かれているメモリ上のアドレスが （ ..... ）番地だとします。 命令のオペランドのフィールドに 5 と

書いてあれば、修飾の結果得られる実効アドレスは、(.....) 番地になります。 相対アドレシングモードの場合は、オペランドフィールドに負の数を書くことがあります。

実際によく使うのは、ジャンプの飛び先の指定をする時です。 例のように、オペランドフィールドに (.....)と書いてあれば、(.....)番地分だけ先の場所にジャンプする、ということになります。 負の数が指定されていれば、指定してある数だけ(.....)ということになるので、「ループの先頭へ戻る」という時などにうまく使えます。

即値アドレッシング: LAD GR3, 123

即値アドレシングは、命令の(.....)に書かれている値を、そのまま値として使う、という命令です。

COMET-Ⅱでは、即値モードは用意されていませんが、ロード命令の即値モード版である LAD (ロード・アドレス)命令があります。 動作は、命令の(.....)に書かれている値(.....)を、直接にそのまま値として、(.....)の GR3 へ書き込みます。

普通の(=直接アドレシングモードの)ロード命令 LD GR3, 123 と動作を比較してみてください。

LD GR3, 123 の結果 GR3 に入る値は (.....)

LAD GR3, 123 の結果 GR3 に入る値は (.....)

(確認問題)

右図はビデオにあった例です。それぞれのアドレシングモードの時にロード命令を実行した場合の、具体的な動作と結果を列挙してください。

6－3. CISC と RISC

(用語・概念)

この節で扱うのは、1つ1つの命令ではなくて、1つ(1種類)の CPU がどんな命令を持っているか(=これを「命令体系」と呼ぶ)、さらにその命令体系が CPU の機種によってどう違うか、違いがどういう利害得失を生むのか、という議論です。

スライドに出てきた2つの用語の意味を説明してください。

略称	RISC	CISC
フルの名称		
それはどういうものか？		
比較 (命令長) (命令の複雑さ)		

(HW の複雑さ) (クロック速度) (プログラム長さ) (全体でどちらが優れているか)		

スライドでも触れているように、RISC の概念が出てきたのは歴史的な背景があり、それが現在ではほとんど失せているので、「どちらが優れている」という議論はあまり意味がないと言われています。

＜歴史的経緯＞　メインフレームの時代に、命令にどんどん複雑な命令が追加された。1 命令で複雑な処理をして、性能を向上させたかったのだと思う。 メインフレームの CPU は 1 チップ化はされておらず、いくつかの（かなり多くの）チップでできていて、容量制限はあまりなかった。 ある時（1978 年ぐらい）に実用的な CPU の 1 チップ化が行われたが、その前後では 1 チップに搭載できる論理回路の分量の上限が小さいのであまり複雑な命令を取り込めず、かつクロック速度があまり早くできなかったので複雑な命令を取り込めなかった。そのために考えられたのが RISC である。1980 年代初頭に RISC の考え方（CPU の設計指針）が議論・評価され、マイクロプロセッサの技術として普及した。 その後 1990 年代初頭には、これらの制約は半導体技術の発展により回路容量、スピードとも問題にならなくなり、最近はこれらを理由にした RISC の議論は聞かない。現に Intel の 86 系プロセッサ（CISC）がパソコンの主流を占めている（主たる理由はおそらく命令セットの継続によるソフト継続性）。 他方、スマホ・携帯端末で使われる ARM は RISC 的な設計になっている。

6－4. マイクロプログラミング

（用語・概念）

マイクロプログラミングは、「命令の実行サイクル（4つのステップ）」を実現するのに、（マイクロ）プログラムの手続きとして実現します。 ～～ そうでない（マイクロプログラムでない）やり方では、4つのステップを順番に進行することとそれぞれのステップの中身を、ハードウェアの配線で実現（布線論理方式と呼ぶ）します。 4ステップの実現が、ハードウェア回路によるのか、プログラムのような書き方ができるのか、の違いです。 両者を比較すると:

（マイクロプログラミングに比べて）布線論理が優れている点: .....

.....

.....

（布線論理に比べて）マイクロプログラミングが優れている点: .....

.....

.....

メインフレームの時代に、同じ命令セットを持つコンピュータを布線論理とマイクロプログラムで作っていた時代があります（今は行われませんが）。 この時、上位モデル（速度が速い、その代わり値段が高い）では布線論理を、下位モデル（速度は遅い、値段は安い）ではマイクロプログラムを使っていました。 同じ命令が使えるので、プログラムは（OSもアプリも含めて）そのまま（共通に）使える、ということが「売り」で、お客さんの懐事情に応じてハードを選択してください、という売り方をしました。 現在のマイクロプロセッサでは、ハードの（布線論理的な）複雑さはコストにあまり影響しないの

で、このような差別化はされていません。 むしろ、マルチメディアなどを 1 命令で扱うために複雑な処理をする命令を作り込むときに、マイクロプログラミングに類する手法が使われているようです。

6－5. 命令の実行性能

（用語・概念）

クロックとはどういうものか、説明してください：

クロックを速くすると何が起こるでしょうか　たとえばクロックが元々3GHz なのに 4GHz にしたらどうなるでしょうか：

CPI (Cycle Per Instruction または Clock Per Instruction)とは何か、説明してください：

1つの命令を実行するのにかかる時間を、CPI とクロック速度で表してください：

1秒間に実行できる命令の数はいくらですか：

MIPS とは何ですか：

同じ CPU でも命令によってかかるクロック数が違うことがあります。 このときはどうしたらいいでしょうか？

CPI は CPU の仕組・構造とどう関係していると思いますか。 CPI を小さくする（＝CPU の命令実行速度が速くなる）にはどうすればよいでしょうか？

（練習問題）

a） 50MIPS のプロセッサの平均命令実行時間はいくらか？

b）平均命令実行時間が 0.2 マイクロ秒のコンピュータがある。このコンピュータの性能は何 MPIS か？

c） ある CPU の CPI が 3 であり、その CPU に 3GHz のクロックを与えたとき、1つの命令の実行にかかる時間はいくらか

d） 命令ごとの出現頻度と、これを実行するプロセッサの実行クロック数を右表に示す。このテストプログラムにおける CPI はいくらか？

命令	出現頻度	実行クロック数
転送	50%	1
演算	30%	2
分岐	20%	5

（おまけ） コンピュータシステムの性能指標

コンピュータシステムの（ハードウェア）性能指標として、MIPS の値が使われたことがありますが、いくつかの理由で「良くない」と言われ、他の指標にとって変わられました。 理由を考えてみてください。

（理由 1） 命令の出現頻度は、プログラムによって違うのではないか？

（どんなことが考えられるか？）

（理由 2） 性能（アプリの実行時間とか）を決めているのは、CPU （の命令実行時間） だけではないだろう？

（何が影響するだろうか？）

現在では、「ベンチマークテスト」が良く使われます。 整数計算、浮動小数計算、トランザクションなどいろいろなベンチマークテストがありますが、どれもすべての面の性能を代表するものではないことに注意する必要があります。