マイクロプロセッサ演習

2004 年

第 2 回

0　はじめに

今回は「コンピュータの性能評価」という内容を取り扱うが、その内容に関連する Web ページを紹介する。「CPU の性能をクロック周波数だけで評価するのは妥当か」という内容である。
『AMD、Athlon1.5GHz でメガヘルツ神話打倒キャンペーン (2001/8 ITMedia)』
http://www.itmedia.co.jp/news/0108/09/e_amd.html
『Intel も脱クロック偏重 (2004/3 ITMedia)』
http://www.itmedia.co.jp/news/articles/0403/15/news003.html

さて、今回の演習を行うための予備知識を以下で説明しておこう。前回説明したように、C 言語で書かれたプログラムをコンパイルするとアセンブリ言語で書かれたプログラムが出力される。その様子を図 1 に示した。このとき、アセンブリ言語で書かれ

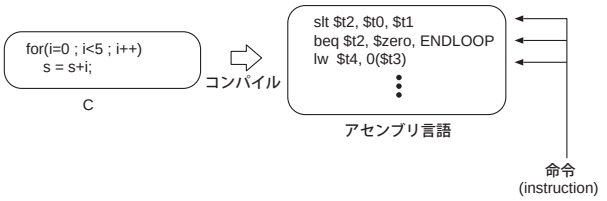

図 1: C 言語からアセンブリ言語へのコンパイル

たプログラムは図にあるように**命令**の列として表現される。

この命令が CPU によって一命令づつ実行されることになる [1]。C 言語による表現に比べ、アセンブリ言語でのプログラムは非常に冗長になっていることに注意しよう。これは CPU の振舞いを逐一記述しているためである。

ある CPU で定義されている命令を集めたものはその CPU の**命令セット**と呼ばれる。CPU によって命令セットは異なるため、アセンブリ言語の文法も各 CPU で異なる。本演習で扱うのは MIPS R2000 と呼ばれる CPU のアセンブリ言語である。MIPS は RISC (Reduced Instruction Set Computer) 型の CPU の一つであり、その構造もシンプルであるため学習には適している。

また、CPU には「クロック」と呼ばれるものがあり、クロックに同期して命令が実行される。このとき、図 2 のように一命令の実行ごとに複数のクロックがかかるのが普通である [2]。

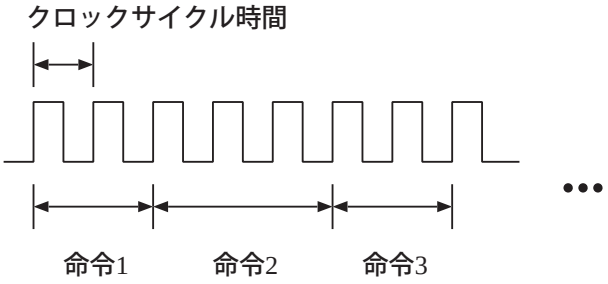

図 2: クロックに同期した命令の実行

また、一命令を実行するのに必要なクロック数の平均値を **CPI** (clock cycle per instruction) と呼ぶ。

最後に、以下の問題を解くためのヒントを図 3 に載せる。

1　[問題] 第二回の演習問題

[問題 1] 相対性能
M1 と M2 の 二つの別のマシンの性能を評価したい [3]。両マシンに関して下記の測定結果が得られている。各プログラムについてどちらのマシンがど

プログラム	M1 上での時間	M2 上での時間
1	10 (秒)	5 (秒)
2	3 (秒)	4 (秒)

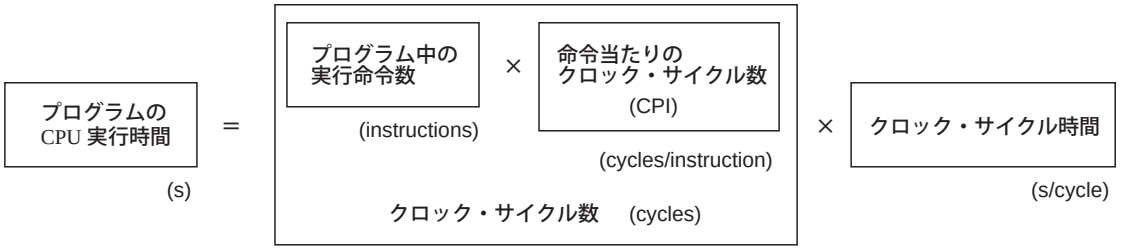

図 3: 性能評価に関する公式 (ヒント)

れだけ速いか。

[**問題 2**] **命令 (instruction)**

問題 1 の両マシンに関して追加の測定を行い、下記のデータが得られた [4]。プログラム 1 を実行した

プログラム	M1 上で実行された命令数	M2 上で実行された命令数
1	200×10^6	160×10^6

ときの各マシンの実行命令率 (1 秒あたりの命令実行回数) を求めよ。

[**問題 3**] **CPI (clock cycle per instruction)**

問題 1 のマシン M1 のクロック周波数は 200 MHz、マシン M2 のクロック周波数は 300 MHz であるとする。問題 1 と問題 2 のデータを使用して、両方のマシンにおけるプログラム 1 の命令当たりのクロック・サイクル数 (CPI) を求めよ。

また、クロック周波数では M2 は M1 の 1.5 倍の性能であるが、プログラム 1 の実行時間で判断すると M2 は M1 の 2 倍の性能である。この違いは何に依るか。

2 [SPIM] SPIM を用いたシミュレーション

今回も前回に引続き SPIM シミュレータを使う練習を行う。前回は文字列 (“Hello World.”) を表示させるプログラムだったが、今回は 2 つの整数値の足し算を行うプログラムである。

まず、本演習の Web ページより data02.zip というファイルを入手する。このファイルを解凍すると “Add.asm” というソースファイルが出て来るので、このファイルを前回 spim をインストールしたフォルダに移動する。(前回の指示に従っていれば w:\Install PCspim である。)

まず Add.asm をテキストエディタ (秀丸) で開き、中身を確認しておく。コメント (# の後に書かれていること) を見れば、大体内容を把握できるはずである。

さて、このプログラムを DOS 版 spim (SPIM.EXE) および Windows 版 spim (pcspim.exe) の両方で実行してもらう。簡単な手順を以下に示す。

DOS 版 (SPIM.EXE) での実行

1. SPIM.EXE をダブルクリックする。
2. (spim) load ”Add.asm” により、プログラムをロード
3. (spim) run により、プログラムを実行 (ここでは何も起きない)
4. (spim) print $t0 により、結果を確認。(print $t1、print $t2 も試してみよう)
5. (spim) exit で DOS 版 spim を終了できるが、ここでは終了せずそのままにしておく。

Windows 版 (pcspim.exe) での実行

1. File → Open で Add.asm をロード。
2. Simulator → Go で実行。(Run Parameters はそのまま OK)
3. Register Display (一番上の窓) の t0, t1, t2 を参照し、DOS 版と同じ結果になっていることを確認する。(Windows 版では 16 進数で結果が表示されることに注意)

なお、レジスタ (register) とは CPU 内部にあるデータを保持する機構である。このシミュレーションではレジスタ内のデータが書き換えられ、計算が

実行されていることが確認できた。(なお、spim は MIPS R2000 という CPU の振舞いを模倣するシミュレータであることに注意しよう)

次回以降はこのレジスタの振舞いを理解しながら、プログラムを自分で記述してもらう予定である。

参考文献

[1] なおアセンブリ言語には疑似命令と呼ばれるものがあり、この中にはアセンブラによって機械語に変換されるときに複数の命令に展開されるものがあるが、本文中ではそれらは考慮していない。

[2] 一命令の実行にかかるクロック数は命令ごとに異なるが、その目安は教科書の 5.4 章で解説される。また、教科書 (下) の 6 章「パイプラインを用いた性能向上」では、全ての命令を実質 1 クロックで処理する手法について解説されている。

[3] マシン M1 と マシン M2 は、例えば「Intel Pentium 4 を搭載した Windows マシン と PowerPC を搭載した Macintosh」や「AMD Athlon を搭載した Windows マシンと Ultra Sparc を搭載したワークステーション」などを想像すればよい。つまり「アーキテクチャの異なる CPU を搭載したマシン」を 2 つ考えるということである。

あるいは「Intel Pentium 4 を搭載した Windows マシンと AMD Athlon を搭載した Windows マシン」を考えても良い。こちらの場合は「命令セットには互換性があるが実装が異なる CPU を搭載したマシン」を 2 つ考えることになる。

[4] 「Pentium と PowerPC」のように命令セットが異なる CPU であれば、この問題のように同じプログラムでも命令数が異なるのは自然である。

一方、「Pentium と Athlon」のように命令セットに互換性がある場合は通常命令数は一致するはずである。この場合、この問題 2 は「M1 (Pentium) と M2 (Athlon) とでコンパイル時の最適化レベルを変えたので命令数が変化した」と解釈すれば良いだろう。