「今度こそわかるＰ≠ＮＰ」（渡辺治）の「こだわり補足」

講談社サイエンティフィク, 2014

(2014.9.18)

原始プログラムのコード化の方法とビット長の評価

※この説明は，東京工業大学で大学院 授業課題の回答レポートを元に作成した．元となったレポートを作成し加藤 大智氏（2013 年度計算工学修士）に感謝する．

１．原始プログラムのコード化の方法

各命令は命令番号 a と目的語となる数 b の対で表され, 任意の原始プログラムはその命令の並びである．そこで，原始プログラム P を次のような形でコード化する．

各命令を（a, r, b）の形で表す．a は用いる命令（番号付をして数で表わす）．r は目的語 b が何を表しているかを示す数値，そして b は目的語である．r の意味は

r の値	意味
0	命令が目的語をとらない
1	b は読むべきレジスタの番号
2	b は読むべきメモリの番地
3	b は読むべきメモリの番地が保存されたレジスタの番号

とする．

以上の形で表現された命令（a_1, r_1, b_1),…, (a_k, r_k, b_k）の a, r, b をビット値化し，$a_1r_1b_1$ ・・・$a_kr_kb_k$ のように直列に接続すると，原始プログラムを 2 進列でコード化できる．a, r ,b のビット長をあらかじめ定数として与えれば，命令の切れ目を判定するのは容易である．

たとえば，本書の図 2.2 の @main は次のようにコーディングされる．ここで，b のビット長は 4 とした. またコーディング結果を読みやすくするため空白を挿入したが実際には不要である．

プログラム	(a, r, b)	コーディング結果
get#1 [1]	(0, 1, 1)	000000 01 0001
mult2#1	(16, 0, 0)	010000 00 0000
add#1 [2]	(8, 1, 2)	001000 01 0010
get#2 [3]	(1, 1, 3)	000001 01 0011
mult2#2	(17, 0, 0)	010001 00 0000
add#2 [4]	(9, 1, 4)	001001 01 0100
get#3 0	(2, 0, 0)	000010 00 0000
@mult	（ここに@mult をコーディングしたものを挿入）	
halt #3	(63, 0, 0)	111111 00 0000

２．プログラムをコード化した際のビット長

次にビット長を評価する．a の値は 64 以下なので 6 ビットで表わすことができる．同様に，r の値は 4 以下なので 2 ビットで表わす．b は不定だがプログラムの命令数 k のビット長（goto 命令で使用）か $l(\ell)$ の大きい方で固定しておく．したがって，プログラム P のコード長 $|P|$ は，

$$|P| = k\big(6 + 2 + \max\big(|k|, l(\ell)\big)\big) < k\big(8 + |k| + l(\ell)\big)$$

と評価できる．