Multiplication by an Integer Constant: Lower Bounds on the Code Length

Vincent Lefèvre

Loria, INRIA Lorraine

RNC'5

September 3 – 5, 2003

Introduction

Problem: to generate (optimal) code with elementary operations (left shifts, i.e. multiplications by powers of 2, additions and subtractions).

Example: compute $1997x$ (constant $n = 1997$).

$$\begin{array}{lrcl}
1. & 17x & \leftarrow & (x \ll 4) + x \\
2. & 51x & \leftarrow & (17x \ll 2) - 17x \\
3. & 1997x & \leftarrow & (x \ll 11) - 51x
\end{array}$$

Can we get a very short code that computes nx?

Same question as with compression methods! (i.e. compress n.)

Other similarities: my heuristic, based on common patterns in the base-2 representation of n.

Formulation of the Problem

Given: odd positive integer n (our constant). We consider a sequence of positive integers $u_0, u_1, u_2, \ldots, u_q$ such that:

- initial value: $u_0 = 1$;
- for all $i > 0$, $u_i = |s_i\, u_j + 2^{\,c_i}\, u_k|$, with

$$j < i, \quad k < i, \quad s_i \in \{-1, 0, 1\}, \quad c_i \geq 0;$$

- final value: $u_q = n$.

Same operations with $u_0 = x$: we get code (called *program* in the following) that computes the $u_i\, x$, and in particular, nx.

Minimal q associated with n (denoted q_n)?

Outline:

1. Introduction / formulation of the problem (done).
2. Bounds on the shift counts.
3. A prefix code for the nonnegative integers.
4. How programs are encoded.
5. Lower bounds on the program length.

Bounds on the Shift Counts

Two data contribute to the *size* σ of a program:

- the number q of elementary operations (i.e. the *length*);
- the size of the parameters, in particular the shift counts c_i.

Information theory will give us information on σ. To deduce lower bounds on q, we need bounds on c_i.

Notation: for any positive integer m, let $\mathcal{P}_m$ be a subset of programs multiplying by m-bit constants; S denotes a function such that for any program $\in \mathcal{P}_m$ and any i, $c_i \leq S(m)$.

$\mathcal{P}_m$: optimal programs, programs generated by some algorithm, etc.

[$S(m)$: bound on the shift counts for any considered program (i.e. in $\mathcal{P}_m$) associated with m-bit constants.]

For $n = 2^m - 1$, the optimal program will always be in $\mathcal{P}_m$.
Therefore, $\boxed{\boldsymbol{S(m) \geq m}}$.

For the set of programs generated by algorithms used in practice, $c_i \leq m$, therefore $S(m) = m$.

Proved upper bound for optimal programs:
$S(m) \leq 2^{\lfloor m/2 \rfloor - 2}(m+1)$, but useless here.

For adequately chosen optimal programs, it seems that $c_i \leq m$.
If this is true, then $S(m) = m$.
$\rightarrow$ Lower bound on the length of *any* program.

But for the set of **all** optimal programs, consider the following example for $m = 6h + 1$: $n = (1 + 2^h)(1 + 2^{2h})(1 + 2^{4h}) - 2^{7h}$.

One of the optimal programs (4 operations):

$$
\begin{array}{lcllcl}
u_0 & = & 1 & & & \\
u_1 & = & u_0 \ll h & + & u_0 \\
u_2 & = & u_1 \ll 2h & + & u_1 \\
u_3 & = & u_2 \ll 4h & + & u_2 \\
u_4 & = & u_3 & - & u_0 \ll 7h\,.
\end{array}
$$

This gives: $S(m) \geq 7h = \frac{7}{6}(m-1)$.

$\rightarrow$ The choice of the optimal program for a constant n is important.

We will also consider $S(m) = k.m$, with $k > 1$.

A Prefix Code for the Nonnegative Integers

Linked to the *unbounded search problem*: there exists a code in $\operatorname{logsum}_2(n) + O(\log^*(n))$.

Here, we are only interested in a code in $\log_2(n) + o(\log_2(n))$.

For $n \geq 4$:

- k: number of bits of n minus 1;
- h: number of bits of k minus 1;
- code word of n: 3 concatenated subwords h digits 1 and a 0 h bits of k without the first 1 k bits of n without the first 1.

integer	code word
0	000
1	001
2	010
3	011
4	10 0 00
5	10 0 01
6	10 0 10
7	10 0 11
8	10 1 000
15	10 1 111

integer	code word
16	110 00 0000
31	110 00 1111
32	110 01 00000
63	110 01 11111
64	110 10 000000
127	110 10 111111
128	110 11 0000000
255	110 11 1111111
256	1110 000 00000000
511	1110 000 11111111

Encoding an Elementary Operation

Elementary operation: $u_i = |s_i\, u_j + 2^{\,c_i}\, u_k|$.
$\rightarrow$ Encode s_i, c_i, j and k.

- s_i: 3 possible values (-1, 0 and 1) $\rightarrow$ 2 bits.
  4th one for the end of the program.
- Integers c_i, j and k: prefix code.
- Concatenate the 4 code words.

Size of the Encoded Program

Bounds on the integers:

- c_i bounded above by $S(m) = k.m$.
- j and k bounded by $i-1$, and without significant loss, by $q-1$.

$\rightarrow$ Upper bound on the size of the encoded program:

$$B(m,q) = q\,(2 + C(S(m)) + 2\,C(q-1)) + 2.$$

with $C(n) = \begin{cases} 3 & \text{if } n \leq 3, \\ \lfloor \log_2(n) \rfloor + 2\,\lfloor \log_2(\log_2(n)) \rfloor + 1 & \text{if } n \geq 4. \end{cases}$

Asymptotically: $B(m,q) \sim q\,(\log_2(S(m)) + 2\,\log_2(q))$.

With $S(m) = k.m$: $B(m,q) \sim q\,(\log_2(m) + 2\,\log_2(q))$.

Lower Bounds: A Notation...

Let f and g be two positive functions on some domain.

$f(x) \gtrsim g(x)$ if there exists a function ε such that

$$|\varepsilon(x)| = o(1) \quad \text{and} \quad f(x) \geq g(x)\,(1 + \varepsilon(x)).$$

Note: it is equivalent to say that there exists a function ε' such that

$$|\varepsilon'(x)| = o(1) \quad \text{and} \quad f(x)\,(1 + \varepsilon'(x)) \geq g(x).$$

Lower Bounds: Worst Case

We consider the 2^{m-2} positive odd integers having exactly m bits in their binary representation, and for each integer, an associated program in $\mathcal{P}_m$. The 2^{m-2} programs must be different.
$\Rightarrow$ There exists a program whose size σ is $\geq m-2$, and its length q satisfies: $m-2 \leq \sigma \leq B(m,q) \leq B(m, q_{\mathrm{worst}})$.

We recall that asymptotically, with $S(m) = k.m$, we have:

$$B(m, q_{\mathrm{worst}}) \sim q_{\mathrm{worst}} \left(\log_2(m) + 2 \log_2(q_{\mathrm{worst}})\right).$$

We can guess that $\log_2(q_{\mathrm{worst}}) \sim \log_2(m)$. Thus we choose to bound q_{worst} by m and write: $q_{\mathrm{worst}} \left(3 \log_2(m)\right) \gtrsim B(m, q_{\mathrm{worst}})$.

We recall that $q_{\text{worst}}\,(3\,\log_2(m)) \gtrsim B(m, q_{\text{worst}}) \geq m-2$.

As a consequence: $\boxed{q_{\text{worst}} \gtrsim \dfrac{m}{3\,\log_2(m)}}$.

Note: this also proves that $\log_2(q_{\text{worst}}) \sim \log_2(m)$, thus we didn't lost anything significant when bounding q_{worst} by m.

Exact lower bound for $m \geq 4$:

$$\frac{m-4}{3\,\log_2(m) + 4\,\lfloor \log_2(\log_2(m)) \rfloor + 2\,\lfloor \log_2(\log_2(k.m)) \rfloor + \log_2(k) + 6}$$

(note: very optimistic for small m — e.g., < 1 for all $m \leq 37$).

Lower Bounds: Average Case

We consider the set O_m of the 2^{m-2} positive odd integers having exactly m bits in their binary representation, and for each integer, an associated program in $\mathcal{P}_m$.

The 2^{m-2} programs must be different:

$$\frac{1}{2^{m-2}} \sum_{i \in O_m} B(m, q_i) \geq \frac{1}{2^{m-2}} \sum_{i=1}^{2^{m-2}} \lfloor \log_2 i \rfloor = m - 4 + \frac{m}{2^{m-2}},$$

As a consequence,

$$2 + (2 + C(S(m)) + 2\,C(m)) \frac{1}{2^{m-2}} \sum_{i \in O_m} q_i \geq m - 4 + \frac{m}{2^{m-2}}.$$

We recall that

$$2 + (2 + C(S(m)) + 2\,C(m))\,\frac{1}{2^{m-2}} \sum_{i \in O_m} q_i \geq m - 4 + \frac{m}{2^{m-2}}.$$

Thus $q_{\mathrm{av}} \geq \dfrac{m - 6 + m/2^{m-2}}{2 + C(S(m)) + 2\,C(m)}$.

Asymptotically, with $S(m) = k.m$, the average length q_{av} satisfies:

$$\boxed{q_{\mathrm{av}} \gtrsim \frac{m}{3\ \log_2(m)}},$$

i.e. the same bound as in the worst case.

m	q_{av}^{+}	q_{av}^{-}	ratio
8	2.6	0.11	24.5
16	4.4	0.34	12.8
32	7.6	0.81	9.35
64	13.4	1.66	8.09
128	23.7	3.21	7.38
256	42.2	5.32	7.93
512	75.5	10.1	7.46
1024	135	19.2	7.05
2048	243	36.5	6.67
4096	440	69.3	6.35
8192	803	132	6.08

For random m-bit constants: approximated upper bounds on q_{av} (obtained with my algorithm), lower bounds on q_{av} and the ratio.