

次の問 8 は必須問題です。必ず解答してください。

問 8 次のプログラムの説明及びプログラムを読んで、設問 1, 2 に答えよ。

プログラム 1 及びプログラム 2 は、いずれも二つの整数 M , N を受け取り、その積 $M \times N$ の値を返す。積は、加減算とシフト演算を使って求める。 M , N 及び求めた積は、いずれも符号付き 2 進数の整数で、負数は 2 の補数で表現する。

〔プログラム 1 の説明〕

プログラム 1 が受け取る M , N は、いずれも 4 ビットで、各値の範囲は $-8 \sim 7$ である。求めた積 $M \times N$ は、8 ビットで返す。ただし、プログラム 1 中では、 M を 5 ビットに、 N を 8 ビットに拡張して処理を行う。

R は 13 ビットの作業用変数であり、最下位から順にビット番号を 0, 1, \dots とし、最上位（符号ビット）のビット番号を 12 とする。 R は、指定した一部の範囲（例えば、ビット番号 12 \sim 8 の上位 5 ビット）だけを符号付き 2 進数とみなして部分的な算術演算ができる。また、値の検査のために指定したビット番号の内容を取り出すこともできる。

なお、⑥の行の加算では、けたあふれが起きても無視する。

$N=3$ として、 $M=5$ と $M=-5$ の場合の処理過程を、それぞれ図 1, 2 に示す。図 1, 2 中の記号① \sim ⑧は、プログラム 1 の① \sim ⑧の行の処理と対応している。

〔プログラム 1〕

○プログラム 1 (M , N)

○符号付き 2 進整数型: M , N , R

○整数型: L

- ① → ・ M を 5 ビットの符号付き 2 進整数に拡張
- ② → ・ N を 8 ビットの符号付き 2 進整数に拡張
- ③ → ・ R のビット番号 7 \sim 0 に N を複写
- ④ → ・ R のビット番号 12 \sim 8 を 0 で初期化
- ⑤ →

■

■

 $L: 1, L \leq 8, 1$
- ⑥ →

■

■

↑
↓

 R のビット番号 0 のビットが 1
・ R のビット番号 12 \sim 8 の内容に M の値を加算
- ⑦ → ・ R の全 13 ビットを右に 1 ビット算術シフト /* 空いたビット位置には */
/* 符号と同じものが入る */
- ⑧ → ・ return (R のビット番号 7 \sim 0 の内容) /* 返却値 (括弧内) を返す */

Figure 1 illustrates the generation of a 128-bit stream S from a 12-bit key K and a 128-bit message M using a linear feedback shift register (LFSR). The key K is 1211109876543210. The message M is 00101. The stream S is generated by repeatedly applying the LFSR to the key and message, resulting in a sequence of 128-bit blocks. The diagram shows the initial state of the LFSR, the key and message inputs, and the resulting stream S blocks, with arrows indicating the flow of data and the generation of the stream.

図2 プログラム1の実行例
($M = -5$, $N = 3$)

〔プログラム 2 の説明〕

プログラム 2 が受け取る M, N は、いずれも 4 ビットで、各値の範囲は $-8 \sim 7$ である。求めた積 $M \times N$ は、8 ビットで返す。ただし、プログラム 2 中では、M を 5 ビットに拡張して処理を行う。

R は 10 ビットの作業用変数であり、最下位から順にビット番号を 0, 1, … とし、最上位（符号ビット）のビット番号を 9 とする。R は、指定した一部の範囲（例えば、ビット番号 9～5 の上位 5 ビット）だけを符号付き 2 進数とみなして部分的な算術演算ができる。また、値の検査のために指定したビット番号の内容を取り出すこともできる。

なお、⑤の行の加算及び⑦の行の減算では、けたあふれが起きても無視する。

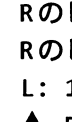
M=-5, N=3 の場合の処理過程を図3に示す。図3中の記号①～⑨は、プログラム2の①～⑨の行の処理と対応している。

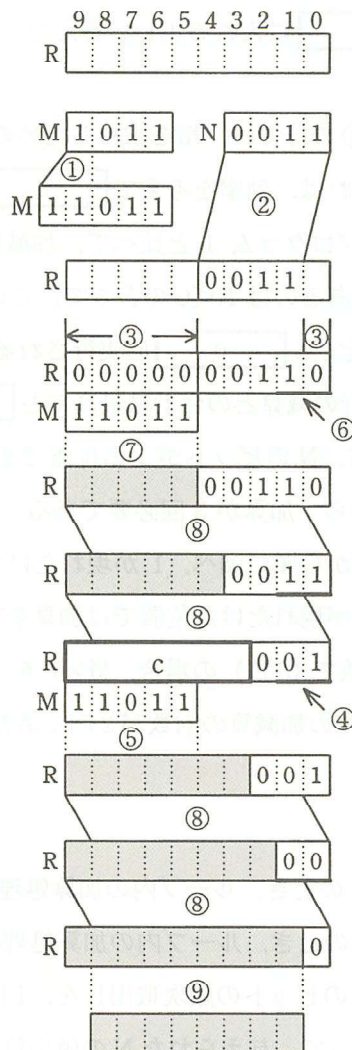
〔プログラム 2〕

○プログラム2 (M, N)

○符号付き2進整数型：M, N, R

○整数型： L

- ① → ・ Mを5ビットの符号付き2進整数に拡張
 - ② → ・ Rのビット番号4～1にNを複写
 - ③ → ・ Rのビット番号9～5及び0を0で初期化
 - ④ →  L: 1, L ≤ 4, 1
 - ⑤ → ・ Rのビット番号9～5の内容にMの値を加算
 - ⑥ → ・ Rのビット番号1のビットが1 かつ Rのビット番号0のビットが0
 - ⑦ → ・ Rのビット番号9～5の内容からMの値を減算
 - ⑧ → ・ Rの全10ビットを右に1ビット算術シフト /* 空いたビット位置には /* 符号と同じものが入る /*
 - ⑨ → ・ return (Rのビット番号8～1の内容) /* 返却値(括弧内)を返す /*



注 網掛けの部分は、表示していない。

図3 プログラム2の実行例
(M=5, N=3)

設問1 図2及び図3中の に入れる正しい答えを、解答群の中から選べ。

aに関する解答群

ア 0 0 0 1 0 1 イ 0 1 1 0 1 1 ウ 1 0 0 1 0 1 エ 1 1 1 0 1 1

b, cに関する解答群

ア 0 0 0 0 1 0 0 イ 0 0 0 0 1 0 1 ウ 0 0 0 1 1 1 1
エ 0 0 1 0 0 0 1 オ 1 1 1 0 0 0 1 カ 1 1 1 1 0 1 1

設問2 次の記述中の に入れる正しい答えを、解答群の中から選べ。

- (1) プログラム1の③の行では、積を求めるためのRの下位8ビットにNの値を設定している。それは、効率を考慮して d ためである。
- (2) プログラム2はプログラム1と比べて、加減算の回数が少なく済む場合がある。特に効果があるのは $N < 0$ のときで、このとき、プログラム1では、⑥の行の加算は最低でも e 回実行されるが、プログラム2では、⑤の行の加算と⑦の行の減算との合計は最高でも f 回で済む。
- (3) プログラム1では、Nのビットが1の位置で加算をするので、例えば $N=7$ (2進数で0111) なら、加算が3回必要である。一方、プログラム2では、Nの各ビットを最下位から順に調べ、1が現れたけた位置では減算をし、次に1の並びが途切れて0が現れたけた位置では加算をする、という処理を繰り返す。例えば $N=7$ (2進数で0111) の場合、 $M \times 7$ を $M \times$ (g) として計算するので、2進数での加減算の回数が2回で済む。

dに関する解答群

- ア 与えられたMの値が1のとき、ループ内の加算処理の実行回数を0にできる
イ 与えられたNの値が1のとき、ループ内の加算処理の実行回数を0にできる
ウ 中間結果のシフトとNのビットの順次取出しを、1回のシフトで済ませる
エ 中間結果のシフトによって、与えられたNの値の符号検査をなくせる

e, fに関する解答群

- | | | |
|-----|-----|-----|
| ア 1 | イ 2 | ウ 3 |
| エ 4 | オ 5 | カ 6 |

gに関する解答群

- | | | | |
|----------------|----------------|----------------|----------------|
| ア $-2^0 + 2^3$ | イ $-2^0 + 2^4$ | ウ $-2^1 + 2^3$ | エ $-2^1 + 2^4$ |
|----------------|----------------|----------------|----------------|