

情報実験第三課題 1.A

情報工学科 15_03602 柿沼 建太郎

情報工学科 15_10588 中田 光

平成 29 年 5 月 2 日

各課題担当者

各課題と担当者を表として以下に示す。

課題番号/名前	柿沼	中田
1	○	○
2	○	○
3	○	○
4	○	○
5(test_io1)		○
5(test_io2)		○
5(test_calc1)	○	

課題プログラムレポート(1,2,3,4)

倍精度乗算

柿沼

流れについての説明

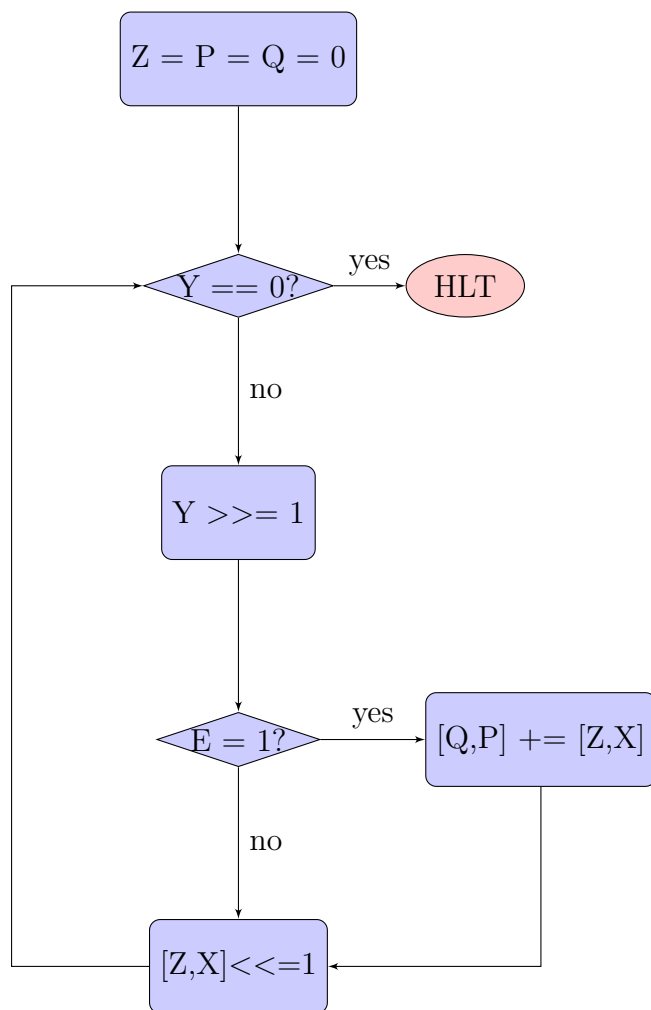
使用した変数について, 以下に示す。

変数名	説明
X	乗算の対象
Y	乗算の対象
Z	乗算の対象 (X の拡張用)
P	乗算の結果 (下位 16bit)
Q	乗算の結果 (上位 16bit)

乗算は筆算式に実現した。Y を右シフトしながら X を左シフトしていき、Y の末尾が 1 だった場合にはその時の X の値を結果に加算していく。このとき、X は 16 回左にシフトするが、結果は倍精度とするためそのままでは上位 16bit の情報が失われてしまう。

そこで、新しく Z という変数を使い [Z,X] を 32bit の変数として扱うことで倍精度計算を実現した。

以下にフローチャートを示す。



工夫点

$[Q,P] += [Z,X]$ を実現する際、P と X の加算によってあふれた bit が E レジスタに入ることを利用し、加算の後 E レジスタの値が 1 だった場合には 1 回 INC することで実現した。

ソースコードと総命令数

Listing 1: report1.1.asm

```

1 | ORG 10
2 | / Y == 0 ? HLT : goto LY

```

```

3  L0,
4    CLE
5    LDA Y
6    SZA
7    BUN LY
8    HLT
9    / Y >>= 1, E == 0 ? goto LX : goto LP
10 LY,
11   CIR
12   STA Y
13   SZE
14   BUN LP
15 / [Z,X] <<= 1, goto L0
16 LX,
17   LDA X
18   CIL
19   STA X
20   LDA Z
21   CIL
22   STA Z
23   BUN L0
24 / [Q,P] += [Z,X]
25 LP,
26   LDA X
27   ADD P
28   STA P
29   LDA Z
30   SZE
31   INC
32   ADD Q
33   STA Q
34   CLE
35   BUN LX
36 / data
37 X, DEC 65535
38 Y, DEC 65535
39 Z, DEC 0
40 P, DEC 0
41 Q, DEC 0
42 END

```

総命令数:26

入力 ($X \times Y$)	ステップ数
11×13	90
0×30	114
30×0	4
65535×65535	403

異なる入力に対する実行命令ステップ数

剰余算

柿沼

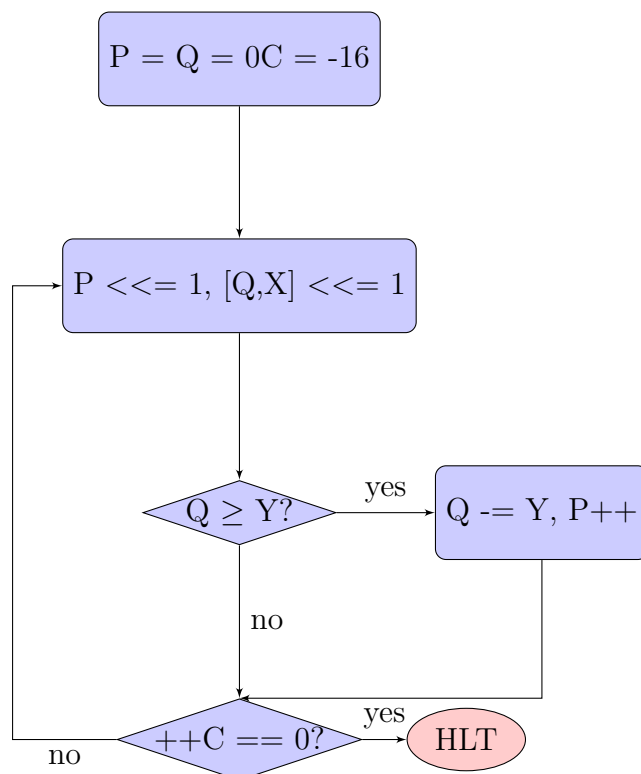
流れについての説明

使用した変数について, 以下に示す。

変数名	説明
X	徐算の対象
Y	徐算の対象
P	徐算の商
Q	徐算の余り
C	徐算の余り

徐算は筆算式に実現した。 $[Q, X]$ を左シフトしながら商も左シフトしていき、Q に溜まった数が Y 以上となったときに Q から Y を引き、商の末尾ビットを立てる。これを 16bit ぶん繰り返した。

以下にフローチャートを示す。



工夫点

$Q \geq Y$ の部分で $Q - Y$ を計算するため、それをそのまま $Q -= Y$ に転用した。

$Q \geq Y$ の計算は符号なし 16bit 整数として計算しなければならないため、E レジスタを含めた符号あり 17bit と考えて、適切な計算の後 SZE で判定するという手法をとった。

Y を E レジスタを含めた 17bit 符号あり整数と考えた場合、符号反転後はほとんどの場合最上位ビット (E レジスタ) が立っていることになるが、唯一 Y が 0 のときのみ符号を反転しても E レジスタは 0 のままなため、反転前の Y に対して SZA で E レジスタの値を定めた。

ソースコードと総命令数

Listing 2: report1.2.asm

```
1 | ORG 10
```

```

2 / P<<= 1, [Q,X] <<= 1
3 LD0,
4     CLE
5     LDA P
6     CIL
7     STA P
8     LDA X
9     CIL
10    STA X
11    LDA Q
12    CIL
13    STA Q
14 / Q >= Y ? goto LD1 : goto LD2
15    LDA Y
16    CLE
17    SZA
18    CME
19    CMA
20    INC
21    ADD Q
22    SZE
23    BUN LD1
24    BUN LD2
25 / Q -= Y, P++
26 LD1,
27    STA Q
28    LDA P
29    INC
30    STA P
31 / ++C == 0 ? HLT : goto LD0
32 LD2,
33    ISZ C
34    BUN LD0
35    HLT
36 / data
37 X, DEC 65535
38 Y, DEC 65535
39 P, DEC 0
40 Q, DEC 0
41 C, DEC -16
42 END

```

総命令数:28

入力 ($X \div Y$)	ステップ数
$30 \div 7$	340
$0 \div 15$	336
$65535 \div 1$	400
$65535 \div 65535$	340

異なる入力に対する実行命令ステップ数

16進 → 10進

柿沼

流れについての説明

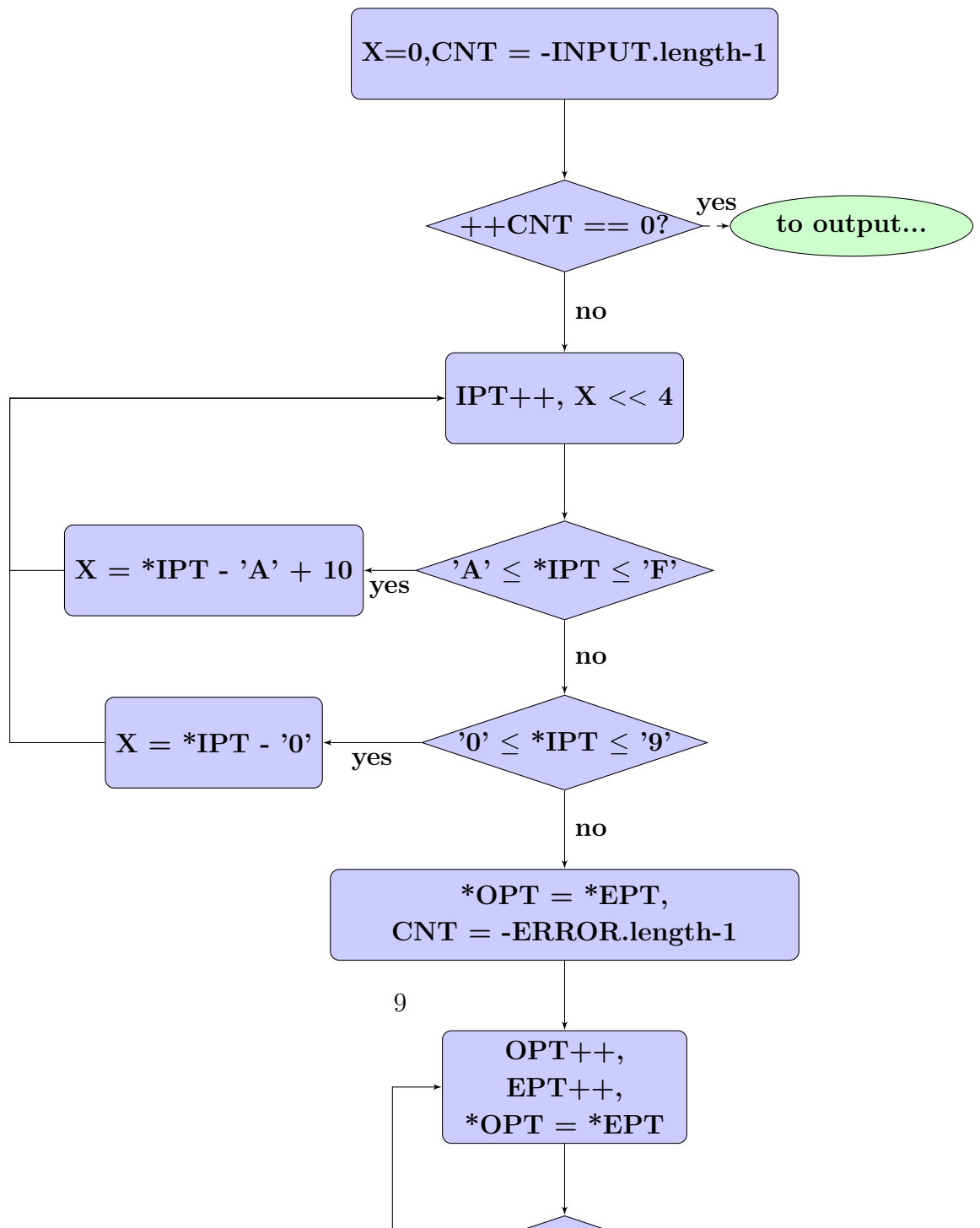
使用した変数について, 以下に示す。使用した定数について, 以下に示す。

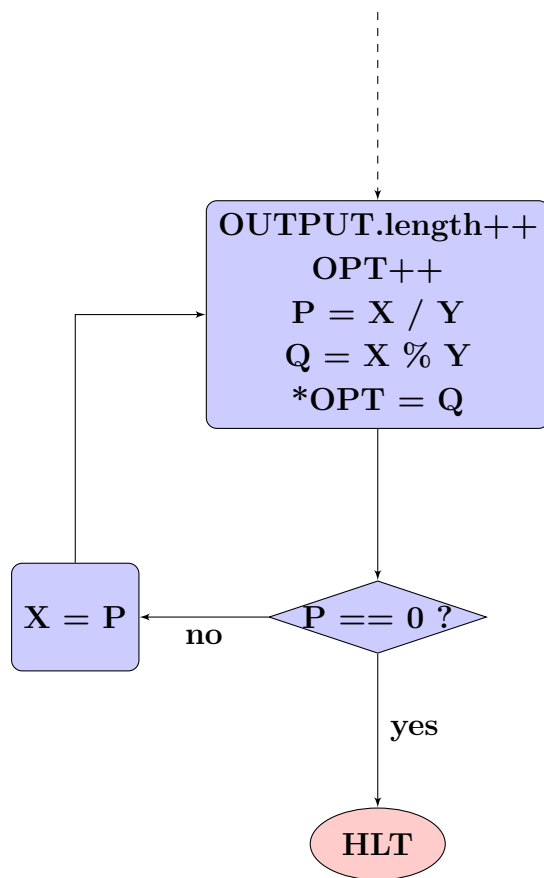
変数名	説明
OUTPUT	出力文字列
IPT	入力文字列の先頭へのポインタ
OPT	出力文字列の先頭へのポインタ
EPT	エラー文字列の先頭へのポインタ
CNT	文字列のカウント用変数
X	徐算の対象
Y	徐算の対象
P	徐算の商
Q	徐算の余り
C	徐算の余り

16進数文字列として格納された INPUT を数値として X に格納し、1桁ずつ OUTPUT に入れていく。X への格納に不備が発生した場合には ERROR を OUTPUT にコピーする。

以下にフローチャートを示す。

定数名	説明
TEN	即値「10」
A	即値「'A'」
F	即値「'F'」
ZERO	即値「'0'」
NINE	即値「'9'」
CINIT	即値「-16」
INPUT	入力文字列
ERROR	エラー文字列





工夫点

ソースコードと総命令数

Listing 3: report1_3.asm

```

1  ORG 10
2  / CNT = -INPUT.length-1
3  CLA
4  ADD IPT I
5  CMA
6  STA CNT
7  / ++CNT == 0 ? LO : goto L1
8  L0,
9  ISZ CNT
10 BUN L1
11 BUN LO
12 / IPT++
13 L1,

```

```

14 |   ISZ IPT
15 | / X <=<= 4
16 |   LDA X
17 |   CIL
18 |   CIL
19 |   CIL
20 |   CIL
21 |   STA X
22 | / AC = INPUT
23 |   CLA
24 |   ADD IPT I
25 | / input > 'F' goto LE
26 |   CMA
27 |   INC
28 |   ADD F
29 |   SPA
30 |   BUN LE
31 | / input >= 'A' goto LA
32 |   LDA A
33 |   CMA
34 |   INC
35 |   ADD IPT I
36 |   SNA
37 |   BUN LA
38 | / input < '0' goto LE
39 |   LDA ZERO
40 |   CMA
41 |   INC
42 |   ADD IPT I
43 |   SPA
44 |   BUN LE
45 | / input <= '9' goto LN
46 |   CLA
47 |   ADD IPT I
48 |   CMA
49 |   INC
50 |   ADD NINE
51 |   SNA
52 |   BUN LN
53 | / output 'ERROR'
54 | / *OPT = *EPT, CNT = -ERROR.length-1
55 | LE,
56 |   CLA
57 |   ADD EPT I
58 |   STA OPT I
59 |   CMA
60 |   STA CNT
61 | / EPT++, OPT++, *OPT = *EPT
62 | LE2,

```

```

63     ISZ EPT
64     ISZ OPT
65     CLA
66     ADD EPT I
67     STA OPT I
68 / ++CNT == 0 ? HLT : goto LE2
69     ISZ CNT
70     BUN LE2
71     HLT
72 / AlphaBet to Dec
73 / X += input - 'A' + 10, goto L0
74 LA,
75     LDA A
76     CMA
77     ADD TEN
78     BUN LX
79 / X += input - '0', goto L0
80 LN,
81     LDA ZERO
82     CMA
83 LX,
84     INC
85     ADD IPT I
86     ADD X
87     STA X
88     BUN L0
89 /binary -> dec
90 /OUTPUT.length++, OPT++, P = X/Y, Q = X%Y, *OPT = Q
91 LO,
92     ISZ OUTPUT
93     ISZ OPT
94     BSA LD
95     LDA Q
96     STA OPT I
97 / P == 0 ? HLT
98     LDA P
99     SZA
100    BUN LD3
101    HLT
102 / X = P
103 LD3,
104     STA X
105     BUN LO
106 LD,
107     HEX 0
108     CLA
109     STA P
110     STA Q
111     LDA TEN

```

```

112     STA Y
113     LDA CINIT
114     STA C
115 / P<<= 1, [Q,X] <<= 1
116 LD0,
117     CLE
118     LDA P
119     CIL
120     STA P
121     LDA X
122     CIL
123     STA X
124     LDA Q
125     CIL
126     STA Q
127 / Q >= Y ? goto LD1 : goto LD2
128     LDA Y
129     CLE
130     SZA
131     CME
132     CMA
133     INC
134     ADD Q
135     SZE
136     BUN LD1
137     BUN LD2
138 / Q -= Y, P++
139 LD1,
140     STA Q
141     LDA P
142     INC
143     STA P
144 / ++C == 0 ? HLT : goto LD0
145 LD2,
146     ISZ C
147     BUN LD0
148     BUN LD I
149 / data
150 TEN, DEC 10
151 A, CHR A
152 F, CHR F
153 ZERO, CHR 0
154 NINE, CHR 9
155 CINIT, DEC -16
156 INPUT, DEC 4
157 CHR A
158 CHR B
159 CHR C
160 CHR X

```

```

161 OUTPUT, DEC 0
162   CHR _
163   CHR _
164   CHR _
165   CHR _
166   CHR _
167 ERROR, DEC 5
168   CHR R
169   CHR O
170   CHR R
171   CHR R
172   CHR E
173 IPT, SYM INPUT
174 OPT, SYM OUTPUT
175 EPT, SYM ERROR
176 X, DEC 0
177 Y, DEC 10
178 P, DEC 0
179 Q, DEC 0
180 C, DEC -16
181 CNT, DEC 0
182 END

```

総命令数:121

異なる入力に対する実行命令ステップ数

入力 (X)	ステップ数
0F	783
FFFF	1957
XX	67
ABCX	157

素数計算

柿沼

流れについての説明

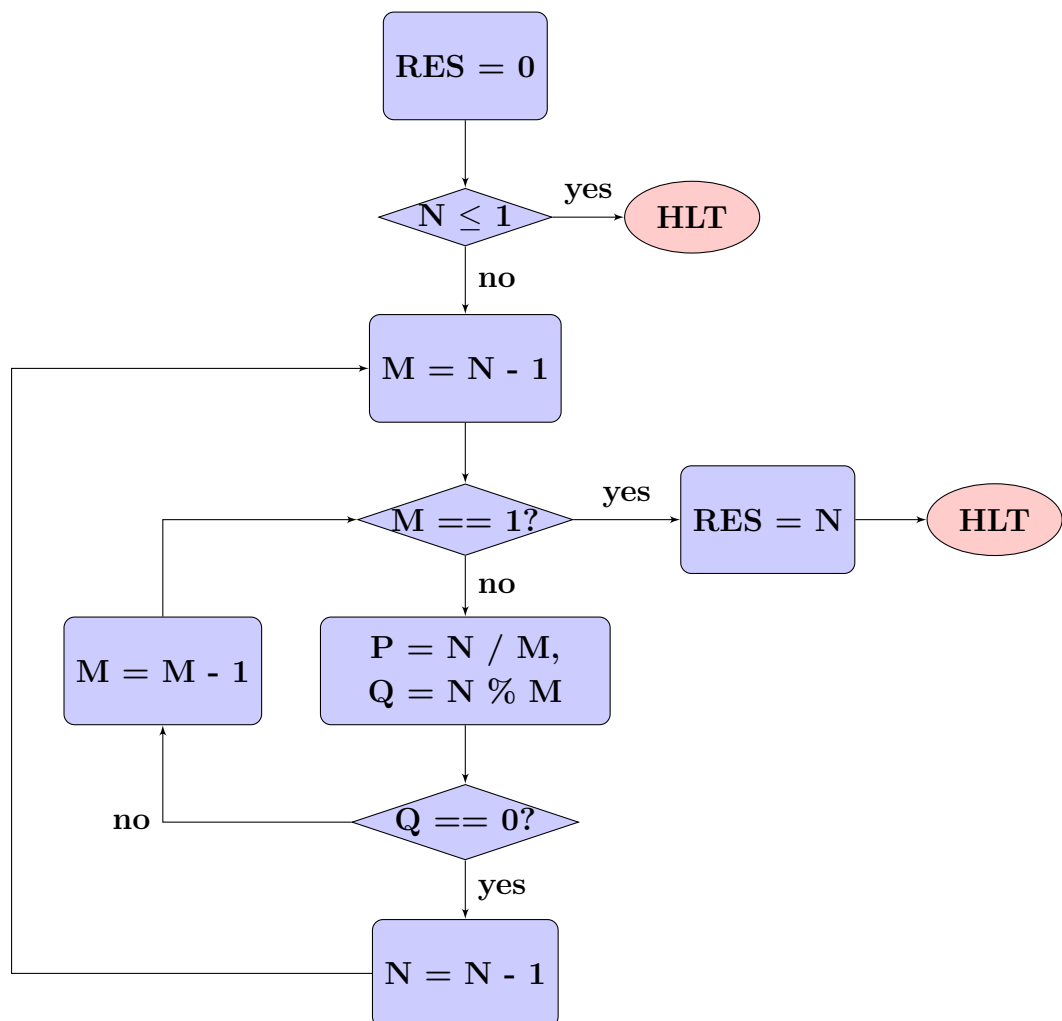
使用した変数について, 以下に示す。使用した定数について, 以下に示す。

変数名	説明
N	現在の素数候補
M	N を割る数
RES	結果
X	徐算の対象
Y	徐算の対象
P	徐算の商
Q	徐算の余り
C	徐算の余り

定数名	説明
CINIT	即値「-16」

自然数 N が素数であるかの判定には N 未満の自然数すべてに対して徐算を実行することで実現した。

以下にフローチャートを示す。



工夫点

$Q \geq Y$ の部分で $Q - Y$ を計算するため、それをそのまま $Q -= Y$ に転用した。

$Q \geq Y$ の計算は符号なし 16bit 整数として計算しなければならないため、E レジスタを含めた符号あり 17bit と考えて、適切な計算の後 SZE で判定するという手法をとった。

Y を E レジスタを含めた 17bit 符号あり整数と考えた場合、符号反転後はほとんどの場合最上位ビット (E レジスタ) が立っていることになるが、唯一 Y が 0 のときのみ符号を反転しても E レジスタは 0 のままなため、反転前の Y に対して SZA で E レジスタの値を定めた。

ソースコードと総命令数

Listing 4: report1_4.asm

```
1  ORG 10
2  /N <= 1 ? HLT
3      LDA N
4      SZA
5      CME
6      CMA
7      ADD TWO
8      SZE
9      HLT
10 / M = N-1
11 L0,
12     LDA N
13     BSA DC
14     STA M
15 / M == 1 ? RES = N, HLT : goto L2
16 L1,
17     LDA M
18     BSA DC
19     SZA
20     BUN L2
21     LDA N
22     STA RES
23     HLT
24 / P = N / M, Q = N % M, Q == 0 ? N--, goto L0 : goto L3
25 L2,
26     LDA N
27     STA X
28     LDA M
29     STA Y
30     BSA LD
31     LDA Q
32     SZA
33     BUN L3
34     LDA N
35     BSA DC
36     STA N
37     BUN L0
38 / M--, goto L1
39 L3,
40     LDA M
41     BSA DC
42     STA M
43     BUN L1
44 /P = 0, Q = 0, C = -16
45 LD,
```

```

46      HEX 0
47      CLA
48      STA P
49      STA Q
50      LDA CINIT
51      STA C
52      / P<<= 1, [Q,X] <<= 1
53 LD0,
54      CLE
55      LDA P
56      CIL
57      STA P
58      LDA X
59      CIL
60      STA X
61      LDA Q
62      CIL
63      STA Q
64      / Q >= Y ? goto LD1 : goto LD2
65      LDA Y
66      CLE
67      SZA
68      CME
69      CMA
70      INC
71      ADD Q
72      SZE
73      BUN LD1
74      BUN LD2
75      / Q -= Y, P++
76 LD1,
77      STA Q
78      LDA P
79      INC
80      STA P
81      / ++C == 0 ? HLT : goto LD0
82 LD2,
83      ISZ C
84      BUN LD0
85      BUN LD I
86      /AC = AC-1
87 DC,
88      HEX 0
89      CMA
90      INC
91      INC
92      CMA
93      INC
94      BUN DC I

```

```

95 / data
96 CINIT, DEC -16
97 TWO, DEC 2
98 RES, DEC 0
99 N, DEC 65535
100 M, DEC 0
101 X, DEC 0
102 Y, DEC 0
103 P, DEC 0
104 Q, DEC 0
105 C, DEC 0
106 END

```

総命令数:72

異なる入力に対する実行命令ステップ数

入力 (N)	ステップ数
2	27
64	61307
255	337101
65535	245377052

サンプルプログラム解析レポート (5)

0.1 test_calc1.asm

0.1.1 プログラムの概要

入力された数値の計算をさせるプログラムである。ユーザーは最大 4 桁の 16 進数表記の数値の足し算と引き算ができる。オーバーフローやアンダーフローは無視し、下位 16bit を答えとして出力する。入力する数値が 4 桁を超えたりするとエラー文を出力して終了する。

各サブルーチンの機能

サブルーチン名	機能の説明
INI	プログラム本体のエントリーポイント。各種変数の初期化をした後、入力可能状態で待機し、BYE が 1 になると停止する。
INLST	各種変数の初期化
ST0	割り込みハンドラのエントリーポイント
CHK_CH	AC と TMI が等しいかどうかを返す。
SET_MSG	PTR_MG に AC の指すアドレスを入れ、その指す文字列長を CNT に入れる。
READ_HX	TMI に入った文字が 16 進数として有効な文字なら HXI にその値を入れ、 $AC \geq$ にする。無効なら $AC < 0$ にする。このとき、文字が改行文字ならば=に置き換える。
CHK_DGT	引数として AC とその他に 2 つの定数を取り、AC がそれらの間に入っているかどうかを返す。
WRITE_Z	Z に入った数値を文字列として ZMG に入れる。

処理の流れ

