修 士 論 文

楕円曲線暗号ハードウェアの設計法と性能評価

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

白勢 政明

2003 年 3 月

修 士 論 文

楕円曲線暗号ハードウェアの設計法と性能評価

指導教官　日比野靖 教授

審査委員主査　日比野靖 教授
審査委員　田中清史 助教授
審査委員　堀口進 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

110062 白勢 政明

提出年月: 2003 年 2 月

概 要

本稿では、楕円曲線暗号の処理をハードウェア処理するにあたり必要となる剰余計算アルゴリズム、暗号演算、制御のための命令を提案する。剰余計算アルゴリズムはハードウェア処理に適しており、除算を使わないので効率的に計算できる。暗号処理では本稿で提案する一種類の暗号演算のみで行うことができ、制御が容易になる。また、それらを取り入れた暗号ハードウェアを設計し性能を評価する。

目 次

第 1 章　はじめに **1**
1.1　本研究の背景と目的 1
1.2　本論文の構成 2

第 2 章　楕円曲線暗号 **3**
2.1　有限体 $\mathbf{F_p}$ 3
2.2　剰余計算アルゴリズム 3
2.3　楕円曲線 7
2.4　楕円曲線暗号 9
2.5　Weierstrass 型と Montgomery 型楕円曲線の比較 12
2.6　楕円曲線の選び方 13

第 3 章　暗号処理の効率化 **15**
3.1　除算の削減 15
3.2　暗号演算 17

第 4 章　暗号ハードウェアの命令セット **19**
4.1　アドレス付け 19
4.2　パイプライン化 20
4.3　オペランド選択命令 22
4.4　カウンタ命令とスカラー命令 23
4.5　無限遠点 $\mathcal{O}$ について 24
4.6　無限遠点 $\mathcal{O}$ を考慮にいれた命令セット 25

第 5 章　暗号ハードウェアの概要 **29**
5.1　キューと出力制御器 29
5.2　キューが空のときの制御 31
5.3　乱数発生器 32
5.4　カウンタ 32
5.5　命令生成器 33
5.6　その他の制御器 33
5.7　暗号ハードウェアの概要図 33

第 6 章　演算器の設計 **35**
6.1 Wallace tree 乗算器 35
6.2 多入力加算器 40
6.3 $\mathbf{F_p}$ 乗算器 41
6.4 $\mathbf{F_p}$ 加算器と $\mathbf{F_p}$ 減算器の設計 44
6.5 ecc 演算器の概要とパイプライン分け 45

第 7 章　キュー、乱数発生器、制御器の設計 **47**
7.1 キューの設計 47
7.2 乱数発生器の設計 48
7.3 制御器の設計 48
7.3.1 カウンタの設計 48
7.3.2 命令シーケンサの設計 49
7.3.3 スカラー設定器の設計 50
7.3.4 qo に関する制御器 50
7.3.5 オペランド選択器 51
7.3.6 書き込み制御器の設計 51
7.3.7 出力制御器の設計 51
7.4 暗号ハードウェアのゲート数と面積 52

第 8 章　性能評価と今後の課題 **55**
8.1 性能評価 55
8.2 今後の課題 56

付 録 A **59**
A.1 MPSCSA*i* が使用する fa*n* について 59
A.2 命令コード 63
A.3 シミュレーション結果 75

第1章　はじめに

1.1　本研究の背景と目的

暗号には2つの鍵、暗号化鍵と復号化鍵が必要である。暗号化鍵とは元のメッセージ (平文という) を暗号文に変換するために必要な情報であり、逆に暗号文を復号するために必要な情報が復号化鍵である。

簡単な例としてアルファベットを右に3つ巡回させる暗号を考えてみる。

アルファベット	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	···
暗号のためのアルファベット	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	···

平文を”hello”とすると暗号文は”khoor”となる。これを復号するには逆に3つずらせば良い。この場合、暗号化鍵は+3(3つずらす) であり、復号化鍵は−3(逆に3つずらす) である。この暗号ではどちらかの鍵を知っていればもう片方の鍵を簡単に知ることができる。このような暗号を秘密鍵暗号または対称暗号という。秘密鍵暗号は鍵を共有できる小さな集団でしか使用することができない。数十年前までは暗号は秘密鍵暗号しか知られていなかった。

1976年にW.DiffieとM.Hellmanが鍵が対称的でない、つまり暗号化鍵から復号化鍵を見つけることができないか非常に困難な暗号を発明した。このような暗号を公開鍵暗号または非対称暗号という。(イギリス政府通信本部が1960年代に公開鍵暗号を発明していたが、このことを最近まで公表していなかった[1]。) 公開鍵暗号では暗号化鍵を公開鍵 (public key)、復号化鍵を秘密鍵 (secret key) ともいう。公開鍵暗号では秘密鍵だけを保管し公開鍵を公開することができる。よって鍵を公開している者へはメッセージを暗号化して送ることができる。A氏は暗号化鍵 P_k を公開し復号化鍵 S_k を保管しているとする。誰もが公開されている P_k を使ってA氏に送りたいメッセージを暗号化できる。A氏は保管している S_k を使ってその暗号文を読むことができる。

情報化社会の進展にともない情報セキュリティの重要性が増大している。暗号技術は情報セキュリティにおける主要な技術の1つである。大きなネットワークによる不特定多数の人との通信には公開鍵暗号が適している。

公開鍵暗号は処理が遅いことが欠点である。このため秘密鍵暗号の鍵の配送に公開鍵暗号を使い、メッセージの暗号化にはその鍵による秘密鍵暗号を使うというハイブリッド暗号を使うことが多い。ハイブリッド暗号で使用する秘密鍵暗号の解読法が見つかった場合、ハイブリッド暗号は機能しなくなるが公開鍵暗号単独の場合は機能している。よってハイブリッド暗号より公開鍵暗号の方が安全である。またハイブリッド暗号では2つのタイプの暗号処理を行うためのデバイスが必要となる。以上から処理速度が十分速いのならば公開鍵暗号を単独で使うのが望ましい。

本研究は公開鍵暗号の一種である楕円曲線暗号のハードウェアの高速かつ効率的な設計法を提案し、性能評価を行う。暗号ハードウェアの目的は、高速性の追求以外にもサーバ用のアクセラレータでの使用がある。

1.2 本論文の構成

第 2 章では楕円曲線暗号に必要な数学的項目、有限体 F_p、楕円曲線について説明し、それから楕円曲線暗号の処理について説明する。また、ハードウェア処理に適している剰余計算アルゴリズムを紹介し、効率的な楕円曲線の選び方も考察する。

第 3 章では暗号処理の効率化を説明する。除算の回数を減らすことで暗号処理のスループットを向上させることができ、また、暗号処理演算を 1 種類にすることにより暗号ハードウェアの制御を簡単にできる。

第 4 章では暗号ハードウェアの制御のために必要な命令について説明する。暗号ハードウェアが 14 段にパイプライン化されているときは、80 ビット長で 882 ワードの命令が必要である。

暗号ハードウェアに必要な演算器は第 3 章で説明するので、第 5 章では暗号ハードウエアのうち演算器以外のコンポーネントについて説明する。暗号ハードウェアには演算器以外にキュー、乱数発生器、カウンタ、各制御器が必要である。それから暗号ハードウェアの概要について説明する。

第 6 章では演算器の設計法について説明する。演算器は加算器、減算器、乗算器、剰余計算器に分けることができる。乗算器は高速性のために Wallace tree 乗算器を用い、規則的なゲートの配置法を考える。加算器は普通の桁上げ先見加算器を用い、減算器は特殊な計算を行うが加算器と同様な構造となる。剰余計算器は第 2 章で説明する剰余計算を行うが、法とする値を限定することで回路規模を小さくし処理時間を短縮させることができる。法とする値を限定することは楕円曲線暗号の処理においては問題とならない。PARTHENON による面積、ゲート数、最大遅延時間の評価も行う。

第 7 章ではキューや乱数発生器や制御器などの演算器以外のコンポーネントの設計について説明する。第 6 章と同様に PARTHENON による評価も行う。

第 8 章では暗号ハードウェアの性能評価とソフトウェア処理との比較や他の公開鍵暗号ハードウェアと比較する。また、本研究の問題点や今後の課題について考察する。

第2章　楕円曲線暗号

この章でははじめに楕円曲線暗号で必要な有限体 F_p での計算、剰余アルゴリズムを説明する。それから楕円曲線の定義と加算公式、楕円曲線暗号の処理の方法を説明し、暗号で使用する楕円曲線の選び方を説明する。

2.1　有限体 $\mathbf{F_p}$

体とは簡単に言うと 0 で割ることを除く四則演算が定義されている集合である。たとえば有理数全体の集合 Q や実数全体の集合 R は体であるが、整数全体の集合 Z は体でない。例えば 1 と 2 は Z の元であるが、1/2 は Z の元ではない。

p を素数とする。p 個の有限集合 $F_p = \{0, 1, 2, \cdots p-1\}$ には四則演算が定義できるので F_p は体である。$x,\ y$ を F_p の元とすると F_p での加減乗算は次のようにに行う。

加算　普通に加算 $x + y$ を計算してから p での剰余をとる。
減算　普通に減算 $x - y$ を計算してから p での剰余をとる。
乗算　普通に乗算 $x \cdot y$ を計算してから p での剰余をとる。

除算には Fermat の小定理と呼ばれる次の定理を使う。

定理 [2]
p を素数とする。任意の整数 x に対して $x^p \equiv x \pmod{p}$ が成り立つ。x が p で割り切れないならば $x^{p-1} \equiv 1 \pmod{p}$ が成り立つ。

この定理から F_p の 0 でない元 y に対して $y^{p-1} = 1$ となることが分かる。よって $y^{-1} = y^{p-2}$ である。F_p における除算は

$$x/y = x \cdot y^{-1} = x \cdot y^{p-2} \bmod p,$$

を計算すれば良い[1]。

2.2　剰余計算アルゴリズム

F_p での四則演算には剰余計算が必要である。(普通の) 除算はコストの高い演算なので、除算を使う剰余計算は効率が悪い。ここではハードウェア処理に適している剰余計算アルゴリズムを紹

[1] F_p での除算は Euchlid のアルゴリズムを使う方が効率的な場合もある。しかし Fermat の小定理を使うならば F_p 乗算器だけで F_p での除算を求められるが、Euchlid のアルゴリズムを使うときはそのためのハードウェアが必要となってしまう。よって F_p 除算には Fermat の小定理を使うことにした。

介する。

剰余計算アルゴリズム A
$m = 2^n - k,\ k > 0,\ k^2 < 2^n$ とする。$0 \le x \le (m-1)^2$ に対して $x \bmod m$ は次のようにして求められる。

1. $x = q \cdot 2^n + r,\ q \ge 0,\ 0 \le r < 2^n$ とする.
   (r は x の下位 n ビット、q はそれより上のビットである.)
2. $s = qk + r$ を計算.
3. $s = q' \cdot 2^n + r',\ q' > 0,\ 0 \le r' < 2^n$ とする.
   (r' は s の下位 n ビット、q' はそれより上のビットである.)
4. $s' = q'k + r',\ s'' = (q'+1)k + r'$ を計算する.
5. $s'' \ge 2^n$ ならば $(x \bmod m) = s'' - 2^n$,
   $s'' < 2^n$ ならば $(x \bmod m) = s'$.

証明
アルゴリズムの過程から

$$x = q \cdot 2^n + r, \qquad q \ge 0,\ 0 \le r \le 2^n. \tag{2.1}$$
$$s = qk + r = q' \cdot 2^n + r', \qquad q' \ge 0,\ 0 \le r' \le 2^n. \tag{2.2}$$
$$s' = q'k + r'. \tag{2.3}$$

という関係が成り立っている。(2.1) より

$$q \le \frac{x}{2^n} \le \frac{(2^n - k - 1)^2}{2^n}. \tag{2.4}$$

となり、アルゴリズムの条件 $k^2 < 2^n$ から

$$2^n + k^2 < 2^{n+1}. \tag{2.5}$$

となる。$R = (x \bmod m)$ とおくと $m = 2^n - k$ であるから

$$x = (q + Q)(2^n - k) + R, \qquad Q \ge 0,\ 0 \le R < 2^n - k. \tag{2.6}$$

と書ける。(2.1) と (2.6) から

$$r - R = -qk + Q \cdot 2^n - Qk. \tag{2.7}$$

となる。

ところで、$0 \le r < 2^n,\ 0 \le R < 2^n - k$ より $r - R < 2^n$ である。

$$\begin{aligned}
-qk + Q \cdot 2^n - Qk &< 2^n, \\
Q(2^n - k) &< 2^n + qk, \\
Q &< \frac{2^n + qk}{2^n - k}, \\
&\le \frac{2^n + \frac{(2^n - k + 1)^2}{2^n} \cdot k}{2^n - k}, \qquad (2.4) \text{ より} \\
&= \frac{2^{2n} + k(2^n - k - 1)^2}{2^n(2^n - k)}, \\
&< k + 1.
\end{aligned}$$

よって Q の範囲は $0 \le Q \le k$ である。(2.2) と (2.7) から

$$\begin{aligned}
R &= r + qk - Q \cdot 2^n + Qk, \\
&= q' \cdot 2^n + r' - Q \cdot 2^n + Qk, \\
&= (q' - Q) \cdot 2^n + r' + Qk.
\end{aligned}$$

$$(Q - q') \cdot 2^n = r' + Qk - R. \tag{2.8}$$

となる。また、$0 \le r' < 2^n,\ 0 \le Q \le k,\ 0 \le R \le 2^n - k$ であるから

$$-2^n < -(2^n - k) < (Q - q') \cdot 2^n = r' + Qk - R < 2^n + k^2 < 2^{n+1}.$$

となる。一番右の不等号は (2.5) から成り立つ。よって、$-1 < Q - q' < 2$ となり、$Q - q'$ は整数だから、$Q = q'$ または $Q = q' + 1$ である。

$$\begin{aligned}
Q = q' &\Leftrightarrow R = r' + Qk = r' + q'k = s' < 2^n - k, && ((2.8) \text{ より}) \\
&\Leftrightarrow R + k = s'' < 2^n. \\
Q = q' + 1 &\Leftrightarrow R + 2^n = r' + Qk = r' + (q' + 1)k = s'' \ge 2^n, && ((2.8) \text{ より}) \\
&\Leftrightarrow R = s'' - 2^n.
\end{aligned}$$

よって $s'' < 2^n$ ならば $(x \bmod m) = s'$、$s'' \le 2^n$ ならば $(x \bmod m) = s'' - 2^n$ となる。∎

アルゴリズム A では剰余計算を行うのに除算を必要としないので、高速に行うことができる。また、剰余計算の結果は s' と $s'' - 2^n$ のどちらかになるが、s' と s'' は並列に求めることができ、その選択には s'' の最上位ビットの値を使うことができるので効率的である。$x,\ y$ が F_p の元ならば $0 \le x,\ y \le m - 1$ であるから $0 \le x \cdot y \le (m - 1)^2$ となり、アルゴリズム A を使って剰余計算ができる。ただし、F_p の元 $x,\ y,\ z$ に対して $(x + y) \cdot z$ を求めるときは、普通の計算の結果で $m \le (x + y) \cdot z$ となることがあるのでアルゴリズム A を使うことはできない。アルゴリズム A を使って F_p での乗算を行う場合は乗数、被乗数とも剰余をとっておかなければならない。

剰余計算アルゴリズム B

$m = 2^n - k,\ 0 < k < 2^n$ とする。$0 \le x \le 2^{2n}/k - 2^n$ に対して $x \bmod m$ は次のようにして求められる。

1. $x = q \cdot 2^n + r,\ q \ge 0,\ 0 \le r < 2^n$ とする.
   (r は x の下位 n ビット、q はそれより上のビットである.)
2. $s = qk + r,\ s' = (q+1)k + r$ を計算する.
3. $s' \ge 2^n$ ならば $(x \bmod m) = s' - 2^n$,
   $s' < 2^n$ ならば $(x \bmod m) = s$.

証明

$x \equiv r + qk \equiv r + (q+1)k - 2^n \pmod m$ であることは次から分かる。

$$\begin{aligned} x &= q \cdot 2^n + r, \\ &= q(m+k) + r, \\ &\equiv r + qk = s && \pmod m, \\ &\equiv r + qk - m && \pmod m, \\ &= r + qk + k - 2^n \\ &= r + (q+1)k - 2^n = s' - 2^n. \end{aligned}$$

次に $s < 2m$ となることを示す。アルゴリズムの条件 $0 \le x \le 2^{2n}/k - 2^n$ から

$$\begin{aligned} kx &\le 2^{2n} - 2^n k, \\ 0 &\le 2^{2n} - 2^n k - kx, \\ 0 &\le 2^n - k - \frac{kx}{2^n}. \end{aligned} \tag{2.9}$$

である。

$$\begin{aligned} 2m - s &= 2(2^n - k) - qk - r \\ &= 2(2^n - k) - \frac{(x-r)k}{2^n} - r && (x = q \cdot 2^n + r \text{ より}) \\ &= 2(2^n - k) - \frac{xk}{2^n} + \left(\frac{k}{2^n} - 1\right) r \\ &> 2(2^n - k) - \frac{xk}{2^n} + (k - 2^n) && (0 \le r < m < 2^n \text{より}) \\ &= 2^n - k - \frac{xk}{2^n} \\ &\ge 0 && ((2.9) \text{ より}) \end{aligned}$$

よって $s < 2m$ が示された。$s' = s + k$ であるから、$s' \ge 2^n$ ならば $0 \le s' - 2^n = s + k - 2^n < 2m + k - 2^n = m$ となるので、$(x \bmod m) = s' - 2^n$ である。$s' < 2^n$ ならば $0 \le s = s' - k < 2^n - k = m$ となるので、$(x \bmod m) = s$ である。∎

アルゴリズム B の操作はステップ 2,3 を除いたアルゴリズム A の操作と同じである。加算結果の剰余に対してアルゴリズム B を使うことができる。加算の場合は、いくつかの普通の加算を行ってから、アルゴリズム B による剰余計算を 1 回行えば良い。

2.3 楕円曲線

この節では楕円曲線について説明する。楕円曲線暗号はその名の通り楕円曲線を使用する。楕円曲線は Weierstrass 型が一般的である。Weierstrass 型楕円曲線とは

$$E : y^2 = x^3 + ax + b,\ 4a^3 + 27b^2 \neq 0, \quad a,\ b \text{ は定数},$$

で表される 3 次曲線である[2]。本研究では Mongomery 型楕円曲線

$$E : By^2 = x^3 + Ax^2 + x,\ A \neq \pm 2,\ B \neq 0, \quad A,\ B \text{ は定数},$$

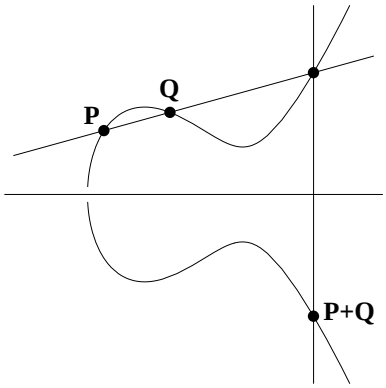
を使用する。$a,\ b$ または $A,\ B$ が体 K の元であるときはこのことを強調するために E/K と書く。Weierstrass 型の条件 $4a^3 + 27b^2 \neq 0$ や Montgomery 型の条件 $A \neq \pm 2$ は右辺の x の 3 次式が重根を持たないための条件である。これらの楕円曲線を実数平面に描くと x 軸に対称な滑らかな曲線となる。Montgomery 型楕円曲線は座標変換によって Wierstrass 型に変換できるが、逆はできない。よって Weierstrass 型楕円曲線は Montgomery 型楕円曲線より一般的な形式であるといえる。

楕円曲線 E/K に対して E の K 有理点の集合 $E(K)$ を次のように定義する。

$$E(K) = \{(x, y) \in K \times K : x, y \text{ は楕円曲線の式を満たす } \} \cup \{\mathcal{O}\}.$$

$\mathcal{O}$ は無限遠点と呼ばれる特別な点である。K が有限体ならば $E(K)$ は有限集合である。

次に $E/\boldsymbol{R}$($\boldsymbol{R}$ は実数全体の集合) のグラフを考える。$P,\ Q$ を $\mathcal{O}$ でない $E(\boldsymbol{R})$ の点とする。まずは $P \neq Q$ とする。E は 3 次曲線なので、P と Q を通る直線 l は別の点 R で E と交わる。(l は P か Q の接線になる場合があるがそのときは R はその接点とする。) x 軸に対する R の対称点を $P + Q$ とする。$P = Q$ の場合は P での接線 l と E とのもう 1 つの交点を R とし、x 軸に対する R の対称点を $2P$ とする。$E(R)$ の任意の点 P に対して $P = P + \mathcal{O} = \mathcal{O} + P$ とする。$P = (x,\ y)$ に対して $-P = (x,\ -y)$ とする。$E(\boldsymbol{R})$ この演算 $+$ に対して $\mathcal{O}$ を零元とする可換群 ($P + Q = Q + P$ となる群) をなす [2, 3]。つまり楕円曲線の点に結合法則を満たす加算が定義できる。



Weierstrass 型楕円曲線に対してこの操作は以下のように表すことができる。

[2] K の標数が 2, 3 の場合は Weierstrass 型楕円曲線の式はもっと複雑になる [3]。

Weierstrass 型楕円曲線の加法公式

$P,\ Q \in E,\ P=(x_P,\ y_P) \neq \mathcal{O},\ Q=(x_Q,\ y_Q) \neq \mathcal{O}$ とする。$R=P+Q$ とする。$R \neq \mathcal{O}$ ならば $R=(x_R,\ y_R)$ とする。

(1)$P+\mathcal{O}=\mathcal{O}+P=P,\ \mathcal{O}+\mathcal{O}=\mathcal{O}.$

(2)$x_P=x_Q,\ y_P=-y_Q \Leftrightarrow P=-Q$ のとき

$$R=\mathcal{O}=P+Q.$$

(3)(**加算公式**) $x_P \neq x_Q$ のとき

$$\begin{aligned} x_R &= \left(\frac{y_Q-y_P}{x_Q-x_P}\right)^2 - x_P - x_Q, \\ y_R &= -y_P + \left(\frac{y_Q-y_P}{x_Q-x_P}\right)(x_P-x_R). \end{aligned}$$

(4)(**2 倍公式**) $x_p=x_Q,\ y_P=y_Q \Leftrightarrow P=Q$ のとき

$$\begin{aligned} x_R &= \left(\frac{3x_P^2+a}{2y_P}\right)^2 - 2x_P, \\ y_R &= -y_P + \left(\frac{3x_P^2+a}{2y_P}\right)(x_P-x_R). \end{aligned}$$

任意の体 K に対して、この公式から $E(K)$ の任意の 2 つの元 $P,\ Q$ の和 $P+Q$ を求めると $P+Q$ の座標も K の元になるから $P+Q$ も $E(K)$ の元である。よって $E(K)$ は群になる。

点 P の t 個の和を

$$P+P+\cdots+P=tP,$$

と書く。

Montgomery 型楕円曲線の場合も同様な公式を導き、更に $x=X/Z,\ y=Y/Z$ とおくと次の公式を得る [4]。

Montgomery 型楕円曲線の加法公式

$P=(x_P,\ y_P),\ Q=(x_Q,\ y_Q),\ P \neq \mathcal{O},\ Q \neq \mathcal{O},\ P+Q=(x_{P+Q},\ y_{P+Q}),\ P-Q=(x_{P-Q},\ y_{P-Q})$ とする。$R=(x_R,\ y_R)$ に対して $x_R=X_R/Z_R$ とおく。

(1)(**加算公式**) $x_P \neq x_Q$ のとき

$$\begin{aligned} X_{P+Q} &= [(X_P-Z_P)(X_Q+Z_Q)+(X_P+Z_P)(X_Q-Z_Q)]^2. \\ Z_{P+Q} &= x_{P-Q}[(X_P-Z_P)(X_Q+Z_Q)-(X_P+Z_P)(Z_Q-Z_Q)]^2. \end{aligned}$$

(2)(**2 倍公式**) $P=(x_P,\ y_P),\ P \neq \mathcal{O},\ 2P=(x_{2P},\ y_{2P})$ のとき

$$\begin{aligned} 4X_PZ_P &= (X_P+Z_P)^2-(X_P-Z_P)^2. \\ X_{2P} &= (X_P+Z_P)^2(X_P-Z_P)^2. \\ Z_{2P} &= (4X_PZ_P)((X_P-Z_P)^2+\frac{A+2}{4}\ (4X_PZ_P)). \end{aligned}$$

Montgomery 楕円曲線では $P+Q$ の x 座標は $P,\ Q,\ P-Q$ の x 座標から求めることができ、y 座標を必要としない。

2.4 楕円曲線暗号

楕円曲線の有理点の群 $E(F_p)$ とある整数 t に対して、tP と P から t を求める問題を楕円曲線の離散対数問題という。$\#E(F_p)$ は $E(F_p)$ の元の個数を表すとする。l を $\#E(F_p)$ の最大素因数とする。このとき l が十分に大きく (l は 160 ビット以上)、

条件 $\mathcal{A}$

1. $1 \leq b < 20$ に対して $l^b - 1$ が p で割りきれない,
2. $\#E(F_p) \neq p$.

を満たすとき、楕円曲線の離散対数問題は困難であることが知られている [5, 6, 7]。

楕円曲線暗号は離散対数問題の困難性を利用している[3]。楕円曲線暗号システムでは楕円曲線 E/F_P と $E(F_p)$ の 1 つの元 B がベースポイントとして公表されている。暗号システム参加者は整数 s を選び $P_k = sB$ を求め、P_k を公開鍵として公開し s は秘密鍵として保管する。離散対数問題の困難性より、他人が B と P_k から s を知ることはできない (か非常に困難である)。

楕円曲線暗号の暗号化処理と復号化処理は次のようになる。

楕円曲線暗号の処理

暗号化	1. 乱数 r を選ぶ。 2.$(C_1,\ c_2) = (rB,\ x(rP_k) \oplus m)$ が暗号文となる。 ここで $x(rP_k)$ は rP_k の x 座標を表す。
復号化	$x(sC_1) \oplus c_2$ を計算する。

l を $\#E(F_p)$ の最大素因数とすると乱数の範囲は $1 \leq r < l$ であり、同様に秘密鍵 s も $1 < s < l$ で選ぶ。

$P_k = sB$ であるから

$$sC_1 = s(rB) = r(sB) = rP_k.$$

$$x(sC_1) \oplus c_2 = x(rP_k) \oplus x(rP_k) \oplus m = m.$$

よって復号化の操作から元のメッセージ m を得ることができる。

ところで Montgomery 型楕円曲線では y 座標を使わないで、$x(C_1)$ から $x(sC_1)$ を計算することができる。よって Montgomery 型楕円曲線を使う場合は暗号文は $(x(C_1),\ x(c_2) \oplus m)$ とすることができる。つまり、暗号文に c_1 の y 座標の情報が必要でない。同様に $x(B),\ x(P_k)$ から $x(rB),\ x(rP_k)$ を計算できる。

Montgomery 型楕円曲線による暗号処理

暗号化	1. 乱数 r を選ぶ。 2.$(x(C_1),\ c_2) = (x(rB),\ x(rP_k) \oplus m)$ が暗号文となる。 $x(rB),\ x(rP_k)$ は $r,\ x(B),\ x(P_k)$ から得られる。
復号化	$x(sC_1) \oplus c_2$ を計算する。

[3] 一般の有限群に対して離散対数問題が定義でき、離散対数問題の困難性を利用する暗号を ElGamal 暗号という。楕円曲線暗号は有限群に $E(F_p)$ を用いた ElGamal 暗号である。

また有名な公開鍵暗号である RSA は素因数分解の困難性を利用している。

$E(F_p)$ の元の個数 $\#E(F_p)$ に関して次の定理がある。

定理 (Hasse)[3]
$E(F_p)$ の元の個数 $\#E(F_p)$ に対して次が成り立つ。

$$p+1-2\sqrt{p} \leq \#E(F_p) \leq p+1+2\sqrt{p}.$$

Hasse の定理から係数を変えることにより $\#E(F_p)$ の値を動かすことができる。楕円曲線暗号の安全性は $\#E(F_p)$ の最大素因数 l のビット長に依存するが、同じ安全性を求める場合は F_p ではなく楕円曲線の係数を変えることで、異なる $E(F_p)$ を得ることができる。よって安全性の強度を変えないのならば必要な剰余計算を固定でき、これは楕円曲線暗号の長所の 1 つである。

$E(F_p)$ の元 P と整数 t に対して楕円曲線暗号の主な操作は、Wierstrass 型楕円曲線を使う場合は P から tP を求めることであり、Montgomery 型楕円曲線では $x(P)$ から $x(tP)$ を求めることである。t は n ビットで 2 進表現が $(t_{n-1}t_{n-2}\cdots t_1t_0)_2$ であるとする。加法公式を使って tP や $x(tP)$ を求めるアルゴリズムは次にようになる。

アルゴリズム C (Wierstrass 型楕円曲線で tP を求める)

$Q = \mathcal{O}$
for$(i = 0\ ;\ i < n\ ;\ i++)\{$
　　if $(t_i == 1)$ then $Q = P + Q$;
　　$P = 2P$;
$\}$
$(tP) = Q$;

$P+Q$ を計算するには加算公式を使い $2P$ を計算するには 2 倍公式を使うので、tP を求めるには加算公式を平均 $n/2$ 回、最大で n 回、2 倍公式を n 回使用する。

アルゴリズム D (Montgomery 型楕円曲線で $x(tP)$ を求める)

$Q' = \mathcal{O}\ ;\ P' = P$;
for$(i = 0\ ;\ i < n\ ;\ i++)\{$
　　if $(t_{n-1-i} == 0)\{$
　　　　$x(P') = x(P' + Q')$;
　　　　$x(Q') = x(2Q')$;
　　$\}$
　　else $\{$
　　　　$x(Q') = x(P' + Q')$;
　　　　$x(P') = x(2P')$;
　　$\}$
$\}$
$x(tP) = x(Q')$;

Montgomery 型では $x(P'+Q')$ を求めるには $x(P'),\ x(Q'),\ x(P'-Q')$ が必要であるが、アルゴリズム D では常に $x(P'-Q') = x(P)$ となっており $x(P)$ は入力する値であるから $x(P'+Q')$ を計算できる。$x(tP)$ を求めるのに加算公式を n 回、2 倍公式を n 回使用する。またアルゴリズム D で求まるのは $x(tP) = X/Z$ を満たす $(X,\ Z)$ であるので最後に除算 X/Z を行わなければならない。

tP を求めるアルゴリズムと同様に x^t を得ることができる。$t = (t_{n-1}t_{n-2}\cdots t_1t_0)$(2 進表現) とする。このとき x^t は次のアルゴリズム E またはアルゴリズム F で求まる。

アルゴリズム E (x^t を求める)

$y = 1$;
for($i = 0$; $i < n$; $i++$){
    if ($t_i == 1$) $y = x \cdot y$;
    $x = x \cdot x$;
}
$(x^t) = y$;

アルゴリズム F (x^t を求める)

$y = 1$;
for($i = 0$; $i < n$; $i++$){
    if($t_{n-1-i} == 0$){
        $x = x \cdot y$;
        $y = y \cdot y$;
    }
    else{
        $y = x \cdot y$;
        $x = x \cdot x$;
    }
}
$(x^t) = y$;

x^t を求めるには n から $2n$ 回の乗算が必要である。

tP や $x(tP)$ を求めるのに必要な計算量 (乗算と除算の回数) を調べてみる。はじめに Weirstrass 型で P から tP を求めるのに必要な乗算と除算の回数を調べる。Weierstrass 型楕円曲線では各公式に除算が含まれるので計算量は多くなるが、座標系を変えることで乗算と除算の回数を減らすことができる。$P = (x,\ y)$ に対して $x = X/Z\ \ y = Y/Z$ とおき, さらに aZ^4 の値を保持する Modified Jacobian 座標系 [8] を使うと加算公式の乗算の回数は 19 回、2 倍公式の乗算の回数は 8 回となる。Modified Jacobian 座標系では $X,\ Y,\ Z$ を使って計算するので、Montgomery 型と同様に最後に 2 つの除算 $x = X/Z,\ y = Y/Z$ を計算しなければならないが、除算はこの 2 回だけで良い。P から tP を求めるのに加算公式は平均で $n/2$ 回で最大で n 回、2 倍公式は n 回使うので、乗算の回数は平均で $19 \cdot n/2 + 8n = 17.5n$ 回、最大で $19n + 8n = 27n$ 回の乗算が必要である。

次に Montgomery 型楕円曲線で $x(P)$ から $x(tP)$ を求めるのに必要な乗算と除算の回数を調べる。加算公式に必要な乗算は

1 回目　$(X_P - Z_P)(X_Q + Z_Q)$,
2 回目　$(X_P + Z_P)(X_Q - Z_Q)$,
3 回目　$[(X_P - Z_P)(X_Q + Z_Q) + (X_P + Z_P)(X_Q - Z_Q)]^2$,
4 回目　$[(X_P - Z_P)(X_Q + Z_Q) - (X_P + Z_P)(X_Q - Z_Q)]^2$,
5 回目　$x_{P-Q}[(X_P - Z_P)(X_Q + Z_Q) - (X_P + Z_P)(X_Q - Z_Q)]^2$.

の 5 回である。2 倍公式に必要な乗算は

$$
\begin{aligned}
&1\text{回目}\quad (X_P+Z_P)^2,\\
&2\text{回目}\quad (X_P-Z_P)^2,\\
&3\text{回目}\quad (X_P+Z_P)^2(X_P+Z_P)^2,\\
&4\text{回目}\quad ((A+2)/4)(4X_pZ_P),\\
&5\text{回目}\quad (4X_PZ_P)((X_P-Z_P)^2-((A+2)/4)(4X_pZ_P)).
\end{aligned}
$$

の 5 つである。$4X_PZ_P$ は $4X_PZ_P=(X_P+Z_P)^2-(X_P-Z_P)^2$ より上 2 つの減算から得られる。$(A+2)/4$ は定数なのであらかじめ求めておく。$x(P)$ から $x(tP)$ を求めるのに加算公式と 2 倍公式を n 回ずつ使うので $10n$ 回の乗算が必要であり、除算は 1 回である。

2.5 Weierstrass 型と Montgomery 型楕円曲線の比較

$E(F_p)$ の最大素因数を l とすると楕円曲線の安全性 (離散対数問題の困難性) は l の大きさに依存している。十分な安全性を確保するために l は 160 ビット以上が必要である [5, 9]。前節の初めに説明した条件 $\mathcal{A}$ も満たす必要がある。l が 160 ビットのときの楕円曲線暗号の安全性は 1024 ビットの RSA 暗号に匹敵する。これは同じ安全性に対しては楕円曲線暗号の方が RSA 暗号より鍵長を小さくできることを意味する。本研究では l を 160 ビットに固定してハードウェアを設計する。

p を 160 ビットの素数とする。$\#E(F_p)$ が素数ならば $l=\#E(F_p)$ となり、Hasse の定理から l は p の値に近くなるので l も 160 ビットになる。(p が 160 ビット素数の中で小さいものならば l は 159 ビットになることもあるので、p は大きめの 160 ビット素数であるとする。また l が 161 ビットになる場合もあるが、この場合は安全性が低くなることはない。) Wierstrass 型楕円曲線では係数をうまく選ぶことにより $\#E(F_p)$ が素数となるものがある。しかし、Montgomery 型楕円曲線では $\#E(F_p)$ は必ず 4 の倍数となる [4]。よって Montgomery 型での最善な選択は $\#E(F_p)=4\,l$ となる楕円曲線を使うことであるが、この時 p は 162 ビットとなる。p が大きくなると演算器の規模が大きくなる。

次に楕円曲線暗号における鍵の大きさと暗号文の長さを考える。暗号に 160 ビット素数 p、$\#E(F_p)=l$ で l が 160 ビット素数となる Wierstrass 型楕円曲線を使うとする。秘密鍵 s は $1<s<l$ の範囲で選ぶので 160 ビット、公開鍵は P_k の $x,\ y$ 座標の情報が必要なので 320 ビットの情報となる。暗号文は 160 ビットの乱数 r と 160 ビットのメッセージ m に対して $(C_1,\ c_2)=(rB,\ x(rP_k)\oplus m)$ となるが、C_1 は $x,\ y$ 座標が必要なので 320 ビット、c_2 は 160 ビット、合計 480 ビットとなる。暗号文の長さは元のメッセージの 3 倍となる。

楕円曲線の点は x 座標が決まると y 座標の取り得る値は多くて 2 つなので、y 座標の情報を 1 ビットにすることができる。実際に y 座標の最下位 1 ビットと x 座標から y 座標を復元できる。この方法を point compression という [6]。point compression を使うと公開鍵の情報は 161 ビット、暗号文は C_1 の情報が 161 ビット、c_2 の情報が 160 ビットで合計 321 ビットとなる。暗号文の長さは元のメッセージの約 2 倍である。$p\equiv 3\pmod 4$ のときは y 座標を復元するのに x^3+ax+b の $\frac{p+1}{4}$ 乗を計算する必要があり、このためには最低 158 回の乗算が必要である。$p\equiv 1\pmod 4$ の場合は y 座標の復元はより複雑になる [2]。

暗号に 162 ビット素数 p、$\#E(F_p)=4\,l$ で l が 160 ビット素数となる Montgomery 型楕円曲線を使う場合を考える。秘密鍵 s は $1<s<l$ の範囲で選ぶので (162 ビットではなく)160 ビットで

表 2.1: Weierstrass 型と Montgomery 型の比較

	Eierstrass 型 (point compression 使用時)	Montgomery 型
tP(あるいは $x(tP)$) を求めるのに必要な乗算の回数	平均 2800 回 最大 4320 回	1600 回
tP(あるいは $x(tP)$) を求めるのに必要な除算の回数	2 回	1 回
除算 1 回あたりの乗算の回数	161〜321	163〜325
p の大きさ	160 ビット	162 ビット
暗号文のビット数	480 (321)	324
暗号文のビット数/メッセージのビット数	3 (2.006)	2
その他	point compression を使用するときは復号化時に 158 回以上の乗算が必要	

$\#E(F_p)$ の最大素因数 l と t は 160 ビットである。

ある。公開鍵は P_k の x 座標だけで良いので 162 ビットである。暗号文は 160 ビットの乱数 r と 162 ビットのメッセージ m に対して $(x(C_1),\ c_2) = (x(rB),\ x(rP_k) \oplus m)$ となる。$x(C_1)$ と c_2 は 162 ビットなので合計 324 ビットであり、暗号文の長さは元のメッセージの 2 倍である。

楕円曲線暗号での使用における Wierstrass 型楕円曲線と Montgomery 型楕円曲線との比較をまとめると表 2.1 となる。F_p の元 $x,\ y$ に対して $x/y = x \cdot y^{-1} = x \cdot y^{p-2}$ である。$p-2$ が n ビットの場合 y^{p-2} は n から $2n$ 回の乗算で求められるので、x/y は $n+1$ から $2n+1$ 回の乗算で求められる。

暗号に Montgomery 型楕円曲線を使用する方が回路規模が若干大きくなるが、その他の点では Wierstrass 型を使用するときと較べて長所が多いので、本研究は Montgomery 型楕円曲線を使用する暗号ハードウェアを設計する。

前節でべき乗の 2 つのアルゴリズム E,F を紹介したが、F_p での除算にアルゴリズム E を使うと 163〜325 回の乗算が必要となり、アルゴリズム F では 325 回と一定である。もし暗号の処理速度を一定にするならば乗算回数が最多になる場合の時間が必要となるので、両方とも除算に必要な時間は同じとなる。また Montgomery 型楕円曲線では $x(tP)$ を求めるときにアルゴリズム D を使うが、アルゴリズム D と F では t の参照の仕方が同じである。よって F_p 除算にはアルゴリズム F を使うこととする。このとき $x(tP)$ を求めるのには 1925 回の乗算が必要となる。

2.6 楕円曲線の選び方

十分な安全性を確保するために $\#E(F_p)$ の最大素因数 l が 160 ビット以上であり、2.4 節の条件 $\mathcal{A}$ を満たす必要がある。Montgomery 型楕円曲線では条件 $\mathcal{A}$ の 2. の場合は起り得ないので、1. だけを気をつければ良い。

暗号化では $1 \leq r < l$ の範囲の乱数 r を発生させる必要がある。Hasse の定理から $l > 2^{160}$ と $l < 2^{160}$ の場合がある。$l > 2^{160}$ の場合は乱数を $1 \leq r \leq 2^{160} - 1 (< l)$ の範囲で選べば良いので、0 を除く任意の 160 ビットの値を乱数として使用できる。$l < 2^{160}$ の場合に $1 \leq r \leq 2^{160} - 1$ とすると、$r \geq l$ に対しては $rP = (r-l)P$ となり、これは乱数が偏っているような影響を与え、更に

$r = l$ のときは $r = 0$ と同じであり $r = l$ とならないようにしなければならない。よって乱数発生機構を簡単にするために $l > 2^{160}$ であることが望ましい。

$E(F_p)$ の元 P に対して $tP = \mathcal{O}$ となる最小の自然数 t を P の位数という。位数は $\#E(F_p)$ の約数のどれかとなる。ある楕円曲線暗号システムではベースポイントとして $E(F_p)$ の元 B を 1 つ選ばなければならないが、位数が l となるように選ぶ。

$p = 2^{162} - 101$ は 162 ビットの素数で、Montgomery 型楕円曲線

$$E : y^2 = x^3 + 3 \cdot 3456x^2 + x,$$

は条件 A を満たし、素数 $l = 1461501637330902918203685718752061191448311153967$ に対して $\#E(F_p) = 4\,l$ となる。また $l > 2^{160}$ である。この楕円曲線は暗号に使用できる。$x = 5472016277628$ $0022257407447653685238536895012836 33$ となる $E(F_p)$ の元 $B = (x,\ y)$ は位数が l である。よってこれらの楕円曲線暗号に使用できる各値の例である。楕円曲線 E の係数と p の値から $\#E(F_p)$ を求める Schoof のアルゴリズムがあるので、これを利用して $y^2 = x^3 + 3 \cdot 3456 + x$ を見つけた。この $E(F_p)$ の元 P の位数は $1,\ 2,\ 4,\ l,\ 2l,\ 4l$ のどれかであるが、P の位数が $l,\ 2l,\ 4l$ ならば $4P$ の位数は l となる。位数が $1,\ 2,\ 4$ の元に対しては

$$\begin{aligned} P \text{ の位数が } 1 \quad &\Leftrightarrow \quad P = \mathcal{O}, \\ P \text{ の位数が } 2 \quad &\Leftrightarrow \quad y(P) = 0, \\ P \text{ の位数が } 4 \quad &\Leftrightarrow \quad x(P) = \pm 1[10], \end{aligned}$$

となる。$f(x) = x^3 + 3{\cdot}3456x^2 + x$ とおく。$1 < x < l-1$ に対して $f(x)$ が F_p での 2 乗数、つまり $y^2 = f(x)$ となる $y \in F_p$ があれば $(x,\ y)$ は位数が $l,\ 2l,\ 4l$ のどれかの $E(F_p)$ の元である。$f(x)$ が F_p での 2 乗数かどうかの判定は簡単で、例えば $f(x)^{(p-1)/2} \equiv 1 \pmod{p}$ ならば $f(x)$ は 2 乗数である。$f(2)$ は 2 乗数であり、$P = (2,\ y)$ とすると、$x(4P) = 5472016277628002225740744765368523853689501283633$ となる。

第3章　暗号処理の効率化

楕円曲線暗号の処理には F_p での四則演算が必要である。2.4 節で説明したように p が 162 ビットならば F_p での 325 回の乗算を必要とするコストの高い演算である。この章では最初に除算の回数を減らす方法を考える。また加減乗算の 3 種類の演算があると暗号ハードウェアの制御が複雑になるので、演算を 1 つにする方法を述べる。

3.1　除算の削減

p を 162 ビットの素数とする。F_p での除算には 325 回の乗算、逆元を求めるには 324 回の乗算が必要である。一般に除算は乗算よりコストが高いので除算を減らす工夫がなされている。2 つの除算

$$a/b,\ c/d.$$

を計算するとする。このとき、

$$a/b = a \cdot d(bd)^{-1},\ c/d = c \cdot b(bd)^{-1}.$$

とすると除算 (逆元を求めること) は $(bd)^{-1}$ を求めるときの 1 回だけで良い。但し 5 回の乗算も必要となる。このようにいくつかの除算をまとめることで除算の回数を減らすことを除算削減法と呼ぶことにする。$a,\ b,\ c,\ d$ が F_p の元ならば、a/b と c/d を個別に計算すると 650 回の乗算が必要だが、除算削減法を使うと 329 回の乗算で計算できる。

除算削減法を使わないときは $x(tP)$ を求めるには、$x(tP) = X/Z$ を満たす $X,\ Z$ を求めるのに 1600 回、X/Z を計算するのに 325 回、合計 1925 回の乗算が必要である。除算削減法を使うと 2 つの $x(tP),\ x(t'P')$ を求めるのに $1600 \times 2 + 329 = 3529$ 回の乗算で求められる。除算削減法を使うと

$$3529/(1925 \times 2) = 0.917 \text{ 倍}$$

に乗算の回数を減らすことができる。

同様に更に多くの除算を 1 度の除算で得るようにすることもできるがあまり効率的ではなさそうである。一般的な場合を考えてみる。逆元を求めるには m 回の乗算が必要であるとする。このとき除算に必要な乗算は $m + 1$ 回である。k 個の除算

$$X_0/Z_0,\ X_1/Z_1,\ \cdots, X_{k-1}/Z_{k-1}.$$

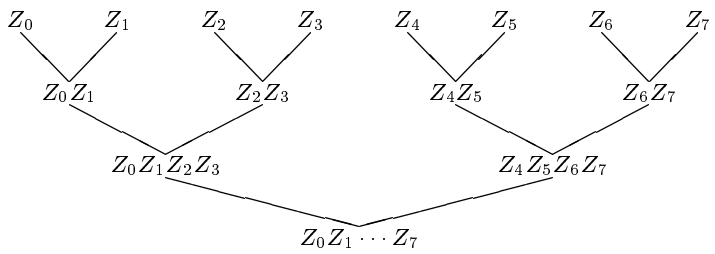
を行うとする。このとき、

$$X_i/Z_i = X_i \cdot (Z_0 Z_1 \cdots Z_{i-1} Z_{i+1} \cdots Z_{k-1})(Z_0 Z_1 \cdots Z_{k-1})^{-1}$$

と計算すれば除算 (逆元計算) は 1 回で済む。k 個の除算をまとめる除算削減法をレベル k の除算削減法と呼ぶことにする。

k は 2 のべきとする。ツリーを考えれば、$Z_0 Z_1 \cdots Z_{k-1}$ を求めるには $k-1$ 回の乗算が必要であることが分かる。

$k = 8$ の場合の例



ツリーのノードの $Z_0 Z_1$ や $Z_4 Z_5 Z_6 Z_7$ などの $k-1$ 個の値は $Z_0 Z_1 \cdots Z_{i-1} Z_{i+1} \cdots Z_{k-1}$ を計算するときに必要なので、保存しなければならない。保存すべき値の個数は $k-1$ である。

この方法で k 個の除算を求めるのに必要な乗算の個数は以下のようになる。

表 3.1: 除算削減法による乗算の回数

$Z_0 Z_1 \cdots Z_{k-1}$ の計算	$k-1$ 回
$(Z_0 Z_1 \cdots Z_{k-1})^{-1}$ の計算	m 回
$Z_0 Z_1 \cdots Z_{i-1} Z_{i+1} \cdots Z_{k-1}$ の計算	$\log_2 k$ 回
$X_i/Z_i = X_i \cdot (Z_0 Z_1 \cdots Z_{i-1} Z_{i+1} \cdot Z_{k-1}) \cdot (Z_0 Z_1 \cdots Z_{k-1})^{-1}$ の計算	2 回
合計	$= k \log_2 k + 2k + m - 1$ 回

除算削減法による乗算の回数と個別に k 個の除算を行うときの乗算の回数の比

$$\frac{k \log_2 k + 2k + m - 1}{k(m+1)}$$

は $k = (m-1)\log 2 = 0.69(m-1)$ のときに最小となる。

$x(tP)$ を求めるときの乗算の回数について考える。F_p (p は 162 ビット) では $m = 324$ である。k 個の $x(tP)$ を計算するのに必要な乗算の回数はレベル k の除算削減法を用いると $1602k + l \log_2 k + 323$ 回、除算削減法を用いない場合は $1925k$ 回である。これらの比

$$\frac{1602k + k \log_2 k + 323}{1925k} \tag{3.1}$$

と保存しなければならない途中結果の個数を表にする (次ページ)。除算削減法のレベルを大きくしても比 (3.1) はあまり小さくならない。保存する途中結果の個数を考えると $k = 2$ が良さそうである。本研究で設計する暗号ハードウェアはレベル 2 の除算削減法による除算を行うことにする。また除算削減法の欠点としてレベルにほぼ比例して暗号処理のレイテンシが増大することである。除算削減法の採用はレイテンシを犠牲にしてスループットの向上を選択することである。

	比 (3.1) の値	保存する途中結果の個数
k=1	1	0
k=2	0.917	1
k=4	0.875	3
k=8	0.855	7
k=16	0.845	15
k=32	0.840	31
k=64	0.838	63
k=128	0.837	127
k=256	0.837	255

3.2 暗号演算

Montgomery 型楕円曲線を使用するとき、楕円曲線の点 P の x 座標 $x(P)$ から tP の x 座標 $x(tP)$ を求めることが主な暗号処理となる。この操作には F_p での加減乗算 (除算は乗算で行える) が必要であるが、暗号ハードウェアは 3 つの演算を制御しなければならない。この制御を簡単にするために ecc(Elliptic Curve Cryptgraphy) 演算を導入する。

F_p の元 $X_0,\ X_1,\ X_2,\ X_3$ と $\{0,\ 1\}$ の元 $i,\ j$ に対して演算 $ecc(X_0,\ X_1,\ X_2,\ X_3,\ i,\ j)$ を次のように定義する。

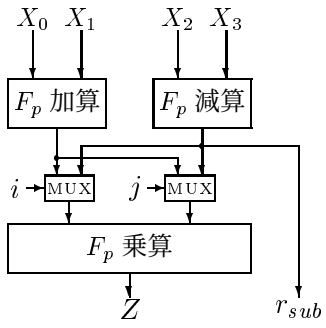
1. $Y_0 = X_0 + X_1$, $Y_1 = X_2 - X_3$,
2. $Y_i \cdot Y_j$ と Y_1 を返す.

$Z, r_{sub} = ecc(X_0,\ X_1,\ X_2,\ X_3,\ i,\ j)$ と表記するときは Z には $Y_i \cdot Y_j$ の値を入れ、Y_1 は r_{sub} に入れることにする。また、r_{sub} を必要としないときは単に $Z = ecc(X_0,\ X_1,\ X_2,\ X_3,\ i,\ j)$ と表記する。

減算は Y_1 の値を利用できるが、加算、乗算も ecc 演算で求めることができる。

$$\begin{aligned} x + y &= ecc(x,\ y,\ 1,\ 0,\ 0,\ 1), \\ x \cdot y &= ecc(x,\ 0,\ y,\ 0,\ 0,\ 1). \end{aligned}$$

よって加減乗算は ecc 演算だけで求められる。ecc 演算を計算するためのハードウェアは次のようになる。



Montgomery 型楕円曲線の加算公式は

$$\begin{aligned} X_{P+Q} &= [(X_P - Z_P)(X_Q + Z_Q) + (X_P + Z_P)(X_Q - Z_Q)]^2, \\ Z_{P+Q} &= x_{P-Q}[(X_P - Z_P)(X_Q + Z_Q) - (X_Q + Z_P)(Z_Q - Z_Q)]^2. \end{aligned}$$

である。ecc 演算を使うと 5 ステップで加算公式を計算できる。

$$\begin{array}{lllll}
1. & r_1 & = & ecc(X_Q,\ Z_Q,\ X_P,\ Z_P,\ 0,\ 1), \\
2. & r_2 & = & ecc(X_P,\ Z_P,\ X_Q,\ Z_Q,\ 0,\ 1), \\
3. & X_{P+Q} & = & ecc(r_1,\ r_2,\ 0,\ 0,\ 0,\ 0), \\
4. & r_1 & = & ecc(0,\ 0,\ r_1,\ r_2,\ 1,\ 1), \\
5. & Z_{P+Q} & = & ecc(r_1,\ 0,\ x_{P-Q},\ 0,\ 0,\ 1).
\end{array}$$

Montgomery 型の 2 倍公式は

$$\begin{aligned}
4X_PZ_P &= (XP + Z_P)^2 - (X_P - Z_P)^2 \\
X_{2P} &= (X_P + Z_P)^2(X_P - Z_P)^2 \\
Z_{2P} &= (4X_PZ_P)((X_P - Z_P)^2 + \tfrac{A+2}{4}(4X_PZ_P))
\end{aligned}$$

である。ecc を使うと 5 ステップで 2 倍公式を計算できる。但し $A' = (A+2)/4$ とする。

$$\begin{array}{lllll}
1. & r_1 & = & ecc(X_P,\ Z_P,\ 0,\ 0,\ 0,\ 0) \\
2. & r_2 & = & ecc(0,\ 0,\ X_P,\ Z_P,\ 1,\ 1) \\
3. & X_{2P} & = & ecc(r_1,\ 0,\ r_2,\ 0,\ 0,\ 1) \\
4. & (r_1,\ r_{sub}) & = & ecc(A',\ 0,\ r_1,\ r_2,\ 0,\ 1) \\
5. & Z_{2P} & = & ecc(r_1,\ r_2,\ r_{sub},\ 0,\ 0,\ 0)
\end{array}$$

$(t_0,\ x(P_0))$ と $(t_1,\ x(P_1))$ から $x(t_0P_0)$ と $x(t_1P_1)$ を求めるのに必要な ecc 演算の回数は、アルゴリズム D と表 2.1 と表 3.1 から次のようになる。ここで $X_0/Z_0 = x(tP_0),\ X_1/Z_1 = x(tP_1)$ とし、表 3.1 では $k = 2,\ m = 325$ とする。

表 3.2: $x(t_0P_0)$ と $x(t_1P_1)$ を求めるのに必要な ecc 演算の回数

	ecc 演算の回数
アルゴリズム D により $(X_0,\ Z_0)$ を求める	1600
アルゴリズム D により $(X_1,\ Z_1)$ を求める	1600
$x(t_0P_0) = X_0/Z_0$ と $x(t_0P_0) = X_0/Z_0$ を計算	329
合計	3529

楕円曲線暗号の処理に必要な演算は ecc だけとなるので、暗号ハードウェアに必要な演算器は ecc 演算器だけである。

第4章　暗号ハードウェアの命令セット

この章では暗号ハードウェアの制御に必要な命令について説明する。

4.1　アドレス付け

$x_0 = x(P_0),\ x_1 = x(P_1)$ から $x(t_0P_0),\ x(t_1P_1)$ を求めることを考える。はじめに $x(t_0P_0) = X_0/Z_0$ を満たす組 $(X_0,\ Z_0)$ を求め、次に $x(t_1P_1) = X_1/Z_1$ を満たす $(X_1,\ Z_1)$ を求め、最後に除算削減法により X_0/Z_0 と X_1/Z_1 を求めるとする。

$X_0,\ Z_0$ を求めるのに途中結果を保存しなければならないが、どのくらいのメモリが必要であるかをアルゴリズム D に従って順番に見ていく。アルゴリズム D の $x(P'),\ x(Q')$ に対して $x(P') = X_P/Z_P,\ x(Q') = X_Q/Z_Q$ とおく。まず $Q' = \mathcal{O}$ をセットするが、無限遠点 $\mathcal{O}$ の扱いは 4.5 節で説明する。次に $P' = P_0$ をセットするが、$X_P = x(P_0), Z_P = 1$ とすれば良い。$t_0 = (t_{0,159}t_{0,158}\cdots t_{0,0})_2$(2進表現) であるとする。アルゴリズム D のループ内の演算は表 4.1 のようになる。

表 4.1: アルゴリズム D のループ内の ecc 演算

$t_{0,159-i} = 0$ の場合			$t_{0,159-i} = 1$ の場合		
r_1	$=$	$ecc(X_Q,\ Z_Q,\ X_P,\ Z_P,\ 0,\ 1),$	r_1	$=$	$ecc(X_Q,\ Z_Q,\ X_P,\ Z_P,\ 0,\ 1),$
r_2	$=$	$ecc(X_P,\ Z_P,\ X_Q,\ Z_Q,\ 0,\ 1),$	r_2	$=$	$ecc(X_P,\ Z_P,\ X_Q,\ Z_Q,\ 0,\ 1),$
X_P	$=$	$ecc(r_1,\ r_2,\ 0,\ 0,\ 0,\ 0),$	X_Q	$=$	$ecc(r_1,\ r_2,\ 0,\ 0,\ 0,\ 0),$
r_1	$=$	$ecc(0,\ 0,\ r_1,\ r_2,\ 1,\ 1),$	r_1	$=$	$ecc(0,\ 0,\ r_1,\ r_2,\ 1,\ 1),$
Z_P	$=$	$ecc(r_1,\ 0,\ X_1,\ 0,\ 0,\ 1),$	Z_Q	$=$	$ecc(r_1,\ 0,\ X_1,\ 0,\ 0,\ 1),$
r_1	$=$	$ecc(X_Q,\ Z_Q,\ 0,\ 0,\ 0,\ 0),$	r_1	$=$	$ecc(X_P,\ Z_P,\ 0,\ 0,\ 0,\ 0),$
r_2	$=$	$ecc(0,\ 0,\ X_Q,\ Z_Q,\ 1,\ 1),$	r_2	$=$	$ecc(0,\ 0,\ X_P,\ Z_P,\ 1,\ 1),$
X_Q	$=$	$ecc(r_1,\ 0,\ r_2,\ 0,\ 0,\ 1),$	X_P	$=$	$ecc(r_1,\ 0,\ r_2,\ 0,\ 0,\ 1),$
$(r_1,\ r_{sub})$	$=$	$ecc(A',\ 0,\ r_1,\ r_2,\ 0,\ 1),$	$(r_1,\ r_{sub})$	$=$	$ecc(A',\ 0,\ r_1,\ r_2,\ 0,\ 1),$
Z_Q	$=$	$ecc(r_1,\ r_2,\ r_{sub},\ 0,\ 0,\ 1),$	Z_P	$=$	$ecc(r_1,\ r_2,\ r_{sub},\ 0,\ 0,\ 1),$

ループが終わったら $X_0 = X_Q,\ Y_0 = Y_Q$ をセットする。$x(t_0P_0) = X_0/Z_0$ となる。$X_0,\ Z_0,\ X_P,\ X_Q,\ Z_P,\ Z_Q,\ r_0,\ r_1,\ r_{sub}$ を記録するためのメモリが必要である。アルゴリズム D の処理が終わったら、$X_0,\ Z_0$ は X_0/Z_0 を計算するために保存しなければならないが、残りはもう必要ない。

$X_1,\ Z_1$ を求めるには $X_1 = x_1,\ Z_1 = 1$ をセットし、ループ内の処理は上と同じで、最後に $X_1 = X_Q,\ Y_1 = Y_Q$ をセットする。ここでは $X_1,\ Z_1,\ X_P,\ X_Q,\ Z_P,\ Z_Q,\ r_0,\ r_1,\ r_{sub}$ のための

メモリを使う。

$X_0,\ X_1,\ Z_0,\ Z_1$ からレベル 2 の除算削減法により $X_0/Z_0,\ X_1/Z_1$ を計算する。つまり

$$X_0/Z_0 = X_0 \cdot Z_1 \cdot (Z_0 Z_1)^{-1},\quad X_1/Z_1 = X_1 \cdot Z_0 \cdot (Z_0 Z_1)^{-1}.$$

を計算する。最初に $x = Z_0 Z_1 = ecc(Z_0,\ 0,\ Z_1,\ 0,\ 0,\ 1)$ を計算する。次にアルゴリズム F により $y = x^{-1} = x^{p-2}$ を計算する。$p-2$ は 162 ビットで、2 進表現が $(t_{161}t_{160}\cdots t_1 t_0)_2$ であるとする。最初に $y = 1$ をセットする。アルゴリズム F のループの中は ecc を使って次のように書ける。

$t_{161-i} = 0$ のとき			$t_{161-i} = 1$ のとき		
x	$=$	$ecc(x,\ 0,\ y,\ 0,\ 0,\ 1)$	y	$=$	$ecc(x,\ 0,\ y,\ 0,\ 0,\ 1)$
y	$=$	$ecc(y,\ 0,\ y,\ 0,\ 0,\ 1)$	x	$=$	$ecc(x,\ 0,\ x,\ 0,\ 0,\ 1)$

次に $Z_0' = Z_1 \cdot y = ecc(Z_1,\ 0,\ y,\ 0,\ 0,\ 1),\ Z_1' = Z_0 \cdot y = ecc(Z_0,\ 0,\ y,\ 0,\ 0,\ 1)$ を計算する。$X_0/Z_0 = X_0 \cdot Z_0',\ X_1/Z_1 = X_1 \cdot Z_1'$ となるから、最後に $X_0/Z_0 = ecc(X_0,\ 0,\ Z_0',\ 0,\ 0,\ 1)$ と $X_1/Z_1 = ecc(X_1,\ 0,\ Z_1',\ 0,\ 0,\ 1)$ を行えば良い。除算過程においては $X_0,\ X_1,\ Z_0,\ Z_1,\ x,\ y,\ Z_0',\ Z_1'$ のためのメモリを使用した。

暗号処理においては定数 0, 1, ベースポイントの x 座標 $x(B)$, 公開鍵の x 座標 $x(P_k)$, 楕円曲線の係数に関する値 $A' = (A+2)/4$ を使うので定数レジスタなどに記憶する。変数のためのメモリと定数レジスタに対して次のようにアドレスを付ける。外部との通信 (入力や出力) のためのアドレスも付ける。

アドレス	変数や定数
0	X_0
1	X_1
2	Z_0
3	Z_1
4	$X_P,\ x$
5	$X_Q,\ y$
6	$Z_P,\ Z_0'$
7	$Z_Q,\ Z_1'$

アドレス	変数や定数
8	r_1
9	r_2
10	0
11	1
12	$x(B),\ x(P_k)$
13	外部
14	A'
15	r_{sub}

なぜ $x(B)$ と $x(P_k)$ が同じアドレスが付けられてているのかは次の節で説明する。

4.2 パイプライン化

この節では本研究で設計する暗号ハードウェアのパイプライン化を考える。パイプライン化されていない場合は 2 組の $(x(P_0),\ t_0)$ と $(x(P_1),\ t_1)$ から 2 つの $x(t_0P_0),\ x(t_1P_1)$ を求めることが一まとまりの処理である。暗号ハードウェアが n 段にパイプライン化されているならば $2n$ 組の $(x(P_i),\ t_i)$ から $2n$ 個の $x(t_iP_i)$ を求めることになる。n 段のパイプラインなのに $2n$ 組のデータを処理するのはレベル 2 の除算削減を用いるからである。n 段のパイプライン化の場合は、処理すべきデータが $2n$ 組集まってから処理を開始することにする。データが集まるまではキューに入れておく。処理された最初のデータが出力されるときに、次の $2n$ 組のデータの最初のものが演算器へ入力される。均等な時間間隔でデータを取り入れる方がレイテンシは小さくなるが、スループッ

トは変わらず、$2n$ 個のデータが集まってから処理する方が制御が簡単になる。データは除算削減法の相手とグループを作るとする。n 段パイプランでは n 個のグループができる。ここまでの過程をまとめると以下のようになる。

1. データが $2n$ 個集まったら処理を開始する。まず n 個の $x(P_{0i})$ を処理する。

グループ		
$n-1$	$x(P_{0n-1})$	$x(P_{1n-1})$
:		
1	$x(P_{01})$	$x(P_{11})$
0	$x(P_{00})$	$x(P_{10})$

↓

処理開始

2. しばらくすると $X_{0i}/Z_{0i} = x(t_{0i}P_{0i})$ を満たす $X_{0i},\ Z_{0i}$ が 1 クロックおきに求まる。
3. 残りの n 個の $x(P_{1i})$ を処理する。

グループ	
$n-1$	$x(P_{1n-1})$
:	
1	$x(P_{11})$
0	$x(P_{10})$

↓

処理開始

4. しばらくすると $X_{1i}/Z_{1i} = x(t_{1i}P_{1i})$ を満たす $X_{1i},\ Z_{1i}$ が 1 クロックおきに求まる。
5. 同時除算法により $x(t_{0i}P_{0i}) = X_{0i}/Z_{0i}$ と $x(t_{1i}P_{1i}) = X_{1i}/Z_{1i}$ を求める。出力の順番は

$$x(t_{00}P_{00}),\ x(t_{01}P_{01}), \cdots, x(t_{0n-1}P_{0n-1}),\ x(t_{10}P_{10}),\ x(t_{11}P_{11}), \cdots, x(t_{1n-1}P_{1n-1}).$$

   であり、1 クロック毎に出力される。
6. 1. の操作から繰り返される。

n 段パイプラインの場合はメモリのアドレスは、グループを指定するグループ番号と前節で説明したアドレスによってアドレスを付けることができる。

メッセージ m の暗号文は $(x(rB),\ x(rP_k) \oplus m)$ となる。r は乱数値である。2 段パイプラインの場合の暗号化処理を考える。まず暗号ハードウェアは $x(B),\ x(P_k)$ を順に取り入れる。$x(B)$ はグループ 0 であり $x(P_k)$ はグループ 1 である。これらが演算器を 1600 回通ると $(X_{00},\ Z_{00})$ と $(X_{01},\ Z_{01})$ が求まる。ここで乱数値 r_0 に対して、$x(r_0B) = X_{00}/Z_{00},\ x(r_0P_k) = X_{01}/Z_{01}$ となる。暗号ハードウェアは再び $x(B),\ x(P_k)$ を順に取り入れる。$x(B)$ はグループ 0 であり $x(P_k)$ はグループ 1 である。それから乱数 r_1 に対して、$x(r_1B) = X_{10}/Z_{10},\ x(r_1P_k) = X_{11}/Z_{11}$ を満たす $(X_{10},\ Z_{10})$ と $(X_{11},\ Z_{11})$ が求まる。ここで $x(r_1B) = X_{10}/Z_{10},\ x(r_1P_k) = X_{11}/Z_{11}$ となる。演算器はグループ 0 の 1 つ目の結果、グループ 1 の 1 つ目の結果、グループ 0 の 2 つ目の結果、グループ 1 の 2 つ目の結果の順、つまり $x(r_0B),\ x(r_0P_k),\ x(r_1B),\ x(r_1P_k)$ の順に計算する。$x(r_iP_k)$ とメッセージ m_i との排他的論理和は演算器とは別の機構 (出力制御器) によって行うとする。

グループ 0 では P_k は使わず B だけを使い、逆にグループ 1 では B は使わず P_k を使っている。暗号化処理において一般にパイプラインの段数が偶数ならば、偶数グループでは P_k は使わず B だけを使い、奇数グループでは B は使わず P_k を使う。これが前節のアドレス付けにおいて B と P_k が一緒にできる理由である。排他的論理和をとる操作もパイプライン段数が偶数の方が制御しやすくなる。

ところでグループ数 > パイプライン段数とすると必要なメモリ容量とレイテンシは増大するが、処理は正しく行われる。よってグループ数とパイプライン段数の関係は

$$\text{グループ数} \geq \text{パイプライン段数}.$$

となる。メモリアクセスに余裕を持たせたいときはグループ数を増やせば良い。本研究ではグループ数=パイプライン段数=14 とした。

4.3 オペランド選択命令

アルゴリズム D でははじめに $Q = \mathcal{O}$ をセットするので、$\mathcal{O}$ との和を求める必要がある。$P = P + \mathcal{O}$ であるから簡単な計算であるが、加算公式は使えないので特別な処理が必要となる。$\mathcal{O}$ の扱いについては次節で説明し、この節では $\mathcal{O}$ を考慮せずに暗号ハードウェアの制御に必要な命令を考える。

$Y = ecc(X_0,\ X_1,\ X_2,\ X_3,\ i,\ j)$ を計算するには $X_0,\ X_1,\ X_2,\ X_3$ のアドレスと $i,\ j$ を指定すれば良い。グループ数が 1 ならば各 $X_k(k = 0, 1, 2, 3)$ と Y は 4.1 節のアドレスにより指定するので 4 ビットずつ、$i,\ j$ を指定するには 1 ビットずつの情報が必要なので、このためには合計で 22 ビットの情報が必要である。本研究ではグループ数が 14 なので、グループ指定のための 4 ビットの情報も必要である。

4.1 節で述べたように、$t = (t_{n-1}t_{n-2}\cdots t_1t_0)$ から $x(tP)$ や x^t を求めるのに t_{n-i} の値によって 2 つの ecc 演算が選択される。よって、1 つの命令に 2 つのデータ指定 (オペランド) が必要となる。グループ指定は 1 つで良い。オペランドを選択する必要が無い場合はオペランド 0 のデータが使用されるとする。

ed(Encrypt - Decrypt) 信号により、暗号化処理と復号化処理を区別することにし、暗号化処理では ed=0 がセットされ、復号化処理では ed=1 がセットされるとする。t の値によるオペランド選択だけでなく、ed によってオペランドが選択される場合がある。例えば処理の最初の時点では、ed=0 ならば $X_0 = x(B)$ または $X_0 = x(P_k)$ がセットされ、ed=1 ならば $X_0 = m$(メッセージ) がセットされる。

このような制御を行うためのオペランド選択命令が必要である。

オペランド選択	00	オペランド 0 を選択
	01	if $(t_{n-i} = 0)$ オペランド 0 を選択 else オペランド 1 を選択
	10	if (ed=0) オペランド 0 を選択 else オペランド 1 を選択

グループ毎の命令はほぼ同一であるので命令をグループ共通なものとして記述できるかもしれない。しかしそのような命令にすると演算器の使用が限定されすぎることになる。本研究ではグループ毎に命令を記述する方針をとった。

4.4 カウンタ命令とスカラー命令

表 3.2 から 1 つのグループを処理するのに ecc 演算が 3529 回必要であるが、更に初期値の設定に ecc 演算を 12 回行う。例えば暗号化処理の時には最初に $X_0 = B,\ Z_0 = 1$ をセットする。この代入は

$$\begin{aligned} X_0 &= ecc(B,\ 0,\ 0,\ 0,\ 0,\ 1), \\ Z_0 &= ecc(1,\ 0,\ 0,\ 0,\ 0,\ 1). \end{aligned}$$

によってなされる。よって命令はグループ数が 14 のときは $14 \times (3529 + 12) = 49574$ ワードとなり、かなり多い。

暗号処理ではアルゴリズム D や F を使うが、ループがあるのでハードウエアにカウンタを設置し、カウンタが桁上りを起こさないときはジャンプさせることにすると命令のワード数を少なくできる。ループの回数はアルゴリズム D と F では異なっているので、カウンタは異なる回数のループを数える必要がある。よってカウンタは任意の値を初期値に設定できる 8 ビットカウンタとする。このようなカウンタを設置することで、命令を $14 \times 63 = 882$ ワードに減らせる。命令にはカウンタの初期値のためのオペランド 2 が必要となる。

命令 ROM を速くアクセスしたいので、次の命令のアドレスを指定する next アドレスを設け、プログラムカウンタは使わないこととする。カウンタの桁上りの有無によってはオペランド 2 が次の命令アドレスとなる。

次のような next 命令とカウンタ命令が必要である。

next	0	(next アドレス) が次の命令アドレス
	1	if (カウンタが桁上り) (next アドレス) が次の命令アドレス else オペランド 2 が次の命令アドレス
カウンタ	00	カウンタを動かさない
	01	カウンタをインクリメント
	10	カウンタにオペランド 2 をセット

アルゴリズム D での t の値は、暗号化処理では乱数値 r であり、復号化では秘密鍵 s である。また、アルゴリズム F では $t = p - 2$ となる。一般に整数 t と楕円曲線の点 P に対して tP を P のスカラー倍と言うので、t をスカラー値と呼ぶことにする。暗号化処理では同じ乱数値 r に対して $x(rB)$ と $x(rP_k)$ を求めなければならないので、発生させた乱数はしばらくの間はスカラーレジスタに保存しなければならない。アルゴリズム D, F のループでは t のあるビットを使うと次の繰り返しでは隣のビットを使う。次の繰り返しでは t を左シフトさせ、常に t の最上位ビットを参照することにする。t を制御する命令をスカラー設定命令ということにする。

スカラー設定	000	何もしない
	001	ed=0 のときスカラー値に乱数の値をセット ed=1 のときスカラー値に s(秘密鍵) をセット
	010	スカラー値にスカラーレジスタの値をセット
	011	スカラー値に $p - 2$ をセット
	100	スカラー値を左シフトする

命令形式は

オペコード	グループ	オペランド 0	オペランド 1	オペランド 2	next アドレス

となる。

4.5 無限遠点 $\mathcal{O}$ について

アルゴリズム D においてはじめに $Q' = \mathcal{O}$ とセットするが、 無限遠点 $\mathcal{O}$ は座標では表されず、加算公式や 2 倍公式を適用することができない。$Q' = \mathcal{O}$ であることを示すための qo ビット (Q' is $\mathcal{O}$) が必要となる。qo=1 ならば $Q' = \mathcal{O}$ を表し、qo=0 ならば $Q' \neq \mathcal{O}$ を表すとする。$Q' = \mathcal{O}$ をセットするときは qo=1 をセットするが、$X_Q,\ Z_Q$ にも $X_Q = 0,\ Z_Q = 0$ とセットすることにする。

今 qo=1 であるとする。アルゴリズム D において $t_{159-i} = 0$ のときは Q' は $\mathcal{O}$ のままであるが、$t_{159-i} = 1$ のときは Q' は P となる。一度 Q' は $\mathcal{O}$ でなくなると、新たな処理のために $Q' = \mathcal{O}$ がセットされるまで $\mathcal{O}$ とは異なる値をとる。qo と t_{159-i} と次の qo は次のような関係にある。

現在の qo	$\overline{t_{159-i}}$	次の qo
0	0	0
0	1	0
1	0	1
1	1	0

よってループの最後のステップで

$$\text{qo=qo \& } \overline{t_{159-i}},$$

を行う。

アルゴリズム D は次のように書き換えられる。

アルゴリズム D'

$P' = P$; qo=1 ;
for($i = 0$, $i < 160$, $i++$){
  if (qo==0 && t_{159-i}==0){
    $x(P') = x(P' + Q')$; (4.1)
    $x(Q') = x(2Q')$; (4.2)
  }
  if (qo==0 && $t_{159-i} == 1$){
    $x(Q') = x(P' + Q')$; (4.1')
    $x(P') = x(2P')$; (4.2')
  }
  if (qo==1 && $t_{159-i} == 0$){
    $x(P') = x(P')$; (4.1")
    $Q' = \mathcal{O}$; (4.2")
  }
  if (qo==1 && $t_{159-i} == 1$){
    $x(Q') = x(P')$; (4.1"')
    $x(P') = x(2P')$; (4.2"')
  }
  qo=qo & $\overline{t_{159-i}}$;
}
$x(tP) = x(Q')$;

4.6 無限遠点 $\mathcal{O}$ を考慮にいれた命令セット

アルゴリズム D' を行えるように命令を追加しなければならない。表 4.1 は qo=1 を考慮しないでアルゴリズム D' の (4.1), (4.1'), (4.2), (4.2') を行うための命令に等しい。

はじめに、(4.1") $x(P') = x(P')$ と (4.1"') $x(Q') = x(P')$ を行えるように命令を追加する。アルゴリズム D' の (4.1), (4.1') のためのオペランド選択とオペランド 0 とオペランド 1 の値は次のようになる。

(4.1), (4.1') を行うためのオペランド選択とオペランドの値

	オペランド選択	オペランド 0							オペランド 1							
0	00	r_1,	X_Q,	Z_Q,	X_P,	Z_P,	0,	1								
1	00	r_2,	X_P,	Z_P,	X_Q,	Z_Q,	0,	1								
2	01	X_P,	r_1,	r_2,	0,	0,	0,	0	X_Q,	r_1,	r_2,	0,	0,	0,	0	
3	00	r_1,	0,	0,	r_1,	r_2,	1,	1								
4	01	Z_P,	r_1,	0,	X_1,	0,	0,	1	Z_Q,	r_1,	0,	X_1,	0,	0,	1	

ここで 0 番目のオペコード 0 の欄には r_1, X_Q, Z_Q, X_P, Z_P, 0, 1 と書かれているが、これはオペコード 0 には

$$r_1 = ecc(X_Q,\ Z_Q,\ X_P,\ Z_P,\ 0,\ 1).$$

を行うためのメモリアドレスが書かれていることを意味している。出力の r_{sub} が必要な場合もあるが r_{sub} は明示的に指定する必要はない。

まず (4.1") $x(P') = x(P')$ を行えるようにする。これは qo=1 のときに 2 番と 4 番においてメモリへの書き込みを禁止すれば良い。$r_1,\ r_2$ は途中結果なので、$X_P,\ Z_P$ が変わらないのならば $r_1,\ r_2$ の値はどんな値になっても構わない。書き込み命令

書き込み	00	演算結果をメモリに書き込む
	01	if (qo=1) 演算結果をメモリに書き込まない else 演算結果をメモリに書き込む

を導入すれば (4.1), (4.1'), (4.1") を行うことができる。

(4.1), (4.1'), (4.1") を行うための各値

	オペランド選択	書き込み	オペランド 0							オペランド 1						
0	00	00	$r_1,$	$X_Q,$	$Z_Q,$	$X_P,$	$Z_P,$	0,	1							
1	00	00	$r_2,$	$X_P,$	$Z_P,$	$X_Q,$	$Z_Q,$	0,	1							
2	01	01	$X_P,$	$r_1,$	$r_2,$	0,	0,	0,	0	$X_Q,$	$r_1,$	$r_2,$	0,	0,	0,	0
3	00	00	$r_1,$	0,	0,	$r_1,$	$r_2,$	1,	1							
4	01	01	$Z_P,$	$r_1,$	0,	$X_1,$	0,	0,	1	$Z_Q,$	$r_1,$	0,	$X_0,$	0,	0,	1

(4.1"') $x(Q') = x(P')$ を行うためには $X_Q = X_P$ と $Z_Q = Z_P$ を行えば良く、ecc 演算では $X_Q = ecc(X_P,\ 0,\ 1,\ 0,\ 0,\ 1)$ と $Z_Q = ecc(Z_P,\ 0,\ 1,\ 0,\ 0,\ 1)$ となる。上の表の 1 番目のオペランド 1 と 3 番目のオペランド 1 を使ってこの計算をできるようにすれば良い。2 番目と 4 番目では ($t_{n-i} = 1$ かつ qo=1) でないとき、つまり $t_{n-i} = 0$ または qo=0 のときオペランド 0 が選択され、そうでないときにオペランド 1 が選択されるようにする。オペランド選択命令に

オペランド選択	11	if ($t_{n-i} = 0$ or qo=0) オペランド 0 を選択 else オペランド 1 を選択

を追加すれば (4.1), (4.1'), (4.1"), (4.1"') を実行するための命令を記述できる。

(4.1), (4.1'), (4.1"),(4.1"') を行うための各値

	オペランド選択	書き込み	オペランド 0							オペランド 1						
0	00	00	$r_1,$	$X_Q,$	$Z_Q,$	$X_P,$	$Z_P,$	0,	1							
1	11	00	$r_2,$	$X_P,$	$Z_P,$	$X_Q,$	$Z_Q,$	0,	1	$X_Q,$	$X_Q,$	0,	0,	0,	0,	1
2	01	01	$X_P,$	$r_1,$	$r_2,$	0,	0,	0,	0	$X_Q,$	$r_1,$	$r_2,$	0,	0,	0,	0
3	11	00	$r_1,$	0,	0,	$r_1,$	$r_2,$	1,	1	$Z_Q,$	$X_Q,$	0,	0,	0,	0,	1
4	01	01	$Z_P,$	$r_1,$	0,	$X_1,$	0,	0,	1	$Z_Q,$	$r_1,$	0,	$X_0,$	0,	0,	1

$Q' = \mathcal{O}$ を考慮しないならば (4.2), (4.2') を行うためのオペランド選択とオペランドは次のようになる。

(4.2),(4.2') を行うためのオペランド選択とオペランドの値

	オペランド選択	オペランド 0							オペランド 1						
0	01	$r_1,$	$X_Q,$	$Z_Q,$	0,	0,	0,	0	$r_1,$	$X_P,$	$Z_P,$	0,	0,	0,	0
1	01	$r_2,$	0,	0,	$X_Q,$	$Z_Q,$	1,	1	$r_2,$	0,	0,	$X_P,$	$Z_P,$	1,	1
2	01	$X_Q,$	$r_1,$	0,	$r_2,$	0,	0,	1	$X_P,$	$r_1,$	0,	$r_2,$	0,	0,	1
3	00	$r_1,$	$A',$	0,	$r_1,$	$r_2,$	0,	1							
4	01	$Z_Q,$	$r_1,$	$r_2,$	$r_{sub},$	0,	0,	1	$Z_P,$	$r_1,$	$r_2,$	$r_{sub},$	0,	0,	1

(4.2"') $x(P') = x(2P')$ は (4.2') と同じだから (4.2") $Q' = \mathcal{O}$ だけを考慮すれば良い。$Q' = \mathcal{O}$ は qo ビットだけの操作を行い、$X_Q,\ Z_Q$ は何もしなくて良い、つまり 2 番と 4 番を書き込み禁止にすれば良い。書き込み命令に

書き込み	10	if ($t_{n-i}=0$ and qo=1) 演算結果をメモリに書き込まない else 演算結果をメモリに書き込む

を追加すると、(4.2),(4.2'),(4.2"),(4.2"') を実行するための命令を記述できる。

(4.2), (4.2'), (4.2"), (4.2"') を行うための各値

	オペランド選択	書き込み	オペランド 0							オペランド 1						
0	01	00	r_1,	X_Q,	Z_Q,	0,	0,	0,	0	r_1,	X_P,	Z_P,	0,	0,	0,	0
1	01	00	r_2,	0,	0,	X_Q,	Z_Q,	1,	1	r_2,	0,	0,	X_P,	Z_P,	1,	1
2	01	10	X_Q,	r_1,	0,	r_2,	0,	0,	1	X_P,	r_1,	0,	r_2,	0,	0,	1
3	00	00	r_1,	A',	0,	r_1,	r_2,	0,	1							
4	01	10	Z_Q,	r_1,	r_2,	r_{sub},	0,	0,	1	Z_P,	r_1,	r_2,	r_{sub},	0,	0,	1

アルゴリズム D の最初に qo を初期化するための qo イニシャル命令と、ループの最後のステップで qo を変更する qo アップデート命令が必要である。

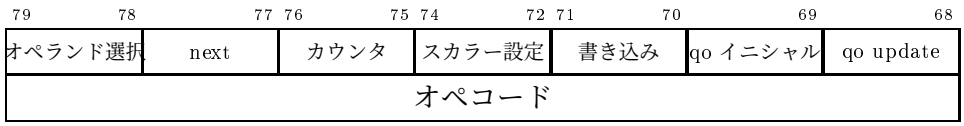
qo イニシャル	0	何もしない
	1	qo=1 をセット
qo アップデート	0	何もしない
	1	qo=qo & $\overline{t_{n-i}}$

ここまでに導入した命令によって $x(P),\ t$ から $x(tP)$ を求めることができる。暗号処理には排他的論理和をとる操作もあるが、これは命令なしで実行できる (5.1 節参照)。以上が本研究で設計する暗号ハードウェアに必要な命令セットである。

各命令の個数と命令を指定するの必要なビット長は

命令	個数	ビット長
オペランド選択	4	2
next	2	1
カウンタ	3	2
スカラー設定	5	3
書き込み	3	2
qo イニシャル	2	1
qo update	2	1

もし命令をエンコードしないとオペコードは 12 ビット必要である。命令は $4\times2\times3\times5\times3\times2\times2=1440$ 通りあるのでエンコードしても 11 ビットが必要である。1 ビットしか違わないので命令はエンコードしないことにする。命令長は 80 ビットで命令形式は次のようになる。



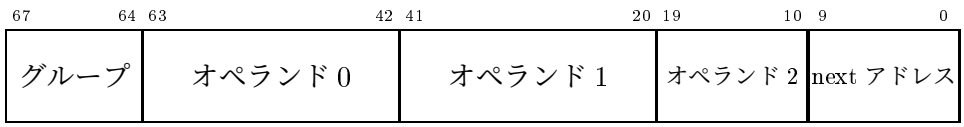


表 4.2: 命令セット一覧

オペランド選択	00	オペランド 0 を選択
	01	if ($t_{n-i} = 0$) オペランド 0 を選択 else オペランド 1 を選択
	10	if (ed=0) オペランド 0 を選択 else オペランド 1 を選択
	11	if ($t_{n-i} = 0$ or qo=0) オペランド 0 を選択 else オペランド 1 を選択
next	0	(next アドレス) が次の命令アドレス
	1	if (カウンタが桁上り) (next アドレス) が次の命令アドレス else オペランド 2 が次の命令アドレス
カウンタ	00	カウンタを動かさない
	01	カウンタをインクリメント
	10	カウンタにオペランド 2 をセット
スカラー設定	000	何もしない
	001	ed=0 のときスカラー値に乱数の値をセット ed=1 のときスカラー値に s(秘密鍵) をセット
	010	スカラー値にスカラーレジスタの値をセット
	011	スカラー値に $p-2$ をセット
	100	スカラー値を左シフトする
書き込み	00	演算結果をメモリに書き込む
	01	if (qo=1) 演算結果をメモリに書き込まない else 演算結果をメモリに書き込む
	10	if ($t_{n-i} = 0$ and qo=1) 演算結果をメモリに書き込まない else 演算結果をメモリに書き込む
qo イニシャル	0	何もしない
	1	qo=1 をセット
qo アップデート	0	何もしない
	1	qo=qo$\cdot\overline{t_{n-i}}$

第5章　暗号ハードウェアの概要

この章では暗号ハードウェア全体の概要について説明するが、その前に演算器以外の暗号ハードウェアに必要なコンポーネントについて説明する。暗号ハードウェアにはデータの入力を調節するためのキューと演算結果を外部へ出力するときに必要な制御を行う出力制御器、乱数発生器が必要である。また命令 ROM からループになっている命令を読み出すときにカウンタを使い、4 章で説明した命令を実行するための制御器も必要である。

5.1　キューと出力制御器

はじめに暗号化処理で必要なキューを考える。ベースポイント B と公開鍵 P_k、乱数値 r によるメッセージ m の暗号文は

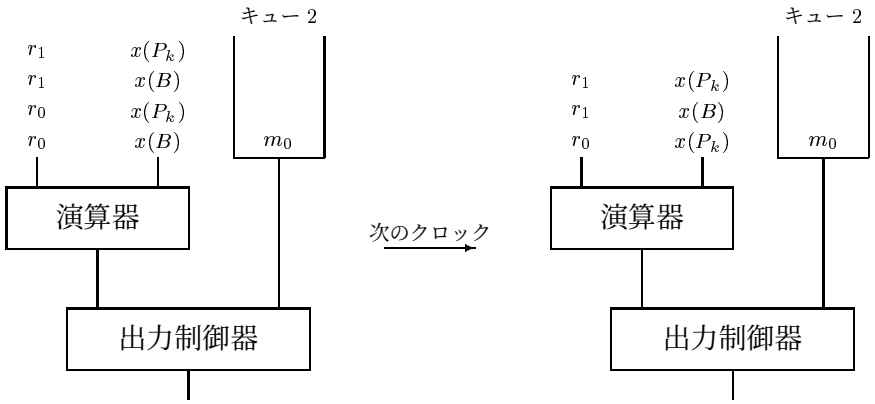
$$(x(C_1),\ c_2) = (x(rB),\ x(rP_k) \oplus m),$$

である。演算器は $x(B),\ x(P_k)$ から $x(rB),\ x(rP_k)$ を計算し、出力制御器が $x(rB)$ と m の排他的論理和をとる。グループ数を n とすると、ある時点で演算器は 1 クロックおきに

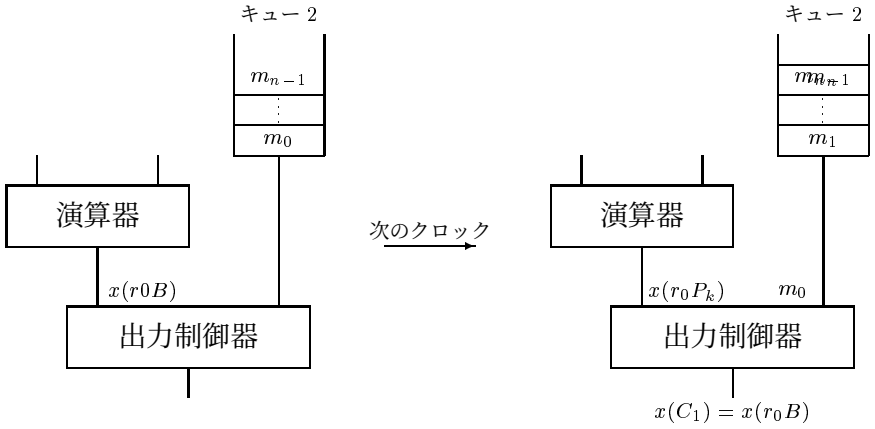
$$x(r_0B),\ x(r_0P_k),\ x(r_1B),\ x(r_1P_k), \cdots x(r_{n-1}B),\ x(r_{n-1}P_k).$$

を出力し、しばらくしてからまた 1 クロックおきに $2n$ 個のデータを出力し、これを繰り返す。なぜデータの出力が n 個ずつでなく $2n$ 個ずつであるのかは、レベル 2 の除算削減法を用いているからである (3.1 節)。このためメッセージ m が一定の割合で暗号ハードウェアに到着するとすると、メッセージをキューに保存しなければならない。このキューをキュー 2 とする。復号化ではもう 1 つキューが必要で、$x(C_1)$ がもう 1 つのキュー (キュー 1) を使い c_2 がキュー 2 を使うので、このようにキューの名前をつけた。キュー 1 は演算器へデータを渡し、キュー 2 は出力制御器へデータを渡す。

暗号化処理では最初のメッセージ m_0 がキュー 2 に入ってから、演算器は 1 クロックおきに $x(B)$ と $x(P_k)$ を交互に取り入れ演算を開始する。



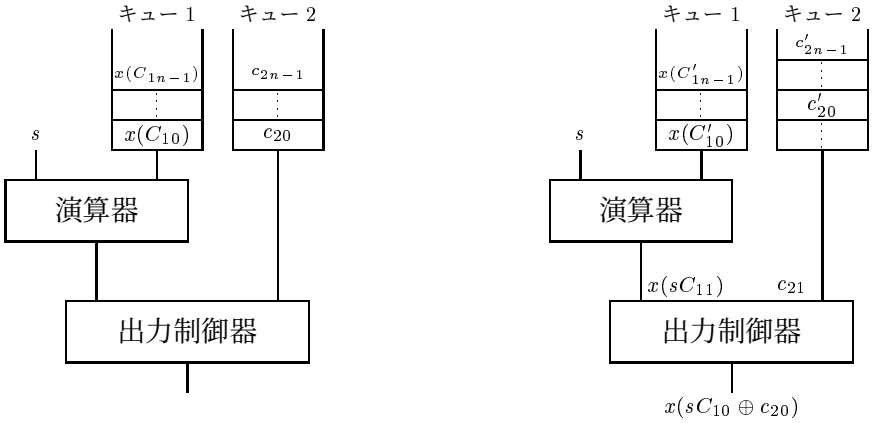
ある時間が経過すると演算器は演算結果の出力を開始するが、このときキュー 2 には n 個のメッセージが入っている (ようにクロックを調節する)。また、このとき演算器は新たに $x(B)$ と $x(P_k)$ の入力も開始する。演算器からの出力データのうち $x(r_iB)$ はそのまま出力し、$x(r_iP_k)$ はキュー 2 にあるメッセージ m_i との排他的論理和をとってから出力する。$2n$ 個 (n 組) の暗号文が 1 クロックおきに出力される。



復号化処理は暗号文 $(x(C_1),\ c_2)$ から秘密鍵 s を使って

$$x(sC_1) \oplus c_2,$$

を計算することであり、これが復号文となる。演算器で s と C_1 から $x(sC_1)$ を求め、出力制御器で演算結果 $x(sC_1)$ と c_2 の排他的論理和をとる。暗号文 $(x(C_1),\ c_2)$ が $2n$ 組集まってから、演算器は $x(C_1)$ を $2n$ を取り入れ演算を開始する。演算結果の出力が始まるときは更に $2n$ 組の暗号文が到着するのでキュー 1 は $2n$ ワード、キュー 2 は $4n$ ワードの容量が必要である。



このように暗号化処理ではキュー 2 のみが使用され、復号化処理ではキュー 1 とキュー 2 が使用されるがデータの入る順番はキュー 1→ キュー 2→ キュー 1→ $\cdots$ と決まっている。よって外部から暗号ハードウェアへデータが到着したときは到着を示す in 信号は必要であるが、そのデータをどちらのキューへ振り分けるかを明示的に示す必要はない。演算器がキュー 1 のデータを使用したら deq1 信号をキュー 1 へ送り、キュー 1 は deq1 を受信したらポインタを進める。同様に出力制御器がキュー 2 のデータを使用したら deq2 を送る。

復号化処理では出力制御器は演算器からの出力データとキュー 2 のデータとの排他的論理和をとる操作を行うが、暗号化処理では $x(rP_k)$ に対してはこの操作を行うが、$x(rB)$ に対しては行わない。グループ数 n が偶数の場合は毎クロック反転するトブル型フリップフロップを使ってデータが $x(rB)$ であるか $x(rP_k)$ であるかを見分けることができる。

5.2 キューが空のときの制御

次にキューが空のときの影響を考える。暗号処理開始時ではキューにはデータが十分にあるが、暗号処理を続けていると通信網の状態によってはキューにデータが無くなることが起り得る。このとき、データがキューに届くまで暗号ハードウェアを止めるか、データ無しで処理を進めなければならない。本研究では後者の方法を採ることにした。

暗号化処理でキュー 2 が空になったとする。キュー 2 が空のときは、キュー 2 は empty 信号を出力制御器へ送る。異なる乱数値 r と r' に対して、$(x(rB),\ x(rP_k) \oplus m)$ と $(x(r'B),\ x(r'P_k) \oplus m)$ はともに m の暗号文であるから、出力制御器は $x(rP_k) \oplus m$ 求めるときに empty を受信したら $x(rP_k)$ を破棄すれば良い。しかし、1 クロック前の $x(rB)$ も破棄しなければならない。

復号化処理で $x(sC_1)$ が求まったときにキュー 2 に c_2 が届いていないとする。そうすると $x(sC_1) \oplus c_2$ を求めることができないが、復号化では $x(sC_1)$ を破棄することはできない。よって c_2 が到着するまで暗号ハードウェアを止めるか、このような事態にならないようにしなければならない。本研究では暗号ハードウェアを途中で止めない立場を取っている。キュー 1 にデータ $x(C_1)$ があってもキュー 2 が空ならば、演算器へ empty 信号と古いデータ (無効データ) を送り、キュー 1 はポインタを動かさない。つまりキュー 1 はキュー 2 が空のときはキュー 1 が空であるようにふるまう。演算器が受信した empty 信号は演算処理が終わるまで保持する。出力制御器が演算器からの empty 信号を受信したときは、$x(sC_1)$ は無効データに対する演算結果であるからこれを破棄し、

キュー 2 からはデータを取り出さない。復号化処理では $x(C_1),\ c_2$ の順で暗号文が入力されるから、キュー 1 が空でキュー 2 が空でないということは起らないので、この場合を考える必要はない。

5.3 乱数発生器

暗号化処理では乱数を発生させなければならない。処理開始時には組 $(x(B), r),\ (x(P_k), r),$ $(x(B), r'),\ (x(P_k), r'),\cdots$ が演算器に 1 クロック毎に入力されるので、乱数発生器は 2 クロックに 1 回の割合で 160 ビット乱数を発生させる機能が必要である。今回は Lehemer 法による乱数を考える。

Lehmer 法は定数 $a,\ c$ と初期値 X_0 から線型合同数列

$$X_n = (aX_{n-1} + c) \bmod n,$$

から順次 X_n を求める方法である。剰余計算を簡単にするために $n = 2^{160}$ とする。この場合実質的に剰余計算が不要である。乱数性を良くするためには a は $a \equiv 5 \pmod 8$ となる $\sqrt{2^{160}}$ に近い値とし c は奇数とする [11]。$aX_{n-1} + c$ を簡単に求めるために $a = 2^{80} + 5$ に固定する。このとき 4 入力加算器だけで $aX_{n-1} + c$ を求めることができる。

暗号処理での乱数は 0 であってはならないので $X_n = 0$ のときは線型合同数列の式をもう一度計算し直すことにするが、$c = a \cdot 0 + c$ であるから、$X_n = 0$ となったら $X_n = c$ とすれば良い。

$a = 2^{80} + 5$ のとき ($a \equiv 1 \pmod 4$ のとき)、Lehmer 法による乱数値の下位 2 ビットは完全な周期を持ってしまい、下位ビットも乱数性が悪い。このため以下のように Y_n を求めて Y_n を乱数として使用することにする。

$$Y_n = X_n[159:80] \ ||\ X_n[79:0] \oplus X_n[159:80].$$

ここで $X_n[k:l]$ は X_n の k ビット目から l ビット目までのビットを表し、$||$ はビットの連接を表している。X_n から Y_n を求める操作は一対一であり $X_n = 0 \Leftrightarrow Y_n = 0$ であるから、X_n に対してゼロチェックを行えば Y_n ではゼロチェックを行う必要はない。

$aX_n + c$ の計算は演算器が 1 クロックで行う計算量にほぼ等しい (6 章参照) から、X_n は 1 クロックで求めることができる。次のクロックで X_n のゼロチェックと Y_n を求める操作を行えば、2 クロックに 1 つの 160 ビット乱数を発生させることができる。

Lehmer 法による乱数は乱数性があまり良くないので、[12] のような物理的な高速乱数発生器を使用するのも考慮にいれた方が良い。

5.4 カウンタ

カウンタはループになっている命令を数えるために使用される。カウンタが数えるループの繰り返し数は 159 と 161 なのでカウンタは 8 ビットカウンタとし、桁上げが起ったとき桁上りを通知する。カウンタの値は必要ないので出力しないことにする。カウンタにはインクリメント機能以外に初期値設定機能が必要である。インクリメントや値の設定はオペコードのカウンタ部によって行われる。

5.5 命令生成器

命令シーケンサは命令アドレスの命令を出力し、オペコードの next 部とカウンタの桁上り信号により、次の命令のアドレスを next アドレスとオペランド 2 から選択する。

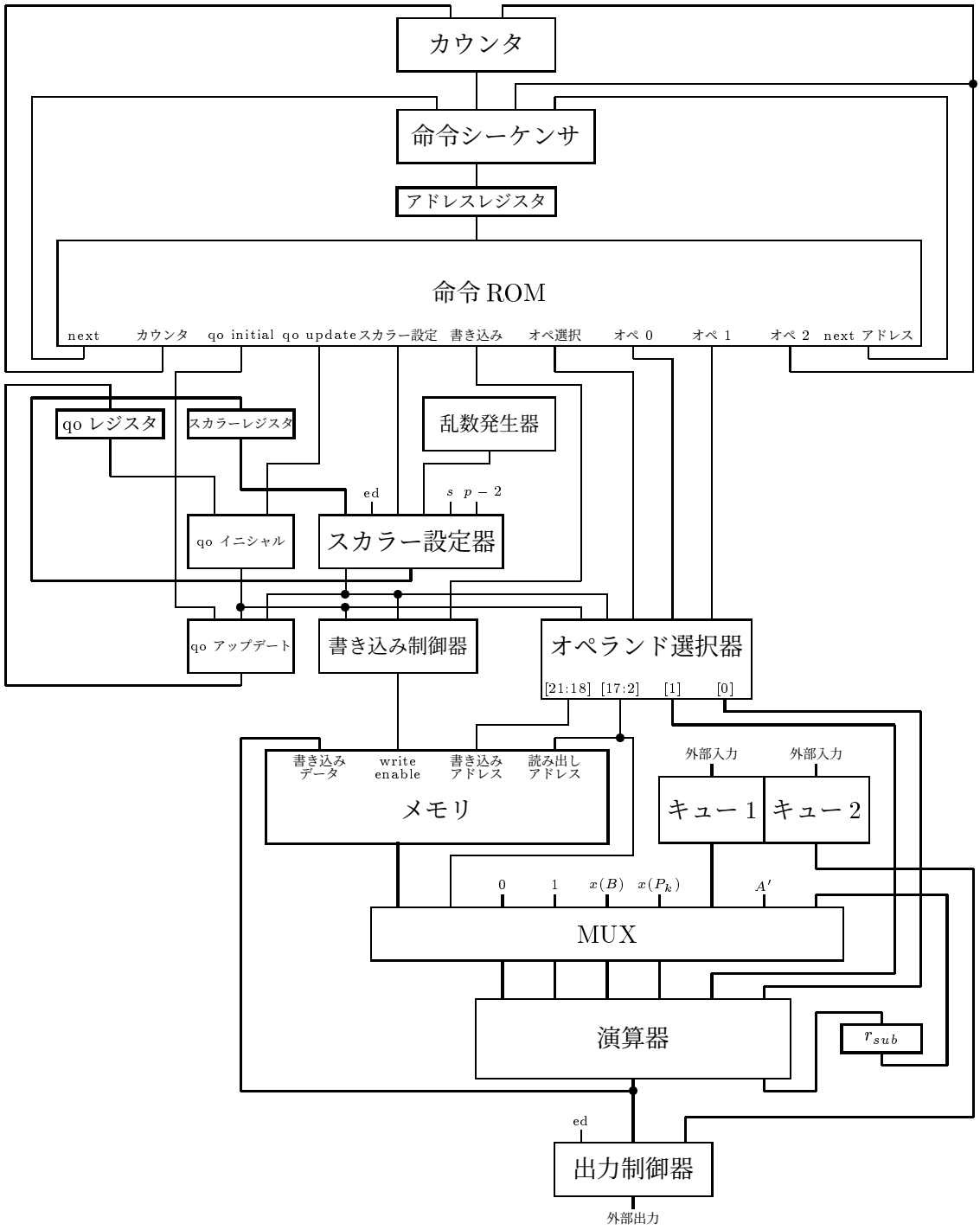
5.6 その他の制御器

表 4.2 の命令を実行するために次のような制御器が必要である。

制御器名	使用するオペコードの部分	使用するその他の信号
アドレス選択器	アドレス選択	スカラー値の最上位ビット, ed, qo
スカラー設定器	スカラー設定	ed
書き込み制御器	書き込み	qo, スカラー値の最上位ビット
qo イニシャル器	qo イニシャル	
qo アップデート器	qo アップデート	スカラー値の最上位ビット

5.7 暗号ハードウェアの概要図

パイプライン化を考慮しないときの暗号ハードウェアの概要図は次のページのようになる。

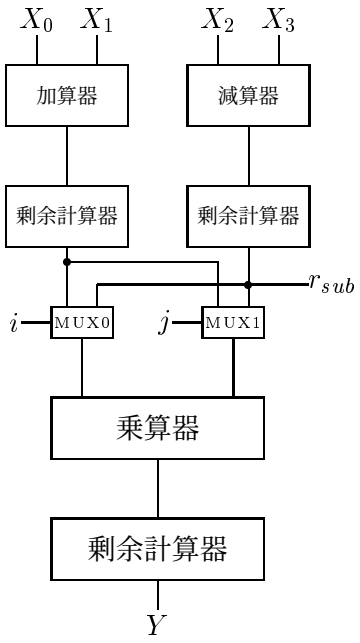


ここで細線は制御信号、太線はデータ信号を表し、細いボックスは論理回路、太線は順序回路やレジスタを表している。

第6章　演算器の設計

この章では暗号ハードウェアのうちの演算器の設計について説明する。暗号ハードウェアの処理速度を向上させるため演算器は 162 ビットを扱うとする。暗号処理で必要な演算は ecc 演算だけであるから、暗号ハードウェアに必要な演算器は ecc 演算器だけである。ecc 演算器の概要は次のようになる。

図 6.1: 演算器の概要



ここで加算器と乗算器は正整数のためのものとなる。減算器は 6.4 節で説明するように特殊な計算を行う。剰余計算器はアルゴリズム A と B を行うが、k を適切な値に固定することで回路規模を小さくできる。この章で扱う加算器と乗算器については [13] を参照。

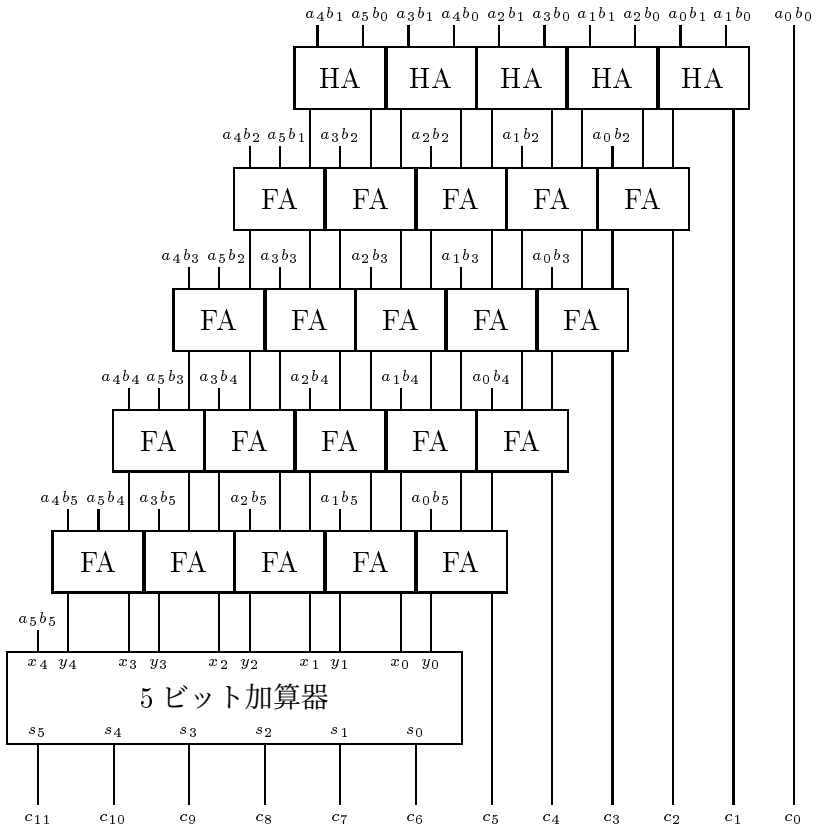
6.1　Wallace tree 乗算器

はじめに正整数乗算器について説明する。一般に n ビット乗算器は n ビットの部分積を n 個生成し、それらを足し合わせる。部分積を生成する部分は n^2 個の AND ゲートから構成され、この部分を MPS(multiple select) 部と呼ぶ。例として 6 ビットの乗算 $a_5a_4a_3a_2a_1a_0 \times b_5b_4b_3b_2b_1b_0 =$

$c_{11}c_{10}\cdots c_1c_0$ を考える。MPS 部が各 a_ib_j を生成する。

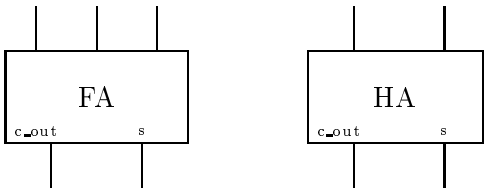
							a_0b_5	a_0b_4	a_0b_3	a_0b_2	a_0b_1	a_0b_0
						a_1b_5	a_1b_4	a_1b_3	a_1b_2	a_1b_1	a_1b_0	
					a_2b_5	a_2b_4	a_2b_3	a_2b_2	a_2b_1	a_2b_0		
				a_3b_5	a_3b_4	a_3b_3	a_3b_2	a_3b_1	a_3b_0			
			a_4b_5	a_4b_4	a_4b_3	a_4b_2	a_4b_1	a_4b_0				
$+$		a_5b_5	a_5b_4	a_5b_3	a_5b_2	a_5b_1	a_5b_0					
	c_{11}	c_{10}	c_9	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0

アレイ型乗算器は次のように構成される。全加算器 (FA) と半加算器 (HA) がアレイ状に並んでいる部分を CSA(carry-save adder) 部といい、加算器の部分を CPA(carry-propagate adder) 部という。



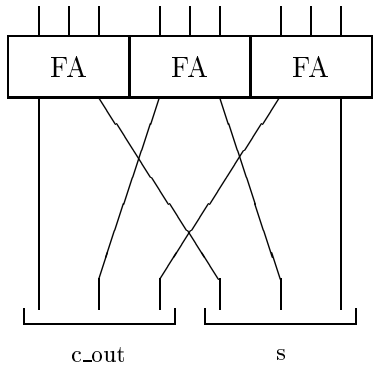
AND ゲート、OR ゲート、XOR ゲートを使うと全加算器の論理段数は 3、半加算器の論理段数は 1 になるので、k ビットアレイ型の CSA 部の論理段数は $1+(k-1)\cdot 3 = 3k-2$ になり、CPA 部は $k-1$ ビット加算器になる。

Wallace tree 乗算器を説明する前に論理回路 fa$n$ を定義する。Wallace tree 乗算器の CSA 部は fa$n$ を使って構成できる。全加算器と半加算器の出力を c_out と s とする。c_out は桁上りである。



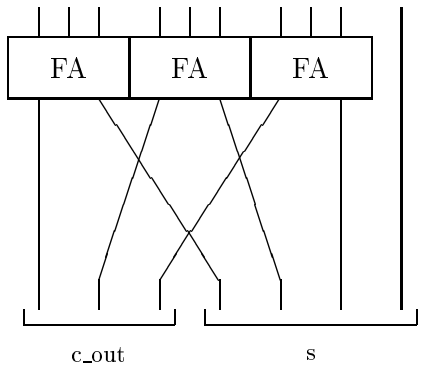
$n \equiv 0 \pmod 3$ の時は全加算器を $n/3$ 個並べ、各全加算器の c_out 同士と s 同士をまとめたものが fa$n$ となる。c_out の集まりを fa$n$ の c_out、s の集まりを fa$n$ の s と呼ぶことにする。fa$n$ の c_out、s はそれぞれ $n/3$ ビットである。例えば fa9 は次のようになる。

fa9



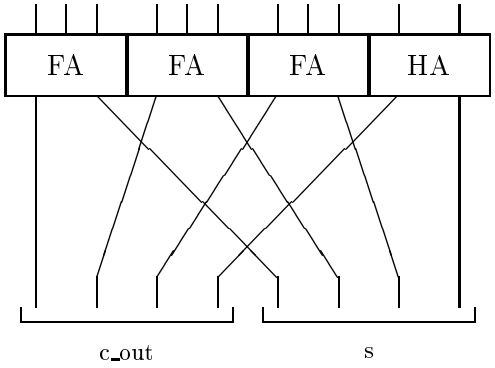
$n \equiv 1 \pmod 3$ の時は全加算器を $\lfloor n/3 \rfloor$ 個並べその隣にスルー線を並べたものが fa$n$ となる。fa$n$ の c_out は各全加算器の c_out をまとめたもの、fa$n$ の s は各全加算器の s とスルー線をまとめたものとなる。fa$n$ の c_out は $\lfloor n/3 \rfloor$ ビット、fa$n$ の s は $\lfloor n/3 \rfloor + 1$ ビットとなる。ここで $\lfloor \cdot \rfloor$ は切り捨てを意味する。fa10 は次のようになる。

fa10



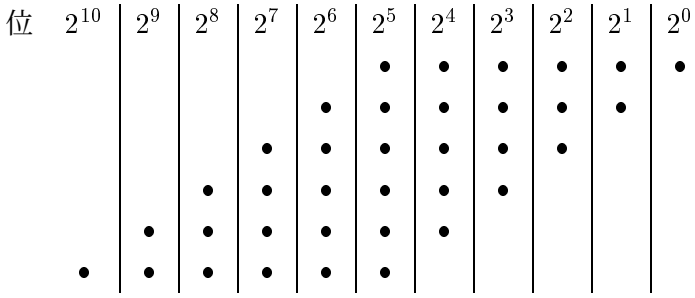
$n \equiv 2 \pmod 3$ の時は全加算器を $\lfloor n/3 \rfloor$ 個と半加算器を 1 つ並べたものが fa$n$ となる。fa$n$ の c_out は各全加算器と半加算器の c_out をまとめたもの、fa$n$ の s は各全加算器と半加算器の s をまとめたものとなる。fa$n$ の c_out と s はともに $\lfloor n/3 \rfloor + 1$ ビットとなる。fa11 は次のようになる。

fa11



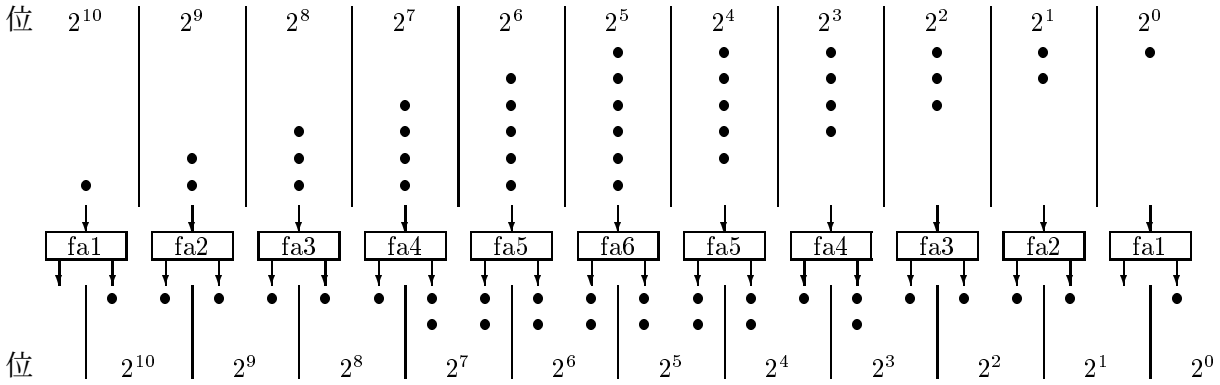
fa$n$ の論理段数とは全加算器の論理段数は同じ 3 である。

Wallace tree 乗算器は CSA 部を fa$n$ で構成した乗算器である。また 6 ビットの乗算を例にする。• をビット値とする。MPS 部の結果は次のようになる。

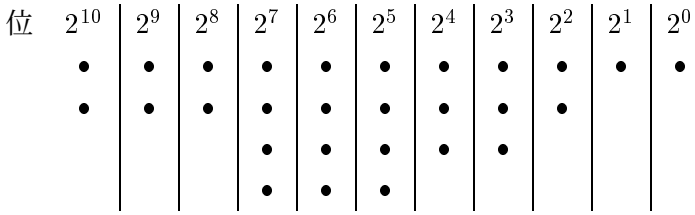


2^m の位のビットが n 個あるならばビットを fa$n$ へ入力する。そうすると fa$n$ の出力のうち c_out が 2^{m+1} の位のビット、s が 2^m の位のビットとなる。

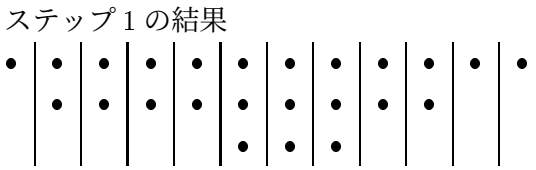
ステップ 0

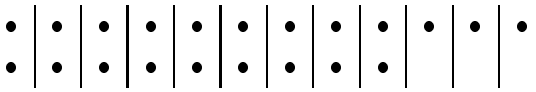


ここまでの操作をステップ 0 とする。ステップ 0 の結果により次のようなビットを得た。



ステップ 0 と同様にステップ 0 の結果の各位のビットを fa$n$ に入力する。この操作をステップ 1 とする。同様にステップ 2 も行う。





ステップ 2 が終わると各位のビットが 2 つ以下になっているので、それらのビットを CPA 部 (9 ビット加算器) へ入力すれば 6 ビット乗算の結果が得られる。

ステップが進むにつれて、位毎のビット数の最大値は約 2/3 になるので、k ビット乗算で必要なステップ数は約 $\log_{2/3}(2/k)$ となる。fa$n$ の論理段数は 3 なので k ビット Wallace tree 乗算器の CSA 部の論理段数は約 $3\cdot\log_{2/3}(2/k)$ 段となる。k が大きくなると Wallace tree 乗算器の CSA 部の論理段数はアレイ型と較べてかなり小さくなる。例えば $k=162$ すると、162 ビット Wallace tree 乗算器の CSA 部のステップ数は 12 なので論理段数は 36 であり、アレイ型では $3\cdot 162-2=484$ 段となる。(全加算器はあるピンでは論理段数が 1 や 2 であることもあるので、実際には Wallace tree 乗算器の CSA 部はより論理段数が少なくなることもある。)

6 ビット乗算器の例からも分かるように、ステップ l の結果では最下位から下位第 $l+1$ 位までのビット数が 1 である。よって k ビット Wallace tree 乗算器の CPA 部は ($2k-$ ステップ数) ビット加算器となる。k が大きい場合、アレイ型と較べて CPA 部のビット数は 2 倍近くになるが、CPA 部を桁上げ先見加算器にすれば論理段数はそれ程大きくはならない。

k ビット Wallace tree 乗算器の CSA 部において、ステップ m、2^i の位で使用する fa$n$ の n を $n_{m,i}$ と書くとする。$m=0$ のときは

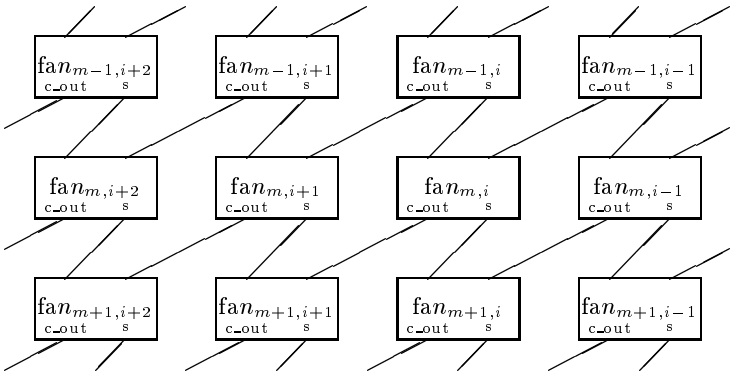
$$\begin{aligned} i<k &\Rightarrow n_{0,i}=i+1,\\ i\geq k &\Rightarrow n_{0,i}=2k-1-i, \end{aligned}$$

となる。$m\geq 1$ では

$$n_{m,i}=\text{fa } n_{m-1,i-1}\text{ の c\_out のビット数}+\text{fa } n_{m-1,i}\text{ の s のビット数.}$$

で求められる。よって Wallace tree 乗算器で使用する fa$n$ はすぐに知ることができる。

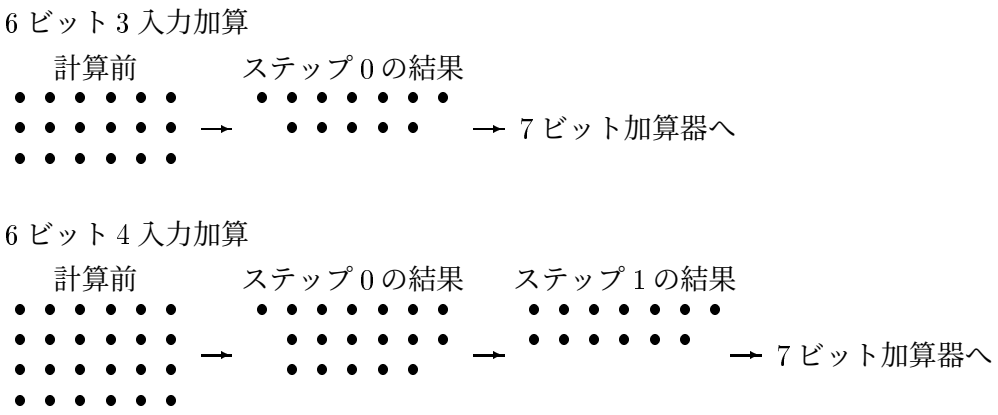
Wallace tree 乗算器の CSA 部を fa$n$ で表すと次のようになり、規則的な配列となっている。次の図では 1 本の線でも複数の配線を表している場合がある。



次に fa2 について考える。fa2 は半加算器と同じものであるが、半加算器は 2 ビットの入力に対して 2 ビットを出力するのでビット数の削減を行わない。よって場合によっては fa2 を 2 ビットのスルー線にすることができる。下位では CPA のビット数に影響を与えるかもしれないので、本研究では上位の fa2 をスルー線に置き換えて乗算器を設計した。このようにすることでわずかだがゲート数を減らすことができる。

6.2 多入力加算器

乗算器の CSA 部と CPA 部は多入力加算器と見なすことができる。一般の多入力加算器も Wallace tree と 2 入力加算器から構成できる。例えば、3 入力 6 ビット加算と 4 入力 6 ビット加算をドット図で書くと次のようになる。



乗算器と同様に各ステップでは位毎に fa$n$ へ通す操作を行うので、1 ステップの論理段数は全加算器と同じ 3 段である。乗算器と同様に多入力加算器に対しても fa$n$ の部分を CSA 部、2 入力加算器の部分を CPA 部と呼ぶことにする。多入力加算器をこのように構成する方が、加算器を複数用いるより処理時間とハードウェア量の両方で効率的になる。剰余計算器と減算器の設計には多入力加算器を使用する。

6.3 F_p 乗算器

ecc 演算器のうち 162 ビット乗算器とその結果の剰余をとるための剰余計算器をあわせて F_p 乗算器と呼ぶことにする。162 ビット乗算器は Wallace tree 乗算器とし、剰余計算器はアルゴリズム A を使用する。

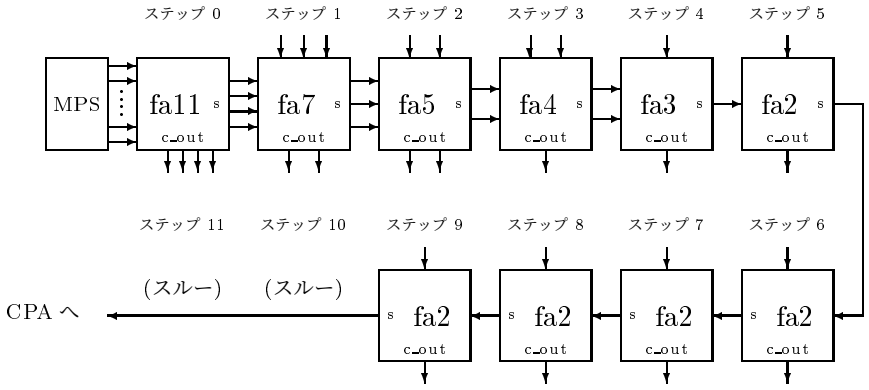
本研究ではハードウェア記述言語は SFL、合成ツールは PARTHENON を使用し、セルライブラリは 0.35μm テクノロジの NEC cmos9 を使用し、ゲート数、面積、最大遅延の評価を行った [14]。遅延と面積の評価は配線を無視している。面積は最小インバータの面積が 18L×14L(基本長) として概算してある。

162 ビット Wallace tree 乗算器は MPS–CSA 部と CPA 部に分けて設計した。CPA は 312 ビット桁上げ先見加算器である。MPS–CPA の PARTHENON による評価は次のようになる。

	ゲート数	段数	面積 (mm^2)	最大遅延 (ps)
162 ビット乗算器の MPS–CSA 部	210853	33	511.4	1.91×10^4

MPS–CSA 部は規模が大きく一度には合成できないので、各位毎にサブモジュール分けして設計した。2^i の位の MPS–CSA 部のサブモジュール名を MPSCSAi とする。例えば MPSCSA10 は次のようになる。

MPSCSA10



各 MPSCSAi が使用する fa$n$ の n の値と PARTHENON による評価をまとめた表を付録 A.1 に掲載する。

乗算器の CPA 部の PARTHENON による評価は次のようになる。

	ゲート数	段数	面積 (mm^2)	段数
162 ビット乗算器の CPA 部	5052	29	0.245	1.19×10^4

MPS–CSA 部の各ステップでの出力ビットを調べると次のようになる。ここで 1 ワードは 162 ビットとしている。

	出力ビット数	出力ワード数
MPS	26244	162
CSA ステップ 0	17610	108.7
CSA ステップ 1	11849	73.1
CSA ステップ 2	8008	49.4
CSA ステップ 3	5447	33.6
CSA ステップ 4	3744	23.1
CSA ステップ 5	2607	16.1
CSA ステップ 6	1857	11.5
CSA ステップ 7	1360	8.4
CSA ステップ 8	1016	6.3
CSA ステップ 9	817	5.0
CSA ステップ 10	681	4.2
CSA ステップ 11	636	3.9

今回は MPS–CSA 部はパイプライン分けしないが、もし MPS–CSA 部を細かくパイプライン分けするとパイプライン・ラッチのハードウェア量が急増する。もし CSA ステップ 2 とステップ 3 をパイプライン分けするならば 49.4 ワード (8008 ビット) のパイプライン・ラッチが必要となる。

次に F_p 乗算の剰余計算器を設計する。剰余計算器はアルゴリズム A

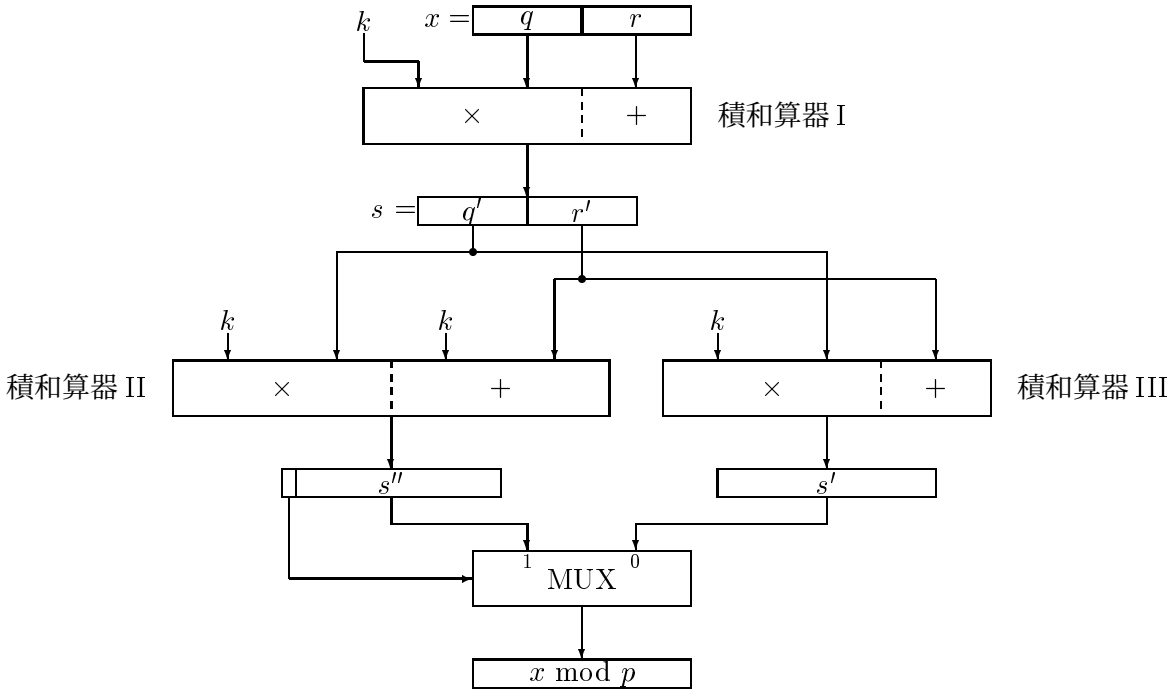
1. $x = q \cdot 2^n + r,\ q \geq 0,\ 0 \leq r < 2^n$ とする.
2. $s = qk + r$ を計算.
3. $s = q' \cdot 2^n + r',\ q' > 0,\ 0 \leq r' < 2^n$ とする.
4. $s' = q'k + r',\ s'' = (q' + 1)k + r'$ を計算する.
5. $s'' \geq 2^n$ ならば $(x \bmod m) = s'' - 2^n$,
   $s'' < 2^n$ ならば $(x \bmod m) = s$'.

を行うハードウェアである。剰余計算器は

$$
\begin{aligned}
s &= qk + r, \\
s' &= q'k + r', \\
s'' &= (q' + 1)k + r'
\end{aligned}
$$

を計算する積和算器とマルチプレクサ mux_m から構成される。積和算器とは乗算と加算を一度に行う演算器のことであり、これも Wallace tree で設計できる。s を求めるための積和算器を I、s' を求めるための積和算器を II、s'' を求めるための積和算器を III とする。剰余計算器は次のようになる。

剰余計算器 II



ここで k を 2 進表現で表したときの 1 の個数が少なくなるように固定すると各積和算器の規模は小さくなる。本研究では剰余計算器の規模を小さくするために $k = 101$ に固定した。つまり素数 p を $p = 2^{162} - 101$ に固定する。101 の 2 進表現は $(1100101)_2$ であるから

$$\begin{aligned} qk + r &= q \cdot 101 + r \\ &= q + q \cdot 2^2 + q \cdot 2^5 + q \cdot 2^6 + r, \end{aligned}$$

となり、積和算器 I は 5 入力加算器となる。このとき積和算器 I の CSA 部のステップ数は 3 で、CPA は 169 ビット加算器となる。積和算器 II は 5 入力加算器であり、CSA 部のステップ数は 3、CPA 部は 162 ビット加算器となる。積和算器 III も 5 入力加算器である。

アルゴリズム A の条件である $0 < k < 2^{81}$ の範囲で k を動かせるような剰余計算器では積和算器 I は 82 入力加算器となり、CSA 部のステップ数は 10、CPA 部は 252 ビット加算器となり、回路規模は 162 ビット乗算器の約半分になる。同様に積和算器 II, III も $k = 101$ に固定する場合と較べて規模が大きくなる。k を固定することは体 F_p を固定することになるが、暗号で使用する $E(F_p)$ は楕円曲線の係数を変えることで変更することができ、この点が楕円曲線暗号の長所の 1 つである。また、楕円曲線暗号のソフトウェア処理でも高速処理のためには体を固定している [9]。

剰余計算器とマルチプレクサ mux_m の PARTHENON による評価は次のようになる。

	ゲート数	段数	面積 (mm^2)	最大遅延
積和算器 I の CSA 部	2918	8	0.160	4.30×10^3
積和算器 I の CPA 部	2623	25	0.127	1.11×10^4
積和算器 II の CSA 部	168	7	0.009	4.30×10^3
積和算器 II の CPA 部	2503	25	0.122	1.10×10^4
積和算器 III の CSA 部	178	7	0.010	4.30×10^3
積和算器 III の CPA 部	2503	25	0.122	1.10×10^4
mux_m	493	4	0.017	1.30×10^3

6.4 F_p 加算器と F_p 減算器の設計

F_p 加算は普通の加算の結果からアルゴリズム B によって p の剰余をとる。アルゴリズム B を行うためには

$$s = qk + r, \quad s' = (q+1)k + r,$$

を計算するが、ここで q は 1 ビットなので $s = k + r$ または $s = r$ となり、s を求めるには加算器と入力のためのセレクタだけで良い。s を求めるコンポーネントを加算器 1 とする。s' も $s' = k + r$ または $s' = 2k + r$ で得られるので加算器と入力のためのセレクタだけで良い。s' を求めるコンポーネントを加算器 2 と呼ぶことにする。F_p 加算器の剰余計算の部分は加算器 1 と加算器 2 の他にマルチプレクサ mux_a が必要である。

	ゲート数	段数	面積 (mm^2)	最大遅延
162 ビット加算器	2503	25	0.122	1.10×10^4
加算器 1	1108	25	0.067	1.06×10^4
加算器 2	1118	25	0.067	1.07×10^4
mux_a	493	4	0.017	1.30×10^3

F_p 減算について説明する。$\overline{x}$ を x のビットを反転させた値とする。n ビットで考えるとすると $\overline{x} = 2^n - 1 - x$ となる。$p = 2^n - k$ とし、$a,\ b$ は n ビットであるとする。

$$\begin{aligned} a - b &\equiv a - b + 2p, && \pmod{p} \\ &= a - b + 2(2^n - k), \\ &= a + (2^n - 1 - b) + (2^n - 1 - (2k-2)), \\ &= a + \overline{b} + \overline{(2k-2)}. \\ (a-b) \bmod p &= (a + \overline{b} + \overline{(2k-2)}) \bmod p. \end{aligned}$$

よってビットの反転と加算と剰余計算から F_p 減算を計算できる。k を固定するならば $\overline{2k-2}$ は定数となる。

F_p 減算 a − b の方法

1. b の NOT をとる。
2. 3 入力加算器を使って $a + \overline{b} + \overline{(2k-2)}$ を計算する。(この結果は 164 ビット。)
3. 2 の結果を剰余計算器に通す。

減算器は上の 1. と 2. を行うコンポーネントとする。減算器は NOT を取る部分、3 入力加算器の CSA 部と CPA 部に分けることができ、CSA 部のステップ数は 1、CPA 部は 162 ビット加算器となる。3. の剰余計算は F_p 加算と同様にアルゴリズム B を使うので、剰余計算器は $s = qk + r$ を計算するための積和算器 a と $s' = (q+1)k + r$ を計算するための積和算器 b 及びマルチプレクサ mux_s から構成される。

F_p 加算器の各コンポーネントの PARTHENON による評価は次のようになる。

	ゲート数	段数	面積 (mm^2)	最大遅延 (ps)
減算器	3510	33	0.167	1.33×10^4
積和算器 a の CSA 部	23	3	0.001	1.30×10^3
積和算器 a の CPA 部	2503	25	0.122	1.10×10^4
積和算器 b の CSA 部	46	3	0.003	1.70×10^3
積和算器 b の CPA 部	2503	25	0.122	1.10×10^4
mux_m	493	4	0.017	1.30×10^3

6.5 ecc 演算器の概要とパイプライン分け

これまでに演算器のうち F_p 乗算器、F_p 加算器、F_p 減算器の設計について説明した。演算器に必要なコンポーネントはこれらの他には 2 個のマルチプレクサ mux0 と mux1 だけである (図 6.1)。mux0 と mux1 は mux_a などと同一の構造である。

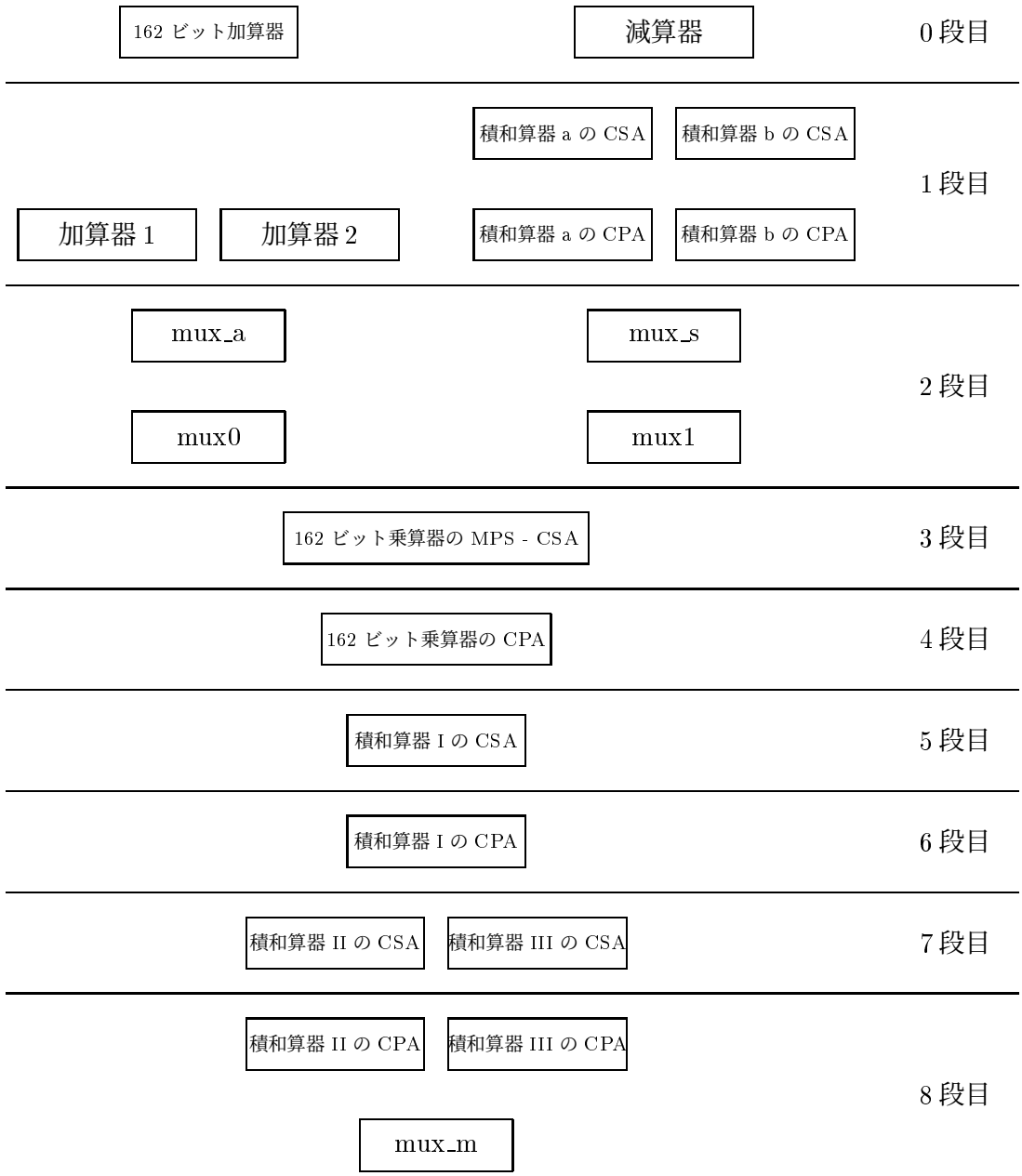
	ゲート数	段数	面積 (mm^2)	最大遅延 (ps)
mux_m	493	4	0.017	1.30×10^3
mux_m	493	4	0.017	1.30×10^3

演算器全体ではゲート数は 242577、面責は $14.49mm^2$(cmos9)、$1.183mm^2$ となる。

本研究では次のページのように演算器を 9 つにパイプライン分けした。また各ステージでの段数と最大遅延時間は以下のようになる。

ステージ	ゲート段数	cmos9 での最大遅延 (ps)
0	33	1.33×10^4
1	29	1.27×10^4
2	8	2.60×10^3
3	33	1.91×10^4
4	29	1.19×10^4
5	8	4.30×10^3
6	25	1.11×10^4
7	7	4.30×10^3
8	29	1.23×10^4

この表は直列にコンポーネントが並んでいるステージの最大遅延は各コンポーネントの最大遅延を足し合わせた値となっている。


162 ビット加算器
減算器
0 段目
積和算器 a の CSA
積和算器 b の CSA
1 段目
加算器 1
加算器 2
積和算器 a の CPA
積和算器 b の CPA
mux_a
mux_s
2 段目
mux0
mux1
162 ビット乗算器の MPS - CSA
3 段目
162 ビット乗算器の CPA
4 段目
積和算器 I の CSA
5 段目
積和算器 I の CPA
6 段目
積和算器 II の CSA
積和算器 III の CSA
7 段目
積和算器 II の CPA
積和算器 III の CPA
8 段目
mux_m

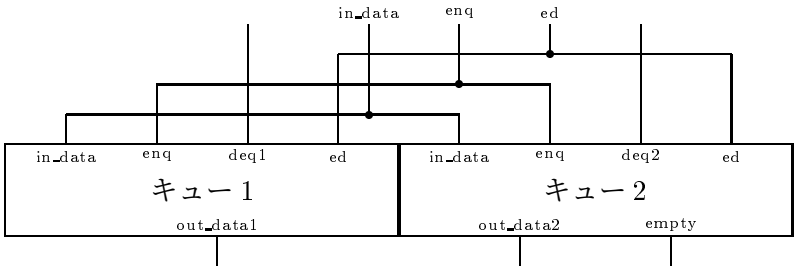
第7章　キュー、乱数発生器、制御器の設計

この章では暗号ハードウェアの演算器以外のコンポーネントの設計を行う。この章で扱うコンポーネントの動作などの詳細は 5 章を参照。

7.1　キューの設計

5.1 節で説明したように暗号ハードウェアにはキュー 1 とキュー 2 の 2 つのキューが必要で、キュー 1 のデータは演算器で使用され、キュー 2 のデータは出力制御器で使用される。キュー 1 は 28 ワード、キュー 2 は 56 ワードの記憶容量が必要である。キューの入力線と出力線、概要図は次のようになる。

	信号名	ビット数	
入力線	in_data	162	キューへの入力データ. キュー 1 とキュー 2 へ自動的に振り分けられる.
	enq	1	キューへ入力されるデータがあるときに enq=1 となる.
	deq1	1	キュー 1 のデータを使用するときに deq1=1 とする.
	deq2	1	キュー 2 のデータを使用するときに deq2=1 とする.
	ed	1	暗号化処理のときは ed=0, 復号化処理のときは ed=1.
出力線	out_data1	162	キュー 1 の出力データ.
	out_data2	162	キュー 2 の出力データ.
	empty	1	キュー 2 が空のとき empty=1 となる.



キューの PARTHENON による評価は次のようになる。

	ゲート数	レジスタ数	面積 (mm^2)
キュー	33876	13870	8.06

7.2 乱数発生器の設計

乱数発生器は 2 クロックに 1 回の割合で 160 ビット乱数を発生させなければならない。今回は線型合同数列

$$X_{n+1} = (aX_n + c) \bmod 2^{162}.$$

による Lehmer の方法による乱数発生器を構成する。ここで c は奇数でなければならず、a は $a \equiv 5 \bmod 8$ となる 2^{81} に近い値が望ましい [11]。剰余計算器の構成と同じ理由、つまり $aX_n + c$ を計算するためのハードウェア量が少なくするために $a = 2^{81} + 5$ とした。そうすると $aX_n + c$ は 4 入力加算器で計算できる。初期値 X_0 と c は外部入力とする。但し c の最下位ビットは常に 1 になるようにする。

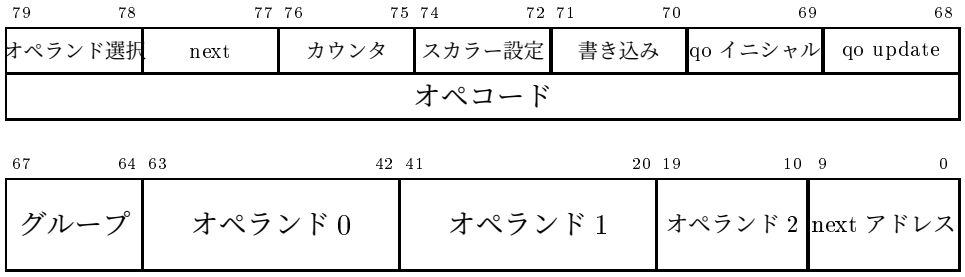
5.2 節で説明したように下位ビットの周期性を打ち消すために X_n から Y_n を生成し、Y_n を乱数として使用する。2 クロックに 1 回乱数を発生されば良いので、1 クロック目に X_n を計算し、2 クロック目に X_n のゼロ判定を行い $X_n = 0$ ならば $X_n = c$ とおき、X_n から Y_n を求める。

乱数発生器の PARTHENON による評価は次のようになる。

	ゲート数	レジスタ数	面積 (mm^2)
乱数発生器	6182	483	0.536

7.3 制御器の設計

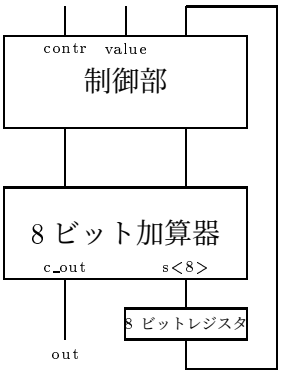
この節ではオペコードや ed 信号、スカラー値などから暗号ハードウェアの制御を行うためのコンポーネントを設計する。各制御器のふるまいは表 4.2 や 5 章を参照。参考のために命令形式をもう一度記しておく。



7.3.1 カウンタの設計

カウンタは 2 ビットの制御信号 contr と 8 ビットのデータ value を入力し、1 ビットの桁上り信号 out を出力する。8 ビットレジスタ reg にカウンタの値を記録する。contr の値により次にように動作する。

contr の値	動作	
00	reg:=reg	(何もしない)
01	reg:=value	(value の値をセット)
10	reg:=reg+1	(インクリメント)

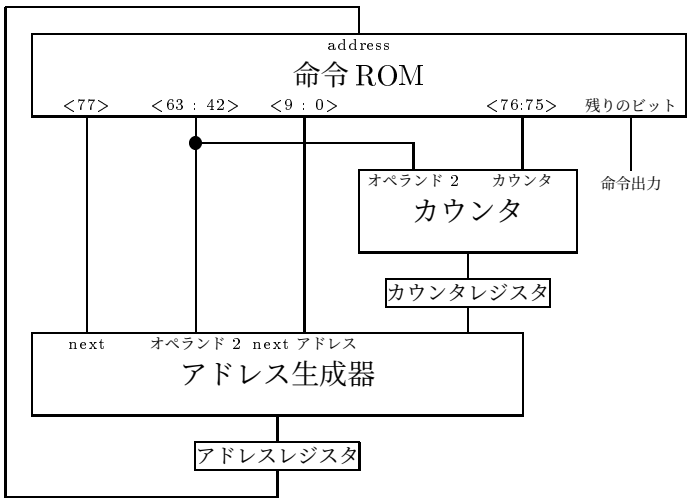


カウンタの PARTHENON による評価は次のようになる。

	ゲート数	レジスタ数	面積 (mm^2)
カウンタ	104	11	0.010

7.3.2 命令シーケンサの設計

命令生成器は命令 ROM、カウンタとこれらの出力から次のアドレスを決定するアドレス生成器から構成され、オペコードの next 値とカウンタからの out 信号を使用する。命令 ROM は 80 ビット ×882 ワードの ROM である。コードについては付録 A.2 に記載する。アドレス生成器はオペコードの next 値とカウンタ値から、次のアドレスとして next アドレス値とオペランド 2 の値から適切な方を選択する。

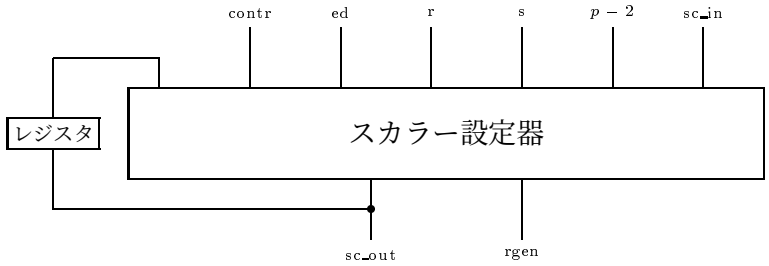


命令 ROM とカウンタを除いた命令シーケンサの PARTHENON による評価は次のようになる。

	ゲート数	レジスタ数	面積 (mm^2)
命令シーケンサ	177	23	0.017

7.3.3 スカラー設定器の設計

スカラー設定器はオペコードのスカラー値と暗号化か復号化かを示す ed から、演算器で使用するスカラー値を乱数値 r(暗号化で使用)、秘密鍵 s(復号化で使用)、$p-2$(除算で使用)、前のスカラー値 (sc_in) の左シフト (演算の途中で使用)、1 クロック前と同じ値 (暗号化で使用) から選択する。乱数値を使った場合は新しい乱数値を発生させるために rgen も出力する。r と s は 160 ビット、$p-2$ と sc_in は 162 ビットであるが、160 ビットのデータが選択された場合は最下位 2 ビットに 00 を連接して出力する。

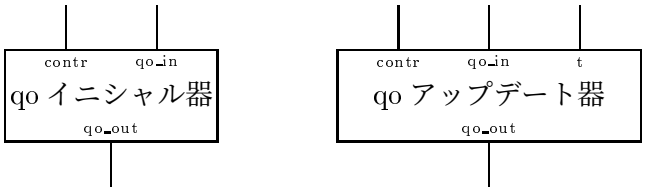


スカラー設定器の PARTHENON による評価は次のようになる。

	ゲート数	レジスタ数	面積 (mm^2)	面積
スカラー設定器	1358	162	0.090	0.007

7.3.4 qo に関する制御器

$E(F_p)$ の元である無限遠点 $\mathcal{O}$ は座標では表すことができない。アルゴリズム D' において Q' が $\mathcal{O}$ であることを示す qo ビットが必要となる。qo イニシャル器はアルゴリズム D' のループの開始時に qo=1 と初期化するためのコンポーネントで、qo アップデート器はループの最後のステップで qo を変更するコンポーネントである。

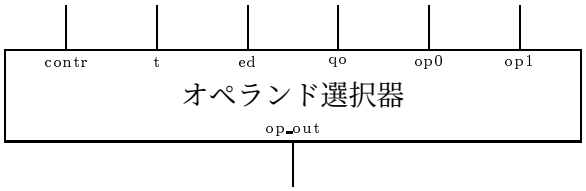


qo に関する制御器の PARTHENON による評価は次のようになる。

	ゲート数	面積 (mm^2)
qo イニシャル器	3	0.000
qo アップデート器	3	0.000

7.3.5 オペランド選択器

オペランド選択器はオペコードのオペランド選択、スカラー値の最上位ビット t、暗号化処理か復号化処理かを示す ed、qo ビットによりオペランド 0 とオペランド 1 から使用するオペランドを選ぶ。

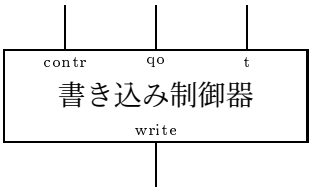


オペランド選択器の PARTHENON による評価は次のようになる。

	ゲート数	面積 (mm^2)
オペランド選択器	86	0.006

7.3.6 書き込み制御器の設計

選択されたオペランドのうち 4 ビットが演算結果を書き込むメモリ (あるいは外部へ出力するか) を指定しているが、場合によってはメモリへの書き込みを禁止する場合がある。書き込み制御器はオペコードの書き込み値と qo ビットとスカラー値の最上位ビット t から write 信号を生成する。write=1 のときメモリへの書き込みが行われ、write=0 のときは書き込みは行われない。



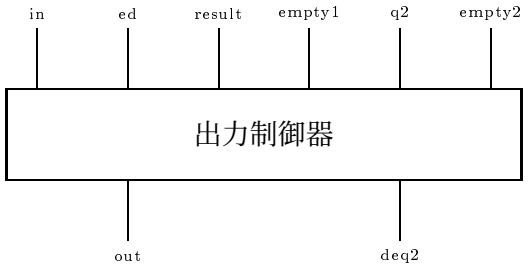
書き込み制御器の PARTHENON による評価は次のようになる。

	ゲート数	面積 (mm^2)
書き込み制御	86	0.006

7.3.7 出力制御器の設計

出力制御器は外部へ出力すべき演算結果をそのまま出力するか、あるいはキュー 2 のデータとの排他的論理和を取ってから出力するかを決定する。出力制御器の各信号の意味と概略図は次のようになる。

	信号名	ビット数	信号の意味
入力信号	in	1	演算器から出力すべきデータが来たことを示す.
	ed	1	暗号化処理か復号化処理かを示す.
	result	162	演算器からのデータ.
	empty1	1	演算開始時のキュー 2 からの empty 信号.
	q2	162	キュー 2 からのデータ.
	empty2	1	現在のキュー 2 からの empty 信号.
出力信号	out	162	出力データ.
	deq2	1	キュー 2 のデータを使用したことを示す.



出力制御器の PARTHENON による評価は次のようになる。

	ゲート数	面積 (mm^2)
出力制御器	1552	0.172

7.4 暗号ハードウェアのゲート数と面積

これまでで暗号ハードウェアに必要な全てのコンポーネントを設計した。今回は次のように暗号ハードウェアを 14 段にパイプライン分けする。演算の箇所は 9 つに分かれている (6.5 節)。

段数		使用するオペコード
0	命令フェッチ	カウンタ,next
1	スカラー設定 qo セット 0	スカラー設定 qo セットの 0 ビット目
2	オペランド選択	オペランド選択
3	データの取り込み	
4〜12	演算	
13	write back qo セット 1 出力制御	書き込み qo セットの 1 ビット目

オペランド選択にはスカラー値が必要なので、オペランド選択はスカラー設定の後でなければならない。

暗号ハードウェア全体での配線を無視した面積とゲート数は次のようになる。ここでメモリや命令 ROM は評価に入れていない。

			ゲート数	レジスタ数	面積 (mm^2)
演算器	F_p 加算器	162 ビット加算器	2503	0	0.122
		加算器 1	1108	0	0.067
		加算器 2	1118	0	0.067
		mux_a	493	0	0.017
	F_p 減算器	減算器	3510	0	0.167
		積和算器 a の CSA	23	0	0.001
		積和算器 a の CPA	2503	0	0.122
		積和算器 b の CSA	46	0	0.003
		積和算器 b の CPA	2503	0	0.122
		mux_s	493	0	0.017
	F_p 乗算器	162 ビット乗算器の MPS-CSA	210853	0	12.90
		162 ビット乗算器の CPA	5052	0	0.245
		積和算器 I の CSA	2918	0	0.160
		積和算器 I の CPA	2623	0	0.127
		積和算器 II の CSA	168	0	0.009
		積和算器 II の CPA	2503	0	0.122
		積和算器 III の CSA	178	0	0.010
		積和算器 III の CPA	2503	0	0.122
		mux_m	493	0	0.017
	mux_0		493	0	0.017
	mux_1		493	0	0.017
	合計		242577	0	14.49
制御器	キュー		33876	13870	8.06
	乱数発生器		6182	483	0.536
	カウンタ		104	11	0.010
	命令シーケンサ		177	23	0.017
	スカラー設定器		1358	162	0,090
	qo イニシャル器		3	0	0.000
	qo アップデート器		3	0	0.000
	オペランド選択器		86	0	0.006
	書き込み制御器		86	0	0.006
	出力制御器		1552	0	0.172
	合計		43427	14549	8.900
その他			18618	8272	4.963
合計			286004	22821	28.35

ここでその他とは全コンポーネントを結合するために必要となるゲートやパイプライン・ラッチ、定数レジスタなどである。1 レジスタを 1 ゲートで換算すると暗号ハードウェアのゲート数は 308825 であり、1 レジスタを 4 ゲートで換算すると 377288 ゲートとなる。

命令 ROM は 80 ビット長で、ワード数はグループ数 n に対して $63n$ ワードとなる。パイプライン段数が 14 の時は 882 ワードとなり、$80 \times 882 = 70560$ ビットの ROM が必要である。。

1 つのメモリは 162 ビット $\times$ 10 ワードである。$ecc(X_0,\ X_1,\ X_2,\ X_3,\ i, j)$ を計算するにあたり、$X_0,\ X_1,\ X_2,\ X_3$ をメモリから読み出すので、一度に 4 つのデータを読み出す必要がある。このため 4 つのメモリを用意し、書き込みは 4 つのメモリが同じ内容になるように 4 つともに行い、X_i は i 番目のメモリから読み出すとする。この 4 つのメモリをメモリセットと呼ぶこととすると、暗号ハードウェア全体ではメモリセットが n 組必要である。$n = 14$ では必要なメモリは $162 \times 10 \times 14 \times 4 = 90720$ ビットである。

次に配線を考慮した暗号ハードウェアの面積を概算する。ゲートとレジスタの配線はそれらの 2.3 倍の面積を必要とする。ROM は 1 ビットあたり $1.44\mu m^2$、RAM は 1 ビットあたり $10\mu m^2$ の面積を必要とし、RAM と ROM はデコーダのために 20%の面積が必要とする。すると暗号ハードウェアの面積は次のように概算される。

ゲートとレジスタの面積	28.35mm^2
ゲートとレジスタの配線の面積	$28.35 \times 2.3 = 65.21\text{mm}^2$
ROM の面積	$7560 \times 1.44 \times 1.2 \times 10^{-6} = 0.12\text{mm}^2$
RAM の面積	$90720 \times 10 \times 1.2 \times 10^{-6} = 1.09\text{mm}^2$
合計	94.77mm^2

第8章　性能評価と今後の課題

8.1　性能評価

2つの整数 t, t' と $E(F_p)$ の2つの点の x 座標 $x(P),\ x(P')$ から $x(tP),\ x(t'P')$ を求めるのに3541回の ecc 演算が必要である。暗号ハードウェアが n 段にパイプライン化されているとすると、1回の ecc 演算には n クロックが必要となる。n 段パイプラインでは n 個の並列処理がなされているので、$x(tP)$ を求めることのスループットは

$$\frac{2n}{3541n} \times (\text{クロック周波数}) = 5.648 \times 10^{-4} \times (\text{クロック周波数})\ (\text{個}/s)$$

となる。パイプラインの段数を大きくすると、クロック周波数を上げることができスループットが向上する。

本研究で設計した暗号ハードウェアでは162ビットのメッセージを暗号化するのに $x(tP)$ を求めることを2回するので、暗号化処理のスループットは

$$162 \times 5.648 \times 10^{-4} \times (\text{クロック周波})/2 = 0.0457 \times (\text{クロック周波数})\ \text{bps} \tag{8.1}$$

となる。復号化は162ビットのメッセージを復元するのに $x(tP)$ を求めることを1回行うので、スループットは暗号化処理の2倍になる。

$$162 \times 5.6673 \times 10^{-4} \times (\text{クロック周波}) = 0.0915 \times (\text{クロック周波数})\ \text{bps} \tag{8.2}$$

演算器でタイムクリティカルな箇所は乗算器のMPS–CSA部であり、セルライブラリNEC/cmos9によるPARTHENONの評価では最大遅延が19.1nsである。暗号ハードウェア全体でのタイムクリティカルな箇所も乗算器のMPS–CSA部と仮定する。配線遅延とラッチのアクセス時間の合計を10nsとするとクロック周波数は33MHzとなり、暗号化のスループットは1.51Mbps、復号化のスループットは3.02Mbpsとなる。

現在最も一般的な公開鍵暗号であるRSAの処理速度を調べてみる。[15]により実装がなされたRSAチップとソフトウェアによるRSAの処理速度は次のようになる。

RSAチップ	2.4ms
ソフトウェア処理	10〜50ms

RSAチップはソフトウェアの4〜20倍の速さである。このRSAチップは処理速度が0.43Mbpsであり、ゲート数は約280kゲートである。また、このRSAチップは0.25μmのCMOSで実装されており、動作周波数は80MHzである。このRSAチップは1024ビットのRSAを扱っているが、これは本研究の暗号ハードウェアと安全性の強度は同じである。本研究で設計した暗号ハードウェアのゲート数は300Kを越えているが、このRSAチップより処理速度は3.5〜7倍速い。

[9] では 0.571μs でソフトウェアにより有限体が 168 ビット長の楕円曲線のスカラー倍を求めており、暗号化処理は 0.147Mbps、復号化処理は 0.294Mbps で行うことができる。なお [9] では Pentium II(450MHz) を使用している。現在では動作周波数が 3GHz の CPU が存在するので、周波数に比例してスループットが向上するならば、楕円曲線暗号のソフトウェア処理でのスループットは暗号化処理で 0.989Mbps、復号化で 1.96Mbps となる。0.35μm テクノロジ、動作周波数 33MHz の本研究の暗号ハードウェアがまだ約 3 倍高速である。

0.1μm テクノロジを想定した仮想セルを作成し、処理時間の最も多い乗算器の MPS–CSA 部での最大遅延を計測したところ、その値は 5.22ns となった。よって暗号ハードウェアの動作周波数を 100MHz にはできると思われる。このときのスループットは暗号化処理で 4.57MHz、復号化処理で 9.14MHz となる。暗号ハードウェアの面積も 0.35μm の cmos9 では 100mm^2(1cm 角) 近くあるが、0.1μm テクノロジでは約 10mm^2(3mm 角) となる。パイプラインの段数を上げることで更に性能を上げることができる。

8.2 今後の課題

本研究は暗号ハードウェアの設計までしか行えず、実装まではできなかった。暗号ハードウェアでは乗算器の CSA 部の実装が最も困難であると思われる。6.1 節で説明した fa$n$ は、部分的な斜め配線が可能ならばきれいな回路となる。もし各 fa$n$ が面積や回路規模が同じならば、乗算器の CSA 部を規則正しく配置することが可能であるが、実際には大きな差があるので fa$n$ の配置法を考えなければならない。このようなことが可能な CAD があれば大規模乗算器の実装が容易に行えるようになる。

暗号ハードウェアには、回路規模と消費電力を無視した高速性を追求する場合と、逆に IC チップの搭載のためのハードウェアのように高速性よりも回路規模や消費電力に重点におく場合とがある。本研究は前者の立場を取っているがまだ十分には高速でない。パイプラインの段数を大きくすることで暗号処理のスループットを上げることはできるが、乗算器の CSA 部を細かく分けるとパイプライン・ラッチの量が急激に増大する。乗算器の CSA 部はウェーブパイプライン化するのが望ましいかもしれない。

本研究では有限体 F_p の p を固定したが、安全性の強度を変えるには p のビット長を変えなければならない。このためには RNS(Residue Number System) という剰余計算法を使うのが良いと思われる。RNS は大きな数を法とする剰余計算を、いくつかの小さな数を法とする剰余計算により求める。この小さな数を法とする剰余計算に本研究で紹介したアルゴリズム A が使用できる。元々は RNS は RSA の処理で必要となる剰余計算のために考えられた。よって RNS に対応するハードウェアでは任意の安全性をもつ楕円曲線暗号と RSA の両方の処理が可能になるかもしれない。

謝辞

本研究を行うにあたり日頃から熱心にかつ適切に御指導頂きました日比野靖教授には心より感謝しております。また、有益な助言を頂いた田中清史助教授、宮崎純助手、日比野研究室の皆様に感謝致します。本研究に関する事項について相談して頂いた相羽明氏にも深く感謝しております。

参考文献

[1] S. Singh, (青木薫 訳), 「暗号解読」, 新潮社, 2001.

[2] N. Koblitz, (林彬 訳), 「暗号の代数理論」, シュプリンガー・フェアラーク東京, 1999.

[3] J. Silverman., *The Arithmetic of Elliptic Curves*, Springer–Verlag, 1986.

[4] P. L. Montgomery., Speeding the Pollard and Elliptic Curve Methods of Factorization, *Mathematics of computation*, Vol.48, No.177, pp243–264, 1987.

[5] 暗号の数理, 「数理科学」, サイエンス社, No.9, 2000.

[6] Certicom Research, SEC 1: Elliptic Curve Cryptography, Certicom Corp., 2000.

[7] A. J. Menezes, T. Okamoto, S. A. Vanstone., The Implemention of Elliptic Curve Logarithms to Logarithms in a Field, *IEEE Transactions on Information Theory*, 39(1993), pp1639–1646.

[8] A. Miyaji, T. Ono, H. Cohen., Efficient Elliptic Curve Exponentiation, *Advanced in Cryptology–ASIACRYPT'98*, LNCS 1514, pp282–290, Springer, 1998.

[9] 青木和麻呂, 楕円曲線暗号はどこまで速くなるか?, http://www.math.is.tohoku.ac.jp/ taya/sendaiNT/2000/aoki_m.pdf .

[10] K. Okeya, H. Kurumatani, K. Sakurai., Elliptic Curves with the Montgomery–Form and Their Cryptographic Applications, *Public Key Cryptography*, LNCS 1751, pp238–257, Springer, 1987.

[11] D. E. Knuth, (渋谷政昭 訳), 準数値算法/乱数, サイエンス社, 1981.

[12] 東芝プレスリリース (2002 年 12 月 9 日付) http://www.toshiba.co.jp/about/press/2002_12/pr_j0501.htm .

[13] A. R. Omondi, *Computer Arithmetic Systems*, Prentice Hall, 1994.

[14] PARTHENON, http://www.kecl.ntt.co.jp/parthenon/html/intro.htm .

[15] 新保淳, 野崎華恵, 川村信一, 高速 RSA 暗号 LSI, 東芝レビュー, Vol. 56, No. 7, pp10–13, 2001.

付 録A

A.1 MPSCSAi が使用する fa$n$ について

6.3 節で説明した MPSCSAi が各ステップで使用する fa$n$ の n は以下の表のようになる。$n = -2$ は 2 ビットスルー線を表している。

\ ステップ	0	1	2	3	4	5	6	7	8	9	10	11
MPSCSA 0	1	1	1	1	1	1	1	1	1	1	1	1
MPSCSA 1	2	1	1	1	1	1	1	1	1	1	1	1
MPSCSA 2	3	2	1	1	1	1	1	1	1	1	1	1
MPSCSA 3	4	3	2	1	1	1	1	1	1	1	1	1
MPSCSA 4	5	3	2	2	1	1	1	1	1	1	1	1
MPSCSA 5	6	4	3	2	2	1	1	1	1	1	1	1
MPSCSA 6	7	5	3	2	2	2	1	1	1	1	1	1
MPSCSA 7	8	5	4	3	2	2	2	1	1	1	1	1
MPSCSA 8	9	6	4	3	2	2	2	2	1	1	1	1
MPSCSA 9	10	7	5	3	2	2	2	2	2	1	1	1
MPSCSA 10	11	7	5	4	3	2	2	2	2	2	1	1
MPSCSA 11	12	8	5	4	3	2	2	2	2	2	2	1
MPSCSA 12	13	9	6	4	3	2	2	2	2	2	2	2
MPSCSA 13	14	9	6	4	3	2	2	2	2	2	2	2
MPSCSA 14	15	10	7	5	3	2	2	2	2	2	2	2
MPSCSA 15	16	11	7	5	4	3	2	2	2	2	2	2
MPSCSA 16	17	11	8	5	4	3	2	2	2	2	2	2
MPSCSA 17	18	12	8	6	4	3	2	2	2	2	2	2
MPSCSA 18	19	13	9	6	4	3	2	2	2	2	2	2
MPSCSA 19	20	13	9	6	4	3	2	2	2	2	2	2
MPSCSA 20	21	14	9	6	4	3	2	2	2	2	2	2
MPSCSA 21	22	15	10	7	5	3	2	2	2	2	2	2
MPSCSA 22	23	15	10	7	5	4	3	2	2	2	2	2
MPSCSA 23	24	16	11	7	5	4	3	2	2	2	2	2
MPSCSA 24	25	17	11	8	5	4	3	2	2	2	2	2
MPSCSA 25	26	17	12	8	6	4	3	2	2	2	2	2
MPSCSA 26	27	18	12	8	6	4	3	2	2	2	2	2
MPSCSA 27	28	19	13	9	6	4	3	2	2	2	2	2
MPSCSA 28	29	19	13	9	6	4	3	2	2	2	2	2
MPSCSA 29	30	20	13	9	6	4	3	2	2	2	2	2
MPSCSA 30	31	21	14	9	6	4	3	2	2	2	2	2
MPSCSA 31	32	21	14	10	7	5	3	2	2	2	2	2
MPSCSA 32	33	22	15	10	7	5	4	3	2	2	2	2
MPSCSA 33	34	23	15	10	7	5	4	3	2	2	2	2
MPSCSA 34	35	23	16	11	7	5	4	3	2	2	2	2
MPSCSA 35	36	24	16	11	8	5	4	3	2	2	2	2
MPSCSA 36	37	25	17	11	8	6	4	3	2	2	2	2
MPSCSA 37	38	25	17	12	8	6	4	3	2	2	2	2
MPSCSA 38	39	26	17	12	8	6	4	3	2	2	2	2
MPSCSA 39	40	27	18	12	8	6	4	3	2	2	2	2
MPSCSA 40	41	27	18	12	8	6	4	3	2	2	2	2
MPSCSA 41	42	28	19	13	9	6	4	3	2	2	2	2
MPSCSA 42	43	29	19	13	9	6	4	3	2	2	2	2
MPSCSA 43	44	29	20	13	9	6	4	3	2	2	2	2
MPSCSA 44	45	30	20	14	9	6	4	3	2	2	2	2
MPSCSA 45	46	31	21	14	10	7	5	3	2	2	2	2
MPSCSA 46	47	31	21	14	10	7	5	4	3	2	2	2
MPSCSA 47	48	32	21	14	10	7	5	4	3	2	2	2
MPSCSA 48	49	33	22	15	10	7	5	4	3	2	2	2
MPSCSA 49	50	33	22	15	10	7	5	4	3	2	2	2
MPSCSA 50	51	34	23	15	10	7	5	4	3	2	2	2
MPSCSA 51	52	35	23	16	11	7	5	4	3	2	2	2
MPSCSA 52	53	35	24	16	11	8	5	4	3	2	2	2
MPSCSA 53	54	36	24	16	11	8	6	4	3	2	2	2
MPSCSA 54	55	37	25	17	11	8	6	4	3	2	2	2
MPSCSA 55	56	37	25	17	12	8	6	4	3	2	2	2
MPSCSA 56	57	38	25	17	12	8	6	4	3	2	2	2
MPSCSA 57	58	39	26	17	12	8	6	4	3	2	2	2
MPSCSA 58	59	39	26	18	12	8	6	4	3	2	2	2
MPSCSA 59	60	40	27	18	12	8	6	4	3	2	2	2
MPSCSA 60	61	41	27	18	12	8	6	4	3	2	2	2
MPSCSA 61	62	41	28	19	13	9	6	4	3	2	2	2

\ ステップ	0	1	2	3	4	5	6	7	8	9	10	11
MPSCSA 62	63	42	28	19	13	9	6	4	3	2	2	2
MPSCSA 63	64	43	29	19	13	9	6	4	3	2	2	2
MPSCSA 64	65	43	29	20	13	9	6	4	3	2	2	2
MPSCSA 65	66	44	29	20	14	9	6	4	3	2	2	2
MPSCSA 66	67	45	30	20	14	10	7	5	3	2	2	2
MPSCSA 67	68	45	30	20	14	10	7	5	4	3	2	2
MPSCSA 68	69	46	31	21	14	10	7	5	4	3	2	2
MPSCSA 69	70	47	31	21	14	10	7	5	4	3	2	2
MPSCSA 70	71	47	32	21	14	10	7	5	4	3	2	2
MPSCSA 71	72	48	32	22	15	10	7	5	4	3	2	2
MPSCSA 72	73	49	33	22	15	10	7	5	4	3	2	2
MPSCSA 73	74	49	33	22	15	10	7	5	4	3	2	2
MPSCSA 74	75	50	33	22	15	10	7	5	4	3	2	2
MPSCSA 75	76	51	34	23	15	10	7	5	4	3	2	2
MPSCSA 76	77	51	34	23	16	11	7	5	4	3	2	2
MPSCSA 77	78	52	35	23	16	11	8	5	4	3	2	2
MPSCSA 78	79	53	35	24	16	11	8	6	4	3	2	2
MPSCSA 79	80	53	36	24	16	11	8	6	4	3	2	2
MPSCSA 80	81	54	36	24	16	11	8	6	4	3	2	2
MPSCSA 81	82	55	37	25	17	11	8	6	4	3	2	2
MPSCSA 82	83	55	37	25	17	12	8	6	4	3	2	2
MPSCSA 83	84	56	37	25	17	12	8	6	4	3	2	2
MPSCSA 84	85	57	38	25	17	12	8	6	4	3	2	2
MPSCSA 85	86	57	38	26	17	12	8	6	4	3	2	2
MPSCSA 86	87	58	39	26	18	12	8	6	4	3	2	2
MPSCSA 87	88	59	39	26	18	12	8	6	4	3	2	2
MPSCSA 88	89	59	40	27	18	12	8	6	4	3	2	2
MPSCSA 89	90	60	40	27	18	12	8	6	4	3	2	2
MPSCSA 90	91	61	41	27	18	12	8	6	4	3	2	2
MPSCSA 91	92	61	41	28	19	13	9	6	4	3	2	2
MPSCSA 92	93	62	41	28	19	13	9	6	4	3	2	2
MPSCSA 93	94	63	42	28	19	13	9	6	4	3	2	2
MPSCSA 94	95	63	42	28	19	13	9	6	4	3	2	2
MPSCSA 95	96	64	43	29	19	13	9	6	4	3	2	2
MPSCSA 96	97	65	43	29	20	13	9	6	4	3	2	2
MPSCSA 97	98	65	44	29	20	14	9	6	4	3	2	2
MPSCSA 98	99	66	44	30	20	14	10	7	5	3	2	2
MPSCSA 99	100	67	45	30	20	14	10	7	5	4	3	2
MPSCSA 100	101	67	45	30	20	14	10	7	5	4	3	2
MPSCSA 101	102	68	45	30	20	14	10	7	5	4	3	2
MPSCSA 102	103	69	46	31	21	14	10	7	5	4	3	2
MPSCSA 103	104	69	46	31	21	14	10	7	5	4	3	2
MPSCSA 104	105	70	47	31	21	14	10	7	5	4	3	2
MPSCSA 105	106	71	47	32	21	14	10	7	5	4	3	2
MPSCSA 106	107	71	48	32	22	15	10	7	5	4	3	2
MPSCSA 107	108	72	48	32	22	15	10	7	5	4	3	2
MPSCSA 108	109	73	49	33	22	15	10	7	5	4	3	2
MPSCSA 109	110	73	49	33	22	15	10	7	5	4	3	2
MPSCSA 110	111	74	49	33	22	15	10	7	5	4	3	2
MPSCSA 111	112	75	50	33	22	15	10	7	5	4	3	2
MPSCSA 112	113	75	50	34	23	15	10	7	5	4	3	2
MPSCSA 113	114	76	51	34	23	16	11	7	5	4	3	2
MPSCSA 114	115	77	51	34	23	16	11	8	5	4	3	2
MPSCSA 115	116	77	52	35	23	16	11	8	6	4	3	2
MPSCSA 116	117	78	52	35	24	16	11	8	6	4	3	2
MPSCSA 117	118	79	53	35	24	16	11	8	6	4	3	2
MPSCSA 118	119	79	53	36	24	16	11	8	6	4	3	2
MPSCSA 119	120	80	53	36	24	16	11	8	6	4	3	2
MPSCSA 120	121	81	54	36	24	16	11	8	6	4	3	2
MPSCSA 121	122	81	54	36	24	16	11	8	6	4	3	2
MPSCSA 122	123	82	55	37	25	17	11	8	6	4	3	2
MPSCSA 123	124	83	55	37	25	17	12	8	6	4	3	2
MPSCSA 124	125	83	56	37	25	17	12	8	6	4	3	2
MPSCSA 125	126	84	56	38	25	17	12	8	6	4	3	2
MPSCSA 126	127	85	57	38	26	17	12	8	6	4	3	2
MPSCSA 127	128	85	57	38	26	18	12	8	6	4	3	2
MPSCSA 128	129	86	57	38	26	18	12	8	6	4	3	2
MPSCSA 129	130	87	58	39	26	18	12	8	6	4	3	2
MPSCSA 130	131	87	58	39	26	18	12	8	6	4	3	2
MPSCSA 131	132	88	59	39	26	18	12	8	6	4	3	2
MPSCSA 132	133	89	59	40	27	18	12	8	6	4	3	2
MPSCSA 133	134	89	60	40	27	18	12	8	6	4	3	2
MPSCSA 134	135	90	60	40	27	18	12	8	6	4	3	2
MPSCSA 135	136	91	61	41	27	18	12	8	6	4	3	2
MPSCSA 136	137	91	61	41	28	19	13	9	6	4	3	2
MPSCSA 137	138	92	61	41	28	19	13	9	6	4	3	2
MPSCSA 138	139	93	62	41	28	19	13	9	6	4	3	2
MPSCSA 139	140	93	62	42	28	19	13	9	6	4	3	2
MPSCSA 140	141	94	63	42	28	19	13	9	6	4	3	2
MPSCSA 141	142	95	63	42	28	19	13	9	6	4	3	2
MPSCSA 142	143	95	64	43	29	19	13	9	6	4	3	2
MPSCSA 143	144	96	64	43	29	20	13	9	6	4	3	2
MPSCSA 144	145	97	65	43	29	20	14	9	6	4	3	2
MPSCSA 145	146	97	65	44	29	20	14	10	7	5	3	2
MPSCSA 146	147	98	65	44	30	20	14	10	7	5	4	3
MPSCSA 147	148	99	66	44	30	20	14	10	7	5	4	3
MPSCSA 148	149	99	66	44	30	20	14	10	7	5	4	3

\ ステップ	0	1	2	3	4	5	6	7	8	9	10	11
MPSCSA 149	150	100	67	45	30	20	14	10	7	5	4	3
MPSCSA 150	151	101	67	45	30	20	14	10	7	5	4	3
MPSCSA 151	152	101	68	45	30	20	14	10	7	5	4	3
MPSCSA 152	153	102	68	46	31	21	14	10	7	5	4	3
MPSCSA 153	154	103	69	46	31	21	14	10	7	5	4	3
MPSCSA 154	155	103	69	46	31	21	14	10	7	5	4	3
MPSCSA 155	156	104	69	46	31	21	14	10	7	5	4	3
MPSCSA 156	157	105	70	47	31	21	14	10	7	5	4	3
MPSCSA 157	158	105	70	47	32	21	14	10	7	5	4	3
MPSCSA 158	159	106	71	47	32	22	15	10	7	5	4	3
MPSCSA 159	160	107	71	48	32	22	15	10	7	5	4	3
MPSCSA 160	161	107	72	48	32	22	15	10	7	5	4	3
MPSCSA 161	162	108	72	48	32	22	15	10	7	5	4	3
MPSCSA 162	161	108	72	48	32	22	15	10	7	5	4	3
MPSCSA 163	160	108	72	48	32	22	15	10	7	5	4	3
MPSCSA 164	159	106	72	48	32	22	15	10	7	5	4	3
MPSCSA 165	158	106	71	48	32	22	15	10	7	5	4	3
MPSCSA 166	157	106	71	48	32	22	15	10	7	5	4	3
MPSCSA 167	156	104	70	48	32	22	15	10	7	5	4	3
MPSCSA 168	155	104	70	47	32	22	15	10	7	5	4	3
MPSCSA 169	154	104	70	47	32	22	15	10	7	5	4	3
MPSCSA 170	153	102	69	46	32	22	15	10	7	5	4	3
MPSCSA 171	152	102	68	46	31	22	15	10	7	5	4	3
MPSCSA 172	151	102	68	46	31	21	14	10	7	5	4	3
MPSCSA 173	150	100	68	46	31	21	14	10	7	5	4	3
MPSCSA 174	149	100	67	46	31	21	14	10	7	5	4	3
MPSCSA 175	148	100	67	45	30	20	14	10	7	5	4	3
MPSCSA 176	147	98	66	44	30	20	14	10	7	5	4	3
MPSCSA 177	146	98	66	44	30	20	14	10	7	5	4	3
MPSCSA 178	145	98	66	44	30	20	14	10	7	5	4	3
MPSCSA 179	144	96	65	44	30	20	14	10	7	5	4	3
MPSCSA 180	143	96	64	44	30	20	14	10	7	5	4	3
MPSCSA 181	142	96	64	43	30	20	14	10	7	5	4	3
MPSCSA 182	141	94	64	43	29	20	14	10	7	5	4	3
MPSCSA 183	140	94	63	42	28	20	14	10	7	5	4	3
MPSCSA 184	139	94	63	42	28	19	14	10	7	5	4	3
MPSCSA 185	138	92	62	42	28	19	13	10	7	5	4	3
MPSCSA 186	137	92	62	42	28	19	13	9	6	4	4	3
MPSCSA 187	136	92	62	42	28	19	13	9	6	4	3	2
MPSCSA 188	135	90	61	42	28	19	13	9	6	4	3	2
MPSCSA 189	134	90	60	40	28	19	13	9	6	4	3	2
MPSCSA 190	133	90	60	40	27	18	12	8	6	4	3	2
MPSCSA 191	132	88	60	40	27	18	12	8	6	4	3	2
MPSCSA 192	131	88	59	40	27	18	12	8	6	4	3	2
MPSCSA 193	130	88	59	40	27	18	12	8	6	4	3	2
MPSCSA 194	129	86	58	40	27	18	12	8	6	4	3	2
MPSCSA 195	128	86	58	39	26	18	12	8	6	4	3	2
MPSCSA 196	127	86	58	39	26	18	12	8	6	4	3	2
MPSCSA 197	126	84	57	38	26	18	12	8	6	4	3	2
MPSCSA 198	125	84	56	38	26	18	12	8	6	4	3	2
MPSCSA 199	124	84	56	38	26	18	12	8	6	4	3	2
MPSCSA 200	123	82	56	38	26	18	12	8	6	4	3	2
MPSCSA 201	122	82	55	38	26	18	12	8	6	4	3	2
MPSCSA 202	121	82	55	37	26	18	12	8	6	4	3	2
MPSCSA 203	120	80	54	36	24	17	12	8	6	4	3	2
MPSCSA 204	119	80	54	36	24	16	12	8	6	4	3	2
MPSCSA 205	118	80	54	36	24	16	11	8	6	4	3	2
MPSCSA 206	117	78	53	36	24	16	11	8	6	4	3	2
MPSCSA 207	116	78	52	36	24	16	11	8	6	4	3	2
MPSCSA 208	115	78	52	35	24	16	11	8	6	4	3	2
MPSCSA 209	114	76	52	35	24	16	11	8	6	4	3	2
MPSCSA 210	113	76	51	34	24	16	11	8	6	4	3	2
MPSCSA 211	112	76	51	34	23	16	11	8	6	4	3	2
MPSCSA 212	111	74	50	34	23	16	11	8	6	4	3	2
MPSCSA 213	110	74	50	34	23	16	11	8	6	4	3	2
MPSCSA 214	109	74	50	34	23	16	11	8	6	4	3	2
MPSCSA 215	108	72	49	34	23	16	11	8	6	4	3	2
MPSCSA 216	107	72	48	32	22	16	11	8	6	4	3	2
MPSCSA 217	106	72	48	32	22	15	10	8	6	4	3	2
MPSCSA 218	105	70	48	32	22	15	10	7	6	4	3	2
MPSCSA 219	104	70	47	32	22	15	10	7	5	4	3	2
MPSCSA 220	103	70	47	32	22	15	10	7	5	4	3	2
MPSCSA 221	102	68	46	32	22	15	10	7	5	4	3	2
MPSCSA 222	101	68	46	31	22	15	10	7	5	4	3	2
MPSCSA 223	100	68	46	31	21	14	10	7	5	4	3	2
MPSCSA 224	99	66	45	30	20	14	10	7	5	4	3	2
MPSCSA 225	98	66	44	30	20	14	10	7	5	4	3	2
MPSCSA 226	97	66	44	30	20	14	10	7	5	4	3	2
MPSCSA 227	96	64	44	30	20	14	10	7	5	4	3	2
MPSCSA 228	95	64	43	30	20	14	10	7	5	4	3	2
MPSCSA 229	94	64	43	29	20	14	10	7	5	4	3	2
MPSCSA 230	93	62	42	28	20	14	10	7	5	4	3	2
MPSCSA 231	92	62	42	28	19	14	10	7	5	4	3	2
MPSCSA 232	91	62	42	28	19	13	10	7	5	4	3	2
MPSCSA 233	90	60	41	28	19	13	9	6	4	4	3	2
MPSCSA 234	89	60	40	28	19	13	9	6	4	3	-2	3
MPSCSA 235	88	60	40	27	18	12	8	6	4	3	-2	2

\ ステップ	0	1	2	3	4	5	6	7	8	9	10	11
MPSCSA 236	87	58	40	27	18	12	8	6	4	3	-2	2
MPSCSA 237	86	58	39	26	18	12	8	6	4	3	-2	2
MPSCSA 238	85	58	39	26	18	12	8	6	4	3	-2	2
MPSCSA 239	84	56	38	26	18	12	8	6	4	3	-2	2
MPSCSA 240	83	56	38	26	18	12	8	6	4	3	-2	2
MPSCSA 241	82	56	38	26	18	12	8	6	4	3	-2	2
MPSCSA 242	81	54	37	26	18	12	8	6	4	3	-2	2
MPSCSA 243	80	54	36	24	17	12	8	6	4	3	-2	2
MPSCSA 244	79	54	36	24	16	12	8	6	4	3	-2	2
MPSCSA 245	78	52	36	24	16	11	8	6	4	3	-2	2
MPSCSA 246	77	52	35	24	16	11	8	6	4	3	-2	2
MPSCSA 247	76	52	35	24	16	11	8	6	4	3	-2	2
MPSCSA 248	75	50	34	24	16	11	8	6	4	3	-2	2
MPSCSA 249	74	50	34	23	16	11	8	6	4	3	-2	2
MPSCSA 250	73	50	34	23	16	11	8	6	4	3	-2	2
MPSCSA 251	72	48	33	22	16	11	8	6	4	3	-2	2
MPSCSA 252	71	48	32	22	15	10	8	6	4	3	-2	2
MPSCSA 253	70	48	32	22	15	10	7	6	4	3	-2	2
MPSCSA 254	69	46	32	22	15	10	7	5	4	3	-2	2
MPSCSA 255	68	46	31	22	15	10	7	5	4	3	-2	2
MPSCSA 256	67	46	31	21	14	10	7	5	4	3	-2	2
MPSCSA 257	66	44	30	20	14	10	7	5	4	3	-2	2
MPSCSA 258	65	44	30	20	14	10	7	5	4	3	-2	2
MPSCSA 259	64	44	30	20	14	10	7	5	4	3	-2	2
MPSCSA 260	63	42	29	20	14	10	7	5	4	3	-2	2
MPSCSA 261	62	42	28	20	14	10	7	5	4	3	-2	2
MPSCSA 262	61	42	28	19	14	10	7	5	4	3	-2	2
MPSCSA 263	60	40	28	19	13	10	7	5	4	3	-2	2
MPSCSA 264	59	40	27	18	12	8	6	4	4	3	-2	2
MPSCSA 265	58	40	27	18	12	8	6	4	3	-2	3	1
MPSCSA 266	57	38	26	18	12	8	6	4	3	-2	-2	3
MPSCSA 267	56	38	26	18	12	8	6	4	3	-2	-2	2
MPSCSA 268	55	38	26	18	12	8	6	4	3	-2	-2	2
MPSCSA 269	54	36	25	18	12	8	6	4	3	-2	-2	2
MPSCSA 270	53	36	24	16	12	8	6	4	3	-2	-2	2
MPSCSA 271	52	36	24	16	11	8	6	4	3	-2	-2	2
MPSCSA 272	51	34	24	16	11	8	6	4	3	-2	-2	2
MPSCSA 273	50	34	23	16	11	8	6	4	3	-2	-2	2
MPSCSA 274	49	34	23	16	11	8	6	4	3	-2	-2	2
MPSCSA 275	48	32	22	16	11	8	6	4	3	-2	-2	2
MPSCSA 276	47	32	22	15	10	8	6	4	3	-2	-2	2
MPSCSA 277	46	32	22	15	10	7	6	4	3	-2	-2	2
MPSCSA 278	45	30	21	14	10	7	5	4	3	-2	-2	2
MPSCSA 279	44	30	20	14	10	7	5	4	3	-2	-2	2
MPSCSA 280	43	30	20	14	10	7	5	4	3	-2	-2	2
MPSCSA 281	42	28	20	14	10	7	5	4	3	-2	-2	2
MPSCSA 282	41	28	19	14	10	7	5	4	3	-2	-2	2
MPSCSA 283	40	28	19	13	10	7	5	4	3	-2	-2	2
MPSCSA 284	39	26	18	12	8	6	4	4	3	-2	-2	2
MPSCSA 285	38	26	18	12	8	6	4	3	-2	3	1	1
MPSCSA 286	37	26	18	12	8	6	4	3	-2	-2	3	1
MPSCSA 287	36	24	17	12	8	6	4	3	-2	-2	-2	3
MPSCSA 288	35	24	16	12	8	6	4	3	-2	-2	-2	2
MPSCSA 289	34	24	16	11	8	6	4	3	-2	-2	-2	2
MPSCSA 290	33	22	16	11	8	6	4	3	-2	-2	-2	2
MPSCSA 291	32	22	15	10	8	6	4	3	-2	-2	-2	2
MPSCSA 292	31	22	15	10	7	6	4	3	-2	-2	-2	2
MPSCSA 293	30	20	14	10	7	5	4	3	-2	-2	-2	2
MPSCSA 294	29	20	14	10	7	5	4	3	-2	-2	-2	2
MPSCSA 295	28	20	14	10	7	5	4	3	-2	-2	-2	2
MPSCSA 296	27	18	13	10	7	5	4	3	-2	-2	-2	2
MPSCSA 297	26	18	12	8	6	4	4	3	-2	-2	-2	2
MPSCSA 298	25	18	12	8	6	4	3	-2	3	1	1	1
MPSCSA 299	24	16	12	8	6	4	3	-2	-2	3	1	1
MPSCSA 300	23	16	11	8	6	4	3	-2	-2	-2	3	1
MPSCSA 301	22	16	11	8	6	4	3	-2	-2	-2	-2	3
MPSCSA 302	21	14	10	8	6	4	3	-2	-2	-2	-2	2
MPSCSA 303	20	14	10	7	6	4	3	-2	-2	-2	-2	2
MPSCSA 304	19	14	10	7	5	4	3	-2	-2	-2	-2	2
MPSCSA 305	18	12	9	6	4	4	3	-2	-2	-2	-2	2
MPSCSA 306	17	12	8	6	4	3	-2	3	1	1	1	1
MPSCSA 307	16	12	8	6	4	3	-2	-2	3	1	1	1
MPSCSA 308	15	10	8	6	4	3	-2	-2	-2	3	1	1
MPSCSA 309	14	10	7	6	4	3	-2	-2	-2	-2	3	1
MPSCSA 310	13	10	7	5	4	3	-2	-2	-2	-2	-2	3
MPSCSA 311	12	8	6	4	4	3	-2	-2	-2	-2	-2	2
MPSCSA 312	11	8	6	4	3	-2	3	1	1	1	1	1
MPSCSA 313	10	8	6	4	3	-2	-2	3	1	1	1	1
MPSCSA 314	9	6	5	4	3	-2	-2	-2	3	1	1	1
MPSCSA 315	8	6	4	4	3	-2	-2	-2	-2	3	1	1
MPSCSA 316	7	6	4	3	-2	3	1	1	1	1	-2	2
MPSCSA 317	6	4	4	3	-2	-2	3	1	1	1	1	1
MPSCSA 318	5	4	3	-2	3	1	1	-2	-2	-2	-2	2
MPSCSA 319	4	4	3	-2	-2	3	1	1	1	1	1	1
MPSCSA 320	3	-2	3	-2	-2	-2	3	1	1	1	1	1
MPSCSA 321	-2	3	1	-2	-2	-2	-2	3	1	1	1	1
MPSCSA 322	1	1	-2	-2	-2	-2	-2	-2	3	1	1	1
MPSCSA 323	0	0	0	0	0	0	0	0	0	1	1	1

A.2　命令コード

16 段パイプラインの暗号ハードウェア・シミュレータで使用した命令コードをここに記す。表記は 10 進法を使っている。オペコードは

オペランド選択, next, カウンタスカラー設定, 書き込み, qo イニシャル, qo アップデート,

の順になっている。演算 $Y = ecc(X_0,\ X_1,\ X_2,\ X_3,\ i,\ j)$ 対して、オペコード 0 とオペコード 1 は

Y のアドレス, X_0 のアドレス, X_1 のアドレス,X_2 のアドレス, X_3 のアドレス, i, j,

の順になっている。

アドレス	オペコード	グループ	オペランド 0							オペランド 1							オペランド 2	next アドレス
0	2 0 0 0 0 0 0	0	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	1
1	2 0 0 0 0 0 0	1	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	2
2	2 0 0 0 0 0 0	2	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	3
3	2 0 0 0 0 0 0	3	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	4
4	2 0 0 0 0 0 0	4	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	5
5	2 0 0 0 0 0 0	5	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	6
6	2 0 0 0 0 0 0	6	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	7
7	2 0 0 0 0 0 0	7	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	8
8	2 0 0 0 0 0 0	8	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	9
9	2 0 0 0 0 0 0	9	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	10
10	2 0 0 0 0 0 0	10	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	11
11	2 0 0 0 0 0 0	11	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	12
12	2 0 0 0 0 0 0	12	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	13
13	2 0 0 0 0 0 0	13	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	14
14	2 0 0 0 0 0 0	14	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	15
15	2 0 0 0 0 0 0	15	0	12	10	11	10	0	0	0	13	10	11	10	0	0	0	16
16	0 0 0 0 0 0 0	0	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	17
17	0 0 0 0 0 0 0	1	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	18
18	0 0 0 0 0 0 0	2	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	19
19	0 0 0 0 0 0 0	3	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	20
20	0 0 0 0 0 0 0	4	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	21
21	0 0 0 0 0 0 0	5	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	22
22	0 0 0 0 0 0 0	6	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	23
23	0 0 0 0 0 0 0	7	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	24
24	0 0 0 0 0 0 0	8	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	25
25	0 0 0 0 0 0 0	9	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	26
26	0 0 0 0 0 0 0	10	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	27
27	0 0 0 0 0 0 0	11	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	28
28	0 0 0 0 0 0 0	12	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	29
29	0 0 0 0 0 0 0	13	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	30
30	0 0 0 0 0 0 0	14	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	31
31	0 0 0 0 0 0 0	15	2	11	10	11	10	0	0	0	0	0	0	0	0	0	0	32
32	0 0 0 0 0 0 0	0	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	33
33	0 0 0 0 0 0 0	1	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	34
34	0 0 0 0 0 0 0	2	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	35
35	0 0 0 0 0 0 0	3	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	36
36	0 0 0 0 0 0 0	4	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	37
37	0 0 0 0 0 0 0	5	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	38
38	0 0 0 0 0 0 0	6	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	39
39	0 0 0 0 0 0 0	7	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	40
40	0 0 0 0 0 0 0	8	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	41
41	0 0 0 0 0 0 0	9	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	42
42	0 0 0 0 0 0 0	10	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	43
43	0 0 0 0 0 0 0	11	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	44
44	0 0 0 0 0 0 0	12	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	45
45	0 0 0 0 0 0 0	13	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	46
46	0 0 0 0 0 0 0	14	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	47
47	0 0 0 0 0 0 0	15	4	0	10	11	10	0	0	0	0	0	0	0	0	0	0	48
48	0 0 0 0 0 0 0	0	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	49
49	0 0 0 0 0 0 0	1	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	50
50	0 0 0 0 0 0 0	2	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	51
51	0 0 0 0 0 0 0	3	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	52
52	0 0 0 0 0 0 0	4	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	53
53	0 0 0 0 0 0 0	5	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	54
54	0 0 0 0 0 0 0	6	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	55
55	0 0 0 0 0 0 0	7	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	56
56	0 0 0 0 0 0 0	8	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	57
57	0 0 0 0 0 0 0	9	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	58
58	0 0 0 0 0 0 0	10	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	59
59	0 0 0 0 0 0 0	11	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	60
60	0 0 0 0 0 0 0	12	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	61
61	0 0 0 0 0 0 0	13	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	62

アドレス	オペコード	グループ	オペランド 0							オペランド 1							オペランド 2	next アドレス
62	0 0 0 0 0 0 0	14	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	63
63	0 0 0 0 0 0 0	15	6	2	10	11	10	0	0	0	0	0	0	0	0	0	0	64
64	0 0 0 0 0 0 0	0	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	65
65	0 0 0 0 0 0 0	1	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	66
66	0 0 0 0 0 0 0	2	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	67
67	0 0 0 0 0 0 0	3	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	68
68	0 0 0 0 0 0 0	4	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	69
69	0 0 0 0 0 0 0	5	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	70
70	0 0 0 0 0 0 0	6	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	71
71	0 0 0 0 0 0 0	7	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	72
72	0 0 0 0 0 0 0	8	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	73
73	0 0 0 0 0 0 0	9	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	74
74	0 0 0 0 0 0 0	10	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	75
75	0 0 0 0 0 0 0	11	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	76
76	0 0 0 0 0 0 0	12	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	77
77	0 0 0 0 0 0 0	13	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	78
78	0 0 0 0 0 0 0	14	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	79
79	0 0 0 0 0 0 0	15	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	80
80	0 0 0 0 0 0 0	0	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	81
81	0 0 0 0 0 0 0	1	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	82
82	0 0 0 0 0 0 0	2	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	83
83	0 0 0 0 0 0 0	3	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	84
84	0 0 0 0 0 0 0	4	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	85
85	0 0 0 0 0 0 0	5	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	86
86	0 0 0 0 0 0 0	6	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	87
87	0 0 0 0 0 0 0	7	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	88
88	0 0 0 0 0 0 0	8	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	89
89	0 0 0 0 0 0 0	9	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	90
90	0 0 0 0 0 0 0	10	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	91
91	0 0 0 0 0 0 0	11	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	92
92	0 0 0 0 0 0 0	12	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	93
93	0 0 0 0 0 0 0	13	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	94
94	0 0 0 0 0 0 0	14	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	95
95	0 0 2 0 0 0 0	15	7	10	10	10	10	0	0	0	0	0	0	0	0	0	97	96
96	0 0 0 1 0 1 0	0	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	97
97	0 0 0 2 0 1 0	1	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	98
98	0 0 0 1 0 1 0	2	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	99
99	0 0 0 2 0 1 0	3	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	100
100	0 0 0 1 0 1 0	4	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	101
101	0 0 0 2 0 1 0	5	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	102
102	0 0 0 1 0 1 0	6	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	103
103	0 0 0 2 0 1 0	7	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	104
104	0 0 0 1 0 1 0	8	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	105
105	0 0 0 2 0 1 0	9	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	106
106	0 0 0 1 0 1 0	10	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	107
107	0 0 0 2 0 1 0	11	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	108
108	0 0 0 1 0 1 0	12	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	109
109	0 0 0 2 0 1 0	13	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	110
110	0 0 0 1 0 1 0	14	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	111
111	0 0 0 2 0 1 0	15	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	128
112	0 0 0 4 0 0 0	0	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	113
113	0 0 0 4 0 0 0	1	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	114
114	0 0 0 4 0 0 0	2	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	115
115	0 0 0 4 0 0 0	3	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	116
116	0 0 0 4 0 0 0	4	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	117
117	0 0 0 4 0 0 0	5	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	118
118	0 0 0 4 0 0 0	6	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	119
119	0 0 0 4 0 0 0	7	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	120
120	0 0 0 4 0 0 0	8	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	121
121	0 0 0 4 0 0 0	9	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	122
122	0 0 0 4 0 0 0	10	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	123
123	0 0 0 4 0 0 0	11	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	124
124	0 0 0 4 0 0 0	12	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	125
125	0 0 0 4 0 0 0	13	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	126
126	0 0 0 4 0 0 0	14	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	127
127	0 0 0 4 0 0 0	15	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	128
128	3 0 0 0 0 0 0	0	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	129
129	3 0 0 0 0 0 0	1	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	130
130	3 0 0 0 0 0 0	2	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	131
131	3 0 0 0 0 0 0	3	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	132
132	3 0 0 0 0 0 0	4	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	133
133	3 0 0 0 0 0 0	5	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	134
134	3 0 0 0 0 0 0	6	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	135
135	3 0 0 0 0 0 0	7	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	136
136	3 0 0 0 0 0 0	8	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	137
137	3 0 0 0 0 0 0	9	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	138
138	3 0 0 0 0 0 0	10	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	139
139	3 0 0 0 0 0 0	11	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	140
140	3 0 0 0 0 0 0	12	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	141
141	3 0 0 0 0 0 0	13	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	142
142	3 0 0 0 0 0 0	14	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	143
143	3 0 0 0 0 0 0	15	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	144
144	1 0 0 0 1 0 0	0	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	145
145	1 0 0 0 1 0 0	1	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	146
146	1 0 0 0 1 0 0	2	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	147
147	1 0 0 0 1 0 0	3	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	148

アドレス	オペコード	グループ	オペランド 0							オペランド 1							オペランド 2	next アドレス
148	1 0 0 0 1 0 0	4	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	149
149	1 0 0 0 1 0 0	5	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	150
150	1 0 0 0 1 0 0	6	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	151
151	1 0 0 0 1 0 0	7	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	152
152	1 0 0 0 1 0 0	8	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	153
153	1 0 0 0 1 0 0	9	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	154
154	1 0 0 0 1 0 0	10	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	155
155	1 0 0 0 1 0 0	11	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	156
156	1 0 0 0 1 0 0	12	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	157
157	1 0 0 0 1 0 0	13	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	158
158	1 0 0 0 1 0 0	14	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	159
159	1 0 0 0 1 0 0	15	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	160
160	3 0 0 0 0 0 0	0	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	161
161	3 0 0 0 0 0 0	1	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	162
162	3 0 0 0 0 0 0	2	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	163
163	3 0 0 0 0 0 0	3	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	164
164	3 0 0 0 0 0 0	4	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	165
165	3 0 0 0 0 0 0	5	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	166
166	3 0 0 0 0 0 0	6	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	167
167	3 0 0 0 0 0 0	7	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	168
168	3 0 0 0 0 0 0	8	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	169
169	3 0 0 0 0 0 0	9	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	170
170	3 0 0 0 0 0 0	10	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	171
171	3 0 0 0 0 0 0	11	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	172
172	3 0 0 0 0 0 0	12	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	173
173	3 0 0 0 0 0 0	13	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	174
174	3 0 0 0 0 0 0	14	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	175
175	3 0 0 0 0 0 0	15	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	176
176	1 0 0 0 1 0 0	0	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	177
177	1 0 0 0 1 0 0	1	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	178
178	1 0 0 0 1 0 0	2	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	179
179	1 0 0 0 1 0 0	3	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	180
180	1 0 0 0 1 0 0	4	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	181
181	1 0 0 0 1 0 0	5	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	182
182	1 0 0 0 1 0 0	6	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	183
183	1 0 0 0 1 0 0	7	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	184
184	1 0 0 0 1 0 0	8	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	185
185	1 0 0 0 1 0 0	9	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	186
186	1 0 0 0 1 0 0	10	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	187
187	1 0 0 0 1 0 0	11	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	188
188	1 0 0 0 1 0 0	12	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	189
189	1 0 0 0 1 0 0	13	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	190
190	1 0 0 0 1 0 0	14	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	191
191	1 0 0 0 1 0 0	15	6	8	10	0	10	0	0	7	8	10	0	10	0	0	0	192
192	1 0 0 0 0 0 0	0	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	193
193	1 0 0 0 0 0 0	1	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	194
194	1 0 0 0 0 0 0	2	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	195
195	1 0 0 0 0 0 0	3	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	196
196	1 0 0 0 0 0 0	4	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	197
197	1 0 0 0 0 0 0	5	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	198
198	1 0 0 0 0 0 0	6	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	199
199	1 0 0 0 0 0 0	7	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	200
200	1 0 0 0 0 0 0	8	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	201
201	1 0 0 0 0 0 0	9	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	202
202	1 0 0 0 0 0 0	10	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	203
203	1 0 0 0 0 0 0	11	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	204
204	1 0 0 0 0 0 0	12	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	205
205	1 0 0 0 0 0 0	13	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	206
206	1 0 0 0 0 0 0	14	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	207
207	1 0 0 0 0 0 0	15	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	208
208	1 0 0 0 0 0 0	0	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	209
209	1 0 0 0 0 0 0	1	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	210
210	1 0 0 0 0 0 0	2	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	211
211	1 0 0 0 0 0 0	3	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	212
212	1 0 0 0 0 0 0	4	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	213
213	1 0 0 0 0 0 0	5	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	214
214	1 0 0 0 0 0 0	6	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	215
215	1 0 0 0 0 0 0	7	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	216
216	1 0 0 0 0 0 0	8	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	217
217	1 0 0 0 0 0 0	9	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	218
218	1 0 0 0 0 0 0	10	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	219
219	1 0 0 0 0 0 0	11	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	220
220	1 0 0 0 0 0 0	12	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	221
221	1 0 0 0 0 0 0	13	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	222
222	1 0 0 0 0 0 0	14	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	223
223	1 0 0 0 0 0 0	15	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	224
224	1 0 0 0 2 0 0	0	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	225
225	1 0 0 0 2 0 0	1	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	226
226	1 0 0 0 2 0 0	2	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	227
227	1 0 0 0 2 0 0	3	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	228
228	1 0 0 0 2 0 0	4	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	229
229	1 0 0 0 2 0 0	5	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	230
230	1 0 0 0 2 0 0	6	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	231
231	1 0 0 0 2 0 0	7	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	232
232	1 0 0 0 2 0 0	8	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	233

アドレス	オペコード	グループ	オペランド 0							オペランド 1							オペランド 2	next アドレス
233	1 0 0 0 2 0 0	9	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	234
234	1 0 0 0 2 0 0	10	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	235
235	1 0 0 0 2 0 0	11	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	236
236	1 0 0 0 2 0 0	12	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	237
237	1 0 0 0 2 0 0	13	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	238
238	1 0 0 0 2 0 0	14	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	239
239	1 0 0 0 2 0 0	15	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	240
240	0 0 0 0 0 0 0	0	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	241
241	0 0 0 0 0 0 0	1	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	242
242	0 0 0 0 0 0 0	2	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	243
243	0 0 0 0 0 0 0	3	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	244
244	0 0 0 0 0 0 0	4	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	245
245	0 0 0 0 0 0 0	5	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	246
246	0 0 0 0 0 0 0	6	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	247
247	0 0 0 0 0 0 0	7	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	248
248	0 0 0 0 0 0 0	8	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	249
249	0 0 0 0 0 0 0	9	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	250
250	0 0 0 0 0 0 0	10	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	251
251	0 0 0 0 0 0 0	11	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	252
252	0 0 0 0 0 0 0	12	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	253
253	0 0 0 0 0 0 0	13	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	254
254	0 0 0 0 0 0 0	14	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	255
255	0 0 0 0 0 0 0	15	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	256
256	1 0 0 0 2 0 1	0	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	257
257	1 0 0 0 2 0 1	1	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	258
258	1 0 0 0 2 0 1	2	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	259
259	1 0 0 0 2 0 1	3	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	260
260	1 0 0 0 2 0 1	4	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	261
261	1 0 0 0 2 0 1	5	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	262
262	1 0 0 0 2 0 1	6	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	263
263	1 0 0 0 2 0 1	7	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	264
264	1 0 0 0 2 0 1	8	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	265
265	1 0 0 0 2 0 1	9	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	266
266	1 0 0 0 2 0 1	10	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	267
267	1 0 0 0 2 0 1	11	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	268
268	1 0 0 0 2 0 1	12	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	269
269	1 0 0 0 2 0 1	13	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	270
270	1 0 1 0 2 0 1	14	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	271
271	1 1 0 0 2 0 1	15	7	8	9	15	10	0	0	6	8	9	15	10	0	0	112	272
272	0 0 0 4 0 0 0	0	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	273
273	0 0 0 4 0 0 0	1	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	274
274	0 0 0 4 0 0 0	2	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	275
275	0 0 0 4 0 0 0	3	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	276
276	0 0 0 4 0 0 0	4	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	277
277	0 0 0 4 0 0 0	5	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	278
278	0 0 0 4 0 0 0	6	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	279
279	0 0 0 4 0 0 0	7	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	280
280	0 0 0 4 0 0 0	8	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	281
281	0 0 0 4 0 0 0	9	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	282
282	0 0 0 4 0 0 0	10	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	283
283	0 0 0 4 0 0 0	11	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	284
284	0 0 0 4 0 0 0	12	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	285
285	0 0 0 4 0 0 0	13	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	286
286	0 0 0 4 0 0 0	14	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	287
287	0 0 0 4 0 0 0	15	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	288
288	3 0 0 0 0 0 0	0	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	289
289	3 0 0 0 0 0 0	1	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	290
290	3 0 0 0 0 0 0	2	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	291
291	3 0 0 0 0 0 0	3	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	292
292	3 0 0 0 0 0 0	4	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	293
293	3 0 0 0 0 0 0	5	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	294
294	3 0 0 0 0 0 0	6	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	295
295	3 0 0 0 0 0 0	7	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	296
296	3 0 0 0 0 0 0	8	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	297
297	3 0 0 0 0 0 0	9	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	298
298	3 0 0 0 0 0 0	10	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	299
299	3 0 0 0 0 0 0	11	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	300
300	3 0 0 0 0 0 0	12	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	301
301	3 0 0 0 0 0 0	13	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	302
302	3 0 0 0 0 0 0	14	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	303
303	3 0 0 0 0 0 0	15	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	304
304	0 0 0 0 1 0 0	0	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	305
305	0 0 0 0 1 0 0	1	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	306
306	0 0 0 0 1 0 0	2	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	307
307	0 0 0 0 1 0 0	3	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	308
308	0 0 0 0 1 0 0	4	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	309
309	0 0 0 0 1 0 0	5	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	310
310	0 0 0 0 1 0 0	6	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	311
311	0 0 0 0 1 0 0	7	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	312
312	0 0 0 0 1 0 0	8	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	313
313	0 0 0 0 1 0 0	9	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	314
314	0 0 0 0 1 0 0	10	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	315
315	0 0 0 0 1 0 0	11	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	316
316	0 0 0 0 1 0 0	12	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	317
317	0 0 0 0 1 0 0	13	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	318
318	0 0 0 0 1 0 0	14	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	319
319	0 0 0 0 1 0 0	15	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	320

アドレス	オペコード	グループ	オペランド 0							オペランド 1							オペランド 2	next アドレス
320	3 0 0 0 0 0 0	0	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	321
321	3 0 0 0 0 0 0	1	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	322
322	3 0 0 0 0 0 0	2	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	323
323	3 0 0 0 0 0 0	3	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	324
324	3 0 0 0 0 0 0	4	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	325
325	3 0 0 0 0 0 0	5	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	326
326	3 0 0 0 0 0 0	6	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	327
327	3 0 0 0 0 0 0	7	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	328
328	3 0 0 0 0 0 0	8	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	329
329	3 0 0 0 0 0 0	9	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	330
330	3 0 0 0 0 0 0	10	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	331
331	3 0 0 0 0 0 0	11	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	332
332	3 0 0 0 0 0 0	12	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	333
333	3 0 0 0 0 0 0	13	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	334
334	3 0 0 0 0 0 0	14	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	335
335	3 0 0 0 0 0 0	15	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	336
336	1 0 0 0 1 0 0	0	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	337
337	1 0 0 0 1 0 0	1	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	338
338	1 0 0 0 1 0 0	2	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	339
339	1 0 0 0 1 0 0	3	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	340
340	1 0 0 0 1 0 0	4	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	341
341	1 0 0 0 1 0 0	5	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	342
342	1 0 0 0 1 0 0	6	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	343
343	1 0 0 0 1 0 0	7	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	344
344	1 0 0 0 1 0 0	8	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	345
345	1 0 0 0 1 0 0	9	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	346
346	1 0 0 0 1 0 0	10	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	347
347	1 0 0 0 1 0 0	11	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	348
348	1 0 0 0 1 0 0	12	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	349
349	1 0 0 0 1 0 0	13	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	350
350	1 0 0 0 1 0 0	14	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	351
351	1 0 0 0 1 0 0	15	6	8	10	0	10	0	0	2	8	10	0	10	0	0	0	352
352	1 0 0 0 3 0 0	0	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	353
353	1 0 0 0 3 0 0	1	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	354
354	1 0 0 0 3 0 0	2	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	355
355	1 0 0 0 3 0 0	3	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	356
356	1 0 0 0 3 0 0	4	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	357
357	1 0 0 0 3 0 0	5	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	358
358	1 0 0 0 3 0 0	6	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	359
359	1 0 0 0 3 0 0	7	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	360
360	1 0 0 0 3 0 0	8	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	361
361	1 0 0 0 3 0 0	9	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	362
362	1 0 0 0 3 0 0	10	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	363
363	1 0 0 0 3 0 0	11	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	364
364	1 0 0 0 3 0 0	12	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	365
365	1 0 0 0 3 0 0	13	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	366
366	1 0 0 0 3 0 0	14	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	367
367	1 0 0 0 3 0 0	15	8	5	7	10	10	0	0	0	4	10	11	10	0	0	0	368
368	4 0 0 0 0 0 0	0	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	369
369	4 0 0 0 0 0 0	1	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	370
370	4 0 0 0 0 0 0	2	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	371
371	4 0 0 0 0 0 0	3	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	372
372	4 0 0 0 0 0 0	4	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	373
373	4 0 0 0 0 0 0	5	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	374
374	4 0 0 0 0 0 0	6	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	375
375	4 0 0 0 0 0 0	7	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	376
376	4 0 0 0 0 0 0	8	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	377
377	4 0 0 0 0 0 0	9	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	378
378	4 0 0 0 0 0 0	10	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	379
379	4 0 0 0 0 0 0	11	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	380
380	4 0 0 0 0 0 0	12	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	381
381	4 0 0 0 0 0 0	13	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	382
382	4 0 0 0 0 0 0	14	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	383
383	4 0 0 0 0 0 0	15	0	10	10	10	10	0	0	0	0	0	0	0	0	0	0	384
384	1 0 0 0 2 0 0	0	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	385
385	1 0 0 0 2 0 0	1	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	386
386	1 0 0 0 2 0 0	2	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	387
387	1 0 0 0 2 0 0	3	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	388
388	1 0 0 0 2 0 0	4	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	389
389	1 0 0 0 2 0 0	5	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	390
390	1 0 0 0 2 0 0	6	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	391
391	1 0 0 0 2 0 0	7	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	392
392	1 0 0 0 2 0 0	8	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	393
393	1 0 0 0 2 0 0	9	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	394
394	1 0 0 0 2 0 0	10	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	395
395	1 0 0 0 2 0 0	11	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	396
396	1 0 0 0 2 0 0	12	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	397
397	1 0 0 0 2 0 0	13	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	398
398	1 0 0 0 2 0 0	14	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	399
399	1 0 0 0 2 0 0	15	0	8	10	9	10	0	0	4	8	10	9	10	0	0	0	400
400	4 0 0 0 0 0 0	0	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	401
401	4 0 0 0 0 0 0	1	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	402
402	4 0 0 0 0 0 0	2	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	403
403	4 0 0 0 0 0 0	3	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	404
404	4 0 0 0 0 0 0	4	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	405
405	4 0 0 0 0 0 0	5	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	406

アドレス	オペコード	グループ	オペランド 0							オペランド 1							オペランド 2	next アドレス
406	4 0 0 0 0 0 0	6	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	407
407	4 0 0 0 0 0 0	7	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	408
408	4 0 0 0 0 0 0	8	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	409
409	4 0 0 0 0 0 0	9	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	410
410	4 0 0 0 0 0 0	10	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	411
411	4 0 0 0 0 0 0	11	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	412
412	4 0 0 0 0 0 0	12	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	413
413	4 0 0 0 0 0 0	13	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	414
414	4 0 0 0 0 0 0	14	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	415
415	4 0 0 0 0 0 0	15	2	10	10	10	10	0	0	0	0	0	0	0	0	0	0	416
416	1 0 0 0 2 0 1	0	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	417
417	1 0 0 0 2 0 1	1	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	418
418	1 0 0 0 2 0 1	2	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	419
419	1 0 0 0 2 0 1	3	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	420
420	1 0 0 0 2 0 1	4	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	421
421	1 0 0 0 2 0 1	5	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	422
422	1 0 0 0 2 0 1	6	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	423
423	1 0 0 0 2 0 1	7	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	424
424	1 0 0 0 2 0 1	8	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	425
425	1 0 0 0 2 0 1	9	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	426
426	1 0 0 0 2 0 1	10	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	427
427	1 0 0 0 2 0 1	11	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	428
428	1 0 0 0 2 0 1	12	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	429
429	1 0 0 0 2 0 1	13	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	430
430	1 0 0 0 2 0 1	14	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	431
431	1 0 0 0 2 0 1	15	2	8	9	15	10	0	0	6	8	9	15	10	0	0	0	432
432	2 0 0 0 0 0 0	0	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	433
433	2 0 0 0 0 0 0	1	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	434
434	2 0 0 0 0 0 0	2	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	435
435	2 0 0 0 0 0 0	3	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	436
436	2 0 0 0 0 0 0	4	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	437
437	2 0 0 0 0 0 0	5	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	438
438	2 0 0 0 0 0 0	6	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	439
439	2 0 0 0 0 0 0	7	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	440
440	2 0 0 0 0 0 0	8	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	441
441	2 0 0 0 0 0 0	9	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	442
442	2 0 0 0 0 0 0	10	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	443
443	2 0 0 0 0 0 0	11	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	444
444	2 0 0 0 0 0 0	12	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	445
445	2 0 0 0 0 0 0	13	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	446
446	2 0 0 0 0 0 0	14	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	447
447	2 0 0 0 0 0 0	15	1	12	10	11	10	0	0	1	13	10	11	10	0	0	0	448
448	0 0 0 0 0 0 0	0	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	449
449	0 0 0 0 0 0 0	1	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	450
450	0 0 0 0 0 0 0	2	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	451
451	0 0 0 0 0 0 0	3	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	452
452	0 0 0 0 0 0 0	4	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	453
453	0 0 0 0 0 0 0	5	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	454
454	0 0 0 0 0 0 0	6	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	455
455	0 0 0 0 0 0 0	7	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	456
456	0 0 0 0 0 0 0	8	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	457
457	0 0 0 0 0 0 0	9	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	458
458	0 0 0 0 0 0 0	10	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	459
459	0 0 0 0 0 0 0	11	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	460
460	0 0 0 0 0 0 0	12	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	461
461	0 0 0 0 0 0 0	13	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	462
462	0 0 0 0 0 0 0	14	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	463
463	0 0 0 0 0 0 0	15	3	11	10	11	10	0	0	0	0	0	0	0	0	0	0	464
464	0 0 0 0 0 0 0	0	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	465
465	0 0 0 0 0 0 0	1	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	466
466	0 0 0 0 0 0 0	2	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	467
467	0 0 0 0 0 0 0	3	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	468
468	0 0 0 0 0 0 0	4	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	469
469	0 0 0 0 0 0 0	5	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	470
470	0 0 0 0 0 0 0	6	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	471
471	0 0 0 0 0 0 0	7	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	472
472	0 0 0 0 0 0 0	8	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	473
473	0 0 0 0 0 0 0	9	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	474
474	0 0 0 0 0 0 0	10	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	475
475	0 0 0 0 0 0 0	11	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	476
476	0 0 0 0 0 0 0	12	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	477
477	0 0 0 0 0 0 0	13	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	478
478	0 0 0 0 0 0 0	14	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	479
479	0 0 0 0 0 0 0	15	4	1	10	11	10	0	0	0	0	0	0	0	0	0	0	480
480	0 0 0 0 0 0 0	0	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	481
481	0 0 0 0 0 0 0	1	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	482
482	0 0 0 0 0 0 0	2	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	483
483	0 0 0 0 0 0 0	3	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	484
484	0 0 0 0 0 0 0	4	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	485
485	0 0 0 0 0 0 0	5	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	486
486	0 0 0 0 0 0 0	6	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	487
487	0 0 0 0 0 0 0	7	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	488
488	0 0 0 0 0 0 0	8	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	489
489	0 0 0 0 0 0 0	9	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	490
490	0 0 0 0 0 0 0	10	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	491
491	0 0 0 0 0 0 0	11	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	492

アドレス	オペコード	グループ	オペランド 0							オペランド 1							オペランド 2	next アドレス
492	0 0 0 0 0 0 0	12	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	493
493	0 0 0 0 0 0 0	13	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	494
494	0 0 0 0 0 0 0	14	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	495
495	0 0 0 0 0 0 0	15	6	3	10	11	10	0	0	0	0	0	0	0	0	0	0	496
496	0 0 0 0 0 0 0	0	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	497
497	0 0 0 0 0 0 0	1	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	498
498	0 0 0 0 0 0 0	2	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	499
499	0 0 0 0 0 0 0	3	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	500
500	0 0 0 0 0 0 0	4	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	501
501	0 0 0 0 0 0 0	5	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	502
502	0 0 0 0 0 0 0	6	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	503
503	0 0 0 0 0 0 0	7	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	504
504	0 0 0 0 0 0 0	8	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	505
505	0 0 0 0 0 0 0	9	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	506
506	0 0 0 0 0 0 0	10	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	507
507	0 0 0 0 0 0 0	11	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	508
508	0 0 0 0 0 0 0	12	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	509
509	0 0 0 0 0 0 0	13	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	510
510	0 0 0 0 0 0 0	14	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	511
511	0 0 0 0 0 0 0	15	5	10	10	10	10	0	0	0	0	0	0	0	0	0	0	512
512	0 0 0 0 0 0 0	0	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	513
513	0 0 0 0 0 0 0	1	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	514
514	0 0 0 0 0 0 0	2	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	515
515	0 0 0 0 0 0 0	3	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	516
516	0 0 0 0 0 0 0	4	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	517
517	0 0 0 0 0 0 0	5	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	518
518	0 0 0 0 0 0 0	6	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	519
519	0 0 0 0 0 0 0	7	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	520
520	0 0 0 0 0 0 0	8	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	521
521	0 0 0 0 0 0 0	9	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	522
522	0 0 0 0 0 0 0	10	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	523
523	0 0 0 0 0 0 0	11	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	524
524	0 0 0 0 0 0 0	12	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	525
525	0 0 0 0 0 0 0	13	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	526
526	0 0 0 0 0 0 0	14	7	10	10	10	10	0	0	0	0	0	0	0	0	0	0	527
527	0 0 2 0 0 0 0	15	7	10	10	10	10	0	0	0	0	0	0	0	0	0	97	528
528	0 0 0 1 0 1 0	0	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	529
529	0 0 0 2 0 1 0	1	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	530
530	0 0 0 1 0 1 0	2	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	531
531	0 0 0 2 0 1 0	3	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	532
532	0 0 0 1 0 1 0	4	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	533
533	0 0 0 2 0 1 0	5	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	534
534	0 0 0 1 0 1 0	6	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	535
535	0 0 0 2 0 1 0	7	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	536
536	0 0 0 1 0 1 0	8	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	537
537	0 0 0 2 0 1 0	9	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	538
538	0 0 0 1 0 1 0	10	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	539
539	0 0 0 2 0 1 0	11	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	540
540	0 0 0 1 0 1 0	12	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	541
541	0 0 0 2 0 1 0	13	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	542
542	0 0 0 1 0 1 0	14	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	543
543	0 0 0 2 0 1 0	15	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	560
544	0 0 0 4 0 0 0	0	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	545
545	0 0 0 4 0 0 0	1	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	546
546	0 0 0 4 0 0 0	2	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	547
547	0 0 0 4 0 0 0	3	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	548
548	0 0 0 4 0 0 0	4	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	549
549	0 0 0 4 0 0 0	5	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	550
550	0 0 0 4 0 0 0	6	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	551
551	0 0 0 4 0 0 0	7	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	552
552	0 0 0 4 0 0 0	8	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	553
553	0 0 0 4 0 0 0	9	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	554
554	0 0 0 4 0 0 0	10	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	555
555	0 0 0 4 0 0 0	11	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	556
556	0 0 0 4 0 0 0	12	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	557
557	0 0 0 4 0 0 0	13	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	558
558	0 0 0 4 0 0 0	14	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	559
559	0 0 0 4 0 0 0	15	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	560
560	3 0 0 0 0 0 0	0	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	561
561	3 0 0 0 0 0 0	1	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	562
562	3 0 0 0 0 0 0	2	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	563
563	3 0 0 0 0 0 0	3	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	564
564	3 0 0 0 0 0 0	4	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	565
565	3 0 0 0 0 0 0	5	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	566
566	3 0 0 0 0 0 0	6	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	567
567	3 0 0 0 0 0 0	7	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	568
568	3 0 0 0 0 0 0	8	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	569
569	3 0 0 0 0 0 0	9	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	570
570	3 0 0 0 0 0 0	10	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	571
571	3 0 0 0 0 0 0	11	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	572
572	3 0 0 0 0 0 0	12	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	573
573	3 0 0 0 0 0 0	13	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	574
574	3 0 0 0 0 0 0	14	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	575
575	3 0 0 0 0 0 0	15	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	576
576	1 0 0 0 1 0 0	0	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	577
577	1 0 0 0 1 0 0	1	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	578

アドレス	オペコード	グループ	オペランド 0							オペランド 1							オペランド 2	next アドレス
578	1 0 0 0 1 0 0	2	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	579
579	1 0 0 0 1 0 0	3	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	580
580	1 0 0 0 1 0 0	4	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	581
581	1 0 0 0 1 0 0	5	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	582
582	1 0 0 0 1 0 0	6	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	583
583	1 0 0 0 1 0 0	7	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	584
584	1 0 0 0 1 0 0	8	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	585
585	1 0 0 0 1 0 0	9	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	586
586	1 0 0 0 1 0 0	10	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	587
587	1 0 0 0 1 0 0	11	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	588
588	1 0 0 0 1 0 0	12	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	589
589	1 0 0 0 1 0 0	13	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	590
590	1 0 0 0 1 0 0	14	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	591
591	1 0 0 0 1 0 0	15	4	8	9	10	10	0	0	5	8	9	10	10	0	0	0	592
592	3 0 0 0 0 0 0	0	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	593
593	3 0 0 0 0 0 0	1	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	594
594	3 0 0 0 0 0 0	2	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	595
595	3 0 0 0 0 0 0	3	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	596
596	3 0 0 0 0 0 0	4	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	597
597	3 0 0 0 0 0 0	5	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	598
598	3 0 0 0 0 0 0	6	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	599
599	3 0 0 0 0 0 0	7	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	600
600	3 0 0 0 0 0 0	8	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	601
601	3 0 0 0 0 0 0	9	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	602
602	3 0 0 0 0 0 0	10	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	603
603	3 0 0 0 0 0 0	11	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	604
604	3 0 0 0 0 0 0	12	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	605
605	3 0 0 0 0 0 0	13	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	606
606	3 0 0 0 0 0 0	14	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	607
607	3 0 0 0 0 0 0	15	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	608
608	1 0 0 0 1 0 0	0	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	609
609	1 0 0 0 1 0 0	1	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	610
610	1 0 0 0 1 0 0	2	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	611
611	1 0 0 0 1 0 0	3	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	612
612	1 0 0 0 1 0 0	4	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	613
613	1 0 0 0 1 0 0	5	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	614
614	1 0 0 0 1 0 0	6	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	615
615	1 0 0 0 1 0 0	7	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	616
616	1 0 0 0 1 0 0	8	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	617
617	1 0 0 0 1 0 0	9	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	618
618	1 0 0 0 1 0 0	10	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	619
619	1 0 0 0 1 0 0	11	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	620
620	1 0 0 0 1 0 0	12	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	621
621	1 0 0 0 1 0 0	13	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	622
622	1 0 0 0 1 0 0	14	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	623
623	1 0 0 0 1 0 0	15	6	8	10	1	10	0	0	7	8	10	1	10	0	0	0	624
624	1 0 0 0 0 0 0	0	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	625
625	1 0 0 0 0 0 0	1	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	626
626	1 0 0 0 0 0 0	2	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	627
627	1 0 0 0 0 0 0	3	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	628
628	1 0 0 0 0 0 0	4	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	629
629	1 0 0 0 0 0 0	5	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	630
630	1 0 0 0 0 0 0	6	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	631
631	1 0 0 0 0 0 0	7	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	632
632	1 0 0 0 0 0 0	8	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	633
633	1 0 0 0 0 0 0	9	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	634
634	1 0 0 0 0 0 0	10	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	635
635	1 0 0 0 0 0 0	11	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	636
636	1 0 0 0 0 0 0	12	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	637
637	1 0 0 0 0 0 0	13	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	638
638	1 0 0 0 0 0 0	14	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	639
639	1 0 0 0 0 0 0	15	8	5	7	10	10	0	0	8	4	6	10	10	0	0	0	640
640	1 0 0 0 0 0 0	0	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	641
641	1 0 0 0 0 0 0	1	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	642
642	1 0 0 0 0 0 0	2	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	643
643	1 0 0 0 0 0 0	3	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	644
644	1 0 0 0 0 0 0	4	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	645
645	1 0 0 0 0 0 0	5	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	646
646	1 0 0 0 0 0 0	6	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	647
647	1 0 0 0 0 0 0	7	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	648
648	1 0 0 0 0 0 0	8	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	649
649	1 0 0 0 0 0 0	9	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	650
650	1 0 0 0 0 0 0	10	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	651
651	1 0 0 0 0 0 0	11	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	652
652	1 0 0 0 0 0 0	12	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	653
653	1 0 0 0 0 0 0	13	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	654
654	1 0 0 0 0 0 0	14	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	655
655	1 0 0 0 0 0 0	15	9	10	10	5	7	0	0	9	10	10	4	6	0	0	0	656
656	1 0 0 0 2 0 0	0	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	657
657	1 0 0 0 2 0 0	1	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	658
658	1 0 0 0 2 0 0	2	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	659
659	1 0 0 0 2 0 0	3	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	660
660	1 0 0 0 2 0 0	4	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	661
661	1 0 0 0 2 0 0	5	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	662
662	1 0 0 0 2 0 0	6	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	663
663	1 0 0 0 2 0 0	7	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	664

アドレス	オペコード	グループ	オペランド 0							オペランド 1							オペランド 2	next アドレス
664	1 0 0 0 2 0 0	8	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	665
665	1 0 0 0 2 0 0	9	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	666
666	1 0 0 0 2 0 0	10	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	667
667	1 0 0 0 2 0 0	11	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	668
668	1 0 0 0 2 0 0	12	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	669
669	1 0 0 0 2 0 0	13	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	670
670	1 0 0 0 2 0 0	14	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	671
671	1 0 0 0 2 0 0	15	5	8	10	9	10	0	0	4	8	10	9	10	0	0	0	672
672	0 0 0 0 0 0 0	0	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	673
673	0 0 0 0 0 0 0	1	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	674
674	0 0 0 0 0 0 0	2	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	675
675	0 0 0 0 0 0 0	3	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	676
676	0 0 0 0 0 0 0	4	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	677
677	0 0 0 0 0 0 0	5	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	678
678	0 0 0 0 0 0 0	6	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	679
679	0 0 0 0 0 0 0	7	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	680
680	0 0 0 0 0 0 0	8	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	681
681	0 0 0 0 0 0 0	9	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	682
682	0 0 0 0 0 0 0	10	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	683
683	0 0 0 0 0 0 0	11	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	684
684	0 0 0 0 0 0 0	12	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	685
685	0 0 0 0 0 0 0	13	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	686
686	0 0 0 0 0 0 0	14	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	687
687	0 0 0 0 0 0 0	15	8	14	10	8	9	0	0	0	0	0	0	0	0	0	0	688
688	1 0 0 0 2 0 1	0	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	689
689	1 0 0 0 2 0 1	1	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	690
690	1 0 0 0 2 0 1	2	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	691
691	1 0 0 0 2 0 1	3	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	692
692	1 0 0 0 2 0 1	4	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	693
693	1 0 0 0 2 0 1	5	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	694
694	1 0 0 0 2 0 1	6	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	695
695	1 0 0 0 2 0 1	7	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	696
696	1 0 0 0 2 0 1	8	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	697
697	1 0 0 0 2 0 1	9	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	698
698	1 0 0 0 2 0 1	10	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	699
699	1 0 0 0 2 0 1	11	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	700
700	1 0 0 0 2 0 1	12	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	701
701	1 0 0 0 2 0 1	13	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	702
702	1 0 1 0 2 0 1	14	7	8	9	15	10	0	0	6	8	9	15	10	0	0	0	703
703	1 1 0 0 2 0 1	15	7	8	9	15	10	0	0	6	8	9	15	10	0	0	544	704
704	0 0 0 4 0 0 0	0	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	705
705	0 0 0 4 0 0 0	1	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	706
706	0 0 0 4 0 0 0	2	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	707
707	0 0 0 4 0 0 0	3	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	708
708	0 0 0 4 0 0 0	4	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	709
709	0 0 0 4 0 0 0	5	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	710
710	0 0 0 4 0 0 0	6	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	711
711	0 0 0 4 0 0 0	7	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	712
712	0 0 0 4 0 0 0	8	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	713
713	0 0 0 4 0 0 0	9	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	714
714	0 0 0 4 0 0 0	10	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	715
715	0 0 0 4 0 0 0	11	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	716
716	0 0 0 4 0 0 0	12	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	717
717	0 0 0 4 0 0 0	13	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	718
718	0 0 0 4 0 0 0	14	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	719
719	0 0 0 4 0 0 0	15	8	5	7	4	6	0	0	0	0	0	0	0	0	0	0	720
720	3 0 0 0 0 0 0	0	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	721
721	3 0 0 0 0 0 0	1	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	722
722	3 0 0 0 0 0 0	2	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	723
723	3 0 0 0 0 0 0	3	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	724
724	3 0 0 0 0 0 0	4	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	725
725	3 0 0 0 0 0 0	5	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	726
726	3 0 0 0 0 0 0	6	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	727
727	3 0 0 0 0 0 0	7	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	728
728	3 0 0 0 0 0 0	8	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	729
729	3 0 0 0 0 0 0	9	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	730
730	3 0 0 0 0 0 0	10	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	731
731	3 0 0 0 0 0 0	11	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	732
732	3 0 0 0 0 0 0	12	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	733
733	3 0 0 0 0 0 0	13	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	734
734	3 0 0 0 0 0 0	14	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	735
735	3 0 0 0 0 0 0	15	9	4	6	5	7	0	0	5	4	10	11	10	0	0	0	736
736	0 0 0 0 1 0 0	0	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	737
737	0 0 0 0 1 0 0	1	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	738
738	0 0 0 0 1 0 0	2	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	739
739	0 0 0 0 1 0 0	3	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	740
740	0 0 0 0 1 0 0	4	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	741
741	0 0 0 0 1 0 0	5	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	742
742	0 0 0 0 1 0 0	6	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	743
743	0 0 0 0 1 0 0	7	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	744
744	0 0 0 0 1 0 0	8	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	745
745	0 0 0 0 1 0 0	9	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	746
746	0 0 0 0 1 0 0	10	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	747
747	0 0 0 0 1 0 0	11	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	748
748	0 0 0 0 1 0 0	12	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	749
749	0 0 0 0 1 0 0	13	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	750

アドレス	オペコード	グループ	オペランド 0							オペランド 1							オペランド 2	next アドレス
750	0 0 0 0 1 0 0	14	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	751
751	0 0 0 0 1 0 0	15	4	8	9	10	10	0	0	0	0	0	0	0	0	0	0	752
752	3 0 0 0 0 0 0	0	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	753
753	3 0 0 0 0 0 0	1	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	754
754	3 0 0 0 0 0 0	2	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	755
755	3 0 0 0 0 0 0	3	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	756
756	3 0 0 0 0 0 0	4	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	757
757	3 0 0 0 0 0 0	5	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	758
758	3 0 0 0 0 0 0	6	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	759
759	3 0 0 0 0 0 0	7	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	760
760	3 0 0 0 0 0 0	8	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	761
761	3 0 0 0 0 0 0	9	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	762
762	3 0 0 0 0 0 0	10	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	763
763	3 0 0 0 0 0 0	11	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	764
764	3 0 0 0 0 0 0	12	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	765
765	3 0 0 0 0 0 0	13	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	766
766	3 0 0 0 0 0 0	14	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	767
767	3 0 0 0 0 0 0	15	8	10	10	8	9	0	0	7	6	10	11	10	0	0	0	768
768	1 0 0 0 1 0 0	0	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	769
769	1 0 0 0 1 0 0	1	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	770
770	1 0 0 0 1 0 0	2	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	771
771	1 0 0 0 1 0 0	3	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	772
772	1 0 0 0 1 0 0	4	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	773
773	1 0 0 0 1 0 0	5	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	774
774	1 0 0 0 1 0 0	6	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	775
775	1 0 0 0 1 0 0	7	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	776
776	1 0 0 0 1 0 0	8	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	777
777	1 0 0 0 1 0 0	9	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	778
778	1 0 0 0 1 0 0	10	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	779
779	1 0 0 0 1 0 0	11	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	780
780	1 0 0 0 1 0 0	12	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	781
781	1 0 0 0 1 0 0	13	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	782
782	1 0 0 0 1 0 0	14	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	783
783	1 0 0 0 1 0 0	15	6	8	10	1	10	0	0	3	8	10	1	10	0	0	0	784
784	1 0 0 0 3 0 0	0	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	785
785	1 0 0 0 3 0 0	1	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	786
786	1 0 0 0 3 0 0	2	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	787
787	1 0 0 0 3 0 0	3	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	788
788	1 0 0 0 3 0 0	4	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	789
789	1 0 0 0 3 0 0	5	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	790
790	1 0 0 0 3 0 0	6	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	791
791	1 0 0 0 3 0 0	7	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	792
792	1 0 0 0 3 0 0	8	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	793
793	1 0 0 0 3 0 0	9	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	794
794	1 0 0 0 3 0 0	10	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	795
795	1 0 0 0 3 0 0	11	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	796
796	1 0 0 0 3 0 0	12	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	797
797	1 0 0 0 3 0 0	13	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	798
798	1 0 0 0 3 0 0	14	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	799
799	1 0 0 0 3 0 0	15	8	5	7	10	10	0	0	1	4	10	11	10	0	0	0	800
800	4 0 0 0 0 0 0	0	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	801
801	4 0 0 0 0 0 0	1	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	802
802	4 0 0 0 0 0 0	2	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	803
803	4 0 0 0 0 0 0	3	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	804
804	4 0 0 0 0 0 0	4	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	805
805	4 0 0 0 0 0 0	5	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	806
806	4 0 0 0 0 0 0	6	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	807
807	4 0 0 0 0 0 0	7	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	808
808	4 0 0 0 0 0 0	8	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	809
809	4 0 0 0 0 0 0	9	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	810
810	4 0 0 0 0 0 0	10	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	811
811	4 0 0 0 0 0 0	11	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	812
812	4 0 0 0 0 0 0	12	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	813
813	4 0 0 0 0 0 0	13	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	814
814	4 0 0 0 0 0 0	14	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	815
815	4 0 0 0 0 0 0	15	1	10	10	10	10	0	0	0	0	0	0	0	0	0	0	816
816	1 0 0 0 2 0 0	0	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	817
817	1 0 0 0 2 0 0	1	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	818
818	1 0 0 0 2 0 0	2	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	819
819	1 0 0 0 2 0 0	3	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	820
820	1 0 0 0 2 0 0	4	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	821
821	1 0 0 0 2 0 0	5	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	822
822	1 0 0 0 2 0 0	6	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	823
823	1 0 0 0 2 0 0	7	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	824
824	1 0 0 0 2 0 0	8	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	825
825	1 0 0 0 2 0 0	9	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	826
826	1 0 0 0 2 0 0	10	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	827
827	1 0 0 0 2 0 0	11	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	828
828	1 0 0 0 2 0 0	12	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	829
829	1 0 0 0 2 0 0	13	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	830
830	1 0 0 0 2 0 0	14	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	831
831	1 0 0 0 2 0 0	15	1	8	10	9	10	0	0	4	8	10	9	10	0	0	0	832
832	4 0 0 0 0 0 0	0	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	833
833	4 0 0 0 0 0 0	1	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	834
834	4 0 0 0 0 0 0	2	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	835
835	4 0 0 0 0 0 0	3	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	836

アドレス	オペコード	グループ	オペランド 0							オペランド 1							オペランド 2	next アドレス
836	4 0 0 0 0 0 0	4	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	837
837	4 0 0 0 0 0 0	5	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	838
838	4 0 0 0 0 0 0	6	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	839
839	4 0 0 0 0 0 0	7	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	840
840	4 0 0 0 0 0 0	8	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	841
841	4 0 0 0 0 0 0	9	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	842
842	4 0 0 0 0 0 0	10	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	843
843	4 0 0 0 0 0 0	11	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	844
844	4 0 0 0 0 0 0	12	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	845
845	4 0 0 0 0 0 0	13	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	846
846	4 0 0 0 0 0 0	14	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	847
847	4 0 0 0 0 0 0	15	3	10	10	10	10	0	0	0	0	0	0	0	0	0	0	848
848	1 0 0 0 2 0 1	0	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	849
849	1 0 0 0 2 0 1	1	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	850
850	1 0 0 0 2 0 1	2	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	851
851	1 0 0 0 2 0 1	3	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	852
852	1 0 0 0 2 0 1	4	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	853
853	1 0 0 0 2 0 1	5	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	854
854	1 0 0 0 2 0 1	6	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	855
855	1 0 0 0 2 0 1	7	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	856
856	1 0 0 0 2 0 1	8	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	857
857	1 0 0 0 2 0 1	9	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	858
858	1 0 0 0 2 0 1	10	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	859
859	1 0 0 0 2 0 1	11	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	860
860	1 0 0 0 2 0 1	12	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	861
861	1 0 0 0 2 0 1	13	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	862
862	1 0 0 0 2 0 1	14	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	863
863	1 0 0 0 2 0 1	15	3	8	9	15	10	0	0	6	8	9	15	10	0	0	0	864
864	0 0 0 0 0 0 0	0	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	865
865	0 0 0 0 0 0 0	1	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	866
866	0 0 0 0 0 0 0	2	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	867
867	0 0 0 0 0 0 0	3	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	868
868	0 0 0 0 0 0 0	4	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	869
869	0 0 0 0 0 0 0	5	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	870
870	0 0 0 0 0 0 0	6	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	871
871	0 0 0 0 0 0 0	7	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	872
872	0 0 0 0 0 0 0	8	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	873
873	0 0 0 0 0 0 0	9	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	874
874	0 0 0 0 0 0 0	10	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	875
875	0 0 0 0 0 0 0	11	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	876
876	0 0 0 0 0 0 0	12	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	877
877	0 0 0 0 0 0 0	13	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	878
878	0 0 0 0 0 0 0	14	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	879
879	0 0 0 0 0 0 0	15	4	2	10	3	10	0	0	0	0	0	0	0	0	0	0	880
880	1 0 0 3 0 0 0	0	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	881
881	1 0 0 3 0 0 0	1	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	882
882	1 0 0 3 0 0 0	2	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	883
883	1 0 0 3 0 0 0	3	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	884
884	1 0 0 3 0 0 0	4	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	885
885	1 0 0 3 0 0 0	5	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	886
886	1 0 0 3 0 0 0	6	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	887
887	1 0 0 3 0 0 0	7	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	888
888	1 0 0 3 0 0 0	8	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	889
889	1 0 0 3 0 0 0	9	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	890
890	1 0 0 3 0 0 0	10	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	891
891	1 0 0 3 0 0 0	11	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	892
892	1 0 0 3 0 0 0	12	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	893
893	1 0 0 3 0 0 0	13	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	894
894	1 0 0 3 0 0 0	14	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	895
895	1 0 0 3 0 0 0	15	4	4	10	11	10	0	0	5	4	10	11	10	0	0	0	896
896	1 0 0 0 0 0 0	0	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	897
897	1 0 0 0 0 0 0	1	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	898
898	1 0 0 0 0 0 0	2	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	899
899	1 0 0 0 0 0 0	3	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	900
900	1 0 0 0 0 0 0	4	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	901
901	1 0 0 0 0 0 0	5	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	902
902	1 0 0 0 0 0 0	6	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	903
903	1 0 0 0 0 0 0	7	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	904
904	1 0 0 0 0 0 0	8	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	905
905	1 0 0 0 0 0 0	9	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	906
906	1 0 0 0 0 0 0	10	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	907
907	1 0 0 0 0 0 0	11	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	908
908	1 0 0 0 0 0 0	12	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	909
909	1 0 0 0 0 0 0	13	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	910
910	1 0 0 0 0 0 0	14	5	11	10	11	10	0	0	4	4	10	4	10	0	0	0	911
911	1 0 2 0 0 0 0	15	5	11	10	11	10	0	0	4	4	10	4	10	0	0	95	912
912	1 0 0 4 0 0 0	0	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	913
913	1 0 0 4 0 0 0	1	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	914
914	1 0 0 4 0 0 0	2	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	915
915	1 0 0 4 0 0 0	3	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	916
916	1 0 0 4 0 0 0	4	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	917
917	1 0 0 4 0 0 0	5	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	918
918	1 0 0 4 0 0 0	6	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	919
919	1 0 0 4 0 0 0	7	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	920
920	1 0 0 4 0 0 0	8	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	921
921	1 0 0 4 0 0 0	9	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	922

アドレス	オペコード	グループ	オペランド 0							オペランド 1							オペランド 2	next アドレス
922	1 0 0 4 0 0 0	10	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	923
923	1 0 0 4 0 0 0	11	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	924
924	1 0 0 4 0 0 0	12	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	925
925	1 0 0 4 0 0 0	13	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	926
926	1 0 0 4 0 0 0	14	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	927
927	1 0 0 4 0 0 0	15	4	4	10	5	10	0	0	5	4	10	5	10	0	0	0	928
928	1 0 0 0 0 0 0	0	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	929
929	1 0 0 0 0 0 0	1	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	930
930	1 0 0 0 0 0 0	2	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	931
931	1 0 0 0 0 0 0	3	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	932
932	1 0 0 0 0 0 0	4	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	933
933	1 0 0 0 0 0 0	5	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	934
934	1 0 0 0 0 0 0	6	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	935
935	1 0 0 0 0 0 0	7	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	936
936	1 0 0 0 0 0 0	8	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	937
937	1 0 0 0 0 0 0	9	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	938
938	1 0 0 0 0 0 0	10	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	939
939	1 0 0 0 0 0 0	11	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	940
940	1 0 0 0 0 0 0	12	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	941
941	1 0 0 0 0 0 0	13	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	942
942	1 0 1 0 0 0 0	14	5	5	10	5	10	0	0	4	4	10	4	10	0	0	0	943
943	1 1 0 0 0 0 0	15	5	5	10	5	10	0	0	4	4	10	4	10	0	0	912	944
944	0 0 0 0 0 0 0	0	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	945
945	0 0 0 0 0 0 0	1	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	946
946	0 0 0 0 0 0 0	2	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	947
947	0 0 0 0 0 0 0	3	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	948
948	0 0 0 0 0 0 0	4	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	949
949	0 0 0 0 0 0 0	5	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	950
950	0 0 0 0 0 0 0	6	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	951
951	0 0 0 0 0 0 0	7	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	952
952	0 0 0 0 0 0 0	8	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	953
953	0 0 0 0 0 0 0	9	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	954
954	0 0 0 0 0 0 0	10	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	955
955	0 0 0 0 0 0 0	11	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	956
956	0 0 0 0 0 0 0	12	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	957
957	0 0 0 0 0 0 0	13	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	958
958	0 0 0 0 0 0 0	14	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	959
959	0 0 0 0 0 0 0	15	6	5	10	3	10	0	0	0	0	0	0	0	0	0	0	960
960	0 0 0 0 0 0 0	0	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	961
961	0 0 0 0 0 0 0	1	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	962
962	0 0 0 0 0 0 0	2	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	963
963	0 0 0 0 0 0 0	3	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	964
964	0 0 0 0 0 0 0	4	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	965
965	0 0 0 0 0 0 0	5	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	966
966	0 0 0 0 0 0 0	6	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	967
967	0 0 0 0 0 0 0	7	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	968
968	0 0 0 0 0 0 0	8	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	969
969	0 0 0 0 0 0 0	9	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	970
970	0 0 0 0 0 0 0	10	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	971
971	0 0 0 0 0 0 0	11	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	972
972	0 0 0 0 0 0 0	12	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	973
973	0 0 0 0 0 0 0	13	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	974
974	0 0 0 0 0 0 0	14	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	975
975	0 0 0 0 0 0 0	15	13	6	10	0	10	0	0	0	0	0	0	0	0	0	0	976
976	0 0 0 0 0 0 0	0	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	977
977	0 0 0 0 0 0 0	1	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	978
978	0 0 0 0 0 0 0	2	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	979
979	0 0 0 0 0 0 0	3	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	980
980	0 0 0 0 0 0 0	4	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	981
981	0 0 0 0 0 0 0	5	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	982
982	0 0 0 0 0 0 0	6	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	983
983	0 0 0 0 0 0 0	7	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	984
984	0 0 0 0 0 0 0	8	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	985
985	0 0 0 0 0 0 0	9	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	986
986	0 0 0 0 0 0 0	10	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	987
987	0 0 0 0 0 0 0	11	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	988
988	0 0 0 0 0 0 0	12	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	989
989	0 0 0 0 0 0 0	13	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	990
990	0 0 0 0 0 0 0	14	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	991
991	0 0 0 0 0 0 0	15	7	5	10	2	10	0	0	0	0	0	0	0	0	0	0	992
992	0 0 0 0 0 0 0	0	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	993
993	0 0 0 0 0 0 0	1	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	994
994	0 0 0 0 0 0 0	2	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	995
995	0 0 0 0 0 0 0	3	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	996
996	0 0 0 0 0 0 0	4	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	997
997	0 0 0 0 0 0 0	5	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	998
998	0 0 0 0 0 0 0	6	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	999
999	0 0 0 0 0 0 0	7	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	1000
1000	0 0 0 0 0 0 0	8	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	1001
1001	0 0 0 0 0 0 0	9	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	1002
1002	0 0 0 0 0 0 0	10	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	1003
1003	0 0 0 0 0 0 0	11	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	1004
1004	0 0 0 0 0 0 0	12	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	1005
1005	0 0 0 0 0 0 0	13	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	1006
1006	0 0 0 0 0 0 0	14	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	1007
1007	0 0 0 0 0 0 0	15	13	7	10	1	10	0	0	0	0	0	0	0	0	0	0	0

A.3　シミュレーション結果

暗号ハードウェアを設計する前にシミュレータをソフトウェアで実装したので、その結果を掲載する。なお、このシミュレータでは 16 段パイプラインを想定していた。$162 = 9 \times 18$ であるので、162 ビットの数字は 512 進数 18 桁で表すことができる。ここでは特に注釈が無い場合は 512 進数で数を表している。楕円曲線暗号の各情報は以下を使っている。

素数	$p = 2^{162} - 101$ (10 進数)
楕円曲線	$E : y^2 = x^3 + 3 \cdot 3456x^2 + x$ (係数は 10 進数)
ベースポイント	$x(B)$= 479 125 356 235 303 364 139 410 209 67 298 175 201 65 108 56 58 305
公開鍵	$x(P_k)$= 402 388 216 289 461 7 379 218 16 219 192 391 17 182 49 412 194 254
秘密鍵	s= 67 327 342 494 506 416 497 466 398 237 112 472 275 116 251 383 97 412

素数、楕円曲線、ベースポイントは 2.6 節で説明したものと同じである。なお乱数は C の乱数を使っている。

メッセージ																		
m_0	191	1	135	481	284	380	479	189	502	340	251	482	507	331	107	385	126	454
m_1	389	294	61	479	142	394	491	153	377	506	0	426	249	304	71	338	23	250
m_2	327	13	470	92	179	23	479	348	156	496	251	98	416	280	497	250	154	64
m_3	117	66	232	103	424	56	220	292	349	123	164	481	460	209	16	61	245	14
m_4	195	235	100	130	245	141	276	9	411	196	204	390	114	497	49	375	14	165
m_5	269	413	266	439	341	142	132	358	470	240	302	366	385	421	82	227	184	283
m_6	194	485	43	476	398	385	11	338	326	353	473	414	301	409	468	136	337	280
m_7	393	274	272	394	150	149	65	431	97	71	109	29	188	502	288	124	91	447
m_8	224	25	337	478	73	481	74	357	102	395	276	313	87	270	16	211	299	205
m_9	469	448	289	50	364	448	276	209	157	386	178	483	150	352	65	207	60	70
m_{10}	274	482	234	289	144	243	416	264	267	25	227	489	455	2	282	495	225	136
m_{11}	107	483	353	225	197	2	63	259	75	239	293	419	211	203	320	144	65	455
m_{12}	77	79	140	255	11	456	389	186	149	495	119	7	339	323	148	45	89	434
m_{13}	112	107	432	289	384	240	463	500	283	215	87	384	252	334	498	289	443	67
m_{14}	0	423	38	365	465	291	324	378	349	466	449	362	0	53	225	220	493	254
m_{15}	502	482	495	362	381	366	498	490	40	392	35	133	199	456	270	197	276	169
m_{16}	333	24	397	297	279	170	170	469	214	200	52	315	487	474	172	141	401	499
m_{17}	307	413	465	468	315	443	250	309	427	485	493	163	92	511	281	222	237	254
m_{18}	258	310	84	129	219	338	446	196	299	292	31	121	503	296	19	275	147	353
m_{19}	183	297	427	131	280	509	102	273	351	318	473	442	282	444	240	144	336	93
m_{20}	124	194	183	6	122	92	398	68	423	437	127	0	42	201	412	332	62	207
m_{21}	104	391	272	390	91	406	400	395	236	399	392	128	86	253	7	244	83	376
m_{22}	206	482	37	258	341	404	259	279	388	356	186	21	472	381	399	209	495	252
m_{23}	213	185	147	80	369	67	223	454	141	132	334	441	3	260	375	137	464	167
m_{24}	123	144	269	10	396	340	26	235	305	17	347	28	502	440	182	341	316	208
m_{25}	465	306	295	219	21	76	241	0	200	125	497	220	486	128	33	316	333	504
m_{26}	252	288	153	121	403	373	128	474	471	124	399	84	476	448	48	219	369	82
m_{27}	14	49	259	352	477	250	104	107	106	174	497	479	8	230	78	207	205	333
m_{28}	150	104	139	298	262	41	4	46	106	16	476	280	320	360	147	179	136	90
m_{29}	97	228	457	56	420	276	461	204	338	217	196	474	64	190	407	99	399	473
m_{30}	29	224	279	4	9	448	238	327	323	104	353	104	441	504	487	220	309	130
m_{31}	97	16	79	24	62	193	48	274	121	326	103	222	129	394	143	340	130	45

乱数値

r_0	47	384	161	504	199	95	119	431	253	341	62	504	382	466	410	480	159	369
r_1	97	201	271	247	419	354	378	422	222	254	256	106	318	204	17	468	261	446
r_2	81	387	245	279	44	389	503	471	39	124	62	408	360	70	124	190	294	272
r_3	29	144	314	25	490	14	55	73	87	158	425	120	243	52	132	15	189	131
r_4	48	442	409	32	317	163	197	2	230	433	51	97	284	380	140	221	387	297
r_5	67	231	194	365	469	163	289	89	373	316	75	347	352	233	148	312	430	70
r_6	48	377	138	503	99	352	130	468	337	344	246	231	331	230	245	34	84	198
r_7	98	196	324	98	293	293	144	235	408	145	411	135	175	125	328	31	22	495
r_8	56	6	212	247	274	248	146	345	153	354	453	78	149	451	260	52	458	435
r_9	117	240	72	140	347	112	69	52	167	224	300	376	421	344	16	179	399	17
r_{10}	68	376	314	328	164	60	488	66	66	390	184	506	241	384	326	379	440	162
r_{11}	26	504	472	184	177	128	271	448	402	443	457	232	436	434	464	36	16	241
r_{12}	19	147	419	63	386	498	97	174	293	251	413	385	468	464	421	11	150	236
r_{13}	28	26	492	72	224	60	115	509	70	437	405	480	63	83	380	328	238	400
r_{14}	0	105	393	347	244	200	465	94	343	244	368	218	256	13	184	183	123	191
r_{15}	125	376	379	474	351	219	380	378	266	98	8	417	177	498	67	305	197	42
r_{16}	83	134	99	202	197	426	298	373	181	306	13	78	505	502	299	35	228	252
r_{17}	76	487	244	245	78	494	446	333	234	505	251	168	407	127	454	183	315	191
r_{18}	64	333	277	32	182	468	367	305	74	457	7	414	253	458	4	452	420	472
r_{19}	45	458	234	416	454	127	153	324	215	463	374	238	326	367	60	36	84	23
r_{20}	31	48	301	385	286	279	99	273	105	493	159	384	10	306	231	83	15	307
r_{21}	26	97	452	97	278	485	356	98	443	99	482	32	21	319	129	445	20	478
r_{22}	51	376	265	192	341	229	64	453	481	89	46	261	246	95	227	436	251	447
r_{23}	53	174	164	404	92	144	439	497	291	161	83	366	128	449	93	418	244	41
r_{24}	30	420	67	130	355	85	6	314	460	132	214	391	125	366	45	341	207	52
r_{25}	116	204	329	438	389	147	60	128	50	31	252	183	121	288	8	207	83	254
r_{26}	63	72	38	158	228	477	160	118	373	415	99	405	119	112	12	54	476	148
r_{27}	3	268	192	472	119	190	282	26	410	299	380	247	386	57	275	307	435	211
r_{28}	37	282	34	458	321	266	129	11	282	260	119	70	80	90	36	428	418	22
r_{29}	24	185	114	142	105	69	115	179	84	310	177	118	272	47	357	408	483	502
r_{30}	7	184	69	385	2	240	59	337	464	410	88	154	110	254	121	439	77	160
r_{31}	24	132	19	390	15	304	140	68	286	209	281	439	288	226	291	469	32	267

暗号文 $x(C_1)$

$C_{1,0}$	228	318	36	297	469	184	447	23	173	56	494	82	30	69	231	202	80	249
$C_{1,1}$	97	8	452	464	359	119	402	103	22	299	430	23	200	8	26	453	222	175
$C_{1,2}$	325	143	385	52	458	281	188	52	331	8	36	483	62	488	145	392	469	290
$C_{1,3}$	91	317	50	375	135	389	209	419	265	439	86	55	299	457	280	143	106	278
$C_{1,4}$	233	496	347	298	212	17	157	484	164	180	20	25	20	214	358	392	481	224
$C_{1,5}$	464	472	377	411	309	471	216	487	296	133	366	428	94	462	131	44	239	392
$C_{1,6}$	13	257	427	212	37	417	294	340	28	243	59	302	301	92	383	255	454	330
$C_{1,7}$	434	263	386	497	360	509	387	179	171	181	502	32	456	164	468	278	20	347
$C_{1,8}$	107	417	418	106	505	62	450	14	307	404	288	278	87	406	231	123	374	470
$C_{1,9}$	131	361	341	31	119	267	498	148	367	371	440	132	157	419	206	178	36	357
$C_{1,10}$	210	5	22	73	318	378	445	133	431	421	403	166	511	212	508	379	428	227
$C_{1,11}$	220	322	7	330	213	218	300	95	151	95	206	243	332	130	153	277	439	100
$C_{1,12}$	268	183	222	436	502	84	119	47	235	344	297	157	28	237	502	433	41	315
$C_{1,13}$	416	399	332	418	249	4	410	263	402	469	393	290	35	228	19	400	80	368
$C_{1,14}$	439	151	272	297	440	85	341	320	451	272	321	449	314	398	28	289	74	272
$C_{1,15}$	364	113	308	117	96	134	1	23	262	0	471	54	471	136	122	62	490	377
$C_{1,16}$	302	338	294	276	162	103	292	122	297	337	69	287	47	99	280	200	114	134
$C_{1,17}$	9	76	215	225	45	467	82	247	126	463	175	507	163	328	261	240	325	478
$C_{1,18}$	464	324	181	246	191	287	203	472	305	278	55	289	186	207	41	306	385	5
$C_{1,19}$	207	71	59	80	225	358	9	266	19	461	5	218	500	460	509	398	85	463
$C_{1,20}$	464	472	281	337	150	75	99	366	299	172	429	55	73	305	132	350	428	1
$C_{1,21}$	38	208	344	454	72	362	303	400	407	111	118	399	116	404	103	219	82	211
$C_{1,22}$	138	433	188	473	157	285	191	294	350	395	195	64	308	123	303	427	377	202
$C_{1,23}$	447	470	48	15	430	92	131	192	27	222	33	362	221	253	205	483	346	220
$C_{1,24}$	64	294	284	333	318	22	48	496	9	339	330	385	371	402	320	77	164	511
$C_{1,25}$	119	135	168	53	286	32	338	269	500	155	315	68	478	287	64	233	426	223
$C_{1,26}$	291	286	231	468	129	104	199	335	94	264	192	369	343	362	496	170	145	391
$C_{1,27}$	58	153	474	467	304	182	77	502	180	127	423	491	71	127	457	446	444	259
$C_{1,28}$	48	75	486	500	295	445	363	301	451	458	192	43	4	424	427	283	250	372
$C_{1,29}$	102	184	491	202	187	238	95	501	343	222	458	434	113	19	421	165	430	468
$C_{1,30}$	508	255	100	205	281	36	108	168	326	72	396	332	357	120	57	259	208	267
$C_{1,31}$	273	432	455	269	287	370	260	346	390	160	5	71	229	277	87	71	353	111

暗号文 c_2

$c_{2,0}$	441	163	102	265	57	417	197	92	327	367	196	14	190	404	51	101	12	161
$c_{2,1}$	76	363	214	383	218	216	450	128	400	172	80	224	421	292	293	139	309	122
$c_{2,2}$	311	467	481	369	107	204	482	211	412	348	14	75	234	93	85	295	337	465
$c_{2,3}$	423	408	64	423	469	50	12	405	177	85	467	243	192	395	138	330	40	164
$c_{2,4}$	363	315	135	287	172	262	298	383	73	310	265	181	150	152	343	409	62	433
$c_{2,5}$	457	284	41	341	239	435	329	72	58	431	303	432	489	132	210	476	227	333
$c_{2,6}$	431	128	382	287	382	273	119	365	442	269	230	94	62	416	282	400	24	394
$c_{2,7}$	241	330	421	214	301	52	313	320	353	395	230	40	90	444	117	270	345	96
$c_{2,8}$	162	300	96	320	418	105	268	445	369	304	289	48	463	102	213	255	309	458
$c_{2,9}$	472	257	54	276	441	122	369	214	265	50	372	475	290	118	312	87	350	107
$c_{2,10}$	403	501	428	426	476	95	8	185	357	394	124	291	103	415	235	57	125	264
$c_{2,11}$	102	92	64	88	245	180	302	329	213	294	463	113	188	484	324	222	92	33
$c_{2,12}$	426	276	309	155	213	359	234	33	165	473	17	153	485	472	320	387	313	407
$c_{2,13}$	295	59	142	294	329	66	470	43	116	142	64	218	393	35	366	50	293	289
$c_{2,14}$	220	510	457	427	206	442	44	188	349	66	252	285	172	81	26	373	486	255
$c_{2,15}$	16	26	136	458	103	352	244	85	318	224	87	419	56	269	325	397	457	84
$c_{2,16}$	213	0	252	473	13	235	437	331	477	68	257	281	416	247	457	95	113	207
$c_{2,17}$	136	120	118	111	263	111	427	332	294	298	221	125	297	115	443	234	357	127
$c_{2,18}$	154	127	390	275	340	397	50	33	397	86	163	238	342	38	110	24	77	325
$c_{2,19}$	281	452	54	462	179	54	388	444	184	222	289	355	455	307	421	416	292	30
$c_{2,20}$	394	368	145	390	143	8	180	26	328	39	453	26	292	222	17	54	419	229
$c_{2,21}$	272	106	210	69	453	389	248	15	209	150	245	254	388	184	490	74	500	56
$c_{2,22}$	380	152	134	316	433	504	222	340	82	192	96	42	41	246	464	254	116	21
$c_{2,23}$	6	70	177	27	137	249	94	131	104	27	482	8	477	467	221	421	33	112
$c_{2,24}$	314	156	290	196	57	64	435	146	89	428	126	175	462	73	294	469	479	294
$c_{2,25}$	255	391	253	252	288	38	338	474	171	398	231	74	479	118	150	129	47	304
$c_{2,26}$	327	376	231	367	218	94	427	360	187	196	203	262	76	225	493	34	323	37
$c_{2,27}$	463	491	294	355	12	499	273	84	376	12	215	176	239	242	191	310	122	319
$c_{2,28}$	327	144	212	175	38	321	403	494	114	106	157	444	191	45	37	24	491	412
$c_{2,29}$	275	387	114	323	280	121	310	339	367	272	3	53	448	190	503	371	243	18
$c_{2,30}$	329	135	228	15	442	442	461	12	319	200	477	503	204	223	225	341	322	156
$c_{2,31}$	193	159	95	262	450	260	238	69	374	307	439	308	109	388	70	296	354	334

復号文

d_0	191	1	135	481	284	380	479	189	502	340	251	482	507	331	107	385	126	454
d_1	389	294	61	479	142	394	491	153	377	506	0	426	249	304	71	338	23	250
d_2	327	13	470	92	179	23	479	348	156	496	251	98	416	280	497	250	154	64
d_3	117	66	232	103	424	56	220	292	349	123	164	481	460	209	16	61	245	14
d_4	195	235	100	130	245	141	276	9	411	196	204	390	114	497	49	375	14	165
d_5	269	413	266	439	341	142	132	358	470	240	302	366	385	421	82	227	184	283
d_6	194	485	43	476	398	385	11	338	326	353	473	414	301	409	468	136	337	280
d_7	393	274	272	394	150	149	65	431	97	71	109	29	188	502	288	124	91	447
d_8	224	25	337	478	73	481	74	357	102	395	276	313	87	270	16	211	299	205
d_9	469	448	289	50	364	448	276	209	157	386	178	483	150	352	65	207	60	70
d_{10}	274	482	234	289	144	243	416	264	267	25	227	489	455	2	282	495	225	136
d_{11}	107	483	353	225	197	2	63	259	75	239	293	419	211	203	320	144	65	455
d_{12}	77	79	140	255	11	456	389	186	149	495	119	7	339	323	148	45	89	434
d_{13}	112	107	432	289	384	240	463	500	283	215	87	384	252	334	498	289	443	67
d_{14}	0	423	38	365	465	291	324	378	349	466	449	362	0	53	225	220	493	254
d_{15}	502	482	495	362	381	366	498	490	40	392	35	133	199	456	270	197	276	169
d_{16}	333	24	397	297	279	170	170	469	214	200	52	315	487	474	172	141	401	499
d_{17}	307	413	465	468	315	443	250	309	427	485	493	163	92	511	281	222	237	254
d_{18}	258	310	84	129	219	338	446	196	299	292	31	121	503	296	19	275	147	353
d_{19}	183	297	427	131	280	509	102	273	351	318	473	442	282	444	240	144	336	93
d_{20}	124	194	183	6	122	92	398	68	423	437	127	0	42	201	412	332	62	207
d_{21}	104	391	272	390	91	406	400	395	236	399	392	128	86	253	7	244	83	376
d_{22}	206	482	37	258	341	404	259	279	388	356	186	21	472	381	399	209	495	252
d_{23}	213	185	147	80	369	67	223	454	141	132	334	441	3	260	375	137	464	167
d_{24}	123	144	269	10	396	340	26	235	305	17	347	28	502	440	182	341	316	208
d_{25}	465	306	295	219	21	76	241	0	200	125	497	220	486	128	33	316	333	504
d_{26}	252	288	153	121	403	373	128	474	471	124	399	84	476	448	48	219	369	82
d_{27}	14	49	259	352	477	250	104	107	106	174	497	479	8	230	78	207	205	333
d_{28}	150	104	139	298	262	41	4	46	106	16	476	280	320	360	147	179	136	90
d_{29}	97	228	457	56	420	276	461	204	338	217	196	474	64	190	407	99	399	473
d_{30}	29	224	279	4	9	448	238	327	323	104	353	104	441	504	487	220	309	130
d_{31}	97	16	79	24	62	193	48	274	121	326	103	222	129	394	143	340	130	45