# アセンブラ短歌

坂井弘亮
(KOZOSプロジェクト)
TwitterID:kozossakai

まず最初に
紹介

# SoftwareDesign誌
# 懐かしの連載
# 「全国津々浦々!
# 勉強会&イベント
# 探訪記」

電子書籍になります！

電子書籍に
~~なります!~~
した!

達人出版会

# OSC東京も収録されてます！(2011年春・秋)

全国津々浦々！
勉強会＆イベント
探訪記　完全版

坂井弘亮

北は北海道から南は
九州まで、各地の
IT系イベントを写真
と記録で振り返る。

# さて本題

# SECCON 2013

# SECCONとは何か

セキュリティ競技のコンテストです

昨年は**CTF**という
競技大会が
メインでしたが

今年のSECCONでは
様々なコンテストが
行われるらしい

10月の長野大会では
アセンブラ短歌の
コンテストと品評会が
行われました

# アセンブラ短歌？

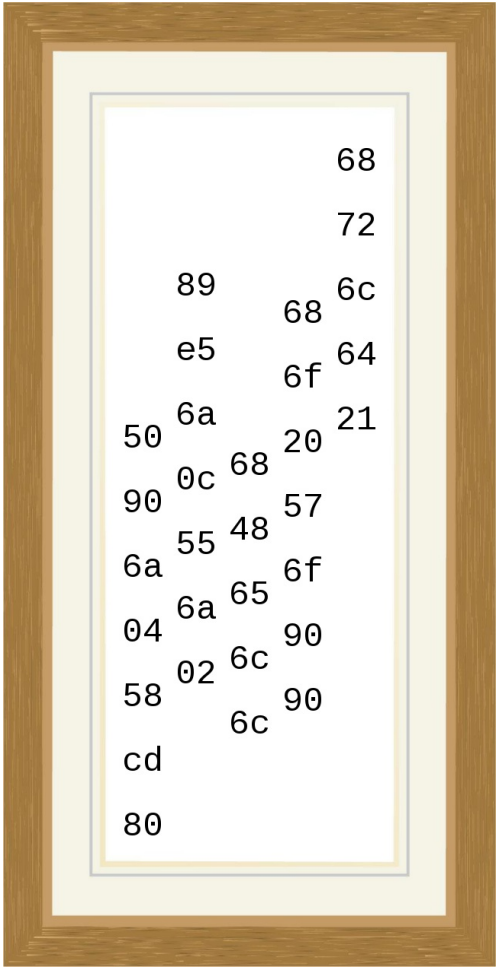
# アセンブラ短歌とは

5・7・5・7・7の機械語
コードでプログラムを
書いてみるという
近未来の文化的趣味

こんな感じです

```
68 72 6c 64 21
68 6f 20 57 6f 90 90
68 48 65 6c 6c
89 e5 6a 0c 55 6a 02
50 90 6a 04 58 cd 80
```

札に詠むと
こんな感じ



# 実行結果

# Hello World!

ちなみに
他の言語では

「XX短歌」は
他の言語では
難しい

スクリプト言語では
これが邪魔

#!/bin/sh
(9文字)

C言語では
これが邪魔

```
#include <stdio.h>
```

(18文字)

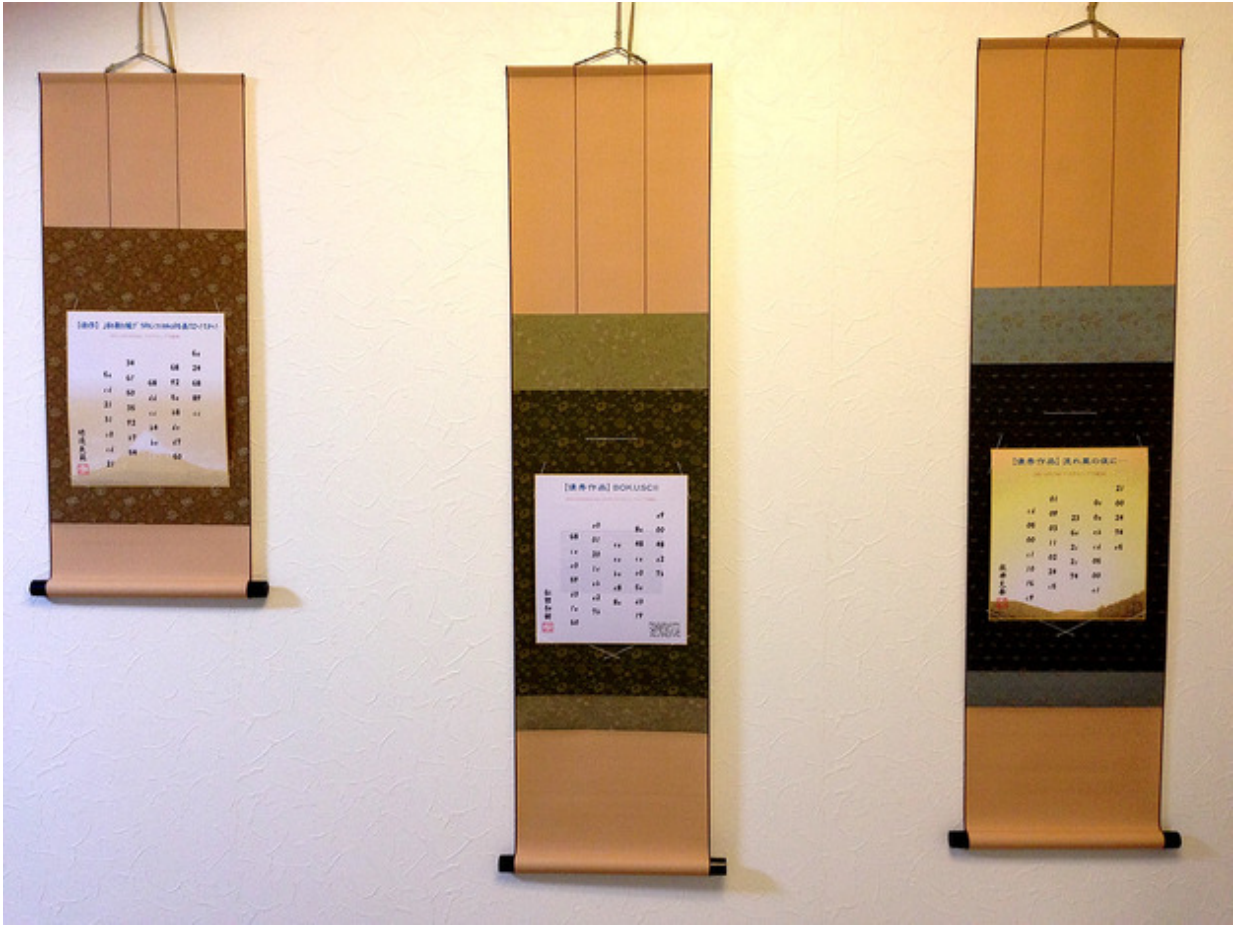
アセンブラは
もっとも短歌に向いた
高級（な）言語です

ところで

# SECCON
# 長野大会の
# 品評会で
# 披露されたもの

# アセンブラ短歌作品の

# 掛軸



拡大すると

# タイトル「新潟」

6e
34
68
24
5a
6f
68
92
68
cd
50
dd
5a
89
21
35
cc
b8
cc
31
92
b4
de
竹迫爽範
c0
b7
bc
d7
cd
54
50
21

ちなみに

アセンブラ短歌も
近日中に
書籍化されます！
(オンデマンド&電子)

# 31バイトでつくる
# アセンブラプログラミング
# アセンブラ短歌の世界

そんな
アセンブラ短歌
ですが

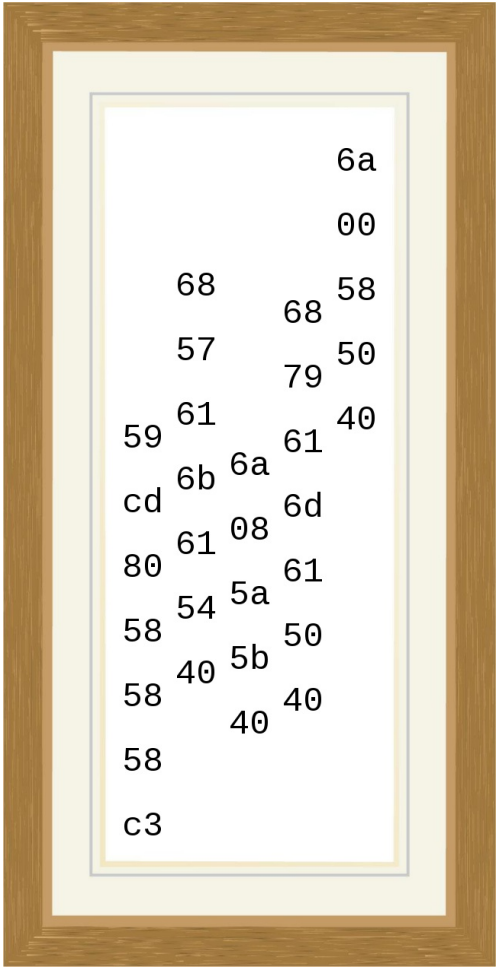
いちおう
セキュリティ教育の
目的があります
(詳しくはHP参照)

http://kozos.jp/asm-tanka/

では

# 一首
# 鑑賞して
# みましょう

# タイトル
# 「夏休み」
## 詠み人：私


6a
00
58
68
50
40
68
57
79
61
59
61
6b
6a
cd
6d
08
61
80
61
54
5a
58
50
40
5b
58
40
40
58
c3

# 鑑賞ポイント

# アセンブラを見ると

上の句

```
push    $0
pop     %eax
push    %eax
inc     %eax

push    $0x616d6179
push    %eax
inc     %eax

push    $8
pop     %edx
pop     %ebx
inc     %eax
```

下の句

```
push    $0x616b6157
push    %esp
inc     %eax

pop     %ecx
int     $0x80
pop     %eax
pop     %eax
pop     %eax
ret
```

# アセンブラを見ると

各句の末尾に同じ命令が

上の句

```
push    $0
pop     %eax
push    %eax
inc     %eax

push    $0x616d6179
push    %eax
inc     %eax

push    $8
pop     %edx
pop     %ebx
inc     %eax
```

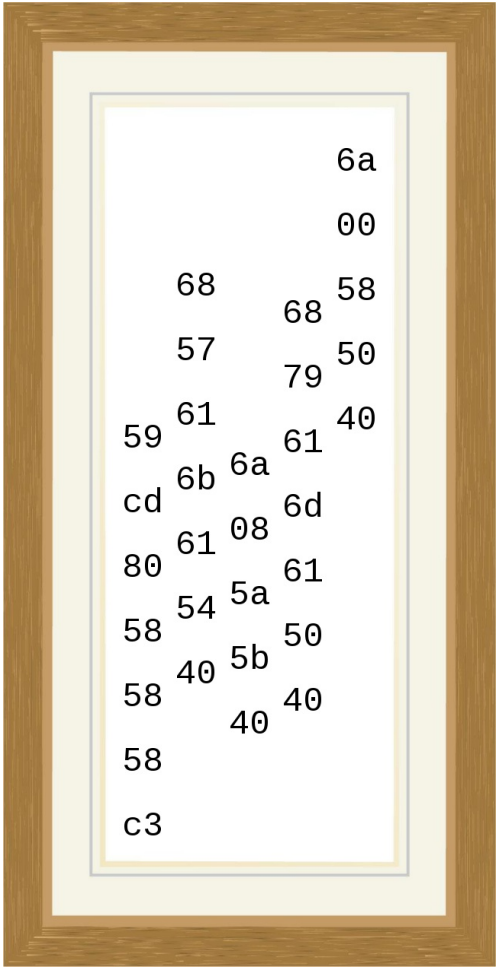
下の句

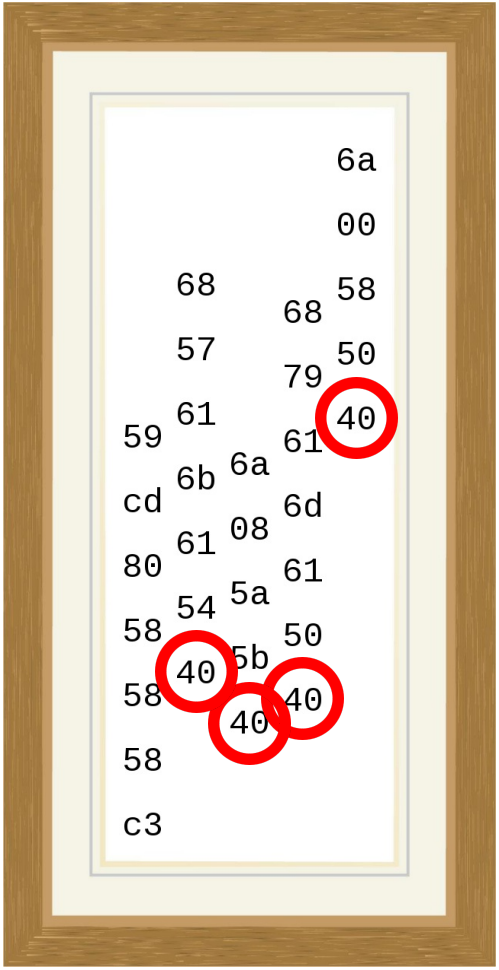
```
push    $0x616b6157
push    %esp
inc     %eax

pop     %ecx
int     $0x80
pop     %eax
pop     %eax
pop     %eax
ret
```

つまり

韻をふんでいる
わけです


6a
00
58
68
50
40
68
57
79
61
59
61
6b
6a
cd
6d
08
61
80
61
54
5a
58
50
40
5b
58
40
40
58
c3


6a
00
58
50
40
68
57
61
6b
61
54
40
68
79
61
6d
61
50
40
59
cd
80
58
58
c3
6a
08
5a
5b
40

**0x40で**
**韻をふむ**

# 他にも

```
push    $0
pop     %eax
push    %eax
inc     %eax

push    $0x616d6179
push    %eax
inc     %eax

push    $8
pop     %edx
pop     %ebx
inc     %eax
```

```
push    $0x616b6157
push    %esp
inc     %eax

pop     %ecx
int     $0x80
pop     %eax
pop     %eax
pop     %eax
ret
```

# EAXの値を流用してレジスタ設定している

```
push    $0
pop     %eax
push    %eax
inc     %eax

push    $0x616d6179
push    %eax
inc     %eax

push    $8
pop     %edx
pop     %ebx
inc     %eax
```

```
push    $0x616b6157
push    %esp
inc     %eax

pop     %ecx
int     $0x80
pop     %eax
pop     %eax
pop     %eax
ret
```

まるで

流れる小川の
せせらぎのよう

命令に注目
すると

```
push    $0
pop     %eax
push    %eax
inc     %eax

push    $0x616d6179
push    %eax
inc     %eax

push    $8
pop     %edx
pop     %ebx
inc     %eax
```

```
push    $0x616b6157
push    %esp
inc     %eax

pop     %ecx
int     $0x80
pop     %eax
pop     %eax
pop     %eax
ret
```

# 命令のリズムもある

```
push   $0
pop    %eax
push   %eax
inc    %eax

push   $0x616d6179
push   %eax
inc    %eax

push   $8
pop    %edx
pop    %ebx
inc    %eax
```

```
push   $0x616b6157
push   %esp
inc    %eax

pop    %ecx
int    $0x80
pop    %eax
pop    %eax
pop    %eax
ret
```

# 総評

最初にゼロ設定した**EAX**が**inc**命令で1ずつ増加していく際に, その値を流用して **EBX**や戻り値の**EAX**を設定していくさまは，流れる小川のせせらぎ のようです. またニーモニックを見ると「**push-pop-push-inc, push-push-inc**」「**push-pop-pop-inc, push-push-inc**」というスキップするようなリズム があり, 小川のほとりで遊ぶ 子供たちの情景 を思い起こさせ,「夏休み」というテーマがよく表現されています.

# 総評

最初にゼロ設定したEAXがinc命令で1ずつ増加していく際に，その値を流用して EBXや戻り値のEAXを設定していくさまは，流れる小川のせせらぎ のようです．またニーモニックを見ると「push-pop-push-inc, push-push-inc」「push-pop-pop-inc, push-push-inc」というスキップするようなリズム があり，小川のほとりで遊ぶ 子供たちの情景 を思い起こさせ，「夏休み」というテーマがよく表現されています．

# 総評

最初にゼロ設定したEAXがinc命令で1ずつ増加していく際に，その値を流用して EBXや戻り値のEAXを設定していくさまは，流れる小川のせせらぎ のようです．またニーモニックを見ると「push-pop-push-inc, push-push-inc」「push-pop-pop-inc, push-push-inc」というスキップするようなリズム があり，小川のほとりで遊ぶ 子供たちの情景 を思い起こさせ，「夏休み」というテーマがよく表現されています．

# 総評

最初にゼロ設定したEAXがinc命令で1ずつ増加していく際に，その値を流用して EBXや戻り値のEAXを設定していくさまは，流れる小川のせせらぎ のようです．またニーモニックを見ると「push-pop-push-inc, push-push-inc」「push-pop-pop-inc, push-push-inc」というスキップするようなリズム があり，小川のほとりで遊ぶ 子供たちの情景 を思い起こさせ，「夏休み」というテーマがよく表現されています．

# 総評

最初にゼロ設定したEAXがinc命令で1ずつ増加していく際に，その値を流用して EBXや戻り値のEAXを設定していくさまは，流れる小川のせせらぎ のようです．またニーモニックを見ると「push-pop-push-inc, push-push-inc」「push-pop-pop-inc, push-push-inc」というスキップするようなリズム があり，小川のほとりで遊ぶ 子供たちの情景 を思い起こさせ，「夏休み」というテーマがよく表現されています．

ここからが
今日の本題

ふだんは趣味で
こんなことを
やっています

GCCが対応している
CPUアーキテクチャの
アセンブラを
片っ端から出力させて
片っ端から読んでみる

この技術を
アセンブラ短歌に
活かしたい

# アセンブラ短歌にもっとも向いたアーキテクチャは何か？

RISC系プロセッサ

→ 4バイト固定長命令

可変長命令でも

→ 偶数バイト命令が多

原理的に不可能

以下のアーキで
原理的に可能な
ことが判明

**x86(Intel) M32C(**三菱**)**
**MN10300(**松下**)**
**RL78(**ルネサス**) RX(**ルネサス**)**
**Xtensa(**テンシリカ**)**

# やってみた

まず、RX

# こんな感じか

```
mov.l #1, r1
mov.l #10, r3
nop

mov.l #0x6c6c6548, [r0]
nop

mov.l #0x6c72, 8[r0]
```

上の句

下の句

```
mov.l #0x6f57206f, 4[r0]

mov.l r0, r2
mov.l #5, r5
int #255
```

# 機械語コードは
# こんな感じ

66

11

66

a3

03

f8

02

48

65

6c

6c

03

f9

02

01

6f

20

57

6f

6c

f9

0a

02

72

ef

02

66

55

75

60

ff

# 実行結果は

Hello Worl

(10文字)

次、RL78

# こんな感じ

```
mov 0xfff10, #72
mov a, #108

mov 0xfff10, #101
mov 0xfff10, a
mov 0xfff10, a

mov 0xfff10, #111
mov a, #32
```

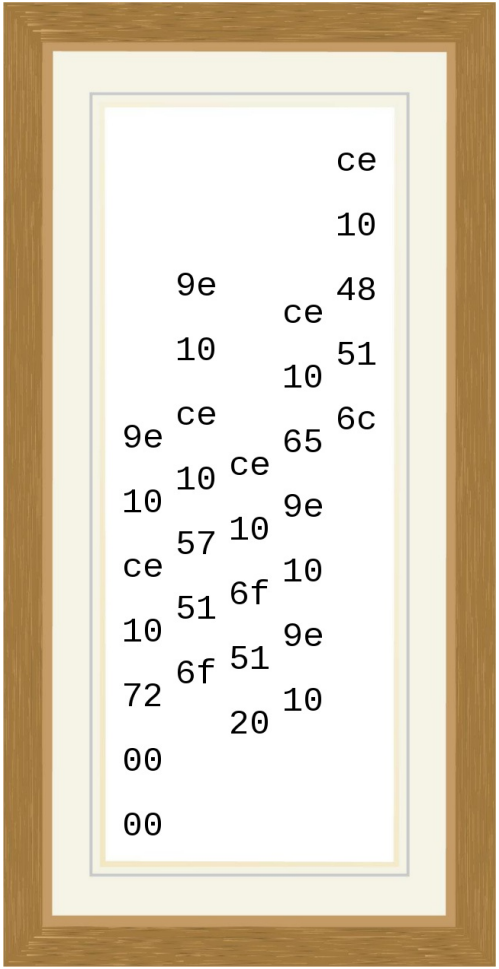
上の句

下の句

```
mov 0xfff10, a
mov 0xfff10, #87
mov a, #111

mov 0xfff10, a
mov 0xfff10, #114
nop
nop
```

こんな


9e 10 ce 10 57 51 6f
9e 10 ce 10 6f 51 20
ce 10 65 9e 10 9e 10
ce 10 48 51 6c
00
00
72

かんじ

# 実行結果は

Hello Wor

(9文字)

他のは
こんな感じ

# MN10300

Hello Wo (8文字)

# M32C

Hello!　　(6文字)

# i386/FreeBSD

Hello World! (12文字)

# x86/Linux

Hello World!! (13文字)

# 比較してみる

# 比較の前に

定量化の
ためには
単位が必要

# 単位を制定する

## BPT (Byte per Tanka)

## ...アセンブラ短歌の
## やりやすさの指標

たとえばRXでは

**hello Worl**

で10文字なので

10BPT

集計すると

| | |
|---|---|
| M32C | 6BPT |
| MN10300 | 8BPT |
| RL78 | 9BPT |
| RX | 10BPT |
| i386/FreeBSD | 12BPT |
| x86/Linux | 13BPT |

# x86は非常に短歌に向いていることが判明(あとLinuxも)

どうもありがとう
ございました