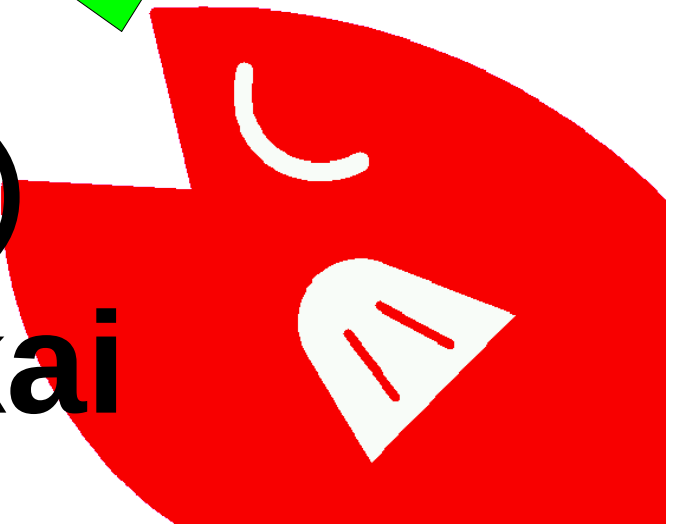
アセンブラ短歌

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



ええっと
いろんなところで
紹介させていた
だいているのですが

こんな本を
書いています

フィーリングで読む
アセンブラ入門

どんな内容？

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

(注)
フィーリングで

対象アーキテクチャ

４０種類

Alpha ARC ARM ARM(Thumb) AVR AVR(8bit) CRIS FR30 FR-V H8/300 H8/300H HP/PA i386 i960 IA-64 M32R 68HC11 68HC11(16bit) M68000 MCORE MIPS MIPS16 MIPS64 MIST32 MMIX MN10300 PDP-11 PowerPC PowerPC64 S/390 SH SH64 SPARC StrongARM V850 VAX x86-64 XScale Xstormy Xtensa

Alpha ARC ARM ARM(Thumb) AVR AVR(8bit) CRIS FR30 FR-V H8/300 H8/300H HP/PA i386 i960 IA-64 M32R 68HC11 68HC11(16bit) M68000 MCORE MIPS MIPS16 MIPS64 MIST32 MMIX MN10300 PDP-11 PowerPC PowerPC64 S/390 SH SH64 SPARC StrongARM V850 VAX x86-64 XScale Xstormy Xtensa

不安な点

これだけだと
物足りない
ですよね

ということで
追加で

いくつかのアーキテクチャでは
シミュレータ対応をして
GDBのシミュレータを使って
Hello World を
動かしてみる

対象アーキテクチャ

18種類

Alpha ARC ARM ARM(Thumb) AVR AVR(8bit) CRIS FR30 FR-V H8/300 H8/300H HP/PA i386 i960 IA-64 M32R 68HC11 68HC11(16bit) M68000 MCORE MIPS MIPS16 MIPS64 MIST32 MMIX MN10300 PDP-11 PowerPC PowerPC64 S/390 SH SH64 SPARC StrongARM V850 VAX x86-64 XScale Xstormy Xtensa

本日は、
ダイジェストとして

いくつか
ピックアップして
見てみましょう

40種類のreturn命令

Alpha	01 80 fa 6b	ret
ARC	20 80 0f 38	j.d [blink]
ARM	e1a0f00e	mov pc, lr
Thumb	4770	bx lr
AVR	08 95	ret
AVR(8)	08 95	ret
CRIS	7fb6	ret
FR30	97 20	ret
FR-V	c0 3a 40 00	bralr
H8/300	54 70	rts
H8/300H	54 70	rts
HP/PA	e8 40 c0 02	bv,n r0(rp)
i386	c3	ret
i960	00 00 00 0a	ret
IA-64	08 00 84 00	br.ret.sptk.many b0;;
M32R	1f ce f0 00	jmp lr || nop
68HC11	39	rts
68HC11(16)	39	rts
M68000	4e75	rts
MCORE	00cf	jmp r15
MIPS	03e00008	jr ra
MIPS16	e820	jr ra
MIPS64	03e00008	jr ra
MIST32	14 40 03 e0	b rret,#al
MMIX	f8000000	pop 0,0
MN10300	f0 fc	rets
PDP-11	0087	rts pc
PowerPC	4e 80 00 20	blr
PowerPC64	4e 80 00 20	blr
S/390	07 fe	br %r14
SH	00 0b	rts
SH64	4401fff0	blink tr0,r63
SPARC	81 c3 e0 08	retl
StrongARM	e1a0f00e	mov pc, lr
V850	7f 00	jmp [lp
VAX	04	ret
x86-64	f3 c3	repz retq
XScale	e12fff1e	bx lr
Xstormy	03 00	ret
Xtensa	1df0	retw.n

18種類のHello World

Hello World! abadface This architecture is i386-elf
Hello World! abadface This architecture is arm-elf
Hello World! abadface This architecture is arm16-elf
Hello World! abadface This architecture is avr-elf
Hello World! abadface This architecture is cris-elf
Hello World! abadface This architecture is frv-elf
Hello World! abadface This architecture is h8300-elf
Hello World! abadface This architecture is m32r-elf
Hello World! abadface This architecture is m6811-elf
Hello World! abadface This architecture is mcore-elf
Hello World! abadface This architecture is mips-elf
Hello World! abadface This architecture is mips16-elf
Hello World! abadface This architecture is mn10300-elf
Hello World! abadface This architecture is powerpc-elf
Hello World! abadface This architecture is sh-elf
Hello World! abadface This architecture is sh64-elf
Hello World! abadface This architecture is sparc-elf
Hello World! abadface This architecture is v850-elf

どうですか

いきなり
おなかいっぱい
かもしれませんが

アセンブラの
流行の兆しが
見えてきたのでは
ないでしょうか

近年、若者を中心に
流行している
アセンブラですが

今のうちに
先取りしておこうと
いうことで

今回は
若者向けに
こんなものを
やってみました

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
```

実行結果

Hello World!

それならば、
これもできるだろうと
いうことで

アセンブラ川柳

```
6a 48 89 e5 90
6a 01 55 6a 02 50 90
6a 04 58 cd 80
```

実行結果

H

1文字が
せいいっぱいでした

下の句が無いと、
けっこう難しい

ちなみに

「XX短歌」は
他の言語では
難しい

スクリプト言語は
これが邪魔

#!/bin/sh

(9文字)

C言語は
これが邪魔

#include <stdio.h>

(18文字)

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

アセンブラ短歌ができるのは
実は、以下くらいしか
ありませんでした

x86、68HC11、
MN10300、
VAX、Xtensa

まてよ

ということは

こういうのも
可能ということか

アセンブラ
かるた

作ってみた

```
68 a5 f3 a5 b0
68 a1 bc a5 ea 90 90
68 a5 d5 a5 a3
89 e5 6a 0c 55 6a 02
50 90 6a 04 58 cd 80
```
実行 → フィーリング

```
68 00 00 a5 a2
68 a5 bb a5 f3 90 90
68 a5 d6 a5 e9
89 e5 6a 0c 55 6a 02
50 90 6a 04 58 cd 80
```
実行 → ブラセンア

しかし問題あり

バイナリダンプを見ると

```
00000000  68 a5 f3 a5 b0 68 a1 bc  |hングh一|
00000008  a5 ea 90 90 68 a5 d5 a5  |リ..hフ|
00000010  a3 89 e5 6a 0c 55 6a 02  |.j.Uj.|
00000018  50 90 6a 04 58 cd 80     |P.j.X.|

00000000  68 00 00 a5 a2 68 a5 bb  |h..アhセ|
00000008  a5 f3 90 90 68 a5 d6 a5  |ン..hブ|
00000010  e9 89 e5 6a 0c 55 6a 02  |.j.Uj.|
00000018  50 90 6a 04 58 cd 80     |P.j.X.|
```

```
00000000  68 a5 f3 a5 b0 68 a1 bc  |hングh一|
00000008  a5 ea 90 90 68 a5 d5 a5  |リ..hフ|
00000010  a3 89 e5 6a 0c 55 6a 02  |.j.Uj.|
00000018  50 90 6a 04 58 cd 80     |P.j.X.|

00000000  68 00 00 a5 a2 68 a5 bb  |h..アhセ|
00000008  a5 f3 90 90 68 a5 d6 a5  |ン..hブ|
00000010  e9 89 e5 6a 0c 55 6a 02  |.j.Uj.|
00000018  50 90 6a 04 58 cd 80     |P.j.X.|
```

つまりバイトコードを
読み上げた時点で
キーワードが
わかってしまう

これでは
競技にできない
(＝流行しない)

xorにかけて難読化してみる

```
b8 5b 0c 5a 29
f7 d8 50 b8 5b 5d 5a 44  (字余り)
f7 d8 50 89 e5
6a 08 55 6a 02 50 90
90 90 6a 04 58 cd 80
```

実行結果

アセンブ

４文字が
せいいっぱい
でした

バイナリダンプは

```
00000000  b8 5b 0c 5a 29 f7 d8 50  |[.Z)P|
00000008  b8 5b 5d 5a 44 f7 d8 50  |[]ZDP|
00000010  89 e5 6a 08 55 6a 02 50  |.j.Uj.P|
00000018  90 90 90 6a 04 58 cd 80  |...j.X.|
```

これなら
読めまい

ここからが
今日の本題

ところで

現在、
アセンブラ出力環境の
GCCバージョン4対応を
やっています

以下のアーキが
利用可能に
なりました

Blackfin CR16 M32C
MicroBlaze Moxie
RL78 RX TIC6X

短歌的には
どうかというと

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]
```

下の句

上の句

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

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

機械語コードにすると

こんな感じ

```
66 11 66 a3 03
f8 02 48 65 6c 6c 03
f9 0a 02 72 6c
f9 02 01 6f 20 57 6f
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
```

機械語コード

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

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)

i386/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は

Hello Worl で
10文字なので
10BPT

集計すると

M32C	6BPT
MN10300	8BPT
RL78	9BPT
RX	10BPT
x86/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)

アセンブラで Quine

Quineとは

自分自身を
出力する
プログラム

C言語だと
こんなふうに
書けるらしい

```
int main() { char *s =
"int main() { char *s =
%c%s%c; printf(s, 34,
s, 34); }"; printf(s,
34, s, 34); }
```

Perlだと
こんな感じ
だとか

```
$prog=q(
$prog="\$prog=q(".
$prog.");";
print $prog;
print "\neval \
$prog;\n";
);
eval $prog;
```

機械語コードだと
ちょう簡単
メモリの値を
ダンプするだけ

やってみた

```
e8 00 00 00 00 58 83 e8
05 6a 14 50 6a 02 50 6a
04 58 cd 80
```

実行結果!

```
e8 00 00 00 00 58 83 e8
05 6a 14 50 6a 02 50 6a
04 58 cd 80
```

Quineを最もやりやすい言語が何かというのは諸説あるようですが

実は、最も
やりやすいのは
機械語だった

どうもありがとう
ございました