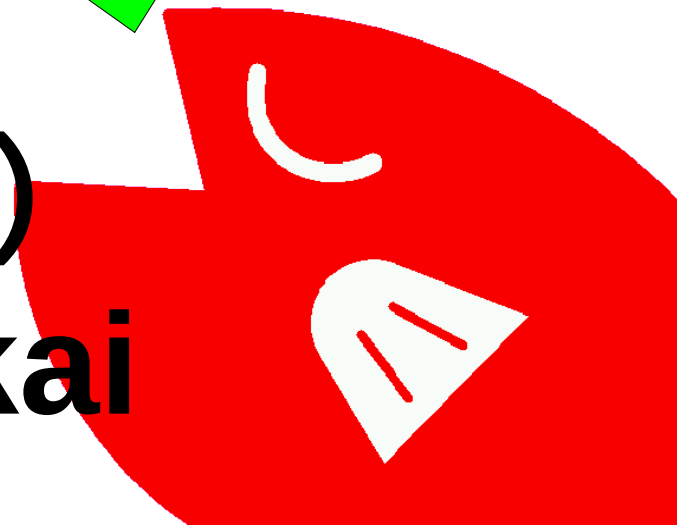


フィーリングで読む
40種類のアセンブラ

坂井弘亮

(KOZOSプロジェクト)

Twitter ID:kozossakai



過去のLTで
何度か紹介
させていただいて
いますが

こんな本を
書いています

フイーリングで読む 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)

フイーリングで

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
「それだけじゃあ
物足りないなあ」
と言われてしまうかも

ということでは

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

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、
これ也能きるだろうと
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

Н

1文字が
せいいっぱいでした

7・7の部分が無いと、
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ー
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
00000008	a5	ea	90	90	68	a5	d5	a5
00000010	a3	89	e5	6a	0c	55	6a	02
00000018	50	90	6a	04	58	cd	80	

	h	ン	グ	h	-	
	リ	..	h	フ		
	.	j	.	U	j	.
	P	.	j	.	X	.

00000000	68	00	00	a5	a2	68	a5	bb
00000008	a5	f3	90	90	68	a5	d6	a5
00000010	e9	89	e5	6a	0c	55	6a	02
00000018	50	90	6a	04	58	cd	80	

	h	..	ア	h	セ	
	ン	..	h	ブ		
	.	j	.	U	j	.
	P	.	j	.	X	.

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

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
ちよう簡単
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

