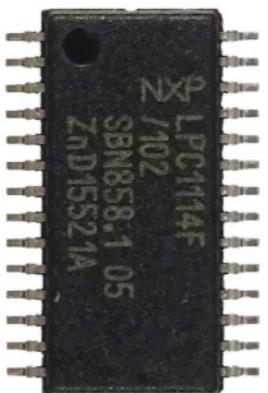
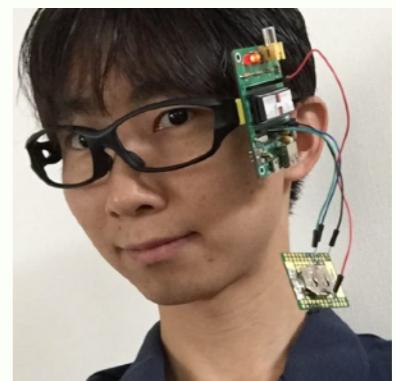


# IchigoJamは、本当に 1秒に5000万回計算できるのか!?

## ～IchigoJam マシン語入門～



jig.jp 会長 / IchigoJam 開発者 福野泰介  
@taisukef <http://fukuno.jig.jp/>





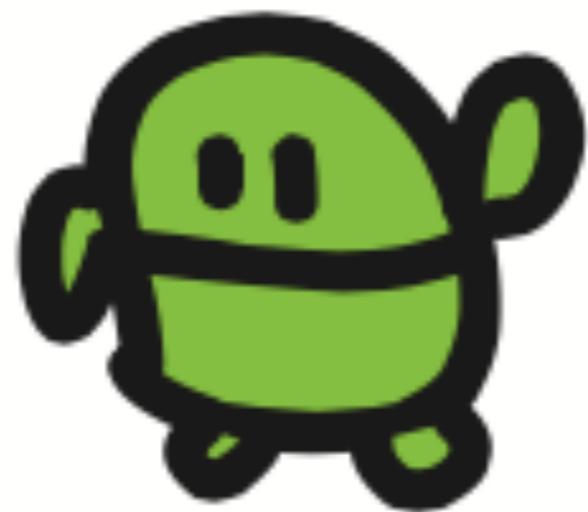
IchigoJam

CPU

1秒に5000万回！

本当に？

はかろう！



じかんのはかりかた

?TICK()

?TICK()

CLT

?TICK()

CLTでリセット

# 足し算1000回の速さ

NEW

10 N=0 : CLT

20 N=N+1 : IF N<1000 CONT

30 ?TICK( )/6

RUN

何秒？

18 = 1.8秒

おや？おそくない？



# 足し算1000回の速さ

NEW

5 VIDEO00

10 N=0 : CLT

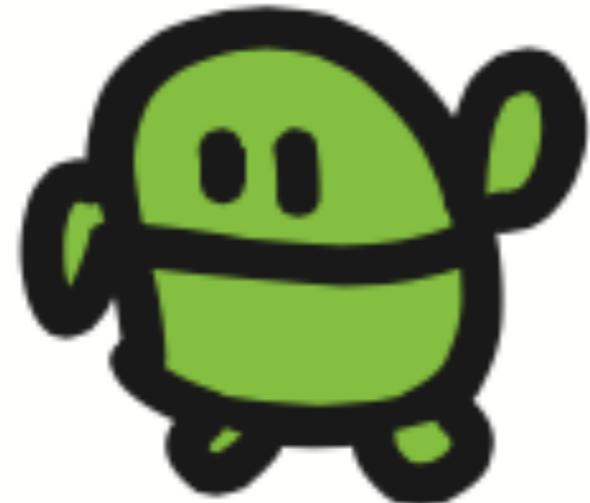
20 N=N+1 : IF N<1000 CONT

30 ?TICK( )×6

40 VIDEO01

表示を消して  
本気出す

1秒1000回くらい  
なぜおそい？



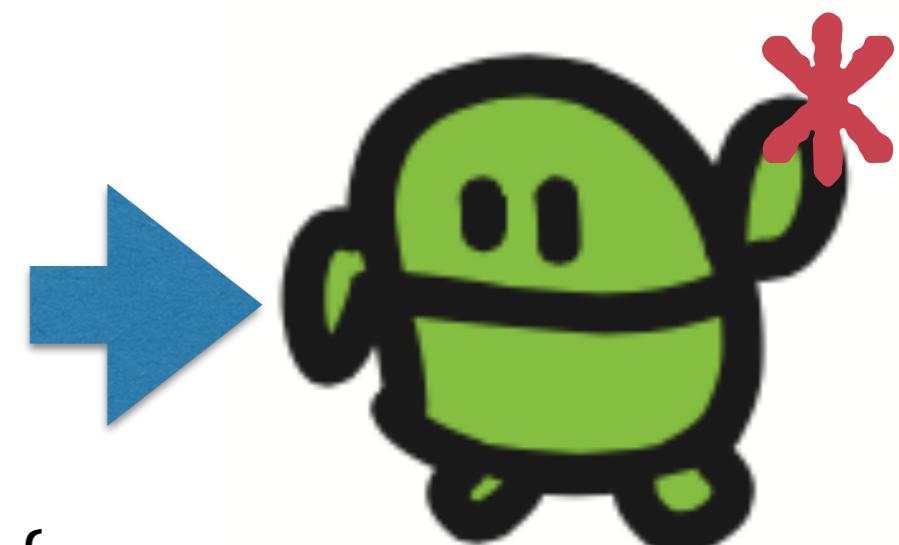
# インタプリタ (ほんやくしゃ)

LED1 →

```
while (program) {  
    cmd = ..., param = ...;  
    if (cmd == 'LED') {  
        if (param) {  
            GPIO_LED |= LED;  
        } else {  
            GPIO_LED &= ~LED;  
        }  
    } else if (cmd == 'WAIT') {  
        ...  
    } ...  
    ...  
}
```

BASICのコマンドを  
マシン語に都度翻訳

人にやさしい



マシン語  
わかる

マシン語ではなそ



# LPC1114 のとりせつ

[http://www.nxp-lpc.com/images/LPC111x\\_UM\\_Rev.00.15\\_Japanese.pdf](http://www.nxp-lpc.com/images/LPC111x_UM_Rev.00.15_Japanese.pdf)

# CPU: Arm Cortex-M0



# Arm Cortex-M0 のとりせつ

## ARM アーキテクチャ リファレンスマニュアル

ARM®

Copyright © 1996-1998, 2000, 2004, 2005 ARM Limited. All rights reserved.  
ARM DDI 0100HJ-00

[http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc\\_subset\\_architecture.reference/index.html](http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc_subset_architecture.reference/index.html)

### ARM アーキテクチャリファレンスマニュアル

Copyright © 1996-1998, 2000, 2004, 2005 ARM Limited. All rights reserved.

#### リリース情報

このドキュメントには、以下の変更が加えられています。

改訂履歴

日付	変更箇所	変更内容
1996年2月	A	初版
1997年7月	B	更新と索引の追加
1998年4月	C	更新
2000年2月	D	ARMアーキテクチャv5に対応した更新
2000年6月	E	ARMアーキテクチャv5TEに対応した更新とパートBの修正
2004年7月	F	ARMアーキテクチャv6に対応した更新(非公開)
2004年12月	G	誤記の修正
2005年3月	H	誤記の修正

#### 著作権表記

ARM、ARM Powered ロゴ、Thumb、StrongARM は ARM 社の登録商標です。

ARM ロゴ、AMBA、Angel、ARMulator、EmbeddedICE、ModelGen、Multi-ICE、PrimeCell、ARM7TDMI、ARM7TDMI-S、ARM9TDMI、ARM9E-S、ETM7、ETM9、TDMI、STRONG は ARM 社の商標です。

このドキュメントに表記されている他の製品やサービスは、対応する所有者の商標の場合があります。

このドキュメントに説明されている製品は、継続的に開発と改良が行われています。このドキュメントにある製品とその使用法に関する記載事項について、ARM は保証しません。

1. 下記の条件に従い、ARM はこの ARM アーキテクチャリファレンスマニュアルを次の目的に利用する永続的、非排他的、移転不可、無料、国際的なライセンスを許可します。使用目的は、(1) ARM からのライセンスにより配布されるマイクロプロセッサコアで実行することを目的としたソフトウェアアプリケーションとオペレーティングシステム(2) ARM からのライセンスにより配布されるマイクロプロセッサコアで実行することを目的としたソフトウェアプログラムの開発用に設計されたツール(3) ARM からのライセンスにより製造されるマイクロプロセッサコアを搭載する集積回路のいずれかの開発に限られます。

2. 条項 1 で明示的に与えられているものを除き、ARM アーキテクチャリファレンスマニュアル、またはそれに含まれるいかなる知的著作物についても、いかなる権利、資格、利益も与えるものではありません。条項 1 に示されている許諾は、いかなる場合でも明示的、暗黙的、禁反言、その他の形で ARM アーキテクチャリファレンスマニュアル以外のいかなる ARM テクノロジに関するライセンスも与えるものではありません。条項 1 で与えられるライセンスには、ARM パテントを使用する、または使用に含める権利は明示的に除外されます。条項 1 の条件では、次に示す権利は与えられません。(1) ARM アーキテクチャリファレンスマニュアルを、この ARM アーキテクチャリファレンスマニュアルで説明されている命令、プログラマモデル、

# まとめ

## Cortex-M0 Armマシン語表 (asm15、抜粋)

代入	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd = u8	0	0	1	0	0		Rd						u8				1
Rd = Rm	0	1	0	0	0	1	1	0	Rd3		Rm			Rd2-0		1,3	

\*Rd3とRd2-0の4bitでRdを指定する、RdがPCの時3cycles

演算	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd += u8	0	0	1	1	0		Rd						u8				1
Rd -= u8	0	0	1	1	1		Rd						u8				1
Rd = PC + u8	1	0	1	0	0		Rd						u8				1
Rd += Rm	0	1	0	0	0	1	0	0	Rd3		Rm			Rd2-0		1,3	
Rd = Rn + u3	0	0	0	1	1	1	0		u3		Rn		Rd			1	
Rd = Rn - u3	0	0	0	1	1	1	1		u3		Rn		Rd			1	
Rd = Rn + Rm	0	0	0	1	1	0	0		Rm		Rn		Rd			1	
Rd = Rn - Rm	0	0	0	1	1	0	1		Rm		Rn		Rd			1	

Rd 32bit レジスタ x 16 (CPU内にあるメモリ)

R0～R7 汎用的に使えるレジスタ (R4～R7は元に戻す)

R0 パラメータの受け渡しに使う

R8～R12 使えるコマンドが限られるレジスタ

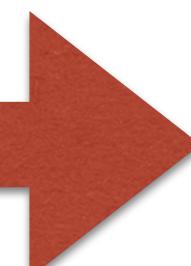
R13～R15 使う用途が特殊なレジスタ

# 足し算させよう

## Cortex-M0 Armマシン語表 (asm15、抜粋)

代入	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd = u8	0	0	1	0	0		Rd						u8				1
Rd = Rm	0	1	0	0	0	1	1	0	Rd3		Rm		Rd2-0				1,3

\*Rd3とRd2-0の4bitでRdを指定する、RdがPCの時3cycles



演算	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd += u8	0	0	1	1	0		Rd						u8				1
Rd -= u8	0	0	1	1	1		Rd						u8				1
Rd = PC + u8	1	0	1	0	0		Rd						u8				1
Rd += Rm	0	1	0	0	0	1	0	0	Rd3		Rm		Rd2-0				1,3
Rd = Rn + u3	0	0	0	1	1	1	0		u3		Rn		Rd				1
Rd = Rn - u3	0	0	0	1	1	1	1		u3		Rn		Rd				1
Rd = Rn + Rm	0	0	0	1	1	0	0		Rm		Rn		Rd				1
Rd = Rn - Rm	0	0	0	1	1	0	1		Rm		Rn		Rd				1

足し算する時は「Rd += u8」

(レジスタdに、符号なし8bitの数を足し込む) を使う

例えばレジスタ0(R0)に1足す時は

00110 000 00000001 (2進数) となる

# 10進数、2進数、16進数

	10進数へ	10進数から
2進数	?`111 ?	?BIN\$(6) 110
16進数	?#F 15	?HEX\$(255) FF

# BASICからマシン語よびだし

[ 0 ] = ^ 00110 000 00000001

?USR(#800, 0)

バグった！？

足し算する時は「Rd += u8」

(レジスタdに、符号なし8bitの数を足し込む) を使う

例えばレジスタ0(R0)に1足す時は

00110 000 0000001 (2進数) となる

# BASICからマシン語呼びだし

[0] = ^00110 000 00000001

[1] = #4770

?USR (#800, 0)

1

?USR (#800, 10)

11

#4770 = 呼び出し元へもどる命令が必要

# 足し算1000回の速さ

R1 = 0

R1 += 1

R1 - R0

IF !0 GOTO -2

RET

R1-R0が  
0じゃない時  
2つ前へ

# 足し算1000回の速さ

## Cortex-M0 Armマシン語表 (asm15、抜粋)

代入	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd = u8	0	0	1	0	0		Rd										1
Rd = Rm	0	1	0	0	0	1	1	0	Rd3		Rm				Rd2-0		1,3

\*Rd3とRd2-0の4bitでRdを指定する、RdがPCの時3cycles

演算	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd += u8	0	0	1	1	0		Rd										1
Rd -= u8	0	0	1	1	1		Rd										1
Rd = PC + u8	1	0	1	0	0		Rd										1
Rd += Rm	0	1	0	0	0	1	0	0	Rd3		Rm				Rd2-0		1,3
Rd = Rn + u3	0	0	0	1	1	1	0		u3		Rn				Rd		1
Rd = Rn - u3	0	0	0	1	1	1	1		u3		Rn				Rd		1
Rd = Rn + Rm	0	0	0	1	1	0	0		Rm		Rn				Rd		1
Rd = Rn - Rm	0	0	0	1	1	0	0		Rm		Rn				Rd		1

```
[0] = ^0010001 00000000 R1=0
[1] = ^0011001 00000001 R1+=1
[2] = ^01000010 10000001 R1-R0
[3] = ^11010001 11111100 IF!0GOTO-2
[4] = ^01000111 01110000 RET
```

\*GOTOは更に-2して2の補数表現 <https://fukuno.jig.jp/1188>

# 足し算1000回の速さ

NEW

```
10 POKЕ#800,0,33,1,49,129,66,252,  
209,112,71
```

```
20 CLT:USR(#800,1000):?TICK()
```

SAVE2

RUN

# 足し算1万回の速さ

LIST

```
10 POK E#800,0,33,1,49,129,66,252,  
209,112,71
```

```
20 CLT:USR (#800,10000):?TICK()
```

RUN

# 足し算100万回の速さ

LIST

```
10 POK E#800,100,34,80,67,0,33,1,  
49,129,66,252,209,112,71
```

```
20 CLT:USR( #800,10000):?TICK()
```

SAVE2

RUN

\*追加 (パラメータを100倍する)

```
R2=100 ^00100 010 1100100
```

```
R0*=R2 ^0100001101 010 000
```

計算してみよう



足し算100万回の速さ

13/60秒 やく0.2秒で100万回

1秒で500万回？

まだ10倍おそい……

# 足し算100万回の速さ

LIST

5 VIDEO0

10 POKE#800,100,34,80,67,0,33,1,  
49,129,66,252,209,112,71

20 CLT:USR(#800,10000):?TICK()

30 VIED01

RUN

表示を消して  
本気出す

# 1秒1000万回！？

## Cortex-M0 Armマシン語表 (asm15、抜粋)

代入	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd = u8	0	0	1	0	0		Rd										1
Rd = Rm	0	1	0	0	0	1	1	0	Rd3	Rm					Rd2-0	1,3	

\*Rd3とRd2-0の4bitでRdを指定する、RdがPCの時3cycles

演算	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd += u8	0	0	1	1	0		Rd										1
Rd -= u8	0	0	1	1	1		Rd										1
Rd = PC + u8	1	0	1	0	0		Rd										1
Rd := Rm	0	1	0	0	0	1	0	0	Rd3	Rm				Rd2-0		1,3	

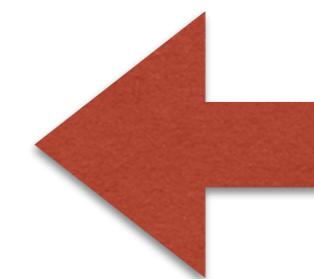
R1 = 0

R1 += 1 ← 1 cycle

R1 -= R0 ← 1 cycle

IF !0 GOTO -2 ← 3cycle

RET =合計 5サイクル



何サイクルで  
処理するか？

計算は1秒5000万回ペースで実施していた！



IchigoJam

CPU

1秒に5000万回！



(C)IchigoJam



(C)Apple

IchigoJam

iPhone 11  
GPU

5000万回

IchigoJam  
何台分？→

1500円

2万台分

8万円



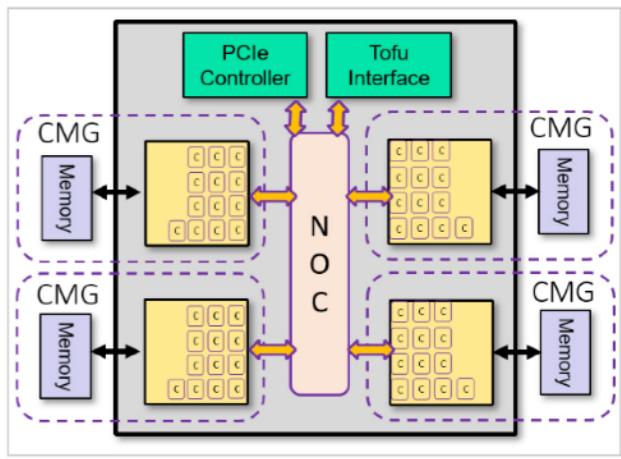
(C)TSUKUMO

パソコン  
GPU

10兆回

20万台分

10万円



SVE: Scalable Vector Extension

(C)RIKEN

スパコン富岳

100京回

200億台分

1100億円



(C)IchigoJam



(C)Apple

IchigoJam

iPhone 11  
CPU

5000万回

IchigoJam  
何台分？→

1500円

40台分

8万円



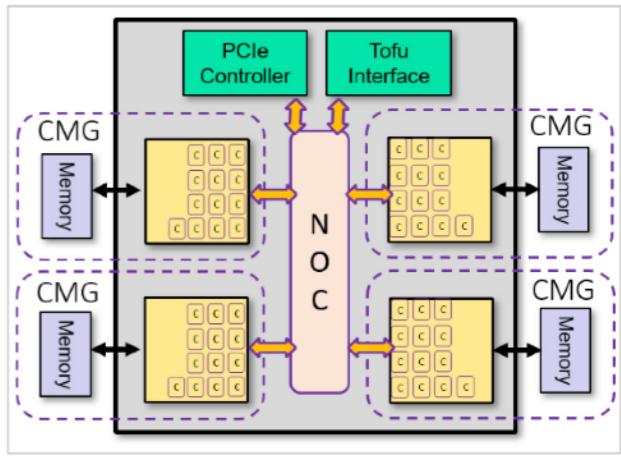
(C)TSUKUMO

パソコン  
CPU

20億回

40台分

10万円



SVE: Scalable Vector Extension

(C)RIKEN

スパコン富岳

100京回

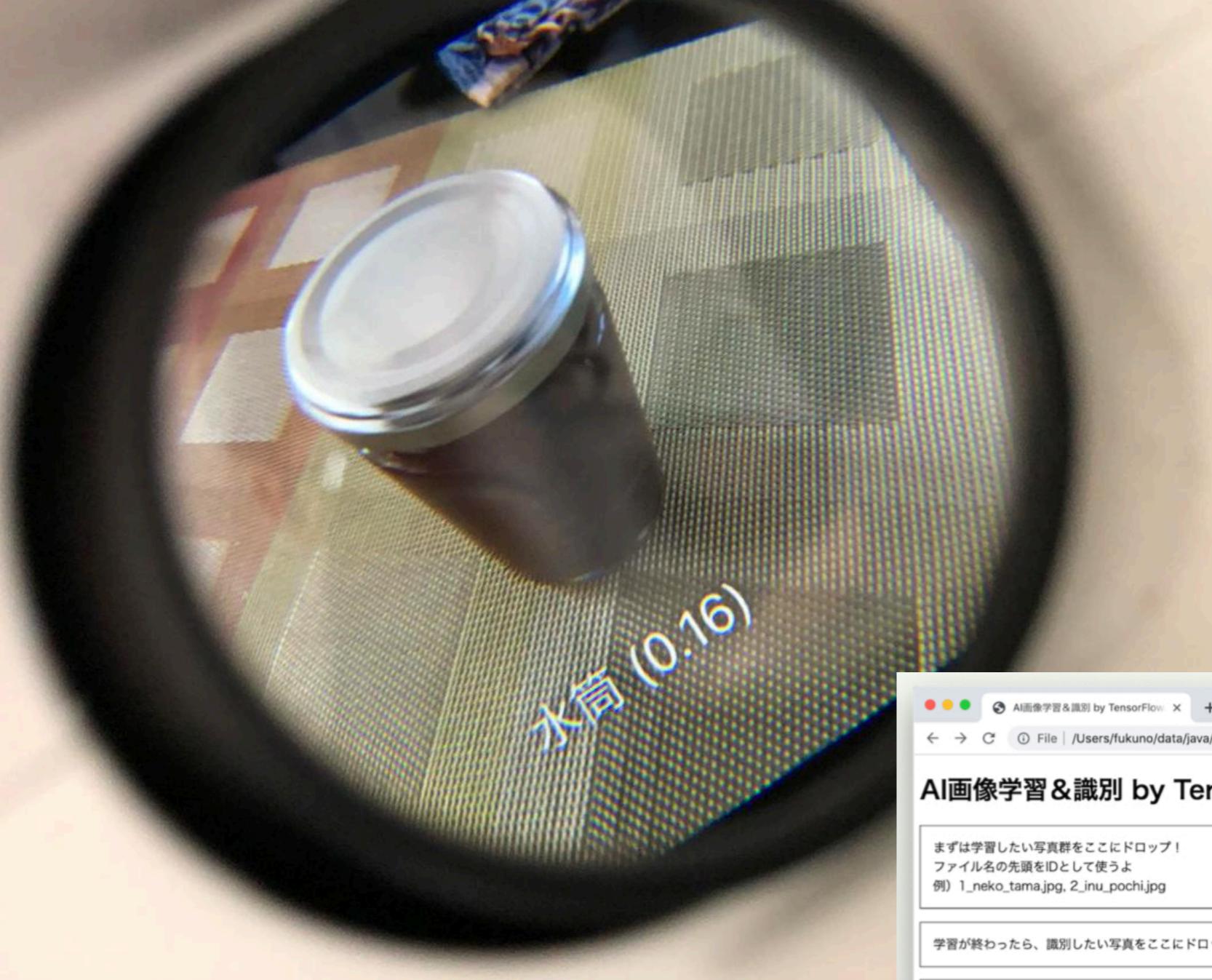
200億台分

1100億円

# クラウド=オンラインレンタルパソコン 20億回級のコンピューターは、1時間1円で借りられる！

The screenshot shows the ConoHa web interface. The top navigation bar includes the ConoHa logo, service icons for VPS and WING, a notification bell, user profile, and a dropdown menu. On the left, a sidebar menu lists various services: サーバー追加 (+), サーバー, ディスク, イメージ, ネットワーク, セキュリティ, オブジェクトストレージ, DNS, ライセンス, ドメイン, and API. The main content area is titled 'VPS' and '東京'. It displays a message 'ご利用中のVPSはありません' (No VPS is currently in use) and a blue '追加' (Add) button. Below this, there are four summary boxes: '今月のご利用金額 2019-10-01~2019-10-30 25 円' (Monthly usage amount 2019-10-01~2019-10-30 25 yen), 'チャージ残高 0 円' (Charge balance 0 yen), 'クーポン残高 675 円' (Coupon balance 675 yen), and a section for the 'ConoHa公式スマートアプリ' (Official ConoHa mobile app) with a smartphone icon.

時給1円のサーバーくんをプログラミングで自在に操ろう！ クラウド入門 ConoHa API編  
<https://fukuno.jig.jp/2656>



JavaScriptで  
使えるよ！

<https://fukuno.jig.jp/2645>

GPUを駆使した応用  
→ 3Dゲーム、AI

AI画像学習&識別 by TensorFlow.js

まずは学習したい写真群をここにドロップ！  
ファイル名の先頭をIDとして使うよ  
例) 1\_neko\_tama.jpg, 2\_inu\_pochi.jpg

学習が終わったら、識別したい写真をここにドロップしてね

識別した結果は8番です！

8番 確度 : 18.00%
7番 確度 : 12.16%
1番 確度 : 10.95%
0番 確度 : 10.92%
4番 確度 : 10.79%
3番 確度 : 10.72%
6番 確度 : 9.26%
2番 確度 : 9.17%
5番 確度 : 8.04%

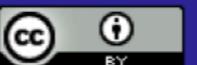
学習

1\_neko.jpg  
2\_inu.jpg

予測

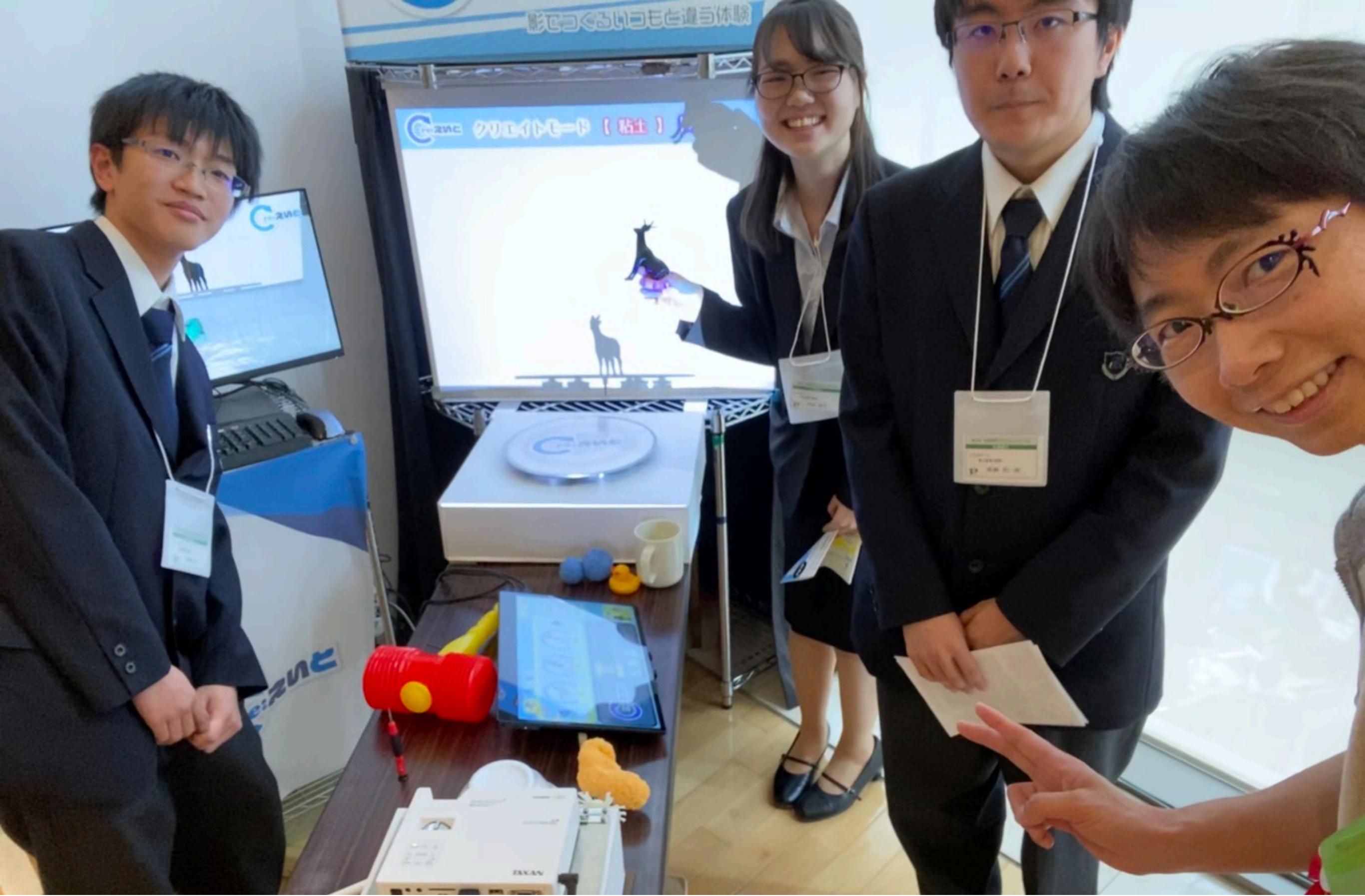

TensorFlow.js はじめの一っぽ  
JavaScript版



CC BY @taisukef <https://fukuno.jig.jp/2645>



高専プロコン2019 in 都城 <https://fukuno.jig.jp/2641>



高専プロコン2019 in 都城 <https://fukuno.jig.jp/2641>



高専プロコン2019 in 都城 <https://fukuno.jig.jp/2641>

- ◆出展料無料
- ◆物販 OK
- ◆1日だけでもOK



#NT 鮎江

CC BY Azulily\_V

みてもらおう！

出展者募集中！！

みてたのしもう！

NT 鮎江 2019

誰でもふらっと見に来ていただける技術を楽しむ祭典

2019/10/26~27

26日（土）10時～17時

27日（日）10時～16時

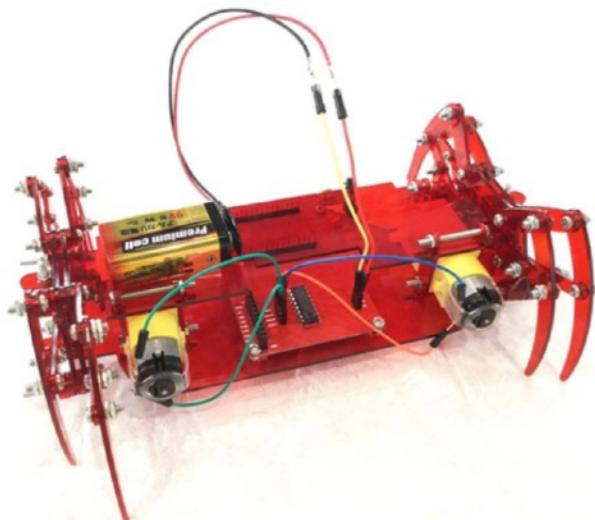
会場 ▶ 鮎江市嚮陽会館 2F 大会議室

NT 鮎江 2020 10/3-4 予定！

# ないものはつくろう！

さばえカニロボット

新商品



¥ 8,700

※こちらの価格には消費税が含まれています。  
※送料は別途発生いたします。詳細は [こちら](#)  
※5,000円以上のご注文で送料が無料になります。

数量

1

カートに入れる

外部サイトに貼る

ツイート シェア 49

通報する



メカ担当：MASAHARU（中2）

基板担当：MISAKI（高2）

Hana道場で販売、子供開発のロボット！

作品応募は

**9/30**

(月まで!)

あなたのアイディアを試すチャンス!

第1回

# みやぎプロコン

みやぎプロコンって？

応募について

ワークショップ

コンセプト  
ストーリー



仙台高専で開催された「みやぎプロコン」  
高専生が小中学生を審査するのおもしろいかも？福井でも！

<https://fukuno.jig.jp/2653>

PCNこどもプログラミングコンテスト2019-2020

# PCN こどもプロコン 2019-2020 開催決定！

PCNプロコンは  
君のプログラムを  
待っている！

2019  
10/1(Tue.)  
START

主催：一般社団法人 プログラミングクラブネットワーク(PCN)

後援：文部科学省、総務省、IT総合戦略本部、福井県、福井市、福井市教育委員会、福井新聞社

PCNこどもプロコン2019-2020 ご協賛企業・団体

I-O DATA

NSD

SAKURA  
internet

ZOZO  
Technologies

PFU  
a Fujitsu company

株式会社アイティプロジェクト  
共立電子産業株式会社  
ソリッドシード株式会社

株式会社秋月電子通販  
一般社団法人ココロエデュケーションラボ  
ワンダーラボ大阪

# 小中学生向け PCNこどもプロコン



ノートPCがもらえる！？



<http://pcn.club/contest/>

後援：総務省、文科省、経産省、IT総合室  
高専機構、未来の学びコンソーシアム



# Hana道場

4才から74才まで通うITものづくり道場

● ただいま修行中です

11:00～17:00

イベントに  
参加する

道具を  
借りる

鯖江、Hana道場へどうぞ！ <http://hanadojo.com/>