プログラムはどこまで小さくできるのか

一般社団法人 未踏 理事
サイボウズ・ラボ株式会社
川合秀実

はじめに

- 質問はいつでもしてください
- 質問を後でしてもいいけど、思いついた時にしたほうが、その後の話が分かりやすくなると思います！
- えっと3ページ前のことについてなんですが・・・
  - こういうのも大歓迎です！！

自己紹介

- 「30日でできる！OS自作入門」の著者
- IPA未踏ユース2002年度 卒業生
- セキュリティキャンプの講師（2009年より）

- 1975年生まれのおじさん

「コードゴルフ」という遊び

- お題が与えられ、それを何バイトのプログラムで書くことができるかを競う遊び

- 例： "hello, world¥n"を出力するプログラムは何バイトで書けるか？

- Linux(x86 ELF)での競争の結果：57バイト
  – すごい！

菊やんさん（2006年）

ELF・Linuxは不利

- でもこの57バイトの半分以上はELFフォーマットによる制約に由来するもの
- ELFフォーマットを考えた人たちもそれを採用したLinux開発陣も、コードゴルファーの気持ちが分かってない！
- こっちは必死の思いで1バイトを削っているのに、そもそもELFフォーマットは無駄がいっぱい
- その証拠に、同じx86でもMS-DOSなら同じプログラムを22バイトで書ける（半分以下）

コードゴルフの先へ

- MS-DOSではヘッダを一切書かないCOM形式というものがあり、それが最強
  - （0バイトよりも少ないヘッダサイズはないから）
- しかし・・・
  - MS-DOSはアプリケーションにとって便利なAPIを十分にそろえているわけではない
  - 呼び出ししやすいように工夫があるわけでもない
- じゃあ、そこを改善したらどこまでいくだろうか？
  - （註：川合は自作OSが得意なので、そういうOSを作ってしまえばいいと、安易に考えた）
- API = アプリケーション・プログラム・インタフェース
  - （この文脈ではシステムコールだと思っても問題ない）

やってみた

- hello対決

57バイト	**Linux ELFバイナリ**
22バイト	**MS-DOS COM形式**
18バイト	**COM64plus**（友人のN氏が作ったOS）（彼は永遠のライバル）
16バイト	**第二世代OSASK**（川合が2008年～2009年に作ったOS）

- ほらね！ （読み方：OSASK=おさすく）
- やっぱりMS-DOS版よりも小さくなった！

ここでN氏について

- 男性で日本人であること以外はいろいろ不明
  - 謎の人でいつづけるのが好きな人
  - 直接会ったことはある
- OSを作るのがうまい人、OS作るのが好きな人
  - 私に似ている！
- プログラムを小さくするのが好き
  - ここも私に似ている！

chars対決(1)

- このOSでは、helloだけが小さくなったわけじゃない
- 課題「画面にキャラクターコード0x20から0x7e（印字可能なASCIIキャラクタのすべて）と改行を出力する」

```
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[¥]^_`
abcdefghijklmnopqrstuvwxyz{|}~
```

- → chars対決

chars対決(2)

- chars対決

17バイト	**MS-DOS COM形式**
14バイト	**COM64plus（N氏）**
13バイト	**第二世代OSASK（川合）**

- 他にも小さくなるものはたくさんあるけど、キリがないので省略

さらに究極を目指す

- OSを都合よく改造していいのなら、CPUも改造OKにすべきでは？
  - それでこその究極では？
- x86がコードゴルフに向いているとは言えない
  - それでも他のCPUよりはかなり向いているけど
  - たとえばある32ビットのレジスタに1を代入したいとき、どうやっても3バイトを要する（こんな簡単なことなのに！）
- Javaのバイトコードと同じ方式で実現
  - だから本当のCPUを作るわけじゃないよ

きっかけ

- 2013年にライバルのN氏が、CPUの機械語もこうしたほうがいいと定義し直して、それで 超絶に小さくできることを実証して、私に自慢してきた。「こんなのできちゃったよー。」

- すごくうらやましかった
- 自分でもやってみたいと思った

- バトル再開！

chars対決(3)

- chars対決（2013年）:

17バイト	**MS-DOS COM形式**
14バイト	**COM64plus**（N氏）
13バイト	**第二世代OSASK**（川合）
10バイト	**CLE**（N氏）
9バイト	**第三世代OSASK**（川合）

- 何とか追いつき追いぬいた！
- しかしN氏も本気を出してきて・・・2014年に8バイト達成
  - ありえない！
- そして・・・最終結果は・・・

chars対決(4)

- chars対決：

352バイト	**Java**
17バイト	**MS-DOS COM形式**
14バイト	**COM64plus（N氏）**
13バイト	**第二世代OSASK（川合）**
8バイト	**CLE（N氏）**
7バイト	**第三世代OSASK（川合）**

- ちなみにJavaも小さくする努力はしているらしいです
  - でも話にならないですね・・・
- すごく小さいと思ってきたMS-DOS版の半分以上は、実は無駄だった！

「ずる」について

- これは無駄を省くための競争
- だからずるはいけない

- ずるの例:
  - OS側にhello専用APIやchars専用APIがあり、これを使うと、小さく書ける
  - CPUの命令にhello専用命令やchars専用命令があり、これを使えば1命令でできてしまう

「ずる」ではないがきわどいもの

- helloやcharsが必要とする命令を優先的に1バイト命令などに割り当てる

- これはhelloやcharsが小さくなるものの、代償として他のアプリが小さく書けなくなる
  - これは望ましくない
  - 純粋に使用頻度の高い命令が短くなるべき
  - だから他のアプリも小さくなるか、検証するべき

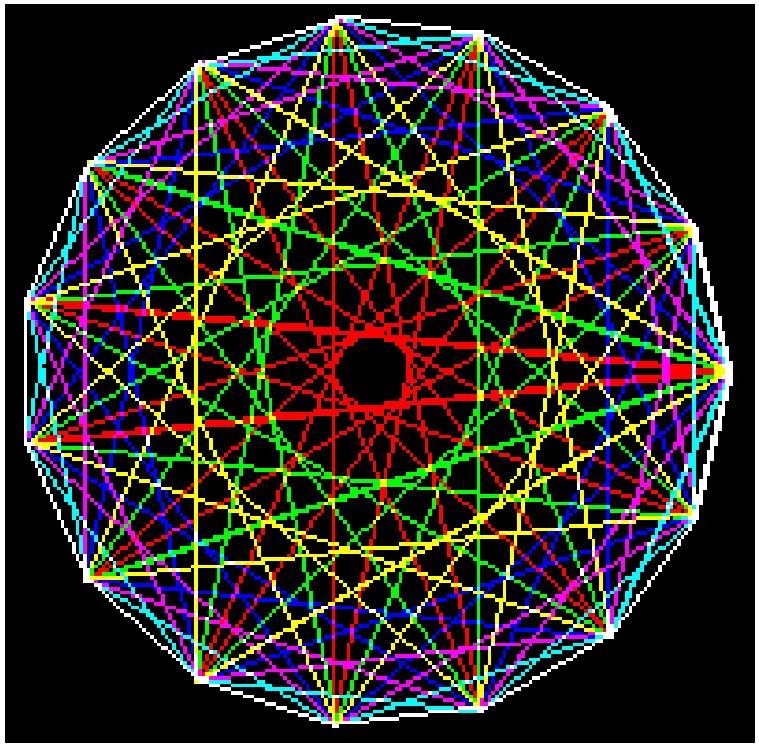
第三世代OSASKの実力(1)

- 実行に必要なもの
  - アプリ
  - osecpu.exeという実行ファイル（Java-VMに相当）
    - これは**29.0KB**
    - ここが大きくなるようでは二流
    - Osecpuというのは開発コードネーム
- 他には何も参照しない
  - 当然
  - 他のものも使ってよかったら、どれだけでも「ずる」ができる

第三世代OSASKの実力(2)

- bball対決

186バイト	**第一世代OSASK**
114バイト	**MS-DOS/PC-9801**
63バイト	**第三世代OSASK**

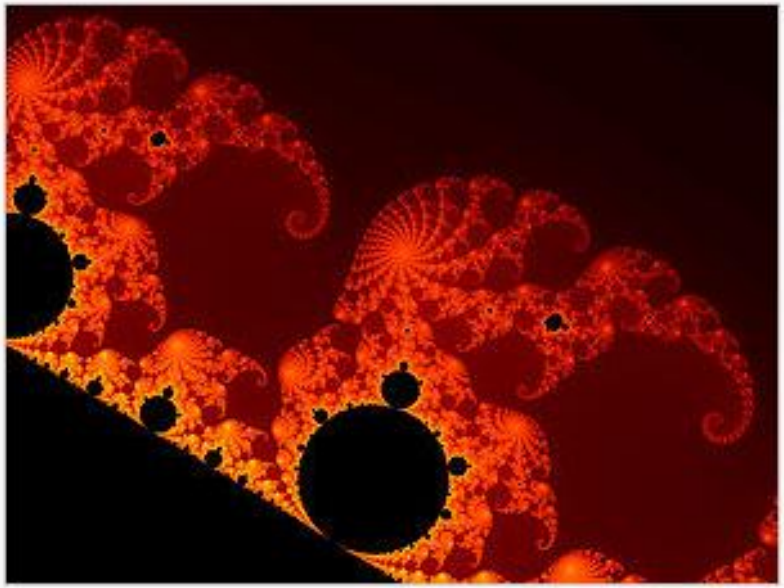


- あなたは、63バイトのプログラムでこの絵が描けますか？

第三世代OSASKの実力(3)

- Wikipediaより:

  – 「遥かに小さい」?
  – よし、はっきりさせよう!

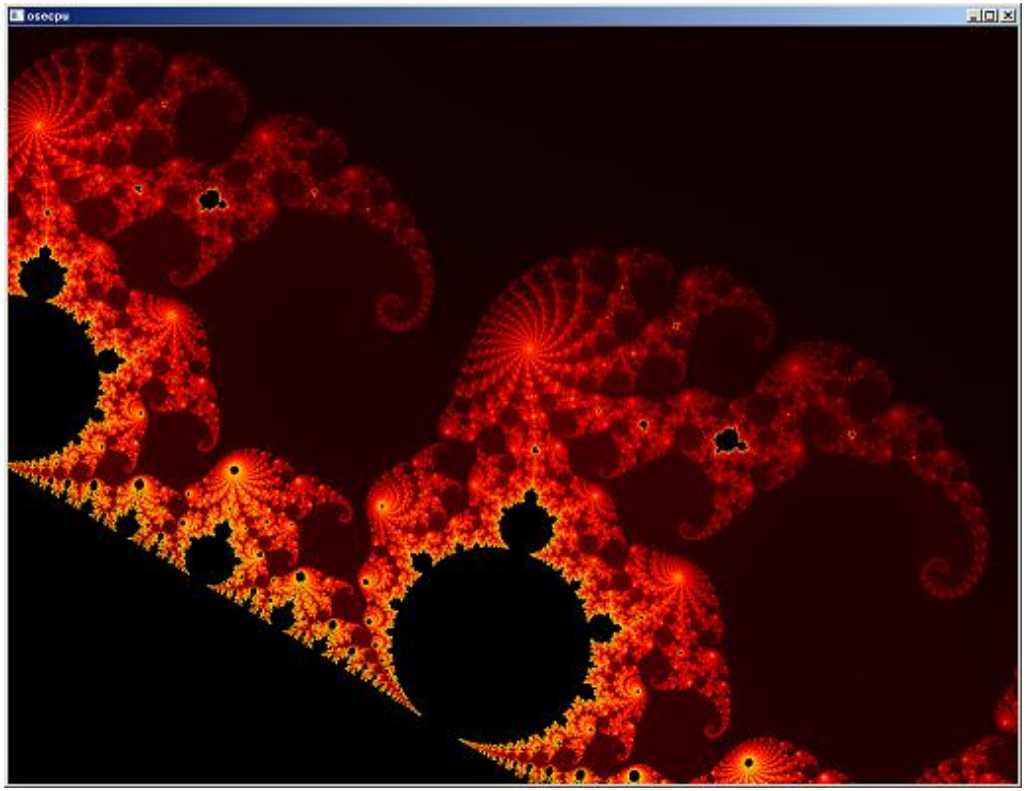


この画像はフラクタル図形であるマンデルブロ集合の一部である。このJPEGファイルのサイズは17KB以上(約140,000ビット)ある。ところが、これと同じファイルは140,000ビットよりも遥かに小さいコンピュータ・プログラムによって作成することが出来る。従って、このJPEGファイルのコルモゴロフ複雑性は140,000よりも遥かに小さい。

  – https://ja.wikipedia.org/wiki/コルモゴロフ複雑性

第三世代OSASKの実力(4)

- コルモゴロフ複雑性対決
  - 111バイト
  - つまり888ビット
  - きっとあのキャプションを書いた人の想定を遥かに超えているに違いない(笑)



第三世代OSASKの実力(5)

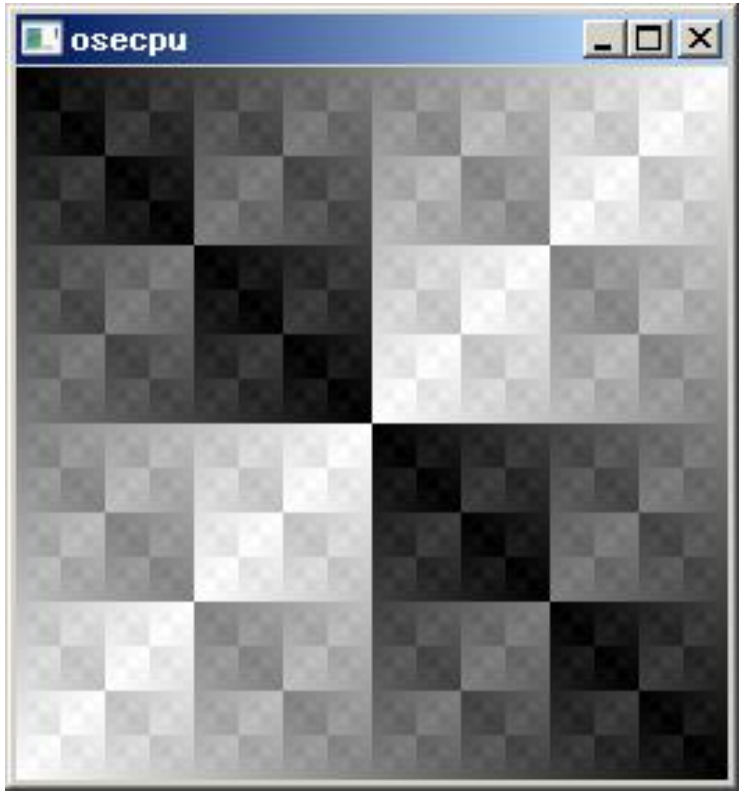
- インベーダ対決

2192バイト	**Windows9x**
1509バイト	**「はりぼてOS」**
1241バイト	**TownsOS**
1108バイト	**第一世代OSASK**
985バイト	**MSX-DOS**
430バイト	**第三世代OSASK**

第三世代OSASKの実力(6)

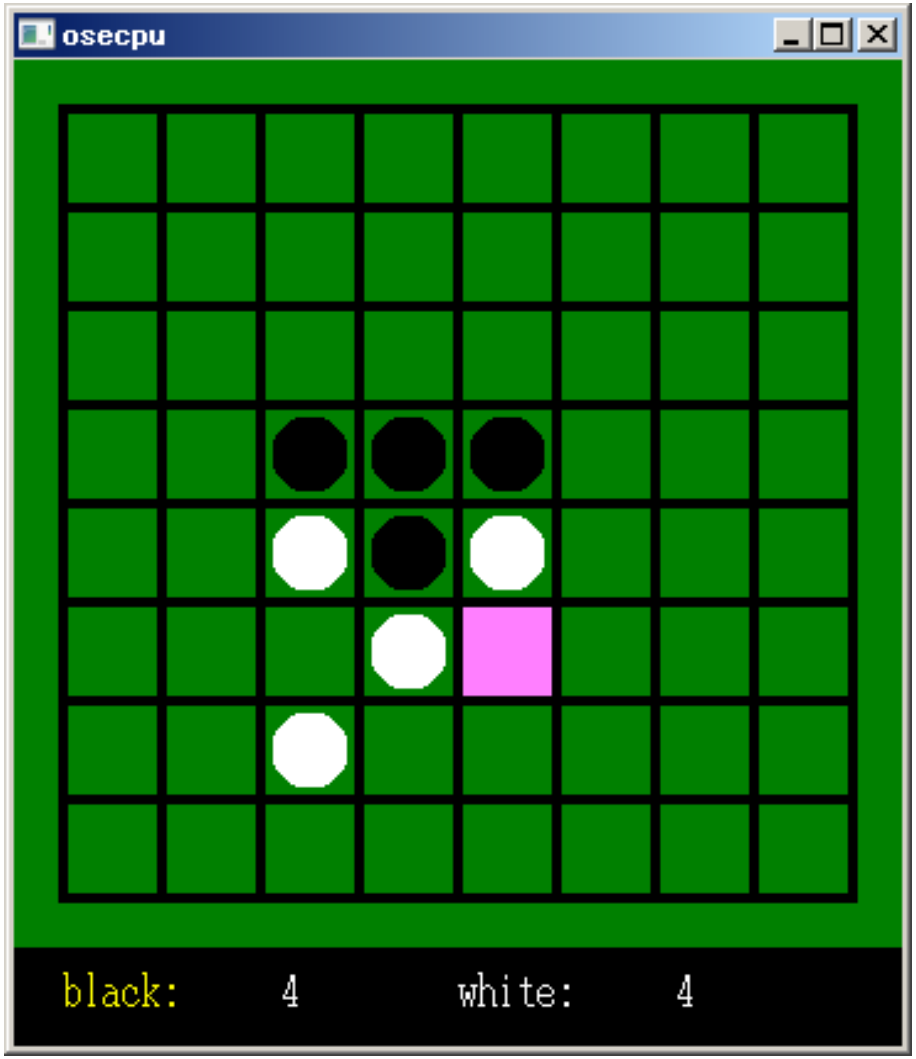
- きれいな模様
  - 9バイト！

```
#include "osecpu_ask.h"

Int32s x:R01, y:R02, c:R00;

api_openWin(256, 256);
for (y = 0; y != 256; y++) {
    for (x = 0; x != 256; x++) {
        c = x ^ y;
        c *= 0x10101;
        api_drawPoint(MODE_COL24, c, x, y);
    }
}
```



第三世代OSASKの実力(7)

- オセロゲーム
  - 476バイト
  - 人間が黒
  - コンピュータが白
  - 白はランダム打ち
    ルールは守る



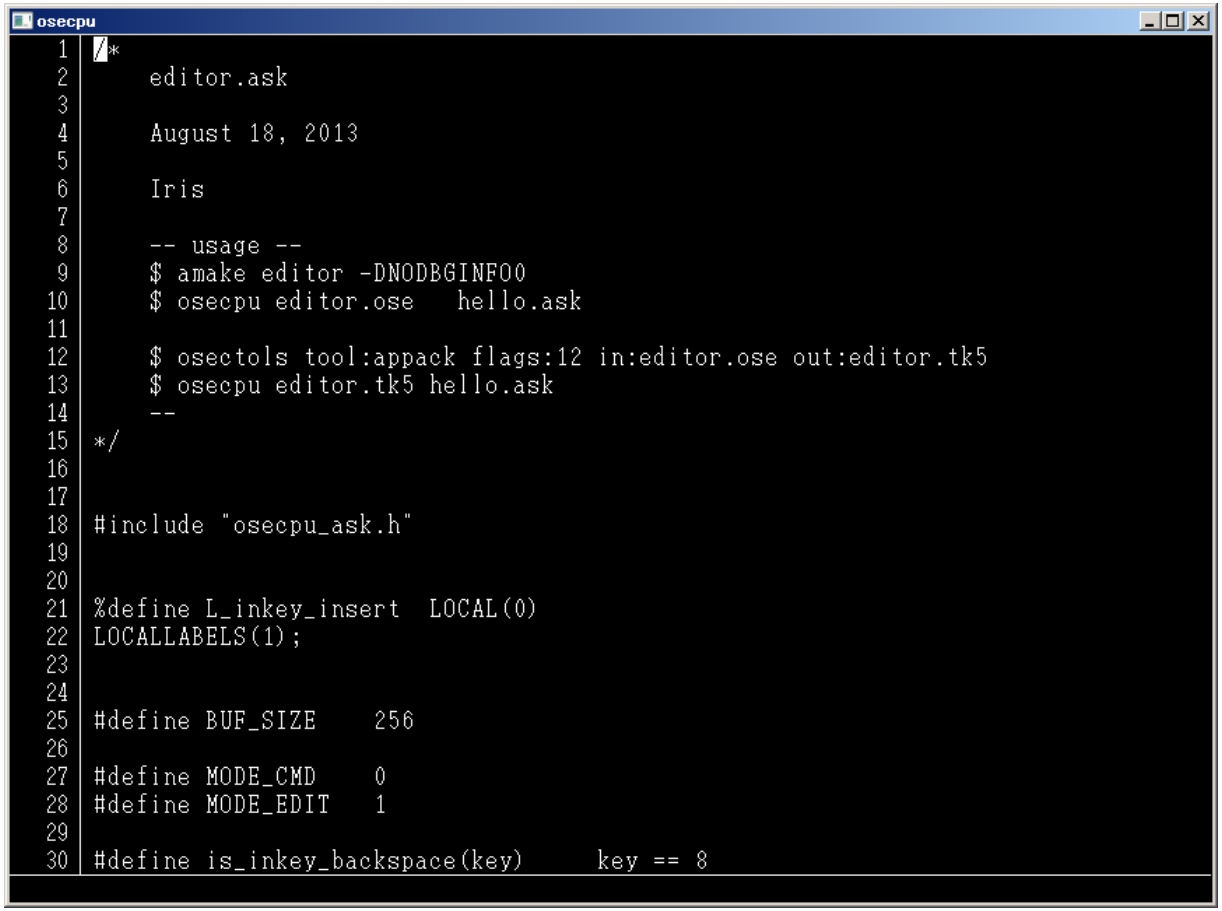
第三世代OSASKの実力(8)

- テキストエディタ

– 881バイト！

ソースコード

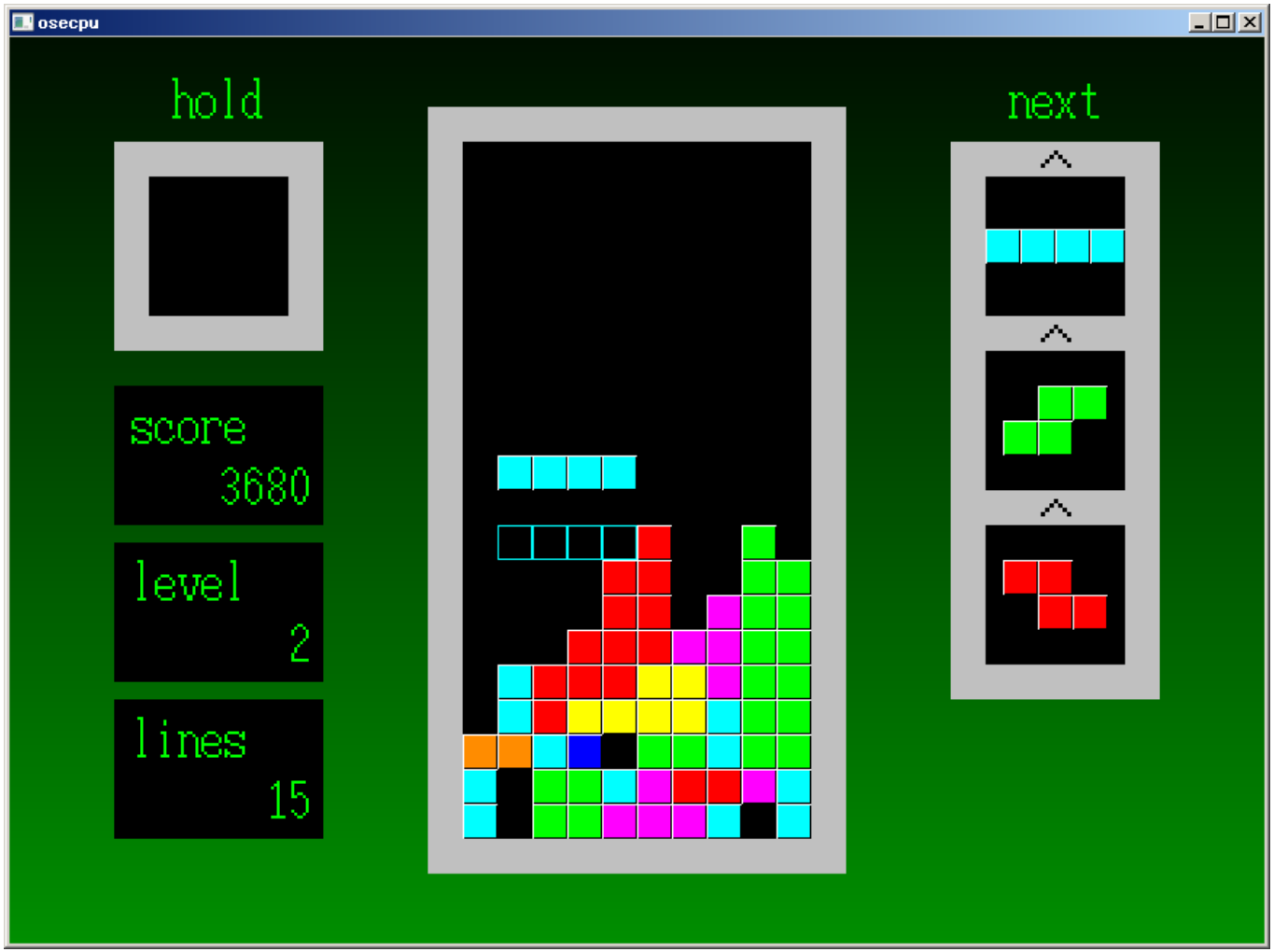
530行



第三世代OSASKの実力(9)

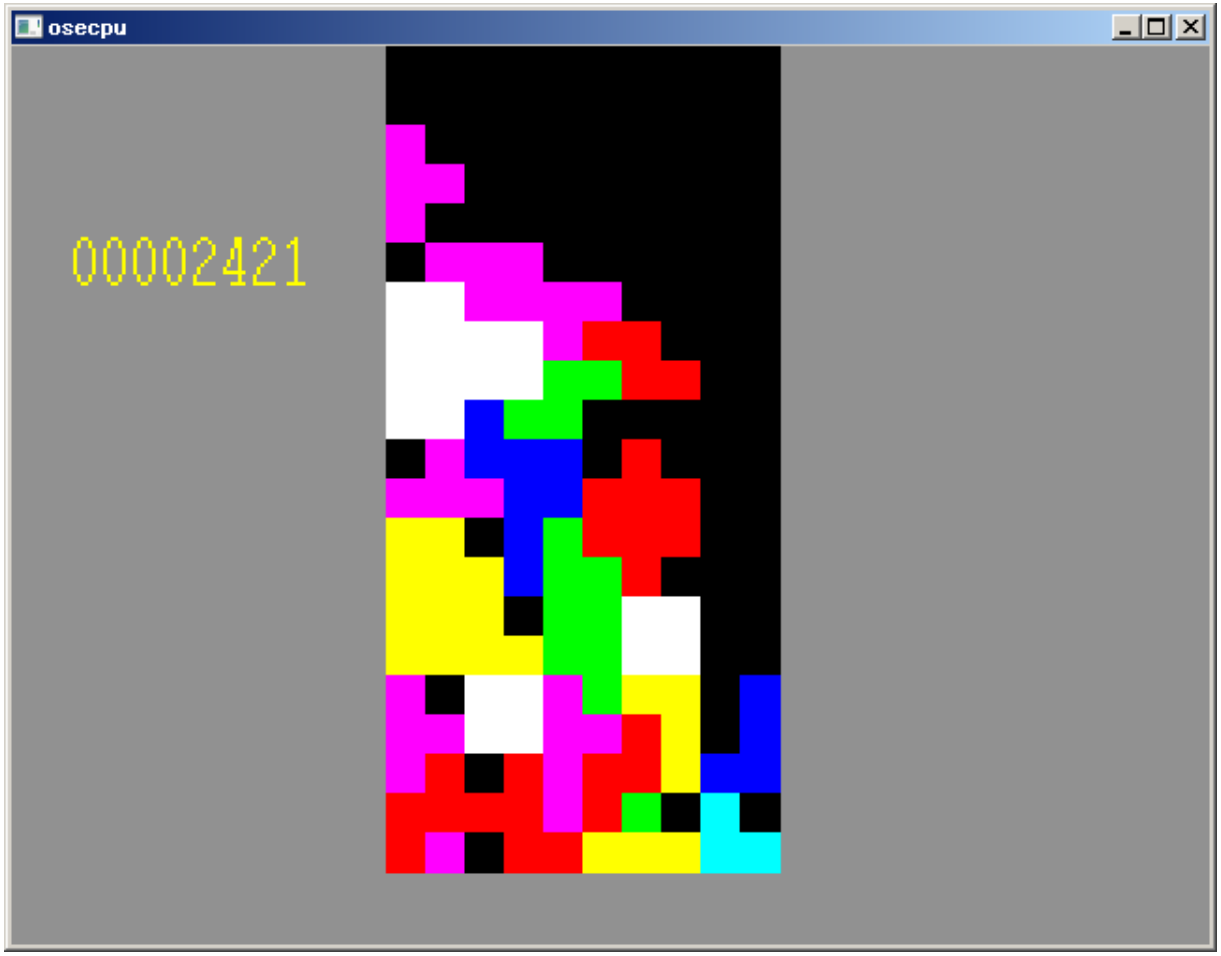
- テトリス

1175バイト



第三世代OSASKの実力(10)

- シンプルなテトリス
  - 296バイト



結論

- 第三世代OSASKは「ずる」してない
- 何を作っても簡単に世界最小の座をとれる

- これは単に無駄がないだけであって、高度なテクニックを要求しているわけではない
  - 学生でも2週間くらい練習すれば、小さなアプリが書けるようになる（セキュリティキャンプで実証済み）

逆に

- 世間のものは無駄がたくさんある
- がんばっても小さくできないので、無駄を強いられているとも言える
  - まあそもそもみんな頑張っていないのだけど
- 究極まで削ったWindows版のインベーダゲーム（2192バイト）の80%は無駄
  - いやでも、あのインベーダゲームを2.14KBで書けるのはとんでもなくすごいことなんですよ！
- これでいいのだろうか？

無駄の具体例

- 「x86で32ビットレジスタに1を代入」
  - EAX = 1
- 普通の書き方:
  - B8 01 00 00 00（普通のMOV命令、なんと5バイト）
- がんばった書き方:
  - 6A 01 58（スタックに1を積んで、EAXで取り出す）
- 第三世代OSASK:
  - 2 1 0（1.5バイト）

B8 01 00 00 00 を考える

- 標準的なレジスタ代入命令
  - B8 というのは1バイト＝8ビット　10111 000
  - このうちの5ビットが命令番号
  - 3ビットがレジスタ番号
  - 正直、ここまでは悪くない（第三世代OSASKも同じようなもの）
- 次の 01 00 00 00 がいけない（4バイト）
  - 32ビットで表した1
  - なぜ32ビットで書かなければいけないのか

なぜ32ビットで書くのか？

- x86の場合、サイズだけではなく速度が重要
  - ビット長にバリエーションがあると、高速に実行するのが難しくなる

- 第三世代OSASKの場合、

  小さいコード → 実行しやすいコード

  に変換してから実行する（だから小さくても遅くない）
    - 変換も十分高速で、ロード時間より短い
- これはJava-VMでも当たり前に行われている技術
  - それなのにJavaは小さくない・・・

定数をどう表すか？

- プログラム中に現れる定数で一番多いものは？
  - 0
  - 1
  - その他1桁の数字（2, 3, 4, …）
  - たまに2桁の数字（15, 23, 97, …）
  - 3桁、4桁、など・・・
- だから小さな数字をいかに無駄なく記述できるかが鍵となる

継続ビット方式

- 定数が3ビット以内で記述できる場合：
  - XXX0：Xの部分がデータ（2進数）
- 6ビット以内で記述できる場合：
  - XXX1 XXX0：「1」が継続を表す（合計で8ビット）
- 9ビット以内で記述できる場合：
  - XXX1 XXX1 XXX0：合計で12ビット

hh4

- 前述の継続ビット方式:
  - 最後まで読まないとデータ長が分からない
  - 別にそれでもいいんだけど、なんか気になる
- だから「継続ビット」を前に集める
  - 0XXX　　[XXX0]
  - 10XXXXXXX　　[XXX1 XXX0]
  - 110XXXXXXXXX　　[XXX1 XXX1 XXX0]
- 第三世代OSASKで採用

なぜ3ビットずつなのか(1)

- デメリット:
  - 3ビットずつの場合、最大で上位2ビットが使われずに無駄になる可能性がある
- メリット:
  - 3ビットずつの場合、継続ビットは全体の25%を占める（継続ビットはデータ長を表すために追加されたビットなので無駄）
  - 比較: 2ビットずつの場合、継続ビットは33%を占める
  - 比較: 1ビットずつの場合、継続ビットは50%を占める

なぜ3ビットずつなのか(2)

- 3ビットずつにしておくと全体長が4の倍数になるので、バイナリエディタで見た時にすごく見やすい

- もしかしたら2ビットずつや4ビットずつのほうがいいかもしれない
  - 詳細な比較検討はしてない

ほぼ全部hh4

- 第三世代OSASKでは、以下をすべてhh4にしている
  - コード内の定数
  - レジスタ番号
  - 命令番号

- だからなんでも短く書ける上に、上限はない
  - x86は命令番号が足りなくなって苦労していた
  - レジスタ番号もいっぱいになってレジスタを増やすときに苦労していた

命令番号(1)

- よく使う命令は小さな命令番号を割り当てておくと、4ビットで記述できる

- 第三世代OSASKで選んだもの:
  - 0 : NOP
  - 1 : ラベル宣言（分岐先の宣言）
  - 2 : レジスタへの定数代入
  - 3 : ジャンプ
  - 4 : プリフィクス（後述）
  - 5 : API呼び出し
  - 6 : ループ開始
  - 7 : ループ閉じ

命令番号(2)

- その他の命令:
  - 0x10 : OR
  - 0x11 : XOR
  - 0x12 : AND
  - 0x14 : ADD
  - 0x15 : SUB
  - 0x16 : MUL
  - 0x18 : 左シフト
  - 0x19 : 右シフト
  - 0x1a : DIV
  - 0x1b : MOD
- このほかに条件分岐などがある(0x20～)
- 整数命令はほぼ 0x3f までに収まっている
  - 命令の種類がかなり少ない(シンプル)

例1

- 整数レジスタR00に3を足す:
  - ADD(R00, R00, 3);
  - 14 0 3 [生データ]
  - 94 0 3 [hh4化]

  - たったの2バイトで書ける！

- x86では3バイトを要する [83 C0 03]
  - 第三世代OSASKは3割も小さい

例2

- 三項形式「 R01 = R00 + 3 」
  - ADD(R01, R00, 3);
  - 4 14 0 3 1 ： 4は演算命令を三項型に切り替える
  - 4 94 0 3 1 [hh4化]
  - たったの3バイトで書ける！

- x86でやるとこうなる [8B C8 83 C1 03]
  - MOV ECX,EAX / ADD ECX,3
  - 合計5バイト（第三世代OSASKは4割も小さい）

さらに・・・(1)

- 頻出の定数への対応：
  - プログラムでは、以下の定数がビット長が長いにもかかわらず、しばしばでてくる
  - 0x20, 0x40, 0x80, 0x100, 0xff, 0x7f, 0x3f
  - hh4では、符号付き整数を扱うと、-0x20～+0x1fの範囲しか8ビットで書けない
  - そこで負の数のいくつかを犠牲にして、上記の定数を8ビットで書けるようにしている
    - 統計を取ってみると、負の数はめったに使わないので
  - 犠牲になった負の数は12ビット形式で書く

さらに・・・(2)

- ADD命令で0を指定することはまずない
  - R00 = R01 + 0;
  - これを書くくらいなら単純に代入にしますよねー
- SUB命令でも0はまずない
- MUL命令では0や1や2はまずない
  - 2は左シフト命令を使わせる
- DIV命令でも0～2は使わない

- だから、4ビット形式でこれらの値が使われたら、他の値を意味するように割り当てを変更する
- これで4ビット形式の出番が増える

レジスタはたくさん

- 第三世代OSASKでは、レジスタは最低でも64本
  - x86では64ビットモードでさえたったの16本
- レジスタ番号的にはもっと増えても構わない
- ほとんどの変数はレジスタに置く
  - 配列や構造体はメモリに置く
- メモリアクセス命令が激減
  - これがサイズを小さくすることに貢献
  - レジスタは指定しやすい

メモリアクセスについて

- メモリアクセスはセキュリティホールになりやすい
  - ポインタのバグ、
  - バッファオーバーラン、
  - メモリリーク、
  - 型を間違えて値が壊れる、など
- メモリアクセスはマルチスレッド環境でアクセス競合関係のややこしいバグの温床にもなる
- 第三世代OSASKでは、メモリアクセスに際して多くのチェックをするのでメモリアクセスが遅い
  - そのかわり安全
- だからレジスタを増やして、本当に必要なメモリアクセス以外をなくすようにしている
  - x86ではレジスタが足りないのでメモリで代用することがしょっちゅうある

開発コードネームOSECPU

- OS : OSとの連携を意識した仕様
- SEC : セキュリティを意識した仕様
- CPU : CPUの命令から作る

- これらを組み合わせて「OSECPU（おせくぷ）」

リピートレジスタ

- レジスタ番号0～7番は4ビットで指定できる
- レジスタ番号8～63番は4ビットでは指定できない（8ビットが必要になる）
- うまい方法はないか？ → リピートレジスタ

- 直前に代入されたレジスタを指定できる
  - もちろん4ビットで
  - 2個前に代入したものも指定できる
  - 結構効果がある

ここまでのまとめ

- このような努力の積み重ねにより、第三世代OSASKは驚異的な小ささを実現している
- 「ずる」はどこにもない！ ただ無駄を削っただけ
- でもhh4なので全体の25%は制御ビットで無駄
  - これをどうしたらいいのかはわからない

本質とは？

- 無駄を削って、削って、それでも残ったもの：
  - これは何だろう？
  - これが本質だろうか？

- プログラムを小さくすることは、メモリもディスクも大きくなった今では無意味かもしれない
- でも本質を知ることは無意味ではないと思う

- 「私はプログラムの本質を世界一知っている」

- ちなみに現状は、本当は430バイトで書けるプログラムを10KBくらいで書いている（もっとひどい人もいる）
  - 500円の価値しかないものに1万円を払っているようなもの
  - しかもみんな疑問すら感じていない
  - これはちょっとひどすぎると思うときはある

私の夢

- 無駄をなくすとどこまで行けるのかを知りたい
- 自分が自力でどこまで迫れたのかを知りたい

- でも無駄の全くない世界で暮らしたいわけではない
  - それはちょっと息苦しそうだもん

データ圧縮との関係

- 私の別の趣味：「データ圧縮」
- 他のOS用のプログラムを圧縮したら、第三世代OSASKと同じ大きさにならないだろうか？
- だって圧縮は無駄を削ることだから
  - 復元できる範囲で削りまくる
- 答え：ならない
  - つまり「既存のOS＋既存のデータ圧縮アルゴリズム」では、このサイズは実現しない
- なぜできないのかを調査したら、データ圧縮技術も進歩するかもしれない

ソースコード行数と実行ファイルバイト数

- ソースコード行数と実行ファイルバイト数

アプリ名	ソースコード行数	実行ファイルサイズ
bball	30行	63バイト
コルモゴルフ	82行	111バイト
インベーダー	241行	430バイト
オセロ	287行	476バイト

- ただし、言語はC言語ではない
  - C言語みたいな独自言語
  - 1行あたり、1.5～2.1バイトになる感じ

bballのソースコード

```
#include "osecpu_ask.h"

%define L_POINT  LOCAL(0)
LOCALLABELS(1);

//  PLIMM(P01, L_POINT);

DAT_SA(L_POINT, T_UINT8, 16 * 2);
    DB(196, 100, 187,  61, 164,  29, 129,   9,  90,   5);
    DB( 53,  17,  23,  44,   7,  81,   7, 119,  23, 156);
    DB( 53, 183,  90, 195, 129, 191, 164, 171, 187, 139);
    DB(196, 100);
DAT_END();

    Int32s col:R00, x0:R01, y0:R02, x1:R03, y1:R04, i:R05, j:R06;

    for (i = 0; i != 15; i++) {
        REM34(); LMEMOPP(32, R01, T_UINT8, P01); LMEMOPP(32, R02, T_UINT8, P01); // x0, y0.
        PLIMM(P02, L_POINT);
        for (j = -8; j != 8; j++) {
            col = i - j;
            REM34(); LMEMOPP(32, R03, T_UINT8, P02); LMEMOPP(32, R04, T_UINT8, P02); // x1, y1.
            if (col <= 0) { col = 1 - col; }
            if (col <= 7) {
                api_drawLine(MODE_OR, col, x0, y0, x1, y1);
            }
        }
    }
```

この成果のために必要だったもの

- 論文とか一つも読んでない
- 二次方程式も微分も積分も必要ない
- 英語も必要ない
- きっと小学生でもわかるレベル

- 執念は必要
- 根性も必要
- 究極への強いあこがれ
- よきライバル
- 「ここで油断したら、誰かに先を越されて、一生後悔するんじゃないか？」と恐怖におののく

参考情報

- osecpu.exe（29.0KB）のソースコード：

ファイル名（計12個）	行数（計7548行）
api.c	1024行
debug.c	204行
decode.c	2807行
driver.c	629行
extend.c	23行
float.c	265行
integer.c	529行
osecpu-vm.c	345行
osecpu-vm.h	239行
other.c	546行
pointer.c	298行
tek.c	639行

3.93バイト/行

第二部

時間があったら話したいけど、
なかったらパスしてもいい内容

第二部の目次

- 「正しい」ことよりも大事なこと（17分）
  - 要するに間違っていても「やった人」が勝者になる話
- チームワークについて（7分）
  - 弱点を克服することって必要？
  - 学校のテストは満点があるからいけない
- 競争・ライバルについて（10分）
  - 40位の人は100位の人を笑えるのか？
    - 平均より上とか下って意味があるのか
  - よきライバルを見つけよう！（よき師匠よりもいいかも）
- お得な情報の紹介
  - 未踏、未踏ジュニア、セキュリティキャンプ、サイボウズ・ラボユース

「正しい」ことよりも大事なこと

要するに、間違っていても
「やった人」が勝者になる話

[1] 私の学生時代の夢

- Windowsを超えるOSを作りたい！
  - 当時のWindowsの仕様に不満だった [1995年]
  - 当時のLinuxにも満足できず
  - 誰も作ってくれないから自分で作る

- 主な不満点:
  - どうも遅い気がする ＆ メモリやディスクを使いすぎでは？

[2] 世間の反応

- 無理だから、やめるべき
  - 個人グループが大企業に勝てるわけない
  - 川合とかいう無名の学生に何ができる？
  - 夢を見過ぎている、人生を無駄にしている
  - こいつは何でも自分で作りたい病なのでは？（誤解）
- むしろ有害だ
  - Linuxに合流して開発力を集結させるべきなのに、川合は悪い例を作っている

[3] 私の反論

- 「無理だから、やめるべき」に対して
  - やってみないと分からない
  - なんとなくできるような気がする（根拠のない自信）
  - 仮にできなくても、あなたの迷惑にはならないよ、なのにどうしてこんなに非難されなきゃなんないのさ！
- 「むしろ有害だ」対して
  - 一つにまとまるべきという発想が理解できない
  - 選択肢は多いほうがいいし、多様性は重要なのでは？

[4] 今にして思えば

- 「世間の反応」は、とても合理的
  - 可能性とかを考えたら、確かに言う通り
  - 批判した人たちも、まったくの善意からだったと思われる
  - 私の反論も、まさに子供の屁理屈レベル

- しかし私はまったくいうことを聞かずに独走

[5] 年表(1)

- 大学2年：OSを作ると友人たちに言い出す [1995年]
- 修士1年：ホームページを作ってアピール開始
  - そして、本気でやりたいから就職活動はしないと決めた
- 無職2年：ちょっと有名になり、それと同時に前述の批判に何ヶ月もさらされる
- 無職3年：IPAの未踏に応募したが落ちる
  - でも未踏ユースでは酷評の上で採択される [2002年]
- 無職4年：また未踏に応募したが落ちる
  - つまり有識者から見ても、私のOS作りは無理っぽい感じだったということ

[6] 未踏ユース採択コメント

- このようにOSをまた一から作り直すというプロジェクトは大きな予算ではできないが、日本の若い人の元気印が脈々と過去を繰り返し、その経験を血とし肉として成長していくプロセスを支援することが重要だとPMは考えた。
- このプロジェクトが中学生とか高校生が自分の手に届くものになればもっとよい。
- 「見せしめ」、もとい、「引き立て」に意義があると強く感じた提案。
- でも、瓢箪から駒もあり得るとPMは期待している。

- つまり新規性も必要性も全く理解されず・・・

[7] 作っていたOSはこんな感じ

- 名称：第一世代OSASK（おさすく）[2005年]
  - メモリ：8MB
  - ディスク：0.1MB
  - CPU：33MHz
  - 1秒で起動
  - 単位に注意！

[8] 年表(2)

- 無職6年：本をせっせと書く
  - 翌年に「30日でできる！OS自作入門」として発売される [2006年]
  - OSは「集団じゃないと作れないもの」ではなくなった！

  - Amazonで全200万冊中の3位になったこともあります

  - おかげさまで今までに4万部以上売れていて、韓国語版、中国語版もあります

[9] 年表(3)

- 無職10年：セキュリティ＆プログラミングキャンプの講師になる [2009年]
  - 横浜創英短期大学の非常勤講師もやり始める [2009年]

- 無職12年目にして初めて就職活動開始（お金が無いので）
  - このときすでに36歳
  - この業界には35歳定年説というのがあり、それを超えているのは非常に不利だと、転職支援サービスに言われた
  - しかも就職の経験がないなんて！
  - こいつ（＝川合）はバカか、アホか？
  - そもそも転職支援サービスの3社中2社には、相手にすらしてもらえなかった・・・
  - 相談に乗ってくれてありがとう、インテリジェンス！！
    - でもインテリジェンスの紹介してくれた企業には就職しなかったけど・・・ごめんね！

[10] お金の話

- 無職時代を乗り切るために私がしたこと
  - 「とにかくお金を使わない」・・・これに尽きる
  - 国民年金と国民健康保険と奨学金の返済をがんばる、他の出費はしない
  - 親にののしられてもとにかく引きこもりを装って耐える、実家から絶対に出ない
  - お金が無くなったら、もう続けられないから
  - 学生時代の貯金を使って、ここまでできた
    - 未踏ユースのお金や、書籍の印税や、講師料で延命して、合計11年間できたことになる

[11] 就職活動

- ずっと憧れていた「サイボウズ・ラボ」にダメ元で応募
  - 「何か新規のものを作るのは得意じゃないけど、高速化は得意です」
  - 「自分のやりたいことはもうたくさんやりました、これからは会社のためになることをやらせていただきます」
  - 「でもとにかく何でもやりますので、どうかよろしくお願いします」
  - もう36歳だし、社会人経験ないし、ダメだろうなと思っていたけど、採用してもらえた！！ [2011年]
  - ありがとうございます！

[12] 就職後の活動(1)

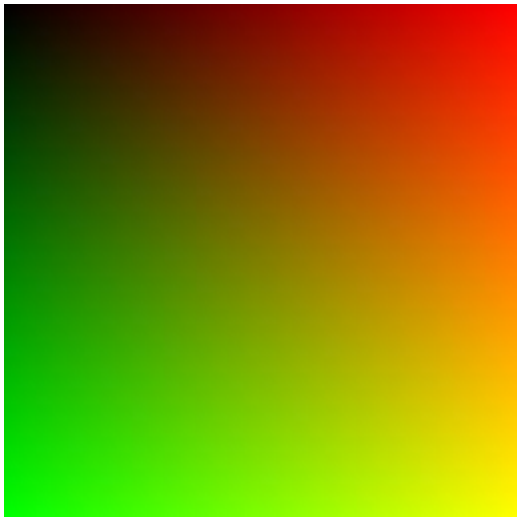
- 本社のサイボウズから、製品が遅くて困っているという話をもらうたびにどんどん解決！
  - 自分の知らないプログラミング言語でも引き受ける
  - 3割くらいしか速くならないこともあるけど、たいていは3倍以上速くできる
    - 8000倍速化を達成したこともありました
- みんなから感謝されて幸せ
  - お役に立ててよかった！
- さすがにOS開発はできないよな･･･と思ったら

[13] 就職後の活動(2)

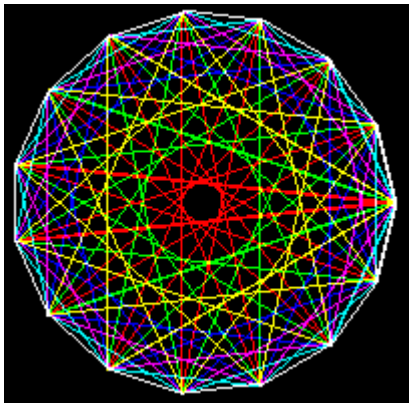
- セキュリティキャンプの講師を会社の業務でやらせてもらえた
  - そこでOS作りの指導ができて、「夢の続き」をやることができた（間接的だけど）
  - OSECPU-VM（おせくぷ・ぶいえむ）[2014年]
    - Java的なものを独自に作った：就職前からやりたかった
    - 自分でも信じられないほど、すごいものができた！（世界最小記録を大幅に更新）

[14] 就職後の活動(3)

- OSECPU-VM（おせくぷ・ぶいえむ）[2014年]
  - VMも、アプリも、おかしいくらいに小さい！
  - Windows用のVMは30KB（OSASKは80KB）



8バイトでグラデーション
（OSASKでは327バイト）



63バイトでこんな模様
（OSASKでは186バイト）

430バイトでこんなゲーム
（OSASKでは1108バイト）

[15] 正しかったのは誰か？

- この20年間を総括すれば:
  - 私はWindowsを超えるOSを実現できていないので、その点において正しかったのは彼らです
  - もちろんもっと未来では実現するかもしれないけど、とにかく20年間ではできていないわけです
  - ある意味では（サイズ面とかでは）もう超えているといえるけど、20年前の私や彼らが目指していたものはもっと大きかったのです

[16]「正しい」よりも重要なこと

- 批判を無視した私はこうなった
  - では、私を批判した「正しい」人たちはどうなっただろうか？
  - 私よりも幸せになれただろうか？ → いいえ

- 正しくないのに大成功した例：クリストファー・コロンブス
  - コロンブスは地球の半径を誤算していた
  - だから、当時の航海技術でも、ヨーロッパからアジアに行けると思った：大間違い
  - ポルトガルは正しい半径を知っていて、コロンブスの提案を無理だろうと拒否 [正しい]
  - スペインはコロンブスの提案を受け入れて、資金援助した

  - で、莫大な富を得たのはどちら？
    - （コロンブスは死ぬまで自分はインドに行ったと信じていた）

[17] いつあきらめるべきか

- 「成功者」はたいていこう言う:
  - 「あきらめない気持ちが一番大事」
  - そりゃ、あんたは成功したから、そう言うだろうさ！
- 誰でもあきらめなければ成功するのか？
  - そんなわけない
- 私は夢を叶えるためではなく、夢をあきらめるために、夢を追いかけた
  - やってダメならあきらめられる
  - 後悔しないためには、全力でやるしかない
  - 期限は重要：〇〇年やってダメならあきらめる、お金が無くなったらあきらめる
  - あとになって誰か成功者が出て、「自分だってやればできたはずだ」と言うのは絶対に嫌だ、素直に「あいつはさすがだ」と言えるようになりたい

チームワークについて

弱点を克服することって必要？
学校のテストは満点があるからいけない

[1] チームワークとは

- 一人ではできないことをみんなでやる
- お互いの得意分野を持ち寄る

- 絵がうまい人はデザインを
- 音楽ができる人はBGMを
- プログラミングが得意ならプログラムを

  – 適材適所

[2] どちらをメンバーに選ぶ？

- [Aさん] 絵も音楽もプログラムもできるけど、どれも70点くらいの人
- [Bさん] 絵は全然描けない、音楽もひどい、でもプログラムを書かせれば誰にも負けない人

- 私だったら、Aさんはいらない（ごめんね）
- Bさんにプログラムを担当してもらう

[3] 点数で考えると

- チームワークではメンバーの「得意分野」の点数のみが重要
- 不得意分野が0点でも50点でも70点でも関係ない（担当しないから）
- 得意分野が2つあって、それが98点と96点とかなら、どっちも重要

[4] 万能型の人の役割

- どの分野もずば抜けてはいないけど、平均点としてはかなりよい（オール80点とか）の人：

- そんな人は、たった一人で頑張るときとか、
- 誰かが病気になった時のための補欠とか、
- 全体を理解して調整する役目とか、

- そういう仕事なら適任かもしれない

[5] 弱点の克服？

- 学校では、できないところをできるようにしましょうと言われる･･･よね？

- そういう弱点の克服はもちろんいいけど、でも本当は「できるところをもっとできるようにする」ほうが社会に出てからは役立つと思います！

[6] 得意を極めると

- 何か一つ圧倒的にできるようになると、すごいチームに入れてもらえることがある
  - まさに一流のチーム
- そこには教え方もすごくうまい人がいて、そのチームにいるだけで苦手が克服できることもあるくらい
  - うまい人がどうやるかを見ているだけでも参考になる
- 数学を極めたら、英語の文献を読む場面が増えて、英語もけっこうできるようになった、などの展開もありうる

[7] 学校の成績のせい？

(1) 学校の試験には満点がある
(2) 満点以上に勉強しても成績は良くならない
(3) だから弱点の克服をやるしかなくなる
(4) それで自分の欠点ばかり意識するようになる
(5) 長所が伸びない

学校の成績がいい人ほど、この罠にかかりやすい

部活や課外活動は、満点がない世界なので、そこで頑張ると、(4)の悪い習慣に染まらずにすむかも

小学3年生が、小学6年生の内容を理解していたら、800点とかをつけてあげられたらいいのに・・・

競争・ライバルについて

40位の人は100位の人を笑えるのか？
よきライバルを見つけよう！

[1] 競争は好きですか？

- おそらくここにいる皆さんは競争が好きだと思います
  - ちなみに私も大好きです！

- 競争に勝てるから競争が好きなのか？
- 競争が好きだから競争に勝てるのか？

- ちなみに私は自分が負けても競争が好きです

[2] 競争を嫌がる人たち

- 競争が嫌いな人
  - 負けるのが嫌だから？
  - 負かすのが嫌だから？

- うーん、負けるってそんなに嫌なこと？

[3] 2位は1位の代用

- 1位は文句なくすごいです！

- お店でお菓子を買います:
  - もし同じ値段なら、あえて2位を買いますか？
  - 1位が売り切れとか、1位に嫌いな食材が入っているとか、そういう事情があって、それで2位を買うわけですよね？

[4] 1位以外は「負け」

- 結局のところ、1位以外は負けだと思います
- まあ3位くらいまではいいかもしれません
- いや、10位でもいいです

- でも40位とかってどうでしょうか？
- もっというと40位は100位を笑えるのでしょうか？
- 40位の人に出番はありますか？出番がないという意味で、もはや100位の人と同じじゃないですか？

[5] 平均との比較に意味があるか？

- 「自分は平均よりできる、できない」とかいう会話を耳にすることがありますが、これに何か意味があるでしょうか？
  - 100人中の、40位と60位にどんな違いがある？？
  - 結局どちらも出番はないのでは？
  - 使わないスキルが高くても低くても関係ないですよね

- 平均より上か下かなんかで一喜一憂せずに、ぜひ1位を目指しましょう！

[6] 負けることは当たり前

- ほとんどのことで1位にはなれないので、私は ほぼいつも負けています

- でもくやしいとは思いません
- 純粋に1位の人がすごいと思います

- 負けることが嫌じゃないです
  - 当たり前の普通のことなんです
  - でも全力を出せずに負けるのは嫌ですよ

- 負かされることも嫌じゃないです
  - すごい人に会えてうれしいです！

- もし競争しなかったら、私は誰を尊敬したらいいんですか？

[7] 負けるのが嫌な人は

- 私の想像ですが・・・

- 負けるのが嫌いな人って、実はあまり負けたことがない人じゃないですか？
  - 勝ってばかりいた人？
- だから負けるのが怖い？

- でもね、負けても何も失わないですよ
  - それであなたの能力が下がるわけじゃない
  - 評判は落ちるかもしれませんが・・・
- 尊敬できる人が増えるだけです

[8] ライバルはいますか？

- N氏は私のライバルです

- 勝ったり負けたりできる相手がいるといいです
- その人と競争しているうちに、実力が付きます
  - 私の経験上では、競争相手がいないと成長が遅いです
  - 仲間がいるのは楽しいけれど、大きく成長する機会をくれるのはいつもライバルです

- 「よい先生」と「よきライバル」はたぶん同じくらいの価値があります
  - もしかしたらライバルのほうが少しいいかも
  - だからよい先生を探すのと同じくらい、よきライバルを探しましょう！
  - 有能な先生の時間を自分のせいで使わせるのが申し訳ないです
  - ライバル相手ならそんな気持ちになることはないです

[9] ライバル vs 仲間

- ライバルは私の得意分野に関心があります
  - それが原因で負けるかもしれないから
- 仲間は私の不得意なところを補って手伝ってくれるけれど、私の得意分野にはあまり関心がありません
  - そこは自分の担当ではないから
- 私の得意な技術について、熱心に話を聞いて一緒に考えてくれる人はどちらですか？

[10] ライバルを大事にしよう

- ライバルが不調な時は、助けましょう
  - 敵に塩を贈る
  - ライバルがいなくなって困るのは私です！
- レベルの高い戦いを続けましょう
  - 油断したら負けるくらいがちょうどいいです
  - 勝ったらすぐに新しい技を教え合うのです
  - 一方的になったら、お互いに成長しませんよ
  - 勝つことが目的じゃないのです

お得な情報の紹介

未踏
未踏ジュニア
セキュリティキャンプ
サイボウズ・ラボユース

未踏

- 経済産業省がIPAを通じて、個人のソフトウェア開発を支援
  - ハードウェア開発の事例もあります
- 何か新しいもの

川合は2002年度の未踏ユース卒業生

- IPAが個人に対して開発を発注してくれる
  - 9ヶ月で200万円くらい
- 受け取った費用は自分の人件費にもできる
- 作ったものは自分のものにできる

  - 優秀な若者がコンビニでアルバイトするのはもったいない、という発想が原点

未踏ジュニア

- アイデアはあるけど未踏には届かない
- そういう人のための支援事業

- 未踏社団が主催（未踏社団＝未踏卒業生の集まり）
- 人件費は出ません（ごめんなさい！）
- 機材購入費はある程度助けられます

川合は未踏社団の人

セキュリティキャンプ

- お盆休みの時期に4泊5日で、セキュリティの勉強をします
- 内容はハイレベルです

川合は2009年からずっと講師です

- セキュリティを学ぶといろいろと実力が付きます
- セキュリティキャンプ卒業生が未踏にいくパターンも最近はよくあります

サイボウズ・ラボユース

- 「未踏」は経済産業省の事業
- それに対してラボユースは民間企業による事業
  - （慈善事業です）

- ラボユースは1人当たり100万円程度の予算
  - 自分の作りたいものを作って、アルバイト代がもらえる仕組み
  - サイボウズ・ラボの社員がアドバイスもします
- ラボユース卒業生が未踏に行くパターンもよくあります

川合はサイボウズ・ラボの社員

まとめ

長い話をまとめました

まとめ

- 第三世代OSASKを開発して、世界一小さいアプリケーションプログラムを作れるようになりました
- 「正しい」ことよりも大事なこと
  - 要するに間違っていても「やった人」が勝者になる話
- チームワークについて
  - 弱点を克服することって必要？
  - 学校のテストは満点があるからいけない
- 競争・ライバルについて
  - 40位の人は100位の人を笑えるのか？
  - よきライバルを見つけよう！（よき師匠よりもいいかも）
- お得な情報の紹介
  - 未踏、未踏ジュニア、セキュリティキャンプ、サイボウズ・ラボユース