

# Emscriptenを用いたx86エミュレータの ブラウザへの移植

---

Arch

B1 sksat

# 背景

- 過去にx86エミュレータを作っていた
- WebAssemblyがアツい

# WebAssemblyとは

- ブラウザ上で走るアセンブリ風(?)言語
- JavaScriptを置換するわけではない
- 大してWebでもAssemblyでもない
- **ブラウザ上で高速に計算できる**
- LLVM 8.0で正式サポート(2019/03/20)



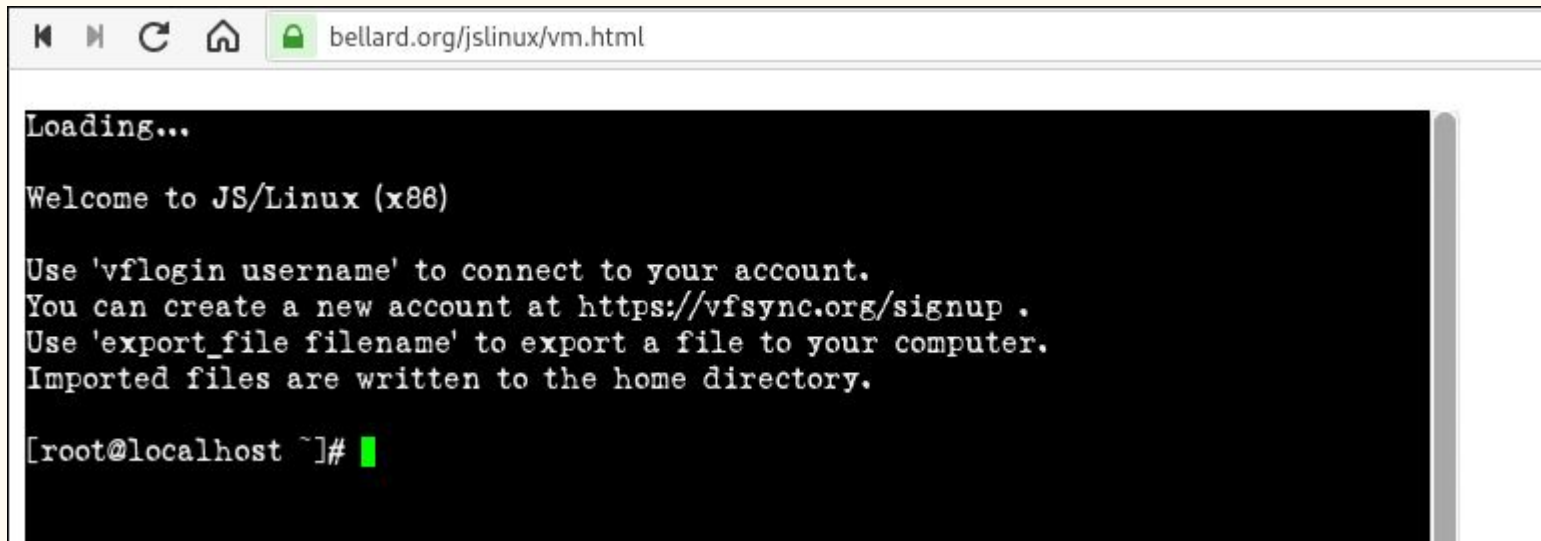
# 自作エミュレータ

- ・高2の時に作ったもの
- ・<https://github.com/sk2sat/emu> で公開
- ・ターゲットはx86
- ・コンピュータの動作原理を知りたい
- ・はりぼてOS(教育用の小さなOS)が動く
- ・マルチプラットフォームで動く

→ ブラウザでも動いたら楽しい！ ←今回のモチベーション

# 先行事例

- JSLinux
- v86



The screenshot shows a web browser window with the address bar displaying `bellard.org/jslinux/vm.html`. The main content area is a black terminal window with white text. The text in the terminal reads: `Loading...`, `Welcome to JS/Linux (x86)`, `Use 'vflogin username' to connect to your account.`, `You can create a new account at https://vfsync.org/signup .`, `Use 'export_file filename' to export a file to your computer.`, `Imported files are written to the home directory.`, and `[root@localhost ~]#` followed by a green cursor.

```
Loading...

Welcome to JS/Linux (x86)

Use 'vflogin username' to connect to your account.
You can create a new account at https://vfsync.org/signup .
Use 'export_file filename' to export a file to your computer.
Imported files are written to the home directory.

[root@localhost ~]#
```

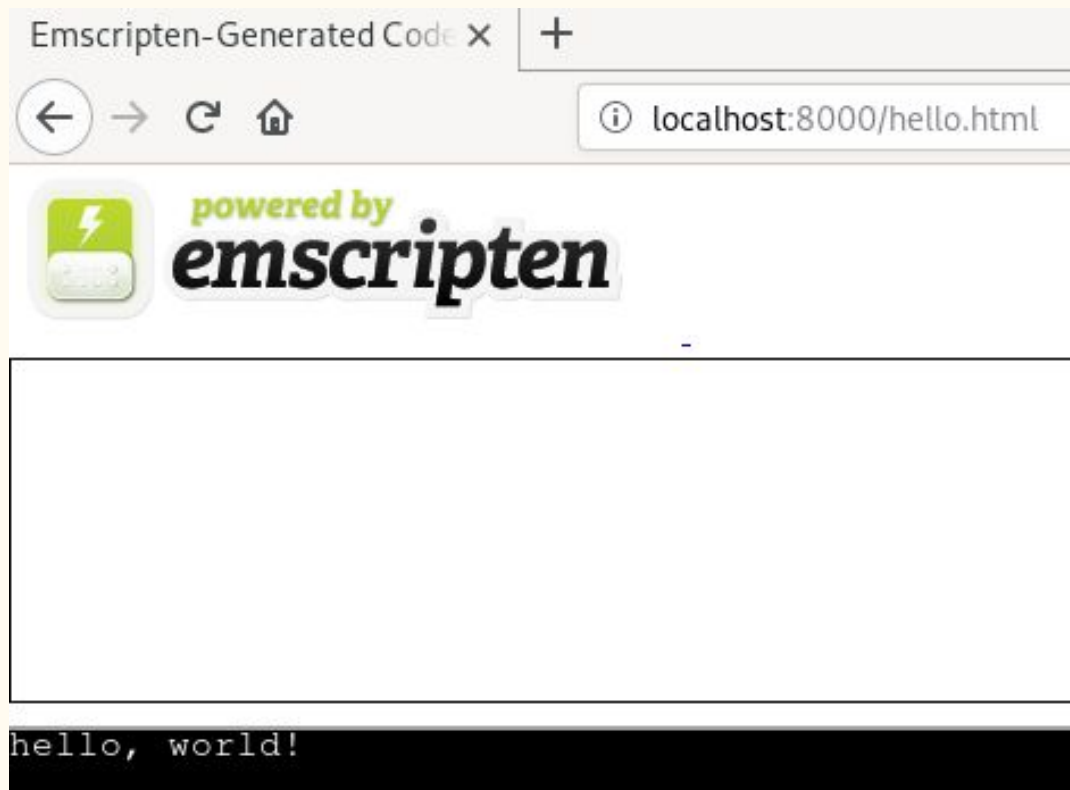
# Emscriptenとは



- asm.js, WebAssemblyへのコンパイラツールチェーン
- C/C++をブラウザ向けにコンパイルできる(! ?)

```
sh-5.0$ cat > hello.c
#include <stdio.h>
int main(int argc, char **argv){
    printf("hello, world!\n");
    return 0;
}
sh-5.0$ emcc hello.c -s WASM=1 -o hello.html
sh-5.0$ python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

# Emscriptenとは(2)



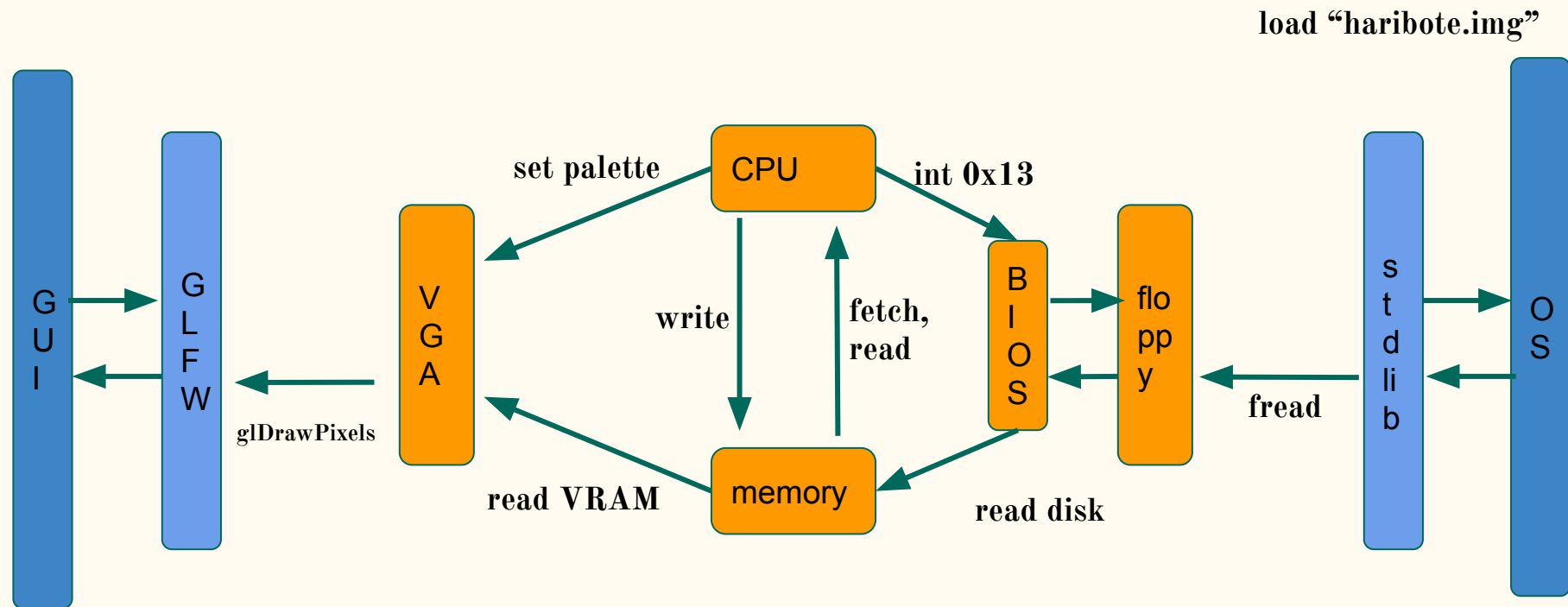
# とりあえずやってみる

```
make CC=emcc CXX=emcc
```

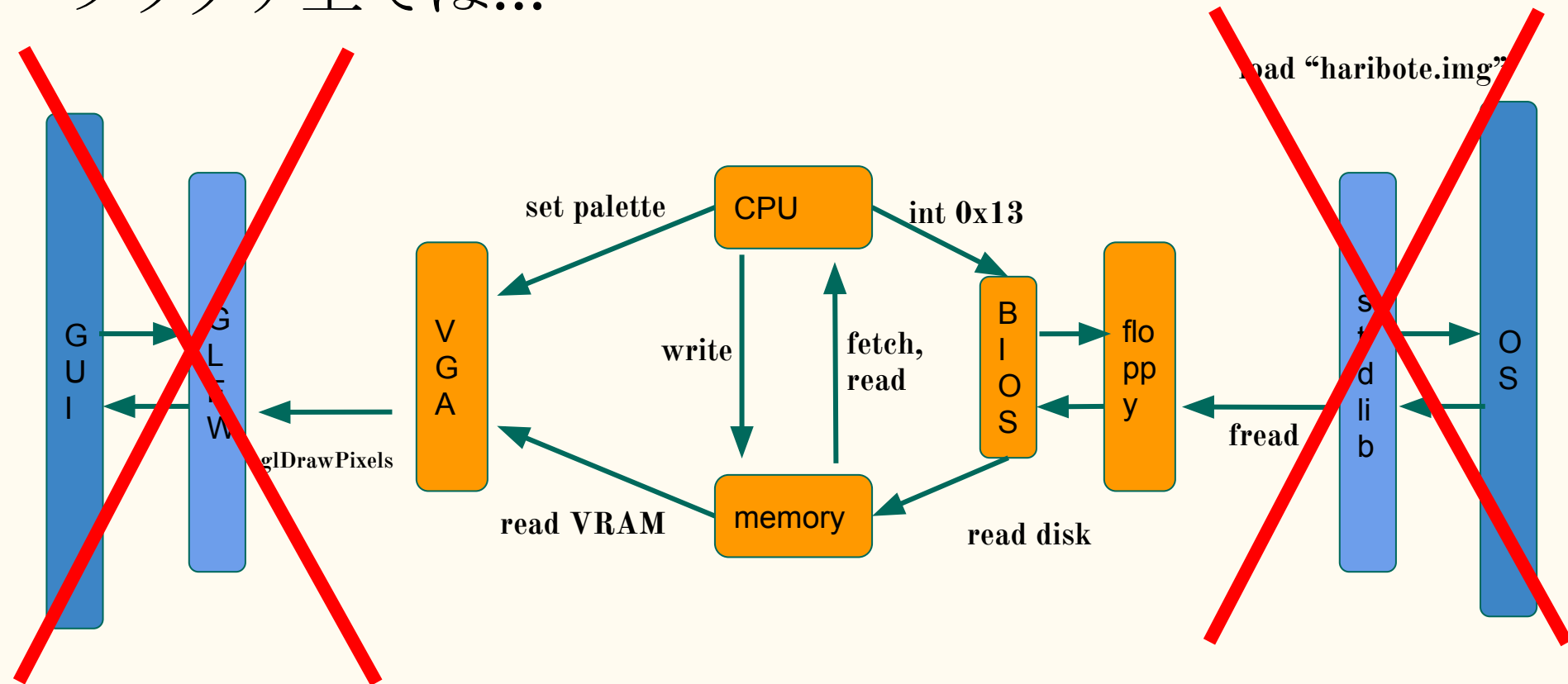
- これだけではうまくいかない
- なぜ？→ファイル読み込みと画面描画ができない



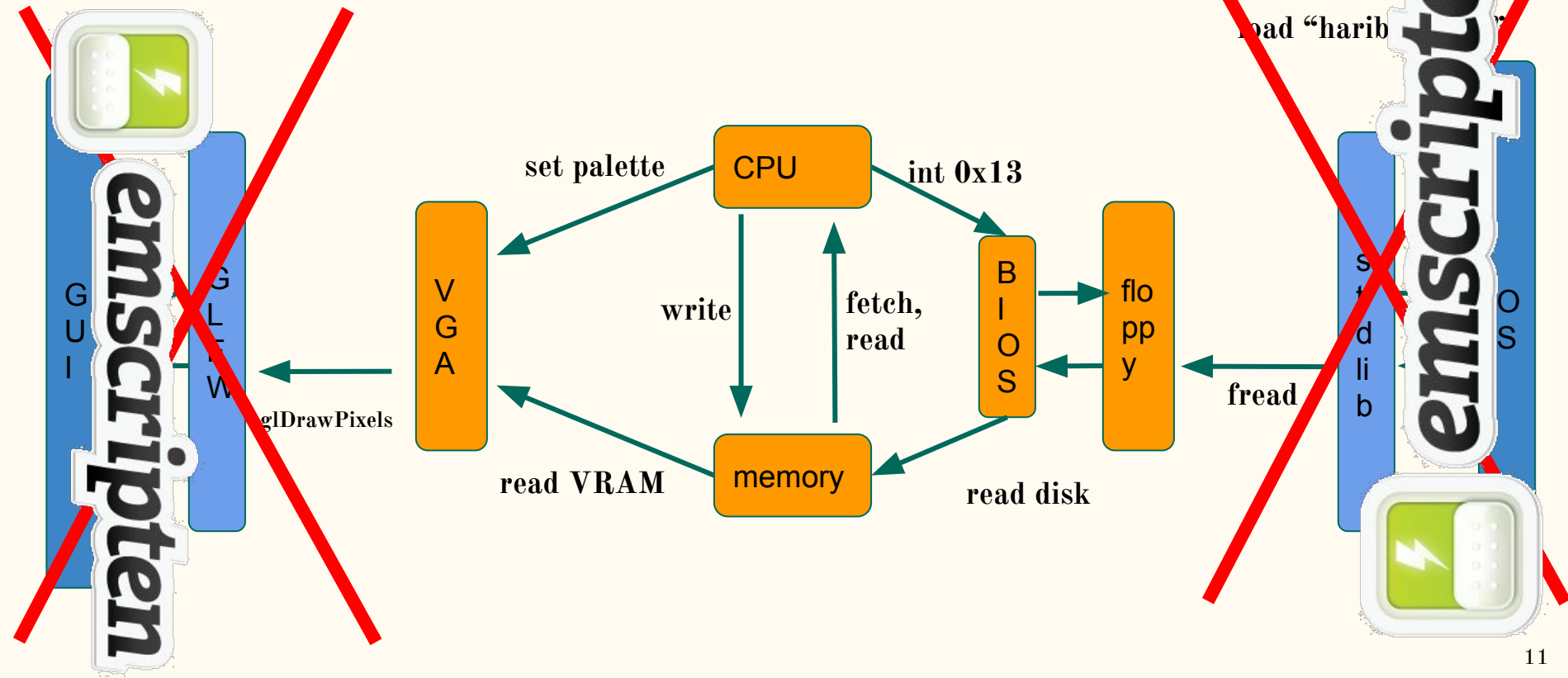
# エミュレータの構造



ブラウザ上では...



今回やろうとしたこと



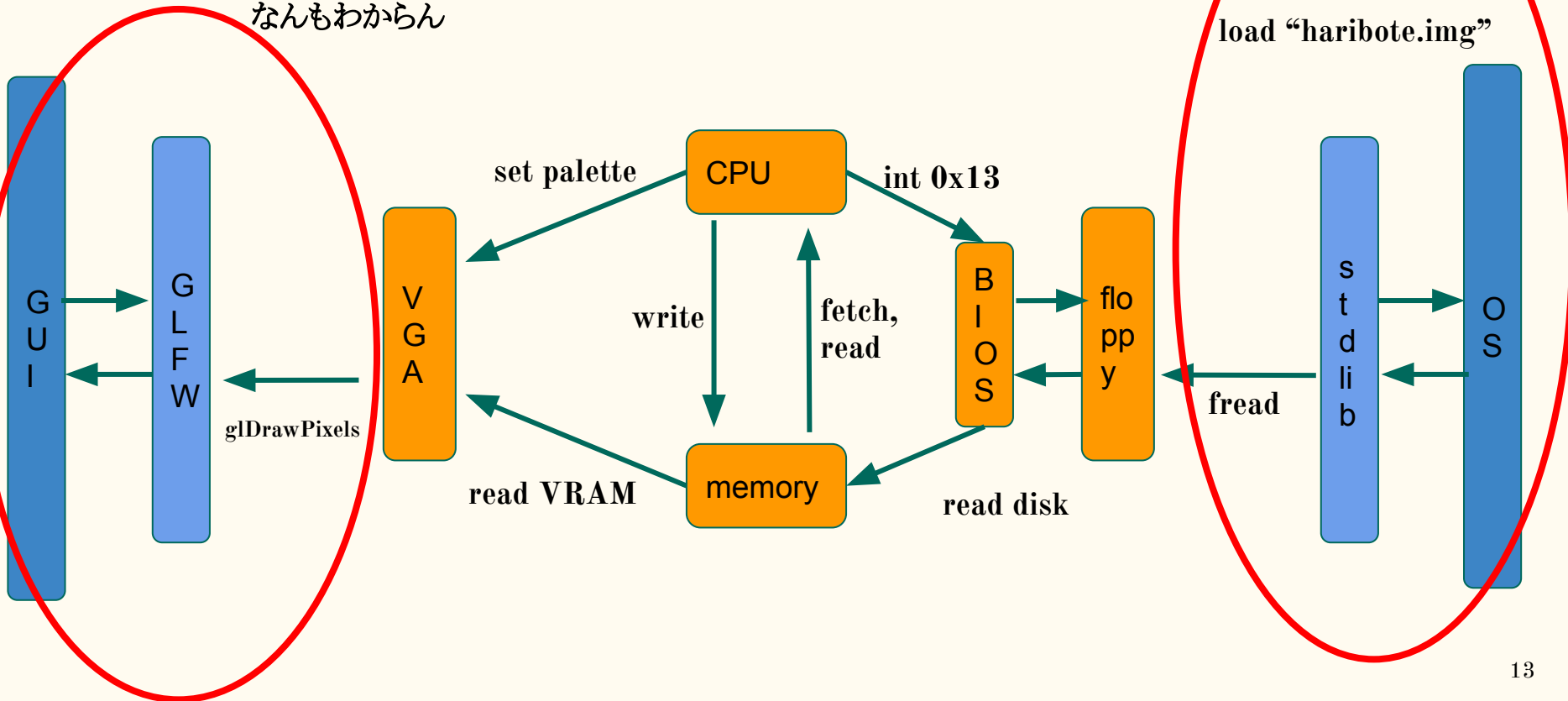
# 進捗どうですか？

できませんでした...

# 進捗

なんもわからん

できた



# なぜできなかったのか？

- WebAssemblyでのpthreadがまだproposalだった

- **WebAssembly threads support.**

- Enables support for the WebAssembly Threads proposal. Implies `#shared-array-buffer` and `#enable-webassembly`. – Mac, Windows, Linux, Chrome OS, Android  
`#enable-webassembly-threads`

- `glDrawPixels()`が使えなかった

`glDrawPixels()`はOpenGL ES 2.0で削除

WebGLはOpenGL ES 2.0相当

# やったこと

- threadが無いなら定期的に描画してしまえばいいじゃない

→main()が終了するまでDOMへの操作が反映されない

emscripten\_sleep()を入れるとそこで更新してくれるらしい

→コンパイラが落ちる

## やったこと(2)

- `glDrawPixels()`が使えないなら...

元々 RGBを1つの関数で表示したかったがためにOpenGLを使っていた

シェーダなんもわからん

そもそも `emscripten_sleep()`が使えないから実行中の描画ができない



## やったこと(3)

- ・実行中の描画はあきらめた

とりあえず終了時の画面が描画できればいい

- ・RGBを描画する関数が無いならcanvasに直書きすればいいじゃない

```
for(int x=0;x<320;x++){
    for(int y=0;y<200;y++){
        int pos = 320 * y + x;
        unsigned char red    = img[pos*3  ];
        unsigned char green  = img[pos*3+1];
        unsigned char blue   = img[pos*3+2];
        EM_ASM_(
            var canvas = document.getElementById("canvas");
            var ctx = canvas.getContext("2d");
            ctx.fillStyle = rgb2hex([$2, $3, $4]);
            c.fillRect($0, $1, 1, 1);
            , x, y, red, green, blue);
    }
}
```

# デモ

<https://sk2sat.github.io/emu/demo/emu.html>

# 今後

- ・新規にx86エミュレータを作りたい

<https://github.com/sk2sat/eucerca>

- ・キーボード/マウスのエミュレーション
- ・メモリ保護機能
- ・FPGAで自作CPU

# エミュレータの構造

