# minc

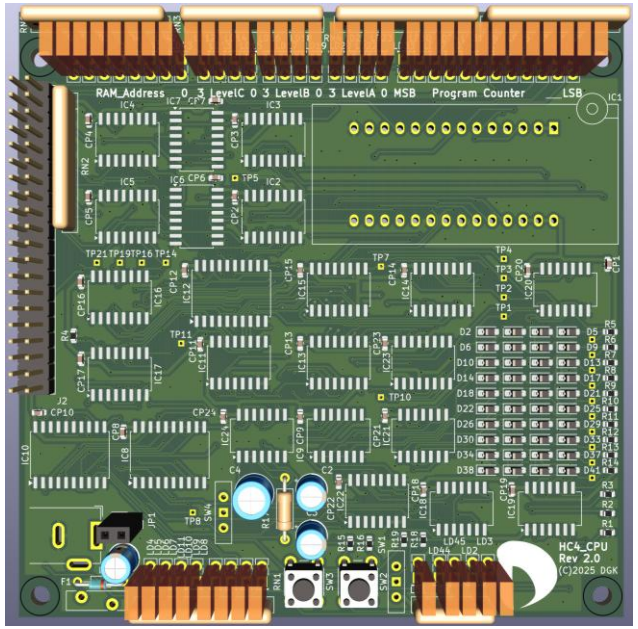
- 最小限のC言語の動くCPUを目指して -

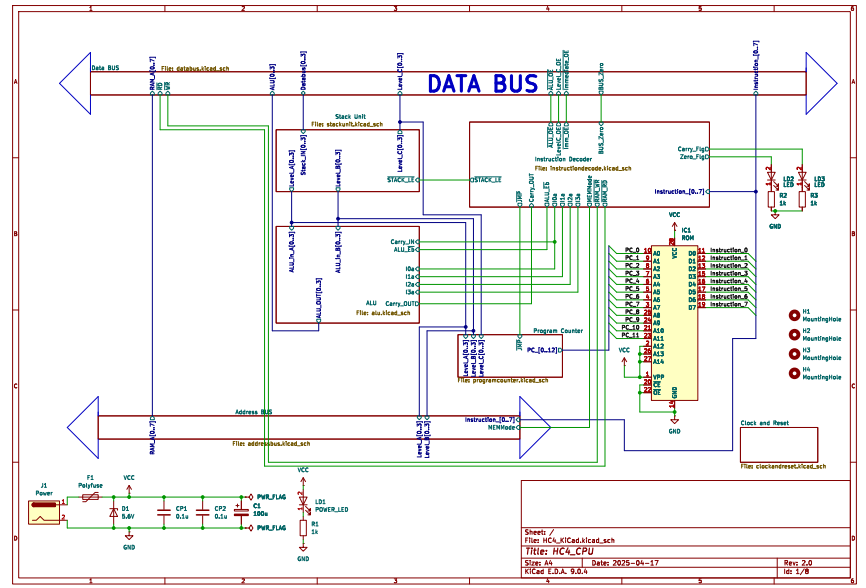
清水星那

2025/12/19

# 自己紹介

- なまえ　清水星那
- 学年　高校２年
- 生息地　東京都
- 趣味
  電子工作ほか




普段は部活で自作CPUを制作している
（語る会で発表したもの）

GitHub : https://github.com/nasu8151

# プロジェクトの概要

- プロジェクト名：minc
  Minimal C-CPUの略
- C言語の動く最小限のCPUを作る！
  →　最小限って何？
- 最小限と言っても、バイナリサイズ、回路規模、いろいろある
- 今は最小限＝最小限のゲート数として考えている

- RV32I、6502、AVRあたりには勝ちたい

# 目指すもの

- C言語に最適化されているCPU
- それに対応したCコンパイラ

- 標準ライブラリ：今のところ既存のものを移植するつもり
  出来ないものは作る

- 実装先として、FPGA（現状はTang Nano 9k）を想定
- エミュレータも作る

# 端緒

- 自作CPUを2個くらい作った
  →どちらも高水準言語が動かない！
- 例えば、スタックがない、4bit幅、メモリ不足、サブルーチンコールができないなど

- アセンブラで高度なプログラムを書くのは難しい…
  他から移植するのも難しい…
  （できないわけではない）

- →高水準言語（たとえばC）が動くようなCPUを作りたい

# 現在の進捗

- 現在、https://www.sigbus.info/compilerbook を参考にCコンパイラの実装について勉強中
- 今は条件分岐の実装を作り始めるところ
- CPUは基本的な命令セットが完成したところ

# CPUの仕様（現状）

- 変わる可能性が大いにある

- データ8bit, 命令15bit
- アドレス空間は8bit幅（256語 or Byte）
  拡張予定あり
- レジスタマシン（汎用レジスタ16本、PC、SP）
- FPGA上に実装する予定なので乗算命令あり
- アセンブリで書くことはあまり想定せず、C言語などのみに最適化する予定（人が書くことはあまり考えない仕様）

# CPUの仕様（現状）

- 命令表

| Mnemonic | Machine code | Description |
|---|---|---|
| mov rd,rs | 000 0000 dddd ssss | rd = rs |
| add rd,rs | 000 0001 dddd ssss | rd = rd + rs |
| sub rd,rs | 000 0010 dddd ssss | rd = rd - rs |
| cmp rd,rs | 000 0011 dddd ssss | rd - rs |
| mul rd,rs | 000 0100 dddd ssss | rd = rd * rs |
| push rs | 000 1000 0000 ssss | (--sp) = rs |
| lds rs | 000 1001 0000 ssss | SP = rs |
| pop rd | 000 1010 dddd 0000 | rd = (SP++) |
| sts rd | 000 1011 dddd 0000 | rd = SP |
| ret | 000 1100 0000 0000 | PC = (SP++) + 1 |
| mvi rd,n | 001 nnnn nnnn dddd | rd = n |
| stm n,rs | 010 nnnn nnnn ssss | [r15+n] = rs |
| ldm rd,n | 011 nnnn nnnn dddd | rd = [r15+n] |

# minc CPUの特徴（現状）

- レジスタマシン、ロードストア型、ハーバードアーキテクチャ
- 2オペランド命令が基本

- メモリアクセスはレジスタからの相対とPUSH/POPのみ
- レジスタ相対→スタックフレーム
- PUSH/POP→いろいろ

- Cコンパイラが必要とする機能のみ実装する

# Cコンパイラ

- 現在ほぼスタックマシンとして動作するコードしか吐けない
  （レジスタマシンなのに！）
- もちろん最適化するつもり

- 現状、参考にしたサイトの実装をほぼそのままにしている
  →今後、最適化にあたって変わっていく？
- たとえば、
  - ローカル変数・一時変数をレジスタに割り振る

# 工夫した・すべきところ

- 今までの（特にローエンドの）マイコンやレガシーなCPUはいづれもアセンブリで書くことも想定していた（I think）

- C言語のみに最適化すればそれらの構造とは異なってくるはず
- たとえば、
  - アドレッシングがレジスタ相対しかない
  - 減算時にCを反転しない（正ならCが立つ）

# 今後の展望

- 年内：
  - 条件分岐、関数呼び出し、ループくらいを実装する
- 2月くらい？まで：
  - 「第一の最適化」：Cコンパイラをレジスタマシンに適合させる
- それ以降：
  - 「第二の最適化」：Cコンパイラに必要な仕様に合わせてCPUの命令セットを調整する

- それ以降
  - 「第三の最適化」：データフロー解析などを用いてさらにコードの最適化を強化

# リンク

- mincリポジトリ
  https://github.com/nasu8151/minc