# RISC-V の Windows のコマンドプロンプトでのクロスコンパイル環境の構築

備忘録として、Windows のコマンドプロンプト上で動作する RISC-V のクロスコンパイル環境の構築手順を記す。

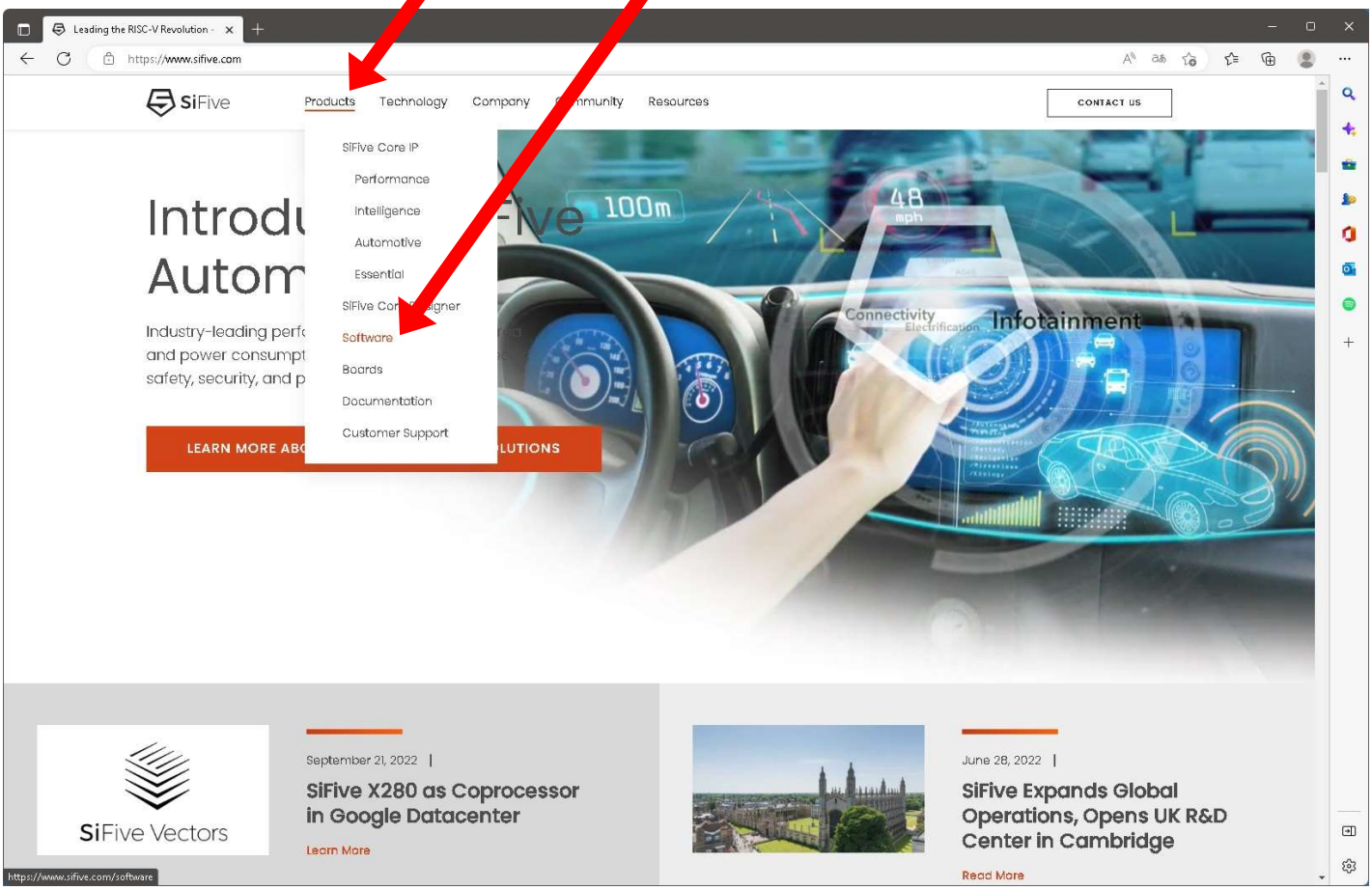
## 環境構築の手順

1.SiFive のホームページから FreedomStudio をダウンロード。
2.適当なフォルダに FreedomStudio のファイルを解凍。
3.FreedomStudio を解凍したフォルダにパスを通す。
以上

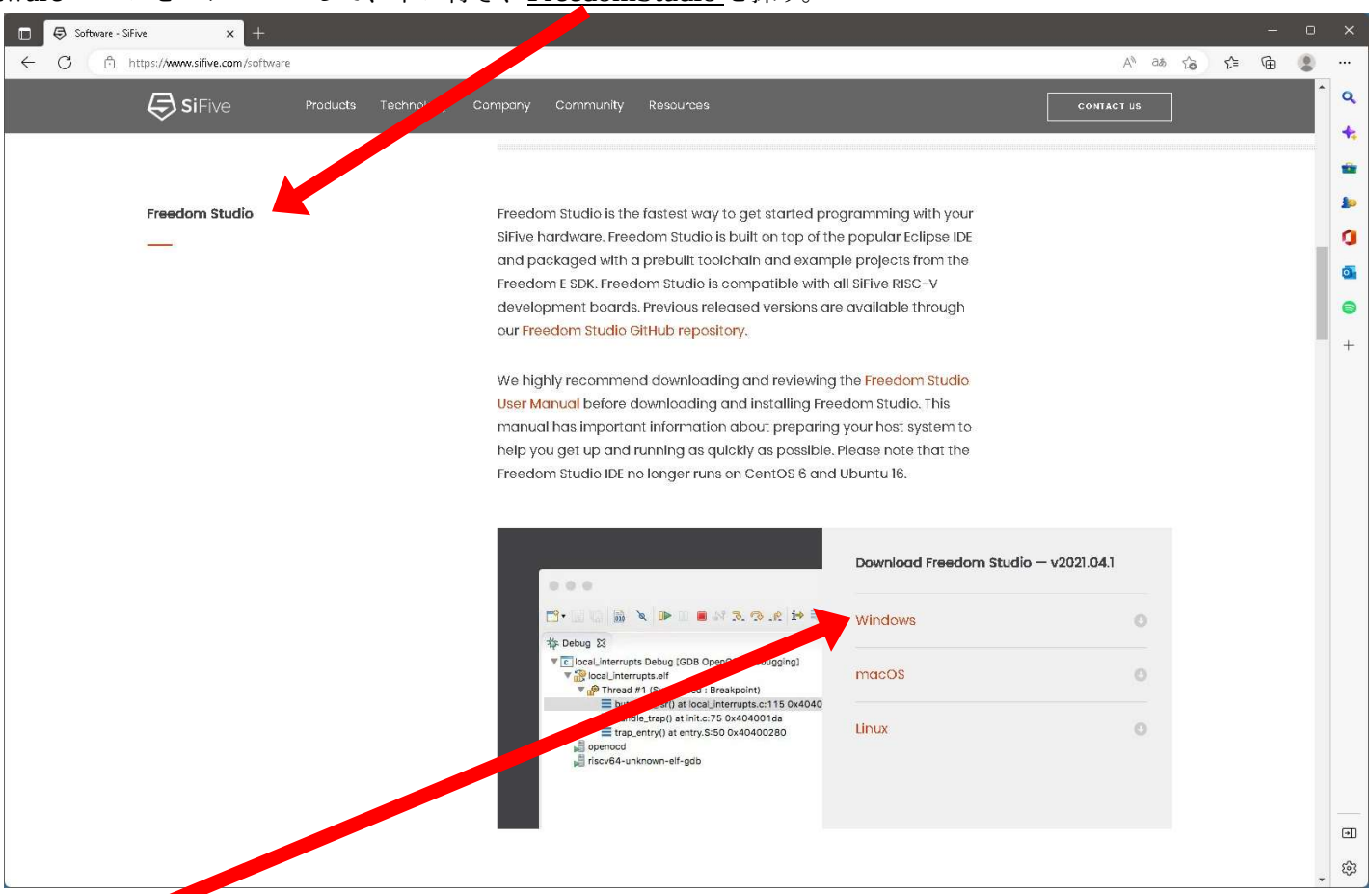
コンパイル、リンク、Verilog 用の hex ファイルの作成方法はこちら → 7 ページ

## RISC-V のビルド環境の構築

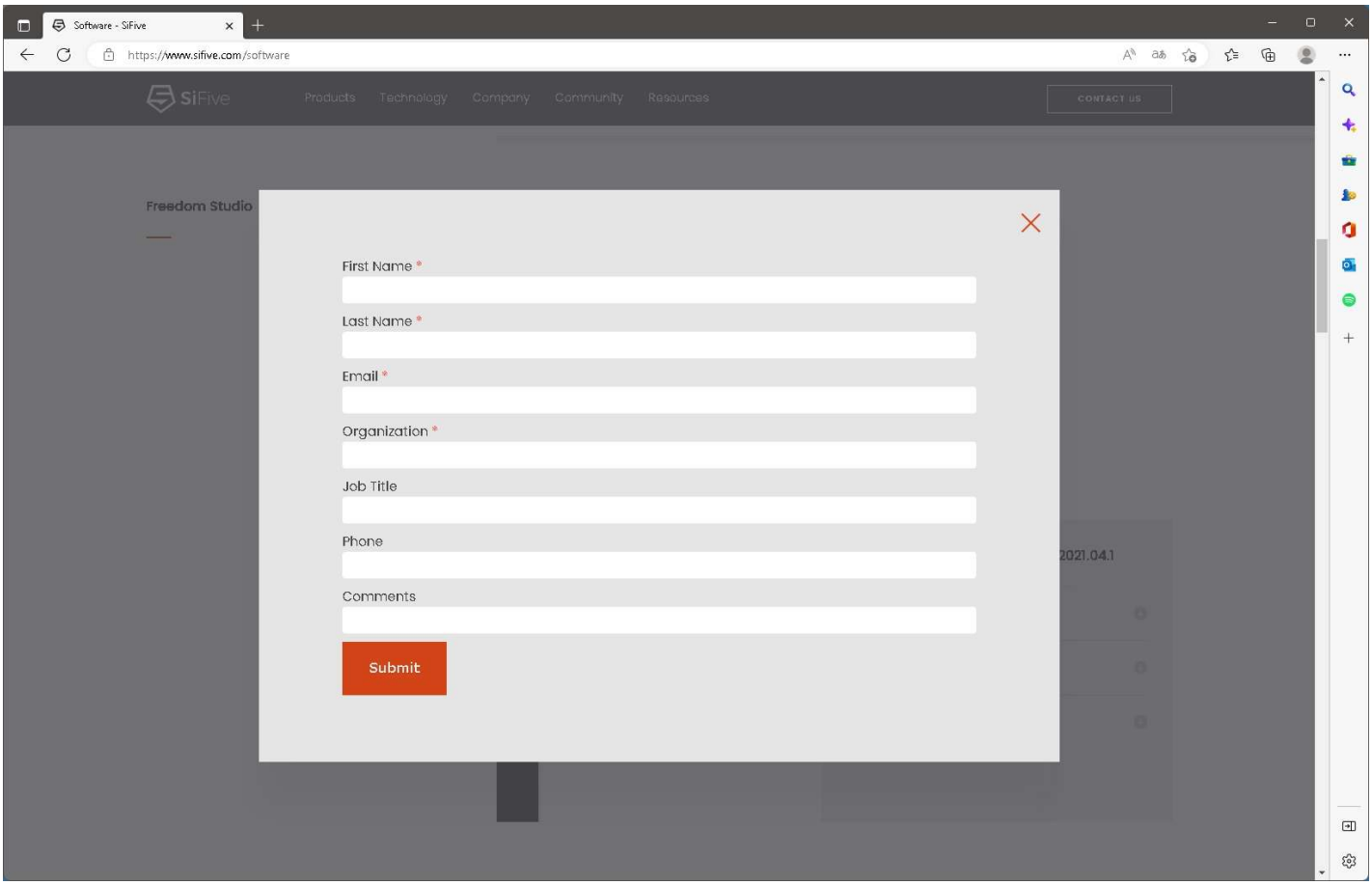
・ツールのダウンロード
SiFive のホームページにアクセス。Products の中から Software を選択する。



Software ページをスクロールして、下に行き、FreedomStudio を探す。



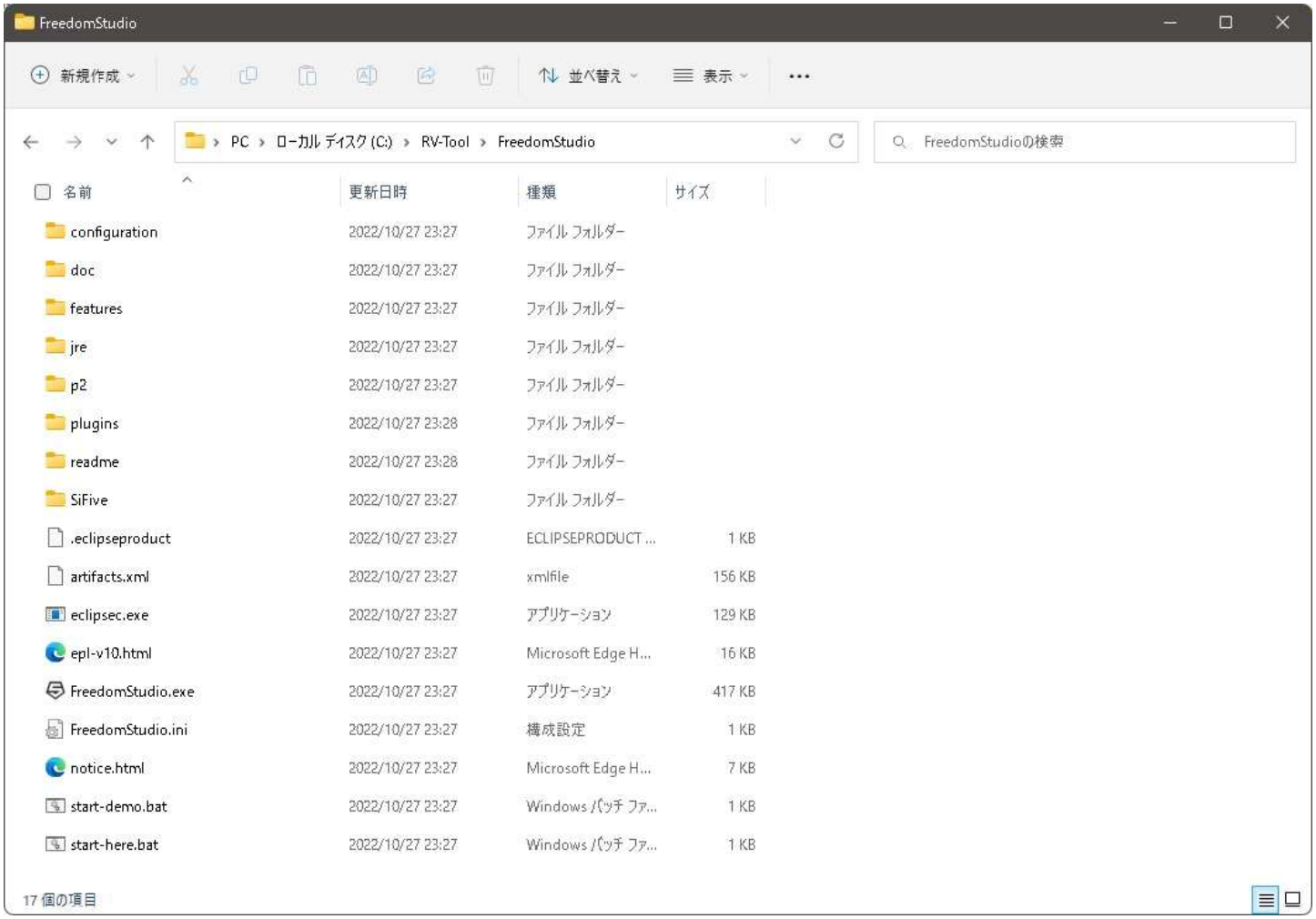

Windows 用の FreedomStudio をクリックする。




名前、姓名、E メールアドレス、所属の必須項目を入れて、「Submit」をクリックするとダウンロードが開始される。

## ダウンロードしたファイルの解凍

適当なフォルダにダウンロードしたファイルを解凍する。
（とりあえず、この説明では「C:\RV-Tool\FreedomStudio」へ展開。）




## パスを通す

解凍したフォルダの内、

C:\RV-Tool\FreedomStudio\SiFive\riscv64-unknown-elf-toolchain-10.2.0-2020.12.8\bin

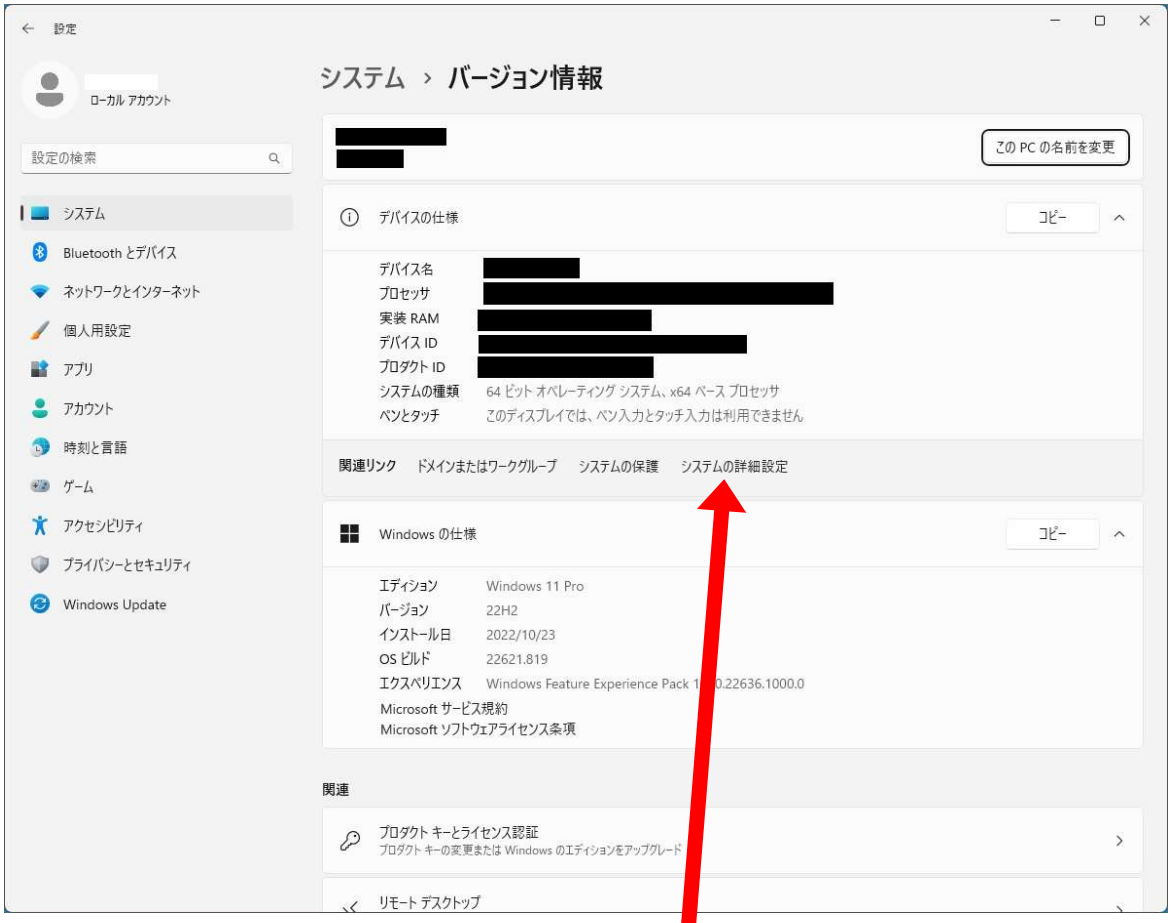
C:\RV-Tool\FreedomStudio\SiFive\msys64-1.0.0-2020.08.1\usr\bin

にパスを通す。「C:\RV-Tool\FreedomStudio」の部分は、個々の環境に合わせて変更する。

Windows11 でのパスの通し方（他の Windows については、「Windows PATH 追加」などの文字列で検索）。

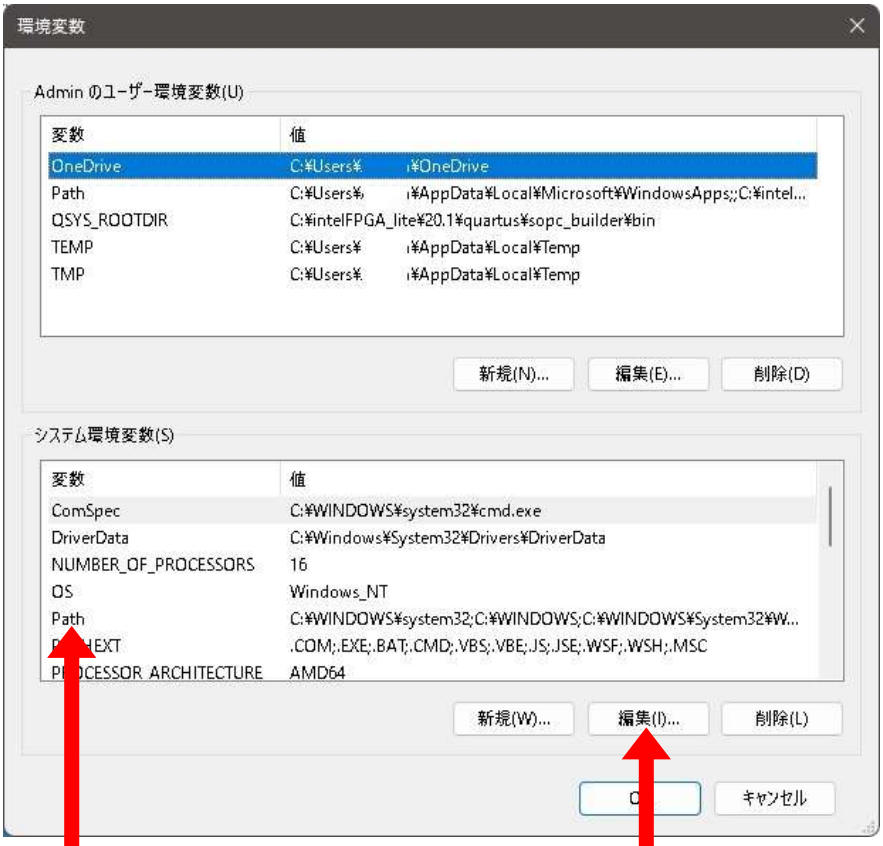

「システム」をクリック

Windos ボタンを右クリック




「システムの詳細設定」をクリック

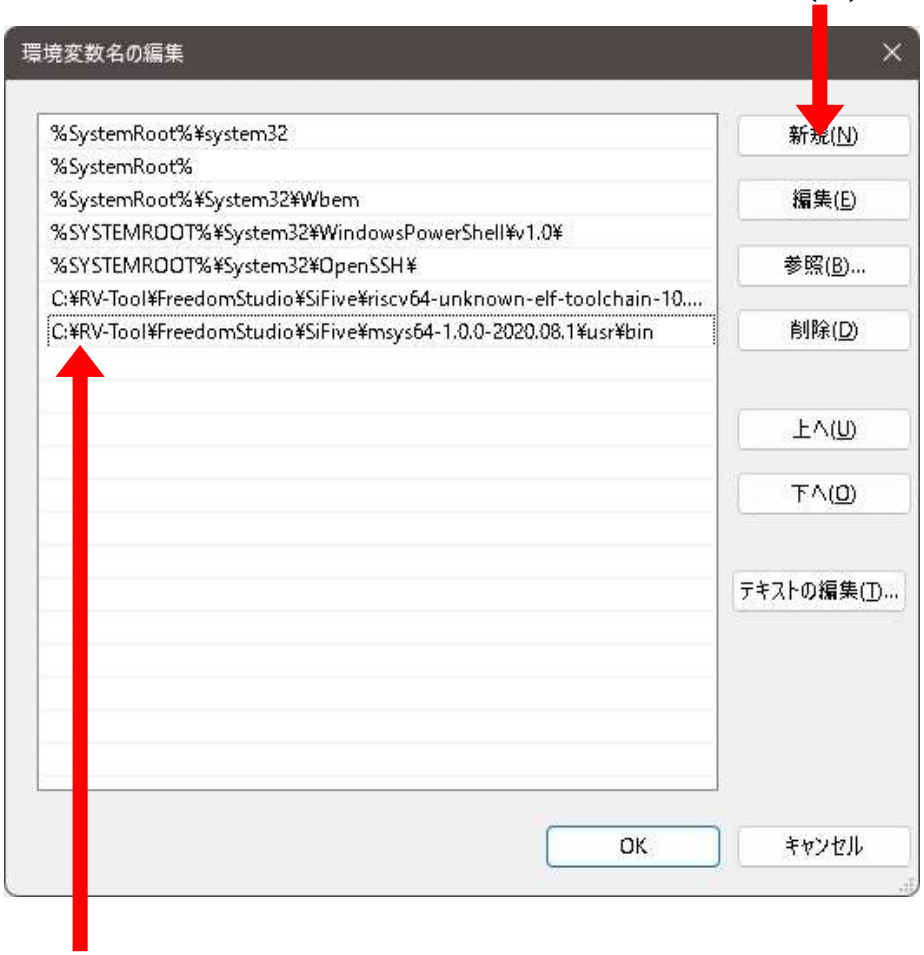

「環境変数 (N)… 」をクリック




「 Path 」を選択

「編集 (I)… 」をクリック

「新規 (N) 」をクリック




「C:\RV-Tool\FreedomStudio\SiFive\riscv64-unknown-elf-toolchain-10.2.0-2020.12.8\bin」
「C:\RV-Tool\FreedomStudio\SiFive\msys64-1.0.0-2020.08.1\usr\bin」を追加

以上で、簡単な RISC-V のクロスコンパイル環境は完成。
「riscv64-unknown-elf-toolchain-10.2.0-2020.12.8」フォルダ下には lib とか include とかサブフォルダがあるが、GPL のコピーレフトとか面倒なことを考えたくない場合は、なるべく使用しない。

# RISC-V のソフトのビルド方法と Verilog 用の hex ファイルの作り方

RV32I を例にして、test.c というファイルを、FreedomStuido のツールで、riscv のバイナリをビルドする方法。
スタートアップコードとして、start.S、リンカスクリプトとして、link.ld というファイルを使った場合で説明する。

コマンド プロンプトの起動

検索ウィンドウに「コマンド プロンプト」を入力


Windos ボタンをクリック

コマンドは以下の通りになる。

・コンパイル
　＞ riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -c -o test.o test.c
　＞ riscv64-unknown-elf-gcc -march=rv32i -mabi=ilp32 -c -o start.o start.S

サポートされるビット幅、拡張機能と mabi、march のパラメータ

| ビット幅 | 拡張機能 | mabi | march | 備考 |
|---|---|---|---|---|
| 32 ビット | E、M、A、C | ilp32e | rv32e[m][a][c] | |

| | | | | |
|---|---|---|---|---|
| 32 ビット | I、M、A、F、D、C | ilp32 | rv32i [m][a][f][d][c]<br>rv32g [c] | ソフト実数演算 |
| 32 ビット | I、M、A、F、D、C | ilp32d | rv32i [m][a][f] d [c]<br>rv32g [c] | ハード実数演算<br>(倍制度) |
| 32 ビット | I、M、A、F、D、C | ilp32f | rv32i [m][a] f [d][c]<br>rv32g [c] | ハード実数演算<br>(単制度) |
| 64 ビット | I、M、A、F、D、C | lp64 | rv64i [m][a][f][d][c]<br>rv64g [c] | ソフト実数演算 |
| 64 ビット | I、M、A、F、D、C | lp64d | rv64i [m][a][f] d [c]<br>rv64g [c] | ハード実数演算<br>(倍制度) |
| 64 ビット | I、M、A、F、D、C | lp64f | rv64i [m][a] f [d][c]<br>rv64g [c] | ハード実数演算<br>(単制度) |

ただし、SiFive のホームページにある Windows 用の Tool-chain では RV32E のリンクでエラーになる。
このバグは、Github の本家の Tool-chain では修正されている。

・リンク
＞ riscv64-unknown-elf-ld test.o start.o -Tlink.ld -static -m elf32lriscv -b elf32-littleriscv -o test.elf
64 ビットの場合、「elf64lriscv」、「elf64-littleriscv」とする。

・バイナリ化
＞ riscv64-unknown-elf-objcopy -O binary test.elf test.bin

・hex ファイル化
＞ hexdump -v -e '"%08X\n"' test.bin > test.hex
Verilog には$readmemh を使用して、メモリに組み込む。

```
reg [31:0] mem[MEM_DEPTH:0];
initial begin
    $readmemh("test.hex", mem);
end
```

Windows の CEERTUTIL コマンドでも HEX ダンプできるらしい。

・ELF ファイルのダンプ
＞ riscv64-unknown-elf-objdump -d test.elf > test.dump

竹田 良之