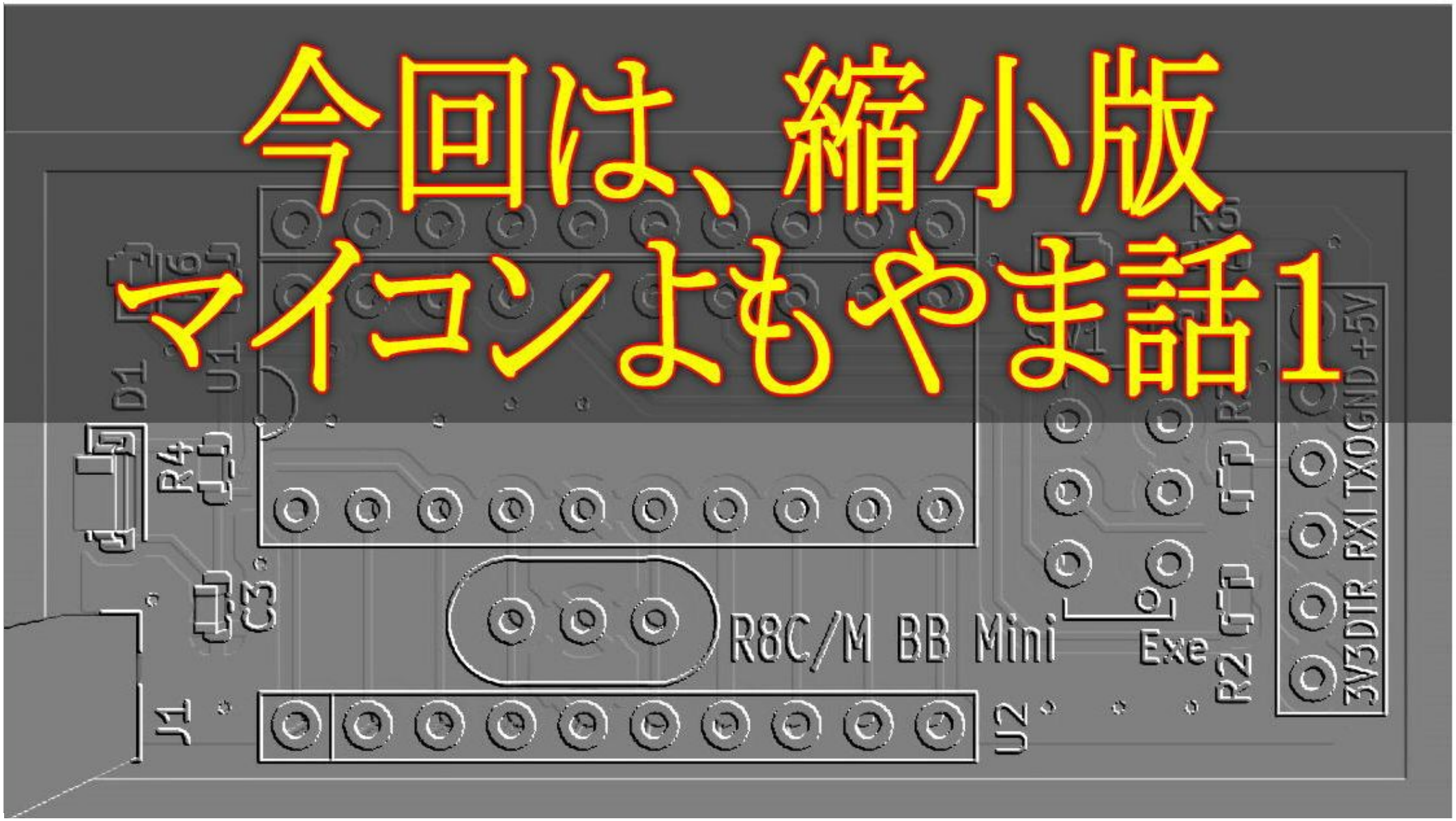

今回は、縮小版
マイコンよもやま話1
R8C/M BB Mini
Exe
3V3DTR RXI TXO GND +5V
D1
U1
R4
C3
J1
U2

R8C/Mシリーズの ZIPファイルについて

これらの資料は、ルネサスのホームページで公開されている評価版ソフト及び、ルネサスのホームページで公開されている資料です。 とくに評価版ソフトが、どこにあるのか分かりにくいので、私がダウンロードした物を 付けました。 その他　個人ユーザさんの資料も 一部含まれています。 今回の、R8Cマイコン開発に関わる圧縮ファイルは、2023年 8月末までの　期間限定と　させて頂きます。

- R8CM1_ハード資料.ZIP
  - R8C_開発環境等
    - R8C用C、C++コンパイラ_アセンブラ_リンカ_HEW
      - nc30v600r00_ev.exe ：　R8Cマイコン用統合開発環境インストーラ（ C,C++コンパイラ、アセンブラ、リンカを含む ）
      - FlashSta.exe　：　R8Cマイコン用書き込みソフト（ 対応OS：　WinXP ～ Win10 は 家の環境で確認してます。）
    - サンプルコードの資料（ 主にメーカーの PDFファイル ）
      - R8C用_C C++コンパイラユーザーズマニュアル.pdf
      - R8C用_アセンブラ_リンカ資料.pdf
      - その他、多数の資料
  - R8CM1_ハード資料　（ R8CMシリーズのデータシート PDFファイル ）

マイコンよもやま話について

何を　書こうかなと思っていたのですが…。
今回、R8Cマイコンで、プログラムを作成するに当たって　ルネサスの 統合開発環境HEWで扱える言語は C、C++、アセンブラです。

私は、ルネサスのマイコンで C++を使った事は ないです。 私は　Cと　アセンブラを組み合わせてプログラムを作ります。Cと　アセンブラを組み合わせて開発するのは、言語の適材適所的なところが あるからです。

アセンブラは、マイコンが持っている全ての命令が使えます。 コンパイラしかやってない方は、勘違いされるのですが、アセンブラは、昔の古い機械語的なコンパイラと　考えておられる方がいます。 アセンブラは、コンパイラではありません。

よってアセンブラを理解するために、アセンブラの勉強をするというのは、全くのナンセンスなのです。 どういうことかというと、特定の CPUのアセンブラを理解するのであれば、そのCPUのアーキテクチャ（設計思想）を理解する必要があるのです。 CPUのレジスタ構成、命令セットやアドレッシングモードの特徴を理解する事です。

言い方を変えると、アセンブラは　CPUが、変わると全く互換性が　ありません。 よって、アセンブラは、Cの様に　移植のため他の CPUに　ソースを持って行く等の移植生が、非常に悪いです。だから、C言語が出来たといえます。

でも C言語の舞台裏を話すと　移植性を損なわないように、どのCPUにも有るような、汎用的な命令しか使用しないため、そのCPUの持つ性能を　フルに引き出す事は出来ません。

例えば、C言語の 苦手な記述として、割り込み処理があります。 割り込み処理は、初期化時に該当する割り込みのベクトルテーブルに、割り込み処理の エントリーアドレスを記述する必要があります。

そして割り込みが発生した時、割り込み処理内で最初に 使用するレジスタを 全てスタックに積み上げる PUSH命令を実行します。

割り込み処理が終わったら、スタックに退避したレジスタ値を POP命令で 戻します。
そして割り込み処理用の リターン命令を使用します。 普通のサブルーチンリターン命令は使用出来ません。 暴走します。

アセンブラを経験した方であれば、何故暴走するか容易に理解できると思いますが、C、C++しか やった事が無い人にとっては、イメージが掴みにくいと思います。

それと、C、C++で 組み込み用途のプログラム開発にて、ルネサスの RXマイコンを使う場合は汎用レジスタを 16本持つ 32bit マイコンです。

C言語で割り込み処理を作成する場合は、割り込み処理スケルトン側で、どのレジスタを使用するのか分からないため、15本全てをスタックメモリ上に積み上げます。 1本少ないと思われるでしょうが、残り1本は、スタックを管理するスタックポインタです。

当然、割り込み処理から抜ける前に、スタック上に積み上げた15本分のレジスタの保存データを レジスタに POP命令で戻す必要があります。

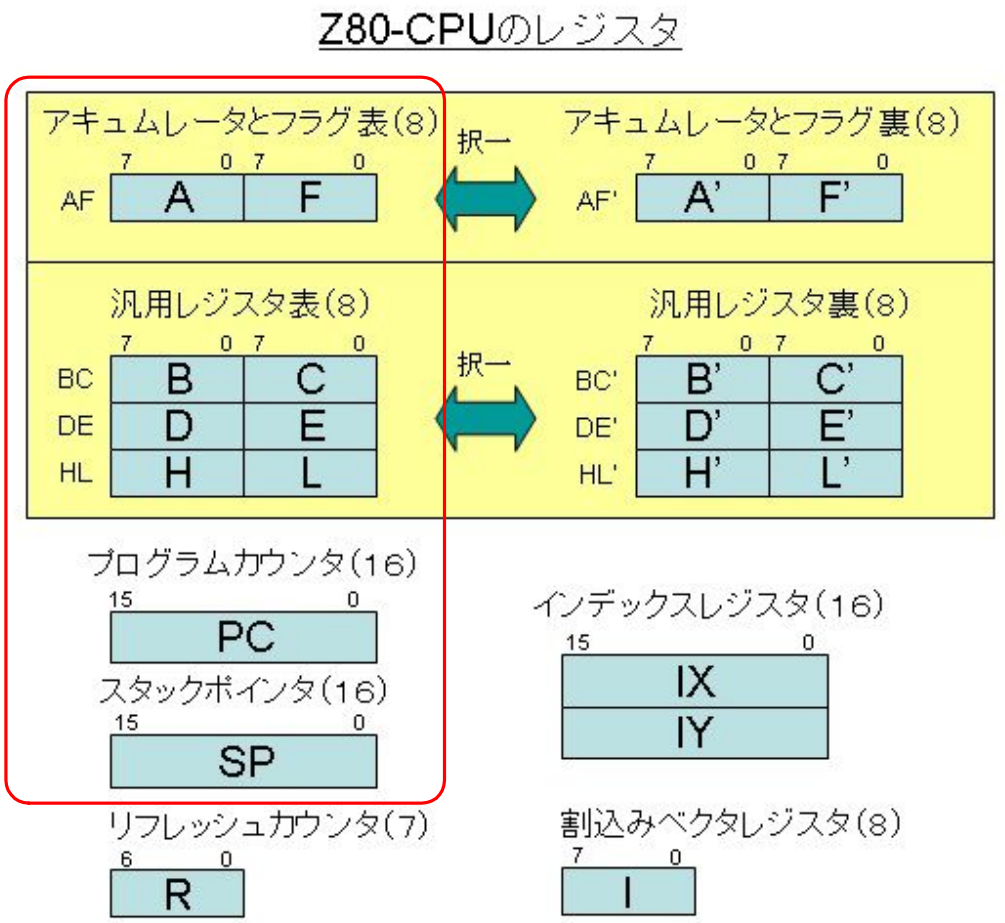
アセンブラで割り込み処理を記述すれば、自分で使用するレジスタは、把握できるでしょうから、使用するレジスタだけ、スタックに積み上げる事が、出来ます。 積み上げる数が少ない分、割り込みの応答速度が、早くなります。

ちょっと話が変わりますが、遥か昔 8bitのパソコンに 使用された Z80というCPUは、ご存じでしょうか。 若い人は 名前は聞いたこと有るけどどのような CPUなのかは知らない。 という方が多いのではないかと思います。

このCPUは、Intelの 世界初の コンピュータらしいマイコンとして登場した i8080、及び i8085とマシン語レベルで、上位互換性があるCPUです。

Intel は、i8080の前に 世界初のマイクロプロセッサ 4bitの i4004 と その 8bit版ともいえる i8008が、あります。 この2つは、DIP16ピンで、コンピュータというよりは、コントローラの一部で、性能もいまいちだったようです。

i8080で、劇的に コンピュータらしくなったようです。i8080は DIP 40ピンで、電源が +5V、+12V、-9Vの3電源が必要でした。 それを +5V単一電源で動くようにしたのが、i8085 です。

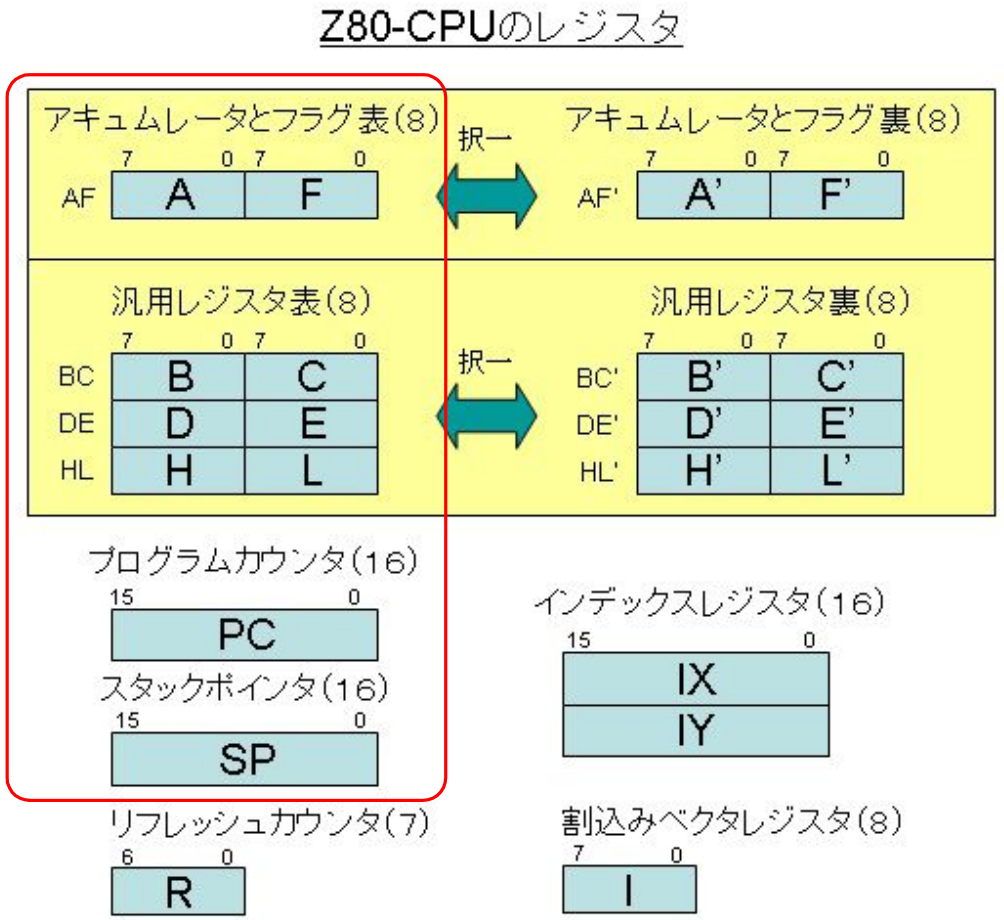


その後、ザイログという半導体メーカが、マシン語レベルで i8080上位互換の Z80を出しました。

前のページで、右のレジスター覧の話まで行き付けませんでしたが、Z80のレジスター覧です。 で、左側の赤い線で囲った部分が、i8080が、持っているレジスタです。 赤い枠線の外側にあるレジスタは、i8080は、持っていません。 よって、Z80のレジスタは、i8080の 倍に 増えています。

i8080のレジスタは、Fのフラグレジスタは、横に置いといて、 A、BC、DE、HL レジスタが ありますが、Aが アキュムレータ、BCレジスタペアが、カウンタ、DEレジスタペアは メモリのアドレス指定として使用できる。と書いてありますが機能が少ないようです。 HLレジスタペアがポインタとして優れているようです。

更に Z80は、インデックスレジスタ IX、IYももっており、ポインタを多数持っています。
そして、A'、B'C'、D'E'、H'L' の裏レジスタセットも持っています。 表レジスタセットと2つ同時



には、使えませんが、1命令で 表と裏のレジスタセットを瞬時に切り替える事が出来ます。
これは、ある事において非常に便利に使えます。

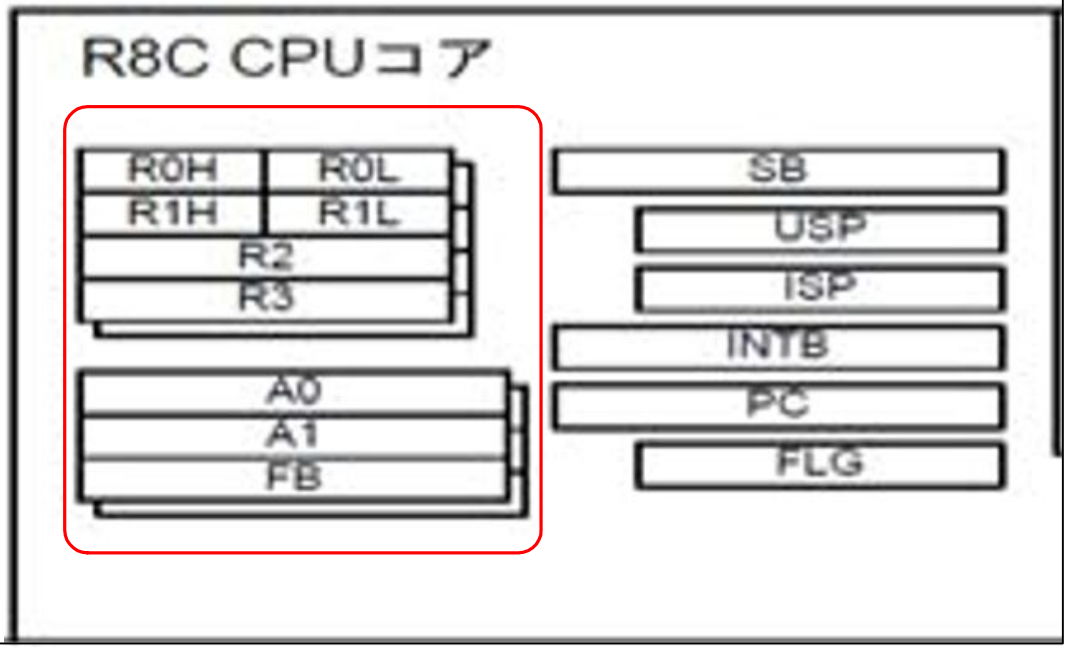
１命令で 表と裏のレジスタセットを瞬時に切り替える事が出来ます。 これは、ある事において非常に便利に使えます。 と書きましたが、ある事とは　何か分かりますか。？

３ページ前に、割り込み処理に入った直後に、CPUの レジスタを、全てスタックに積み上げて保存して、割り込み処理が終わったら、リターンする直前に　積み上げたレジスタの値を　レジスタに戻す処理を行ってリターンしています。

このレジスタセットの退避と復帰に、スタックにデータを積み上げるのではなくて、表と 裏のレジスタセットが　あるならば、１命令で表と　裏のレジスタを切り替えた方が、ずっと高速にレジスタを退避する事が出来ます。 但し、この方法で多重割り込みは、出来ません。 よって応答速度をあまり気にしない割り込みは、スタックを使用して、高速応答が必要な割り込み処理にのみ レジスタの表裏切り替えを使うと、効率のいい　制御系のシステムを　構築出来ます。

実は、Z80の表裏レジスタのような ハード構成は R8Cマイコンも持っているのです。

R0H、R0L、R1H、R1L、R2、R3、A0、A1、FB　レジスタは、裏レジスタが　あります。 R8Cも１命令で、レジスタセットを瞬時に切り替えられるので、高速な割り込み応答が可能になります。



ローエンドマイコンにとっては、このようなレジスタセットの 表裏切り替えが出来ると、マイコンの性能を フルに発揮させる事が出来ると思います。

まあ、その前に割り込み処理が、どのようなものなのかをしっかり理解しておく必要があります。 制御系のプログラムであれば、割り込みは避けて通れないと思います。

そして割り込み処理は、CPUの割り込みシーケンス等のハードに近い部分も理解しておく必要があります。 そしてハードを理解するという事は、アセンブラを理解する事にもつながってきます。 ローエンドマイコンで、最高のパフォーマンスを実現するには、アセンブラも必須になると思います。

但し、アセンブラは、万能ではありません。 正直 作りにくい用途もあります。

例えば、複雑な演算処理は、C言語で行うべきです。R8Cは 整数演算は結構速いと思います。

浮動小数点は、私は R8Cで 使った事がありません。ソフトで演算を行うので、整数演算に比べ数十倍時間が かかります。 そして演算ライブラリもリンクするので、メモリも圧迫します。

特にリアルタイム処理内で、浮動小数点演算を行うのであれば、FPU(浮動小数点演算ユニット)を 持ってないと 処理が間に合わない恐れがあります。 ルネサスの場合は、RX600シリーズに、FPUが 実装されてます。

あと、アセンブラで適さないと思われるのは複雑な判定処理です。 これは、C言語で if 文、switch ～ case 文で ループ処理は、for 文、while 文、そして continue、break などで表す方が ロジックを 簡潔に表現できるでしょう。