TOPPERS 活用アイデア・アプリケーション開発コンテスト

部門　：　(活用アイデア部門)
　　アプリケーション開発部門
　　（フリークラス，がじぇるね IoT クラス，R2CA クラス）

作品のタイトル　：　TOPPERS/ASP3 カーネルの AArch64 対応

作成者　：　加藤 丈治

共同作業者　：

対象者　：　64 ビット ARM の利用を検討している組込み開発者

使用する開発成果物　：　TOPPERS/ASP3 カーネル

目的・狙い

以下の目的から 64 ビット ARM 環境へ TOPPERS/ASP3 カーネルの移植を行った
1）64 ビット ARM 市場に TOPPERS の適用範囲の拡大を図ること
2）次世代主力 RTOS である TOPPES/ASP3 を利用可能にすることで、64 ビット ARM 環境が目指すエコシステムの発展に貢献すること
3）OSS のシミュレータ環境向けに実装を行い、成果物（ソースコード）および動作デモの公開を通して 64bit 組込み環境への参入障壁の低減を図ること

アイデア/アプリケーションの概要

TOPPERS/ASP3 に 64 ビット ARM 用ターゲット依存部を追加し, OSS のシステムシミュレータである QEmu 環境での動作を確認した。主要な移植項目は以下の通り:
1）64 ビット ARM 向けのコンテキスト操作部の実装
2）64 ビット ARM 向けの時間管理機構の実装
3）64 ビット ARM 向けの割り込み/例外処理機構の実装

はじめに

近年発表されたARMv8アーキテクチャでは、64ビット実行モード（以下AArch64）が追加されており、組込み分野でも64ビット実行環境が普及する兆しを見せつつある。

その一方で，次世代TOPPERSカーネルであるTOPPERS/ASP3カーネルは,64ビットARM環境のサポートが行われていないため，AArch64で改善された命令セットを利用した高性能な組込み機器の実装が行いにくいという課題がある。

本稿では本課題を解決するために，TOPPES/ASP3カーネルをAArch64で実行可能とした。

TOPPERS/ASP3カーネルをAArch64へ移植することで，以下のような効果を得ることが期待できる。

- 64ビット環境へのTOPPERS普及の足がかりとすることでTOPPERSカーネルの適用範囲の拡大を図ること
- 国内外で大きなシェアを持つTOPPERSをAArch64で利用可能とすることで64ビットARM環境が目指すエコシステムの拡大を図ること

64ビット版ARMアーキテクチャについて

ARM社により新たなARMアーキテクチャとして，ARMv8アーキテクチャが公開されている。ARMv8では，従来の32bitアプリケーション向けの実行モードであるAArch32に加え，64ビットの実行モードであるAArch64が追加されている。

AArch64の特徴は以下の通り：

- アドレス長が48ビットに拡大
- 32bit固定長の命令（レジスタデコードの改善）
- 汎用レジスタ構成を従来の32bit長レジスタ16個から64bit長レジスタ31個に強化
- プログラムカウンタ（PC）とスタックポインタ(SP)の利用方法の変更
- Advanced SIMDやVFP機能における128ビットレジスタの拡張（16個から32個），倍精度演算サポートやIEEE754への準拠

上記のうちアドレス長の拡張は，スマートフォンやタブレットなどの比較的大規模な組込み機器への応用を意図した機能改善項目である。

その他の項目は，既存ARM命令セットに対する以下のような機能・性能改善を意図した項目であるため、中・小規模組込み機器でも大きな適用効果が期待できる。

- パイプラインの活用効率など処理性能面での改善
- コンパイラでのより高度な最適化を行うための命令体系・レジスタ体系の改善

- 浮動小数演算の標準仕様準拠による他アーキとの演算結果の共通化

今後、ARMv8 アーキテクチャの普及に伴い,中小規模組込み機器を含めた組込み分野全般に AArch64 を用いた応用事例が増えていくと考えられる。

AArch64 版 TOPPERS/ASP3 カーネルの開発環境・ターゲット環境

前節で説明した背景から，既存の組込みソフトウエア資産を AArch64 環境で活用可能とするために，本稿では TOPPERS/ASP3 カーネルを AArch64 ターゲットに移植する。

開発環境・ターゲット環境を表 1 に示す。

表 1 開発環境・ターゲット環境

開発ホスト環境	CentOS7.1
開発ツールチェイン	2016 年 2 月版 x86-64 用 AArch64 ターゲット向けクロスコンパイラ(gcc-5.3)[1] (gcc-linaro-5.3-2016.02-x86_64_aarch64-linux-gnu)
ターゲット環境	QEMU emulator version 2.6.0[2]
ターゲット略称	aarch64_gcc

本稿では，AArch64 版の TOPPERS をより気軽に試してもらえる環境を構築することで，AArch64 や TOPPERS 環境への新規参入障壁を軽減することを主眼にホスト環境やターゲット環境を選定した。選定の背景は以下の通り：

- 開発ホスト環境・・・インターネット経由で容易に入手可能で，かつ，商用利用者が多く，運用実績が十分に確立されている Red Hat 系の Linux OS を採用
- 開発ツールチェイン・・・安価に導入可能な OSS のコンパイラで，かつ,導入が容易なクロスコンパイラとして linaro 版のクロスコンパイラを採用
- ターゲット環境・・・比較的高価な実機を導入することなく AArch64 環境に触れられる環境として OSS のシステムシミュレータである QEmu を採用

---

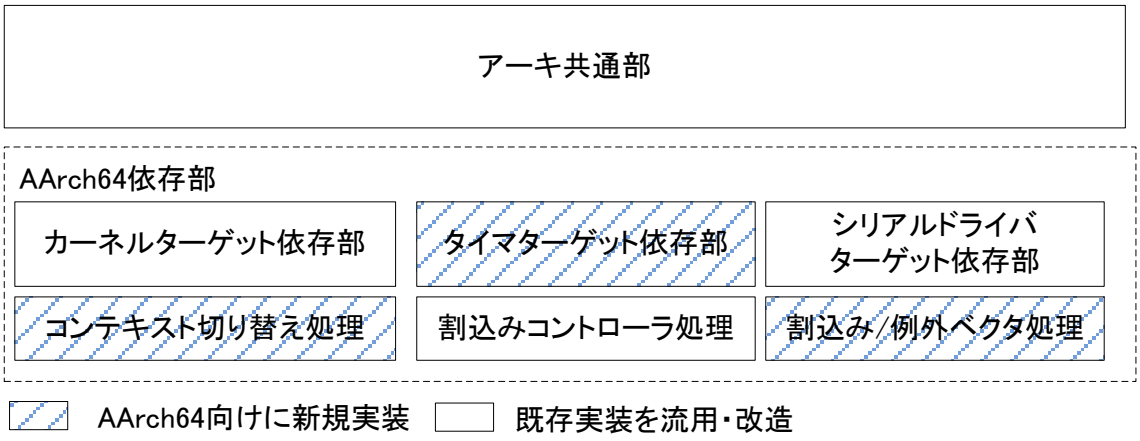
1 https://releases.linaro.org/components/toolchain/binaries/latest-5/aarch64-linux-gnu/gcc-linaro-5.3-2016.02-x86_64_aarch64-linux-gnu.tar.xz より入手可能

2 http://wiki.qemu-project.org/download/qemu-2.6.0.tar.bz2 より入手可能

AArch64 対応版 TOPPERS/ASP3 カーネルの構成

AArch64 対応版 TOPPERS/ASP3 カーネルの構成を図 1 に示す。

図 1 AArch64 版 TOPPERS/ASP3 カーネルの構成



- **アーキ共通部**・・・特定のアーキテクチャに依存しないカーネル本体部。
- **AArch64 依存部**・・・ARMv8 の AArch64 モードに依存したカーネル実装部。主に以下のサブモジュールから構成される。
  - **カーネルターゲット依存部(target/aarch64_gcc/start.S, target_kernel_impl.c)**
    ターゲット依存の初期化・終了処理やカーネル共通部から呼ばれるディスパッチャの動作開始処理を実装しているモジュール。既存の ct11mpcore の実装を流用・改造して実装した。
  - **タイマターゲット依存部(target/aarch64_gcc /target_timer.c)**
    AArch64 上で動作する OS が標準的に使用しているチップ内仮想タイマ機能を使用してティックレスタイマ処理を実装するモジュール。AArch64 向けに新規に実装した。
  - **シリアルドライバターゲット依存部(PL011 用のドライバを流用)**
    QEmu AArch64 システムシミュレータで一般に使用される virt ターゲット向けの UART ドライバ。既存の ARM PrimCell UART (PL011) 用ドライバを流用した。
  - **コンテキスト切り替え処理(target/aarch64_gcc/swctx.S)**

AArch64 で拡張された汎用レジスタ(X0-X31)の待避復元処理を実装するモジュール。AArch64 向けに新規に実装した。

- **割込みコントローラ処理(target/aarch64_gcc/ gic_support.S)**
  AArch64 の GIC（Generic Interrupt Controller）を用いた割込み属性の設定・マスク制御・割込み応答処理などを実装するモジュール。既存の GICv2 用の割込み制御処理を流用・改造して実装した。

- **割込み/例外ベクタ処理(target/aarch64_gcc/vector.S, traps.c)**
  AArch64 の割込み/例外ベクタの定義と割込みの出入り口となる処理を実装するモジュール。AArch64 向けに新規に実装した。

AArch64 では，レジスタの拡張や命令セットの見直しにより，既存の 32bit 版のコードをそのまま流用することはできないものの，UART や割込みコントローラなどの周辺チップについては，既存資産で確立された処理方式を活用可能とするように設計されている。

例えば，UART シリアルや割込みコントローラのレジスタ仕様には，32bit ARM の環境と同様の仕様が採用されている。本仕様は、32bit 環境の資産を 64 ビット環境に容易に流用することを視野に入れた設計によるものである。

本稿での AArch64 への移植作業では，32bit 版 ARM 向けの処理ロジックや既存ドライバを流用することで，大きな設計変更を伴うことなくカーネルを AArch64 に対応させることができた。このことは，AArch64 の設計方針の妥当性と TOPPERS/ASP3 の持つ移植性の高さの証左となったと考えている。

次節以降で，AArch64 向けに新規に実装したモジュールに関する設計について述べる。

AArch64 のタイマターゲット依存部

本稿での AArch64 への移植作業では， AArch64 の標準タイマである Generic Timer 機構を利用してタイマ処理のターゲット依存部を実現した。

Generic Timer 機構では，仮想化やセキュアモニタの実装を想定し，「仮想タイマ」，「物理タイマ」，「物理セキュアタイマ」の 3 種類のタイマを搭載可能となっている。オペレーティングシステムの時間管理には，「仮想タイマ」を使用するのが一般的である[3]。

Generic Timer 機構の仮想タイマ関係のレジスタの概要を表 2 に示す。

[3] OS v https://github.com/cloudius-systems/osv/wiki/AArch64 や Linux カーネル，FreeBSD カーネルでも仮想タイマを用いた時間管理を実装している。

表 2　AArch64 Generic Timer 機構（仮想タイマ）関連レジスタ

レジスタ名	使用用途
CNTV_CTL_EL0	仮想タイマのカウント開始・停止・割込み発生有無を制御する制御レジスタ
CNTVCT_EL0	仮想タイマのカウント値を保持するフリーカウンタ
CNTV_TVAL_EL0	次のタイマ割込みをあげるまでのカウント値（現在値からの相対値）を設定するレジスタ
CNTFRQ_EL0	カウンタの増加周期（単位:Hz）を取得する読み取り専用レジスタ

TOPPERS/ASP3 カーネルのティックレスタイマ処理内の高精度タイマイベント設定 IF(target_hrt_set_event)では，次のタイマ割込みが上がるまでの現在時間からの相対時間を引数に指定することでタイマのリプログラミングを行う。

本稿で実施した AArch64 への移植では，TOPPERS/ASP3 の仕様に従ったティックレスタイマ機構を実装している。実装したタイマ割込処理機構の処理内容を表 3 に示す。

表 3 AArch64 依存部でのタイマ処理

ターゲット依存部のタイマ処理関数	処理の概要
target_hrt_initialize	割込み発生カウンタ（CNTV_TVAL_EL0）を最大値に設定して初期化し，タイマを開始する
target_hrt_terminate	タイマを停止する
target_hrt_get_current	タイマカウンタ(CNTVCT_EL0)を詠み出す
target_hrt_raise_event	GIC を使用してタイマ割込み(27 番)の割込みを発生させる
target_hrt_set_event	タイマカウンタの周期(CNTFRQ_EL0)と現在のカウンタ値(CNTVCT_EL0)を元に，次に割込みを上げるまでの相対カウント値を算出して，割込み発生カウンタ（CNTV_TVAL_EL0）を設定する
target_hrt_handler	target_hrt_int_clear を呼び出して,タイマ割込みをクリアしアーキ非依存のタイマ処理(signal_time)を呼び出す

Generic Timer機構は，表 2に示した通り，タイマ割込み契機を現在のカウンタ値からの相対時間を割込み発生時間として引き渡す仕様であるため，Generic Timer 機構の特性を生かすことで，ティックレスタイマ機構を比較的容易に実装できたと考える。

AArch64 でのタスクコンテキスト切り替え処理

AArch64 では，図 2 に示すレジスタが利用可能である[4]。

図 2 AArch64 モードでのレジスタ

一般に RTOS のタスクコンテキスト切り替え処理では，プロセッサの ABI(Application Binary Interface)で規定された関数呼び出し規約に従って，コンテキストの退避・復元を行

[4] 「OS のタスク切り替え処理と aarch64(ARM64)の関数呼び出し規約」（ http://qiita.com/tkato/items/82f095a8f8c55be07cb5 ） より図を引用。

う。AArch64 の場合, Procedure Call Standard for the ARM 64-bit Architecture[5]により, 表 4 に示すように関数呼び出し時のレジスタ使用規約が規定されている。

表 4 AArch64 での関数呼び出し規約

レジスタ	保存契機	用途
SP	保存不要	スタックポインタ
X30(LR)	呼び出された側で保存	リンクレジスタ。関数復帰アドレスを格納する
X29(FP)	呼び出された側で保存	フレームポインタ。関数内のローカル変数などの格納領域アドレスを格納する
X19-X28	呼び出された側で保存	汎用レジスタ
X18	呼び出された側で保存	プラットフォームレジスタ
X16-X17	呼び出された側で保存	ライブラリの動的リンクに使用するレジスタ
X9-X15	呼び出した側で保存	テンポラリレジスタ
X8	呼び出された側で保存	間接リゾルトレジスタ
X0-X7	呼び出された側で保存	関数の引数や返値（x0)の格納に使用する

本稿で実施した AArch64 への移植作業では, 上記仕様に従って, 呼び出された側の関数で保存するべきレジスタを退避・復元することでタスク切り替え処理を実現している(表 5 参照)。

表 5　AArch64 でのコンテキスト切り替え処理の主要関数

コンテキスト切り替え処理関数	処理の概要
dispatcher	次に実行するタスクのスタックからコンテキストを復元する。アイドル時は割込み待ちで CPU を停止させる(wfi 命令を実行する)
dispatch	呼び出し側で保存するべきレジスタをスタックに保存して, ディスパッチャを呼び出す

[5] http://infocenter.arm.com/help/topic/com.arm.doc.ihi0055b/IHI0055B_aapcs64.pdf

exit_and_dispatch	カーネル終了処理の呼出し

浮動小数点レジスタについては，浮動小数点演算処理時に発生する CPU 例外を契機としてレジスタの退避・復元を行うことで，浮動小数点レジスタの退避/復元に伴う処理コストを軽減する設計とすることが一般的である[6]。

また，組込機器の場合，特定のタスクでのみ浮動小数点演算を行うような構成の場合には，浮動小数点レジスタの退避復元は不要となるため、浮動小数点レジスタの退避/復元の要否はアプリケーションのタスク構成に依存する。

以上の背景から，浮動小数点レジスタの退避・復元処理は，アプリケーションに任せるべきと判断し，本稿での実装の対象外とした。

AArch64 での割込み/例外ベクタ処理

AArch64 では，以下の 4 種類の例外/割込みを受け付けることが可能となっている。

1) 同期割込み・・・ソフトウエアトラップや MMU 例外などの同期的に発生する例外
2) IRQ・・・外部割り込み
3) FIQ・・・高速外部割り込み
4) エラー・・・異常例外

AArch64 の割込みベクタテーブルは，上記の例外/割込みを例外発生前の実行レベル/スタックポインタ/実行モード(AArch32)毎に受け付け可能な 16 個のエントリから構成される[7]。

本稿での AArch64 移植作業では，AArch64 のカーネルモードで動作するアプリケーションを実行することを想定し，カーネルモードでの例外・割込みのみを処理する方針で割込み・例外処理を実装した。

本稿での移植作業では，IRQ および FIQ 割込みに対する割込みハンドラの定義・実行，同期割込み，エラー割込みに対する例外ハンドラの定義・実行をサポートした。

TOPPERS のポーティングガイドに従い割込み・例外処理時の非タスクコンテキストへの切り替えや割込みマスク操作を実装できており，AArch64 移植に際して特別な考慮は不要であった。

---

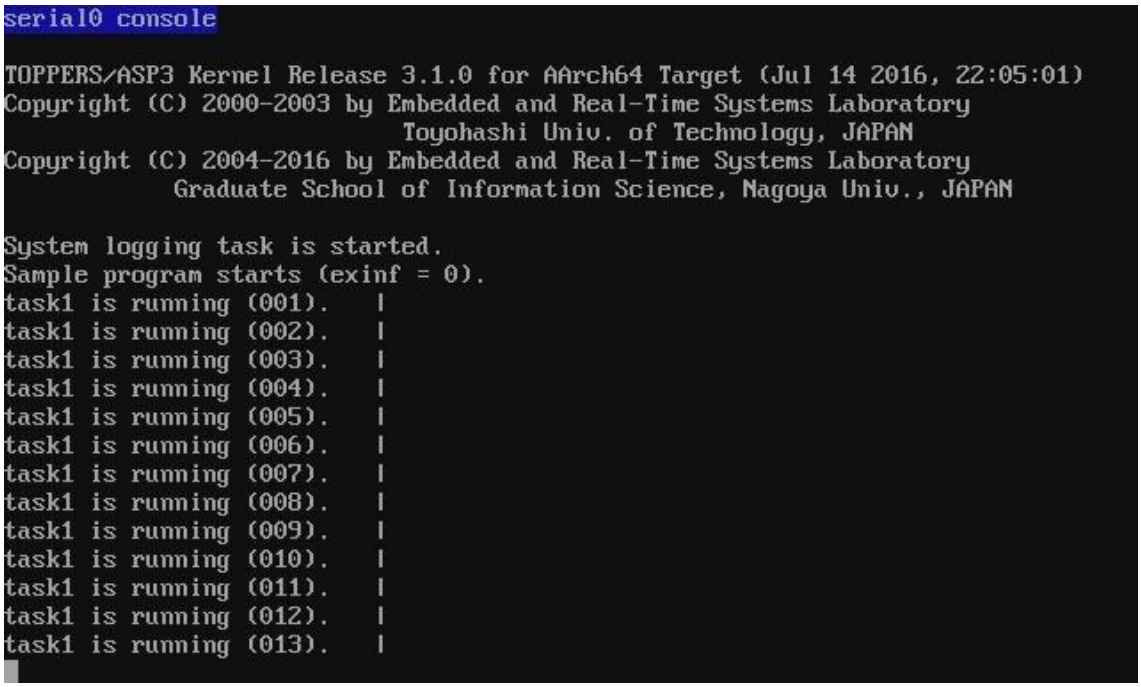
[6] インテル® 64 および IA-32 アーキテクチャー・ソフトウェア・デベロッパーズ・マニュアル内の「D.3.6.2 Tracking x87 FPU Ownership」を参照
（ http://www.intel.co.jp/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf ）

[7] ARM® Architecture Reference Manual 内の「D1.10.2 Exception vectors」を参照
https://silver.arm.com/download/ARM_and_AMBA_Architecture/AR150-DA-70000-r0p0-00eac0/DDI0487A_j_armv8_arm.pdf

動作例/ソースコード

本稿で実装した AArch64 対応版カーネルの動作画面例を図 3 に示す。また実際の動作の様子を YouTube(https://www.youtube.com/watch?v=-VH_JOxVme8) で公開済みである。

図 3 AArch64 版 TOPPERS/ASP3 カーネルの動作画面



また，今回の作業で実装したソースコードも GitHub(https://github.com/takeharukato/asp3-aarch64)にて公開済みである。

まとめ/将来課題

64 ビット ARM 環境を組込み分野でより広く活用するための環境整備の一環として，TOPPERS/ASP3 カーネルを AArch64 に移植し，一般的なシミュレータ環境での動作確認を通して実現可能性を示した。 これにより，将来的に TOPPERS と AArch64 の双方の利用が促進されることで本稿での目的が達成されると考える。

現時点での実装では，既存実装部のように Core/Chip/Target の 3 者が明に分離された構成になっていないなどの課題が残っている。AArch64 実機への展開も含め今後の課題としたい。

謝辞

本開発に際し, ruby 版コンフィグレータ利用時のトラブル対応などで名古屋大学 高田広章先生はじめ toppers-users メーリングリストの方々にお世話になりました。この場をお借りして感謝の意を表します。

また, TOPPERS 初心者の私に, TOPPERS の利用法から設計思想・実装に対する適切な助言と励ましを下さいました石岡 之也氏に感謝の意を表します。

付録:QEmu 上での AArch64 版 TOPPERS の動作手順

x86-64 上の CentOS7.1 環境に 3 ページの表 1 に示したコンパイラ(aarch64-linux-gnu-gcc)および QEmu(qemu-system-aarch64)を導入する。

環境変数 PATH 上に両者のコマンドへのパスが設定されている状態で, 以下のコマンドを実行することで、QEmu 上で AArch64 版 TOPPERS を動作させることが可能である。

1) Git Hub からソースを取得する
   git clone https://github.com/takeharukato/asp3-aarch64.git
2) カーネルソースツリー中に OBJ ディレクトリを作成後、OBJ ディレクトリに移動する
   cd asp3-aarch64
   mkdir OBJ
   cd OBJ
3) aarch64_gcc 用の Makefile を生成する
   ../configure.rb -T aarch64_gcc
4) カーネルをコンパイルする
   make
5) Qemu 上でサンプルプログラムを動作させる
   qemu-system-aarch64 -M virt -cpu cortex-a53 -kernel asp

—以上—