第5章

ARM7コア搭載CPUのためのGCC＆GDB環境

ARM対応クロス開発環境の構築とその使い方

本稿も第3章と同様に，CPU固有の部分に焦点を絞って解説する．ここではARMアーキテクチャについて解説した後，ARM用のスタートアップ・ルーチンやリンカ・スクリプトの書き方，ADuC7026用のGDBスタブの移植事例について解説する．（編集部）

山際 伸一

本稿では第3章と同じ構成で，ターゲットCPUとしてARMを取り上げ，GNUツールを使った組み込みプログラム開発手法と，GDBによるデバッグ環境の構築方法について説明していきます．

1. ARMを理解する

● ARMアーキテクチャ

ARMアーキテクチャにはいくつか種類が存在しますが，ここではもっとも基本的なコアであるARM7 TDMIについて取り上げます．

1) プロセッサ概要

ARMはリトル・エンディアンのパイプライン・プロセッサですが，遅延スロットを持ちません．CPUの実行モードとしてARMステートとThumbステートと呼ばれる二つのモードがあり，それぞれ使用される命令セットが異なります．ARMステートでは32ビット長の命令セットが，Thumbステートでは16ビット長の命令セットが用いられます．BXまたはBLX命令を用いることでThumbステートとARMステートのそれぞれに移行することが可能です．これらの命令にはCPSRレジスタのTビットをそのオペランドのアドレス即値，またはレジスタの内容の最下位ビットに設定することで，異なる動作モードへの移行が可能になっています．

2) レジスタ

ARMのレジスタを**図1**に示します．汎用レジスタについては，それぞれの使用方法が規定されています．GNUツールではこれらのレジスタの定義に加えて，C言語関数の引き数と返り値のための仕様を追加しています．

また，ARMでは，いくつかのレジスタが例外ごとに重複して用意されており，それぞれの例外が発生した場合，専用のレジスタが用いられます．

3) 命令セット

表1にARMの命令セット一覧を示します．ARMには複数のレジスタの書き込み，そして読み出しが可能なSTM，

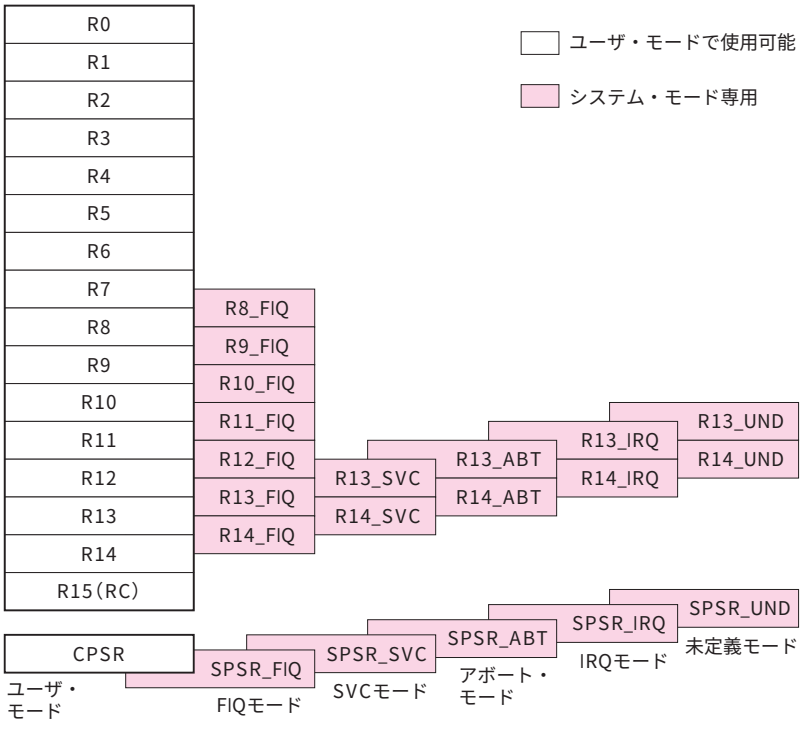

図1　ARMレジスター覧

KEYWORD ―ARM，ARM7TDMI，ARMモード，Thumbモード，ADuC7026，UART，ウォッチドッグ・タイマ，スタートアップ・ルートン，startup.s，リンカ・スクリプト，例外ハンドラ

表1　ARM命令セット

ニモニック	構　文	意　味	動　作
AND	AND{cond}{S} Rd, Rn, Op2	論理積	R*d*=R*n* AND Op2
EOR	EOR{cond}{S} Rd, Rn, Op2	排他的論理和	R*d*=R*n* EOR Op2
ORR	ORR{cond}{S} Rd, Rn, Op2	論理和	R*d*=R*n* OR Op2
BIC	BIC{cond}{S} Rd, Rn, Op2	ビット・クリア	R*d*=R*n* AND not(Op2)
ADD	ADD{cond}{S} Rd, Rn, Op2	加算	R*d*=R*n*+Op2
ADC	ADC{cond}{S} Rd, Rn, Op2	キャリ付き加算	R*d*=R*n*+Op2+C
SUB	SUB{cond}{S} Rd, Rn, Op2	減算	R*d*=R*n*−Op2
RSB	RSB{cond}{S} Rd, Rn, Op2	逆減算	R*d*=Op2−R*n*
SBC	SBC{cond}{S} Rd, Rn, Op2	キャリ付き減算	R*d*=R*n*−Op2−not(C)
RSC	RSC{cond}{S} Rd, Rn, Op2	キャリ付き逆減算	R*d*=Op2−R*n*−not(C)
TST	TST{cond}　Rn, Op2	テスト	Flag <= R*n* AND Op2
TEQ	TEQ{cond}　Rn, Op2	等価テスト	Flag <= R*n* EOR Op2
CMP	CMP{cond}　Rn, Op2	比較	Flag <= R*n*−Op2
CMN	CMN{cond}　Rd, Op2	否定比較	Flag <= R*n*+Op2
MOV	MOV{cond}{S} Rd, Op2	移動	R*d*=Op2

R*d*：デスティネーション・レジスタ　　“ <= ”はフラグの更新を意味する
R*n*：第1ソース・レジスタ　　“ C ”は条件フラグのキャリ・フラグ

（**a**）データ処理命令

ニモニック	構　文	意　味	動　作
MUL	MUL{cond}{S}　Rd,　Rm,　Rs	32ビット積算	R*d*=(R*m*＊R*s*)[31 : 0]
MLA	MLA{cond}{S}　Rd,　Rm,　Rs, Rn	32ビット積和算	R*d*=(R*m*＊R*s*+R*n*)[31 : 0]
UMULL	UMULL{cond}{S}RdLo, RdHi, Rm, Rs	符号なし64ビット積算	R*d*/Hi : R*d*/Lo=R*m*＊R*s*
UMLAL	UMLAL{cond}{S}RdLo, RdHi, Rm, Rs	符号なし64ビット積和算	R*d*/Hi : R*d*/Lo=R*d*/Hi : R*d*/Lo+R*m*＊R*s*
SMULL	SMULL{cond}{S}RdLo, RdHi, Rm, Rs	符号あり64ビット積算	R*d*/Hi : R*d*/Lo=R*m*＊R*s*

R*d*：デスティネーション・レジスタ　　R*m*：被乗数レジスタ(32ビット)　R*s*：乗数レジスタ(32ビット)
R*d*/Hi：デスティネーション・レジスタ(上位32ビット)　　R*n*：積に加算する値
R*d*/Lo：デスティネーション・レジスタ(下位32ビット)

（**b**）乗算命令

ニモニック	構　文	意　味
LDR	LDR{cond}　Rd, addr_mode	ワード・ロード
LDRB	LDR{cond}B　Rd, addr_mode	バイト・ロード
LDRT	LDR{cond}T　Rd, addr_mode	ユーザ・モード，特権ワード・ロード
LDRBT	LDR{cond}BT　Rd, addr_mode	ユーザ・モード，特権バイト・ロード
STR	STR{cond}　Rd, addr_mode	ワード・ストア
STRB	STR{cond}B　Rd, addr_mode	バイト・ストア
STRT	STR{cond}T　Rd, addr_mode	ユーザ・モード，特権ワード・ストア

R*d*：デスティネーション・レジスタ　　addr_mode：アドレッシング・モード

（**c**）ワードおよび符号なしバイト転送命令

ニモニック	構　文	意　味
LDRH	LDR{cond}H　Rd, addr_mode	ハーフワード・ロード
LDRSH	LDR{cond}SH Rd, addr_mode	符号付きハーフワード・ロード
LDRSB	LDR{cond}SB Rd, addr_mode	符号付きバイト・ロード

R*d*：デスティネーション・レジスタ　　addr_mode：アドレッシング・モード

（**d**）ハーフワードおよび符号付きバイト転送命令