第3章

2007年5月号付属CPU基板を活用

V850対応クロス開発環境の構築とその使い方

第2章のGDBの解説では、リモート・シリアル・プロトコルなどの非CPU依存部について解説した．実際にシリアル・ポートを使ってGDBと通信するには，CPU固有のUART機能を使って通信する必要がある．また実際にブレークがかかった後の例外処理の内容は，それぞれのCPUによって異なる．ここではV850ESのアーキテクチャについて解説した後，V850ES/JG2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，V850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，実際のCPUや評価ボードを想定し，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ます．まずは本誌2007年5月号付属基板に搭載されているV850ES/JG2を取り上げます．

1. V850ESシリーズを理解する

ここではV850ESのCPUアーキテクチャについて，特にプログラマの立場からその概要を説明します．

● V850ESアーキテクチャ

1) プロセッサ概要

V850ES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．従ってアセンブリ言語で書かれたプログラムの各命令は，分岐命令を含む場合でも記述した順番に命令を実行します．基本命令長は16ビットですが，命令の種類によって（例えば，即値分岐命令jrなど）は32ビット長のものもあります．

2) レジスタ

V850ES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．

3) 命令セット

表1に命令セットの一覧を示します．プロセッサ・コアは乗算器をハードウェアとして実装しているため，乗算命令はハードウェアで実行されます．

ちなみに，**図1**に示した，専用レジスタが用意されているcalltの命令とは，サブルーチン・コールを16ビット長で実行する命令です．これを利用すると，通常の32ビット長命令を使用するサブルーチン・コールよりプログラム・サイズを小さくすることができます．

4) 割り込み・例外処理

割り込み・例外処理のための例外ベクタはV850ESでは固定されており，必ず0x0番地からのオフセットであると決められています．割り込みや例外が発生すると，**表2**に示すアドレスにある命令を実行します．その際，例外コードはシステム・レジスタのECRレジスタに設定されます．NMI（Non-maskable Interrupt）の割り込みの際にはECRの上位16ビットを，そうでないときにはECRの下位16ビットを例外ベクタ領域からのオフセットとして，発生した割り込みや例外処理のハンドラへ分岐することが可能です．

割り込みや例外が発生したときには，EIPCレジスタに**表2**の復帰プログラム・カウンタ（PC）に示すアドレスがセットされます．この値から，どの命令の実行時に例外が発生したかを特定できます．また，マスカブル割り込みのイネーブルとディセーブルにはそれぞれeiの命令とdiの命令が用いられます．そして割り込み・例外処理から復帰する際にはretiの命令が用いられます．

なお，デバッグ・トラップ例外から復帰する際にはdbretの命令が用いられます．

● ターゲットCPUはV850ES/JG2

ここでは実際のCPUとして，本誌2007年5月号付属

KEYWORD ── V850，850ES/JG2，μPD70F3716GC，シリアル通信モジュール，UARTA，スタートアップ・ルーチン，startup.s，リンカ・スクリプト，例外ハンドラ

表1　V850ESの命令一覧

命令の種類	ニモニック	オペランド	バイト	実行クロック数
ロード命令	ld.b	disp16[reg1],reg2	4	1
	ld.h	disp16[reg1],reg2	4	1
	ld.w	disp16[reg1],reg2	4	1
	ld.bu	disp16[reg1],reg2	4	1
	ld.hu	disp16[reg1],reg2	4	1
	sld.b	disp7[ep],reg2	2	1
	sld.bu	disp4[ep],reg2	2	1
	sld.h	disp8[ep],reg2	2	1
	sld.hu	disp5[ep],reg2	2	1
	sld.w	disp8[ep],reg2	2	1
ストア命令	st.b	reg2,disp16[reg1]	4	1
	st.h	reg2,disp16[reg1]	4	1
	st.w	reg2,disp16[reg1]	4	1
	sst.b	reg2,disp7[ep]	2	1
	sst.h	reg2,disp8[ep]	2	1
	sst.w	reg2,disp8[ep]	2	1
乗算命令	mul	reg1,reg2,reg3	4	1
	mul	imm9,reg2,reg3	4	1
	mulh	reg1,reg2	2	1
	mulh	imm5,reg2	2	1
	mulhi	imm16,reg1,reg2	4	1
	mulu	reg1,reg2,reg3	4	1
	mulu	imm9,reg2,reg3	4	1
算術演算命令	add	reg1,reg2	2	1
	add	imm5,reg2	2	1
	addi	imm16,reg1,reg2	4	1
	cmov	cccc,reg1,reg2,reg3	4	1
	cmov	cccc,imm5,reg2,reg3	4	1
	cmp	reg1,reg2	2	1
	cmp	imm5,reg2	2	1
	div	reg1,reg2,reg3	4	35
	divh	reg1,reg2	2	35
	divh	reg1,reg2,reg3	4	35
	divhu	reg1,reg2,reg3	4	34
	divu	reg1,reg2,reg3	4	34
	mov	reg1,reg2	2	1
	mov	imm5,reg2	2	1
	mov	imm32,reg1	6	2
	movea	imm16,reg1,reg2	4	1
	movhi	imm16,reg1,reg2	4	1
	sasf	cccc,reg2	4	1
	setf	cccc,reg2	4	1
	sub	reg1,reg2	2	1
	subr	reg1,reg2	2	1
飽和演算命令	satadd	reg1,reg2	2	1
	satadd	imm5,reg2	2	1
	satsub	reg1,reg2	2	1
	satsubi	imm16,reg1,reg2	4	1
	satsubr	reg1,reg2	2	1
論理演算命令	and	reg1,reg2	2	1
	andi	imm16,reg1,reg2	4	1

命令の種類	ニモニック	オペランド	バイト	実行クロック数
論理演算命令	bsh	reg2,reg3	4	1
	bsw	reg2,reg3	4	1
	hsw	reg2,reg3	4	1
	not	reg1,reg2	2	1
	or	reg1,reg2	2	1
	ori	imm16,reg1,reg2	4	1
	sar	reg1,reg2	4	1
	sar	imm5,reg2	2	1
	shl	reg1,reg2	4	1
	shl	imm5,reg2	2	1
	shr	reg1,reg2	4	1
	shr	imm5,reg2	2	1
	sxb	reg1	2	1
	sxh	reg1	2	1
	tst	reg1,reg2	2	1
	xor	reg1,reg2	2	1
	xori	imm16,reg1,reg2	4	1
	zxb	reg1	2	1
	zxh	reg1	2	1
分岐命令	bcond	disp9（条件成立時）	2	2＊1
		disp9（条件不成立時）	2	1
	jarl	disp22,reg2	4	2
	jmp	[reg1]	2	3
	jr	disp22	4	2
ビット操作命令	clr1	bit#3,disp16[reg1]	4	3＊2
	clr1	reg2,[reg1]	4	3＊2
	not1	bit#3,disp16[reg1]	4	3＊2
	not1	reg2,[reg1]	4	3＊2
	set1	bit#3,disp16[reg1]	4	3＊2
	set1	reg2,[reg1]	4	3＊2
	tst1	bit#3,disp16[reg1]	4	3＊2
	tst1	reg2,[reg1]	4	3＊2
特殊命令	callt	imm6	2	4
	ctret	—	4	3
	di	—	4	1
	dispose	imm5,list12	4	*n*+1＊3
	dispose	imm5,list12,[reg1]	4	*n*+3＊3
	ei	—	4	1
	halt	—	4	1
	ldsr	reg2,regID	4	1
	nop	—	2	1
	prepare	list12,imm5	4	*n*+1＊3
	prepare	list12,imm5,sp	4	*n*+2＊3
	prepare	list12,imm5,imm16	6	*n*+2＊3
	prepare	list12,imm5,imm32	8	*n*+3＊3
	reti	—	4	3
	stsr	regID, reg2	4	1
	switch	reg1	2	5
	trap	vector	4	3
デバッグ機能用命令	dbret	—	4	3
	dbtrap	—	2	3

＊1 直前にPSWの内容を書き換える命令がある場合は3　＊2 ウェイト・ステートがない場合（3＋リード・アクセス・ウェイト・ステート数）
＊3 *n*は，list xで指定されるレジスタの合計数（ウェイト・ステート数による．ウェイト・ステートがない場合，*n*はlist xで指定されるレジスタの合計数．*N*＝0の場合は，*n*＝1の場合と同じ）

（**a**）命令一覧