組み込みマイコンの仕組みを理解しよう

第2回

FRマイコンの命令セットとメモリ・アーキテクチャ

平石 郁雄

本連載では，富士通の32ビット・マイコン「FRファミリ」を例に，マイコンの仕組みや使い方について解説している．今回は，命令セットとメモリ・アーキテクチャを中心に説明する．（編集部）

高級言語（C言語など）に慣れ親しんでいても，アセンブリ言語を使いこなしている人は少ないのではないでしょうか．アセンブリ言語は，機械語とほとんど等価な命令の集まり（命令セット）で，CPUの種類に依存しています．最近では，アセンブリ言語で作成したプログラムを読解できる技術者がいないという話も聞こえてきます．今日では高級言語を機械語に変換するコンパイラ（**図1**）と呼ばれるツールの性能が向上したこともあり，組み込みソフトウェア開発者のほとんどが，アセンブリ言語ではなく，CPUの種類を意識する必要がない高級言語でプログラム開発を行っています．

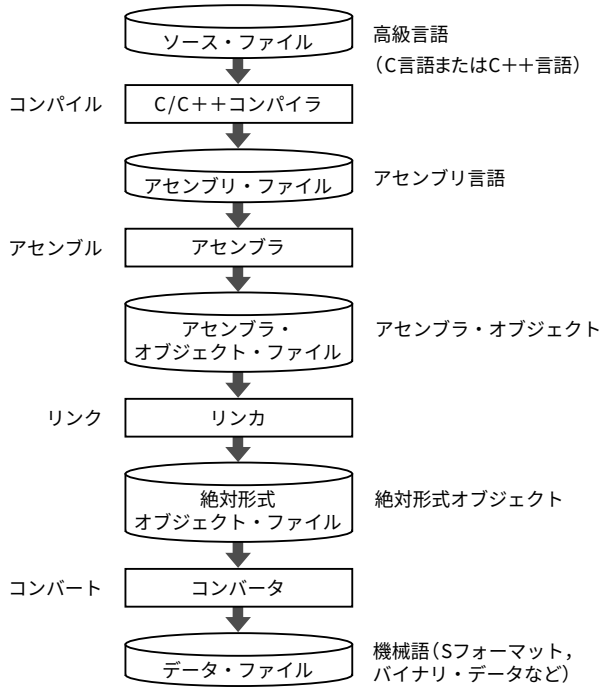


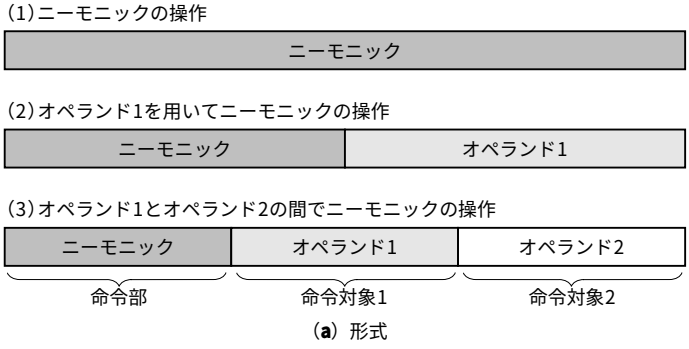
図1　C/C++コンパイルの例

C/C++コンパイラは，高級言語（C言語またはC++言語）ファイルをコンパイルしてアセンブリ言語ファイルを生成する．アセンブラは，アセンブリ言語ファイルをアセンブルしてアセンブラ・オブジェクトを生成する．リンカは，アセンブラが生成した複数のアセンブラ・オブジェクト・ファイルをリンクして，絶対形式オブジェクト・ファイルを生成する．コンバータは，絶対形式オブジェクト・ファイルを変換して，データ・ファイル（機械語）を生成する．

それではアセンブリ言語を学習する必要はないかというと，組み込み系のソフトウェア開発はそう簡単にはいきません．アプリケーションによっては，使用しているマイコンの最高性能を引き出すために，アセンブリ言語でゴリゴリと記述することを強いられます．また，ハードウェアの動きを正確に知るためには，アセンブリ言語を読み取る能力が要求されます．そのため，アセンブリ言語を知ることは，効率的な開発を行う上で必要なことと言えます．

● ニーモニック＋オペランドから構成される

アセンブリ言語は，「ニーモニック」と「オペランド」か



（**a**）形式

(1)ニーモニックの操作

```
<ニーモニック>
  RET
```

オペレーション：RPの値をPC（プログラム・カウンタ）へ格納する

(2)オペランド1を用いてニーモニックの操作

```
<ニーモニック><オペランド1>
  JMP          @R1
```

オペレーション：R1(オペランド1)の値をPCへ格納する

(3)オペランド1とオペランド2の間でニーモニックの操作

```
<ニーモニック><オペランド1><オペランド2>
  ADD          R1,          R2
```

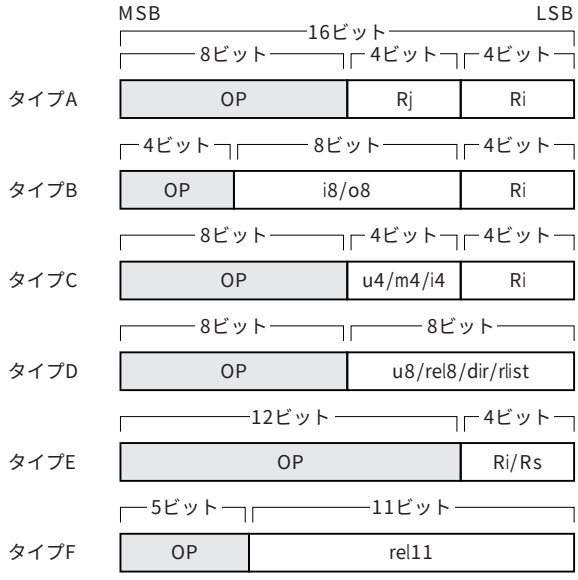
オペレーション：　R1（オペランド1）とR2（オペランド2）の値を加算して，R2へ格納する

（**b**）記述例

図2　命令の記述形式

FRの命令セットは，ニーモニックとオペランドを組み合わせた3種類の命令記述形式で構成している．

KEYWORD――マイコン，FRファミリ，アセンブリ言語，命令セット，加減算命令，乗除算命令，即値データ転送命令，シフト演算命令，比較命令，レジスタ間転送命令，専用レジスタ転送命令，メモリ・アーキテクチャ



（**a**）フォーマット

Ri/Rj	レジスタ
0000	R0
0001	R1
0010	R2
0011	R3
0100	R4
0101	R5
0110	R6
0111	R7
1000	R8
1001	R9
1010	R10
1011	R11
1100	R12
1101	R13
1110	R14
1111	R15

RS	レジスタ
0000	TBR
0001	RP
0010	SSP
0011	USP
0100	MDH
0101	MDL
0110	予約
0111	予約
1000	予約
1001	予約
1010	予約
1011	予約
1100	予約
1101	予約
1110	予約
1111	予約

（**b**）汎用/専用レジスタのフィールド・ビット・パターン

上位4ビット

下位4ビット	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	LD @(R13, Rj),Ri	ST Ri, @(R13,Rj)	LD@(R14,disp10),R	STRi,@(R14,disp10)	LDUH@(R14,disp9),Ri	STHRi,@(R14,disp9)	LDUB@(R14,disp8),Ri	STBRi,@(R14,disp8)	BANDL #u4,@Ri	BORL #u4,@Ri	ADDN #i4,Ri	LSR #u4,Ri	LDI:8 #i8,Ri	CALL label12	BRA label9	BRA:D label9
1	LDUH @ (R13,Rj),Ri	STH Ri, @(R13,Rj)							BANDH #u4,@Ri	BORH #u4,@Ri	ADDN2 #i4,Ri	LSR2 #u4,Ri			BNO label9	BNO:D label9
2	LDUB @ (R13,Rj),Ri	STB Ri, @(R13,Rj)							AND Rj,Ri	OR Rj,Ri	ADDN Rj,Ri	LSR Rj,Ri			BEQ label9	BEQ:D label9
3	LD @(R15, udisp6),Ri	ST Ri, @(R15,ud6)							ANDCCR #u8	ORCCR #u8	ADDSP #s10	MOV Ri,Rs			BNE label9	BNE:D label9
4	LD @Rj,Ri	ST Ri,@Rj							AND Rj, @Ri	OR Rj, @Ri	ADD #i4,Ri	LSL #u4,Ri			BC label9	BC:D label9
5	LDUH @Rj,Ri	STH Ri, @Rj							ANDH Rj,@Ri	ORH Rj, @Ri	ADD2 #i4,Ri	LSL2 #u4,Ri			BNC label9	BNC:D label9
6	LDUB @Rj,Ri	STB Ri, @Rj							ANDB Rj,@Ri	ORB Rj, @Ri	ADD Rj,Ri	LSL Rj,Ri			BN label9	BN:D label9
7	E format	E format							STILM #u8	E format	ADDC Rj,Ri	MOV Rs,Ri			BP label9	BP:D label9
8	DMOV @ d10,R13	DMOV R13,@d10							BTSTL #u4,@Ri	BEORL #u4,@Ri	CMP #i4,Ri	ASR #u4,Ri		CALL:D label12	BV label9	BV:D label9
9	DMOVH @d9,R13	DMOVH R13,@d9							BTSTH #u4,@Ri	BEORH #u4,@Ri	CMP2 #i4,Ri	ASR2 #u4,Ri			BNV label9	BNV:D label9
A	DMOVB @d8,R13	DMOVB R13,@d8							XCHB @Rj,Ri	EOR Rj,Ri	CMP Rj,Ri	ASR Rj,Ri			BLT label9	BLT:D label9
B	DMOV @ d10,@－R15	DMOV @ R15＋,@d10							MOV Rj,Ri	LD:20 #i20,Ri	MULU Rj,Ri	MULUH Rj,Ri			BGE label9	BGE:D label9
C	DMOV @ d10,@R13＋	DMOV @ R13＋,@d10							LDM0 (reglist)	EOR Rj,@Ri	SUB Rj,Ri	LDRES @Ri＋,#u4			BLE label9	BLE:D label9
D	DMOVH @d9,@R13＋	DMOVH @R13＋,@d9							LDM1 (reglist)	EORH Rj,@Ri	SUBC Rj,Ri	STRES #u4,@Ri＋			BGT label9	BGT:D label9
E	DMOVB @d8,@R13＋	DMOVB @R13＋,@d8							STM0 (reglist)	EORB Rj,@Ri	SUBN Rj,Ri				BLS label9	BLS:D label9
F	ENTER #u10	INT #u8							STM1 (reglist)	E format	MUL Rj,Ri	MULH Rj,Ri			BHI label9	BHI:D label9

（**c**）命令マップ

図3　命令フォーマット

命令コードの基本長が16ビットなので，限られた資源を有効に活用するため，メモリマップがぎっしり詰まっている．