ICF3-Z 命令セットアーキテクチャ仕様(2020年1月15日版)

平山 直紀

https://icf3z.idletime.tokyo/

1. 概要

ICF3-Z は 1999 年に製品出荷された暗号 LSI ICF3 をベースに作られた 8bit CPU のオープンソースハードウェア。高性能で面積が小さいことが特徴。命令コードが 32bit であるため 1 命令で重複しない複数の処理が実行できます。32bit の命令コードは 8bit CPU にしてはメモリ効率が悪い問題がありますが 16bit の圧縮命令を実装することで対策しています。圧縮命令は自由に命令を作れるため、互換性やコンパイラの開発が容易になるなどのメリットがあります。割込み機能を含めたプロセッサ全体が、とても簡潔に実装できます。ライセンスについて Web サイトで確認してください。

2. 基本仕様

・命令コード 1 ワード 32bit 固定長(圧縮命令は 16bit)
・すべての命令コードは 1 サイクルで実行される。
・1 サイクルの遅延分岐(圧縮命令や J 系の命令に遅延スロットはなく 1 サイクルストール)
  LOOP 命令のみ 2 サイクルの遅延分岐
・汎用レジスタ 16 本×2 バンク
・スクラッチパッド 最大 256 バイト(32 バイト or 128 バイト or 256 バイト)
  先頭 32 バイトは汎用レジスタ
・データメモリ(オプション) 最大 64KB 先頭 256 バイトはスクラッチパッドメモリ
・プログラムメモリ 最大 64K ワード(256KB)
  将来的には最大 256K ワード(1024KB)までの拡張可能性
  ROM 実装可能
・スタック用メモリ 16bit×最大 16 段(4 段、8 段、16 段)
  圧縮命令を使う場合は 1 段少なくなります。
・8bit の I/O ポート 256 個

3. レジスタ、フラグ

#	名前	bit 幅	用途
1	A	8	演算レジスタ
2	B	8	演算レジスタ
3	C	8	演算レジスタ
4	D	8	演算レジスタ
5	CF	1	キャリーフラグ
6	ZF	1	ゼロフラグ
7	I	4	ループ制御カウンタ
8	R0～R15	8	汎用レジスタ 16 本
9	PC	16	プログラムカウンタ

4. 演算レジスタと汎用レジスタ間の転送

汎用レジスタから演算レジスタ(A,B,C,D)に転送するのに 1 サイクル
A レジスタから汎用レジスタに転送するのに 1 サイクル
スクラッチパッドメモリも汎用レジスタと同じ 1 サイクルで転送

5. キャリーフラグ(CF)とゼロフラグ(ZF)

キャリーフラグ(CF)はオペコードが ADD 命令の場合にセットされる。
ゼロフラグ(ZF)はオペコードが AND/OR/XOR 命令の場合にセットされる。CF ビット(bit9)を 1 にすると ZF を伝搬する。つまり演算結果の ZF と前回の ZF との AND を新しい ZF にする。

6. ループ制御レジスタ

レジスタ I はループ制御専用のレジスタ。I=X 命令で値を代入。LOOP 命令や WAIT 命令で使われる。

7. 命令コードフォーマット

通常の 32bit 命令

bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24
0	COPR	オペコード					A レジ

bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16
BC レジスタの入力				D レジ	STORE	R(N/ADDR/REGADDR)	

bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
ONE	NOT	ADDR=[B,C,D]		ADDL=[A,DIVA,N]		CF	MUL

bit7-0 は、次の 3 通りの意味

●X=n,Z=n 命令 汎用レジスタの番号

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
X(リード レジスタ番号)				Z(ライトレジスタ番号)			

●N=n 命令

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
即値(8bit)							

●分岐命令のアドレス

bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
アドレス上位 8bit							

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
アドレス下位 8bit							

16bit アドレスを使う分岐系の命令と同時にできるのは B=LSH のみ

●LOOP 命令の相対アドレス

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
相対アドレス							

16bit 圧縮命令

bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24
1	圧縮命令オペコード						

bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16
オペランド							

圧縮命令のオペランドは bit7-0 にマッピングされる

通常命令で COPR ビットが 1 の場合、IR レジスタの 7-0 bit を圧縮命令のオペランドとして格納されている CO レジスタに置換する。

8. ニモニック

ニモニック	bit	値	意味
NOP	29-25	0	何もしない
I=X		1	I レジスタに即値 X を代入
B/JB(ラベル)		2	無条件分岐
BCF0/BCF1(ラベル)		3	CF が 0 あるいは 1 なら分岐
BZF0/BZF1(ラベル)		4	ZF が 0 あるいは 1 なら分岐
BC0/BC1(ラベル)		5	C の最下位が 0 あるいは 1 なら分岐
CALL/JCALL(ラベル)		6	サブルーチンコール
CALL0/CALL1(ラベル)		7	ZF が 0 あるいは 1 ならコール
RETURN		8	リターン(遅延スロット有)
JRETURN		9	リターン(遅延スロット無)
RETURN0		10	ZF が 0 ならリターン
RETURN1		11	ZF が 1 ならリターン
LOOP(ラベル)		12	I≠0 なら分岐と I=I-1
WAIT		13	I≠0 なら WAIT と I=I-1
D=REG		14	D=ANS と併用することで D=REG
(空)		15	未定義
ADD		16	デフォルト ADD 命令だが CF セットに
AND		17	CF ビットが 1 のときは CF と連携した結
XOR		18	果になる。
OR		19	

MOV		20	STORE する値を A レジスタから REG に
REGBANK		21	X でバンク番号を指定(0,1 のみ)
ENABLE		22	割込み可能状態
DISABLE		23	割込み禁止状態
RETURNI		24	割込みからの復帰
POPSP		25	圧縮命令内の分岐制御。SP=SP+1,HL=0
IOSTROBE		26	xIOSTROBE_P 信号を 1 にする
OUTPUTK		27	N を出力 xWSTROBEK_P を 1 に
INPUTOUTPUT		28	INPUT 命令と OUTPUT 命令の同時
INPUT		29	xINPORT_P[7:0]を ANS に xRSTROBE_P を 1
OUTPUT		30	D レジスタを出力 xWSTROBE_P を 1 に
(空)		31	未定義
A=ANS	24		A レジスタに代入
B=[ANS,LSH,RSH,REG] C=[ANS,LSH,RSH,REG]	23-20		B レジスタに代入 C レジスタに代入
D=[ANS,REG]	19,29-25		D に代入、D=REG はオペコードと併用
STORE	18		A レジスタを汎用レジスタに書き込み
R(N) R(ADDR) R(REGADDR)	17-16		データメモリへのアクセス方法を指定
MUL	15		乗算制御ビット
NOT	14		ADDR の出力を反転
ADDR=[B,C,D]	13-12		ADDR の選択 B、C、D
ADDL=[A,DIVA,N]	11-10		ADDL の選択 DIVA は除算用の A
CF	9		加算器の入力キャリーに CF の値
ONE	8		加算器の入力キャリーに 1 の値を入れる
X=n	7-4		汎用レジスタのリードアドレス
Z=n	3-0		汎用レジスタのライトアドレス OUTPUTK の即値
N=n	7-0		汎用レジスタのアドレス ADDL の即値、I レジスタへの即値

9. B、C レジスタ

演算レジスタ B と C は乗算、除算、余算を演算するためにレジスタへの入力が作られています。どちらのレジスタも ANS、LSH、RSH、REG の値を選択できるニモニックになっている。しかし B と C 合わせて 4bit しか制御信号がなく全部を表現できない。B=LSH、C=RSH と B=RSH、C=LSH はほとんど可能性のない組み合わせです。それを使って B、C レジスタの入力選択の信号を 4bit にします。前述の組み合わせができない他、B=REG を選択すると C は選択できません。C=REG を選択すると B は選択できません。

#	B				C				REG
	0	ANS	LSH	RSH	0	ANS	LSH	RSH	
0	○				○				
1	○					○			
2	○						○		
3	○							○	
4		○			○				
5		○				○			
6		○					○		
7		○						○	
8			○		○				
9			○			○			
10			○				○		
11			×					×	B=REG
12				○	○				
13				○		○			
14				×			×		C=REG
15				○				○	

10. D レジスタ

演算レジスタ D は ANS と REG の値を代入すること可能ですが REG の値を代入するにはオペコード D=REG と D=ANS の両方を設定する必要があります。アセンブラでは D=REG のニモニックを記述すると自動的に D=ANS も設定されます。

11. R(xx)

データメモリの先頭 16 バイトは汎用レジスタの 0 番～15 番です。R(N)を指定すると即値の N を使って 0 番地～255 番にアクセスできます。R(ADDR)を指定すると ADDR の値を使って 0 番地～255 番地にアクセスできます。すなわちレジスタ B、C、D の値をアドレスとしてメモリアクセスできます。256 バイト以上データメモリが実装されている場合は、R(REGADDR)を指定します。16bit のアドレスの下位 8bit は ADDR で指定されたもの。上位 8bit は前サイクルで読み出した REG の値です。
例えば汎用レジスタの 14 番と 15 番に 16bit アドレスのポインタが格納されている場合、そのポインタを使ってデータメモリからリードするコードは次のようになります。

```
    X=14
    B=REG;X=15
    ADDR=B;R(REGADDR)
    C=REG
```

A レジスタの値をライトする場合

```
    X=14
    B=REG;X=15
    ADDR=B;R(REGADDR);STORE
```

16bit のポインタは、スクラッチパッドメモリ上にも格納できます。
62 番と 63 番に 16bit ポインタを格納している場合のリード

```
    R(N);N=62
    B=REG;R(N);N=63
    ADDR=B;R(REGADDR)
    C=REG
```

12-13 番のレジスタで示されるアドレスのメモリを 14-15 番のレジスタで示されるアドレスに 8 バイトコピーをするコード(上位 8bit のアドレスは 8 バイトの先頭と最後で同じ)

```
    X=12
    X=14;D=REG
    C=REG;I=X;X=7
LBL0:
    X=13
    ADDR=D;R(REGADDR);ONE;D=ANS
    LOOP(LBL0);B=REG
    ADDR=B;A=ANS;X=15
    ADDR=C;R(REGADDR);ONE;C=ANS;STORE
```

LOOP 命令は 2 サイクルの遅延分岐なので実行サイクルは 3+5×8 サイクル

12. 圧縮命令

圧縮命令ではオペコード×8+4 のアドレスにサブルーチンコールする。遅延スロットにある命令は無効化される。圧縮命令は 128 命令定義できる。通常、1 命令は 8 ワードで実装するが、分岐命令を使ってそれ以上のワードで実装しても構わない。圧縮命令実行中、圧縮命令を実行することはできません。

13. I=X 命令

I=X 命令は X=n と合わせて I レジスタに即値 n(0～15)を入れる。汎用レジスタへのストアや、A,B,C,D の演算レジスタを使った演算などが並列に動作する。

14. 遅延分岐と遅延スロット

基本的に分岐命令には 1 サイクルの遅延スロットがある。ただし一文字目が"J"の分岐命令(JB、JBI、JCALL、JRETURN)は 1 サイクルの遅延分岐だが遅延スロットの命令の実行はキャンセルされる。RETURN 系以外の分岐命令で COPR ビットは使用不可。
LOOP 命令のみ 2 サイクルの遅延スロット。

15. LOOP 命令

I=0 なら、プログラムアドレスの後方への相対ジャンプ。I≠0 なら I=I-1。
2 サイクルの遅延スロットがある。PC = PC+2 – N(8bit)

16. WAIT 命令

I≠0 なら、もう一度、同じ命令コードを実行し、I=I-1。
I=0 なら、次の命令コードに移る。遅延スロットはなく WAIT 命令がある次の命令が実行される。

17. 間接ジャンプ命令

ジャンプ命令のアドレスに 1 番地を指定すると、遅延スロットの命令を実行後、1 番地ある命令を実行して、レジスタ A,B のアドレスにジャンプ。

18. POPSP 命令

圧縮命令からジャンプ命令をする場合には、必ず POPSP 命令を実行してからジャンプ命令を実行すること。内部的な動作はスタックポインタ SP を 1 つポップして HL ビットをクリアする。

19. REGBANK 命令

汎用レジスタ 0 から 15 はスクラッチパッドメモリの 0 から 15 にマップされている。これを別の場所にマップする命令。マップする位置は X=n で指定する。X=0 は 0-15 にマップ。X=1 は 16-31 にマップ。X=2 以降はオプション。

20. INPUT 命令

INPUT 命令はレジスタのアドレスを指定する Z と一緒に使います。

Z	内容
0	xINPORT_P
1	リターンアドレスの下位バイト
2	リターンアドレスの上位バイト
3	割込み復帰アドレスの下位バイト
4	割込み復帰アドレスの上位バイト

21. プログラムにデータを置く方法

定数のデータをプログラムに置いて 1 バイトづつ読み出す方法(A レジスタに読み出し)

```
RETURN;A=ANS;ADDL=N;N=n
```

22. メモリマップ

プログラムをする上での仕様です。**必ず守る必要があります。**

アドレス	内容
0 番地	JB 命令でプログラムの開始アドレスにジャンプさせます。
1 番地	NOP 命令
2 番地	割込み 0 番(最優先の割込み) 割込み 0 番を使う場合は JB 命令で割込み処理のアドレスに分岐
3 番地	割込み 1 番 割込み 1 番を使う場合は JB 命令で割込み処理のアドレスに分岐
4 番地	圧縮命令の 0 番目

5 番地	リザーブ（割込み 2 番）
6 番地	リザーブ（割込み 3 番）
7 番地	リザーブ（割込み 4 番）

割込みや圧縮命令を使わない場合、2 番地以降にプログラムを置くことができます。

23. 入出力ポート

入力は xPORTID_P[7:0]でポートを指定して xINPORT_P[7:0]で受ける。
出力は xPORTID_P[7:0]でポートを指定して xOUTPORT_P[7:0]で出力する。
B レジスタの値がそのまま xPORTID_P[7:0]になっている。そのことを考慮して設計しなければならない。OUTPUTK 命令実行中以外では D レジスタの値がそのまま xOUTPORT_P[7:0]になっている。そのことを考慮して設計しなければならない。

IOSTROBE 命令

xIOSTROBE_P 信号が 1 サイクル間 1 になる。割込み以外では IOSTROBE 命令を連続させている間、xIOSTROBE_P 信号が途中で 0 になることはない。

INPUT 命令

xRSTROBE_P が 1 サイクル間 1 になる。得られる値を Z で指定する。

OUTPUT 命令

D レジスタが xOUTPORT_P[7:0]に出力され xWSTROBE_P が 1 サイクル間 1 になる。

OUTPUTK 命令

即値 Z が xOUTPORT_P[7:0]に出力され、xWSTROBEK_P が 1 サイクル間、1 になる。

INPUTOUTPUT 命令

INPUT 命令と OUTPUT 命令の同時

例 ポート ID 3 の入力を R4 に代入

```
ADDL=N;N=3;B=ANS
INPUT;C=ANS
A=ANS;ADDR=C
R4=A
```

例 R7 をポート ID 5 に出力

```
X=7
ADDL=N;N=5;B=ANS;D=REG
OUTPUT
```

24. 割込み

割込みは0番と1番がある。割込みから復帰するにはRETURNI命令を使います。RETURNI命令の後には必ずNOP命令を入れてください。通常のRETURN命令の遅延スロットとは異なりNOP命令は実行されたり、されなかったりします。同時に割り込みが来た場合、0番が優先されます。割込み復帰命令のRETURNI命令を実行するときに0番の割込み信号が来ていれば、復帰することなく0番の割込みへ分岐します(このとき遅延スロットの命令が実行されます)。RETURNI命令を実行するときに1番の割込み信号が来ていても、一度、復帰して1サイクル実行後、1番の割込みに分岐する。つまり1番の割込み信号を、ずっと1にしていると、毎サイクル、1番の割込みルーチンに分岐する。これを使ってデバッガの実装をする。INPUT命令で割込み復帰アドレス後に実行する命令の次の命令のアドレスを取得することができる。ただし分岐命令の直後に置かれる命令は、キャンセルされるので、かならずその命令が実行されるとは限らない。

```
    JB(LBL1)
LBL0:
    ADD
LBL1:
```

LBL0 に分岐してくる回数を計算してブレークする処理を割込みで行っても、上にある JB命令の後にも、割込みが入って回数に加算されてしまう。
正しい回数(パスカウント)にするためには分岐命令の後のキャンセルされるような命令にブレークポイントを設定しない。(させない)
また圧縮命令は1つのアドレスに2個の命令があるため、2回計算されます。

Apenddix A 乗算サンプルコード

(1) 乗算　8bit × 8bit　= 16bit (9 サイクル)

BC = A×C+B

```
I=X;X=7
MUL;ADDL=A;ADDR=B;B=RSH;C=RSH;WAIT
```

(2) 8bit × 3bit + 8bit = 8bit

B = A×C+B　4 サイクル

```
NOT;ONE;ADD
MUL;ADDL=A;ADDR=B;B=ANS;C=RSH
MUL;ADDL=DIVA;ADDR=B;B=ANS;C=RSH
MUL;ADDL=DIVA;ADDR=B;B=ANS
```

(3) 8bit × 5bit + 8bit = 8bit

B = A×C+B　6 サイクル

```
I=X;X=3;NOT;ONE;ADD
MUL;ADDL=A;ADDR=B;B=ANS;C=RSH
MUL;ADDL=DIVA;ADDR=B;B=ANS;C=RSH;WAIT
```

Apenddix B 除算サンプルコード

(1) 除算 16bit÷8bit (17 サイクル)

BC = BC ÷ D

A = BC mod D

```
I=X;X=15;A=ANS
ADDL=DIVA;ADDR=D;ONE;NOT;A=ANS;B=LSH;C=LSH;WAIT
```

(2) 除算 8bit÷8bit (9 サイクル)

C = B ÷ D

A = B mod D

```
I=X;X=7;A=ANS
ADDL=DIVA;ADDR=D;ONE;NOT;A=ANS;B=LSH;C=LSH;WAIT
```

(3) 除算(A<D ケース) 16bit÷8bit　(9 サイクル)

C = AB ÷ D

A = AB mod D

```
I=X;X=7
ADDL=DIVA;ADDR=D;ONE;NOT;A=ANS;B=LSH;C=LSH;WAIT
```

Apenddix C メモリ転送のサンプルコード

このサンプルは拡張 RAM が装備されている必要があります。
12-13 番のレジスタで示されるアドレスのメモリを 14-15 番のレジスタで示されるアドレスに 16 バイトコピーをするコード

```
    X=12
    X=14;D=REG
    C=REG;I=X;X=15
LBL0:
    X=13
    ADDR=D;R(REGADDR);ONE;D=ANS;B=REG
    B=REG;ADDR=B;CF;A=ANS
    STORE;Z=13;ADDR=B;A=ANS;X=15
    ADDR=C;R(REGADDR);ONE;C=ANS;B=REG;STORE
    LOOP(LBL0)
    ADDR=B;CF;A=ANS
    STORE;Z=15
```

Apenddix D INPUT したデータの高速な処理

ポート ID3 を INPUT 命令で取得したデータの bit 0 を判定して分岐

```
    ADDL=N;N=3;B=ANS
    INPUT;C=ANS
    BC0(xxxx)
```

Apenddix E INPUTOUTPUT 命令を使った高速なデータ転送

これはポート ID が INPUT と OUTPUT で同じである必要があるので、使いにくいかもしれません。
256 バイトのデータを xINPORT_P から読み込み、そのまま xOUTPORT_P に出力する。
ポート ID は、いづれも 6。

```
    I=N;N=31
    ADDL=N;N=6;B=ANS
loop:
    INPUT;C=ANS;D=ANS
    INPUTOUTPUT;C=ANS;D=ANS
    INPUTOUTPUT;C=ANS;D=ANS
    INPUTOUTPUT;C=ANS;D=ANS
    INPUTOUTPUT;C=ANS;D=ANS
    INPUTOUTPUT;C=ANS;D=ANS
    INPUTOUTPUT;C=ANS;D=ANS
    INPUTOUTPUT;C=ANS;D=ANS
    OUTPUT
    BI(loop)
```