WZeta 命令セット SDog コア設計資料(β版 2022/08/22)

平山 直紀(Naoki Hirayama)

https://wzeta.idletime.tokyo/

1 概要

WZeta SDog はトランジスタ数が非常に少ない超軽量な 8bit CPU です。AES 暗号が動作する最小クラスの CPU を目標として設計しています。8bit レジスタ 3 個の CPU です。WZeta の命令コードは 16bit 固定長ですが 8bit 単位に送信できる命令セットです。通常とは逆順のオペランドから送信することでパイプライン化をすることなく高速に動作します。命令数は少ないですが平易でわかりやすい命令セットです。レジスタが少ないのでメモリを 16 ビットポインタにすることが便利な命令(INCX、DECX)があります。乗算命令のない 8bit CPU で剰余演算を高速かつ省メモリに実行できることを考慮しています。利用者定義命令をソフトウェアで追加可能なためメモリ効率にも優れる。スタックポインタのハードウェアが全く存在しないノイマン型アーキテクチャ(プログラムメモリとデータメモリを共有)とすることで部品を少なくして製造コストが下がることが期待されています。

2 基本仕様

・命令コード 1 ワード 16bit 固定長
・1 命令は 4 サイクルで実行
　分岐を含めた全命令が 4 サイクルであるため PWM 制御などで使い易い
・ノイマン型アーキテクチャ
・8bit 汎用レジスタ 2 本、8bit ループ制御用レジスタ 1 本
・メモリ　標準 2KB(最小 512 バイト、最大 192KB)
　　1 つのメモリをプログラム領域とデータ領域に分けて利用するモードと混在する 2 つのモードがある
　　プログラム領域とデータ領域の分割位置を変更可能。変更方法は実装依存。
　　HALF モデルは搭載メモリの前半がプログラム、後半がデータ。
　　後ろから 4 分の一、 8 分の一、16 分の一などの設定可。

　　偶数サイクルのアクセスがプログラムメモリへのアクセス、奇数サイクルのアクセスが
　　データメモリへのアクセスなのでプログラムとデータで物理的に異なるメモリも可能

・8bit の I/O ポートと 8bit のバス
・外部割込 2 本(優先度付き)
・2 の補数

3 レジスタ、フラグ、ステータス

3 - 1　レジスタ

#	名前	bit 幅	用途
1	A	8	汎用レジスタ
2	B	8	汎用レジスタ
3	C	8	ループ制御用
4	PC	9-17	プログラムカウンタ(デフォルト 12)
5	MC	9-17	ミリコードカウンタ(デフォルト 12)
6	IC	9-17	割込みカウンタ(デフォルト 12)
7	JRA	8-16	相対分岐アドレス(デフォルト 11)

3 - 2 フラグ

bit	名前	用途
0	ZF	ゼロフラグ
1	MF	ミリフラグ(ミリコード実行中を示す)
2	UF	ユーザーモード フラグ(1 の場合、ユーザーモードを示す)
3	OF0	オフセットの 0bit 目([n]で示す 7bit のアドレスに OF を追加)
4	OF1	オフセットの 1bit 目(　　〃　　)
5		(未使用)
6		(未使用)
7	CF	キャリーフラグ

3 - 3 ステータス

bit	名前	用途
0	AE	ユーザーモードフラグ(UF)が 1 のとき保護された低位のアドレスにアクセス
1	TM	タイマー割込み(実装依存) 試作未実装
2		(未使用)
3		(未使用)
4		(未使用)
5		(未使用)
6		(未使用)
7	EF	割込み許可状態　0: 割込み禁止　1: 割込み許可

４命令コードフォーマット

16bit 固定長の命令コード。

bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
opmsb	macro	OP コード					

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
オペランド xx(8bit)							

opmsb ビット ： モードによって役割が異なる。「デバッグ」、「パリティ」、「高速」が存在する

macro ビット ： ハードマクロ命令の開始と終了

オペランド xx は値によってアドレッシングモードを変更する。オペランドを先に転送してもオペコードを待つことなく投機的にメモリをリードすることが可能。(投機的にすることを強制するわけではない)

xx≦126 アドレスは xx

xx=127 アドレスは A:B

xx≧128 アドレスは(xx-80h):B

OF ビットが設定される場合、xx≦126 を指定すると OF×128+xx になる。ただし割込み実行中は OF は影響しません。常に OF=2'b00 と同じです。

ユーザモード・フラグ(UF=1)の場合は、アクセス制限を有効化する。制限内容は各実装で決められる。
SDog では、あらかじめ決められたアドレスより低位へのアクセスは制限される。具体的には各命令で[n:B]、[A:B]形式による読み書きが制限されます。

例えばシステムで 2KB が設定されると OF ビットが 3 であっても 512B までしかアクセスできないため、512B～2KB に暗号鍵を置けばユーザーモードによる通信では、バグで鍵が読み取られるリスクが低減されます。

もし制限されたアドレスにアクセスした場合、AE ビットが 1 になり、割込み 1 番が発生します。
CLEAR 命令が発行されるまで AE は 1 を保持します。

SDog には UF を省略する verilog の parameter USER_MODE がある。UF に関係する AE bit や INTR1 割込みが省略されます。

「高速モード」の opmsb は B レジスタのインクリメントを OP コードの命令の実行と並行して同時にする。
B レジスタをインクリメントやデクリメントする OP コードの命令では B レジスタの操作をキャンセルする。
それ以外の B レジスタを操作する OP コードの命令では使用不可

JSIG121 オプション

このオプションが有効のときは外部からの信号(JSIG121)が 0 のとき JMP 命令や BAL 命令で 121(242 番地)があっても分岐しない。このオプションを装備した場合、必ず無効化できる設定を用意すること。

n : 8bit の数字(相対分岐では符号付)、m : 7bit の数字、k : 4bit の数字、xx:命令コードのオペランド 8bit

ニモニック	動作内容(★は未定義命令、コアによっては利用可)		OP		F
NOP	何もしない		00	00 0000	
LD A, B	01 h	A=B	01	00 0001	
LD B, A	02 h	B=A			
SWAP	03 h	A=B , B=A			
LD B, C	20 h	B=C			
GETPC	21 h	AB=PC			
GETMC	22 h	AB=MC			
SETPC	24 h	A,B の値を PC にセットする。			
SETMC	28 h	A,B の値を MC にセットする。			
USERMODE k	3k h	ユーザーモードを k にする。k=0、1 のみ			
OFFSET k	4k h	オフセットを k にする。 k=0、1、2、 3 のみ			
GETSTAT	51 h	A={EF,0,0,0,0,0,TM,AE} 主に割込み要因の取得			
GETVER	52 h	**C=(バージョン番号)(試作未実装)**			
GETID	54 h	**C=(識別 ID)(試作未実装)**			
CPUINFO	58 h	**C=(機能情報)(試作未実装)**			
CLEAR k	6k h	k で示される割込み要因のクリア(k=1 は AE)			
(LOAD)/(SAVE)	C1/C2	(オプション)			
CLC	C4 h	キャリフラグのクリア CF=0			
PMEMACC	C8 h	(オプション)次の 1 命令だけﾌﾟﾛｸﾞﾗﾑﾒﾓﾘにアクセス			
INT k	Dk h	ソフトウェア割込 0≦k≦15			
DISABLE/ENABLE	E0/E1	割込み禁止/割込み可能			
INPUTA	F1 h	ポート ID C から A=IN			
OUTPUTB	F2 h	ポート ID C に B を出力			
INOUT	F3 h	ポート ID C に B を出力、A=IN			
LD A, n	A=n (n≦255) 即値代入		02	00 0010	
LD B, n	B=n (n≦255) 即値代入		03	00 0011	
LD C, n	C=n (n≦255) 即値代入		04	00 0100	
SETFLAG [m]	{flag register}=[m] (m≦126、xx≦126)		05	00 0101	○
SETFLAG [A:B]	{flag register}=[A:B](xx=127)				○
SETFLAG [m:B]	{flag register}=[m:B] (m≦127、128≦xx≦255)				○
ST [m],A	[m]=A(m≦126、xx≦126)		06	00 0110	-
ROOPSHLC(未定)	*LOOPDEC 0 と SHLC [A:B]の複合命令(n=127)割込み OK*				CF
ST [m:B],A	[m:B]=A (m≦127、128≦xx≦255)				-
ST [m],B	[m]=B(m≦126、xx≦126)		07	00 0111	-
ROOPSHRC(未定)	*LOOPINC 0 と SHLR [A:B]の複合命令(n=127)割込み OK*				CF
ZERO [m]	[m]=0 (m≦127、128≦xx≦255)				-

SHL [m] SHL [A:B] SHL [m:B]	[m]={[m]<<1, 0}、CF=[m]>>7 (m≦126、xx≦126) [A:B]={[A:B]<<1, 0}、CF=[m]>>7 (m=127、xx=127) [m:B]={[m:B]<<1, 0}、CF=[m]>>7 (m≦127、128≦xx≦255)	08	00 1000	CF CF CF
SHR [m] SHR [A:B] SHR [m:B]	[m]={0,[m]>>1}、CF=[m]&1　(m≦126、xx≦126) [A:B]={0,[A:B]>>1}、CF=[m]&1　(m=127、xx=127) [m:B]={0,[m:B]>>1}、CF=[m]&1　(m≦127、128≦xx≦255)	09	00 1001	CF CF CF
AND [m],A **AND [A:B],C** AND [m:B],A	[m]=[m]&A (m≦126、xx≦126) [A:B]=[A:B]&C (xx=127) [m:B]=[m:B]&A (m≦127、128≦xx≦255)	0A	00 1010	ZF ZF ZF
ST [m],C ST [A:B],C ST [m:B],C	[m]=C (m≦126、xx≦126) [A:B]=C (xx=127) [m:B] = C (m≦127、128≦xx≦255)	0B	00 1011	
XOR [m],A **XOR [A:B],C** XOR [m:B],A	[m]=[m]^A (m≦126、xx≦126) [A:B]=[A:B]^A (xx=127) [m:B]=[m:B]^A (m≦127、128≦xx≦255)	0C	00 1100	ZF ZF ZF
GETFLAG [m] GETFLAG [A:B] GETFLAG [m:B]	[m] = {CF, 0, 0, OF[1:0], LF, MF, ZF} (m≦126、xx≦126) [A:B] ={CF, 0, 0, OF[1:0], LF, MF, ZF} (xx=127) [m:B]= {CF, 0, 0, OF, LF, MF, ZF}(m≦127、128≦xx≦255)	0D	00 1101	
OR [m],A **OR [A:B],C** OR [m:B],A	[m]=[m]|A (m≦126、xx≦126) [A:B]=[A:B]|C (xx=127) [m:B]=[m:B]|A (m≦127、128≦xx≦255)	0E	00 1110	ZF ZF ZF
(未使用)	(オペコード 0F は未使用)	0F	00 1111	
JMP m JMPC0 A:B JMPC1 A:B JMPZ0 A:B JMPZ1 A:B JMP A:B JMP m:B	m×2 番地へ分岐 **(m≦121、xx≦121)** CF=0 なら A:B×2 番地へ分岐(xx =122) CF=1 なら A:B×2 番地へ分岐(xx =123) ZF=0 なら A:B×2 番地へ分岐(xx =124) ZF=1 なら A:B×2 番地へ分岐(xx =125) A:B×2 番地へ分岐(xx=127) m:B×2 番地へ分岐(m≦127、128≦xx≦255)	10	01 0000	
BAL m BALC0 A:B BALC1 A:B BALZ0 A:B BALZ1 A:B BAL A:B BAL m:B	m×2 番地へ分岐、AB=PC/2 **(m≦121、xx≦121)** CF=0 なら A:B×2 番地へ分岐、AB=PC/2 (xx =122) CF=1 なら A:B×2 番地へ分岐、AB=PC/2 (xx =123) ZF=0 なら A:B×2 番地へ分岐、AB=PC/2 (xx =124) ZF=1 なら A:B×2 番地へ分岐、AB=PC/2 (xx =125) A:B×2 番地へ分岐(xx=127) 、AB=PC/2 m:B×2 番地へ分岐、AB=PC/2 (m≦127、128≦xx≦255) PC: 次の命令のアドレス、サブルーチンからの戻り値	11	01 0001	
JR n JR B	PC+n×2 へ分岐(-127≦n≦126、n=127 は使用不可) PC+B×2 へ分岐(n=-128、符号なしの値は 0x80)	12	01 0010	

BR n BR B	PC+n×2 へ分岐(-127≦n≦126、127 は不可) AB=PC/2 PC+B×2 へ分岐(n=-128) AB=PC/2	13	01 0011	
JRC0 n JRC0 B	CF=0 なら PC+n×2 へ分岐(-127≦n≦126、127 は使用不可) CF=0 なら PC+B×2 へ分岐(n=-128)	14	01 0100	
BRC0 n BRC0 B	CF=0 なら PC+n×2 へ分岐(-127≦n≦126、127 不)AB=PC/2 CF=0 なら PC+B×2 へ分岐(n=-128) AB=PC/2	15	01 0101	
JRC1 n JRC1 B	CF=1 なら PC+n×2 へ分岐(-127≦n≦126、127 は使用不可) CF=1 なら PC+B×2 へ分岐(n=-128)	16	01 0110	
BRC1 n BRC1 B	CF=1 なら PC+n×2 へ分岐(-127≦n≦126、127 不) AB=PC/2 CF=1 なら PC+B×2 へ分岐(n=-128) AB=PC/2	17	01 0111	
JRZ0 n JRZ0 B	ZF=0 なら PC+n×2 へ分岐(-127≦n≦126、127 は使用不可) ZF=0 なら PC+B×2 へ分岐(n=-128)	18	01 1000	
BRZ0 n BRZ0 B	ZF=0 なら PC+n×2 へ分岐(-127≦n≦126、127 不) AB=PC/2 ZF=0 なら PC+B×2 へ分岐(n=-128) AB=PC/2	19	01 1001	
JRZ1 n JRZ1 B	ZF=1 なら PC+n×2 へ分岐(-127≦n≦126、127 は使用不可) ZF=1 なら PC+B×2 へ分岐(n=-128)	1A	01 1010	
BRZ1 n BRZ1 B	ZF=1 なら PC+n×2 へ分岐(-127≦n≦126、127 不) AB=PC/2 ZF=1 なら PC+n×2 へ分岐(n=-128) AB=PC/2	1B	01 1011	
(未使用)	(オペコード 1C は未使用)	1C	01 1100	
IN [m] IN [A:B] IN [m:B]	ポート ID C から[m] = IN (m≦126、xx≦126) ポート ID C から[A:B] = IN (m=127、xx=127) ポート ID C から[m:B] = IN (m≦127、128≦xx≦255)	1D	01 1101	
OUT [m] OUT [A:B] OUT [m:B]	ポート ID C に[m]を出力(m≦126、xx≦126) ポート ID C に[m]を出力(m=127、xx=127) ポート ID C に[m:B]を出力 (m≦127、128≦xx≦255)	1E	01 1110	
RETI [m]	SETFLAG [m]を実行して割込みから復帰(m≦126、xx≦126)	1F	01 1111	
INC [m] INC [A:B] INC [m:B]	[m]=[m]+1 (m≦126、xx≦126) [A:B]=[A:B]+1 (xx=127) [m:B]=[m:B]+1 (m≦127、128≦xx≦255)	20	10 0000	○ ○ ○
INCC [m] INCC [A:B] INCC [m:B]	[m]=[m]+CF (m≦126、xx≦126) [A:B]=[A:B]+CF (xx=127) [m:B]=[m:B]+CF (m≦127、128≦xx≦255)	21	10 0001	○ ○ ○
INCX [m] INCX [A:B] INCX [m:B]	A=[m], B=A, [m]=[m]+1 (m≦126、xx≦126) A=[A:B], B=A , [A:B]=[A:B]+1 (x=127) A=[m:B], B=A , [m:B]=[m:B]+1 (m≦127、128≦xx≦255)	22	10 0010	○ ○ ○
INCXC [m] INCXC [A:B] INCXC [m:B]	A=[m], B=A, [m]=[m]+CF (m≦126、xx≦126) A=[A:B], B=A, [A:B]=[A:B]+CF(xx=127) A=[m:B], B=A, [m:B]=[m:B]+CF(m≦127、128≦xx≦255)	23	10 0011	○ ○ ○
ADD [m],A	[m]=[m]+A (m≦126、xx≦126)	24	10 0100	○

ADD [A:B],C	[A:B]=[A:B]+C (xx=127)			○
ADD [m:B],A	[m:B]=[m:B]+A (m≦127、128≦xx≦255)			○
ADDC [m],A	[m]=[m]+A+CF (m≦126、xx≦126)	25	10 0101	○
ADDC [A:B],C	[A:B]=[A:B]+C+CF (xx=127)			○
ADDC [m:B],A	[m:B]=[m:B]+A+CF (m≦127、128≦xx≦255)			○
ADD A,[m]	A=[m]+A (m≦126、xx≦126)	26	10 0110	○
ADD A,C	A=A+C(xx=127)			○
ADD A,[m:B]	A=[m:B]+A (m≦127、128≦xx≦255)			○
ADDC A,[m]	A=[m]+A+CF (m≦126、xx≦126)	27	10 0111	○
ADDC A,C	A=A+C+CF(xx=127)			○
ADDC A,[m:B]	A=[m:B]+A+CF (m≦127、128≦xx≦255)			○
SHLC [m]	[m]={[m]<<1, CF} 、CF=[m]>>7 (m≦126、xx≦126)	28	10 1000	CF
SHLC [A:B]	[A:B]={[A:B]<<1, CF}、CF=[A:B]>>7 (xx=127)			CF
SHLC [m:B]	[m:B]={[m:B]<<1, CF}、CF=[m:B]>>7 (m≦127、128≦xx≦255)			CF
SHRC [m]	[m]={CF,[m]>>1}、CF=[m]&1 (m≦126、xx≦126)	29	10 1001	CF
SHRC [A:B]	[A:B]={CF,[A:B]>>1}、CF=[A:B]&1 (xx≦127)			CF
SHRC [m:B]	[m:B]={CF,[m:B]>>1}、CF=[m:B]&1 (m≦127、128≦xx≦255)			CF
LD A,[m]	A=[m] (m≦126、xx≦126)	2A	10 1010	
LD A,[A:B]	A=[A:B] (xx=127)			
LD A,[m:B]	A=[m:B] (m≦127、128≦xx≦255)			
LD B,[m]	B=[m] (m≦126、xx≦126)	2B	10 1011	
LD B,[A:B]	B=[A:B] (xx=127)			
LD B,[m:B]	B=[m:B] (m≦127、128≦xx≦255)			
LD C,[m]	C=[m] (m≦126、xx≦126)	2C	10 1100	
LD C,[A:B]	C=[A:B] (xx=127)			
LD C,[m:B]	C=[xx:B] (m≦127、128≦xx≦255)			
STACP [m]	[A:C]=[m],B=B+1,C=C+1(m≦126、xx≦126)	2D	10 1101	
STACP [A:B]	[A:C]=[A:B],B=B+1,C=C+1(xx=127)			
STACP [m:B]	[A:C]=[m:B],B=B+1,C=C+1(m≦127、128≦xx≦255)			
STAB [m]	[A:B] = [m] (m≦126、xx≦126)	2E	10 1110	
STAC [A:B]	**[A:C]=[A:B] (xx=127)**			
STAB [m:B]	[A:B] = [m:B] (m≦127、128≦xx≦255)			
STABP [m]	[A:B] = [m],B=B+1 (m≦126、xx≦126)	2F	10 1111	
STZCP [A:B]	**[C]=[A:B],B=B+1,C=C+1(xx=127)**			
STABP [m:B]	[A:B] = [m:B],B=B+1(m≦127、128≦xx≦255)			
DEC [m]	[m]=[m]-1 (m≦126、xx≦126)	30	11 0000	○
DEC [A:B]	[A:B]=[A:B]-1 (xx=127)			○
DEC [m:B]	[m:B]=[m:B]-1 (m≦127、128≦xx≦255)			○
DECC [m]	[m]=[m]-CF (m≦126、xx≦126)	31	11 0001	○

DECC [A:B]	[A:B]=[A:B]-CF (xx=127)			○
DECC [m:B]	[m:B]=[m:B]-CF (m≦127、128≦xx≦255)			○
DECX [m]	A=[m], B=A, [m]=[m]-1 (m≦126、xx≦126)	32	11 0010	○
DECX [A:B]	A=[A:B], B=A, [A:B]=[A:B]-1(xx=127)			○
DECX [m:B]	A=[m:B], B=A, [m:B]=[m:B]-1(m≦127、128≦xx≦255)			○
DECXC [m]	A=[m], B=A, [m]=[m]-CF (m≦126、xx≦126)	33	11 0011	○
DECXC [A:B]	A=[A:B], B=A, [A:B]=[A:B]-CF(xx=127)			○
DECXC [m:B]	A=[m:B], B=A, [m:B]=[m:B]-CF(m≦127、128≦xx≦255)			○
SUB [m],A	[m]=[m]-A (m≦126、xx≦126)	34	11 0100	○
SUB [A:B],C	[A:B]=[A:B]-C(xx=127)			○
SUB [m:B],A	[m:B]=[m:B]-A (m≦127、128≦xx≦255)			○
SUBC [m],A	[m]=[m]-A-CF (m≦126、xx≦126)	35	11 0101	○
SUBC [A:B],C	[A:B]=[A:B]-C-CF (xx=127)			○
SUBC [m:B],A	[m:B]=[m:B]-A-CF (m≦127、128≦xx≦255)			○
SUB A,[m]	A=[m]-A　(m≦126、xx≦126)　　引き算の方向に注意	36	11 0110	○
SUB A,C	A=C-A (xx=127)			○
SUB A,[m:B]	A=[m:B]-A (m≦127、128≦xx≦255)			○
SUBC A,[m]	A=[m]-A-CF (m≦126、xx≦126)　引き算の方向に注意	37	11 0111	○
SUBC A,C	A=C-A-CF(xx=127)			○
SUBC A,[m:B]	A=[m:B]-A-CF(m≦127、128≦xx≦255)			○
LOOPDEC n	C≠0 なら JR n , C=C-1 , B=B-1 (-127≦n≦126)	38	11 1000	-
LOOPSHRC	LOOPDEC 0 と SHRC [A:B]の複合命令(n=127)割込み OK			CF
~~LOOPDEC B~~	C≠0 なら JR B , C=C-1 , B=B-1★　**LOOPDEC 0 は WAIT 用！**			-
LOOPINC n	C≠0 なら JR n, C=C-1 , B=B+1 (-127≦n≦126,n=0 除く)	39	11 1001	-
LOOPZERO	LOOPINC 0 と[A:B]=0 の複合命令(n=0)割込み OK			-
LOOPSHLC	LOOPINC 0 と SHLC [A:B]の複合命令(n=127)割込み OK			CF
~~LOOPINC B~~	C≠0 なら JR B, C=C-1 , B=B+1★			-
AND A,[m]	A=A&[m] (m≦126、xx≦126)	3A	11 1010	ZF
AND A,C	A=A&C			ZF
AND A,[m:B]	A=A&[m:B] (m≦127、128≦xx≦255)			ZF
AND A, n	A=A & n (8bit 即値)	3B	11 1011	ZF
XOR A,[m]	A=A^[m] (m≦126、xx≦126)	3C	11 1100	ZF
XOR A,C	A=A^C(xx=127)			ZF
XOR A,[m:B]	A=A^[m:B] (m≦127、128≦xx≦255)			ZF
XOR A, n	A=A ^ n (8bit 即値)	3D	11 1101	ZF
OR A,[m]	A=A|[m] (m≦126、xx≦126)	3E	11 1110	ZF
OR A,C	A=A|C (xx=127)			ZF
OR A,[m:B]	A=A|[m:B] (m≦127、128≦xx≦255)			
OR A, n	A=A｜n (8bit 即値)	3F	11 1111	ZF

未使用オペコードは 0x0F、0x1C。それ以上、命令を拡張したい場合は macro ビットを修正する。命令コードの bit 14 が macro ビットだが、これを bit14、bit13 が 1 になったときに macro ビットが 1 になったのと同じ処理をする。32 命令のオペコードが利用可能になる。ただしハードマクロは 64 命令から 32 命令に減る。また macro ビットの仕様の都合で、拡張された 32 命令をハードマクロ命令の最後に使うことはできない。将来の拡張に備えるならハードマクロ命令は 32 命令までとしたほうがいい場合もある。

6 PMEMACC 命令(オプション)

次の 1 命令だけメモリモデル TINY と同等のアクセスになる。メモリモデルを HALF などにしてプログラムとデータに分けた場合、この命令を使うとプログラムメモリにアクセスできる。ユーザーモードではメモリへの書き込みは抑止される。割込み禁止状態で実行されなかった場合、最悪、プログラムメモリは破壊される。この命令は宇宙線などのソフトエラーになったメモリを修復するために使われます。
割込状態が不明の場合、次のようなコードが参考になります

```
GETSTAT
AND A,0x80
DISABLE
PMEMACC
LD A,[n]
JRZ1 1
ENABLE
```

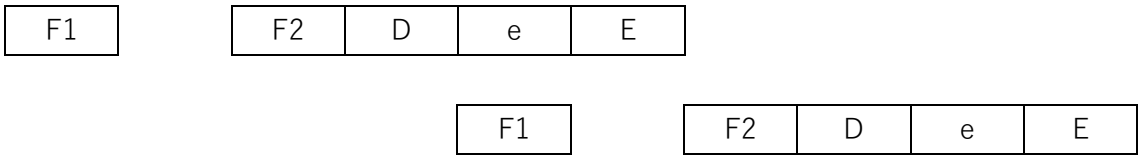
7 割込

外部割込は INT0、INT1 の 2 つ。INT0 優先でプログラムの 32 番地に分岐する。INT1 は 62 番地に分岐。INT 命令によってプログラムが割込みを起動できる。通常の割込みよりも優先順位が上で DISABLE の状態でも割込を起動できる。INT n の場合、プログラムの n ×2+32 番地に分岐する。n=0 は INT0 と同じアドレス。n=15 は INT1 と同じ。割込みによって ENABLE、DISABLE の状態は変化しません。割込み実行中、割込み信号は受け付けません。割込み中、INT 命令やハードマクロ命令は使えません。

８パイプライン

SDog ではオペランドを先に転送する逆順で 6 ステージ 4 サイクル。

6 ステージ 4 サイクル



分岐命令は e ステージで分岐先が確定しているので即座に次の命令の F1 をキャンセルできる。
つまりパイプライン処理というより逐次処理になるので少ないトランジスタ数で実装ができる。

参考までに正順の転送の 7 ステージ 3 サイクルとしてみた場合。

第 1 命令

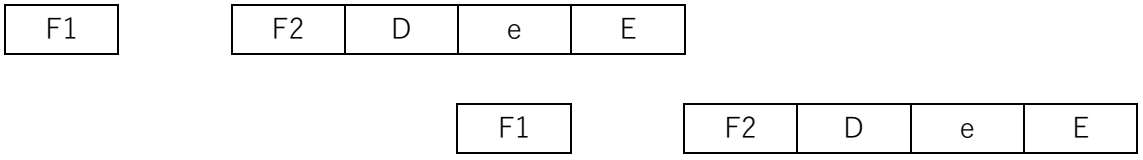
第 2 命令

第 3 命令

第 1 命令の E と第 3 命令の F1 がコンフリクトするためノイマン型アーキテクチャでは 3 サイクルピッチの実行は不可能。パイプラインをストールさせて 3-5 サイクルとすることも考えられるが PWM 制御用では使いにくくなる。

９相対ジャンプのアドレス計算

6 ステージ 4 サイクル



通常ケース アドレス

PC　　PC+1　　PC+2

相対分岐ケース　アドレス

PC　　PC+1　　PC+2+N

N は xx×2。最上位ビットは符号。符号拡張すること。

１０ハードマクロ命令

ハードマクロ命令は macro bit(bit14)が 1 である命令。複数の命令列にハード的に展開して処理をする命令です。命令列にはマイクロコードのような命令ではなく通常の命令コードを使います。命令列の中にハードマクロ命令が使えないこと以外は通常の命令コードと同じです。これを IBM はミリコードと呼んでいるようです。ハードマクロ命令のオペランドはメモリアドレス 0(オフセット付き)に書き込まれます。
またオペランドの値は C レジスタに書き込まれます。

ミリコードの命令列はプログラム領域の低位のアドレスに配置する。macro bit が 1 の命令は OP コード×16+512 番地(実装に依存)のアドレスに分岐します。再び macro bit が 1 の命令のを実行してハードマクロ命令を終了して、ハードマクロ命令の次の命令の実行を再開します。ハードマクロ命令の実装は通常の分岐と同じ e ステージで分岐処理をします。このとき MF を 1 にしてミリコード実行中、ずっと 1 を保持し続けます。ミリコードの終了命令では e ステージで MC から PC への切り替えをして MF を 0 にします。

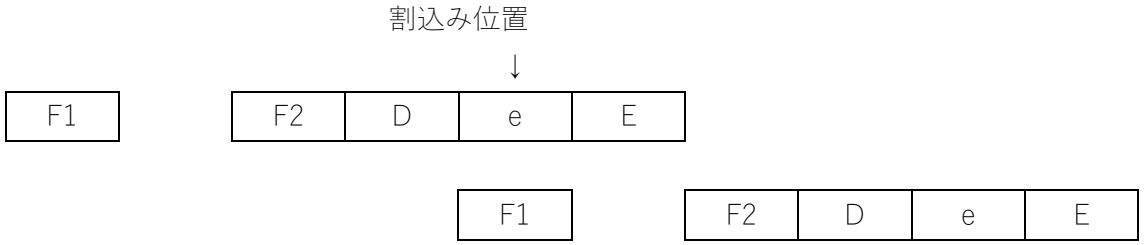
１１割込み実装

１１-１割込みプログラム

割込みプログラム中にハードマクロ命令は使用できません。

１１-２実装

一般命令の分岐のように処理する。



●割込み処理に分岐

IF を 1 にセット。カウンタを IC に変更。割込みルーチンの先頭でフラグの状態を退避。

●割込からの復帰

割込み先頭で保存していた A、B、C レジスタやフラグを復元して RETI 命令を実行。