luna0 Developper's Manual

for **luna0** **Rev. 1.1.2b**

DigiCat (Takashi Kohno)

Rev. 0.1.0 / 2003.10.20

目 次

1 Legal notice

2 Interface

2.1 HDL Interface

luna0 は VHDL で記述されており、IEEE ライブラリ、std_logic_1164、std_logic_arith パッケージを使用している。

```
entity luna0 is
    generic(    IAWidth : integer := 16 ;
                DAWidth : integer := 16
                ) ;
    port(   CLK, nRST   : in std_logic ;

            -- Instruction Bus
            InstAdrs    : out std_logic_vector(IAWidth - 1 downto 0) ;
            InstIn      : in std_logic_vector(7 downto 0) ;
            InstRd      : out std_logic ;  -- Inst. Mem read enable

            -- Data Bus
            DataAdrs    : out std_logic_vector(DAWidth - 1 downto 0) ;
            DataOut     : out std_logic_vector(7 downto 0) ;
            DataIn      : in std_logic_vector(7 downto 0) ;
            DataWr      : out std_logic ;
            DataRd      : out std_logic ;

            -- Control
            nWait       : in std_logic ;

            -- Interruption
            Int         : in std_logic
            ) ;
end luna0 ;
```

2.2 Pin Description

luna0 の入出力信号は以下の通り。

Instruction Bus

インストラクション・メモリ空間へのアクセス用バス。ALTERA M4K メモリの非同期データ出力モード ($LPM\_OUTDATA$ => "$UNREGISTERED$") に対応。

InstAdrs	インストラクション・メモリ空間のアドレス。IAWidth ビット。
InstIn	インストラクション入力。8 ビット。
InstRd	インストラクション・メモリのリード・イネーブル信号。InstAdrs に有効なアドレスが出力されているクロックでこの信号が '1' となる。

Data Bus

データ・メモリ空間と I/O 空間へのアクセス用バス。ALTERA M4K メモリの非同期データ出力モード ($LPM\_OUTDATA$ => "$UNREGISTERED$") に対応。

DataAdrs	データ・メモリ, I/O 空間のアドレス。DAWidth ビット。
DataOut	データ出力。8 ビット。
DataIn	データ入力。8 ビット。
DataWr	データ・メモリ, I/O のライト・イネーブル信号。DataOut に有効なデータが出力されているクロックで'1' となる。
DataRd	データ・メモリ, I/O のリード・イネーブル信号。DataAdrs に有効なアドレスが出力されているクロックで'1' となる。

Control

ウェイト、割り込み、リセット等

nWait	ウェイト信号。この信号が '0' の場合、クロックは無視される。クロック・イネーブル信号とも見ることができる。
Int	割り込み信号。'1' で割り込み。
CLK	クロック入力。
nRST	非同期リセット入力。3 クロック以上の間'0' とすることでアサートとなる。

3 Base Architecture

(図 1) に **luna0** のブロック図を示す。ハーバード・アーキテクチャの 8bit-CPU であり、RISC ライクな構造により CPI=1 を実現している。インストラクション・メモリ空間、データ・メモリ空間、共に最大 16bit アドレスをサポート。GENERIC 変数の IAWidth, DAWidth によりそれぞれインストラクション・メモリ空間、データ・メモリ空間のアドレス・ビット幅を指定できる。

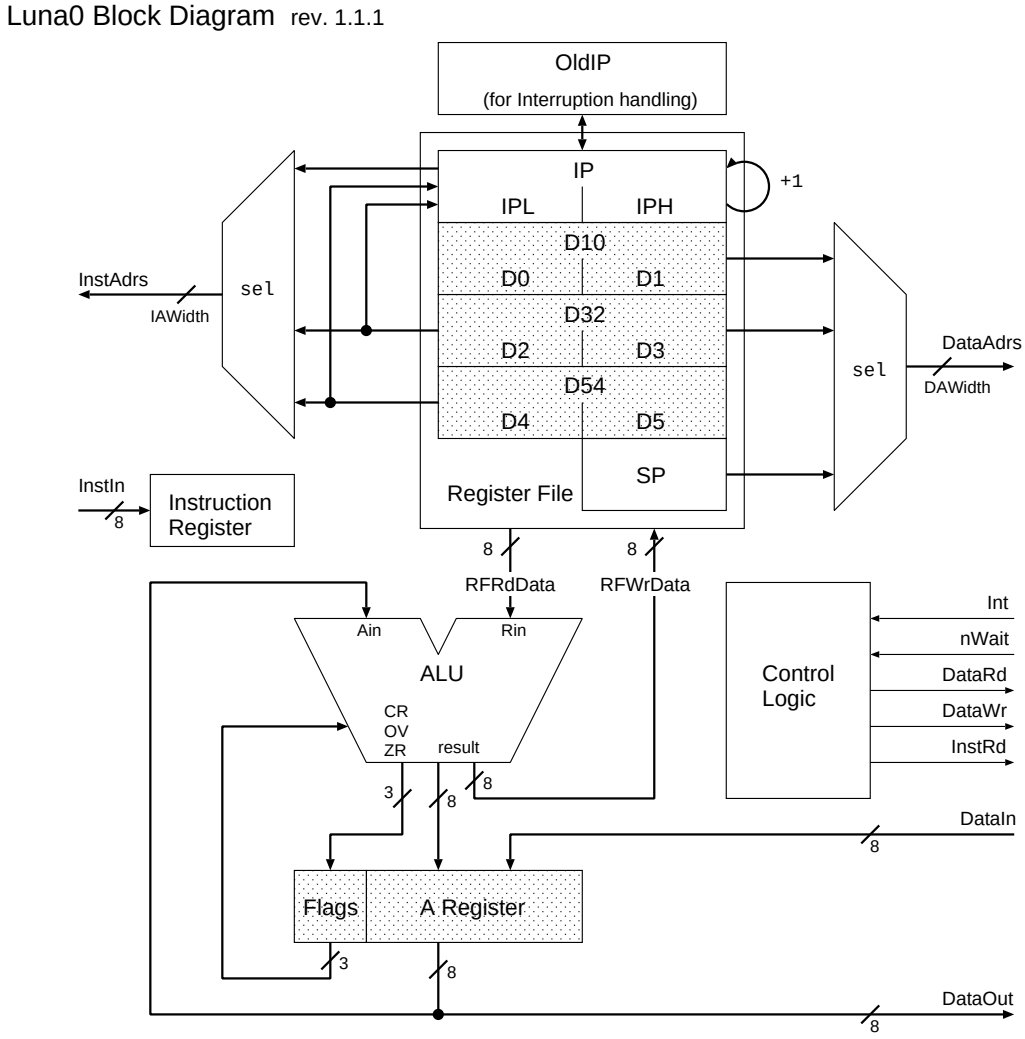


図 1: **luna0** のブロック図

3.1 Structural Description

- Register File

  レジスタファイルには 9 個の 8bit 幅レジスタが実装されている。IPL と IPH はそれぞれインストラクション・ポインタの下位 8bits と上位 8bits として機能する。D0〜D5 は汎用レジスタで、隣接レジスタと連結して 16bit 幅のレジスタとしてメモリ空間へのポインタとして機能することができる。これらの 16bit 幅レジスタは D10(D0 が下位 8bits、D1 が上位 8bits)、D32(D2 が下位 8bits、D3 が上位 8bits)、D54(D4 が下位 8bits、D5 が上位 8bits) であり、D10 はデータ・メモリ空間のポインタとして、

D32 はデータ・メモリ空間とインストラクション・メモリ空間のポインタとして、D54 はインストラクション・メモリ空間のポインタとして、利用できる。SP は 8bit 幅のスタック・ポインタであり、上位 8bits に”00000000b” を連結した 16bits の値がスタック参照アドレスとなる。すなわち、スタックはデータ・メモリの先頭 256bytes のみを用いることとなる。

- A Register
  A Register はアキュムレータとして動作する 8bit 幅のレジスタであり、データ・メモリへの入出力、ALU からの演算出力はこのレジスタに直結される。

- ALU
  加減算 (ADD, SUB, INC, DEC) と論理演算 (AND, OR, XOR, NOT, SHL, SHR) をサポート。キャリー (CR)、オーバーフロー (OV)、ゼロ (ZR) の 3 種類のフラグを生成する。

- OldIP
  OldIP は 16bit 幅の 2 個 1 組のレジスタであり、割り込み発生時の IP を保存する。iret 命令の実行時にこのレジスタの内容が IP にロードされる。

3.2 Functional Description

3.2.1 Pipelines

(図 2) に示すように、**luna0** のパイプラインはデコードと実行の 2 段で構成されている。

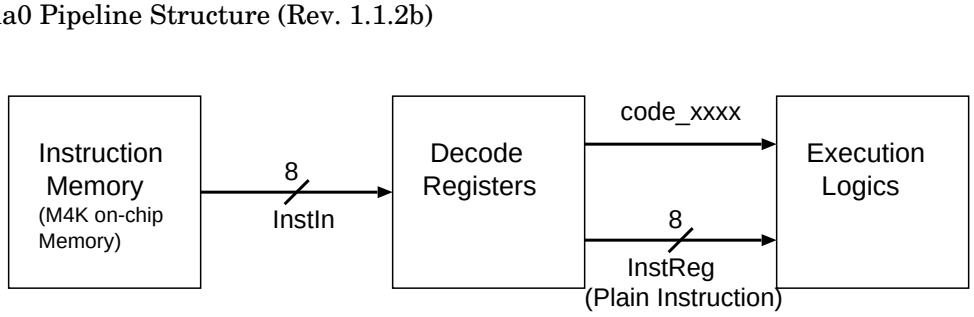


図 2: **luna0** のブロック図

パイプラインの動作例を (図 3) に示す。インストラクション・メモリへアドレスが提示された (InstAdrs) 次のクロックで命令コードが出力される (InstIn)。命令コードはデコードされて次のクロックでレジスタに格納され (InstReg, code_xxxx)、さらに次のクロックで実行される。

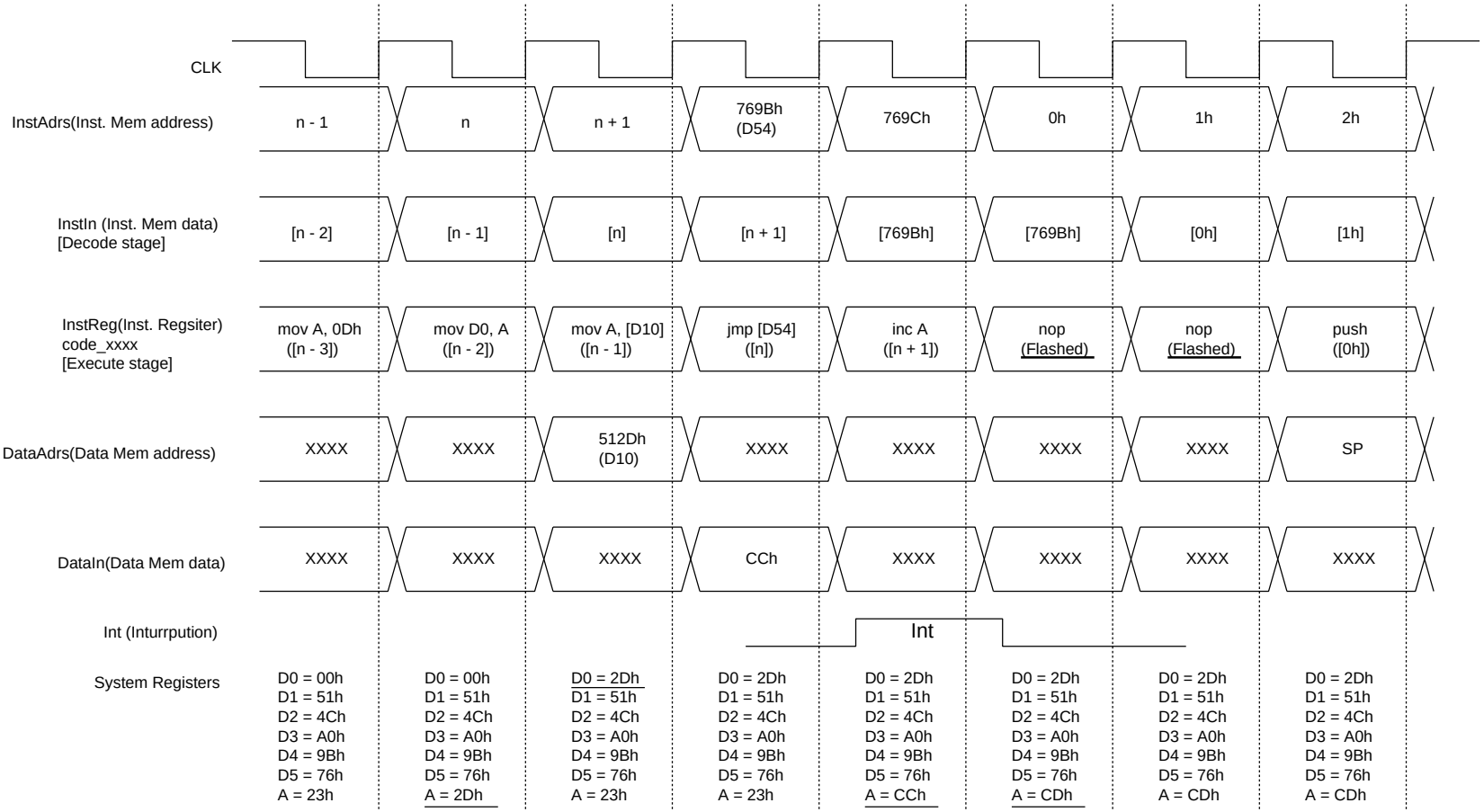
- 遅延分岐は、デコード時にジャンプ命令が検出された場合に、実行時に分岐先アドレスとして指定されたレジスタをインストラクション・メモリ・アドレス (InstAdrs) に出力することによって、1 命令におさえている。

- 割り込みが検出された場合、パイプラインをフラッシュするため 2CLK の間 nop 動作となる。つまり、割り込み検出後 3CLK 目で割り込みルーチン (インストラクション・メモリ空間の先頭に固定) の実行が開始される。

- nWait 信号がアサート ('0') されたクロックではパイプラインは停止する (図 4)。

図 3: luna0 のブロック図

An example of luna0 pipeline execution (Rev. 1.1.2b)



3.2.2 Bus Timing

メモリ・インターフェースの動作例を (図 4) に示す。メモリ・インターフェースは'M4K' メモリのデータ出力非同期モードに対応しており、有効アドレスの出力されているタイミングでイネーブル信号 (DataRd, InstRd, DataWr) がアサート ('1') される。リード動作では、イネーブル信号がアサートされたクロックの立ち上がりでメモリがアドレスを読み込み、データ出力を行うことが期待されている。ライト動作では、イネーブル信号がアサートされたクロックの立ち上がりでメモリがアドレスとデータを読み込むことが期待されている。

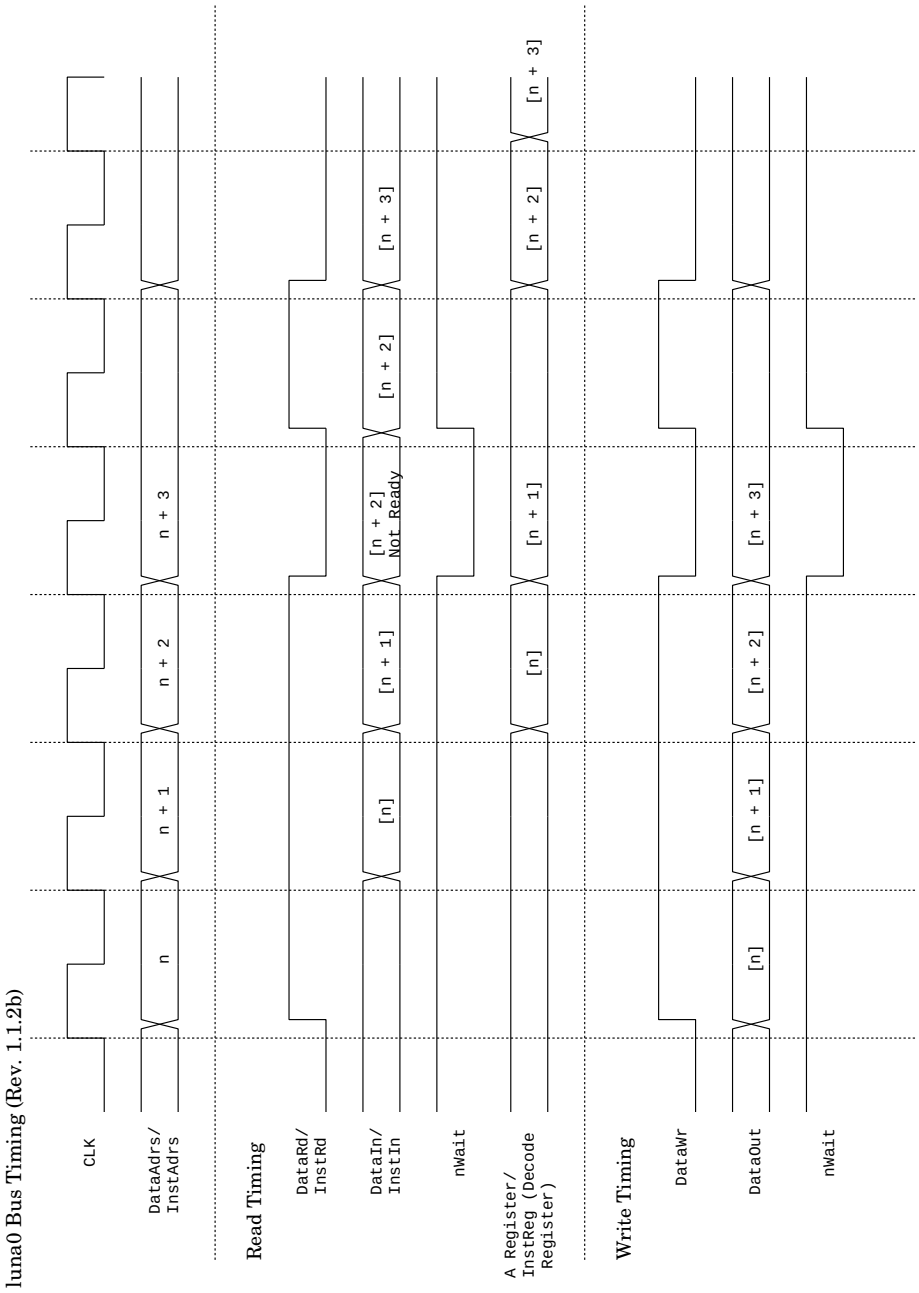


図 4: **luna0** のバス・タイミング

4 Software Acrchitecture

4.1 Programming Codes

マシン・コードとニーモニックを (表 4) に示す。表中、”_rrr” は 8bit 幅レジスタを指定するコードであり、ニーモニックの”reg” に対応する (表 1(a))。“_ww_” は 16bit 幅レジスタを指定するコードであり、ニーモニックの”wreg” に対応する (表 1(b))。“cc_c” は条件分岐用の条件コードであり、ニーモニックの”cond” に対応する (表 1(c))。nop は jc FALSE, [D54] と同義となっている。

Code	Nimonic	Code	Nimonic
000	IPL	001	IPH
010	D0	011	D1
100	D4	101	D5
110	D2	111	D3

(a) reg(レジスタコード) [_rrr]

Code	Nimonic
00	IP
01	D10
11	D32
10	D54

(b) wreg(ワイドレジスタコード) [_ww_]

Code	Nimonic	Code	Nimonic
00_0	**FALSE**	01_0	**TRUE**
00_1	ZR(ZeRo)	01_1	NZ(Not Zero)
10_0	CR(CaRry)	11_0	NC(Not Carry)
10_1	OF(OverFlow)	11_1	NO(Not Overflow)

(c) cond(コンディションコード) [cc_c]

表 1: reg(レジスタコード) [_rrr]

リセット後、インストラクション・ポインタ (IP) は 0040h にセットされ、このアドレスから実行が開始される。

4.2 Assembler

luna0 には専用アセンブラ (**lunasm**) が用意されており、前節のニーモニックによって記述されたプログラムをマシン・コードに変換することができる。**lunasm** 用命令 (ディレクティブ) として (表 2) に示す 3 つが定義されており、即値の並び、データ・メモリ空間への配置指定、インストラクション・メモリ空間への配置指定、が行える。即値表現 (表 3) については、2 進数、8 進数、10 進数、16 進数の他、文字コード、文字列、が使用可能となっている。16 進数表現において、先頭文字が A〜F となる場合は最初に 0 を付ける必要がある。また、ラベル・アドレス中の特定の 4bits や 8bits を指定することができる。

db inst, inst, ...	即値宣言
.data	データメモリへ切替
.inst	インストラクションメモリへ切替

表 2: **lunasm** ディレクティブ

表記	意味	例
xxh	16 進数	0A2h = 162, 2Fh = 47
xxo	8 進数	25o = 23, 77o = 63
xxb	2 進数	11b = 3, 10101b = 21
xxd(or xx)	10 進数	22d = 22
' 文字'	文字コード	'c' = 63h = 72
"文字列"	文字コードの並び	"luna" = 6ch, 75h, 6Eh, 61h = 108, 117, 110, 97
#n:(ラベル名)	ラベルの指すアドレスの 16 進表現での n 桁目 (n=0,1,2,3)	ラベル"START" が 028Fh を指すとき、#2:START = 2, #0:START = 0Fh
#s:(ラベル名)	ラベルの指すアドレスの上位/下位バイト (s=h,l)	#l:START = 8Fh, #h:START = 02h

表 3: **lunasm** における即値表現

4.3 Memory Latency

プログラミング上、メモリ・レイテンシが問題となるのはデータ・メモリ空間への読み書き、分岐、割り込み、の 3 つの場合である。

4.3.1 データ・メモリ空間からの読み出し

データ・メモリ空間からの読み出し命令は mov A, [wreg] と pop の 2 命令である。データ・メモリ空間から A Register への読み出しは、M4K メモリの構造上、1CLK 遅延する (図 4)。したがって、データ・メモリ空間からの読み出し命令の直後に A Register を参照する命令を配置した場合には**データ・メモリ空間からの読み出し内容が反映される前の A Register の内容が参照される**(ただし、割り込みが発生した場合にはこの限りではない (4.3.4 節参照))。下記の例では D2 レジスタには [02C4h] の内容ではなく 02h がロードされる (割り込みが発生しない場合) が、D3 レジスタには [02C4h] の内容がロードされる。

```
例:  mov   AL, 4h
     mov   AH, 0Ch
     mov   D0, A
     mov   AL, 2h
     mov   AH, 0h
     mov   D1, A
     mov   A, [D10]
     mov   D2, A
     mov   D3, A
```

4.3.2 データ・メモリ空間への書き込み

データ・メモリ空間への書き込み命令は mov [wreg], A と push の 2 命令である。M4K メモリのコンフィグレーションによっては書き込み動作を行ってから読み出し内容に結果が反映されるまでに 1CLK のレイテンシが生じる場合がある。したがって、データ・メモリ空間への書き込みを行った直後に同アドレスからの読み出しを行うことは I/O アクセス以外では避けるべきである。

種別 (Flag)	Nimonic	Machine Codes		動作
		opcode	opland	
転送命令	mov A, reg	0010	0rrr	A reg. に reg の内容をロード
	mov A, [wreg]	0010	1wwx	A reg. に wreg(*1) の指すメモリ内容をロード
	mov reg, A	0011	0rrr	reg に A reg. の内容をストア
	mov [wreg], A	0011	1wwx	Areg. の内容を wreg(*1) の指すメモリにストア
	mov AL, inst	0100	xxxx	A reg. の下位 4bits に即値 inst をロード
	mov AH, inst	0101	xxxx	A reg. の上位 4bits に即値 inst をロード
二項算術 (ZR,CR,OV)	add A, reg	1000	0rrr	A ← A + reg
	addv A, reg	1000	1rrr	A + reg の結果のフラグのみセット
	sub A, reg	1001	0rrr	A ← A - reg
	subv A, reg	1001	1rrr	A - reg の結果のフラグのみセット
二項論理 (ZR)	and A, reg	1010	0rrr	A ← A and reg
	andv A, reg	1010	1rrr	A and reg の結果のフラグのみセット
	or A, reg	1011	0rrr	A ← A or reg
	orv A, reg	1011	1rrr	A or reg の結果のフラグのみセット
	xor A, reg	0111	0rrr	A ← A xor reg
	xorv A, reg	0111	1rrr	A xor reg の結果のフラグのみセット
単項算術 (ZR,CR,OV)	inc A	1100	0000	A ← A + 1
	inc reg	1100	1rrr	reg ← reg + 1 (*2)
	dec A	1101	0000	A ← A - 1
	dec reg	1101	1rrr	reg ← reg - 1 (*2)
単項論理 (ZR,CR)	shr A	1110	0000	A ← A >> 1
	shr reg	1110	1rrr	reg ← reg >> 1 (*2)
	shl A	1111	0000	A ← A << 1
	shl reg	1111	1rrr	reg ← reg << 1 (*2)
単項論理 (ZR)	not A	0110	0000	A ← not A
	not reg	0110	1rrr	reg ← not reg (*2)
分岐	jmp wreg	0000	01w0	wreg(*3) の指すアドレスにジャンプ
	jc cond, wreg	0000	ccwc	cond が真の場合 wreg(*3) のアドレスにジャンプ
nop	nop	0000	0000	
スタック	pop	0001	0000	A reg. にスタックをポップ
	push	0001	0001	A reg. をスタックにプッシュ
	dcsp	0001	0010	スタック・ポインタをデクリメント
	icsp	0001	0011	スタック・ポインタをインクリメント
	ldsp	0001	0100	スタック・ポインタを A reg. にロード
	stsp	0001	0101	Areg. をスタック・ポインタにセット
フラグ	ldfl	0001	0110	フラグを Areg. にロード (*4)
	stfl	0001	0111	Areg. をフラグにセット (*4)
割り込み	dsi	0001	1000	割り込みを不可にする
	eni	0001	1001	割り込みを可能にする
	iret	0001	1111	割り込みが発生した場所に戻る

表 4: Machine codes for luna0 (Rev. 2.2)

(*1): D10 or D32, (*2): IPL, IPH に対する演算は nop 動作, (*3): D32 or D54, (*4): (Bit2=OF/Bit1=CR/Bit0=ZR)

4.3.3 分岐

luna0 では遅延分岐が発生するが、その命令数は 1 である (3.2.1 節参照)。従って、分岐命令 (jmp [wreg] 及び jc cond, [wreg]) の直後に配置された命令は必ず実行される。この特性を用いて、分岐命令を連続配置することによって任意のアドレスの命令を 1 命令のみ実行させることができる。

4.3.4 割り込み

eni 命令によって割り込みマスクが解除された場合、**luna0** は割り込み入力信号 Int が '1' になったことを検出すると割り込みをマスクし、実行中のアドレスを OldIP レジスタに待避した上でインストラクション・メモリ空間の先頭 (0000h 番地) にジャンプする。**luna0** 側の動作はこれだけなので、インストラクション・メモリ空間の先頭から、リセット後の実行開始アドレス 0040h までの間にレジスタ内容の待避/復元や割り込みルーチンへのジャンプを行うコードを記述しておく必要がある。また、復帰アドレスをメモリではなく単一レジスタに待避するため、多重割り込みには対応していない。

0000h 番地へのジャンプの際、パイプラインがフラッシュされるため 2CLK の間は nop 動作となる (3.2.1 節参照) ため、実行中のプログラムにとって割り込み発生は nop 命令が 2 つ挿入されることに相当する。従って、データ・メモリ空間からの読み出し命令の実行時に割り込みが発生した場合、続く命令が読み出し命令の 1CLK 後に実行されるとは限らない。よって、**割り込みが発生しうるプログラムにおいてはデータ・メモリ空間からの読み出し命令の直後に A Registe を読み書きする命令は配置してはならない**(割り込みが発生した場合と発生しなかった場合で結果が異なる)。4.3.1 節の例では、D2 レジスタには mov A, [d10] の実行時に割り込みが発生した場合は [02C4h] の内容が、それ以外の場合には 02h がロードされる。

4.4 Stack

スタック・ポインタ (SP) は 8bit 幅のレジスタであり、上位 8bits に 00h を結合したデータ・メモリ空間内のアドレスがスタックの先頭アドレス + 1 となる。つまり、データ・メモリ空間の先頭 256bytes がスタック・メモリとして使用される。

SP の初期値は 00h であり、データは上位アドレスに向かって積まれていく。push 命令は現 SP の指すアドレスにデータをストアした後、SP をインクリメントする。pop 命令は SP をデクリメントした後、SP の指すアドレスから A Register にデータをロードする。SP の操作においてオーバーフロー、アンダーフローはチェックされないため、スタックに積まれるデータが 256bytes を越えた場合はスタックの後尾が上書きされることになる。