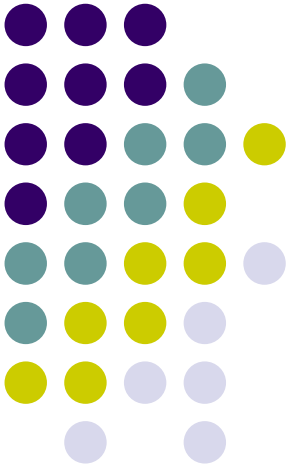
數位信號處理實務

Ch4. 指令集



大綱

4.1　數字表示與符號定義

4.2　指令總覽

大綱

4.1 數字表示與符號定義

4.2 指令總覽

4.1 數字表示與符號定義

- 數字表示有二進制、十進制以及十六進制等3種方式，以數字0、1、8及15為例，其中以0b開頭代表二進制；以0x開頭代表十六進制，英文字母大小寫不區分。

類型	數字表示方式			
十進制	0	1	8	15
二進制	0b0000	0b0001	0b1000	0b1111
十六進制	0x0	0x1	0x8	0xF

一般符號和指令內之符號說明(1/4)

符號	說明
Address Generator	資料記憶體(RAM)的位址產生器
ALU	16位元的算術邏輯運算單元
Multiplier	17X17 硬體乘法器
Program Counter	程式計數器(15位元可定址32K的程式記憶體)
Peripheral Control Registers	週邊控制暫存器，用於控制週邊的指令，例如ADC/DAC/INT等
RAM	內部資料記憶體(2K，0x0000 ~ 0x07FFF)
SR	狀態暫存器(Status register, SR)，內含carry/zero/overflow等狀態
Stack	16 位元軟體位址堆疊，用於副程式呼叫與中斷
W/C	Word/Cycle(指令長度/執行週期)
+, -, *, /	加減乘除四則運算
[ ]	指向資料記憶體(RAM)的位址

一般符號和指令內之符號說明(2/4)

運算元(Operand)	
符號	說明
R0~R7	一般功能暫存器
Rd	目的地暫存器(R0 ~ R7)
Rs	來源暫存器(R0 ~ R7)
Rt	第2來源暫存器(R0 ~ R7)
Rn	計算重覆或迴圈數目之暫存器(R0 ~ R7)
.b	指定位元，b的範圍為0~15
#imm6	直接定義6位元之資料，範圍為0 ~ 63
#imm8	直接定義8位元之資料，範圍為0 ~ 255
#imm16	直接定義16位元之資料，範圍為0 ~ 65535
RAM16	直接定址RAM之16位元位址
RAM8	直接定址RAM之8位元位址
B	借位，進位的反向 /C
P[ ]	指向程式(ROM)記憶體
IO[ ]	指向I/O space
<abs ADDR>	絕對位址(16位元)

- 一般符號和指令內之符號說明(3/4)

旗標(SR內狀態)	
符號	說明
T	測試旗標(test flag)
N	負旗標(negative flag)
Z	零旗標(zero flag)
V	溢位旗標(overflow flag)
C	進位旗標(carry flag)
+	旗標受指令執行影響
-	旗標不受指令執行影響(旗標不變)
*	指令執行後旗標未定義

一般符號和指令內之符號說明(4/4)

指令說明	
符號	說明
$addr16	直接定址之16位元RAM位址數值
$addr8	直接定址之8位元RAM位址數值
Long_addr	long jump之16位元絕對位址
Short_addr	short jump 16位元位址之偏移數值(PC + 1 + #offset)
L_addr	long call1之16位元絕對位址
S_addr	short call之14位元絕對位址
+, –, *, /	加減乘除四則運算(也是運算子)
&, |, ^, ~	AND、OR、XOR以及1的補數等邏輯運算(也是運算子)
==	相等

4.2 指令總覽

功能分類	指令之功能描述簡稱
資料搬移指令	MOV, IN, OUT, PUSH, POP, SP
算術運算指令	ADD, ADC, SUB, SUBB, INC, DEC
邏輯運算指令	AND, OR, XOR, COM, NEG, SWAP, CMP, CLR
移位指令	SHL, SHR, ROL, ROR, ASR
位元運算指令	BS, BC, BTEST, BTG IO
分支指令	JCC, JCS, JLS, JGE, (S)JMP, (L)JMP…等
控制指令	(S)CALL, (L)CALL, RET, RETI, RPT, LOOP, TRAP, NOP
DSP指令	MUL.UU, MUL.US, MUL.SU, MUL.SS, MAC, MAS, DIV, DIVS

4.2.1 資料搬移指令 (1/4)

- 資料搬移指令包括MOV、IN、OUT、PUSH以及POP等，主要功能是將資料從來源端搬移到目的地端，MOV相關指令將資料從記憶體搬移至暫存器稱為下載(LOAD)資料；而從暫存器搬移至記憶體稱為儲存(STORE)資料。IN和OUT指令可經由I/O搬移資料。PUSH和POP指令可從暫存器與堆疊(或I/O與堆疊)之間搬移資料。

4.2.1 資料搬移指令(2/4)

MOV指令

指令格式	執行功能	W/C
Rd = Rs	Rs→Rd	1/1
Rd = [Rs]	[Rs]→Rd	1/1
Rd = [Rs++]	[Rs++]→Rd	1/1
Rd = [Rs--]	[Rs--]→Rd	1/1
Rd = P[Rs]	P[Rs]→Rd	1/2(1)
Rd = P[Rs++]	P[Rs++]→Rd	1/2(1)
[Rd++] = P[Rs--]	P[Rs--]→[Rd++]	1/2(1)
[Rd++] = P[Rs++]	P[Rs++]→[Rd++]	1/2(1)
Rd = P[Rs--]	P[Rs--]→Rd	1/2(1)
[Rd] = P[Rs++]	P[Rs++]→[Rd]	1/2(1)
[Rd--] = P[Rs--]	P[Rs--]→[Rd--]	1/2(1)

4.2.1 資料搬移指令(3/4)

MOV指令

指令格式	執行功能	W/C
[Rd] = P[Rs]	P[Rs]→[Rd]	1/2(1)
[Rd++] = P[Rs]	P[Rs]→[Rd++]	1/2(1)
[Rd] = Rs	Rs→[Rd]	1/1
[Rd++] = Rs	Rs→ [Rd++]	1/1
[Rd--] = Rs	Rs→[Rd--]	1/1
Rd = RAM16	$addr16 → Rd	2/2
RAM16 = Rs	Rs→ $addr16	2/2
[Rd] = #imm16	#imm16 → [Rd]	2/2
Rd.l = #imm8	#imm8→Rd.l; 0→Rd.h	1/1
Rd.h = #imm8	#imm8→Rd.h	1/1

4.2.1 資料搬移指令(4/4)

IN/OUT指令

指令格式	執行功能	W/C
Rd = IO[Addr]	IO[Addr] → Rd	1/1
IO[Addr] = Rs	Rs → IO[Addr]	1/1

PUSH/POP/SP指令

指令格式	執行功能	W/C
PUSH Rn	1. Rn → TOS　2. SP−1 → SP	1/1
PUSH IO[Addr]	1. I/O → TOS　2. SP−1 → SP	1/1
POP Rn	1. SP+1 → SP　2. TOS → Rn	1/1
POP IO[Addr]	1. SP+1 → SP　2.TOS → IO[Addr]	1/1
SP=SP + #imm6	SP + #imm6 → SP	1/1
SP=SP – #imm6	SP – #imm6 → SP	1/1

4.2.2 算數運算指令(1/4)

ADD指令

指令格式	執行功能	W/C
Rd = Rs+Rt	Rs+Rt → Rd	1/1
[Rd] = Rs+Rt	Rs+Rt → [Rd]	1/1
Rd = [Rs]+Rt	[Rs]+Rt → Rd	1/1
Rd = Rd+#imm6	Rd+#imm6 → Rd	1/1
Rd = Rs+#imm16	Rs+#imm16 → Rd	2/2
[Rd] = Rs+#imm16	Rs+#imm16→ [Rd]	2/2
Rd = Rs+RAM16	Rs+ $addr16 → Rd	2/2
Rd++	Rd + 1 → Rd	1/1
Rd - -	Rd – 1 → Rd	1/1

4.2.2 算數運算指令(2/4)

ADC指令

指令格式	執行功能	W/C
Rd = Rs+Rt+C	Rs+Rt+C → Rd	1/1
[Rd] = Rs+Rt+C	Rs+Rt+C → [Rd]	1/1
Rd = [Rs]+Rt+C	[Rs]+Rt+C → Rd	1/1
Rd = Rd+#imm6+C	Rd+#imm6+C → Rd	1/1
Rd = Rs+#imm16+C	Rs+#imm16+C → Rd	2/2
[Rd] = Rs+#imm16+C	Rs+#imm16+C→ [Rd]	2/2
Rd = Rs+RAM16+C	Rs+ $addr16+C → Rd	2/2

4.2.2 算數運算指令(3/4)

SUB指令

指令格式	執行功能	W/C
Rd = Rs–Rt	Rs–Rt → Rd	1/1
[Rd] = Rs–Rt	Rs–Rt → [Rd]	1/1
Rd = [Rs]–Rt	[Rs]–Rt → Rd	1/1
Rd = Rd–#imm6	Rd–#imm6 → Rd	1/1
Rd = Rs–#imm16	Rs–#imm16 → Rd	2/2
[Rd] = Rs–#imm16	Rs–#imm16 → [Rd]	2/2
Rd = Rs–RAM16	Rs–$addr16 → Rd	2/2

4.2.2 算數運算指令(4/4)

SUBB指令

指令格式	執行功能	W/C
Rd=Rs–Rt–B	Rs–Rt–/C → Rd	1/1
[Rd]=Rs–Rt–B	Rs–Rt–/C → [Rd]	1/1
Rd=[Rs]–Rt–B	[Rs]–Rt–/C → Rd	1/1
Rd=Rd–#imm6-B	Rd–#imm6-/C → Rd	1/1
Rd=Rs–#imm16-B	Rs–#imm16-/C → Rd	2/2
[Rd]=Rs–#imm16–B	Rs–#imm16-/C → [Rd]	2/2
Rd=Rs–RAM16–B	Rs– $addr16-/C → Rd	2/2

4.2.3 邏輯運算指令器(1/5)

AND指令

指令格式	執行功能	W/C
Rd=Rs AND Rt	Rs&Rt→ Rd	1/1
[Rd]=Rs AND Rt	Rs&Rt→[Rd]	1/1
Rd=[Rs] AND Rt	[Rs]&Rt → Rd	1/1
Rd=Rd AND #imm6	Rd&#imm6 → Rd	1/1
Rd=Rs AND #imm16	Rs&#imm16 → Rd	2/2
[Rd]=Rs AND #imm16	Rs&#imm16→ [Rd]	2/2
Rd=Rs AND RAM16	Rs&$addr16→ Rd	2/2

4.2.3 邏輯運算指令器(2/5)

OR指令

指令格式	執行功能	W/C
Rd=Rs OR Rt	Rs | Rt → Rd	1/1
[Rd]=Rs OR Rt	Rs | Rt → [Rd]	1/1
Rd=[Rs] OR Rt	[Rs] | Rt → Rd	1/1
Rd=Rd OR #imm6	Rd | #imm6 → Rd	1/1
Rd=Rs OR #imm16	Rs | #imm16 → Rd	1/2(1)
[Rd]=Rs OR #imm16	Rs | #imm16 → [Rd]	1/2(1)
Rd=Rs OR RAM16	Rs | $addr16 → Rd	1/2(1)

4.2.3 邏輯運算指令器(3/5)

XOR指令

指令格式	執行功能	W/C
Rd=Rs XOR Rt	Rs^Rt → Rd	1/1
[Rd]=Rs XOR Rt	Rs^Rt → [Rd]	1/1
Rd=[Rs] XOR Rt	[Rs]^Rt → Rd	1/1
Rd=Rd XOR #imm6	Rd^#imm6 → Rd	1/1
Rd=Rs XOR #imm16	Rs^#imm16 → Rd	2/2
[Rd]=Rs XOR #imm16	Rs^#imm16→ [Rd]	2/2
Rd=Rs XOR RAM16	Rs^ $addr16 → Rd	2/2

4.2.3 邏輯運算指令器(4/5)

COM指令

指令格式	執行功能	W/C
Rd = COM Rs	~Rs → Rd	1/1

NEG指令

指令格式	執行功能	W/C
Rd=NEG Rs	~Rs+1 → Rd	1/1

SWAP指令

指令格式	執行功能	W/C
SWAP Rs	Rs[15:8]↔Rs[7:0]	1/1

4.2.3 邏輯運算指令器(5/5)

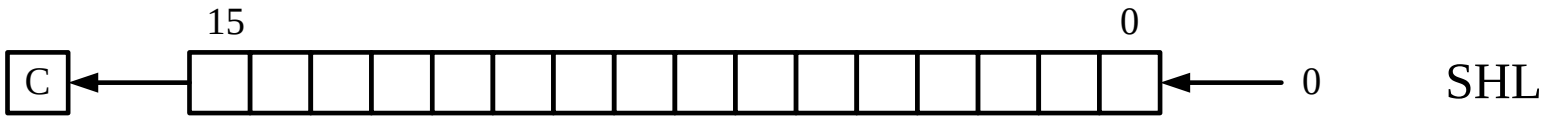
CMP指令

指令格式	執行功能	W/C
CMP Rs, Rt	Rs–Rt, update SR register	1/1
CMP Rs, [Rt]	Rs–[Rt], update SR register	1/1
CMP Rs, [Rt++]	Rs–[Rt++],update SR register	1/1
CMP Rs, [Rt--]	Rs–[Rt--],update SR register	1/1
CMP Rs, P[Rt]	Rs–P[Rt] ,update SR register	1/2(1)
CMP Rs, P[Rt++]	Rs–P[Rt++],update SR register	1/2(1)
CMP[Rs++], P[Rt--]	[Rs++]–P[Rt--],update SR register	1/2(1)
CMP [Rs++], P[Rt++]	[Rs++]–P[Rt++],update SR register	1/2(1)

4.2.4 移位指令(1/3)

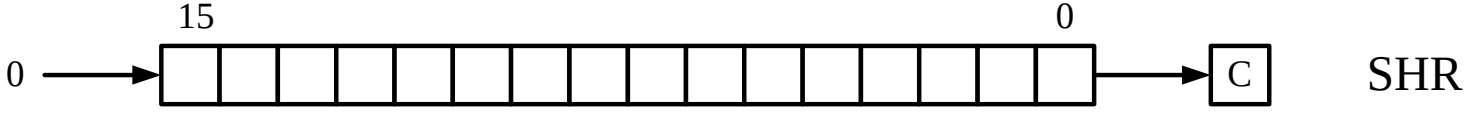
SHL指令

指令格式	執行功能	W/C
Rd=SHL Rs	1. Rs→ Rd 2. Rd[15] → C，Rd[14:0] → Rd[15:1]，0 → Rd[0]	1/1



SHR指令

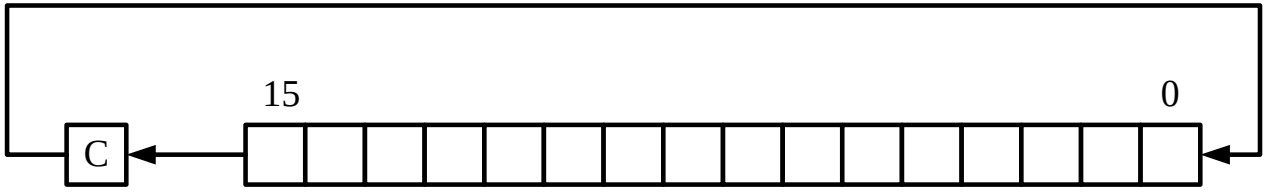
指令格式	執行功能	W/C
Rd=SHR Rs	1. Rs → Rd 2. Rd[0] → C， Rd[15:1] → Rd[14:0]， 0 → Rd[15]	1/1



4.2.4 移位指令(2/3)

ROL指令

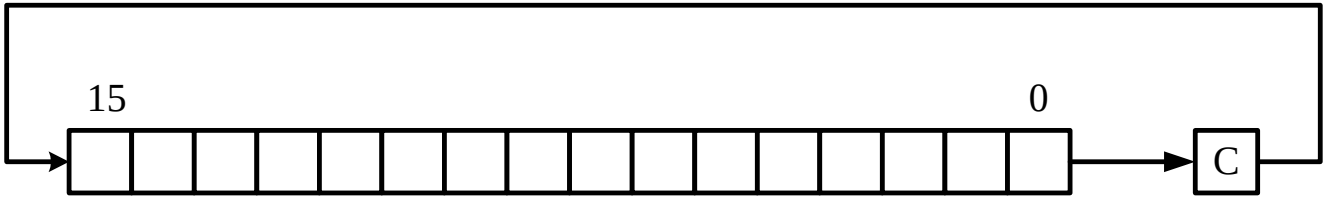
指令格式	執行功能	W/C
Rd = ROL Rs	1. Rs → Rd 2. C → Rd[0]，Rd[14:0] → Rd[15:1]，Rd[15] → C	1/1



ROL

ROR指令

指令格式	執行功能	W/C
Rd = ROR Rs	1. Rs → Rd 2. C → Rd[15]，Rd[15:1] → Rd[14:0]，Rd[0] → C	1/1

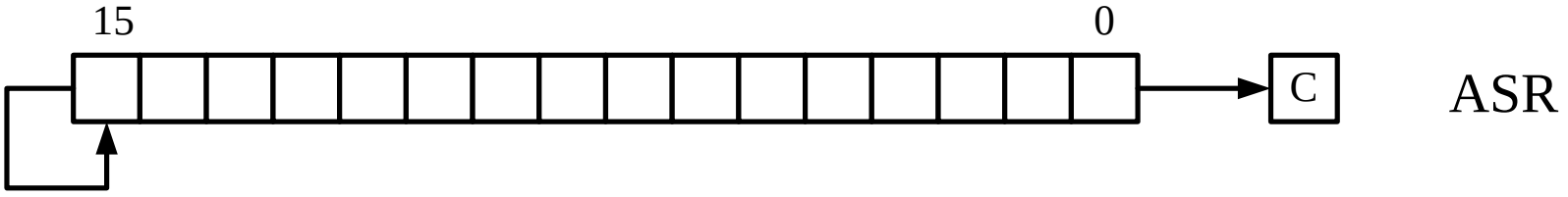


ROR

4.2.4 移位指令(3/3)

ASR指令

指令格式	執行功能	W/C
Rd=ASR Rs	1. Rs → Rd 2. Rd[0] → C，Rd[15:1] → Rd[14:0]，Rd[15] → Rd[15]	1/1



4.2.5 位元(Bit)運算指令(1/2)

BS指令

指令格式	執行功能	W/C
BS Rd.b	1→Rd[bit b]	1/1
BS IO[IO_addr].b	1→IO_addr[bit b]	1/1
BS RAM_addr.b	1→RAM_addr[bit b]	1/1

BC指令

指令格式	執行功能	W/C
BC Rd.b	0→Rd[bit b]	1/1
BC IO[IO_addr].b	0→IO_addr[bit b]	1/1
BC RAM_addr.b	0→RAM_addr[bit b]	1/1

4.2.5 位元(Bit)運算指令(2/2)

BTEST指令

指令格式	執行功能	W/C
BTEST Rd.b	Test Rd.b → Test Flag	1/1
BTEST IO[IO_addr].b	Test IO_addr.b → Test Flag	1/1
BTEST RAM_addr.b	Test RAM_addr.b → Test Flag	1/1

BTG指令

指令格式	執行功能	W/C
BTG Rd.b	~ Rd.b → Rd.b	1/1
BTG IO[IO_addr].b	~IO_addr.b→ IO_addr.b	1/1
BTG RAM_addr.b	~RAM_addr.b → RAM_addr.b	1/1

4.2.6 分支指令 (1/6)

JMP指令

指令格式	執行功能	W/C
JMP Rs	Rs→ PC	1/2
JMP Long_addr	Long_addr→PC	2/2
JMP Short_addr	PC+1+Offset→PC	1/1(2)

JCC指令

指令格式	執行功能	W/C
IF CC JMP Long_addr	If C = = 0, Long_addr→PC	2/2
IF CC JMP Short_addr	If C = = 0, PC+1+Offest→PC	1/1(2)

4.2.6 分支指令(2/6)

JCS指令

指令格式	執行功能	W/C
IF CS JMP Long_addr	If C = = 1, Long_addr → PC	2/2
IF CS JMP Short_addr	If C = = 1, PC+1+Offset → PC	1/1(2)

JVC指令

指令格式	執行功能	W/C
IF VC JMP Long_addr	If V== 0, Long_addr→PC	2/2
IF VC JMP Short_addr	If V== 0, PC+1+Offset→PC	1/1(2)

JVS指令

指令格式	執行功能	W/C
IF VS JMP Long_addr	If V = = 1, Long_addr→PC	2/2
IF VS JMP Short_addr	If V = = 1, PC+1+Offset→PC	1/1(2)

4.2.6 分支指令(3/6)

JMI指令

指令格式	執行功能	W/C
IF MI JMP Long_addr	If N = = 1,Long_addr→PC	2/2
IF MI JMP Short_addr	If N = = 1,PC+1+Offset→PC	1/1(2)

JPL指令

指令格式	執行功能	W/C
IF PL JMP Long_addr	If N= =0, Long_addr→PC	2/2
IF PL JMP Short_addr	If N = = 0,PC+1+Offset→PC	1/1(2)

JTC指令

指令格式	執行功能	W/C
IF TC JMP Long_addr	If T= =0, Long_addr→PC	2/2
IF TC JMP Short_addr	If T = = 0,PC+1+Offset→PC	1/1(2)

4.2.6 分支指令(4/6)

JTS指令

指令格式	執行功能	W/C
IF TS JMP Long_addr	If T= =1, Long_addr→PC	2/2
IF TS JMP Short_addr	If T = = 1,PC+1+Offset→PC	1/1(2)

JGT指令

指令格式	執行功能	W/C
IF GT JMP Long_addr	If [Z | (N^V) == 0],Long_addr→PC	2/2
IF GT JMP Short_addr	If [Z | (N^V) == 0],PC+1+Offset→PC	1/1(2)

JGE指令

指令格式	執行功能	W/C
IF GE JMP Long_addr	If [(N ^ V) = = 0],Long_addr→PC	2/2
IF GE JMP Short_addr	If [(N ^ V) = = 0],PC+1+Offset→PC	1/1(2)

4.2.6 分支指令(5/6)

JHS指令

指令格式	執行功能	W/C
IF HS JMP Long_addr	If C = = 1, Long_addr → PC	2/2
IF HS JMP Short_addr	If C = = 1, PC+1+Offset → PC	1/1(2)

JEQ指令

指令格式	執行功能	W/C
IF EQ JMP Long_addr	If Z = = 1, Long_addr → PC	2/2
IF EQ JMP Short_addr	If Z = = 1, PC+1+Offset→PC	1/1(2)

JNE指令

指令格式	執行功能	W/C
IF NE JMP Long_addr	If Z = = 0, Long_addr → PC	2/2
IF NE JMP Short_addr	If Z = = 0, PC+1+Offset→PC	1/1(2)

4.2.6 分支指令(6/6)

JLE指令

指令格式	執行功能	W/C
IF LE JMP Long_addr	If [Z | (N^V) = = 1],Long_addr→PC	2/2
IF LE JMP Short_addr	If [Z | (N^V) = = 1],PC+1+Offset→PC	1/1(2)

JLO指令

指令格式	執行功能	W/C
IF LO JMP Long_addr	If C = = 0, Long_addr→PC	2/2
IF LO JMP Short_addr	If C = = 0, PC+1+Offest→PC	1/1(2)

JLS指令

指令格式	執行功能	W/C
IF LS JMP Long_addr	If [(C = = 0) | (Z = = 1)],Long_addr→PC	2/2
IF LS JMP Short_addr	If [(C = = 0) | (Z = = 1)],PC+1+Offset→PC	1/1(2)

4.2.7 控制指令(1/3)

CALL指令

指令格式	執行功能	W/C
CALL Rd	1. SP−1 → SP　　2. PC+1 → TOS 3. Rd→ PC	1/1
CALL L_addr	1. SP −1 → SP　　2. PC+1 → TOS 3. 16 bit address→ PC	2/2
CALL S_addr	1. SP −1 → SP　　2. PC+1 → TOS 3. {00,14 bit address} → PC	1/1

RET/RETI指令

指令格式	執行功能	W/C
RET	1. Top of stack → PC 2. SP+1→SP	1/2
RETI	1. Top of stack→ PC 2. SP+1→SP　　3. 0→ GIE	1/2

4.2.7 控制指令(2/3)

RPT指令

指令格式	執行功能	W/C
RPT Rn	Rn→RC	1/1
RPT #imm6	#imm6→RC	1/1

LOOP指令

指令格式	執行功能	W/C
LOOP Rn .ENDL	1. Rn →LC 2. PC + 1→LSA 3. End_loop→LEA	1/1
LOOP #imm6 .ENDL	1. #imm6 →LC 2. PC + 1→LSA 3. End_loop→LEA	1/1

4.2.7 控制指令(3/3)

TRAP指令

指令格式	執行功能	W/C
TRAP #imm6	1. PC+1→TOS 2. SP−1→SP 3. (#imm6 vector number*2) → PC 4. GIE→0	1/1

NOP指令

指令格式	執行功能	W/C
NOP	PC+1→ PC	1/1

4.2.8 DSP指令

MUL.SS指令

指令格式	執行功能	W/C
D=Rs*Rt(SS)	Rs.S*Rt.S→D	1/1
D=Rs*[Rt] (SS)	Rs.S*[Rt].S→D	1/1
D=Rs*[Rt++](SS)	Rs.S*[Rt++].S→D	1/1
D=Rs*[Rt--](SS)	Rs.S*[Rt––].S→D	1/1
D=Rs*P[Rt](SS)	Rs.S*P[Rt].S→ D	1/2(1)
D=Rs*P[Rt++](SS)	Rs.S*P[Rt++].S→D	1/2(1)
D=[Rs++]*P[Rt--](SS)	[Rs++].S*P[Rt––].S→D	1/2(1)
D=[Rs++]*P[Rt++](SS)	[Rs++].S*P[Rt++].S → D	1/2(1)

4.2.8 DSP指令

MUL.SU指令

指令格式	執行功能	W/C
D=Rs*Rt(SU)	Rs.S*Rt.U → D	1/1
D=Rs*[Rt] (SU)	Rs.S*[Rt].U → D	1/1
D=Rs*[Rt++](SU)	Rs.S*[Rt++].U → D	1/1
D=Rs*[Rt--](SU)	Rs.S*[Rt--].U → D	1/1
D=Rs*P[Rt](SU)	Rs.S*P[Rt].U → D	1/2(1)
D=Rs*P[Rt++](SU)	Rs.S*P[Rt++].U → D	1/2(1)
D=[Rs++]*P[Rt--](SU)	[Rs++].S *P[Rt−−].U → D	1/2(1)
D=[Rs++]*P[Rt++](SU)	[Rs++].S *P[Rt++].U → D	1/2(1)

4.2.8 DSP指令

MUL.US指令

指令格式	執行功能	W/C
D=Rs*Rt(US)	Rs.U*Rt.S → D	1/1
D=Rs*[Rt] (US)	Rs.U*[Rt].S → D	1/1
D=Rs*[Rt++](US)	Rs.U*[Rt++].S → D	1/1
D=Rs*[Rt--](US)	Rs.U*[Rt--].S → D	1/1
D=Rs*P[Rt](US)	Rs.U*P[Rt].S → D	1/2(1)
D=Rs*P[Rt++](US)	Rs.U*P[Rt++].S → D	1/2(1)
D=[Rs++]*P[Rt--](US)	[Rs++].U*P[Rt––].S → D	1/2(1)
D=[Rs++]*P[Rt++](US)	[Rs++].U*P[Rt++].S → D	1/2(1)

4.2.8 DSP指令

MUL.UU指令

指令格式	執行功能	W/C
D=Rs*Rt(UU)	Rs.U*Rt.U → D	1/1
D=Rs*[Rt] (UU)	Rs.U*[Rt].U → D	1/1
D=Rs*[Rt++](UU)	Rs.U*[Rt++].U → D	1/1
D=Rs*[Rt--](UU)	Rs.U*[Rt--].U → D	1/1
D=Rs*P[Rt](UU)	Rs.U*P[Rt].U → D	1/2(1)
D=Rs*P[Rt++](UU)	Rs.U*P[Rt++].U → D	1/2(1)
D=[Rs++]*P[Rt--](UU)	[Rs++].U*P[Rt––].U → D	1/2(1)
D=[Rs++]*P[Rt++](UU)	[Rs++].U*P[Rt++].U → D	1/2(1)

4.2.8 DSP指令

DIV.S指令

指令格式	執行功能	W/C
D=D/Rs(S)	Signed [R1:R0] / Rs → R0 = Quotient	1/17(16)

DIV.U指令

指令格式	執行功能	W/C
D=D/Rs	Unsigned [R1:R0] / Rs→R0 = Quotient	1/17(16)

4.2.8 DSP指令

MAC指令

指令格式	執行功能	W/C
D=D+Rs*Rt	D+Rs*Rt → D	1/1
D=D+Rs*[Rt]	D+Rs*[Rt] → D	1/1
D=D+Rs*[Rt++]	D+Rs*[Rt++] → D	1/1
D=D+Rs*[Rt--]	D+Rs*[Rt--] → D	1/1
D=D+Rs*P[Rt]	D+Rs*P[Rt] → D	1/2(1)
D=D+Rs*P[Rt++]	D+Rs*P[Rt++] → D	1/2(1)
D=D+[Rs++]*P[Rt--]	D+[Rs++]*P[Rt--] → D	1/2(1)
D=D+[Rs++]*P[Rt++]	D+[Rs++]*P[Rt++] → D	1/2(1)

4.2.8 DSP指令

MAS指令

指令格式	執行功能	W/C
D=D-Rs*Rt	D-Rs*Rt → D	1/1
D=D-Rs*[Rt]	D-Rs*[Rt] → D	1/1
D=D-Rs*[Rt++]	D-Rs*[Rt++] → D	1/1
D=D-Rs*[Rt--]	D-Rs*[Rt--] → D	1/1
D=D-Rs*P[Rt]	D-Rs*P[Rt] → D	1/2(1)
D=D-Rs*P[Rt++]	D-Rs*P[Rt++] → D	1/2(1)
D=D-[Rs++]*P[Rt--]	D-[Rs++]*P[Rt--] → D	1/2(1)
D=D-[Rs++]*P[Rt++]	D-[Rs++]*P[Rt++] → D	1/2(1)