吳俊德實驗室 W5 處理器指令集介紹

DATA Transfer Instruction

MOV RD, N

OPCODE

RD <= RA Op L		
23 21	20 16	15 0
010	RD	LITERAL

Note: 數值 N 表示有兩種形式

(1) 16 進位: 0xN ex: 0x1000

(2) 10 進位: #N ex: #15

MOV RD, RA

OPCODE

RD <= RA				
23 20	19 16	15 11	10 6	5 0
0000	0001	RD	RA	NO USE

PORT 輸出輸入指令:

總共支援 8 組 PORT

MOV P#, RA

OPCODE

I/O port <= RA				
23 20	19 16	15 11	10 6	5 0
1100	0#	NO USE	RA	NO USE

MOV RD, P#

OPCODE

RD <= I/O port				
23 20	19 16	15 11	10 6	5 0
1100	1#	RD	NO USE	NO USE

以上"#"為 port 的數值(0~7)

譬如:

mov p1,r1 → 1100_0001_00000_00001_000000 //將 r1 的值輸出到 port1

mov r1,p1 → 1100_1001_00001_00000_000000 //將 port1 的值輸入存到 r1

記憶體定址:

MOV RAM[RA], RB

OPCODE

RAM[RA] <= RB					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
1110	**1000**	**NO USE**	**RA**	**RB**	**NO USE**

MOV RD, RAM[RA]
MOV RD, ROM[RA]

OPCODE

RD <= RAM[RA]				
23 20	**19 16**	**15 11**	**10 6**	**5 0**
1110	**0000**	**RD**	**RA**	**NO USE**
RD <= ROM[RA]				
23 20	**19 16**	**15 11**	**10 6**	**5 0**
1111	**0000**	**RD**	**RA**	**NO USE**

MOV RD, RAM[DS+RA]
MOV RD, ROM[DS+RA]

OPCODE

RD <= RAM[DS op RA]				
23 20	**19 16**	**15 11**	**10 6**	**5 0**
1110	**0100**	**RD**	**RA**	**NO USE**
RD <= ROM[DS op RA]				
23 20	**19 16**	**15 11**	**10 6**	**5 0**
1111	**0100**	**RD**	**RA**	**NO USE**

MOV RAM[DS+RA], RB

OPCODE

RAM[DS op RA] <= RB					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
1110	**1100**	**NO USE**	**RA**	**RB**	**NO USE**

PUSH RA

OPCODE

RAM<= RA				
23 20	**19 16**	**15 11**	**10 6**	**5 0**
1110	**1110**	**NO USE**	**RA**	**NO USE**

POP RD

OPCODE

RD <= RAM				
23 20	**19 16**	**15 11**	**10 6**	**5 0**
1110	**0110**	**RD**	**NO USE**	**NO USE**

Boolean Type instruction

ABS RA (RD = RA)

ABS RD, RA

OPCODE

RD <= RA				
23 20	**19 16**	**15 11**	**10 6**	**5 0**
0000	**0011**	**RD**	**RA**	**NO USE**

NOT RA (RD = RA)

NOT RD, RA

OPCODE

RD <= RA				
23 20	**19 16**	**15 11**	**10 6**	**5 0**
0000	**0101**	**RD**	**RA**	**NO USE**

AND RD, RA, RB

OPCODE

RD <= RA Op RB					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
0000	**0110**	**RD**	**RA**	**RB**	**NO USE**

OR RD, RA, RB

OPCODE

RD <= RA Op RB					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
0000	**0111**	**RD**	**RA**	**RB**	**NO USE**

XOR RD, RA, RB

OPCODE

RD <= RA Op RB					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
0000	**1000**	**RD**	**RA**	**RB**	**NO USE**

Add/Sub Type instruction

INC RA (RD=RA)

INC RD, RA

OPCODE

RD <= RA					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
0001	**0000**	**RD**	**RA**	**NO USE**	**NO USE**

ADD RD, RA, RB

OPCODE

RD <= RA Op RB					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
0001	**0001**	**RD**	**RA**	**RB**	**NO USE**

DEC RA (RA=RD)

DEC RD, RA

OPCODE

RD <= RA					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
0001	**0100**	**RD**	**RA**	**NO USE**	**NO USE**

SUB RD, RA, RB

OPCODE

RD <= RA Op RB					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
0001	**0101**	**RD**	**RA**	**RB**	**NO USE**

DIV/MOD Type instruction

DIV RD, RA, RB

OPCODE

RD <= RA Op RB					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
0001	**1111**	**RD**	**RA**	**RB**	**NO USE**

MOD RD, RA, RB

OPCODE

RD <= RA Op RB					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
0001	**1011**	**RD**	**RA**	**RB**	**NO USE**

Note: 由於除法器硬體需要消耗大量計算量，因此在 firmware 裡如需用到除法器的功能(除法 DIV & 餘數 MOD)需再額外呼叫多行以確保輸出之值為正確在此請遵照以下範例以 call function 的方式執行

```
1   mov r1,#45
2   mov r2,#7
3   call @mod
4   nop
.   .
.   .
.   .

30  mod: div r0,r1,r2
31          div r0,r1,r2
32          div r0,r1,r2
33          div r0,r1,r2
34          ret
35          nop
```

Shift/Rotate Type

LSHL RD, RA, RB

OPCODE

RD <= RA Op RB (unsigned shift)					
23 20	19 16	15 11	10 6	5 1	0 0
0011	0000	RD	RA	RB	NO USE

LSHR RD, RA, RB

OPCODE

RD <= RA Op RB (unsigned shift)					
23 20	19 16	15 11	10 6	5 1	0 0
0011	0100	RD	RA	RB	NO USE

ASHL RD, RA, RB

OPCODE

RD <= RA Op RB (signed shift)					
23 20	19 16	15 11	10 6	5 1	0 0
0011	0010	RD	RA	RB	NO USE

ASHR RD, RA, RB

OPCODE

RD <= RA Op RB (signed shift)					
23 20	19 16	15 11	10 6	5 1	0 0
0011	0110	RD	RA	RB	NO USE

ROL RD, RA, RB

OPCODE

RD <= RA Op RB (rotated shift)					
23 20	19 16	15 11	10 6	5 1	0 0
0011	1000	RD	RA	RB	NO USE

ROR RD, RA,RB

OPCODE

RD <= RA Op RB (rotated shift)					
23 20	19 16	15 11	10 6	5 1	0 0
0011	1100	RD	RA	RB	NO USE

Mac Type instruction

MUL RD, RA, RB

OPCODE

RD <= RA Op RB					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
0001	**1000**	**RD**	**RA**	**RB**	**NO USE**

MACA RD, RA, RB

OPCODE

RD <= RA Op RB					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
0001	**1001**	**RD**	**RA**	**RB**	**NO USE**

MACS RD, RA, RB

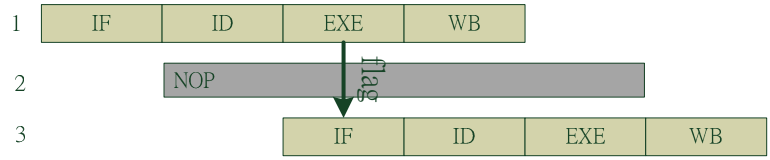
OPCODE

RD <= RA Op RB					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
0001	**1010**	**RD**	**RA**	**RB**	**NO USE**

Control Type instruction

NOTE:

1. jump 分支指令: JN, JFN, JC, JFC, JZ, JFZ, JBZ, JFBZ 需要在下一行多加一行 NOP，原因是分支執行判斷所需的 flag(negative, carry, zero, Boolean zero) 需要在 pipeline 中 execution 階段之後才能決定，故在此我們需要在上述這些分支指令後多加一行 NOP 指令(原先需要兩個 NOP，但因使用 forward 技術因此可減少至一個 NOP)(PC=PC+1 屬於 ID stage)



2. 其他 CONTROL type 指令在 IF 階段就執行完成，故不需要加上任何的 NOP

NOP

OPCODE

NOP					
23 20	**19 16**	**15 11**	**10 6**	**5 1**	**0 0**
0000	**0000**	**0000**	**0000**	**0000**	**000000000000000**

JMP N

PC <= Op L			
23 20	**19 16**	**15 14**	**13 0**
1101	**0010**	**NO USE**	**LITERAL**

JN N

PC <= Op L (jump if arithmetic result was negative)			
23 20	**19 16**	**15 14**	**13 0**
1101	**0100**	**00**	**LITERAL**

JFN N

PC <= Op L(jump if arithmetic result was not negative)			
23 20	**19 16**	**15 14**	**13 0**
1101	**0110**	**00**	**LITERAL**

JC N

PC <= Op L(jump if arithmetic result have carry)			
23 20	**19 16**	**15 14**	**13 0**
1101	**0100**	**01**	**LITERAL**

JFC N

PC <= Op L(jump if arithmetic result don't have carry)			
23 20	**19 16**	**15 14**	**13 0**
1101	**0110**	**01**	**LITERAL**

JZ N

PC <= Op L(jump if arithmetic result was zero)			
23 20	19 16	15 14	13 0
1101	0100	10	LITERAL

JFZ N

PC <= Op L(jump if arithmetic result was not zero)			
23 20	19 16	15 14	13 0
1101	0110	10	LITERAL

JBZ N

PC <= Op L(jump if logic result was zero)			
23 20	19 16	15 14	13 0
1101	0100	11	LITERAL

JFBZ N

PC <= Op L (jump if logic result was not zero)			
23 20	19 16	15 14	13 0
1101	0110	11	LITERAL

CALL N

PC <= Op L			
23 20	19 16	15 14	13 0
1101	1000	NO USE	LITERAL

RET

PC <= Op L		
23 20	19 16	15 0
1101	1010	NO USE

LOOP RA

PC <= RA				
23 20	19 16	15 11	10 6	5 0
1101	1111	NO USE	RA	NO USE

END

PC <= Op L		
23 20	19 16	15 0
1101	1110	NO USE

Note: 1. loop 指令的 RA 暫存器可使用 32 bits

2. loop layer 可支援 8 層運作

Interrupt Type Instruction

總共包含 8 組中斷向量，其中 Interrupt 序號 4~7 為計數器中斷向量，Interrupt 序號 0~3 為外部中斷向量。[Interrupt 序號大者]優先權高於[Interrupt 序號小者]，例如: TINT3 > … > TINT0> INT3 > … > INT0。當中斷向量觸發時 PC 會跳到最上層的 PC address 直到 reti 執行為止再跳回原跳出之 PC address。其中 以下為各組中斷向量跳至位址

Interrupt 序號	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Interrupt 名稱	**INT0**	**INT1**	**INT2**	**INT3**	**TINT0**	**TINT1**	**TINT2**	**TINT3**
跳躍後的 PC address (PC address 從 0 開始計算)	1	2	3	4	5	6	7	8

1. 優先權高的中斷向量任務還沒執行完時(沒有執行 RETI 指令)，優先權低的中斷向量進來，此時優先權低的中斷向量就不會被執行。
2. 優先權低的中斷向量任務還沒執行完時(沒有執行 RETI 指令)，優先權高的中斷向量進來，此時原先正在執行的 PC address 會被儲存進 stack，待優先權高的中斷向量任務執行完畢再返回優先權低的任務(跳回原被 stack 住的 pc address)。
3. 若多個中斷向量同時間被觸發時，只有優先權最高的中斷向量任務會被執行，其他的中斷向量都不會被執行。

SETI N (中斷設定指令)

IR <= Enable, Priority			
23 20	**19 16**	**15 12**	**11 0**
1101	**1101**	**Enable**	**NO USE**

Enable = N

中斷向量指令中 Enable 的數值，其對應的二進位位元值若為 1，則其代表的那一根中斷(interrupt)向量可以使用。反之若為 0，則其代表的那一根中斷(interrupt)向量不可以使用

Ex: seti 0xf

所有計數器中斷向量都可以使用。

SET TINT#, RA (計數器中斷產生指令)

OPCODE

TIME <= **RA**				
23 20	**19 16**	**15 11**	**10 6**	**5 0**
1000	**0010**	**TINT#**	**RA**	**NO USE**

總共支援 4 個 interrupt counter

TINT 指令的 RA 暫存器可使用 32 bits

Ex: set TINT0, r0

計數器數到 r0 的質時會觸發 INT0

RETI

PC <= Op L		
23 20	**19 16**	**15 0**
1101	**1100**	**NO USE**

PWM Instruction

Note: **PWM** 指令的 RA 暫存器可使用 32 bits

SET TCMP#, RA

OPCODE

TCMP <= **RA**				
23 20	**19 16**	**15 11**	**10 6**	**5 0**
1000	**0000**	TCMP#	**RA**	**NO USE**

SET TCNT#, RA

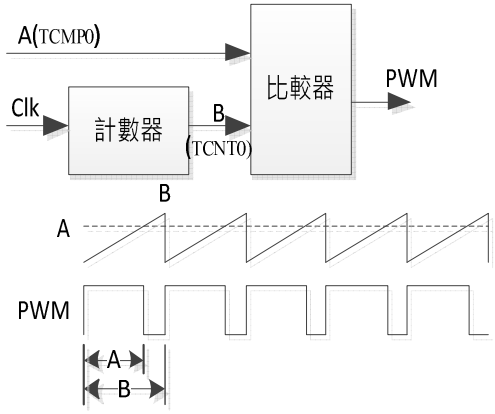
OPCODE

TCNT <= **RA**				
23 20	**19 16**	**15 11**	**10 6**	**5 0**
1000	**0001**	TCNT#	**RA**	**NO USE**

NOTE:

1. set TCMP0, r0
   PWM0 每個 PLUS 寬度為 r0 裡的數值
   換句話說如果 r0=100, PWM0 每個 PLUS 寬度為 100 個 clock cycle

2. set TCNT0,r0
   PWM0 每個周期寬度為 r0 裡的數值
   換句話說如果 r0=200, PWM0 每個周期寬度為 200 個 clock cycle



3. 總共支援 16 個 PWM 輸出:
   TCMP0~TCMP15 與 TCNT0~TCNT15