ADDC——无符号带进位加法指令

统一化指令

语法	操作	编译结果

addc rz, rx	RZ ← RZ + RX + C，
	C←进位
		根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<16) and (z<16), then
		addc16 rz, rx;
		else
		addc32 rz, rz, rx;

addc rz, rx, ry	RZ ← RX + RY + C，
	C←进位
		根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (y==z) and (x<16) and (z<16), then
		addc16 rz, rx;
		else
		addc32 rz, rx, ry;

说明： 将 RZ/RY、RX 与 C 位的值相加，并把结果存在 RZ，进位存在 C 位。

影响标志位： C ← 进位

异常： 无

16位指令

操作： RZ ← RZ + RX + C，C←进位

语法： addc16 rz, rx

说明： 将 RZ、RX 与 C 位的值相加，并把结果存在 RZ，进位存在 C 位。

影响标志位： C ← 进位

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15 14 10 9 6 5 2 1 0

0	1 1 0 0 0	RZ	RX	0 1

32位指令

操作： RZ ← RX + RY + C，C←进位

语法： addc32　rz, rx, ry

说明： 将 RX、RY 与 C 位的值相加，并把结果存在 RZ，进位存在 C 位。

影响标志位： C ← 进位

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	RY	RX	0 0 0 0 0 0	0 0 0 1 0	RZ

ADDI——无符号立即数加法指令

统一化指令

语法	操作	编译结果

addi rz, oimm12	RZ ← RZ + zero_extend(OIMM12)	根据立即数和寄存器的范围编译为对应的 16 位或 32 位指令。
		if (z<8) and (oimm12<257),
		addi16 rz, oimm8;
		else
		addi32 rz, rz, oimm12;

addi rz, rx, oimm12	RZ ← RX + zero_extend(OIMM12)	根据立即数和寄存器的范围编译为对应的 16 位或 32 位指令。
		if (oimm12<9) and (z<8) and (x<8),
		addi16 rz, rx, oimm3;
		elsif (oimm12<257) and (x==z) and (z<8),
		addi16 rz, oimm8;
		else
		addi32 rz, rx, oimm12;

说明： 将带偏置 1 的立即数零扩展至 32 位，然后与 RX/RZ 的值相加，把结果存入 RZ。

影响标志位： 无影响

限制： 若源寄存器是 R28，立即数的范围为 0x1-0x40000。

若源寄存器不是 R28，立即数的范围为 0x1-0x1000。

异常： 无

16位指令----1

操作： RZ ← RZ + zero_extend(OIMM8)

语法： addi16 rz, oimm8

说明： 将带偏置 1 的 8 位立即数（OIMM8）零扩展至 32 位，然后与 RZ 的值相加，把结果存入 RZ。

注意：二进制操作数 IMM8 等于 OIMM8 - 1。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7；立即数的范围为 1-256。

异常： 无

指令格式:

15	14 11	10 8	7 0
0	0 1 0 0	RZ	IMM8

IMM8 域——指定不带偏置立即数的值。

注意：加到寄存器里的值 OIMM8 比起二进制操作数 IMM8 需偏置 1。

00000000——加 1

00000001——加 2

……

11111111——加 256

16位指令----2

操作： RZ ← RX + zero_extend(OIMM3)

语法： addi16 rz, rx, oimm3

说明： 将带偏置 1 的 3 位立即数（OIMM3）零扩展至 32 位，然后与 RX 的值相加，把结果存入 RZ。

注意：二进制操作数 IMM3 等于 OIMM3 - 1。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7；立即数的范围为 1-8。

异常： 无

指令格式:

15	14 10	9 8	7 5	4 2	1 0
0	1 0 1 1	RX	RZ	IMM3	1 0

IMM3 域——指定不带偏置立即数的值。

注意：加到寄存器里的值 OIMM3 比起二进制操作数 IMM3 需偏置 1。

000——加 1

001——加 2

……

111——加 8

32位指令

操作： RZ ← RX + zero_extend(OIMM12)

语法： addi32　rz, rx, oimm12

说明： 将带偏置 1 的 12 位立即数（OIMM12）零扩展至 32 位，然后与 RX 的值相加，把结果存入 RZ。

注意：二进制操作数 IMM12 等于 OIMM12 - 1。

影响标志位： 无影响

限制： 立即数的范围为 0x1-0x1000。

异常： 无

指令格式：

31	30 26	25 21	20 16	15 12	11 0
1	1 1 0 0 1	RZ	RX	0 0 0 0	IMM12

IMM12 域——指定不带偏置立即数的值。

注意：加到寄存器里的值 OIMM12 比起二进制操作数 IMM12 需偏置 1。

000000000000——加 0x1

000000000001——加 0x2

……

111111111111——加 0x1000

ADDI(SP)——无符号（堆栈指针）立即数加法指令

统一化指令

语法	操作	编译结果
addi rz, sp, imm	RZ ← SP + zero_extend(IMM)	仅存在 16 位指令。 addi rz, sp, imm
addi sp, sp, imm	SP ← SP+ zero_extend(IMM)	仅存在 16 位指令。 addi sp, sp, imm

说明： 将立即数（IMM）零扩展至 32 位，然后与堆栈指针（SP）的值相加，把结果存入 RZ 或者 SP。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7；立即数的范围为 0x0-0x3fc。

异常： 无

16位指令----1

操作： RZ ← SP + zero_extend(IMM)

语法： addi16 rz, sp, imm8

说明： 将立即数（IMM）零扩展至 32 位，然后与堆栈指针（SP）的值相加，把结果存入 RZ。

注意：立即数（IMM）等于二进制操作数 IMM8 << 2。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7；立即数的范围为(0x0-0xff) << 2。

异常： 无

指令格式：

15 14 11 10 8 7 0

0	0 0 1 1	RZ	IMM8

IMM8 域——指定不带移位立即数的值。

注意：加到寄存器里的值 IMM 比起二进制操作数 IMM8 需左移 2 位。

00000000——加 0x0

00000001——加 0x4

……

11111111——加 0x3fc

16位指令----2

操作： SP ← SP + zero_extend(IMM)

语法： addi16　sp, sp, imm

说明： 将立即数（IMM）零扩展至 32 位，然后与堆栈指针（SP）的值相加，把结果存入 RZ。

注意：立即数（IMM）等于二进制操作数{IMM2，IMM5} << 2。

影响标志位： 无影响

限制： 源与目的寄存器均为堆栈指针寄存器（R14）；立即数的范围为(0x0-0x7f) << 2。

异常： 无

指令格式：

15	14 13 12 11	10	9 8	7 6 5	4 3 2 1 0
0	0 0 1 0	1	IMM2	0 0 0	IMM5

IMM 域——指定不带移位立即数的值。

注意：加到寄存器里的值 IMM 比起二进制操作数{IMM2，IMM5}需左移 2 位。

{00, 00000}——加 0x0

{00, 00001}——加 0x4

……

{11, 11111}——加 0x1fc

ADDU——无符号加法指令

统一化指令

语法	操作	编译结果

addu rz, rx	RZ ← RZ + RX	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (z<16) and (x<16), then
		addu16 rz, rx;
		else
		addu32 rz, rz, rx;

addu rz, rx, ry	RZ ← RX + RY	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (z<8) and (x<8) and (y<8), then
		addu16 rz, rx, ry;
		elsif (y==z) and (x<16) and (z<16), then
		addu16 rz, rx;
		else
		addu32 rz, rx, ry;

说明： 将 RZ/RY 与 RX 的值相加，并把结果存在 RZ。

影响标志位： 无影响

异常： 无

16位指令----1

操作： RZ ← RZ + RX

语法： addu16 rz, rx

说明： 将 RZ 与 RX 的值相加，并把结果存在 RZ。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

1514 10 9 6 5 2 1 0

0	1 1 0 0 0	RZ	RX	0 0

16位指令----2

操作： RZ ← RX + RY

语法： addu16 rz, rx, ry

说明： 将 RX 与 RY 的值相加，并把结果存在 RZ。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7。

异常： 无

指令格式：

15 14 11 10 8 7 5 4 2 1 0

0	1 0 1 1	RX	RZ	RY	0 0

32 位指令

操作： RZ ← RX + RY

语法： addu32 rz, rx, ry

说明： 将 RX 与 RY 的值相加，并把结果存在 RZ。

影响标志位： 无影响

异常： 无

指令格式：

31 30 26 25 21 20 16 15 10 9 5 4 0

1	1 0 0 0 1	RY	RX	0 0 0 0 0 0	0 0 0 0 1	RZ

AND——按位与指令

统一化指令

语法	操作	编译结果

and rz, rx	RZ ← RZ and RX	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<16) and (z<16), then
		and16 rz, rx;
		else
		and32 rz, rz, rx;

and rz, rx, ry	RZ ← RX and RY	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (y==z) and (x<16) and (z<16), then
		and16 rz, rx;
		else
		and32 rz, rx, ry;

说明： 将 RZ/RY 与 RX 的值按位与，并把结果存在 RZ；

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← RZ and RX

语法： and16 rz, rx

说明： 将 RZ 与 RX 的值按位与，并把结果存在 RZ。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15 14 10 9 6 5 2 1 0

0	1 1 0 1 0	RZ	RX	0 0

32位指令

操作： RZ ← RX and RY

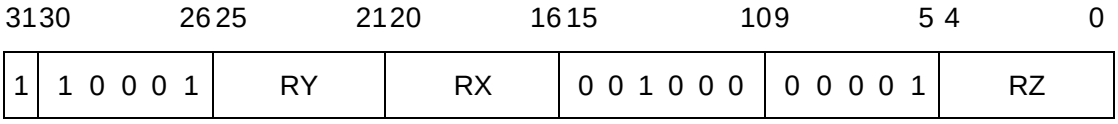
语法： and32　rz, rx, ry

说明： 将 RX 与 RY 的值按位与，并把结果存在 RZ。

影响标志位： 无影响

异常： 无

指令格式：



ANDI——立即数按位与指令

统一化指令

语法	操作	编译结果
andi rz, rx, imm12	RZ ← RX and zero_extend(IMM12)	仅存在 32 位指令 andi32 rz, rx, imm12

说明： 将 12 位立即数零扩展至 32 位，然后与 RX 的值进行按位与操作，把结果存入 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0x0-0xFFF。

异常： 无

32位指令

操作： RZ ← RX and zero_extend(IMM12)

语法： andi32 rz, rx, imm12

说明： 将 12 位立即数零扩展至 32 位，然后与 RX 的值进行按位与操作，把结果存入 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0x0-0xFFF。

异常： 无

指令格式：

31	30	26	25	21	20	16	15	12	11	0
1	1 1 0 0 1		RZ		RX	0 0 1 0			IMM12	

ANDN——按位非与指令

统一化指令

语法	操作	编译结果

andn rz, rx	RZ ← RZ and (!RX)	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<16) and (z<16), then
		andn16 rz, rx;
		else
		andn32 rz, rz, rx;

andn rz, rx, ry	RZ ← RX and (!RY)	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x==z) and (y<16) and (z<16), then
		andn16 rz, ry;
		else
		andn32 rz, rz, rx;

说明： 对于 andn rz, rx，将 RZ 的值与 RX 的非值按位与，并把结果存在 RZ；对于 andn rz, rx, ry，将 RX 的值与 RY 的非值按位与，并把结果存在 RZ。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← RZ and (!RX)

语法： andn16 rz, rx

说明： 将 RZ 的值与 RX 的非值按位与，并把结果存在 RZ。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15	14 10	9 6	5 2	1 0
0	1 1 0 1 0	RZ	RX	0 1

32位指令

操作： RZ ← RX and (!RY)

语法： andn32 rz, rx, ry

说明： 将 RX 的值与 RY 的非值按位与，并把结果存在 RZ。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	RY	RX	0 0 1 0 0 0	0 0 0 1 0	RZ

ANDNI——立即数按位非与指令

统一化指令

语法	操作	编译结果
andni rz, rx, imm12	RZ ← RX and !(zero_extend(IMM12))	仅存在 32 位指令 andni32 rz, rx, imm12

说明： 将 12 位立即数零扩展至 32 位并取非，然后与 RX 的值进行按位与操作，把结果存入 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0x0-0xFFF。

异常： 无

32位指令

操作： RZ ← RX and !(zero_extend(IMM12))

语法： andni32 rz, rx, imm12

说明： 将 12 位立即数零扩展至 32 位并取非，然后与 RX 的值进行按位与操作，把结果存入 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0x0-0xFFF。

异常： 无

指令格式：

31	30	26	25	21	20	16	15	12	11	0
1	1 1 0 0 1		RZ		RX		0 0 1 1		IMM12	

ASR——算术右移指令

统一化指令

语法	操作	编译结果

asr rz, rx	RZ ← RZ >>> RX[5:0]	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<16) and (z<16), then
		asr16 rz, rx;
		else
		asr32 rz, rz, rx;

asr rz, rx, ry	RZ ← RX >>> RY[5:0]	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x==z) and (y<16) and (z<16), then
		asr16 rz, ry;
		else
		asr32 rz, rx, ry;

说明： 对于 asr rz, rx，将 RZ 的值进行算术右移（原值右移，左侧移入原符号位的拷贝），结果存入 RZ，右移位数由 RX 低 6 位（RX[5:0]）的值决定；如果 RX[5:0]的值大于 30，那么 RZ 的值（0 或-1）由 RZ 原值的符号位决定；

对于 asr rz, rx, ry，将 RX 的值进行算术右移（原值右移，左侧移入原符号位的拷贝），结果存入 RZ，右移位数由 RY 低 6 位（RY[5:0]）的值决定；如果 RY[5:0]的值大于 30，那么 RZ 的值（0 或-1）由 RX 的符号位决定。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← RZ >>> RX[5:0]

语法： asr16 rz, rx

说明： 将 RZ 的值进行算术右移（原值右移，左侧移入原符号位的拷贝），结果存入 RZ，右移位数由 RX 低 6 位（RX[5:0]）的值决定；如果 RX[5:0]的值大于 30，那么 RZ 的值（0 或-1）由 RZ 原值的符号位决

定。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15	14 10	9 6	5 2	1 0
0	1 1 1 0 0	RZ	RX	1 0

32位指令

操作： RZ ← RX >>> RY[5:0]

语法： asr32 rz, rx, ry

说明： 将 RX 的值进行算术右移（原值右移，左侧移入原符号位的拷贝），结果存入RZ，右移位数由RY低6位（RY[5:0]）的值决定；如果RY[5:0]的值大于 30，那么 RZ 的值（0 或-1）由 RX 的符号位决定。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	RY	RX	0 1 0 0 0 0	0 0 1 0 0	RZ

ASRC——立即数算术右移至 C 位指令

统一化指令

语法	操作	编译结果
asrc rz, rx, oimm5	RZ ← RX >>> OIMM5, C ← RX[OIMM5 - 1]	仅存在 32 位指令 asrc32 rz, rx, oimm5

说明： 将 RX 的值进行算术右移（原值右移，左侧移入原符号位的拷贝），把移出最末位存入条件位 C，移位结果存入 RZ，右移位数由带偏置 1 的 5 位立即数（OIMM5）的值决定。如果 OIMM5 的值等于 32，那么条件位 C 为 RX 的符号位（最高位），RZ 的值（0 或-1）由 RX 的符号位决定。

影响标志位： C ← RX[OIMM5 - 1]

限制： 立即数的范围为 1-32。

异常： 无

32位指令

操作： RZ ← RX >>> OIMM5, C ← RX[OIMM5 - 1]

语法： asrc32 rz, rx, oimm5

说明： 将 RX 的值进行算术右移（原值右移，左侧移入原符号位的拷贝），把移出最末位存入条件位 C，移位结果存入 RZ，右移位数由带偏置 1 的 5 位立即数（OIMM5）的值决定。如果 OIMM5 的值等于 32，那么条件位 C 为 RX 的符号位（最高位），RZ 的值（0 或-1）由 RX 的符号位决定。注意：二进制操作数 IMM5 等于 OIMM5 - 1。

影响标志位： C ← RX[OIMM5 - 1]

限制： 立即数的范围为 1-32。

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	IMM5	RX	0 1 0 0 1 1	0 0 1 0 0	RZ

IMM5 域——指定不带偏置立即数的值。

注意：移位的值 OIMM5 比起二进制操作数 IMM5 需偏置 1。

00000——移 1 位

00001——移 2 位

……

11111——移 32 位

ASRI——立即数算术右移指令

统一化指令

语法	操作	编译结果

asri　rz, rx, imm5	RZ ← RX >>> IMM5	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<8) and (z<8), then
		　asri16　rz, rx, imm5;
		else
		　asri32　rz, rx, imm5;

说明： 对 asri　rz, rx, imm5 而言，将 RX 的值进行算术右移（原值右移，左侧移入原符号位的拷贝），结果存入 RZ，右移位数由 5 位立即数（IMM5）的值决定；如果 IMM5 的值等于 0，那么 RZ 的值将与 RX 相同。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← RX >>> IMM5

语法： asri16　rz, rx, imm5

说明： 将 RX 的值进行算术右移（原值右移，左侧移入原符号位的拷贝），结果存入 RZ，右移位数由 5 位立即数（IMM5）的值决定；如果 IMM5 的值等于 0，那么 RZ 的值将不变。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7；立即数的范围为 0-31。

异常： 无

指令格式：

15 14　　11 10　8 7　5 4　0

0	1 0 1 0	RX	RZ	IMM5

32位指令

操作： RZ ← RX >>> IMM5

语法： asri32 rz, rx, imm5

说明： 将 RX 的值进行算术右移（原值右移，左侧移入原符号位的拷贝），结果存入 RZ，右移位数由 5 位立即数（IMM5）的值决定；如果 IMM5 的值等于 0，那么 RZ 的值将与 RX 相同。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	IMM5	RX	0 1 0 0 1 0	0 0 1 0 0	RZ

BCLRI——立即数位清零指令

统一化指令

语法	操作	编译结果

bclri rz, imm5	RZ ← RZ[IMM5]清零	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (z<8)，then
		bclri16 rz, imm5;
		else
		bclri32 rz, rz, imm5;

bclri rz, rx, imm5	RZ ← RX[IMM5]清零	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x==z) and (z<8), then
		bclri16 rz, imm5;
		else
		bclri32 rz, rx, imm5;

说明： 将 RZ/RX 的值中，由 IMM5 域值所指示的位清零，其余位保持不变，把清零后的结果存入 RZ；

影响标志位： 无影响

限制： 立即数的范围为 0-31。

16位指令

操作： RZ ← RZ[IMM5]清零

语法： bclri16 rz, imm5

说明： 将 RZ 的值中，由 IMM5 域值所指示的位清零，其余位保持不变，把清零后的结果存入 RZ。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7；

立即数的范围为 0-31。

异常： 无

指令格式：

15 14 10 8 7 5 4 0

0	0 1 1 1	RZ	1 0 0	IMM5

32位指令

操作： RZ ← RX[IMM5]清零

语法： bclri32　rz, rx, imm5

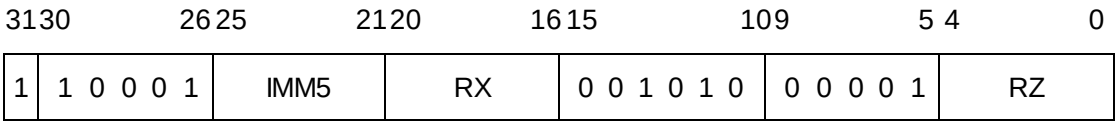
说明： 将 RX 的值中，由 IMM5 域值所指示的位清零，其余位保持不变，把清零后的结果存入 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

指令格式：



BF——C 为 0 分支指令

统一化指令

语法	操作	编译结果

bf　label	C 等于零则程序转移。
	if(C==0)
	　PC←PC + sign_extend(offset << 1);
	else
	　PC ← next PC;
		根据跳转的范围编译为对应的 16 位或 32 位指令。
		if (offset<1KB), then
		　bf16　label;
		else
		　bf32　label;

说明： 如果条件标志位 C 等于零，则程序转移到 label 处执行；否则程序执行下一条指令。

Label 由当前程序 PC 加上左移 1 位的相对偏移量有符号扩展到 32 位后的值得到。BF 指令的转移范围是±64KB 地址空间。

影响标志位： 无影响

异常： 无

16位指令

操作： C 等于零则程序转移。

if(C==0)

　PC ← PC + sign_extend(offset << 1)

else

　PC ← PC + 2

语法： bf16　label

说明： 如果条件标志位 C 等于 0，则程序转移到 label 处执行；否则程序执行下一条指令，即 PC ← PC + 2。

Label 由当前程序 PC 加上左移 1 位的 10 位相对偏移量有符号扩展到 32 位后的值得到。BF16 指令的转移范围是±1KB 地址空间。

影响标志位： 无影响

异常： 无

指令格式：

15	14	10	9	0
0	0 0 0 1 1		Offset	

32位指令

操作： C 等于零则程序转移

```
if(C == 0)
   PC ← PC + sign_extend(offset << 1)
else
   PC← PC + 4
```

语法： bf32　label

说明： 如果条件标志位 C 等于零，则程序转移到 label 处执行；否则程序执行下一条指令，即 PC ← PC + 4。

Label 由当前程序 PC 加上左移 1 位的 16 位相对偏移量有符号扩展到 32 位后的值得到。BF 指令的转移范围是±64KB 地址空间。

影响标志位： 无影响

异常： 无

指令格式:

31	30	26	25	21	20	16	15	0
1	1 1 0 1 0		0 0 0 1 0		0 0 0 0 0		Offset	

BGENI——立即数位产生指令#

统一化指令

语法	操作	编译结果
bgeni rz, imm5	RZ ←（2）^{IMM5}	仅存在 32 位指令。 bgeni32 rz, imm5

说明： 对由 5 位立即数确定的 RZ 的位（RZ[IMM5]）置位，并清除 RZ 的其它位。

注意，如果 IMM5 小于 16，该指令是 movi rz, (2)^{IMM5} 的伪指令；

如果 IMM5 大于等于 16，该指令是 movih rz, (2)^{IMM5} 的伪指令。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

32位指令

操作： RZ ←（2）^{IMM5};

语法： bgeni32 rz, imm5

说明： 对由 5 位立即数确定的 RZ 的位（RZ[IMM5]）置位，并清除 RZ 的其它位。

注意，如果 IMM5 小于 16，该指令是 movi32 rz, (2)^{IMM5} 的伪指令；

如果 IMM5 大于等于 16，该指令是 movih32 rz, (2)^{IMM5} 的伪指令。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

指令格式：

如果 IMM5 小于 16：

31	30 26	25 21	20 16	15 0
1	1 1 0 1 0	1 0 0 0 0	RZ	（2）^{IMM5}

如果 IMM5 大于等于 16：

31	30 26	25 21	20 16	15 0
1	1 1 0 1 0	1 0 0 0 1	RZ	（2）^{IMM5}

BKPT——断点指令

统一化指令

语法	操作	编译结果
bkpt	引起一个断点异常或者进入调试模式	总是编译为 16 位指令。 bkpt16

说明： 断点指令

影响标志位： 无影响

异常： 断点异常

16位指令

操作： 引起一个断点异常或者进入调试模式

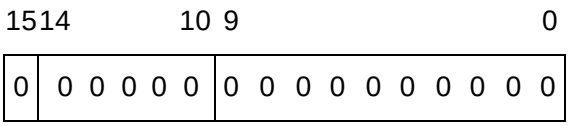
语法： bkpt16

说明： 断点指令

影响标志位： 无影响

异常： 断点异常

指令格式：



BMASKI——立即数位屏蔽产生指令

统一化指令

语法	操作	编译结果
bmaski rz, oimm5	RZ ← (2)^{OIMM5} - 1	仅存在 32 位指令 bmaski32 rz, oimm5

说明： 产生连续低位为 1、高位为 0 的立即数，并将该立即数存入 RZ。带偏置的立即数 OIMM5 指定被置 1 的连续低位（RX[OIMM5-1:0]）的位数，其余高位清零。当 OIMM5 为 0 或 32 时，RX 所有位均被置 1。

注意，OIMM5 为 1-16 时由 movi 指令执行。

影响标志位： 无影响

限制： 立即数的范围为 0，17-32；

异常： 无

32位指令

操作： RZ ← (2)^{OIMM5} - 1

语法： bmaski32 rz, oimm5

说明： 产生连续低位为 1、高位为 0 的立即数，并将该立即数存入 RZ。带偏置的立即数 OIMM5 指定被置 1 的连续低位（RX[OIMM5-1:0]）的位数，其余高位清零。当 OIMM5 为 0 或 32 时，RX 所有位均被置 1。

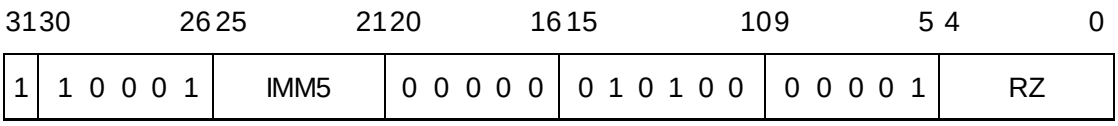
注意，OIMM5 为 1-16 时由 movi 指令执行；二进制操作数 IMM5 等于 OIMM5 - 1。

影响标志位： 无影响

限制： 立即数的范围为 0，17-32；

异常： 无

指令格式：



IMM5 域——指定被置 1 的连续低位的最高位。

注意：立即数 OIMM5 比起二进制操作数 IMM5 需偏置 1。

10000——0-16 位置位

10001——0-17 位置位

……

11111——0-31 位置位

BMCLR——BCTM 位清零指令

统一化指令

语法	操作	编译结果
bmclr	清除状态寄存器的 BM 位。 PSR(BM) ← 0	仅存在 32 位指令。 bmclr32

说明： PSR 的 BM 位被清零。

影响标志位： 无影响

异常： 无

注意： 该指令仅实现于支持二进制代码转译机制的CK802处理器。

32位指令

操作： 清除状态寄存器的 BM 位

PSR(BM) ← 0

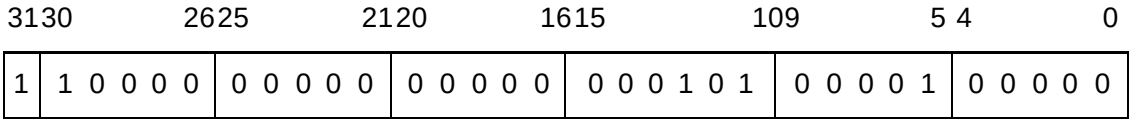
语法： bmclr32

说明： PSR 的 BM 位被清零。

影响标志位： 无影响

异常： 无

指令格式：



BPOP.H——二进制转译半字压栈指令

统一化指令

语法	操作	编译结果

bpop.h　rz	更新二进制转译堆栈寄存器到二进制转译堆栈存储器的顶端，然后从二进制转译堆栈存储器加载半字到寄存器 RZ 中；
	if (BSP - 2 < FP')
	　R15 ← next PC
	　PC ← SVBR - 12
	else
	　BSP ← BSP - 2;
	　RZ ← zero_extend(MEM[BSP]);
		仅存在 16 位指令
		　bpop.h　rz;

说明： 将二进制转译堆栈指针寄存器（BSP）减 2 的值与二进制转译帧指针寄存器（FP'）进行比较。如果 BSP 减 2 的值小于 FP'，则将子程序的返回地址（下一条指令的 PC）保存在链接寄存器 R15 中，程序转移到 SVBR-12 处执行；否则，更新 BSP 到二进制转译堆栈存储器的顶端，然后将二进制转译堆栈存储器中的半字经过零扩展到 32 位后，加载到寄存器 RZ 中。采用堆栈寄存器直接寻址方式。

影响标志位： 无影响

异常： 未对齐访问异常、未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

16位指令

操作： 更新二进制转译堆栈寄存器到二进制转译堆栈存储器的顶端，然后从二进制转译堆栈存储器加载半字到寄存器 RZ 中；

```
if (BSP - 2 < FP')
    R15 ← next PC
    PC ← SVBR - 12
else
  BSP ← BSP - 2;
  RZ ← zero_extend(MEM[BSP]);
```

语法： bpop16.h　rz

说明： 将二进制转译堆栈指针寄存器（BSP）减 2 的值与二进制转译帧指针寄存器（FP'）进行比较。如果 BSP 减 2 的值小于 FP'，则将子程序的返回地址（下一条指令的 PC）保存在链接寄存器 R15 中，程序转移到 SVBR-12 处执行；否则，更新 BSP 到二进制转译堆栈存储器的顶端，然后将二进制转译堆栈存储器中的半字经过零扩展到 32 位后，加载到寄存器 RZ 中。采用堆栈寄存器直接寻址方式。

影响标志位： 无影响

限制： 寄存器的范围为 r0 – r7。

异常： 未对齐访问异常、未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

指令格式：

15	14	10	9	8	7	6	5	4	2	1	0
0	0 0 1 0 1		0	0	1	0	1	RZ		0	0

BPOP.W——二进制转译字压栈指令

统一化指令

语法	操作	编译结果

bpop.w　rz	更新二进制转译堆栈寄存器到二进制转译堆栈存储器的顶端，然后从二进制转译堆栈存储器加载字到寄存器 RZ 中；
	if (BSP - 4 < FP')
	　R15 ← next PC
	　PC ← SVBR - 12
	else
	　BSP ← BSP - 4;
	　RZ ← MEM[BSP];
		仅存在 16 位指令
		　bpop.w　rz;

说明： 将二进制转译堆栈指针寄存器（BSP）减 4 的值与二进制转译帧指针寄存器（FP'）进行比较。如果 BSP 减 4 的值小于 FP'，则将子程序的返回地址（下一条指令的 PC）保存在链接寄存器 R15 中，程序转移到 SVBR-12 处执行；否则，更新 BSP 到二进制转译堆栈存储器的顶端，然后将二进制转译堆栈存储器中的字加载到寄存器 RZ 中。采用堆栈寄存器直接寻址方式。

影响标志位： 无影响

异常： 未对齐访问异常、未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

16位指令

操作： 更新二进制转译堆栈寄存器到二进制转译堆栈存储器的顶端，然后从二进制转译堆栈存储器加载字到寄存器 RZ 中；

```
if (BSP - 4 < FP')
    R15 ← next PC
    PC ← SVBR - 12
else
   BSP ← BSP - 4;
   RZ ← MEM[BSP];
```

语法： bpop16.w　rz

说明： 将二进制转译堆栈指针寄存器（BSP）减 4 的值与二进制转译帧指

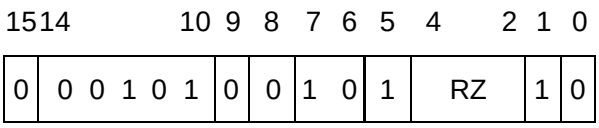
针寄存器（FP’）进行比较。如果 BSP 减 4 的值小于 FP’，则将子程序的返回地址（下一条指令的 PC）保存在链接寄存器 R15 中，程序转移到 SVBR-12 处执行；否则，更新 BSP 到二进制转译堆栈存储器的顶端，然后将二进制转译堆栈存储器中的字加载到寄存器 RZ 中。采用堆栈寄存器直接寻址方式。

影响标志位： 无影响

限制： 寄存器的范围为 r0 – r7。

异常： 未对齐访问异常、未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

指令格式：



BPUSH.H——二进制转译半字压栈指令

统一化指令

语法	操作	编译结果

bpush.h rz	将寄存器 RZ 中的半字存储到二进制转译堆栈存储器中，然后更新二进制转译堆栈寄存器到二进制转译堆栈存储器的顶端；
	if (BSP + 2 > TOP)
	R15 ← next PC
	PC ← SVBR - 12
	else
	MEM[BSP] ← RZ[15:0];
	BSP ← BSP + 2;
		仅存在 16 位指令
		bpush.h rz;

说明： 将二进制转译堆栈指针寄存器（BSP）加 2 的值与二进制转译栈顶寄存器（TOP）进行比较。如果 BSP 加 2 的值大于 TOP，则将子程序的返回地址（下一条指令的 PC）保存在链接寄存器 R15 中，程序转移到 SVBR-12 处执行；否则，将寄存器 RZ 中的低半字存储到二进制转译堆栈存储器中，然后更新 BSP 到二进制转译堆栈存储器的顶端。采用堆栈寄存器直接寻址方式。

影响标志位： 无影响

异常： 未对齐访问异常、未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

16位指令

操作： 将寄存器 RZ 中的半字存储到二进制转译堆栈存储器中，然后更新二进制转译堆栈寄存器到二进制转译堆栈存储器的顶端；

if (BSP + 2 > TOP)

R15 ← next PC

PC ← SVBR - 12

else

MEM[BSP] ← RZ[15:0];

BSP ← BSP + 2;

语法： bpush16.h rz

说明： 将二进制转译堆栈指针寄存器（BSP）加 2 的值与二进制转译栈顶

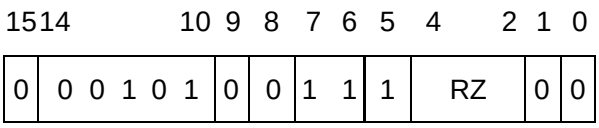
寄存器（TOP）进行比较。如果 BSP 加 2 的值大于 TOP，则将子程序的返回地址（下一条指令的 PC）保存在链接寄存器 R15 中，程序转移到 SVBR-12 处执行；否则，将寄存器 RZ 中的低半字存储到二进制转译堆栈存储器中，然后更新 BSP 到二进制转译堆栈存储器的顶端。采用堆栈寄存器直接寻址方式。

影响标志位： 无影响

限制： 寄存器的范围为 r0 – r7。

异常： 未对齐访问异常、未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

指令格式：



BPUSH.W——二进制转译字压栈指令

统一化指令

语法	操作	编译结果

bpush.w rz	将寄存器 RZ 中的字存储到二进制转译堆栈存储器中，然后更新二进制转译堆栈寄存器到二进制转译堆栈存储器的顶端；
	if (BSP + 4 > TOP)
	R15 ← next PC
	PC ← SVBR - 12
	else
	MEM[BSP] ← RZ[31:0];
	BSP ← BSP + 4;
		仅存在 16 位指令
		bpush.w rz;

说明： 将二进制转译堆栈指针寄存器（BSP）加 4 的值与二进制转译栈顶寄存器（TOP）进行比较。如果 BSP 加 4 的值大于 TOP，则将子程序的返回地址（下一条指令的 PC）保存在链接寄存器 R15 中，程序转移到 SVBR-12 处执行；否则，将寄存器 RZ 中的字存储到二进制转译堆栈存储器中，然后更新 BSP 到二进制转译堆栈存储器的顶端。采用堆栈寄存器直接寻址方式。

影响标志位： 无影响

异常： 未对齐访问异常、未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

16位指令

操作： 将寄存器 RZ 中的字存储到二进制转译堆栈存储器中，然后更新二进制转译堆栈寄存器到二进制转译堆栈存储器的顶端；

if (BSP + 4 > TOP)

R15 ← next PC

PC ← SVBR - 12

else

MEM[BSP] ← RZ[31:0];

BSP ← BSP + 4;

语法： bpush16.w rz

说明： 将二进制转译堆栈指针寄存器（BSP）加 4 的值与二进制转译栈顶

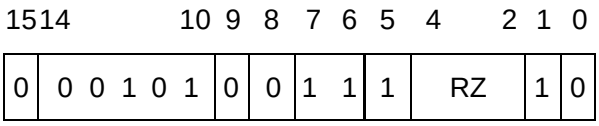
寄存器（TOP）进行比较。如果 BSP 加 4 的值大于 TOP，则将子程序的返回地址（下一条指令的 PC）保存在链接寄存器 R15 中，程序转移到 SVBR-12 处执行；否则，将寄存器 RZ 中的字存储到二进制转译堆栈存储器中，然后更新 BSP 到二进制转译堆栈存储器的顶端。采用堆栈寄存器直接寻址方式。

影响标志位： 无影响

限制： 寄存器的范围为 r0 – r7。

异常： 未对齐访问异常、未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

指令格式：



BMSET——BCTM 位置位指令

统一化指令

语法	操作	编译结果
bmset	设置状态寄存器的 BM 位。 PSR(BM) ← 1	仅存在 32 位指令。 bmset32

说明： PSR 的 BM 位被置位。

影响标志位： 无影响

异常： 无

注意： 该指令仅实现于支持二进制代码转译机制的CK802处理器。

32位指令

操作： 设置状态寄存器的 BM 位

PSR(BM) ← 1

语法： bmset32

说明： PSR 的 BM 位被置位。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 1 0 0	0 0 0 0 1	0 0 0 0 0

BR——无条件跳转指令

统一化指令

语法	操作	编译结果

br label	PC ← PC + sign_extend(offset << 1)	根据跳转的范围编译为对应的 16 位或 32 位指令
		if(offset<1KB), then
		br16 label;
		else
		br32 label;

说明： 程序无条件跳转到 label 处执行。

Label 由当前程序 PC 加上左移 1 位的相对偏移量有符号扩展到 32 位后的值得到。

影响标志位： 无影响

异常： 无

16位指令

操作： PC ← PC + sign_extend(offset << 1)

语法： br16 label

说明： 程序无条件跳转到 label 处执行。

Label 由当前程序 PC 加上左移 1 位的 10 位相对偏移量有符号扩展到 32 位后的值得到。BR16 指令的跳转范围是±1KB 地址空间。

影响标志位： 无影响

异常： 无

指令格式：

15	14 10	9 0
0	0 0 0 0 1	Offset

32位指令

操作： PC ← PC + sign_extend(offset << 1)

语法： br32 label

说明： 程序无条件跳转到 label 处执行。

Label 由当前程序 PC 加上左移 1 位的 16 位相对偏移量有符号扩展到 32 位后的值得到。BR 指令的跳转范围是±64KB 地址空间。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 0
1	1 1 0 1 0	0 0 0 0 0	0 0 0 0 0	Offset

BSETI——立即数位置位指令

统一化指令

语法	操作	编译结果

bseti rz, imm5	RZ ← RZ[IMM5]置位	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (z<8), then
		bseti16 rz, imm5；
		else
		bseti32 rz, rz, imm5；

bseti rz, rx, imm5	RZ ← RX[IMM5]置位	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if((x==z) and (z<8)，then
		bseti16 rz, imm5；
		else
		bseti32 rz, rx, imm5；

说明： 将 RZ/RX 的值中，由 IMM5 域值所指示的位置 1，其余位保持不变，把置位后的结果存入 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

16位指令

操作： RZ ← RZ[IMM5]置位

语法： bseti16 rz, imm5

说明： 将 RZ 的值中，由 IMM5 域值所指示的位置 1，其余位保持不变，把置位后的结果存入 RZ。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7；立即数的范围为 0-31。

异常： 无

指令格式：

15 14 10 8 7 5 4 0

0	0 1 1 1	RZ	1 0 1	IMM5

32位指令

操作： RZ ← RX[IMM5]置位

语法： bseti32　rz, rx, imm5

说明： 将 RX 的值中，由 IMM5 域值所指示的位置 1，其余位保持不变，把置位后的结果存入 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		IMM5		RX		0 0 1 0 1 0		0 0 0 1 0		RZ	

BSR——跳转到子程序指令

统一化指令

语法	操作	编译结果
bsr label	链接并跳转到子程序： R15 ← next PC PC ← PC + sign_extend(offset << 1)	仅存在于 32 位指令 bsr32 label

说明： 子程序跳转，将子程序的返回地址（下一条指令的 PC）保存在链接寄存器 R15 中，程序转移到 label 处执行。

Label 由当前程序 PC 加上左移 1 位的相对偏移量有符号扩展到 32 位后的值得到。

影响标志位： 无影响

异常： 无

32位指令

操作： 链接并跳转到子程序：

R15 ← PC+4

PC ← PC + sign_extend(offset << 1)

语法： bsr32 label

说明： 子程序跳转，将子程序的返回地址（下一条指令的 PC，即当前 PC+4）保存在链接寄存器 R15 中，程序转移到 label 处执行。

Label 由当前程序 PC 加上左移 1 位的 26 位相对偏移量有符号扩展到 32 位后的值得到。BSR 指令的跳转范围是±64MB 地址空间。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 0
1	1 1 0 0 0	Offset

BT——C 为 1 分支指令

统一化指令

语法	操作	编译结果

bt label	if(C == 1)
	PC ← PC + sign_extend(offset << 1);
	else
	PC ← next PC;
		根据跳转的范围编译为对应的 16 位或 32 位指令。
		if (offset<1KB)，then
		bt16 label;
		else
		bt32 label;

说明： 如果条件标志位 C 等于 1，则程序转移到 label 处执行；否则程序执行下一条指令。

Label 由当前程序 PC 加上左移 1 位的相对偏移量有符号扩展到 32 位后的值得到。BT 指令的转移范围是±64KB 地址空间。

影响标志位： 无影响

异常： 无

16位指令

操作： C 等于一则程序转移

if(C == 1)

PC ← PC + sign_extend(offset << 1)

else

PC ← PC + 2

语法： bt16 label

说明： 如果条件标志位 C 等于 1，则程序转移到 label 处执行；否则程序执行下一条指令，即 PC ← PC + 2。

Label 由当前程序 PC 加上左移 1 位的 10 位相对偏移量有符号扩展到 32 位后的值得到。BT16 指令的转移范围是±1KB 地址空间。

影响标志位： 无影响

异常： 无

指令格式：

15	14 10	9 0
0	0 0 0 1 0	Offset

32位指令

操作： C 等于一则程序转移

```
if(C == 1)
   PC ← PC + sign_extend(offset << 1)
else
   PC ← PC + 4
```

语法： bt32　label

说明： 如果条件标志位 C 等于 1，则程序转移到 label 处执行；否则程序执行下一条指令，即 PC ← PC + 4。

Label 由当前程序 PC 加上左移 1 位的 16 位相对偏移量有符号扩展到 32 位后的值得到。BT 指令的转移范围是±64KB 地址空间。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 0
1	1 1 0 1 0	0 0 0 1 1	0 0 0 0 0	Offset

BTSTI——立即数位测试指令

统一化指令

语法	操作	编译结果
btsti rx, imm5	C ← RX[IMM5]	仅存在 32 位指令 btsti32 rx, imm5

说明： 对由 IMM5 决定的 RX 的位（RX[IMM5]）进行测试，并使条件位 C 的值等于该位的值。

影响标志位： C ← RX[IMM5]

限制： 立即数的范围为 0-31。

异常： 无

16位指令

操作： C ← RX[IMM5]

语法： Btsti16 rx, imm5

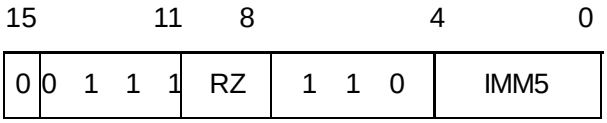
说明： 对由 IMM5 决定的 RX 的位（RX[IMM5]）进行测试，并使条件位 C 的值等于该位的值。

影响标志位： C ← RX[IMM5]

限制： 立即数的范围为 0-31。

异常： 无

指令格式：



32位指令

操作： C ← RX[IMM5]

语法： btsti32 rx, imm5

说明： 对由 IMM5 决定的 RX 的位（RX[IMM5]）进行测试，并使条件位 C 的值等于该位的值。

影响标志位： C ← RX[IMM5]

限制： 立即数的范围为 0-31。

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		IMM5		RX		0 0 1 0 1 0		0 0 1 0 0		0 0 0 0 0	

CMPHS——无符号大于等于比较指令

统一化指令

语法	操作	编译结果

cmphs rx, ry	RX与RY作无符号比较。
	If RX >= RY, then
	C ← 1;
	else
	C ← 0;
		仅存在 16 位指令 cmphs16 rx, ry

说明： 将RX的值减去RY的值，结果与0作比较，并对C位进行更新。cmphs 进行无符号比较，即操作数被认为是无符号数。如果 RX 大于等于 RY，即减法结果大于等于 0，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据比较结果设置条件位 C

异常： 无

16位指令

操作： RX与RY作无符号比较。

If RX >= RY, then

C ← 1;

else

C ← 0;

语法： cmphs16 rx, ry

说明： 将 RX 的值减去 RY 的值，结果与 0 作比较，并对 C 位进行更新。cmphs16 进行无符号比较，即操作数被认为是无符号数。如果 RX 大于等于 RY，即减法结果大于等于 0，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据比较结果设置条件位 C

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15 14 10 9 6 5 2 1 0

0	1 1 0 0 1	RY	RX	0 0

CMPHSI——立即数无符号大于等于比较指令

统一化指令

语法	操作	编译结果

cmphsi rx, oimm16	RX与立即数作无符号比较。
	If RX >= zero_extend(OIMM16),
	C ← 1;
	else
	C ← 0;
		根据立即数和寄存器的范围编译为对应的16位或32位指令。
		if (oimm16<33) and (x<8),then
		cmphsi16 rx, oimm5;
		else
		cmphsi32 rx, oimm16;

说明： 将带偏置 1 的 16 位立即数（OIMM16）零扩展至 32 位，然后用 RX 的值减去该 32 位值，结果与 0 作比较，并对 C 位进行更新。cmphsi 进行无符号比较，即操作数被认为是无符号数。如果 RX 大于等于零扩展后的 OIMM16，即减法结果大于等于 0，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据比较结果设置条件位 C

限制： 立即数的范围为 0x1-0x10000。

异常： 无

16位指令

操作： RX与立即数作无符号比较。

If RX >= zero_extend(OIMM5), then

C ← 1;

else

C ← 0;

语法： cmphsi16 rx, oimm5

说明： 将带偏置 1 的 5 位立即数（OIMM5）零扩展至 32 位，然后用 RX 的值减去该 32 位值，结果与 0 作比较，并对 C 位进行更新。cmphsi16 进行无符号比较，即操作数被认为是无符号数。如果 RX 大于等于零扩展后的 OIMM5，即减法结果大于等于 0，则设置条件位 C；否则，清除条件位 C。

注意：二进制操作数 IMM5 等于 OIMM5 - 1。

影响标志位： 根据比较结果设置条件位 C

限制： 寄存器的范围为 r0-r7；立即数的范围为 1-32。

异常： 无

指令格式：

15	14 10	8 7 5	4 0
0	0 1 1 1	RX	0 0 0	IMM5

IMM5 域——指定不带偏置立即数的值。

注意：参与比较的立即数 OIMM5 比起二进制操作数 IMM5 需偏置 1。

00000——与 1 比较

00001——与 2 比较

……

11111——与 32 比较

32位指令

操作： RX与立即数作无符号比较。

If RX >= zero_extend(OIMM16), then

C ← 1;

else

C ← 0;

语法： cmphsi32 rx, oimm16

说明： 将带偏置 1 的 16 位立即数（OIMM16）零扩展至 32 位，然后用 RX 的值减去该32位值，结果与0作比较，并对C位进行更新。cmphsi32 进行无符号比较，即操作数被认为是无符号数。如果 RX 大于等于零扩展后的 OIMM16，即减法结果大于等于 0，则设置条件位 C；否则，清除条件位 C。

注意：二进制操作数 IMM16 等于 OIMM16 - 1。

影响标志位： 根据比较结果设置条件位 C

限制： 立即数的范围为 0x1-0x10000。

异常： 无

指令格式：

31	30 26	25 21	20 16	15 0
1	1 1 0 1 0	1 1 0 0 0	RX	IMM16

IMM16 域——指定不带偏置立即数的值。

注意：参与比较的立即数 OIMM16 比起二进制操作数 IMM16 需偏置 1。

0000000000000000——与 0x1 比较

0000000000000001——与 0x2 比较

……

1111111111111111——与 0x10000 比较

CMPLT——有符号小于比较指令

统一化指令

语法	操作	编译结果

cmplt rx, ry	RX与RY作有符号比较。
	If RX < RY, then
	C ← 1;
	else
	C ← 0;
		仅存在 16 位指令 cmplt16 rx, ry

说明： 将 RX 的值减去 RY 的值，结果与 0 作比较，并对 C 位进行更新。cmplt 进行有符号比较，即操作数被认为是补码形式的有符号数。如果 RX 小于 RY，即减法结果小于 0，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据比较结果设置条件位 C

异常： 无

16位指令

操作： RX与RY作有符号比较。

If RX < RY, then

C ← 1;

else

C ← 0;

语法： cmplt16 rx, ry

说明： 将 RX 的值减去 RY 的值，结果与 0 作比较，并对 C 位进行更新。cmplt16 进行有符号比较，即操作数被认为是补码形式的有符号数。如果 RX 小于 RY，即减法结果小于 0，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据比较结果设置条件位 C

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15 14 10 9 6 5 2 1 0

0	1 1 0 0 1	RY	RX	0 1

CMPLTI——立即数有符号小于比较指令

统一化指令

语法	操作	编译结果

cmplti rx, oimm16	RX与立即数作有符号比较。
	If RX < zero_extend(OIMM16),
	C ← 1;
	else
	C ← 0;
		根据立即数和寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<8) and (oimm16<33), then
		cmplti16 rx, oimm5;
		else
		cmplti32 rx, oimm16;

说明： 将带偏置 1 的 16 位立即数（OIMM16）零扩展至 32 位，然后用 RX 的值减去该 32 位值，结果与 0 作比较，并对 C 位进行更新。cmplti 进行有符号比较，即 RX 的值被认为是补码形式的有符号数。如果 RX 小于零扩展后的 OIMM16，即减法结果小于 0，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据比较结果设置条件位 C

限制： 立即数的范围为 0x1-0x10000。

异常： 无

16位指令

操作： RX与立即数作有符号比较。

If RX < zero_extend(OIMM5), then

C ← 1;

else

C ← 0;

语法： cmplti16 rx, oimm5

说明： 将带偏置 1 的 5 位立即数（OIMM5）零扩展至 32 位，然后用 RX 的值减去该 32 位值，结果与 0 作比较，并对 C 位进行更新。cmplti16 进行有符号比较，即 RX 的值被认为是补码形式的有符号数。如果 RX 小于零扩展后的 OIMM5，即减法结果小于 0，则设置条件位 C；否则，清除条件位 C。

注意：二进制操作数 IMM5 等于 OIMM5 - 1。

影响标志位： 根据比较结果设置条件位 C

限制： 寄存器的范围为 r0-r7；立即数的范围为 1-32。

异常： 无

指令格式：

15	14	10	8	7	5	4	0
0	0 1 1 1	RX		0 0 1		IMM5	

IMM5 域——指定不带偏置立即数的值。

注意：参与比较的立即数 OIMM5 比起二进制操作数 IMM5 需偏置 1。

00000——与 1 比较

00001——与 2 比较

……

11111——与 32 比较

32位指令

操作： RX与立即数作有符号比较。

If RX < zero_extend(OIMM16), then

C ← 1;

else

C ← 0;

语法： cmplti32 rx, oimm16

说明： 将带偏置 1 的 16 位立即数（OIMM16）零扩展至 32 位，然后用 RX 的值减去该 32 位值，结果与 0 作比较，并对 C 位进行更新。cmplti32 进行有符号比较，即 RX 的值被认为是补码形式的有符号数。如果 RX 小于零扩展后的 OIMM16，即减法结果小于 0，则设置条件位 C；否则，清除条件位 C。

注意：二进制操作数 IMM16 等于 OIMM16 - 1。

影响标志位： 根据比较结果设置条件位 C

限制： 立即数的范围为 0x1-0x10000。

异常： 无

指令格式：

31	30	26	25	21	20	16	15	0
1	1 1 0 1 0		1 1 0 0 1		RX		IMM16	

IMM16 域——指定不带偏置立即数的值。

注意：参与比较的立即数 OIMM16 比起二进制操作数 IMM16 需偏置 1。

0000000000000000——与 0x1 比较

0000000000000001——与 0x2 比较

……

1111111111111111——与 0x10000 比较

CMPNE——不等比较指令

统一化指令

语法	操作	编译结果

cmpne rx, ry	RX与RY作比较。
	If RX != RY, then
	C ← 1;
	else
	C ← 0;
		仅存在 16 位指令
		cmpne16 rx, ry

说明： 将 RX 的值减去 RY 的值，结果与 0 作比较，并对 C 位进行更新。如果 RX 不等于 RY，即减法结果不等于 0，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据比较结果设置条件位 C

异常： 无

16位指令

操作： RX与RY作比较。

If RX != RY, then

C ← 1;

else

C ← 0;

语法： cmpne16 rx, ry

说明： 将 RX 的值减去 RY 的值，结果与 0 作比较，并对 C 位进行更新。如果 RX 不等于 RY，即减法结果不等于 0，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据比较结果设置条件位 C

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15 14 10 9 6 5 2 1 0

0	1 1 0 0 1	RY	RX	1 0

CMPNEI——立即数不等比较指令

统一化指令

语法	操作	编译结果

cmpnei rx,
imm16
	RX与立即数作比较。
	If RX != zero_extend(imm16),
	C ← 1;
	else
	C ← 0;
		根据立即数和寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<7) and (imm16<33), then
		cmpnei16 rx, imm5;
		else
		cmpnei32 rx, imm16;

说明： 将 RX 的值减去零扩展至 32 位的 16 位立即数的值，结果与 0 作比较，并对 C 位进行更新。如果 RX 不等于零扩展后的 IMM16，即减法结果不等于 0，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据比较结果设置条件位 C

限制： 立即数的范围为 0x0-0xFFFF。

异常： 无

16位指令

操作： RX与立即数作比较。

If RX != zero_extend(IMM5), then

C ← 1;

else

C ← 0;

语法： cmpnei16 rx, imm5

说明： 将 RX 的值减去零扩展至 32 位的 5 位立即数的值，结果与 0 作比较，并对 C 位进行更新。如果 RX 不等于零扩展后的 IMM5，即减法结果不等于 0，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据比较结果设置条件位 C

限制： 寄存器的范围为 r0-r7；

立即数的范围为 0-31。

异常： 无

指令格式：

15 14 10 8 7 5 4 0

0	0 1 1 1	RX	0 1 0	IMM5

32位指令

操作： RX与立即数作比较。

If RX != zero_extend(imm16), then

C ← 1;

else

C ← 0;

语法： cmpnei rx, imm16

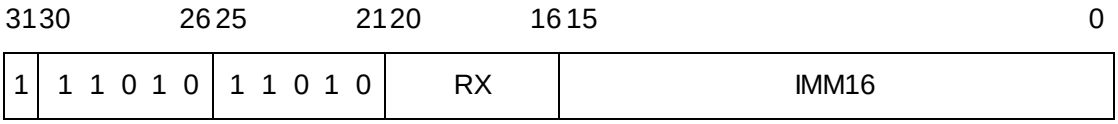
说明： 将 RX 的值减去零扩展至 32 位的 16 位立即数的值，结果与 0 作比较，并对 C 位进行更新。如果 RX 不等于零扩展后的 IMM16，即减法结果不等于 0，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据比较结果设置条件位 C

限制： 立即数的范围为 0x0-0xFFFF。

异常： 无

指令格式：



DECF——C 为 0 立即数减法指令

统一化指令

语法	操作	编译结果

decf　rz, rx, imm5	if C==0, then
	　RZ ← RX - zero_extend(IMM5);
	else
	　RZ ← RZ;
		仅存在 32 位指令
		decf32　rz, rx, imm5

说明： 如果条件位 C 为 0，将 5 位立即数零扩展至 32 位，用 RX 的值减去该 32 位值，把结果存在 RZ；否则，RZ 与 RX 的值不变。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

32位指令

操作：
```
if C==0, then
   RZ ← RX - zero_extend(IMM5);
else
   RZ ← RZ;
```

语法： decf32　rz, rx, imm5

说明： 如果条件位 C 为 0，将 5 位立即数零扩展至 32 位，用 RX 的值减去该 32 位值，把结果存在 RZ；否则，RZ 与 RX 的值不变。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		RZ		RX		0 0 0 0 1 1		0 0 1 0 0		IMM5	

DECT——C 为 1 立即数减法指令

统一化指令

语法	操作	编译结果

dect rz, rx, imm5	if C==1, then
	RZ ← RX - zero_extend(IMM5);
	else
	RZ ← RZ;
		仅存在 32 位指令
		dect32 rz, rx, imm5

说明： 如果条件位 C 为 1，将 5 位立即数零扩展至 32 位，用 RX 的值减去该 32 位值，把结果存在 RZ；否则，RZ 与 RX 的值不变。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

32位指令

操作：
if C==1, then
RZ ← RX - zero_extend(IMM5);
else
RZ ← RZ;

语法： dect32 rz, rx, imm5

说明： 如果条件位 C 为 1，将 5 位立即数零扩展至 32 位，用 RX 的值减去该 32 位值，把结果存在 RZ；否则，RZ 与 RX 的值不变。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	RZ	RX	0 0 0 0 1 1	0 1 0 0 0	IMM5

DOZE——进入低功耗睡眠模式指令

统一化指令

语法	操作	编译结果
doze	进入低功耗睡眠模式	仅存在 32 位指令 doze32

说明： 此指令使处理器进入低功耗睡眠模式，并等待一个中断来退出这个模式。此时，CPU 时钟停止，相应的外围设备也被停止。

影响标志位： 不影响

异常： 特权违反异常

32位指令

操作： 进入低功耗睡眠模式

语法： doze32

属性： 特权指令

说明： 此指令使处理器进入低功耗睡眠模式，并等待一个中断来退出这个模式。此时，CPU 时钟停止，相应的外围设备也被停止。

影响标志位： 不影响

异常： 特权违反异常

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	10000	00000	00000	010100	00001	00000

FF0——快速找 0 指令

统一化指令

语法	操作	编译结果
ff0　rz, rx	RZ ← find_first_0(RX);	仅存在 32 位指令 ff0.32　rz, rx

说明： 查找 RX 第一个为 0 的位，并把查找结果返回到 RZ。查找顺序是从 RX 的最高位扫描到最低位。如果 RX 的最高位（RX[31]）为 0，返回 RZ 的值为 0。如果在 RX 没有为 0 的位，返回 RZ 的值为 32。

影响标志位： 无影响

异常： 无

32位指令

操作： RZ ← find_first_0(RX);

语法： ff0.32　rz, rx

说明： 查找 RX 第一个为 0 的位，并把查找结果返回到 RZ。查找顺序是从 RX 的最高位扫描到最低位。如果 RX 的最高位（RX[31]）为 0，返回 RZ 的值为 0。如果在 RX 没有为 0 的位，返回 RZ 的值为 32。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	0 0 0 0 0	RX	0 1 1 1 1 1	0 0 0 0 1	RZ

FF1——快速找 1 指令

统一化指令

语法	操作	编译结果
ff1　rz, rx	RZ ← find_first_1(RX);	仅存在 32 位指令 ff1.32　rz, rx

说明： 查找 RX 第一个为 1 的位，并把查找结果返回到 RZ。查找顺序是从 RX 的最高位扫描到最低位。如果 RX 的最高位（RX[31]）为 1，返回 RZ 的值为 0。如果在 RX 没有为 1 的位，返回 RZ 的值为 32。

影响标志位： 无影响

异常： 无

32位指令

操作： RZ ← find_first_1(RX);

语法： ff1.32　rz, rx

说明： 查找 RX 第一个为 1 的位，并把查找结果返回到 RZ。查找顺序是从 RX 的最高位扫描到最低位。如果 RX 的最高位（RX[31]）为 1，返回 RZ 的值为 0。如果在 RX 没有为 1 的位，返回 RZ 的值为 32。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	0 0 0 0 0	RX	0 1 1 1 1 1	0 0 0 1 0	RZ

INCF——C 为 0 立即数加法指令

统一化指令

语法	操作	编译结果

incf　rz, rx, imm5	if C==0, then
	　RZ ← RX + zero_extend(IMM5);
	else
	　RZ ← RZ;
		仅存在 32 位指令
		incf32　rz, rx, imm5

说明： 如果条件位 C 为 0，将 5 位立即数零扩展至 32 位，用该 32 位值加 RX 的值，把结果存在 RZ；否则，RZ 与 RX 的值不变。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

32位指令

操作： if C==0, then
　RZ ← RX + zero_extend(IMM5);
else
　RZ ← RZ;

语法： incf32　rz, rx, imm5

说明： 如果条件位 C 为 0，将 5 位立即数零扩展至 32 位，用该 32 位值加 RX 的值，把结果存在 RZ；否则，RZ 与 RX 的值不变。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	RZ	RX	0 0 0 0 1 1	0 0 0 0 1	IMM5

INCT——C 为 1 立即数加法指令

统一化指令

语法	操作	编译结果

inct rz, rx, imm5	if C==1, then
	RZ ← RX + zero_extend(IMM5);
	else
	RZ ← RZ;
		仅存在 32 位指令
		inct32 rz, rx, imm5

说明： 如果条件位 C 为 1，将 5 位立即数零扩展至 32 位，用该 32 位值加 RX 的值，把结果存在 RZ；否则，RZ 与 RX 的值不变。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

32位指令

操作：
if C==1, then
RZ ← RX + zero_extend(IMM5);
else
RZ ← RZ;

语法： inct32 rz, rx, imm5

说明： 如果条件位 C 为 1，将 5 位立即数零扩展至 32 位，用该 32 位值加 RX 的值，把结果存在 RZ；否则，RZ 与 RX 的值不变。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	RZ	RX	0 0 0 0 1 1	0 0 0 1 0	IMM5

IPUSH——中断压栈指令

统一化指令

语法	操作	编译结果
ipush	将中断的通用寄存器现场{R0~R3, R12, R13 }存储到堆栈存储器中，然后更新堆栈指针寄存器到堆栈存储器的顶端； MEM[SP-4]~MEM[SP-24] ←{R13,R12,R3~R0}; SP←SP-24;	仅存在 16 位指令 ipush16

说明： 将中断的通用寄存器现场{ R0~R3, R12, R13 }保存到堆栈存储器中，然后更新堆栈指针寄存器到堆栈存储器的顶端。采用堆栈指针寄存器直接寻址方式。

影响标志位： 无影响

异常： 访问错误异常、未对齐异常

16位指令

操作： 将中断的通用寄存器现场{R0~R3, R12, R13 }存储到堆栈存储器中，然后更新堆栈指针寄存器到堆栈存储器的顶端；
MEM[SP-4]~MEM[SP-24] ←{R13,R12,R3~R0};
SP←SP-24;

语法： IPUSH16

属性 无

说明： 将中断的通用寄存器现场{ R0~R3, R12, R13 }保存到堆栈存储器中，然后更新堆栈指针寄存器到堆栈存储器的顶端。采用堆栈指针寄存器直接寻址方式。

影响标志位： 无影响

异常： 访问错误异常、未对齐异常

指令格式：

15	14 10	9	8	7 5	4 0
0	0 0 1 0 1	0	0	0 1 1	0 0 0 1 0

IPOP——中断出栈指令

统一化指令

语法	操作	编译结果
ipop	从堆栈指针寄存器指向堆栈中载入中断的通用寄存器现场{R0~R3, R12, R13 }，然后更新堆栈指针寄存器到堆栈存储器的顶端； {R0~R3,R12,R13} ← MEM[SP]~MEM[SP+20]; SP←SP+24;	仅存在 16 位指令 ipop16

说明： 从堆栈指针寄存器指向堆栈中载入中断的通用寄存器现场{R0~R3, R12, R13 }，然后更新堆栈指针寄存器到堆栈存储器的顶端。采用堆栈指针寄存器直接寻址方式。

影响标志位： 无影响

异常： 访问错误异常、未对齐异常

16位指令

操作： 从堆栈指针寄存器指向堆栈中载入中断的通用寄存器现场{R0~R3, R12, R13 }，然后更新堆栈指针寄存器到堆栈存储器的顶端；
{R0~R3,R12,R13}← MEM[SP]~MEM[SP+20];
SP←SP+24;

语法： IPOP16

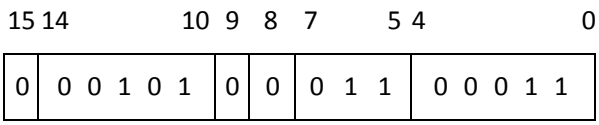
属性 无

说明： 从堆栈指针寄存器指向堆栈中载入中断的通用寄存器现场{R0~R3, R12, R13 }，然后更新堆栈指针寄存器到堆栈存储器的顶端。采用堆栈指针寄存器直接寻址方式。

影响标志位： 无影响

异常： 访问错误异常、未对齐异常

指令格式：



IXH——索引半字指令

统一化指令

语法	操作	编译结果
ixh rz, rx, ry	RZ ← RX + (RY << 1)	仅存在 32 位指令 ixh32 rz, rx, ry

说明： 将 RY 的值左移一位后加上 RX 的值，并把结果存入 RZ。

影响标志位： 无影响

异常： 无

32位指令

操作： RZ ← RX + (RY << 1)

语法： ixh32 rz, rx, ry

说明： 将 RY 的值左移一位后加上 RX 的值，并把结果存入 RZ。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	RY	RX	0 0 0 0 1 0	0 0 0 0 1	RZ

IXW——索引字指令

统一化指令

语法	操作	编译结果
ixw rz, rx, ry	RZ ← RX + (RY << 2)	仅存在 32 位指令 ixw32 rz, rx, ry

说明： 将 RY 的值左移两位后加上 RX 的值，并把结果存入 RZ。

影响标志位： 无影响

异常： 无

32位指令

操作： RZ ← RX + (RY << 2)

语法： ixw32 rz, rx, ry

说明： 将 RY 的值左移两位后加上 RX 的值，并把结果存入 RZ。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	RY	RX	0 0 0 0 1 0	0 0 0 1 0	RZ

JMP——寄存器跳转指令

统一化指令

语法	操作	编译结果
jmp　rx	跳转到寄存器指定的位置 PC ← RX & 0xfffffffe	仅存在 16 位指令。 jmp16　rx

说明： 程序跳转到寄存器 RX 指定的位置，RX 的最低位被忽略。JMP 指令的跳转范围是全部 4GB 地址空间。

影响标志位： 无影响

异常： 无

16位指令

操作： 跳转到寄存器指定的位置

PC ← RX & 0xfffffffe

语法： jmp16　rx

说明： 程序跳转到寄存器 RX 指定的位置，RX 的最低位被忽略。JMP 指令的跳转范围是全部 4GB 地址空间。

影响标志位： 无影响

异常： 无

指令格式：

15	14 10	9 6	5 2	1 0
0	1 1 1 1 0	0 0 0 0	RX	0 0

JMPIX——寄存器索引跳转指令

统一化指令

语法	操作	编译结果
jmpix rx, imm	跳转到寄存器索引指定的位置 PC ← SVBR + (RX & 0xff) * IMM	仅存在 16 位指令。 jmpix16 rx, imm;

说明： 程序跳转到 SVBR + RX[7:0] * IMM 的位置，IMM ∈ {16, 24, 32, 40}。RX 的高 24 位被忽略。

影响标志位： 无影响

异常： 无

注意： 该指令仅实现于支持二进制代码转译机制的CK802处理器。

16位指令

操作： 跳转到寄存器索引指定的位置

PC ← SVBR + (RX & 0xff) * IMM

语法： jmpix16 rx, imm

说明： 程序跳转到 SVBR + RX[7:0] * IMM 的位置，IMM ∈ {16, 24, 32, 40}。RX 的高 24 位被忽略。

影响标志位： 无影响

异常： 无

指令格式：

15 14 11 10 8 7 2 0

0	0 1 1 1	RX	1 1 1 0 0 0	IMM2

IMM2 域——指定立即数的值。

注意：二进制编码的 IMM2 值与此跳转指令中 IMM 值的对应关系如下：

2’b00——乘 16

2’b01——乘 24

2’b10——乘 32

2’b11——乘 40

JSR——寄存器跳转到子程序指令

统一化指令

语法	操作	编译结果
jsr　rx	链接并跳转到寄存器指定的子程序位置 R15 ← PC + 4， PC ← RX & 0xfffffffe	仅存在 16 位指令。 jsr16　rx

说明： 子程序寄存器跳转，将子程序的返回地址（下一条指令的 PC，即当前 PC+4）保存在链接寄存器 R15 中，程序跳转到寄存器 RX 的内容指定的子程序位置处执行，RX 的最低位被忽略。JSR 指令的跳转范围是全部 4GB 地址空间。

影响标志位： 无影响

异常： 无

16位指令

操作： 链接并跳转到寄存器指定的子程序位置

R15 ← PC + 4，PC ← RX & 0xfffffffe

语法： jsr16　rx

说明： 子程序寄存器跳转，将子程序的返回地址（下一条指令的 PC，即当前 PC+4）保存在链接寄存器 R15 中，程序跳转到寄存器 RX 的内容指定的子程序位置处执行，RX 的最低位被忽略。JSR 指令的跳转范围是全部 4GB 地址空间。

影响标志位： 无影响

异常： 无

指令格式：

15	14 10	9 6	5 2	1 0
0	1 1 1 1 0	0 0 0 0	RX	0 1

LD.B——无符号扩展字节加载指令

统一化指令

语法	操作	编译结果

ld.b rz,(rx, disp)	RZ ← zero_extend(MEM[RX + zero_extend(offset)])	根据偏移量和寄存器的范围编译为对应的 16 位或 32 位指令。
		if (disp<32)and(x<7) and (z<7), then
		ld16.b rz, (rx, disp);
		else
		ld32.b rz, (rx, disp);

说明： 从存储器加载的字节经零扩展到 32 位后存放于寄存器 RZ 中。采用寄存器加立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上 12 位相对偏移量无符号扩展到 32 位后的值得到。LD.B 指令可以寻址+4KB 地址空间。

注意，偏移量 DISP 即二进制操作数 Offset。

影响标志位： 无影响

异常： 访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

16位指令

操作： 从存储器加载字节到寄存器，无符号扩展

RZ ← zero_extend(MEM[RX + zero_extend(offset)])

语法： ld16.b rz, (rx, disp)

说明： 从存储器加载的字节经零扩展到 32 位后存放于寄存器 RZ 中。采用寄存器加立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上 5 位相对偏移量无符号扩展到 32 位后的值得到。LD16.B 指令可以寻址+32B 的地址空间。

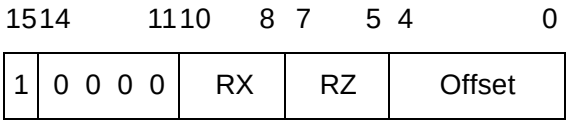
注意，偏移量 DISP 即二进制操作数 Offset。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7。

异常： 访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：



32位指令

操作： 从存储器加载字节到寄存器，无符号扩展

RZ ← zero_extend(MEM[RX + zero_extend(offset)])

语法： ld32.b rz, (rx, disp)

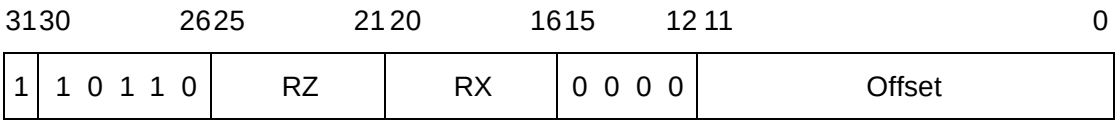
说明： 从存储器加载的字节经零扩展到 32 位后存放于寄存器 RZ 中。采用寄存器加立即数偏移量的寻址方式。存储器的有效地址由基址寄存器RX加上12位相对偏移量无符号扩展到32位后的值得到。LD32.B指令可以寻址+4KB 地址空间。

注意，偏移量 DISP 即二进制操作数 Offset。

影响标志位： 无影响

异常： 访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：



LD.BS——有符号扩展字节加载指令

统一化指令

语法	操作	编译结果
ld.bs rz, (rx, disp)	RZ ← sign_extend(MEM[RX + zero_extend(offset)])	仅存在 32 位指令。 ld32.bs rz, (rx, disp)

说明： 从存储器加载的字节经有符号扩展到 32 位后存放于寄存器 RZ 中。采用寄存器加立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上 12 位相对偏移量无符号扩展到 32 位后的值得到。LD.BS 指令可以寻址+4KB 地址空间。

注意，偏移量 DISP 即二进制操作数 Offset。

影响标志位： 无影响

异常： 访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

32位指令

操作： 从存储器加载字节到寄存器，有符号扩展

RZ ← sign_extend(MEM[RX + zero_extend(offset)])

语法： ld32.bs rz, (rx, disp)

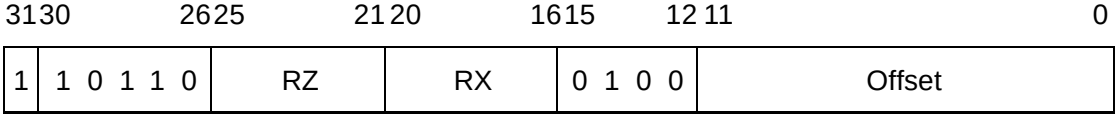
说明： 从存储器加载的字节经有符号扩展到 32 位后存放于寄存器 RZ 中。采用寄存器加立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上 12 位相对偏移量无符号扩展到 32 位后的值得到。LD32.BS 指令可以寻址+4KB 地址空间。

注意，偏移量 DISP 即二进制操作数 Offset。

影响标志位： 无影响

异常： 访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：



LD.H——无符号扩展半字加载指令

统一化指令

语法	操作	编译结果

ld.h rz, (rx, disp)	RZ ← zero_extend(MEM[RX + zero_extend(offset << 1)])	根据偏移量和寄存器的范围编译为对应的 16 位或 32 位指令。
		if (disp<64)and(x<7) and (z<7), then
		ld16.h rz, (rx, disp);
		else
		ld32.h rz, (rx, disp);

说明： 从存储器加载的半字经零扩展到 32 位后存放于寄存器 RZ 中。采用寄存器加立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上左移 1 位的 12 位相对偏移量无符号扩展到 32 位后的值得到。LD.H 指令可以寻址+8KB 地址空间。

注意，偏移量 DISP 是二进制操作数 Offset 左移 1 位得到的。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

16位指令

操作： 从存储器加载半字到寄存器，无符号扩展

RZ ← zero_extend(MEM[RX + zero_extend(offset << 1)])

语法： ld16.h rz, (rx, disp)

说明： 从存储器加载的半字经零扩展到 32 位后存放于寄存器 RZ 中。采用寄存器加立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上左移 1 位的 5 位相对偏移量无符号扩展到 32 位后的值得到。LD16.H 指令可以寻址+64B 的地址空间。

注意，偏移量 DISP 是二进制操作数 Offset 左移 1 位得到的。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7。

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：

1514 1110 8 7 5 4 0

1	0 0 0 1	RX	RZ	Offset

32位指令

操作： 从存储器加载半字到寄存器，无符号扩展

RZ ← zero_extend(MEM[RX + zero_extend(offset << 1)])

语法： ld32.h rz, (rx, disp)

说明： 从存储器加载的半字经零扩展到 32 位后存放于寄存器 RZ 中。采用寄存器加立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上左移 1 位的 12 位相对偏移量无符号扩展到 32 位后的值得到。LD32.H 指令可以寻址+8KB 地址空间。

注意，偏移量 DISP 是二进制操作数 Offset 左移 1 位得到的。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：

31 30 26 25 21 20 16 15 12 11 0

1	1 0 1 1 0	RZ	RX	0 0 0 1	Offset

LD.HS——有符号扩展半字加载指令

统一化指令

语法	操作	编译结果
ld.hs rz, (rx, disp)	RZ ← sign_extend(MEM[RX + zero_extend(offset << 1)])	仅存在 32 位指令。 ld32.hs rz, (rx, disp)

说明： 从存储器加载的半字经有符号扩展到 32 位后存放于寄存器 RZ 中。采用寄存器加立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上左移 1 位的 12 位相对偏移量无符号扩展到 32 位后的值得到。LD.HS 指令可以寻址+8KB 地址空间。

注意，偏移量 DISP 是二进制操作数 Offset 左移 1 位得到的。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

32位指令

操作： 从存储器加载半字到寄存器，有符号扩展

RZ ← sign_extend(MEM[RX + zero_extend(offset << 1)])

语法： ld32.hs rz, (rx, disp)

说明： 从存储器加载的半字经有符号扩展到 32 位后存放于寄存器 RZ 中。采用寄存器加立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上左移 1 位的 12 位相对偏移量无符号扩展到 32 位后的值得到。LD32.HS 指令可以寻址+8KB 地址空间。

注意，偏移量 DISP 是二进制操作数 Offset 左移 1 位得到的。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	1	0	RZ					RX					0	1	0	1	Offset											

LD.W——字加载指令

统一化指令

语法	操作	编译结果

ld.w rz, (rx, disp)	RZ ← MEM[RX + zero_extend(offset << 2)]	根据偏移量和寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x=sp) and (z<7) and (disp < 1024),
		ld16.w rz, (sp, disp);
		else if (disp<128) and (x<7) and (z<7),
		ld16.w rz, (rx, disp);
		else
		ld32.w rz, (rx, disp);

说明： 从存储器加载字到寄存器 RZ 中。采用寄存器加立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上左移两位的 12 位相对偏移量无符号扩展到 32 位后的值得到。LD.W 指令可以寻址+16KB 地址空间。

注意，偏移量 DISP 是二进制操作数 Offset 左移两位得到的。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

16位指令

操作： 从存储器加载字到寄存器

RZ ← MEM[RX + sign_extend(offset << 2)]

语法： ld16.w rz, (rx, disp)

ld16.w rz, (sp, disp)

说明： 从存储器加载字到寄存器 RZ 中。采用寄存器加立即数偏移量的寻址方式。当 RX 为 SP 时，存储器的有效地址由基址寄存器 RX 加上左移 2 位的 8 位相对偏移量无符号扩展到 32 位后的值得到。当 rx 为其它寄存器时，存储器的有效地址由基址寄存器 RX 加上左移 2 位的 5 位相对偏移量无符号扩展到 32 位后的值得到。LD16.W 指令可以寻址+1KB 的地址空间。

注意，偏移量 DISP 是二进制操作数 IMM5 左移两位得到的。当基

址寄存器RX为SP时，偏移量DISP是二进制操作数{IMM3, IMM5}左移两位得到的。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7。

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：

ld16.w rz, (rx, disp)

15	14 11	10 8	7 5	4 0
1	0 0 1 0	RX	RZ	IMM5

ld16.w rz, (sp, disp)

15	14 11	10 8	7 5	4 0
1	0 0 1 1	IMM3	RZ	IMM5

32位指令

操作： 从存储器加载字到寄存器

RZ ← MEM[RX + zero_extend(offset << 2)]

语法： ld32.w rz, (rx, disp)

说明： 从存储器加载字到寄存器 RZ 中。采用寄存器加立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上左移两位的 12 位相对偏移量无符号扩展到 32 位后的值得到。LD32.W 指令可以寻址+16KB 地址空间。

注意，偏移量 DISP 是二进制操作数 Offset 左移两位得到的。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：

31	30 26	25 21	20 16	15 12	11 0
1	1 0 1 1 0	RZ	RX	0 0 1 0	Offset

LDM——连续多字加载指令

统一化指令

语法	操作	编译结果

ldm ry-rz, (rx)	从存储器加载连续的多个字到一片连续的寄存器堆中
	dst ← Y; addr ← RX;
	for (n = 0; n <= (Z-Y); n++){
	Rdst ← MEM[addr];
	dst ← dst + 1;
	addr ← addr + 4;
	}
		仅存在 32 位指令。
		ldm32 ry-rz, (rx);

说明： 从存储器依次加载连续的多个字到寄存器 RY 开始的一片连续寄存器堆中，即将存储器指定地址开始的第一个字加载到寄存器 RY 中，第二个字加载到寄存器 RY+1 中，依次类推，最后一个字加载到寄存器 RZ 中。存储器的有效地址由基址寄存器 RX 的内容决定。

影响标志位： 无影响

限制： RZ 应当大于等于 RY。

RY-RZ 范围内不应该包含基址寄存器 RX，否则结果不可预测。

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

32位指令

操作： 从存储器加载连续的多个字到一片连续的寄存器堆中

dst ← Y; addr ← RX;

for (n = 0; n <= IMM5; n++){

Rdst ← MEM[addr];

dst ← dst + 1;

addr ← addr + 4;

}

语法： ldm32 ry-rz, (rx)

说明： 从存储器依次加载连续的多个字到寄存器 RY 开始的一片连续寄存器堆中，即将存储器指定地址开始的第一个字加载到寄存器 RY 中，第二个字加载到寄存器 RY+1 中，依次类推，最后一个字加载到寄存器 RZ 中。存储器的有效地址由基址寄存器 RX 的内容决定。

影响标志位： 无影响

限制： RZ 应当大于等于 RY。

RY-RZ 范围内不应该包含基址寄存器 RX，否则结果不可预测。

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 1 0 0	RY	RX	0 0 0 1 1 1	0 0 0 0 1	IMM5

IMM5 域——指定目标寄存器的个数，IMM5 = Z - Y。

00000——1 个目的寄存器

00001——2 个目的寄存器

……

11111——32 个目的寄存器

LDQ——连续四字加载指令#

统一化指令

语法	操作	编译结果

ldq　r4-r7, (rx)	从存储器加载连续的四个字到寄存器 R4—R7 中
	dst ← 4; addr ← RX;
	for (n = 0; n <= 3; n++){
	　Rdst ← MEM[addr];
	　dst ← dst + 1;
	　addr ← addr + 4;
	}
		仅存在 32 位指令。
		ldq32　r4-r7, (rx);

说明： 从存储器依次加载连续的 4 个字到寄存器堆[R4, R7]（包括边界）中，即将存储器指定地址开始的第一个字加载到寄存器 R4 中，第二个字加载到寄存器 R5 中，第三个字加载到寄存器 R6 中，第四个字加载到寄存器 R7 中。存储器的有效地址由基址寄存器 RX 的内容决定。注意，该指令是 ldm r4-r7, (rx)的伪指令。

影响标志位： 无影响

限制： R4-R7 范围内不应该包含基址寄存器 RX，否则结果不可预测。

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

32位指令

操作： 从存储器加载连续的四个字到寄存器 R4—R7 中

```
dst ← 4; addr ← RX;
for (n = 0; n <= 3; n++){
   Rdst ← MEM[addr];
   dst ← dst + 1;
   addr ← addr + 4;
}
```

语法： ldq32　r4-r7, (rx)

说明： 从存储器依次加载连续的 4 个字到寄存器堆[R4, R7]（包括边界）中，即将存储器指定地址开始的第一个字加载到寄存器 R4 中，第二个字加载到寄存器 R5 中，第三个字加载到寄存器 R6 中，第四

个字加载到寄存器 R7 中。存储器的有效地址由基址寄存器 RX 的内容决定。

注意，该指令是 ldm32 r4-r7, (rx)的伪指令。

影响标志位： 无影响

限制： R4-R7 范围内不应该包含基址寄存器 RX，否则结果不可预测。

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 1 0 0	0 0 1 0 0	RX	0 0 0 1 1 1	0 0 0 0 1	0 0 0 1 1

LRW——存储器读入指令

统一化指令

语法	操作	编译结果

lrw rz, label
lrw rz, imm32
	从存储器加载字到寄存器
	RZ ← zero_extend(MEM[(PC + zero_extend(offset << 2)) & 0xfffffffc])
		根据加载的范围编译为对应的 16 位或 32 位指令
		if(offset<512B), then
		lrw16 label;
		lrw16 imm32;
		else
		lrw32 label;
		lrw32 imm32;

说明： 加载 label 所在位置的字，或 32 位立即数（IMM32）至目的寄存器 RZ。存储器地址根据 PC 加左移两位的相对偏移量，并无符号扩展到 32 位后，再经最低两位强制清零得到。LRW 指令的加载范围是全部 4GB 地址空间。

影响标志位： 无影响

异常： 访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

16位指令

操作： 从存储器加载字到寄存器

RZ ← zero_extend(MEM[(PC + zero_extend(offset << 2)) & 0xfffffffc])

语法： lrw16 rz, label

lrw16 rz, imm32

说明： 加载 label 所在位置的字，或 32 位立即数（IMM32）至目的寄存器 RZ。存储器地址根据 PC 加左移两位的 10 位相对偏移量，并无符号扩展到 32 位后，再经最低两位强制清零得到。LRW 指令的加载范围是全部 4GB 地址空间。

注意，若 IMM1 为 1，相对偏移量 Offset 等于二进制编码{0，{IMM2，IMM5}}。

若 IMM1 为 0， 相对偏移量 Offset 等于二进制编码{1，{！{IMM2，IMM5}}}。

IMM1,IMM2,IMM5,RZ 同时为零则该指令为 BKPT 指令，并非 LRW16。

影响标志位： 无影响

异常： 访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：

15	14 13	12	11 10	9 8	7 5	4 0
0	00	IMM1	0 0	IMM2	RZ	IMM5

32位指令

操作： 从存储器加载字到寄存器

RZ ← zero_extend(MEM[(PC + zero_extend(offset << 2)) & 0xfffffffc])

语法： lrw32 rz, label

lrw32 rz, imm32

说明： 加载 label 所在位置的字，或 32 位立即数（IMM32）至目的寄存器 RZ。存储器地址根据 PC 加左移两位的 16 位相对偏移量，并无符号扩展到 32 位后，再经最低两位强制清零得到。LRW 指令的加载范围是全部 4GB 地址空间。

影响标志位： 无影响

异常： 访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：

31	30 26	25 21	20 16	15 0
1	1 1 0 1 0	1 0 1 0 0	RZ	Offset

LSL——逻辑左移指令

统一化指令

语法	操作	编译结果

lsl　rz, rx	RZ ← RZ << RX[5:0]	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<16) and (z<16), then
		　lsl16　rz, rx;
		else
		　lsl32　rz, rz, rx;

lsl　rz, rx, ry	RZ ← RX << RY[5:0]	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x==z) and (y<16) and (z<16), then
		　lsl16　rz, ry
		else
		　lsl32　rz, rx, ry

说明： 对于 lsl　rz, rx，将 RZ 的值进行逻辑左移（原值左移，右侧移入 0），结果存入 RZ，左移位数由 RX 低 6 位（RX[5:0]）的值确定；如果 RX[5:0]的值大于 31，那么 RZ 将被清零；

对于 lsl　rz, rx, ry，将 RX 的值进行逻辑左移（原值左移，右侧移入 0），结果存入 RZ，左移位数由 RY 低 6 位（RY[5:0]）的值确定；如果 RY[5:0]的值大于 31，那么 RZ 将被清零。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← RZ << RX[5:0]

语法： lsl16　rz, rx

说明： 将 RZ 的值进行逻辑左移（原值左移，右侧移入 0），结果存入 RZ，左移位数由 RX 低 6 位（RX[5:0]）的值确定；如果 RX[5:0]的值大于 31，那么 RZ 将被清零。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15	14	10	9	6	5	2	1	0
0	1 1 1 0 0		RZ		RX		0 0	

32位指令

操作： RZ ← RX << RY[5:0]

语法： lsl32 rz, rx, ry

说明： 将 RX 的值进行逻辑左移（原值左移，右侧移入 0），结果存入 RZ，左移位数由 RY 低 6 位（RY[5:0]）的值确定；如果 RY[5:0]的值大于 31，那么 RZ 将被清零。

影响标志位： 无影响

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		RY		RX		0 1 0 0 0 0		0 0 0 0 1		RZ	

LSLC——立即数逻辑左移至 C 位指令

统一化指令

语法	操作	编译结果
lslc rz, rx, oimm5	RZ ← RX << OIMM5, C ← RX[32 – OIMM5]	仅存在 32 位指令。 lslc32 rz, rx, oimm5

说明： 将 RX 的值进行逻辑左移（原值左移，右侧移入 0），把移出最末位存入条件位 C，移位结果存入 RZ，左移位数由带偏置 1 的 5 位立即数（OIMM5）的值决定。如果 OIMM5 的值等于 32，那么条件位 C 为 RX 的最低位，RZ 被清零。

影响标志位： C ← RX[32 – OIMM5]

限制： 立即数的范围为 1-32。

异常： 无

32位指令

操作： RZ ← RX << OIMM5, C ← RX[32 – OIMM5]

语法： lslc32 rz, rx, oimm5

说明： 将 RX 的值进行逻辑左移（原值左移，右侧移入 0），把移出最末位存入条件位 C，移位结果存入 RZ，左移位数由带偏置 1 的 5 位立即数（OIMM5）的值决定。如果 OIMM5 的值等于 32，那么条件位 C 为 RX 的最低位，RZ 被清零。

注意：二进制操作数 IMM5 等于 OIMM5 - 1。

影响标志位： C ← RX[32 – OIMM5]

限制： 立即数的范围为 1-32。

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		IMM5		RX		0 1 0 0 1 1		0 0 0 0 1		RZ	

IMM5 域——指定不带偏置立即数的值。

注意：移位的值 OIMM5 比起二进制操作数 IMM5 需偏置 1。

00000——移 1 位

00001——移 2 位

……

11111——移 32 位

LSLI——立即数逻辑左移指令

统一化指令

语法	操作	编译结果

lsli rz, rx, imm5	RZ ← RX << IMM5	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<8) and (z<8), then
		lsli16 rz, rx, imm5
		else
		lsli32 rz, rx, imm5

说明： 将 RX 的值进行逻辑左移（原值左移，右侧移入 0），结果存入 RZ，左移位数由 5 位立即数（IMM5）的值决定；如果 IMM5 的值等于 0，那么 RZ 的值将不变。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

16位指令

操作： RZ ← RX << IMM5

语法： lsli16 rz, rx, imm5

说明： 将 RX 的值进行逻辑左移（原值左移，右侧移入 0），结果存入 RZ，左移位数由 5 位立即数（IMM5）的值决定；如果 IMM5 的值等于 0，那么 RZ 的值将不变。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7；
立即数的范围为 0-31。

异常： 无

指令格式：

15 14 11 10 8 7 5 4 0

0	1 0 0 0	RX	RZ	IMM5

32位指令

操作： RZ ← RX << IMM5

语法： lsli32　rz, rx, imm5

说明： 将 RX 的值进行逻辑左移（原值左移，右侧移入 0），结果存入 RZ，左移位数由 5 位立即数（IMM5）的值决定；如果 IMM5 的值等于 0，那么 RZ 的值将与 RX 相同。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	IMM5	RX	0 1 0 0 1 0	0 0 0 0 1	RZ

LSR——逻辑右移指令

统一化指令

语法	操作	编译结果

lsr　rz, rx	RZ ← RZ >> RX[5:0]	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (z<16) and (x<16), then
		　lsr16　rz, rx;
		else
		　lsr32　rz, rz, rx;

lsr　rz, rx, ry	RZ ← RX >> RY[5:0]	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x==z) and (z<16) and (y<16), then
		　lsr16　rz, ry;
		else
		　lsr32　rz, rx, ry;

说明： 对于 lsr　rz, rx，将 RZ 的值进行逻辑右移（原值右移，左侧移入 0），结果存入 RZ，右移位数由 RX 低 6 位（RX[5:0]）的值确定；如果 RX[5:0]的值大于 31，那么 RZ 将被清零。

对于 lsr　rz, rx, ry，将 RX 的值进行逻辑右移（原值右移，左侧移入 0），结果存入 RZ，右移位数由 RY 低 6 位（RY[5:0]）的值确定；如果 RY[5:0]的值大于 31，那么 RZ 将被清零。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← RZ >> RX[5:0]

语法： lsr16　rz, rx

说明： 将 RZ 的值进行逻辑右移（原值右移，左侧移入 0），结果存入 RZ，右移位数由 RX 低 6 位（RX[5:0]）的值确定；如果 RX[5:0]的值大于 31，那么 RZ 将被清零。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15	14	10	9	6	5	2	1	0
0	1 1 1 0 0		RZ		RX		0 1	

32位指令

操作： RZ ← RX >> RY[5:0]

语法： lsr32 rz, rx, ry

说明： 将 RX 的值进行逻辑右移（原值右移，左侧移入 0），结果存入 RZ，右移位数由 RY 低 6 位（RY[5:0]）的值确定；如果 RY[5:0]的值大于 31，那么 RZ 将被清零。

影响标志位： 无影响

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		RY		RX	0 1 0 0 0 0			0 0 0 1 0		RZ	

LSRC——立即数逻辑右移至 C 位指令

统一化指令

语法	操作	编译结果
lsrc rz, rx, oimm5	RZ ← RX >> OIMM5, C ← RX[OIMM5 - 1]	仅存在 32 位指令。 lsrc32 rz, rx, oimm5

说明： 将 RX 的值进行逻辑右移（原值右移，左侧移入 0），把移出最末位存入条件位 C，移位结果存入 RZ，右移位数由带偏置 1 的 5 位立即数（OIMM5）的值决定。如果 OIMM5 的值等于 32，那么条件位 C 为 RX 的最高位，RZ 被清零。

影响标志位： C ← RX[OIMM5 - 1]

限制： 立即数的范围为 1-32。

异常： 无

32位指令

操作： RZ ← RX >> OIMM5, C ← RX[OIMM5 - 1]

语法： lsrc32 rz, rx, oimm5

说明： 将 RX 的值进行逻辑右移（原值右移，左侧移入 0），把移出最末位存入条件位 C，移位结果存入 RZ，右移位数由带偏置 1 的 5 位立即数（OIMM5）的值决定。如果 OIMM5 的值等于 32，那么条件位 C 为 RX 的最高位，RZ 被清零。

注意：二进制操作数 IMM5 等于 OIMM5 - 1。

影响标志位： C ← RX[OIMM5 - 1]

限制： 立即数的范围为 1-32。

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		IMM5		RX		0 1 0 0 1 1		0 0 0 1 0		RZ	

IMM5 域——指定不带偏置立即数的值。

注意：移位的值 OIMM5 比起二进制操作数 IMM5 需偏置 1。

00000——移 1 位

00001——移 2 位

……

11111——移 32 位

LSRI——立即数逻辑右移指令

统一化指令

语法	操作	编译结果

lsri rz, rx, imm5	RZ ← RX >> IMM5	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<8) and (z<8), then
		lsri16 rz, rx, imm5
		else
		lsri32 rz, rx, imm5

说明： 将 RX 的值进行逻辑右移（原值右移，左侧移入 0），结果存入 RZ，右移位数由 5 位立即数（IMM5）的值决定；如果 IMM5 的值等于 0，那么 RZ 的值不变或者将与 RX 相同。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

16位指令

操作： RZ ← RX >> IMM5

语法： lsri16 rz, rx, imm5

说明： 将 RX 的值进行逻辑右移（原值右移，左侧移入 0），结果存入 RZ，右移位数由 5 位立即数（IMM5）的值决定；如果 IMM5 的值等于 0，那么 RZ 的值将不变。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7；立即数的范围为 0-31。

异常： 无

指令格式：

15 14 11 10 8 7 5 4 0

0	1 0 0 1	RX	RZ	IMM5

32位指令

操作： RZ ← RX >> IMM5

语法： lsri32 rz, rx, imm5

说明： 将 RX 的值进行逻辑右移（原值右移，左侧移入 0），结果存入 RZ，

右移位数由 5 位立即数（IMM5）的值决定；如果 IMM5 的值等于 0，那么 RZ 的值将与 RX 相同。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		IMM5		RX		0 1 0 0 1 0		0 0 0 1 0		RZ	

MFCR——控制寄存器读传送指令

统一化指令

语法	操作	编译结果
mfcr rz, cr<x, sel>	将控制寄存器的内容传送到通用寄存器中 RZ ← CR<X, sel>	仅存在 32 位指令。 mfcr32 rz, cr<x, sel>

属性： 特权指令

说明： 将控制寄存器 CR<x, sel>的内容传送到通用寄存器 RZ 中。

影响标志位： 无影响

异常： 特权违反异常

32位指令

操作： 将控制寄存器的内容传送到通用寄存器中

RZ ← CR<X, sel>

语法： mfcr32 rz, cr<x, sel>

属性： 特权指令

说明： 将控制寄存器 CR<x, sel>的内容传送到通用寄存器 RZ 中。

影响标志位： 无影响

异常： 特权违反异常

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 0	sel	CRX	0 1 1 0 0 0	0 0 0 0 1	RZ

MOV——数据传送指令#

统一化指令

语法	操作	编译结果
mov rz, rx	RZ ← RX	总是编译为 16 位指令。 mov16 rz, rx

说明： 把 RX 中的值复制到目的寄存器 RZ 中。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← RX

语法： mov16 rz, rx

说明： 把 RX 中的值复制到目的寄存器 RZ 中。

注意，该指令寄存器索引范围为 r0-r31。

影响标志位： 无影响

异常： 无

指令格式：

15	14 10	9 6	5 2	1 0
0	1 1 0 1 1	RZ	RX	1 1

32位指令

操作： RZ ← RX

语法： mov32 rz, rx

说明： 把 RX 中的值复制到目的寄存器 RZ 中。

注意，该指令是 lsli32 rz, rx, 0x0 的伪指令。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	0 0 0 0 0	RX	0 1 0 0 1 0	0 0 0 0 1	RZ

MOVF——C 为 0 数据传送指令#

统一化指令

语法	操作	编译结果

movf　rz, rx	if C==0, then
	　RZ ← RX;
	else
	　RZ ← RZ;
		仅存在 32 位指令。
		movf32　rz, rx

说明： 如果 C 为 0，把 RX 的值复制到目的寄存器 RZ 中；否则，RZ 的值不变。

注意，该指令是 incf　rz, rx, 0x0 的伪指令。

影响标志位： 无影响

异常： 无

32位指令

操作： if C==0, then

　RZ ← RX;

else

　RZ ← RZ;

语法： movf32　rz, rx

说明： 如果 C 为 0，把 RX 的值复制到目的寄存器 RZ 中；否则，RZ 的值不变。

注意，该指令是 incf32　rz, rx, 0x0 的伪指令。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	RZ	RX	0 0 0 0 1 1	0 0 0 0 1	0 0 0 0 0

MOVI——立即数数据传送指令

统一化指令

语法	操作	编译结果

movi16 rz, imm16	RZ ← zero_extend(IMM16);	根据立即数和寄存器的范围编译为对应的 16 位或 32 位指令。
		if (imm16<256) and (z<7), then
		  movi16 rz, imm8;
		else
		  movi32 rz, imm16;

说明： 将 16 位立即数零扩展至 32 位，然后传送至目的 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0x0-0xFFFF。

异常： 无

16位指令

操作： RZ ← zero_extend(IMM8);

语法： movi16 rz, imm8

说明： 将 8 位立即数零扩展至 32 位，然后传送至目的 RZ。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7；立即数的范围为 0-255。

异常： 无

15	14 11	10 8	7 0
0	0 1 1 0	RZ	IMM8

32位指令

操作： RZ ← zero_extend(IMM16);

语法： movi32 rz, imm16

说明： 将 16 位立即数零扩展至 32 位，然后传送至目的 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0x0-0xFFFF。

异常： 无

31	30 26	25 21	20 16	15 0
1	1 1 0 1 0	1 0 0 0 0	RZ	IMM16

MOVIH——立即数高位数据传送指令

统一化指令

语法	操作	编译结果
movih rz, imm16	RZ ← zero_extend(IMM16) << 16	仅存在 32 位指令。 movih32 rz, imm16

说明： 将 16 位立即数零扩展至 32 位，然后逻辑左移 16 位，传送结果至目的 RZ。

该指令可配合 ori rz, imm16 指令产生任意 32 位立即数。

影响标志位： 无影响

限制： 立即数的范围为 0x0-0xFFFF。

异常： 无

32位指令

操作： RZ ← zero_extend(IMM16) << 16

语法： movih32 rz, imm16

说明： 将 16 位立即数零扩展至 32 位，然后逻辑左移 16 位，传送结果至目的 RZ。

该指令可配合 ori32 rz, imm16 指令产生任意 32 位立即数。

影响标志位： 无影响

限制： 立即数的范围为 0x0-0xFFFF。

异常： 无

指令格式：

31	30 26	25 21	20 16	15 0
1	1 1 0 1 0	1 0 0 0 1	RZ	IMM16

MOVT——C 为 1 数据传送指令#

统一化指令

语法	操作	编译结果

movt　rz, rx	if C==1, then
	　RZ ← RX;
	else
	　RZ ← RZ;
		仅存在 32 位指令。
		movt32　rz, rx

说明： 如果 C 为 1，把 RX 的值复制到目的寄存器 RZ 中；否则，RZ 的值不变。

注意，该指令是 inct　rz, rx, 0x0 的伪指令。

影响标志位： 无影响

异常： 无

32位指令

操作： if C==1, then

　RZ ← RX;

else

　RZ ← RZ;

语法： movt32　rz, rx

说明： 如果 C 为 1，把 RX 的值复制到目的寄存器 RZ 中；否则，RZ 的值不变。

注意，该指令是 inct32　rz, rx, 0x0 的伪指令。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	RZ	RX	0 0 0 0 1 1	0 0 0 1 0	0 0 0 0 0

MTCR——控制寄存器写传送指令

统一化指令

语法	操作	编译结果
mtcr rx, cr<z, sel>	将通用寄存器的内容传送到控制寄存器中 CR<Z, sel> ← RX	仅存在 32 位指令。 mtcr32 rx, cr<z, sel>

属性： 特权指令

说明： 将通用寄存器 RX 的内容传送到控制寄存器 CR<z, sel>中。

影响标志位： 如果目标控制寄存器不是 PSR，则该指令不会影响标志位。

异常： 特权违反异常

32位指令

操作： 将通用寄存器的内容传送到控制寄存器中
CR<Z, sel> ← RX

语法： mtcr32 rx, cr<z, sel>

属性： 特权指令

说明： 将通用寄存器 RX 的内容传送到控制寄存器 CR<z, sel>中。

影响标志位： 如果目标控制寄存器不是 PSR，则该指令不会影响标志位。

异常： 特权违反异常

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 0	sel	RX	0 1 1 0 0 1	0 0 0 0 1	CRZ

MULT——乘法指令

统一化指令

语法	操作	编译结果

mult rz, rx	两个数相乘，结果的低 32 位放入通用寄存器中
	RZ ← RX × RZ
		根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<16) and (z<16), then
		mult16 rz, rx;
		else
		mult32 rz, rz, rx;

mult rz, rx, ry	两个数相乘，结果的低 32 位放入通用寄存器中
	RZ ← RX × RY
		根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (y==z) and (x<16)and (z<16), then
		mult16 rz, rx;
		else
		mult32 rz, rx, ry;

说明： 将两个源寄存器的内容相乘后结果的低 32 位存放到目的寄存器中，结果的高 32 位舍弃。不管源操作数被认为是有符号数还是无符号数，结果都相同。

影响标志位： 无影响

异常： 无

16位指令

操作： 两个数相乘，结果的低 32 位放入通用寄存器中

RZ ← RX × RZ

语法： mult16 rz, rx

说明： 将通用寄存器 RX 和 RZ 的内容相乘后得到结果的低 32 位存放到通用寄存器 RZ 中，结果的高 32 位舍弃。不管源操作数被认为是有符号数还是无符号数，结果都相同。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15 14 10 9 6 5 2 1 0

0	1 1 1 1 1	RZ	RX	0 0

32位指令

操作： 两个数相乘，结果的低 32 位放入通用寄存器中

RZ ← RX × RY

语法： mult32 rz, rx, ry

说明： 将通用寄存器 RX 和 RY 的内容相乘后结果的低 32 位存放到通用寄存器 RZ 中，结果的高 32 位舍弃。不管源操作数被认为是有符号数还是无符号数，结果都相同。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	RY	RX	1 0 0 0 0 1	0 0 0 0 1	RZ

MVC——C 位传送指令

统一化指令

语法	操作	编译结果
mvc rz	RZ ← C	仅存在 32 位指令。 mvc32 rz;

说明： 把条件位 C 传送到 RZ 的最低位，RZ 的其它位清零。

影响标志位： 无影响

异常： 无

32位指令

操作： RZ ← C

语法： mvc32 rz

说明： 把条件位 C 传送到 RZ 的最低位，RZ 的其它位清零。

影响标志位： 无影响

异常： 无

指令格式：

31	30 ~ 26	25 ~ 21	20 ~ 16	15 ~ 10	9 ~ 5	4 ~ 0
1	1 0 0 0 1	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0 1	0 1 0 0 0	RZ

MVCV——C 位取反传送指令

统一化指令

语法	操作	编译结果
mvcv　rz	RZ ← (!C)	仅存在 16 位指令。 mvcv16　rz

说明： 把条件位 C 取反后传送到 RZ 的最低位，RZ 的其它位清零。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← (!C)

语法： mvcv16　rz

说明： 把条件位 C 取反后传送到 RZ 的最低位，RZ 的其它位清零。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15	14 10	9 6	5 2	1 0
0	1 1 0 0 1	RZ	0 0 0 0	1 1

NIE——中断嵌套使能指令

统一化指令

语法	操作	编译结果
nie	将中断的控制寄存器现场{EPSR, EPC}存储到堆栈存储器中，然后更新堆栈指针寄存器到堆栈存储器的顶端，打开 PSR.IE 和 PSR.EE； MEM[SP-4] ←EPC; MEM[SP-8] ←EPSR; SP←SP-8; PSR({EE,IE}) ← 1	仅存在 16 位指令。 nie16

说明： 将中断的控制寄存器现场{EPSR, EPC }保存到堆栈存储器中，然后更新堆栈指针寄存器到堆栈指针存储器的顶端，打开中断和异常使能位 PSR.IE 和 PSR.EE。采用堆栈指针寄存器直接寻址方式。

影响标志位： 无影响

异常： 访问错误异常、未对齐异常、特权违反异常

16位指令

操作： 将中断的控制寄存器现场{EPSR, EPC}存储到堆栈存储器中，然后更新堆栈指针寄存器到堆栈存储器的顶端，打开 PSR.IE 和 PSR.EE；

MEM[SP-4] ←EPC;

MEM[SP-8] ←EPSR;

SP←SP-8;

PSR({EE,IE}) ← 1

语法： NIE16

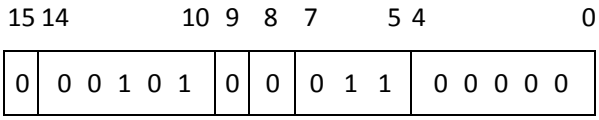
属性 特权指令

说明： 将中断的控制寄存器现场{EPSR, EPC }保存到堆栈存储器中，然后更新堆栈指针寄存器到堆栈指针存储器的顶端，打开中断和异常使能位 PSR.IE 和 PSR.EE。采用堆栈指针寄存器直接寻址方式。

影响标志位： 无影响

异常： 访问错误异常、未对齐异常、特权违反异常

指令格式：


15 14 10 9 8 7 5 4 0
0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0

NIR——中断嵌套返回指令

统一化指令

语法	操作	编译结果
nir	从堆栈存储器中载入中断的控制寄存器现场到{EPSR, EPC }中，然后更新堆栈指针寄存器到堆栈存储器的顶端；并中断返回 EPSR←MEM[SP] EPC←MEM[SP+4]; SP←SP+8; PSR←EPSR; PC←EPC	仅存在 16 位指令。 nir16

说明： 从堆栈存储器中载入中断的现场到{EPSR, EPC }中，然后更新堆栈指针寄存器到堆栈存储器的顶端；PC 值恢复为控制寄存器 EPC 中的值，PSR 值恢复为 EPSR 的值，指令执行从新的 PC 地址处开始。采用堆栈指针寄存器直接寻址方式。

影响标志位： 无影响

异常： 访问错误异常、未对齐异常、特权违反异常

16位指令

操作： 从堆栈存储器中载入中断的控制寄存器现场到{EPSR, EPC }中，然后更新堆栈指针寄存器到堆栈存储器的顶端；并中断返回

EPSR←MEM[SP]

EPC←MEM[SP+4];

SP←SP+8;

PSR←EPSR;

PC←EPC

语法： NIR16

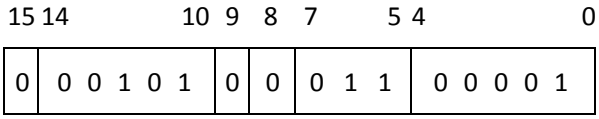
属性 特权指令

说明： 从堆栈存储器中载入中断的现场到{EPSR, EPC }中，然后更新堆栈指针寄存器到堆栈存储器的顶端；PC 值恢复为控制寄存器 EPC 中的值，PSR 值恢复为 EPSR 的值，指令执行从新的 PC 地址处开始。采用堆栈指针寄存器直接寻址方式。

影响标志位： 无影响

异常： 访问错误异常、未对齐异常、特权违反异常

指令格式：



NOR——按位或非指令

统一化指令

语法	操作	编译结果

nor rz, rx	RZ ← !(RZ | RX)	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<16) and (z<16), then
		nor16 rz, rx;
		else
		nor32 rz, rz, rx;

nor rz, rx, ry	RZ ← !(RX | RY)	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (y==z) and (x<16) and (z<16), then
		nor16 rz, rx
		else
		nor32 rz, rx, ry

说明： 将 RX 与 RY/RZ 的值按位或，然后按位取非，并把结果存在 RZ。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← !(RZ | RX)

语法： nor16 rz, rx

说明： 将 RZ 与 RX 的值按位或，然后按位取非，并把结果存在 RZ。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15	14 10	9 6	5 2	1 0
0	1 1 0 1 1	RZ	RX	1 0

32位指令

操作： RZ ← !(RX | RY)

语法： nor32 rz, rx, ry

说明： 将 RX 与 RY 的值按位或，然后按位取非，并把结果存在 RZ。

影响标志位： 无影响

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		RY		RX		0 0 1 0 0 1		0 0 1 0 0		RZ	

NOT——按位非指令#

统一化指令

语法	操作	编译结果

not rz	RZ ← !(RZ)	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (z<16), then
		not16 rz;
		else
		not32 rz, rz;

not rz, rx	RZ ← !(RX)	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x==z) and (z<16), then
		not16 rz;
		else
		not32 rz, rx;

说明： 将 RZ/RX 的值按位取反，把结果存在 RZ。

注意，该指令是 nor rz, rz 和 nor rz, rx, rx 的伪指令。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← !(RZ)

语法： not16 rz

说明： 将 RZ 的值按位取反，把结果存在 RZ。

注意，该指令是 nor16 rz, rz 的伪指令。

影响标志位： 无影响

异常： 无

指令格式：

15	14 10	9 6	5 2	1 0
0	1 1 0 1 1	RZ	RZ	1 0

32位指令

操作： RZ ← !(RX)

语法： not32 rz, rx

说明： 将 RX 的值按位取反，把结果存在 RZ。

注意，该指令是 nor32 rz, rx, rx 的伪指令。

影响标志位： 无影响

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		RX		RX		0 0 1 0 0 1		0 0 1 0 0		RZ	

OR——按位或指令

统一化指令

语法	操作	编译结果

or rz, rx	RZ ← RZ | RX	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<16) and (z<16), then
		or16 rz, rx ;
		else
		or32 rz, rz, rx;

or rz, rx, ry	RZ ← RX | RY	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (y==z) and (x<16) and (z<16), then
		or16 rz, rx
		else
		or32 rz, rx, ry

说明： 将 RX 与 RY/RZ 的值按位或，并把结果存在 RZ。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← RZ | RX

语法： or16 rz, rx

说明： 将 RZ 与 RX 的值按位或，并把结果存在 RZ。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15 14 10 9 6 5 2 1 0

0	1 1 0 1 1	RZ	RX	0 0

32位指令

操作： RZ ← RX | RY

语法： or rz, rx, ry

说明： 将 RX 与 RY 的值按位或，并把结果存在 RZ。

影响标志位： 无影响

异常： 无

指令格式：

31	30 26	25 21	20 16	15 10	9 5	4 0
1	1 0 0 0 1	RY	RX	0 0 1 0 0 1	0 0 0 0 1	RZ

ORI——立即数按位或指令

统一化指令

语法	操作	编译结果
ori　rz, rx, imm16	RZ ← RX | zero_extend(IMM16)	仅存在 32 位指令。 ori32　rz, rx, imm16

说明： 将 16 位立即数零扩展至 32 位，然后与 RX 的值进行按位或操作，把结果存入 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0x0-0xFFFF。

异常： 无

32位指令

操作： RZ ← RX | zero_extend(IMM16)

语法： ori32　rz, rx, imm16

说明： 将 16 位立即数零扩展至 32 位，然后与 RX 的值进行按位或操作，把结果存入 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0x0-0xFFFF。

异常： 无

指令格式：

31	30	26	25	21	20	16	15	0
1	1 1 0 1 1		RZ		RX		IMM16	

POP——出栈指令

统一化指令

语法	操作	编译结果

pop reglist	从堆栈存储器加载连续的多个字到一片连续的寄存器堆中，然后更新堆栈寄存器到堆栈存储器的顶端，并子程序返回；
	dst ← {reglist}; addr ← SP;
	foreach (reglist){
	Rdst ← MEM[addr];
	dst ← next {reglist};
	addr ← addr + 4;
	}
	sp ← addr;
	PC ← R15 & 0xfffffffe;
		pop16 reglist

说明： 从堆栈存储器加载连续的多个字到一片连续的寄存器堆中，更新堆栈指针寄存器，然后实现子程序返回功能，即程序跳转到链接寄存器 R15 指定的位置，链接寄存器的最低位被忽略。采用堆栈寄存器直接寻址方式。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

16位指令

操作： 从堆栈存储器加载连续的多个字到一片连续的寄存器堆中，然后更新堆栈寄存器到堆栈存储器的顶端，并子程序返回。

```
dst ← {reglist}; addr ← SP;
foreach ( reglist ){
   Rdst ← MEM[addr];
   dst ← next {reglist};
   addr ← addr + 4;
}
sp ← addr;
PC ← R15 & 0xfffffffe;
```

语法： pop16 reglist

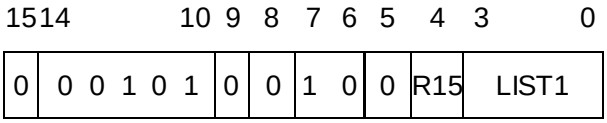
说明： 从堆栈存储器加载连续的多个字到一片连续的寄存器堆中，更新堆栈指针寄存器，然后实现子程序返回功能，即程序跳转到链接寄存器 R15 指定的位置，链接寄存器的最低位被忽略。采用堆栈指针寄存器直接寻址方式。

影响标志位： 无影响

限制： 寄存器的范围为 r4 – r11, r15。

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

指令格式：



LIST1 域——指定寄存器 r4-r11 是否在寄存器列表中。

0000——r4-r11 不在寄存器列表中

0001——r4 在寄存器列表中

0010——r4-r5 在寄存器列表中

0011——r4-r6 在寄存器列表中

……

1000——r4-r11 在寄存器列表中

R15 域——指定寄存器 r15 是否在寄存器列表中。

0——r15 不在寄存器列表中

1——r15 在寄存器列表中

PSRCLR——PSR 位清零指令

统一化指令

语法	操作	编译结果
psrclr　ee, ie, fe, af 或者操作数也可以为ee、ie、fe、af的任意组合。	清除状态寄存器的某一位或几位。 PSR({EE，IE，FE，AF}) ← 0	仅存在 32 位指令。 psrclr32　ee, ie, fe, af

属性： 特权指令

说明： 选中的 PSR 位被清零（1 表示选中）。五位立即数 IMM5 用于编码要清除的控制位，对应关系如下：

立即数 IMM5 各位	对应的 PSR 控制位
Imm5[0]	AF
Imm5[1]	FE
Imm5[2]	IE
Imm5[3]	EE
Imm5[4]	保留

影响标志位： 无影响

异常： 特权违反异常

32位指令

操作： 清除状态寄存器的某一位或几位

PSR({EE，IE，FE，AF}) ← 0

语法： psrclr32　ee, ie, fe, af

或者操作数也可以为 ee、ie、fe、af 的任意组合。

属性： 特权指令

说明： 选中的 PSR 位被清零（1 表示选中）。五位立即数 IMM5 用于编码要清除的控制位，对应关系如下：

立即数 IMM5 各位	对应的 PSR 控制位
Imm5[0]	AF
Imm5[1]	FE
Imm5[2]	IE
Imm5[3]	EE
Imm5[4]	保留

影响标志位： 无影响

异常： 特权违反异常

指令格式:

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 0		IMM5		0 0 0 0 0		0 1 1 1 0 0		0 0 0 0 1		0 0 0 0 0	

PSRSET——PSR 位置位指令

统一化指令

语法	操作	编译结果
psrset　ee, ie, fe, af 或者操作数也可以为ee、ie、fe、af的任意组合。	设置状态寄存器的某几位 PSR({EE，IE，FE，AF}) ← 1	仅存在 32 位指令。 psrset32　ee, ie, fe, af

属性： 特权指令

说明： 选中的 PSR 位被置位（1 表示选中）。五位立即数 IMM5 用于编码要清除的控制位，对应关系如下：

立即数 IMM5 各位	对应的 PSR 控制位
Imm5[0]	AF
Imm5[1]	FE
Imm5[2]	IE
Imm5[3]	EE
Imm5[4]	保留

影响标志位： 无影响

异常： 特权违反异常

32位指令

操作： 设置状态寄存器的某几位

PSR({EE，IE，FE，AF}) ← 1

语法： psrset32　ee, ie, fe, af

或者操作数也可以为 ee、ie、fe、af 的任意组合。

属性： 特权指令

说明： 选中的 PSR 位被置位（1 表示选中）。五位立即数 IMM5 用于编码要清除的控制位，对应关系如下：

立即数 IMM5 各位	对应的 PSR 控制位
Imm5[0]	AF
Imm5[1]	FE
Imm5[2]	IE
Imm5[3]	EE
Imm5[4]	保留

影响标志位： 无影响

异常： 特权违反异常

指令格式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	IMM5					0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0

PUSH——压栈指令

统一化指令

语法	操作	编译结果

push reglist	将寄存器列表中的字存储到堆栈存储器中，然后更新堆栈寄存器到堆栈存储器的顶端；
	src ← {reglist}; addr ← SP;
	foreach (reglist){
	addr ← addr - 4; MEM[addr] ← Rsrc;
	src ← next {reglist};
	}
	sp ← addr;
		push16 reglist

说明： 将寄存器列表中的字存储到堆栈存储器中，然后更新堆栈寄存器到堆栈存储器的顶端。采用堆栈寄存器直接寻址方式。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

16位指令

操作： 将寄存器列表中的字存储到堆栈存储器中

```
src ← {reglist}; addr ← SP;
foreach ( reglist ){
   MEM[addr] ← Rsrc;
   src ← next {reglist};
   addr ← addr - 4;
}
sp ← addr
```

语法： push16 reglist

说明： 将寄存器列表中的字存储到堆栈存储器中，然后更新堆栈寄存器到堆栈存储器的顶端。采用堆栈寄存器直接寻址方式。

影响标志位： 无影响

限制： 寄存器的范围为 r4 – r11, r15。

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异

常、TLB 写无效异常

指令格式：

15	14 10	9	8	7	6	5	4	3 0
0	0 0 1 0 1	0	0	1	1	0	R15	LIST1

LIST1 域——指定寄存器 r4-r11 是否在寄存器列表中。

0000——r4-r11 不在寄存器列表中

0001——r4 在寄存器列表中

0010——r4-r5 在寄存器列表中

0011——r4-r6 在寄存器列表中

……

1000——r4-r11 在寄存器列表中

R15 域——指定寄存器 r15 是否在寄存器列表中。

0——r15 不在寄存器列表中

1——r15 在寄存器列表中

REVB——字节倒序指令

统一化指令

语法	操作	编译结果
revb rz, rx	RZ[31:24] ← RX[7:0]; RZ[23:16] ← RX[15:8]; RZ[15:8] ← RX[23:16]; RZ[7:0] ← RX[31:24];	仅存在 16 位指令 revb16 rz, rxjin

说明： 把 RX 的值按字节取倒序，各字节内部的位顺序保持不变，结果存入 RZ。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ[31:24] ← RX[7:0];
RZ[23:16] ← RX[15:8];
RZ[15:8] ← RX[23:16];
RZ[7:0] ← RX[31:24];

语法： revb16 rz, rx

说明： 把 RX 的值按字节取倒序，各字节内部的位顺序保持不变，结果存入 RZ。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15	14 10	9 6	5 2	1 0
0	1 1 1 1 0	RZ	RX	1 0

REVH——半字字节倒序指令

统一化指令

语法	操作	编译结果
revh rz, rx	RZ[31:24] ← RX[23:16]; RZ[23:16] ← RX[31:24]; RZ[15:8] ← RX[7:0]; RZ[7:0] ← RX[15:8];	仅存在 16 位指令。 revh16 rz, rx

说明： 把 RX 的值在半字内按字节取倒序，即分别交换高半字内的两个字节和低半字内的两个字节，两个半字间的顺序和各字节内的位顺序保持不变，结果存入 RZ。

影响标志位： 无影响

异常： 无

16位指令

操作：
RZ[31:24] ← RX[23:16];
RZ[23:16] ← RX[31:24];
RZ[15:8] ← RX[7:0];
RZ[7:0] ← RX[15:8];

语法： revh16 rz, rx

说明： 把 RX 的值在半字内按字节取倒序，即分别交换高半字内的两个字节和低半字内的两个字节，两个半字间的顺序和各字节内的位顺序保持不变，结果存入 RZ。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15	14 10	9 6	5 2	1 0
0	1 1 1 1 0	RZ	RX	1 1

ROTL——循环左移指令

统一化指令

语法	操作	编译结果

rotl rz, rx	RZ ← RZ <<<< RX[5:0]	根据寄存器的范围编译为对应的16位或32位指令。
		if (x<16) and (z<16), then
		rotl16 rz, rx ;
		else
		rotl32 rz, rz, rx;

rotl rz, rx, ry	RZ ← RX <<<< RY[5:0]	根据寄存器的范围编译为对应的16位或32位指令。
		if (x==z) and (y<16) and (z<16), then
		rotl16 rz, ry
		else
		rotl32 rz, rx, ry

说明： 对于 rotl rz, rx，将 RZ 的值进行循环左移（原值左移，右侧移入左侧移出的位），结果存入 RZ，左移位数由 RX 低 6 位（RX[5:0]）的值决定；如果 RX[5:0]的值大于 31，那么 RZ 将被清零。

对于 rotl rz, rx, ry，将 RX 的值进行循环左移（原值左移，右侧移入左侧移出的位），结果存入 RZ，左移位数由 RY 低 6 位（RY[5:0]）的值决定；如果 RY[5:0]的值大于 31，那么 RZ 将被清零。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← RZ <<<< RX[5:0]

语法： rotl16 rz, rx

说明： 将 RZ 的值进行循环左移（原值左移，右侧移入左侧移出的位），结果存入 RZ，左移位数由 RX 低 6 位（RX[5:0]）的值决定；如果 RX[5:0]的值大于 31，那么 RZ 将被清零。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15	14	10	9	6	5	2	1	0
0	1 1 1 0 0		RZ		RX		1	1

32位指令

操作： RZ ← RX <<<< RY[5:0]

语法： rotl32 rz, rx, ry

说明： 将 RX 的值进行循环左移（原值左移，右侧移入左侧移出的位），结果存入 RZ，左移位数由 RY 低 6 位（RY[5:0]）的值决定；如果 RY[5:0] 的值大于 31，那么 RZ 将被清零。

影响标志位： 无影响

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		RY		RX		0 1 0 0 0 0		0 1 0 0 0		RZ	

ROTLI——立即数循环左移指令

统一化指令

语法	操作	编译结果
rotli rz, rx, imm5	RZ ← RX <<<< IMM5	rotli32 rz, rx, imm5;

说明： 将 RX 的值进行循环左移（原值左移，右侧移入左侧移出的位），结果存入 RZ，左移位数由 5 位立即数（IMM5）的值决定；如果 IMM5 的值等于 0，那么 RZ 的值将与 RX 相同。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

32位指令

操作： RZ ← RX <<<< IMM5

语法： rotli32 rz, rx, imm5

说明： 将 RX 的值进行循环左移（原值左移，右侧移入左侧移出的位），结果存入 RZ，左移位数由 5 位立即数（IMM5）的值决定；如果 IMM5 的值等于 0，那么 RZ 的值将与 RX 相同。

影响标志位： 无影响

限制： 立即数的范围为 0-31。

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		IMM5		RX		0 1 0 0 1 0		0 1 0 0 0		RZ	

RSUB——反向减法指令#

统一化指令

语法	操作	编译结果
rsub rz, rx, ry	RZ ← RY - RX	仅存在 32 位指令。 rsub32 rz, rx, ry

说明： 将 RY 的值减去 RX 值，并把结果存在 RZ 中。

注意，该指令是 subu rz, ry, rx 的伪指令。

影响标志位： 无影响

异常： 无

32位指令

操作： RZ ← RY - RX

语法： rsub32 rz, rx, ry

说明： 将 RY 的值减去 RX 值，并把结果存在 RZ 中。

注意，该指令是 subu32 rz, ry, rx 的伪指令。

影响标志位： 无影响

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		RX		RY		0 0 0 0 0 0		0 0 1 0 0		RZ	

RTS——子程序返回指令#

统一化指令

语法	操作	编译结果
rts	程序跳转到链接寄存器指定的位置 PC ← R15 & 0xfffffffe	总是编译为 16 位指令。 rts16

说明： 程序跳转到链接寄存器 R15 指定的位置，链接寄存器的最低位被忽略。RTS16 指令的跳转范围是全部 4GB 地址空间。

该指令用于实现子程序返回功能。

注意，该指令是 jmp r15 的伪指令。

影响标志位： 无影响

异常： 无

16位指令

操作： 程序跳转到链接寄存器指定的位置

PC ← R15 & 0xfffffffe

语法： rts16

说明： 程序跳转到链接寄存器 R15 指定的位置，链接寄存器的最低位被忽略。RTS16 指令的跳转范围是全部 4GB 地址空间。

该指令用于实现子程序返回功能。

注意，该指令是 jmp16 r15 的伪指令。

影响标志位： 无影响

异常： 无

指令格式：

15	14 13 12 11 10	9 8 7 6	5 4 3 2	1 0
0	1 1 1 1 0	0 0 0 0	1 1 1 1	0 0

RTE——异常和普通中断返回指令

统一化指令

语法	操作	编译结果
rte	异常和普通中断返回 PC ← EPC，PSR ← EPSR	仅存在 32 位指令。 rte32

属性： 特权指令

说明： PC 值恢复为保存在控制寄存器 EPC 中的值，PSR 值恢复为保存在 EPSR 的值，指令执行从新的 PC 地址处开始。

影响标志位： 无影响

异常： 特权违反异常

32位指令

操作： 异常和普通中断返回

PC ← EPC，PSR ← EPSR

语法： rte32

属性： 特权指令

说明： PC 值恢复为保存在控制寄存器 EPC 中的值，PSR 值恢复为保存在 EPSR 的值，指令执行从新的 PC 地址处开始。

影响标志位： 无影响

异常： 特权违反异常

指令格式：

31	30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10	9 8 7 6 5	4 3 2 1 0
1	1 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 1 0 0 0 0	0 0 0 0 1	0 0 0 0 0

SEXTB——字节提取并有符号扩展指令#

统一化指令

语法	操作	编译结果
sextb rz, rx	RZ ← sign_extend(RX[7:0]);	仅存在 16 位指令。 sextb16 rz, rx

说明： 将 RX 的低字节（RX[7:0]）符号扩展至 32 位，结果存在 RZ 中。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← sign_extend(RX[7:0]);

语法： sextb16 rz, rx

说明： 将 RX 的低字节（RX[7:0]）符号扩展至 32 位，结果存在 RZ 中。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15	14 10	9 6	5 2	1 0
0	1 1 1 0 1	RZ	RX	1 0

SEXTH——半字提取并有符号扩展指令#

统一化指令

语法	操作	编译结果
sexth rz, rx	RZ ← sign_extend(RX[15:0]);	仅存在 16 位指令 sexth16 rz, rx

说明： 将 RX 的低半字（RX[15:0]）符号扩展至 32 位，结果存在 RZ 中。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← sign_extend(RX[15:0]);

语法： sexth16 rz, rx

说明： 将 RX 的低半字（RX[15:0]）符号扩展至 32 位，结果存在 RZ 中。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式

15	14 10	9 6	5 2	1 0
0	1 1 1 0 1	RZ	RX	1 1

ST.B——字节存储指令

统一化指令

语法	操作	编译结果

st.b rz, (rx, disp)	将寄存器中的最低字节存储到存储器中
	MEM[RX + zero_extend(offset)] ← RZ[7:0]
		根据偏移量和寄存器的范围编译为对应的 16 位或 32 位指令。
		if (disp<32) and (x<7) and (z<7), then
		st16.b rz, (rx, disp);
		else
		st32.b rz, (rx, disp);

说明： 将寄存器 RZ 中的最低字节存储到存储器中。采用寄存器加无符号立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上 12 位相对偏移量无符号扩展到 32 位后的值得到。ST.B 指令可以寻址+4KB 地址空间。

注意，偏移量 DISP 即二进制操作数 Offset。

影响标志位： 无影响

异常： 访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

16位指令

操作： 将寄存器中的最低字节存储到存储器中

MEM[RX + zero_extend(offset)] ← RZ[7:0]

语法： st16.b rz, (rx, disp)

说明： 将寄存器 RZ 中的最低字节存储到存储器中。采用寄存器加无符号立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上 5 位相对偏移量无符号扩展到 32 位后的值得到。ST16.B 指令可以寻址+32B 的空间。

注意，偏移量 DISP 即二进制操作数 Offset。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7。

异常： 访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：

15	14	11	10	8	7	5	4	0
1	0 1 0 0		RX		RZ		IMM5	

32位指令

操作： 将寄存器中的最低字节存储到存储器中

MEM[RX + zero_extend(offset)] ← RZ[7:0]

语法： st32.b rz, (rx, disp)

说明： 将寄存器 RZ 中的最低字节存储到存储器中。采用寄存器加无符号立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上 12 位相对偏移量无符号扩展到 32 位后的值得到。ST32.B 指令可以寻址+4KB 地址空间。

注意，偏移量 DISP 即二进制操作数 Offset。

影响标志位： 无影响

异常： 访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

指令格式：

31	30	26	25	21	20	16	15	12	11	0
1	1 0 1 1 1		RZ		RX	0 0 0 0		Offset		

ST.H——半字存储指令

统一化指令

语法	操作	编译结果

st.h rz, (rx, disp)	将寄存器中的最低字节存储到存储器中
	MEM[RX + zero_extend(offset<< 1)] ← RZ[15:0]
		根据偏移量和寄存器的范围编译为对应的 16 位或 32 位指令。
		if (disp<64)and(x<7)and(z<7), then
		st16.h rz, (rx, disp);
		else
		st32.h rz, (rx, disp);

说明： 将寄存器 RZ 中的低半字存储到存储器中。采用寄存器加无符号立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上左移 1 位的 12 位相对偏移量无符号扩展到 32 位后的值得到。ST.H 指令可以寻址+8KB 地址空间。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

16位指令

操作： 将寄存器中的低半字存储到存储器中

MEM[RX + zero_extend(offset << 1)] ← RZ[15:0]

语法： st16.h rz, (rx, disp)

说明： 将寄存器 RZ 中的低半字存储到存储器中。采用寄存器加无符号立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上左移 1 位的 5 位相对偏移量无符号扩展到 32 位后的值得到。ST16.H 指令可以寻址+64B 的空间。

注意，偏移量 DISP 是二进制操作数 Offset 左移 1 位得到的。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7。

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：

15 14 11 10 8 7 5 4 0

1	0 1 0 1	RX	RZ	IMM5

32位指令

操作： 将寄存器中的低半字存储到存储器中

MEM[RX + zero_extend(offset << 1)] ← RZ[15:0]

语法： st32.h rz, (rx, disp)

说明： 将寄存器 RZ 中的低半字存储到存储器中。采用寄存器加无符号立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上左移 1 位的 12 位相对偏移量无符号扩展到 32 位后的值得到。ST32.H 指令可以寻址+8KB 地址空间。

注意，偏移量 DISP 是二进制操作数 Offset 左移 1 位得到的。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

指令格式：

31	30	26	25	21	20	16	15	12	11	0
1	1 0 1 1 1		RZ		RX	0 0 0 1			Offset	

ST.W——字存储指令

统一化指令

语法	操作	编译结果

st.w rz, (rx, disp)	将寄存器中的字存储到存储器中
	MEM[RX + zero_extend(offset<< 2)] ← RZ[31:0]
		根据偏移量和寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x=sp) and (z<7) and (disp < 1024),
		st16.w rz, (sp, disp);
		else if (disp<128) and (x<7) and (z<7),
		st16.w rz, (rx, disp);
		else
		st32.w rz, (rx, disp);

说明： 将寄存器 RZ 中的字存储到存储器中。采用寄存器加无符号立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上左移两位的 12 位相对偏移量无符号扩展到 32 位后的值得到。ST.W 指令可以寻址+16KB 地址空间。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

16位指令

操作： 将寄存器中的字存储到存储器中

MEM[RX + zero_extend(offset << 2)] ← RZ[31:0]

语法： st16.w rz, (rx, disp)

st16.w rz, (sp, disp)

说明： 将寄存器 RZ 中的字存储到存储器中。采用寄存器加无符号立即数偏移量的寻址方式。当 rx=sp 时，存储器的有效地址由基址寄存器 RX 加上左移两位的 8 位相对偏移量无符号扩展到 32 位后的值得到。当 rx 为其它寄存器时，存储器的有效地址由基址寄存器 RX 加上左移两位的 5 位相对偏移量无符号扩展到 32 位后的值得到。ST16.W 指令可以寻址+1KB 的空间。

注意，偏移量 DISP 是二进制操作数 IMM5 左移两位得到的。当基址寄存器 RX 为 SP 时，偏移量 DISP 是二进制操作数{IMM3, IMM5}

左移两位得到的。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7。

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 读无效异常

指令格式：

st16.w rz, (rx, disp)

15	14 13 12 11	10 9 8	7 6 5	4 3 2 1 0
1	0 1 1 0	RX	RZ	IMM5

st16.w rz, (sp, disp)

15	14 13 12 11	10 9 8	7 6 5	4 3 2 1 0
1	0 1 1 1	IMM3	RZ	IMM5

32位指令

操作： 将寄存器中的字存储到存储器中

MEM[RX + zero_extend(offset << 2)] ← RZ[31:0]

语法： st32.w rz, (rx, disp)

说明： 将寄存器 RZ 中的字存储到存储器中。采用寄存器加无符号立即数偏移量的寻址方式。存储器的有效地址由基址寄存器 RX 加上左移两位的 12 位相对偏移量无符号扩展到 32 位后的值得到。ST32.W 指令可以寻址+16KB 地址空间。

注意，偏移量 DISP 是二进制操作数 Offset 左移两位得到的。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

指令格式：

31	30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12	11 10 9 8 7 6 5 4 3 2 1 0
1	1 0 1 1 1	RZ	RX	0 0 1 0	Offset

STM——连续多字存储指令

统一化指令

语法	操作	编译结果

stm ry-rz, (rx)	将一片连续的寄存器堆中内容依次存储到一片连续的存储器地址上。
	src ← Y; addr ← RX;
	for (n = 0; n <=(Z-Y); n++){
	MEM[addr] ← Rsrc;
	src ← src + 1;
	addr ← addr + 4;
	}
		仅存在 32 位指令。
		stm32 ry-rz, (rx)

说明： 将从RY开始的一片连续的寄存器堆中的内容依次存储到一片连续的存储器地址上，即将寄存器 RY 的内容存到存储器指定地址开始的第一个字的地址上，寄存器 RY+1 的内容存到存储器指定地址开始的第二个字的地址上，依次类推，将寄存器 RZ 的内容存到存储器指定地址开始的最后一个字的地址上。存储器的有效地址由基址寄存器 RX 的内容决定。

影响标志位： 无影响

限制： RZ 应当大于等于 RY。

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

32位指令

操作： 将一片连续的寄存器堆中内容依次存储到一片连续的存储器地址上。

```
src ← Y; addr ← RX;
for (n = 0; n <= IMM5;  n++){
   MEM[addr] ← Rsrc;
   src ← src + 1;
   addr ← addr + 4;
}
```

语法： stm32 ry-rz, (rx)

说明： 将从RY开始的一片连续的寄存器堆中的内容依次存储到一片连续的存储器地址上，即将寄存器 RY 的内容存到存储器指定地址开始的第

一个字的地址上，寄存器 RY+1 的内容存到存储器指定地址开始的第二个字的地址上，依次类推，将寄存器 RZ 的内容存到存储器指定地址开始的最后一个字的地址上。存储器的有效地址由基址寄存器 RX 的内容决定。

影响标志位： 无影响

限制： RZ 应当大于等于 RY。

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 1 0 1		RY		RX		0 0 0 1 1 1		0 0 0 0 1		IMM5	

IMM5 域——指定目标寄存器的个数，IMM5 = Z - Y。

00000——1 个目的寄存器

00001——2 个目的寄存器

……

11111——32 个目的寄存器

STQ——连续四字存储指令#

统一化指令

语法	操作	编译结果
stq r4-r7, (rx)	将寄存器 R4—R7 中的字依次存储到一片连续的存储器地址上。 src ← 4; addr ← RX; for (n = 0; n <= 3; n++){ MEM[addr] ← Rsrc; src ← src + 1; addr ← addr + 4; }	仅存在 32 位指令。 stq32 r4-r7, (rx);

说明： 将寄存器堆[R4,R7]（包括边界）中的字依次存储到一片连续的存储器地址上，即将寄存器 R4 的内容存到存储器指定地址开始的第一个字的地址上，寄存器 R5 的内容存到存储器指定地址开始的第二个字的地址上，寄存器 R6 的内容存到存储器指定地址开始的第三个字的地址上，寄存器 R7 的内容存到存储器指定地址开始的第四个字的地址上。存储器的有效地址由基址寄存器 RX 的内容决定。注意，该指令是 stm r4-r7, (rx)的伪指令。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

32位指令

操作： 将寄存器 R4—R7 中的字依次存储到一片连续的存储器地址上。

```
src ← 4; addr ← RX;
for (n = 0; n <= 3; n++){
   MEM[addr] ← Rsrc;
   src ← src + 1;
   addr ← addr + 4; }
```

语法： stq32 r4-r7, (rx)

说明： 将寄存器堆[R4,R7]（包括边界）中的字依次存储到一片连续的存储器地址上，即将寄存器 R4 的内容存到存储器指定地址开始的第一个字的地址上，寄存器 R5 的内容存到存储器指定地址开始的第二个字的地址上，寄存器 R6 的内容存到存储器指定地址开始的第三

个字的地址上，寄存器 R7 的内容存到存储器指定地址开始的第四个字的地址上。存储器的有效地址由基址寄存器 RX 的内容决定。注意，该指令是 stm r4-r7, (rx)的伪指令。

影响标志位： 无影响

异常： 未对齐访问异常、访问错误异常、TLB 不可恢复异常、TLB 失配异常、TLB 写无效异常

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 1 0 1		0 0 1 0 0		RX		0 0 0 1 1 1		0 0 0 0 1		0 0 0 1 1	

STOP——进入低功耗暂停模式指令

统一化指令

语法	操作	编译结果
stop	进入低功耗暂停模式	仅存在 32 位指令。 stop32

说明： 此指令使处理器进入低功耗模式，并等待一个中断来退出这个模式。此时，CPU 时钟停止，大部分外围设备也被停止。

影响标志位： 无影响

异常： 特权违反异常

32位指令

操作： 进入低功耗暂停模式

语法： stop32

属性： 特权指令

说明： 此指令使处理器进入低功耗模式，并等待一个中断来退出这个模式。此时，CPU 时钟停止，大部分外围设备也被停止。

影响标志位： 无影响

异常： 特权违反异常

指令格式：

31	30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11	10 9 8 7 6	5 4 3 2 1	0
1	1 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 1 0 0 1 0	0 0 0 0 1	0 0 0 0 0	

SUBC——无符号带借位减法指令

统一化指令

语法	操作	编译结果

subc　rz, rx	RZ ← RZ - RX -（!C），
	C ← 借位
		根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<16) and (z<16), then
		　subc16　rz, rx;
		else
		　subc32　rz, rz, rx;

subc　rz, rx, ry	RZ ← RX - RY -（!C），
	C ← 借位
		根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x==z) and (y<16) and (z<16), then
		　subc16　rz, ry;
		else
		　subc32　rz, rx, ry;

说明： 对于 subc　rz, rx，将 RZ 的值减去寄存器 RX 的值和 C 位的非值；对于 subc　rz, rx, ry，将 RX 的值减去寄存器 RY 的值和 C 位的非值。把结果存在 RZ，借位存在 C 位。对于该减法指令来说，如果发生借位，将清 C 位，反之置 C 位。

影响标志位： C ← 借位

异常： 无

16位指令

操作： RZ ← RZ - RX -（!C），C ← 借位

语法： subc16　rz, rx

说明： 将 RZ 的值减去寄存器 RX 的值和 C 位的非值，并把结果存在 RZ，借位存在 C 位。对于该减法指令来说，如果发生借位，将清 C 位，反之置 C 位。

影响标志位： C ← 借位

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15 14　　10 9　　6 5　　2 1 0

0	1 1 0 0 0	RZ	RX	1 1

32位指令

操作： RZ ← RX - RY -（!C），C ← 借位

语法： subc32　rz, rx, ry

说明： 将 RX 的值减去寄存器 RY 的值和 C 位的非值，并把结果存在 RZ，借位存在 C 位。对于该减法指令来说，如果发生借位，将清 C 位，反之置 C 位。

影响标志位： C ← 借位

异常： 无

指令格式：

31	30	29	28	27	26	25	21	20	16	15	10	9	5	4	0
1	1	0	0	0	1	RY		RX		0	0	0	0	0	0	0	1	0	0	0	RZ

SUBI——无符号立即数减法指令

统一化指令

语法	操作	编译结果

subi rz, oimm12	RZ ← RZ - zero_extend(OIMM12)	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (oimm12<257) and (z<8), then
		subi16 rz, oimm8;
		else
		subi32 rz, rz, oimm12;

subi rz, rx, oimm12	RZ ← RX - zero_extend(OIMM12)	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (oimm12<8) and (z<8) and (x<8), then
		subi16 rz, rx, oimm3;
		elsif (x==z) and (z<8) and (oimm12<257), then
		subi16 rz, oimm8;
		else
		subi32 rz, rx, oimm12;

说明： 将带偏置 1 的 12 位立即数（OIMM12）零扩展至 32 位，然后用 RZ/RX 的值减去该 32 位数，把结果存入 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0x1-0x1000。

异常： 无

16位指令----1

操作： RZ ← RZ - zero_extend(OIMM8)

语法： subi16 rz, oimm8

说明： 将带偏置 1 的 8 位立即数（OIMM8）零扩展至 32 位，然后用 RZ 的值减去该 32 位数，把结果存入 RZ。

注意：二进制操作数 IMM8 等于 OIMM8 - 1。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7；立即数的范围为 1-256。

异常： 无

指令格式：

1514 1110 8 7 0

0	0 1 0 1	RZ	IMM8

IMM8 域——指定不带偏置立即数的值。

注意：寄存器减去的值 OIMM8 比起二进制操作数 IMM8 需偏置 1。

00000000——减 1

00000001——减 2

……

11111111——减 256

16位指令----2

操作： RZ ← RX - zero_extend(OIMM3)

语法： subi16 rz, rx, oimm3

说明： 将带偏置 1 的 3 位立即数（OIMM3）零扩展至 32 位，然后用 RX 的值减去该 32 位数，把结果存入 RZ。

注意：二进制操作数 IMM3 等于 OIMM3 - 1。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7；立即数的范围为 1-8。

异常： 无

指令格式：

1514 10 8 7 5 4 2 1 0

0	1 0 1 1	RX	RZ	IMM3	1 1

IMM3 域——指定不带偏置立即数的值。

注意：寄存器减去的值 OIMM3 比起二进制操作数 IMM3 需偏置 1。

000——减 1

001——减 2

……

111——减 8

32位指令

操作： RZ ← RX - zero_extend(OIMM12)

语法： subi32 rz, rx, oimm12

说明： 将带偏置 1 的 12 位立即数（OIMM12）零扩展至 32 位，然后用 RX 的值减去该 32 位数，把结果存入 RZ。

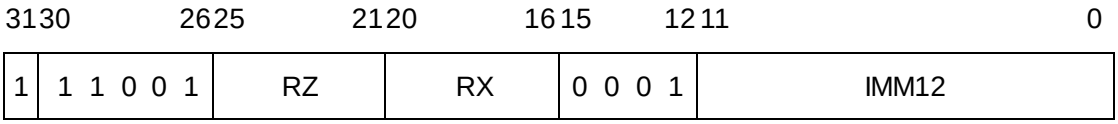
注意：二进制操作数 IMM12 等于 OIMM12 - 1。

影响标志位： 无影响

限制： 立即数的范围为 0x1-0x1000。

异常： 无

指令格式：



IMM12 域——指定不带偏置立即数的值。

注意：寄存器减去的值 OIMM12 比起二进制操作数 IMM12 需偏置 1。

000000000000——减 0x1

000000000001——减 0x2

……

111111111111——减 0x1000

SUBI(SP)——无符号（堆栈指针）立即数减法指令

统一化指令

语法	操作	编译结果
subi sp, sp, imm	SP ← SP- zero_extend(IMM)	仅存在 16 位指令。 subi sp, sp, imm

说明： 将立即数（IMM）零扩展至 32 位并左移 2 位，然后与堆栈指针（SP）的值相减，把结果存入 SP。

影响标志位： 无影响

限制： 立即数的范围为 0x0-0x1fc。

异常： 无

16位指令

操作： SP ← SP - zero_extend(IMM)

语法： subi sp, sp, imm

说明： 将立即数（IMM）零扩展至 32 位并左移 2 位，然后与堆栈指针（SP）的值相减，把结果存入堆栈指针。

注意：立即数（IMM）等于二进制操作数{IMM2，IMM5} << 2。

影响标志位： 无影响

限制： 源与目的寄存器均为堆栈指令寄存器（R14）；立即数的范围为(0x0-0x7f) << 2。

异常： 无

指令格式：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	IMM2		0	0	1	IMM5				

IMM 域——指定不带移位的立即数的值。

注意：加到寄存器里的值 IMM 比起二进制操作数{IMM2，IMM5}需左移 2 位。

{00, 00000}——减 0x0

{00, 00001}——减 0x4

……

{11, 11111}——减 0x1fc

SUBU——无符号减法指令

统一化指令

语法	操作	编译结果

subu rz, rx
sub rz, rx
	RZ ← RZ - RX	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (z<16) and (x<16), then
		subu16 rz, rx;
		else
		subu32 rz, rz, rx;

subu rz, rx, ry	RZ ← RX - RY	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (z<8) and (x<8) and (y<8), then
		subu16 rz, rx, ry;
		elsif (x==z) and (z<16) and (y<16), then
		subu16 rz, ry;
		else
		subu32 rz, rx, ry;

说明： 对于 subu rz, rx，将 RZ 的值减去 RX 值，并把结果存在 RZ 中。

对于 subu rz, rx, ry，将 RX 的值减去 RY 值，并把结果存在 RZ 中。

影响标志位： 无影响

异常： 无

16位指令----1

操作： RZ ← RZ - RX

语法： subu16 rz, rx

sub16 rz, rx

说明： 将 RZ 的值减去 RX 值，并把结果存在 RZ 中。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15	14	10	9	6	5	2	1	0
0	1 1 0 0 0		RZ		RX		1 0	

16位指令----2

操作： RZ ← RX- RY

语法： subu16 rz, rx, ry

sub16 rz, rx, ry

说明： 将 RX 的值减去 RY 值，并把结果存在 RZ 中。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r7。

异常： 无

指令格式：

15	14	11	10	8	7	5	4	2	1	0
0	1 0 1 1		RX		RZ		RY		0 1	

32位指令

操作： RZ ← RX - RY

语法： subu32 rz, rx, ry

说明： 将 RX 的值减去 RY 值，并把结果存在 RZ 中。

影响标志位： 无影响

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		RY		RX		0 0 0 0 0 0		0 0 1 0 0		RZ	

SYNC——CPU 同步指令

统一化指令

语法	操作	编译结果
sync	使 CPU 同步	仅存在 32 位指令。 sync32

说明： 当处理器碰到 sync 指令时，指令就会被悬挂起来直到所有外面的操作全都完成，即没有未完成的指令。

影响标志位： 无影响

异常： 无

32位指令

操作： 使 CPU 同步

语法： sync32

说明： 当处理器碰到 sync 指令时，指令就会被悬挂起来直到所有外面的操作全都完成，即没有未完成的指令。

影响标志位： 无影响

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 0		0 0 0 0 0		0 0 0 0 0		0 0 0 0 0 1		0 0 0 0 1		0 0 0 0 0	

TRAP——操作系统陷阱指令

统一化指令

语法	操作	说明
trap 0, trap 1 trap 2, trap 3	引起陷阱异常发生	仅存在 32 位指令。 trap32 0, trap32 1 trap32 2, trap32 3

说明： 当处理器碰到 trap 指令时，发生陷阱异常操作。

影响标志位： 无影响

异常： 陷阱异常

32位指令

操作： 引起陷阱异常发生

语法： trap32 0,
trap32 1,
trap32 2,
trap32 3

说明： 当处理器碰到 trap 指令时，发生陷阱异常操作。

影响标志位： 无影响

异常： 陷阱异常

指令格式：

trap32 0

31	30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9	8 7 6 5 4	3 2 1 0
1	1 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 1 0 0 0	0 0 0 0 1	0 0 0 0 0

trap32 1

31	30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9	8 7 6 5 4	3 2 1 0
1	1 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 1 0 0 1	0 0 0 0 1	0 0 0 0 0

trap32 2

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 0		0 0 0 0 0		0 0 0 0 0		0 0 1 0 1 0		0 0 0 0 1		0 0 0 0 0	

trap32 3

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 0		0 0 0 0 0		0 0 0 0 0		0 0 1 0 1 1		0 0 0 0 1		0 0 0 0 0	

TST——零测试指令

统一化指令

语法	操作	编译结果

tst rx, ry	If (RX & RY) != 0, then
	C ← 1;
	else
	C ← 0;
		仅存在 16 位指令。
		tst16 rx, ry

说明： 测试 RX 和 RY 的值按位与的结果。

如果结果不等于 0，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据按位与结果设置条件位 C

异常： 无

16位指令

操作： If (RX & RY) != 0, then

C ← 1;

else

C ← 0;

语法： tst16 rx, ry

说明： 测试 RX 和 RY 的值按位与的结果。

如果结果不等于 0，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据按位与结果设置条件位 C

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15 14 10 9 6 5 2 1 0

0	1 1 0 1 0	RY	RX	1 0

TSTNBZ——无字节等于零寄存器测试指令

统一化指令

语法	操作	编译结果
tstnbz16　rx	If　((RX[31:24] != 0) 　&(RX[23:16] != 0) 　&(RX[15: 8] != 0) 　&(RX[7 : 0] != 0)), then 　C ← 1; else 　C ← 0;	仅存在 16 位指令。 tstnbz16　rx

说明： 测试 RX 中是否没有字节等于零。如果 RX 没有字节等于零，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据按位与结果设置条件位 C

异常： 无

16位指令

操作：

```
If　( (RX[31:24] != 0)
　&(RX[23:16] != 0)
　&(RX[15: 8 ] != 0)
　&(RX[ 7 : 0 ] != 0) ), then
　C ← 1;
else
　C ← 0;
```

语法： tstnbz16　rx

说明： 测试 RX 中是否没有字节等于零。如果 RX 没有字节等于零，则设置条件位 C；否则，清除条件位 C。

影响标志位： 根据按位与结果设置条件位 C

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15	14	10	9	6	5	2	1	0
0	1 1 0 1 0		0 0 0 0		RX		1	1

WAIT——进入低功耗等待模式指令

统一化指令

语法	操作	编译结果
wait	进入低功耗等待模式	仅存在 32 位指令。 wait32

属性： 特权指令

说明： 此指令停止当前指令执行，并等待一个中断，此时 CPU 时钟停止。所有的外围设备都仍在继续运行，并有可能会产生中断而引起 CPU 从等待模式退出。

影响标志位： 无影响

异常： 特权违反指令

32位指令

操作： 进入低功耗等待模式

语法： wait32

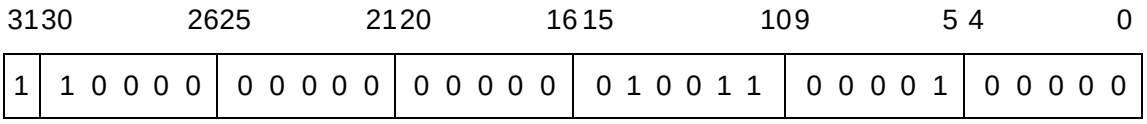
属性： 特权指令

说明： 此指令停止当前指令执行，并等待一个中断，此时 CPU 时钟停止。所有的外围设备都仍在继续运行，并有可能会产生中断而引起 CPU 从等待模式退出。

影响标志位： 无影响

异常： 特权违反指令

指令格式：



XOR——按位异或指令

统一化指令

语法	操作	编译结果

xor rz, rx	RZ ← RZ ^ RX	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (x<16) and (z<16), then
		xor16 rz, rx;
		else
		xor32 rz, rz, rx;

xor rz, rx, ry	RZ ← RX ^ RY	根据寄存器的范围编译为对应的 16 位或 32 位指令。
		if (y==z) and (z<16) and (x<16), then
		xor16 rz, rx;
		else
		xor32 rz, rx, ry;

说明： 将 RX 与 RZ/RY 的值按位异或，并把结果存在 RZ。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← RZ ^ RX

语法： xor16 rz, rx

说明： 将 RZ 与 RX 的值按位异或，并把结果存在 RZ。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15 14 10 9 6 5 2 1 0

0	1 1 0 1 1	RZ	RX	0 1

32位指令

操作： RZ ← RX ^ RY

语法： xor32 rz, rx, ry

说明： 将 RX 与 RY 的值按位异或，并把结果存在 RZ。

影响标志位： 无影响

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		RY		RX		0 0 1 0 0 1		0 0 0 1 0		RZ	

XORI——立即数按位异或指令

统一化指令

语法	操作	编译结果
xori rz, rx, imm16	RZ ← RX ^ zero_extend(IMM12)	仅存在 32 位指令。 xori32 rz, rx, imm12

说明： 将 12 位立即数零扩展至 32 位，然后与 RX 的值进行按位异或操作，把结果存入 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0x0-0xFFF。

异常： 无

32位指令

操作： RZ ← RX ^ zero_extend(IMM12)

语法： xori32 rz, rx, imm12

说明： 将 12 位立即数零扩展至 32 位，然后与 RX 的值进行按位异或操作，把结果存入 RZ。

影响标志位： 无影响

限制： 立即数的范围为 0x0-0xFFF。

异常： 无

指令格式：

31	30 26	25 21	20 16	15 12	11 0
1	1 1 0 0 1	RZ	RX	0 1 0 0	IMM12

XSR——扩展右移指令

统一化指令

语法	操作	编译结果
xsr rz, rx, oimm5	{RZ,C} ←{RX,C} >>>> OIMM5	仅存在 32 位指令。 xsr32 rz, rx, oimm5

说明： 将 RX 带条件位 C 的值（{RX,C}）进行循环右移（原值右移，左侧移入右侧移出的位），把移位结果的最低位（[0]）存入条件位 C，高位（[32:1]）存入 RZ，右移位数由带偏置 1 的 5 位立即数（OIMM5）的值决定。如果 OIMM5 的值等于 32，那么条件位 C 为 RX 的最高位。

影响标志位： C ← RX[OIMM5 - 1]

限制： 立即数的范围为 1-32。

异常： 无

32位指令

操作： {RZ,C} ←{RX,C} >>>> OIMM5

语法： xsr32 rz, rx, oimm5

说明： 将 RX 带条件位 C 的值（{RX,C}）进行循环右移（原值右移，左侧移入右侧移出的位），把移位结果的最低位（[0]）存入条件位 C，高位（[32:1]）存入 RZ，右移位数由带偏置 1 的 5 位立即数（OIMM5）的值决定。如果 OIMM5 的值等于 32，那么条件位 C 为 RX 的最高位。

注意：二进制操作数 IMM5 等于 OIMM5 - 1。

影响标志位： C ← RX[OIMM5 - 1]

限制： 立即数的范围为 1-32。

异常： 无

指令格式：

31	30	26	25	21	20	16	15	10	9	5	4	0
1	1 0 0 0 1		IMM5		RX		0 1 0 0 1 1		0 1 0 0 0		RZ	

IMM5 域——指定不带偏置立即数的值。

注意：移位的值 OIMM5 比起二进制操作数 IMM5 需偏置 1。

00000——移 1 位

00001——移 2 位

……

11111——移 32 位

XTRB0——提取字节 0 并无符号扩展指令

统一化指令

语法	操作	编译结果

xtrb0　rz, rx	RZ ← zero_extend(RX[31:24]);
	if (RX[31:24] == 0), then
	　C ← 0;
	else
	　C ← 1;
		仅存在 32 位指令。
		xtrb0.32　rz, rx

说明： 提取 RX 的字节 0（RX[31:24]）到 RZ 的低位（RZ[7:0]），并进行零扩展。如果结果等于 0，则清除 C 位，反之设置 C 位。

影响标志位： 如果结果等于 0，则清除 C 位，反之设置 C 位。

异常： 无

32位指令

操作：
```
RZ ← zero_extend(RX[31:24]);
if (RX[31:24] == 0), then
  C ← 0;
else
  C ← 1;
```

语法： xtrb0.32　rz, rx

说明： 提取 RX 的字节 0（RX[31:24]）到 RZ 的低位（RZ[7:0]），并进行零扩展。如果结果等于 0，则清除 C 位，反之设置 C 位。

影响标志位： 如果结果等于 0，则清除 C 位，反之设置 C 位。

异常： 无

指令格式：

31	30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10	9 8 7 6 5	4 3 2 1 0
1	1 0 0 0 1	0 0 0 0 0	RX	0 1 1 1 0 0	0 0 0 0 1	RZ

XTRB1——提取字节 1 并无符号扩展指令

统一化指令

语法	操作	编译结果

xtrb1　rz, rx	RZ ← zero_extend(RX[23:16]);
	if (RX[23:16] == 0), then
	　C ← 0;
	else
	　C ← 1;
		仅存在 32 位指令。
		xtrb1.32　rz, rx

说明： 提取 RX 的字节 1（RX[23:16]）到 RZ 的低位（RZ[7:0]），并进行零扩展。如果结果等于 0，则清除 C 位，反之设置 C 位。

影响标志位： 如果结果等于 0，则清除 C 位，反之设置 C 位。

异常： 无

32位指令

操作：
```
RZ ← zero_extend(RX[23:16]);
if (RX[23:16] == 0), then
  C ← 0;
else
  C ← 1;
```

语法： xtrb1.32　rz, rx

说明： 提取 RX 的字节 1（RX[23:16]）到 RZ 的低位（RZ[7:0]），并进行零扩展。如果结果等于 0，则清除 C 位，反之设置 C 位。

影响标志位： 如果结果等于 0，则清除 C 位，反之设置 C 位。

异常： 无

指令格式：

31	30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10	9 8 7 6 5	4 3 2 1 0
1	1 0 0 0 1	0 0 0 0 0	RX	0 1 1 1 0 0	0 0 0 1 0	RZ

XTRB2——提取字节 2 并无符号扩展指令

统一化指令

语法	操作	编译结果

xtrb2　rz, rx	RZ ← zero_extend(RX[15:8]);
	if (RX[15:8] == 0), then
	　C ← 0;
	else
	　C ← 1;
		仅存在 32 位指令。
		xtrb2.32　rz, rx

说明： 提取 RX 的字节 2（RX[15:8]）到 RZ 的低位（RZ[7:0]），并进行零扩展。如果结果等于 0，则清除 C 位，反之设置 C 位。

影响标志位： 如果结果等于 0，则清除 C 位，反之设置 C 位。

异常： 无

32位指令

操作：
```
RZ ← zero_extend(RX[15:8]);
if (RX[15:8] == 0), then
  C ← 0;
else
  C ← 1;
```

语法： xtrb2.32　rz, rx

说明： 提取 RX 的字节 2（RX[15:8]）到 RZ 的低位（RZ[7:0]），并进行零扩展。如果结果等于 0，则清除 C 位，反之设置 C 位。

影响标志位： 如果结果等于 0，则清除 C 位，反之设置 C 位。

异常： 无

指令格式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	1	0	0	0	0	0	RX					0	1	1	1	0	0	0	0	1	0	0	RZ				

XTRB3——提取字节 3 并无符号扩展指令

统一化指令

语法	操作	编译结果

xtrb3　rz, rx	RZ ← zero_extend(RX[7:0]);
	if (RX[7:0] == 0), then
	　C ← 0;
	else
	　C ← 1;
		仅存在 32 位指令。
		xtrb3.32　rz, rx

说明： 提取 RX 的字节 3（RX[7:0]）到 RZ 的低位（RZ[7:0]），并进行零扩展。如果结果等于 0，则清除 C 位，反之设置 C 位。

影响标志位： 如果结果等于 0，则清除 C 位，反之设置 C 位。

异常： 无

32位指令

操作：
```
RZ ← zero_extend(RX[7:0]);
if (RX[7:0] == 0), then
  C ← 0;
else
  C ← 1;
```

语法： xtrb3.32　rz, rx

说明： 提取 RX 的字节 3（RX[7:0]）到 RZ 的低位（RZ[7:0]），并进行零扩展。如果结果等于 0，则清除 C 位，反之设置 C 位。

影响标志位： 如果结果等于 0，则清除 C 位，反之设置 C 位。

异常： 无

指令格式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	1	0	0	0	0	0	RX					0	1	1	1	0	0	0	1	0	0	0	RZ				

ZEXTB——字节提取并无符号扩展指令#

统一化指令

语法	操作	编译结果
zextb rz, rx	RZ ← zero_extend(RX[7:0]);	仅存在 16 位指令。 zextb16 rz, rx;

说明： 将 RX 的低字节（RX[7:0]）零扩展至 32 位，结果存在 RZ 中。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← zero_extend(RX[7:0]);

语法： zextb16 rz, rx

说明： 将 RX 的低字节（RX[7:0]）零扩展至 32 位，结果存在 RZ 中。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式：

15	14 10	9 6	5 2	1 0
0	1 1 1 0 1	RZ	RX	0 0

ZEXTH——半字提取并无符号扩展指令#

统一化指令

语法	操作	编译结果
zexth rz, rx	RZ ← zero_extend(RX[15:0]);	仅存在 16 位指令。 zexth16 rz, rx

说明： 将 RX 的低半字（RX[15:0]）零扩展至 32 位，结果存在 RZ 中。

影响标志位： 无影响

异常： 无

16位指令

操作： RZ ← zero_extend(RX[15:0]);

语法： zexth16 rz, rx

说明： 将 RX 的低半字（RX[15:0]）零扩展至 32 位，结果存在 RZ 中。

影响标志位： 无影响

限制： 寄存器的范围为 r0-r15。

异常： 无

指令格式

15 14 10 9 6 5 2 1 0

0	1 1 1 0 1	RZ	RX	0 1