1. 介绍

UniCore32 指令系统体系结构是北大众志自主定义的 RISC 类型指令系统体系结构。UniCore32 采用装入/存储形式的通用寄存器结构。内存地址空间采用32 位线性寻址，每一个内存单元是一个字节，字长为 32 位，采用小尾(Little Endian)存储格式。UniCore32 指令系统体系结构的设计目标是高性能、高代码密度、低功耗和实时性。

MPRC CONFIDENTIAL

2. 编程模型

2.1. 数据类型

2.1.1. 定点数据类型

UniCore32 指令系统定义如下数据类型：

- 位 (Bit)
- 字节 (Byte, 8 Bits)
- 半字 (Halfword, 16 Bits)
- 字 (Word, 32 Bits)

2.1.2. 内存字节序

针对位宽大于字节的数据类型，UniCore32 指令系统采用小尾（Little-Endian）存储格式。图 2-1 简要描述了 UniCore32 中 Word 数据类型的内存字节序。

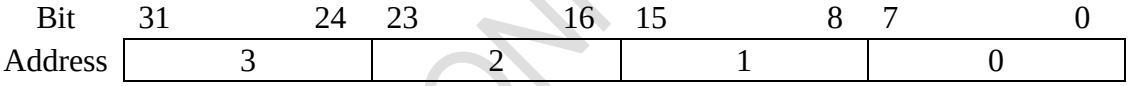

图 2-1 UniCore32 字（Word）数据类型的内存字节序

2.2. 寄存器定义

2.2.1. 定点通用寄存器

UniCore32 指令系统定义了 32 个通用寄存器，这些寄存器中有两个寄存器被赋予特殊用途：

- r30 寄存器在转移连接指令执行中以及异常处理流程中，用来保留返回地址，用 LR 表示。
- r31 固定作为程序计数寄存器，用 PC 表示。

UniCore32 中某些通用寄存器在不同的运行模式会使用不同的影子寄存器（shadow register），其具体分配详见《UniCore32 特权体系结构手册》。图 2-2 简要描述了定点通用寄存器的布局。

MRC CONFIDENTIAL

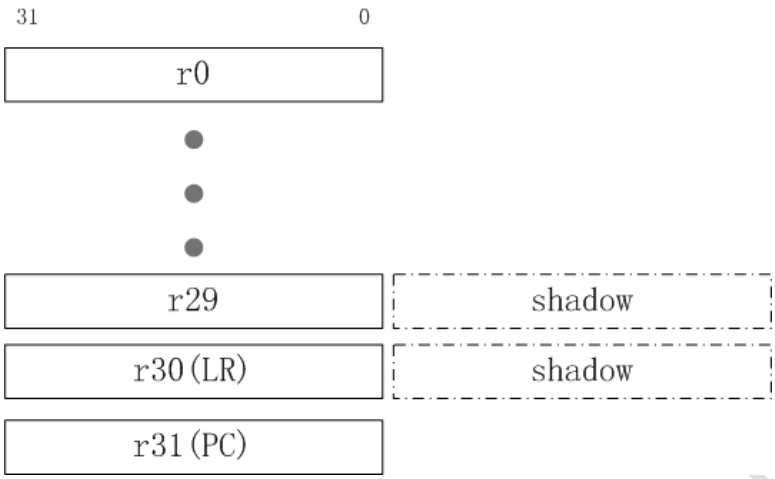

图 2-2 通用寄存器布局

MPRC CONFIDENTIAL

3. 指令系统综述

在 UniCore32 指令系统中，每条指令采用 32 位固定长度编码，指令存放地址为 32 位字对齐，汇编指令采用“助记符”加“操作数”的基本指令格式。

3.1. 指令助记符

指令助记符分为两部分，它们之间采用“.”作为分割符，第一部分说明了该指令的主要操作，第二部分说明了该指令在此操作下的变化，说明如表 3-1。

表 3-1 助记符第二部分说明

符号	说明
A	该指令将改变 ASR 中的内容
F	该指令将改变 ASR 或 BSR 的标志位 F （包括溢出标志 V、进位标志 C、全零标志 Z、符号标志 S）
L	该指令将返回地址放入 r30 寄存器中
U	该指令执行用户地址空间的存取操作，并回写基址寄存器 （该指令只能在非用户模式下运行）
W	该指令回写基址寄存器

部分指令助记符中会采用条件位编码，指令编码中占 4 位位域，说明如表 3-2。

表 3-2 条件位编码说明

条件位编码（bc）	标志位条件（fc）	条件执行指令标志助记符（cc）		条件跳转指令标志助记符（cc）	
0000	Z=1	FZ	全零标志	EQ	等于
0001	Z=0	NZ	无全零标志	NE	不等于
0010	C=1	FC	进位标志	EA	无符号数的大于等于
0011	C=0	NC	无进位标志	UB	无符号数的小于
0100	S=1	FS	符号标志		
0101	S=0	NS	无符号标志		
0110	V=1	FV	溢出标志		
0111	V=0	NV	无溢出标志		
1000	C=1 AND Z=0			UA	无符号数的大于
1001	C=0 OR Z=1			EB	无符号数的小于等于
1010	S=V			EG	有符号数的大于等于
1011	NOT S=V			SL	有符号数的小于
1100	Z=0 AND (S=V)			SG	有符号数的大于
1101	Z=1 OR (NOT S=V)			EL	有符号数的小于等于
1110~1111	保留				

MRC CONFIDENTIAL

3.2. 伪码描述中特殊符号说明

后续的指令详述中会使用伪码方式描述指令的功能，表 3-3 列出所有伪码中特殊符号的含义。

表 3-3 伪码特殊符号说明

符号	含义
←	赋值，变量的位宽在赋值时确定
==	等于
>=	大于等于
!=	不等于
GPR[rn]	通用寄存器中第 rn 寄存器，当 rn 寄存器具有影子寄存器(shadow)时，具体访问的寄存器由当前运行模式确定
{m, n}	将二进制数 m 和 n 拼接，m 在高位
m{n}	将二进制数 n 复制 m 份并拼接起来
MODE	当前处理器运行模式
USER	用户模式
ASR	ASR 寄存器
SPECIAL	SPECIAL 寄存器
BSR.mode	当前运行模式下的 BSR 影子寄存器(shadow)
%	算数取模
INSTRUCTION	表示正在执行指令的编码
PC	表示正在执行指令的指令地址值
AZ	All Zero，表示编码为全 0
AO	All One，表示编码为全 1

3.3. 指令操作数

指令操作数分为目的操作数、源操作数和地址操作数；在排列顺序上，目的操作数总是在最前面，地址操作数总是在最后面，不同操作数之间均采用逗号“,”作为分隔符。

操作数的寻址方式分为两大类，一类为简单寻址方式，另一类为带移位的寻址方式，共有 9 种寻址方式，操作数和寻址方式的对应关系如表 3-4 所示。

程序计数寄存器 r31 作为数据寄存器，其使用方式和取值较为特殊。RSR 寻址方式中的移位次数寄存器不能采用 r31；偏移地址操作数中的寄存器不能采用 r31；当 r31 作为源操作数、基址地址操作数或者 RSI/RSR 寻址方式中的操作数时，所取得的值为指令的当前地址加 4，但是在 STB/STH/STW 指令和 MOV.pp 指令中，当 r31 作为源操作数时，所取得的值为指令的当前地址加 8。

XPRC CONFIDENTIAL

表 3-4 操作数与寻址方式的对应关系

寻址方式		目的操作数		源操作数			地址操作数	
		运算	存取	运算	存取	控制	基址	偏移
简单寻址方式	BSR 寻址方式	可以		可以				
	ASR 寻址方式	可以	可以	可以				
	REG 寻址方式	可以	可以	可以	可以	可以	可以	可以
	CPR 寻址方式	可以		可以				
	R16 寻址方式		可以		可以			
	IMM 寻址方式			可以		可以		可以
带移位的寻址方式	ISI 寻址方式			可以				
	RSI 寻址方式			可以				可以
	RSR 寻址方式			可以				

3.3.1. 简单寻址方式

简单寻址方式有六种，如表 3-5 所示。

表 3-5 简单寻址方式说明

寻址方式	通用格式	含义
ASR 寻址方式	ASR	ASR 寄存器
BSR 寻址方式	BSR	BSR 寄存器
REG 寻址方式	Rnn	数据寄存器
CPR 寻址方式	P(pp).C(nn)	协处理器寄存器
R16 寻址方式	{Rnn~Rnn}	多个寄存器的列表
IMM 寻址方式	u#imm	无符号立即数
	s#imm	有符号立即数

ASR 寻址方式表示的是活动状态寄存器 ASR；BSR 寻址方式表示的是后台状态寄存器 BSR，该寻址方式只能在特权模式、中断模式、陷入模式、扩展模式或实时模式下运行，并且所访问的是当前运行模式下可见的后台状态寄存器。

CPR 寻址方式表示的是协处理器寄存器，格式为 P(pp).C(nn)，其中(pp) 为协处理器编码，在指令编码中占 4 位位域，说明如表 3-6。

表 3-6 协处理器编码说明

编码	协处理器名称	格式 (pp)
0000	系统控制协处理器	P0
0001	片内调试协处理器	P1
0010	浮点协处理器	P2
0100~1111	保留	

TBRC CONFIDENTIAL

(nn)为协处理器的寄存器编码，在指令编码中占 5 位位域，编码说明如表 3-7。

表 3-7 协处理器寄存器的编码说明

编码	寄存器	编码	寄存器	编码	寄存器	编码	寄存器
00000	C00	00001	C01	00010	C02	00011	C03
00100	C04	00101	C05	00110	C06	00111	C07
01000	C08	01001	C09	01010	C10	01011	C11
01100	C12	01101	C13	01110	C14	01111	C15
10000	C16	10001	C17	10010	C18	10011	C19
10100	C20	10101	C21	10110	C22	10111	C23
11000	C24	11001	C25	11010	C26	11011	C27
11100	C28	11101	C29	11110	C30	11111	C31

R16 寻址方式表示的是多个寄存器的列表，通常 16 个一组，采用括号“(”和“)”括在一起，逗号“,”作为分隔符，减号“-”作为连续多个寄存器的省略符。例如，(r0, r3-r5, r8)表示为以下 5 个数据寄存器的列表：r0，r3，r4，r5 和 r8。

IMM 寻址方式表示的是立即数，区分为无符号数和有符号数。

3.3.2. 带移位的寻址方式

带移位的寻址方式所获取的值都是采用操作数进行移位操作后的结果。

移位操作有四种，编码采用 2 位位域，说明如表 3-8 所示。

表 3-8 移位操作描述

移位操作	符号表示	编码	移空部分	移位进位
左移	<<	00	补 0	移出部分的最低有效位
逻辑右移	>>	01	补 0	移出部分的最高有效位
算术右移	|>	10	操作数的最高位	移出部分的最高有效位
循环右移	<>	11	操作数的移出部分	移出部分的最高有效位

左移(<<)对操作数的每一位向左移动指定的位数，移空的部分补 0，移出的部分舍弃；逻辑右移(>>)对操作数的每一位向右移动指定的位数，移空的部分补 0，移出的部分舍弃；算术右移(|>)类似于逻辑右移，区别在于移空的部分填入的是操作数的最高位，而不是 0，这样保存了二进制补码的符号位；循环右移(<>)类似于逻辑右移，区别在于移空的部分填入的是移出的数据位，而不是 0。

当采用移位操作的指令是逻辑运算时，如果需要根据指令执行的结果设置活动状态寄存器 ASR 中的标志位，则移位进位将被保存到 ASR 的进位标志上，否则，移位进位被舍弃。

带移位的寻址方式有三种，如表 3-9 所示，其中(s)泛指上述 4 种移位操作。

MPRC CONFIDENTIAL

表 3-9 带移位的寻址方式说明

寻址方式	通用格式	操作数	移位次数	其它说明
ISI 寻址方式	u# <> u#5	无符号数	5 位无符号数	当 u#5==0 时，可缩写为 u#
RSI 寻址方式	reg (s) u#5	寄存器	5 位无符号数	当(s)为左移，且 u#5==0 时，可缩写为 reg
RSR 寻址方式	reg (s) reg	寄存器	另一个寄存器的最低字节中	移位次数寄存器不能采用程序计数寄存器 r31

ISI 寻址方式表示的是无符号数根据循环右移进行移位的结果，移位次数存放在一个 5 位的无符号数中，移位进位被舍弃。

RSI 寻址方式表示的是寄存器根据某种移位类型进行移位的结果，移位次数存放在一个 5 位的无符号数中；当移位次数的取值范围为 1~31 时，移位结果和移位进位均按照移位操作的类型进行；当移位次数的取值为 0 时，针对不同的移位操作的类型，移位结果和移位进位的对应关系如表 3-10 所示。

表 3-10 移位次数为 0 时 RSI 寻址方式的移位结果与移位进位

移位操作	格式	移位结果	移位进位
左移	reg << 0 或 reg	reg	不影响
逻辑右移	reg >> #32	全’0’	reg[31]
算术右移	reg >> #32	全’0’：如果 reg[31]为’0’ 全’1’：如果 reg[31]为’1’	reg[31]
循环右移	reg <> #33	reg 的每一位向右移动一位，同时，reg[31]填入进位标志	reg[0]

表 3-11 RSR 寻址方式的移位结果与移位进位

移位操作		移位次数 N		
		0	**1~31**	**32~255**
移位结果	左移	reg	reg << N	全’0’
	逻辑右移		reg >> N	全’0’
	算术右移		reg |> N	全’0’：如果 reg[31]为’0’ 全’1’：如果 reg[31]为’1’
	循环右移		reg <> N	reg：如果 N%32==0 reg <> N%32：如果 N%32!=0
移位操作		**移位次数 N**		
		0	**1~32**	**33~255**
移位进位	左移	不影响	reg[32-N]	0
	逻辑右移		reg[N-1]	0
	算术右移			reg[31]
	循环右移			reg[31]：如果 N%32==0 reg[N%32-1]：如果 N%32!=0

MPRC CONFIDENTIAL

RSR 寻址方式表示的是寄存器根据某种移位类型进行移位的结果，移位次数存放在另一个寄存器的最低字节中；移位次数寄存器不能采用程序计数寄存器r31，根据该寄存器的最低字节的取值，RSR 寻址方式的移位结果与移位进位如表 3-11 所示。

3.3.3. 基址地址与偏移地址

基址地址操作数只能采用 REG 寻址方式，因此，简称为基址寄存器。为了区分基址寄存器，在含有基址寄存器的操作数列表中，统一采用中括号“[”和“]”把基址寄存器括起来。

存/取数据指令的地址操作数都含有偏移操作，在指令格式中，采用加号“+”和减号“-”与右中括号“]”的位置作为区别，如表 3-12 所示。

表 3-12 偏移操作描述

偏移操作	格式	地址	是否回写
后减	[reg]-	基址寄存器	回写
后加	[reg]+	基址寄存器	回写
先减	[reg-]	基址寄存器 减 偏移地址	不影响
先加	[reg+]	基址寄存器 加 偏移地址	不影响

如果基址寄存器为程序计数寄存器，那么所取得的值为指令的当前地址加 4，而且不能回写基址寄存器。

偏移地址操作数有 REG/IMM/RSI 三种寻址方式，其中的寄存器都不能采用程序计数寄存器 R31。

基址寄存器和偏移地址操作数中的寄存器不能采用同一个寄存器。

取数据类指令包括取字节操作，取半字操作，取整字操作和取多字操作，其中的地址操作数都采用“基址寄存器”加/减“偏移地址”的方式，并且偏移地址操作数都支持 IMM 寻址方式，如表 3-13 所示。

表 3-13 取数据类指令的数据类型与操作数说明

	取数据操作			偏移地址操作数	
	数据类型	符号扩展	目的操作数	IMM 寻址方式	其它寻址方式
LDB	字节	无符号扩展	REG	14 位无符号数	RSI
LDSB	字节	有符号扩展	REG	10 位无符号数	REG
LDH	半字	无符号扩展	REG	10 位无符号数	REG
LDSH	半字	有符号扩展	REG	10 位无符号数	REG
LDW	整字		REG	14 位无符号数	RSI
LDM	多字		R16	4（省略不写）	
LDUR	多字		R16	4（省略不写）	

存数据类指令包括存字节操作，存半字操作，存整字操作和存多字操作，其

MPRC CONFIDENTIAL

中的地址操作数都采用“基址寄存器”加/减“偏移地址”的方式，并且偏移地址操作数都支持 IMM 寻址方式，如表 3-14 所示。

表 3-14 存数据类指令的数据类型与操作数说明

	数据类型	目的操作数	偏移地址操作数	
			IMM 寻址方式	其它寻址方式
STB	字节	REG	14 位无符号数	RSI
STH	半字	REG	10 位无符号数	REG
STW	整字	REG	14 位无符号数	RSI
STM	多字	R16	4（省略不写）	
STUR	多字	R16	4（省略不写）	

MPRC CONFIDENTIAL

4. 指令系统详述

4.1. ADD

ADD register ADD, immediate logic shift to left, no flag-update

ADD rd, rs1, rs2 << #s_imm5 ; s_imm5 != 0

ADD rd, rs1, rs2 ; s_imm5 = = 0

31	23	18	13	8	4
operation 08H	rs1	rd	s_imm5	function 0H	rs2

指令描述：

该指令首先对 rs2 寄存器中的数据进行逻辑左移 s_imm5 位操作，移位后的结果与 rs1 寄存器中的数据进行**算术加**操作，计算结果存入 rd 寄存器。

指令操作：

operatorA ← GPR[rs1]

operatorB ← {GPR[rs2][31-s_imm5:0], s_imm5{0}}

GPR[rd] ← operatorA + operatorB

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

ADD immediate ADD, immediate arithmetic rotate to right, no flag-update

ADD rd, rs1, imm9 ; s_imm5 == 0

ADD rd, rs1, imm9 <> s_imm5 ; s_imm5 != 0

31	23	18	13	8
operation 28H	rs1	rd	s_imm5	imm9

指令描述：

该指令对 IMM 立即数与 rs1 寄存器中的数据进行**算术加**操作，计算结果存入 rd 寄存器中。其中 IMM 立即数是 imm9 循环右移 s_imm5 位的结果。

指令操作：

```
tmp ← {23{0},  imm9}
IF (s_imm5 == 0)
    operatorA ← tmp
ELSE
    operatorA ← {tmp[s_imm5-1:0],  tmp[31: s_imm5]}
ENDIF

operatorB ← GPR[rs1]

GPR[rd] ← operatorA + operatorB
```

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

4.2. SUB

SUB register SUB, immediate logic shift to left, no flag-update

SUB rd, rs1, rs2 << #s_imm5 ; s_imm5 != 0

SUB rd, rs1, rs2 ; s_imm5 == 0

31	23	18	13	8	4
operation 4H	rs1	rd	s_imm5	function 0H	rs2

指令描述：

该指令首先对 rs2 寄存器的数据逻辑左移 s_imm5 位，然后将 rs1 寄存器中的数据**算术减**去移位后的结果，并将结果存入 rd 寄存器中。

指令操作：

operatorA ← GPR[rs1]

operatorB ← {GPR[rs2][31-s_imm5:0], s_imm5{0}}

GPR[rd] ← operatorA + (~operatorB) + 1

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

SUB immediate SUB, immediate arithmetic rotate to right, no flag-update

SUB rd, rs1, imm9 ; s_imm5 == 0

SUB rd, rs1, imm9 <> s_imm5 ; s_imm5 != 0

31	23	18	13	8
operation 24H	rs1	rd	s_imm5	imm9

指令描述：

该指令将 rs1 寄存器中数据**算术减**去 IMM 立即数，将结果存入 rd 寄存器中。其中 IMM 立即数是 imm9 循环右移 s_imm5 位的结果。

指令操作：

tmpA ← {23{0}, imm9}
IF (s_imm5 == 0)
operatorA ← ~tmpA
ELSE
operatorA ← ~{tmpA[s_imm5-1:0], tmpA[31: s_imm5]}
ENDIF

operatorB ← GPR[rs1]

GPR[rd] ← operatorA + operatorB + 1

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

4.3. MUL

MUL MUL, no flag-update

MUL rd, rs1, rs2

31	23	18	13	8	4
operation 00H	rs1	rd	s_imm5 0H	function 9H	rs2

指令描述：

该指令将 rs1 寄存器中的数据和 rs2 寄存器中的数据进行**算术乘**，并将结果的低 32 位存入 rd 寄存器中。

指令操作：

GPR[rd] ← GPR[rs1] * GPR[rs2]

MPRC CONFIDENTIAL

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

MULSL signed long MUL, no flag-update

MULSL rd1, rd2, rs1, rs2

31	23	18	13	8	4
operation 08H	rs1	rd1	rd2	function 9H	rs2

指令描述：

该指令将 rs1 寄存器中的数据和 rs2 寄存器中的数据进行**算术乘**，并将结果的低 32 位存入 rd1 寄存器中，高 32 位存入 rd2 寄存器中。其中所有的操作数都作为有符号数。

指令操作：

{GPR[rd2], GPR[rd1]} ← GPR[rs1] * GPR[rs2]

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

4.4. AND

AND register AND, immediate logic shift to left, no flag-update

AND rd, rs1, rs2 << #s_imm5 ; s_imm5 != 0

AND rd, rs1, rs2 ; s_imm5 == 0

31	23	18	13	8	4
operation 00H	rs1	rd	s_imm5	function 0H	rs2

指令描述：

该指令首先对 rs2 寄存器中的数据进行逻辑左移 s_imm5 位操作，移位后的结果与 rs1 寄存器中的数据进行**逻辑与**操作，计算结果存入 rd 寄存器。

指令操作：

operatorA ← GPR[rs1]

operatorB ← {GPR[rs2][31-s_imm5:0], s_imm5{0}}

GPR[rd] ← operatorA & operatorB

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

AND immediate,, immediate arithmetic rotate to right, no flag-update

AND rd, rs1, imm9 ; s_imm5 == 0

AND rd, rs1, imm9 <> s_imm5 ; s_imm5 != 0

31	23	18	13	8
operation 20H	rs1	rd	s_imm5	imm9

指令描述：

该指令将 IMM 立即数与 rs1 寄存器中内容进行**逻辑与**操作，将结果放入 rd 寄存器中。需要说明的是，IMM 立即数是由 imm9 循环右移 s_imm5 位得到。

指令操作：

```
tmp ← {23{0},  imm9}
IF (s_imm5 == 0)
    operatorA ← tmp
ELSE
    operatorA ← {tmp[s_imm5-1:0],  tmp[31: s_imm5]}
ENDIF

operatorB ← GPR[rs1]

GPR[rd] ← operatorA & operatorB
```

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

4.5. OR

OR register OR, immediate logic shift to left, no flag-update

OR rd, rs1, rs2 << #s_imm5 ; s_imm5 != 0

OR rd, rs1, rs2 ; s_imm5 == 0

31	23	18	13	8	4
operation 18H	rs1	rd	s_imm5	function 0H	rs2

指令描述：

该指令首先对 rs2 寄存器的数据逻辑左移 s_imm5 位，然后将移位后的结果与 rs1 寄存器中的数据进行**逻辑或**，并将结果存入 rd 寄存器中。

指令操作：

operatorA ← GPR[rs1]

operatorB ← {GPR[rs2][31-s_imm5:0], s_imm5{0}}

GPR[rd] ← operatorA | operatorB

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

OR immediate OR, immediate arithmetic rotate to right, no flag-update

OR rd, rs1, imm9 ; s_imm5 == 0

OR rd, rs1, imm9 <> s_imm5 ; s_imm5 != 0

31	23	18	13	8
operation 38H	rs1	rd	s_imm5	imm9

指令描述：

该指令将 IMM 立即数与 rs1 寄存器中数据进行**逻辑或**，将结果存入 rd 寄存器中。其中 IMM 立即数是 imm9 循环右移 s_imm5 位的结果。

指令操作：

```
tmp ← {23{0},  imm9}
IF (s_imm5 == 0)
    operatorA ← tmp
ELSE
    operatorA ← {tmp[s_imm5-1:0],  tmp[31: s_imm5]}
ENDIF

operatorB ← GPR[rs1]

GPR[rd] ← operatorA | operatorB
```

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

4.6. MOV

MOV immediate MOV, immediate arithmetic rotate to right, no flag-update

MOV rd, imm9 ; s_imm5 == 0

MOV rd, imm9 <> s_imm5 ; s_imm5 != 0

31	23	18	13	8
operation 3AH	rs1 0H	rd	s_imm5	imm9

指令描述：

该指令将 IMM 立即数存入 rd 寄存器中。其中 IMM 立即数是 imm9 循环右移 s_imm5 位的结果。

指令操作：

MPRC CONFIDENTIAL

```
tmp ← {23{0},  imm9}
IF (s_imm5 == 0)
    operatorA ← tmp
ELSE
    operatorA ← {tmp[s_imm5-1:0],  tmp[31: s_imm5]}
ENDIF

GPR[rd] ← operatorA
```

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

MOV register move, immediate logic shift to left, no flag-update

MOV rd, rs1, rs2 << #s_imm5 ; s_imm5 != 0

MOV rd, rs1, rs2 ; s_imm5 == 0

31	23	18	13	8	4
operation 1AH	rs1 0H	rd	s_imm5	function 0H	rs2

指令描述：

该指令首先对 rs2 寄存器中的数据逻辑左移 s_imm5 位，然后将移位后的结果放到 rd 寄存器中。

指令操作：

operatorA ← {GPR[rs2][31-s_imm5:0], s_imm5{0}}

GPR[rd] ← operatorA

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

4.7. LDB

LDB pre-add load byte, immediate offset, no write back

LDB rd, [rs1+], #imm14 ; imm14 ! = 0

LDB rd, [rs1] ; imm14 = = 0

31	23	18	13
operation 7DH	rs1	rd	imm14

指令描述：

该指令将该 imm14 进行无符号扩展，然后与 rs1 寄存器中的数据相加，并将加的结果作为地址访问内存，取回来一个字节的数据并进行无符号扩展，最后将该数值放到 rd 寄存器中。

指令操作：

base ← GPR[rs1]

offset ← {18{0}， imm14}

address ← base + offset

tmpByte ← Memory[address].Byte

GPR[rd] ← {24{0}， tmpByte}

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

LDB pre-add load byte, register offset, immediate logic shift to left, no write back

LDB rd, [rs1+], rs2 << #s_imm5 ; s_imm5 ! = 0

LDB rd, [rs1+], rs2 ; s_imm5 = = 0

31	23	18	13	8	4
operation 5DH	rs1	rd	s_imm5	function 0H	rs2

指令描述：

该指令首先对 rs2 寄存器中的数据逻辑左移 s_imm5 位，然后将该数值与 rs1 寄存器中的数据相加，并将加的结果作为地址访问内存，取回来一个字节的数据并进行无符号扩展，最后将该数值放到 rd 寄存器中。

指令操作：

base ← GPR[rs1]

offset ← {GPR[rs2][31-s_imm5:0], s_imm5{0}}

address ← base + offset

tmpByte ← Memory[address].Byte

GPR[rd] ← {24{0}, tmpByte}

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

4.8. LDW

LDW pre-add load word, immediate offset, no write back

LDW rd, [rs1+], #imm14 ; imm14 ! = 0

LDW rd, [rs1] ; imm14 = = 0

31	23	18	13
operation 79H	rs1	rd	imm14

指令描述：

该指令将该 imm14 进行无符号扩展，然后与 rs1 寄存器中的数据相加，并将加的结果作为地址访问内存，取回来一个字的数据，最后将该数值放到 rd 寄存器中。

指令操作：

base ← GPR[rs1]

offset ← {18{0}, imm14}

MRRC CONFIDENTIAL

address ← base + offset

GPR[rd] ← Memory[address].Word

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

LDW pre-add load word, register offset, immediate logic shift to left, no write back

LDW rd, [rs1+], rs2 << #s_imm5 ; s_imm5 ! = 0

LDW rd, [rs1+], rs2 ; s_imm5 = = 0

31	23	18	13	8	4
operation 59H	rs1	rd	s_imm5	function 0H	rs2

指令描述：

该指令首先对 rs2 寄存器中的数据逻辑左移 s_imm5 位，然后将该数值与 rs1 寄存器中的数据相加，并将加的结果作为地址访问内存，取回来一个字的数据，最后将该数值放到 rd 寄存器中。

指令操作：

base ← GPR[rs1]
offset ← {GPR[rs2][31-s_imm5:0], s_imm5{0}}
address ← base + offset

GPR[rd] ← Memory[address].Word

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

4.9. STB

STB pre-sub store byte, immediate offset, no write back

STB rd, [rs1-], #imm14 ; imm14 ! = 0

STB rd, [rs1] ; imm14 = = 0

31	23	18	13
operation 74H	rs1	rd	imm14

指令描述：

该指令将该 imm14 进行无符号扩展，然后将 rs1 寄存器中的数据减去，并将减的结果作为地址访问内存，向该地址存入 rd 寄存器中低 8 位的数据。

指令操作：

base ← GPR[rs1]
offset ← {18{0}, imm14}
tmpOffset ← ~offset

MPRC CONFIDENTIAL

address ← base + tmpOffset + 1

Memory[address].Byte ← GPR[rd][7:0]

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

STB pre-add store byte, register offset, immediate logic shift to left, no write back

STB rd, [rs1+], rs2 << #s_imm5 ; s_imm5 ! = 0

STB rd, [rs1+], rs2 ; s_imm5 = = 0

31	23	18	13	8	4
operation 5CH	rs1	rd	s_imm5	function 0H	rs2

指令描述：

该指令首先对 rs2 寄存器的数据逻辑左移 s_imm5 位，然后将该数值与 rs1 寄存器中的数据相加，并将加的结果作为地址访问内存，向该地址存入 rd 寄存器中低 8 位的数据。

指令操作：

base ← GPR[rs1]

offset ← {GPR[rs2][31-s_imm5:0], s_imm5{0}}

address ← base + offset

Memory[address].Byte ← GPR[rd][7:0]

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

4.10. STW

STW pre-add store word, immediate offset, no write back

STW rd, [rs1+], #imm14 ; imm14 ! = 0

STW rd, [rs1] ; imm14 = = 0

31	23	18	13
operation 78H	rs1	rd	imm14

指令描述：

该指令将该 imm14 进行无符号扩展，然后与 rs1 寄存器中的数据相加，并将加的结果作为地址访问内存，将 rd 寄存器中的数据存入内存中。

指令操作：

base ← GPR[rs1]

offset ← {18{0}, imm14}

address ← base + offset

Memory[address].Word ← GPR[rd]

MPRS CONFIDENTIAL

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

STW pre-add store word, register offset, immediate logic shift to left, no write back

STW rd, [rs1+], rs2 << #s_imm5 ; s_imm5 ! = 0

STW rd, [rs1+], rs2 ; s_imm5 = = 0

31	23	18	13	8	4
operation 58H	rs1	rd	s_imm5	function 0H	rs2

指令描述：

该指令首先对 rs2 寄存器的数据逻辑左移 s_imm5 位，然后将该数值与 rs1 寄存器中的数据相加，并将加的结果作为地址访问内存，将 rd 寄存器中的数据存入内存中。

指令操作：

base ← GPR[rs1]

offset ← {GPR[rs2][31-s_imm5:0], s_imm5{0}}

address ← base + offset

Memory[address].Word ← GPR[rd]

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

4.11. CMPSUB.A

CMPSUB.A register compare subtraction, immediate logic shift to left

CMPSUB.A rd, rs1, rs2 << #s_imm5 ; s_imm5 != 0

CMPSUB.A rd, rs1, rs2 ; s_imm5 == 0

31	23	18	13	8	4
operation 15H	rs1	rd 0H	s_imm5	function 0H	rs2

指令描述：

该指令首先对 rs2 寄存器中的数据逻辑左移 s_imm5 位，然后将 rs1 寄存器中的数据减去移位后的结果，并根据计算过程修改条件码，但是并不把结果存入 rd 寄存器。

MPRC CONFIDENTIAL

指令操作：

operatorA ← GPR[rs1]

operatorB ← {GPR[rs2][31-s_imm5:0], s_imm5{0}}

tmp ← operatorA + (~operatorB) + 1

标志位影响：

V 位为 1 有两种情况：（operatorA [31] == 1 && operatorB[31] == 0 && tmp[31] == 0）或者（operatorA [31] == 0 && operatorB[31] == 1 && tmp[31] == 1），其他情况 V 位为 0。

如果加法中产生进位则 C 位为 1，否则为 0。

当结果为全 0 时，Z 位为 1，否则为 0。

N 位被设置成结果的最高位的值。

相关例外：

无

CMPSUB.A immediate compare subtraction, immediate arithmetic rotate to right

CMPSUB.A rd, rs1, imm9 ; s_imm5 == 0

CMPSUB.A rd, rs1, imm9 <> s_imm5 ; s_imm5 != 0

31	23	18	13	8
operation 35H	rs1	rd 0H	s_imm5	imm9

指令描述：

该指令将 rs1 寄存器中内容减去 IMM 立即数，并根据计算过程修改条件码，但是并不把结果存入 rd 寄存器。需要说明的是，IMM 立即数是由 imm9 循环右移 s_imm5 位得到。

指令操作：

tmpA ← {23'b0, imm9}

operatorB ← GPR[rs1]

IF (s_imm5 == 0)

operatorA ← tmpA

ELSE

operatorA ← {tmpA[s_imm5-1:0], tmpA[31: s_imm5]}

ENDIF

tmp ← (~operatorA) + operatorB + 1

标志位影响：

如果加法中产生进位则 C 位为 1，否则为 0。

V 位为 1 有两种情况：（operatorA[31] == 1 && operatorB[31] == 1 && tmp[31] == 0）或者（operatorA[31] == 0 && operatorB[31] == 0 && tmp[31] == 1），其他情况 V 位为 0。

当结果为全 0 时，Z 位为 1，否则为 0。

MPRC CONFIDENTIAL

N 位被设置成结果的最高位的值。

相关例外：

无

4.12. B

B branch always, no link

B #imm24

31	23
operation BCH	imm24

指令描述：

该指令将 imm24 符号扩展成 32 位，然后将其与 PC + 4 相加，将结果写入 r31 寄存器。

指令操作：

offset ← {8{imm24[23]}, imm24}

PC ← PC + 4 + offset

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

4.13. Bcc

Bcc conditional branch, no link

Bcc #imm24

31	28	24	23
Operation 5H	bc	0H	imm24

指令描述：

如果相应的条件成立，则该指令将 imm24 符号扩展成 32 位，然后将其与 PC + 4 相加，将结果写入 r31 寄存器。

指令操作：

IF (fc)

offset ← {8{imm24[23]}, imm24}

MPRC CONFIDENTIAL

PC ← PC + 4 + offset

ENDIF

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

4.14. B.L

B.L conditional branch, link

B.L #imm24

31	28	24	23
Operation 5H	bc	1H	imm24

指令描述：

如果相应的条件成立，则该指令将 imm24 符号扩展成 32 位，然后将其与 PC + 4 相加，将结果写入 r31 寄存器，同时将 PC 写入 LR 寄存器中。

指令操作：

```
IF (fc)
  offset ← {8{imm24[23]}, imm24}
  GPR[r30] ← PC
  GPR[r31] ← PC + 4 + offset
ENDIF
```

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL

B.L branch always, link

B.L #imm24

31	23
operation BDH	imm24

指令描述：

该指令将 imm24 符号扩展成 32 位，然后将其与 PC + 4 相加，将结果写入 r31 寄存器，同时将 PC 写入 LR 寄存器中。

指令操作：

offset ← {8{imm24[23]}, imm24}

GPR[r30] ← PC

GPR[r31] ← PC + 4 + offset

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

4.15. JUMP

JUMP jump, no link

JUMP rs2

31	23	18	13	8	4
operation 10H	rs1 1FH	rd 1FH	s_imm5 0H	function 9H	rs2

指令描述：

该指令把 rs2 寄存器中的数据写入 PC 寄存器中。其中 rs2 不可以是 PC 寄存器。

指令操作：

PC ← GPR[rs2]

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

CONFIDENTIAL

JUMP.L jump, link

JUMP.L rs2

31	23	18	13	8	4
operation 11H	rs1 1FH	rd 1FH	s_imm5 0H	function 9H	rs2

指令描述：

该指令把 rs2 寄存器中的数据写入 PC 寄存器中，并把当前 PC 寄存器的值（当前指令的地址加 4）保存到链接寄存器 LR（R30）中，且 LR 寄存器最低两位被清零。其中 rs2 不可以是 PC 寄存器。

指令操作：

LR[31:2] ← PC[31:2]

LR[1:0] ← 0

PC ← GPR[rs2]

标志位影响：

不影响 N、Z、C、V 标志位。

相关例外：

无

MPRC CONFIDENTIAL