

RISC8B 内核单片机指令集与汇编工具

版本：2B

1. 概述

WCH-RISC8B 是 8 位数据宽度的精简指令集单片机内核，RISC8B 基于 RISC8A 新增了一些位传送指令。所有指令宽度均为 16 位，指令由操作码和操作数组成。RISC8B 共有 66 条指令，根据操作对象分为：控制类，面向字节操作类，常数操作类，面向位操作类，转移类。

2. 指令集

f 代表 SFR 或 RAM 寄存器，有效数值是 0x00-0xFF；

F 代表扩展地址 SFR 或 RAM 寄存器，有效数值是 0x000-0x1FF；

A 为工作寄存器，C 为进位标志，Z 为零标志；

d 代表目的寄存器，有效数值是 0、1、A、F、空（指没有该操作数），

当 d=0 或 d=A 时，操作结果放入 A 寄存器，当 d=1 或 d=F 或 d 为空时，操作结果放入 f 寄存器；

b 代表位选择，有效数值是 0-7；

a 代表独立位选择，有效数值是 0-3，0 对应 C，1-3 对应自定义的独立位；

k2 代表一个 2 位的常数，有效数值是 0-3；

k 或 k8 代表一个 8 位的常数，有效数值是 0x00-0xFF；

K7 代表一个 7 位的常数，有效数值是 0x00-0x7F；

k9 代表一个 9 位的常数，有效数值是 0x000-0x1FF；

k10 代表一个 10 位的常数，有效数值是 0x000-0x3FF；

k12 代表一个 12 位的常数，有效数值是 0x000-0xFFF；

TOS 代表 Top Of Stack，指当前堆栈单元；

1#ff、2#ff，分别指自定义的 1#、2#快速 SFR 寄存器；

1#bf、2#bf，分别指自定义的 1#、2#位操作 SFR 寄存器；

{*,*} 表示集合，f[b] 表示 f 的 b 位，[@a] 表示地址 a 指向的寄存器，(k) 表示执行时带参数 k。

底纹标灰的指令为 RISC8B 内核新增或重新定义，RISC8A 内核可能不支持这些指令。

二进制指令码	HEX	指令说明	助记符	操作数	执行操作	影响状态
控制类（18 条）						
00000000 000000xx	0000	空操作	NOP		无	无
00000000 000010xx	0008	清除看门狗计时器	CLRWDT		0→WatchDogTimer	无
00000000 00001100	000C	睡眠	SLEEP		clock stop	无
00000000 000011kk	000C	进入指定睡眠模式	SLEEPX	k2	k2→SleepMode, clock stop (k2)	无
00000000 00010bbb	0010	等待 b 选择的位为 1	WAITB	b	wait bit[b]==1	无
00000000 00010000	0010	等待从并口读出	WAITRD		wait SB_IF_READ==1	无
00000000 00010100	0014	等待从并口写入 或者等待 SPI 中断	WAITWR WAITSPI		wait SB_IF_WRITE==1 wait SB_IF_SPI==1	无

00000000 00011000	0018	从程序空间读取代码	RDCODE		ROM_CODE → {SFR, A}	无
00000000 000110kk	0018	从程序空间读取代码	RCODE	k2	ROM_CODE (k2) → {SFR, A}	无
00000000 00011100	001C	将代码写入程序空间	WRCODE		{SFR} → ROM_CODE	短时暂停
00000000 000111kk	001C	带参数的自定义操作	EXEC	k2	custom operation (k2)	自定义
00000000 001000xx	0020	保存状态到堆栈	PUSHAS		{A, Z, C, ...} → TOS	无
00000000 001001xx	0024	从堆栈恢复状态	POPAS		TOS → {A, Z, C, ...}	Z, C
00000000 001010xx	0028	保存间接寻址寄存器 2 和页面位到堆栈	PUSHA2		{SFR_INDIR_ADDR2, INDIR_RAM_PAGE} → TOS	无
00000000 001011xx	002C	从堆栈恢复间接寻址寄存器 2 和页面位	POPA2		TOS → {INDIR_RAM_PAGE, SFR_INDIR_ADDR2}	INDIR_RAM_PAGE_BIT
00000000 001100xx	0030	子程序返回	RET		TOS → PC	无
00000000 001101xx	0034	子程序置零状态返回	RETZ		TOS → PC, 1 → Z	Z
00000000 001110xx	0038	中断返回	RETIE		TOS → PC, 1 → IE_global	无
面向字节操作类 (16 条)						
00000000 000001xx	0004	A 清零	CLRA		0x00 → A, 1 → Z	Z
00000001 ffffffff	01ff	f 清零	CLR	f	0x00 → f, 1 → Z	Z
0001000F FFFFFFFF	10FF	A 传送到 F, 位 9 指定扩展地址的页面位	MOVA	F	A → F	无
000d001F FFFFFFFF	d2FF	传送 F 到 d, 位 9 指定扩展地址的页面位	MOV	F, d	F → d	Z
000d0100 ffffffff	d4ff	f 递增	INC	f, d	f+1 → d	Z
000d0101 ffffffff	d5ff	f 递减	DEC	f, d	f-1 → d	Z
000d0110 ffffffff	d6ff	f 递增, 为零则跳	INCSZ	f, d	f+1 → d, skip if Z==1	无
000d0111 ffffffff	d7ff	f 递减, 为零则跳	DECSZ	f, d	f-1 → d, skip if Z==1	无
000d1000 ffffffff	d8ff	f 高低半字节交换	SWAP	f, d	f[0:3] <> f[4:7] → d	无
000d1001 ffffffff	d9ff	A 和 f 做与运算	AND	f, d	A&f → d	Z
000d1010 ffffffff	dAff	A 和 f 做或运算	IOR	f, d	A f → d	Z
000d1011 ffffffff	dBff	A 和 f 做异或运算	XOR	f, d	A^f → d	Z
000d1100 ffffffff	dCff	A 加 f	ADD	f, d	A+f → d	Z, C
000d1101 ffffffff	dDff	f 减去 A	SUB	f, d	f-A → d	Z, C
000d1110 ffffffff	dEff	f 带 C 循环左移	RCL	f, d	{f, C} <<1 → d, f[7] → C	C
000d1111 ffffffff	dFff	f 带 C 循环右移	RCR	f, d	{C, f} >>1 → d, f[0] → C	C
常数类 (16 条)						

00100000	kkkkkkkk	20kk	子程序带参数返回	RETL	k	k→A, TOS→PC	无
00100001	kkkkkkkk	21kk	子程序带参数且置非零状态返回	RETLN	k	k→A, 0→Z, TOS→PC	Z
0010001k	kkkkkkkk	22kk	常数置入间接地址寄存器 1 和专用页面位	MOVIP	k9	k9→{INDIR_RAM_PAGE, SFR_INDIR_ADDR}	INDIR_RAM_PAGE_BIT
001001kk	kkkkkkkk	24kk	常数置入间接寻址寄存器 2	MOVIA	k10	k10→SFR_INDIR_ADDR2	无
00100011	kkkkkkkk	23kk	常数置入 1#快速寄存器	MOVA1F	k	k→1#ff	无
00100101	kkkkkkkk	25kk	常数置入 2#快速寄存器	MOVA2F	k	k→2#ff	无
00100110	kkkkkkkk	26kk	常数置入间接地址寄存器 2 指向的寄存器	MOVA2P	k	k→[@SFR_INDIR_ADDR2]	无
00100111	kkkkkkkk	27kk	常数置入间接地址寄存器 1 指向的寄存器	MOVA1P	k	k→[@SFR_INDIR_ADDR]	无
00101000	kkkkkkkk	28kk	常数置入 A	MOVL	k	k→A	无
00101001	kkkkkkkk	29kk	常数和 A 做与运算	ANDL	k	k&A→A	Z
00101010	kkkkkkkk	2Akk	常数和 A 做或运算	IORL	k	k A→A	Z
00101011	kkkkkkkk	2Bkk	常数和 A 做异或运算	XORL	k	k^A→A	Z
00101100	kkkkkkkk	2Ckk	常数加 A	ADDL	k	k+A→A	Z, C
00101101	kkkkkkkk	2Dkk	常数减去 A	SUBL	k	k-A→A	Z, C
00101110	kkkkkkkk	2Ekk	常数加 A 做比较	CMPLN	k	k+A	Z, C
00101111	kkkkkkkk	2Fkk	常数减去 A 做比较	CMPL	k	k-A	Z, C
面向位操作类 (9 条)							
01000bbb	ffffffff	40ff	清除 f 的位 b	BC	f, b	0→f[b]	无
01001bbb	ffffffff	48ff	设置 f 的位 b	BS	f, b	1→f[b]	无
01010bbb	ffffffff	50ff	f 的位 b 为 0 则跳	BTSC	f, b	skip if f[b]==0	无
01011bbb	ffffffff	58ff	f 的位 b 为 1 则跳	BTSS	f, b	skip if f[b]==1	无
00000000	000111aa	001C	传送 a 选择的位到 C	BCTC	a	bit[a]→C	C
00000000	100aabbb	008b	传送 1#寄存器的位 b 到 a 选择的位	BP1F	a, b	1#bf[b]→bit[a]	C if a==0
00000000	101aabbb	00Ab	传送 2#寄存器的位 b 到 a 选择的位	BP2F	a, b	2#bf[b]→bit[a]	C if a==0
00000000	110aabbb	00Cb	传送 a 选择的位到 1#寄存器的位 b	BG1F	a, b	bit[a]→1#bf[b]	无
00000000	111aabbb	00Eb	传送 a 选择的位到 2#寄存器的位 b	BG2F	a, b	bit[a]→2#bf[b]	无

转移类 (7 条)						
0110kkkk kkkkkkkk	6kkk	跳转	JMP	k12	k12→PC	无
0111kkkk kkkkkkkk	7kkk	调用子程序	CALL	k12	PC+1→TOS, k12→PC	无
001100kk kkkkkkkk	3kkk	Z=0 则跳转	JNZ	k10	k10→PC[9:0] if Z==0	无
001101kk kkkkkkkk	3kkk	Z=1 则跳转	JZ	k10	k10→PC[9:0] if Z==1	无
001110kk kkkkkkkk	3kkk	C=0 则跳转	JNC	k10	k10→PC[9:0] if C==0	无
001111kk kkkkkkkk	3kkk	C=1 则跳转	JC	k10	k10→PC[9:0] if C==1	无
1Kkkkkkk kkkkkkkk	8Kkk	常数与 A 做比较, 相等则跳转	CMPZ	K7, k	k→PC[7:0] if A==K7	无

3. 指令周期

RISC8B 的指令长度都是一个字节，但是指令周期分为四种：单周期、双周期、多周期、以及特殊多周期。其中，多周期的周期数要根据程序 ROM 的工艺而定，通常情况下多周期就是两个周期（同双周期），对于个别采用特殊工艺的芯片（目前仅 CH537X 芯片）则有可能超过两个周期。

指令周期由系统工作时钟频率 SCLK 决定，计算公式是：指令周期的时间 = 1 / SCLK。

特殊多周期适用于个别特殊指令，例如：RDCODE 通常为 3 个周期；WRCODE 最少为 4 个周期，最多执行时间为数百毫秒，EXEC 为自定义。除此之外，所有修改程序计数器 PC 以及 SFR_PRG_COUNT 的指令和跳转指令都是双周期或者多周期，除此之外的都是单周期。双周期和多周期指令列表如下：

指令助记符	指令说明	成为双周期或者多周期的附加条件
JMP、CALL	直接跳转	总是多周期
RET、RETZ、RETIE、RETL、RETLN	跳转返回	总是多周期
JNZ、JZ、JNC、JC	条件跳转	当跳转条件成立发生跳转时为多周期，否则单周期
CMPZ	条件跳转	当跳转条件成立发生跳转时为多周期，否则单周期
BTSC、BTSS	条件跳转	当跳转条件成立发生跳转时为双周期，否则单周期
所有面向字节操作类的指令中，其目标寄存器为 SFR_PRG_COUNT 的指令，例如：ADD SFR_PRG_COUNT	修改程序计数器 PC	最终是修改 SFR_PRG_COUNT 寄存器的指令为多周期，不修改或者不保存的指令为单周期，例如“ADD SFR_PRG_COUNT, A”是读操作，是单周期

4. 等效和重新定义指令

4.1. 等效指令助记符

RISC8B 的汇编工具 WASM53B 支持以下等效指令助记符（别名）：

指令说明	原指令助记符	等效指令助记符
清除看门狗计时器	CLRWDT	WDT
睡眠	SLEEP	HALT
等待 SPI 中断或者等待 SPI 写入或者读出	WAITSPI	WAITWR
保存状态到堆栈	PUSHAS	PUSH
从堆栈恢复状态	POPAS	POP
子程序返回	RET	RETURN
子程序置零状态返回（成功返回）	RETZ	RETOK

中断返回	RETIE	RETI
子程序带参数返回（定义单字节数据）	RETL	DB
子程序带参数且置非零状态返回（带错误码返回）	RETLN	RETER
跳转	JMP	GOTO
面向字节操作类的指令，除了 CLRA, INCSZ, DECSZ 面向位操作类的指令，除了 BTSC, BTSS	原助记符 （例如：INC、BC）	在原助记符后加 F （例如：INCF、BCF）
f 递增, 为零则跳	INCSZ	INCFSZ
f 递减, 为零则跳	DECSZ	DECFSZ
f 带 C 循环左移	RCL	RCLF 或者 RLF
f 带 C 循环右移	RCR	RCRF 或者 RRF
f 的位 b 为 0 则跳	BTSC	BTFSC
f 的位 b 为 1 则跳	BTSS	BTFSS
定义双字节数据（定义新指令，操作数为新指令码）		DW

4.2. 重新定义指令码

在不同芯片或不同应用中，RISC8B 的部分指令码被重新定义/复用，以下指令码表示不同指令：

二进制指令码范围	基本指令	复用指令
00000000 00001100 → 00000000 000011kk	SLEEP	SLEEPX k2
00000000 00010000 → 00000000 00010bbb	WAITRD	WAITB b
00000000 00010100 → 00000000 00010bbb	WAITWR WAITSPI	WAITB b
00000000 00011000 → 00000000 000110kk	RDCODE	RCODE k2
00000000 00011100 → 00000000 000111kk	WRCODE	EXEC k2
00000000 00011100 → 00000000 000111aa	WRCODE	BCTC a
0010001k kkkkkkkk → 00100010 kkkkkkkk	MOVIP k9	MOVIP k8
0010001k kkkkkkkk → 00100011 kkkkkkkk	MOVIP k9	MOVA1F k
001001kk kkkkkkkk → 00100100 kkkkkkkk	MOVIA k10	MOVIA k8
001001kk kkkkkkkk → 00100101 kkkkkkkk	MOVIA k10	MOVA2F k
001001kk kkkkkkkk → 00100110 kkkkkkkk	MOVIA k10	MOVA2P k
001001kk kkkkkkkk → 00100111 kkkkkkkk	MOVIA k10	MOVA1P k

5. 寻址方式

RISC8B 的寻址方式包括：立即数寻址、立即数快速寻址、普通直接寻址、扩展直接寻址、间接寻址、位寻址，后 4 种用于寻址寄存器。

5.1. 立即数寻址

立即数寻址用于常数类指令，指令码中的操作数为立即数，直接用于操作。

例： MOVIA 0x246 ;SFR_INDIR_ADDR2=0x246, 将数据 0x246 写入间接寻址 2 的地址寄存器
MOVIP 0x135 ;SB_INDIR_RAM_PAGE=1, SFR_INDIR_ADDR=0x35
ADDL 0x23 ;A=A+0x23

5.2. 扩展直接寻址

扩展直接寻址的寻址范围是 0x000-0x1FF，由指令码直接提供 9 位寄存器地址，仅适用于以下两条直

接读写寄存器的指令：MOV register, A 或 F 指令，MOVA register 指令。需要寻址更大范围时应该采用间接寻址方式。

例： MOVA 0x167 ; [0x167]=A, 将 A 中数据写入地址为 0x167 的寄存器, 建议用标号代替
MOV 0x289, A ; A=[0x289], 读取地址为 0x289 的寄存器中的数据到 A 中, 建议用标号代替

5.3. 普通直接寻址

普通直接寻址的寻址范围是 0x000-0x0FF，由指令码直接提供 8 位寄存器地址，适用于除上述扩展直接寻址指令之外的所有直接地址指令。例如 CLR register、ADD register、BS register, bit 等。需要寻址更大范围时应该采用间接寻址方式。

例： CLR 0x67 ; [0x67]=0, 将地址为 0x67 的寄存器中的数据清 0, 建议用标号代替
INC 0x89 ; [0x89]=[0x89]+1, 将地址为 0x89 的寄存器中的数据加 1, 建议用标号代替
BS 0x9B, 3 ; [0x9B]=[0x9B] | 0x08, 将地址为 0x9B 的寄存器的位 3 置 1, 建议用标号代替

5.4. 间接寻址

间接寻址的寻址范围是 0x000-0x3FF，覆盖 RISC8B 的所有寄存器。RISC8B 共提供了两组间接寻址寄存器，每组间接寻址寄存器由一个地址寄存器和一个数据读写端口组成。先向地址寄存器写入目的寄存器的地址，再通过读写数据读写端口就可以读写目的寄存器。

读写间接寻址的数据读写端口 SFR_INDIRE_PORT，就是读写由间接寻址的地址寄存器 SFR_INDIRE_ADDR 及其页面位 SB_INDIRE_RAM_PAGE 指定地址的目的寄存器，地址范围是 0x000-0x1FF，其中 SFR_INDIRE_ADDR 提供低 8 位地址。读写间接寻址的数据读写端口 SFR_INDIRE_PORT2，就是读写由间接寻址的地址寄存器 SFR_INDIRE_ADDR2 指定地址的目的寄存器，地址范围是 0x000-0x3FF，全部地址由 SFR_INDIRE_ADDR2 提供。

例： MOVL 0x45 ; A=0x45, 这是立即数寻址
MOVIA 0x357 ; SFR_INDIRE_ADDR2=0x357, 间接寻址 0x357
MOVA SFR_INDIRE_PORT2 ; [SFR_INDIRE_ADDR2]=0x45, 将数据 0x45 写入地址为 0x357 的寄存器
MOVIP 0x123 ; SB_INDIRE_RAM_PAGE=1, SFR_INDIRE_ADDR=0x23, 间接寻址 0x123
ADD SFR_INDIRE_PORT ; [SB_INDIRE_RAM_PAGE, SFR_INDIRE_ADDR] += 0x45

5.5. 立即数快速寻址

立即数快速寻址用于常数类指令，指令码中的操作数为立即数，无需通过 A 中转而直接快速写入目标寄存器。MOVA1F 和 MOVA2F 适用于自定义的 2 个快速寄存器，相当于立即数寻址再普通直接寻址；MOVA1P 和 MOVA2P 通过间接寻址适用于所有寄存器，相当于立即数寻址再间接寻址。

例： MOVA1F 0x46 ; 将数据 0x46 直接写入自定义的 1#快速寄存器
MOVA2F 0x35 ; 将数据 0x35 直接写入自定义的 2#快速寄存器
MOVA1P 0x78 ; 将数据 0x78 直接写入 SFR_INDIRE_ADDR 间接寻址指向的寄存器
MOVA2P 0x92 ; 将数据 0x92 直接写入 SFR_INDIRE_ADDR2 间接寻址指向的寄存器

5.6. 位寻址

位寻址由指令码直接提供 3 位位地址，而对寄存器的寻址可分别采用普通直接寻址或者间接寻址，分别实现 0x000-0x0FF 或者 0x000-0x3FF 的寻址范围，从而可以寻址任何一个寄存器的任何一位。

位传送指令用于在选中的自定义独立位与自定义位寄存器的选中位之间实现单周期位数据复制。

例： BC 0x67, 4 ; 0x67. 4=0, 将地址为 0x67 的寄存器的位 4 置为 0, 建议用标号代替
MOVIA 0x135 ; SFR_INDIRE_ADDR2=0x135, 间接寻址 0x135
BS SFR_INDIRE_PORT2, 3 ; [SFR_INDIRE_ADDR2]. 3=1, 将地址为 0x135 的寄存器的位 3 置为 1
BCTC 2 ; 将自定义的 2#独立位复制到 C 中
BP2F 0, 5 ; 将自定义的 2#位寄存器的位 5 复制到 0#独立位即 C 中
BG1F 3, 6 ; 将自定义的 3#独立位复制到自定义的 1#寄存器的位 6 中

6. 汇编程序

6.1. 汇编工具

WASM53B.EXE 是 RISC8B 单片机的汇编工具，用于将汇编源程序编译为指令码目标数据。WASM53B 可以运行在 DOS 或者 Windows 的 DOS 下，命令行语法是：

```
WASM53B 源程序文件名
```

在 Windows 下，也可以将源程序文件直接拖向 WASM53B 工具的图标实现编译，如果源程序文件中使用伪指令 INCLUDE 包含其它文件，那么应该为被包含文件指明含有盘符在内的完整路径。

源程序文件的扩展名默认是 ASM，如果源程序文件的扩展名不是 ASM 则需要加上小数点和扩展名，如果没有扩展名则需要加上小数点。

WASM53B 运行后对源程序文件进行两遍扫描，第一遍扫描记录所有标号和常量，第二遍扫描将源程序中的指令助记符编译为相应的指令码，并产生与源程序文件同名的列表文件和数据文件，同时将错误信息和警告信息等显示在屏幕上。

列表文件的扩展名是 LST，列表文件中包含全部源程序、对应的指令码目标数据、标号、常量、以及各种错误和警告信息，并且最末行显示所有错误和警告信息的数量，用于检查错误和核对数据。

数据文件的扩展名是 BIN，数据文件中包含指令码目标数据，每条指令码占用两个字节，低地址字节为指令码的低 8 位，高地址字节为指令码的高 8 位。由于 RISC8B 的程序空间可以容纳 4096 条指令，所以数据文件的长度总是小于或者等于 8KB，没有用到的程序空间不会被汇编工具填充任何数据。数据文件中的数据可以直接置入 RISC8B 的程序 ROM 中。

例： WASM53B SAMPLE.ASM

产生： 列表文件 SAMPLE.LST 和 数据文件 SAMPLE.BIN

6.2. 字符集

标号是程序空间的实际地址的符号表示，通常情况下，标号是汇编工具在编译过程中自动计算出来的，而不是事先定义的数值。例如，将要跳转的地址用标号表示作为 JMP 指令的操作数，将被调用子程序的起始地址用标号表示作为 CALL 指令的操作数。

常量是各种数值的符号表示，通常情况下，常量是通过汇编伪指令 EQU 事先定义的数值，包括寄存器地址、寄存器位地址、常数或者立即数等。

RISC8B 的汇编工具在内部不区分标号和常量，而是以完全相同的方式进行处理，所以后续说明中不再区分标号和常量，关于标号的所有说明都同样适用于常量。

RISC8B 的汇编工具不区分字符的大小写。标号名称的有效字符包括：数字(0-9)、字母(A-Z、a-z)、下划线(_)、美元符(\$)、井字符(#)、地址符(@)，其中数字(0-9)不能作为标号名称的第一个字符，标号名称的长度不能超过 20，也就是不能超过 20 个字符。

以下单词是保留字，所以不能作为标号名称：

```
END、ORG、INCLUDE
```

以下单词是保留字，可以作为标号名称，但是不易阅读理解，所以不建议：

```
EQU、所有的指令助记符、以及相应的等效指令助记符
```

6.3. 数值

RISC8B 的汇编工具支持二进制数、十六进制数、十进制数、字符。有效数值范围是从 -32768 到 65535。

二进制数有两种表示方式：前缀(0B)，或者前缀(B')且后缀(')，有效数值字符是 0 和 1；

十六进制数有两种表示方式：前缀(0X)，或者前缀(H')且后缀(')，有效数值字符是 0-9 和 A-F；

十进制数有三种表示方式：直接表示，或者前缀(0D)，或者前缀(D')且后缀(')，有效数值字符是 0-9；

字符只有一种表示方式：前缀(')且后缀(')，有效数值字符是所有的 ASCII 码。

例：0b01101001、B'01101001' 都表示 0x69 或 105

0x5A、H'5A' 都表示 0x5A 或 90
 0d43、D'43'、43 都表示 0x2B 或 43
 'a' 表示 0x61 或 97, '6' 表示 0x36 或 54

6.4. 表达式

RISC8B 的汇编工具支持从右向左自然优先级的表达式。

运算符包括：相加(+)、相减(-)、相乘(*)、相除(/)、取余(%)、位与(&)、位或(|)、位异或(^)、位反(~)、左移(<<)、右移(>>)。

位反(~)只需要右边的一个操作数，除此之外的每个运算符都需要两个操作数，分别在运算符的左边和右边，操作数可以是数值、先前已经定义的标号或者嵌套表达式。对于相减(-)操作，如果没有左边的操作数则左边的操作数默认为 0，相当于负号。

位反(~)、左移(<<)、右移(>>)，这 3 种运算的结果自动限制在 0 到 255 数值范围，即最大 0xFF。

嵌套表达式是指表达式中存在多个运算符，这种情况下，右边的运算符具有较高的优先级。汇编工具首先执行右边的运算符，对其左操作数和右操作数进行运算，产生的结果再作为更左边的运算符的右操作数，然后执行该更左边的运算符，对其左操作数和右边的结果进行运算。

例：表达式 { 0x52 | H'0A' } 的结果是 0x52H|0x0A=0x5A

表达式 { 2 * 7 - 4 } 的结果是 2*(7-4)=6

表达式 { - 0b11010110 & 'C' ^ 0xFF } 的结果是 0x0000-(0xD6&(0x43^0xFF))=0xFF6C

NEXT_COUNT EQU THIS_COUNT + 1 ;标号 NEXT_COUNT=THIS_COUNT+1

ADDL 0xFF & 0 - LEN ;A 加上 (0xFF&(0-LEN))，相当于 A 减去 LEN

6.5. 语句格式

RISC8B 的汇编源程序的语句以行为单位，格式如下：

```
LABEL INSTRUCTION PARAMETER1, PARAMETER2 ; REMARK
```

其中：

LABEL 为标号。标号必须从一行的第一个字符开始，标号后面可以加冒号(:)，但是冒号只是作为分隔符处理而不起任何作用。汇编工具将所有从一行的第一个字符开始的除保留字之外的单词作为标号处理。指令助记符前面的标号是可选的，EQU 伪指令前面必须有标号(常量)，其它伪指令前面不能有标号。

INSTRUCTION 为指令助记符或者伪指令。为了区分于标号，指令助记符或者 EQU 伪指令不能从一行的第一个字符开始，但是其它伪指令可以从一行中的任何位置开始。

PARAMETER1 和 PARAMETER2 是指令或者伪指令的第一操作数和第二操作数。操作数可以是数值、标号或者表达式。根据指令或者伪指令的不同，有可能只有一个操作数或者没有操作数，如果有两个操作数，那么两者之间必须用逗号或者空格隔开。

REMARK 是注释。注释可以在一行中的任何位置以分号(;)开始，汇编工具将一行中所有在分号后面的字符作为注释而忽略处理。注释可以是英文字符或者中文字符。

LABEL 与 INSTRUCTION 之间，INSTRUCTION 与 PARAMETER1 之间，PARAMETER1 与 PARAMETER2 之间必须用至少一个空格或者其它等效字符隔开，等效字符包括：制表符(TAB)，冒号(:)，逗号(,)，其中冒号通常只有可能用于 LABEL 与 INSTRUCTION 之间，逗号通常只有可能用于 PARAMETER1 与 PARAMETER2 之间。

RISC8B 的汇编源程序中，每行语句的长度不能超过 249，也就是不能超过 249 个字符，源程序中可以有空行，每个源程序文件中不能超过 9999 行语句，多于 9999 行的源程序可以分成多个包含文件。

```
例： PORTA EQU 0x05 ;port A
      PA2 EQU 2
      ORG 0x80
START: CLR 0x0B ;建议用标号代替寄存器地址 0x0B
      MOVL B'11111011'
```

```
        MOVA  0x15          ;建议用标号代替寄存器地址 0x15
WAIT:   BTSC  PORTA, PA2    ;skip if PA2=0
        JMP   WAIT
        BC   3, 0          ;status register, C=0, 建议用标号代替寄存器地址 3
        MOV  PORTA, A      ;从 A 端口读取数据到 A 中
        PUSHAS              ;save A and status register to stack
```

6.6. 伪指令

6.6.1. 标号赋值伪指令 EQU

EQU 用于为标号赋值。在 EQU 左边是被赋值的标号名称，在 EQU 右边是数值、先前已经定义的标号或者表达式。

对于面向字节操作类和面向位操作类的指令，如果有操作数，通常其第一操作数可能是寄存器地址，建议对该寄存器地址使用标号，避免直接使用立即数作为寄存器地址。

汇编工具在编译源程序时，将自动用标号的值代替源程序中作为操作数的相应标号，通过 EQU 伪指令用标号代替数值，可以方便程序修改，并且容易阅读理解。

例： MY_COUNT EQU 0x23 ;标号 MY_COUNT=0x23

6.6.2. 地址定义伪指令 ORG

ORG 用于定义后续指令在程序空间中的起始地址。在 ORG 右边是地址，地址可以是数值、先前已经定义的标号或者表达式。

没有 ORG 伪指令时，整个源程序文件中第一条指令的默认起始地址为 0x0000。当 ORG 伪指令向前定义地址时，汇编工具将提供“go back by ORG”的警告信息，当 ORG 伪指令向后定义地址时，所跳过的程序空间将被汇编工具自动填充空操作指令 NOP。

例： ORG 0x0004 ;下一条指令从程序空间的 0x0004 地址开始

6.6.3. 文件包含伪指令 INCLUDE

INCLUDE 用于直接引用其它汇编源程序文件。在 INCLUDE 右边是被引用的文件名，如果被引用的文件在当前目录下，那么只需要提供文件名及其扩展名，否则需要提供完整的路径。如果在 Windows 下编译，那么被引用的文件应该提供含有盘符在内的完整的路径。

汇编工具处理到 INCLUDE 伪指令时，将打开被引用的源程序文件，将其作为当前源程序文件的一部分进行编译，处理完被引用的文件后再回到当前源程序文件的下一行继续编译，相当于将被引用的文件插入到当前文件中该 INCLUDE 伪指令所在的行。被引用的文件还可以再通过 INCLUDE 伪指令引用其它文件，汇编工具支持 INCLUDE 伪指令最多 8 级嵌套。通过 INCLUDE 伪指令可以将常用的标号（常量）或者子程序作为通用模块文件，直接被其它汇编源程序引用，而不必重复同样的定义或者编写。

例： INCLUDE C:\RISC8B\CH533\INC.ASM ;在此引用文件 C:\RISC8B\CH533\INC.ASM

6.6.4. 源程序结束伪指令 END

END 用于表示源程序到此结束。

汇编工具处理到 END 伪指令时，将结束整个源程序的编译过程，不再继续处理后续源程序。如果没有处理到 END 伪指令，那么汇编工具将一直处理完整个源程序文件，并提供“END not found”的警告信息。在被引用的文件中可以有 END 伪指令，但是汇编工具只会结束当前被引用文件的编译过程，而不会结束整个源程序的编译过程。

例： END ;在此结束编译过程，当前文件的后续部分将不会被处理

7. 指令详细说明

(暂无)

8. 常见问题

8.1. 每次修改程序后必须要做的人工检查：单字节查表程序

检查单字节查表程序过页情况，RISC8B 查表短跳转页面大小为 256，使用“ADD SFR_PRG_COUNT”及“RETL”指令查表时，只能访问当前 256 字节的地址页面，必须不能越过 0xFFFF 地址。

汇编工具软件 WASM53B 会对程序做智能分析，发现这类异常通常会做警告提醒。

建议使用 ORG 定义表的起始地址，避免超界。

8.2. 每次修改程序后必须要做的人工检查：双字节查表程序

使用“RDCODE”及“DW”指令设计双字节查表程序，要考虑地址输入方法，避免过页情况。

8.3. 每次修改程序后必须要做的人工检查：长表达式或者数学计算

汇编工具比较简单，表达式较长的或者关键的数学计算，建议人工检查 LST 文件中的目标代码。

8.4. 每次修改程序后必须要做的人工检查：程序翻页

检查程序空间翻页情况，RISC8B 程序长跳转页面大小为 4K，在两个程序页面之间切换时必须提前手工设置待跳转的页面。如果子程序在另一个页面，调用前需手工设置页面，调用返回后会自动恢复页面。

汇编工具软件 WASM53B 会对程序做智能分析，发现这类异常通常会做警告提醒。

8.5. 提供子程序库或者编程范例

可以提供以下编程范例：32 位数据的加减算法程序，单字节常数查表程序，双字节常数查表程序，延时子程序，串口和 SPI 操作程序，读写 24CXX 系列 EEPROM 和 25FX 系列 FLASH 的程序，USB 产品的基本框架程序，兼容 CH341 芯片的 USB 转串口程序。

8.6. 如果程序 ROM 为 OTP 工艺，那么建议预留空操作指令

OTP 工艺 ROM 原则上只能烧录一次，ROM 数据位空白时为 0，烧录后可能置 1。在调试过程中，为便于再次修改局部代码，可以临时选择不加读保护就允许再次烧录，为 0 的 ROM 数据位可以再烧录为 1，数据位 1 仍保持 1。程序代码中建议在关键模块前面预留 NOP 空操作指令，其指令码为 0，需要修改代码时将其替换为其它指令例如跳转指令后再次烧录即可。