Instruction Set H08A、H08C、H08D User's Manual

© 2008-2019 HYCON Technology Corp.

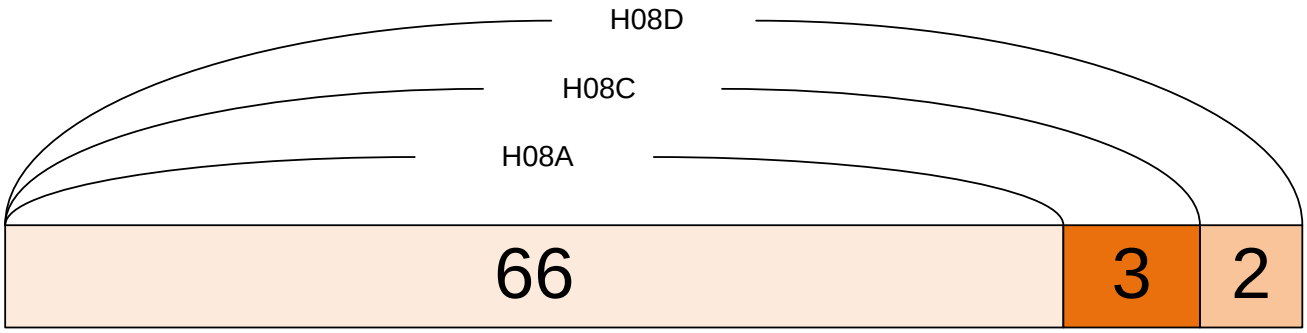
目录

© 2008-2019 HYCON Technology Corp

© 2008-2019 HYCON Technology Corp

1 简介

H08C、H08D 相较于 H08A 增加的指令部分，只要是提高 C 语言的编译效率。本文主要介绍指令集，其中包括指令快速索引表格和指令详解部分。为了使使用者尽快熟悉本文档的内容，需要对几点进行相应的了解。



H08A有基本66个指令集
H08C有69个指令，与原本66的指令增加的3个主要是C语言使用
H08D有71个指令，相较于H08C增加了2个指令，使C语言编译效率更加提高

© 2008-2019 HYCON Technology Corp

在本文档里‘w’表示工作寄存器，‘f’ 表示寄存器（可以包括用户自己定义的一般功能的寄存器或特殊功能的寄存器）；‘b’表示寄存器的第 b 个位；‘n’内存位置或者程序内存的位置，‘k’表示 8 位常数，‘d’ 表示资料存放地方; d = 0 表示存放在 W 工作寄存器 ;d = 1 表示存放在 f 寄存器，‘a’ 表示数据存放内存的位置,a=0 存放在目前内存位置；a=1 存放在 BSRCN 特性功能寄存器内所指定内存位置。另外的还包括：

特殊功能寄存器(SPR)	功能
工作寄存器 WREG	数据搬移，运算，判断
STATUS	C，OV，DC,等的判断，旗标会根据相关指令的执行结果改变
PSTATUS	芯片进入休眠或待机模式，看门狗计数器溢出等状态的判断，旗标会根据状态改变
程序计数器 PC	指向程序执行地址，包含 PCLATH 与 PCLATL
堆栈的迭顶寄存器 TOS	执行 CALL 指令或发生中断（INT）服务时存放程序计数器 PC 地址，子程序或中断返回时又会将地址返回给 PC
堆栈控制器 STKCN	包含堆栈满位、欠位、溢位旗标和堆栈指针 STKPRT
堆栈层寄存器 STKn	当被堆栈指针 STKPRT 指定时，会将寄存器中的数据传送到 TOS
FSR	特殊功能寄存器 FRS0(FSR0H/FSR0L)或者 FSR1(FSR1H/FSR1L)
PRODH	特殊功能寄存器，用于存放剩法运算结果的高字节
PRODL	特殊功能寄存器，用于存放剩法运算结果的低字节
关键词	含义
fs	寄存器
fd	寄存器
REG	寄存器
REG1	寄存器
MSB	最高位
LSB	最低位
Bit	表示位

在指令快速索引表里面每一条指令都可以通过 超链接 的方式可以使读者快速进入指令详解部分。本指令集里需注意的指令有：‘LBSR k’、‘LDPR k,f’、‘MVLP k’、‘TBLK *’DAW 及减法指令。为了避免使用错误，此些指令建议阅读者结合指令详解部分进行熟悉。

© 2008-2019 HYCON Technology Corp

2 指令索引

指令快速索引

Instruction		Description	Cycles	Status Affected
BYTE-ORIENTED FILE REGISTER OPERATIONS				
ADDC	f,d,a	将 W 与 F 和 C 做相加，并将结果放至 W 或 F。	1	C,DC,N,OV,Z
ADDF	f,d,a	将 W 与 F 做相加，并将结果放至 W 或 F。	1	C,DC,N,OV,Z
ADDL	k	将常数 k 与 W 做相加，并将结果放至 W。	1	C,DC,N,OV,Z
ANDF	f,d,a	将 W 与 F 做 AND 运算，并将结果放至 W 或 F。	1	N,Z
ANDL	k	将常数 k 与 W 做 AND 运算，并将结果放至 W。	1	N,Z
ARLC	f,d,a	将 F 内的值与 C 一起做左移动作，并将结果放至 W 或 F。	1	C,N,OV,Z
ARRC	f,d,a	将 F 内的值做右移动作,MSB 保留不变,LSB 移至 C	1	C,N,Z
CLRF	f,a	将 F 内的值都清为 0。	1	None
COMF	f,d,a	将 F 内的值取补码，并将结果放至 W 或 F。	1	N,Z
CPSE	f,a	若 F 与 W 的值相等，则跳过下一个指令。	1(2)(3)	None
CPSG	f,a	若 F 大于 W，则跳过下一个指令。	1(2)(3)	None
CPSL	f,a	若 F 小于 W，则跳过下一个指令。	1(2)(3)	None
DCF	f,d,a	将 F 内的值减 1，并将结果放至 W 或 F。	1	C,DC,N,OV,Z
DCSUZ	f,d,a	将 F 内的值减 1，若不为 0 则跳过下一个指令，并将结果放至 W 或 F。	1(2)(3)	None
DCSZ	f,d,a	将 F 内的值减 1，若为 0 则跳过下一个指令，并将结果放至 W 或 F。	1(2)(3)	None
INF	f,d,a	将 F 内的值加 1，并将结果放至 W 或 F。	1	C,DC,N,OV,Z
INSUZ	f,d,a	将 F 内的值加 1，若不为 0 则跳过下一个指令，并将结果放至 W 或 F。	1(2)(3)	None
INSZ	f,d,a	将 F 内的值加 1，若为 0 则跳过下一个指令，并将结果放至 W 或 F。	1(2)(3)	None
IORF	f,d,a	将 W 与 F 做 OR 运算，并将结果放至 W 或 F。	1	N,Z
IORL	k	将常数 k 与 W 做 OR 运算，并将结果放至 W。	1	N,Z
LBSR	k	将常数 k 搬到 BSRCN 缓存器去。	1	None
LDPR	k,f	将常数 k (9-bit) 搬到第 f 个 FSR 缓存器去 (f = 0 ~ 1) 。	2	None
MULF	f,a	将 W 与 F 做相乘。	2	None
MULL	k	将常数 k 与 W 做乘法运算。	2	None
MVF	f,d,a	将 W 内的值搬到 F 中(d=1)或 F 内的值搬到 W(d=0)。	1	None
MVFF	fs,fd	将 Fs 内的资料搬到 Fd 中。	2	None
MVL	k	将常数 k 搬到 W 去。	1	None
RETL	k	将堆栈最上层的值取出来放至 PC 中，并将 W 的值设为 k ，而主程序由目前 PC 值开始执行	2	None
RLF	f,d,a	将 F 内的值做左移动作，并将结果放至 W 或 F。	1	N,Z
RLFC	f,d,a	将 F 内的值与 C 一起做左移动作，并将结果放至 W 或 F。	1	C,N,Z
RRF	f,d,a	将 F 内的值做右移动作，并将结果放至 W 或 F。	1	N,Z
RRFC	f,d,a	将 F 内的值与 C 一起做右移动作，并将结果放至 W 或 F。	1	C,N,Z

Remark				
	f	缓存器	b	缓存器第 b 个位
	n	内存位置	k	8 位常数
	d	数据存放地方; d = 0 表示存放在 W 累加器；d = 1 表示存放在 f 缓存器。		
	a	数据存放在哪个内存位置,a=0 存放在目前内存位置；a=1 存放在 BSRCN 缓存器内所指定内存位置		

© 2008-2019 HYCON Technology Corp

指令快速索引(续)

Instruction		Description	Cycles	Status Affected
CONTROL OPERATIONS				
SETF	f,a	将 F 内的值设为 0xFF。	1	None
SUBC	f,d,a	将 F 内的值减掉 W 及反向 C，并将结果放至 W 或 F。	1	C,DC,N,OV,Z
SUBF	f,d,a	将 F 内的值减掉 W，并将结果放至 W 或 F。	1	C,DC,N,OV,Z
SUBL	k	将常数 k 与 W 做减法，并将结果放至 W。	1	C,DC,N,OV,Z
SWPF	f,d,a	将 F 内的值高 4 位与低 4 位对调，并将结果放至 W 或 F。	1	None
TFSZ	f,a	测试 F 内的值是否等于 0，若为 0 则跳过下一个指令。	1(2)(3)	None
XORF	f,d,a	将 W 与 F 做 XOR 运算，并将结果放至 W 或 F。	1	N,Z
XORL	k	将常数 k 与 W 做 XOR 运算，并将结果放至 W。	1	N,Z
JNN	n	若 N = 0 则跳到地址 n。	1(2)	None
JNO	n	若 OV = 0 则跳到地址 n。	1(2)	None
JNZ	n	若 Z = 0 则跳到地址 n。	1(2)	None
JO	n	若 OV = 1 则跳到地址 n。	1(2)	None
JZ	n	若 Z = 1 则跳到地址 n。	1(2)	None
NOP		空指令。	1	None
POP		将堆栈指针缓存器减 1 后，取出该堆栈指针所指向堆栈层中堆栈的值放回 TOS 缓存器中。	1	None
RCALL	n	将下一个指令的 PC 值存到堆栈的最上层，并跳到地址 n,-1024≦n≦1023	2	None
RET	s	由子程序返回主程序，并将堆栈最上层的值取出来放至 PC 中，而主程序由目前 PC 值开始执行。	2	None
RETI	s	由中断子程序返回主程序，并将堆栈最上层的值取出来放至 PC 中，而主程序由目前 PC 值开始执行。	2	GIE
RJ	n	无条件跳到地址 n,-1024≦n≦1023	2	None
SLP	f,a	进入睡眠状态。		PD

Remark

f	缓存器	b	缓存器第 b 个位
n	内存位置	k	8 位常数
d	数据存放地方; d = 0 表示存放在 W 累加器；d = 1 表示存放在 f 缓存器。		
a	数据存放在哪个内存位置,a=0 存放在目前内存位置；a=1 存放在 BSRCN 缓存器内所指定内存位置		

© 2008-2019 HYCON Technology Corp

指令快速索引(续)

Instruction		Description	Cycles	Status Affected
BIT-ORIENTED FILE REGISTER OPERATIONS				
BCF	f,b,a	将 F 内某个位 (Bit) 设定为 0。	1	None
BSF	f,b,a	将 F 内某个位 (Bit) 设定为 1。	1	None
BTGF	f,b,a	将 F 内某个位 (Bit) 做 NOT 运算。	1	None
BTSS	f,b,a	测试 F 内某个位 (Bit) 的值是否等于 1 ·若为 1 则跳过下一个指令。	1(2)(3)	None
BTSZ	f,b,a	测试 F 内某个位 (Bit) 的值是否等于 0 ·若为 0 则跳过下一个指令。	1(2)(3)	None
PROGRAM MEMORY OPERATIONS				
MVLP	k	将常数 k(0≦k≦16384)搬到 TABLE 指标器 (TBLPTRH/TBLPTRL)	2	None
TBLR	*	以 TBLPTR 记录器之内容为地址指针，读取程序内存之内容至 TBLDH/TBLDL 缓存器中。	2	None
TBLR	*+	以 TBLPTR 记录器之内容为地址指针，读取程序内存之内容至 TBLDH/TBLDL 缓存器中。然后将地址指针自动+1。	2	None

Remark				
	f	缓存器	b	缓存器第 b 个位
	n	内存位置	k	8 位常数
	d	数据存放地方; d = 0 表示存放在 W 累加器；d = 1 表示存放在 f 缓存器。		
	a	数据存放在哪个内存位置,a=0 存放在目前内存位置；a=1 存放在 BSRCN 缓存器内所指定内存位置		

© 2008-2019 HYCON Technology Corp

3 指令详解

3.1 ADDC

ADD w and Carry bit to f

Syntax: ADDC f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (W) + (f) + (Status<C>) → destination

Status Affected: C, DC, N, OV, Z

Description: W 累加器中的值与 f 缓存器中的值与进位旗标 C 三者相加，并将运算结果放回 d 所指定的暂存器中；
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: ADDC REG, 0, 0

Before Instruction:	**After Instruction:**
W=001H	W=020H
REG(080H)=01FH	REG(080H)=01FH
C=DC=N=OV=Z=0	DC=1, C= N=OV=Z=0

Remark: d=0，则执行结果放回 W 累加器中。

Example 2: ADDC REG, 1, 1 (if BSRCN=001H)

Before Instruction:	**After Instruction:**
W=001H	W=001H
REG(170H)=00EH	REG(170H)=010H
C=1, DC=N=OV=Z=0	DC=1, C= N=OV=Z=0

Remark: d=1，则执行结果放回 f 缓存器中。
a=1，则执行结果放回 BSRCN 缓存器所指定的 RAM 地址中。

© 2008-2019 HYCON Technology Corp

3.2 ADDF

ADD w to F

Syntax: ADDF f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (W) + (f) → destination

Status Affected: C, DC, N, OV, Z

Description: W 累加器中的值与 f 缓存器中的值相加，并将运算结果放回 d 所指定的缓存器中。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 的地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: ADDF REG, 0, 0

Before Instruction:
W=001H
REG(080H)=01FH
C=DC=N=OV=Z=0

After Instruction:
W=020H
REG(080H)=01FH
DC=1, C= N=OV=Z=0

Remark: d=0，则执行结果放回 W 累加器中。
a=0 为默认值，若程序中 a=0 时，则程序中可以不加入此参数。

Example 2: ADDF REG, 1

Before Instruction:
W=001H
REG(080H)=01FH
C=DC=N=OV=Z=0

After Instruction:
W=001H
REG(080H)=020H
DC=1, C= N=OV=Z=0

Remark: d=1，则执行结果放回 f 缓存器中。

a=0 为默认值，若程序中 a=0 时，则程序中可以不加入此参数。

Example 3: ADDF REG, 1, 1 (if BSRCN=001H)

Before Instruction:
W=001H
REG(070H)=00EH
REG1(170H)=01FH
C=DC=N=OV=Z=0

After Instruction:
W=001H
REG(070H)=00EH
REG1(170H)=020H
DC=1, C= N=OV=Z=0

Remark: d=1，则执行结果放回 f 缓存器中。
a=1，则执行结果放回 BSRCN 缓存器所指定的 RAM 地址中。

虽然 REG RAM 地址为 070H，但因为 BSRCN=001H，所以会和 170H 缓存器中的数值作运算后，将执行果放回地址 170H 处。

© 2008-2019 HYCON Technology Corp

3.3 ADDL

ADD Literal to w

Syntax: ADDL k

Operands: 0 ≦ k ≦ 255

Operation: (W) + K → W

Status Affected: C, DC, N, OV, Z

Description: W 累加器中的值与 k 值相加，并将运算结果放回 W 累加器中。

Words: 1

Cycles: 1

Example 1: ADDL 00FH

Before Instruction:

W=001H

C=DC=N=OV=Z=0

After Instruction:

W=010H

DC=1, C= N=OV=Z=0

Remark: 低四位相加后，有进位情形，半进位旗标 DC=1.

Example 2: ADDL 00FH

Before Instruction:

W=071H

C=DC=N=OV=Z=0

After Instruction:

W=080H

DC=N=OV=1, C=Z=0

Remark: 低四位相加后，有进位情形，半进位旗标 DC=1。
执行结果大于 127，则视为负数，负号旗标 N=1。
执行后，位 7=1，溢位旗标 OV=1。

Example 3: ADDL 00FH

Before Instruction:

W=081H

C=DC=N=OV=Z=0

After Instruction:

W=090H

DC=N=1, C=OV=Z=0

Remark: 低四位相加后，有进位情形，半进位旗标 DC=1。
执行结果大于 127，则视为负数，负号旗标 N=1。
执行前位 7=1，执行后位 7 仍为 1 的状态，则溢位旗标不变 OV=0。

Example 4: ADDL 00FH

Before Instruction:

W=0F1H

C=DC=N=OV=Z=0

After Instruction:

W=000H

C=DC= Z=1, N=OV= 0

Remark: 执行结果大于 0FFH，进位旗标 C=1。
低四位相加后，有进位情形，半进位旗标 DC=1。
执行结果为 000H，则零位旗标 Z=1。

© 2008-2019 HYCON Technology Corp

3.4 ANDF

AND w with F

Syntax: ANDF f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (W) AND (f) → destination

Status Affected: N, Z

Description: W 累加器中的值与 f 缓存器中的值做逻辑 AND 运算，并将运算结果放回 d 所指定的缓存器中。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中存放；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: ANDF REG, 0

Before Instruction:

W=055H

REG(080H)=0AAH

C=DC=N=OV=Z=0

After Instruction:

W=000H

REG(080H)=0AAH

Z=1, C=DC=N=OV=0

Remark: 执行结果为 000H，零位旗标 Z=1。
a=0 为默认值，若程序中 a=0 时，则程序中可以不加入此参数。

Example 2: ANDF REG, 1, 1 (if BSRCN=001H)

Before Instruction:

W=080H

REG(170H)=0FFH

C=DC=N=OV=Z=0

After Instruction:

W=080H

REG(170H)=080H

N=1, C=DC= OV=Z=0

Remark: 执行结果大于 127，视为负数，负号旗标 N=1。

© 2008-2019 HYCON Technology Corp

3.5 ANDL

AND Literal with w

Syntax: ANDL k

Operands: 0 ≦ k ≦ 255

Operation: (W) AND k → W

Status Affected: N, Z

Description: W 累加器中的值与 k 值做逻辑的 AND 运算，并将运算结果放回 W 累加器中。

Words: 1

Cycles: 1

Example 1: ANDL 0A0H

Before Instruction:

W=055H

C=DC=N=OV=Z=0

After Instruction:

W=000H

Z=1, C=DC=N=OV=0

Remark: 执行结果为 000H，零位旗标 Z=1。

Example 2: ANDL 0FF0H

Before Instruction:

W=080H

C=DC=N=OV=Z=0

After Instruction:

W=080H

N=1, C=DC= OV=Z=0

Remark: 执行结果大于 127，视为负数，负号旗标 N=1。

© 2008-2019 HYCON Technology Corp

3.6 ARLC

Another Rotate Left f through Carry

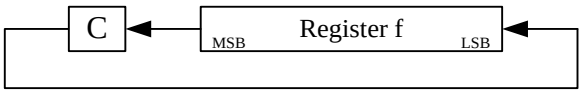
Syntax: ARLC f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f<n>) → destination <n+1 >, (f<7>) → Status< C >,
Status< C > → destination < 0 >

Status Affected: C, N, OV, Z

Description: 将 f 暂存内的值与进位旗标 C 一起向左旋转。(此指令相同于 RLFC 功能，只差异在 OV 旗标)
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。



Words: 1

Cycles: 1

Example 1: ARLC REG, 1, 0

Before Instruction:
WREG(02CH)=00FH
REG(080H)=0AAH
C=N=OV=Z=0

After Instruction:
WREG(02CH)=00FH
REG(080H)=054H
C=OV=1, N=Z=0

Remark: 执行结果 BIT7 由 1→0，所以溢位旗标 OV=1。

Example 2: ARLC REG, 0, 1

Before Instruction:
WREG(02CH)=0FH
REG(170H)=0EAH
C=N=OV=Z=0

After Instruction:
WREG(02CH)=0D4H
REG(170H)=0EAH
C=N=1, OV=Z=0

Example 3: ARLC REG, 1, 1

Before Instruction:
WREG(02CH)=00FH
REG(170H)=080H
C=N=OV=Z=0

After Instruction:
WREG(02CH)=00FH
REG(170H)=000H
C=OV=Z=1, N=0

© 2008-2019 HYCON Technology Corp

3.7 ARRC

Another Rotate Right f through Carry

Syntax: ARRC f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f<n>) → destination <n-1 >, (f<7>) → destination < 7 >,
(f<0>) → Status< C >

Status Affected: C, N, Z

Description: 将 f 暂存内的值向右旋转，BIT0 向右旋转到进位旗标 C，BIT7 保留在 BIT7 位置。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: ARRC REG, 1, 0

Before Instruction:
WREG(02CH)=00FH
REG(080H)=0AAH
C=N=Z=0

After Instruction:
WREG(02CH)=00FH
REG(080H)=0D5H
N=1, C= Z=0

Example 2: ARRC REG, 0, 1

Before Instruction:
WREG(02CH)=00FH
REG(17FH)=055H
C=N=Z=0

After Instruction:
WREG(02CH)=02AH
REG(17FH)=055H
C=1, N=Z=0

Example 3: ARRC REG, 1, 1

Before Instruction:
WREG(02CH)=00FH
REG(17FH)=001H
C=N=Z=0

After Instruction:
WREG(02CH)=00FH
REG(17FH)=000H
C=Z=1, N=0

© 2008-2019 HYCON Technology Corp

3.8 BCF

Bit Clear F

Syntax: BCF f, b, a

Operands: 0 ≦ f ≦ 255 ; 0 ≦ b ≦ 7 ; a € (0, 1)

Operation: 0 → f < b >

Status Affected: None

Description: 将 f 缓存器中的设定的位清除为 0。

Words: 1

Cycles: 1

Example 1: BCF REG,2

Before Instruction:

REG(080H)=1111 1111B

After Instruction:

REG(080H)=1111 1011B

Remark: 将 REG 缓存器中 BIT2 清除为 0，其他位值不变。

© 2008-2019 HYCON Technology Corp

3.9 BSF

Bit Set F

Syntax: BCF f, b, a

Operands: 0 ≦ f ≦ 255 ; 0 ≦ b ≦ 7 ; a € (0, 1)

Operation: 1 → f < b >

Status Affected: None

Description: 将 f 缓存器中的设定的位设定为 1。

Words: 1

Cycles: 1

Example 1: BSF REG,2

Before Instruction: REG(080H)=00000000B

After Instruction: REG(080H)=00000100B

Remark: 将 REG 缓存器中 BIT2 设定为 1，其他位值不变。

© 2008-2019 HYCON Technology Corp

3.10 BTGF

Bit ToGgle F

Syntax: BTGF f, b, a

Operands: 0 ≦ f ≦ 255 ; 0 ≦ b ≦ 7 ; a € (0, 1)

Operation: $\overline{(f < b >)} \rightarrow f < b >$

Status Affected: None

Description: 将缓存器中的某一个位取补码。

Words: 1

Cycles: 1

Example 1: BTGF REG, 7, 0

Before Instruction:

REG(080H)=0111 1111B

After Instruction:

REG(080H)=1111 1111B

Remark: 针对 REG 缓存器中 BIT7 取补码。

© 2008-2019 HYCON Technology Corp

3.11 BTSS

Bit Test and Skip if Set

Syntax: BTSS f, b, a

Operands: 0 ≦ f ≦ 255 ; 0 ≦ b ≦ 7 ; a € (0, 1)

Operation: skip if (f < b >)=1

Status Affected: None

Description: 比较缓存器中的某一个位是否为 1。若为 1 则跳过下一个指令，若不为 1 则往下执行。

Words: 1

Cycles: 1(2)(3)

Example 1:

```
BTSS    REG, 7, 0
MVL     001H
NOP
```

Before Instruction:

WREG(02CH)=000H

REG(080H)=0FFH

After Instruction:

WREG(02CH)=000H

REG(080H)=0FFH

Remark: REG 缓存器中 BIT7 为 1，所以跳过下一个指令。

Example 2:

```
BTSS    REG, 7, 0
MVL     001H
NOP
```

Before Instruction:

WREG(02CH)=000H

REG(080H)=07FH

After Instruction:

WREG(02CH)=001H

REG(080H)=07FH

Remark: REG 缓存器中 BIT7 为 0，所以程序直接往下执行。

© 2008-2019 HYCON Technology Corp

3.12 BTSZ

Bit Test and Skip if Zero

Syntax: BTSZ f, b, a

Operands: 0 ≦ f ≦ 255 ; 0 ≦ b ≦ 7 ; a € (0, 1)

Operation: skip if (f < b >)=0

Status Affected: None

Description: 比较缓存器中的某一个位是否为 0。若为 0 则跳过下一个指令，若不为 0 则往下执行。

Words: 1

Cycles: 1(2)(3)

Example 1:

```
BTSZ    REG, 7, 0
MVL     001H
NOP
```

Before Instruction:

WREG(02CH)=000H

REG(080H)=0FFH

After Instruction:

WREG(02CH)=001H

REG(080H)=0FFH

Remark: REG 缓存器中 BIT7 不为 0，所以程序直接往下执行。

Example 2:

```
BTSZ    REG, 7, 0
MVL     001H
NOP
```

Before Instruction:

WREG(02CH)=000H

REG(080H)=07FH

After Instruction:

WREG(02CH)=000H

REG(080H)=07FH

Remark: REG 缓存器中 BIT7 为 0，所以跳过下一个指令。

© 2008-2019 HYCON Technology Corp

3.13 CALL

subroutine CALL

Syntax: CALL n, s

Operands: 0 ≦ n ≦ 16384(03FFFH); s € (0, 1)

Operation: (PC) + 1 → TOS, n → PC,
If s=1,
(WREG) → WREGSDW,
(STATUS) → STASDW
(BSRCN) → BSRSDW

Status Affected: STKPTR<STKFL>, STKPTR<STKOV>, Pstatus<SKERR>.

Description: 呼叫子程序，呼叫的范围最大到 16Kbytes 的内存范围。

If s=1, WREG, STATUS, BSRCN 缓存器内的值会被放进相对应的遮蔽缓存器(shadow register)中。
若呼叫子程序之后堆栈层为该产品最后一层堆栈，则 STKFL 旗标会被设定为 1。
在 SBMSET1<7>=0 的条件下，堆栈层满之后再进行 CALL 指令则 STKOV 旗标会被设定为 1，SKERR 也会被设定为 1。PC 正常执行。
在 SBMSET1<7>=1 的条件下，堆栈层满之后再进行 CALL 指令则 STKOV 旗标会被设定为 1，SKERR 也会被设定为 1。芯片重置，PC 回到 000H。
STKFL 或 STKOV 发生时，只要其中一个旗标被清除时，两者同时都会被清除。

Words: 2

Cycles: 2

Example 1:

```
LABEL:  CALL    NEXT, 1
        .
        .
        .
NEXT:   NOP
```

Before Instruction:

PC = address (LABEL)

After Instruction:

PC= address (NEXT)
TOS= address (LABEL + 2)
WREGSDW= WREG
BSRSDW= BSRCN
STASDW=STATUS

Remark: s=1, 则 WREG, STATUS, BSRCN 缓存器内的值会被放进相对应的遮蔽缓存器(shadow register)中

© 2008-2019 HYCON Technology Corp

3.14 CLRF

CLeaR F

Syntax: CLRF f, a

Operands: 0 ≦ f ≦ 255 ; a € (0, 1)

Operation: 000H → f

Status Affected: None

Description: 将缓存器 f 的值全部清除为 0。

Words: 1

Cycles: 1

Example 1: CLRF REG, 0

Before Instruction:

REG(080H)=055H

After Instruction:

REG(080H)=000H

Remark: REG 缓存器中的数值被清为 0。

© 2008-2019 HYCON Technology Corp

3.15 COMF

COMplement F

Syntax: COMF f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: $\overline{(f)}$ → destination

Status Affected: N, Z

Description: 将缓存器中的值取补码，并将运算结果放回 d 所指定的缓存器中。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: COMF REG, 0, 0

Before Instruction:
WREG(02CH)=055H
REG(080H)=0FFH
N=Z=0

After Instruction:
WREG(02CH)=000H
REG(080H)=0FFH
Z=1, N=0

Example 2: COMF REG, 1, 1 (if BSRCN=001H)

Before Instruction:
WREG(02CH)=055H
REG(170H)=055H
N=Z=0

After Instruction:
WREG(02CH)=055H
REG(170H)=0AAH
N=1, Z=0

© 2008-2019 HYCON Technology Corp

3.16 CPSE

ComPare f with w, Skip if f Equal w

Syntax: CPSE f, a

Operands: 0 ≦ f ≦ 255 ; a € (0, 1)

Operation: skip if (f) = (W)

Status Affected: None

Description: 将缓存器的值与 W 累加器的值作比较，
若是两个值相等则跳过下一个指令，若大于或是小于则往下执行。

Words: 1

Cycles: 1(2)

Example 1:

```
CPSE    REG, 0
MVL     001H
NOP
```

Before Instruction:

WREG(02CH)=000H

REG(080H)=000H

After Instruction:

WREG(02CH)=000H

REG(080H)=000H

Remark: f 缓存器和 W 累加器数值相等，所以跳过下一个指令。

Example 2:

```
CPSE    REG, 1    (if BSRCN=001H)
MVL     001H
NOP
```

Before Instruction:

WREG(02CH)=000H

REG(170H)=07FH

After Instruction:

WREG(02CH)=001H

REG(170H)=07FH

Remark: f 缓存器和 W 累加器数值不相等，所以程序继续往下执行。

© 2008-2019 HYCON Technology Corp

3.17 CPSG

ComPare f with w, Skip if f Greater than w

Syntax: CPSG f, a

Operands: 0 ≦ f ≦ 255 ; a € (0, 1)

Operation: skip if (f) > (W)

Status Affected: None

Description: 将缓存器的值与 W 累加器的值作比较，
若是缓存器的值大于 W 累加器的值则跳过下一个指令，若小于或是等于则往下执行。

Words: 1

Cycles: 1(2)

Example 1:

```
CPSG    REG, 0
MVL     00FH
NOP
```

Before Instruction:

WREG(02CH)=005H

REG(080H)=006H

After Instruction:

WREG(02CH)=005H

REG(080H)=006H

Remark: f 缓存器的数值大于 W 累加器数值，所以跳过下一个指令。

Example 2:

```
CPSG    REG, 1    (if BSRCN=001H)
MVL     00FH
NOP
```

Before Instruction:

WREG(02CH)=005H

REG(170H)=005H

After Instruction:

WREG(02CH)=00FH

REG(170H)=005H

Remark: f 缓存器数值等于 W 累加器数值，所以程序继续往下执行。

© 2008-2019 HYCON Technology Corp

3.18 CPSL

ComPare f with w, Skip if f Less than w

Syntax: CPSL f, a

Operands: 0 ≦ f ≦ 255 ; a € (0, 1)

Operation: skip if (f) < (W)

Status Affected: None

Description: 将缓存器的值与 W 累加器的值作比较，
若是缓存器的值小于 W 累加器的值则跳过下一个指令，若大于或是等于则往下执行。

Words: 1

Cycles: 1(2)

Example 1:

```
CPSL    REG, 0
MVL     00FH
NOP
```

Before Instruction:

WREG(02CH)=005H

REG(080H)=004H

After Instruction:

WREG(02CH)=005H

REG(080H)=004H

Remark: f 缓存器的数值小于 W 累加器数值，所以跳过下一个指令。

Example 2:

```
CPSL    REG, 1    (if BSRCN=001H)
MVL     00FH
NOP
```

Before Instruction:

WREG(02CH)=005H

REG(170H)=005H

After Instruction:

WREG(02CH)=00FH

REG(170H)=005H

Remark: f 缓存器数值等于 W 累加器数值，所以程序继续往下执行。

© 2008-2019 HYCON Technology Corp

3.19 CWDT

Clear WatchDog Timer

Syntax: CWDT

Operands: None

Operation: 000H → Watch dog counter

Status Affected: Pstatus<TO>

Description: 将看门狗定时器的值全部清除为 0。

Words: 1

Cycles: 1

Example 1: CWDT

Before Instruction:

WDT counter = ???

After Instruction:

WDT counter = 000H
Pstatus<TO> = 0

© 2008-2019 HYCON Technology Corp

3.20 DCF

DeCrement F

Syntax: DCF f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f) – 1 → destination

Status Affected: C, DC, N, OV, Z

Description: 将缓存器 f 的值减 1，并将运算结果放回 d 所指定的缓存器中。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: DCF REG, 0, 0

Before Instruction:	After Instruction:
WREG(02CH)=055H	WREG(02CH)=0FEH
REG(080H)=0FFH	REG(080H)=0FFH
C=DC=N=OV=Z=0	C=DC=N=1, OV=Z=0

Remark: C, DC 未被借位，所以 C=DC=1；执行后结果大于 127，所以 N=1。

Example 2: DCF REG, 1, 1 (if BSRCN=001H)

Before Instruction:	After Instruction:
WREG(02CH)=055H	WREG(02CH)=055H
REG(170H)=000H	REG(170H)=0FFH
C=DC=N=OV=Z=0	N=1, C=DC=OV=Z=0

Remark: C, DC 皆被借位，所以 C=DC=0；执行后结果仍大于 127，所以 N=1。

Example 3: DCF REG, 1, 0

Before Instruction:	After Instruction:
WREG(02CH)=055H	WREG(02CH)=055H
REG(080H)=080H	REG(080H)=07FH
C=DC=N=OV=Z=0	C= OV=1, DC= N= Z=0

Remark: 只有 DC 被借位，所以 DC=0, C=1；执行后 BIT7 被变动，由 1 变 0，所以 OV=1。

Example 4: DCF REG, 0, 0

Before Instruction:	After Instruction:
WREG(02CH)=055H	WREG(02CH)=000H
REG(080H)=001H	REG(080H)=001H
C=DC=N=OV=Z=0	C= DC= Z=1, N=OV=0

Remark: C, DC 未被借位，所以 C=DC=1；执行后结果为 0，所以 Z=1。

© 2008-2019 HYCON Technology Corp

3.21 DCSUZ

DeCrement f, Skip if Un-Zero

Syntax: DCSUZ f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f) – 1 → destination, skip if destination≠0

Status Affected: None

Description: 将缓存器的值减 1 后与 0 作比较，若是缓存器的值不等于 0 则跳过下一个指令，若等于 0 则往下执行，并将运算结果放回 d 所指定的缓存器中。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1(2)(3)

Example 1:

```
DCSUZ   REG, 1, 0
MVL     00AH
NOP
```

Before Instruction:

WREG(02CH)=00FH

REG(080H)=001H

After Instruction:

WREG(02CH)=00AH

REG(080H)=000H

Remark: 执行结果为 0，所以继续往下执行程序段，执行结果被放回 REG 缓存器。

Example 2:

```
DCSUZ   REG, 0, 1  (if BSRCN=001H)
MVL     00AH
NOP
```

Before Instruction:

WREG(02CH)=055H

REG(170H)=000H

After Instruction:

WREG(02CH)=0FFH

REG(170H)=000H

Remark: 执行结果不为 0，所以跳过下一个指令，执行结果被放回 W 累加器。

© 2008-2019 HYCON Technology Corp

3.22 DCSZ

DeCrement f, Skip if Zero

Syntax: DCSZ f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f) – 1 → destination, skip if destination=0

Status Affected: None

Description: 将缓存器的值减 1 后与 0 作比较，若是缓存器的值等于 0 则跳过下一个指令，若不等于 0 则往下执行，并将运算结果放回 d 所指定的缓存器中。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1(2)(3)

Example 1:

```
DCSZ    REG, 0, 0
MVL     00AH
NOP
```

Before Instruction:

WREG(02CH)=00FH

REG(080H)=001H

After Instruction:

WREG(02CH)=000H

REG(080H)=001H

Remark: 执行结果为 0，所以跳过下一个指令。

Example 2:

```
DCSZ    REG, 1, 1  (if BSRCN=001H)
MVL     00AH
NOP
```

Before Instruction:

WREG(02CH)=055H

REG(170H)=000H

After Instruction:

WREG(02CH)=00AH

REG(170H)=0FFH

Remark: 执行结果不为 0，所以继续往下执行程序段，执行结果被放回 REG 缓存器。

© 2008-2019 HYCON Technology Corp

3.23 IDLE

IDLE mode

Syntax: IDLE

Operands: None

Operation: CPU Halt

Status Affected: Pstatus<IDLEB>

Description: CPU 进入暂停模式，程序空指令执行动作。
IDLE 指令后，建议加上 NOP 指令。

Words: 1

Cycles: 1

Example 1:
```
IDLE
NOP
```

Before Instruction:

Pstatus<IDLEB>=0

After Instruction:

Pstatus<IDLEB>=1

IDLE

NOP……program break here

© 2008-2019 HYCON Technology Corp

3.24INF

INcrement F

Syntax: INF f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f) + 1 → destination

Status Affected: C, DC, N, OV, Z

Description: 将缓存器 f 的值加 1，并将运算结果放回 d 所指定的缓存器中。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: INF REG, 0, 0

Before Instruction:
WREG(02CH)=055H
REG(080H)=0FFH
C=DC=N=OV=Z=0

After Instruction:
WREG(02CH)=000H
REG(080H)=0FFH
C=DC= Z=1, N=OV=0

Remark: C, DC 进位所以 C=DC=1；执行后结果等于 0，所以 Z=1。

Example 2: INF REG, 1, 1 (if BSRCN=001H)

Before Instruction:
WREG(02CH)=055H
REG(170H)=00FH
C=DC=N=OV=Z=0

After Instruction:
WREG(02CH)=055H
REG(170H)=010H
DC=1, C=N=OV=Z=0

Remark: DC 进位所以 DC=1。

Example 3: INF REG, 1, 0

Before Instruction:
WREG(02CH)=055H
REG(080H)=07FH
C=DC=N=OV=Z=0

After Instruction:
WREG(02CH)=055H
REG(080H)=080H
DC= N=OV=1, C=Z=0

Remark: DC 进位所以 DC=1；执行后 BIT7 被变动，由 0 变 1，所以 OV=1；执行结果大于 127，所以 N=1。

© 2008-2019 HYCON Technology Corp

3.25 INSUZ

INcrement f, Skip if Un-Zero

Syntax: INSUZ f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f) + 1 → destination, skip if destination≠0

Status Affected: None

Description: 将缓存器的值加 1 后与 0 作比较，若是缓存器的值不等于 0 则跳过下一个指令，若等于 0 则往下执行，并将运算结果放回 d 所指定的缓存器中。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1(2)(3)

Example 1:

```
INSUZ   REG, 1, 0
MVL     00AH
NOP
```

Before Instruction:

WREG(02CH)=00FH

REG(080H)=0FFH

After Instruction:

WREG(02CH)=00AH

REG(080H)=000H

Remark: 执行结果为 0，所以继续往下执行程序段，执行结果被放回 REG 缓存器。

Example 2:

```
INSUZ   REG, 0, 1  (if BSRCN=001H)
MVL     00AH
NOP
```

Before Instruction:

WREG(02CH)=055H

REG(170H)=000H

After Instruction:

WREG(02CH)=001H

REG(170H)=000H

Remark: 执行结果不为 0，所以跳过下一个指令，执行结果被放回 W 累加器。

© 2008-2019 HYCON Technology Corp

3.26 INSZ

INcrement f, Skip if Zero

Syntax: INSZ f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f) + 1 → destination, skip if destination=0

Status Affected: None

Description: 将缓存器的值加 1 后与 0 作比较，若是缓存器的值等于 0 则跳过下一个指令，若不等于 0 则往下执行，并将运算结果放回 d 所指定的缓存器中。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1(2)(3)

Example 1:

```
INSZ    REG, 0, 0
MVL     00AH
NOP
```

Before Instruction:

WREG(02CH)=00FH

REG(080H)=0FFH

After Instruction:

WREG(02CH)=000H

REG(080H)=0FFH

Remark: 执行结果为 0，所以跳过下一个指令。

Example 2:

```
INSZ    REG, 1, 1  (if BSRCN=001H)
MVL     00AH
NOP
```

Before Instruction:

WREG(02CH)=055H

REG(170H)=000H

After Instruction:

WREG(02CH)=00AH

REG(170H)=001H

Remark: 执行结果不为 0，所以继续往下执行程序段，执行结果被放回 REG 缓存器。

© 2008-2019 HYCON Technology Corp

3.27 IORF

Inclusive OR w with F

Syntax: IORF f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (W) OR (f) → destination

Status Affected: N, Z

Description: W 累加器中的值与 f 缓存器中的值作逻辑的 OR 运算，并将运算结果放回 d 所指定的缓存器中。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: IORF REG, 0, 0

Before Instruction:
WREG(02CH)=055H
REG(080H)=0AAH
N=Z=0

After Instruction:
WREG(02CH)=0FFH
REG(080H)=0AAH
N=1, Z=0

Remark: 执行结果大于 127，所以 N=1。

Example 2: IORF REG, 1, 1 (if BSRCN=001H)

Before Instruction:
WREG(02CH)=00FH
REG(170H)=0F0H
N=Z=0

After Instruction:
WREG(02CH)=00FH
REG(170H)=0FFH
N=1, Z=0

Remark: 执行结果大于 127，所以 N=1。

© 2008-2019 HYCON Technology Corp

3.28 IORL

Inclusive OR Literal with w

Syntax: IORL k

Operands: 0 ≦ k ≦ 255

Operation: (W) OR k → W

Status Affected: N, Z

Description: W 累加器中的值与 k 值作逻辑的 OR 运算，并将运算结果放回 W 累加器中。

Words: 1

Cycles: 1

Example 1: IORL 055H

Before Instruction:

WREG(02CH)=0AAH

N=Z=0

After Instruction:

WREG(02CH)=0FFH

N=1, Z=0

Remark: 执行结果大于 127，所以 N=1。

Example 2: IORL 000H

Before Instruction:

WREG(02CH)=000H

N=Z=0

After Instruction:

WREG(02CH)=000H

Z=1, N=0

Remark: 执行结果等于 0，所以 Z=1。

© 2008-2019 HYCON Technology Corp

3.29 JC

Jump if Carry

Syntax: JC n

Operands: -128 ≦ n ≦ 127

Operation: If Status <carry bit> is 1, jump to n

Status Affected: None

Description: 当状态缓存器中的进位旗标 C 等于 1 时，就跳到指定的地址 n。

Words: 1

Cycles: 1(2)

Example 1:

```
LABEL:  JC   NEXT
          .
          .
          .
NEXT:   NOP
```

Before Instruction:

PC = address (LABEL)

After Instruction:

If C=0, PC= address (LABEL + 1)

If C=1, PC= address (NEXT)

© 2008-2019 HYCON Technology Corp

3.30 JMP

unconditional JuMP

Syntax: JMP n

Operands: 0 ≦ n ≦ 16384(03FFFH)

Operation: n → PC

Status Affected: None

Description: 无条件跳跃至指定的地址 n。

Words: 2

Cycles: 2

Example 1:

```
LABEL:  JMP  NEXT
          .
          .
          .
NEXT:   NOP
```

Before Instruction:

PC = address (LABEL)

After Instruction:

PC= address (NEXT)

© 2008-2019 HYCON Technology Corp

3.31 JN

Jump if Negative

Syntax: JN n

Operands: -128 ≦ n ≦ 127

Operation: If Status <negative bit> is 1, jump to n

Status Affected: None

Description: 当状态缓存器中的负号旗标 N 等于 1 时，就跳到指定的地址 n。

Words: 1

Cycles: 1(2)

Example 1:

```
LABEL:  JN   NEXT
        .
        .
        .
NEXT:   NOP
```

Before Instruction:

PC = address (LABEL)

After Instruction:

If N=0, PC= address (LABEL + 1)

If N=1, PC= address (NEXT)

© 2008-2019 HYCON Technology Corp

3.32 JNC

Jump if Not Carry

Syntax: JNC n

Operands: -128 ≦ n ≦ 127

Operation: If Status <carry bit> is 0, jump to n

Status Affected: None

Description: 当状态缓存器中的进位旗标 C 等于 0 时，就跳到指定的地址 n。

Words: 1

Cycles: 1(2)

Example 1:

```
LABEL:  JNC  NEXT
          .
          .
          .
NEXT:   NOP
```

Before Instruction:

PC = address (LABEL)

After Instruction:

If C=0, PC= address (NEXT)

If C=1, PC= address (LABEL + 1)

© 2008-2019 HYCON Technology Corp

3.33 JNN

Jump if Not Negative

Syntax: JNN n

Operands: -128 ≦ n ≦ 127

Operation: If Status <negative bit> is 0, jump to n

Status Affected: None

Description: 当状态缓存器中的负号旗标 N 等于 0 时，就跳到指定的地址 n。

Words: 1

Cycles: 1(2)

Example 1:

```
LABEL:  JNN  NEXT
          .
          .
          .
NEXT:   NOP
```

Before Instruction:

PC = address (LABEL)

After Instruction:

If N=0, PC= address (NEXT)

If N=1, PC= address (LABEL + 1)

© 2008-2019 HYCON Technology Corp

3.34 JNO

Jump if Not Overflow

Syntax: JNO n

Operands: -128 ≦ n ≦ 127

Operation: If Status <overflow bit> is 0, jump to n

Status Affected: None

Description: 当状态缓存器中的溢位旗标 OV 等于 0 时，就跳到指定的地址 n。

Words: 1

Cycles: 1(2)

Example 1:

```
LABEL:  JNO NEXT
          .
          .
          .
NEXT:   NOP
```

Before Instruction:

PC = address (LABEL)

After Instruction:

If OV=0, PC= address (NEXT)

If OV=1, PC= address (LABEL + 1)

© 2008-2019 HYCON Technology Corp

3.35 JNZ

Jump if Not Zero

Syntax: JNZ n

Operands: -128 ≦ n ≦ 127

Operation: If Status <zero bit> is 0, jump to n

Status Affected: None

Description: 当状态缓存器中的零位旗标 Z 等于 0 时，就跳到指定的地址 n。

Words: 1

Cycles: 1(2)

Example 1:

```
LABEL:  JNZ  NEXT
          .
          .
          .
NEXT:   NOP
```

Before Instruction:

PC = address (LABEL)

After Instruction:

If Z=0, PC= address (NEXT)

If Z=1, PC= address (LABEL + 1)

© 2008-2019 HYCON Technology Corp

3.36 JO

Jump if Overflow

Syntax: JO n

Operands: -128 ≦ n ≦ 127

Operation: If Status <overflow bit> is 1, jump to n

Status Affected: None

Description: 当状态缓存器中的溢位旗标 OV 等于 1 时，就跳到指定的地址 n。

Words: 1

Cycles: 1(2)

Example 1:

```
LABEL:  JO   NEXT
          .
          .
          .
NEXT:   NOP
```

Before Instruction:

PC = address (LABEL)

After Instruction:

If OV=0, PC= address (LABEL + 1)

If OV=1, PC= address (NEXT)

© 2008-2019 HYCON Technology Corp

3.37 JZ

Jump if Zero

Syntax: JZ n

Operands: -128 ≦ n ≦ 127

Operation: If Status <zero bit> is 1, jump to n

Status Affected: None

Description: 当状态缓存器中的零位旗标 Z 等于 1 时，就跳到指定的地址 n。

Words: 1

Cycles: 1(2)

Example 1:

```
LABEL:  JZ    NEXT
          .
          .
          .
NEXT:   NOP
```

Before Instruction:

PC = address (LABEL)

After Instruction:

If Z=0, PC= address (LABEL + 1)

If Z=1, PC= address (NEXT)

© 2008-2019 HYCON Technology Corp

3.38 LBSR

Load literal into Bank Select Register

Syntax: LBSR k

Operands: 0 ≦ k ≦ 7

Operation: k → BSRCN

Status Affected: None

Description: 将常数 k 搬到分页缓存器 (Bank Select Register, BSRCN) 中，以设定存取数据的起始地址。

Words: 1

Cycles: 1

Example 1:

```
LBSR    001H            …… total instruction cycles = 1
```

Before Instruction: BSRCN=000H

After Instruction: BSRCN=001H

Remark: 设定 BSRCN=001H。

Example 2:

```
MVL     001H
MVF     BSRCN, 1, 0     …… total instruction cycles = 2
```

Before Instruction: BSRCN=000H

After Instruction: BSRCN=001H

Remark: 此范例程序动作相同于 Example 1。

© 2008-2019 HYCON Technology Corp

3.39 LDPR

LoaD Point into fsR

Syntax: LDPR k, f

Operands: 0 ≦ k ≦ 1279(04FFH) ; 0 ≦ f ≦ 1

Operation: k → FSR (FSRxH, FSRxL)

Status Affected: None

Description: 数据的寻址方式除了有直接寻址与立即寻址外，还有一种就是间接寻址。此指令目的就是简化间接寻址的设定方式。
间接寻址所使用的缓存器是 FSR(File Select Register)缓存器，FSR 缓存器所放的是数据的位址，而数据数值就放在 INDF 这个缓存器中。目前间接寻址提供可以使用的 FSR 缓存器有 2 个，分别是 FSR0、FSR1 缓存器。而定义的内存地址长度可以到达 11 bits, 分别为高低位两个缓存器；分别将 FSR0 分成 FSR0H 与 FSR0L；FSR1 分成 FSR1H 与 FSR1L。而相对应的记忆体数据分别存放于 INDF0, INDF1 之间。

Words: 2

Cycles: 2

Example 1:

```
LDPR    017FH, 0
MVL     0AAH
MVF     INDF0, 1, 0        …… total instruction cycles = 4
```

Before Instruction:	After Instruction:
FSR0H=000H	FSR0H=001H
FSR0L=080H	FSR0L=07FH
INDF0=0FFH	INDF0=0AAH
Address (017FH)=055H	Address (017FH)=0AAH

Remark: f=0 为默认值 FSR0，若程序中 f 所下参数为 0，则可以省略不下。
范例程序中说明使用间接寻址方式对数据地址 address(017FH)地址处写入数据数值 0AAH。

Example 2:

```
MVL     07FH
MVF     FSR0L, 1, 0
MVL     001H
MVF     FSR0H, 1, 0
MVL     0AAH
MVF     INDF0, 1, 0        …… total instruction cycles = 6
```

Remark: 此范例程序动作相同于 Example 1。

© 2008-2019 HYCON Technology Corp

3.40 MULF

MULtiply w with F

Syntax: MULF f, a

Operands: 0 ≦ f ≦ 255 ; a € (0, 1)

Operation: (W) × (f) → PRODH (high byte), PRODL (low byte)

Status Affected: None

Description: 将 W 累加器中的值与缓存器 f 的值相乘，并将结果放到 PRODH, PRODL 这二个缓存器中。
a = 0 or a =1 的设定须取决 f 缓存器位于 RAM 地址：
若 a = 0，则表示 f 缓存器存在于 080H 到 0FFH 所指定的 RAM 地址中(BSRCN=000H)。
若 a = 1，则表示 f 缓存器存在于 100H 到 17FH 所指定的 RAM 地址中(BSRCN=001H)。

Words: 1

Cycles: 2

Example 1: MULF REG, 1

Before Instruction:

WREG(02CH)=00FH
REG(017FH)=0FFH
PRODH=??
PRODL=??

After Instruction:

WREG(02CH)=00FH
REG(017FH)=0FFH
PRODH=00EH
PRODL=0F1H

Remark: 使用直接寻址执行乘法运算。

Example 2: MULF INDF0, 0

Before Instruction:

WREG(02CH)=00FH
FSR0H=001H, FSR0L=07FH
Address (017FH)=0FFH
PRODH=??
PRODL=??

After Instruction:

WREG(02CH)=00FH
FSR0H=001H, FSR0L=07FH
Address (017FH)=0FFH
PRODH=00EH
PRODL=0F1H

Remark: 使用间接寻址执行乘法运算。

© 2008-2019 HYCON Technology Corp

3.41 MULL

MULtiply Literal with w

Syntax: MULL k

Operands: 0 ≦ k ≦ 255

Operation: (W) × k → PRODH (high byte), PRODL (low byte)

Status Affected: None

Description: 将常数 k 与 W 累加器中的值相乘，并将结果放到 PRODH, PRODL 这二个缓存器。

Words: 1

Cycles: 2

Example 1: MULL 0FFH

Before Instruction:

WREG(02CH)=00FH

PRODH=??

PRODL=??

After Instruction:

WREG(02CH)=00FH

PRODH=00EH

PRODL=0F1H

© 2008-2019 HYCON Technology Corp

3.42 MVF

MoVe F to w or MoVe w to F

Syntax: MVF f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f) → W, or (W) → f

Status Affected: None

Description: 将 f 缓存器的值搬到 W 累加器中；或是将 W 累加器的值搬到 f 缓存器中。

若 d = 0，则表示将 f 缓存器的值搬到 W 累加器中；
若 d = 1，则表示将 W 累加器的值搬到 f 缓存器中；
a = 0 or a =1 的设定须取决 f 缓存器位于 RAM 地址：
若 a = 0，则表示 f 缓存器存在于 080H 到 0FFH 所指定的 RAM 地址中(BSRCN=000H)。
若 a = 1，则表示 f 缓存器存在于 100H 到 17FH 所指定的 RAM 地址中(BSRCN=001H)。

Words: 1

Cycles: 1

Example 1: MVF REG, 0, 0

Before Instruction:

WREG(02CH)=055H
REG(080H)=0AAH
REG1(170H)=0FFH

After Instruction:

WREG(02CH)=0AAH
REG(080H)=0AAH
REG1(170H)=0FFH

Remark: d=0，表示将 REG 缓存器的值搬到 W 累加器中。

Example 2: MVF REG1, 1, 1 (if BSRCN=001H)

Before Instruction:

WREG(02CH)=055H
REG1(170H)=0FFH
REG(080H)=0AAH

After Instruction:

WREG(02CH)=055H
REG1(170H)=055H
REG(080H)=0AAH

Remark: d=1，表示将将 W 累加器的值搬到 f 缓存器中。

© 2008-2019 HYCON Technology Corp

3.43 MVFF

MoVe F to F

Syntax: MVFF fs, fd

Operands: 0 ≦ fs ≦ 1279(04FFH) ; 0 ≦ fd ≦ 1279(04FFH)

Operation: (fs) → fd

Status Affected: None

Description: 将缓存器 fs 的数值般到缓存器 fd 去。

Words: 2

Cycles: 2

Example 1: MVFF REG, REG1

Before Instruction:	**After Instruction:**
REG=055H	REG=055H
REG1=0AAH	REG1=055H

Remark: 将缓存器 fs 的数值般到缓存器 fd 去，数据迁移过程不经由 W 累加器。

© 2008-2019 HYCON Technology Corp

3.44 MVL

MoVe Literal to w

Syntax: MVL k

Operands: 0 ≦ k ≦ 255

Operation: k → W

Status Affected: None

Description: 将常数 k 搬到 W 累加器中。

Words: 1

Cycles: 1

Example 1: MVL 0FFH

Before Instruction:

WREG(02CH)=000H

After Instruction:

WREG(02CH)=0FFH

© 2008-2019 HYCON Technology Corp

3.45 MVLP

MoVe Literal to Pointer

Syntax: MVLP k

Operands: 0 ≦ k ≦ 16384(03FFFh)

Operation: k → TBLPTR (TBLPTRH, TBLPTRL)

Status Affected: None

Description: MVLP 是一个设定程序内存指针的指令，多用在查表指令搭配 TBLR 使用。

Words: 2

Cycles: 2

Example 1: MVLP 001FF0H

Before Instruction:	**After Instruction:**
TBLPTRH=000H	TBLPTRH=01FH
TBLPTRL=000H	TBLPTRL=0F0H

Remark: 进行程序内存指针设定，将常数 k 加载 TBLPTR 缓存器。

© 2008-2019 HYCON Technology Corp

3.46 NOP

No OPeration

Syntax: NOP

Operands: None

Operation: No operation

Status Affected: None

Description: 不做任何运算，只延迟一个指令时间。

Words: 1

Cycles: 1

Example 1: NOP

Remark: 空指令，只做一个指令周期的延迟时间。

© 2008-2019 HYCON Technology Corp

3.47 POP

POP return stack

Syntax: POP

Operands: None

Operation: (TOS) → Bit bucket, then ((TOS) at STKPTR-1) → TOS

Status Affected: None

Description: 将堆栈指针(Stack Pointer)所指向堆栈层中的堆栈值丢弃，并将堆栈指针缓存器减 1 后，取出该堆栈指针所指向堆栈层中堆栈的值，将其放在 TOS 这个缓存器中。
而 TOS 缓存器被分为 TOSH, TOSL。

Words: 1

Cycles: 1

Example 1:

```
LABEL:  POP
RJ      LABEL1
LABEL1: NOP
```

Before Instruction:

STKPTR=003H
TOS= 001666H
TOS (STKPTR=002H) = 001234H
TOS (STKPTR=001H) = 000567H
PC=LABEL

After Instruction:

STKPTR=002H
TOS= 001234H
PC=LABEL1

Remark: 如果 STKPTR=00H，则执行 POP 指令并不会造成任何影响，TOS 仍为 0，STKPTR 亦为 0。

© 2008-2019 HYCON Technology Corp

3.48 RCALL

Relative subroutine CALL

Syntax: RCALL n

Operands: -1024 ≦ n ≦ 1023

Operation: (PC) + 1 → TOS, n → PC,

Status Affected: STKPTR<STKFL>, STKPTR<STKOV>, Pstatus<SKERR>.

Description: 呼叫子程序，呼叫的范围最大到±1K bytes 的内存范围。
若呼叫子程序之后堆栈层为该产品最后一层堆栈，则 STKFL 旗标会被设定为 1。
在 SBMSET1<7>=0 的条件下，堆栈层满之后再进行 RCALL 指令则 STKOV 旗标会被设定为 1，SKERR 也会被设定为 1。PC 正常执行。
在 SBMSET1<7>=1 的条件下，堆栈层满之后再进行 RCALL 指令则 STKOV 旗标会被设定为 1，SKERR 也会被设定为 1。芯片重置，PC 回到 000H。
STKFL 或 STKOV 发生时，只要其中一个旗标被清除时，两者同时都会被清除。

Words: 1

Cycles: 2

Example 1:

```
LABEL:  RCALL   NEXT
          .
          .
          .
NEXT:   NOP
```

Before Instruction:

PC = address (LABEL)

TOS=??

After Instruction:

PC= address (NEXT)

TOS= address (LABEL + 2)

© 2008-2019 HYCON Technology Corp

3.49 RET

RETurn from subroutine

Syntax: RET s

Operands: s € (0, 1)

Operation: (TOS) → PC,
If s=1,
(WREGSDW) → WREG,
(STASDW) → STATUS,
(BSRSDW) → BSRCN

Status Affected: STKPTR<STKUN>, Pstatus<SKERR>

Description: 离开子程序，并将推迭指针缓存器的数值存到 PC 中。
If s=1, 遮蔽缓存器(shadow register)中的值，会被放回相对应缓存器中(WREG, STATUS, BSRCN)。
当未呼叫子程序且 STKPTR=000H 时，执行 RET 指令时会造成芯片重置，STKUN 旗标会被设定为 1，SKERR 旗标会被设定为 1。

Words: 1

Cycles: 2

Example 1: RET 1

Before Instruction:

None

After Instruction:

PC=TOS
WREG = WREGSDW
BSRCN = BSRSDW
STATUS = STASDW

Remark: s=1, 所以遮蔽缓存器(shadow register)中的值，会被放回相对应缓存器中(WREG, STATUS, BSRCN)。

© 2008-2019 HYCON Technology Corp

3.50 RETI

RETurn from Interrupt

Syntax: RETI s

Operands: s € (0, 1)

Operation: (TOS) → PC, 1 → GIE
If s=1,
(WREGSDW) → WREG,
(STASDW) → STATUS,
(BSRSDW) → BSRCN

Status Affected: GIE, STKPTR<STKUN>, Pstatus<SKERR>

Description: 离开中断程序子程序，并将推迭指针缓存器的值存到 PC 中，中断致能接脚再度被设定为 1。
If s=1, 遮蔽缓存器(shadow register)中的值，会被放回相对应缓存器中(WREG, STATUS, BSRCN)。
当未呼叫子程序且 STKPTR=000H 时，执行 RETI 指令时会造成芯片重置，STKUN 旗标会被设定为 1， SKERR 旗标会被设定为 1。

Words: 1

Cycles: 2

Example 1: RETI 1

Before Instruction:

None

After Instruction:

PC=TOS
WREG = WREGSDW
BSRCN = BSRSDW
STATUS = STASDW
GIE=1

© 2008-2019 HYCON Technology Corp

3.51 RETL

RETurn Literal to w

Syntax: RETL k

Operands: 0 ≦ k ≦ 255

Operation: k → W, (TOS) → PC

Status Affected: STKPTR<STKUN>, Pstatus<SKERR>

Description: 从子程序返回主程序的指令，但本指令在返回的时候，还会顺便将常数 k 载入到 W 累加器中。此指令多半在查表法时会用到。
当未呼叫子程序且 STKPTR=000H 时，执行 RETL 指令时会造成芯片重置，STKUN 旗标会被设定为 1，SKERR 旗标会被设定为 1。

Words: 1

Cycles: 2

Example 1:

```
LABEL:  MVL     001H
        CALL    TABLE
        .
        .
        .
TABLE:  ADDF    PCLATL, 1, 0
        RETL        055H
        RETL        0AAH
```

Before Instruction:

WREG(02CH)=001H

After Instruction:

WREG(02CH)=0AAH

Remark: 返回主程序时，顺便将常数 k 加载到 W 累加器中。
此范例为藉由写入 PCLATL 的 Offset 数值来决定取得 TABLE 表中第几笔数值。

© 2008-2019 HYCON Technology Corp

3.52 RJ

unconditional Relative Jump

Syntax: RJ n

Operands: -1024 ≦ n ≦ 1023

Operation: n → PC

Status Affected: None

Description: 无条件跳跃至指定的地址 n。

Words: 1

Cycles: 2

Example 1:

```
LABEL:  RJ   NEXT
          .
          .
          .
NEXT:   NOP
```

Before Instruction:

PC = address (LABEL)

After Instruction:

PC= address (NEXT)

© 2008-2019 HYCON Technology Corp

3.53 RLF

Rotate Left F (no carry)

Syntax: RLF f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f<n>) → destination <n+1 >,

(f<7>) → destination < 0 >

Status Affected: N, Z

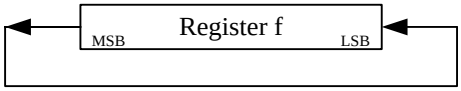
Description: 将 f 暂存内的值向左旋转。

若 d = 0，则运算后的结果放到 W 累加器中；

若 d = 1，则运算后的结果放到 f 缓存器中；

若 a = 0，则运算后的结果放到目前 RAM 地址中；

若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。



Words: 1

Cycles: 1

Example 1: RLF REG, 1, 0

Before Instruction:

WREG(02CH)=00FH

REG(080H)=0AAH

N=Z=0

After Instruction:

WREG(02CH)=00FH

REG(080H)=055H

N=Z=0

Example 2: RLF REG, 0, 1

Before Instruction:

WREG(02CH)=00FH

REG(17FH)=000H

N=Z=0

After Instruction:

WREG(02CH)=000H

REG(17FH)=000H

Z=1, N=0

Example 3: RLF REG, 0, 0

Before Instruction:

WREG(02CH)=00FH

REG(080H)=055H

N=Z=0

After Instruction:

WREG(02CH)=0AAH

REG(080H)=055H

N=1, Z=0

© 2008-2019 HYCON Technology Corp

3.54 RLFC

Rotate Left F through Carry

Syntax: RLFC f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f<n>) → destination <n+1 >,

(f<7>) → Status< C >,

Status< C > → destination < 0 >

Status Affected: C, N, Z

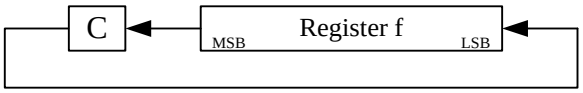
Description: 将 f 暂存内的值与进位旗标 C 一起向左旋转。

若 d = 0，则运算后的结果放到 W 累加器中；

若 d = 1，则运算后的结果放到 f 缓存器中；

若 a = 0，则运算后的结果放到目前 RAM 地址中；

若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。



Words: 1

Cycles: 1

Example 1: RLFC REG, 1, 0

Before Instruction:

WREG(02CH)=00FH

REG(080H)=0AAH

C=N=Z=0

After Instruction:

WREG(02CH)=00FH

REG(080H)=054H

C=1, N=Z=0

Example 2: RLFC REG, 0, 1

Before Instruction:

WREG(02CH)=0FH

REG(170H)=0EAH

C=N=Z=0

After Instruction:

WREG(02CH)=0D4H

REG(170H)=0EAH

C=N=1, Z=0

Example 3: RLFC REG, 1, 1

Before Instruction:

WREG(02CH)=00FH

REG(170H)=080H

C=N=Z=0

After Instruction:

WREG(02CH)=00FH

REG(170H)=000H

C=Z=1, N=0

© 2008-2019 HYCON Technology Corp

3.55 RRF

Rotate Righ F (no carry)

Syntax: RRF f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f<n>) → destination <n - 1 >,

(f<0>) → destination < 7 >

Status Affected: N, Z

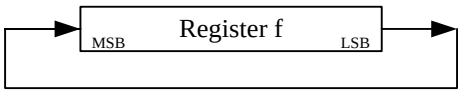
Description: 将 f 暂存内的值向右旋转。

若 d = 0，则运算后的结果放到 W 累加器中；

若 d = 1，则运算后的结果放到 f 缓存器中；

若 a = 0，则运算后的结果放到目前 RAM 地址中；

若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。



Words: 1

Cycles: 1

Example 1: RRF REG, 1, 0

Before Instruction:

WREG(02CH)=00FH

REG(080H)=0AAH

N=Z=0

After Instruction:

WREG(02CH)=00FH

REG(080H)=055H

N=Z=0

Example 2: RRF REG, 0, 1

Before Instruction:

WREG(02CH)=00FH

REG(17FH)=000H

N=Z=0

After Instruction:

WREG(02CH)=000H

REG(17FH)=000H

Z=1, N=0

Example 3: RRF REG, 0, 0

Before Instruction:

WREG(02CH)=00FH

REG(080H)=055H

N=Z=0

After Instruction:

WREG(02CH)=0AAH

REG(080H)=055H

N=1, Z=0

© 2008-2019 HYCON Technology Corp

3.56 RRFC

Rotate Right F through Carry

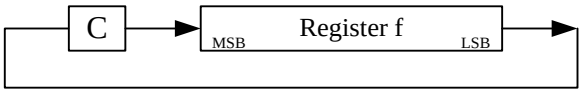
Syntax: RRFC f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f<n>) → destination <n-1 >,
(f<0>) → Status< C >,
Status< C > → destination < 7 >

Status Affected: C, N, Z

Description: 将 f 暂存内的值与进位旗标 C 一起向右旋转。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。



Words: 1

Cycles: 1

Example 1: RRFC REG, 1, 0

Before Instruction:
WREG(02CH)=00FH
REG(080H)=0AAH
C=N=Z=0

After Instruction:
WREG(02CH)=00FH
REG(080H)=055H
C=N=Z=0

Example 2: RRFC REG, 0, 1

Before Instruction:
WREG(02CH)=00FH
REG(17FH)=055H
C=1, N=Z=0

After Instruction:
WREG(02CH)=0AAH
REG(17FH)=055H
C=N=1, Z=0

Example 3: RRFC REG, 1, 1

Before Instruction:
WREG(02CH)=00FH
REG(17FH)=001H
C=N=Z=0

After Instruction:
WREG(02CH)=00FH
REG(17FH)=000H
C=Z=1, N=0

© 2008-2019 HYCON Technology Corp

3.57 SETF

SET F

Syntax: SETF f, a

Operands: 0 ≦ f ≦ 255 ; a € (0, 1)

Operation: 0FFH → f

Status Affected: None

Description: 将 f 暂存内的值全部设定为 1。

a = 0 or a =1 的设定须取决 f 缓存器位于 RAM 地址：

若 a = 0，则表示 f 缓存器存在于 080H 到 0FFH 所指定的 RAM 地址中(BSRCN=000H)。

若 a = 1，则表示 f 缓存器存在于 100H 到 17FH 所指定的 RAM 地址中(BSRCN=001H)。

Words: 1

Cycles: 1

Example 1: SETF REG, 0

Before Instruction:

WREG(02CH)=00FH

REG(080H)=0AAH

After Instruction:

WREG(02CH)=00FH

REG(080H)=0FFH

© 2008-2019 HYCON Technology Corp

3.58 SLP

enter SLeeP mode

Syntax: SLP

Operands: None

Operation: 1 → PD

Status Affected: Pstatus<PD>

Description: CPU 进入睡眠状态，震荡器停止动作。

Words: 1

Cycles: 1

Example 1:

```
SLP
NOP
```

Before Instruction:	After Instruction:
PD=0	PD=1

3.59 SUBC

SUBtract w from f with Carry

Syntax: SUBC f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f) - (W) - $\overline{(C)}$ → destination

Status Affected: C, DC, N, OV, Z

Description: f 缓存器的值减去 W 累加器与进位旗标 C 的反向值，并将结果放置 d。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: SUBC REG, 0, 0

Before Instruction:	**After Instruction:**
WREG=001H	WREG=000H
REG(080H)=001H	REG(080H)=001H
C=1, DC=N=OV=Z=0	C= DC=Z=1, N=OV= 0

Remark: C, DC 皆未借位，所以 C=DC=1，执行结果为 0，所以 Z=1。

Example 2: SUBF REG, 1, 1

Before Instruction:	**After Instruction:**
WREG=000H	WREG=000H
REG(17FH)=080H	REG(17FH)=07FH
C=DC=N=OV=Z=0	C=OV=1, DC= N=Z=0

Remark: C 未借位，所以 C=1；DC 被借位，所以 DC=0；
OV=1 条件判断标准：(负数) - (正数) = 正数 or (正数) - (负数) = 负数；
此范例符合(负数) - (正数) = 正数，所以 OV=1。

© 2008-2019 HYCON Technology Corp

3.60 SUBF

SUBtract w from F

Syntax: SUBF f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f) - (W) → destination

Status Affected: C, DC, N, OV, Z

Description: 将 f 缓存器的值减掉 W 累加器的值，并将结果放置 d。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: SUBF REG, 0, 0

Before Instruction:	After Instruction:
WREG=001H	WREG=000H
REG(080H)=001H	REG(080H)=001H
C=DC=N=OV=Z=0	C= DC=Z=1, N=OV= 0

Remark: C, DC 皆未借位，所以 C=DC=1，执行结果为 0，所以 Z=1。

Example 2: SUBF REG, 1, 1

Before Instruction:	After Instruction:
WREG=001H	WREG=001H
REG(17FH)=080H	REG(17FH)=07FH
C=DC=N=OV=Z=0	C=OV=1, DC= N=Z=0

Remark: C 未借位，所以 C=1；DC 被借位，所以 DC=0；
OV=1 条件判断标准：(负数) - (正数) = 正数 or (正数) - (负数) = 负数；
此范例符合(负数) - (正数) = 正数，所以 OV=1。

© 2008-2019 HYCON Technology Corp

3.61 SUBL

SUBtract w from Literal

Syntax: SUBL k

Operands: 0 ≦ k ≦ 255

Operation: K - (W) → W

Status Affected: C, DC, N, OV, Z

Description: 将常数 k 与 W 累加器的值相减并将结果放回 W 累加器中。

Words: 1

Cycles: 1

Example 1: SUBL 001H

Before Instruction:

WREG=001H

C=DC=N=OV=Z=0

After Instruction:

WREG=000H

C= DC=Z=1, N=OV= 0

Remark: C, DC 皆未借位，所以 C=DC=1，执行结果为 0，所以 Z=1。

Example 2: SUBL 080H

Before Instruction:

WREG=001H

C=DC=N=OV=Z=0

After Instruction:

WREG=07FH

C=OV=1, DC= N=Z=0

Remark: C 未借位，所以 C=1；DC 被借位，所以 DC=0；
OV=1 条件判断标准：(负数) - (正数) = 正数 or (正数) - (负数) = 负数；
此范例符合(负数) - (正数) = 正数，所以 OV=1。

Example 3: SUBL 07FH

Before Instruction:

WREG=0FFH

C=DC=N=OV=Z=0

After Instruction:

WREG=080H

DC=N=OV=1, C= Z=0

Remark: DC 未借位，所以 DC=1；C 被借位，所以 C=0；执行结果大于 127，所以 N=1
此范例符合(正数) - (负数) = 负数，所以 OV=1。

Example 4: SUBL 000H

Before Instruction:

WREG=001H

C=DC=N=OV=Z=0

After Instruction:

WREG=0FFH

N= 1, C=DC=OV=Z=0

Remark: C, DC 均被借位，所以 C=DC=0；执行结果大于 127，所以 N=1。

© 2008-2019 HYCON Technology Corp

3.62 SWPF

SWaP F

Syntax: SWPF f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (f<3:0>) → destination<7:4>

(f<7:4>) → destination<3:0>

Status Affected: None

Description: 将 f 缓存器内的高 4 位值与低 4 位值做交换。

若 d = 0，则运算后的结果放到 W 累加器中；

若 d = 1，则运算后的结果放到 f 缓存器中；

若 a = 0，则运算后的结果放到目前 RAM 地址中；

若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: SWPF REG, 1, 0

Before Instruction:	**After Instruction:**
WREG=001H	WREG=001H
REG(080H)=05AH	REG(080H)=0A5H

© 2008-2019 HYCON Technology Corp

3.63 TBLR

TaBLe Read

Syntax: TBLR (* , *+)

Operands: *, or *+

Operation: # If (TBLR *)

(Program Memory (TBLPTRH, TBLPTRL)) → TBLDH, TBLDL,

TBLPTR (TBLPTRH, TBLPTRL) do not change.

If (TBLR *+)

(Program Memory (TBLPTRH, TBLPTRL)) → TBLDH, TBLDL,

(TBLPTR) +1 ->TBLPTR.

Status Affected: None

Description: TBLR 是一个读取程序内存内容的指令，多用在查表指令使用，目前它提供以下 2 种用法：

- TBLR *

  以 TBLPTRH, TBLPTRL 缓存器数值为地址指针，读取对应的程序内存内容至 TBLD (TBLDH, TBLDL)缓存器中。

- TBLR *+

  以 TBLPTRH, TBLPTRL 缓存器数值为地址指针，读取对应的程序内存内容至 TBLD (TBLDH, TBLDL)缓存器中，然后将地址指针自动加 1。

Words: 1

Cycles: 2

Example 1: TBLR *

Before Instruction:

TBLDH, TBLDL= 0123H

At TBLPTR=0017FFH

Address(0017FFH) = data (5678H)

After Instruction:

TBLDH, TBLDL= 5678H

TBLPTR=0017FFH

Remark: 取出 TBLPTR 地址处的 2 bytes 数据放回 TBLD (TBLDH, TBLDL)，TBLPTR 指标内容不变。

Example 2: TBLR *+

Before Instruction:

TBLDH, TBLDL= 0123H

At TBLPTR=0017FFH

Address(0017FFH) = data (5678H)

After Instruction:

TBLDH, TBLDL= 5678H

TBLPTR=001800H

Remark: 取出 TBLPTR 地址处的 2 bytes 数据放回 TBLD (TBLDH, TBLDL)，TBLPTR 指标内容+1。

© 2008-2019 HYCON Technology Corp

3.64 TFSZ

Test F, Skip if Zero

Syntax: TFSZ f, a

Operands: 0 ≦ f ≦ 255 ; a € (0, 1)

Operation: skip if f = 0

Status Affected: None

Description: 假如 f 缓存器内的值等于 0 则跳过下一个指令 ;若 f 缓存器内的值不等于 0，则执行下一个指令。
a = 0 or a =1 的设定须取决 f 缓存器位于 RAM 地址：
若 a = 0，则表示 f 缓存器存在于 080H 到 0FFH 所指定的 RAM 地址中(BSRCN=000H)。
若 a = 1，则表示 f 缓存器存在于 100H 到 17FH 所指定的 RAM 地址中(BSRCN=001H)。

Words: 1

Cycles: 1(2)(3)

Example 1:

```
TFSZ    REG, 0
MVL     00FH
NOP
```

Before Instruction:

WREG(02CH)=005H

REG(080H)=000H

After Instruction:

WREG(02CH)=005H

REG(080H)=000H

Remark: f 缓存器的数值等于 0，所以跳过下一个指令。

Example 2:

```
TFSZ    REG, 1    (if BSRCN=001H)
MVL     00FH
NOP
```

Before Instruction:

WREG(02CH)=005H

REG(170H)=001H

After Instruction:

WREG(02CH)=00FH

REG(170H)=001H

Remark: f 缓存器数值不等于 0，所以程序继续往下执行。

© 2008-2019 HYCON Technology Corp

3.65XORF

eXclusive OR w with F

Syntax: XORF f, d, a

Operands: 0 ≦ f ≦ 255 ; d € (0, 1) ; a € (0, 1)

Operation: (W) XOR (f) → destination

Status Affected: N, Z

Description: 将常数 k 与 W 累加器的值做逻辑互斥或(XOR)运算，并将结果放回 d 中。
若 d = 0，则运算后的结果放到 W 累加器中；
若 d = 1，则运算后的结果放到 f 缓存器中；
若 a = 0，则运算后的结果放到目前 RAM 地址中；
若 a = 1，则运算后的结果放到 BSRCN 缓存器所指定的 RAM 地址中。

Words: 1

Cycles: 1

Example 1: XORF REG, 0, 0

Before Instruction:	After Instruction:
WREG(02CH)=0AAH	WREG(02CH)=0FFH
REG(080H)=055H	REG(080H)=055H
N=Z=0	N=1, Z=0

Remark: 执行结果大于 127，所以 N=1。 XOR: 两者同则结果为 0；两者不同则结果为 1。

Example 2: XORF REG, 1, 1

Before Instruction:	After Instruction:
WREG(02CH)=0FFH	WREG(02CH)=0FFH
REG(170H)=0FFH	REG(170H)=000H
N=Z=0	Z=1, N=0

Remark: 执行结果等于 0，所以 Z=1。 XOR: 两者同则结果为 0；两者不同则结果为 1。

Example 3: XORF REG, 0, 0

Before Instruction:	After Instruction:
WREG(02CH)=000H	WREG(02CH)=000H
REG(080H)=000H	REG(080H)=000H
N=Z=0	Z=1, N=0

Remark: 执行结果等于 0，所以 Z=1。 XOR: 两者同则结果为 0；两者不同则结果为 1。

© 2008-2019 HYCON Technology Corp

3.66 XORL

eXclusive OR Literal with w

Syntax: XORL k

Operands: 0 ≦ f ≦ 255

Operation: (W) XOR k → W

Status Affected: N, Z

Description: 将常数 k 与 W 累加器的值做逻辑互斥或(XOR)的运算，并将结果放回 W 累加器中。

Words: 1

Cycles: 1

Example 1: XORL 055H

Before Instruction:

WREG(02CH)=0AAH

N=Z=0

After Instruction:

WREG(02CH)=0FFH

N=1, Z=0

Remark: 执行结果大于 127，所以 N=1。
XOR: 两者同则结果为 0；两者不同则结果为 1。

Example 2: XORL 0FFH

Before Instruction:

WREG(02CH)=0FFH

N=Z=0

After Instruction:

WREG(02CH)=000H

Z=1, N=0

Remark: 执行结果等于 0，所以 Z=1。
XOR: 两者同则结果为 0；两者不同则结果为 1。

Example 3: XORL 000H

Before Instruction:

WREG(02CH)=000H

N=Z=0

After Instruction:

WREG(02CH)=000H

Z=1, N=0

Remark: 执行结果等于 0，所以 Z=1。
XOR: 两者同则结果为 0；两者不同则结果为 1。

© 2008-2019 HYCON Technology Corp

4 修订纪录

以下描述本文件差异较大的地方，而标点符号与字形的改变不在此描述范围。

版本	页次	变更摘要
V01	ALL	初版发行
V02	ALL	修正格式
V03	-	删除 DAW 指令
V04	ALL	支持 H08C、H08D 指令说明。並與繁體文件同步

© 2008-2019 HYCON Technology Corp