填空题 1分

设置

ARM指令集是多少位的？ [填空1]

正常使用填空题需3.0以上版本雨课堂

作答

填空题 1分

设置

```
MOV R1,1
MOV R2,2
ADD  R0,R1,R2,LSL #3
```

RO=? [填空1]

正常使用填空题需3.0以上版本雨课堂

作答

填空题 1分

设置

MVN R0,#0x0FF ；结果R0=? [填空1]

正常使用填空题需3.0以上版本雨课堂

作答

主观题 10分

设置

SWP R1,R2,[R0]

MOV R1,#0x1DDD

CLZ R2,R1 R2=?

BX R2 ARM OR THUMB-?

正常使用主观题需2.0以上版本雨课堂

作答

第四章 Thumb 指令集

4.1 Thumb 指令集概述

4.2 Thumb 指令详细介绍

4.1 Thumb指令集概述

ARM开发工具完全支持Thumb指令，应用程序可以灵活的将ARM和Thumb子程序混合编程以便在例程的基础上提高性能或代码密度。在编写Thumb指令时，先要用伪指令CODE16声明（ADS的编译环境下），而且在ARM指令中要使用BX指令跳转到Thumb指令，以切换处理器状态。

4.1.1 Thumb指令集编码

4.1.2 Thumb状态切换

4.1.3 编程模型

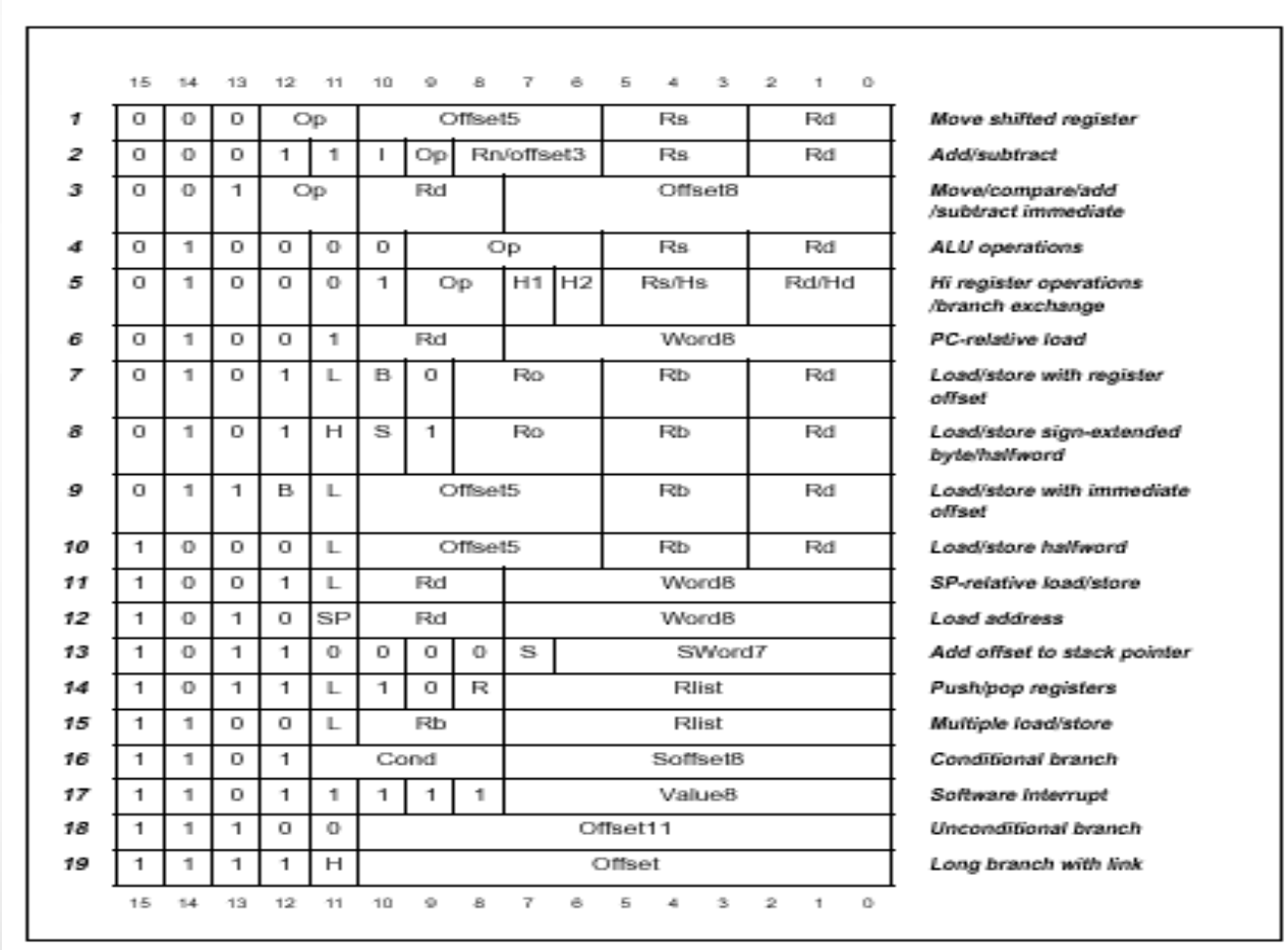
4.1.4 Thumb指令集特性

Thumb指令集特点：

- 采用16位二进制编码，而ARM指令是32位。
- Thumb是压缩指令，先动态解压缩，然后作为标准的ARM指令执行。
- 由CPSR的T位决定指令流。T置位，执行Thumb指令流，T清0，执行ARM指令流
- 由ARM模式进入Thumb模式时，是显式的进入；由Thumb进入ARM模式时，可以是隐式的进入，也可以是显式的进入。
- Thumb指令集没有协处理器指令、信号量指令、乘加指令、64位乘法指令以及访问CPSR和SPSR的指令，而且指令的第2操作数受到限制。
- 除了分支指令B有条件执行功能外，其他指令均无条件执行。
- 大多数Thumb数据处理指令采用2地址格式。

4.1.1Thumb指令集编码

Thumb指令集编码如下：



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	0	0	0	Op		Offset5					Rs			Rd			Move shifted register
2	0	0	0	1	1	I	Op	Rn/offset3			Rs			Rd			Add/subtract
3	0	0	1	Op		Rd		Offset8									Move/compare/add /subtract immediate
4	0	1	0	0	0	0	Op				Rs			Rd			ALU operations
5	0	1	0	0	0	1	Op		H1	H2	Rs/Hs			Rd/Hd			Hi register operations /branch exchange
6	0	1	0	0	1	Rd			Word8								PC-relative load
7	0	1	0	1	L	B	0	Ro			Rb			Rd			Load/store with register offset
8	0	1	0	1	H	S	1	Ro			Rb			Rd			Load/store sign-extended byte/halfword
9	0	1	1	B	L	Offset5					Rb			Rd			Load/store with immediate offset
10	1	0	0	0	L	Offset5					Rb			Rd			Load/store halfword
11	1	0	0	1	L	Rd			Word8								SP-relative load/store
12	1	0	1	0	SP	Rd			Word8								Load address
13	1	0	1	1	0	0	0	0	S	SWord7							Add offset to stack pointer
14	1	0	1	1	L	1	0	R	Rlist								Push/pop registers
15	1	1	0	0	L	Rb			Rlist								Multiple load/store
16	1	1	0	1	Cond				Soffset8								Conditional branch
17	1	1	0	1	1	1	1	1	Value8								Software interrupt
18	1	1	1	0	0	Offset11											Unconditional branch
19	1	1	1	1	H	Offset											Long branch with link
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

4.1.2Thumb状态切换

在任何时刻，CPSR的第5位（位T）决定了ARM微处理器执行的是ARM指令流还是Thumb指令流。当T置1，则认为是16位的Thumb指令流；

当T清0，则认为是32位的ARM指令流。

- 进入Thumb模式

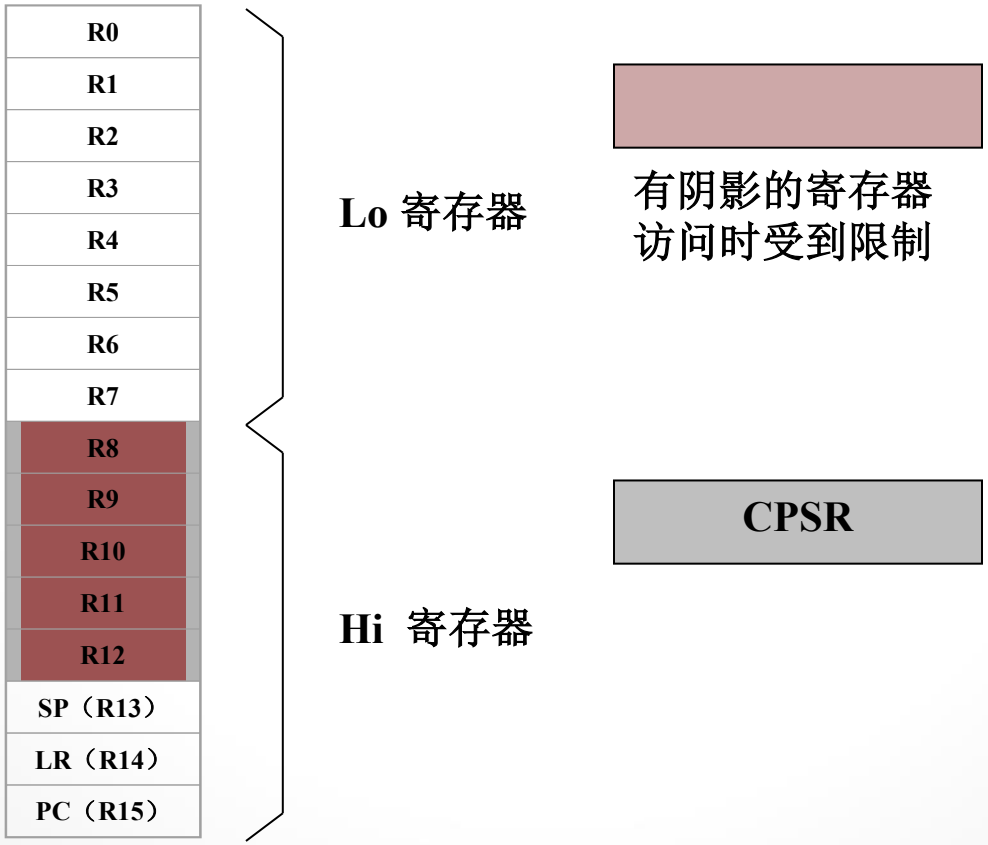
进入Thumb指令模式有两种方法：一种是执行一条交换转移指令BX，另一种方法是利用异常返回，也可以把微处理器从ARM模式转换为Thumb模式。

- 退出Thumb模式

退出Thumb指令模式也有两种方法：一种是执行Thumb指令中的交换转移BX指令可以显式的返回到ARM指令流。另一种是利用异常进入ARM指令流 。

4.1.3编程模型

Thumb指令集是ARM指令集的一个子集，并只能对限定的ARM寄存器进行操作。



4.1.4Thumb指令集特点

- Thumb指令继承了ARM指令集的许多特点

Thumb指令也是采用Load/Store结构，有数据处理、数据传送及流控制指令等。

- Thumb指令集丢弃了ARM指令集一些特性

大多数Thumb指令是无条件执行的（除了转移指令B），而所有ARM指令都是条件执行的。许多Thumb数据处理指令采用2地址格式，即目的寄存器与一个源寄存器相同，而大多数ARM数据处理指令采用的是3地址格式。

- Thumb异常时表现的一些特点

所有异常都会使微处理器返回到ARM模式状态，并在ARM的编程模式中处理。由于ARM微处理器字传送地址必须可被4整除（即字对准），半字传送地址必须可被2整除（即半字对准）。而Thumb指令是2个字节长，而不是4个字节，所以，由Thumb执行状态进入异常时其自然偏移与ARM不同。

4.2 Thumb指令集详细介绍

16位Thumb指令集是从32位ARM指令集提取指令格式的，每条Thumb指令有相同处理器模型所对应的32位ARM指令。

- 数据处理指令
- 转移指令
- Load/Store指令
- 异常中断指令

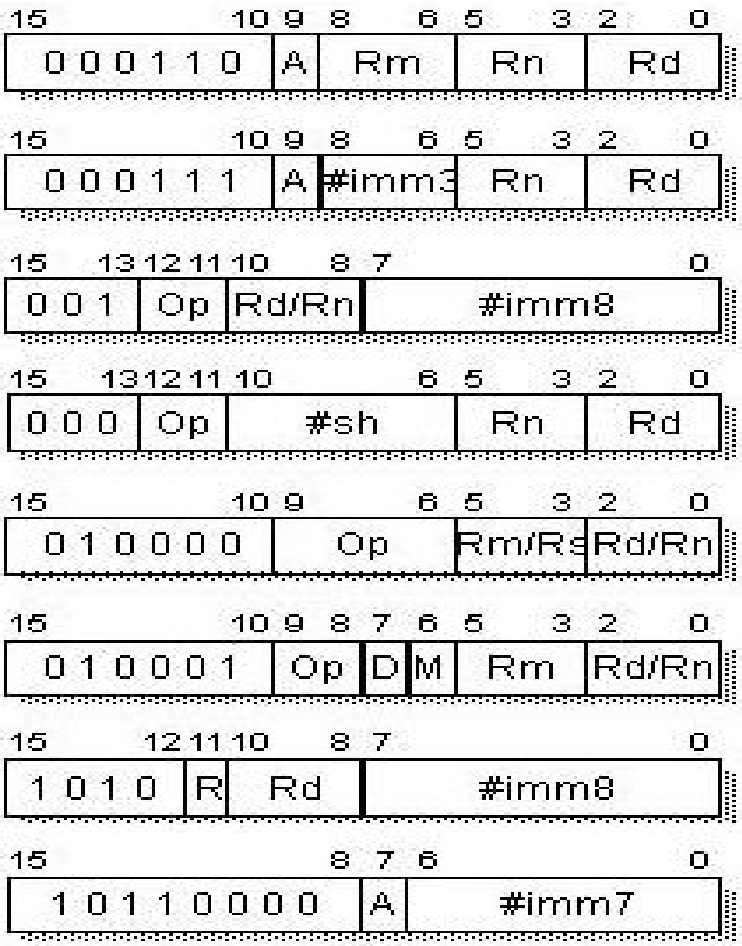
4.2.1 Thumb数据处理指令

Thumb数据处理指令包括一组高度优化且相当复杂的指令，范围涵盖编译器通常需要的大多数操作。ARM指令支持在单条指令中完成一个操作数的移位及一个ALU操作，但Thumb指令集将移位操作和ALU操作分离为不同的指令。本部分从以下几个方面介绍：

- 数据处理指令的二进制编码
- 数据处理指令的分类
- ARM指令与Thumb指令比较

数据处理指令

数据处理指令的二进制编码如下图：



(1)ADD|SUB Rd,Rn,Rm

(2)ADD|SUB Rd,Rn,#imm3

(3)<Op> Rd/Rn,#imm8

(4)LSL|LSR|ASR Rd,Rn,#shi

(5)<Op> Rd/Rn,Rm/Rs

(6)ADD|CMP|MOV Rd/Rn,Rm

(7)ADD Rd,SP|PC,#imm8

(8)ADD|SUB SP,SP,#imm7

数据处理指令

按照数据处理指令的功能，可以将其分为以下几类：

- 算术运算指令，它又分为以下几类：
  - ADD与SUB—低寄存器加法和减法
  - ADD—高或低寄存器
  - ADD与SUB—SP
  - ADD—PC或SP相对偏移
  - ADC，SBC和MUL
- 移位和循环移位操作（ASR，LSL，LSR和ROR）
- 比较指令（CMP和CMN）
- 传送和取负指令（MOV，MVN和NEG）
- 测试指令（TST）

算术运算指令

ADD与SUB——低寄存器加法和减法

句法：

op Rd,Rn,Rm

op Rd,Rn,#expr3

op Rd,#expr8

用法：

指令中Rd、Rn、Rm必须是低寄存器（R0~R7）。

指令更新NZCV标志。

例子：

ADD R3,R1,R5　　ADD R7,R2,R6

SUB R0,R4,#5　　SUB R4,R5,#201

ADD－－高或低寄存器

句法：

ADD Rd,Rm

用法：

Rd←Rd+Rm

Rd和Rm是低寄存器时，更新标志NZCV

例子：

ADD R12,R4

ADD R10,R11

ADD R2,R4

ADD与SUB－－SP

句法：

ADD SP,#expr

SUB SP,#expr

用法：

SP←SP+expr

不影响条件标志码

例子：

ADD SP,#312

ADD－－PC或SP偏移量

句法：

ADD Rd,Rp,#expr

Rp：SP或PC

用法：

Rd←Rp+expr

不影响条件标志码

例子：

ADD R6,SP,#64

ADD R2,PC,#980

ADC、SBC和MUL

（带进位位的加法、带进位位的加法、乘法）

句法：

op Rd,Rm

用法：

ADC：Rd←Rd+Rm+C

SBC：Rd ←Rd-Rm+C-1

MUL：Rd ←Rd×Rm

限制：Rd和Rm必须是低寄存器（R0~R7）

ADC和SBC影响NZCV

MUL影响NZ

AND、ORR、EOR和BIC（按位逻辑操作）

句法：

op Rd,Rm

用法：

AND逻辑“与”操作

ORR逻辑“或”操作

EOR逻辑“异或”操作

BIC：Rd AND NOT Rm

必须是低寄存器，影响NZ标志

ASR、LSL、LSR和ROR（逻辑和循环位移）

句法：

op Rd,Rs

op Rd,Rm,#expr

Rd、Rs、Rm必须是R0~R7范围内的寄存器

例子：

```
ASR R3,R5
LSR R0,R2,#6

ROR R2,R7,#2
LSL R7,R1
```

CMP和CMN（比较和比较负值）

句法：

CMP Rn,#expr

CMP Rn,Rm

CMN Rn,Rm

用法：

只更新条件码标志，不存放结果

CMP：Rn-expr（或Rm）

CMN：Rm+Rn

其中：CMP Rn,Rm指令允许使用高寄存器

例子：

CMP R2,#255　　CMP R9,#24

CMN R1,R5　　CMN R0,R10

MOV、MVN和NEG（传送、传送非和取负）

句法：

MOV Rd,#expr

MOV Rd,Rm

MVN Rd,Rm

NEG Rd,Rm

用法：

MOV：Rd←#expr（或Rm）

MVN：Rd ←NOT Rm

NEG： Rd ←Rm×（－1）

TST（测试）

句法：

TST Rn,Rm

用法：

Rm AND Rn，丢弃结果，更新条件码标志NZ

Rn、Rm必须在R0～R7范围内

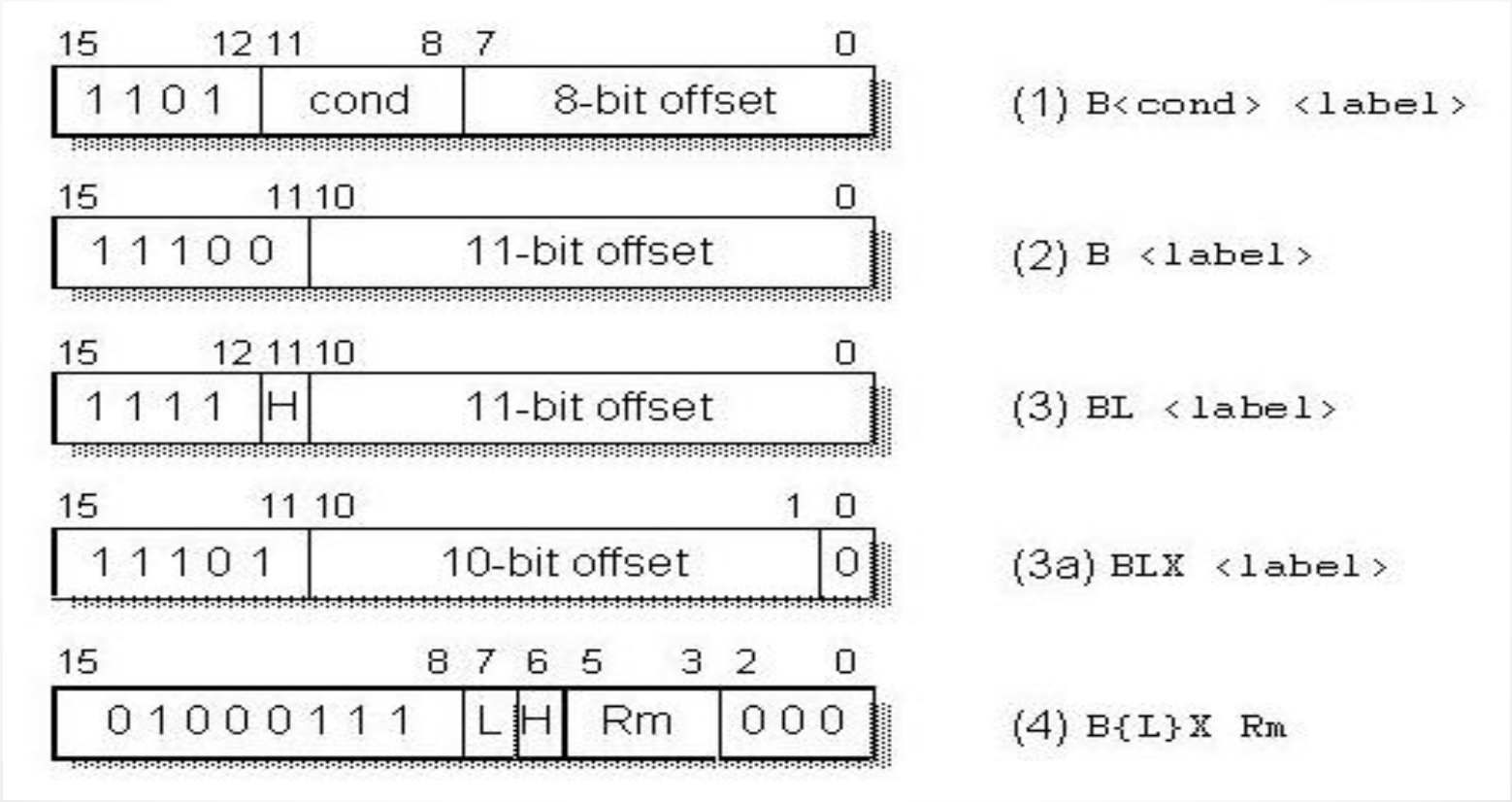
4.2.2Thumb转移指令

ARM指令有一个大的（24位）偏移域（offset field），这不可能在16位Thumb指令格式中表示。为此Thumb指令集有多种方法实现其子功能。

- 转移指令的二进制编码
- 转移指令的汇编格式
- 转移指令的分类

转移指令

转移指令的二进制编码如下：



转移指令

转移指令的汇编格式如下：

```
B     <cond> <label>  ；格式1   目标为Thumb代码
B     <label>         ；格式2   目标为Thumb代码
BL    <label>         ；格式3   目标为Thumb代码
BLX<label>            ；格式3a  目标为ARM代码
B{L}X  Rm             ；格式4   目标为ARM或Thumb代码
```

转移指令

转移指令分类如下：

- B—分支指令，Thumb指令集惟一可条件执行的指令。
- BL—带链接的长分支。
- BX—分支指令，并可选择地切换指令集。
- BLX—带链接分支，并可选地交换指令集。

4.2.3数据存取指令

Thumb的数据存取指令又可以分为：

单寄存器数据存取指令（LDR和STR）

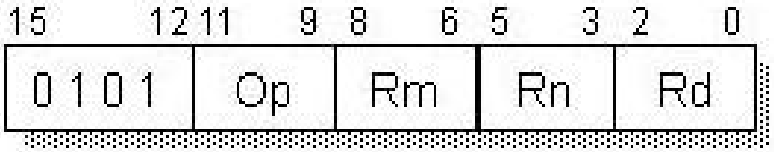
多寄存器数据存取指令（LDM和STM）

单寄存器数据存取指令（LDR和STR）

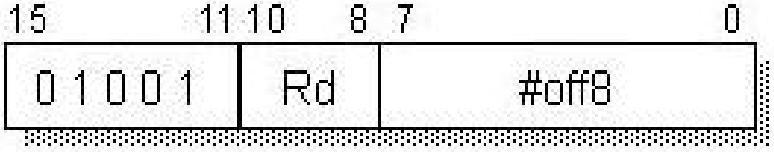
☞二进制编码如下：

(1) LDR|STR{B} Rd,[Rn,#off5]

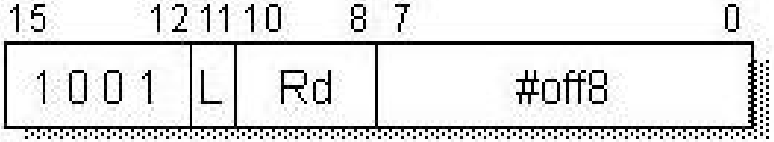
(2) LDRH|STRH Rd,[Rn,#off5]



(3) LDR|STR{S}{H|B} Rd,[Rn,Rm]



(4) LDR Rd,[PC,#off8]



(5) LDR|STR Rd,[SP,#off8]

单寄存器数据存取指令（LDR和STR）

汇编格式如下：

<op> Rd，[Rn，#<#off5>]　　；<op> = LDR|LDRB|STR|STRB

<op> Rd，[Rn，#<#off5>]　　；<op> = LDRH| STRH

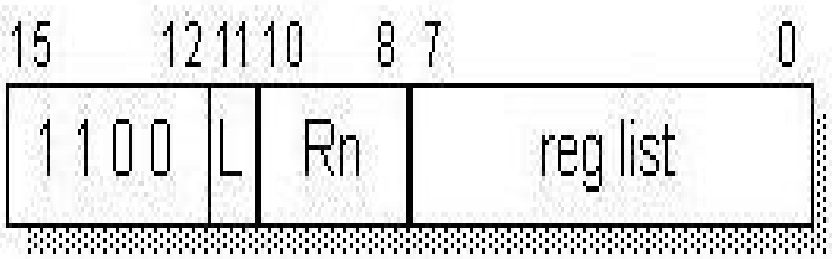
<op> Rd，[Rn，Rm]；<op> =LDR|LDRH|LDRSH|LDRB|LDRSB|STR|STRH|STRB

<op> Rd，[PC，#<#off8>]

<op> Rd，[SP，#<#off8>]；<op> = LDR| STR//该两条指令偏移量为8位

多寄存器数据存取指令

二进制编码如下：



(1) LDMIA|STMIA Rn!, {<reg list>}

(2) POP|PUSH {<reg list>{,R}}

多寄存器数据存取指令

汇编格式如下：

LDMIA　　Rn!，{<reg list>}

STMIA Rn!，{<reg list>}

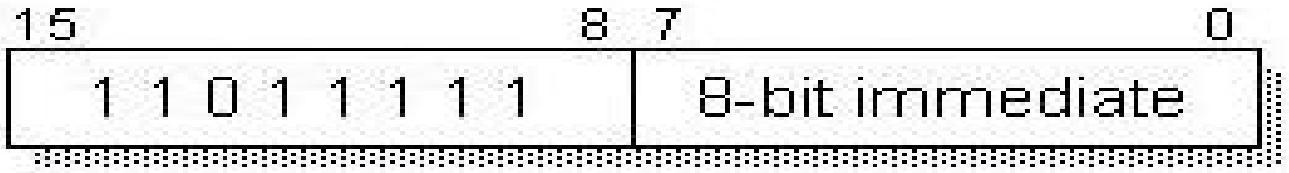
POP　　{<reg list>{，PC}}

PUSH　{<reg list>{，LR}}

4.2.4 异常中断指令

Thumb软件中断指令

Thumb软件中断指令的二进制编码如下：



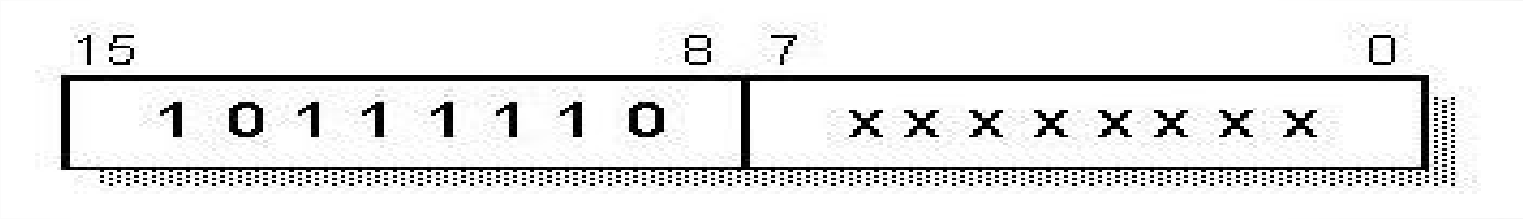
Thumb软件中断指令的汇编格式如下：

SWI <8位立即数>；<8位立即数>为数字表达式，其取值为0~255范围内的整数。

异常中断指令

Thumb断点指令

Thumb断点指令的二进制编码如下：



Thumb断点指令的汇编格式如下：

BKPT immed_8