ARM指令格式

- 了解ARM指令集编码
- 理解ARM指令的寻址方式
- 掌握ARM指令的使用

ARM指令集数据和指令类型

- ARM 采用的是32位架构.
- ARM 约定:
  - Byte（字节）：8 bits
  - Halfword（半字）：16 bits (2 byte)
  - Word （字）：　　32 bits (4 byte)

- 大部分ARM core 提供：
  - ARM 指令集（32-bit）
  - Thumb 指令集(T变种)（16-bit ）

3.1 ARM指令集概述

- ARM指令集是以32位二进制编码的方式给出的，大部分的指令编码中定义了第一操作数、第二操作数、目的操作数、条件标志影响位以及每条指令所对应的不同功能实现的二进制位。

31-28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
cond	0	0	1	opcode				s	Rn				Rd				operand2												数据处理/PSR状态转换
cond	0	0	0	0	0	0	A	s	Rd				Rn				Rs				1	D	0	1	Rm				乘法
cond	0	0	0	0	1	U	A	S	RdHi				RdLO				Rn				1	D	0	1	Rm				长乘
cond	0	0	0	1	0	B	0	0	Rn				Rd				0	0	0	0	0	0	0	0	Rm				数据交换
cond	0	0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	Rn				分支与交换
cond	0	0	0	P	U	0	W	L	Rn				Rd				0	0	0	0	1	S	H	1	Rm				半字存取寄存器偏移
cond	0	0	0	P	U	1	W	L	Rn				Rd				Offiset				1	S	H	1	Offiset				半字存取立即数偏移
cond	0	1	1	P	U	B	W	L	Rn				Rd				Offiset												单数据存取
cond	0	1	1																		1								未定义
cond	1	0	0	P	U	S	W	L	Rn				Register List																数据块存取
cond	1	0	1	L	Offiset																								分支
cond	1	1	0	P	U	N	W	L	Rn				CRd				CP#				Offiset								协处理器数据存取
cond	1	1	1	0	CP Opc				CRn				CRd				CP#				CP			0	GRm				协处理器数据操作
cond	1	1	1	0	CP Opc			L	CRn				Rd				CP#				CP			1	GRm				协处理器寄存器传送
cond	1	1	1	1	Ignored by processor																								软中断
31-28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

SOFTEEM 软帝

- 每条指令的多功能

ARM指令一个重要的特点是它所有的指令都带有条件，例如用户可以测试某个寄存器的值,但是,直到下次使用同一条件进行测试时，才能有条件的执行这些指令。ARM指令另一个重要的特点是具有灵活的第2操作数，既可以是立即数，也可以是逻辑运算数，使得ARM指令可以在读取数值的同时进行算术和移位操作。它可以在几种模式下操作，包括通过使用SWI（软件中断）指令从用户模式进入系统模式。

SOFTEEM 软帝

3.1.2 条件执行

- ARM指令集——条件码

ARM指令的基本格式如下：

```
<opcode> {<cond>} {S}    <Rd> ,<Rn>{,<operand2>}
```

使用条件码“cond”可以实现高效的逻辑操作，提高代码效率。

所有的ARM指令都可以条件执行，而Thumb指令只有B（跳转）指令具有条件执行功能。如果指令不标明条件代码，将默认为无条件（AL）执行。

SOFTEEM 软帝

• 指令条件码表

操作码	条件助记符	标志	含义
0000	EQ	Z=1	相等
0001	NE	Z=0	不相等
0010	CS/HS	C=1	无符号数大于或等于
0011	CC/LO	C=0	无符号数小于
0100	MI	N=1	负数
0101	PL	N=0	正数或零
0110	VS	V=1	溢出
0111	VC	V=0	没有溢出
1000	HI	C=1,Z=0	无符号数大于
1001	LS	C=0,Z=1	无符号数小于或等于
1010	GE	N=V	有符号数大于或等于
1011	LT	N!=V	有符号数小于
1100	GT	Z=0,N=V	有符号数大于
1101	LE	Z=1,N!=V	有符号数小于或等于
1110	AL	任何	无条件执行 (指令默认条件)
1111	NV	任何	从不执行(不要使用)

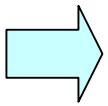
SOFTEEM 软帝

- ARM指令集——条件码

示例1：

C代码：

```
If(a > b)
        a++;
Else
        b++;
```



对应的汇编代码如下.其R0为a,R1为b

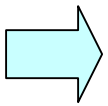
```
CMP    R0,R1        ;R0与R1比较
ADDHI  R0,R0,#1     ;若R0>R1，则R0=R0+1
ADDLS  R1,R1,#1     ;若R0≤R1，则R1=R1+1
```

- ARM指令集——条件码

示例2：

C代码：

```
If(a!=10)&&(b!=20)
a=a+b
```



对应的汇编代码如下.其R0为a,R1为b

```
CMP    R0,#10  ;比较R0是否为10
CMPNE  R1,#20  ;若R0不为10,则比较
               ;R1是否为20
ADDNE  R0,R0,R1;若R0不为10且R1
                ;不为20,则执行
                ;R0=R0+R1
```

SOFTEEM 软帝

3.1.3 指令格式

- ARM指令集——指令格式

ARM指令的基本格式如下：

```
<opcode> {<cond>} {S}    <Rd> ,<Rn>{,<operand2>}
```

其中<>号内的项是必须的，{}号内的项是可选的。各项的说明如下：

opcode：指令助记符；　　**Rd**：目标寄存器；

cond：执行条件；　　**Rn**：第1个操作数的寄存器；

S：是否影响CPSR寄存器的值；　　**operand2**：第2个操作数；

- ARM指令格式举例:

```
LDR   R0,[R1]      ;读取R1地址上的存储单元内容,执行条件AL;
BEQ   DATAEVEN     ;条件执行分支指令,执行条件EQ,即相等则跳转到DATAEVEN;
ADDS  R2,R1,#1     ;加法指令,R2<-R1+1,影响CPSR寄存器;
SUBNES R2,R1,#0x20  ;条件执行的减法运算,执行条件NE,R1-0x20->R2,影响CPSR寄存器;
```

SOFTEEM 软帝

- ARM指令集——第2个操作数

ARM指令的基本格式如下：

```
<opcode> {<cond>} {S}    <Rd> ,<Rn>{,<operand2>}
```

灵活的使用第2个操作数“operand2”能够提高代码效率。它有如下的形式：

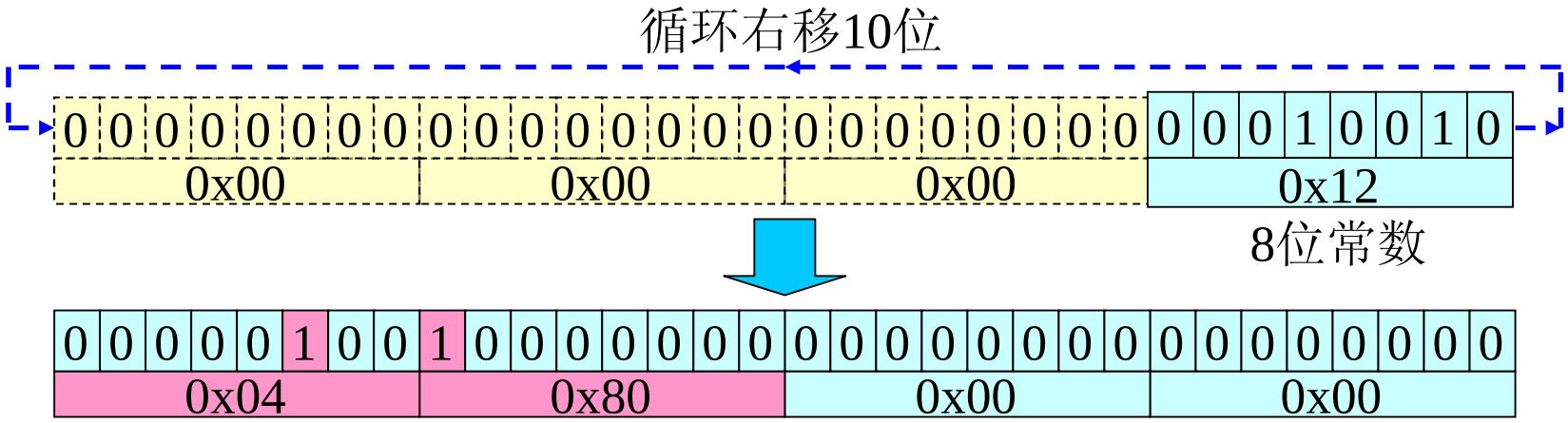
- #immed_8r——常数表达式；
- Rm——寄存器方式（操作数即为寄存器的数值）；
- Rm,shift——寄存器移位方式；

SOFTEEM 软帝

ARM指令集——第2个操作数

- #immed_8r——常数表达式

该常数必须对应8位位图(pattern)，即一个8位的常数通过循环右移偶数位得到,为合法常量。



SOFTEEM 软帝

- ARM指令集——第2个操作数

例如：

```
MOV    R0,#1              ; R0 = 1
AND    R1,R2,#0x0F        ; R2与0x0F,结果保存在R1中
LDR    R0,[R1],#-4        ;读取R1地址上的存储器单元内容,
                          ;且R1=R1-4
```

合法常量：

0xFF、0x104、0xFF0、0xFF000、0xFF000000、0xF000000F

非法常量：

0x101、0x102、0xFF1、0xFF04,0xFF003、0xFFFFFFFF

SOFTEEM 软帝

- ARM指令集——第2个操作数
  - Rm——寄存器方式

在寄存器方式下，操作数即为寄存器的数值。

例如：

```
SUB    R1,R1,R2          ;R1-- R2→R1
MOV    PC,R0             ;PC=R0,程序跳转到指定地址
LDR    R0，[R1]，-R2     ;读取R1地址上的存储器单元内容
                         ;并存入R0，且R1=R1-R2
```

SOFTEEM 软帝

◆ ARM指令集——第2个操作数

▪Rm,shift——寄存器移位方式

将寄存器的移位结果作为操作数，但Rm值保持不变，移位方法如下：

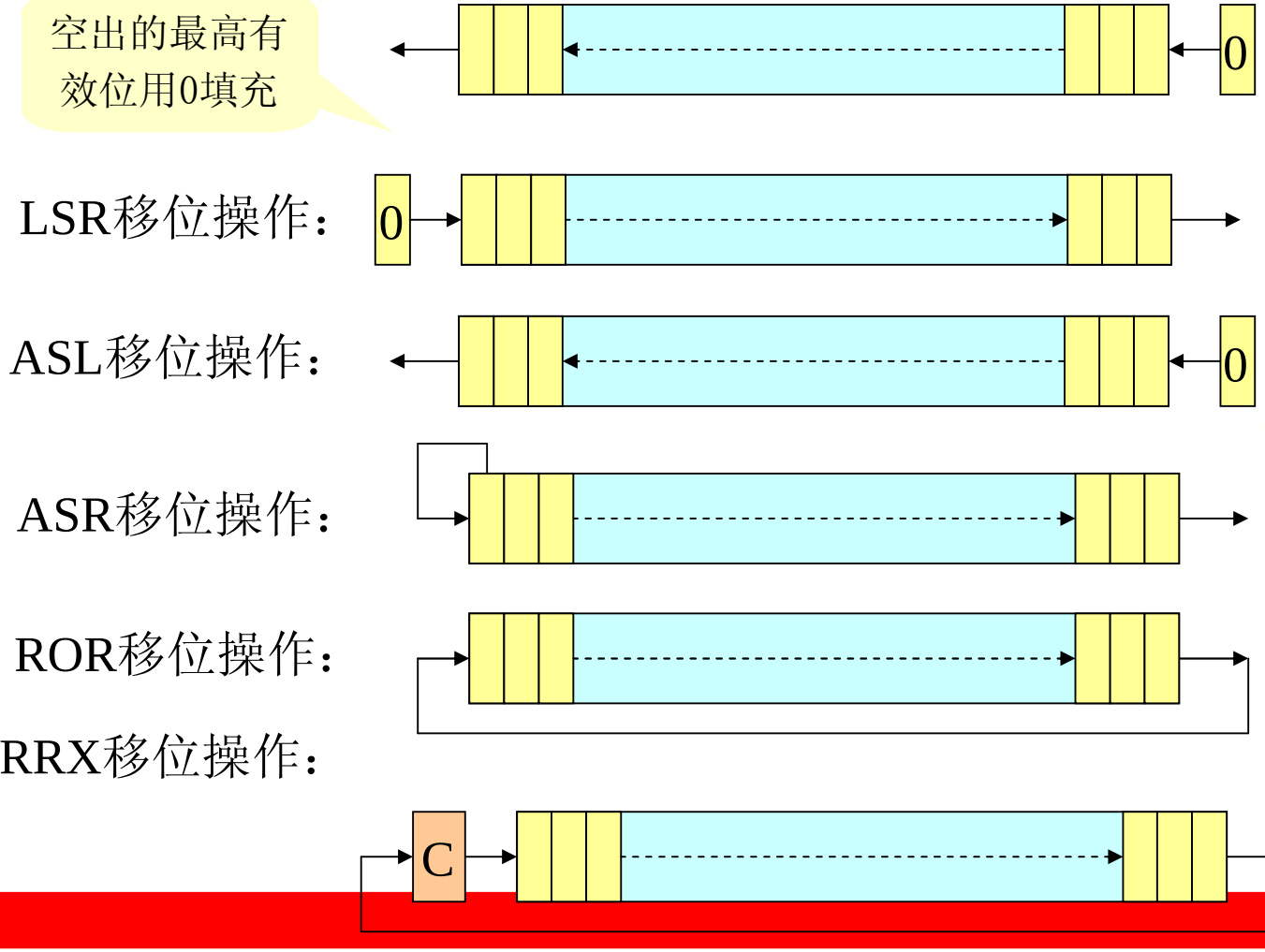
操作码	说明	操作码	说明
LSL #n	逻辑左移n位(1≤n≤31)	ASR #n	算术右移n位(1≤n≤32)
LSR #n	逻辑右移n位(1≤n≤32)	ROR #n	循环右移n位(1≤n≤31)
ASL #n	算术左移n位(与LSL同义)	RRX	带扩展的循环右移1位

◆ ARM指令集——第2个操作数

空出的最高有效位用0填充

空出的最低有效位用0填充

算术移位的对象是有符号数，移位过程中必须保持操作数的符号不变。即源操作数是正数，则空出的最高有效位用0填充，如果是负数，则用1填充。



将寄存器的内容循环右移1位，空位用原来的C标志位填充。

SOFTEEM 软帝

- ARM指令集——第2个操作数
  - Rm,shift——寄存器移位方式

例如：

```
ADD    R1,R1,R1,LSL #3     ;R1=R1+R1*8=9R1
SUB    R1,R1,R2,LSR R3     ;R1=R1-(R2/2^R3)
```

错例：

```
ADD    R3,R7,#1023         ;1023(0x3FF)不是一个循环移位的8位
                           ;位图
SUB    R11,R12,R3,LSL #32  ;#32超出了LSL范围
MOVS   R4,R4,RRX #3        ;RRX不需指定移位量， RRX总是移1位
```

作业任务

简答题

1. 哪些ARM指令可以有条件的执行?
2. 什么是ARM的寻址方式，ARM处理器支持的基本寻址方式有那几种?
3. 什么是堆栈寻址,堆栈寻址的四种模式?
4. ARM数据处理指令的基本原则有那些?
5. 完成ARM子程序调用要使用什么转移指令,基原理是什么?
6. 能不能通过状态寄存器与通用寄存器之间的传送指令将程序状态切换到Thumb状态,为什么?

Q & A