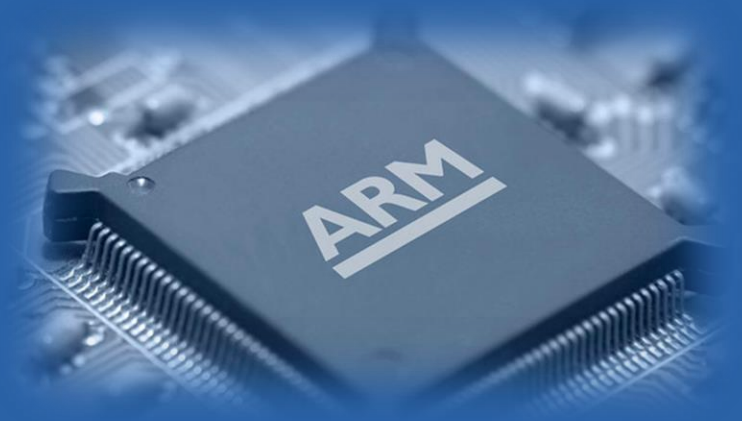
仪器科学与光电工程学院
School of Instrumentation Science and Opto-electronics Engineering

微机原理与接口技术

第三讲 ARM 概 述




```
bl   __isoc99_scanf
ldr r2, [sp, #12]
ldr r3, [sp, #8]
cmp r2, r3
bgt .L7
ldrlt   r1, [sp, #4]
ldrge   r1, [sp, #4]
sublt   r1, r1, #1
strlt   r1, [sp, #4]
```

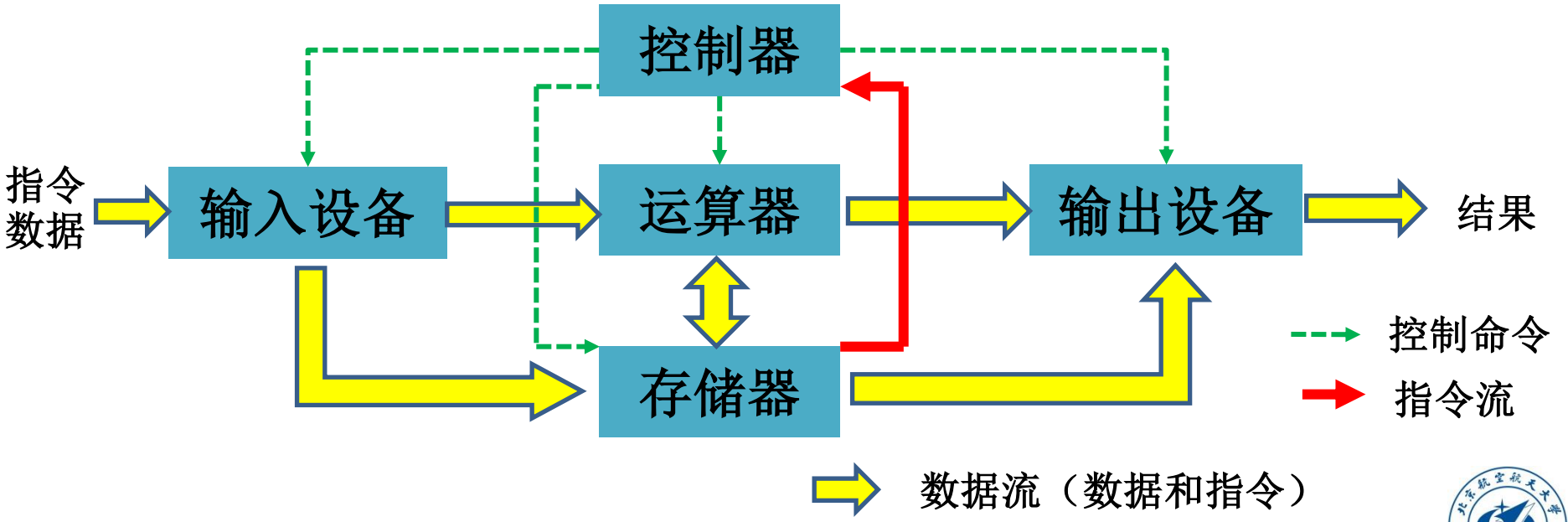
输入设备：用于指令、数据的输入

存储器：存放由指令构成的程序，输入的数据，运算的中间结果

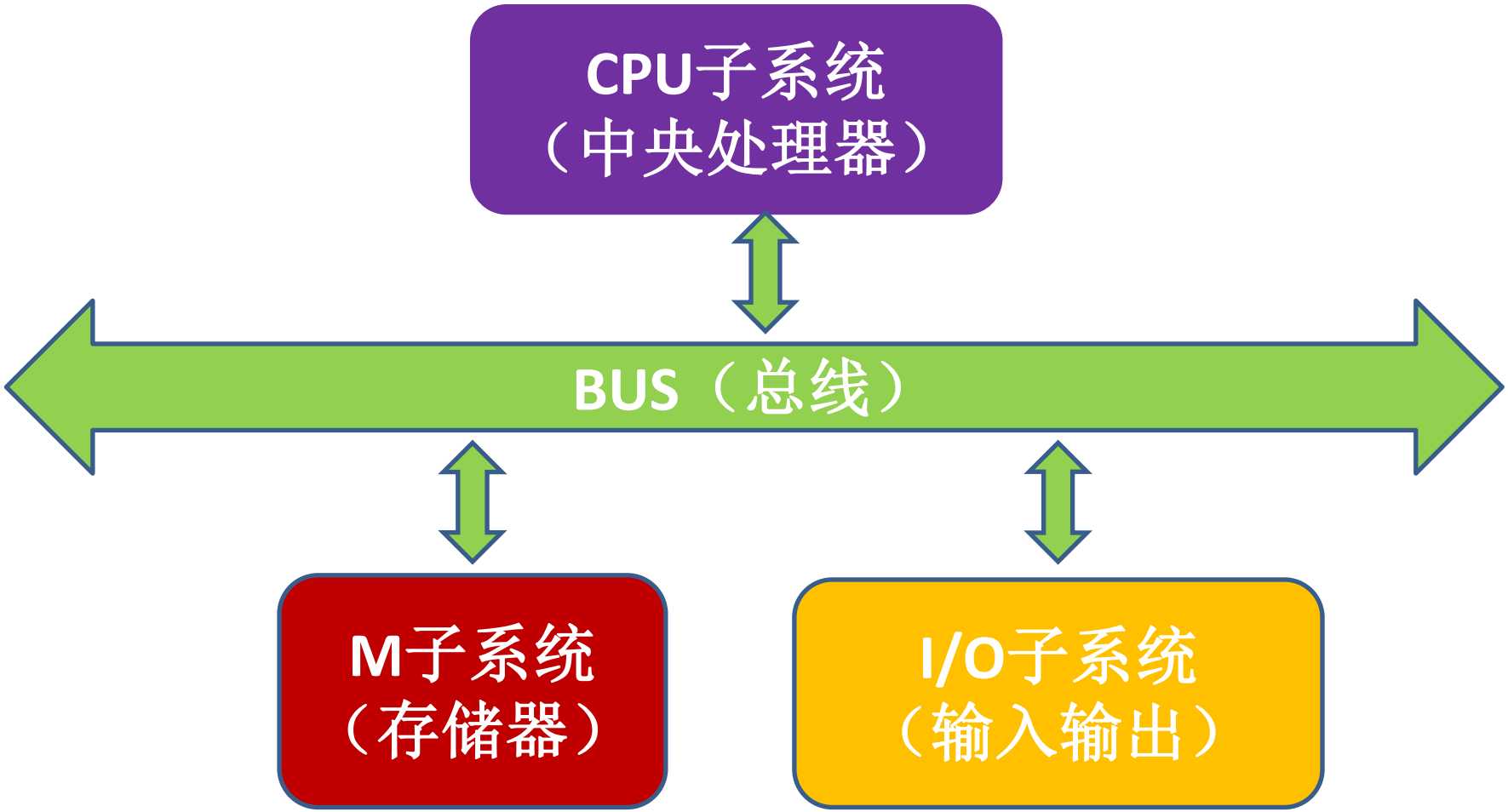
运算器：完成指令指定的各种算术运算和逻辑运算

控制器：根据指令产生相应的控制信号，控制其他4个部件

输出设备：用于运算结果的输出



现代冯·诺依曼计算机架构



1.4 微型计算机的基本组成结构

微型计算机指令系统：

按照指令的执行方式和指令集的复杂程度来划分可以分为两类：

(1)复杂指令集（Complex Instruction Set Computer , CISC）

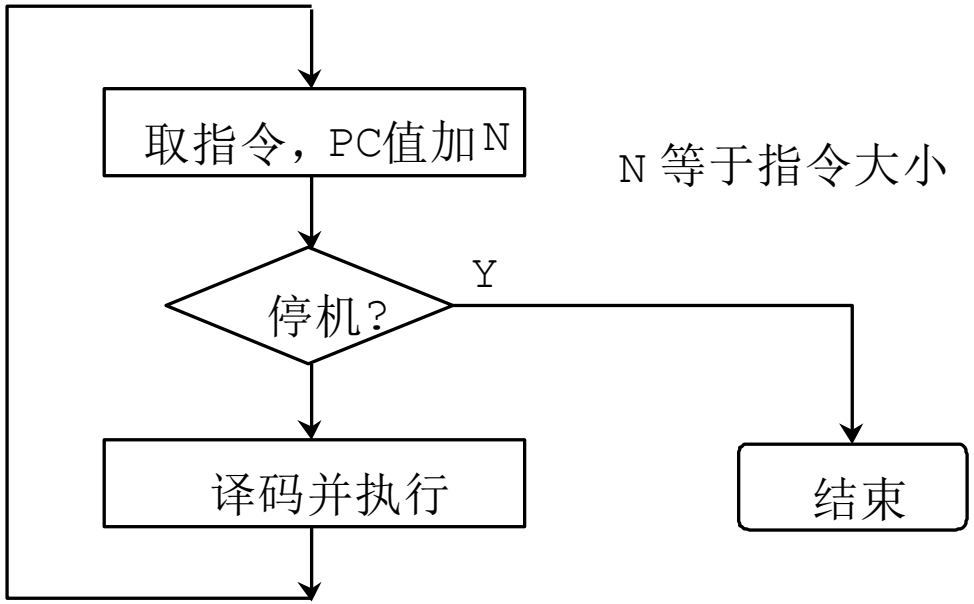
(2) 精简指令集（Reduce Instruction Set Computer , RISC）

CISC和RISC的指令集特征对比

	CISC	RISC
指令集数目	一般大于250条	常小于100
指令长度	不定（3~6字节）	基本固定（2、4字节）
寻址方式	多样、复杂	少，简单
存储访问指令	比较多	只有load/store指令
指令使用频率	差异大	相差不大
指令执行时间	差异大	基本一个周期完成
程序代码长度	较短	较长

1.5 微型计算机的基本工作原理

- 存储程序工作原理



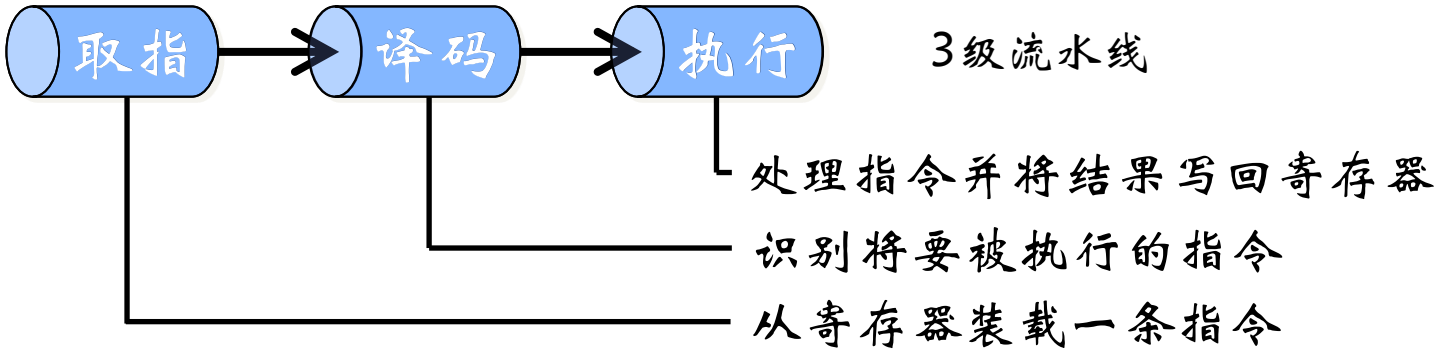
存储程序工作原理：先存储，再执行。即先把程序和数据送到具有记忆功能的存储器中保存，然后将程序第一条指令的地址送到程序计数器（PC），控制器依次取出存储器中的指令，结合相应的数据，执行每条指令，直至结束。

1.5 微型计算机的基本工作原理

- **流水线**

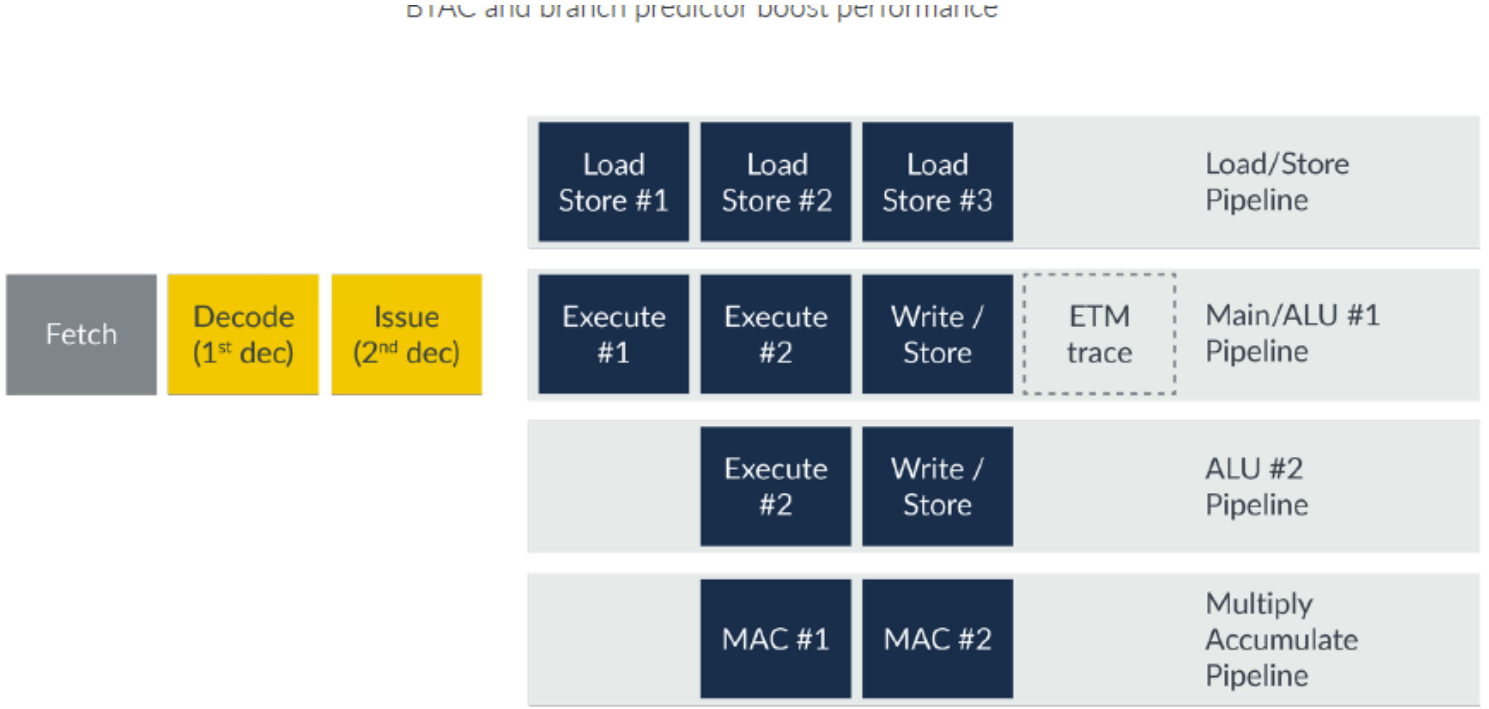
ARM处理器使用流水线来增加处理器指令流的速度，这样可使几个操作同时进行，并使处理和存储器系统连续操作，能提供XX MIPS/MHz的指令执行速度。

正常操作过程中，在执行一条指令的同时对下一条(第二条)指令进行译码，并将第三条指令从存储器中取出。



1.5 微型计算机的基本工作原理

- Cortex-M7的6级超标量流水线



信息的表示

1. 计算机中数的表示及运算

1.1 无符号整数的表示
1.2 有符号整数的表示
1.3 符号位扩展
1.4 整数的运算
1.5 小数的表示
1.6 字符的表示
1.7 其他数据

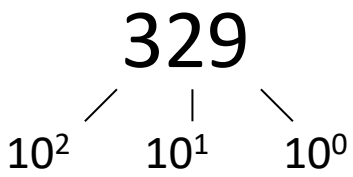
2. 存储器中的字

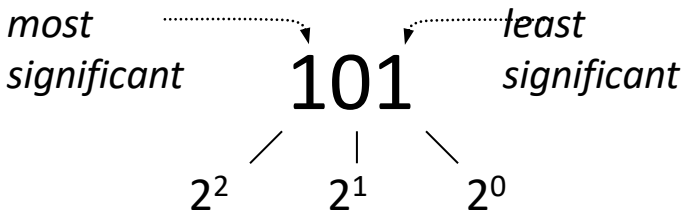
1.计算机中整数的表示及运算

1.1 无符号整数

日常生活中：十进制，十二进制，六十进制等。

计算机中：　采用二进制数表示。





3x100 + 2x10 + 9x1 = 329

1x4 + 0x2 + 1x1 = 5

$Y = \text{“abc”} = a \cdot 2^2 + b \cdot 2^1 + c \cdot 2^0$

（这里的a, b, c的值只能是0或1）

表示范围:

$0 \le Y \le 2^W - 1$

（*W = number of bits*）

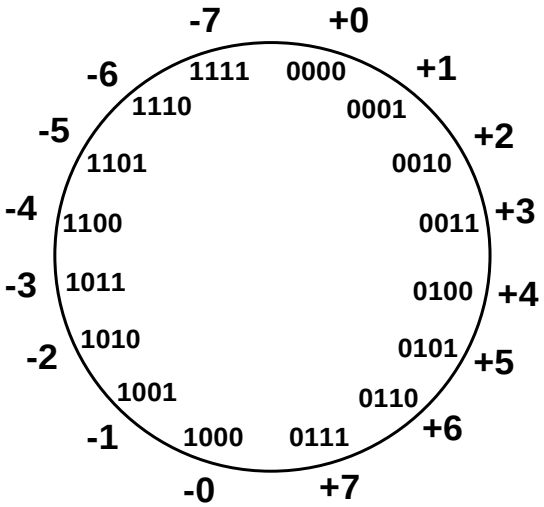
1.2 有符号整数的表示

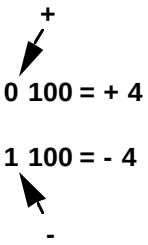
- 有符号整数的表示

  最高位表示符号位。

$$Y = \text{“abc”} = (-1)^a (b \cdot 2^1 + c \cdot 2^0)$$

Range is: $-2^{W-1} + 1 \le i \le 2^{W-1} - 1$





Problems:

- How do we do addition/subtraction?
  - e.g., try 2 + (-3)
- We have two numbers for zero (+/-)!

数的转换

1、二进制→十进制

展开求和

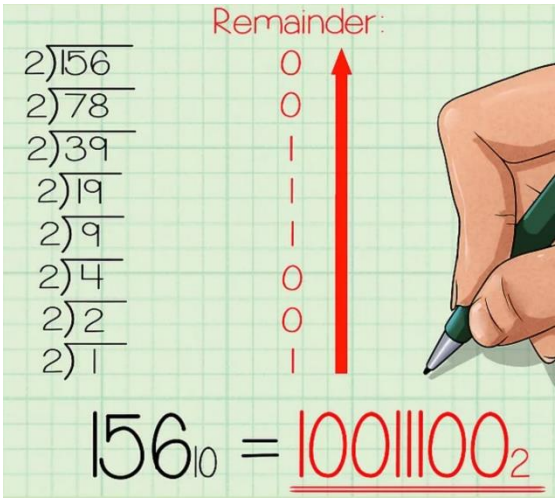
例子：4位二进制 $A = a_3a_2a_1a_0$ 对应:

$$a_3.2^3 + a_2.2^2 + a_1.2^1 + a_0.2^0 = a_3.8 + a_2.4 + a_1.2 + a_0.1$$

$$(a_i = 0 \text{ 、 } 1)$$

2、十进制→二进制

余数短除法除以二



十六进制表示

- Base 16 (hexadecimal)
  - More a convenience for us humans than a true computer data type
  - 0 to 9 represented as such
  - 10, 11, 12, 13, 14, 15 represented by A, B, C, D, E, F
  - $16 = 2^4$: i.e. every hexadecimal digit can be represented by a 4-bit binary (unsigned) and vice-versa.

Example

$$16AB_{16} = 0x16AB$$
$$= 1x16^3 + 6x16^2 + 10x16 + 11$$
$$= 5803_{10} = \#5803$$
$$= 0001\ 0110\ 1010\ 1011_2$$

典型长度整数的范围

数	字长w			
	8	16	32	64
$UMax_w$	0xFF 255	0xFFFF 65 535	0xFFFFFFFF 4 294 967 295	0xFFFFFFFFFFFFFFFF 18 446 744 073 709 551 615
$TMin_w$	0x80 −128	0x8000 −32 768	0x80000000 −2 147 483 648	0x8000000000000000 −9 223 372 036 854 775 808
$TMax_w$	0x7F 127	0x7FFF 32 767	0x7FFFFFFF 2 147 483 647	0x7FFFFFFFFFFFFFFF 9 223 372 036 854 775 807
−1	0xFF	0xFFFF	0xFFFFFFFF	0xFFFFFFFFFFFFFFFF
0	0x00	0x0000	0x00000000	0x0000000000000000

图 2-14　重要的数字。图中给出了数值和十六进制表示

单选题 1分

设置

-200，用16位二进制补码表示为（）

A -200

B 1000 0000 1100 1000

C 1111 1111 0011 1000

D 1011 1000

E 以上都不对

提交

1.3 符号位扩展

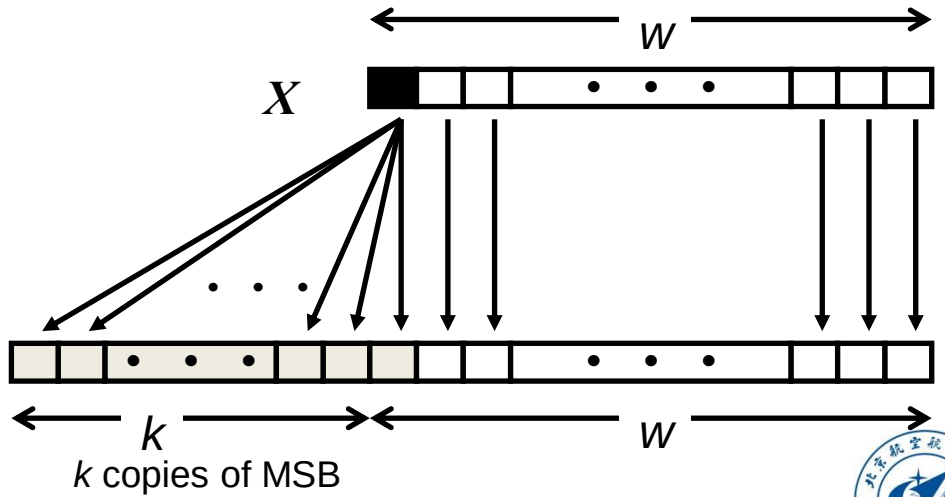
两个数相加，要求两个数表示的位数必须相等，不等时需要进行“符号位扩展”（Sign Extension-SEXT）

◆ 目标：

- 将w位的有符号整数x，扩展为（$w+k$）位同样值的有符号整数。

◆ 方法：

- 低位对其，高位复制k份符号位。



1.3 符号位扩展

```
short int x =  15213;
int       ix = (int) x;
short int y = -15213;
int       iy = (int) y;
```

	Decimal	Hex	Binary
x	15213	3B 6D	00111011 01101101
ix	15213	00 00 3B 6D	00000000 00000000 00111011 01101101
y	-15213	C4 93	11000100 10010011
iy	-15213	FF FF C4 93	11111111 11111111 11000100 10010011

1.4 数的运算

整数运算：算术运算和逻辑运算

- 算术运算：加、减，（乘、除暂不考虑）
- 逻辑运算：与、或、非等。

运算前提（假设）：

- 运算的两个数具有相同的位宽
- 忽略进位（最高位）

1.4.1 无符号整数算术运算

2^2	2^1	2^0	Val(dec)
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

- 无符号数加法和十进制一样

```
 10010
+ 1001
 11011
```

carry

```
 10010
+ 1011
 11101
```

```
  1111
+    1
 10000
```

```
 10111
+  111
 11110
```

- 无符号数减法

1.4.2 有符号数的算数运算

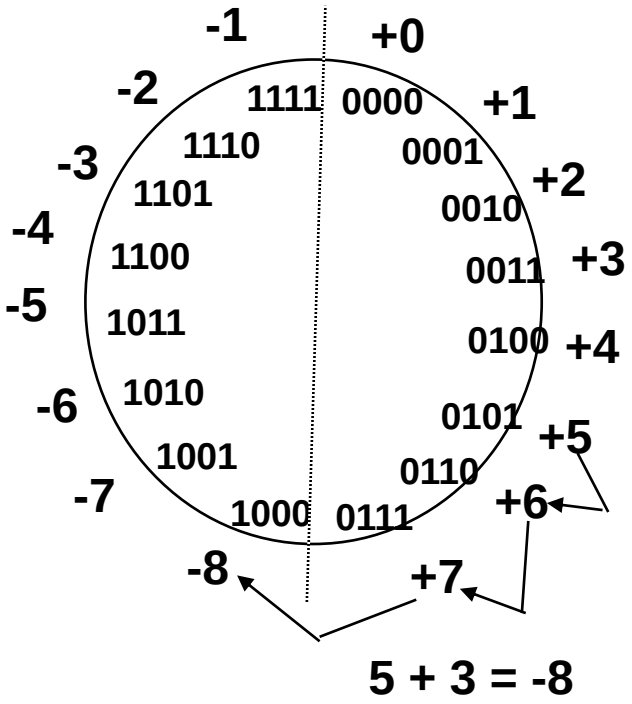
4	0100	-4	1100
+ 3	0011	+ (-3)	1101
7	0111	-7	11001
4	0100	-4	1100
+(- 3)	1101	+ 3	0011
1	10001	-1	1111

溢出：对于两个同符号数相加得到相异的结果，则表示计算发生了溢出。

注意：ALU只对两个二进制数进行加法运算，不考虑其他因素（如正数、负数），也是忽略最高位进位的主要原因。最高位进位通常用来控制。

1.4.3 算术运算过程中的溢出

- 在二进制补码整数运算过程中：
  - 符号位相异的两个数相加，永远不会溢出
  - 同符号数运算，结果符号相反，则溢出
- 无符号数运算
  - 高位进位即代表溢出



1.4.3 算术运算过程中的溢出

5	0 1 0 1
3	0 0 1 1
-8	1 0 0 0

Overflow

-7	1 0 0 1
-2	1 1 1 0
7	1 0 1 1 1

Overflow

5	0 1 0 1
2	0 0 1 0
7	0 1 1 1

No overflow

-3	1 1 0 1
-5	1 0 1 1
-8	1 1 0 0 0

No overflow

1.4.4 逻辑运算

这里讲的逻辑运算是按位逻辑运算，运算结果TRUE 或 FALSE。

位于位之间的与、或和非运算真值表：

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	NOT A
0	1
1	0

1.4.4 逻辑运算

- **XOR and XNOR**

XOR

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

XNOR

A	B	(A⊕B)’
0	0	1
0	1	0
1	0	0
1	1	1

1.4.4 逻辑运算

例：

- AND
  - useful for clearing bits
    - AND with zero = 0
    - AND with one = no change

```
  11000101
AND 00001111
  ________
  00000101
```

- OR
  - useful for setting bits
    - OR with zero = no change
    - OR with one = 1

```
  11000101
OR 00001111
  ________
  11001111
```

- NOT
  - unary operation -- one argument
  - flips every bit

```
NOT 11000101
  ________
  00111010
```

24-Bit, Wide Bandwidth Analog-to-Digital Converter

Table 9. Ideal Output Code versus Input Signal

INPUT SIGNAL V_{IN} (AINP − AINN)	IDEAL OUTPUT CODE(1)
$\geq +V_{REF}$	7FFFFFh
$\frac{+V_{REF}}{2^{23}-1}$	000001h
0	000000h
$\frac{-V_{REF}}{2^{23}-1}$	FFFFFFh
$\leq -V_{REF}\left(\frac{2^{23}}{2^{23}-1}\right)$	800000h

参考电压Vref = 2.5V

- 转换结果0x500000，输入电压？
- 转换结果0x900000，输入电压？

1.5 小数的表示法

采用科学记数法表示小数：$F \times 2^E$。因为科学记数法中小数点位置是变化的，所以称为浮点（float），表示的数称为浮点（float-Point）数。

这样的数需要表示：F，E和符号位。

IEEE 754 Floating-Point Standard (32-bits):

$$N = -1^S \times 1.\text{fraction} \times 2^{\text{exponent}-127},\ 1 \le \text{exponent} \le 254$$ 规约

$$N = -1^S \times 0.\text{fraction} \times 2^{-126},\ \text{exponent} = 0$$ 非规约

1.5 小数的表示法

浮点数例子：

IEEE单精度浮点数：

- 符号：1 – 表示负数.
- 指数：01111110 = 126 (decimal).
- 尾数：0.10000000000… = 0.5 (decimal).

则： Value = $-1.5 \times 2^{(126-127)} = -1.5 \times 2^{-1} = -0.75$.

1.5 小数的表示法

- 浮点数几个惯用表示:
  - Infinity (+ and -):
    $2^{e}-2$
    exponent = 255 and fraction = 0
  - NaN (not a number):
    - exponent = 255 and fraction ≠ 0
  - Zero (0): exponent = 0 and fraction = 0

    (note: exponent = 0 => fraction is *de-normalized*)

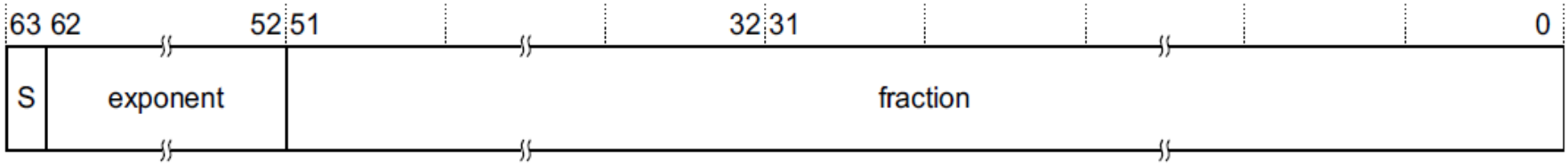
形式	指数	小数部分
零	0	0
非规约形式	0	非0
规约形式	1到2^e-2	任意
无穷	2^e-1	0
NaN	2^e-1	非零

双精度浮点数表示

The double-precision floating-point format used by the Floating-point Extension is as defined by the IEEE 754 standard.

This description includes Floating-point Extension-specific details that are left open by the standard. It is only intended as an introduction to the formats and to the values they can contain. For full details, especially of the handling of infinities, NaNs and signed zeros, see the IEEE 754 standard.

A double-precision value is a 64-bit doubleword, with the format:



1.6 字符的表示

字符：通常用ASCII（美国信息交互标准代码）表示，即：用一个字节（8位）来表示字符。存储非常方便，且通常ASCII编码占据低7位，高位通常为0。包含了可见字符和特殊意义字符（ESC,DEL等）

00	nul	10	dle	20	sp	30	0	40	@	50	P	60	`	70	p
01	soh	11	dc1	21	!	31	1	41	A	51	Q	61	a	71	q
02	stx	12	dc2	22	"	32	2	42	B	52	R	62	b	72	r
03	etx	13	dc3	23	#	33	3	43	C	53	S	63	c	73	s
04	eot	14	dc4	24	$	34	4	44	D	54	T	64	d	74	t
05	enq	15	nak	25	%	35	5	45	E	55	U	65	e	75	u
06	ack	16	syn	26	&	36	6	46	F	56	V	66	f	76	v
07	bel	17	etb	27	'	37	7	47	G	57	W	67	g	77	w
08	bs	18	can	28	(	38	8	48	H	58	X	68	h	78	x
09	ht	19	em	29	)	39	9	49	I	59	Y	69	i	79	y
0a	nl	1a	sub	2a	*	3a	:	4a	J	5a	Z	6a	j	7a	z
0b	vt	1b	esc	2b	+	3b	;	4b	K	5b	[	6b	k	7b	{
0c	np	1c	fs	2c	,	3c	<	4c	L	5c	\	6c	l	7c	|
0d	cr	1d	gs	2d	-	3d	=	4d	M	5d	]	6d	m	7d	}
0e	so	1e	rs	2e	.	3e	>	4e	N	5e	^	6e	n	7e	~
0f	si	1f	us	2f	/	3f	?	4f	O	5f	_	6f	o	7f	del

1.7 其他数据

字符串

一系列字符组成，以NULL（0）结束。

How to stop pandemics?

48 6F 77 20 74 6F 20 73 74 6F 70 20
61 6E 64 65 6D 69 63 73 3F 00

中国USA

D6 D0 B9 FA 55 53 41 00

1.7 其他数据

图片

- 象素阵列
  - 单色图片：1位(1/0 = black/white)
  - 彩色：: red, green, blue (RGB) components (e.g., 8 bits each)

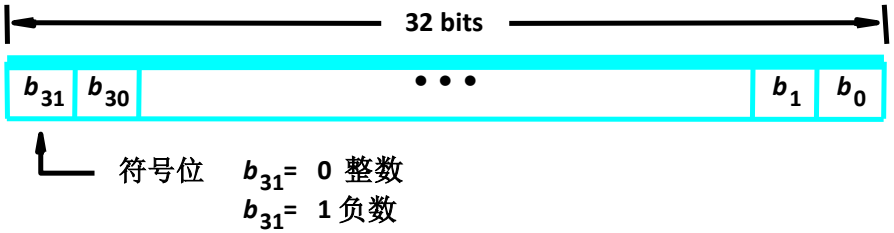
声

一系列定点数表示

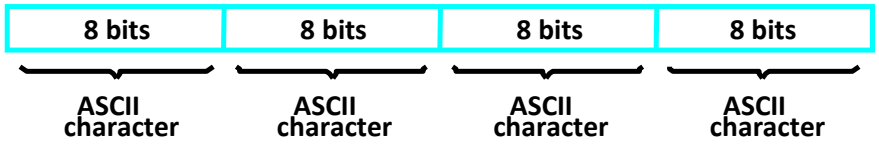
AD采集结果？

2. 存储器中的字

- 存储器是由大量存储单元（cell）组成
- 一般按固定长度的位（n位）进行组合处理
- 固定大小称为字（Word），长度n称为字长
- 主流计算机字长范围从16位~64位不等
- 对于32位字长的计算机：



(a) A signed integer



(b) Four characters

2.1 存储器中字节寻址

对存储器中信息进行访问时，需要知道每一个信息的地址。

一个k位地址的存储器，具有2^k个地址空间（0～ 2^k-1）。例如：

- 24位存储器，地址空间就是2^{24}，16777216个存储地址空间，即16M
- 32位存储器：2^{32} = 4G (1G=2^{30})

常用地址分配是以字节为单位划分，这样的存储器称为字节可寻址存储器（byte-addressable memory）。

对于32位字长的计算机，连续字被分配的地址是0，4，8，....（每一个字包含4个字节）。

- 地址位于A的字由地址为A、A＋1、A＋2和A＋3的字节组成；
- 地址位于A的半字由地址为A和A＋1的字节组成；
- 地址位于A＋2的半字由地址为A＋2和A＋3的字节组成；
- 地址位于A的字由地址为A和A＋2的半字组成。

2.2 大小端分配

- Big-Endian就是高位字节排放在内存的低地址端，低位字节排放在内存的高地址端。

  低地址存放高位

- Little-Endian就是低位字节排放在内存的低地址端，高位字节排放在内存的高地址端。

  低地址存放低位

- 比特序：

  字节比特序：LSB (Least Significant Bit)，
  MSB (Most Significant Bit)

  网络比特序

3.3.1 Big endian format

In Big Endian format, the most significant byte of a word is stored at the lowest numbered byte and the least significant byte at the highest numbered byte. Byte 0 of the memory system is therefore connected to data lines 31 through 24.

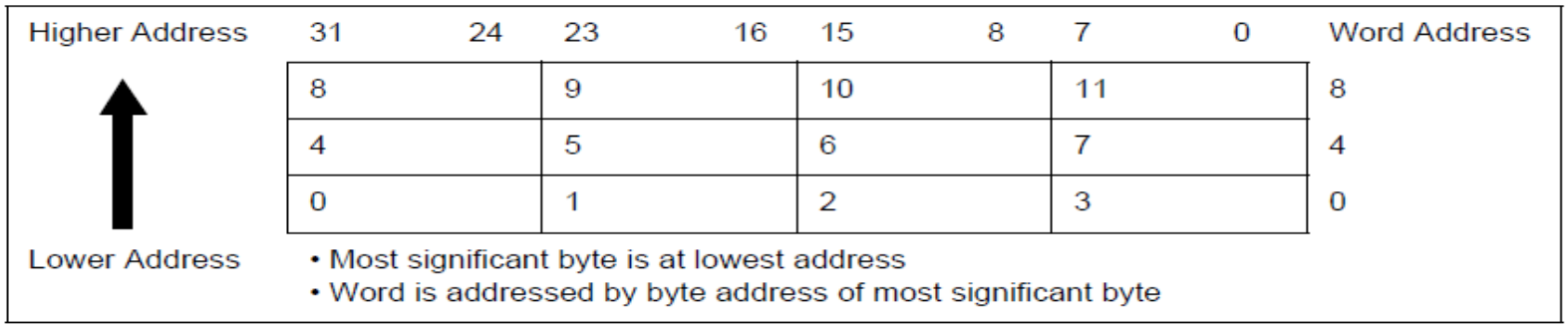


Figure 3-1: Big endian addresses of bytes within words

3.3.2 Little endian format

In Little Endian format, the lowest numbered byte in a word is considered the word's least significant byte, and the highest numbered byte the most significant. Byte 0 of the memory system is therefore connected to data lines 7 through 0.

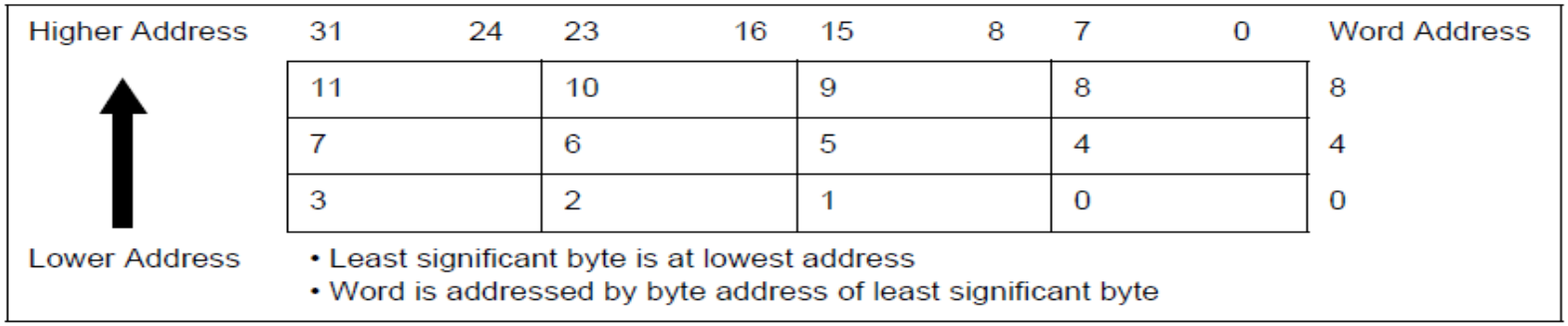
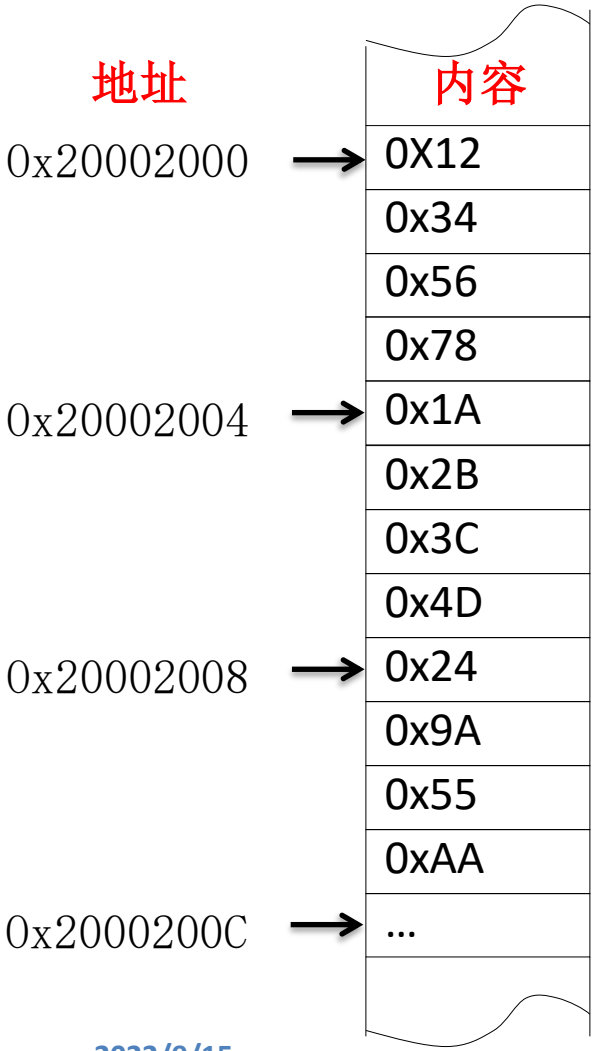


Figure 3-2: Little endian addresses of bytes within words

2.3 存储器内容查看



地址	Big-endian	Little-endian
0x20002000	0x12345678	0x78563412
0x20002004	0x1A2B3C4D	0x4D3C2B1A
0x20002008	0x249A55AA	0xAA559A24
0x20002008	…	…

2.3 存储器内容查看

Memory 1

Address: 0x800125c

```
0x0800125C: 24 4C 10 33 15 60 0A 68 42 F0 01 02 0A 60 18 60 0D 68 1F 4A 25 40 0D 60 18 32 13 1D 1C 1D 1C 4D 10 60
0x0800127E: 28 35 18 60 2A 1D 20 60 13 1D 1C 1D 28 60 25 1D 10 60 2A 1D 18 60 13 1D 20 60 1C 1D 28 60 10 60 18 60
0x080012A0: 20 60 0B 68 4F F0 B0 42 10 4C 23 F4 80 23 0B 60 60 34 20 60 D2 F8 E4 00 0E 49 40 F4 80 40 C2 F8 E4 00
0x080012C2: 08 68 6F F3 0F 00 B0 F1 00 5F 03 D2 0A 49 01 20 C1 F8 08 01 43 F2 D2 00 08 49 02 4B 4A 03 80 3B 08 60
0x080012E4: 1A 60 30 BD 88 ED 00 E0 00 44 02 58 7F ED F6 EA 00 10 00 5C 00 80 00 51 00 40 00 52 FE E7 09 07 00 28
0x08001306: 4F EA 11 61 04 DB 00 F1 E0 20 80 F8 00 14 70 47 00 F0 0F 00 00 F1 E0 20 80 F8 14 1D 70 47 02 E0 08 C8
0x08001328: 12 1F 08 C1 00 2A FA D1 70 47 70 47 00 20 01 E0 01 C1 12 1F 00 2A FB D1 70 47 00 00 86 B0 FF F7 33 F8
0x0800134A: 28 4F 4F F6 FF 75 2C 46 38 68 00 04 03 D5 20 1E A4 F1 01 04 F8 DC 00 2C 00 DA FE E7 FF F7 ED F9 FF F7
0x0800136C: 1D FF 1F 4E E0 36 30 68 40 F0 00 70 30 60 30 68 00 F0 00 70 00 90 00 20 FF F7 BC F9 00 21 08 46 FF F7
0x0800138E: C8 F9 38 68 00 04 03 D4 28 1E A5 F1 01 05 F8 DC 00 2D 00 DA FE E7 00 20 FE F7 A7 FF 30 68 40 F0 04 00
0x080013B0: 30 60 30 68 00 22 0E 49 00 F0 04 00 05 90 4F F4 00 50 8D E8 07 00 0B 48 69 46 FF F7 69 F8 00 22 02 21
0x080013D2: 28 20 FF F7 12 FA 28 20 FF F7 02 FA 01 22 0D 20 11 04 FF F7 28 F8 FE E7 00 00 00 44 02 58 00 00 11 11
```

```
   149:
   150:   /* Reset D1CFGR register */
0x0800125C 4C24      LDR           r4,[pc,#144]  ; @0x080012F0
0x0800125E 3310      ADDS          r3,r3,#0x10
0x08001260 6015      STR           r5,[r2,#0x00]
0x08001262 680A      LDR           r2,[r1,#0x00]
0x08001264 F0420201  ORR           r2,r2,#0x01
0x08001268 600A      STR           r2,[r1,#0x00]
0x0800126A 6018      STR           r0,[r3,#0x00]
0x0800126C 680D      LDR           r5,[r1,#0x00]
   151:   RCC->D1CFGR = 0x00000000;
   152:
   153:   /* Reset D2CFGR register */
0x0800126E 4A1F      LDR           r2,[pc,#124]  ; @0x080012EC
0x08001270 4025      ANDS          r5,r5,r4
```

2.3 存储器内容查看

内存数据查看：

```
0x00000000: 18 F0 9F E5 18 F0 9F E5 18 F0 9F E5 18 F0 9F E5 18 F0 9F E5 00 00 A0
0x00000017: E1 08 F8 1F E5 18 F0 9F E5 58 00 00 40 40 00 00 40 44 00 00 40 48 00
0x0000002E: 00 40 4C 00 00 40 00 00 00 00 50 00 00 40 54 00 00 40 FE FF FF EA FE
```

读取下列存储地址的内容：

0x0000000C：

0x00000010:

0x00000014:

0x0000001C:

小　结

计算机中信息的表示方法、运算和存储。

其它基础知识在学习过程中随时补充。

第2章 ARM 处理器

2.1 ARM及其应用

2.2 ARM处理器的体系

2.3 ARM的编程模型

2.1 ARM及其应用

- ARM Ltd 成立于1990年11月
  - 前身为 Acorn计算机公司
- 主要设计ARM系列RISC处理器内核
- 授权ARM内核给生产和销售半导体的合作伙
  - ARM 公司不生产芯片
- 另外也提供基于ARM架构的开发设计技术
  - 软件工具, 评估板, 调试工具,应用软件,总线架构, 外围设备单元， 等等

2.1 ARM及其应用

- ARM (Acorn RISC Machine)公司

全球领先的半导体知识产权（IP)提供商。

ARM

将技术授权给
其它芯片厂商

intel IBM LG NEC Newco SONY . . .

形成各具特色
的ARM芯片

Arm CPU Ecosystem

累计73亿（每年22%），2020年900/秒，700万/天.

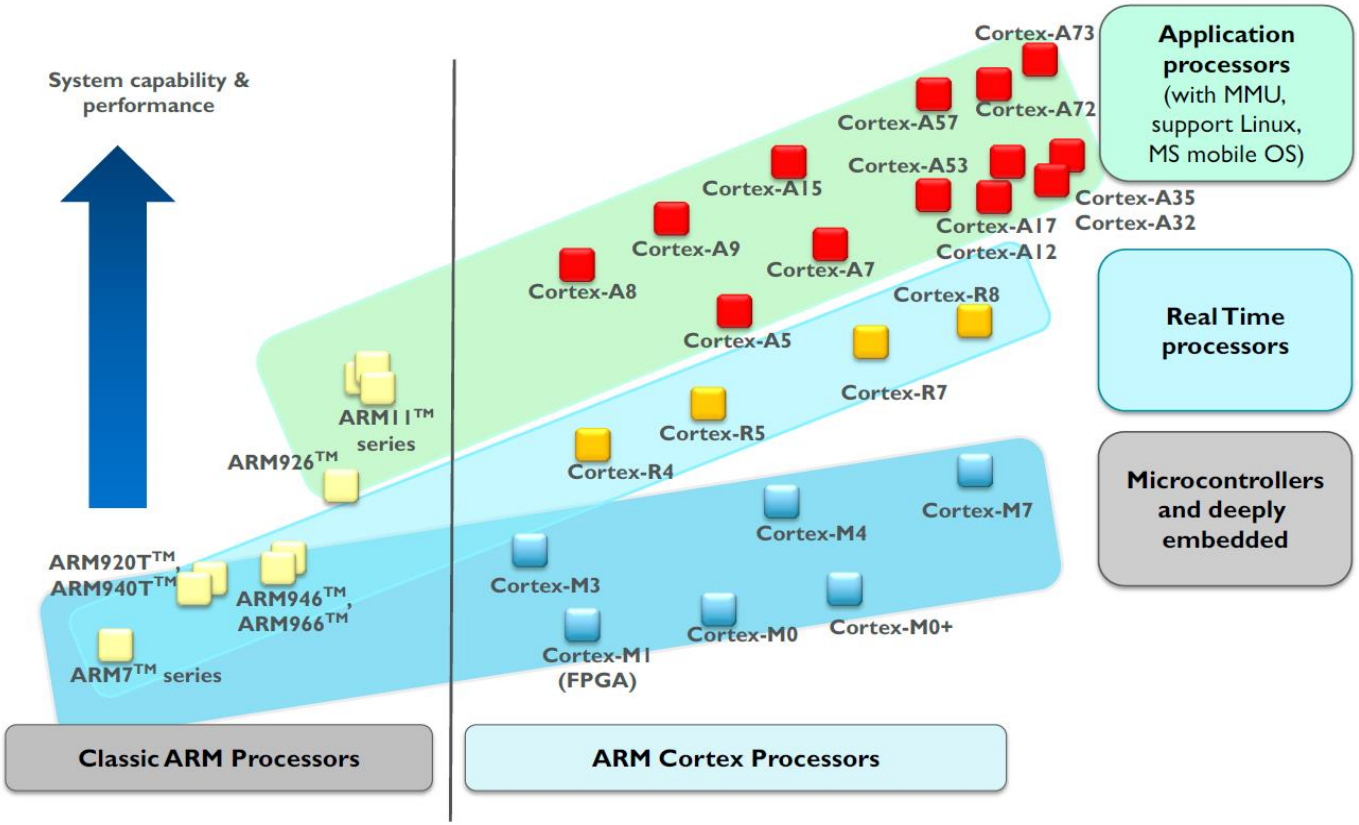
2.2 ARM处理器的体系

架构	处理器家族
ARMv1	ARM1
ARMv2	ARM2、ARM3
ARMv3	ARM6、ARM7
ARMv4	StrongARM、ARM7TDMI、ARM9TDMI
ARMv5	ARM7EJ、ARM9E、ARM10E、XScale
ARMv6	ARM11、ARM Cortex-M
ARMv7	ARM Cortex-A、ARM Cortex-M、ARM Cortex-R
ARMv8	Cortex-A35、Cortex-A50系列[14]、Cortex-A72、Cortex-A73

ARMv9　Cortex-A710

2.2 ARM处理器的体系

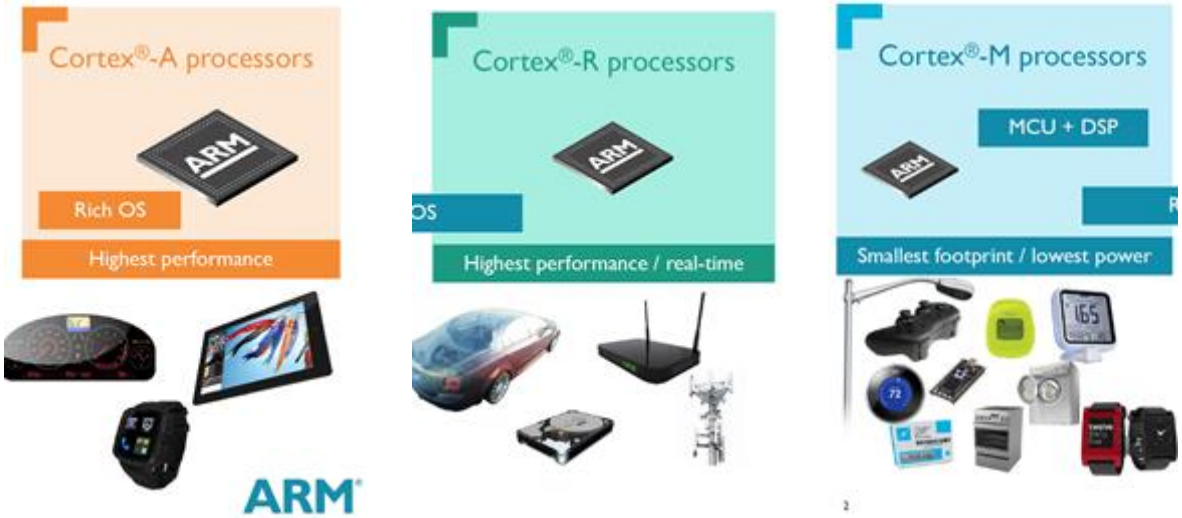
- ARM处理器系列



2.2 ARM处理器的体系

- ARM Cortex系列简介

ARMv7版架构以后的ARM处理器按Cortex系列命名：产品由A、R、M三个系列组成，具体分类延续了一直以来ARM面向具体应用设计CPU的思路。



	Application processors	Real-time processors	Microcontroller processors
设计特点	高时钟频率，长流水线，高性能，对媒体处理支持（NEON 指令集扩展），	高时钟频率，较长的流水线，高确定性（中断延迟低）	通常较短的流水线，超低功耗
系统特性	内存管理单元 (MMU), cache memory, ARM TrustZone® 安全扩展	内存保护单元 (MPU), cache memory, 紧耦合内存 (TCM)	内存保护单元(MPU), 嵌套向量中断控制器 (NVIC), 唤醒中断控制器 (WIC), 最新 ARM TrustZone® 安全扩展.
目标市场	移动计算，智能手机，高能效服务器，高端微处理器	工业微控制器，汽车电子，硬盘控制器，基带	微控制器，深度嵌入系统（例[illegible]传感器，MEMS，混合信号 IC, IoT）

ARM® Cortex®-A Current Portfolio

1H 2015

High Efficiency

Cortex-A53
ARMv8-A
highest efficiency 64/32-bit CPU

Cortex-A5
ARMv7-A
smallest and
lowest power CPU

Cortex-A7
ARMv7-A
highest efficiency
32-bit CPU

Cortex-A9
ARMv7-A 32-bit CPU
Shipping since 2009

High Performance

Cortex-A57
ARMv8-A
64/32-bit proven
high-end CPU

Cortex-A72
ARMv8-A
highest 64/32-bit
performance CPU

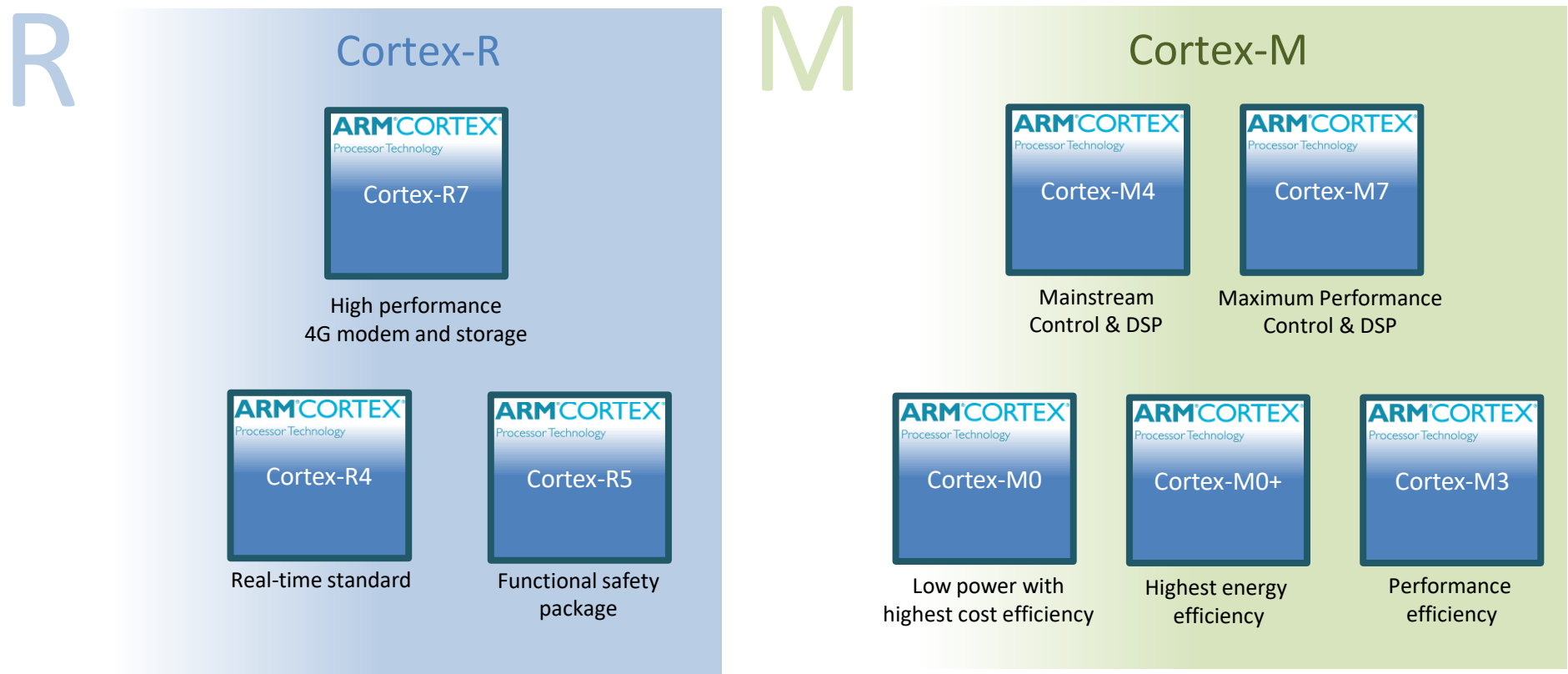
Cortex-A15
ARMv7-A
32-bit performance with
enterprise class feature set

Cortex-A17
ARMv7-A
high performance 32-bit only
CPU for mobile and consumer

摘自 www.arm.com

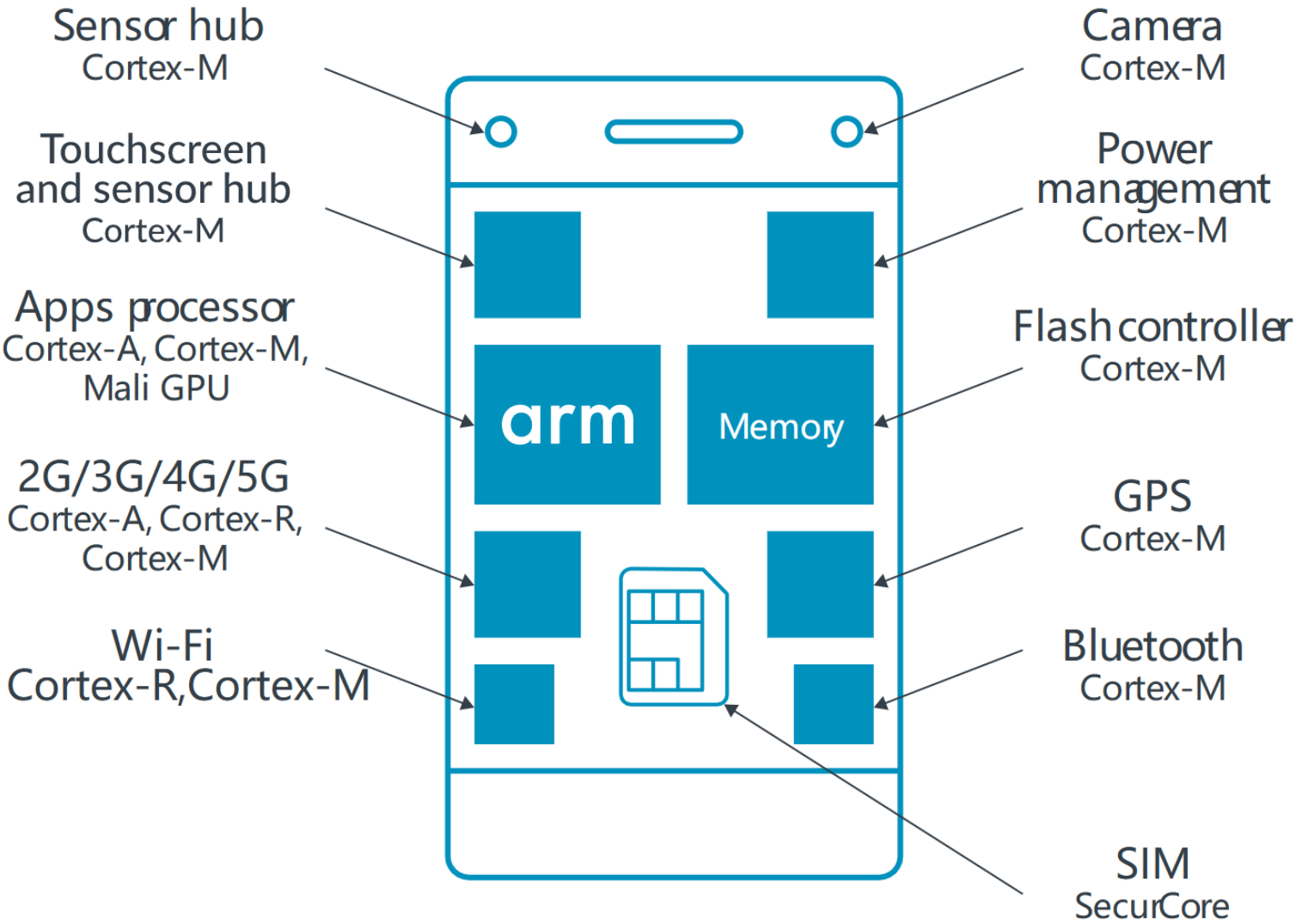
ARM® Cortex®-R and Cortex®-M Processor

1H 2015



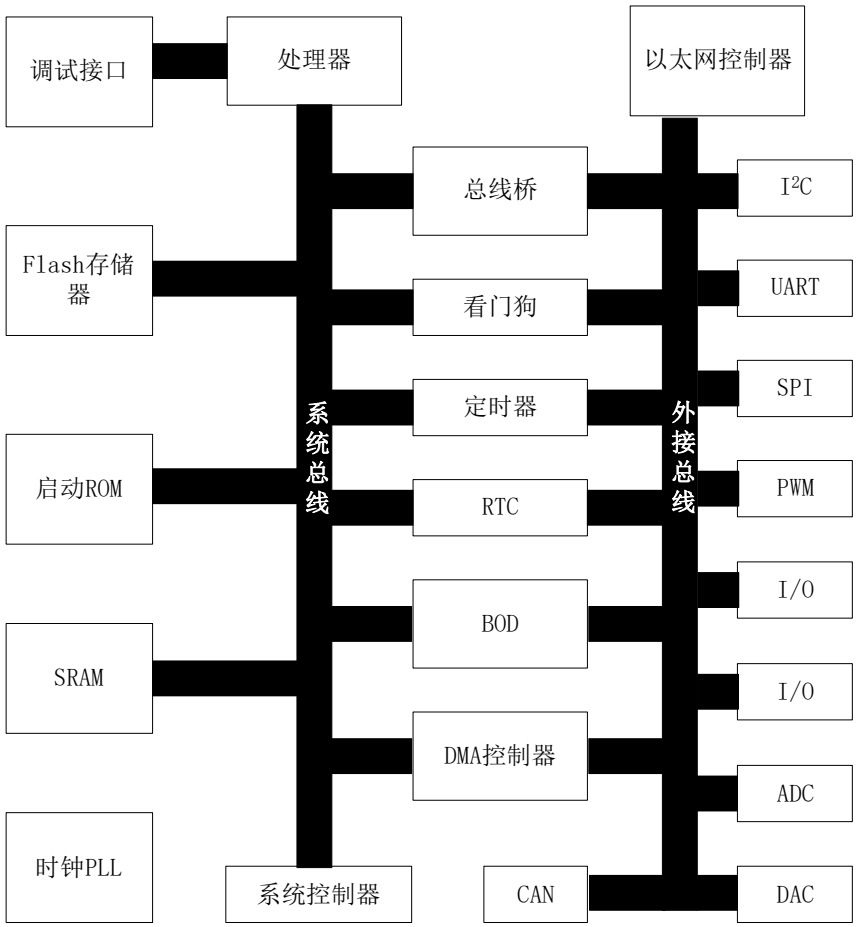
摘自 www.arm.com

智能手机


Sensor hub
Cortex-M
Touchscreen and sensor hub
Cortex-M
Apps processor
Cortex-A, Cortex-M, Mali GPU
2G/3G/4G/5G
Cortex-A, Cortex-R, Cortex-M
Wi-Fi
Cortex-R,Cortex-M
Camera
Cortex-M
Power management
Cortex-M
Flash controller
Cortex-M
GPS
Cortex-M
Bluetooth
Cortex-M
SIM
SecurCore
arm
Memory

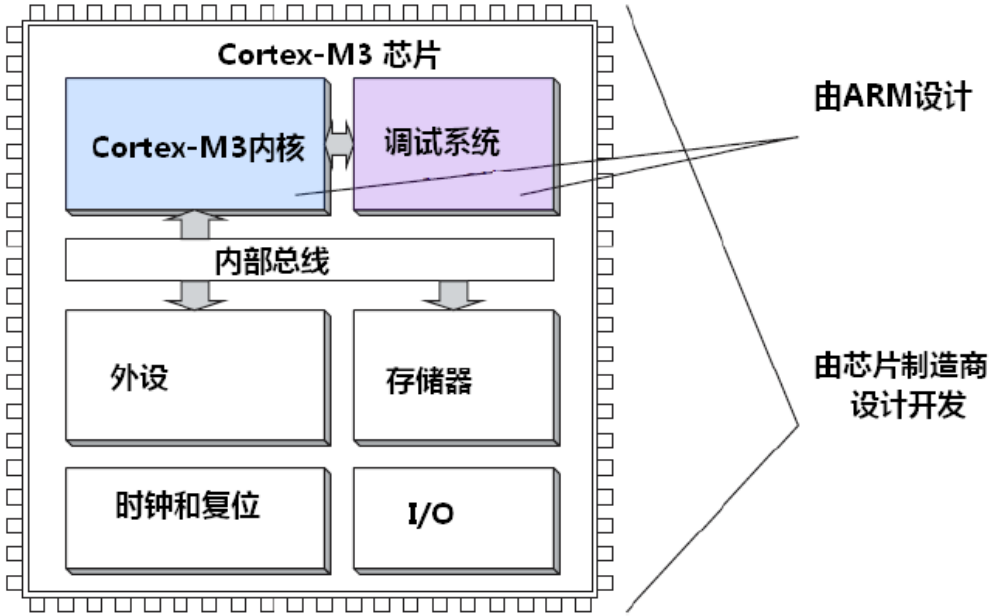
ARM微处理器特点及功能结构

基于ARM架构的微控制器组成如图所示

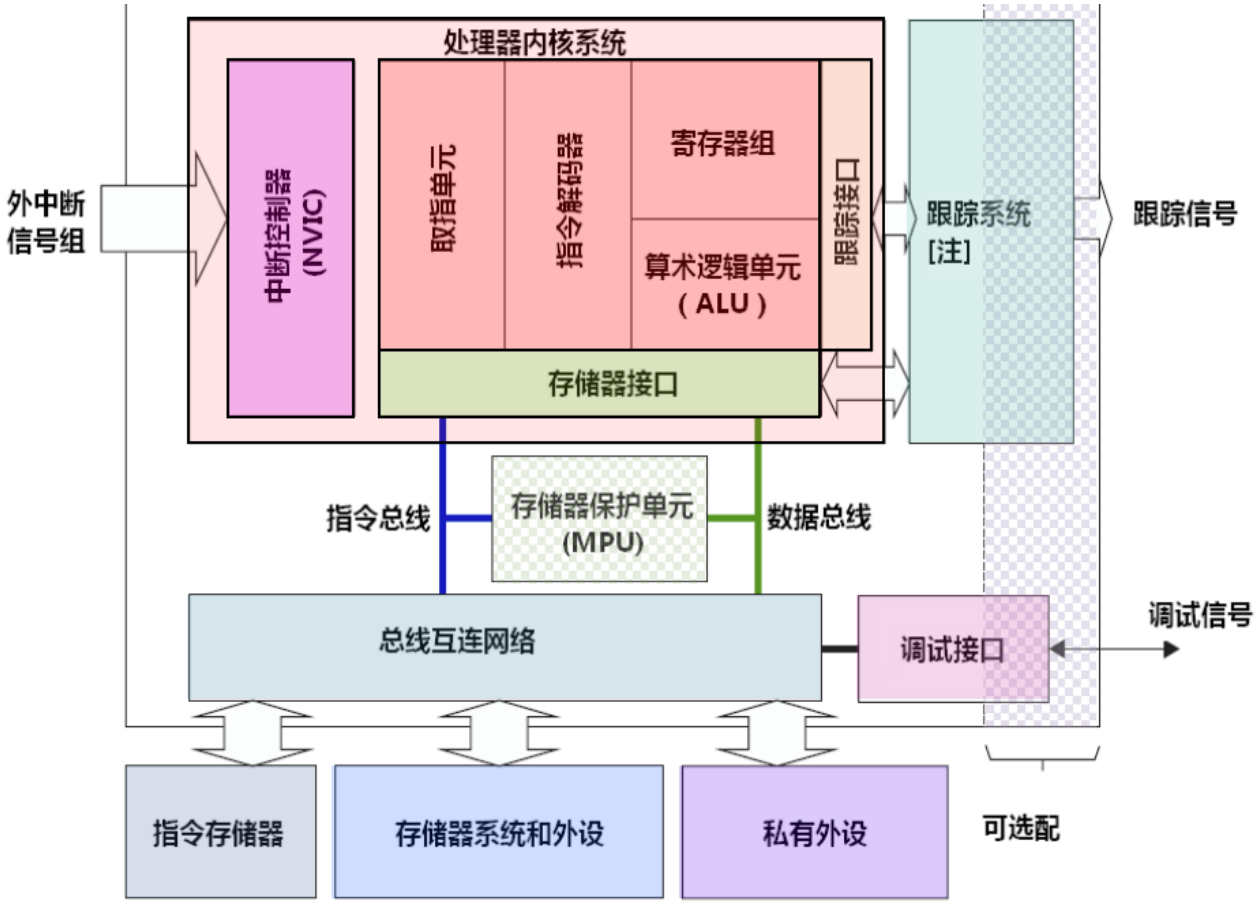


ARM体系结构及特点

- 任何微处理器都是由至少内核、存储器、总线、IO构成。ARM公司的芯片特点是内核部分都是统一的，由ARM设计，但是其它部分各个芯片制造商可以有自己的设计。



Cortex-M内部图

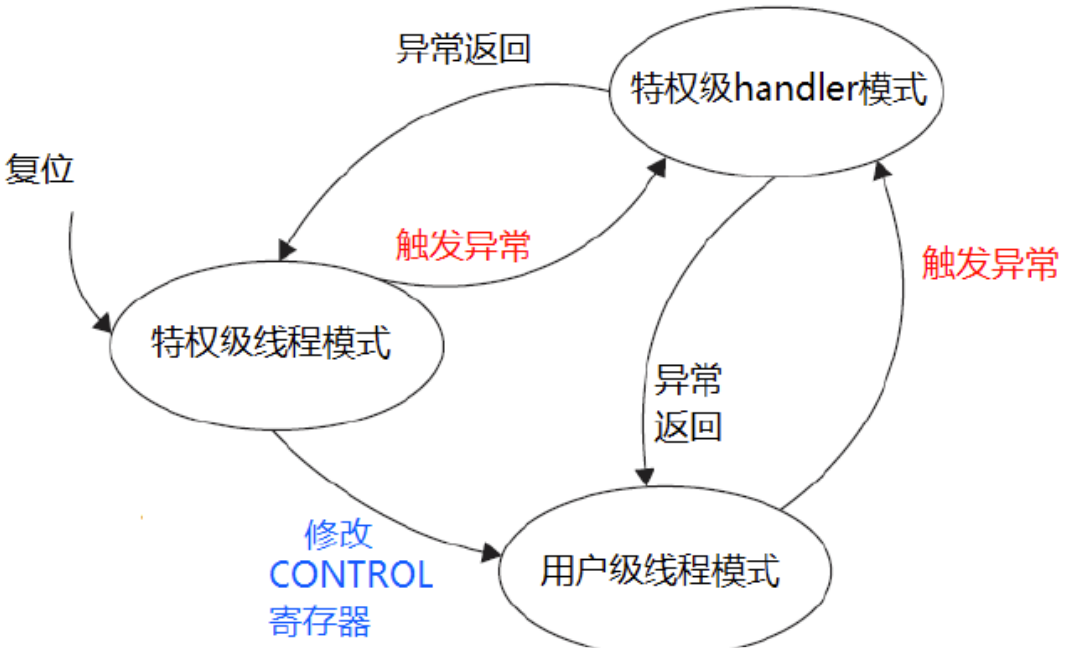


- 寄存器
- ALU
- 存储器
- 总线

2.3 编程模型

Cortex-M处理的编程模型统一

- 只有2种模式，
- 处理器（handler）模式和线程（thread）模式



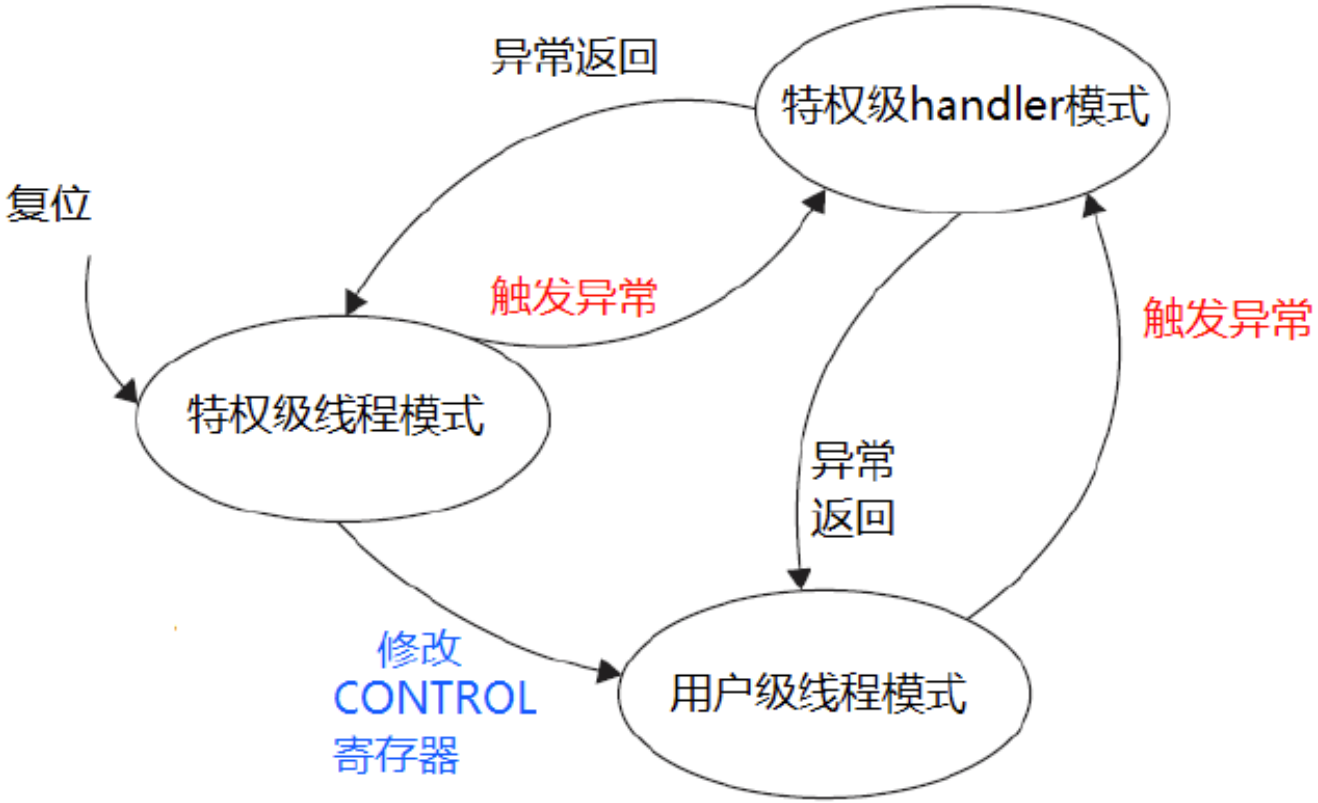
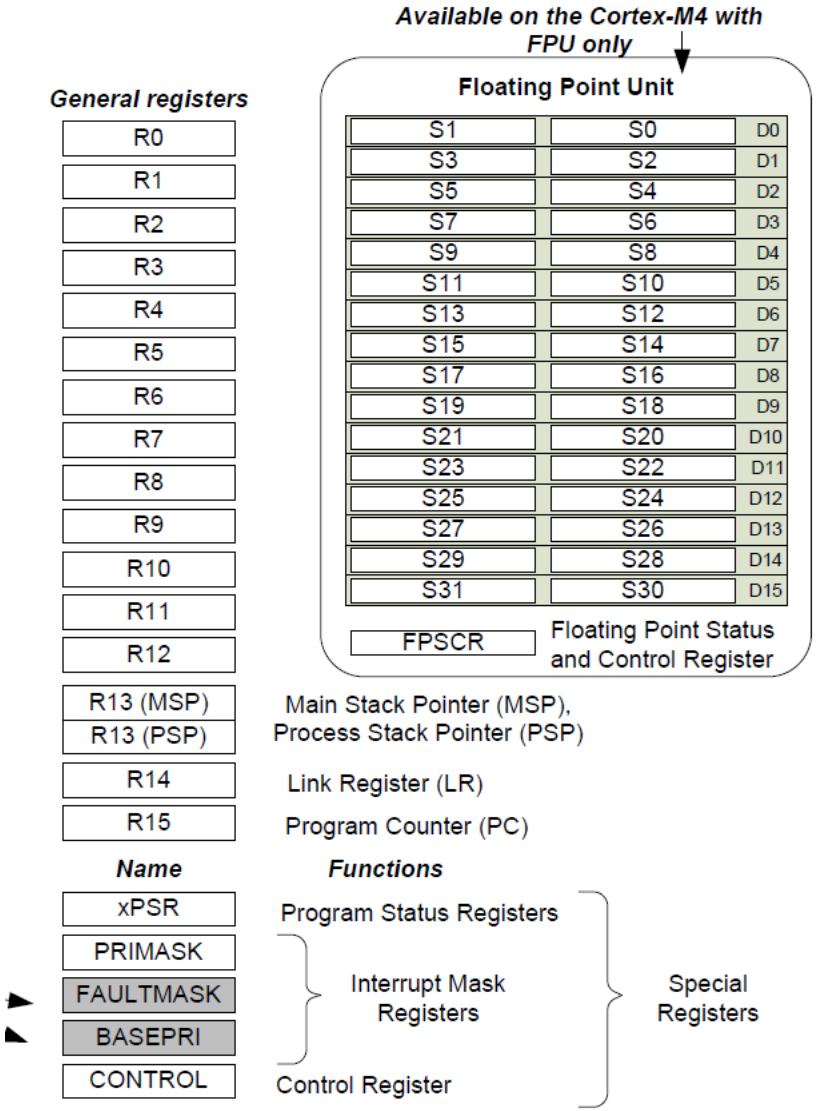
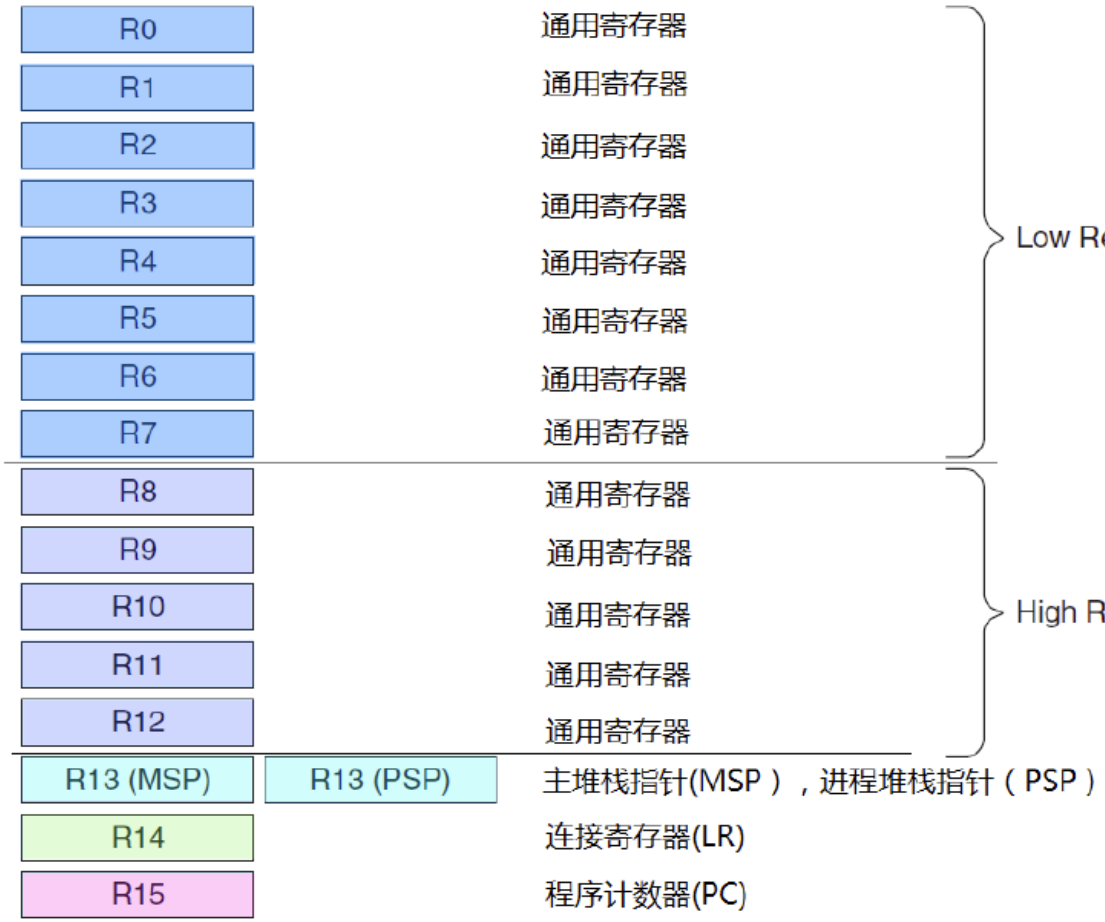


图 2.5　合法的操作模式转换图


Available on the Cortex-M4 with FPU only
Floating Point Unit
S1 S0 D0
S3 S2 D1
S5 S4 D2
S7 S6 D3
S9 S8 D4
S11 S10 D5
S13 S12 D6
S15 S14 D7
S17 S16 D8
S19 S18 D9
S21 S20 D10
S23 S22 D11
S25 S24 D12
S27 S26 D13
S29 S28 D14
S31 S30 D15
FPSCR
Floating Point Status and Control Register
General registers
R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13 (MSP)
R13 (PSP)
Main Stack Pointer (MSP), Process Stack Pointer (PSP)
R14
Link Register (LR)
R15
Program Counter (PC)
Name
Functions
xPSR
Program Status Registers
PRIMASK
FAULTMASK
BASEPRI
Interrupt Mask Registers
CONTROL
Control Register
Special Registers

通用寄存器



主堆栈指针（**MSP**），或写作**SP_main**。是缺省的堆栈指针，它由**OS**内核、异常服务例程以及所有需要特权访问的应用程序代码来使用。可应用于线程模式和处理器模式。

进程堆栈指针（**PSP**）或写作**SP_process**。用于常规的应用程序代码（不处于异常服用例程中时）。只用于线程模式。

连接寄存器R14

- R14是连接寄存器（LR）。在汇编程序中写成LR和R14。
- LR用于在调用子程序时存储返回地址。
- 例如，使用BL(分支并连接，Branch and Link)指令时，就自动填充LR的值。

```
main ;主程序
…
BL function1 ; 使用“分支并连接”指令呼叫function1
; PC= function1，并且LR=main 的下一条指令地址
…
Function1
… ; function1 的代码
BX LR ; 函数返回
```

程序计数器R15

- R15 是程序计数器，“PC”
- 指令流水线，读PC 时返回的值与当前指令所在的地址存在差异

The *Program Counter* (PC) is register R15. It contains the current program address. On reset, the processor loads the PC with the value of the reset vector, which is at address 0x00000004. Bit[0] of the value is loaded into the EPSR T-bit at reset and must be 1.

状态寄存器

- 应用程序 PSR（APSR）
- 中断号 PSR（IPSR）
- 执行 PSR（EPSR）

	31	30	29	28	27	26:25	24	23:20	19:16	15:10	9	8	7	6	5	4:0
APSR	N	Z	C	V	Q											
IPSR												Exception Number				
EPSR						ICI/IT	T			ICI/IT						

Table 4. APSR bit assignments

Bits	Name	Description
[31]	N	Negative flag.
[30]	Z	Zero flag.
[29]	C	Carry or borrow flag.
[28]	V	Overflow flag.
[27]	Q	DSP overflow and saturation flag
[26:20]	-	Reserved
[19:16]	GE[3:0]	Greater than or Equal flags. See *SEL on page 108* for more information.
[15:0]	-	Reserved

Table 5. IPSR bit assignments

Bits	Name	Function
[31:9]	-	Reserved
[8:0]	ISR_NUMBER	This is the number of the current exception: 0 = Thread mode. 1 = Reserved. 2 = NMI. 3 = HardFault. 4 = MemManage. 5 = BusFault 6 = UsageFault 7-10 = Reserved 11 = SVCall. 12 = Reserved for debug 13 = Reserved 14 = PendSV. 15 = SysTick. 16 = IRQ0. . . 256 = IRQ239. see *Exception types on page 39* for more information.

PRIMASK, FAULTMASK 和BASEPRI

名字	功 能 描 述
PRIMASK	只有1 位。为1 时，关掉所有可屏蔽的异常，只剩NMI和硬fault 可以响应。缺省值是0，表示没有关中断。
FAULTMASK	只有1 位。置1 时，只有NMI 才能响应，所有其它的异常，包括中断和fault，通通关闭。它的缺省值也是0，表示没有关异常。
BASEPRI	最多有9 位（由表达优先级的位数决定）。定义了被屏蔽优先级的阈值。当它被设成某个值后，所有优先级号大于等于此值的中断都被关（优先级号越大，优先级越低）。但若被设成0，则不关闭任何中断，0 也是缺省值。

(2)中断屏蔽寄存器

中断屏蔽寄存器分为3组，分别是PRIMASK、FAULTMASK和BASEPRI,.用于控制异常或中断的使能和禁止。

PRIMASK:该寄存器只有一位，为片上外设中断总开关。当该位为0时，响应所有外设中断；当该位为1时，将阻止不可屏蔽中断(NMI)和HardFault异常之外的所有中断/异常。它的默认值是0，表示开放中断。

(2)中断屏蔽寄存器

FAULTMASK:异常屏蔽寄存器，相当于异常的总开关。该寄存器只有最低位有效，即第0位。当该位为0时，响应所有的异常；当该位为1时，除NMI外的所有异常被屏蔽。它的默认值也是0，表示开放中断。

BASEPRI:定义屏蔽优先级的阈值，该寄存器最多有9位。当它被设为某个值后，优先级号大于或等于阀值的中断都被屏蔽。例如，将BASEPRI设置为3，则3和3以上优先级的中断都被屏蔽，而0、1和2优先级的中断不会被屏蔽。其默认值是0，即不屏蔽任何中断。

控制寄存器（CONTROL）

位	功能
CONTROL[1]	堆栈指针选择 0=选择主堆栈指针MSP（复位后缺省值） 1=选择进程堆栈指针PSP 在线程或基础级（没有在响应异常），可以使用PSP。在handler 模式下，只允许使用MSP，所以此时不得往该位写1。
CONTROL[0]	0=特权级的线程模式 1=用户级的线程模式 Handler 模式永远都是特权级的

(3)控制寄存器

控制寄存器**CONTROL**有两个作用，一是定义线程模式的访问级别（特权或非特权）。二是选择堆栈指针（主栈指针或进程栈指针）。

位	功　能
CONTROL[1]	堆栈指针选择 0＝选择主堆栈指针 MSP(复位后默认值) 1＝选择进程堆栈指针 PSP 在线程或基础级(没有在响应异常),可以使用 PSP。在 handler 模式下,只允许使用 MSP,所以此时不得往该位写 1
CONTROL[0]	0＝特权级的线程模式 1＝用户级的线程模式 handler 模式永远都是特权级的

模式改变

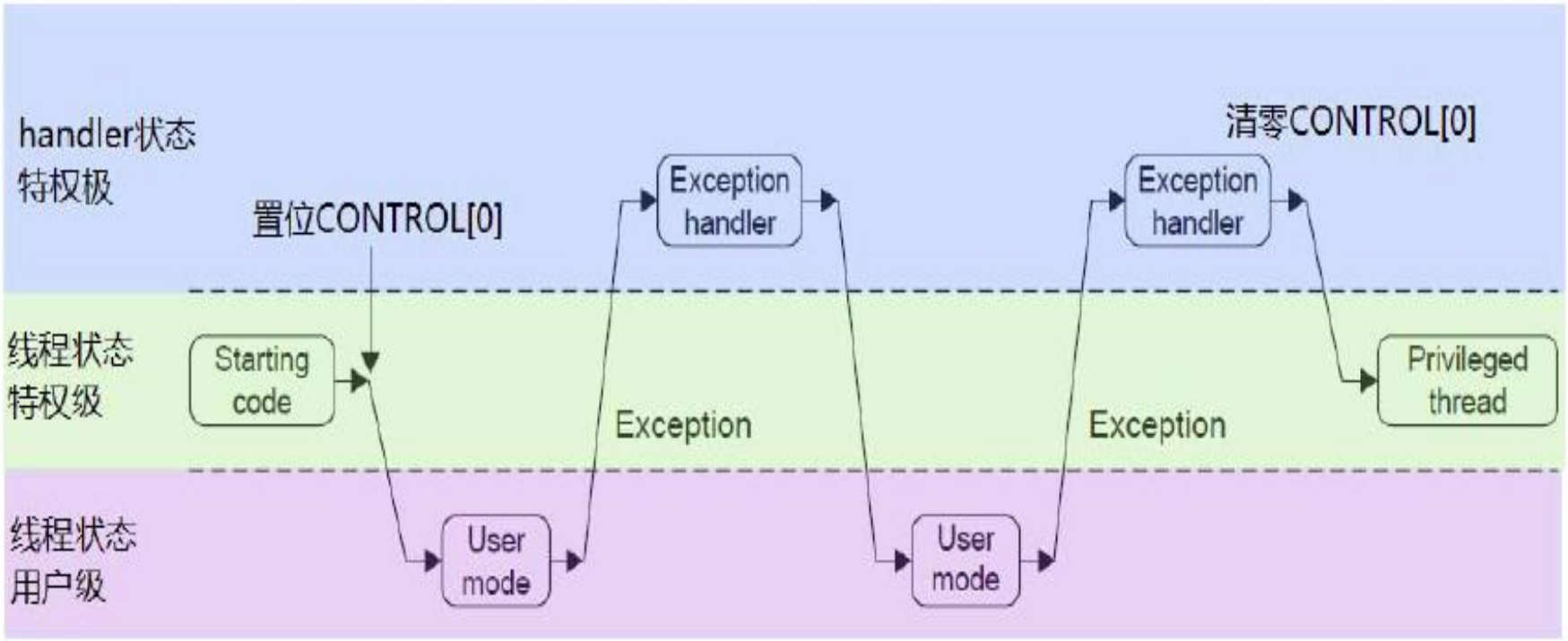
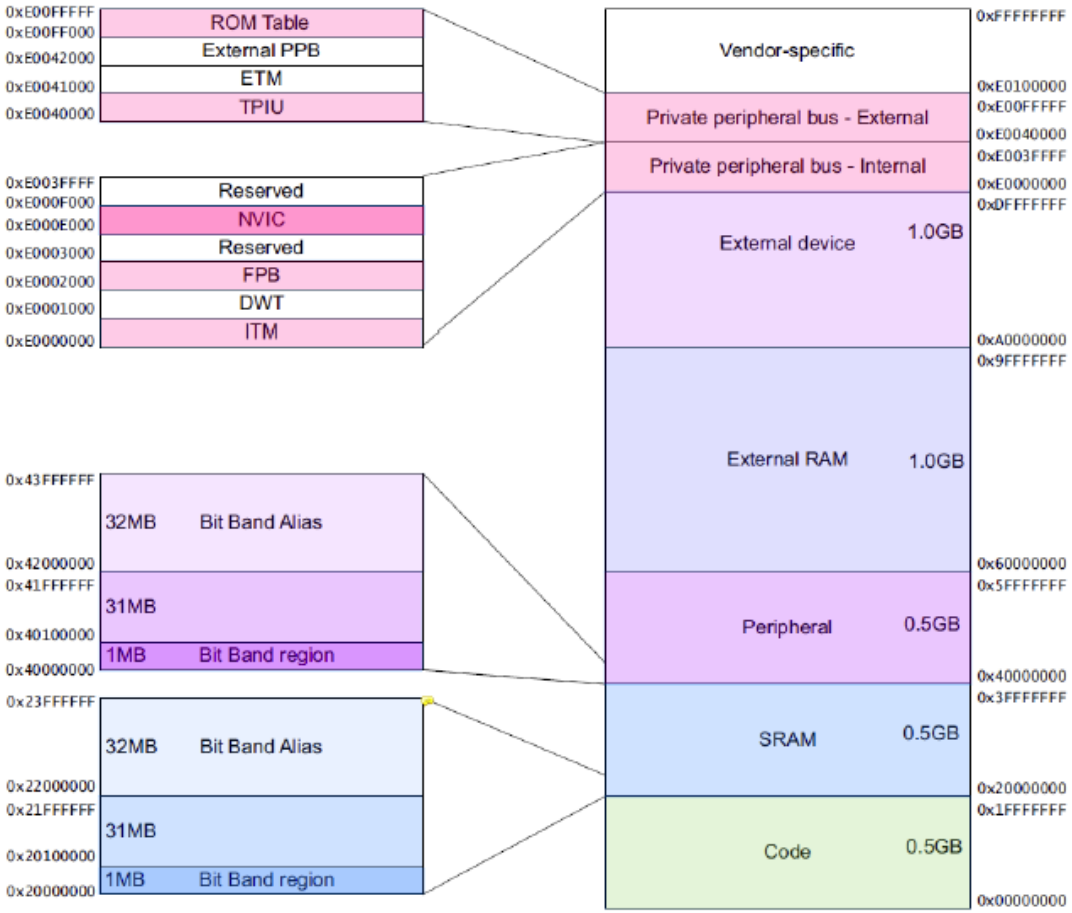
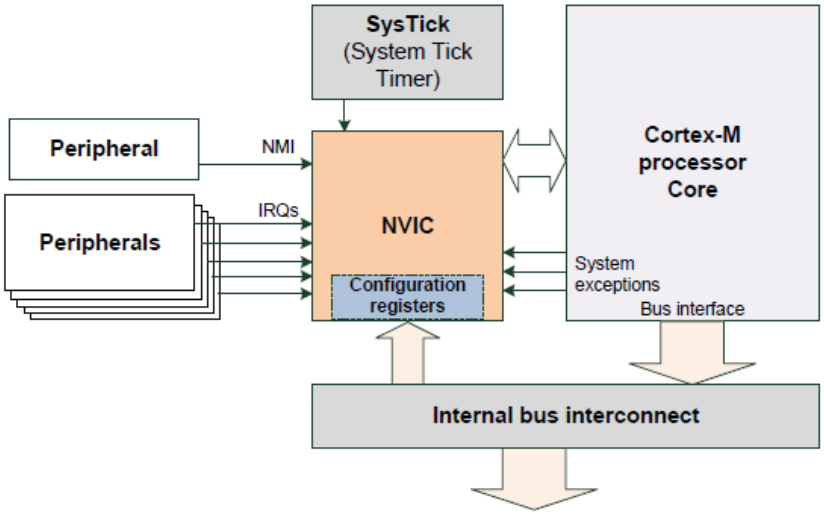


图 3.7　特权级和处理器模式的改变图

地址空间


0xE00FFFFF
0xE00FF000
ROM Table
0xE0042000
External PPB
0xE0041000
ETM
0xE0040000
TPIU
0xE003FFFF
0xE000F000
Reserved
0xE000E000
NVIC
0xE0003000
Reserved
0xE0002000
FPB
0xE0001000
DWT
0xE0000000
ITM
0x43FFFFFF
32MB
Bit Band Alias
0x42000000
0x41FFFFFF
31MB
0x40100000
1MB
Bit Band region
0x40000000
0x23FFFFFF
32MB
Bit Band Alias
0x22000000
0x21FFFFFF
31MB
0x20100000
1MB
Bit Band region
0x20000000
Vendor-specific
0xFFFFFFFF
0xE0100000
Private peripheral bus - External
0xE00FFFFF
0xE0040000
Private peripheral bus - Internal
0xE003FFFF
0xE0000000
External device
1.0GB
0xDFFFFFFF
0xA0000000
External RAM
1.0GB
0x9FFFFFFF
0x60000000
Peripheral
0.5GB
0x5FFFFFFF
0x40000000
SRAM
0.5GB
0x3FFFFFFF
0x20000000
Code
0.5GB
0x1FFFFFFF
0x00000000

中断控制



- Cortex-M支持大量异常，包括16-4-1=11 个系统异常，和最多240 个外部中断——简称IRQ。
- 由外设产生的中断信号，除了SysTick 的之外，全都连接到NVIC 的中断输入信号线。

向量表查表机制

表 3.5　向量表结构

异常类型	表项地址偏移量	异常向量
0	0x00	MSP 的初始值
1	0x04	复位
2	0x08	NMI
3	0x0C	硬 fault
4	0x10	MemManage fault
5	0x14	总线 fault
6	0x18	用法 fault
7-10	0x1c-0x28	保留
11	0x2c	SVC
12	0x30	调试监视器
13	0x34	保留
14	0x38	PendSV
15	0x3c	SysTick
16	0x40	IRQ #0
17	0x44	IRQ #1
18-255	0x48-0x3FF	IRQ #2 - #239