

船说-计算机基础：



条件判断指令处理

CPU设计与结构





1.CPU设计与结构

\ 1.6.1 CPU如何处理判断

- 01 为什么以ARM内核为参考
- 02 CPU中几个有用的寄存器
- 03 CPU处理分支
- 04 更完整的指令处理过程





为什么以ARM内核作为学习参考



1. 简洁的指令集
2. 广泛应用
3. 低功耗设计
4. 开放的资源
5. 系统支持
6. 软硬件生态



Architecture ◆	Core bit-width ◆	Cores		Profile ◆	References ◆
		Arm Ltd. ◆	Third-party ◆		
ARMv1	32	ARM1		Classic	[a 1]
ARMv2	32	ARM2, ARM250, ARM3	Amber, STORM Open Soft Core ^[63]	Classic	[a 1]
ARMv3	32	ARM6, ARM7		Classic	[a 2]
ARMv4	32	ARM8	StrongARM, FA526, ZAP Open Source Processor Core	Classic	[a 2] [64]
ARMv4T	32	ARM7TDMI, ARM9TDMI, SecurCore SC100		Classic	[a 2]
ARMv5TE	32	ARM7EJ, ARM9E, ARM10E	XScale, FA626TE, Feroceon, PJ1/Mohawk	Classic	
ARMv6	32	ARM11		Classic	
ARMv6-M	32	ARM Cortex-M0, ARM Cortex-M0+, ARM Cortex-M1, SecurCore SC000		Microcontroller	
ARMv7-M	32	ARM Cortex-M3, SecurCore SC300	Apple M7 motion coprocessor	Microcontroller	
ARMv7E-M	32	ARM Cortex-M4, ARM Cortex-M7		Microcontroller	
ARMv8-M	32	ARM Cortex-M23, ^[65] ARM Cortex-M33 ^[66]		Microcontroller	[67]
ARMv8.1-M	32	ARM Cortex-M55, ARM Cortex-M85		Microcontroller	[68]
ARMv7-R	32	ARM Cortex-R4, ARM Cortex-R5, ARM Cortex-R7, ARM Cortex-R8		Real-time	
ARMv8-R	32	ARM Cortex-R52		Real-time	[69][70][71]
	64	ARM Cortex-R82 ↗		Real-time	
ARMv7-A	32	ARM Cortex-A5, ARM Cortex-A7, ARM Cortex-A8, ARM Cortex-A9, ARM Cortex-A12, ARM Cortex-A15, ARM Cortex-A17	Qualcomm Scorpion/Krait, PJ4/Sheeva, Apple Swift (A6, A6X)	Application	



ARMv8-A	32	ARM Cortex-A32 ^[72]		Application	
	64/32	ARM Cortex-A35, ^[73] ARM Cortex-A53, ARM Cortex-A57, ^[74] ARM Cortex-A72, ^[75] ARM Cortex-A73 ^[76]	X-Gene, Nvidia Denver 1/2, Cavium ThunderX, AMD K12, Apple Cyclone (A7)/Typhoon (A8, A8X)/Twister (A9, A9X)/Hurricane+Zephyr (A10, A10X), Qualcomm Kryo, Samsung M1/M2 ("Mongoose") /M3 ("Meerkat")	Application	^[77] ^[1] ^[78] ^[79] ^[80] ^[81]
	64	ARM Cortex-A34 ^[82]		Application	
ARMv8.1-A	64/32	TBA	Cavium ThunderX2	Application	^[83]
ARMv8.2-A	64/32	ARM Cortex-A55, ^[84] ARM Cortex-A75, ^[85] ARM Cortex-A76, ^[86] ARM Cortex-A77, ARM Cortex-A78, ARM Cortex-X1, ARM Neoverse N1	Nvidia Carmel, Samsung M4 ("Cheetah"), Fujitsu A64FX (ARMv8 SVE 512-bit)	Application	^[87] ^[88] ^[89]
	64	ARM Cortex-A65, ARM Neoverse E1 with simultaneous multithreading (SMT), ARM Cortex-A65AE ^[90] (also having e.g. ARMv8.4 Dot Product; made for safety critical tasks such as advanced driver-assistance systems (ADAS))	Apple Monsoon+Mistral (A11) (September 2017)	Application	
ARMv8.3-A	64/32	TBA		Application	
	64	TBA	Apple Vortex+Tempest (A12, A12X, A12Z), Marvell ThunderX3 (v8.3+) ^[91]	Application	
ARMv8.4-A	64/32	TBA		Application	
	64	ARM Neoverse V1	Apple Lightning+Thunder (A13), Apple Firestorm+Icestorm (A14, M1)	Application	
ARMv8.5-A	64/32	TBA		Application	
	64	TBA		Application	
ARMv8.6-A	64	TBA	Apple Avalanche+Blizzard (A15, M2), Apple Everest+Sawtooth (A16) ^[92]	Application	





ARM系列内核

ARMv8.7-A	64	TBA		Application	[93]
ARMv8.8-A	64	TBA		Application	
ARMv8.9-A	64	TBA		Application	
ARMv9.0-A	64	ARM Cortex-A510, ARM Cortex-A710, ARM Cortex-A715, ARM Cortex-X2, ARM Cortex-X3, ARM Neoverse E2, ARM Neoverse N2, ARM Neoverse V2		Application	[94][95]
ARMv9.1-A	64	TBA		Application	
ARMv9.2-A	64	ARM Cortex-A520, ARM Cortex-A720, ARM Cortex-X4		Application	
ARMv9.3-A	64	TBA		Application	[96]
ARMv9.4-A	64	TBA		Application	[97]





ARM内核中的典型寄存器

usr	sys	svc	abt	und	irq	fiq
R0						
R1						
R2						
R3						
R4						
R5						
R6						
R7						
R8						R8_fiq
R9						R9_fiq
R10						R10_fiq
R11						R11_fiq
R12						R12_fiq
R13	R13_svc	R13_abt	R13_und	R13_irq	R13_fiq	
R14	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq	
R15						
CPSR						
	SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq	





CPSR寄存器

You
ARMV7 core 中的CPSR内容

ChatGPT
在ARMv7架构中，CPSR代表当前程序状态寄存器（Current Program Status Register），它是一个32位的寄存器，用于控制处理器的状态以及存放处理器的状态标志。CPSR包含以下几个部分：

1. 条件标志位（Flags）：
 - **N（Negative）**：当结果为负时置位。
 - **Z（Zero）**：当结果为0时置位。
 - **C（Carry）**：在算术运算中，如果最后一次操作产生了进位或借位，则置位。
 - **V（Overflow）**：当发生有符号数运算溢出时置位。
2. 中断禁用位：
 - **I（IRQ disable）**：当置位时，关闭所有普通中断请求。
 - **F（FIQ disable）**：当置位时，关闭所有快速中断请求。
3. 控制位：
 - **T（Thumb）**：当置位时，处理器运行在Thumb状态，执行16位压缩指令集。如果未置位，处理器运行在ARM状态，执行32位指令集。
 - **M（Mode）**：低5位定义了处理器的运行模式，例如用户模式（User）、系统模式（System）和各种特权模式，如中断模式（IRQ）、快速中断模式（FIQ）、监控模式（Monitor）、管理模式（Abort）等。
4. 扩展标志：
 - **GE（Greater than or Equal）**：在SIMD操作中使用，分别对应每个字节/半字的比较结果。
5. 保留位：
 - ARM架构保留了一些位，用于未来扩展或者特定的处理器实现。

在ARM的编程和操作中，CPSR寄存器非常关键，因为它不仅控制着中断的开关，也提供了条件执行指令的依据，并且标明了当前的处理器模式。这对于系统级编程和操作系统的开发者来说尤为重要，因为它们需要根据不同的处理器状态来管理程序的执行和资源的分配。

🔄 Regenerate response





CPSR寄存器

1. 条件标志位 (Flags) :

- **N (Negative)** : 当结果为负时置位。
- **Z (Zero)** : 当结果为0时置位。
- **C (Carry)** : 在算术运算中, 如果最后一次操作产生了进位或借位, 则置位。
- **V (Overflow)** : 当发生有符号数运算溢出时置位。

2. 中断禁用位:

- **I (IRQ disable)** : 当置位时, 关闭所有普通中断请求。
- **F (FIQ disable)** : 当置位时, 关闭所有快速中断请求。

3. 控制位:

- **T (Thumb)** : 当置位时, 处理器运行在Thumb状态, 执行16位压缩指令集。如果未置位, 处理器运行在ARM状态, 执行32位指令集。
- **M (Mode)** : 低5位定义了处理器的运行模式, 例如用户模式 (User)、系统模式 (System) 和各种特权模式, 如中断模式 (IRQ)、快速中断模式 (FIQ)、监控模式 (Monitor)、管理模式 (Abort) 等。

4. 扩展标志:

- **GE (Greater than or Equal)** : 在SIMD操作中使用, 分别对应每个字节/半字的比较结果。

5. 保留位:

- ARM架构保留了一些位, 用于未来扩展或者特定的处理器实现。





CPU如何判断

1. 条件标志位 (Flags) :

- **N (Negative)** : 当结果为负时置位。
- **Z (Zero)** : 当结果为0时置位。
- **C (Carry)** : 在算术运算中, 如果最后一次操作产生了进位或借位, 则置位。
- **V (Overflow)** : 当发生有符号数运算溢出时置位。





CPSR寄存器

asm

Copy code

```
CMP R0, R1      ; Compare R0 with R1  
BEQ equal       ; If R0 equals R1, branch to label "equal"
```

这里`BEQ`是一个条件分支指令，表示“Branch if Equal”。如果`CMP`指令结果导致零标志（Z-flag）被设置，那么`BEQ`指令会跳转到标签`equal`。

类似地，有许多不同的条件码，它们可以附加到许多ARM指令上：

- **EQ** (equal)
- **NE** (not equal)





VisUAL2-ARM汇编模拟器

VisUAL2

RunResetStepStep

HexBinDecuDec

1.3.条件演示.s

1STARTmovr0,#3

2subr0,r0,#1

3cmpr0,#0

4moveqr1,#100

5subr0,r0,#1

6cmpr0,#0

7moveqr1,#101

8subr0,r0,#1

9cmpr0,#0

10moveqr1,#102

Registers

Memory

Symbols

R0	0
R1	0
R2	0
R3	0
R4	0
R5	0
R6	0
R7	0
R8	0
R9	0
R10	0
R11	0
R12	0
R13	-16777216
R14	0
R15	0

N0

Z0

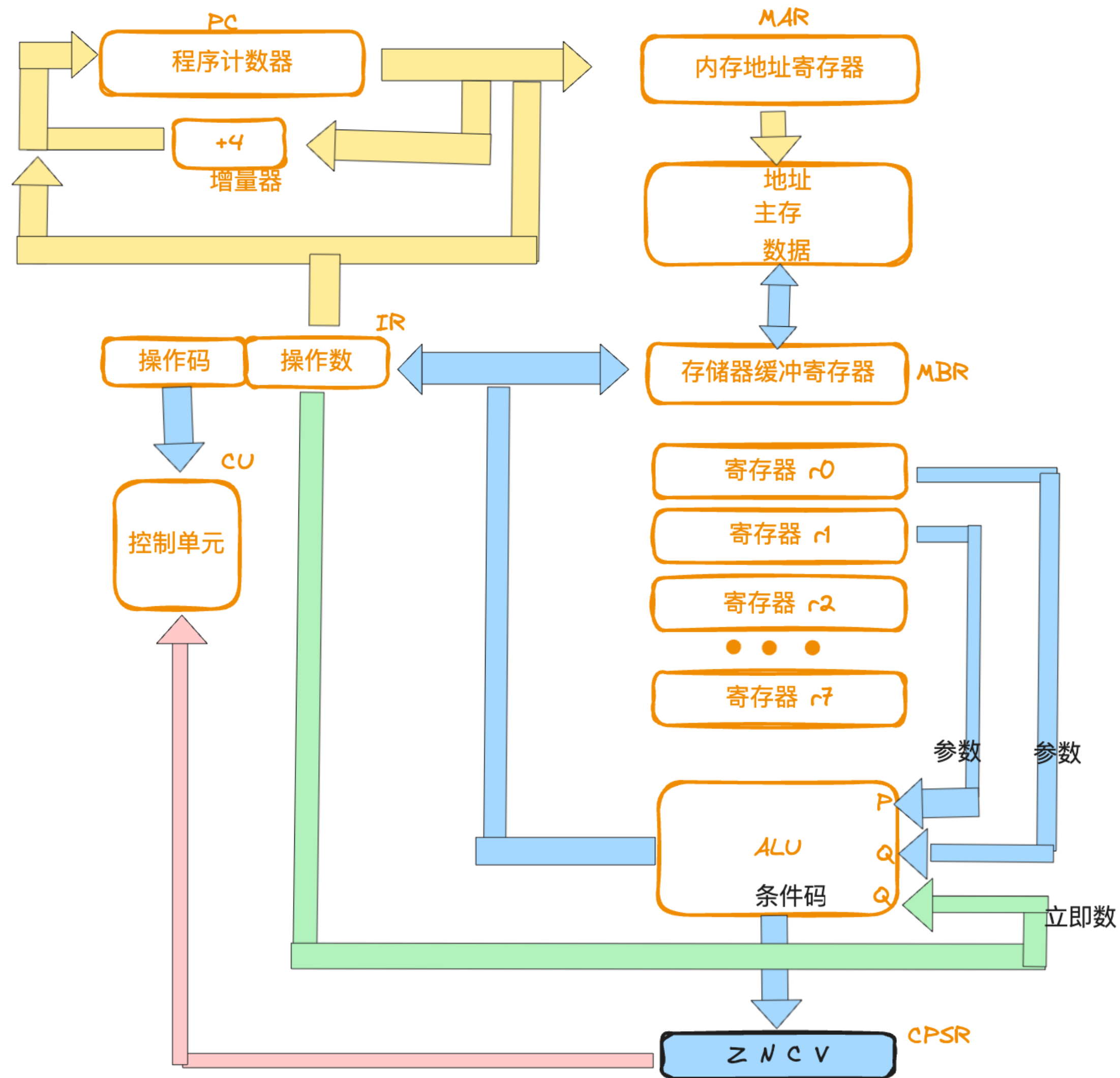
C0

V0





更完整的指令执行过程

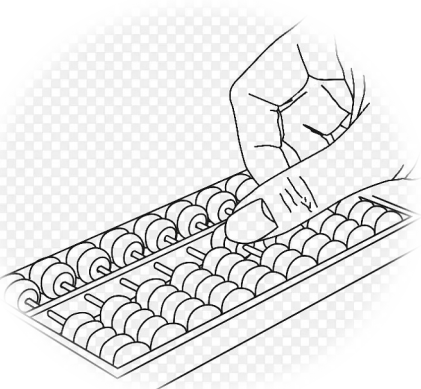




总结

1.6.1 CPU如何处理条件判断

1. CPU中的寄存器分类：特殊，通用
2. 有个CPSR寄存器里有NZCV
3. 执行指令时可以有条件和无条件
4. 更完整的CPU执行指令的流程





欢迎参与学习

W E L C O M E F O R Y O U R J O I N I N G

嵌入式宋船长