ARM 安全比较

; 函数输入：

; r0 = 第一个字符串的指针

; r1 = 第二个字符串的指针

; r2 = 字符串长度

; 函数输出：

; r0 = 0 (字符串不相等) 或 1 (字符串相等)

.global safe\_compare

safe\_compare:

mov r3, #0 ; r3用于存储结果，初始化为0（假定字符串相等）

mov r4, #0 ; r4是循环计数器

compare\_loop:

cmp r4, r2 ; 比较r4和r2（字符串长度）

bge end\_compare ; 如果r4 >= r2，跳到结束比较

ldrb r5, [r0, r4] ; 从第一个字符串加载一个字节

ldrb r6, [r1, r4] ; 从第二个字符串加载一个字节

eor r5, r6 ; XOR比较两个字节

orr r3, r5 ; 将结果与r3做OR操作

add r4, r4, #1 ; 增加循环计数器

b compare\_loop ; 跳回到循环开始

end\_compare:

cmp r3, #0 ; 检查r3是否为0

moveq r0, #1 ; 如果是0，设置返回值为1

movne r0, #0 ; 如果不是0，设置返回值为0

bx lr ; 返回到调用者

ARM 安全拷贝

; 函数输入：

; r0 = 目标缓冲区指针

; r1 = 源缓冲区指针

; r2 = 要拷贝的字节数

; r3 = 目标缓冲区的最大大小

; 函数输出：

; r0 = 拷贝的字节数 (0表示没有拷贝，可能由于缓冲区太小)

.global safe\_copy

safe\_copy:

cmp r2, r3 ; 比较要拷贝的字节数与目标缓冲区的大小

bhi copy\_fail ; 如果r2 > r3, 跳到拷贝失败处理

mov r4, #0 ; r4用作已拷贝的字节数计数器

copy\_loop:

cmp r4, r2 ; 检查是否已拷贝完所有字节

bge copy\_done ; 如果已拷贝足够的字节，结束拷贝

ldrb r5, [r1, r4] ; 从源缓冲区加载一个字节

strb r5, [r0, r4] ; 存储一个字节到目标缓冲区

add r4, r4, #1 ; 更新已拷贝的字节数

b copy\_loop ; 继续拷贝下一个字节

copy\_done:

mov r0, r2 ; 将已拷贝的字节数返回

bx lr ; 返回到调用者

copy\_fail:

mov r0, #0 ; 如果拷贝失败（缓冲区太小），返回0

bx lr ; 返回到调用者

80C51 安全比较

; 函数输入:

; DPTR -> 第一个数组的起始地址

; R0 -> 第二个数组的起始地址

; R1 -> 数组的长度

; 函数输出:

; R0 = 0xFF (如果数组相等)

; R0 = 0x00 (如果数组不相等)

SAFE\_COMPARE:

MOV R2, #0x00 ; R2 用作结果寄存器，初始化为 0x00

MOV R3, #0x00 ; R3 用作索引寄存器，初始化为 0x00

COMPARE\_LOOP:

CLR A ; 清除累加器 A

MOVC A, @A+DPTR ; 将 DPTR 指向的数据加载到 A

MOV B, A ; 将 A 的值存入 B

MOV A, @R0 ; 将 R0 指向的数据加载到 A

XOR A, B ; 比较 A 和 B

ORL R2, A ; 将结果与 R2 做或操作，任何不相等将使 R2 非零

INC DPTR ; 增加 DPTR

INC R0 ; 增加 R0

INC R3 ; 索引自增

CJNE R3, R1, COMPARE\_LOOP ; 如果还没有比较完所有字节，继续循环

MOV A, R2 ; 将最终结果移动到累加器 A

JZ MATCH ; 如果 A 为 0，跳转到匹配标签

MOV R0, #0x00 ; 不匹配，设置返回值为 0x00

SJMP END\_COMPARE ; 跳转到比较结束

MATCH:

MOV R0, #0xFF ; 匹配，设置返回值为 0xFF

END\_COMPARE:

RET ; 返回到调用函数

80C51 安全拷贝

; 函数输入:

; DPTR -> 源数组的起始地址

; R0 -> 目标数组的起始地址

; R1 -> 要拷贝的字节数

; R2 -> 目标数组的最大大小

; 函数输出:

; R0 = 拷贝的字节数 (0表示没有拷贝，可能由于目标缓冲区太小)

SAFE\_COPY:

MOV R3, #0x00 ; R3用作已拷贝的字节数计数器

CJNE R1, R2, CHECK\_SIZE ; 检查要拷贝的字节数是否超过目标缓冲区的大小

SJMP COPY\_START

CHECK\_SIZE:

MOV R1, R2 ; 如果请求的字节数大于目标缓冲区的大小，调整为目标缓冲区大小

COPY\_START:

MOV R3, R1 ; 将要拷贝的字节数设置到R3，用于最后的返回

COPY\_LOOP:

MOVX A, @DPTR ; 从源地址使用间接寻址模式读取数据到累加器A

MOVX @R0, A ; 将数据从累加器A写入目标地址

INC DPTR ; 源地址指针递增

INC R0 ; 目标地址指针递增

DJNZ R1, COPY\_LOOP; 减少R1并检查是否为0，不为0则继续循环

MOV R0, R3 ; 返回已拷贝的字节数

RET ; 返回到调用函数