

ARM

MC404

Um pouco sobre o processador

- Processador mais utilizado no mundo
- Assembly com grande variedade de instruções
- Diversas versões do processador
- Várias extensões ao conjunto de instruções
- Focaremos em instruções de 32 bits
- Começaremos pelas instruções mais simples

Registradores

- 16 registradores de uso geral: r0, r1, ... r15
- Convenção de uso
 - 13 de uso geral: r0 ... r12
 - SP: r13
 - LR: r14
 - PC: r15
- Por hoje, usaremos apenas r0 ... r12

PC

- Endereços de memória organizados de byte em byte
- Todas as instruções têm 32 bits
- As instruções são alinhadas em múltiplos de 4 bytes
- Todos os valores de PC possuem os dois últimos bits [1..0] zerados

Instruções Básicas

- Formato geral: Mnemônico X, Y, Z
 - $X = Y \text{ op } Z$
- Ex.: add r1, r2, r3
 - $r1 = r2 + r3$
- Ex.: add r1, r2, #50
 - $r1 = r2 + 50$

add	$X = Y + Z$
adc	$X = Y + Z + \text{carry}$
sub	$X = Y - Z$
sbc	$X = Y - Z + \text{carry} - 1$
rsb	$X = Z - Y$
rsc	$X = Z - Y + \text{carry} - 1$
cmp	$Y - Z$
cmn	$Y + Z$
tst	$Y \text{ AND } Z$
teq	$Y \text{ XOR } Z$
and	$X = Y \text{ AND } Z$
eor	$X = Y \text{ XOR } Z$
orr	$X = Y \text{ OR } Z$
bic	$X = Y \text{ AND NOT } Z$
mov	$X = Y$
mvn	$X = \text{NOT } Y$

Flags de Condição

Ativados conforme instrução executada. Para gerar flag de condição, é necessário incluir o sufixo S no mnemônico da instrução.

Flag	Instruções Lógicas	Instruções Aritméticas
Negative (N = 1)	-	Bit 31 do resultado foi ativado e significa que o resultado da operação sinalizada é negativo
Zero (Z = 1)	Resultado tem todos os bits com valor 0	Resultado da operação é zero
Carry (C = 1)	Após uma operação de deslocamento, o valor 1 é colocado no carry	Resultado é maior que 32 bits
Overflow (V = 1)	-	Resultado é maior que 31 bits, indicando possivelmente que o bit de sinal foi corrompido em números sinalizados

Execução Condicional

Cada instrução pode ser executada condicionalmente de acordo com os flags atuais. Basta colocar o sufixo adequado da condição.

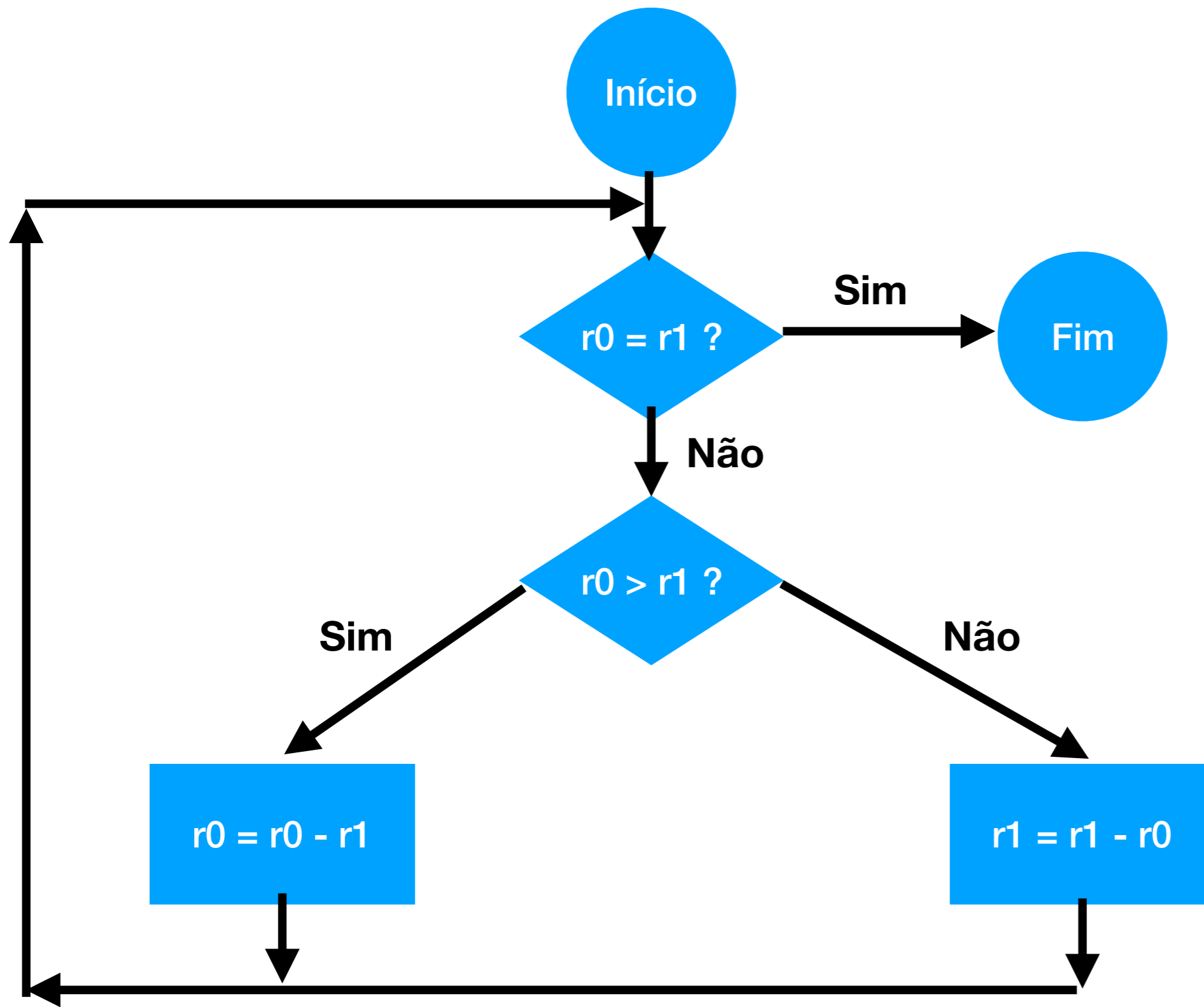
Cond	Descrição
EQ	Z = 1 (igual)
NE	Z = 0 (não igual)
HS / CS	C = 1 (maior ou igual não sinalizado)
LO / CC	C = 0 (menor não sinalizado)
MI	N = 1 (negativo)
PL	N = 0 (positivo ou zero)
VS	V = 1 (overflow)
VC	V = 0 (sem overflow)

Cond	Descrição
HI	C = 1 e Z = 0 (maior sinalizado)
LS	C = 0 OU Z = 1 (menor ou igual sem sinal)
GE	N=1 E V=1 OU N=0 E V=0 (maior ou igual)
LT	N=1 E V=0 OU N=0 E V=1 (menor)
GT	Z=0 E ((N=1 E V=1) OU (N=0 E V=1)) (maior)
LE	Z=0 OU (N=1 E V=0) OU (N=0 E V=1) (menor ou igual)
AL	Sempre
NV	Reservado

Instruções de Salto

- B<cond> destino
 - Salta para o endereço destino se a condição for satisfeita
- BL<cond> função (próxima aula)
 - Salta e guarda o endereço da próxima instrução no registrador LR
 - Para retornar basta executar
 - MOV PC, LR

Maior Divisor Comum



- Implementar usando apenas CMP, B, SUB
- Implementar uma versão onde apenas saltos podem ser condicionais
- Implementar uma versão onde todas as instruções podem ser condicionais

Respostas

```
gcd    cmp    r0, r1
      beq    fim
      blt    menor
      sub    r0, r0, r1
      bal    gcd
menor  sub    r1, r1, r0
      bal    gcd
fim
```

```
gcd    cmp    r0, r1
      subgt  r0, r0, r1
      subl  r1, r1, r0
      bne    gcd
```

Arquitetura Load/Store

- Somente instruções explícitas de acesso à memória conseguem ler e escrever dados na memória
- Load: Lê dados da memória
 - LDR ou LDM
- Store: Escreve dados na memória
 - STR ou STM

Acesso pré-indexado

- `ldr r0, [r1, #12]`
 - Supondo $r1 = 20$, acessa a posição 32 ($=20+12$) de memória e guarda o valor no registrador `r0`.
- `str r5, [r9, #-20]`
 - Supondo $r9 = 100$, grava o valor do registrador `r5` na posição 80 ($=100-20$) de memória.
- Se incluir uma exclamação (!) após a instrução, atualiza o registrador de índice (`r1`, `r9`) pela constante fornecida.

Exercícios

1. Dado um vetor de números inteiros positivos, cujo endereço do primeiro elemento está no registrador r0 e a quantidade de elementos está no registrador r1, faça um programa em assembly do ARM que encontre o menor elemento deste vetor e coloque no registrador r2.
2. Dado um número inteiro fornecido no registrador r0, calcule o número de Fibonacci dele e coloque o resultado no registrador r1.
 - $\text{fib}(x) = \text{fib}(x-1) + \text{fib}(x-2)$
 - Os dois primeiros números são 1 e 1.

Blocos de Código

```
for (i = 0; i < 100; i ++)
```

```
    a += i;
```

```
if (x > 10)
```

```
    y += 7;
```

```
else
```

```
    y -= 7;
```