PSI3441 – Arquitetura de Sistemas Embarcados

- Arquitetura do µProcessador

Escola Politécnica da Universidade de São Paulo

Prof. Gustavo Rehder – grehder@lme.usp.br

Prof. Sergio Takeo – kofuji@usp.br

Prof. Antonio Seabra – acseabra@lsi.usp.br

Objetivo da Aula

- Entender como o processador executa um programa
- O que é o ciclo de máquina
- Entender a microarquitetura de processadores

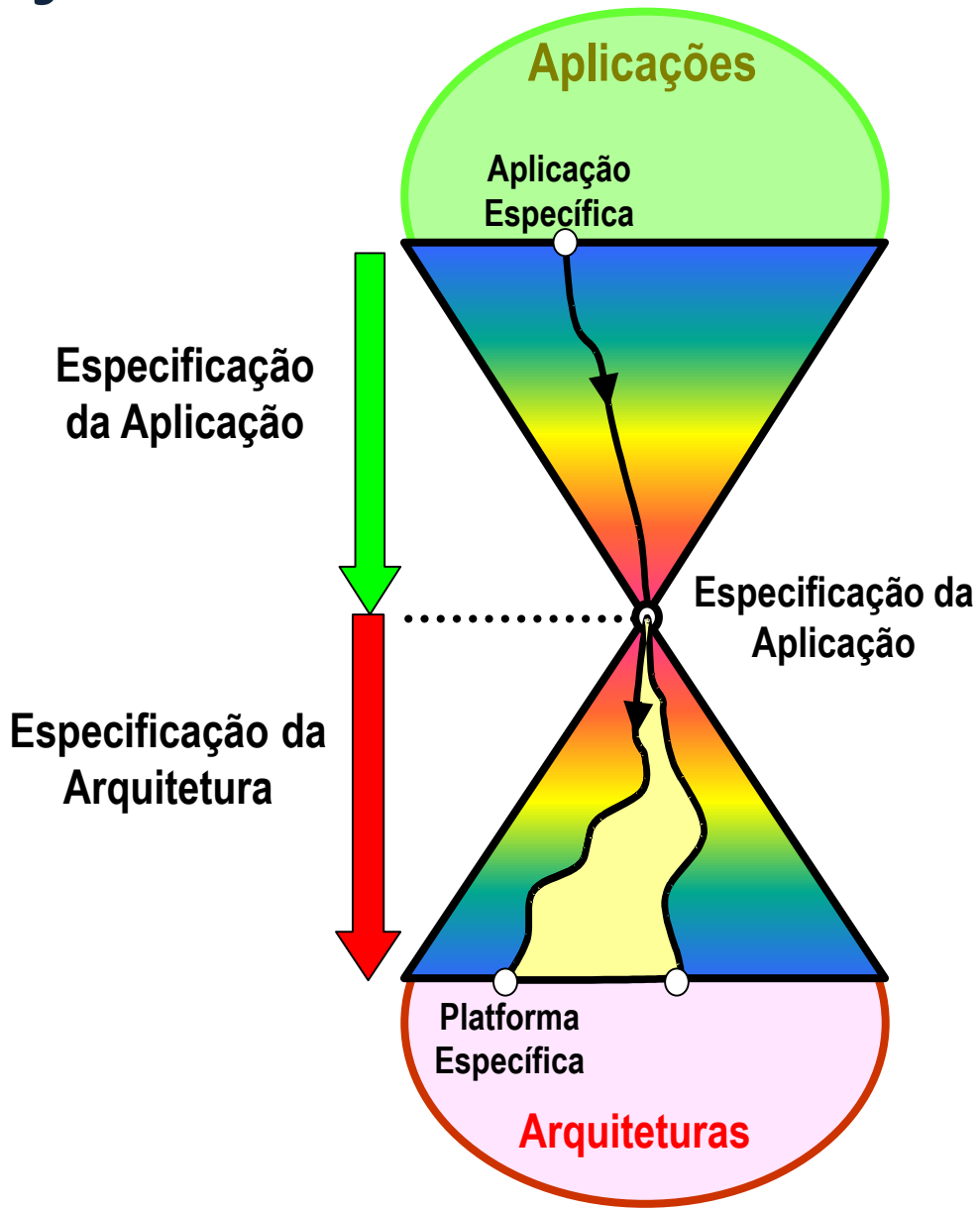
  (focado no Cortex M0+ da ARM)
- Conhecer diferentes instruções executadas pelo processador

Metodologia de Projeto (Baseado em Plataforma)

Especificação da Arquitetura

Definição do Hardware

CPU ?



Definição da CPU

- Processador, Microcontrolador, DSP, FPGA, ASIC?
  (http://www.eetimes.com/document.asp?doc_id=1275272)
- Arquitetura: Load/Store, Register Memory, Stack?
- Von Neumann ou Harvard?
  (http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/ka3839.html)
- RISC ou CISC?
  (https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/)
- 8 bits, 16 bits, 32 bits ou 64 bits?
- 64 Mb, 128 Mb,256 Mb, ... ?
- Velocidade do Clock? 16 MHz, 48 MHz, ... ?
- Memória Externa?
- Periféricos? AD/AD, timers, comunicação serial, ... ?
- Consumo de potência?
- Preço?
- Cache? Pipeline? DMA?

Necessidade de conhecer a arquitetura e funcionamento de CPUs

Tarefa a ser executada pela CPU

Programa

(C, C++, Basic, Pascal, Python, Assembler, VLPs)

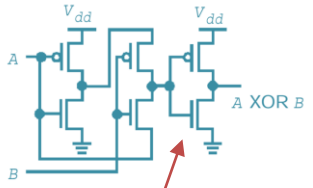
2+2

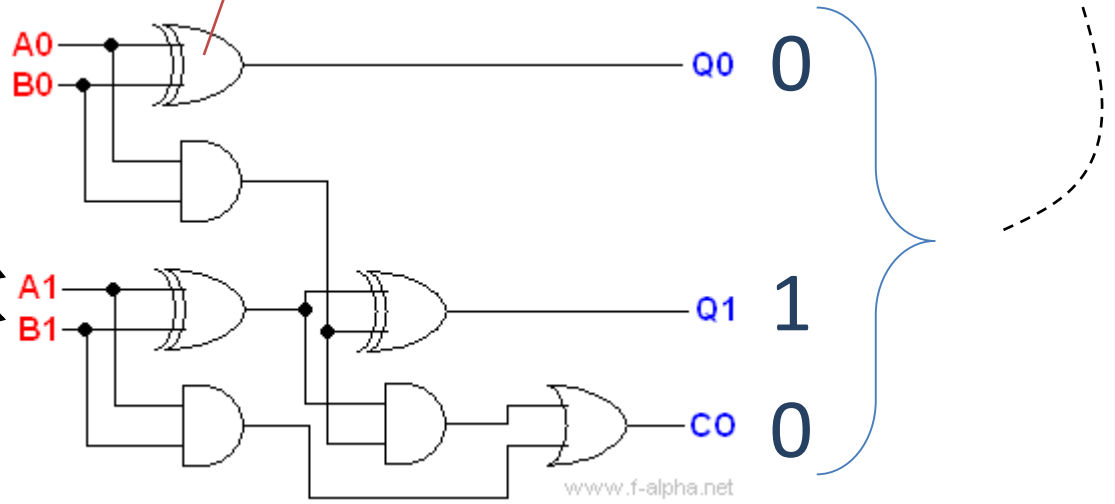
Compilação

= 4

Binário 10 10

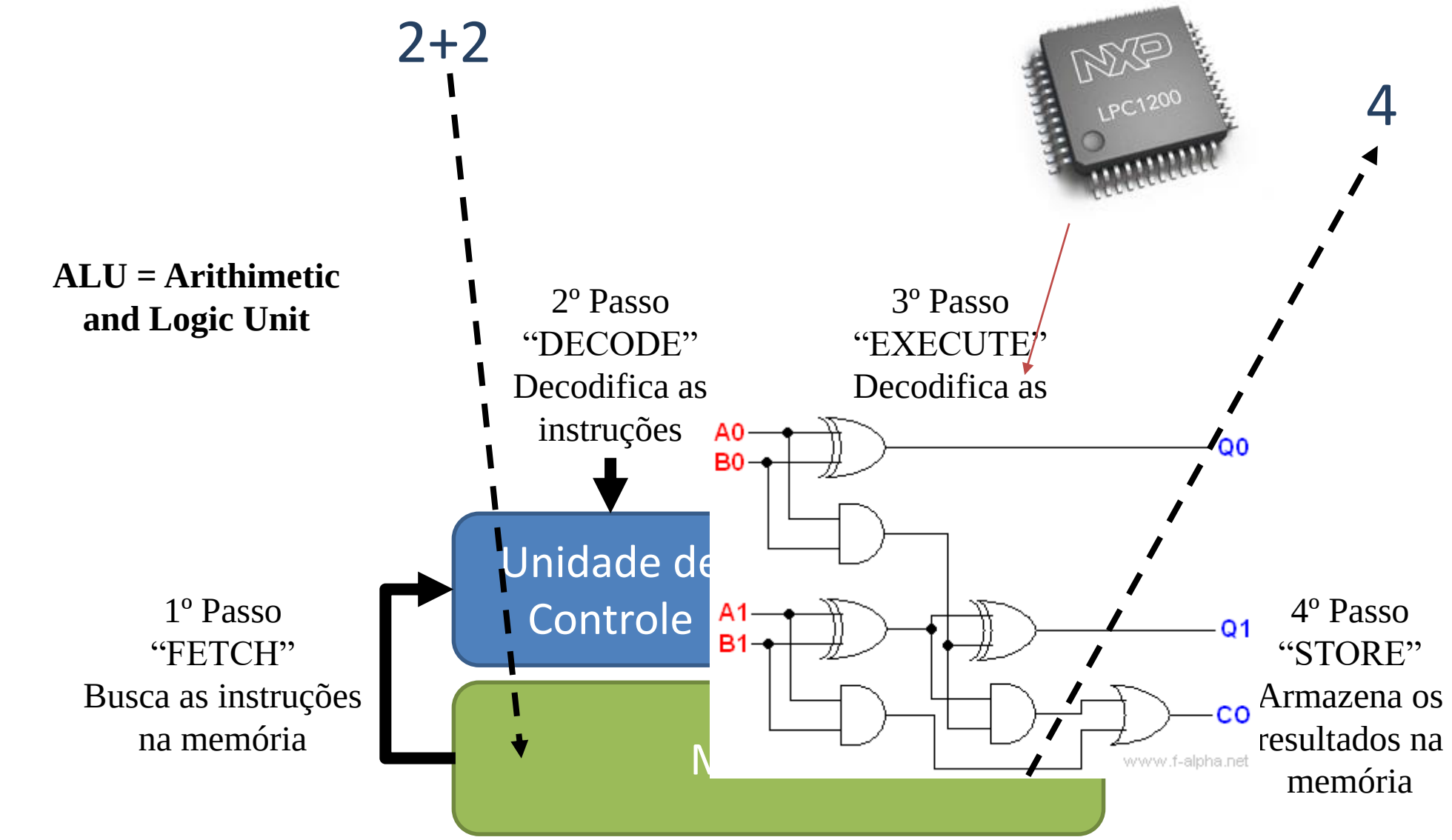
Código de máquina





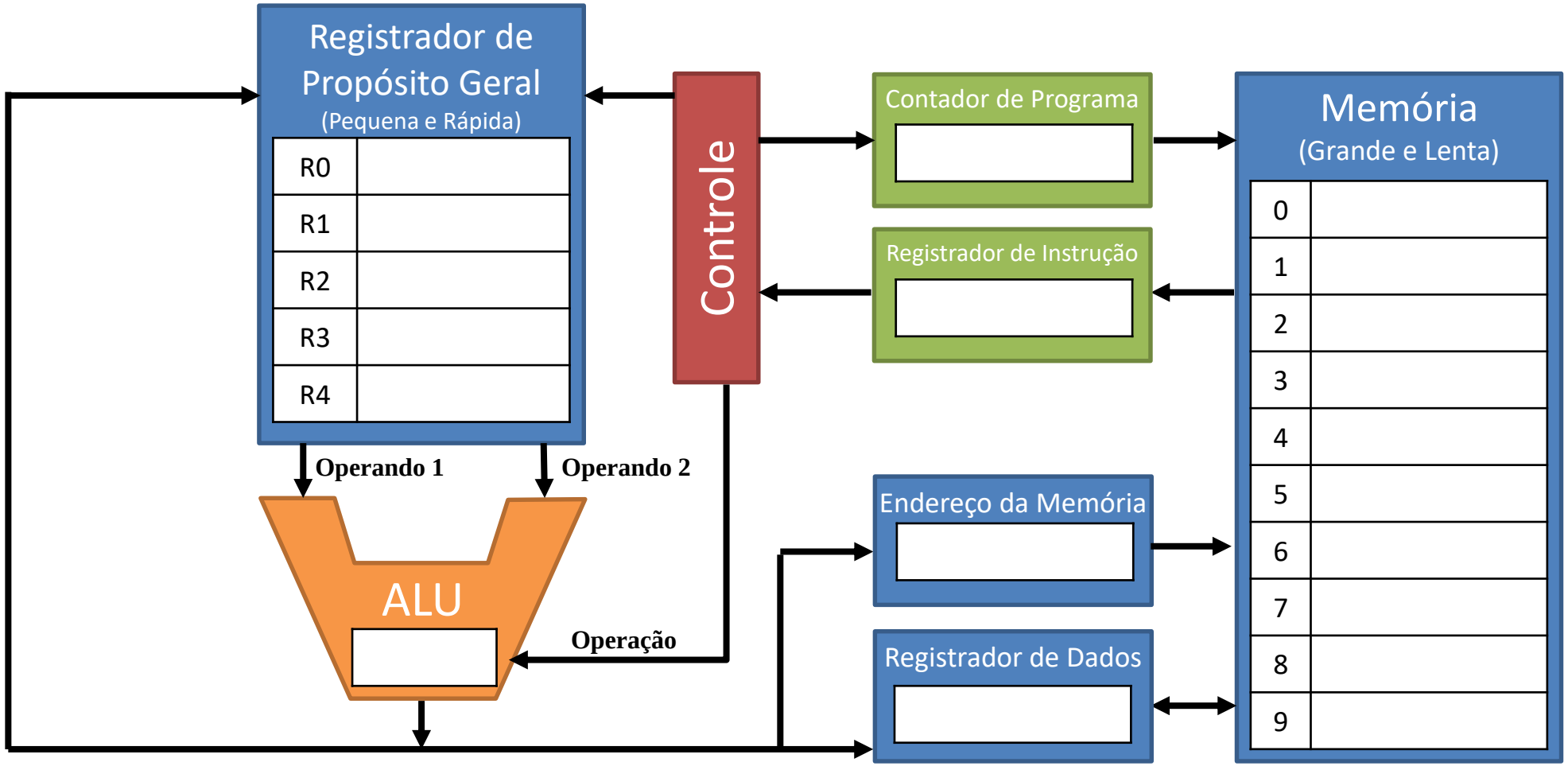
Somador de 2 bits

Dentro da CPU – Ciclo de Máquina



Arquitetura Simplificada “Load/Store”

- A CPU não faz operações na Memória, todas as operações são realizadas pelo ALU utilizando dados do Registrador
- O programa e os dados são gravados na Memória
- O programa é executado sequencialmente, uma instrução em cada linha da memória

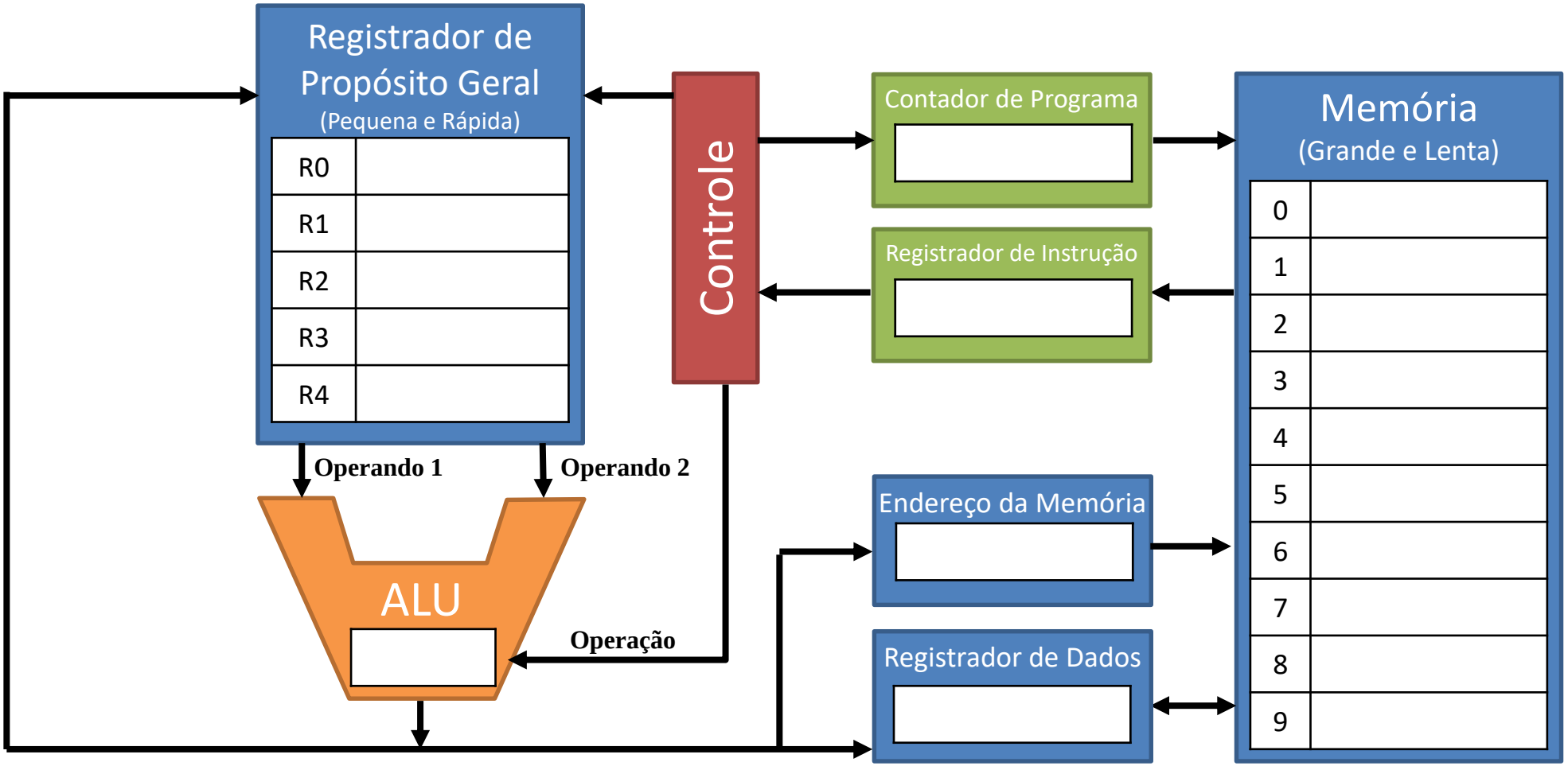


Arquitetura Simplificada “Load/Store”

3 Tipos de Operação (Instruction Set Architecture – ISA):

- Transferência – Leitura, Escrita...
- Dados – Adição, Subtração, Multiplicação, Divisão, Lógica...
- Desvio – Jump, Branch...

Depende do Processador



Arquitetura Simplificada "Load/Store" – ARM Cortex M0+

Formato da Instrução:

Operação Destino, Operando 1, Operando 2

↓

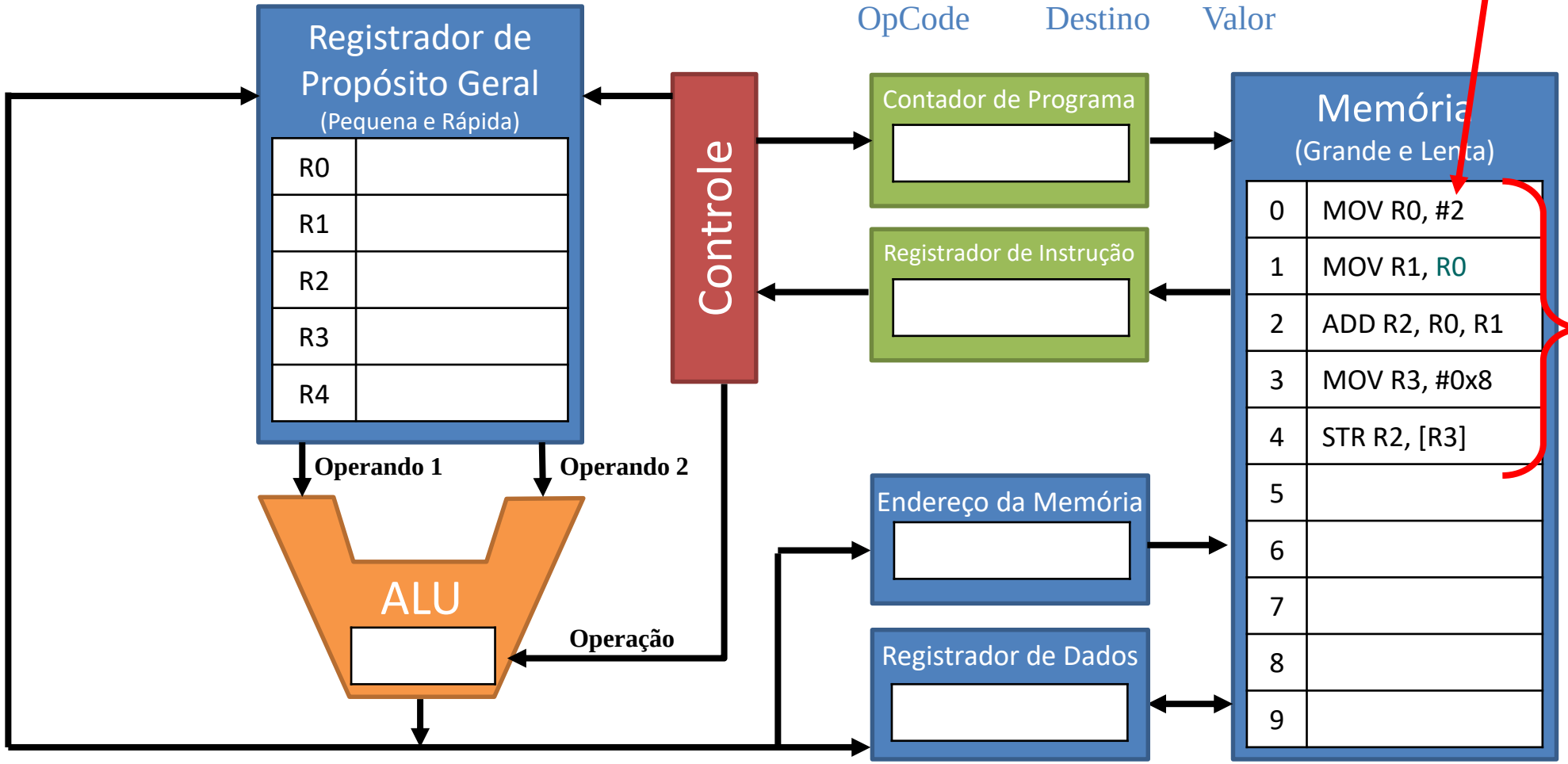
Dependendo da operação não é usado

Instrução: MOVE – atribui o valor de Op2 ao registrador Rn

MOV Rn, Op2

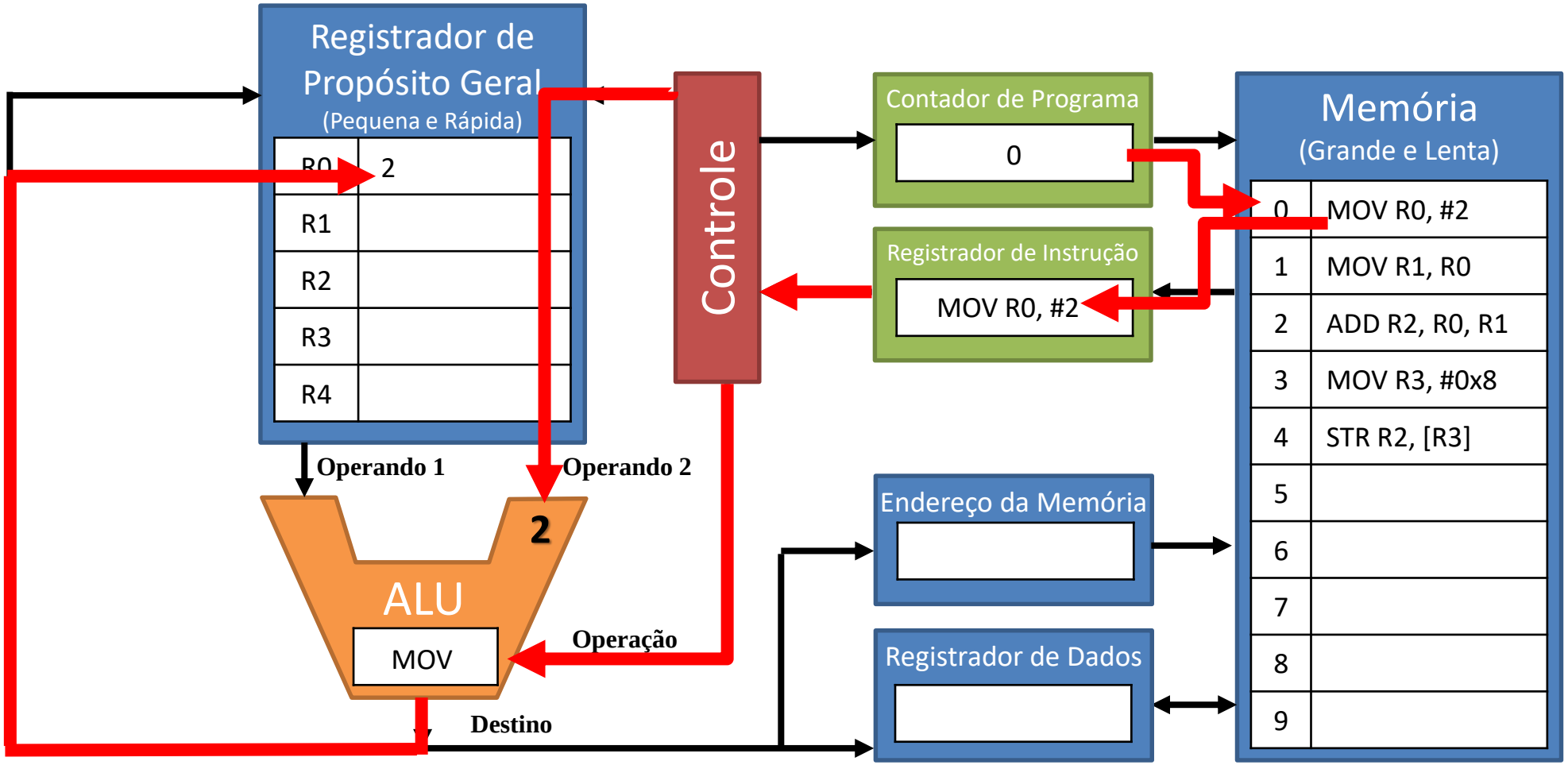
OpCode Destino Valor

Constante de 8 bits



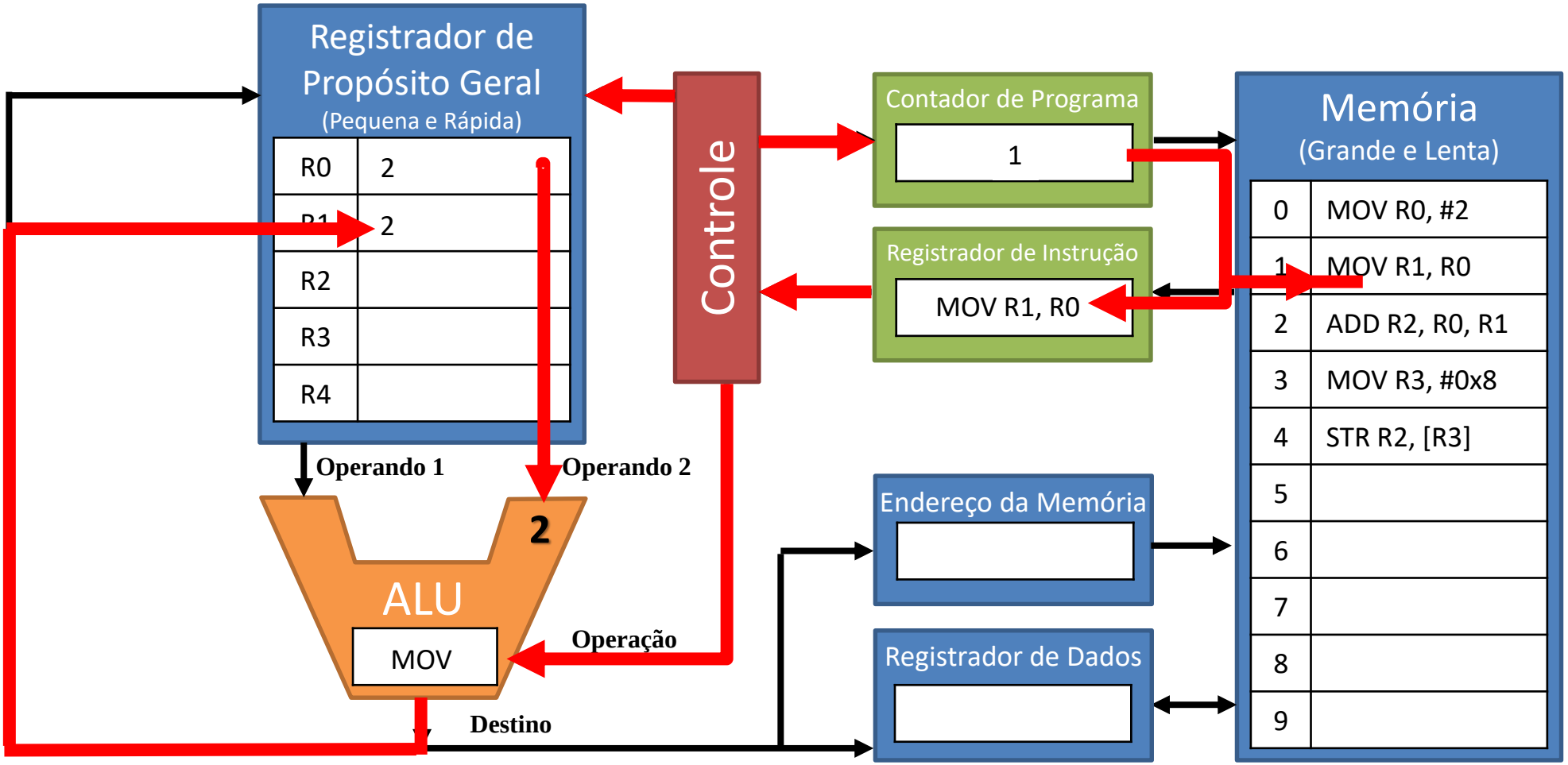
Arquitetura Simplificada "Load/Store" – ARM Cortex M0+

- Depois de inicializar, Contador de Programa (PC) aponta para a próxima instrução a ser executada.
- A instrução é copiada para o Registrador de Instrução (IR) – "FETCH".
- O Controle decodifica a instrução e carrega variáveis – "DECODE".
- A operação é executada pelo ALU – "EXECUTE".
- O resultado é armazenado no registrador R0 – "STORE".



Arquitetura Simplificada "Load/Store" – ARM Cortex M0+

- Ao fim do ciclo o PC é incrementado, apontando para a próxima instrução.
- A instrução é copiada para o Registrador de Instrução (IR) – "FETCH".
- O Controle decodifica a instrução e carrega variáveis – "DECODE".
- A operação é executada pelo ALU – "EXECUTE".
- O resultado é armazenado no registrador R1– "STORE".



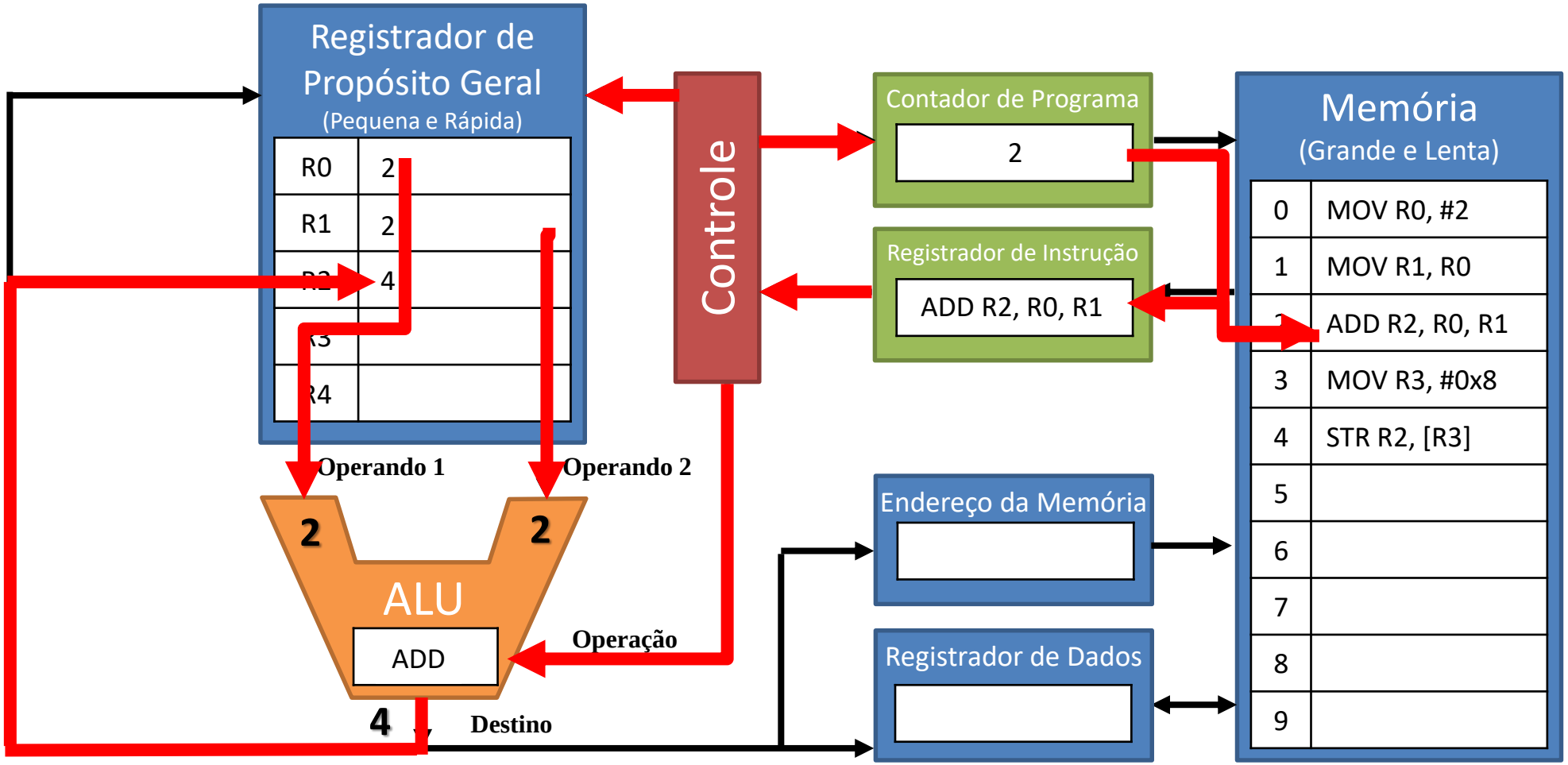
Arquitetura Simplificada "Load/Store" – ARM Cortex M0+

- Ao fim do ciclo o PC é incrementado, apontando para a próxima instrução.
- A instrução é copiada para o Registrador de Instrução (IR) – "FETCH".
- O Controle decodifica a instrução e carrega variáveis – "DECODE".
- A operação é executada pelo ALU – "EXECUTE".
- O resultado é armazenado no registrador R2– "STORE".

Instrução: ADD – Soma valores de Op1 e Op2 e armazena no registrador Rn.

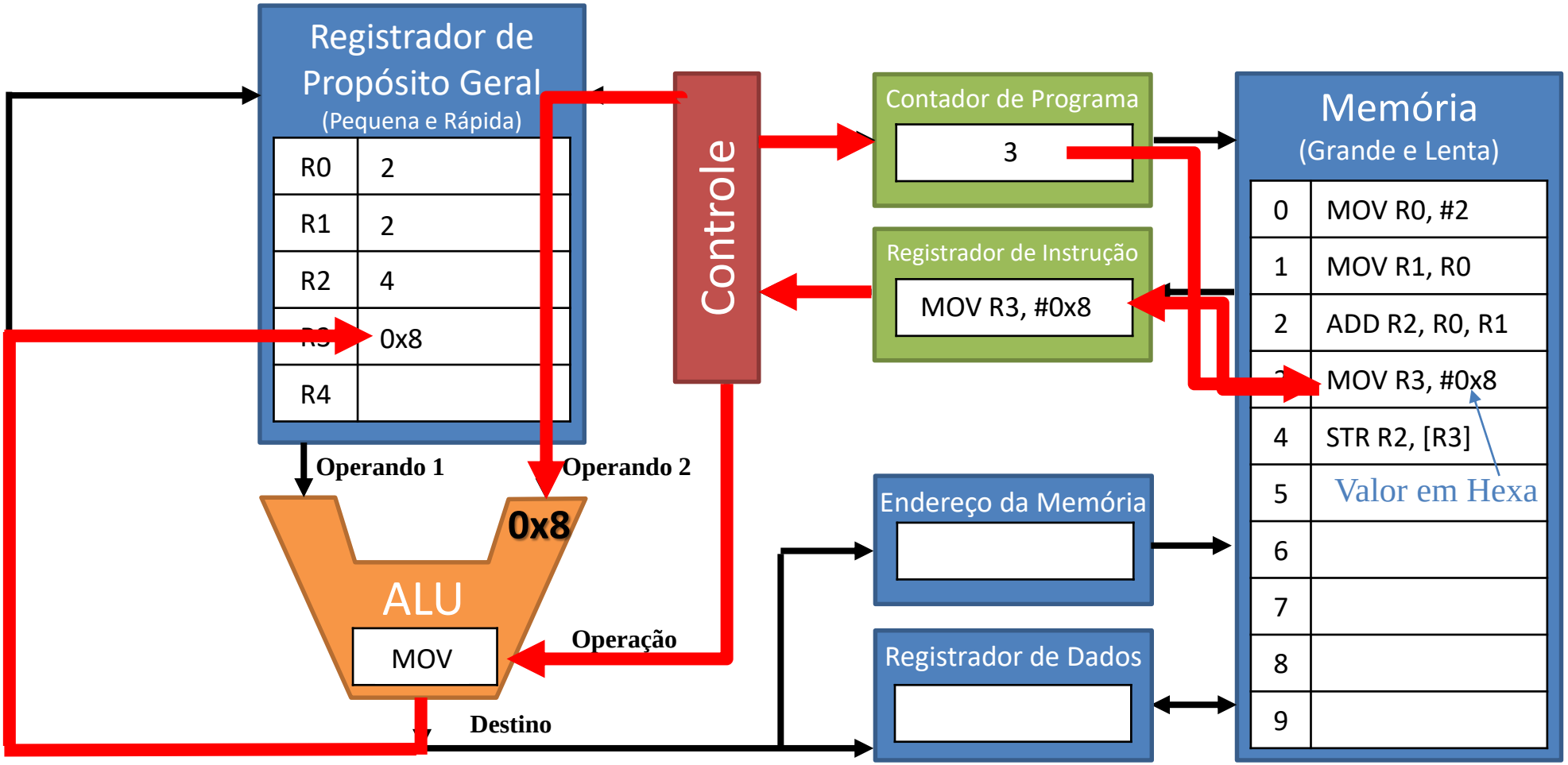
MOV Rn, Op1, Op2 ; Rn = Op1 + Op2

Operação Destino Valor 1 Valor 2



Arquitetura Simplificada "Load/Store" – ARM Cortex M0+

- Ao fim do ciclo o PC é incrementado, apontando para a próxima instrução.
- A instrução é copiada para o Registrador de Instrução (IR) – "FETCH".
- O Controle decodifica a instrução e carrega variáveis – "DECODE".
- A operação é executada pelo ALU – "EXECUTE".
- O resultado é armazenado no registrador R3 – "STORE".

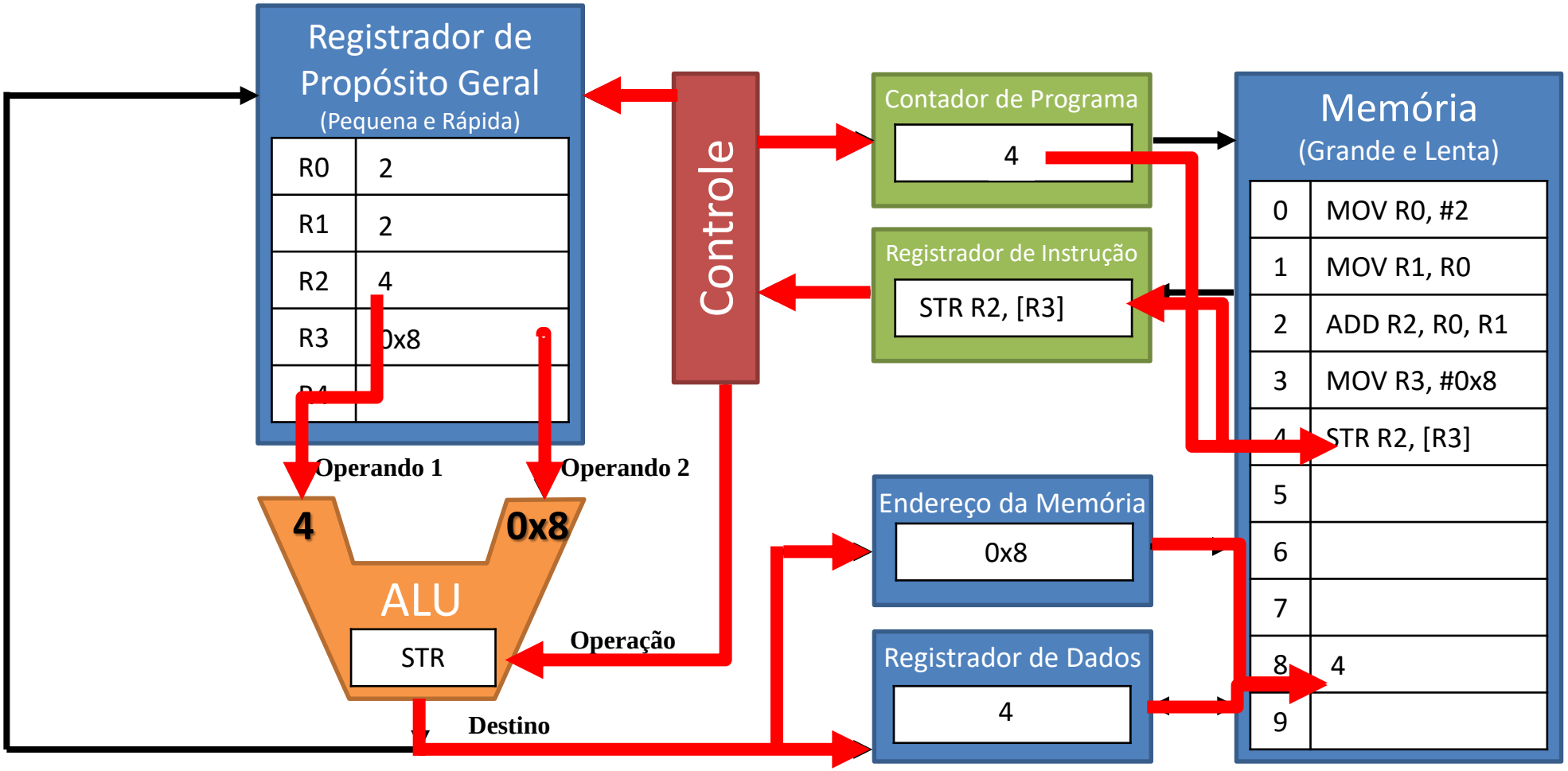


Arquitetura Simplificada "Load/Store" – ARM Cortex M0+

- Ao final do STORE – armazena o valor do registrador Rn no endereço da memória indicado pelo registrador Rd.
- Instrução STORE – Armazena o valor do registrador Rn no endereço da memória indicado pelo registrador Rd.
- A instrução é copiada para o Registrador de Instrução (IR) – "FETCH".
- O Controle decodifica a instrução e carrega variáveis – "DECODE".
- A operação é executada pelo ALU – "EXECUTE".
- O resultado é armazenado na memória no endereço especificado pelo registrador R3 – "STORE".

STR Rn, [Rd]

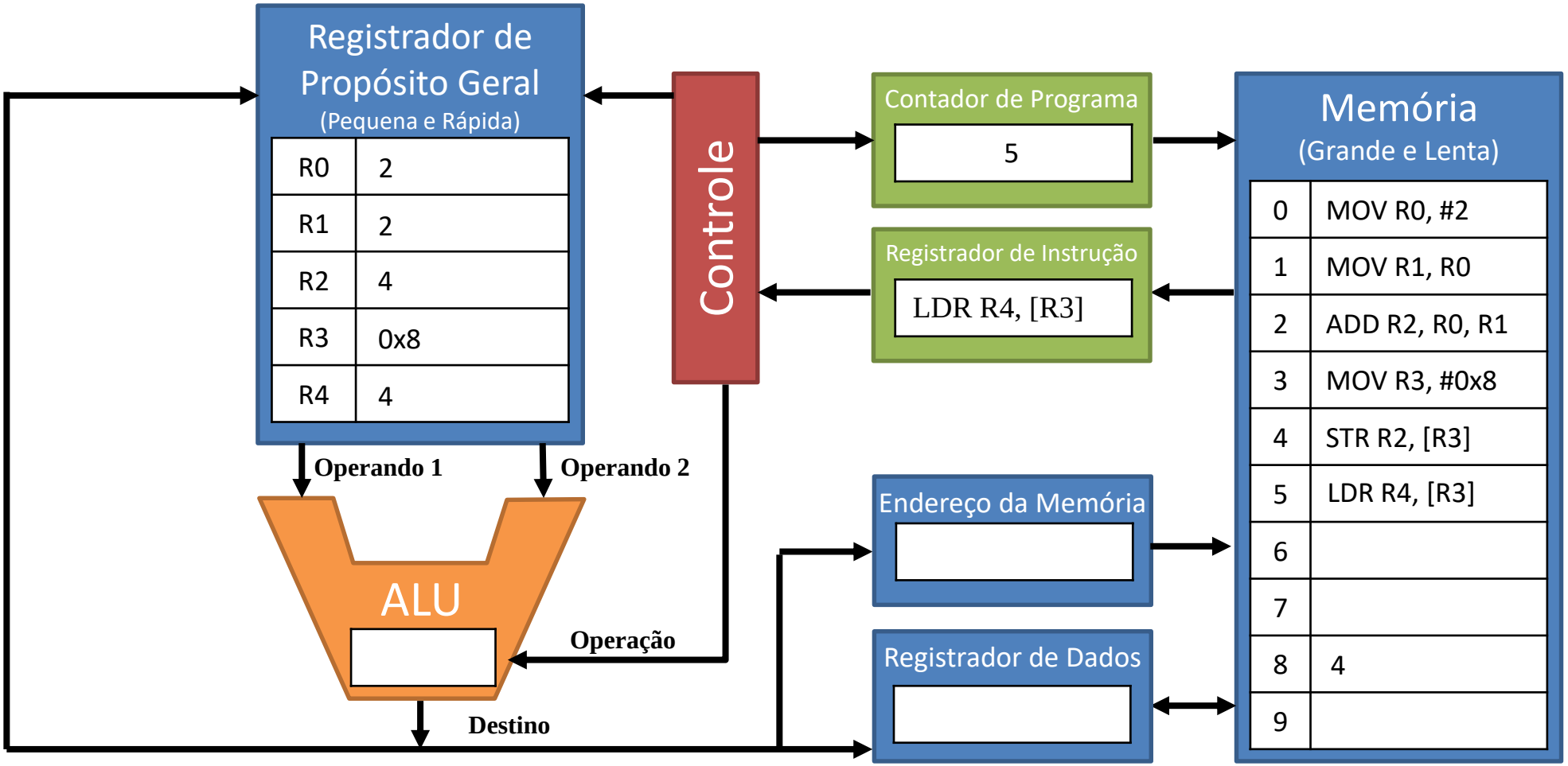
OpCode Valor Endereço



Arquitetura Simplificada "Load/Store" – ARM Cortex M0+

O que acontece no próximo ciclo?

FALHA!! Não tem instrução para executar!! O processador deve sempre rodar em loop infinito!!



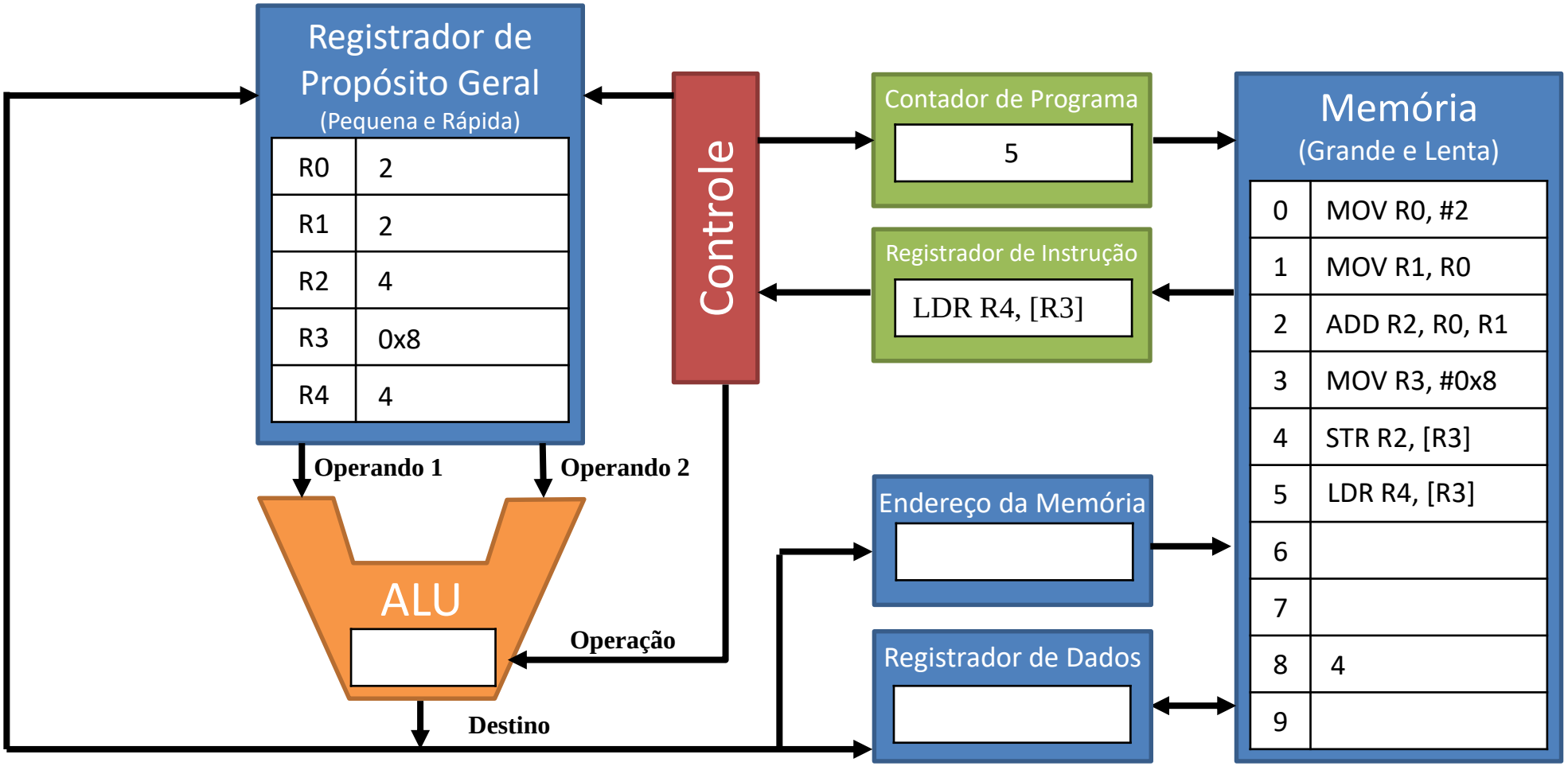
Arquitetura Simplificada "Load/Store" – ARM Cortex M0+

Instrução: LOAD – Carrega o registrador Rn com o valor do endereço da memória indicado pelo registrador Rd.

LDR Rn, [Rd]

OpCode Destino Endereço

Como colocar a posição 8 da memória no R4? Qual é o fluxo?



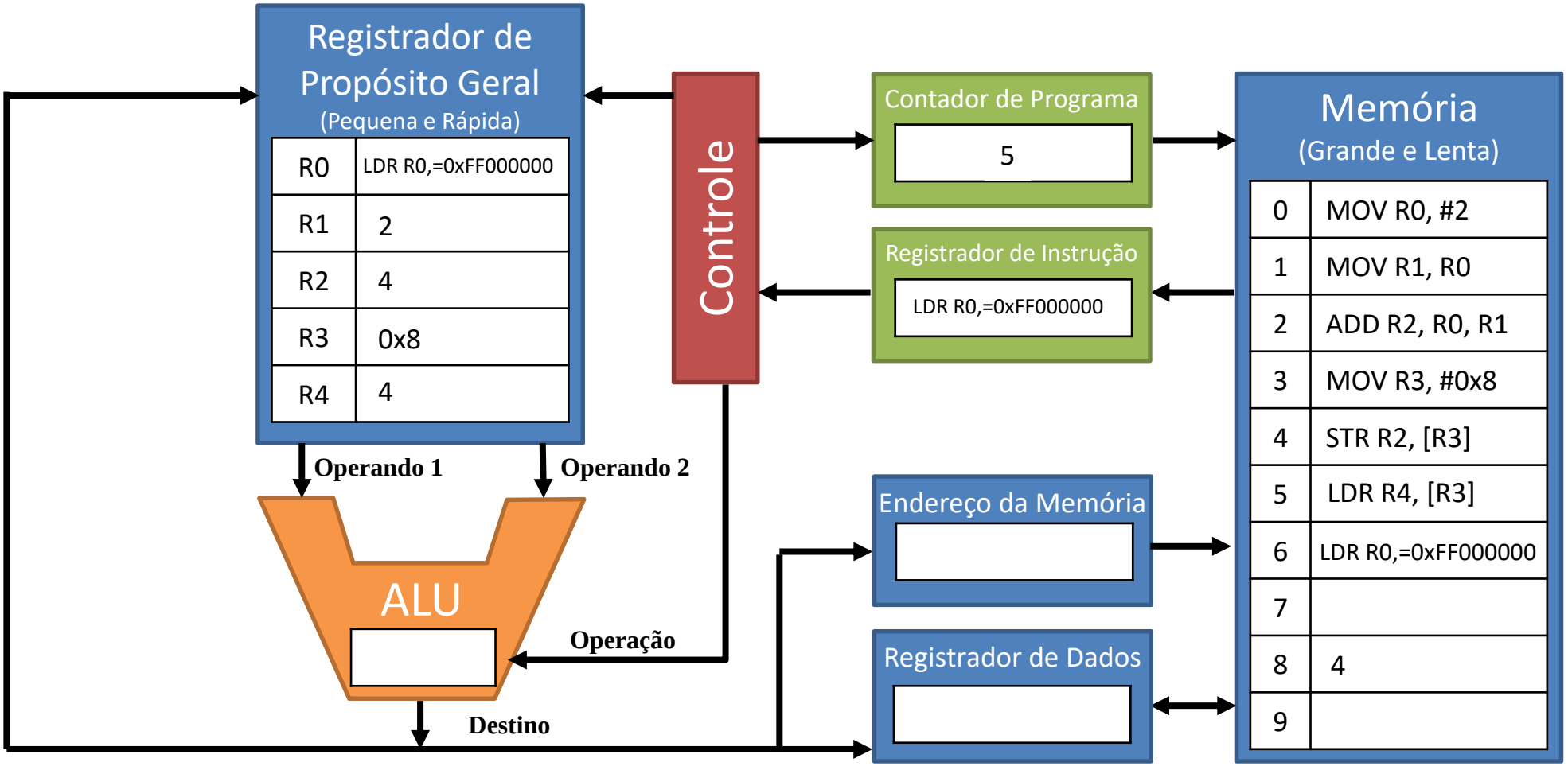
Arquitetura Simplificada "Load/Store" – ARM Cortex M0+

Instrução: LOAD – Carrega o registrador Rn com o valor do endereço da memória indicado pelo registrador Rd.

LDR Rn, [Rd]

OpCode Destino Endereço

LOAD por der usado para colocar endereços de 32 bits nos registradores!

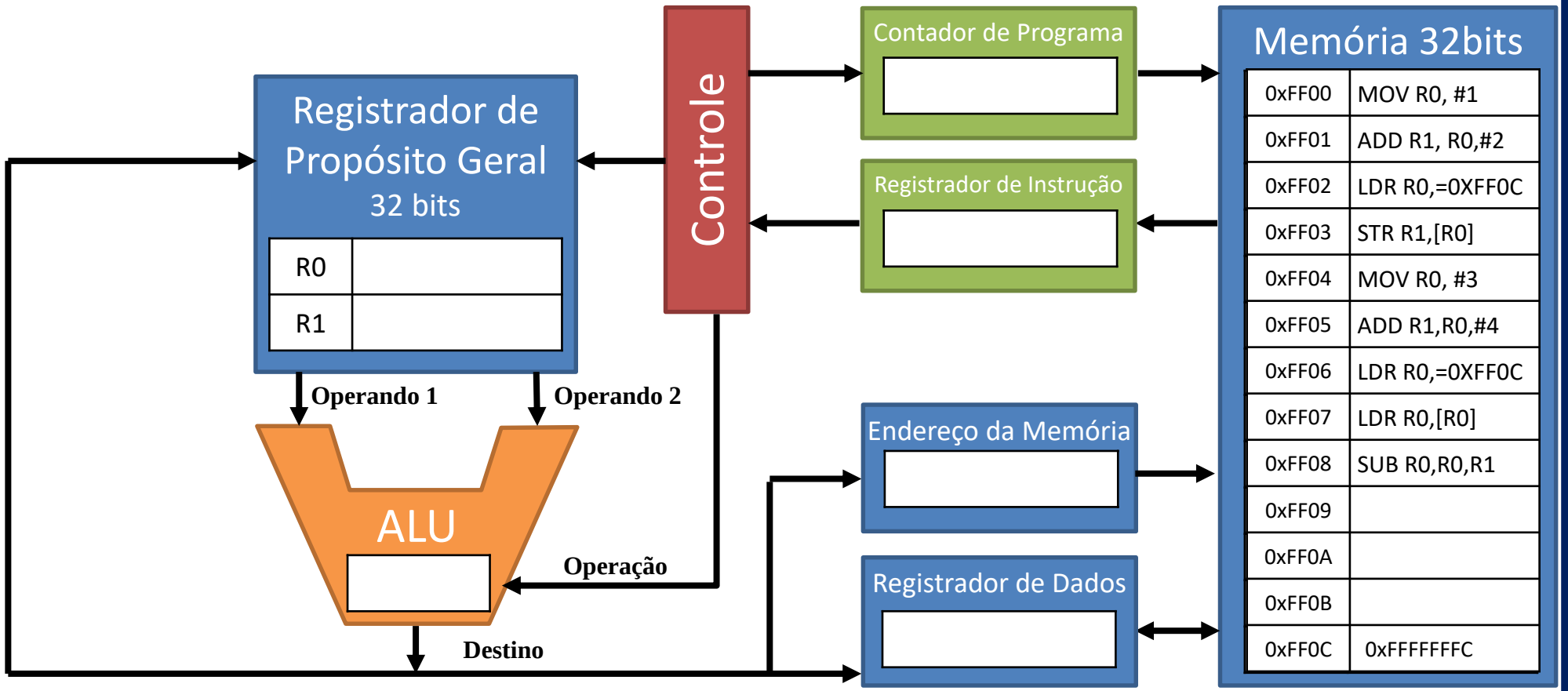


Exercício

Dado o diagrama do processador abaixo, escreva o código para executar a função abaixo e armazene X em 0xFF0C. Qual é o valor de X?

X = (a+b)-(c+d), sendo a = 1, b = 2, c = 3, d = 4

Instrução de subtração: SUB Rd, Rn, Op2 ; Rd=Rn-Op2



Subtração utilizando complemento de 2

X = (a+b)-(c+d), sendo a = 1, b = 2, c = 3, d = 4
X = (1+2)-(3+4) = 3-7 = 3 + (-7)

SUB Rd, Rn, Op2

Procedimento:

1. Tirar o complemento de 2 do subtraendo (Op2) – **inverter os bits e somar 1**
2. Adiciona-lo ao minuendo (Rn)

1. 7 em Hexa expresso em 32 bits — 0000 0007
   Complemento de 2 — FFFF FFF9

2. Este bit indica que é negativo

```
  0000 0003
+ FFFF FFF9
  FFFF FFFC = -4
```

FFFF FFFC → 1111 1111 1111 1111 1111 1111 1111 1100
Complemento de 2 → 000 0000 0000 0000 0000 0000 0000 0100 → 4

O que foi visto na aula