

Programação em Baixo Nível

C, Assembly e execução de programas
na arquitetura Intel 64

Igor Zhirkov

Novatec

Original English language edition published by Manning Publications Co, Copyright © 2017 by Manning Publications. Portuguese-language edition for Brazil copyright © 2018 by Novatec Editora. All rights reserved.

Edição original em inglês publicada pela Manning Publications Co, Copyright © 2017 pela Manning Publications. Edição em português para o Brasil copyright © 2018 pela Novatec Editora. Todos os direitos reservados.

Copyright © 2018 da Novatec Editora Ltda.

Todos os direitos reservados e protegidos pela Lei 9.610 de 19/02/1998. É proibida a reprodução desta obra, mesmo parcial, por qualquer processo, sem prévia autorização, por escrito, do autor e da Editora.

Editor: Rubens Prates

Tradução: Lúcia A. Kinoshita

Revisão gramatical: Tássia Carvalho

Editoração eletrônica: Carolina Kuwabata

ISBN: 978-85-7522-667-4

Histórico de impressões:

Abril/2018 Primeira edição

Novatec Editora Ltda.

Rua Luís Antônio dos Santos 110

02460-000 – São Paulo, SP – Brasil

Tel.: +55 11 2959-6529

Email: novatec@novatec.com.br

Site: www.novatec.com.br

Twitter: twitter.com/novateceditora

Facebook: facebook.com/novatec

LinkedIn: linkedin.com/in/novatec

LIS20180403

Sumário

Sobre o autor	15
Sobre o revisor técnico.....	16
Agradecimentos.....	17
Introdução	19
Parte I • Linguagem Assembly e arquitetura de computadores.....	23
Capítulo 1 • Básico sobre arquitetura de computadores.....	24
1.1 Arquitetura do núcleo	24
1.1.1 Modelo de computação	24
1.1.2 Arquitetura de von Neumann.....	25
1.2 Evolução	27
1.2.1 Desvantagens da arquitetura de von Neumann.....	27
1.2.2 Arquitetura Intel 64.....	28
1.2.3 Extensões da arquitetura	29
1.3 Registradores	31
1.3.1 Registradores de propósito geral.....	33
1.3.2 Outros registradores	37
1.3.3 Registradores de sistema.....	38
1.4 Anéis de proteção.....	39
1.5 Pilha de hardware	40
1.6 Resumo	42
Capítulo 2 • Linguagem Assembly	44
2.1 Configurando o ambiente.....	44
2.1.1 Trabalhando com os códigos de exemplo.....	45
2.2 Escrevendo “Hello, world”	45
2.2.1 Entrada e saída básicas	45
2.2.2 Estrutura do programa	47

2.2.3 Instruções básicas	49
2.3 Exemplo: exibindo o conteúdo de registradores	51
2.3.1 Rótulos locais	53
2.3.2 Endereçamento relativo	54
2.3.3 Ordem de execução	55
2.4 Chamadas de função.....	56
2.5 Trabalhando com dados.....	60
2.5.1 Endianness.....	60
2.5.2 Strings.....	63
2.5.3 Pré-processamento de constantes.....	63
2.5.4 Ponteiros e diferentes tipos de endereçamento	63
2.6 Exemplo: calculando o tamanho de uma string	65
2.7 Exercício: biblioteca de entrada/saída	68
2.7.1 Autoavaliação	70
2.8 Resumo	71
Capítulo 3 • Legado.....	74
3.1 Modo real	74
3.2 Modo protegido	76
3.3 Segmentação mínima em modo longo.....	81
3.4 Acessando partes de registradores	83
3.4.1 Um comportamento inesperado	83
3.4.2 CISC e RISC.....	83
3.4.3 Explicação	85
3.5 Resumo	85
Capítulo 4 • Memória virtual	86
4.1 Caching.....	86
4.2 Motivação.....	87
4.3 Espaços de endereçamento	88
4.4 Recursos	89
4.5 Exemplo: acessando um endereço proibido.....	91
4.6 Eficiência	93
4.7 Implementação	94
4.7.1 Estrutura dos endereços virtuais.....	94
4.7.2 Tradução de endereços em detalhes	95
4.7.3 Tamanhos de página	98
4.8 Mapeamento de memória.....	99
4.9 Exemplo: mapeando um arquivo na memória	100
4.9.1 Nomes mnemônicos para constantes	100
4.9.2 Exemplo completo	102
4.10 Resumo.....	104

Capítulo 5 • Pipeline de compilação	106
5.1 Pré-processador.....	107
5.1.1 Substituições simples	107
5.1.2 Substituições com argumentos	109
5.1.3 Substituição condicional simples	110
5.1.4 Condicionais sobre a definição	111
5.1.5 Condicionais sobre a identidade de textos	112
5.1.6 Condicionais sobre o tipo de argumento.....	113
5.1.7 Ordem de avaliação: define, xdefine, assign	114
5.1.8 Repetição.....	116
5.1.9 Exemplo: calculando números primos	117
5.1.10 Rótulos em macros.....	119
5.1.11 Conclusão.....	120
5.2 Tradução	120
5.3 Ligação.....	121
5.3.1 ELF (Executable and Linkable Format)	121
5.3.2 Arquivos-objeto relocáveis	125
5.3.3 Arquivos-objeto executáveis.....	130
5.3.4 Bibliotecas dinâmicas.....	131
5.3.5 Loader.....	137
5.4 Exercício: dicionário.....	139
5.5 Resumo	142
Capítulo 6 • Interrupções e chamadas de sistema	144
6.1 Entrada e saída	144
6.1.1 O registrador TR e o Task State Segment	146
6.2 Interrupções	147
6.3 Chamadas de sistema.....	152
6.3.1 Registradores específicos de modelo.....	153
6.3.2 syscall e sysret	153
6.4 Resumo	155
Capítulo 7 • Modelos de computação	157
7.1 Máquinas de estado finitas	157
7.1.1 Definição	157
7.1.2 Exemplo: paridade de bits	159
7.1.3 Implementação em linguagem Assembly.....	160
7.1.4 Importância prática.....	163
7.1.5 Expressões regulares.....	164
7.2 Máquina de Forth	168
7.2.1 Arquitetura.....	169
7.2.2 Trace de um exemplo de programa em Forth	171

7.2.3 Dicionário	172
7.2.4 Como as palavras são implementadas	172
7.2.5 Compilador	178
7.3 Exercício: compilador e interpretador de Forth.....	179
7.3.1 Dicionário estático e interpretador.....	180
7.3.2 Compilação	184
7.3.3 Forth com bootstrap	185
7.4 Resumo.....	188

Parte II • A linguagem de programação C.....190

Capítulo 8 • Básico da linguagem C..... 191

8.1 Introdução	192
8.2 Estrutura do programa.....	193
8.2.1 Tipos de dados	196
8.3 Controle de fluxo	197
8.3.1 if	197
8.3.2 while	199
8.3.3 for	199
8.3.4 goto.....	201
8.3.5 switch	202
8.3.6 Exemplo: divisor.....	203
8.3.7 Exemplo: é um número de Fibonacci?.....	204
8.4 Instruções e expressões	205
8.4.1 Tipos de instrução	206
8.4.2 Construindo expressões.....	208
8.5 Funções.....	209
8.6 Pré-processador	212
8.7 Resumo	214

Capítulo 9 • Sistema de tipos..... 215

9.1 Sistema básico de tipos em C	215
9.1.1 Tipos numéricos	216
9.1.2 Casting de tipo	218
9.1.3 Tipo booleano	219
9.1.4 Conversões implícitas.....	219
9.1.5 Ponteiros	221
9.1.6 Arrays.....	223
9.1.7 Arrays como argumentos de função.....	225
9.1.8 Inicializadores designados em arrays	226
9.1.9 Aliases de tipo	227
9.1.10 Revendo a função main	228

9.1.11 Operador sizeof.....	229
9.1.12 Tipos const	231
9.1.13 Strings	233
9.1.14 Tipos função	234
9.1.15 Escrevendo um bom código.....	236
9.1.16 Exercício: produto escalar.....	242
9.1.17 Exercício: verificador de número primo	243
9.2 Tipos com tags.....	243
9.2.1 Estruturas	243
9.2.2 Uniões.....	246
9.2.3 Estruturas e uniões anônimas	248
9.2.4 Enumerações	249
9.3 Tipos de dados nas linguagens de programação	250
9.3.1 Espécies de tipagem	250
9.3.2 Polimorfismo	253
9.4 Polimorfismo in C	254
9.4.1 Polimorfismo paramétrico	255
9.4.2 Inclusão.....	257
9.4.3 Sobrecarga.....	258
9.4.4 Coerções.....	260
9.5 Resumo	260
Capítulo 10 • Estrutura do código	262
10.1 Declarações e definições.....	262
10.1.1 Declarações de função.....	263
10.1.2 Declarações de estrutura	265
10.2 Acessando código a partir de outros arquivos	266
10.2.1 Funções de outros arquivos.....	266
10.2.2 Dados em outros arquivos.....	268
10.2.3 Arquivos de cabeçalho	270
10.3 A biblioteca-padrão	272
10.4 Pré-processador.....	275
10.4.1 Guarda de inclusão	277
10.4.2 Por que o pré-processador é maléfico?.....	279
10.5 Exemplo: soma de um array dinâmico	281
10.5.1 Uma espiada na alocação dinâmica de memória.....	281
10.5.2 Exemplo	282
10.6 Exercício: lista ligada.....	283
10.6.1 Exercício.....	283
10.7 Palavra reservada static.....	286
10.8 Ligação	287
10.9 Resumo	288

Capítulo 11 • Memória.....	290
11.1 Revendo os ponteiros	290
11.1.1 Por que precisamos de ponteiros?.....	290
11.1.2 Aritmética de ponteiros	291
11.1.3 Tipo <code>void*</code>	293
11.1.4 <code>NULL</code>	293
11.1.5 Uma palavrinha sobre <code>ptrdiff_t</code>	294
11.1.6 Ponteiros de função	295
11.2 Modelo de memória	297
11.2.1 Alocação de memória	298
11.3 Arrays e ponteiros	301
11.3.1 Detalhes da sintaxe	302
11.4 Strings literais	303
11.4.1 String Interning.....	306
11.5 Modelos de dados	306
11.6 Streams de dados	308
11.7 Exercício: funções de alta ordem e listas.....	312
11.7.1 Funções de alta ordem comuns	312
11.7.2 Exercício	313
11.8 Resumo.....	315
Capítulo 12 • Sintaxe, semântica e pragmática.....	317
12.1 O que é uma linguagem de programação?.....	317
12.2 Sintaxe e gramáticas formais	318
12.2.1 Exemplo: números naturais.....	320
12.2.2 Exemplo: aritmética simples	321
12.2.3 Descida recursiva	322
12.2.4 Exemplo: aritmética com prioridades	326
12.2.5 Exemplo: linguagem imperativa simples.....	327
12.2.6 Hierarquia de Chomsky.....	328
12.2.7 Árvore de sintaxe abstrata.....	329
12.2.8 Análise lexical.....	330
12.2.9 Resumo sobre o parsing.....	330
12.3 Semântica	330
12.3.1 Comportamento indefinido.....	331
12.3.2 Comportamento não especificado	334
12.3.3 Comportamento definido pela implementação	334
12.3.4 Pontos de sequência	335
12.4 Pragmática.....	336
12.4.1 Alinhamento.....	336
12.4.2 Preenchimento de estruturas de dados.....	337
12.5 Alinhamento no C11	341
12.6 Resumo	342

Capítulo 13 • Boas práticas de programação.....344

13.1 Fazendo escolhas	344
13.2 Elementos do código	346
13.2.1 Nomenclatura em geral	346
13.2.2 Estrutura de arquivos	347
13.2.3 Tipos.....	347
13.2.4. Variáveis.....	349
13.2.5 Sobre variáveis globais	350
13.2.6 Funções.....	351
13.3 Arquivos e documentação	351
13.4 Encapsulamento.....	353
13.5 Imutabilidade.....	358
13.6 Asserções.....	359
13.7 Tratamento de erros	359
13.8 Sobre a alocação de memória.....	363
13.9 Sobre a flexibilidade	363
13.10 Exercício: rotação de imagem.....	365
13.10.1 Formato de arquivo BMP.....	366
13.10.2 Arquitetura	367
13.11 Exercício: alocador de memória personalizado	369
13.12 Resumo.....	373

Parte III • Entre o C e o Assembly374**Capítulo 14 • Detalhes sobre a tradução..... 375**

14.1 Sequência de chamadas de função	375
14.1.1 Registradores XMM	375
14.1.2 Convenção de chamadas	376
14.1.3 Exemplo: uma função simples e sua pilha	379
14.1.4 Zona vermelha	382
14.1.5 Número variável de argumentos	383
14.1.6 vprintf e seus companheiros	384
14.2 volatile	385
14.2.1 Alocação de memória em modo preguiçoso	386
14.2.2 Código gerado	387
14.3 Jumps não locais – setjmp	388
14.3.1 volatile e setjmp	390
14.4 inline	394
14.5 restrict	395
14.6 Aliasing rigoroso	398
14.7 Problemas de segurança.....	399
14.7.1 Stack buffer overrun (transbordamento de buffer na pilha)	399

14.7.2 return-to-libc	400
14.7.3 Vulnerabilidades de formatação de saída.....	401
14.8 Mecanismos de proteção	403
14.8.1 Cookie de segurança	404
14.8.2 Address Space Layout Randomization (Randomização do Espaço de Endereçamento).....	404
14.8.3 DEP.....	404
14.9 Resumo.....	405
Capítulo 15 • Objetos compartilhados e modelos de código	407
15.1 Carga dinâmica.....	407
15.2 Relocações e PIC.....	410
15.3 Exemplo: biblioteca dinâmica em C.....	410
15.4 GOT e PLT	412
15.4.1 Acessando variáveis externas.....	412
15.4.2 Chamando funções externas	415
15.4.3 Exemplo de PLT.....	418
15.5 Pré-carga	420
15.6 Resumo sobre o endereçamento de símbolos.....	422
15.7 Exemplos	422
15.7.1 Chamando uma função	423
15.7.2 Sobre vários linkers dinâmicos	425
15.7.3 Acessando uma variável externa	426
15.7.4 Exemplo completo em Assembly.....	428
15.7.5 Misturando C e Assembly	429
15.8 Quais objetos são ligados?.....	431
15.9 Otimizações.....	435
15.10 Modelos de código	439
15.10.1 Modelo de código pequeno (sem PIC)	441
15.10.2 Modelo de código grande (sem PIC).....	442
15.10.3 Modelo de código médio (sem PIC)	443
15.10.4 Modelo de código pequeno com PIC.....	444
15.10.5 Modelo de código grande com PIC	445
15.10.6 Modelo de código médio com PIC	448
15.11 Resumo.....	451
Capítulo 16 • Desempenho	453
16.1 Otimizações.....	453
16.1.1 Mito sobre linguagens rápidas	454
16.1.2 Conselhos gerais	455
16.1.3 Omitindo o ponteiro do stack frame	457
16.1.4 Recursão de cauda.....	459
16.1.5 Eliminação de subexpressões comuns.....	463

16.1.6 Propagação de constantes.....	464
16.1.7 Otimização de valores de retorno (nomeados).....	466
16.1.8 Influência da previsão de desvio.....	469
16.1.9 Influência das unidades de execução.....	470
16.1.10 Agrupando leituras e escritas no código.....	472
16.2 Caching.....	472
16.2.1 Como usar o cache de modo eficaz?.....	472
16.2.2 Prefetching.....	474
16.2.3 Exemplo: busca binária com prefetching.....	475
16.2.4 Ignorando o cache.....	479
16.2.5 Exemplo: inicialização de matriz.....	480
16.3 Classe de instruções SIMD.....	483
16.4 Extensões SSE e AVX.....	484
16.4.1 Exercício: filtro sépia.....	487
16.5 Resumo.....	492
Capítulo 17 • Multithreading.....	494
17.1 Processos e threads.....	494
17.2 O que deixa o multithreading complicado?.....	495
17.3 Ordem de execução.....	496
17.4 Modelos de memória fortes e fracos.....	498
17.5 Exemplo de reordenação.....	499
17.6 O que é e o que não é volátil.....	502
17.7 Barreiras de memória.....	503
17.8 Introdução à pthreads.....	505
17.8.1 Quando usar multithreading.....	506
17.8.2 Criando threads.....	507
17.8.3 Gerenciando threads.....	512
17.8.4 Exemplo: fatoração distribuída.....	513
17.8.5 Mutexes.....	518
17.8.6 Deadlocks.....	522
17.8.7 Livelocks.....	523
17.8.8 Variáveis de condição.....	525
17.8.9 Spinlocks.....	528
17.9 Semáforos.....	530
17.10 Quão robusto é o Intel 64?.....	533
17.11 O que é uma programação sem lock?.....	537
17.12 Modelo de memória do C11.....	540
17.12.1 Visão geral.....	540
17.12.2 Atômicos.....	541
17.12.3 Ordenações de memória no C11.....	542
17.12.4 Operações.....	544
17.13 Resumo.....	546

Apêndice A • Usando o gdb.....	549
Apêndice B • Usando o Make	558
Apêndice C • Chamadas de sistema	564
Apêndice D • Informações sobre testes de desempenho	571
Bibliografia.....	574