



Getting Started

Pedro Garcia

<http://www.sawp.com.br>

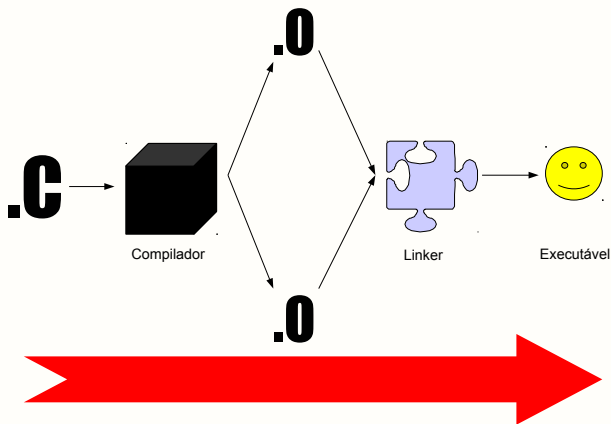
22 de Janeiro de 2010

Assembly Working Party

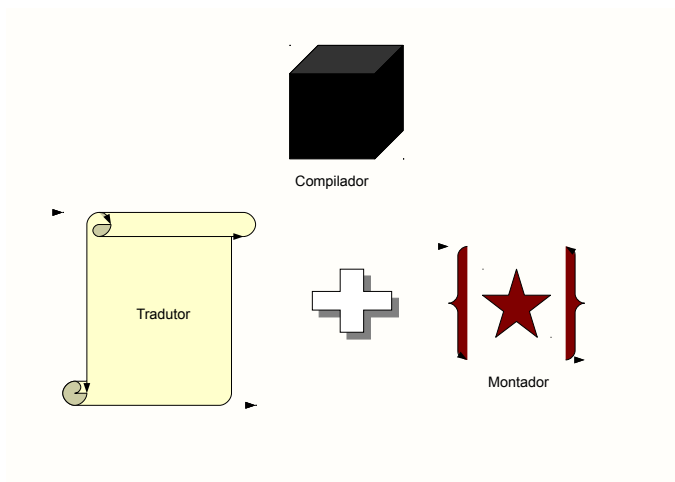
Tópicos

- 1 **Motivação**
- 2 Assembly
- 3 Assembler
- 4 GNU
- 5 Objdump
- 6 Objdump e gcc
- 7 Conclusões

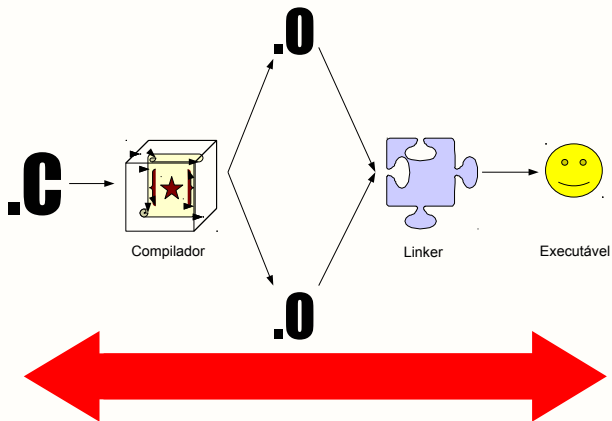
Compiladores como uma caixa-preta



O Compilador



O Programador



Tópicos

- 1 Motivação
- 2 Assembly**
- 3 Assembler
- 4 GNU
- 5 Objdump
- 6 Objdump e gcc
- 7 Conclusões

ASM como linguagem: “O último nível.”

- Conjunto de caracteres, símbolos e regras que representam cada instrução processável pelo computador.
- O **Assembler** (montador) que converte as instruções representadas pela linguagem em código de máquina (binário).
- Não é padronizada: AT&T, Intel, HLA.
- Específica de arquitetura, assembler, ideologia, etc.
- Formada basicamente por:
 - Opcode + modificador de instrução (lógica).
 - Seção de dados.
 - Diretivas (Macros).

Maior fator de incompatibilidade.

Tópicos

- 1 Motivação
- 2 Assembly
- 3 Assembler**
- 4 GNU
- 5 Objdump
- 6 Objdump e gcc
- 7 Conclusões

Nossa maior ferramenta.



As source.s -o source.o

Assemblers

Em ordem crescente de maior importância (para o contexto desse trabalho).

- **HLA:** High Level Assembly
 - Randall Hyde.
 - “The Art of Assembly Language”.
 - Propósito didático.
 - Extenso uso de diretivas: fator “high level”.
 - Hyde está para HLA assim como Guido van Rossum está para o Python.
- **MASM:** Microsoft Assembler
 - Ferramenta importante para quem pretende ganhar algum dinheiro com programação.
 - Extensa documentação, livros e suporte.
 - Atualmente é parte da suite de desenvolvimento do MVS.
 - MASM32: <http://www.masm32.com>

Assemblers

- **NASM:** Netwide Assembler
 - Usado pelo Ladeira: Software Básico. (UnB)
 - Desenvolvido inicialmente como software **comercial** para UNIX.
 - Atualmente é Opensource e portado para diversos S.Os.
 - <http://www.nasm.sourceforge.net>.
- **GAS:** GNU Assembler
 - Ferramenta da suite GNU.
 - Extensa documentação.
 - Portado para diversos S.Os, favorecendo a portabilidade.
 - Permite *cross-compilation*.

Tópicos

- 1 Motivação
- 2 Assembly
- 3 Assembler
- 4 GNU**
- 5 Objdump
- 6 Objdump e gcc
- 7 Conclusões

Por que usarmos as ferramentas da GNU?

- 1 Ambiente “real”:
 - mesmo montador usado pelo **gcc**.
 - o aprendizado “servirá para alguma coisa” (na prática ASM é utilizado de forma embutida em código C/C++).
 - os programas do pacote *binutils* são usados por diversos profissionais.
- 2 Software livre.
- 3 Presente no cotidiano do LCC.
- 4 ASM gerado pelo **gcc** segue a sintaxe AT&T.

Ferramental

gcc

```
$ gcc sample1.c -o sample  
$ gcc -c sample1.c  
$ gcc -S sample2.c -o sample.s  
$ gcc -gstabs+ -pg -O source.c -o binary.o
```

gas

```
$ as sample1.s -o sample.o  
$ as -statistics sample1.s -o sample.o  
$ as -gstabs+ source.s -o binary.o
```

Ferramental

linker

```
$ ld source.o -o program.exe  
$ ld -shared source.o -o program.exe  
$ ld source1.o source2.o source3.o -o program.exe
```

gdb

```
$ gcc -g source.c -o program.exe  
$ gdb program.exe
```

objdump

```
$ gcc -c source.c -o source.o  
$ objdump -d source.o $ objdump -d -S -G source.o
```

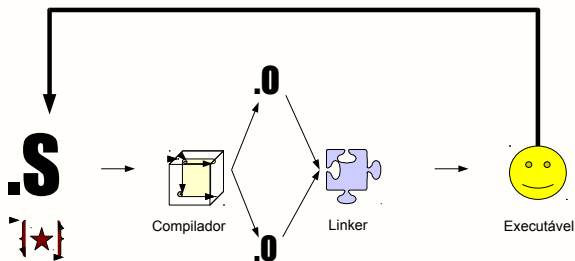
GDB e Objdump

- Fundamentais para analisarmos o comportamento do programa.
- **CIC e FIS:** funções de I/O são usadas para extrair informações de *debug*. (Já houveram relatos de pessoas que depuraram seus programas utilizando Microsoft Excel 2003).
- **AWP:** usaremos GDB e Objdump para verificar o estado da variável após certa execução do nosso programa (fazendo papel das funções de IO).

Tópicos

- 1 Motivação
- 2 Assembly
- 3 Assembler
- 4 GNU
- 5 Objdump**
- 6 Objdump e gcc
- 7 Conclusões

Objdump: Usando o disassembler



objdump -d program.o

Objdump: Parâmetros fundamentais.

- **-d**: Desassembla o código compilado: 1010011010 → *ASM*
- **-G**: Exibe conteúdo das seções de *debug*. Este recurso também costuma ser usado para obter informações adicionais do programa de código fechado. As vezes, o programador descuidado esquece de recompilar o programa sem informação de depuração antes de distribuí-lo.
- **-S**: Exibe código fonte misturado com código desassemblado.
- **-start-address** e **-stop-address**: Indica um endereço específico de início e fim para o processo de *dump*.

Objdump: Exemplo.

Seja compilar o seguinte programa:

source.c

```
1 int main(void) {  
2     int i, j, k;  
3  
4     i = 3;  
5     j = 8;  
6     k = j << i;  
7  
8     return 0;  
9 }
```

Objdump: Exemplo.

objdump

```
$ gcc -c hw.c -o hw
```

```
$ objdump -d -f hw
```

Note que o parâmetro **-c** não foi passado para o compilador, o objdump irá desassemblar todo código “linkado”, inclusive das bibliotecas compartilhadas.

Objdump: Saída

```
$ objdump -d -f hw
```

```
hw:      file format elf32-i386
architecture: i386, flags 0x00000010:
HAS_SYMS
start address 0x00000000
```

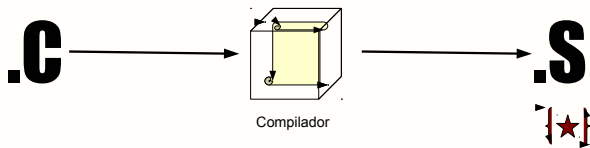
Disassembly of section .text:

```
00000000 <main>:
 0: 55                push    %ebp
 1: 89 e5             mov     %esp,%ebp
 3: 53                push    %ebx
 4: 83 ec 10          sub     $0x10,%esp
 7: c7 45 f8 03 00 00 00  movl   $0x3,-0x8(%ebp)
 e: c7 45 f4 08 00 00 00  movl   $0x8,-0xc(%ebp)
15: 8b 45 f8           mov     -0x8(%ebp),%eax
18: 8b 55 f4           mov     -0xc(%ebp),%edx
1b: 89 d3             mov     %edx,%ebx
1d: 89 c1             mov     %eax,%ecx
1f: d3 e3             shl     %cl,%ebx
21: 89 d8             mov     %ebx,%eax
23: 89 45 f0           mov     %eax,-0x10(%ebp)
26: b8 00 00 00 00    mov     $0x0,%eax
2b: 83 c4 10          add     $0x10,%esp
2e: 5b                pop     %ebx
2f: 5d                pop     %ebp
30: c3                ret
```

Tópicos

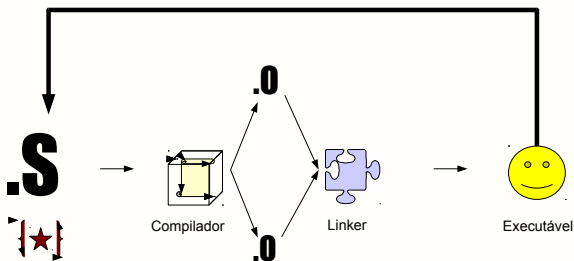
- 1 Motivação
- 2 Assembly
- 3 Assembler
- 4 GNU
- 5 Objdump
- 6 Objdump e gcc**
- 7 Conclusões

Convertendo **código-fonte** em assembly.



`gcc -S source.c -o source.s`

Convertendo código binário em assembly.



objdump -d program.o

Instruções geradas pelo gcc

```
$ gcc -S hw.c -o hw.s
```

```
$ more hw.s
```

```
.file    "hw.c"  
.text  
.globl  main  
.type   main, @function  
main:  
    pushl   %ebp  
    movl   %esp, %ebp  
    pushl   %ebx  
    subl   $16, %esp  
    movl   $3, -8(%ebp)  
    movl   $8, -12(%ebp)  
    movl   -8(%ebp), %eax  
    movl   -12(%ebp), %edx  
    movl   %edx, %ebx  
    movl   %eax, %ecx  
    sall   %cl, %ebx  
    movl   %ebx, %eax  
    movl   %eax, -16(%ebp)  
    movl   $0, %eax  
    addl   $16, %esp  
    popl   %ebx  
    popl   %ebp  
    ret  
.size   main, .-main
```

Instruções geradas Objdump

```
$ objdump -d -f hw
```

```
hw:      file format elf32-i386
architecture: i386, flags 0x00000010:
HAS_SYMS
start address 0x00000000
```

Disassembly of section .text:

```
00000000 <main>:
 0: 55                      push  %ebp
 1: 89 e5                   mov   %esp,%ebp
 3: 53                      push  %ebx
 4: 83 ec 10                sub   $0x10,%esp
 7: c7 45 f8 03 00 00 00    movl  $0x3,-0x8(%ebp)
 e: c7 45 f4 08 00 00 00    movl  $0x8,-0xc(%ebp)
15: 8b 45 f8                mov   -0x8(%ebp),%eax
18: 8b 55 f4                mov   -0xc(%ebp),%edx
1b: 89 d3                   mov   %edx,%ebx
1d: 89 c1                   mov   %eax,%ecx
1f: d3 e3                   shl   %cl,%ebx
21: 89 d8                   mov   %ebx,%eax
23: 89 45 f0                mov   %eax,-0x10(%ebp)
26: b8 00 00 00 00         mov   $0x0,%eax
2b: 83 c4 10                add   $0x10,%esp
2e: 5b                      pop   %ebx
2f: 5d                      pop   %ebp
30: c3                      ret
```

Note que as instruções geradas por ambas ferramentas são idênticas. Contudo, observamos que a única divergência entre ambos os códigos está na presença ou falta de diretivas do primeiro. Nada mais natural, uma vez que as instruções mostradas pelo Objdump são geradas pelo gcc. Isto mostra que, provavelmente, um outro assembler geraria o mesmo conjunto de instruções, na montagem final. Contudo, sua sintaxe e diretivas poderiam ser totalmente diferentes.

Tópicos

- 1 Motivação
- 2 Assembly
- 3 Assembler
- 4 GNU
- 5 Objdump
- 6 Objdump e gcc
- 7 Conclusões**

- Conhecer ASM é fundamental para aprimorar as habilidades de qualquer programador, uma vez que é o recurso-chave na interface entre *software* e *hardware*.
- Embora existam diversos esforços para o estudo e desenvolvimento de decompiladores, este tipo de aplicação ainda é muito pouco desenvolvido, sendo os desassemblers peça essencial para programadores que desejam estudar o código de um programa funcional.

▶ Voltar ao Início