Software Básico (INF1018)

Noemi Rodriguez
Ana Lúcia de Moura
Raúl Renteria
Alexandre Meslin

http://www.inf.puc-rio.br/~inf1018

Conteúdo do Curso

- Representação de dados
- Manipulação de bits
- Complemento a dois
- Armazenamento de vetores e estruturas
- Linguagem de máquina
- Estruturas de controle
- Chamadas a procedimentos
- Variáveis locais e globais
- Representação de ponto flutuante
- Interrupção
- Link-edição

Material Básico de Referência

Computer Systems, A Programmer's Perspective

Randal Bryant and David O'Hallaron

Slides, vídeos e exercícios no *site* da disciplina

http://www.inf.puc-rio.br/~inf1018

DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Critério de Avaliação

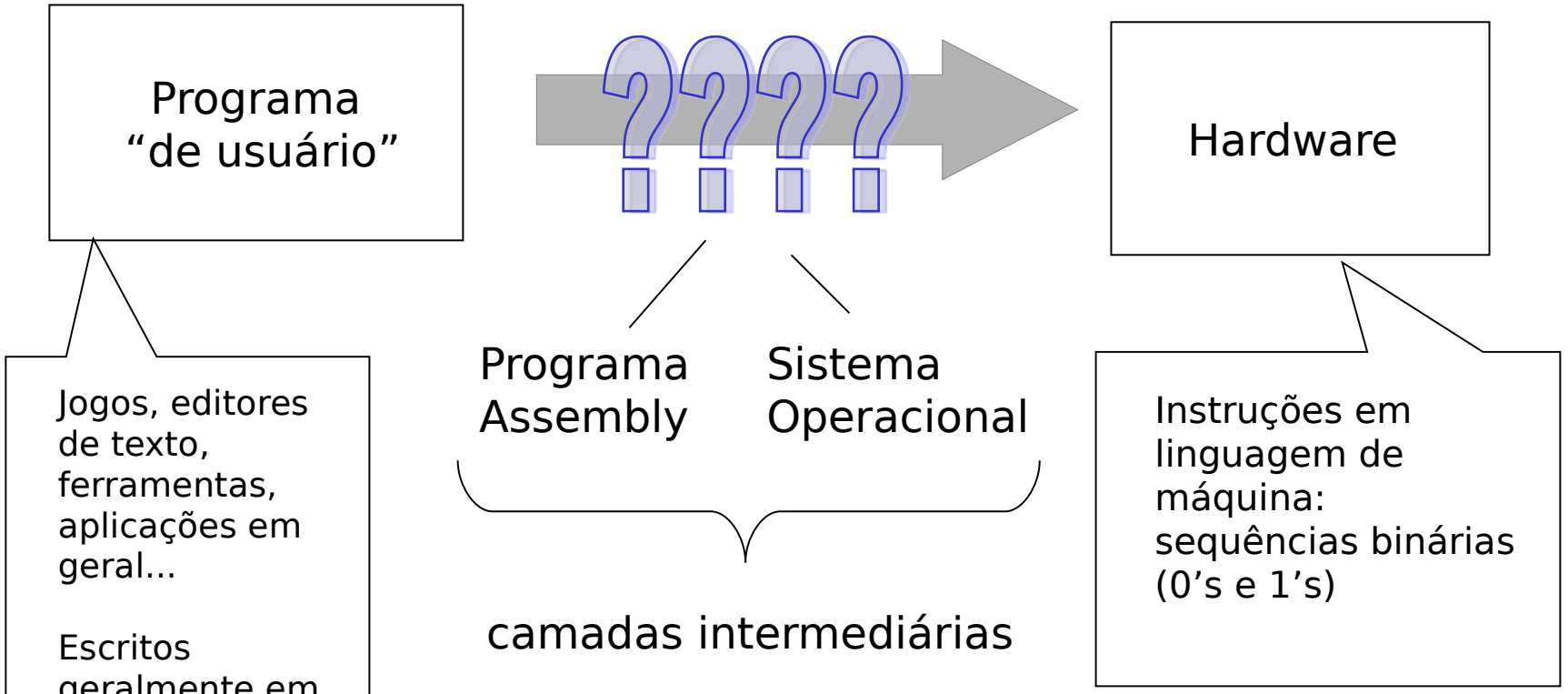
Cada grau é a média geométrica de uma prova (peso 2) e um trabalho (em dupla)

$$G1 = \sqrt[3]{(P1^2 \times T1)}$$
$$G2 = \sqrt[3]{(P2^2 \times T2)}$$

Média = (G1 + G2) / 2

- se **G1 e G2 ≥ 3.0** e **M ≥ 6.0**, é a nota final (NF)
- caso contrário: **NF = (G1 + G2 + 2 x PF) / 4**
- se **NF >= 5.0** o aluno está aprovado

Hierarquia de Abstrações



Objetivo do Curso

Entender como funciona um computador típico, como visto no nível de Linguagem de Montagem

Perspectiva de *software* (foco no **programador**)

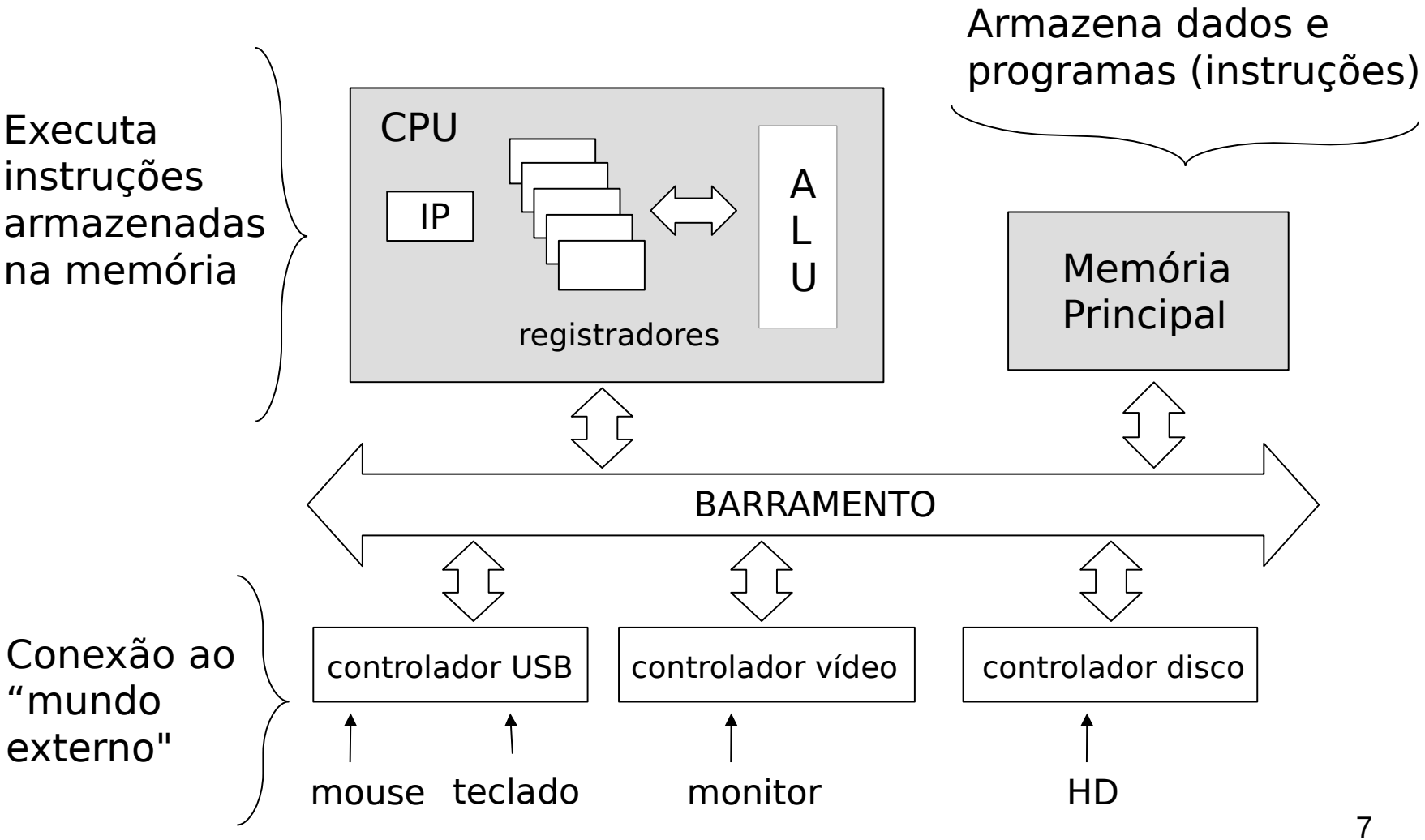
- suporte para abstrações de uma linguagem de programação (C)

Programadores precisam de um entendimento sólido da hierarquia de abstrações de um Sistema de Computação

- otimizar o uso de recursos/desempenho de programas
- entender e saber evitar bugs (representação de dados, manipulação de memória, estouro da pilha...)

porque abstrações vazam !

Arquitetura Típica



Geração de um Executável

code/intro/hello.c

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("hello, world\n");
6 }
```

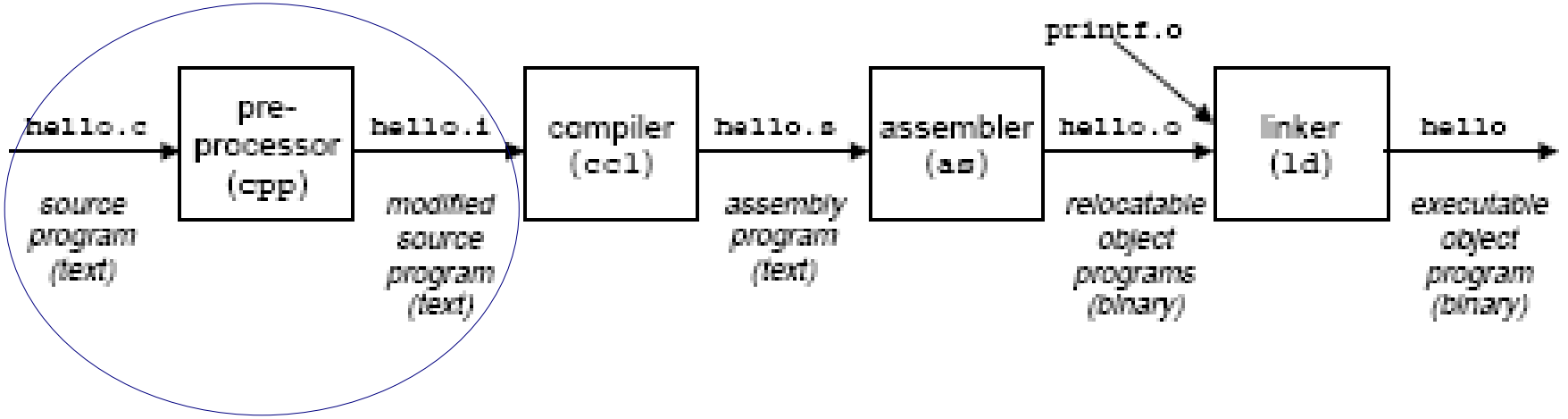
code/intro/hello.c

O programa **fonte** (arquivo texto) deve ser "traduzido" para uma sequência de instruções de linguagem de máquina, que é armazenada em um arquivo binário (**executável**)

➡ essa tradução é realizada em 4 passos

Passo 1: pré-processamento

```
ana@sol:~/inf1018$ gcc -o hello hello.c
```



Modifica o programa fonte C de acordo com as diretivas começadas com #

#include <stdio.h> faz com que o pré-processador leia o arquivo **stdio.h** e o insira no programa fonte

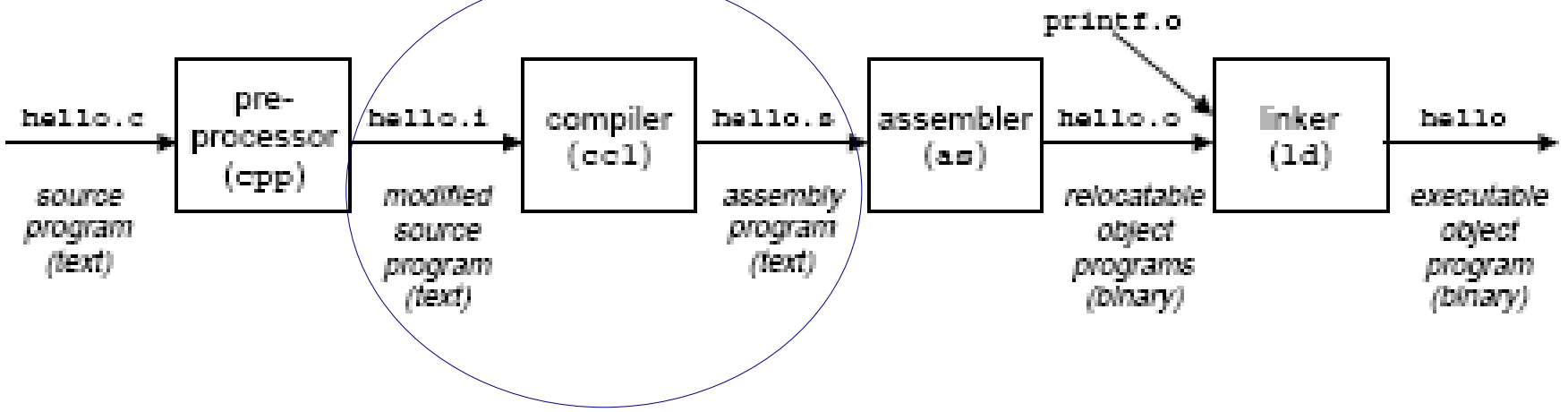
Passo 1: pré-processamento

```
$ gcc -E -o alomundo.i alomundo.c
$ cat alomundo.i
# 1 "alomundo.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 31 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 32 "<command-line>" 2
# 1 "alomundo.c"
# 1 "/usr/include/stdio.h" 1 3 4
# 27 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/libc-header-
start.h" 1 3 4
# 33 "/usr/include/x86_64-linux-gnu/bits/libc-header-
start.h" 3 4
# 1 "/usr/include/features.h" 1 3 4
# 461 "/usr/include/features.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 1 3 4
# 452 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 453 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/long-double.h" 1
3 4
# 454 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 2 3 4
# 462 "/usr/include/features.h" 2 3 4
# 485 "/usr/include/features.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 1 3 4
# 10 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 3 4
...
```

```
...
extern int sys_nerr;
extern const char *const sys_errlist[];
# 782 "/usr/include/stdio.h" 2 3 4
extern int fileno (FILE *__stream) __attribute__
((__nothrow__ , __leaf__)) ;
extern int fileno_unlocked (FILE *__stream) __attribute__
((__nothrow__ , __leaf__)) ;
# 800 "/usr/include/stdio.h" 3 4
extern FILE *popen (const char *__command, const char *__modes)
;
extern int pclose (FILE *__stream);
extern char *ctermid (char *__s) __attribute__ ((__nothrow__ ,
__leaf__));
# 840 "/usr/include/stdio.h" 3 4
extern void flockfile (FILE *__stream) __attribute__
((__nothrow__ , __leaf__));
extern int ftrylockfile (FILE *__stream) __attribute__
((__nothrow__ , __leaf__)) ;
extern void funlockfile (FILE *__stream) __attribute__
((__nothrow__ , __leaf__));
# 858 "/usr/include/stdio.h" 3 4
extern int __uflow (FILE *);
extern int __overflow (FILE *, int);
# 873 "/usr/include/stdio.h" 3 4
# 2 "alomundo.c" 2
# 3 "alomundo.c"
int main(void) {
 printf("Alô mundo!\n");
 return 0;
}
```

Passo 2: compilação

```
ana@sol:~/inf1018$ gcc -o hello hello.c
```



Traduz o programa fonte modificado para um programa em linguagem de montagem (**assembly**)

- é um formato de saída comum para os compiladores de linguagens de programação de alto nível

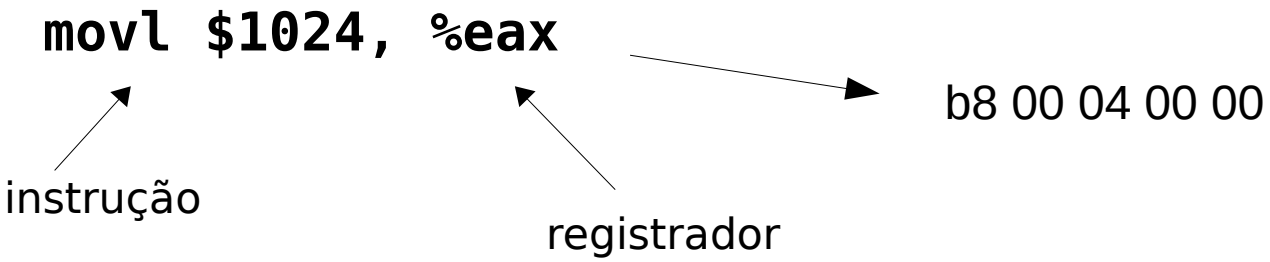
Linguagem de Montagem

É um mapeamento bastante direto para a linguagem de máquina

- a linguagem de montagem (e de máquina) é específica para uma plataforma
- cada linha do código fonte corresponde a uma instrução para o processador

Tem várias facilidades para um programador

- tipos básicos de dados (inteiros, endereços)
- uso de *mnemônicos* (nomes) para representar instruções, registradores



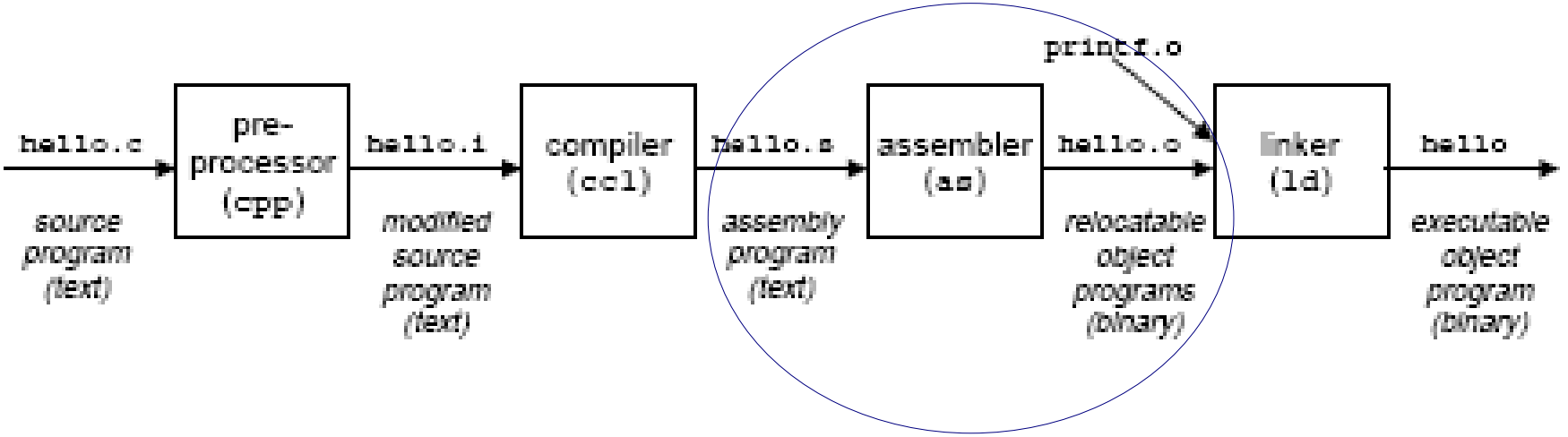
Passo 2: compilação

```
$ gcc -S -o alomundo.s alomundo.c
$ cat alomundo.s
.file "alomundo.c"
.text
.section    .rodata
.LC0:
.string     "Al\303\264 mundo!"
.text
.globl      main
.type main, @function
main:
.LFB0:
.cfi_startproc
endbr64
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq  %rsp, %rbp
.cfi_def_cfa_register 6
leaq    .LC0(%rip), %rdi
movl    $0, %eax
call    printf@PLT
movl  $0, %eax
popq  %rbp
.cfi_def_cfa 7, 8
ret
.cfi_endproc
.LFE0:
.size main, .-main
.ident      "GCC: (Ubuntu 9.3.0-10ubuntu2) 9.3.0"
.section    .note.GNU-stack,"",@progbits
.section    .note.gnu.property,"a"
.align 8
.long  1f - 0f
.long  4f - 1f
.long  5
0:
.string      "GNU"
1:
.align 8
.long  0xc0000002
.long  3f - 2f
2:
.long  0x3
3:
.align 8
4:
```

DEPARTAMENTO DE INFORMÁTICA
PUC·RIO

Passo 3: montagem

```
ana@sol:~/inf1018$ gcc -o hello hello.c
```



Traduz o programa fonte assembly para instruções da linguagem de máquina (**objeto**)

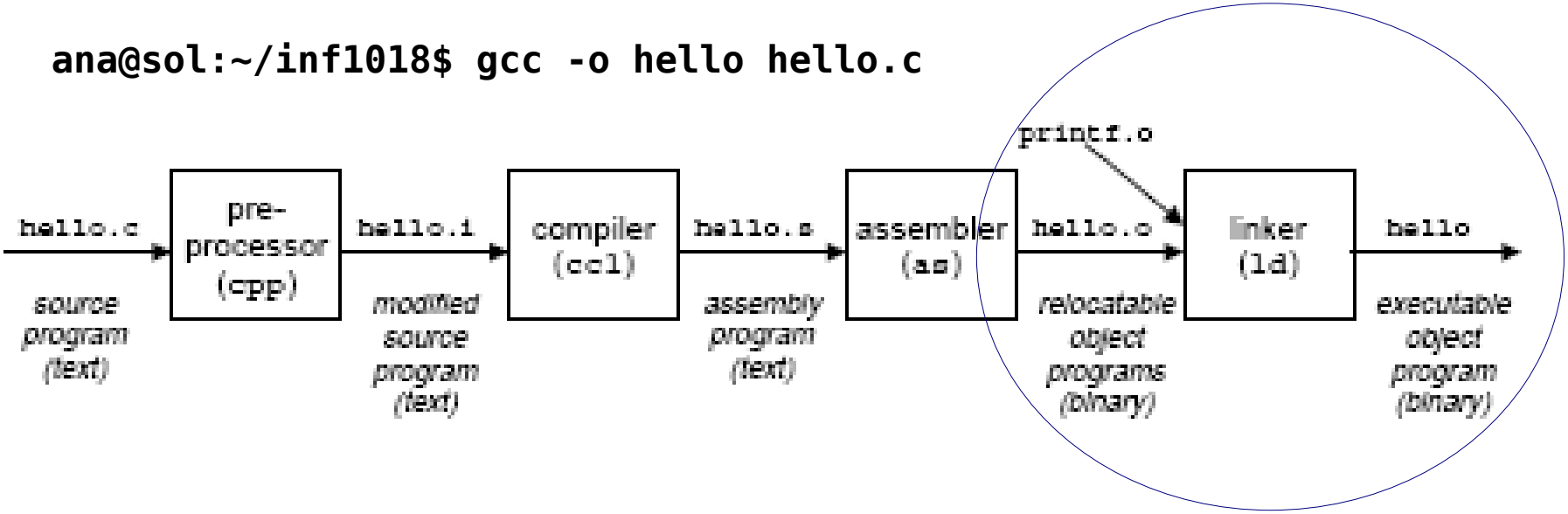
- armazenado como um arquivo **binário** com extensão **.o**

Passo 3: montagem

```
$ gcc -c -o alomundo.o alomundo.c
$ objdump -d alomundo

alomundo.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <main>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   48 89 e5                mov    %rsp,%rbp
   8:   48 8d 3d 00 00 00 00    lea    0x0(%rip),%rdi        # f
<main+0xf>
   f:   b8 00 00 00 00          mov    $0x0,%eax
  14:   e8 00 00 00 00          callq  19 <main+0x19>
  19:   b8 00 00 00 00          mov    $0x0,%eax
  1e:   5d                      pop    %rbp
  1f:   c3                      retq
```

Passo 4: ligação (amarração)

```
ana@sol:~/inf1018$ gcc -o hello hello.c
```



Gera um **executável** a partir do(s) módulo(s) objeto

- uma aplicação pode ser composta por vários arquivos fonte, cada um gerando um módulo objeto diferente
- alguns módulos objeto podem estar armazenados em **bibliotecas**
- o ligador faz a união dos módulos necessários para gerar o executável

Passo 4: ligação (amarração)

```
$ gcc -o alomundo alomundo.c
$ objdump -d alomundo

alomundo:     file format elf64-x86-64
...
0000000000001050 <printf@plt>:
    1050: f3 0f 1e fa           endbr64
    1054: f2 ff 25 75 2f 00 00  bnd jmpq *0x2f75(%rip)        # 3fd0 <printf@GLIBC_2.2.5>
    105b: 0f 1f 44 00 00        nopl   0x0(%rax,%rax,1)
...
0000000000001149 <main>:
    1149: f3 0f 1e fa           endbr64
    114d: 55                    push   %rbp
    114e: 48 89 e5              mov    %rsp,%rbp
    1151: 48 8d 3d ac 0e 00 00  lea    0xeac(%rip),%rdi        # 2004 <_IO_stdin_used+0x4>
    1158: b8 00 00 00 00        mov    $0x0,%eax
    115d: e8 ee fe ff ff        callq  1050 <printf@plt>
    1162: b8 00 00 00 00        mov    $0x0,%eax
    1167: 5d                    pop    %rbp
    1168: c3                    retq
    1169: 0f 1f 80 00 00 00 00  nopl   0x0(%rax)
...
```

Software Básico (INF1018)

Noemi Rodriguez
Ana Lúcia de Moura
Raúl Renteria
Alexandre Meslin

http://www.inf.puc-rio.br/~inf1018