

# Níveis de Abstração

## Guião II

### Apresentação

Este guião serve como base para a apresentação dos vários níveis de abstração e de representação envolvidos no processo de desenvolvimento de programas. Complementarmente pretende-se compreender e utilizar os mecanismos de conversão entre os diferentes níveis.

Para atingir aqueles objetivos, torna-se necessário que os alunos tomem contacto com ferramentas de uso generalizado usadas para a criação e depuração de executáveis. Em particular, serão apresentados os mecanismos de compilação de programas escritos em linguagem **C**, recorrendo aos utilitários **gcc**, **gdb** e **objdump** em ambiente Linux.

### Edição de programas

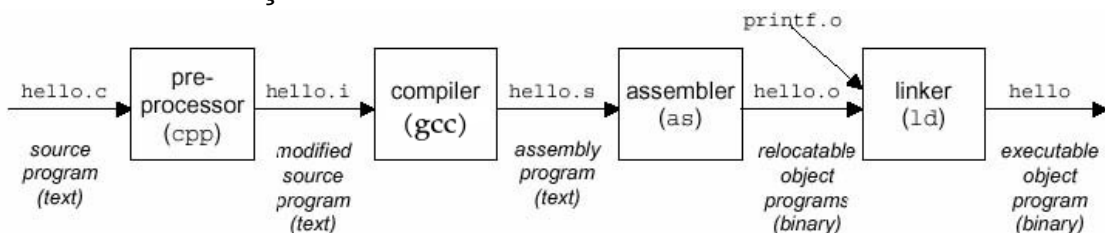
main.c	soma.c
<pre>main () {     int x;      soma(x); }</pre>	<pre>int accum=0;  void soma (int p) {     accum += p; }</pre>

#### Exercício 1

Use o editor de texto mais conveniente para escrever as funções C apresentadas no quadro acima.

- Identifique o formato de representação das funções com o comando: `file <nomeFicheiro>`
- Que relação existe entre aqueles dois módulos?

### Modelo de Transformação



Por **compilação** entende-se a conversão de um programa escrito num dado nível de abstração noutro de nível inferior. Historicamente o termo surgiu da conversão de um programa escrito numa HLL para o nível de “*montagem*”. Contudo, a maior parte dos utilitários atuais conhecidos como “compiladores” permitem, com uma única linha de comando, passar diretamente do nível HLL para o nível da linguagem máquina, executando na realidade 4 programas distintos, correspondentes a 4 fases diferentes: **pré-processamento**, **compilação**, **montagem** (com *assembler*) e **união** (com o *linker*).

#### Exercício 2

- Use o comando `man <util>` para conhecer os utilitários **gcc**, **gdb** e **objdump**.

---

## Compilação

### Exercício 3

Compile os módulos *soma.c* e *main.c* usando o comando:

**\$ gcc -Wall -O2 -S soma.c main.c**

- Identifique as diferentes opções usadas no comando acima.
  - Identifique o formato dos ficheiros produzidos, use o comando: *file <nomeFicheiro>*
  - Visualize os ficheiros acima e interprete o respetivo conteúdo, determinando o nível de abstração implícito.
- 

## Montagem

### Exercício 4

Use o comando

**\$ gcc -Wall -O2 -c soma.s**

- Qual o objetivo de usar a opção *-c* ?
  - Identifique o formato do ficheiro resultante através do comando: *file <nomeFicheiro>*
  - Obteria o mesmo resultado se usasse como entrada o ficheiro *soma.c* ?
  - Qual o nível de abstração implícito?
  - O ficheiro *soma.o* pode ser executado diretamente pela máquina?
- 

## Desmontagem

### Exercício 5

Execute o comando abaixo e visualize o resultado para interpretar o respetivo conteúdo.

**\$ objdump -d soma.o**

- O ficheiro contém informação simbólica?
- Identifique as diferentes representações das instruções presentes.
  - Quanto ao tamanho
  - Quanto ao formato

### Exercício 6

Baseado na visualização do conteúdo do resultado produzido pelo comando acima,

- que relação existe entre a informação observada acima e a presente no ficheiro *soma.s* ?
- identifique o código relativo à representação da variável *accum* ?
  - apresente razões para o modo de representação usado.

### Exercício 7

Execute o comando abaixo (para apoio consulte o anexo *gdb-resumo-comandos*)

**\$ gdb soma.o**

- Identifique o resultado dos seguintes sub-comandos do gdb:
    - *x/23xb soma*
    - *disass soma*
- 

## **Ligação e execução de módulos**

### Exercício 8

Use o comando

**\$ gcc -Wall -O2 -o prog main.c soma.o**

- Identifique o formato do ficheiro resultante, através do comando: *file <nomeFicheiro>*
  - Como procederia para executar o programa implícito?
- 

## **Depuração de programas**

### Exercício 9

Use o comando:

**\$ objdump -d prog**

- Repita o exercício 6 para o resultado obtido pelo comando acima.
- Relacione a representação da variável *accum* com os conceitos de *Little/Big-endian*.

### Exercício 10

Use o comando:

**\$ gdb prog**

- Visualize o resultado da sequência de sub-comandos que se segue:
  - *disass soma*
  - *disass main*
- Identifique no código acima a passagem do controlo à função *soma*?
- Execute o programa, usando o *sub-comando run*.

## Anexo: Introdução ao depurador GDB

O depurador GNU GDB disponibiliza facilidades para a análise e avaliação do funcionamento de programas em linguagem máquina. Permite a execução controlada de programas, com indicação explícita de quando interromper essa execução, através da definição de pontos de paragens e da execução de programas passo-a-passo, e possibilitando ainda a análise do conteúdo de registos e de posições de memória, após cada interrupção.

Comando	Resultado
<b>Starting and Stopping</b>	
<i>quit</i>	Exit GDB
<i>run</i>	Run your program (give command line argument here)
<i>kill</i>	Stop your program
<b>Breakpoints</b>	
<i>break sum</i>	Set breakpoint at entry to function <code>sum</code>
<i>break *0x80483c3</i>	Set breakpoint at address <code>0x80483c3</code>
<i>disable 3</i>	Disable breakpoint 3
<i>enable 2</i>	Enable breakpoint 2
<i>clear sum</i>	Clear any breakpoint at entry to function <code>sum</code>
<i>delete 1</i>	Delete breakpoint 1
<i>delete</i>	Delete all breakpoints
<b>Execution</b>	
<i>stepi</i>	Execute one instruction
<i>stepi 4</i>	Execute four instructions
<i>nexti</i>	Like <i>stepi</i> , but proceed through function calls
<i>continue</i>	Resume execution
<i>finish</i>	Run until current function returns
<b>Examining code</b>	
<i>disas</i>	Disassemble current function
<i>disas sum</i>	Disassemble function <code>sum</code>
<i>disas 0x80483b7</i>	Disassemble function around address <code>0x80483b7</code>
<i>disas 0x80483b7 0x80483c7</i>	Disassemble code within specified address range
<i>print /x \$eip</i>	Print program counter in hex
<b>Examining data</b>	
<i>print \$eax</i>	Print contents of <code>%eax</code> in decimal
<i>print /x \$eax</i>	Print contents of <code>%eax</code> in hex
<i>print /t \$eax</i>	Print contents of <code>%eax</code> in binary
<i>print 0x100</i>	Print decimal representation of <code>0x100</code>
<i>print /x 555</i>	Print hex representation of <code>555</code>
<i>print /x (\$ebp+8)</i>	Print contents of <code>%ebp</code> plus 8 in hex
<i>print *(int *) 0xbffff890</i>	Print integer at address <code>0xbffff890</code>
<i>print *(int *) (\$ebp+8)</i>	Print integer at address <code>%ebp + 8</code>
<i>x/2w 0xbffff890</i>	Examine 2(4-byte) words starting at addr <code>0xbffff890</code>
<i>x/20b sum</i>	Examine first 20 bytes of function <code>sum</code>
<b>Useful information</b>	
<i>info frame</i>	Information about current stack frame
<i>info registers</i>	Values of all the registers
<i>help</i>	Get information about GDB

**Nota:** retirado do livro CSAPP