Níveis de Abstração

TPC4 e Guião laboratorial *(com ajuda...)*

Alberto José Proença

Objetivo geral

Este documento é o **guião** para apoio ao funcionamento da primeira sessão laboratorial de SC. **É indispensável a sua leitura/estudo prévio**, incluindo a resolução e **entrega dos exercícios propostos**.

Objetivo concreto

Assimilar, ao longo de uma sessão laboratorial, os vários **níveis de abstração** envolvidos no processo de desenvolvimento de *software* e respetivas representações usadas em cada nível, bem como os **mecanismos de conversão** entre esses níveis.

Para atingir os objetivos: desenvolver um programa em C, constituído por 2 módulos (em ficheiros distintos), e acompanhar e visualizar as várias fases, **usando as ferramentas do Unix** `gcc`, `gdb` e `objdump`. Para garantir idênticos resultados em todos os trabalhos, **usar sempre** a máquina virtual que se disponibiliza remotamente, em ambiente Unix.

O acesso remoto faz-se com o protocolo `ssh`. Uma vez na máquina remota, usar o ***username*** e ***password*** enviada por correio eletrónico na semana anterior à realização deste trabalho, pela equipa técnica do DI.

Se estiver num ambiente Windows, use a aplicação `PuTTY` para a conetividade com sistemas Unix através do protocolo `ssh`: preencha no campo "`Host`" na janela do `PuTTY` "`sc.di.uminho.pt`" e selecione o botão "Open" para aceder à máquina virtual remota.

Em Unix/Linux ou MacOS usar simplesmente o comando `ssh` para aceder à máquina remota com o seu novo identificador, "*`username`*`@sc.di.uminho.pt`" indicando depois a sua nova ***password***.

---

1. Linguagem de alto nível (*HLL*)

Considere os módulos em C apresentados na tabela, que são para colocar em 2 ficheiros separados:

prog.c	**soma.c**

`void soma (int);`
`int main ()`
`{`
`  int x=2;`
`  soma (x);`
`  return 1;`
`}`
	`int accum=0;`
	
	`void soma (int p)`
	`{`
	`  accum += p;`
	`}`

Exercício 1. (**TPC**) Escolha um editor de texto e crie no servidor remoto os ficheiros **`prog.c`** e **`soma.c`**. Em que formato está representada a informação contida nestes ficheiros?

Tal como já foi sugerido em mensagens anteriores, deverá usar um editor de texto (VI ou VIM) para criar e compor estes ficheiros.

Então se está a usar um "editor de texto", que formato acha que o texto usa para ficar armazenado neste ficheiro?

E nesse formato, quantas células de memória são necessárias para codificar e armazenar cada um dos carateres?

Exercício 2. (**TPC**) Qual o tamanho de cada um dos ficheiros?
(Calcule manualmente **e** verifique com um comando da *shell* do Linux)

Se entendeu a questão anterior, então é fácil calcular manualmente a dimensão de cada um dos ficheiros...

E que comandos do Linux pode usar para saber a dimensão dum ficheiro? Há vários, procure-os e teste!

2. Compilação

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 do *assembly.* Contudo, a maior parte dos utilitários actuais 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 o *assembler*) e união (com o *linker*). Uma descrição mais detalhada destas fases encontra-se no texto que acompanha as aulas teóricas (*Introdução aos Sistemas de Computação*, Cap.3, com material retirado do livro CSAPP).

As diversas versões do manual do compilador de C distribuído pelo projeto GNU, o `gcc`, estão disponíveis *online* em http://www.gnu.org/software/gcc/onlinedocs/, em que a versão disponível na máquina virtual é a 3.2.3. Um sumário muito compacto do manual duma versão do `gcc` é incluído no fim deste guião.

Compile o módulo `soma.c` usando o comando

```
gcc -Wall –O2 –S soma.c
```

O *switch* `Wall` ativa o envio de mensagens de diagnóstico relatando a existência de estruturas de código que poderão conter imprecisões ou potenciais fontes de erro, o *switch* `O2` indica ao compilador para usar o nível dois de otimização do código, enquanto o *switch* `S` indica que deve gerar apenas o código *assembly*. Este comando gera o ficheiro `soma.s`.

Exercício 3. (**TPC**) Em que formato está representada a informação contida neste novo ficheiro?

O ficheiro gerado pelo compilador é um ficheiro com o código do programa em *assembly*.

E porque foi criado essa versão intermédia de código? Foi para mostrar a um ser humano o código binário que a PU vai executar, mas numa forma facilmente compreensível.

E que forma será essa então?...

Exercício 4. Usando um programa adequado visualize o conteúdo de `soma.s`. Encontra texto para além das mnemónicas das operações, nomes de registos do IA-32 ou constantes? Como caracteriza esse texto e com que fim foi lá colocado?

Se já entendeu a questão anterior, então é fácil encontrar um programa adequado para visualizar o ficheiro (um editor de texto, por exemplo).

Daquilo que já viu nas aulas teóricas, em particular o conjunto de instruções em *assembly* do IA-32, então deve certamente encontrar neste ficheiro vários outros caracteres e termos que são distintos das instruções em *assembly* do IA-32.

Que texto extra é esse e que estará lá a fazer? Pense um pouco...

Algumas dicas:

- esse texto extra inclui linhas que começam com um "**.**", que terminam em "**:**" ou ainda texto no meio das instruções em *assembly* que não designam constantes ou endereços nem designações de registos;

- o *assembler* vai precisar de indicações para saber **(i)** se este ficheiro contém o programa todo ou apenas uma ou mais funções do programa, **(ii)** se este ficheiro contém a definição de variáveis globais que o restante do programa precisará de ter acesso, **(iii)** que parte do programa em *assembly* é código e que parte é definição de variáveis globais, para reservar espaço em memória para estas duas componentes, **(iv)** no caso de variáveis globais quantas células de memória irá ocupar cada variável e se necessita de ser inicializada com algum valor, e qual é esse valor, e ainda outras indicações; todas estas estão neste ficheiro em *assembly*.

Exercício 5. (**TPC**) Este programa (o ficheiro `soma.s`) pode ser executado diretamente pela máquina? Em que nível de abstração se encontra?

Se este é apenas um ficheiro de texto, acha que o modo normal de execução de instruções numa PU é indo buscar instruções à memória que estejam codificadas como texto?

Foi assim que trabalhou na peça de teatro?

3. Compilação e montagem (uso do *assembler*)

Use o comando

```
gcc -Wall –O2 –c soma.c
```

para gerar o ficheiro `soma.o` (código binário resultante de compilação e montagem do módulo `soma.c`); o *switch* –c indica que o processo termina após a montagem. O código binário não pode ser visualizado usando um editor de texto, pois o formato da informação já não é ASCII.

Para visualizar o conteúdo de um ficheiro objeto (binário) pode-se usar um ***debugger*** (depurador) fornecido com o Linux. Neste caso, para se iniciar o processo de depuração, far-se-ia:

```
gdb soma.o
```

Uma vez dentro do depurador, pode-se ativar o comando:

```
(gdb) x/23xb soma
```

o qual irá e**x**aminar e mostrar (abreviado "**x**") **23** "he**x**-formatted **b**ytes" (abreviado para "**xb**") a partir do início do código da função **`soma`**.

Exercício 6. O que representam os valores que está a visualizar?

Leu bem o texto em cima que coloquei a cor diferente?

E lembra-se ainda das "instruções" que se foram buscar à memória na peça de teatro?

Estas não se parecem com essas instruções?

Então o que acha que estes valores da memória que está a visualizar representam?

E tem alguma pista porque razão o depurador se recusa a mostrar para além do nº de *bytes* correspondentes ao tamanho do código da função?...

Exercício 7. (**TPC**) Este programa (o ficheiro `soma.o`) pode ser executado diretamente pela máquina? Em que nível de abstração se encontra?

Acha que um ficheiro de código C sem o `main` pode ser considerado um programa?

E não só...

É possível ainda visualizar o código *assembly* a partir do ficheiro objeto, quer dentro do depurador (com o comando **`disassemble soma`**), quer ainda usando um ***disassembler*** (desmontador) do Linux. Este tem a vantagem de mostrar ainda o código binário para além do código *assembly*.

Assim, abandone o `gdb` (com o comando **`quit`**) e execute o comando

```
objdump –d soma.o
```

Exercício 8. O ficheiro **`soma.o`** desmontado, deveria conter apenas linhas de código *assembly* da GNU com instruções ISA-32 e respetivos operandos; contudo, encontra lá ainda outro tipo de informação. Identifique-o e encontre uma explicação para esse texto que lá encontra.

Analise bem o conteúdo deste ficheiro objeto desmontado, i.e., em *assembly* para nós entendermos melhor que código vai a UP executar?

Já reparou que o código da função começa no endereço "`0`" e que a localização da variável `accum` é também no endereço "`0`"? Isto faz sentido?

Este ficheiro necessita ainda de ter uma tabela de símbolos que ainda não foram definidos...

Essa tabela de símbolos pode ser visualizada com `objdump -t soma.o`.

Exercício 9. Como está representada a variável `accum`? Porque razão é ela representada desta forma?

Esta variável deixou de aparecer como uma "*string*": já foi convertida num número, "`0`".

Significa isto que está no endereço "`0`"? Ora tente explicar...

Exercício 10. Quantas instruções tem a função `soma`? Quantos *bytes* ocupa? Quais são as instruções mais curtas e quais as mais longas?

Use o comando `objdump -d` e responda a esta questão.

4. União (uso do *linker*) e execução

Para gerar o programa executável é necessário ligar os dois módulos entre si e com quaisquer outras funções de bibliotecas que sejam utilizadas, assim como acrescentar código que faz o interface com o sistema operativo. Este é o papel do *linker*. Execute o comando

```
gcc -Wall –O2 –o prog prog.c soma.o
```

Exercício 11. O resultado da execução deste comando é colocado no ficheiro `prog`. Qual o formato da informação aí contida? Este ficheiro pode ser executado diretamente pela máquina?

Este já é o ficheiro executável...

Desmonte este programa executável e guarde-o num ficheiro de texto, usando o comando

```
objdump –d prog > prog.dump
```

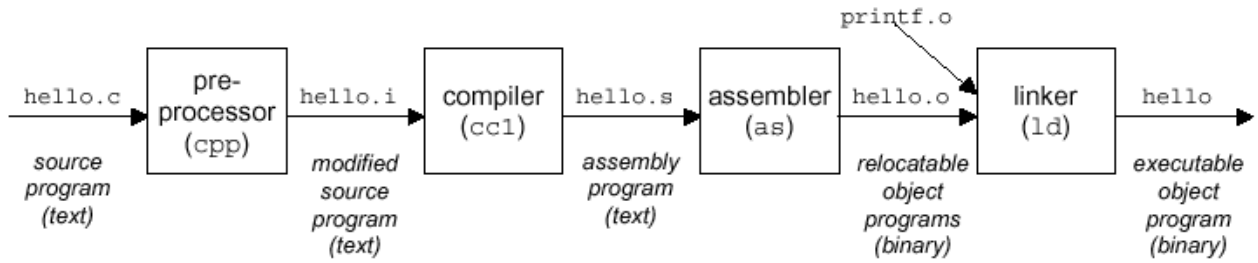
Exercício 12. Localize no ficheiro `prog.dump` a função `soma`. Como está representada a variável `accum`?

Ainda continua representada pelo valor "`0`"?

Exercício 13. Porque ordem são armazenados na memória os 4 *bytes* correspondentes ao endereço de `accum`? *Little-endian* ou *big-endian*? Confirme no ficheiro `prog.dump` (como?).

Exercício 14. Como é que a função `main` passa o controlo (invoca) a função `soma`?

5. Sumário do manual de gcc



GCC(1) GNU Tools GCC(1)

NAME

gcc, *g*++ - GNU project C and *C*++ Compiler (gcc-3.43)

SYNOPSIS

gcc [option | filename]...

DESCRIPTION

The C and *C*++ compiler are integrated. Both process input files through one or more of four stages: preprocessing, compilation, assembly, and linking. Source file-name suffixes identify the source language, but which name you use for the compiler governs default assumptions:

gcc assumes preprocessed (.i) files are C and assumes C style linking.

Suffixes of source file names indicate the language and kind of processing to be done:

.c C source; preprocess, compile, assemble

.i preprocessed C; compile, assemble

.s Assembler source; assemble

.o Object file: pass to the linker.

OPTIONS

Overall Options

-c -S -E -o file -pipe -v -x language

C Language Options

Warning Options

Debugging Options

Optimization Options

Preprocessor Options

Assembler Option

Linker Options

Machine Dependent Options

Nº	**Nome:**	**Turma:**

Resolução dos exercícios

(**Nota**: Apresente sempre os cálculos que efectuar no verso da folha; o não cumprimento desta regra equivale à não entrega do trabalho.)

1. **Crie** os ficheiros `prog.c` e `soma.c` no servidor remoto.

2. **Indique** em que formato está representada a informação contida nos ficheiros `prog.c` e `soma.c`.

3. **Indique** qual o tamanho da cada um dos ficheiros. Calcule manualmente (indique como fez) **e** verifique com um comando da *shell* do Linux (indique o comando usado).

4. **Indique** em que formato está representada a informação contida no ficheiro `soma.s`.

5. **Indique (i)** se o programa no ficheiro `soma.s` pode ser executado diretamente pela máquina (justifique a resposta) e **(ii)** em que nível de abstração se encontra.

7. **Indique (i)** se o programa no ficheiro `soma.o` pode ser executado diretamente pela máquina (justifique a resposta) e **(ii)** em que nível de abstração se encontra.