x86-64
Diretivas *assembly*

Programação em Sistemas Computacionais

João Pedro Patriarca (joao.patriarca@isel.pt), Gabinete F.0.23 do edifício F

ISEL, ADEETC, LEIC

Agenda

- Considerações sobre um ficheiro em *assembly*
- Diretivas que definem secções lógicas
- Diretivas que contribuem para a alocação de espaço
- Outras diretivas

Considerações sobre um ficheiro em *assembly*

- A extensão de um ficheiro em *assembly* em ambiente Linux é `.s`
- O compilador de *assembly* da GNU é a ferramenta `as`. No entanto pode usar-se a ferramenta `gcc` desde que o ficheiro tenha extensão `.s`
- Comentários:

  # comentário de linha
  / comentário de bloco *\/*
- O conteúdo de um ficheiro em *assembly* deve ser organizado em três colunas

1ª coluna	2ª coluna	3ª coluna
	diretiva do compilador	operandos da diretiva
nome/símbolo definido pelo programador (ex: nome de funções, labels, ...) terminada pelo caracter ':'		
	mnemónica da instrução	operandos da instrução

Secções lógicas

- As secções lógicas dividem o programa em áreas com diferentes objetivos
- O GCC usa o formato ELF (*Executable and Linkable Format*) para produzir o resultado da compilação
- Genericamente uma secção é definida com a diretiva

  .section <section-name>
- Exemplos de algumas secções com significados pré-estabelecidos

Secção	Descrição
.text	Código executável (contribui para a imagem do programa em memória)
.data	Variáveis globais iniciadas (contribui para a imagem do programa em memória)
.bss	Variáveis globais não iniciadas (contribui para a imagem do programa em memória e o espaço correspondente é iniciado com 0)
.section .rodata	Dados constantes e imutáveis (contribui para a imagem do programa em memória. Ex: *strings*, literais)
.section .debug	Informação simbólica para *degub* (quando se compila com a opção –g)

Diretivas que alocam espaço

Diretiva		Secção	Descrição
.align	*exp1, exp2*	*.bss, .data, .rodata*	Alinha o próximo dado num endereço múltiplo de *exp1* e, se *exp2* presente, preenche com o byte *exp2*. Se *exp2* omisso, por norma, preenche com 0
.ascii	*“string”*	*.data, .rodata*	Zero ou mais *strings* separadas por ‘,’
.asciz	*“string”*	*.data, .rodata*	Zero ou mais *strings* terminadas por 0 separadas por ‘,’
.byte	*expressions*	*.data, .rodata*	Zero ou mais expressões codificadas em 8 bits
.long/.int	*expressions*	*.data, .rodata*	Zero ou mais expressões codificadas em 32 bits
.quad	*expressions*	*.data, .rodata*	Zero ou mais expressões codificadas em 64 bits
.space	*size, fill*	*.bss*	Aloca *size* bytes iniciados, opcionalmente, pelo valor *fill*. Se *fill* omisso, por norma, preenche com 0
.word	*expressions*	*.data, .rodata*	Zero ou mais expressões codificadas em 16 bits

Outras diretivas

Diretiva		Descrição
.equ/.set	*symbol, exp*	Define símbolo *symbol* associando-lhe o valor *exp*
.globl/.global	*symbols*	Torna a sequência de símbolos separados por ‘,’ públicos (visível para o *linker*). Por omissão, os símbolos são privados ao módulo
.end		Última diretiva de um módulo em *assembly*

Exemplo de um ficheiro em *assembly*

```
        .globl    f0, f1    # Os símbolos f0 e f1 ficam públicos e visíveis externamente
        .text               # Secção com código
f0:     # Function f0 code
f1:     # Function f1 code
        .data               # Secção com variáveis globais iniciadas
var1:   .byte     1, 2, 3   # Variável var1 com uma sequência de 3 bytes
        .align    4         # Alinha var2 num endereço múltiplo de 4 (introduz 1 byte)
var2:   .long     1, 2      # Variável var2 com uma sequência de 2 bytes
        .align    8         # Alinha var3 num endereço múltiplo de 8 (introduz 4 bytes)
var3:   .quad     0xABC     # Variável 3 com o valor 0xABC
        .bss                # Secção com variáveis globais não iniciadas
var4:   .space    16        # Variável var4 com uma sequência de 16 bytes
        .align    8         # Alinha var5 num endereço múltiplo de 8 (introduz 0 bytes)
var5:   .space    8         # Variável var5 com uma sequência de 8 bytes
        .end
```