Arquitetura de Computadores

LEETC | LEIC | LEIRT

Programação em Assembly

Aula teórico-prática #3

ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia Eletrónica e Telecomunicações e de Computadores

Março de 2024

Introdução

Este trabalho tem como principal objetivo o exercício da programação em linguagem *assembly* do processador P16, incluindo a organização dos programas em rotinas e a exploração das regras da convenção P16 no seu desenvolvimento.

Os exercícios propostos envolvem *i*) operações com caracteres e números inteiros, com e sem sinal, *ii*) processamento de *arrays* em memória, *iii*) invocação e retorno de funções, *iv*) passagem de argumentos e devolução de valores de rotinas e *v*) utilização de variáveis locais e globais.

Tipos

Na especificação dos exercícios adota-se a sintaxe da linguagem C [4] e consideram-se os tipos definidos na biblioteca C [5], porquanto os tipos numéricos apresentados são representados na base 2 e têm os seguintes significados:

`int8_t` – inteiro a 8 bits, com sinal	`uint8_t` – inteiro a 8 bits, sem sinal
`int16_t` – inteiro a 16 bits, com sinal	`uint16_t` – inteiro a 16 bits, sem sinal
`int32_t` – inteiro a 32 bits, com sinal	`uint32_t` – inteiro a 32 bits, sem sinal

O tipo de dados `char` é utilizado para representar caracteres segundo o padrão *American Standard Code for Information Interchange* (ASCII) [1], ocupando cada carácter 8 bits com valores possíveis entre 0 e 127. A implementação deste tipo é feita como `uint8_t`.

Considera-se ainda o tipo de dados composto *array* da linguagem C para representar coleções de um ou mais valores, todos do mesmo tipo, armazenados em posições de memória contíguas e acessíveis por um único nome.

Uma *string* é um caso particular do tipo de dados composto *array*, em que os seus elementos são do tipo `char` e o último elemento da coleção é o carácter '\0'.

Exercícios

1. Implemente, usando a linguagem *assembly* do P16, a função `which_vowel` que identifica se o argumento de `letter` é uma vogal. A função deve retornar 0, quando `letter` toma o valor `a`; retorna 1, no caso do valor `e`; e assim sucessivamente. Caso o conteúdo não corresponda a uma vogal, deve ser retornado o valor -1.

```
int16_t which_vowel( char letter ) {
     int16_t i;
     switch( letter ) {
          case 'a' : i = 0; break;
          case 'e' : i = 1; break;
          case 'i' : i = 2; break;
          case 'o' : i = 3; break;
          case 'u' : i = 4; break;
          default  : i = -1;
     }
     return i;
}
```

2. Implemente, usando a linguagem *assembly* do P16, a função `vowel_histogram` que calcula o número de ocorrências de cada vogal na *string* `phrase`, codificada em ASCII e terminada pelo caráter '\0', atualizando o *array* `occurences` com essa informação. Só devem ser considerados os primeiros `max_letters` carateres.

```
void vowel_histogram( char phrase[], uint16_t max_letters, uint16_t
    occurrences[5] ) {
      int16_t idx;
      uint16_t i;
      for( i = 0; phrase[i]!='\0' && i < max_letters ; i++ ) {
           if ( (idx = which_vowel( phrase[i] ) ) != -1 ) {
                occurrences[idx]++;
           }
      }
}
```

3. Implemente, usando a linguagem *assembly* do P16, o seguinte troço de código, que permite verificar e demonstrar o comportamento da função `vowel_histogram` em dois cenários de utilização distintos.

```
#define SIZE 5

uint16_t occurrences1 [SIZE];
uint16_t occurrences2 [SIZE];

uint8_t phrase1 [] = "hello, world";
uint8_t phrase2 [] = "the quick brown fox jumps over the lazy dog";

void main( void ){
     vowel_histogram( phrase1, 7, occurrences1 );
     vowel_histogram( phrase2, 50, occurrences2 );
}
```

a) Implemente, usando a sintaxe do assembler `p16as` (v1.5.0), as definições de todos os símbolos apresentados.

b) Implemente, usando a sintaxe do assembler `p16as` (v1.5.0), as definições de todas as variáveis globais apresentadas, definindo as secções necessárias.

c) Implemente a rotina *main*.

d) Implemente outros troços de código que considere fundamentais para a construção e correta execução de um programa, definindo as secções necessárias. Justifique a sua resposta.

e) Usando a ferramenta `p16as`, proceda à geração de um ficheiro binário correspondente ao programa desenvolvido e, de seguida, utilize as ferramentas `p16sim` e `p16dbg` para executar esse programa e verificar os resultados produzidos nas variáveis `occurrences1` e `occurrences2`.

Bibliografia

[1] ANSI: **ISO-IR-6: ASCII Graphic character set**, 1975. https://iselpt.sharepoint.com/:b:/s/acp/EUmzOiW_UNpLo4a3F0Z1LRwBDfkDyCFfVUumgRS04Y9HtA?e=kscsNz, acedido em 23-02-2024.

[2] Dias, Tiago: **Manual de consulta rápida das instruções do P16**. ISEL, Lisboa, Portugal, 2024. https://iselpt.sharepoint.com/:b:/s/acp/EVR0vj3IxJZHp--3eH88wQUBspGUrKP0VXqGcR_USuoeBQ?e=Jwtpvx (Acedido em 26-02-2024).

[3] Harris, Sarah e David Harris: **Digital Design and Computer Architecture: ARM Edition**. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1a edição, 2015, ISBN 978-0128000564.

[4] Kernighan, Brian W. e Dennis M. Ritchie: **The C Programming Language**. Prentice Hall Professional Technical Reference, 2nd edição, 1988, ISBN 0131103709.

[5] Loosemore, Sandra, Richard M. Stallman, Roland McGrath, Andrew Oram e Ulrich Drepper: **The GNU C Library Reference Manual**, 2022. https://www.gnu.org/software/libc/manual/html_node/Integers.html, acedido em 23-02-2024.