Arquitetura de Computadores

Exercícios compilados pelo prof. Tiago Dias para serem resolvidos em *assembly* do P16

João Pedro Patriarca (jpatri@cc.isel.ipl.pt), Gabinete F.0.23 do edifício F

ISEL, ADEETC, LEIC

Sumário

- Os exercícios propostos permitem a prática de *assembly* nos seguintes domínios:
  - Manipulação de dados no domínio dos naturais, relativos a 8 e a 16 bits
  - Manipulação de sequências de dados em memória
  - Manipulação de caracteres e sequências de caracteres
  - Ordenação de *arrays*

Sequência de inteiros de 16 bits sem sinal em memória

Implementar a função `max`, que devolve o maior valor presente no *array* `a`, com `n` elementos. A constante `UINT16_MIN` corresponde ao menor valor possível de codificar numa variável com tipo `uint16_t`.

```
# define UINT16_MIN 0
uint16_t max( uint16_t a[], uint16_t n ) {
   uint16_t max = UINT16_MIN ;
   for ( i = 0; i < n; i++) {
      if ( a[i] > max )
         max = a[i];
   }
   return max;
}
```

Sequência de inteiros de 16 bits com sinal em memória

Com as adaptações necessárias, repetir o exercício anterior considerando que os elementos do *array* são do tipo `int16_t`.

```
# define INT16_MIN 0x8000
uint16_t max( int16_t a[], uint16_t n ) {
   int16_t max = INT16_MIN ;
   for ( i = 0; i < n; i++) {
      if ( a[i] > max )
         max = a[i];
   }
   return max;
}
```

Manipulação de caracteres

Implementar a função `to_upper`, que devolve a maiúscula correspondente ao carácter do tipo letra passado à função no parâmetro `ch`. Caso o argumento de `ch` não corresponda a um carácter do tipo letra, a função devolve esse valor inalterado.

```
char to_upper ( char ch ) {
   char uc = ch;
   if ( ch >= 'a' && ch <= 'z') {
      uc = ch + ( 'A' - 'a' );
   }
   return uc;
}
```

Manipulação de sequência de caracteres (1 de 3)

Implementar a função `str_upper`, que altera a *string* `str` de modo a que todos os seus caracteres do tipo letra sejam maiúsculas. A função devolve o número total de caracteres que compõem a *string*.

```
uint16_t str_upper ( char str [] ) {
   for ( uint16_t i = 0; str[i] != '\0'; i++ ) {
      str[i] = to_upper( str[i] );
   }
   return i;
}
```

Manipulação de sequência de caracteres (2 de 3)

Implementar a função `str_copy`, que copia todos os caracteres que compõem a *string* `src` para a *string* `dst`.

```
void str_copy ( char dst [], char src [] ) {
   uint16_t i = 0;
   while ( src[i] != '\0' ) {
      dst[i] = src[i];
      i++;
   }
   dst[i] = '\0 ';
}
```

Manipulação de sequência de caracteres (3 de 3)

Implementar o programa de teste para a função `str_copy`.

```
# define MAX_PHRASE_SIZE 100

uint8_t phrase_orig [] = "the quick brown fox jumps over the lazy dog";
uint8_t phrase_copy [ MAX_PHRASE_SIZE ];

void main ( void ){
   str_copy( phrase_copy , phrase_orig );
}
```

Ordenação de *arrays* (1 de 2)

Implementar a função `selection_sort`, que ordena os `n` elementos (com `n` > 0) do *array* `a` por ordem crescente dos seus valores.

A função `min_idx` devolve o índice do *array* `a` correspondente ao menor valor presente entre os índices `f` e `l`.

```
void selection_sort ( int16_t a[],
                      uint16_t n ) {
   uint16_t i, j;
   int16_t temp ;
   for ( i = 0; i < n - 1; i++ ) {
      j = min_idx ( a, i, n );
      temp = a[i];
      a[i] = a[j];
      a[j] = temp ;
   }
}
```

```
uint16_t min_idx ( int16_t a[], uint16_t f,
                   uint16_t l ) {
   uint16_t i, j;
   i = f;
   for ( j = f + 1; j < l; j++ ) {
      if( a[j] < a[i] ) {
         i = j;
      }
   }
   return i;
}
```

Ordenação de *arrays* (1 de 2)

Implementar o programa de teste para a função `selection_sort`.

```
int16_t array [] = {-2, 45, 0, 11, -9};

void main ( void ) {
   selection_sort ( array , 5 );
}
```