String Procedures in Irvine32

The Irvine32 library provides a number of procedures for manipulating null-terminated strings. These procedures are similar to the functions in the standard C library.

Str_copy

The Str_copy procedure copies a source string to a target string. It takes two arguments: a pointer to the source string and a pointer to the target string. The procedure returns a pointer to the target string.

```
140 ; Copy a source string to a target string.
141 Str_copy PROTO,
142 source:PTR BYTE,
143 target:PTR BYTE
```

The following code shows how to use the **Str_copy procedure** to copy the string "Hello, world!" to the string "buffer":

```
145 mov eax, OFFSET "Hello, world!"
146 mov ebx, OFFSET buffer
147 call Str_copy
148 ; buffer now contains the string "Hello, world!"
```

Str_length

The Str_length procedure returns the length of a string (excluding the null byte) in the EAX register. It takes one argument: a pointer to the string.

```
152 ; Return the length of a string (excluding the null byte) in EAX.
153 Str_length PROTO,
154 pString:PTR BYTE
```

The following code shows how to use the Str_length procedure to get the length of the string "Hello, world!" and store it in the EAX

register:

```
157 mov eax, OFFSET "Hello, world!"
158 call Str_length
159 ; EAX now contains the value 12, which is the length of the string "Hello, world!"
```

Str_compare

The Str_compare procedure compares two strings and sets the Zero and Carry flags in the same way as the CMP instruction. It takes two arguments: pointers to the two strings.

```
162 ; Compare string1 to string2. Set the Zero and
163 ; Carry flags in the same way as the CMP instruction.
164 Str_compare PROTO,
165 string1:PTR BYTE,
166 string2:PTR BYTE
```

The following code shows how to use the Str_compare procedure to compare the strings "Hello, world!" and "Hello, world!":

```
170 mov eax, OFFSET "Hello, world!"
171 mov ebx, OFFSET "Hello, world!"
172 call Str_compare
173 ; The Zero flag will be set, indicating that the two strings are equal.
```

Str_trim

The Str_trim procedure trims a given trailing character from a string. It takes two arguments: a pointer to the string and the character to trim.

```
180 ; Trim a given trailing character from a string.
181 ; The second argument is the character to trim.
182 Str_trim PROTO,
183 pString:PTR BYTE,
184 char:BYTE
```

The following code shows how to use the Str_trim procedure to trim the trailing newline character from the string "Hello, world!\n":

```
mov eax, OFFSET "Hello, world!\n"
mov ebx, AL ; '\n' character
call Str_trim
; The string "Hello, world!" is now stored in the memory location pointed to by EAX.
```

Str_ucase

The Str_ucase procedure converts a string to upper case. It takes one argument: a pointer to the string.

```
; Convert a string to upper case.
Str_ucase PROTO,
pString:PTR BYTE
```

The following code shows how to use the Str_ucase procedure to convert the string "hello, world!" to upper case:

```
mov eax, OFFSET "hello, world!"
call Str_ucase
; The string "HELLO, WORLD!" is now stored in the memory location pointed to by EAX.
```