# 10 Character Sets (cset.hhf)

The HLA Standard Library contains several routines that provide the power of the HLA compile-time character set facilities at run-time (i.e., within your programs).

HLA uses a 128-bit bitmap (16 consecutive bytes) to implement sets of seven-bit ASCII characters. This has a very important implication: you cannot pass byte values greater than $7F to a character set function. Currently, the HLA Standard Library routines do not check for values out of range (for performance reasons). In the future, this checking may be added as a compilable option. For the time being, however, it is your responsibility to verify that all character values are in the range #$0..#$7F (and, in general, #$0 is an exceeding bad value to specify in many cases since the null character terminates strings).

The bitmap consists of 128 consectutive bits numbered 0..127. If a bit in a character set is one, then the corresponding character (whose ASCII code matches the bit number) is a member of the character set. Conversely, if a bit is zero, the corresponding character is not a member of the set.

Note that many routines pass character sets by value. This means you can pass HLA character set constants as parameters to these procedures/functions. HLA emits four MOV (doubleword) instructions to copy a character set by value, so passing character sets by value is not horribly inefficient (though not quite as fast as a 32-bit integer!).

**Warning**: All of the character set routines are members of the cs namespace. This means you cannot use the name cs within your programs. (cs is a common character set name that lazy programmers use; sorry, it's already been taken!)

The following sections describe each of the character set routines in the HLA Standard Library.

**A Note About Thread Safety**: The routines in this module are all thread safe.

**Note about stack diagrams**: this documentation includes stack diagrams for those functions that pass parameters on the stack. To conserve space, this documentation does not include a stack diagram for any function that does not pass data on the stack (that is, only a return address appears on the stack).

## 10.1 Predicates (tests)

Although the "returns" value for each of the following functions is "AL", these tests always set EAX to zero or one. Therefore, you may refer to the AL or EAX register after these tests, whichever is more convenient for you. If you use instruction composition and bury one of these function calls in another statement, that statement will use the AL register as the operand.

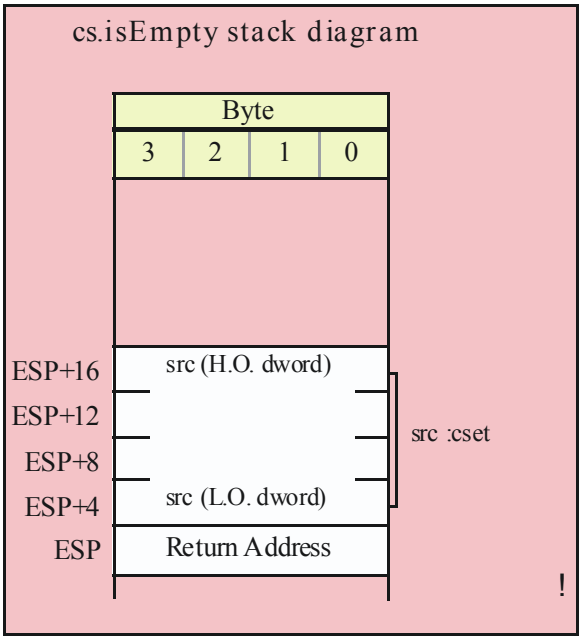
Note that these functions generally pass their character set parameters by value. This involves pushing 16 bytes on the stack for each cset parameter (typically four push instructions). Keep this in mind if efficiency is your utmost concern. Be sure to read the section on "Passing CSET Parameters on the Stack" in the chapter on "Passing Parameters to Standard Library Routines".

```
procedure  cs.IsEmpty( src: cset ); @returns( "AL" );
```

This function returns true (1) in the AL register if the specified character set is empty (has no members). It returns false (0) in AL/EAX otherwise.

```
HLA high-level calling sequence examples:


cs.IsEmpty( csetVar );
mov( al, booleanResult );


HLA low-level calling sequence examples:

// cs.IsEmpty is really intended to be used as a high-level
// type function. It's actually just as easy to compute the
// function manually as it is to call it. Here's the low-level
// calling sequence:

push( (type dword csetVar[12]));
push( (type dword csetVar[8]));
push( (type dword csetVar[4]));
```

```
    push( (type dword csetVar[0]));
    call cs.IsEmpty;
    mov( al, booleanResult;

    // Here's the same thing using bare machine instructions:

    mov( (type dword csetVar[0]), eax );
    or( (type dword csetVar[4]), eax );
    or( (type dword csetVar[8]), eax );
    or( (type dword csetVar[12]), eax );
    setz( al );
    mov( al, booleanResult );
```

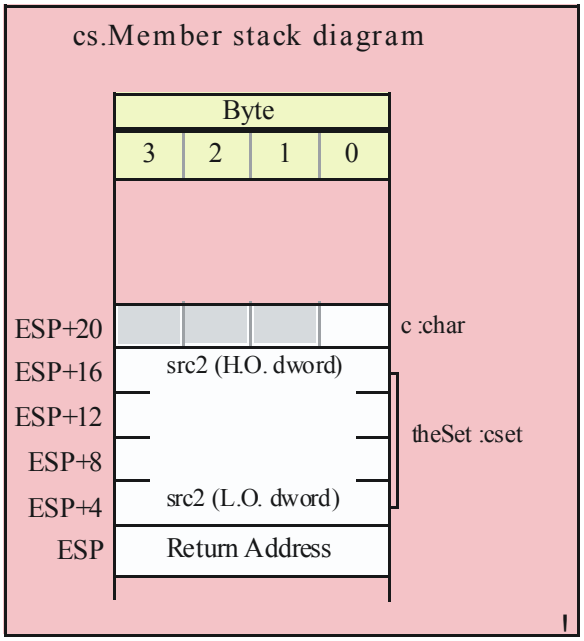



**procedure cs.member( c:char; theSet:cset ); @returns( "AL" );**

This function returns true (1) or false (0) in AL/EAX if the specified character is a member of the specified character set.

```
    HLA high-level calling sequence examples:

    cs.member( charVar, csetVar );
    mov( al, booleanResult );


    HLA low-level calling sequence examples:

    // cs.member is really intended to be used as a high-level
    // type function. It's actually just as easy to compute the
    // function manually as it is to call it. Here's the low-level
    // calling sequence:

    movzx( charVar, eax );
    push( eax );
    push( (type dword csetVar[12]));
    push( (type dword csetVar[8]));
```

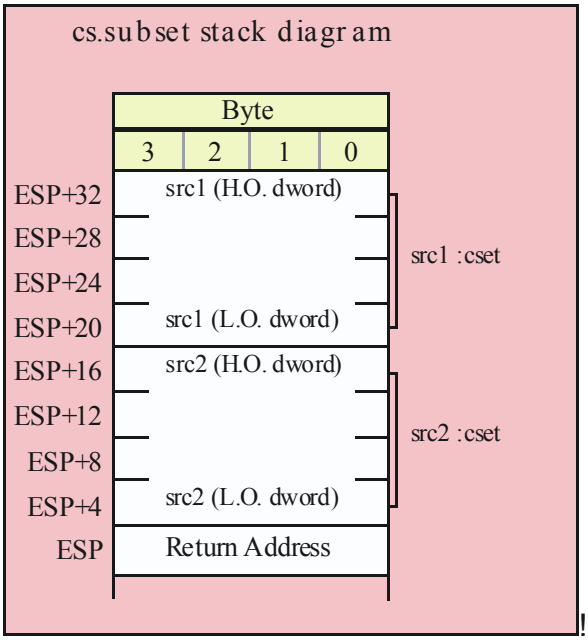
```
push( (type dword csetVar[4]));
push( (type dword csetVar[0]));
call cs.member;
mov( al, booleanResult;

// Here's the same thing using bare machine instructions:

movzx( charVar, eax );
bt( eax, csetVar );
setc( al );
mov( al, booleanResult );
```




**procedure cs.subset( src1:cset; src2:cset ); @returns( "AL" );**

The cs.subset function returns true in AL/EAX if src1 <= src2 (that is, all of src1's members are also members of src2).

```
HLA high-level calling sequence examples:

cs.subset( subsetVar, supersetVar );
mov( al, booleanResult );

HLA low-level calling sequence examples:

push( (type dword subsetVar[12]));
push( (type dword subsetVar[8]));
push( (type dword subsetVar[4]));
push( (type dword subsetVar[0]));

push( (type dword supersetVar [12]));
push( (type dword supersetVar [8]));
push( (type dword supersetVar [4]));
push( (type dword supersetVar [0]));
call cs.subset;
mov( al, booleanResult );
```

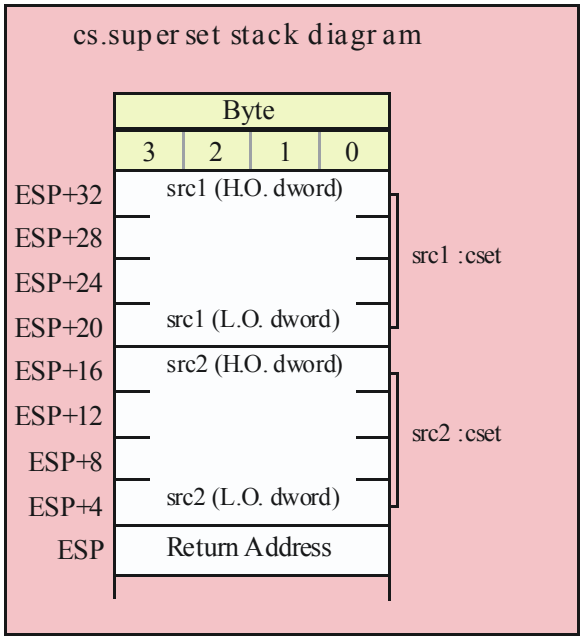



!

**procedure cs.superset( src1:cset; src2:cset ); @returns( "AL" );**

The cs.superset function returns true in AL/EAX if src1 >= src2 (that is, all of src2's members are members of src1).

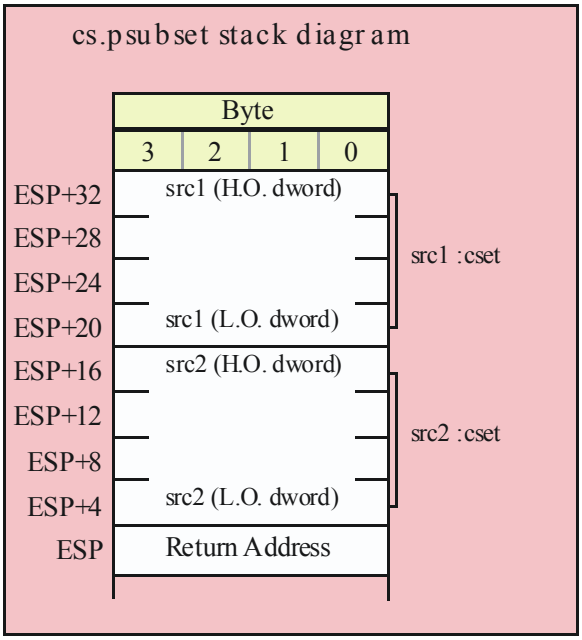
```
HLA high-level calling sequence examples:

cs.superset( supersetVar, subsetVar );
mov( al, booleanResult );

HLA low-level calling sequence examples:

push( (type dword supersetVar[12]));
push( (type dword supersetVar[8]));
push( (type dword supersetVar[4]));
push( (type dword supersetVar[0]));

push( (type dword subsetVar[12]));
push( (type dword subsetVar[8]));
push( (type dword subsetVar[4]));
push( (type dword subsetVar[0]));
call cs.superset;
mov( al, booleanResult );
```




**procedure cs.psubset( src1:cset; src2:cset ); @returns( "AL" );**

The cs.psubset (proper subset) function returns true in AL/EAX if src1 < src2 (that is, all of src1's members are members of src2 but src1 <> src2).

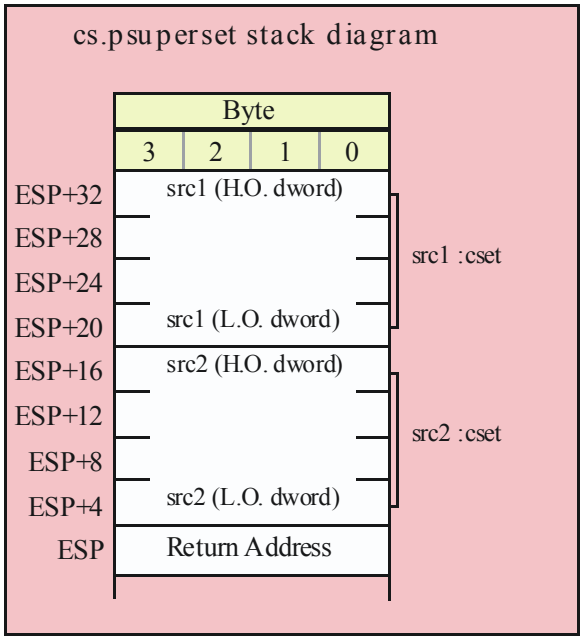
```
HLA high-level calling sequence examples:

cs.psubset( subsetVar, supersetVar );
mov( al, booleanResult );

HLA low-level calling sequence examples:

push( (type dword subsetVar[12]));
push( (type dword subsetVar[8]));
push( (type dword subsetVar[4]));
push( (type dword subsetVar[0]));

push( (type dword supersetVar [12]));
push( (type dword supersetVar [8]));
push( (type dword supersetVar [4]));
push( (type dword supersetVar [0]));
call cs.psubset;
mov( al, booleanResult );
```




**`procedure cs.psuperset( src1:cset; src2:cset ); @returns( "AL" );`**

The cs.spsuperset (proper superset) function returns true in AL/EAX if src1 > src2 (that is, all of src2's members are members of src1 but src2 <> src1).

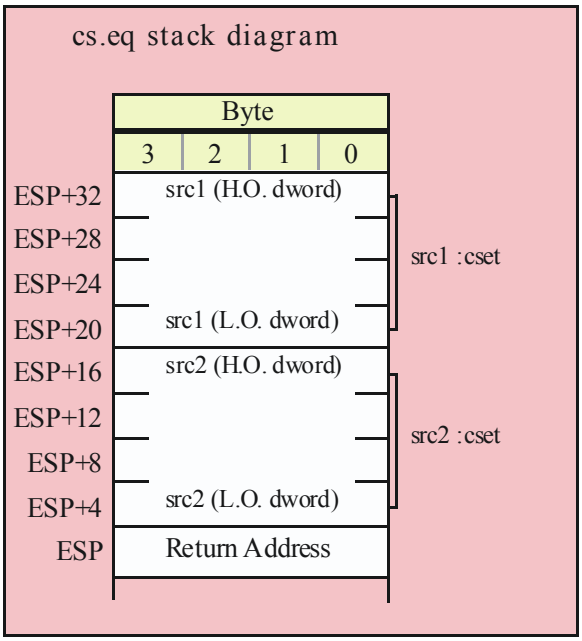
```
HLA high-level calling sequence examples:

cs.psuperset( supersetVar, subsetVar );
mov( al, booleanResult );

HLA low-level calling sequence examples:

push( (type dword supersetVar[12]));
push( (type dword supersetVar[8]));
push( (type dword supersetVar[4]));
push( (type dword supersetVar[0]));

push( (type dword subsetVar[12]));
push( (type dword subsetVar[8]));
push( (type dword subsetVar[4]));
push( (type dword subsetVar[0]));
call cs.psuperset;
mov( al, booleanResult );
```




**procedure cs.eq( src1:cset; src2:cset ); @returns( "AL" );**

The cs.eq function compares the two sets and returns true/false in AL/EAX; true if the two sets are equal, false if they are not.

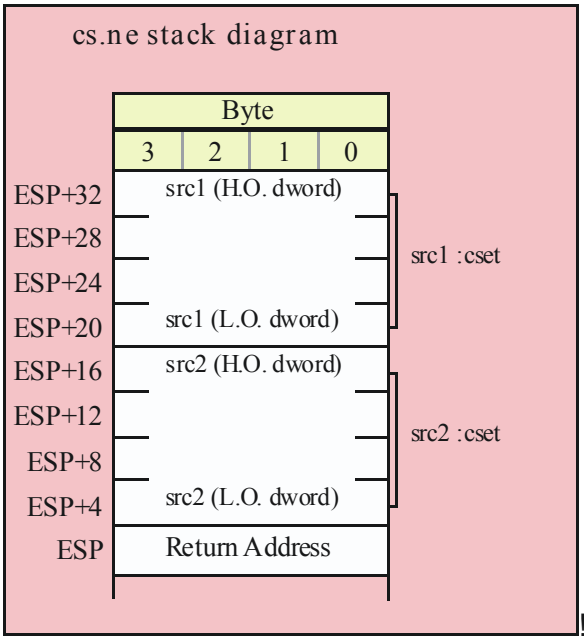
```
HLA high-level calling sequence examples:

cs.eq( src1, src2 );
mov( al, booleanResult );

HLA low-level calling sequence examples:

push( (type dword src1[12]));
push( (type dword src1[8]));
push( (type dword src1[4]));
push( (type dword src1[0]));

push( (type dword src2[12]));
push( (type dword src2[8]));
push( (type dword src2[4]));
push( (type dword src2[0]));
call cs.eq;
mov( al, booleanResult );
```




**procedure cs.ne( src1:cset; src2:cset ); @returns( "AL" );**

The cs.eq function compares the two sets and returns true/false in AL/EAX; true if the two sets are not equal, false if they are equal.

```
HLA high-level calling sequence examples:

cs.ne( src1, src2 );
mov( al, booleanResult );

HLA low-level calling sequence examples:

push( (type dword src1[12]));
push( (type dword src1[8]));
push( (type dword src1[4]));
push( (type dword src1[0]));

push( (type dword src2[12]));
push( (type dword src2[8]));
push( (type dword src2[4]));
push( (type dword src2[0]));
call cs.ne;
mov( al, booleanResult );
```




## 10.2 Character Set Construction and Manipulation

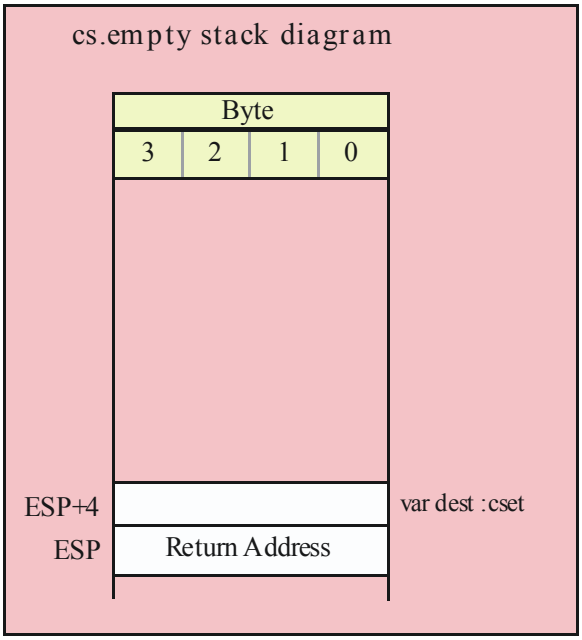
The functions in this group create character set objects, extract data from character set objects, or transfer data between character set objects.

**`procedure cs.empty( var dest:cset );`**

This function clears all the bits in a character set to create the empty set. Note that the single character set parameter is passed by reference.

```
HLA high-level calling sequence examples:

cs.empty( csetVar );

HLA low-level calling sequence examples:

// cset_s is a variable declared in the static/storage section:

pushd( &cset_s );
call cs.empty;

// cset_v is a variable declared in the var section or
// is a parameter:

lea( eax, cset_v );
push( eax );
call cs.empty;

// Alternative call passing cset_v if no 32-bit registers
// are available (this code assumes that EBP points at the current
// activation record/stack frame that contains cset_v):

push( ebp );
add( @offset( cset_v ), (type dword [esp] ));
call cs.empty;
```

```
// Low-level call assuming a 32-bit register (esi in this case)
// contains the address of the cset:

push( esi );
call cs.empty;

// Low-level call assuming a dword or pointer variable contains the
// address of the cset that will receive the delimiter character set:

push( ptrToDelims );
call cs.empty;
```

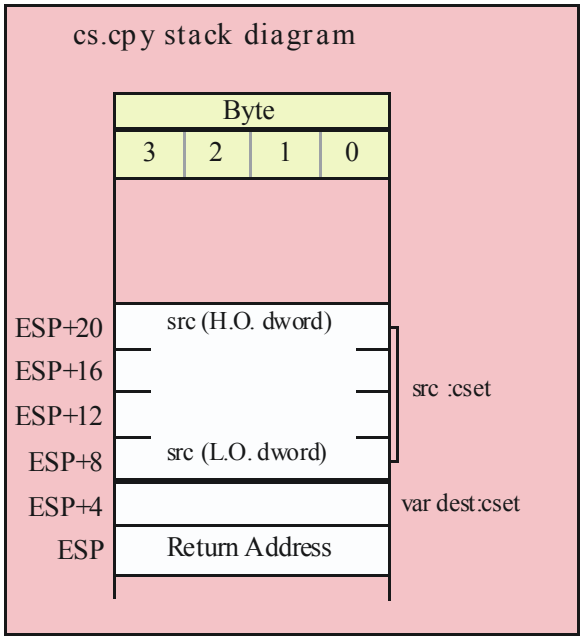



**procedure cs.cpy( src:cset; var dest:cset );**

This routine copies the data from the source character set (src) to the destination character set (dest). Note that the dest set is passed by reference. Although this routine is convenient, you should consider writing a macro to do this same function (copy 16 bytes from src to dest) if you call this function in time critical sections of your code.

```
HLA high-level calling sequence examples:

cs.cpy( csetSrc, csetDest );
cs.cpy( {'a'..'z'}, lowerCaseCset );


HLA low-level calling sequence examples:

// csetDest_s is a variable declared
// in the static/storage section:

push( (type dword csetSrc_s[12]));
push( (type dword csetSrc_s[8]));
push( (type dword csetSrc_s[4]));
push( (type dword csetSrc_s[0]));
```

```
pushd( &csetDest_s );
call cs.cpy;

// csetDest_v is a variable declared in the var section or
// is a parameter:

push( (type dword csetSrc_v[12]));
push( (type dword csetSrc_v[8]));
push( (type dword csetSrc_v[4]));
push( (type dword csetSrc_v[0]));
lea( eax, csetDest_v );
push( eax );
call cs.cpy;

// Alternative call passing csetDest_v if no 32-bit registers
// are available (this code assumes that EBP points at the current
// activation record/stack frame that contains csetDest_v):

push( (type dword csetSrc_v[12]));
push( (type dword csetSrc_v[8]));
push( (type dword csetSrc_v[4]));
push( (type dword csetSrc_v[0]));
push( ebp );
add( @offset( csetDest_v ), (type dword [esp] ));
call cs.cpy;

// Low-level call assuming a 32-bit register (edi in this case)
// contains the address of the destination cset:

push( (type dword csetSrc_v[12]));
push( (type dword csetSrc_v[8]));
push( (type dword csetSrc_v[4]));
push( (type dword csetSrc_v[0]));
push( edi );
call cs.cpy;

// Low-level call assuming a dword or pointer variable contains the
// address of the destination cset:

push( (type dword csetSrc[12]));
push( (type dword csetSrc[8]));
push( (type dword csetSrc[4]));
push( (type dword csetSrc[0]));
push( ptrTodest );
call cs.cpy;
```

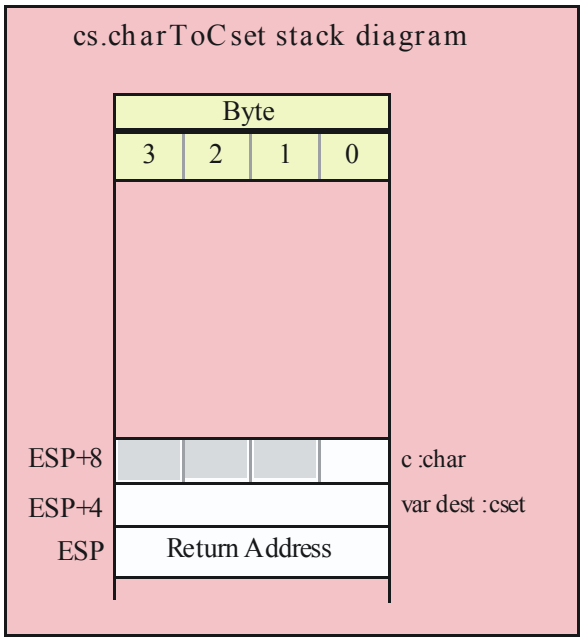



**procedure cs.charToCset( c:char; var dest:cset );**

The cs.charToCset procedure takes the character passed as a parameter and creates a singleton set containing that character (a singleton is a set with exactly one member). The resulting set is stored into the destination parameter (which is passed by reference).

```
HLA high-level calling sequence examples:

cs.charToCset( 'c', csetDest );
cs.charToCset( charVar, lowerCaseCset );
cs.charToCset( (type char [esi]), (type cset [edi]));


HLA low-level calling sequence examples:

// The following low-level examples all assume that csetDest_s is
// a statically-declared object (static or storage section). For
// examples of additional ways to pass the destination cset by
// reference, please see the examples for cs.cpy given earlier.
//
// Passing a single character constant:

pushd( 'c' );
pushd( &csetDest_s );
call cs.charToCset;

// Passing a character variable, assuming a 32-bit register is
// available or one of the 8-bit register: AL, BL, CL, or DL

mov( charVar, al );
push( eax );
pushd( &csetDest_s );
call cs.charToCset;
```

```
// If the character variable is guaranteed not to be in the last
// three bytes of allocated storage, you could also do this:

push( type dword charVar );
pushd( &csetDest_s );
call cs.charToCset;

// If the character is in one of the 8-bit registers: AH, BH, CH, DH

sub( 4, esp );
mov( ah, [esp] );
pushd( &csetDest_s );
call cs.charToCset;
```

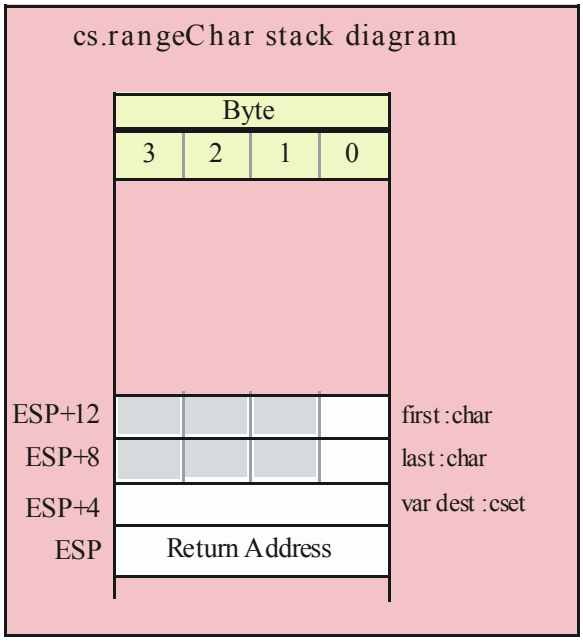



**procedure cs.rangeChar( first:char; last:char; var dest:cset );**

This function creates a set whose member range between the first character specified and the last character specified. For example, cs.rangeChar( 'A', 'Z', UpperCaseSet) will create a character set whose members are the upper case alphabetic characters. Any previous members in the destination set are lost.

```
HLA high-level calling sequence examples:

cs.rangeChar( 'a', 'z', csetDest );
cs.rangeChar( charVar, endCharVal, lowerCaseCset );
cs.rangeChar( (type char [esi]), '0', (type cset [edi]));


HLA low-level calling sequence examples:

// The following low-level examples all assume that csetDest_s is
// a statically-declared object (static or storage section). For
// examples of additional ways to pass the destination cset by
// reference, please see the examples for cs.cpy given earlier.
// Also note that both the "first" and "last" parameters are
// character objects that you pass the same way. For brevity,
// the following examples only demonstrate variations on the
```

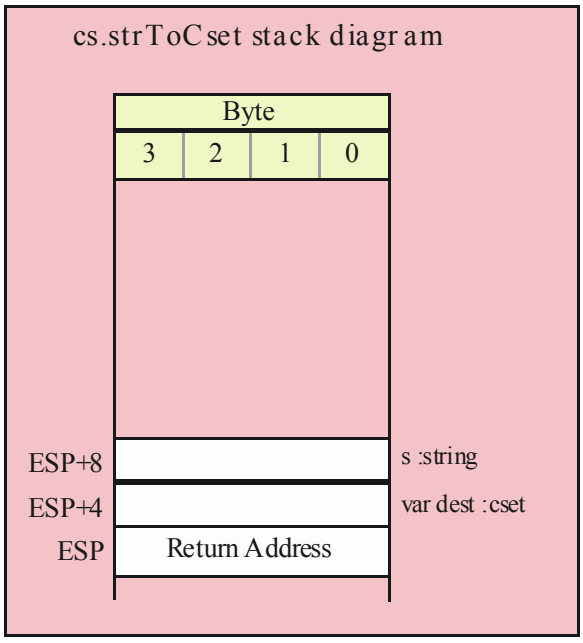
```
// "first" parameter; the same principles apply to the "last"
// parameter.
//
// Passing two character constants:

pushd( 'a' );
pushd( 'z' );
pushd( &csetDest_s );
call cs.rangeChar;

// Passing a character variable, assuming a 32-bit register is
// available or one of the 8-bit register: AL, BL, CL, or DL

mov( charVar, al );
push( eax );
pushd( 'z' );
pushd( &csetDest_s );
call cs.rangeChar;

// If the "first" character variable is guaranteed not to be in the
// last three bytes of allocated storage, you could also do this:

push( type dword charVar );
pushd( 'z' );
pushd( &csetDest_s );
call cs.rangeChar;

// If the "first" character is in one of the 8-bit
// registers: AH, BH, CH, DH

sub( 4, esp );
mov( ah, [esp] );
pushd( 'z' );
pushd( &csetDest_s );
call cs.rangeChar;
```



**procedure cs.strToCset( s:string; var dest:cset );**

This function first sets the destination character set to the empty set. Then it "unions in" all the characters found in the string parameter to the destination set.

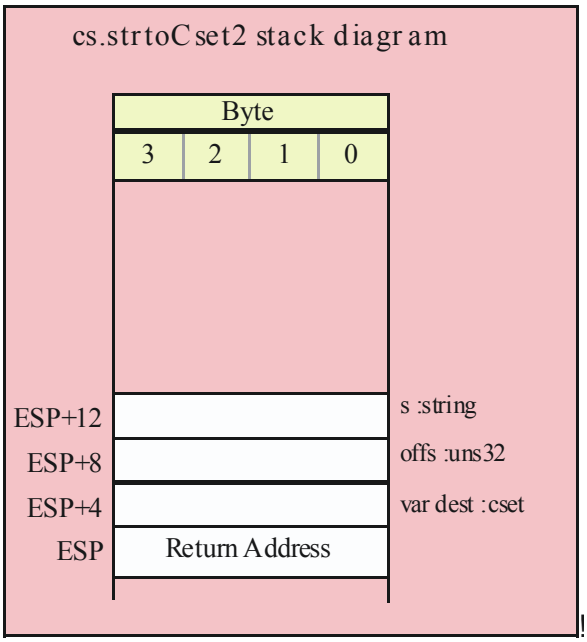
```
HLA high-level calling sequence examples:

cs.strToCset( strSrc, csetDest );
cs.strToCset( "ABCDEFabcdef", hexCset );


HLA low-level calling sequence examples:

// csetDest_s is a variable declared
// in the static/storage section:

push( strSrc );
pushd( &csetDest_s );
call cs.strtoCset;

// You could pass a string literal thusly (though there is
// no benefit to doing this over creating a statically
// initialized string variable and passing that string variable).

lea( eax, "abcdefABCDEF" );
push( eax );
pushd( &csetDest_s );
call cs.strtoCset;
```




**procedure cs.strToCset2( s:string; offs:uns32; var dest:cset );**

This function first sets the destination character set to the empty set. Then it "unions in" all the characters starting at offset offs in the string parameter to the destination character set.

```
HLA high-level calling sequence examples:

cs.strToCset2( strSrc, 2, csetDest );
cs.strToCset2( "ABCDEF", offsetIntoStr, partialHexCset );


HLA low-level calling sequence examples:

// csetDest_s is a variable declared
// in the static/storage section:

push( strSrc );
pushd( 2 )
pushd( &csetDest_s );
call cs.strToCset2;

// Assume the offset is in the variable "offsetIntoStr":

push( strSrc );
push( offsetIntoStr );
pushd( &csetDest_s );
call cs.strToCset2;
```



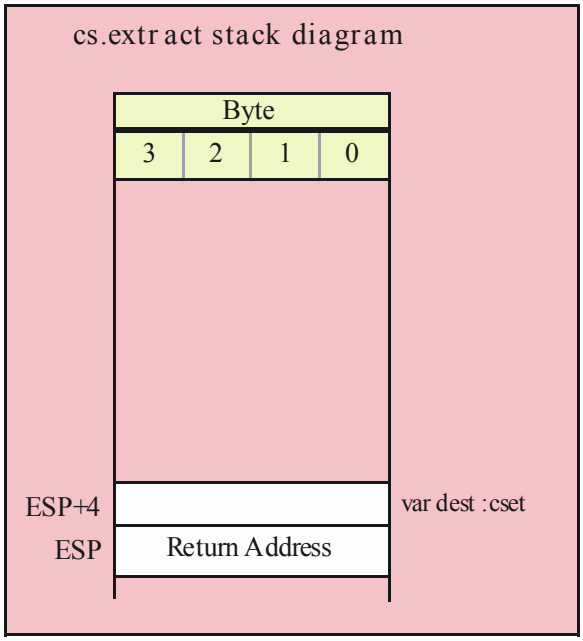

!

**procedure cs.extract( var dest:cset ); @returns( "EAX" );**

This function removes a single character from the character set and returns that character in the AL register. Currently, this function removes characters by order of their ASCII character codes (that is, each call returns the character in the set with the lowest ASCII code). However, you should not make this assumption. You should assume that this function could return the characters in an arbitrary order. If the specified character set is empty, this routine returns -1 ($FFFF_FFFF) in the EAX register; in all other cases the H.O. three bytes of EAX contain zero upon return.

Note: unlike the HLA compile-time function "@extract", this function actually removes the character from the character set ("@extract" leaves the character in the set). Keep this in mind. (In the future, the name of the HLA @extract function will probably be changed to something else to clean up this conflict.)

```
HLA high-level calling sequence examples:

cs.extract( csetVar );
if( eax <> -1 ) then

  mov( al, charExtracted );

endif;


HLA low-level calling sequence example:

// The following low-level example assumes that cset_s is
// a statically-declared object (static or storage section). For
// examples of additional ways to pass the destination cset by
// reference, please see the examples for cs.cpy given earlier.

pushd( &cset_s );
call cs.extract;
cmp( eax, -1 );
je noCharExtracted;
mov( al, charExtracted;
noCharExtracted:
```




## 10.3 Set Operations

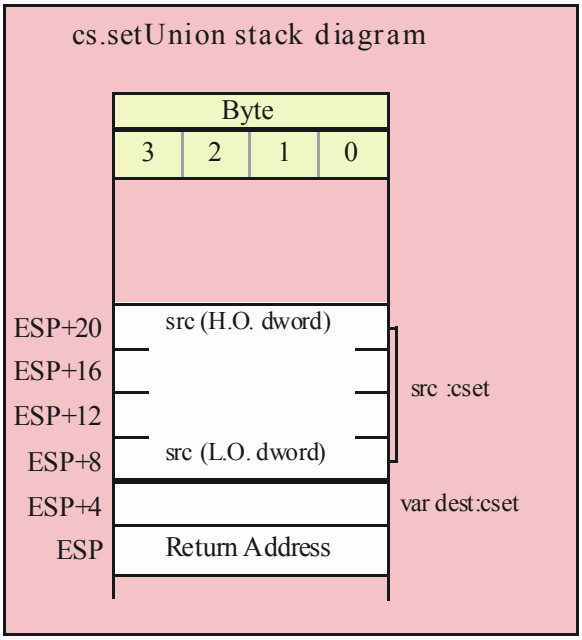
The following set functions perform what is generally considered to be set arithmetic: operations like set union, intersection, difference, and so on.

**procedure cs.setunion( src:cset; var dest:cset );**

This function computes the union of two sets, storing the result back into the destination set. Note that the destination set parameter is passed by reference.

Note: The name "setunion" was used rather than the more obvious choice of "union" because "union" is an HLA reserved word.

```
HLA high-level calling sequence examples:

cs.setunion( csetSrc, csetDest );
cs.setunion( {'a'..'z'}, lowerCaseUnion );


HLA low-level calling sequence example:

// The following low-level example assumes that csetDest_s is
// a statically-declared object (static or storage section). For
// examples of additional ways to pass the destination cset by
// reference, please see the examples for cs.cpy given earlier.

push( (type dword csetSrc_s[12]));
push( (type dword csetSrc_s[8]));
push( (type dword csetSrc_s[4]));
push( (type dword csetSrc_s[0]));
pushd( &csetDest_s );
call cs.setunion;
```

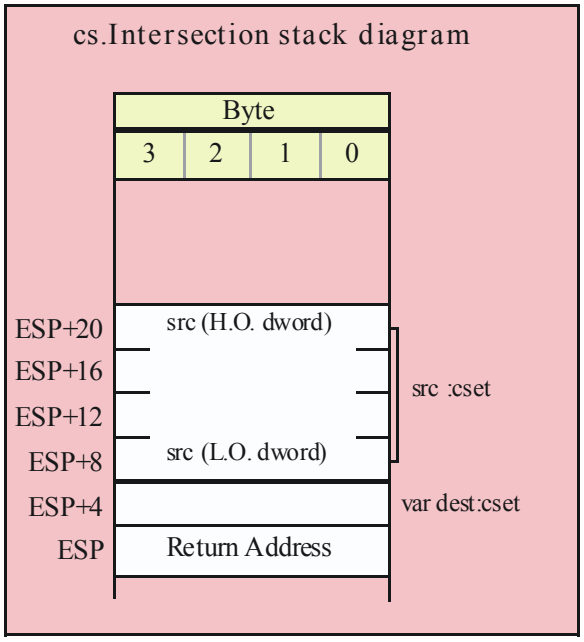



**procedure cs.intersection( src:cset; var dest:cset );**

This function computes the set intersection of the two sets passed as parameters and stores the result back into the destination set. Note that the dest parameter is passed by reference.

```
HLA high-level calling sequence examples:

cs.intersection( csetSrc, csetDest );
```

```
cs.intersection( {'a'..'z'}, lowerCaseUnion );


HLA low-level calling sequence example:

// The following low-level example assumes that csetDest_s is
// a statically-declared object (static or storage section). For
// examples of additional ways to pass the destination cset by
// reference, please see the examples for cs.cpy given earlier.

push( (type dword csetSrc_s[12]));
push( (type dword csetSrc_s[8]));
push( (type dword csetSrc_s[4]));
push( (type dword csetSrc_s[0]));
pushd( &csetDest_s );
call cs.intersection;
```

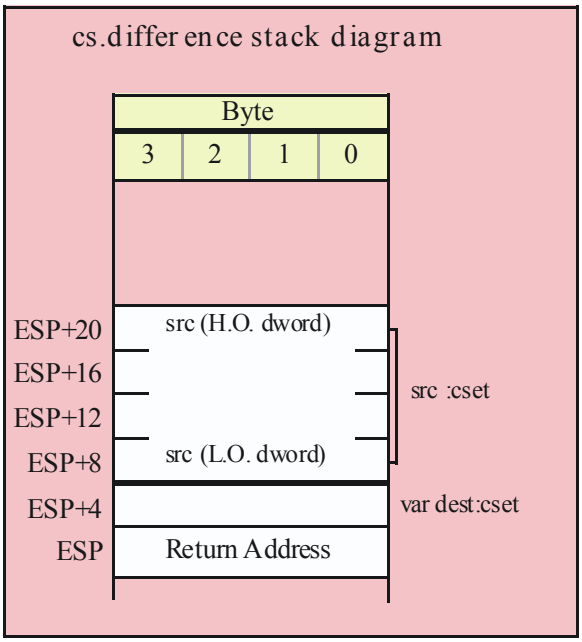



**procedure cs.difference( src:cset; var dest:cset );**

This function computes the set difference of two sets (i.e., the members in the destination set that are not also members of the source set). It stores the result back into the dest set (which is passed by reference).

```
HLA high-level calling sequence examples:

cs.difference( csetSrc, csetDest );
cs.difference( {'a'..'z'}, csetWithoutLowerCase );


HLA low-level calling sequence example:

// The following low-level example assumes that csetDest_s is
// a statically-declared object (static or storage section). For
// examples of additional ways to pass the destination cset by
// reference, please see the examples for cs.cpy given earlier.
```

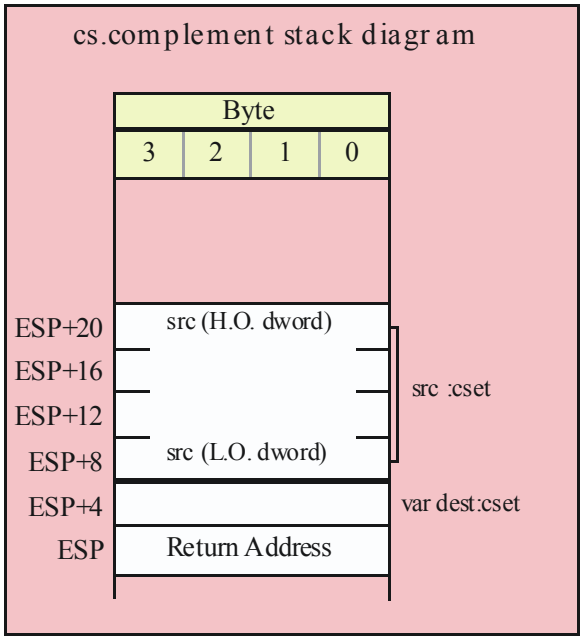
```
push( (type dword csetSrc_s[12]));
push( (type dword csetSrc_s[8]));
push( (type dword csetSrc_s[4]));
push( (type dword csetSrc_s[0]));
pushd( &csetDest_s );
call cs.difference;
```




**procedure cs.complement( src:cset; var dest:cset );**

This function computes the set complement of a set (i.e., the members in the destination set are those elements that are not in the source set.). It stores the complemented version of the set in the destination operand (which is passed by reference).

```
HLA high-level calling sequence examples:

cs.complement( csetSrc, negatedCsetDest );
cs.complement( {'a'..'z'}, allButLowercase );


HLA low-level calling sequence example:

// The following low-level example assumes that csetDest_s is
// a statically-declared object (static or storage section). For
// examples of additional ways to pass the destination cset by
// reference, please see the examples for cs.cpy given earlier.

push( (type dword csetSrc_s[12]));
push( (type dword csetSrc_s[8]));
push( (type dword csetSrc_s[4]));
push( (type dword csetSrc_s[0]));
pushd( &csetDest_s );
call cs.complement;
```

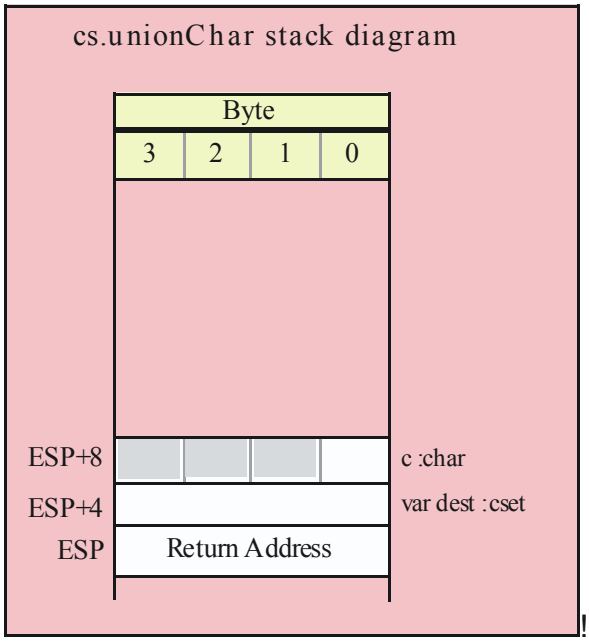



**procedure cs.unionChar( c:char; var dest:cset );**

The cs.unionChar function adds the character (supplied as a parameter) to the specified destination character set (passed by reference). If the character was already a member of the set, this function does not affect the character set.

```
HLA high-level calling sequence examples:

cs.unionChar( 'c', csetDest );
cs.unionChar( charVar, lowerCaseCset );
cs.unionChar( (type char [esi]), (type cset [edi]));


HLA low-level calling sequence examples:

// The following low-level examples all assume that csetDest_s is
// a statically-declared object (static or storage section). For
// examples of additional ways to pass the destination cset by
// reference, please see the examples for cs.cpy given earlier.
//
// Passing a single character constant:

pushd( 'c' );
pushd( &csetDest_s );
call cs.unionChar;

// Passing a character variable, assuming a 32-bit register is
// available or one of the 8-bit register: AL, BL, CL, or DL

mov( charVar, al );
push( eax );
pushd( &csetDest_s );
call cs.unionChar;
```

```
// If the character variable is guaranteed not to be in the last
// three bytes of allocated storage, you could also do this:

push( type dword charVar );
pushd( &csetDest_s );
call cs.unionChar;

// If the character is in one of the 8-bit registers: AH, BH, CH, DH

sub( 4, esp );
mov( ah, [esp] );
pushd( &csetDest_s );
call cs.unionChar;
```



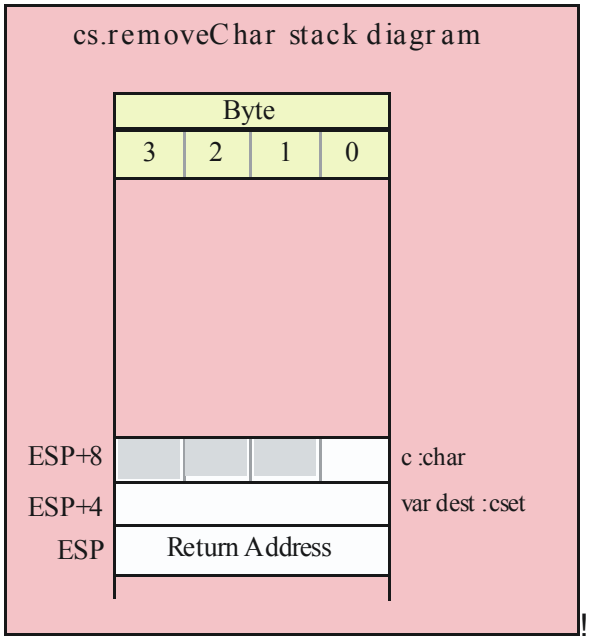

!

**procedure cs.removeChar( c:char; var dest:cset );**

This function removes a single character from the specified destination set (passed by reference). If the character was not previously a member of the destination set, this function does not affect that set.

```
HLA high-level calling sequence examples:

cs.removeChar( 'c', csetDest );
cs.removeChar( charVar, lowerCaseCset );
cs.removeChar( (type char [esi]), (type cset [edi]));


HLA low-level calling sequence examples:

// The following low-level examples all assume that csetDest_s is
// a statically-declared object (static or storage section). For
// examples of additional ways to pass the destination cset by
// reference, please see the examples for cs.cpy given earlier.
//
```

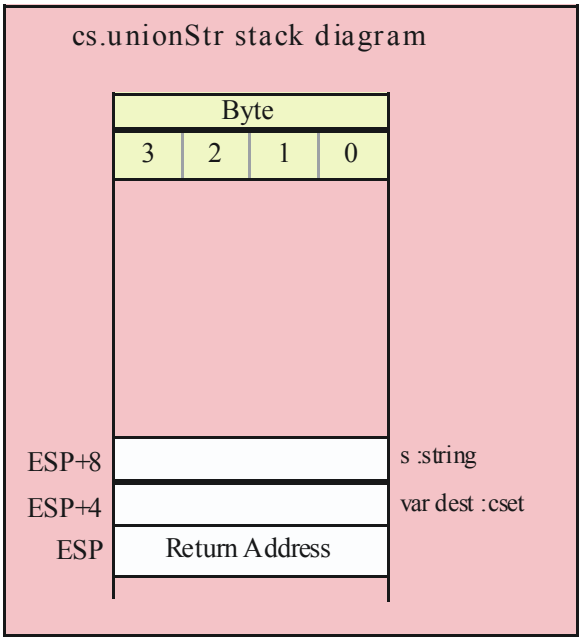
```
// Passing a single character constant:

pushd( 'c' );
pushd( &csetDest_s );
call cs.removeChar;

// Passing a character variable, assuming a 32-bit register is
// available or one of the 8-bit register: AL, BL, CL, or DL

mov( charVar, al );
push( eax );
pushd( &csetDest_s );
call cs.removeChar;

// If the character variable is guaranteed not to be in the last
// three bytes of allocated storage, you could also do this:

push( type dword charVar );
pushd( &csetDest_s );
call cs.removeChar;

// If the character is in one of the 8-bit registers: AH, BH, CH, DH

sub( 4, esp );
mov( ah, [esp] );
pushd( &csetDest_s );
call cs.removeChar;
```




**procedure cs.unionStr( s:string; var dest:cset );**

This function will union in all the characters in a string to the destination set. Unlike the cs.strToCset function, this function does not clear the destination character set before processing the characters in the string.

```
HLA high-level calling sequence examples:

cs.unionStr( strSrc, csetDest );
cs.unionStr( "ABCDEFabcdef", csetPlusHexChars );


HLA low-level calling sequence examples:

// csetDest_s is a variable declared
// in the static/storage section:

push( strSrc );
pushd( &csetDest_s );
call cs.unionStr;

// You could pass a string literal thusly (though there is
// no benefit to doing this over creating a statically
// initialized string variable and passing that string variable).

lea( eax, "abcdefABCDEF" );
push( eax );
pushd( &csetDest_s );
call cs.unionStr;
```

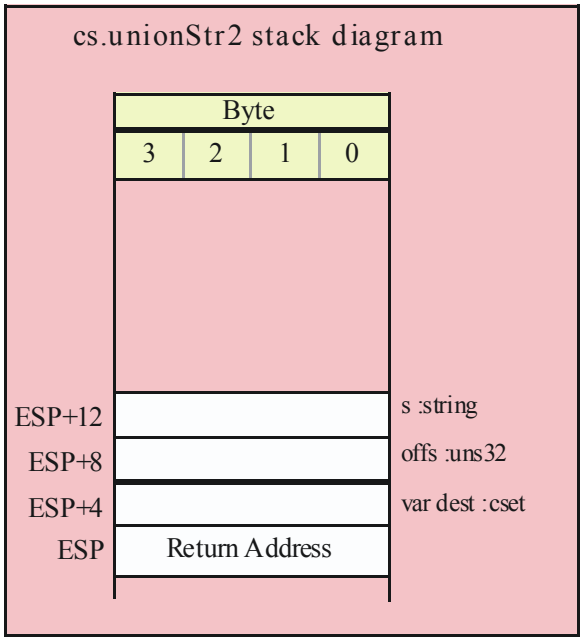



**procedure cs.unionStr2( s:string; offs:uns32;  offs:uns32; var dest:cset );**

This function will union in all the characters in a string to the destination set. Unlike the cs.unionStr function, this function starts at character position offs in s rather than at character position zero.

```
HLA high-level calling sequence examples:

cs.unionStr2( strSrc, 2, csetDest );
cs.unionStr2( "ABCDEF", offsetIntoStr, partialHexUnion );
```

```
HLA low-level calling sequence examples:

// csetDest_s is a variable declared
// in the static/storage section:

push( strSrc );
pushd( 2 )
pushd( &csetDest_s );
call cs.unionStr2;

// Assume the offset is in the variable "offsetIntoStr":

push( strSrc );
push( offsetIntoStr );
pushd( &csetDest_s );
call cs.unionStr2;
```

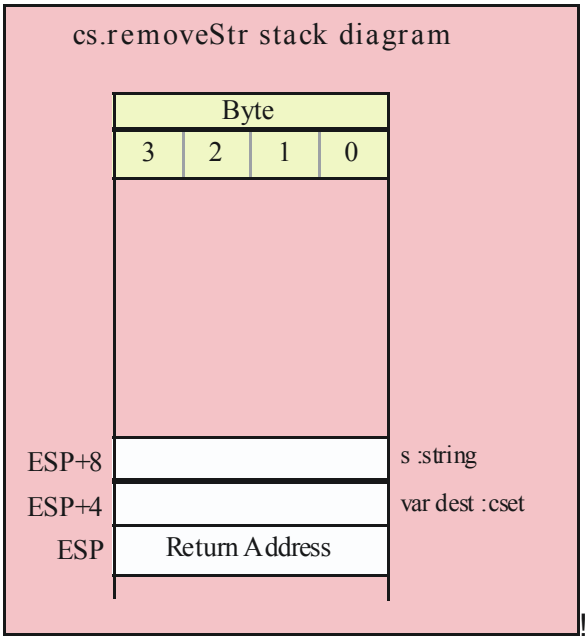



**procedure cs.removeStr( s:string; var dest:cset );**

This function removes characters found in the string from the specified character set. If a character in the string was not previously a member of the character set, the specified character has no effect on the destination set.

```
HLA high-level calling sequence examples:

cs.removeStr( strSrc, csetDest );
cs.removeStr( "ABCDEFabcdef", csetMinusHexChars );


HLA low-level calling sequence examples:

// csetDest_s is a variable declared
// in the static/storage section:
```

```
push( strSrc );
pushd( &csetDest_s );
call cs.removeStr;

// You could pass a string literal thusly (though there is
// no benefit to doing this over creating a statically
// initialized string variable and passing that string variable).

lea( eax, "abcdefABCDEF" );
push( eax );
pushd( &csetDest_s );
call cs.removeStr;
```




**procedure cs.removeStr2( s:string; offs:uns32; var dest:cset );**

This function removes characters found in the string at character position offs and beyond from the specified character set. If a character in the string was not previously a member of the character set, the specified character has no effect on the destination set.

```
HLA high-level calling sequence examples:

cs.removeStr2( strSrc, 2, csetDest );
cs.removeStr2( "ABCDEF", offsetIntoStr, csetMinusSomeHexChars );


HLA low-level calling sequence examples:

// csetDest_s is a variable declared
// in the static/storage section:

push( strSrc );
pushd( 2 )
pushd( &csetDest_s );
call cs.removeStr2;
```

```
// Assume the offset is in the variable "offsetIntoStr":

push( strSrc );
push( offsetIntoStr );
pushd( &csetDest_s );
call cs.removeStr2;
```