CS271 Irvine Library Procedures List

CloseFile	2	ReadHex	18
Clrscr	2	ReadInt	18
CreateOutputFile	3	ReadKey	19
Crlf	3	ReadString	20
Delay	4	SetTextColor	21
DumpMem	4	ShowFPUStack	22
DumpRegs	5	Str_compare	23
GetCommandTail	5	Str_copy	24
GetDateTime	6	Str_length	25
GetMaxXY	7	Str_trim	26
GetMseconds	8	Str_ucase	26
GetTextColor	9	WaitMsg	27
Gotoxy	9	WriteBin	27
IsDigit	10	WriteBinB	28
MsgBox	10	WriteChar	28
MsgBoxAsk	11	WriteDec	29
OpenInputFile	12	WriteFloat	29
ParseDecimal32	12	WriteHex	30
ParseInteger32	13	WriteHexB	30
Random32	13	WriteInt	31
Randomize	14	WriteStackFrame	31
RandomRange	14	WriteStackFrameName	32
ReadChar	15	WriteString	33
ReadDec	15	WriteToFile	34
ReadFloat	16	WriteWindowsMsg	34
ReadFromFile	17		

Oregon State University
College of Engineering

CloseFile

Closes a disk file that was previously opened

receives:
EAX = file handle
returns:
EAX = return code (0 if error)

Example code:

```
mov  eax,fileHandle
call CloseFile
```

Clrscr

Clears the console window

receives:
None
returns:
None

Example code:

```
call WaitMsg        ; "Press any key..."
call Clrscr
```

CreateOutputFile

Creates a new disk file for writing in output mode.

receives:

EDX = address of filename

returns:

EAX = file handle (INVALID_HANDLE_VALUE if error)

*ECX may be changed by this procedure

Example code:

```
.data
filename BYTE "newfile.txt",0

.code
mov  edx,OFFSET filename
call CreateOutputFile
cmp  eax, INVALID_HANDLE_VALUE
je   Error
```

Crlf

Writes an end-of-line sequence to the console window.

receives:

None

returns:

None

Example code:

```
call Crlf
```

Delay

Pauses the program for a number of milliseconds.

receives:

EAX = number of milliseconds

returns:

None

Example code:

```
mov  eax,1000       ; 1 second
call Delay
```

DumpMem

Writes a range of memory to the console window in hexadecimal.

receives:

ESI = starting address

ECX = number of units

EBX = unit size (1,2, or 4)

returns:

None

Example code:

```
.data
array DWORD 1,2,3,4,5,6,7,8,9,0Ah,0Bh

.code
mov  esi,OFFSET array        ; starting OFFSET
mov  ecx,LENGTHOF array      ; number of units
mov  ebx,TYPE array          ; doubleword format
call DumpMem
```

DumpRegs

Displays the EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP, EIP, and EFL (EFLAGS) registers in hexadecimal.

receives:
None
returns:
None

Example code:

```
call DumpRegs
```

GetCommandTail

Copies the program's command line into a null-terminated string.

receives:
EDX = address of array ; array must be at least 129 bytes
returns:
None

Example code:

```
.data
cmdTail BYTE 129 DUP(0)     ; empty buffer

.code
mov  edx,OFFSET cmdTail
call GetCommandTail         ; fills the buffer
```

GetDateTime

Gets current local date and time, stored in a 64-bit integer in Win32 FILETIME format.

receives:
  PTR QWORD - reference to store datetime
returns:
  PTR QWORD - date and time in FILETIME format

Example code:

```
.data
time        QWORD ?
tenmil      DWORD 10000000
shift       REAL8 11644473600.
unixtime    DWORD ?

.code
push  offset time
call  GetDateTime

; convert Win32 FILETIME to Unix time
finit
fild  time
fidiv tenmil
fsub  shift
fist  unixtime

; display time
mov   EAX unixtime
call  WriteDec
```

GetMaxXY

Gets the size of the console window's buffer.

receives:

None

returns:

DX = number of columns ; max 255

AX = number of rows ; max 255

Example code:

```
.data
rows BYTE ?
cols BYTE ?

.code
call GetMaxXY
mov rows,al
mov cols,dl
```

GetMseconds

Gets the number of milliseconds elapsed since midnight on the host computer.

receives:

None

returns:

EAX = time in milliseconds

Example code:

```
.data
    startTime DWORD ?

.code
    call GetMseconds
    mov  startTime,eax

L1:
    loop L1
    call GetMseconds
    sub  eax,startTime      ; EAX = loop time, in
milliseconds
```

GetTextColor

Gets the current foreground and background colors of the console window.

receives:

None

returns:

AL = color ; upper 4 bits = background, lower 4 bits = foreground

Example code:

```
.data
color byte ?

.code
call GetTextColor
mov  color,AL
```

Gotoxy

Move the cursor at a given row and column in the console window.

receives:

DH = X coordinate (column)

DL = Y coordinate (row)

returns:

None

Example code:

```
mov  dh,10          ; row 10
mov  dl,20          ; column 20
call Gotoxy         ; locate cursor
```

IsDigit

Determines whether the value in AL is the ASCII code for a valid decimal digit.

receives:

AL = character

returns:

Zero flag is set if valid digit, else clear

Example code:

```
mov  AL,somechar
call IsDigit
```

MsgBox

Displays a graphical popup message box with an optional caption.

receives:

EDX = address of message string

EBX = address of title string ; 0 for blank

returns:

None

Example code:

```
.data
caption  BYTE "Dialog Title", 0
HelloMsg BYTE "This is a pop-up message box.", 0dh,0ah
         BYTE "Click OK to continue...", 0

.code
mov  ebx, OFFSET caption
mov  edx, OFFSET HelloMsg
call MsgBox
```

MsgBoxAsk

Displays a graphical popup message box with Yes and No buttons.

receives:

EDX = address of question string

EBX = address of title string ; 0 for blank

returns:

EAX = answer ; IDYES (6) or IDNO (7)

Example code:

```
.data
caption  BYTE "Survey Completed",0
question BYTE "Thank you for completing the survey."
         BYTE 0dh,0ah
         BYTE "Would you like to receive the results?",0

.code
mov  ebx, OFFSET caption
mov  edx, OFFSET question
call MsgBoxAsk
;(check return value in EAX)
```

OpenInputFile

Opens an existing file for input.

receives:

EDX = address of filename

returns:

EAX = file handle ; INVALID_HANDLE_VALUE if file open failed

Example code:

```
.data
filename BYTE "myfile.txt",0

.code
mov  edx,OFFSET filename
call OpenInputFile
```

ParseDecimal32

Converts an unsigned decimal integer string to 32-bit binary.

receives:

EDX = address of string

ECX = length of string

returns:

EAX = parsed integer

Example code:

```
.data
buffer BYTE "8193"
bufSize = ($ - buffer)

.code
mov  edx,OFFSET buffer
mov  ecx,bufSize
call ParseDecimal32 ; returns EAX
```

ParseInteger32

Converts an signed decimal integer string to 32-bit binary.

receives:

EDX = address of string

ECX = length of string

returns:

EAX = parsed integer

Example code:

```
.data
buffer BYTE "-8193"
bufSize = ($ - buffer)

.code
mov  edx,OFFSET buffer
mov  ecx,bufSize
call ParseInteger32 ; returns EAX
```

Random32

Generates and returns a 32-bit random integer.

receives:

None

returns:

EAX = random integer

Example code:

```
.data
randVal DWORD ?

.code
call Random32
mov  randVal,eax
```

Randomize

Initializes the starting seed value of the Random32 and RandomRange procedures.

receives:
None
returns:
None

Example code:

```
call Randomize
```

RandomRange

Produces a random integer within a range.

receives:
EAX = upper limit (exclusive)
returns:
EAX = random integer

Example code:

```
.data
randVal DWORD ?

.code
mov  eax, 5000
call RandomRange    ; generates 0 - 4999
mov  randVal, eax
```

ReadChar

Reads a single character from the keyboard.

receives:
None

returns:
AL = character
AH = scan code (optional ; if extended key pressed)

Example code:

```
.data
char BYTE ?

.code
call ReadChar
mov  char,al
```

ReadDec

Reads a 32-bit unsigned decimal integer from the keyboard.

receives:
None

returns:
EAX = unsigned integer
CF = 1 if value is zero or invalid, else 0

Example code:

```
.data
intVal DWORD ?

.code
call ReadDec
mov  intVal,eax
```

ReadFloat

Read a floating-point value from the keyboard and push it onto the FPU stack.

receives:

None

returns:

ST(0) = user entered floating-point value

Example code:

```
finit                  ; initialize FPU
call ReadFloat         ; get first float
call ReadFloat         ; get second float
fmul ST(0),ST(1)       ; multiply
call WriteFloat        ; first x second
```

ReadFromFile

Reads an input disk file into a memory buffer.

receives:

EAX = open file handle

EDX = address of buffer

ECX = buffer size

returns:

EAX = bytes read (CF = 0) or error (CF = 1)

CF = error indicator

Example code:

```
.data
BUFFER_SIZE = 5000
buffer BYTE BUFFER_SIZE DUP(?)
bytesRead DWORD ?

.code
mov  eax, fileHandle     ; open file handle
mov  edx, OFFSET buffer  ; points to buffer
mov  ecx, BUFFER_SIZE    ; max bytes to read
call ReadFromFile        ; read the file
```

ReadHex

Reads a 32-bit hexadecimal integer from the keyboard.

receives:

None

returns:

EAX = integer

Example code:

```
.data
hexVal DWORD ?

.code
call ReadHex
mov  hexVal,eax
```

ReadInt

Reads a 32-bit signed decimal integer from the keyboard.

receives:

None

returns:

EAX = integer

Example code:

```
.data
intVal SDWORD ?

.code
call ReadInt
mov  intVal,eax
```

ReadKey

Performs a no-wait keyboard check to see if any key has been pressed.

receives:

None

returns:

AL = ASCII code or 0 (if special key)

AH = scan code (if AL = 0)

DX = virtual key code (if AL = 0)

EBX = keyboard flag bits (if AL = 0)

ZF = 0 (key pressed) or 1 (no key)

Example code:

```
.data
pressedKey BYTE ?

.code
    call ReadKey
    cmp ZF, 0
    jne NoKey
    mov pressedKey, AL
NoKey:
```

ReadString

Reads a string from the keyboard, stopping when the user presses the Enter key.

receives:

EDX = address of buffer

ECX = buffer size

returns:

EDX = address of user string

EAX = number of characters entered

Example code:

```
.data
buffer    BYTE 21 DUP(0) ; input buffer
byteCount DWORD ?        ; holds counter

.code
mov  edx,OFFSET buffer   ; point to the buffer
mov  ecx,SIZEOF buffer   ; specify max characters
call ReadString          ; input the string
mov  byteCount,eax       ; number of characters
```

SetTextColor

Sets the foreground and background colors for text output.

receives:

EAX = colors

returns:

None

Example code:

```
; foreground color + (background color x 16)
mov eax, white + (blue * 16)   ; white on blue
call SetTextColor
```

Color values are:

black	0
blue	1
green	2
cyan	3
red	4
magenta	5
brown	6
lightGray	7
gray	8
lightBlue	9
lightGreen	10
lightCyan	11
lightRed	12
lightMagenta	13
yellow	14
white	15

ShowFPUStack

Display the contents of the FPU stack.

receives:

None

returns:

None

Example code:

```
.data
first   REAL8 123.456
second  REAL8 10.0

.code
finit           ; initialize FPU
fld   first
fld   second
call  ShowFPUStack
```

Str_compare

Compares two strings, setting the Zero and Carry flags.

receives:

PTR BYTE - first string

PTR BYTE - second string

returns:

CF and ZF are set according to the CMP instruction

Example code:

```
.data
stringA   BYTE "abcde", 0
stringB   BYTE "xyz", 0

.code
push offset stringA
push offset stringB
call Str_compare
```

Str_copy

Copy a string.

receives:

PTR BYTE - source string

PTR BYTE - target string

returns:

PTR BYTE - target string copied from source

Example code:

```
.data
oldString   BYTE "abcde",0
newString   BYTE LENGTHOF oldString dup(0)

.code
push offset newString
push offset oldString
call Str_copy           ; newString = oldString
```

Str_length

Returns the length of a null-terminated string.

receives:

EDX = address of string

returns:

EAX = string length

Example code:

```
.data
buffer    BYTE "abcde",0
bufLength DWORD ?

.code
mov  edx,OFFSET buffer  ; point to string
call Str_length         ; EAX = 5
mov  bufLength,eax      ; save length
```

Str_trim

Removes occurrences of a character from the end of a string.

receives:

PTR BYTE - string to trim

CHAR - character to remove

returns:

PTR BYTE - trimmed string

Example code:

```
.data
string   BYTE "abcde###", 0
target   CHAR '#'

.code
push target
push offset string
call Str_trim
```

Str_ucase

Converts string to upper case.

receives:

PTR BYTE - string to convert

returns:

PTR BYTE - upper case string

Example code:

```
.data
string   BYTE "abcde", 0

.code
push offset string
call Str_ucase
```

WaitMsg

Displays the message “Press any key to continue. . .” and waits for the user to press a key.

receives:

    None

returns:

    None

Example code:

```
call WaitMsg
```

WriteBin

Writes an integer to the console window in ASCII binary format.

receives:

    EAX = integer

returns:

    None

Example code:

```
mov  eax,12346AF9h
call WriteBin
```

WriteBinB

Writes a 32-bit integer to the console window in ASCII binary format.

receives:

EAX = integer

EBX = display size (1,2, or 4)

returns:

None

Example code:

```
mov  eax, 1234h
mov  ebx, TYPE WORD
call WriteBinB
```

WriteChar

Displays a character to the output.

receives:

AL = character

returns:

None

Example code:

```
.data
myChar CHAR '+'

.code
mov  al, '+'
call WriteChar
```

WriteDec

Displays a 32-bit unsigned integer to output.

receives:

EAX = integer

returns:

None

Example code:

```
mov  eax, 256
call WriteDec
```

WriteFloat

Write the floating-point value from ST(0) to the output.

receives:

ST(0)

returns:

None

Example code:

```
.data
val REAL8 25.0

.code
finit                ; initialize FPU
fld val              ; push val to ST(0)
call WriteFloat
```

WriteHex

Writes a 32-bit unsigned integer to output in 8-digit hexidecimal.

receives:

EAX = integer

returns:

None

Example code:

```
mov  eax,7FFFh
call WriteHex       ; displays: "00007FFF"
```

WriteHexB

Writes a 32-bit unsigned integer to output in hexidecimal.

receives:

EAX = integer

EBX = display size (1,2, or 4)

returns:

None

Example code:

```
mov  eax,7FFFh
mov  ebx, TYPE WORD
call WriteHexB       ; displays: "7FFF"
```

WriteInt

Displays a 32-bit signed integer to output.

receives:

EAX = integer

returns:

None

Example code:

```
.data
myInt SWORD 216543

.code
mov  eax, myInt
call WriteInt       ; displays: "+216543"
```

WriteStackFrame

Writes the stack frame of a procedure.

receives:

DWORD - number of parameters passed

DWORD - number of DWORD local variables

DWORD - number of saved registers

returns:

None

Example code:

```
myProc PROC USES ebx, ecx, edx
    val:DWORD
    LOCAL a:DWORD, b:DWORD

.code
; inside myProc ...
INVOKE WriteStackFrame, 1, 2, 3
```

WriteStackFrameName

Writes the stack frame of a procedure with the procedure name.

receives:

DWORD - number of parameters passed

DWORD - number of DWORD local variables

DWORD - number of saved registers

PTR BYTE - reference to procedure name

returns:

None

Example code:

```
myProc PROC USES ebx, ecx, edx
    val:DWORD
    LOCAL a:DWORD, b:DWORD

.data
procName BYTE "myProc", 0

.code
; inside myProc ...
INVOKE WriteStackFrameName, 1, 2, 3, ADDR procName
```

WriteString

Writes a null-terminated string to output.

receives:

EDX = address of string

returns:

None

Example code:

```
.data
prompt BYTE "Enter your name: ", 0

.code
mov  edx, OFFSET prompt
call WriteString        ; "Enter your name: "
call Crlf
```

WriteToFile

Writes the contents of a buffer to an output file. Use with OpenInputFile procedure.

receives:

EAX = file handle

EDX = address of buffer

ECX = number of bytes to write

returns:

EAX = number of bytes written (0 if error)

Example code:

```
BUFFER_SIZE = 5000
.data
fileHandle DWORD ?
buffer BYTE BUFFER_SIZE DUP(?)

.code
mov  eax, fileHandle
mov  edx, OFFSET buffer
mov  ecx, BUFFER_SIZE
call WriteToFile
```

WriteWindowsMsg

Writes a string containing the most recent error generated by your application to the output when executing a call to a system function.

receives:

None

returns:

None

Example code:

```
call WriteWindowsMsg
```