Practical Reverse Engineering Exercises - Write Ups
Chapter 1 – Exercise 3 – part2 (question 4) (20th of July 2014)

TASK

"4. Implement the following functions in x86 assembly: strlen, strchr, memcpy, memset, strcmp, strset."

Excerpt from: *"Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation",*

Bruce Dang, Alexandre Gazet, Elias Bachaalany, Sebastien Josse, ISBN: 978-1-118-78731-1

MY ANSWERS

STRLEN ASSEMBLY VERSION

C function prototype: size_t strlen(const char *str);

The following function is included in **ex3_4-strlen.asm** file in github.

strlenAsm – my version of the function

```
strlenAsm:
        ;function equivalent to strlen C size_t strlen(const char *str)
        push    ebp
        mov     ebp, esp

        mov     edi, [ebp+8]        ;move the function argument into EDI
        xor     eax, eax ; we will search for 0 terminator
        or      ecx, 0FFFFFFFFh  ; set ECX to -1
        repne   scasb
        add     ecx, 2
        neg     ecx
        mov     eax, ecx ; move the result to eax
        mov     esp, ebp
        pop     ebp
        retn
```

STRCHR ASSEMBLY VERSION

C function prototype: char *strchr(const char *s, int c);

The following function is included in **ex3_4-strchr.asm** file in github.

Author: ePsiLoN (info at epsilon-labs dot com)

strchrAsm – my version of the function

```
strchrAsm:
        ;function analogue to  char *strchr(const char *s, int c); which locates the first occurrence of a char in a string
        ; prologue
        push    ebp
        mov     ebp, esp
        push    edi
        push    edx

        ;begin the function body
        mov     edi, [ebp+8]
        mov     edx, edi  ;save it so we can restore it later
        xor     eax, eax ; we are going to search null char
        or      ecx, 0FFFFFFFFh ; set EAX -1
        repne   scasb
        add     ecx, 1
        neg     ecx                 ; the length of the string including the null character
        mov     edi, edx  ; move back the EDI to the firts char
        mov     al, [ebp+0Ch]       ; load the second parameter into AL
        repne   scasb
        test    ecx, ecx  ; check if we reached the end of the string without exit i.e. no match
        jz      notFound
        mov     eax, edi  ; the edi will point to the
        jmp     commonEnd
 notFound:
        xor     eax, eax
 commonEnd:
        ;epilogue
        pop     edx
        pop     edi
```

Author: ePsiLoN (info at epsilon-labs dot com)

```
        mov     esp, ebp
        pop     ebp
        retn
```

MEMCPY ASSEMBLY VERSION

C function prototype: void *memcpy(void *dest, const void *src, size_t n);

The following function is included in **ex3_4-memcpy.asm** file in github.

memcpyAsm – my version of the function

```
memcpyAsm:
        ; function implementing void *memcpy(void *dest, const void *src, size_t n);
        ; the function copies n bytes from memory area src to memory area dest and returns a pointer to dest.

        ;prologue
        push    ebp
        mov     ebp, esp

        ;function body
        mov     edi, [ebp+8]        ;load dest address in EDI
        mov     edx, edi ; store for further use
        mov     esi, [ebp+0Ch]      ;load src address in  ESI
        mov     ecx, [ebp+10h]      ; load n (number of bytes to copy) in ECX
        rep     movsb               ; move byte from src to dest n times
        mov     eax, edx

        ;epilogue
        mov     esp,ebp
        pop     ebp
        retn
```

Author: ePsiLoN (info at epsilon-labs dot com)

MEMSET ASSEMBLY VERSION

C function prototype: void *memset(void *s, int c, size_t n);

The following function is included in **ex3_4-memset.asm** file in github.

memsetAsm – my version of the function

```
memsetAsm:

        ; function implementing void *memset(void *s, int c, size_t n) which fills the first n bytes of the memory area pointed to by s with the constant byte c


        ;prologue
        push    ebp
        mov     ebp, esp
        ; function body
        mov     ecx, [ebp+010h]  ; load the number of bytes n in ECX
        mov     al,  [ebp+0Ch]   ; load the c in AL
        mov     edi, [ebp+8]     ; load the target string in EDI
        rep     stosb


        ;epilogue
        mov     esp, ebp
        pop     ebp
        retn
```

STRCMP ASSEMBLY VERSION

C function prototype: int strcmp(const char *s1, const char *s2);

The following function is included in **ex3_4-strcmp.asm** file in github.

strcmpAsm – my version of the function

```
strcmpAsm:
```

```
        ; function implementing int strcmp(const char *s1, const char *s2);
        ; The function compares the two strings s1 and s2 and It returns an integer less than, equal to, or greater than zero if s1 is found
        ; respectively, to be less than, to match, or be greater than s2.

        ;prologue
        push    ebp
        mov     ebp, esp

        ;function body
        ; first calculate the string length of both strings and compare them
        xor     eax, eax
        or      ecx, 0FFFFFFFFh
        mov     edi, [ebp+8]
        repne   scasb
        add     ecx, 2
        neg     ecx
        mov     edx, ecx
        mov     ecx, 0FFFFFFFFh
        mov     edi, [ebp+0Ch]
        repne   scasb
        add     ecx,2
        neg     ecx
        cmp     edx, ecx
        je      compareSymbols
        jg      s1abs2
 s1bels2:
        mov     eax, -1
        jmp     commonEnd
 s1abs2:
        mov     eax, 1
```

Author: ePsiLoN (info at epsilon-labs dot com)

```
        jmp     commonEnd
 s1eqs2:
        mov     eax, 0
        jmp     commonEnd
 compareSymbols:
        mov     esi, [ebp+8]
        mov     edi, [ebp+0Ch]
        repe    cmpsb
        je      s1eqs2
        ja      s1bels2
        jmp     s1abs2
commonEnd:
        mov     esp,ebp
        pop     ebp
        retn
```

STRSET ASSEMBLY VERSION

C function prototype: char *strset(char *string, int c);

The following function is included in **ex3_4-strset.asm** file in github.

strsetAsm – my version of the function

```
strsetAsm:
        ; function implementing char *strset(char *string, int c);
        ; the function replaces all characters form a srting with given character c

        ;prologue
        push    ebp
        mov     ebp, esp
        ; function body
```

```
;first find the string length
xor     eax, eax
or      ecx, 0FFFFFFFFh
mov     edi, [ebp+8]
mov     edx, edi
repne   scasb
add     ecx,2
neg     ecx
xor     eax, eax
mov     al,  [ebp+0Ch]     ; load the c in AL
mov     edi, [ebp+8]       ; load the target string in EDI
rep     stosb
mov     eax, edx

;epilogue
mov     esp, ebp
pop     ebp
retn
```