Source File: `~/4301/12/lab12.asm`
Input: Under control of `main` function
Output: Under control of `main` function
Value: 2

For $\Sigma = \{a, b\}$, construct a boolean assembly function that accepts a string of unknown length as its argument and determines if the string is a member of the set consisting of

$$\{x \mid x \in \{a, b\}^* \text{ and that contain an odd number of occurrences of the substring } aa\}$$

Note that *aaa* has two occurrences of *aa*. Valid strings include `aab`, `baa`, `aabbb`, `bbbaa`, `aabbbbb`, `bbbbbbbaa`, `aa`, `baa`, `bbaaaabaaa`, and `bababaabababa`, but not ϵ, `babab`, `abbba`, `bbbabbbabbb`, `a`, `b`, `ab`, `ba`, `bb`, `aaa`, `aba`, `bbb`, `abba`, `baba`, `babba`, `abbbba`, `bbabababa`, `babaaabaaa`, or `abaaaaaaaaaaa`.

A sample `main` function for testing your function is shown in Figure 1, a template for starting the assembly function is shown in Figure 2, and a sample execution sequence is shown in Figure 3.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int isValid(char *s);
5
6  int main()
7  {
8    char s[81];
9
10   while (fgets(s, sizeof(s), stdin))
11   {
12     // remove trailing newline from s
13     s[strcspn(s, "\n")] = 0;
14     printf("%s is %s\n", s, (isValid(s) ? "valid" : "invalid"));
15   }
16
17   return 0;
18 }
```

Figure 1. `/usr/local/4301/src/lab12main.c`

```
1  ; Your Name - CS 3304 - Lab 12
2
3  ;---------------------------------------------------------------------
4  global  isValid:function
5  ; Returns true if its argument is valid and false otherwise.
6  ; HLL prototype: int isValid(char *s);
7  ; Implements the following HLL function:
8  ;      int isValid(char *s)
9  ;      {
10 ;        int sum = 0;
11 ;        for ( ; *s != '\0'; ++s)
12 ;        {
13 ;          if (*s < 'a' || *s > 'b')
14 ;             return 0;
15 ;          if (*s == 'a' && *(s + 1) != '\0' && *(s + 1) == 'a')
16 ;            sum += 1;
17 ;        }
18 ;        return sum % 2 != 0;
19 ;      }
20 ; Receives: one 32-bit pointer containing the address of a string
21 ;           via the system stack
22 ; Returns:  a 32-bit integer in EAX
23 ;---------------------------------------------------------------------
24 isValid:
25 SECTION .text
26         push    ebp
27         mov     ebp,esp
28         sub     esp,4              ; create space on stack for one 32-bit int
29         pushad                     ; push eax, ecx, edx, ebx, esp, ebp, esi, edi
30 ; Stack looks like
31 ;
32 ;       | param *s    |   [ebp + 8]
33 ;       | return addr |   [ebp + 4]
34 ;       | ebp         |   ebp
35 ;       | local var   |   [ebp -  4] sum from above pseudocode
36 ;       | eax         |   [ebp -  8] from pushad
37 ;       | ecx         |   [ebp - 12] from pushad
38 ;       | edx         |   [ebp - 16] from pushad
39 ;       | ebx         |   [ebp - 20] from pushad
40 ;       | esp         |   [ebp - 24] from pushad
41 ;       | ebp         |   [ebp - 28] from pushad
42 ;       | esi         |   [ebp - 32] from pushad
43 ;       | edi         |   [ebp - 36] from pushad
44 ;
45         mov     dword [ebp-4],0    ; clear accumulator; sum in pseudocode
46         mov     esi,[ebp+8]        ; place addr of param in esi
47
```

Figure 2. `/usr/local/4301/src/lab12tmpl.asm` (Part 1 of 2)

```
48 ;
49 ;  Put your code here
50 ;
51
52         popad                       ; pop edi, esi, ebp, esp, ebx, edx, ecx, eax
53         mov     eax,[ebp-4]         ; move local accumulator to eax
54         add     esp,4               ; clean local var from stack
55         pop     ebp
56         ret
```

Figure 2. `/usr/local/4301/src/lab12tmpl.asm` (Part 2 of 2)

```
1  newuser@csunix ~> cd 4301
2  newuser@csunix ~/4301> mkdir 12
3  newuser@csunix ~/4301> cd 12
4  newuser@csunix ~/4301/12> # Copy the sample input and output files
5  newuser@csunix ~/4301/12> cp /usr/local/4301/data/12/* .
6  newuser@csunix ~/4301/12> ls
7  00.dat  00.out
8  newuser@csunix ~/4301/12> # Copy the main function
9  newuser@csunix ~/4301/12> cp /usr/local/4301/src/lab12main.c .
10 newuser@csunix ~/4301/12> ls
11 00.dat  00.out  lab12main.c
12 newuser@csunix ~/4301/12> # Copy the template for the assembly code
13 newuser@csunix ~/4301/12> # and save it as lab12.asm
14 newuser@csunix ~/4301/12> cp /usr/local/4301/src/lab12tmpl.asm lab12.asm
15 newuser@csunix ~/4301/12> # Edit lab12.asm to complete the code
16 newuser@csunix ~/4301/12> # Compile the C main function
17 newuser@csunix ~/4301/12> gcc -m32 -c lab12main.c
18 newuser@csunix ~/4301/12> # Assemble the assembly function
19 newuser@csunix ~/4301/12> nasm -f elf32 -o lab12.o lab12.asm
20 newuser@csunix ~/4301/12> # Invoke the linker to create the executable
21 newuser@csunix ~/4301/12> gcc -m32 -o lab12 lab12main.o lab12.o
22 newuser@csunix ~/4301/12> ls
23 00.dat  00.out  lab12  lab12.asm  lab12.o  lab12main.c  lab12main.o
24 newuser@csunix ~/4301/12> # Execute lab12 on dataset 00
25 newuser@csunix ~/4301/12> ./lab12 < 00.dat
26 aab is valid
27 aba is invalid
28 baa is valid
29 babab is invalid
30 aabbb is valid
31 bbbaa is valid
32 abbba is invalid
33 aabbbbb is valid
34 bbbbbbbaa is valid
35 bbbabbbabbb is invalid
```

Figure 3. Commands to Compile, Assemble, Link, & Run Lab 12 (Part 1 of 2)

```
 is invalid
a is invalid
b is invalid
aa is valid
ab is invalid
ba is invalid
bb is invalid
aaa is invalid
baa is valid
aba is invalid
bbb is invalid
abba is invalid
baba is invalid
babba is invalid
abbbba is invalid
bbabababa is invalid
bbaaaabaaa is valid
babaaabaaa is invalid
abaaaaaaaaaaa is invalid
bababaababababa is valid
newuser@csunix ~/4301/12> ./lab12 < 00.dat > my.out
newuser@csunix ~/4301/12> diff 00.out my.out
newuser@csunix ~/4301/12>
```

Figure 3. Commands to Compile, Assemble, Link, & Run Lab 12 (Part 2 of 2)