作業系統概論 hw4

學號: 408410113 姓名: 王 X 彥

1. 撰寫一支程式名為：「stdin_read」，在這個程式中使用組合語言呼叫system call，從stdin讀進一個字元，假設讀入的字元為a，隨後使用printf在螢幕上印出『讀入的字元為“a”』

2. 將程式碼反組譯，然後「大致」解釋組語的意義

```
Dump of assembler code for function main:
4   int main(int argc, char** argv) {
   0x0000000000401c4d <+0>:   push   %rbp
   0x0000000000401c4e <+1>:   mov    %rsp,%rbp
   0x0000000000401c51 <+4>:   push   %rbx
   0x0000000000401c52 <+5>:   sub    $0x38,%rsp
   0x0000000000401c56 <+9>:   mov    %edi,-0x34(%rbp)
   0x0000000000401c59 <+12>:  mov    %rsi,-0x40(%rbp)
   0x0000000000401c5d <+16>:  mov    %fs:0x28,%rax
   0x0000000000401c66 <+25>:  mov    %rax,-0x18(%rbp)
   0x0000000000401c6a <+29>:  xor    %eax,%eax                //將reg原本的資料放到記憶體，並把rax歸0

5       char *ch;
6       size_t size = 1;
   0x0000000000401c6c <+31>:  movq   $0x1,-0x20(%rbp)         //賦值

7       long ret;
8
9       printf("輸入一個字元:\n");
   0x0000000000401c74 <+39>:  lea    0x93389(%rip),%rdi        # 0x495004
   0x0000000000401c7b <+46>:  callq  0x418840 <puts>           //call puts function將buffer印出

10
11      __asm__ volatile (                          //將參數放到指定reg並呼叫syscall
   0x0000000000401c80 <+51>:  mov    $0x0,%rax
   0x0000000000401c87 <+58>:  mov    $0x1,%rdi
   0x0000000000401c8e <+65>:  mov    -0x30(%rbp),%rsi
   0x0000000000401c92 <+69>:  mov    -0x20(%rbp),%rdx
   0x0000000000401c96 <+73>:  syscall
   0x0000000000401c98 <+75>:  mov    %rax,-0x28(%rbp)

12          "mov $0, %%rax\n"   //system call number 0(read)
13          "mov $1, %%rdi\n"   //stdout (fd)
14          "mov %1, %%rsi\n"   //buffer (char *buf)
```

```
15          "mov %2, %%rdx\n"   //size (size_t count)
16          "syscall\n"
17          "mov %%rax, %0\n"
18          : "=m"(ret), "=m"(ch)
19          : "g"(size)
20          : "rax", "rbx", "rcx", "rdx");
21
22      getchar();
   0x0000000000401c9c <+79>:  callq  0x41a950 <getchar>
44
45  23
46  24      printf("讀入的字元是""%c""\n", *ch);
47     0x0000000000401ca1 <+84>:  mov    -0x30(%rbp),%rax
48     0x0000000000401ca5 <+88>:  movzbl (%rax),%eax
49     0x0000000000401ca8 <+91>:  movsbl %al,%eax
50     0x0000000000401cab <+94>:  mov    %eax,%esi
51     0x0000000000401cad <+96>:  lea    0x93364(%rip),%rdi        # 0x495018
52     0x0000000000401cb4 <+103>: mov    $0x0,%eax
53     0x0000000000401cb9 <+108>: callq  0x410b80 <printf>   //將要印出的訊息整理好放到buffer，並call printf function將buffer印出
54
55  25      printf("回傳值是%ld\n", ret);
56     0x0000000000401cbe <+113>: mov    -0x28(%rbp),%rax
57     0x0000000000401cc2 <+117>: mov    %rax,%rsi
58     0x0000000000401cc5 <+120>: lea    0x93362(%rip),%rdi        # 0x49502e
59     0x0000000000401ccc <+127>: mov    $0x0,%eax
60     0x0000000000401cd1 <+132>: callq  0x410b80 <printf>   //將要印出的訊息整理好放到buffer，並call printf function將buffer印出
61
62  26
63  27      return 0;
64     0x0000000000401cd6 <+137>: mov    $0x0,%eax           //將回傳值放入eax reg
```

```
65
66  28 }
67     0x0000000000401cdb <+142>: mov    -0x18(%rbp),%rdx
68     0x0000000000401cdf <+146>: xor    %fs:0x28,%rdx
69     0x0000000000401ce8 <+155>: je     0x401cef <main+162>
70     0x0000000000401cea <+157>: callq  0x454590 <__stack_chk_fail_local>
71     0x0000000000401cef <+162>: add    $0x38,%rsp
72     0x0000000000401cf3 <+166>: pop    %rbx
73     0x0000000000401cf4 <+167>: pop    %rbp
74     0x0000000000401cf5 <+168>: retq   //將一開始放到mem的值移回reg，並回傳
75
76  End of assembler dump.
```