システムプログラミング 2
期末レポート

氏名: 山田　敬汰 (Yamada，Keita)
学生番号: 09430559

出題日: 2019 年 12 月 9 日
提出日: 2019 年??月??日
締切日: 2020 年 1 月 27 日

1　概要

2　課題 2-1

2.1　課題内容

SPIM が提供するシステムコールを C 言語から実行できるようにしたい．教科書 A.6 節「手続き呼出し規約」に従って，各種手続きをアセンブラで記述せよ．ファイル名は，`syscalls.s` とすること．

また，記述した `syscalls.s` の関数を C 言語から呼び出すことで，ハノイの塔 (`hanoi.c` とする) を完成させよ．

2.2　C 言語で記述したプログラム例

```
 1 #include <stdio.h>
 2
 3 void hanoi(int n, int start, int finish, int extra)
 4 {
 5     if (n != 0)
 6     {
 7         hanoi(n - 1, start, extra, finish);
 8         print_string("Move disk ");
 9         print_int(n);
10         print_string(" from peg ");
11         print_int(start);
12         print_string(" to peg ");
13         print_int(finish);
14         print_string(".\n");
15         hanoi(n - 1, extra, finish, start);
16     }
```

```
17 }
18 main()
19 {
20     int n;
21     print_string("Enter number of disks> ");
22     n = read_int();
23     hanoi(n, 1, 2, 3);
24 }
```

2.3 作成したプログラム

```
 1 .text
 2 .align 2
 3
 4 _print_int:
 5  subu    $sp,$sp,24
 6  sw      $ra,20($sp)
 7
 8  li      $v0, 1
 9  syscall
10
11  lw      $ra,20($sp)
12  addu    $sp,$sp,24
13  j       $ra
14
15 _print_string:
16  subu    $sp,$sp,24
17  sw      $ra,20($sp)
18
19  li      $v0, 4
20  syscall
21
22  lw      $ra,20($sp)
23  addu    $sp,$sp,24
24  j       $ra
25
26 _read_int:
27  subu    $sp,$sp,24
28  sw      $ra,20($sp)
29
30  li      $v0, 5
31  syscall
32
33  lw      $ra,20($sp)
```

```
34  addu    $sp,$sp,24
35  j       $ra
36
37 _read_string:
38  subu    $sp,$sp,24
39  sw      $ra,20($sp)
40
41  li      $v0, 8
42  syscall
43
44  lw      $ra,20($sp)
45  addu    $sp,$sp,24
46  j       $ra
47
48 _exit:
49  li      $v0, 10
50  syscall
51
```

2.4 実行結果

```
Enter number of disks>  4
Move disk 1 from peg 1 to peg 3.
Move disk 2 from peg 1 to peg 2.
Move disk 1 from peg 3 to peg 2.
Move disk 3 from peg 1 to peg 3.
Move disk 1 from peg 2 to peg 1.
Move disk 2 from peg 2 to peg 3.
Move disk 1 from peg 1 to peg 3.
Move disk 4 from peg 1 to peg 2.
Move disk 1 from peg 3 to peg 2.
Move disk 2 from peg 3 to peg 1.
Move disk 1 from peg 2 to peg 1.
Move disk 3 from peg 3 to peg 2.
Move disk 1 from peg 1 to peg 3.
Move disk 2 from peg 1 to peg 2.
Move disk 1 from peg 3 to peg 2.
```

2.5 プログラムの解説

ここでは_print_int 部分での手続きを例に解説する.

1. スタックの領域を 24 バイト確保し，戻りアドレスを格納しておく．戻りアドレスを退避させている理由としては，syscall 内の手続きが OS に一任され，アセンブリのプログラムか

ら分からないようになっているからである.（つまり，OS が勝手に $ra レジスタの値を壊している可能性を考慮している.）

2. $v0 レジスタに適切な番号（_print_int の場合は 1）を格納し，syscall 命令を発行する．

3. スタックに格納しておいた戻りアドレスを $ra レジスタに再び格納し，呼び出し元に帰る．

その他の手続きも $v0 レジスタの値を変更することで，同様の手順で実行することができる．（OS によって抽象化されている.）

また，exit 手続きのみ，スタックに戻りアドレスを格納していない．その理由としては syscall の発行によってプロセスが終了するので，値を退避させたところで復帰させる方法がないからである．

2.6 考察

今回のプログラムでは syscall を呼び出す部分のみをアセンブリ言語で記述している．これは，C 言語の中から直接 syscall 命令を発行する方法が存在しないからである．そして，実行する時に C 言語の部分をアセンブリ言語にコンパイルし，syscalls.s と共に正しい順序でメモリ上に読み込むことで，プログラムを実行することができる．ここで，ファイル読み込みを間違えた場合は関数の参照先が未定義となり，プログラムが例外を発生する．

また，手続き呼び出し規約によって「引数はどのレジスタに入っていて，戻り値はこのレジスタに入っている」というのが決められているので，この規約を守っている限りは C 言語とアセンブリ言語との連携を円滑に行うことができる．言い換えれば，プログラマはコンパイラがどのように C 言語のプログラムを変換するのか（レジスタを決定するルール）を知っていなければアセンブリ言語を書くことができない，ということである．

3 課題 2-2

3.1 課題内容

hanoi.s を例に spim-gcc の引数保存に関するスタックの利用方法について，説明せよ．そのことは，規約上許されるスタックフレームの最小値 24 とどう関係しているか．このスタックフレームの最小値規約を守らないとどのような問題が生じるかについて解説せよ．

3.2 与えられたプログラム

```
 1        .file 1 "hanoi.c"
 2
        （中略）
13
14
15      .rdata
16      .align 0
17      .align 2
18 $LC0:
19      .ascii "Move disk \000"
```

```
20      .align 2
21 $LC1:
22      .ascii " from peg \000"
23      .align 2
24 $LC2:
25      .ascii " to peg \000"
26      .align 2
27 $LC3:
28      .ascii ".\n\000"
29      .text
30      .align 2
31      .set nomips16
32 _hanoi:
33      subu $sp,$sp,24
34      sw $ra,20($sp)
35      sw $fp,16($sp)
36      move $fp,$sp
37      sw $a0,24($fp)
38      sw $a1,28($fp)
39      sw $a2,32($fp)
40      sw $a3,36($fp)
41      lw $v0,24($fp)
42      beq $v0,$zero,$L3
43      lw $v0,24($fp)
44      addu $v0,$v0,-1
45      move $a0,$v0
46      lw $a1,28($fp)
47      lw $a2,36($fp)
48      lw $a3,32($fp)
49      jal _hanoi
50      la $a0,$LC0
51      jal _print_string
52      lw $a0,24($fp)
53      jal _print_int
54      la $a0,$LC1
55      jal _print_string
56      lw $a0,28($fp)
57      jal _print_int
58      la $a0,$LC2
59      jal _print_string
60      lw $a0,32($fp)
61      jal _print_int
62      la $a0,$LC3
63      jal _print_string
```

```
 64      lw $v0,24($fp)
 65      addu $v0,$v0,-1
 66      move $a0,$v0
 67      lw $a1,36($fp)
 68      lw $a2,32($fp)
 69      lw $a3,28($fp)
 70      jal _hanoi
 71 $L3:
 72      move $sp,$fp
 73      lw $ra,20($sp)
 74      lw $fp,16($sp)
 75      addu $sp,$sp,24
 76      j $ra
 77      .rdata
 78      .align 0
 79      .align 2
 80 $LC4:
 81      .ascii "Enter number of disks> \000"
 82      .text
 83      .align 2
 84      .set nomips16
 85 main:
 86      subu $sp,$sp,32
 87      sw $ra,28($sp)
 88      sw $fp,24($sp)
 89      move $fp,$sp
 90      la $a0,$LC4
 91      jal _print_string
 92      jal _read_int
 93      sw $v0,16($fp)
 94      lw $a0,16($fp)
 95      li $a1,1 # 0x1
 96      li $a2,2 # 0x2
 97      li $a3,3 # 0x3
 98      jal _hanoi
 99      move $sp,$fp
100      lw $ra,28($sp)
101      lw $fp,24($sp)
102      addu $sp,$sp,32
103      j $ra
```