システムプログラミング 1
レポート

氏名: 今田　将也 (IMADA, Masaya)
学生番号: 09430509

出題日: 2019 年 10 月 07 日
提出日: 2019 年 11 月 20 日
締切日: 2019 年 11 月 25 日

1　概要

本演習では，PIM という MIPS CPU シミュレータのハードウェア上に C 言語とアセンブリ言語を使用して文字の表示と入力のためのシステムコールライブラリを作成する． さらに，そのライブラリを使用して printf 及び gets 相当を C 言語で作成する． 最後に，それらを利用した応用プログラムを動作させる．

なお、与えられた課題内容を以下に述べる．

1.1　課題内容

以下の課題についてレポートをする． プログラムは，MIPS アセンブリ言語及び C 言語で記述し，SPIM を用いて動作を確認している．

2-1 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．

```
 1: void hanoi(int n, int start, int finish, int extra)
 2: {
 3:   if (n != 0){
 4:     hanoi(n - 1, start, extra, finish);
 5:     print_string("Move disk ");
 6:     print_int(n);
 7:     print_string(" from peg ");
 8:     print_int(start);
 9:     print_string(" to peg ");
10:     print_int(finish);
11:     print_string(".\n");
12:     hanoi(n - 1, extra, finish, start);
```

```
13:   }
14: }
15: main()
16: {
17:   int n;
18:   print_string("Enter number of disks> ");
19:   n = read_int();
20:   hanoi(n, 1, 2, 3);
21: }
```

spim-gcc によって hanoi.s ができたら， hanoi.s, syscalls.s の順に SPIM 上でロードして実行.

実行例は以下の通り:

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

2-2 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.

2-3 以下のプログラム report2-1.c をコンパイルした結果をもとに， auto 変数と static 変数の違い，ポインタと配列の違いについてレポートせよ.

```
 1: int primes_stat[10];
 2:
 3: char * string_ptr   = "ABCDEFG";
 4: char   string_ary[] = "ABCDEFG";
 5:
 6: void print_var(char *name, int val)
 7: {
 8:   print_string(name);
 9:   print_string(" = ");
10:   print_int(val);
11:   print_string("\n");
12: }
13:
14: main()
15: {
16:   int primes_auto[10];
```

```
17:
18:   primes_stat[0] = 2;
19:   primes_auto[0] = 3;
20:
21:   print_var("primes_stat[0]", primes_stat[0]);
22:   print_var("primes_auto[0]", primes_auto[0]);
23: }
```

2-4 printf など，一部の関数は，任意の数の引数を取ることができる． これらの関数を可変引数関数と呼ぶ． MIPS の C コンパイラにおいて可変引数関数の実現方法について考察し，解説せよ．

2-5 printf のサブセットを実装し， SPIM 上でその動作を確認する応用プログラム (自由なデモプログラム) を作成せよ． フルセットにどれだけ近いか，あるいは，よく使う重要な仕様だけをうまく切り出して， 実用的なサブセットを実装しているかについて評価する． ただし，浮動小数は対応しなくてもよい (SPIM 自体がうまく対応していない)． 加えて，この printf を利用した応用プログラムの出来も評価の対象とする．

1.2 xspim の実行方法

```
$ xspim -mapped_io&
```

でコンソール上で実行後，必要なアセンブリファイルを load し，run することで実行した．

1.3 c ソースコードからアセンブリファイルへの変換方法

```
$ spim-gcc file.c
```

でコンソール上で実行後，file.c に対応する file.s というアセンブリファイルが作られる．

2 課題レポート

2.1 課題 1-1

2.1.1 作成したプログラム

2.1.2 考察

3 感想