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 xx 月 xx 日
提出日: 20xx 年 xx 月 xx 日
締切日: 20xx 年 xx 月 xx 日

1　概要

本演習では，プログラミングに関する理解を深めるために不可欠なアセンブラと C 言語の境界部分についての演習を行った．具体的には，SPIM 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仕様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した．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た．

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 アセンブリ言語で記述し，SPIM を用いて動作を確認している．

課題 1-1 教科書 A.8 節 「入力と出力」に示されている方法と， A.9 節 最後「システムコール」に示されている方法のそれぞれで ”Hello World” を表示せよ．両者の方式を比較し考察せよ．

課題 1-2 アセンブリ言語中で使用する .data， .text および .align とは何か解説せよ． 下記コード中の 6 行目の .data がない場合，どうなるかについて考察せよ．

```
 1:          .text
 2:          .align  2
 3: _print_message:
 4:          la      $a0, msg
 5:          li      $v0, 4
 6:          .data
 7:          .align  2
 8: msg:
 9:          .asciiz "Hello!!\n"
10:          .text
11:          syscall
```

```
12:         j       $ra
13: main:
14:         subu    $sp, $sp, 24
15:         sw      $ra, 16($sp)
16:         jal     _print_message
17:         lw      $ra, 16($sp)
18:         addu    $sp, $sp, 24
19:         j       $ra
```

課題 1-3 教科書 A.6 節 「手続き呼出し規約」に従って，関数 fact を実装せよ．(以降の課題においては，この規約に全て従うこと) fact を C 言語で記述した場合は，以下のようになるであろう．

```
 1: main()
 2: {
 3:   print_string("The factorial of 10 is ");
 4:   print_int(fact(10));
 5:   print_string("\n");
 6: }
 7:
 8: int fact(int n)
 9: {
10:   if (n < 1)
11:     return 1;
12:   else
13:     return n * fact(n - 1);
14: }
```

課題 1-4 素数を最初から 100 番目まで求めて表示する MIPS のアセンブリ言語プログラムを作成してテストせよ． その際，素数を求めるために下記の 2 つのルーチンを作成すること．

関数名	概要
test_prime(n)	n が素数なら 1，そうでなければ 0 を返す
main()	整数を順々に素数判定し，100 個プリント

C 言語で記述したプログラム例:

```
 1: int test_prime(int n)
 2: {
 3:   int i;
 4:   for (i = 2; i < n; i++){
 5:     if (n % i == 0)
 6:       return 0;
 7:   }
 8:   return 1;
 9: }
```

```
10:
11: int main()
12: {
13:   int match = 0, n = 2;
14:   while (match < 100){
15:     if (test_prime(n) == 1){
16:       print_int(n);
17:       print_string(" ");
18:       match++;
19:     }
20:     n++;
21:   }
22:   print_string("\n");
23: }
```

実行結果（行を適当に折り返している）:

```
  2   3   5   7  11  13  17  19  23  29
 31  37  41  43  47  53  59  61  67  71
 73  79  83  89  97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229
233 239 241 251 257 263 269 271 277 281
283 293 307 311 313 317 331 337 347 349
353 359 367 373 379 383 389 397 401 409
419 421 431 433 439 443 449 457 461 463
467 479 487 491 499 503 509 521 523 541
```

課題 1-5 素数を最初から 100 番目まで求めて表示する MIPS のアセンブリ言語プログラムを作成してテストせよ． ただし，配列に実行結果を保存するように main 部分を改造し， ユーザの入力によって任意の番目の配列要素を表示可能にせよ．

C 言語で記述したプログラム例：

```
 1: int primes[100];
 2: int main()
 3: {
 4:   int match = 0, n = 2;
 5:   while (match < 100){
 6:     if (test_prime(n) == 1){
 7:       primes[match++] = n;
 8:     }
 9:     n++;
10:   }
11:   for (;;){
12:     print_string("> ");
```

```
13:     print_int(primes[read_int() - 1]);
14:     print_string("\n");
15:   }
16: }
```

実行例：

```
> 15
47
> 100
541
```

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

A.8 節「入力と出力」に示されている方法

```
        .text
        .align 2
main:
        li      $a0,72
putc:
        lw      $t0,0xffff0008
        li      $t1,1
        and     $t0,$t0,$t1
        beqz    $t0,putc
        sw      $a0,0xffff000c
        li      $a0,101
putc2:
        lw      $t0,0xffff0008
        li      $t1,1
        and     $t0,$t0,$t1
        beqz    $t0,putc2
        sw      $a0,0xffff000c
        li      $a0,108
putc3:
        lw      $t0,0xffff0008
        li      $t1,1
        and     $t0,$t0,$t1
        beqz    $t0,putc3
        sw      $a0,0xffff000c
        li      $a0,108
```

```
putc4:
        lw      $t0,0xffff0008
        li      $t1,1
        and     $t0,$t0,$t1
        beqz    $t0,putc4
        sw      $a0,0xffff000c
        li      $a0,111
putc5:
        lw      $t0,0xffff0008
        li      $t1,1
        and     $t0,$t0,$t1
        beqz    $t0,putc5
        sw      $a0,0xffff000c
        li      $a0,32

putc6:
        lw      $t0,0xffff0008
        li      $t1,1
        and     $t0,$t0,$t1
        beqz    $t0,putc6
        sw      $a0,0xffff000c
        li      $a0,87
putc7:
        lw      $t0,0xffff0008
        li      $t1,1
        and     $t0,$t0,$t1
        beqz    $t0,putc7
        sw      $a0,0xffff000c
        li      $a0,111
putc8:
        lw      $t0,0xffff0008
        li      $t1,1
        and     $t0,$t0,$t1
        beqz    $t0,putc8
        sw      $a0,0xffff000c
        li      $a0,114
putc9:
        lw      $t0,0xffff0008
        li      $t1,1
        and     $t0,$t0,$t1
        beqz    $t0,putc9
        sw      $a0,0xffff000c
        li      $a0,108
putc10:
```

```
        lw      $t0,0xffff0008
        li      $t1,1
        and     $t0,$t0,$t1
        beqz    $t0,putc10
        sw      $a0,0xffff000c
        li      $a0,100
putc11:
        lw      $t0,0xffff0008
        li      $t1,1
        and     $t0,$t0,$t1
        beqz    $t0,putc11
        sw      $a0,0xffff000c
        j       $ra
```

A.9 節「システムコール」に示されている方法

```
        .data
        .align 2
str:    .asciiz "Hello World"

        .text
        .align 2

main:   li  $v0,4   #print_str のシスコールを$v0 にロード
        la  $a0,str #プリントする文字列のアドレスを syscall の引数
                    #$a0 にロードアドレス命令を行う
        syscall
        j   $ra     #$ra レジスタへ戻り、プログラム終了
```

2.1.2 考察

2.2 課題 1-2

2.2.1 考察

2.3 課題 1-2

2.3.1 考察

2.4 課題 1-3

2.4.1 作成したプログラム

```
        .data
        .align 2
str:
        .ascii "The factorial of 10 is "
```

```
        .text
        .align 2

print_int:
        li      $v0,1
        syscall
        j       $ra

print_str:
        li      $v0,4
        syscall
        j       $ra

main:
        subu    $sp,$sp,32  #スタックフレームは３２バイト長で確保をする
        sw      $ra,20($sp) #戻りアドレスを退避させる
        sw      $fp,16($sp) #古いフレームポインタを退避
        addiu   $fp,$sp,28  #新しくフレームポインタを設定

        #fact を呼び出して戻ってから、syscall で$LC と fact の返り値をプリントする

        li      $a0,10      #引数は 10
        jal     fact

        #move    $t1,$v0     #返り値を a1 に退避
        #la      $a0,str     #a0 にテンプレ文のアドレスを記入
        #jal     print_str

        move    $a0,$v0     #fact の返り値を保存した t1 を$a0 に収める
        jal     print_int

        #退避してあったレジスタを復元したあと呼出側へ戻る

        lw      $ra,20($sp) #戻りアドレスを復元
        lw      $fp,16($sp) #フレームポインタを復元
        addiu   $sp,$sp,32  #スタックポインタをポップする
        j       $ra

fact:
        subu    $sp,$sp,32  #スタックフレームは３２バイト長
        sw      $ra,20($sp) #戻りアドレスを退避させる
        sw      $fp,16($sp) #古いフレームポインタを退避
        addiu   $fp,$sp,28  #新しくフレームポインタを設定
```

```
        sw      $a0,0($fp)  #引数を退避させる# 28($sp) にもってきているもの

        #引数＞０かどうかを調べる。
        #引数＜＝０なら１を返す。
        #引数＞０なら fact ルーチンを呼出 (n-1) を計算し、その結果に n をかける
        #上記を再帰的に繰り返す

        lw      $v0,0($fp)  #n を road させておく
        bgtz    $v0,factsub #引数が 0 より大きければ再帰処理に飛ぶ
        li      $v0,1       #0 以下なら 1
        j       return

factsub:
        lw      $v1,0($fp)  #n をロードする
        subu    $v0,$v1,1   #n-1
        move    $a0,$v0     #a0 に n-1 に戻る
        jal     fact

        lw      $v1,0($fp)
        mul     $v0,$v0,$v1 #n*fact(n-1)

return: #return 処理
        lw      $ra,20($sp) #戻りアドレスを復元
        lw      $fp,16($sp) #フレームポインタを復元
        addiu   $sp,$sp,32  #スタックポインタをポップする
        j       $ra
```

2.4.2 内容

2.5 課題 1-4

2.5.1 考察

3 感想