基礎演習 IP2 レポート

1411045 風間　健流　I1 クラス

2016 年 2 月 22 日 (月)

1　課題 1 - 1

1.1　ソースコード

リスト 1: 課題 1-1 プログラム

```
1  .data
2  newline:.asciiz "\n"
3    .text
4    .globl  main
5    .ent  main
6  main:
7    subu  $sp, 16
8    sw  $31, 12($sp)
9    li  $8, 10
10   li  $9, 15
11   add $4, $8, $9
12   li  $10, 5
13   sub $4, $4, $10
14   li  $v0, 1
15   syscall
16   la  $4, newline
17   li  $v0, 4
18   syscall
19   move  $2, $0
20   lw  $31, 12($sp)
21   addu  $sp, 16
22
23   jr  $ra
24   .end
```

1.2 プログラムの説明

まず、値 10 と 15 をレジスタ$8,$9 にコピーする。次に 8 と$9 に保持されている値の和とり、$4 にコピー。値 5 を$10 にコピーし、$4 と$10 の差をとる。表示方法は example1-1.asm と同じ。

1.3 実行結果

```
(spim) load "1-1.asm"
(spim) run
20
```

2 課題 1-2

2.1 ソースコード

リスト 2: 課題 1-2 プログラム

```
1  .data
2  newline:.asciiz "\n"
3    .text
4    .globl  main
5    .ent  main
6  main:
7    subu  $sp, 16
8    sw  $31, 12($sp)
9    li  $v0, 5
10   syscall
11   move  $8, $v0
12   li  $v0, 5
13   syscall
14   move  $9, $v0
15   add $4, $8, $9
16   li  $v0, 1
17   syscall
18   la  $4, newline
19   li  $v0, 4
20   syscall
21   move  $2, $0
22   lw  $31, 12($sp)
23   addu  $sp, 16
24
25   jr  $ra
26   .end
```

2.2 プログラムの説明

$v0 に 5 を代入してからシステムコールを行うことで整数値を読み込む。この処理を二回行いそれぞれをレジスタ$8,$9 にコピーする。その後、命令「add」で$4 にレジスタの値の和をとり出力する。

2.3 実行結果

```
(spim) load "1-2.asm"
(spim) run
234
54
288
```

3 課題１考察

システムコールは使うのが初めてで$v0 レジスタにコピーする値によって動作が変わるなど少しとっつきにくいところはあるが、うまく使えば MIPS でやれることの幅が広がることがわかった。

4 課題 2-1

4.1 ソースコード

リスト 3: 課題 2-1 プログラム

```
1     .data
2  hoge:   .word 1,4,1,4,2,1,3,5 # int hoge[8] = {1, 4, 1, 4, 2, 1, 3, 5}
3  bar:    .word 0:8
4  newline:  .asciiz "\n"
5     .text
6     .align  2
7     .globl  main
8     .ent  main
9  main:
10    subu  $sp, 16
11    sw  $31, 12($sp)
12
13    li  $16, 0    # i = 0;
```

```
14      la  $17, hoge # $17に、hoge[0]のアドレスをコピー
15      la  $18, bar
16      li  $19, 0    # sum = 0;
17  loop:
18      lw  $4, 0($17)  # hoge[i]の値をロード
19      add $19, $19, $4
20      sw  $19, 0($18)
21      lw  $4, 0($18)
22      li  $v0, 1
23      syscall     # hoge[i]の値を画面に表示
24      la  $4, newline
25      li  $v0, 4
26      syscall     # 改行を画面に表示
27      add $17, $17, 4 # $17は、hoge[i + 1]のアドレス
28      add $18, $18, 4
29      add $16, $16, 1 # i++
30      slt $10, $16, 8 # i < 8 ?
31      bgtz  $10, loop # i < 8ならばloopへジャンプ
32
33      move  $2, $0
34      lw  $31, 12($sp)
35      addu  $sp, 16
36      j $31
37    .end
```

4.2 プログラムの説明

コードの各行における、レジスタとメモリの状態は次の図のようになる。

4.3 実行結果

```
(spim) load "2-1.asm"
(spim) run
1
5
6
10
12
13
16
21
```

5 課題 2-2

5.1 ソースコード

リスト 4: 課題 2-2 プログラム

```
1     .data
2  newline: .asciiz "\n"
3     .text
4     .align  2
5     .globl  main
6     .ent  main
7  main:
8     subu  $sp, 16
9     sw  $31, 12($sp)
10
11    li  $v0, 5
12    syscall
13    move  $8, $v0
14    li  $9, 0
15 loop:
16    add $9, $9, 1
17    div $8, $9
18    mfhi  $4
19    bne $4, 0,l0
20    move  $4, $9
21    li  $v0, 1
22    syscall
23    la  $4, newline
24    li  $v0, 4
25    syscall
```

```
26 l0:
27     slt $10, $9, $8
28     bgtz  $10, loop
29
30     move  $2, $0
31     lw  $31, 12($sp)
32     addu  $sp, 16
33     j $31
34   .end
```

5.2 プログラムの説明

まずレジスタ$8 に入力した値をコピーする。次にレジスタ$9 の値を 1 から 1 ずつ増やしながらループもしも $8 ÷ $9 の余りが 0 だったら画面に出力する。$9 が$8 以上になったらループを抜ける。

5.3 実行結果

```
(spim) load "2-2.asm"
(spim) run
36
1
2
3
4
6
9
12
18
36
```

6 課題２考察

普段何気なく使っている for 文や配列が内部でどのように処理されているのかが分かった。課題２－２のプログラムでは調べる必要のない部分も判定しているので無駄な部分をなくせば効率がさらによくなると思う。