情報領域演習第二 C 演習（クラス 3）

学籍番号: 1810678 クラス: 3
名前: 山田朔也

2019 年 6 月 28 日

課題 1　ソースコード　まず、作成したソースコードを、Listing1 に示す。

Listing 1　課題 1 のソースコード

```
1            .data
2            .text
3        newline:.asciiz "\n"
4            .globl main
5            .ent main
6        main:
7            subu $sp, 16
8            sw $ra, 12($sp)
9            li $t0, 10
10           li $t1, 15
11           li $t2, 5
12           add $a0, $t0, $t1
13           sub $a0, $a0, $t2
14           li $v0, 1
15           syscall
16           la $a0, newline
17           li $v0, 4
18           syscall
19           move $v0, $zero
20           lw $ra, 12($sp)
21           addu $sp, 16
22           jr $ra
23           .end
```

実行例　また、この実行例は以下の Listing2 ようになる。

Listing 2　課題 1 の実行例

```
1        (spim) load "prob1.asm"
2        (spim) run
3        20
```

解説と考察　まずこのコードでは、t0,t1,t2 レジスタに計算に必要な各値を入れていく。その後 add 命令と sub 命令を用いて求める値を計算して a0 レジスタに値を入れる。そして syscall のサービス

の一つである、print_int を行い a0 レジスタ内の整数値を表示、最後に改行を画面に表示してプログラムは終了する。
また、このコードであるが、まだ使用するレジスタ数を減らす方法がある。計算するためにわざわざ t0 t2 レジスタに値を代入していたが、それ以前から a0 レジスタを使用することによって、a0,t0,t1 だけで実行したい計算を行うことができるだろう。また、即値演算命令を用いれば a0 レジスタのみで計算を行うことも可能である。

課題 2　ソースコード　まず、作成したソースコードを、Listing3 に示す。

Listing 3　課題 2 のソースコード

```
1              .data
2          newline:.asciiz "\n" 改行を画面に表示するおまじない#
3              .text
4              .globl main
5              .ent main
6          main:
7              subu $sp, 16
8              sw $ra, 12($sp)
9              li $t0, 9
10             li $t1, 2
11             div $t0, $t1
12
13             mflo $a0
14             li $v0, 1 整数を出力する命令#
15             syscall #の値を画面に表示$a0
16
17             la $a0, newline #にで定義された文字列をコピー$a0newline
18             li $v0, 4 文字列を出力する命令#
19             syscall 改行を画面に表示#
20
21             mfhi $a0
22             li $v0, 1
23             syscall
24
25             la $a0, newline
26             li $v0, 4
27             syscall
28
29             move $v0, $zero
30             lw $ra, 12($sp)
31             addu $sp, 16
32             jr $ra
33             .end
```

実行例　また、この実行例は以下の Listing4 ようになる。

Listing 4　課題 2 の実行例

```
1          (spim) load "prob2.asm"
2          (spim) run
3          4
4          1
```

解説と考察　まずこのコードでは、t0,t1 レジスタに計算に必要な各値を入れていく。その後 div 命令用いて求める値を計算して,LO レジスタと HI レジスタの中身を a0 レジスタに入れ、表示していく。

今回の課題は前課題と異なり、使用するレジスタをこれ以上減らすことはできないだろう。なぜなら div 命令は即値による演算ができず、値はレジスタを経由して指定するしかないからである。また、move 命令は汎用レジスタ名しか指定できないため、HI,LO からデータを移す時は mflo,mfhi を利用することに注意が必要だと思われる。

課題 3　ソースコード　まず、作成したソースコードを、Listing5 に示す。

Listing 5　課題 3 のソースコード

```
1              .data
2          newline:.asciiz "\n" 改行を画面に表示するおまじない#
3              .text
4              .globl main
5              .ent main
6          main:
7              subu $sp, 16
8              sw $ra, 12($sp)
9
10             li $v0, 5
11             syscall #read_int
12             move $t0, $v0
13
14             li $v0, 5
15             syscall #read_int
16             move $t1, $v0
17
18             add $a0, $t0, $t1
19
20             li $v0, 1 整数を出力する命令#
21             syscall #の値を画面に表示$a0
22
23             la $a0, newline #にで定義された文字列をコピー$a0newline
24             li $v0, 4 文字列を出力する命令#
25             syscall 改行を画面に表示#
26
27             move $v0, $zero
28             lw $ra, 12($sp)
```

```
29              addu $sp, 16
30              jr $ra
31              .end
```

実行例　また、この実行例は以下の Listing6 ようになる。

Listing 6　課題 3 の実行例

```
1           (spim) load "prob3.asm"
2           (spim) run
3           150
4           12
5           162
```

解説と考察　まずこのコードでは、syscall のサービスの一つの read_int を用いて 2 つの数値を t0,t1 レジスタにそれぞれ代入していく。その後 add 命令用いて求める値を計算して, その結果を a0 レジスタに入れ、表示していく。

今回の課題は前課題と異なり、逆に move 命令を利用するのが良いだろう。また、その他にも add 命令を利用して代入する方法もあるが、move が一番単純明快だと思われる。

課題 4　ソースコード　まず、作成したソースコードを、Listing7 に示す。

Listing 7　課題 4 のソースコード

```
1               .data
2           newline:.asciiz "\n" 改行を画面に表示するおまじない#
3           hoge:.word 1 4 1 4 2 1 3 5
4           bar:.word 0 0 0 0 0 0 0 0
5               .text
6               .align 2
7               .globl main
8               .ent main
9           main:
10              subu $sp, 16
11              sw $ra, 12($sp)
12
13              la $t0, hoge
14              la $t1, bar
15              li $t2, 0 #i
16              li $t3, 8 定数#
17              li $t4, 0 #sum
18
19          k1: bge $t2, $t3, k2
20
21              sll $t5, $t2, 2
22              add $t6, $t5, $t1
23              add $t5, $t5, $t0
24              lw $t7, 0($t5)
```

```
25
26              add $t4, $t4, $t7
27
28              sw $t4, 0($t6)
29
30              lw $t7, 0($t7)
31
32              move $a0, $t7
33
34              li $v0, 1 整数を出力する命令#
35              syscall #の値を画面に表示$a0
36
37              la $a0, newline #にで定義された文字列をコピー$a0newline
38              li $v0, 4 文字列を出力する命令#
39              syscall 改行を画面に表示#
40
41              add $t2, $t2, 1
42              b k1
43
44          k2: move $v0, $zero
45              lw $ra, 12($sp)
46              addu $sp, 16
47              jr $ra
48
49              .end
```

実行例　また、この実行例は以下の Listing8 ようになる。

Listing 8　課題 4 の実行例

```
1           (spim) load "prob4.asm"
2           (spim) run
3           1
4           5
5           6
6           10
7           12
8           13
9           16
10          21
```

解説と考察　まずこのコードでは、hoge と bar という名前の配列を作成し、変数 i,sum と定数用のレジスタを確保する。次にループの中身を実装で、まず最初に条件分岐命令を用いてループを実行するかどうかを判定する。その後、sll を用いて、配列が 0 番目から i 番目までのアドレスのズレを計算して t5 レジスタに保存しておく。また、これに乗じて bar の i 番目のアドレスを t6 レジスタに、hoge の i 番目のアドレスを t5 レジスタに保存する。そして sum の値を計算した後 bar の i 番目に書き込み、これをもう一度 t7 にロードし表示して正しく書き込めているかを確かめる。

この課題であるが、一番重要なのはきちんと書き込めているかを、再度ロードして確かめる部分である。ただ、読み込み書き込みのテストのための課題なため今回はこのコードが正しいが、本来であれば余計な処理であるため再読込みをせずにそのまま sum を書き出すのが処理速度を考える上で重要となるだろう。