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 7 月 13 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 newline:.asciiz "\n"
3         .text
4         .globl main
5         .ent main
6 main:
7         subu $sp, 16
8         sw $ra, 12($sp)
9 #ここまでおまじない--------------------------------------------------
10
11        li $v0, 5
12        syscall #read_int
13        move $t0, $v0
14
15        li $t1, 1
16        li $t2, 1
17
18 k1: beq $t0, $t1, k2
19
20        mult $t2, $t0
21        mflo $t2
22
23        sub $t0, $t0, 1
24
25        b k1
26
27 k2: move $a0, $t2
28        li $v0, 1 #整数を出力する命令
29        syscall #$a0の値を画面に表示
30
31        la $a0, newline #$a0にnewlineで定義された文字列をコピー
32        li $v0, 4 #文字列を出力する命令
```

```
33          syscall #改行を画面に表示
34
35          move $v0, $zero
36          lw $ra, 12($sp)
37          addu $sp, 16
38          jr $ra
39
40          .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  4
4  24
5  (spim) run
6  1
7  1
8  (spim) run
9  10
10 3628800
```

考察　まず、このコードを実装する際に工夫した点を記述していく。

工夫した点としてはどれだけ少ない行数で記述できるかという点である。ただ、その結果として残念ながら必要であろう機能を一部欠くことになってしまった。本当であれば実行例のように”Input a nuber: ”や”Output: ”などと表示したり、LO レジスタの中身だけでなく、値が大きくなったときのため HI レジスタの中身も表示できるようにするとよりよいプログラムとなることが推測される。

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"
3  hoge:.word 1 4 1 4 2 1 3 5
4          .text
5          .globl main
6          .ent main
7  main:
8          subu $sp, 16
9          sw $ra, 12($sp)
10 #ここまでおまじない----------------------------------------------------
11         li $s0, 0 #ループ用のi
12         li $s1, 8 #ループ回数の最大数
13         la $s2, hoge
```

```
14 k1: bge $s0, $s1, k2
15
16         sll $s3, $s0, 2 #hoge読み込み準備
17         add $s3, $s3, $s2
18         lw $a0, 0($s3)
19
20         jal print
21
22         add $s0, $s0, 1
23
24         b k1
25
26 k2: move $v0, $zero
27         lw $ra, 12($sp)
28         addu $sp, 16
29         jr $ra
30
31         .end
32 #ここからサブルーチン----------------------------------------------------
33 print: li $v0, 1 #整数を出力する命令
34         syscall #$a0の値を画面に表示
35
36         la $a0, newline #$a0にnewlineで定義された文字列をコピー
37         li $v0, 4 #文字列を出力する命令
38         syscall #改行を画面に表示
39
40         jr $ra
```

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 1
4 4
5 1
6 4
7 2
8 1
9 3
10 5
```

考察　まず、今回サブルーチンを使用するため、気をつけなければいけないこととして main 部にて $t0~$t7 の使用は控えなければいけない。そして、一番重要なのはサブルーチン周りのレジスタの扱いである。ただ、この課題では返り値は存在しないため、注意するべきは引数を格納する$a0 レジスタである。基本的にこのレジスタに引数を入れることによってサブルーチンに値を渡すため、

必ず呼び出し元ではサブルーチンを呼ぶ前に渡したい値を$a0 におさめておかなくてはならない。

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"
3 hoge:.word 1 4 1 4 2 1 3 5
4         .text
5         .globl main
6         .ent main
7 main:
8         subu $sp, 16
9         sw $ra, 12($sp)
10 #ここまでおまじない--------------------------------------------------
11        li $s0, 0 #ループ用のi
12        li $s1, 8 #ループ回数の最大数
13        la $s2, hoge
14        li $s3, 0 #sum
15 k1: bge $s0, $s1, k2
16
17        sll $s4, $s0, 2 #hoge読み込み準備
18        add $s4, $s4, $s2
19        lw $a0, 0($s4)
20
21        jal func
22        add $s3, $s3, $v0
23
24
25        add $s0, $s0, 1
26
27        b k1
28
29 k2: move $a0, $s3
30        jal print
31
32        move $v0, $zero
33        lw $ra, 12($sp)
34        addu $sp, 16
35        jr $ra
36
37        .end
38
39 #ここからサブルーチン-----------------------------------------------
40 func: move $v0, $a0
41
```

```
42          mult $v0, $v0
43          mflo $v0
44
45          add $v0, $v0, 1
46
47          j $ra
48
49  print: li $v0, 1 #整数を出力する命令
50          syscall #$a0の値を画面に表示
51
52          la $a0, newline #$a0にnewlineで定義された文字列をコピー
53          li $v0, 4 #文字列を出力する命令
54          syscall #改行を画面に表示
55
56          jr $ra
```

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  81
```

考察　この課題では課題 2 と異なり、サブルーチンの返り値について考えなくてはならない。サブルーチンの返り値はサブルーチン内でまず$v0 レジスタに返したい値を格納し、その後呼び出し元に帰った後$v0 レジスタから取り出して使用することとなる。無論$v0 レジスタも汎用レジスタ同様計算に用いることができるため、今回実装したコードでは使用するレジスタを無駄に増やさないよう$v0 を計算に用いている。

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"
3  hoge: .word 1 4 1 4 2 1 3 5
4          .text
5          .globl main
6          .ent main
7  main:
8          subu $sp, 16
9          sw $ra, 12($sp)
10 #ここまでおまじない--------------------------------------------------
11         li $s0, 0 #ループ用のi
12         li $s1, 8 #ループ回数の最大数
13         la $s2, hoge
14         li $s3, 0 #sum
```

```
15 k1: bge $s0, $s1, k2
16
17         sll $s4, $s0, 2 #hoge読み込み準備
18         add $s4, $s4, $s2
19         lw $a0, 0($s4)
20
21         jal func
22         add $s3, $s3, $v0
23
24
25         add $s0, $s0, 1
26
27         b k1
28
29 k2: move $a0, $s3
30         jal print
31
32         move $v0, $zero
33         lw $ra, 12($sp)
34         addu $sp, 16
35         jr $ra
36
37         .end
38
39 #ここからサブルーチン--------------------------------------------------
40 #--関数calc1-----------------------------------------------------
41 calc1: li $t0, 1 #factorial
42         move $t1, $a0 #i
43
44 k3: ble $t1, $zero, k4 #forにおけるi>0
45         mul $t0,$t0,$t1 #factorial *= i
46         sub $t1, 1 #i--
47         b k3
48
49 k4: move $v0, $t0 #戻り値を専用レジスタに格納
50         jr $ra #親に帰る
51
52 #--関数calc2-----------------------------------------------------
53 calc2: li $t0, 0 #sum
54         li $t1, 1 #i
55
56 k5: bgt $t1,$a0,k6 #forにおけるi<=a
57         add $t0,$t0,$t1 #sum += i
58         add $t1, 1 #i++
59         b k5
60
```

```
61 k6: move $v0, $t0 #戻り値を専用レジスタに格納
62         jr $ra #親に帰る
63
64 #--関数func-----------------------------------------------------
65 func: move $s7, $ra #戻り番地の保存
66
67         li $t0, 2 #条件分岐の準備
68         div $a0, $t0
69         mfhi $t0
70
71         beq $t0,$zero,toc1 #条件分岐
72
73         jal calc2 #calc2呼び出し
74         b rtmain
75
76 toc1: jal calc1 #calc1呼び出し
77
78 rtmain: move $ra, $s7 #戻り番地を保存先から復元する
79         jr $ra #親に帰る
80
81 #--関数print-----------------------------------------------------
82 print: li $v0, 1 #整数を出力する命令
83         syscall #$a0の値を画面に表示
84
85         la $a0, newline #$a0にnewlineで定義された文字列をコピー
86         li $v0, 4 #文字列を出力する命令
87         syscall #改行を画面に表示
88
89         jr $ra
```

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 74
```

考察　この課題は、課題 2,3 と異なりサブルーチン内でサブルーチンを呼び出している。その場合困るのは$ra レジスタの値が最後に呼び出されたサブルーチンの戻る先しか保存できていないという点にある。ではこれをどのように解決するかというと、今回の課題では一度目に呼び出されたサブルーチン func にてサブルーチン呼び出しだ行われても値が保存される$s0~$s7 レジスタのどれかに$a0 の内容を保存しておくというものである。これをしておく事でサブルーチンからサブルーチンを呼び出したとしても最後に呼び出されたサブルーチンから帰ってきた際に保存しておいた値をもとに戻すことにより呼び出し元に帰ることができる。

また、仮に呼び出し数が増えたとしても、スタックフレームを利用すれば多くのアドレスを保存し

ておくことも可能である。