

2020年度電子情報工学科実験III
ボードコンピュータ実験
第3部 QtSPIMを用いた実験
課題4-1, 4

課題4-1：実行前

- “Single Step”で動作を確認
- シミュレータを初期化して再ロードし、ラベルがloopの命令とhaltの命令にブレークポイントを設定

Data	Text
Text	
	<pre>[00400000] 00004020 add \$8, \$0, \$0 ; 9: add \$t0, \$0, \$0 [00400004] 3c091001 lui \$9, 4097 [A] ; 10: la \$t1, 4097(A) → [00400008] 8d2a0000 lw \$10, 0(\$9) ; 11: lw \$t2, 0(\$t1) [0040000c] 11400004 beq \$10, \$0, 16 [halt-0x0040000c] [00400010] 010a4020 add \$8, \$8, \$10 ; 13: add \$t0, \$t0, \$t2 [00400014] 21290004 addi \$9, \$9, 4 ; 14: addi \$t1, \$t1, 4 [00400018] 08100002 j 0x00400008 [loop] ; 15: j loop → [0040001c] 2002000a addi \$2, \$0, 10 ; 16: addi \$t3, \$0, 10 [00400020] 0000000c syscall ; 17: syscall</pre>

課題4-1：実行結果

- 観測表に出力を記入

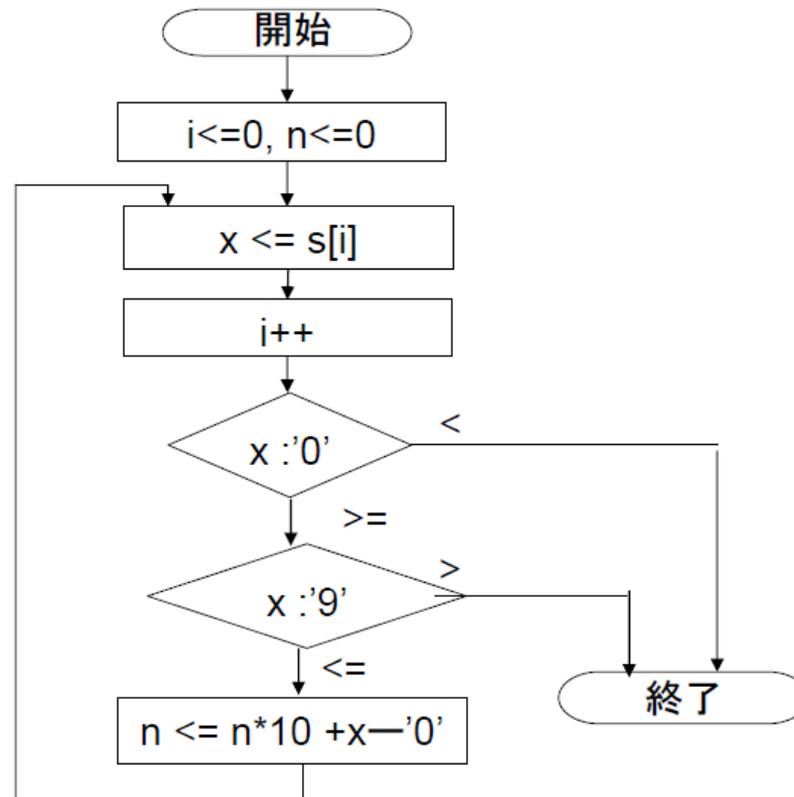
```
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
```

この値を観測

レジスタ	1回目	2回目	3回目	4回目	5回目	6回目
\$t0	0	1	3	7	a	a
\$t1	10010000	10010004	10010008	1001000c	10010010	10010010
\$t2	1	2	4	3	0	0

課題4-4：概要

- atoi関数
 - 入力されたASCIIコードをintegerに変換する
 - 数字でない文字は無視し終了する



課題4-4：ソースコード例(前半)

```
1      .data
2  A:   .space 80      # 文字列A
3  Text1: .asciiz "input is "
4  Text2: .asciiz "output is "
5
6      .text
7  .globl main
8  main: li $t0, 0      # n初期化
9
10     la $a0, Text1
11     li $v0, 4
12     syscall          # Text1の出力
13     la $a0, A
14     li $a1, 80
15     li $v0, 8
16     syscall          # 文字列Aの入力
17     li $t2, 48      # '0'
18     li $t3, 57      # '9'
19     li $t4, 10      # 10
```

入出力の際に表示させる
文字列(任意の文字列)

システムコール(表4参照)

""で囲ったものはASCII
コードの値、囲ってな
いものは数値(定数)

課題4-4 : ソースコード例(後半)

```
18 LOOP:  lb    $t1, 0($a0)
19      add  $a0, $a0, 1    # ポインタのインクリメント
20      blt  $t1, $t2, NEXT # if $t1 < $s0 then NEXT
21      bgt  $t1, $t3, NEXT # if $t1 > $s1 then NEXT
22      mul  $t0, $t0, $t4  # $t0 = $t0 * 10
23      add  $t0, $t0, $t1  # $t0 = $t0 + $t1
24      sub  $t0, $t0, $t2  # $t0 = $t0 - "0"
25      j    LOOP
```

```
26 NEXT: la    $a0, Text2
27      li    $v0, 4
28      syscall    # Text2
29      move  $a0, $t0
30      li    $v0, 1
31      syscall    # $t0 (integer) の出力
32      li    $a0, 0xa
33      li    $v0, 11
34      syscall    # Return の出力
35      li    $v0, 10
36      syscall    # exit
```

文字が"0"~"9"の場合実行し、次の文字を見る

課題4-4：実行前

- 初期値を確認

表示させようとしている
文字列を確認

The screenshot shows a debugger window with two panes. The left pane, titled 'Int Regs [16]', displays the initial values of various registers: PC, EPC, Cause, BadVAddr, Status, HI, LO, R0-R7. The right pane, titled 'Data', shows a memory dump of the 'User data segment' and 'User Stack'. The 'User data segment' contains the text 'input is . output'. A red box highlights this text, and a red arrow points from the text box above to it.

Register	Value
PC	= 0
EPC	= 0
Cause	= 0
BadVAddr	= 0
Status	= 3000ff10
HI	= 0
LO	= 0
R0 [r0]	= 0
R1 [at]	= 0
R2 [v0]	= 0
R3 [v1]	= 0
R4 [a0]	= 2
R5 [a1]	= 7ffff5d4
R6 [a2]	= 7ffff5e0
R7 [a3]	= 0

Address	Hex	Hex	Hex	Hex	Text
[10000000]..[1001004f]	00000000				
[10010050]	75706e69	73692074	756f0020	74757074	input is . output
[10010060]	20736920	00000000	00000000	00000000	is
[10010070]..[1003ffff]	00000000				

Address	Hex	Hex	Hex	Hex	Text
[7ffff5d0]	00000002	7ffff6b3	7ffff68f	00000000
[7ffff5e0]	7fffffe1	7fffffb2	7fffff81	7fffff45 E . . .
[7ffff5f0]	7fffff14	7ffffef7	7ffffed3	7ffffea1
[7ffff600]	7ffffe70	7ffffe48	7ffffe3b	7ffffe1d	p . . . H . . . /
[7ffff610]	7ffffdec	7ffffdba	7ffffd9c	7ffffd77 w . . .
[7ffff620]	7ffffd60	7ffffd37	7ffffd29	7ffffa01	` . . . 7 . . .)
[7ffff630]	7ffff9c3	7ffff9a6	7ffff95d	7ffff94b] . . . K . . .
[7ffff640]	7ffff933	7ffff918	7ffff8fa	7ffff8d1	3
[7ffff650]	7ffff8b3	7ffff848	7ffff831	7ffff81d H . . . 1
[7ffff660]	7ffff80e	7ffff7f8	7ffff7ce	7ffff7a5

課題4-4：文字の入力

- “Console”画面で文字を入力する
 - デバッグのため数字+文字, 文字+数字等、複数パターン試す

 Console

```
input is |
```

課題4-4：実行後①

- “12”を入力した場合

 Console

```
input is 12
output is 12
|
```

課題4-4：実行後②

- “a12”, “12a”を入力した場合

 Console

```
input is a12
output is 0
|
```

 Console

```
input is 12a
output is 12
|
```