

## Worksheet #7: Worksheet on Loops – Solutions

Below is the usual (“standard”) snapshot of MIPS registers, with a data declaration and the SPIM memory dump from the data section. Remember: All register and memory contents are shown in hexadecimal form. Also, if any register or memory location is changed in a problem, the change does not carry over to any other problem.

.data
n: .word
p: .word
q: .word
r: .word
s: .word
t: .word
u: .word
v: .word
w: .word
x: .word
y: .word
z: .word
str:
.asciiz
"hello
world\n"

### MIPS Registers

R0 (r0): 0x00000000	R8 (t0): 0x0f0f0f0f	R16 (s0): 0x00000000	R24 (t8): 0x00000000
R1 (at): 0x10010000	R9 (t1): 0x0000ffff	R17 (s1): 0x00000000	R25 (t9): 0x00000001
R2 (v0): 0x0000000b	R10 (t2): 0x00000000	R18 (s2): 0x00000058	R26 (k0): 0x00000000
R3 (v1): 0x00000000	R11 (t3): 0x10010020	R19 (s3): 0x00000000	R27 (k1): 0x00000000
R4 (a0): 0x00000058	R12 (t4): 0x100100f0	R20 (s4): 0x00400020	R28 (gp): 0x10008000
R5 (a1): 0x10010010	R13 (t5): 0x10010030	R21 (s5): 0x00000000	R29 (sp): 0x7ffffeff0
R6 (a2): 0x0000000c	R14 (t6): 0x80000080	R22 (s6): 0x800c1001	R30 (s8): 0x00000000
R7 (a3): 0x00000010	R15 (t7): 0xffff0000	R23 (s7): 0x00000050	R31 (ra): 0x00400070

### Data

[0x10000000]...[0x1000fffc] 0x00000000

[0x10010000]	0x5350494d	0x20697320	0x61207573	0x6566756c
[0x10010010]	0x9073696d	0x756c6174	0xcf722066	0x6f72206c
[0x10010020]	0x6561726e	0xf96e6720	0x4d495053	0xa0617373
[0x10010030]	0x68656c6c	0x6f20776f	0x726c640a	0x00000000
[0x10010040]	0x726f6772	0x616d6d69	0x6e672061	0x6e642063
[0x10010050]	0xf7f7f7f7	0x74657220	0x61726368	0x69746563

[0x10010060]...[0x10020000] 0x00000000

1. In the program, answer the following questions:
  - 1.1. What is the program doing? Copying the data located at memory locations 0x 10010000-0x 1001001c to 0x10010040-0x1001005c.
  - 1.2. Where is the source “data?” The eight words at addresses 0x10010000 to 0x1001001c.
  - 1.3. Where is the destination location of the data? Memory addresses 0x10010040 to 0x1001005c.
  - 1.4. How many cycles will the loop make? 8.

move \$t0,\$at
li \$t2,8
loop: lw \$t1,0(\$t0)
sw \$t1,64(\$t0)
sub \$t2,\$t2,1
addi \$t0,\$t0,4
bgtz \$t2,loop
li \$v0,10
syscall

2. In the program segment at the right:
  - 2.1. What is the program doing? Converting the lower case “hello world” to upper case.
  - 2.2. What is the purpose of incrementing the value in \$t8 by 1 each cycle? The pointer to the current character must be changed as the loop proceeds from letter to letter, changing the value to upper case.
  - 2.3. What do blt \$t0,0x61,incr and bgt \$t0,0x7a,incr do? They make sure that we do the capitalization function only on lower-case letters – doing it to any other character would result in changing the actual character.

la \$t8,str
go: lb \$t0,0(\$t8)
beqz \$t0,done
blt \$t0,0x61,incr
bgt \$t0,0x7a,incr
sub \$t0,\$t0,32
sb \$t0,0(\$t8)
incr: addi \$t8,\$t8,1
j go
done: li \$v0,10
syscall

3. In the space to the right:
  - 3.1. Compose a program to load the numbers w-z into a register and determine whether any are – (negative) . Count the number of negative data locations and store in \$t3. You do not need to do a data declaration (since it is on page 1); just do the text (program).  
See Program.

.text
main: li \$t0,4
move \$t3,\$0
la \$t1,w
loop: lw \$t2,0(\$t1)
bltz \$t2,count
j incr
count: addi \$t3,\$t3,1
incr: sub \$t0,\$t0,1
beqz \$t0,done
addi \$t1,\$t1,4
j loop
done: li \$v0,10
syscall

- 3.2. How many negative numbers are there? 2.