

Lab 10: Introduction to RISC-V Assembly SOLUTIONS

Digital Design and Computer Architecture: RISC-V Edition (Harris & Harris, Elsevier © 2021)

1. Please indicate how many hours you spent on this lab. This will be helpful for calibrating the workload for next time the course is taught.
2. Please submit each of your code snippets: div9.s, big2little.s, and bubblesort.s.

Answers:

div9.s

```
# sarah.harris@unlv.edu
# October 2021
# Description: a0 holds a positive integer. If it is divisible by 9,
# a0 is 1 after the code completes. Otherwise, a0 is 0.
#

test:    addi a0, zero, 25      # test case

check:   beq  a0, zero, isdiv9
        blt  a0, zero, notdiv9
        addi a0, a0, -9        # a0 = a0 - 9
        j    check
notdiv9: addi a0, zero, 0       # a0 = 0 (not divisible by 9)
        j    done
isdiv9:  addi a0, zero, 1       # a0 = 1 (is divisible by 9)
done:
```

big2little.s

```
initarray:
    addi a0, zero, 0x300
    addi t0, zero, 32        # number of bytes
    addi t1, zero, 0         # index
    lui  t2, 0xABCDE         # t2 = 0xABCDE123 (random value)
    addi t2, t2, 0x123

L1:
    beq  t0, t1, doneinit
    add  t3, a0, t1           # t3 = address to write to
    sw   t2, 0(t3)            # [t3] = t2
    addi t1, t1, 4            # t1 += 4 (address of next word)
    slli t2, t2, 8            # change value in t2
    add  t2, t2, t1
    j    L1                  # repeat loop
doneinit:

    addi a0, zero, 0x300
    addi t0, zero, 0x320     # just past last byte

L2:
    beq  a0, t0, done        # branch when at end of 8 words
    lb   t2, 0(a0)           # t2 = byte 0
    lb   t3, 1(a0)           # t3 = byte 1
    lb   t4, 2(a0)           # t4 = byte 2
```

```

lb    t5, 3(a0)      # t5 = byte 3
sb    t2, 3(a0)      # byte 3 = t2
sb    t3, 2(a0)      # byte 2 = t3
sb    t4, 1(a0)      # byte 1 = t4
sb    t5, 0(a0)      # byte 0 = t5
addi  a0, a0, 4      # increment by 1 word (4 bytes)
j     L2             # repeat loop
done:

```

bubblesort.c

```

int tmp, swapped = 1; // s0 = tmp, s1 = swapped

while (swapped) {
    swapped = 0;
    for (i=0; i<9; i++)
        if (sortarray[i] > sortarray[i+1]) {
            tmp = sortarray[i];
            sortarray[i] = sortarray[i+1];
            sortarray[i+1] = tmp;
            swapped = 1;
        }
}

```

bubblesort.s

```

initarray:
    addi a0, zero, 0x400
    addi t0, zero, 89
    sw    t0, 0(a0)
    addi t0, zero, 63
    sw    t0, 4(a0)
    addi t0, zero, -55
    sw    t0, 8(a0)
    addi t0, zero, -107
    sw    t0, 0xC(a0)
    addi t0, zero, 42
    sw    t0, 0x10(a0)
    addi t0, zero, 98
    sw    t0, 0x14(a0)
    addi t0, zero, -425
    sw    t0, 0x18(a0)
    addi t0, zero, 203
    sw    t0, 0x1C(a0)
    addi t0, zero, 0
    sw    t0, 0x20(a0)
    addi t0, zero, 303
    sw    t0, 0x24(a0)

# s0 = tmp, s1 = swapped
bubblesort:
    addi a0, zero, 0x400 # base address of array
    addi t0, a0, 36      # just past second to last element of array
    addi s1, zero, 1     # swapped = 1
L1:

```

```

    beq  s1, zero, done    # if (swapped == 0), done
    addi s1, zero, 0      # swapped = 0
    addi t3, a0, 0        # t3 = base address of sortarray[]
L2:
    bge  t3, t0, L1        # if i >= 9, end of for loop
    lw   s0, 0(t3)         # tmp = sortarray[i]
    lw   t1, 4(t3)         # t1 = sortarray[i+1]
    bge  t1, s0, L3        # if sa[i+1] >= sa[i], continue
    sw   t1, 0(t3)         # sortarray[i] = sortarray[i+1]
    sw   s0, 4(t3)         # sortarray[i+1] = tmp
    addi s1, zero, 1       # swapped = 1
L3:
    addi t3, t3, 4         # t3 = address of next element in array
    j    L2
done:
    addi zero, zero, 0     # nop - to be able to set breakpoint here

```

Please indicate any bugs you found in this lab manual, or any suggestions you would have to improve the lab.