Department of Electrical and Computer Engineering
University of Wisconsin – Madison
ECE 552 Introduction to Computer Architecture

In-Class Exercise (09/09)

Write all answers within the boxes provided for each question. Any response outside the designated boxes will be disregarded. Answers may be either handwritten or typed. If handwritten, ensure that your writing is clear and legible. Please upload the pdf with your answers to gradescope through Canvas (under Canvas assignment tab).

1. *Translating C statement to RV32I assembly.*
   In the following sub-problems, assume variables f, g, h, i, and j are stored in registers $s0, $s1, $s2, $s3, $s4 respectively. Assume the base address of int arrays A and B are in registers $s6 and $s7 respectively. For the C statement in each sub-problem, translate it into corresponding RV32I code using the minimum number of assembly instructions. Use only $s*, $t*, and $zero registers. **Assume memory is byte addressable.**

   (a) f = g + (-5 - f)

```
addi t0, x0, -5
sub t1, t0, $s0
add $s0, $s1, t1
```

   (b) B[8] = A[j-i] // assume j ≥ i

```
sub t0, $s4, $s3
slli t0, t0, 2
add t0, t0, $s6
lw t1, 0(t0)
sw t1, 32($s7)
```

2. *Translating RISCV assembly code to C statements.*
   In the following sub-problems, assume variables f, g, h, i, and j are stored in registers $s0, $s1, $s2, $s3, and $s4 respectively and their values are multiples of 4. Assume that &A (the base address of int array A) and &B (the base address of int array B) are in registers $s6, and $s7 respectively. For the RV32I assembly code in each sub-problem, write down the corresponding C statements. **Assume memory is byte addressable.**

   (a)

```
add $t0, $s6, $s0
add $t1, $s7, $s1
lw $s0, 0($t0)
lw $t0, 4($t0)
add $t0, $t0, $s0
sw $t0, 0($t1)
```

```
temp = &A + f;
temp2 = &B + g;
f = *temp;
temp = *(temp + 4);
temp = temp + f;
*temp2 = temp;
```

(b)

```
addi $t0, $s6, 4
add $t1, $s6, $zero
sw $t1, 0($t0)
lw $t0, 0($t0)
add $s0, $t1, $t0
```

```
temp = &A + 4;
temp2 = &A;
*temp = temp2;
temp = *temp;
f = temp2 + temp;
```

A[1] = &A;
f = 2 * &A

3. Consider the following RISCV assembly program:

```
Loop:   slli $t1, $s3, 2
        add  $t1, $t1, $s6
        lw   $t0, 0($t1)
        bne  $t0, $s5, Exit
        addi $s3, $s3, 1
        jal  $zero, Loop
Exit:
```

The above assembly is compiled from the following C statement (A is an **integer** array):

```
while (A[i] == k) i++;
```

Which registers correspond to:

a. Integer variable i?
b. Integer variable k?
c. Base address of integer array A (&A[0])?

a. Register $ s3

b. Register $ s5

c. Register $ s6

4. The following RISCV assembly code has just executed on your machine:

```
andi $s1, $s1, 20
add  $s1, $s0, $s1
lw   $s3, 0($s1)
```

20 = 0b10100   s1 can be 20, 4, 16, or 0

The current contents of registers and memory immediately **after** executing this code are:

Register	Contents
$s0	4000
$s1	???
$s2	3
$s3	-5

Memory Address	Contents
4000	-5
4004	1
4008	3
4012	7
4016	-5
4020	-5
4024	-1
4028	7
4032	-5
4036	3
4040	3
4044	-1

Given the current state of registers and memory, list all the possible values of [$s1] after executing the code.

4000, 4016, or 4020