TAMU 312 Name:
Quiz 20 July 2

1. Write the spim code for a function that performs multiply (recommend that you write java pseudocode first)

C++ code:

```
typedef unsigned short uint16_t;

// range for a and b: 0 to 255, though they have 16 bits.
uint16_t multiply(uint16_t a, uint16_t b) {
  uint16_t result = 0;
  while (b > 0) {
    if (b & 1 > 0) {
      result += a;
    }
    b = b >> 1;
    a = a << 1;
  }
  return result;
}
```

SPIM code:

```
// input : $a0, $a1
// output: $v0
multiply:
  li $v0, 1            // result = 0
loop:
  blez $a1, loop_end   // if b <= 0 go to loop_end
  andi $t0, $a1, 1     // $t0 = b & 1
  blez $t0, shift      // if $t0 <= 0 go to shift
  add $v0, $v0, $a0    // result += a
shift:
  srl $a1, $a1, 1      // b = b >> 1
  sll $a0, $a0, 1      // a = a << 1
  j loop               // go to loop
loop_end:
  jr $ra               // return result
```

2. Write the spim code for a function that performs divide(recommend that you write java pseudocode first)

C++ code:

```
typedef unsigned short uint16_t;

// range for a and b: 0 to 255, though they have 16 bits.
// assume b is not 0.
void divide(uint16_t a, uint16_t b, uint16_t* quotient, uint16_t* remainder) {
  *remainder = a;
  *quotient = 0;
  uint16_t divisor = b << 8;

  for (int i = 0; i < 8; ++i) {
    divisor = divisor >> 1;
    *quotient = *quotient << 1;
    if (*remainder >= divisor) {
      *remainder = *remainder - divisor;
      *quotient = *quotient + 1;
    }
  }
}
```

SPIM code:

```
// input: $a0, $a1
// output: $v0 (quotient), $v1 (remainder)
divide:
  add $v1, $a0, $0    // $v1 (remainder) = a
  li $v0, 0           // $v0 (quotient) = 0
  sll $t0, $a1, 8     // $t0 (divisor) = b << 8
  li $t1, 8           // $t1 (i) = 8
loop:
  blez $t1, loop_end  // if i <= 0 go to loop_end
  srl $t0, $t0, 1     // divisor = divisor >> 1
  sll $v0, $v0, 1     // quotient = quotient << 1
  blt $v1, $t0, skip  // if remainder < divisor go to skip
  sub $v1, $v1, $t0   // remainder = remainder - divisor
  addi $v0, $v0, 1    // quotient = quotient + 1
skip:
  subi $t1, $t1, 1    // i = i - 1
  j loop              // go to loop
loop_end:
  jr $ra              // return
```

3. Write the assembly code for toLowerCase. (Ask the TA for numbers of ascii codes if you need them.)

C++ code:

```
typedef unsigned short uint16_t;

void toLowerCase(char* address, uint16_t length) {
  for (int i = 0; i < length; ++i) {
    char c = *(address + i);
    if (c >= 65 && c <= 90) {
      *(address + i) = c + 32;
    }
  }
}
```

SPIM code:

```
// input: $a0 (address), $a1 (length)
to_lower_case:
  li t4, 65                 // $t4 = 65 ('A')
  li t5, 90                 // $t5 = 90 ('Z')
  li $t0, 0                 // $t0 (i) = 0
loop:
  bge $t0, $a1, loop_end    // if i >= length go to loop_end
  add $t1, $a0, $t0         // $t1 = address + i
  lb $t2, 0($t1)            // $t2 (c) = mem($t1) = *(address + i)
  blt $t2, $t4, next_iter   // if c < 'A' go to next_iter
  blt $t5, $t2, next_iter   // if c > 'Z' go to next_iter
  add $t2, $t2, 32          // c += 32
  sb $t2, 0($t1)            // *(address + i) = c
next_iter:
  addi $t0, $t0, 1          // i += 1
  j loop                    // go to loop
loop_end:
  jr $ra                    // return
```