IT5002 Computer Organization
Tutorial 2

Tutorial Questions:

1. Below is a C code that performs palindrome checking. A palindrome is a sequence of characters that reads the same backward or forward. For example, “madam” and “rotator” are palindromes.

```
char string[size] = { ... }; // some string
int low, high, matched;

// Translate to MIPS from this point onwards
low = 0;
high = size-1;
matched = 1;          // assume this is a palindrome
                      // In C, 1 means true and 0 means false
while ((low < high) && matched) {
   if (string[low] != string[high])
      matched = 0;    // found a mismatch
   else {
      low++;
      high--;
   }
}
// "matched" = 1 (palindrome) or 0 (not palindrome)
```

Given the following variable mappings:

low ➔ \$s0;
high➔ \$s1;
matched ➔ \$s3;
base address of string[] ➔ \$s4;
size ➔ \$s5

a. Translate the C code into MIPS code by keeping track of the indices.

b. Translate the C code into MIPS code by using the idea of “array pointer”. Basically, we keep track of the actual addresses of the elements to be accessed, rather than the indices. Refer to lecture set #8, slide 34 for an example.

Note: Recall the “short circuit” logical AND operation in C. Given condition (A && B), condition B will not be checked if A is found to be false.

2. **MIPS Bitwise Operations**

Implement the following in MIPS assembly. Assume that integer variables **a**, **b** and **c** are mapped to registers $s0, $s1 and $s2 respectively. Each part is independent of all the other parts. **For bitwise instructions (e.g. ori, andi, etc),** any immediate values you use should be written in binary for this question. This is optional for non-bitwise instructions (e.g. addi, etc).

Note that bit 31 is the most significant bit (MSB) on the left, and bit 0 is the least significant bit (LSB) on the right, i..e.:

MSB					LSB
Bit 31	Bit 30	Bit 29	…	Bit 1	Bit 0

a. Set bits 2, 8, 9, 14 and 16 of **b** to 1. Leave all other bits unchanged.

b. Copy over bits 1, 3 and 7 of **b** into **a**, without changing any other bits of **a**.

c. Make bits 2, 4 and 8 of **c** the inverse of bits 1, 3 and 7 of **b** (i.e. if bit 1 of **b** is 0, then bit 2 of **c** should be 1; if bit 1 of **b** is 1, then bit 2 of **c** should be 0), without changing any other bits of **c**

3. **MIPS Arithmetic**

Write the following in MIPS Assembly, using as few instructions as possible. You may rewrite the equations if necessary to minimize instructions.

In all parts you can assume that integer variables **a**, **b**, **c** and **d** are mapped to registers $s0, $s1, $s2 and $s3 respectively. Each part is independent of the others.

a. c = a + b

b. d = a + b – c

c. c = 2b + (a – 2)

d. d = 6a + 3(b – 2c)

4. [AY2013/14 Semester 2 Exam]

The mysterious MIPS code below assumes that **$s0 is a 31-bit binary sequence**, i.e. the MSB (most significant bit) of **$s0** is assumed to be zero at the start of the code.

```
     add  $t0, $s0, $zero   # make a copy of $s0 in $t0
     lui  $t1, 0x8000
lp:  beq  $t0, $zero, e
     andi $t2, $t0, 1
     beq  $t2, $zero, s
     xor  $s0, $s0, $t1
s:   srl  $t0, $t0, 1
     j    lp
e:
```

a) For each of the following initial values in register **$s0** at the beginning of the code, give the hexadecimal value of the content in register $**s0** at the end of the code.

i. Decimal value **31**.
ii. Hexadecimal value **0x0AAAAAAA**.

b) Explain the purpose of the code in one sentence.