## CS2100 (AY2019/2020 Semester 2)

# /15

Assignment #1

Name: Yap Dian Hao

# Tutorial Grp: 3

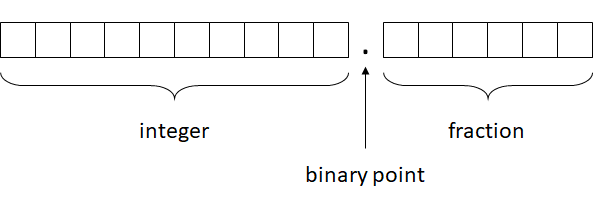
You are to do this assignment **on your own**. (Students found copying will be penalised.) Please fill in your **name** and **tutorial group number** (T01 – T20) in the boxes above, and your answers in the space indicated below. You are not required to show workings.

You should know your tutorial group number, and you need to fill in your tutorial group number in your midterm test paper. If you do not fill in your group number above, or fill in the wrong tutorial group number, one mark will be deducted.

Please submit this assignment by **15 February 2020, Saturday, 1pm** to the submission File on LumiNUS according to your tutorial group**.** Please submit either a .docx or .pdf file. **Late submission and email submission will not be accepted**. We are very strict on this, so please submit early if you think you may forget.

**Introduction**

At the start of the module, you learn how to code in C. Then we talked about number system to know more about how C actually handles certain numeric values. We then moved to MIPS and how to compile from C to MIPS, as well as how to encode MIPS instructions as binary/hexadecimal values.

In this assignment, we will go from C to hexadecimal to enhance your understanding of number systems and how to perform calculations on them. At the end of this assignment, we hope that you have a complete understanding of number system. Our aim is to implement a 16-bit fixed point number represented as 1s-complement with 10-bit integer and 6-bit fraction. The bit arrangement are shown below. We will subscript this number with 10\_6\_1s to differentiate it from the usual 1s-complement number system.

1. [*Warmup*] How is **-36.0312510** represented in IEEE 754 single-precision floating-point format? Write your answer in ***hexadecimal***. [1 mark]

***Answer:*** **0x**C2102000

1. [*Warmup*] How is **-36.0312510** represented in 10\_6\_1s representation above? Write your answer in ***binary*** with exactly 10-bit integer and 6-bit fraction. Also include the binary point for ease of marking. [1 mark]

***Answer:*** 1111011011.111101

1. Consider an implementation of the 10\_6\_1s number system in C using an array of integers of size 16 (e.g., int number[16]). Write a function to add two 10\_6\_1s numbers. The function prototype is given as:

int add(int num1[], int num2[], int res[]);

The function takes in two numbers (num1 and num2) as arrays of 16 integers, performs addition of the two numbers and stores the result in res (another array of 16 integers). The function also returns an integer. It returns 1 if there is an overflow, otherwise it returns 0.

***Answer:*** [3 marks]

|  |
| --- |
| int add (int num1[], int num2[], int res[]) {  bool carry = false;  for (int i = 6; i > -1; i--) {  int sum;  if (carry) {  sum = 1 + num1[i] + num2[i];  } else {  sum = num1[i] + num2[i];  }  if (sum > 1) {  if (sum == 2) {  res[i] = 0;  } else if (sum == 3) {  res[i] = 1;  }  carry = true;  } else {  res[i] = sum;  carry = false;  }  }  int i = 15;  while (carry) {  int sum = num1[i] + num2[i] + i; // carry out of MSB  if (sum > 1) {  if (sum == 2) {  res[i] = 0;  } else if (sum == 3) {  res[i] = 1;  }  } else {  carry = false;  res[i] = sum;  }  i--;  }  if (arr1[0] == arr2[0] && arr1[0] != res[0]) {  return 1;  } else {  return 0;  }  } |

You are given the implementation of void negate(int num[], int res[]); below that negates a 10\_6\_1s number num and stores the result in res. Both variables are integer arrays of size 16.

void negate(int num[], int res[]) {

  int i;

  for (i=0; i<16; i++) {

    if (num[i] == 0) {

      res[i] = 1;

    } else {

      res[i] = 0;

    }

  }

}

1. Write an equivalent MIPS code to perform the negation. You may assume the following mapping of variables to registers.

|  |  |
| --- | --- |
| **$s0: base address of num** | **$s1: base address of res** |

*Answer:* [3 marks]

|  |
| --- |
| addi $t0, $zero, 0  addi $t1, $zero, 16  loop: beq $t0, $t1, exit  lw $t2, 0($t0)  addi $s0, $s0, 4  addi $t0, $t0, 1  beq $t2, $zero, inv  sw $zero, 0($s1)  addi $s1, $s1, 4  j loop  inv: addi $t3, $zero, 1  sw $t3, 0($s1)  addi $s1, $s1, 4  j loop  exit: |

You are then given the following MIPS code with $s2 initially storing the base address of an array of 16 integers representing a 10\_6\_1s number. Assume that the 10\_6\_1s number represented as this array is greater than 0. Lastly, you are also given the data at some memory locations.

|  |  |
| --- | --- |
| **Memory** | |
| **Address** | **Value** |
| *11060038* | 1 |
| *1106003C* | 0 |
| *11060040* | 0 |
| *11060044* | 0 |
| *11060048* | 0 |
| *1106004C* | 0 |
| *11060050* | 1 |
| *11060054* | 0 |
| *11060058* | 0 |
| *1106005C* | 1 |
| *11060060* | 0 |
| *11060064* | 0 |
| *11060068* | 0 |
| *1106006C* | 0 |
| *11060070* | 0 |
| *11060074* | 0 |
| *11060078* | 1 |
| *1106007C* | 0 |
| *11060080* | 1 |
| *11060084* | 0 |

    addi $t1  , $s2  ,   4

    addi $t2  , $s2  ,  64

    addi $t3  , $zero, 136

LA: lw   $t4  , 0($t1)

    addi $t1  , $t1  ,  4

    addi $t3  , $t3  , -1

    beq  $t4  , $zero, LA

    add  $t5  , $zero, $zero

LB: beq  $t1  , $t2  , EX

    lw   $t4  , 0($t1)

    sll  $t5  , $t5  , 1

    add  $t5  , $t5  , $t4

    addi $t1  , $t1  , 4

E:  j    LB

EX: addi $t6  , $t3  , -136

    sll  $t3  , $t3  ,   23

    sll  $t5  , $t5  ,    8

SL: beq  $t6  , $zero, AN

    sll  $t5  , $t5  , 1

    addi $t6  , $t6  , 1

    j    SL

AN: or   $s3  , $zero, $t3

    or   $s3  , $s3  , $t5

1. Assume that the first instruction addi $t1, $s2, 4 is at address 0x21004208 and the initial value of $s2 is 0x11060040.
2. What is the value of $s3, in ***hexadecimal***, at the end of the execution?

***Answer:*** **0x42102000**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [2 marks]

1. What is the purpose of the code?

***Answer:*** Invert the sign of 10\_6\_1s number and express it in IEEE 754 single precision format. [2 marks]

1. What is the encoding, in ***hexadecimal***, for the j SL instruction?

***Answer:*** **0x08401093**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [2 marks]

1. How many **R-format** instructions are there in the MIPS program above?

***Answer:*** \_8\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [1 mark]