COMP 2421 Computer Organization
Programming Assignment One

> Do not copy codes from any resources. We can detect that.

1 OBJECTIVES

The objective of this assignment is to practice some simple programming tasks by using the MIPS assembly language using QtSpim. This program allows a user to input an integer. Then, the program prints out the corresponding number in different number systems including: i) Binary, ii) Quaternary, and iii) Octal. After completing the assignment, student should be able to understand the followings:

- how a program reads data and prints them out through the console window in different formats; (Sample code will be provided to show the operations. You need to understand how system calls work properly.)
- how the content of a register at the bit level can be read and how to use bitwise operations to perform logical transformations;
- how labels (as main memory addresses) and constants are declared and used in a program;
- how loop structures are used.

2 ASSIGNMENT DESCRIPTION

Write a program in MIPS assembly language that realizes the following functionalities:

(1) Prompt a message in the “Console” to ask for an integer. (It is assumed that the input type is correct, i.e., the input must be an integer.)
(2) Capture the input and store the number in a register.
(3) Use your own algorithm to convert the value stored in the register. (Note that you may store the output as a string or in an array.)
(4) Print out the results (i.e., the binary number, quaternary number and octal number) on the console.
(5) Ask if the user wants to "Continue" by inputting either “1” or “0” (1=Yes and 0=No).
(6) If “Yes”, repeat steps “(1)” to “(5)” again.
(7) If “No”, terminate the program properly.

The following is a sample of input and output of the program:

```
Enter a number: (assume the user inputs 23)
Input number is 23
Binary: 00000000000000000000000000010111
Quaternary: 0000000000000113
Octal: 00000000027
Continue? (1=Yes/0=No) (assume that the user inputs 1)
Enter a number: (assume the user inputs 24)
Input number is 24
Binary: 00000000000000000000000000011000
Quaternary: 0000000000000120
Octal: 00000000030
Continue? (1=Yes/0=No) (assume that the user inputs 0)
Bye!
```

3 WHAT TO SUBMIT

(1) **Your program code**. Store the code in a file and name it as "`p1_XXXXXXXX.s`", where "`XXXXXXXX`" represents your 8-dight student number (the last letter is not needed).
(2) **At the beginning of the program code file**, you should write a "readme" section to explain briefly how the program runs and what your design is. In addition, any other important/special information which user should know before running the program should also be included in the paragraph.
(3) Along with the program code, you should properly write **comments** for your code so that other programmers can easily follow your code.
(4) Upload the file to Blackboard. No other form of submission will be accepted. DO NOT zip the file.

4 ASSESSMENT

Your program will be tested on QtSpim. Note that QtSpim will be configured as ignoring the branch delay slots (so that you can ignore this point in your program).

The program will be assessed from these two aspects:

Correctness (Total 70 points)

- program can be successfully loaded (10 pts)
- input handling (10 pts)
- number conversion (30 pts)
- result output (10 pts)
- repetition (10 pts)

Documentation (Total 30 points)

- description of the program and comments (20 pts)
- code formatting and style (you are suggested to follow the style of the codes in the lab materials) (10 pts)

The following programs serve as the hints for you to complete this assignment. You are strongly encouraged to understand the functionalities of these programs before starting your own program. Please note that these programs are just used for illustration, you may not need to use them *exactly* in your own program. In addition, the materials in the labs are also good sources for your consideration.

(1) This example illustrates how to capture a string from the console and then to store it in a memory location (here in the example, it is "buffer").

```
      .text
      .globl main
main:
      la $a0, ask
      li $v0, 4
      syscall

      la $a0, buffer
      la $a1, buffer

      li $v0, 8
      syscall

      la $a0, reply
      li $v0, 4
      syscall

      la $a0, buffer
      li $v0, 4
      syscall

      li $v0, 10
      syscall

      .data
ask:        .asciiz "\n Enter string: "
reply:      .asciiz "\n You wrote: "
buffer:     .space 10
```

(2) This example illustrates how to transform an input value to an ASCII character. You may need to google ASCII code and know how different characters are represented as decimal numbers.

```
    .data
prompt: .asciiz "Enter a number (32 to 126): "
ans1:   .asciiz "\n Number: "
ans2:   .asciiz "\n ASCII : "
letter: .space 1

    .text
    .globl main
main:
    la $a0, prompt
    li $v0, 4
    syscall

    li $v0, 5
    syscall
    move $t0, $v0

    la $a0, ans1
    li $v0, 4
    syscall

    move $a0, $t0
    li $v0, 1
    syscall

    la $a0, ans2
    li $v0, 4
    syscall

    la $t1, letter
    sb $t0, 0($t1)

    la $a0, letter
    li $v0, 4
    syscall
```

(3) This example illustrates how to print out the value of the last two bits of a register. The program checks the last two bits of an input value (integer) which is stored in one of the registers.

```
    .data
prompt: .asciiz "\n Enter a number: "
ans:    .asciiz "\n Last bit is: "

    .text
    .globl main
main:
    la $a0, prompt
    li $v0, 4
    syscall

    li $v0, 5
    syscall

    move $t0, $v0

    la $a0, ans
    li $v0, 4
    syscall

    li $t1, 3

    and $t2, $t0, $t1

    move $a0, $t2
    li $v0, 1
    syscall
```