CS 245 Fall 2015
Assignment 3 – Using Procedures to Parse an I- or R-formatted Instruction

For all assignments, be sure to:

- INCLUDE REGISTER CONVENTIONS, CALLING SEQUENCE, STACK USAGE.
- Include file comments at the top, listing your name, the name of the program, and briefly describe what the program does.
- Describe the register convention, and calling sequence for procedures.
- Line up instruction parts in columns: 0=labels; 1tab=opcode (instruction mnemonic); 2tabs=operands; 3-4tabs=comments.
- Include pseudo-code comments to the right of your assembly code. **Avoid** comments that tell me what the assembly does, such as: move 25 to register $s3 – I already know this. Keep your comments to the logic.

Program 3: ***Using Procedures to Parse an I- or R-Formatted Instruction***

In the previous assignment, you took **an R-formatted instruction stored in memory** (i.e., in the data section), and printed its hexadecimal or decimal value for each field in the instruction.

In this assignment you will differentiate between an I-format and R-format instruction, because R-format instructions have opcodes=0. You will then print an instruction either as an I-format or an R-format instruction, selecting one format from the two formats below:

```
Example:
      Instruction: 0x0388000F                    Instruction: 0x24AB0020
      Opcode: 0x0A                               Opcode: 0xA0
      RS: 5                                      RS: 10
      RT: 6                                      RT: 12
      RD: 7                                      Immediate: 32
      Shift amount: 0
      Function Code: 0x0F

(These are not true examples – don't assume this data is correct.)
```

Note that the registers, shift amounts and immediates are printed in decimal, while opcode, function code and instruction are printed in hexadecimal.

To do:
Create two procedures: Print_I_Format and Print_R_Format. In main, determine whether the instruction is an I-format or R-format, then call (or jal to) the appropriate procedure. From each of these procedures, you will call separate print procedures as discussed next.

The previous program was excessively long. So create two procedures to help with the printing. Here are the two procedures, with their MANDATORY calling sequence:

```
##########################################
#  Print_Decimal()
#  This procedure will print a label and a number: e.g., "RS: 8"
```

```
#  Calling Sequence:
#               la      $a0,Label       #  PrintDecimal(Label,Value)
#               lw      $a1,Value
#               jal     PrDec
#  The routine then prints:  Label: value
#   Stack Usage:  (You specify)
##########################################

##########################################
#  Print_Hexadecimal()
#  This procedure will print a label and a base 16 number: e.g., "RS: 0x0024"
#  Calling Sequence:
#               la      $a0,label       # PrintHex(Label,Value)
#               sw      $a0,-4($sp)
#               lw      $a0,value
#               sw      $a0,-8($sp)
#               jal     PrHex
#  The routine then prints:  Label: 0xValue
#  Stack Usage:  (You specify)
#
##########################################
```

You will be graded on proper use of procedures, stack and documentation. Be sure to include storage of the return address on the stack, as well as any $s registers you use. Be sure to document calling sequence, registers conventions, and stack usage for each procedure.

Submission

Turn this program in as homework3.asm **via paper and electronic copy**:

```
$ submit 245 homework3.asm
```

Grading

Each homework assignment is worth 10 points. Be careful to include all comments and format correctly, as directed in the beginning of this file.