Class	CS47, Sec 01
Homework	I
Due Date	October 12, 2015 11:59 PM PST

Instructions

1. There are 7 questions with total 10 points.
2. Please create electronic document with your answer.
3. There is no need to include the question itself. However, you **MUST** include question number and sub-part index if any. Example: 9(b)
4. Please create a PDF document **hw1.pdf** and ***upload that in Canvas*** assignment page by the due date.
5. **NO** handwritten document is accepted.
6. **NO LATE SUBMISSION.**

1. A 16-bit system (i.e. address and data are both 16-bit) uses 8 registers (0-7). It supports 3 types of instructions. Type I has machine code format [opcode (4bit) | rs (3bit) | rt (3bit) | rd (3bit) | funct (3bit)] and assembly code format '<mnemonic> <rd>, <rs>, <rt>'. Type II has machine code format [opcode (4bit) | rs (3bit) | rt (3bit) | immediate (6bit)] and assembly code format '<mnemonic> <rt>, <rs>, <immediate>'. Type III has machine code format [opcode (4bit) | address (12-bit)]. Using this information compute the machine code in 16-bit hex format for the following instructions assuming anything after '#' is comment. [**3 pts**]

   a) add r2, r5, r6 # opcode: 0x2 / funct: 0x2 **[0x2B92]**
   b) sub r1, r7, r7 # opcode: 0x2 / funct: 0x3 **[0x2FCB]**
   c) addi r1, r2, 0x3A #opcode: 0x4 **[0x447A]**
   d) ori r2, r3, 0x1B #opcode: 0x3 **[0x369B]**
   e) jmp 0x23C # opcode: 0x5 **[0x423C]**
   f) jal 0x00F # opcode: 0x6 **[0x600F]**

2. Assume 'addi' in question 1 expands the 6-bit immediate number to a 16-bit number using signed extension. Similarly 'ori' in question 1 expands the 6-bit immediate number to a 16-bit number using zero extension. What are the expanded 16-bit hex representation for the immediate number in the following. [**1 pts**]

   a) addi r1, r2, 0x3A **[0xFFFA]**
   b) ori r2, r3, 0x1B **[0x001B]**

3. For the given two macro definition expand the macro call and write down the expanded text for 'my_para (Frank, 28Sep2085, 30, pizza)' [**1 pts**]

```
.macro my_msg($name,$date)
I am $name born on $date,
therefore I am a computer guru
and my id is $name_$date
.end_macro
```

```
.macro my_para ($name, $date, $age, $food)
my_msg($name,$date)
I am $age year old and love to eat $food
.end_macro
```

Ans:

```
I am Frank born on 28Sep2085,
therefore I am a computer guru
and my id is Frank_28Sep2085
I am 30 year old and love to eat pizza
```

4. Write a macro in MIPS assembly to print value of Hi and Lo registers in 32-bit integer format. This macro takes 4 arguments of string names of strHi, strComma, strLo, strEqual and prints <strHi> <strEqual> <value of Hi> <strComma> <strLo> <strSequal> <value of Lo>. Use this macro in program which will ask a number of Hi reg, and then for Lo reg. It prints the content of Hi and Lo using the macro you have defined. Then the program swaps the value of Hi and Lo and then again print its content. Include your complete code (without any .include) in this answer, so that one can copy paste your code in MARS and run it. It should generate output similar to the following. [**2 pts**]

```
Enter number for Hi ? 45
Enter number for Lo ? 20
Before swapping Hi = 45 , Lo = 20
After swapping Hi = 20 , Lo = 45
```

[cut paste the program in MARS and run to watch the result – a possible answer is as following]

```
# Macro : print_str
# Usage: print_str(<address of the string>)
.macro print_str($arg)
 li      $v0, 4     # System call code for print_str
 la      $a0, $arg  # Address of the string to print
 syscall          # Print the string
 .end_macro

 # Macro: read_int
 # Usage: read_int(<reg>)
 .macro read_int($arg)
 li      $v0,5   # Read intger
 syscall
 move    $arg, $v0 # move the data to target reg
 .end_macro

# Macro: print_reg_int
 # Usage: print_reg_int(<reg>)
 .macro print_reg_int ($arg)
 li      $v0, 1         # print_int call
 move    $a0, $arg      # move the source reg value to $a0
 syscall
 .end_macro

 # Macro : exit
  # Usage: exit
  .macro exit
 li      $v0, 10
 syscall
 .end_macro
```

```
.data
msg1:   .asciiz "Enter number for "
strHi:  .asciiz "Hi"
strLo:      .asciiz "Lo"
strQuery:      .asciiz " ? "
strEqual:      .asciiz " = "
strComma:      .asciiz " , "
strNewline:    .asciiz "\n"
msg2:          .asciiz "Before swaping "
msg3:          .asciiz "After swaping "

# prints : 'Hi = <value> , Lo = <value> '
# Expects 'strHi' (Hi), 'strLo'(Lo),
#         'strComma'( , ), 'strEqual' ( = ) is defined
.macro print_hi_lo ($strHi, $strEqual, $strComma, $strLo)
       print_str($strHi)
       print_str($strEqual)
       mfhi $t0
       print_reg_int($t0)
       print_str($strComma)
       print_str($strLo)
       print_str($strEqual)
       mflo $t0
       print_reg_int($t0)
.end_macro

.text
.globl main
main:
       # Get and store Hi value
       print_str(msg1)        # Prints: Enter a number for
       print_str(strHi)       # Prints: Hi
       print_str(strQuery)    # Prints: ?
       read_int($t0)          # Read integer into $t0
       mthi $t0               # Move $t0 value to Hi

       # Get and store Lo value
       print_str(msg1)        # Prints: Enter a number for
       print_str(strLo)       # Prints: Hi
       print_str(strQuery)    # Prints: ?
       read_int($t0)          # Read integer into $t0
       mtlo $t0               # Move $t0 value to Lo

       #print content of Hi and Lo
       print_str(msg2)        # Prints: Before swaping
       print_hi_lo(strHi,strEqual, strComma, strLo)
                              # Prints: Hi = <val> , Lo = <val>
       print_str(strNewline)  # Prints: newline

       # Swap the content
       mfhi $t0               # $t0 = Hi
       mflo $t1               # $t1 = Lo
       mthi $t1               # Hi  = $t1
       mtlo $t0               # Lo  = $t0

       #print content of Hi and Lo
       print_str(msg3)        # Prints: After swaping
       print_hi_lo (strHi, strEqual, strComma, strLo)
                              # Prints: Hi = <val> , Lo = <val>
       print_str(strNewline)  # Prints: newline

       exit
```

5. Write macro 'push' and 'pop' which takes register name as argument. The 'push' operation implements push the given register value onto MIPS stack and 'pop' operation implements pop value from the MIPS stack to the given register. [**1 pts**]

```
       # Macro: push
       # Usage: push (<reg>)
       .macro push($reg)
```

```
sw      $reg, 0x0($sp) # M[$sp] = R[reg]
addi    $sp, $sp, -4   # R[sp] = R[sp] - 4
.end_macro

# Macro: push
# Usage: push (<reg>)
.macro pop($reg)
addi    $sp, $sp, +4   # R[sp] = R[sp] + 4
lw      $reg, 0x0($sp) # M[$sp] = R[reg]
.end_macro
```

6. In a byte addressable system byte sequences are following from address 0x10010000 0x23, 0x1a, 0x25, 0xaf, 0xef, 0xa5, 0x5a, 0x61, 0x6f, 0x73. If the system uses 48-bit register and supports a load load command 'ld48bit <rt>, <address>' to load 48-bit information from memory. What would be the content of register t0 after 'ld48bit $t0, 0x10010002' in following scenarios? [**1pts**]
   a) System uses little endian convention. **[0x615aa5efaf25]**
   b) System uses big endian convention. **[0x25afefa55a61]**

7. A number system muNote uses symbol Do, Re, Mi, Fa, So, La, Ti with equivalent decimal weight 0, 1, 2, 3, 4, 5, 6 respectively. In that case, answer the following. [**1pts**]
   a) What is the decimal equivalent of MiReReDo?
   b) What is muNote equivalent of decimal number 1045673?

Ans:

a) This is 7 base number system. Therefore, assuming equivalent decimal weight of Mi, Re and Do, MiReReDo is ($2 * 7^3 + 1 * 7^2 + 1 * 7^1 + 0 * 7^0$) = 742.

b) 1045673 / 7 → Q = 149381 , R = 6 (Ti)
149381 / 7 → Q = 21340 , R = 1 (Re)
21340 / 7 → Q = 3048 , R = 4 (So)
3048 / 7 → Q = 435 , R = 3 (Fa)
435 / 7 → Q = 62 , R = 1 (Re)
62 / 7 → Q = 8, R = 6 (Ti)
8 / 7 → Q = 1, R = 1 (Re)
1 / 7 → Q = 0, R = 1 (Re)

Hence the muNote equivalent is 'ReReTiReFaSoReTi'