COMP1047 Lab Week 03

1. Work out the unsigned binary representation for the following decimal numbers by hand.
   (a) 45 (b) 1026
   remainder
   45 / 2 1
   22 / 2 0
   11 / 2 1
   5 /2 1
   2 /2 0
   1 /2 1
   0

   The result is 101101

   remainder
   1026 / 2 0
   513 / 2 1
   256 / 2 0
   128 / 2 0
   64 / 2 0
   32 / 2 0
   16 / 2 0
   8/2 0
   4/2 0
   2/2 0
   1/2 1
   0
   The result is 10000000010

2. Write a MIPS program to load the numbers above into registers $s0 and $s1 as unsigned numbers. You can place the binary value directly in the data memory segment and then use `lw` instruction to load them into registers. For example, the following program stores two unsigned integers $0000000A_{16}$ and 10000000_{16} in the data segment of the memory and then loads the first integer in $s0 using `lw` instruction.

```
      .data
uint: .word 0x0000000A 0x10000000
      .text
      .globl main
main:
      la $t0, uint            #load the base address
      lw $s0, 0($t0)    #load the first integer into $s0
```

   Note that here we use assembler directive **.word**. You can find more assembler directives from pages A-47 to A-49 of the textbook. Now print out both numbers to the console using the **syscall** function. Check the output to see whether it is expected.

Solution

```
# In QtSpim, change the register to binary,
# run the program step by step, and see the value change
# in R16 [s0], R17 [s1] and R4[a0].

      .data
uint: .word 45 1026
      #.word 0x0000002D 0x00000402 # 101101 10000000010
nl:   .asciiz "\n"
      .text
      .globl main
main:
      la $t0, uint   #load the base address
      lw $s0, ($t0)  #load the first integer into $s0
      lw $s1, 4($t0) #load the second integer into $s1

      move $a0, $s0  # move $s0 to $a0 for printing
      li $v0, 1
      syscall

      la $a0, nl # print a new line
      li $v0, 4
      syscall

      move $a0, $s1 # move $s1 to $a0 for printing
      li $v0, 1
      syscall

      li $v0, 10 # exit
      syscall
```

3. Work out the 2’s complement representation for the following decimal numbers by hand.
(a) 45 (b) -130

Write a similar program in the previous question, load both numbers into registers and print them out to the QtSpim console, check whether your outputs are correct.

Solution

```
# In QtSpim, change the register to binary,
# run the program step by step, and see the value change
# in R16 [s0], R17 [s1] and R4[a0].

            .data
uint:       .word 45 -130
            #.word 0x0000002D 0xFFFFFFFFFFFFFF7E
            #.word 00…00101101 00…0011111111101111110 (16 bits each)
nl:         .asciiz "\n"
            .text
            .globl main
main:
            la $t0, uint       #load the base address
            lw $s0, ($t0)      #load the first integer into $s0
            lw $s1, 4($t0)     #load the second integer into $s1

            move $a0, $s0      # move $s0 to $a0 for printing
            li $v0, 1
            syscall

            la $a0, nl         # print a new line
            li $v0, 4
            syscall

            move $a0, $s1      # move $s1 to $a0 for printing
            li $v0, 1
            syscall

            li $v0, 10 # exit
            syscall
```

4. Write a program in MIPS32 assembly language which reads two integer numbers x and y from the console, calculates, then prints x – 2y – 40. *Hint: no multiplication is necessary and proper user prompts are expected.*

   To read an integer from the console:

```
li  $v0, 5   # read_int
syscall
# $v0 contains the number just entered
```

   To print an integer to the console:

```
# $a0 contains the number to be printed
li  $v0, 1   # print_int
syscall
```

Solution

```
            .data
prompt1:    .asciiz "Please input x: "
prompt2:    .asciiz "Please input x: "
rs_string:  .asciiz "The result of (x - 2y - 40) is: "
            .text
            .globl main
main:
            # prompt for input
            la $a0, prompt1 # prompt x
            li $v0, 4
            syscall

            li $v0, 5       # read input x
            syscall

            or $s0, $zero, $v0  # Save x to s0

            la $a0, prompt2    # prompt y
            li $v0, 4
            syscall

            li $v0, 5       # read input y
            syscall

            or $s1, $zero, $v0     # Save y to s1

            la $a0, rs_string     # The result is
            li $v0, 4
            syscall

            # calculation
            sll $s1, $s1, 1      # 2y
            sub $s0, $s0, $s1   # x - 2y
            addi $a0, $s0, -40   # a0 = x -2y – 40
```

```
li $v0, 1        # output result
syscall

# exit
li $v0, 10
syscall
```