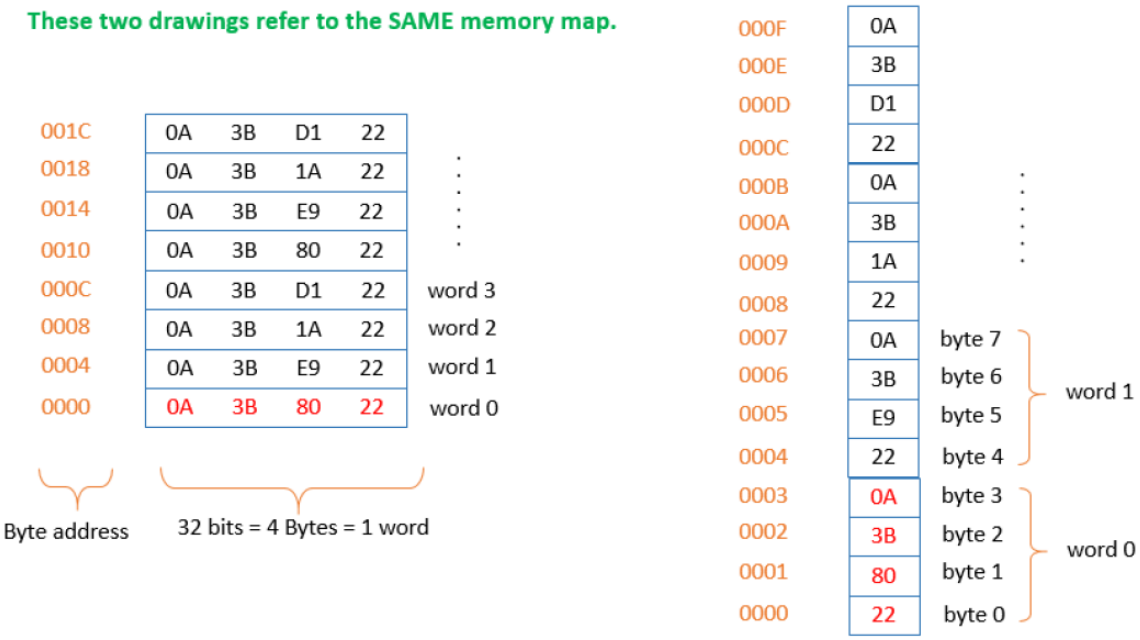
COMP1047 Lab Week 03

1. This part aims to provide with you deeper insights into the memory organization and representation. Observe the figure below for the same piece of memory, which is represented in words and in bytes, respectively. Provide answers to the questions below.



   a. What is the byte address of word number 42? Can you represent the address in Hex format?
   Solution Word 0 starts at 0 (the initial byte), Word 1 starts at 4, etc. Keeping counting, Word 42 starts at 42 × 4 = 168, which is A8 in Hex.

   b. What are the byte addresses that word 42 spans?
   Solution 0xA8, 0xA9, 0xAA, 0xAB.

   c. Write the MIPS assembly code to load word 3 into $t0. Hint: (1) Use lw; (2) Use $t1 to contain the address of word 3 directly.
   Solution
   ```
   addi $t1, $0, 12 (or $t1, $0, 0xC)
   lw   $t0, 0($t1)
   ```

   d. Write the MIPS assembly code to add the values in word 0 and word 1, and store the result back to word 42.
   Solution
   ```
   lw   $t6, 0($0)
   addi  $t1, $0, 4
   lw   $t7, 0($t1)
   add  $t8, $t6, $t7
   addi $t1, $0, 0xA8
   sw   $t8, 0($t1)
   ```

2. Write a program in MIPS assembly which reads three integer numbers x, y and z from the console, then calculates and prints out *m*, the minimum of the three. The following C segment shows how *m* can be calculated:

```
m = x;
if (m > y) m = y;
if (m > z) m = z;
```

Solution

```
.data
prompt1:.asciiz "Please input x: "
prompt2:.asciiz "Please input y: "
prompt3:.asciiz "Please input z: "
rs_string:.asciiz "The minimum number 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, prompt3       # prompt z
  li $v0, 4
  syscall

  li $v0, 5             # read input z
  syscall

  or $s2, $zero, $v0    # Save z to s2
  # Print rs_string first
  la $a0, rs_string
  li $v0, 4
  syscall

  # calculation
  move $a0, $s0         # m = x
  slt $t0, $s1, $a0     # t0 = 1 if s1 < s0
```

```
    beq $t0, $zero, compare2 # jump to compare2 if s1 >= s0

    move $a0, $s1          #if y<m, m=y

# compare smaller one in (s0, s1) with s2
compare2:
    slt $t0, $s2, $a0      #s2 < a0
    beq $t0, $zero, print_res # jump to print_res if s2 >= s0
    or $a0, $s2, $0        # if z<m, m = z

print_res:
    li $v0, 1              #output result
    syscall
    li $v0, 10             # exit
    syscall
```

3. Write a program in MIPS assembly language to read two integer numbers A and B. The program should indicate if one of these numbers is multiple of the other one. Hint: You may need to use the "div" and "mfhi" instructions, whose definition can be found from the MIPS reference card. Note for "div", the "Lo" and "Hi" are two special registers that are used to store the results of division and multiplication operations.

Solution

```
.data
prompt1: .asciiz "Please input A: "
prompt2: .asciiz "Please input B: "
AofB_string: .asciiz "A is the multiple of B.\n"
BofA_string: .asciiz "B is the multiple of A.\n"
no_string: .asciiz "They are not the multiple of each other.\n"
.text
.globl main

main:
## prompt for input
la $a0, prompt1        # prompt A
li $v0, 4
syscall
li $v0, 5              # read input A
syscall
or $s0, $zero, $v0     # save A to s0

la $a0, prompt2        # prompt B
li $v0, 4
syscall
li $v0, 5              # read input B
syscall
or $s1, $zero, $v0     # save B to s1
```

```
## calculation
div $s0, $s1              # Lo = $s0 / $s1, Hi = $s0 mod $s1
mfhi $t0 # move quantity in special register Hi to $t0: $t0 = Hi, i.e. the remainder
beq $t0, $zero, AofB      # if the remainder is 0, jump to branch AofB

div $s1, $s0              # Lo = $s1 / $s0, Hi = $s1 mod $s0
mfhi $t0 # move quantity in special register Hi to $t0: $t0 = Hi, i.e. the remainder
beq $t0, $zero, BofA      # if the remainder is 0, jump to branch BofA

no:                       # otherwise, move to branch no
la $a0, no_string         # "They are not the multiple of each other."
li $v0, 4
syscall
j exit # exit the program

AofB:
la $a0, AofB_string       # "A is the multiple of B."
li $v0, 4
syscall
j exit

BofA:
la $a0, BofA_string       # "B is the multiple of A."
li $v0, 4
syscall

exit:                     # exit the program
li $v0, 10
syscall
```

4. Given two integer arrays A and B, in which each integer is represented in 32-bit two's complement format. Assume that A and B are defined as follows.

```
        .data
A:      .word 4  6  12  -8  5
B:      .word 3  2   1   4  0
```

Update B[0]= 2*A[3]+B[4] and then print out all elements in B.

Solution

```
   .data
rs_string:
   .asciiz "B[i]= \n"
nline:
   .asciiz "\n"
A:
   .word 4 6 12 -8 5
B:
   .word 3 2 1 4 0
```

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

    la $s0, A
    la $s1, B
    li $s2, 5                # array length of B
    lw $t0, 12($s0)          # $t0 = A[3]
    lw $t1, 16($s1)          # $t1 = B[4]
    sll $t0, $t0, 1          # 2*A[3]
    add $t0, $t0, $t1        # B[0] = $t0

    sw $t0, ($s1)

loop:
    lw $a0, ($s1)
    li $v0, 1
    syscall                  # print integer

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

    addi $s2, $s2, -1
    addi $s1, $s1, 4         #next integer
    bne $s2, $zero, loop     #continue until all elements are printed

    li $v0, 10
    syscall
```