**Laboratory 3 – ITITIU22240 – Đàm Nguyễn Trọng Lễ**

**Exercise 1: Write a program that use only**

and andi nor or ori sll srl xor xori

* 1. Put the number 0xABC22240 into register $t1 **without** using pseudo-instruction **li**. (**lab4\_1\_1.s**)

**Code:**

*.data*

*.text*

    .globl  main

main:

    andi    $t1,    $t1,    0x0     #clear the value of t1

    ori     $t1,    $t1,    0xABC2  #load the upper 16 bits to t1

                                    #t1=0x0000ABC2

    sll     $t1,    $t1,    16      #shift the value of t1 to the left by 16 bits

                                    #t1=0xABC20000

    ori     $t1,    $t1,    0x2240  # or bit the lower 16 bits of t1, 0x2240

                                    #t1=0xABC22240

    jr      $ra

**Explain:**

**1st: We use andi $t0 to make sure that we return $t1 value to 0  
2nd. Load the first 16 bits of 0xABC22240 (0xABC2) to $t1 by LOGIC OR  
3rd. Then shift 16 bits left to OR with 16 bits left (0x2240)**

**QTSPIM”**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.

* 1. Redo 1.1 as follows: use **ori** to load **each letter** into register. (**lab4\_1\_2.s**)

**Code:**

*.data*

*.text*

    .globl  main

main:

    and     $t1,    $t1,    $zero   # Clear $t1

    ori     $t1,    $t1,    0xA

    sll     $t1,    $t1,    4

    ori     $t1,    $t1,    0xB

    sll     $t1,    $t1,    4

    ori     $t1,    $t1,    0xC

    sll     $t1,    $t1,    4

    ori     $t1,    $t1,    0x2

    sll     $t1,    $t1,    4

    ori     $t1,    $t1,    0x2

    sll     $t1,    $t1,    4

    ori     $t1,    $t1,    0x2

    sll     $t1,    $t1,    4

    ori     $t1,    $t1,    0x4

    sll     $t1,    $t1,    4

    ori     $t1,    $t1,    0x0

    #printing the value of $t1

    li      $v0,    1

    move    $a0,    $t1

    syscall

    # Exit the program

    li      $v0,    10

    syscall

**Explain:**

**With same method to problem 1.1 but rather than each step ORI we use full (short) 16 bits we just use 4 bits = 1 digits of hexadecimal, thus after each ORI we shift left 4 bits**

**QTSPIM:**

**After add the 0xA  
Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Nội dung do AI tạo ra có thể không chính xác.**

**Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Nội dung do AI tạo ra có thể không chính xác.**

**Ảnh có chứa ảnh chụp màn hình, văn bản

Nội dung do AI tạo ra có thể không chính xác.**

* 1. Suppose that $t1 = 0xABC22240. Using only register-to-register logic and shift instructions, Reverse the order of the bytes in $t1 so that register $t2 get the bit pattern 0x04222CBA (**lab4\_1\_3.s**)

**Code:**

*.data*

*.text*

    .globl  main

main:

    and     $t1,    $t1,    $zero       #clear the value of t1

    and     $t2,    $t2,    $zero       #clear the value of t2

    and     $t3,    $t3,    $zero       #clear the value of t3

    ori     $t1,    $t1,    0xABC2

    sll     $t1,    $t1,    16

    ori     $t1,    $t1,    0x2240

    andi    $t3,    $t1,    0xF         # mask the lower 4 bits of t1 => t3(temporary register) for storing the last 4 bits of t1

    or      $t2,    $t2,    $t3         # or bit the last 4 bits of t1 to t2

    sll     $t2,    $t2,    4           # shift t2 left by 4 bits

    srl     $t1,    $t1,    4           # shift t1 right by 4 bits

    andi    $t3,    $t1,    0xF

    or      $t2,    $t2,    $t3

    sll     $t2,    $t2,    4

    srl     $t1,    $t1,    4

    andi    $t3,    $t1,    0xF

    or      $t2,    $t2,    $t3

    sll     $t2,    $t2,    4

    srl     $t1,    $t1,    4

    andi    $t3,    $t1,    0xF

    or      $t2,    $t2,    $t3

    sll     $t2,    $t2,    4

    srl     $t1,    $t1,    4

    andi    $t3,    $t1,    0xF

    or      $t2,    $t2,    $t3

    sll     $t2,    $t2,    4

    srl     $t1,    $t1,    4

    andi    $t3,    $t1,    0xF

    or      $t2,    $t2,    $t3

    sll     $t2,    $t2,    4

    srl     $t1,    $t1,    4

    andi    $t3,    $t1,    0xF

    or      $t2,    $t2,    $t3

    sll     $t2,    $t2,    4

    srl     $t1,    $t1,    4

    andi    $t3,    $t1,    0xF

    or      $t2,    $t2,    $t3         # or bit the last 4 bits of t1 to t2

    srl     $t1,    $t1,    4           # clear the last 4 bits of t1

    srl     $t3,    $t3,    4           # clear t3

    jr      $ra

**Explain:**

**After we set $t1 = 0xABC22240 and set $t2, $t3 = 0, we:**

1. **Mask the last 4 bits by ANDI $t1 with 0xF and store the result into temp register($t3)**
2. **Then we use OR to store the result into last 4 bits of $t2**
3. **Then we shift left $t2 4 bits to hold the place for the incoming value**
4. **After that we shift right $t1 4 bits to prepare the newest last digit(hex) for the loop**
5. **Repeat until we get all the digits in $t1 (hexadecimal)**

**QTSPIM:**

**Set $t1 = 0xABC22240**

**Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Nội dung do AI tạo ra có thể không chính xác.**

**After the 1st iteration:**

**Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Nội dung do AI tạo ra có thể không chính xác.**

**Result: R10 [t2] = 4222cba (the last digits is 0 => the result only 7 rather than 8 digits in total)**

**Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Nội dung do AI tạo ra có thể không chính xác.**

* 1. Redo 1.3 using only and, or, and rotate instructions. **(lab4\_1\_4.s)**

**Code:**

*.data*

*.text*

    .globl  main

main:

    lui     $t1,    0xABC2

    ori     $t1,    $t1,    0x2240

    ori     $t4,    $zero,  0xF

    and     $t2,    $t2,    $zero       #clear the value of t2

    and     $t3,    $t3,    $zero       #clear the value of t3

    and     $t3,    $t1,    $t4

    or      $t2,    $t2,    $t3

    sll     $t2,    $t2,    4

    srl     $t1,    $t1,    4

    and     $t3,    $t1,    $t4

    or      $t2,    $t2,    $t3

    sll     $t2,    $t2,    4

    srl     $t1,    $t1,    4

    and     $t3,    $t1,    $t4

    or      $t2,    $t2,    $t3

    sll     $t2,    $t2,    4

    srl     $t1,    $t1,    4

    and     $t3,    $t1,    $t4

    or      $t2,    $t2,    $t3

    sll     $t2,    $t2,    4

    srl     $t1,    $t1,    4

    and     $t3,    $t1,    $t4

    or      $t2,    $t2,    $t3

    sll     $t2,    $t2,    4

    srl     $t1,    $t1,    4

    and     $t3,    $t1,    $t4

    or      $t2,    $t2,    $t3

    sll     $t2,    $t2,    4

    srl     $t1,    $t1,    4

    and     $t3,    $t1,    $t4

    or      $t2,    $t2,    $t3

    sll     $t2,    $t2,    4

    srl     $t1,    $t1,    4

    and     $t3,    $t1,    $t4

    or      $t2,    $t2,    $t3         # or bit the last 4 bits of t1 to t2

    srl     $t1,    $t1,    4           # clear the last 4 bits of t1

    srl     $t3,    $t3,    4           # clear t3

    jr      $ra

**Explain:**

**We use the same method with 1.3, except the load to $t1, $t4 we use ori, all the line left is free of addi and ori.**

**QTSPIM:**

**Ảnh có chứa văn bản, ảnh chụp màn hình, số

Nội dung do AI tạo ra có thể không chính xác.**

**Exercise 2: Write a program that**

* 1. Set the corresponding bit in register $t1 through $t8. That is, in register $t1 set bit 1, register $t2 set bit 2, and so on. (**lab4\_2\_1.s**)

**Code:**

*.data*

*.text*

    .globl  main

main:

    ori     $t1,    $t1,    0x1     # Load the value 0x1 into register $t1

    sll     $t2,    $t1,    1       # Shift the value in $t1 left by 1 bit and store it in $t2

    sll     $t3,    $t2,    1

    sll     $t4,    $t3,    1

    sll     $t5,    $t4,    1

    sll     $t6,    $t5,    1

    sll     $t7,    $t6,    1

    sll     $t8,    $t7,    1

    # Exit the program

    li      $v0,    10

    syscall

**Explain:**

**The idea behind this is simple, we store the value 0x1 into $t1 by using ori. Then we using shift left 1 bits to act as activation for bits in 2nd position in $t2. After that we treat the $t2 as cascading step to activate the bits in 3rd position in $t3. Looping until we set the bits in 8th position in $t8 on.**

**QTSPIM:**

**Ảnh có chứa văn bản, số, hàng, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.**

* 1. By using **ONLY** shift instructions and register to register logic instructions (no **li** pseudoinstruction or **addi**), put the pattern 0xFFFFFFFF into register $t1. (**lab4\_2\_2.s**)

**Code:**

*.data*

*.text*

    .globl  main

main:

    ori     $t1,    $t1,    0x1     # Load the value 0x1 into register $t1

    sll     $t2,    $t1,    1       # Shift the value in $t1 left by 1 bit and store it in $t2

    or      $t3,    $t2,    $t1     # t3 = 0x0011 = 0x3

    sll     $t4,    $t3,    2       # t4 = 0x1100 = 0xC

    or      $t5,    $t4,    $t3     # t5 = 0x1111 = 0xF

    sll     $t6,    $t5,    4       # t6 = 0x11110000 = 0xF0

    or      $t7,    $t6,    $t5     # t7 = 0x11111111 = 0xFF

    sll     $t8,    $t7,    8       # t8 = 0x1111111100000000 = 0xFF00

    or      $t9,    $t8,    $t7     # t9 = 0x1111111111111111 = 0xFFFF

    #clear the value of t1, t2, t3, t4, t5, t6, t7, t8,

    and     $t1,    $t1,    $zero

    and     $t2,    $t2,    $zero

    and     $t3,    $t3,    $zero

    and     $t4,    $t4,    $zero

    and     $t5,    $t5,    $zero

    and     $t6,    $t6,    $zero

    and     $t7,    $t7,    $zero

    and     $t8,    $t8,    $zero

    # put 0xFFFFFFFF in t1

    or      $t1,    $t1,    $t9

    sll     $t1,    $t1,    16

    or      $t1,    $t1,    $t9

    # Exit the program

    li      $v0,    10

    syscall

**Explain:**

**First, we set the $t1 = 0b1 then we use sll 1 bit to get $t2 = 0b10**

**We or $t1 and $t2 to get 0b0011 and store that in $t3, then sll $t3 2 bit and store that in $t4 to get 0b1100 , then we or $t3 and $t4 to get 0b1111 and store that in $t5**

**Then we repeat the same method to $t6 , $t7 , $t8 , $t9 to get 0xFFFF**

**Then we clear $t1 -> $t8 and apply the method we use in 1.1 to store the 0xFFFF in upper half 16 bits and lower half 16 bits of $t1**

**QTSPIM:**

We got the $t9 = 0xFFFF

Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Nội dung do AI tạo ra có thể không chính xác.