

Logical Operations

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Up Until Now

- Up until now, we've done
 - Arithmetic: `add`, `sub`, `addi`
 - Memory access: `lw`
 - branches and jumps: <https://eduassistpro.github.io/>
- These instructions view contents of register as a **single quantity** (such as a signed or unsigned integer)

Assignment Project Exam Help

Add WeChat edu_assist_pro

Bitwise Operations

- View contents of register as 32 **independent bits**
 - Since registers are composed of 32 bits, we may want to access individual bits (or groups of bits) rather than the whole.
- Two new classes of MIPS instructions for bitwise operations:
 - Logical Operators
 - Shift Operators

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Bitwise Operations

- Truth Table: lists all combinations of inputs and outputs

A	B	AND	OR	NOT
0	0	0	0	1
0	1	0	1	0
1	0	0	1	0
1	1	1	1	0

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

if both inputs are 1

if at least one input is 1

NOR:

if both inputs are 0

Bitwise Operations

- Bitwise result **applies function to each bit independently**
- The i^{th} bit of inputs produce the i^{th} bit of outputs

Assignment Project Exam Help

A	B	AND	OR	NO
0	0	0	0	1
0	1	0	1	0
1	0	0	1	0
1	1	1	1	0

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

AND (01101001, 11001100) = 01001000

Bit-wise OR

OR (01101001, 11001100) = 11101101

Boolean function applied at each bit position

Assignment Project Exam Help
MIP **ators**
<https://eduassistpro.github.io/>
Add WeChat edu_assist_pro

MIPS Logical Operations

- MIPS Logical Operators are *bitwise operations*
- Basic MIPS logical operators

Assignment Project Exam Help

<https://eduassistpro.github.io/>
and TargetReg, SourceReg1, SourceReg2
or TargetReg, SourceReg1, SourceReg2
nor TargetReg, SourceReg1, SourceReg2

- Like many MIPS instructions, logical operations accept exactly 2 inputs and produce 1 output

Use Logical Operator in Conditional Statement

- Conditional statements

```
// C
```

```
if ( A < 0 && B < 0 ) {  
    ...  
}
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

```
                # MIPS code  
slt $t0, $s0, $zero    # $t0 = $s0 < 0?  
slt $t1, $s1, $zero    # $t1 = $s1 < 0?  
and $t2, $t0, $t1      # $t2 = ( $s0 < 0 && $s1 < 0 ) ?
```

\$t0 = 0000 0000 0000 0000 0000 0000 0000 0001
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000

Use Logical Operator in Conditional Statement

- Conditional statements

```
// C
```

```
if ( A < 0 || B < 0 ) {  
    ...  
}
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

```
                # MIPS code  
slt $t0, $s0, $zero # $t0 = $s0 < 0?  
slt $t1, $s1, $zero # $t1 = $s1 < 0?  
or  $t2, $t0, $t1   # $t2 = ( $s0 < 0 || $s1 < 0 ) ?
```

```
$t0 = 0000 0000 0000 0000 0000 0000 0000 0001  
0000 0000 0000 0000 0000 0000 0000 0000  
0000 0000 0000 0000 0000 0000 0000 0001
```

Logical Operators with Immediate

- Similar to `and`, `or`, `nor`, but the third argument is an immediate

Syntax

`andi TargetReg immediate`
`ori TargetReg immediate`

<https://eduassistpro.github.io/>
Add WeChat edu_assist_pro

NOR for NOT

- Boolean expressions are made with **AND** and **OR** and **NOT**
- Why is **NOT** not a MIPS instruction?
 - **NOT** takes one operand and produces a result, which is not in keeping with the other instructions
 - How do we do **NOT** with **NOR**?

```
nor $t1, $t0, $zero
```

\$zero	Input	NOR
0	0	1
0	1	0

Assignment Project Exam Help

<https://eduassistpro.github.io/>

An Application of It
Add WeChat edu_assist_pro



No, not this one

Use AND for Mask

- Any bit *and* 0 produces an output 0

0	Input	AND
0	0	
0	1	

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

This can be used to
create a *mask*.

- Any bit *and* 1 produces the original bit

1	Input	AND
1	0	0
1	1	1

Use AND for Mask

Example:

A = 1011 0110 1010 0100 0011 1101 1001 1010

B = 0000 0000 0000 0000 0000 0000 1111 1111

A and B = 0000 0000 0000 0000 0000 0000 1001 1010

Assignment Project Exam Help

<https://eduassistpro.github.io/>

- In this example, **B** is called a **mask**
- **B** is used to isolate the rightmost 8 bits of **A** by *masking* out the rest of the string (e.g., setting it to all 0s)
- Thus, the **and** operator can be used to set certain portions of a bitstring to 0s, while leaving the rest alone.

Add WeChat edu_assist_pro

Use AND for Mask

Example: If A = 0xB6A43D9A is saved in \$t0, then what is \$t1 and \$t2 after the following instructions?

`andi $t1, $t0, 0xFF`

\$t0 = 1011 0110 1010 0100 0011 1101 1001 1010
0xFF = 0000 0000 0000 0000 0000 0000 1111 1111
\$t1 = 0000 0000 0000 0000 0000 0000 1001 1010 = 0x9A

`andi $t2, $t0, 0x000000FF`

\$t0 = 1011 0110 1010 0100 0011 1101 1001 1010
0x000000FF = 0000 0000 0000 0000 0000 0000 1111 1111
\$t2 = 0000 0000 0000 0000 0000 0000 1001 1010 = 0x9A

Uses OR for Mask

- Any bit `or` 0 produces the original bit

0	Input	OR
0	0	0
0	1	1

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assistpro

- Any bit `or` 1 produces 1

0	Input	OR
1	0	1
1	1	1

This can also be used
to create a *mask*.

Uses OR for Mask

- Can be used to force certain bits of a string to 1s.

Example: if \$t0 contains 0x12345678, then after

Assignment Project Exam Help

```
ori $
```

<https://eduassistpro.github.io/>

\$t1 contains 0x1234FFFF (high-order 16 bits untouched, low-order 16 bits are forced to 1s).

\$t0 = 0001 0010 0011 0100 0101 0110 0111 1000

0xFFFF = 0000 0000 0000 0000 1111 1111 1111 1111

\$t1 = 0001 0010 0011 0100 1111 1111 1111 1111

Assignment Project Exam Help

Shift Operators

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Shift Logical Instructions

- Shift Instruction Syntax:

Operation **TargetReg**, **SourceReg**, **ShiftAmount**

1) **Operation**:

2) **TargetReg**: register that receives value

3) **SourceReg**: register that provides original value

4) **ShiftAmount**: shift amount (non-negative constant < 32)

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat: edu_assist_pro

Shift Logical Instructions

- Shift left logical **sll**

- shifts left and fills emptied bits with 0s

Assignment Project Exam Help

```
sll TargetReg, SourceReg, ShiftAmount
```

<https://eduassistpro.github.io/>

- Shift right logical **srl**

Add WeChat edu_assist_pro

- shifts right and fills emptied bits with 0s

```
srl TargetReg, SourceReg, ShiftAmount
```

Shift Logical Instructions

Example:

Assume \$t0 contains 0001 0010 0011 0100 0101 0110 0111 1000
What are \$t1 and \$t2

Assignment Project Exam Help

<https://eduassistpro.github.io/>

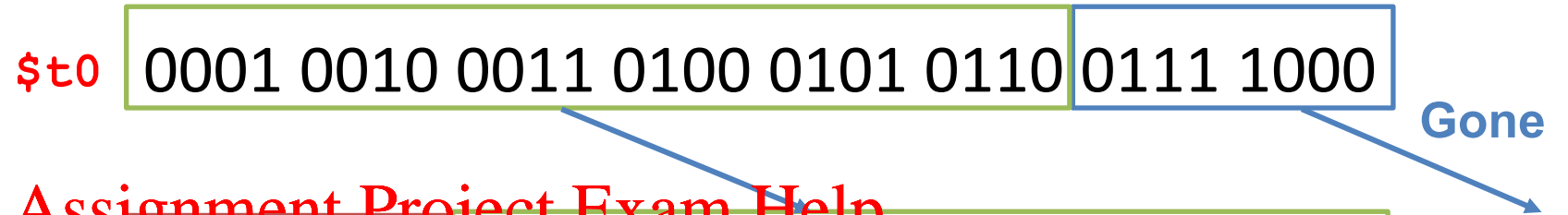
```
# shift right  
srl $t1, $t0, 8
```

Add WeChat

```
edu_assist_pro  
sll $t2, $t0, 8
```

Shift Logical Instructions

```
# shift right  
srl $t1, $t0, 8
```



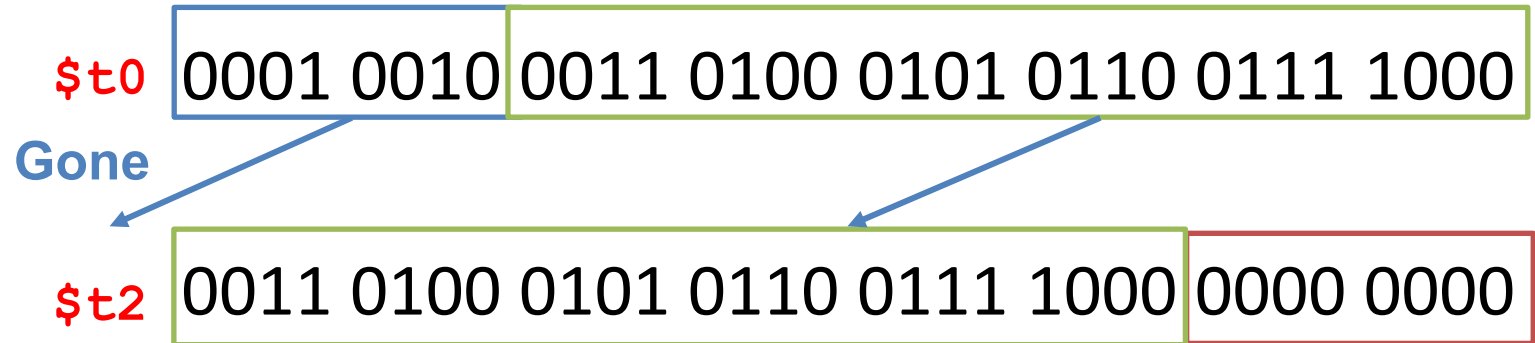
\$t1

	0011 0100 0101 0110
--	---------------------

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

```
# shift left  
sll $t2, $t0, 8
```



Fill with eight 0s

Shift Arithmetic Instructions

- Shift right arithmetic **sra**

- Shifts right and fills emptied bits by sign extending

Assignment Project Exam Help

sra **TargetReg** **if****tAmount**

<https://eduassistpro.github.io/>

- Why? A negative number should be active after shifting

Add WeChat edu_assist_pro

- If MSB = 0, shift and fill the new bits with 0s
- If MSB = 1, shift and fill the new bits with 1s

Shift Arithmetic Instructions

Example: SRA (shift right arithmetic) by 8 bits

\$t3 = 0001 0010 0011 0100 0101 0110 0111 1000

\$t4 = 1001 0010 0011 0100 0101 0110 0111 1000

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

What happen after shift right arith 8 bits?

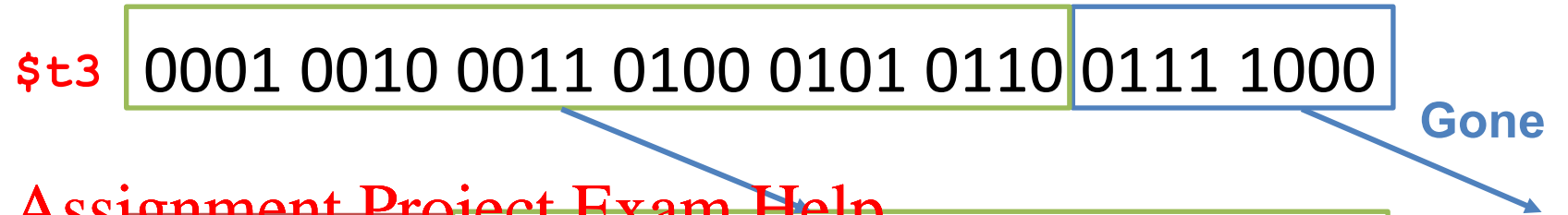
```
sra $t5, $t3, 8
```

```
sra $t5, $t4, 8
```


Shift Arithmetic Instructions

```
# shift right  
sra $t5, $t3, 8
```

If MSB = 0, the new bit
after shifting = 0



Assignment Project Exam Help

\$t5

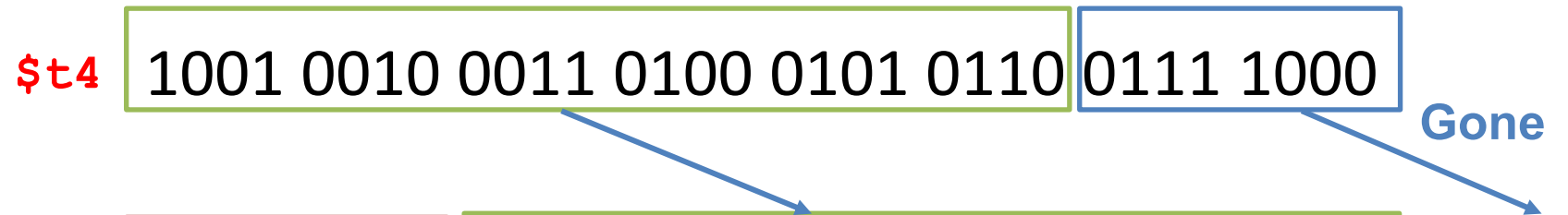
	0011 0100 0101 0110
--	---------------------

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

```
# shift right  
sra $t5, $t4, 8
```

If MSB = 1, the new bit
after shifting = 1



\$t5

1111 1111	1001 0010 0011 0100 0101 0110
-----------	-------------------------------

Fill with eight 1s

Use Shift Instructions in Multiplication

- In decimal:
 - Multiplying by 10 = shifting left by 1: $714_{10} \times 10_{10} = 7140_{10}$
 - Multiplying by 100 = $714_{10} \times 100_{10} = 71400_{10}$
 - Multiplying by 10^n = <https://eduassistpro.github.io/>
- In binary:
 - Multiplying by 2 = shifting left by 1: $11_2 \times 10_2 = 110_2$
 - Multiplying by 4 = shifting left by 2: $11_2 \times 100_2 = 1100_2$
 - Multiplying by 2^n = shifting left by n

Use Shift Instructions in Multiplication

- Shifting maybe faster than multiplication!
 - a good compiler usually notices when C code multiplies by a power of 2 and compiles it to a shift instruction:

`a = a * 8 ; //` <https://eduassistpro.github.io/> `# MIPS`

Add WeChat edu_assist_pro

- Likewise, shift right to divide by powers of 2
 - Use `sra` but watch out for negative numbers as the result is rounded down

`b = b / 2 ; //` `C`



`sra $s1, $s1, 1 # MIPS`

with `$s1 = -3` answer is -2

Use Shift to Extract Information

- Suppose we want to isolate byte 0 (rightmost 8 bits) of a word stored in \$t0

`andi $t0, $t0, 0xFF`

- Suppose we want to isolate byte 3 (fourth byte) of a word stored in \$t0.

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

`andi $t0, $t0, 0xFF00`

- How do we “extract” the information?

word



Use Shift to Extract Information

```
sll $t0, $t0, 16  
srl $t0, $t0, 24
```

0001 0010 0011 0100 0101 0110 0111 1000

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

0101 0110 0111 1000 0000 0000 0000 0000

0000 0000 0000 0000 0000 0000 0101 0110

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Application:
Pixel Data for Images

Example: Packing pixel data for images



PNG

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

- Suppose each pixel of an image
 - *Red* r [0 ~ 255]
 - *Green* g [0 ~ 255]
 - *Blue* b [0 ~ 255]
 - *Alpha* a , transparency value [0 ~ 255]

Each of them can be
represented by a byte (8-bit)

Example: Packing pixel data for images



PNG

Assignment Project Exam Help

<https://eduassistpro.github.io/>

- Instead of using 4 registers for r, g, b, a, we **pack** each of 8 bits into a 32-bit integer

```
// C code
int packARGB ( int r, int g, int b, int a ) {
    return a << 24 | r << 16 | g << 8 | b ;
}
```


Example: Packing pixel data for images

```
// C code
int packARGB ( int r, int g, int b, int a ) {
    return a << 24 | r << 16 | g << 8 | b ;
}
```

Assignment Project Exam Help

```
# MIPS
sll    $t0, $a3, 24    # shift a to the left
or     $v0, $t0, $zero # combine a with $v0
sll    $t0, $a2, 16    # shift r to the left
or     $v0, $t0, $v0    # combine r with $v0
sll    $t0, $a1, 8     # shift g to the left
or     $v0, $t0, $v0    # combine g with $v0
or     $v0, $a0, $v0    # combine b with $v0
jr     $ra             # return to $ra
```

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Example 2: Changing pixel formats



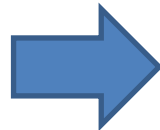
- **ARGB to ABGR**

- Write a conversion function (C or Java syntax):

`int argb2abgr(int ARGB)`

- Write function **processImage** <https://eduassistpro.github.io/>

- Convert to MIPS assembly [Add WeChat edu_assist_pro](#)



```
.data
# reserve space for 256x256 display
display: .space 0x40000
# filename of image to load
fname: .asciiz "wall-eImage.bin"

.text
main:      jal loadImage
           jal touchDisplay
           jal processImage
           li $v0, 10      # exit
           syscall

argb2abgr:
# TODO: finish this function
jr $ra

processImage:
# TODO: finish this function
jr $ra
```

MARS Bitmap Display Tool

Assignment Project Exam Help

- Open and connect to <https://eduassistpro.github.io/>
- For this example, change width to 1000 and height to 1000
Add WeChat edu_assist_pro
- Base address is static .data segment
- Display updates when memory written
 - Loading sets the memory, but not the display
 - Must call another function to touch the memory for image to show

Review

- Logical and Shift Instructions operate on bits individually, unlike arithmetic, which operate on entire word
- Use Logical and Shift Instructions to extract fields, either by masking or by shift
- New Instructions:
 - Logical: `and, andi, or, ori, nor`
 - Shift: `sll, srl, sra`
- Practice: try writing MIPS functions
- Textbook 2.6

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro