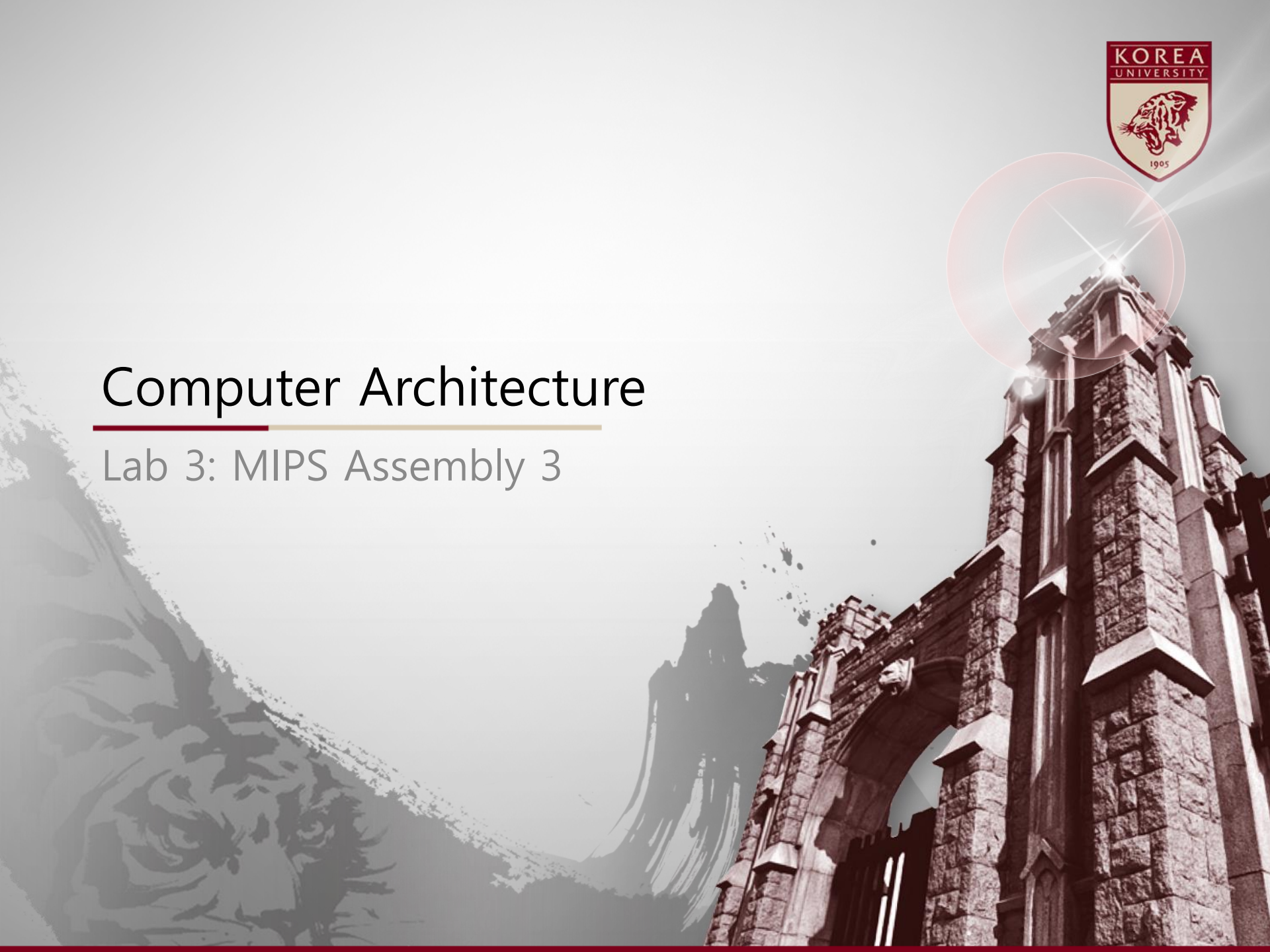


# Computer Architecture

## Lab 3: MIPS Assembly 3


# Getting Started

- Open a web browser and go to https://dannyqiu.me/mips-interpreter/


# Registers

| Preserved<br>*Callee-Saved* | Nonpreserved<br>*Caller-Saved* |
|---|---|
| `$s0-$s7` | `$t0-$t9` |
| `$ra` | `$a0-$a3` |
| `$sp` | `$v0-$v1` |
| **stack above `$sp`** | **stack below `$sp`** |

# Lab 3-1

- Copy and paste the code on the right to the code area.
- Click “Reset”
- Click “Step” and observe the changes of the registers

- Note
  - Add “nop” after every jump/branch instruction (delay slot)

```
addiu $sp, $0, 0x1000
j main
nop

func2:
  jr $ra
  nop

func1:
  addiu $sp, $sp, -4
  sw $ra, 0($sp)
  jal func2
  nop
  lw $ra, 0($sp)
  addiu $sp, $sp, 4
  jr $ra
  nop

main:
  jal func1
  nop
```


# Lab Assignment

- Write MIPS assembly code to compute the factorial(!) of the given input.
- The input is given in $s0, and the output should be in $s1.
- You can assume that the input ($s0) is always greater than 0 ($s0>0).
- The multiplication function is given in the next slide.
  - $v0 = $a0 * $a1
- You should use the multiplication function following the MIPS function call convention, but your factorial implementation does not have to use a recursive algorithm (you may, though, if you want).
- Submit your source code to Blackboard


# Multiplication Function

```
addiu $sp, $0, 0x1000     # stack pointer
addiu $s0, $0, 5          # input
j main
nop

# multiplication
mult:
  addiu $v0, $0, 0
  addiu $t0, $0, 0
multloop:
  slt $t1, $t0, $a1
  beq $t1, $0, multreturn
  nop
  addu $v0, $v0, $a0
  addiu $t0, $t0, 1
  j multloop
  nop
multreturn:
  jr $ra
  nop

main:
# implement your algorithm here replacing following lines
  addu $a0, $0, $s0
  addiu $a1, $0, 4
  jal mult
  nop
  addu $s1, $0, $v0        # output
```