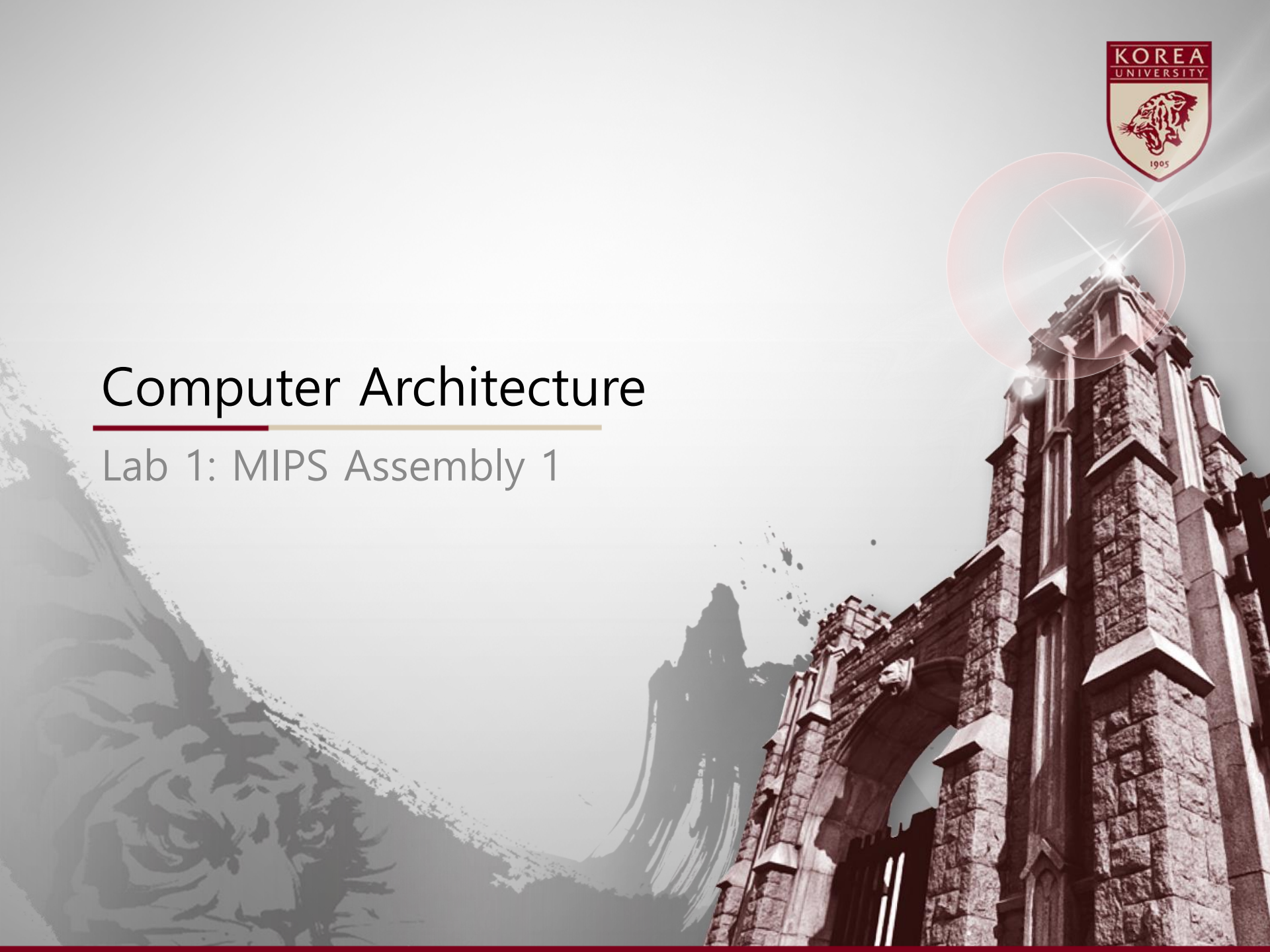

Computer Architecture

Lab 1: MIPS Assembly 1

Getting Started

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

MIPS Interpreter

Input your MIPS code here:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

Reset | Stop | CPU: 32 Hz

```
[line 17]: sw $t3, 4($s1)
[line 18]: addiu $s0, $s0, 1
[line 20]: j for
[line 11]: beq $s0, $t0, done
[line 13]: addiu $s1, $s1, 4 (Delay Slot)
No more instructions to run! Press Reset to reload the code!
```

Features

- *Reset* to load the code, *Step* one instruction, or *Run* all instructions
- Set a breakpoint by clicking on the line number (only for *Run*)
- View registers on the right, memory on the bottom of this page
- **A Delay slot is used for all jumps/branches. Branch offset addresses are relative to the delay slot instruction.**

Supported Instructions

- Immediate Arithmetic: `ADDIU` , `ANDI` , `ORI` , `XORI` , `SLTI` , `SLTIU`
- Register Arithmetic: `ADDU` , `SUBU` , `AND` , `OR` , `XOR` , `NOR` , `SLT` , `SLTU`
- Move: `MOVN` , `MOVZ`
- Shifts: `SLL` , `SRL` , `SRA` , `SLLV` , `SRLV` , `SRAV`
- Immediate Load: `LUI`
- Control: `J` , `JR` , `JAL` , `JALR` , `BEQ` , `BNE` , `BLEZ` , `BGTZ` , `BLTZ` , `BGEZ`
- Memory: `LW` , `LB` , `LBU` , `SW` , `SB`

MIPS Reference: mips_vol2.pdf

Init Value	Register	Decimal	Hex	Binary
0	$0 ($zero)	0	0x00000000	0b00000000000000000000000000000000
0	$1 ($at)	0	0x00000000	0b00000000000000000000000000000000
0	$2 ($v0)	0	0x00000000	0b00000000000000000000000000000000
0	$3 ($v1)	0	0x00000000	0b00000000000000000000000000000000
0	$4 ($a0)	0	0x00000000	0b00000000000000000000000000000000
0	$5 ($a1)	0	0x00000000	0b00000000000000000000000000000000
0	$6 ($a2)	0	0x00000000	0b00000000000000000000000000000000
0	$7 ($a3)	0	0x00000000	0b00000000000000000000000000000000
0	$8 ($t0)	10	0x0000000a	0b00000000000000000000000000001010
0	$9 ($t1)	21	0x00000015	0b00000000000000000000000000010101
0	$10 ($t2)	34	0x00000022	0b00000000000000000000000000100010
0	$11 ($t3)	55	0x00000037	0b00000000000000000000000000110111
0	$12 ($t4)	0	0x00000000	0b00000000000000000000000000000000
0	$13 ($t5)	0	0x00000000	0b00000000000000000000000000000000
0	$14 ($t6)	0	0x00000000	0b00000000000000000000000000000000
0	$15 ($t7)	0	0x00000000	0b00000000000000000000000000000000
0	$16 ($s0)	10	0x0000000a	0b00000000000000000000000000001010
0	$17 ($s1)	36	0x00000024	0b00000000000000000000000000100100
0	$18 ($s2)	0	0x00000000	0b00000000000000000000000000000000
0	$19 ($s3)	0	0x00000000	0b00000000000000000000000000000000
0	$20 ($s4)	0	0x00000000	0b00000000000000000000000000000000
0	$21 ($s5)	0	0x00000000	0b00000000000000000000000000000000
0	$22 ($s6)	0	0x00000000	0b00000000000000000000000000000000
0	$23 ($s7)	0	0x00000000	0b00000000000000000000000000000000
0	$24 ($t8)	0	0x00000000	0b00000000000000000000000000000000
0	$25 ($t9)	0	0x00000000	0b00000000000000000000000000000000
0	$26 ($k0)	0	0x00000000	0b00000000000000000000000000000000
0	$27 ($k1)	0	0x00000000	0b00000000000000000000000000000000
0	$28 ($gp)	0	0x00000000	0b00000000000000000000000000000000
0	$29 ($sp)	0	0x00000000	0b00000000000000000000000000000000
0	$30 ($fp)	0	0x00000000	0b00000000000000000000000000000000
0	$31 ($ra)	0	0x00000000	0b00000000000000000000000000000000

Memory Address 0x00000000 Go Download!

Memory Address	Decimal	Hex	Binary
0x00000000	1	0x00000001	0b00000000000000000000000000000001
0x00000004	1	0x00000001	0b00000000000000000000000000000001
0x00000008	2	0x00000002	0b00000000000000000000000000000010
0x0000000c	3	0x00000003	0b00000000000000000000000000000011
0x00000010	5	0x00000005	0b00000000000000000000000000000101
0x00000014	8	0x00000008	0b00000000000000000000000000001000
0x00000018	13	0x0000000d	0b00000000000000000000000000001101
0x0000001c	21	0x00000015	0b00000000000000000000000000010101
0x00000020	34	0x00000022	0b00000000000000000000000000100010
0x00000024	55	0x00000037	0b00000000000000000000000000110111

Lab 1-1

- Copy and paste the code on the right to the code area.
- Note
  - Add “nop” instruction after every jump/branch instruction (delay slot)
- Click “Reset”
- Click “Step” and observe the changes of the registers

```
main:
addiu $s0, $0, 0
addiu $t0, $0, 10

for:
beq $s0, $t0, done
nop
addiu $s0, $s0, 1
j for
nop

done:
```

Lab Assignment

- Write MIPS assembly code to perform modular operation.
- Two inputs are stored in $s0 and $s1, and the output is stored in $s2.
  - $s2 = $s0 % $s1
- Inputs and output are unsigned integers.
- Submit your source code to Blackboard.

Expected Result