# CSE301 – Computer Organization

## Lab 2: Input/Output to console - Arithmetic operation - control the flow

### Input/Output to console - System Calls (syscall) — how to use

Procedure:

1. Put syscall number in $v0.
2. Put parameters in $a0, $a1, $a2, ... (or $f12 for some FP syscalls).
3. Execute syscall.
4. Return values come back in $v0 (and sometimes $v1, $f0).

Common MIPS SPIM/PCSpim syscall numbers (standard simulator set):

| $v0 | Operation | Args / Return |
|---|---|---|
| 1 | print integer | $a0 = int |
| 2 | print float | $f12 = float |
| 3 | print double | $f12 = double |
| 4 | print string | $a0 = address (null-terminated) |
| 5 | read integer | returns in $v0 |
| 6 | read float | returns in $f0 |
| 7 | read double | returns in $f0 |
| 8 | read string | $a0 = buffer, $a1 = length |
| 9 | sbrk (allocate) | $a0 = bytes, returns pointer in $v0 |
| 10 | exit | no args |
| 11 | print char | $a0 = char (int) |
| 12 | read char | returns ASCII in $v0 |
| 13-16 | file ops (open/read/write/close) | simulator dependent |
| 17 | exit2 | $a0 = exit code |

### Arithmetic operations

| Instruction | Type | Example | Effect |
|---|---|---|---|
| add | R | add $t0,$t1,$t2 | signed add, traps on overflow |
| addu | R | addu $t0,$t1,$t2 | unsigned add, no overflow trap |
| sub | R | sub $t0,$t1,$t2 | signed subtract, traps on overflow |
| subu | R | subu $t0,$t1,$t2 | unsigned subtract |
| addi | I | addi $t0,$t1,10 | signed add immediate, traps on overflow |
| addiu | I | addiu $t0,$t1,-1 | unsigned immediate add (commonly used even for signed constants) |

| | | | |
|---|---|---|---|
| mult / multu | R | mult $t1,$t2 | 64-bit result in HI/LO |
| div / divu | R | div $t1,$t2 | quotient → LO, remainder → HI |
| mfhi / mflo | R | mflo $t0 | move from HI/LO |

### Logical / Bitwise

| Instruction | Example | Effect |
|---|---|---|
| and | and $t0,$t1,$t2 | bitwise AND |
| or | or $t0,$t1,$t2 | bitwise OR |
| xor | xor $t0,$t1,$t2 | bitwise XOR |
| nor | nor $t0,$t1,$t2 | bitwise NOR |
| andi/ori/xori | ori $t0,$t1,0xFF | immediate logical ops (zero-extended immediate) |
| sll/srl/sra | sll $t0,$t1,2 | shift left logical / right logical / right arithmetic |

**Lab Exercise 1: Add Two Numbers**

Write a MIPS assembly program that performs the following tasks:

- Prompts the user to enter two integers.
- Reads both numbers from the console.
- Adds the two numbers together.
- Displays the result with an appropriate message.

**Task:**

**Task 1:** Write an assembly language program using MIPS architecture to enter four integers from user and determine and display the average of four numbers. Program should include prompt message for user inputs and appropriate message for displayed result.

### Branch Instructions

| Instr | Example | Effect |
|---|---|---|
| beq | beq $t0, $t1, label | branch if equal |
| bne | bne $t0, $t1, label | branch if not equal |
| blez / bgtz | blez $t0, label | branch on <=0 / >0 |
| j | j target | jump (pseudo converts 26-bit target) |
| jal | jal function | jump and link (stores return address in $ra) |
| jr | jr $ra | jump to register (return) |

# Pseudo-instructions (assembler expands them)

Pseudo-instructions make code readable; assembler rewrites them using real instructions. Examples:

| Pseudo | Expands to | Example / notes |
|---|---|---|
| move rd, rs | add rd, rs, $zero | copy register |
| li rd, imm | addi or lui+ori | load 32-bit constant (assembler picks sequence) |
| la rd, label | addi/lui+ori/add | load address of label |
| bgt / ble / blt / bge | combination of slt/bne/beq | branch pseudo |
| mul rd, rs, rt | mult + mflo (or mul real on some ISAs) | multiplies and stores low 32 bits |

**Lab Exercise 2: If Condition Example**

Write a MIPS assembly program that reads an integer from the user and checks whether it is positive, negative, or zero. Display an appropriate message for each case.

**Lab Exercise 3: While Loop Example**

Write a MIPS program that prints numbers from 1 to 10 using a **while loop** structure.

**Lab Exercise 4: Do-While Loop Example**

Write a MIPS assembly program that reads numbers from the user and sums them until the user enters zero. Display the final sum at the end.

**Lab Exercise 5: For Loop Example**

Write a MIPS program that calculates the sum of the first *n* natural numbers using a **for loop** structure.

**Task:**

**Task 2:** Write an assembly language program using MIPS architecture to enter n integers from user and determine and display the average of n numbers where number of integer n is entered by user. Program should include prompt message for user inputs and appropriate message for displayed result.

**Task 3:** Write an assembly language program using MIPS architecture to implement switch...case.