King Fahd University of Petroleum and Minerals
College of Computer Sciences and Engineering
Computer Engineering Department
COE 301: Computer Architecture

LAB 02: Introduction to MIPS Assembly Programing

Saleh AlSaleh

Agenda

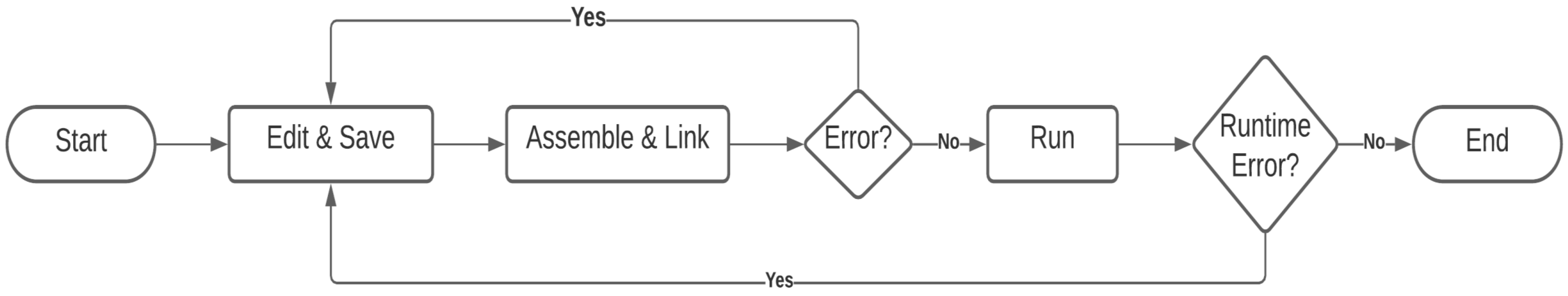
- MIPS Assembly Language Program Template
- The Edit-Assemble-Link-Run Cycle
- MIPS Instructions Format
- MIPS General Purpose Registers
- System Calls
- Live Examples
- Tasks

MIPS Assembly Program Language

```
# Title:
# Author:
# Date:
# Description:
# Input:
# Output:
################### Data segment #####################
.data
 . . .
################### Code segment #####################
.text
.globl main
main:                      # main function entry
 . . .
li $v0, 10
syscall                    # system call to exit program
```

- Comments start with “#”
- Directives starts with “.”
- Instructions are usually written in lower case; however, you can write them in uppercase as well.
- Registers name or number starts with “$”.

Edit-Assemble-Link-Run Cycle



MIPS Instruction Formats

- R-Type Format
  - Both operands are registers
  - E.g. add $t0, $t1, $t2
- I-Type Instructions
  - One operand is a register and the other one is a 16-bit immediate value
  - E.g. addi $t0, $t1, 5
- J-Type Instructions
  - Used for jump instructions with 26-bit immediate value
  - E.g. j loop

Op^6	Rs^5	Rt^5	Rd^5	sa^5	funct^6

Op^6	Rs^5	Rt^5	Immediate^{16}

Op^6	immediate^{26}

MIPS General Purpose Registers

Register Name	Register Number	Register Usage
$zero	$0	Always zero, forced by hardware
$at	$1	Assembler Temporary register, reserved for assembler use
$v0 - $v1	$2 - $3	Results of a function
$a0 - $a3	$4 - $7	Arguments of a function
$t0 - $t7	$8 - $15	Registers for storing temporary values
$s0 - $s7	$16 - $23	Registers that should be saved across function calls
$t8 - $t9	$24 - $25	Registers for storing more temporary values
$k0 - $k1	$26 - $27	Registers reserved for the OS kernel use
$gp	$28	Global Pointer register that points to global data
$sp	$29	Stack Pointer register that points to top of stack
$fp	$30	Frame Pointer register that points to stack frame
$ra	$31	Return Address register used to return from a function call

System Calls

Service	Code in $v0	Arguments	Results
Print Integer	1	$a0 = integer to print	
Print String	4	$a0 = address of null-terminated string	
Read Integer	5		$v0 = integer read
Read String	8	$a0 = address of input buffer $a1 = maximum number of characters to read	
Exit Program	10		Terminate program
Print Char	11	$a0 = character to print	
Read Char	12		$v0 = character read

Live Examples