

Lab# 12 THE SINGLE CYCLE DATAPATH

Instructor: I Putu Danu Raharja.

Objectives:

Learn how to implement instructions for a CPU.

Method:

Learn to implement the single cycle datapath for a subset of 16-bit MIPS-like processor.

Preparation:

Read the slides.

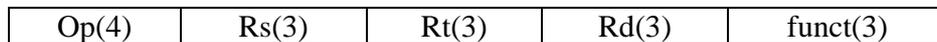
File To Use:

12.1 OVERVIEW:

Suppose we would like to design a simple 16-bit MIPS-like processor with seven 16-bit general-purpose registers: R1 through R7. R0 is hardwired to zero and cannot be written, so we are left with seven registers. There is also one special-purpose 16-bit register, which is the program counter (PC). All instructions are also 16 bits. There are three instruction formats, R-type, I-type, and J-type as shown below:

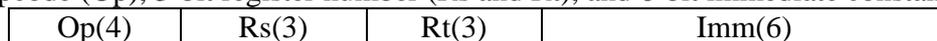
R-type format:

4-bit opcode (Op), 3-bit register numbers (Rs, Rt, and Rd), and 3-bit function field (funct)



I-type format:

4-bit opcode (Op), 3-bit register number (Rs and Rt), and 6-bit immediate constant



J-type format:

4-bit opcode (Op) and 12-bit immediate constant



For R-type instructions, Rs and Rt specify the two source register numbers, and Rd specifies the destination register number. The function field can specify at most eight functions for a given opcode. We will reserve opcode 0 and opcode 1 for R-type instructions.

For I-type instructions, Rs specifies a source register number, and Rt can be a second source or a destination register number. The immediate constant is only 6 bits because of

the fixed size nature of the instruction. The size of the immediate constant is suitable for our uses. The 6-bit immediate constant is signed (and sign-extended) for all I-type instructions.

For J-type, a 12-bit immediate constant is used for instructions such as J (jump), JAL (jump-and-link), and LUI (load upper immediate) instructions.

Instruction Encoding:

Eight R-type instructions, six I-type instructions, and three J-type instructions are defined. These instructions, their meanings, and their encodings are shown below:

Instr	Meaning	Encoding				
		Op	Rs	Rt	Rd	f
OR	$\text{Reg}(\text{Rd}) = \text{Reg}(\text{Rs}) \mid \text{Reg}(\text{Rt})$	Op = 0000	Rs	Rt	Rd	f = 000
AND	$\text{Reg}(\text{Rd}) = \text{Reg}(\text{Rs}) \& \text{Reg}(\text{Rt})$	Op = 0000	Rs	Rt	Rd	f = 001
NOR	$\text{Reg}(\text{Rd}) = \sim(\text{Reg}(\text{Rs}) \mid \text{Reg}(\text{Rt}))$	Op = 0000	Rs	Rt	Rd	f = 010
XOR	$\text{Reg}(\text{Rd}) = \text{Reg}(\text{Rs}) \wedge \text{Reg}(\text{Rt})$	Op = 0000	Rs	Rt	Rd	f = 011
ADD	$\text{Reg}(\text{Rd}) = \text{Reg}(\text{Rs}) + \text{Reg}(\text{Rt})$	Op = 0000	Rs	Rt	Rd	f = 100
SUB	$\text{Reg}(\text{Rd}) = \text{Reg}(\text{Rs}) - \text{Reg}(\text{Rt})$	Op = 0000	Rs	Rt	Rd	f = 101
SLT	$\text{Reg}(\text{Rd}) = \text{Reg}(\text{Rs}) < \text{Reg}(\text{Rt})$	Op = 0000	Rs	Rt	Rd	f = 110
JR	Jump register: $\text{PC} = \text{Reg}(\text{Rs})$	Op = 0000	Rs	000	000	f = 111
ADDI	$\text{Reg}(\text{Rt}) = \text{Reg}(\text{Rs}) + \text{ext}(\text{im}^6)$	Op = 0100	Rs	Rt	Immediate ⁶	
SLTI	$\text{Reg}(\text{Rt}) = \text{Reg}(\text{Rs}) < \text{ext}(\text{im}^6)$	Op = 0110	Rs	Rt	Immediate ⁶	
LW	$\text{Reg}(\text{Rt}) = \text{Mem}(\text{Reg}(\text{Rs}) + \text{ext}(\text{im}^6))$	Op = 1000	Rs	Rt	Immediate ⁶	
SW	$\text{Mem}(\text{Reg}(\text{Rs}) + \text{ext}(\text{im}^6)) = \text{Reg}(\text{Rt})$	Op = 1001	Rs	Rt	Immediate ⁶	
BEQ	Branch if $(\text{Reg}(\text{Rs}) == \text{Reg}(\text{Rt}))$	Op = 1010	Rs	Rt	Immediate ⁶	
BNE	Branch if $(\text{Reg}(\text{Rs}) \neq \text{Reg}(\text{Rt}))$	Op = 1011	Rs	Rt	Immediate ⁶	
J	$\text{PC} = \text{PC} + 1 + \text{ext}(\text{im}^{12})$	Op = 1100	Immediate ¹²			
JAL	$\text{R7} = \text{PC} + 1, \text{PC} = \text{PC} + 1 + \text{ext}(\text{im}^{12})$	Op = 1101	Immediate ¹²			
LUI	$\text{R1} = \text{Immediate}^{12} \ll 4$	Op = 1111	Immediate ¹²			

12.2 LAB EXERCISE

Based on the above requirement, implement only the datapath.