MIPS ISA

Jonathan Shihata

Chanartip Soonthornwan

TABLE OF CONTENTS

INTRODUCTION #

PURPOSE #

INSTRUCTION SET ARCHITECTURE #

* R-TYPE INSTRUCTION #
* I-TYPE INSTRUCTION #
* J-TYPE INSTRUCTION #
* ENHANCED INSTRUCTION #

EXAMPLE #

R-TYPE INSTRUCTION #

shift left logic (SLL) #

shift right logic (SRL) #

shift right arithmetic (SRA) #

jump register(JR) #

move from hi (MFHI) #

move from lo (MFLO) #

multiplication (MULT) #

division (DIV) #

addition signed (ADD) #

addition unsigned (ADDU) #

subtraction signed(SUB) #

subtraction unsigned(SUBU) #

and (AND) #

or (OR) #

exclusive or (XOR) #

not or (NOR) #

set less than (SLT) #

set less than unsigned (SLTU) #

breakpoint (BREAK) #

(SETIE) #

I-TYPE INSTRUCTION #

branch equal (BEQ) #

branch not equal (BNE) #

branch less than or equal to zero (BLEZ) #

branch greater than zero (BGTZ) #

addition immediate (ADDI) #

set less than immediate (SLTI) #

set less than immediate unsigned (SLTIU) #

and immediate (ANDI) #

or immediate (ORI) #

exclusive or immediate (XORI) #

load upper immediate (LUI) #

load word (LW) #

store word (SW) #

J-TYPE INSTRUCTION #

jump (J) #

jump and link (JAL) #

ENHANCED INSTRUCTION #

INPUT #

OUTPUT #

RETI #

“E\_KEY” #

CONCLUSION #

**this page is intentionally left blank**

example instruction name

**31 26 25 21 20 16 15 11 10 6 5 0**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **opcode**  **000000** | **rs**  **00000** | **rt**  **00000** | **rd**  **00000** | **shamt**  **00000** | **funct**  **000000** |

**6 bits 5 bits 5 bits 5 bits 5 bits 6 bits**

format: example rd, rs, rt

purpose: to do an exampleop on 32-bit integers.

description: rd <= exampleop rt

this section describe the operation of the instruction in text, tables, and pictures. it will include information about the instruction that is hard to encode in the operation section so they must be used together to completely understand the instruction.

restrictions:

this section lists any restrictions for the instruction. things which may be restricted include the values of instruction encoding fields such as register specifiers, operand values, operand formats, address alignment, instruction scheduling hazards, and type of memory access for addressed locations.

operation:

/\* this section describes the operation of the instruction in a high-level

\* language. it is precise in ways that the *description* section is not, but

\* it is also missing information that hard to express in pseudocode.\*/

temp <- gpr[rs] **exampleop** gpr[rt]

gpr[rd <= signExtend[temp31…0]

exceptions:

a list of the exceptions taken by the instruction such as integer Overflow

**programming notes:**

this contains information useful for programmers, but not necessary to describe the operation of the instruction.

**implementation notes:**

like *programming notes* except for processor implementers.

branch on equal

**31 26 25 21 20 16 15 11 10 6 5 0**

|  |  |  |  |
| --- | --- | --- | --- |
| **opcode**  **000100** | **rs**  **00000** | **rt**  **00000** | **16-bit offset**  **0000000000000000** |

**6 bits 5 bits 5 bits 5 bits 5 bits 6 bits**

format: beq rs, rt, offset

purpose: equal register-register compare and branch with 16-bit offest.

description: if(r[rs] == r[rt]) pc ← pc + 4 + branch\_addr

if the condition is true, the branch is taken and the follwed instruction will not be executed. if the branch is taken, 16-bit offset will be used to reform a pc-relative effective address by concatenated with 14-bit of offset[15], the offset, and two zeroes.

restrictions:

the branch may be followed by another branch or jump that would delay the execution

operation:

branch\_addr ← {14{offset[15], offset, 2’b0}}

beq: cond ← gpr[rs] = gpr[rt]

if cond then

pc ← (pc+4+ branch\_addr)

end if

exceptions:

n/a

**programming notes:**

**example:**