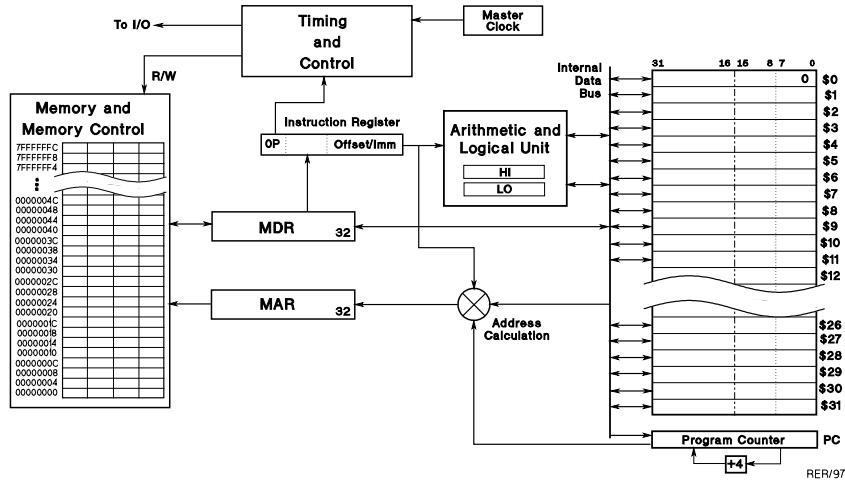
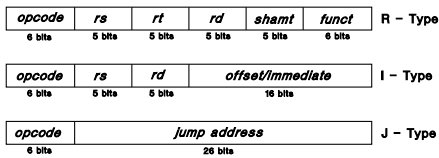


MIPS R2000 – Instruction Reference Card

Programmer's Model of the MIPS R2000 Microprocessor



• MIPS Binary Instruction Formats



Fields in the instruction formats are:

- opcode* - (6 bits)
- rs* - (5 bits) first source register
- rt* - (5 bits) second source register
- rd* - (5 bits) destination register
- shamt* - (5 bits) shift amount
- funct* - (6 bits) additional opcode
- off/imm* - (16 bits) immediate value/address offset
- jmp addr* - (26 bits) jump address

• Register Conventional Usage

Name	Register	Description/Usage
	\$0	Hardwired constant 0
\$at	\$1	Reserved for assembler
\$v0-\$v1	\$2-\$3	Return results from functions
\$a0-\$a3	\$4-\$7	Used for subroutine arguments.
\$t0-\$t7	\$8-\$15	"Temporary," not saved across a call
\$s0-\$s7	\$16-\$23	"Saved," saved across a call
\$t8-\$t9	\$24-\$25	More temporary
\$k0-\$k1	\$26-\$27	"Kernel" - reserved for OS use
\$gp	\$28	Pointer to global data area
\$sp	\$29	Stack Pointer
\$fp	\$30	Frame pointer
\$ra	\$31	Return address of subroutine

• Addressing Modes Used with Load/Stores

Mode	Syntax	Effective Address
Relative	RSYMB	EA = [PC] + offset
Register Indirect	(\$n)	EA = [\$n]
Base	offset(\$n)	EA = [\$n] + offset

• Load/Store, Data Movement Instructions

Description	Opcode	Operand	Format
Load Address	la	rd, mem	†
Load Byte	lb	rd, mem	I
Load Byte Unsigned	lbu	rd, mem	I
Load Halfword	lh	rd, mem	I
Load Half Unsigned	lhu	rd, mem	I
Load Word	lw	rd, mem	I
Load Immediate	li	rd, imm	†
Load Upper Immed	lui	rd, imm	I
Store Word	sw	rs, mem	I
Store Halfword	sh	rs, mem	I
Store Byte	sb	rs, mem	I
Move	move	rd, rs	†

• Arithmetic/Logical Instructions

Description	Opcode	Operand	Format
Add	add	rd, rs, rt	R
Add Immediate	addi	rd, rs, imm	I
Add Unsigned	addu	rd, rs, rt	R
Add Unsigned Immed	addiu	rd, rs, imm	I
Divide	div	rd, rs, rt	†
Divide Unsigned	divu	rd, rs, rt	†
Multiply	mul	rd, rs, rt	†
Multiply Unsigned	mulu	rd, rs, rt	†
Negate	neg	rd, rs	†
Remainder	rem	rd, rs, rt	†
Subtract	sub	rd, rs, rt	R
Subtract Unsigned	subu	rd, rs, rt	R
And	and	rd, rs, rt	R
And Immediate	andi	rd, rs, imm	I
Nor	nor	rd, rs, rt	R
Not	not	rd, rs	†
Or (Inclusive)	or	rd, rs, rt	R
Or Immediate	ori	rd, rs, imm	I
eXclusive OR	xor	rd, rs, rt	R
eXclusive OR Immed	xori	rd, rs, imm	I

• Branch/Jump Instructions

Description	Opcode	Operand*	Fmt
Branch (always)	b	label	†
Jump	j	label	J
Branch if Equal	beq	rs, rt, label	I
Branch if Not Equal	bne	rs, rt, label	I
Branch if Greater	bgt	rs, rt, label	†
Branch if Greater or Equal	bge	rs, rt, label	†
Branch if Greater Unsigned	bgtu	rs, rt, label	†
Branch if Greater or Equal Unsigned	bgeu	rs, rt, label	†
Branch if Less Than	blt	rs, rt, label	†
Branch if Less or Equal	ble	rs, rt, label	†
Branch if Less Unsigned	bltu	rs, rt, label	†
Branch if Less or Equal Unsigned	bleu	rs, rt, label	†
Branch if Equal to Zero	beqz	rs, label	†
Branch if Not Equal to Zero	bnez	rs, label	†
Branch if Greater Than Zero	bgtz	rs, label	I
Branch if Greater or Equal to Zero	bgez	rs, label	I
Branch if Less Than Zero	bltz	rs, label	I
Branch if Less or Equal to Zero	blez	rs, label	I

* In the three operand branch instructions, *rt* can be replaced with a constant, i.e., operands can be *rs,imm,label*. All such instructions are pseudoinstructions.

† - pseudoinstruction

• Shift and Rotate Instructions

Description	Opcode	Operand	Fmt
Rotate Left	rol	rd, rs, rt	†
Rotate Right	ror	rd, rs, rt	†
Shift Left Logical	sll	rd, rs, sa	R
Shift Left Log Variable	sllv	rd, rs, rt	R
Shift Right Logical	srl	rd, rs, sa	R
Shift Right Log Variable	srlv	rd, rs, rt	R
Shift Right Arithmetic	sra	rd, rs, sa	R
Shift Right Arith Variable	srav	rd, rs, rt	R

• Comparison Instructions

Description	Opcode	Operand	Fmt
Set if Equal	seq	rd, rs, rt	†
Set if Not Equal	sne	rd, rs, rt	†
Set if Less Than	slt	rd, rs, rt	R
Set if Less or Equal	sle	rd, rs, rt	†
Set if Greater Than	sgt	rd, rs, rt	†
Set if Greater or Equal	sge	rd, rs, rt	†
Set if Less Unsigned	sltu	rd, rs, rt	R
Set if Less or Equal Unsigned	sleu	rd, rs, rt	†
Set if Greater Unsigned	sgtu	rd, rs, rt	†
Set if Greater or Equal Unsigned	sgeu	rd, rs, rt	†
Set if Less Than Immed	slti	rd,rs,imm	I
Set if Less Than Imm Unsigned	sltiu	rd,rs,imm	I

• Subroutine Instructions

Description	Opcode	Operand	Format
Jump and Link (Call)	jal	label	J
Jump Register	jr	rs	R

• Miscellaneous Instructions

Description	Opcode	Operand	Format
No Operation	nop		†
System Call	syscall	code	R

• Assembler Syntax

```
label: mnemonic operands # comment
```

• Assembler Directives - SPIM Subset

Directive	Description
.text	Put next code into <i>text</i> section
.data	Put next code into <i>data</i> section
.globl name	Make <i>name</i> be a global symbol
.space n	Allocate <i>n bytes</i> of space
.word val1,val2, ...	Allocate one word for each value
.half val1,val2, ...	Allocate a halfword for each value
.byte val1,val2, ...	Allocate a byte for each value
.ascii "string"	Allocate space for the ASCII chars
.asciiz "string"	Allocate space for string, NULL terminated
.align n	Start next on 2 ⁿ byte boundary

• SPIM System Services

Routine	Code in \$2	Arguments	Result
print_int	1	\$4 = integer	
print_str	4	\$4 = addr of string	
read_int	5		integer in \$2
read_str	8	\$4 = addr of buffer (memory) \$5 = max length of str + 1	str in buffer
malloc	9	\$4 = # of bytes desired	address in \$2
exit	10		