Core Instruction Set

Name	Mnemonic	Format
Add	add	R
Add Immediate	addi	I
Add Imm. Unsigned	addiu	I
Add Unsigned	addu	R
And	and	R
And Immediate	andi	I
Branch On Equal	beq	I
Branch on Not Equal	bne	I
Jump	j	J
Jump and Link	jal	J
Jump Register	jr	R
Load Byte Unsigned	ibu	I
Load Halfword Unsigned	lhu	I
Load Linked	ll	
Load Upper Imm.	lui	I
Load Word	lw	I
Nor	nor	R
Or	or	R
Or Immediate	ori	I
Set Less Than	slt	R
Set Less Than Imm.	slti	I
Set Less Than Imm. Unsigned	sltiu	I
Set Less Than Unsigned	sltu	I
Shift Left Logical	sll	R
Shift Right Logical	srl	R
Store Byte	sb	I
Store Conditional	sc	I
Store Halfword	sh	I
Store Word	*sw*	I
Subtract	sub	I
Subtract Unsigned	subu	R

Arithmetic Core Instruction Set

Name	Mnemonic	Format
Branch On FP True	bclt	FI
Branch On FP False	*bclf*	Fi
Divide	div	R
Divide Unsigned	divu	R
FP Add Single	add.s	FR
FP Add Double	add.d	FR
FP Compare Single	c.x.s*	FR
FP Compare Double	c.x.d*	FR
FP Divide Single	div.s	FR

FP Divide Double	div.d	FR
FP Multiply Single	mul.s	FR
FP Multiply Double	mul.d	FR
FP Subtract Single	sub.s	FR
FP Subtract Double	sub.d	FR
Load FP Single	lwcl	I
Load FP Double	ldcl	I
Move From Hi	mfhi	R
Move From Lo	mflo	R
Move From Control	*mfcO*	R
Multiply	mult	R
Multiply Unsigned	*multu*	R
Shift Right Ari th.	*sra*	R
Store FP Single	swcl	I
Store FP Double	sdcl	I

Operation

R[rd] = R[rs] + R[rt]
R[rt] = R[rs] + SignExtImm
R[rt] = R[rs] + SignExtImm
R[rd] = R[rs] + R[rt]
R[rd] = R[rs] & R[rt]
R[rt] = R[rs] & ZeroExtImm
if(R[rs]==R[rt]) PC = PC + 4 + BranchAddr
if(R[rs]!=R[rt]) PC = PC + 4 + BranchAddr
PC = JumpAddr
R[31] = PC + 8; PC = JumpAddr
PC = R[rs]
R[rt]={24'b0,M[R[rs] + SignExtImm](7 :0)}
R[rt]={ 16'b0,M[R[rs] + SignExtImm](15 :0)}
R[rt] = M[R[rs]+SignExtImm]
R[rt] = {imm, 16'b0}
R[rt] = M[R[rs]+SignExtImm]
R[rd] = ~ (R[rs] I R[1t])
R[rd] = R[rs] I R[rt]
R[rt] = R[rs] I ZeroExtImm
R[rd] = (R[rs] < R[rt])? 1 : 0
R[rt] = (R[rs] < SignExtImm)? 1 : 0 (2) ahex
R[rt] = (R[rs] < SignExtImm) ? 1:0
R[rd] = (R[rs] < R[rt])? 1 : 0
R[rd] = R[rt] « shamt
R[rd] = R[rt] >» shamt
M[R[rs]+SignExtImm](7:0) = R[rt](7:0)
M[R[rs]+SignExtImm] = R[rt]; R[rt] =(atomic)? 1 : 0
M[R[rs]+SignExtImm](I5:0) = R[rt](I5:0)
M[R[rs]+SignExtImm] = R[rt]
R[rd] = R[rs] - R[rt]
R[rd] = R[rs] - R[rt]

Operation

if(FPcond)PC=PC+4+BranchAddr (4)
if(!FPcond)PC=PC+4+BranchAddr(4)
Lo=R[rs]/R[rt]; Hi=R[rs]¾R[rt]
Lo=R[rs]/R[rt]; Hi=R[rs]¾R[rt]
F[fd]= F[fs] + F[ft]
{F[fd],F[fd+I]} = {F[fs],F[fs+I]} + {F[ft],F[ft+ 1]}
FPcond = (F[fs] op F[ft])? 1 : 0
FPcond = ({F[fs],F[fs+I]} op {F[ft],F[ft+I]})? 1 : 0
F[fd] = F[fs] / F[ft]

{F[fd],F[fd+l]} = {F[fs],F[fs+l]} / {F[ft],F[ft+ l]}
F[fd] = F[fs] * F[ft]
{F[fd],F[fd+l]} = {F[fs],F[fs+l]} * {F[ft],F[ft+ l]}
F[fd]=F[fs] - F[ft]
{F[fd],F[fd+l]} = {F[fs],F[fs+l]} - {F[ft],F[ft+1]}
F[1t]=M[R[rs]+SignExtlmm]
F[rt]=M[R[rs]+SignExtlmm]; F[rt+ l]=M[R[rs]+SignExtlmm+4]
R[rd] = Hi
R[rd] = Lo
R[rd] = CR[rs]
*{Hi,Lo} = R[rs] * R[rt]*
{Hi,Lo} = R[rs] * R[rt]
R[rd] = R[rt] » shamt
M[R[rs]+SignExtlmm] = F[rt]
M[R[rs]+SignExtlmm] = F[rt]; M[R[rs]+SignExtlmm+4] = F[rt+ l]